Modeling Dependent Data:

An Excursion

Reinhard Furrer
and the Applied Statistics Group

Version June 20, 2023
Git 15e8507

Contents

Preface

This document accompanies the lecture *STA 330 Modeling Dependent Data* for the spring semester of 2023. The lecture is given in the framework of the minor in applied probability and statistics (www.math.uzh.ch/aws). It typically compromises 12 weeks of two hours of lecture and one hour of exercises per week (14 weeks minus two weeks with holidays - on average).

The lecture has evolved over the spring semesters 2015, 2016, 2019, 2021, 2023 and older versions of this script may be floating around.

We assume prerequisites at the level of *STA 121 Statistical Modeling* or *STA 402 Likelihood Inference*.

In my view, it is very important that this document contains a structure that is tailored to the content I cover in class each week. This inherently leads to 12 "chapters." However, as we essentially cover four broad topics, I have cast the material of each of these in *seemingly* different chapters. This structure helps me to frame the lectures better: each week has a start, a set of learning goals, and a predetermined end. Figure 1 illustrates the four topics of this script.

At the beginning of each chapter, there is a direct link to an R file that contains all the code to re-create the analysis and figures of the chapter. Moreover, some of these files include additional concise illustrations, marked with `### Code not shown in script`. To not clutter this script with plenty of `par()` commands, we do not include them here and in the available R files. Datasets that are not part of regular CRAN packages are available at the URL www.math.uzh.ch/furrer/download/sta330/.

The script uses as "classical" notation consistently as possible: upper case for random quantities, lower case for realizations, boldface for vectors (lower case) and matrices (upper case). Inherently, a matrix can not be differentiated from a random vector based on notation only. The context and, if necessary, explanations will clarify.

Many have contributed to this document (alphabetical order), especially Clément Chevalier, Jakob Dambon, Roman Flury, Florian Gerber, Tim Gyger, Michael Hediger, Craig Wang (alphabetical order). Without their help, you would not be reading these lines. No textbook material is perfect. Please let me know of any necessary improvements, and I highly appreciate all forms of contributions in the form of errata, examples, or text blocks. Contributions can be deposited directly in the following Google Doc sheet. Major gaps in the document are indicated with the following icon

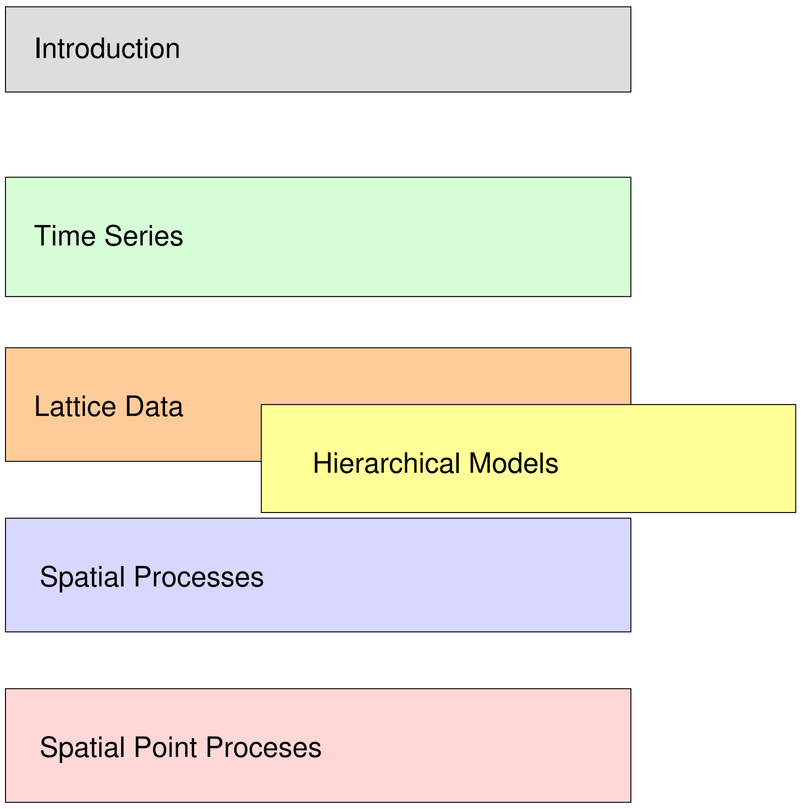


Figure 1: Structure of the manuscript

Reinhard Furrer
Summer 2023

Chapter 1

Introduction

In this chapter, we start motivating the need to model dependent data through two simple examples. We will observe temporally and spatially correlated data, violating the classical iid assumption.
The second part of the chapter contains a brief recap of the most important statistical concepts that we assume (known) throughout this manuscript.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter01.R.

1.1 Illustrative Examples

Example 1.1. The concentration of carbon dioxide (CO_2) in Earth's atmosphere has increased in the last century and is likely to increase in the future. CO_2 has a greenhouse effect and thus contributes to global warming. The concentration has increased markedly in the 21st century and exceeded 400 ppm daily average at Mauna Loa on May 10th, 2013. The left panels of Figure 1.1 shows recent monthly mean carbon dioxide measured at Mauna Loa Observatory, Hawaii. The right panel annual averages with a cubic fit. The data is available here with more information here. Data credits to Dr. Pieter Tans, NOAA/ESRL (`www.esrl.noaa.gov/gmd/ccgg/trends/`) and Dr. Ralph Keeling, Scripps Institution of Oceanography (`scrippsco2.ucsd.edu`). More information is available here, see also this article about the *Keeling Curve*.

The inconvenient truth is the increasing rate.

Our goal here is to propose *adequate* statistical models for the data. Figure 1.1 includes a lowess smooth and a third order polynomial, respectively (see R-Code 1.1). R-Code 1.2 fits a linear model to the monthly values. Relatively, we have a very good fit. However, the residuals show a strange behavior occurring after 400 months (the reason can be found here). Even after including indicator functions (through the vector `indi`) the residuals show structure, also reflected in the lag-one scatterplot of the right panel in Figure 1.2. For inference, we need to take this positive correlation into account. If we would not do so, our variance estimates are biased and our conclusions may not be appropriate. ♣

R-Code **1.1** Monthly and annual mean carbon dioxide measured at Mauna Loa Observatory, Hawaii. (See Figure 1.1.)

```r
URL <- "http://www.math.uzh.ch/furrer/download/sta330/"
raw <- read.csv(paste0(URL, "CO2_monthly.csv"))
raw <- t(raw[,-1]) # no years needed and by row format
dim(raw)

## [1] 12 62

craw <- c(raw[!is.na(raw)])
n <- length(craw)

mnts <- rep(1:12, dim(raw)[2])[!is.na(raw)]
time <- (1:n)/n
co2 <- ts(craw, start = c(1958, 3), frequency = 12)
str(co2)

##  Time-Series [1:731] from 1958 to 2019: 316 317 318 317 316 ...

plot(co2, ylab="CO2 [ppm]")
lines(lowess(co2, f=.2), col=2) # robust wrt f value!
am <- read.csv(paste0(URL, "CO2_annual.csv"))
dim(am)

## [1] 60  2

### Provided data is slightly different compared to averaged on:
# apply(raw[,-1],1,weighted.mean,na.rm=T)
plot(CO2~year, data=am, cex=.5, ylab="CO2 [ppm]")
lines(am$year, fitted(lm(CO2~poly(year, 3), data=am)), col=3)
```

R-Code **1.2** Fitting an annual and a cubic trend. (See Figure 1.2.)

```r
lm1 <- lm(craw ~ poly(time,3) + as.factor(mnts))
plot(resid(lm1))
### lots of structure!!
indi <- c(rep(0,400), rep(1,n-400))
lm2 <- lm(craw ~ indi * (poly(time,3)) + as.factor(mnts))
plot(resid(lm2))
plot(resid(lm2)[-1], resid(lm2)[-n]) # lag one scatterplot
AIC(lm1, lm2) # the smaller the better

##     df    AIC
## lm1 16 1699.7
## lm2 20 1020.8
```

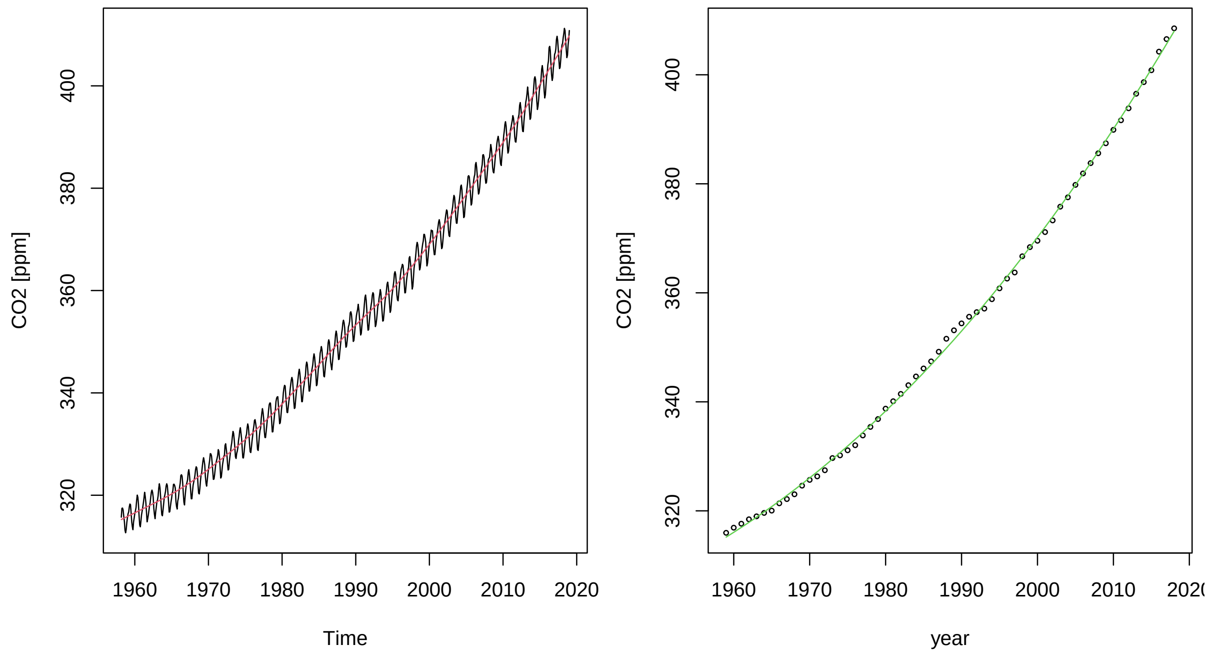


Figure 1.1: Monthly and annual mean carbon dioxide measured at Mauna Loa Observatory, Hawaii. The red and green curves are a lowess smooth and a third-order polynomial fit, respectively. (See R-Code 1.1.)

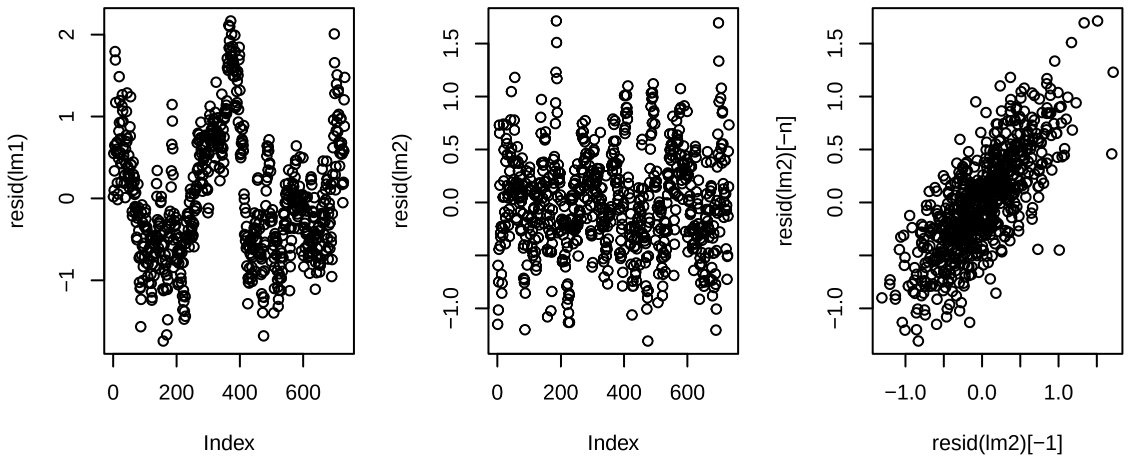


Figure 1.2: Residuals of annual and cubic trend fit without (left) and with (middle) Mount Pinatubo effect. Lag one scatterplot of residuals (right). (See R-Code 1.2.)

The iid assumptions of the noise in the regression model of Example 1.1 are clearly violated. In the coming two chapters, we introduce a few models for such a dependency in the temporal domain. We call such data *time series data* and its analysis *time series analysis*. The next example considers data in space, i.e., the data is georeferenced with two coordinates (x, y or latitude/longitude). Starting from Chapter 4, we consider data in space, illustrated in the next examples. We will briefly see some concepts of spatiotemporal analysis in Chapter 11.

Example 1.2. In this example, we consider the number of cases of oral cavity cancer during a five year period (1986–1990) in the $n = 544$ districts (Landkreise) of Germany (Knorr-Held and Raßer, 2000; Held *et al.*, 2005) The raw counts, shown in Figure 1.3, reflect areas of high population density. It is better to explore the spatial distribution of the relative risk. The expected number of cases e_i was derived using demographical data that allows us to display the standardized mortality ratios y_i/e_i (right panel of Figure 1.3). These ratios clearly show spatial structure and patches of higher (or lower) rates. The data is provided in the package `spam`. ♣

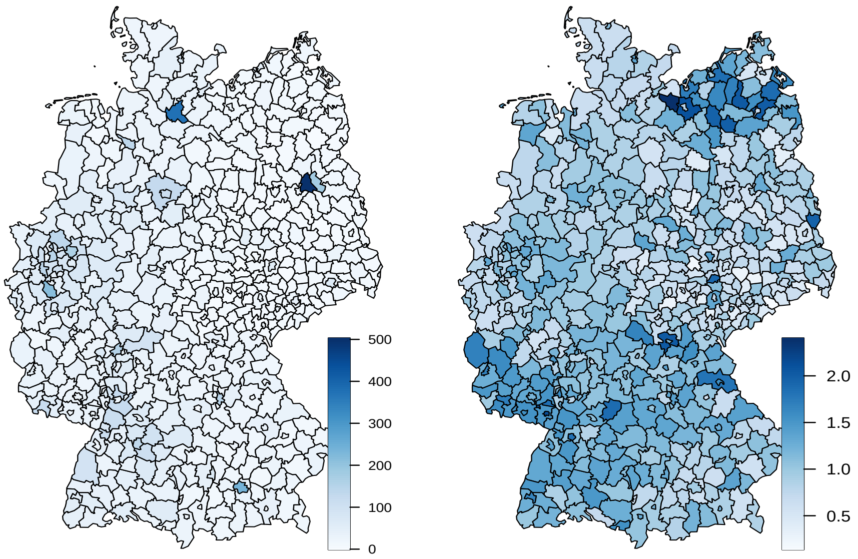


Figure 1.3: Raw counts and standardized mortality ratios of oral cavity cancer deaths observed between 1986–1990 in Germany.

As the number of districts in Example 1.2 is known, we need to work with a multivariate vector instead of individual independent random variables. The difficulty in this example is to model a "multivariate" Poisson vector. With (Bayesian) hierarchical models, we render the problem tractable, however. In Chapters 6 to 8, we introduce this powerful hierarchical formulation with the necessary Bayesian concepts.

Example 1.3. The mission of CIPEL (Commission internationale pour la protection des eaux du Léman) is to organize the studies required to determine the nature, extent, and source of any pollution that has occurred within the waters of Lake Geneva (www.cipel.org). The Commission also advises the contracting Governments (Switzerland, France) about the measures required to correct any existing pollution and prevent future pollution. In this document, we will mainly focus on the lake itself. Figure 1.4 gives a quick layout of the shore, main tributaries, major cities, and bathymetric information. Lake Geneva lies at a mean altitude of 372 meters and has a surface area of $580 km^2$, the most extensive body of fresh water in Western Europe.

We consider measurements from sediments of the lake of, e.g., the 1983 campaign, consisting of almost 300 samples. The data, lake boundaries, and convenient plotting functions are given in the file '`LacLeman.RData`', see R-Code 1.3. Figure 1.5 shows the mercury (Hg) readings and clearly shows spatial patterns: nearby readings have similar values. ♣

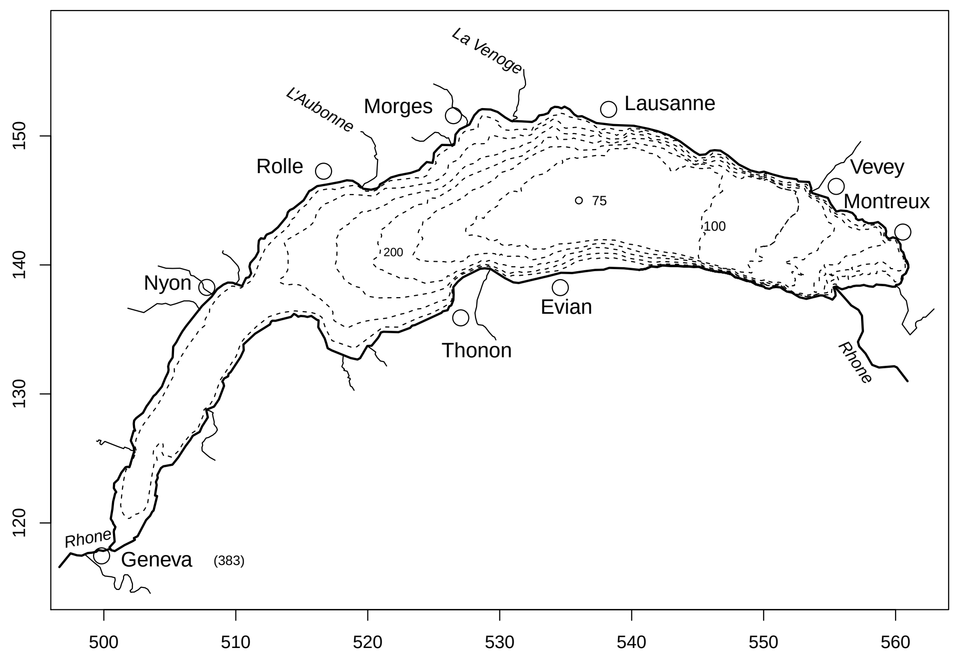


Figure 1.4: Lake Geneva: shore, main tributaries, major cities, and bathymetric information.

R-Code 1.3 *leman* data. (See Figure 1.5.)

```
load("data/LacLeman.RData") # contains the following objects:
ls(pattern = "l[ae]")
## [1] "lake"       "lake.data"  "leman.info" "leman78"    "leman83"
## [6] "leman88"
str(leman83)
## 'data.frame': 295 obs. of  6 variables:
##  $ x : num  501 502 501 503 502 ...
##  $ y : num  119 120 121 121 122 ...
##  $ Hg: num  0.17 0.21 0.06 0.24 0.35 0.14 0.08 0.26 0.23 0.18 ...
##  $ Pb: num  11 15 15 18 24 22 7 19 23 7 ...
##  $ Cd: num  0.23 0.37 0.14 0.3 0.56 0.3 0.17 0.44 0.39 0.26 ...
##  $ Zn: num  72.2 98.2 81.6 131 160 125 48 131 127 106 ...
### Call the following for more info:
# cat(leman.info)
library(fields) # we use this package quite often...
lake("Hg", leman83)
```

The following examples further illustrate spatial data. However, the (spatial) dependency is slightly less standard than in the previous one.

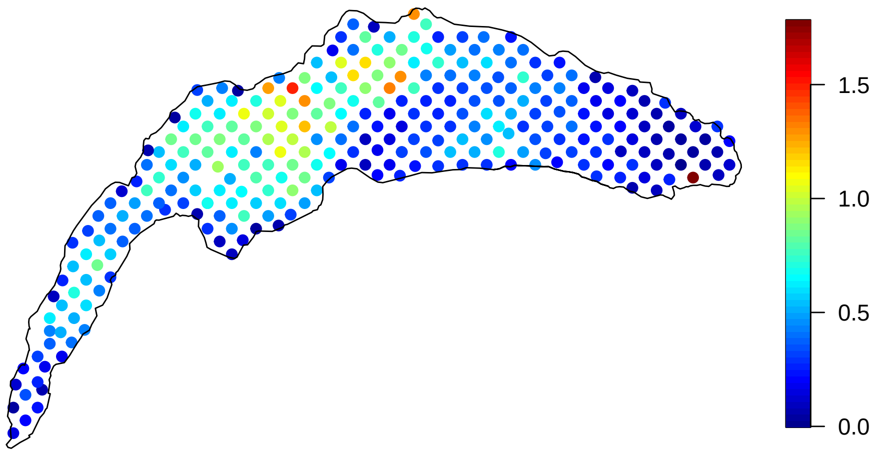


Figure 1.5: Mercury values (mg/kg) measured in 295 sediment samples in 1983 in Lake Geneva. (See R-Code 1.3.)

Example 1.4. In the road or highway construction framework, the topsoil is removed and replaced with appropriate earth material. Typically, this is done by placing a layer of material that is subsequently compacted. A compacted layer is about 20–30cm high. This step is repeated several times. The compaction is performed with moderate-size vibratory rollers equipped with one smooth drum. Sensors within this drum measure the vertical acceleration from which soil "stiffness" is derived (this process is quite complicated but not of our concern here). Depending on the roller type, several measurements per second are recorded. If the roller is equipped with a GPS-type device, it is possible to record the exact (up to a few centimeters) spatial location and the soil stiffness, as illustrated in Figure 1.6.

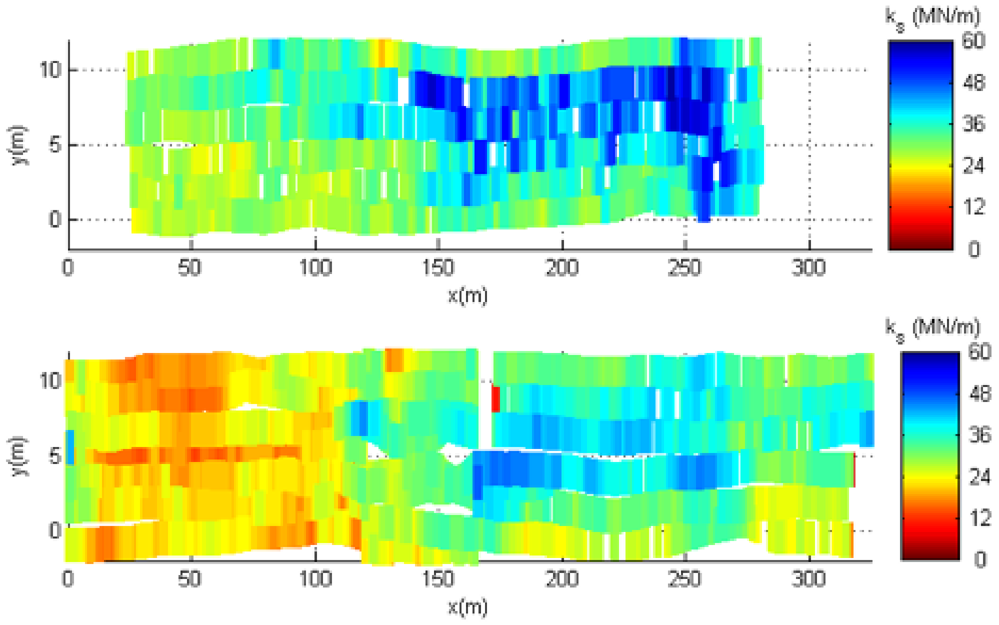


Figure 1.6: Soil stiffness for two layers of a construction site in Iowa, USA. (Red bad, blue good).

Typical (statistical) questions associated with such datasets are:

- What is the magnitude of the measurement error? Is the (spatial) variation in the data typical?
- What is the (best guess) soil stiffness at locations where we do not have any observations?
- Are (specific) low values induced by soft spots or an artifact of the measuring process?
- Is information ported over several layers?
- Can the collected information be used for intelligent compaction? ♣

Example 1.5. The influence of human activities on the Earth's climate is largely undisputed, and the potential for even more significant changes in this century confronts us. A grand challenge facing geosciences is to provide accurate predictions of these changes along with quantifications of uncertainties. Because future climate may be very different from the observational record, a primary tool for assessing changes are large computer models, termed general circulation models (GCMs) that simulate the Earth's climate system under different circumstances. The results of such models are complex spatial fields. Statistical analysis is particularly useful for synthesizing information from several models and providing statistical measures of uncertainty. The application of conventional statistics here is interesting because the models are deterministic computer codes, but the variation and biases among different models can fit into a probabilistic framework. Accordingly, we will refer to the model output as "data" even though it may not fit the conventional perception of a statistical sample.

The *climate* at a given location is the joint distribution of meteorological variables describing the atmosphere averaged over a given time period. In statistical language, given a stationary time series, the climate is simply the marginal or stationary distribution. A standard working definition of climate is a twenty to thirty-year average around a particular time. The temporal variability of meteorology about a climatological mean is termed weather. Weather is observed both in the real world and in what is simulated by models. Thus, any analysis for differences in climate must account for the intrinsic variability of weather and the fact that climate can not be determined precisely with a finite sample. The number of cutting-edge climate system models is limited. However, it is ironic that despite the voluminous spatial output for a given model, the sample size for comparison across different models is small. Hence, careful statistical work

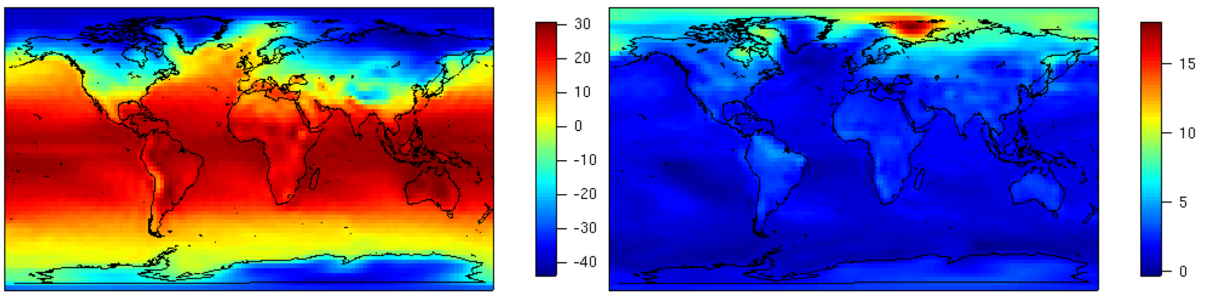


Figure 1.7: Winter temperature fields (left panel) winter climate change (right panel) for one specific model (Unit: °C).

addresses this problem with spatial models that borrow strength across adjacent spatial regions and provide a more statistically accurate assessment of model bias and variability. Based on a statistical framework for the response of a suite of climate models, it is possible to produce a synthesized estimate of climate change at a regional scale along with a measure of the uncertainty.

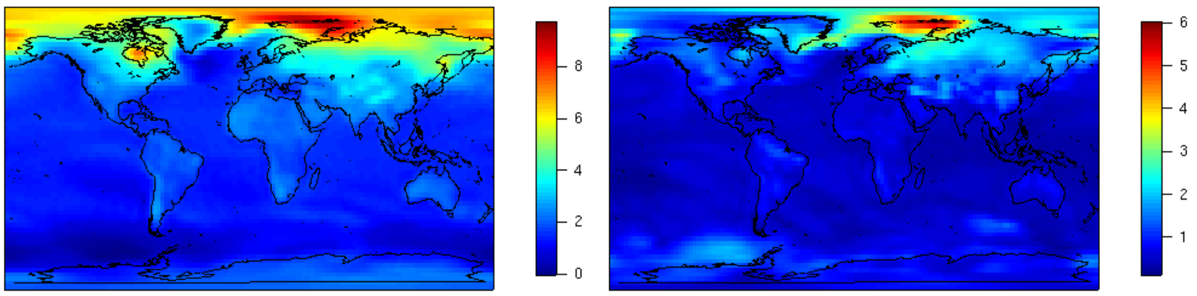


Figure 1.8: Mean and standard deviation of boreal winter temperature climate change (Unit: °C).

As an example of the model output, Figure 1.7 shows winter temperature and winter temperature change (i.e., climate change) for a specific model. As a helpful summary, Figure 1.8 shows the mean and standard deviation over nine models of boreal winter temperature climate change.

Typical (statistical) questions associated with such datasets are:

- How can we statistically model the uncertainties in future climate change at different times and spatial scales?
- How can we statistically relate the changes in the mean to the changes in variability and extreme events?
- By how much can we improve statistical prediction when using joint climate variables?
- To what degree does agreement with observations imply predictive skill for the future?
- How can we efficiently model the evident non-stationarity of climate variability on the globe? ♣

Example 1.6. Improvised Explosive Devices (IED) or roadside bomb has emerged as a weapon of strategic influence on today's battlefield. Insurgencies in Northern Ireland, Lebanon, Chechnya, Iraq, and Afghanistan have all used the IED to influence the battlefield significantly. They are difficult to identify, easy to produce, and extremely lethal. As of April 2009, IEDs account for 75 percent of casualties to coalition forces in Afghanistan (Vanden Brook, 2009). IEDs have produced more casualties than any other weapon in Iraq and accounted for 60 percent of coalition casualties in 2006-2007 (iCasualties, 2009). Units that routinely operate in a certain area gain a greater understanding of the local enemy activity, but today's battlefield requires patrols to often travel in unfamiliar areas. Due to the continual adaptation of friendly and enemy forces, assessing the likelihood of enemy activity in unfamiliar areas of operation is time and resource intensive. Although historical data is available, we often assess tactical risk subjectively. Furthermore,

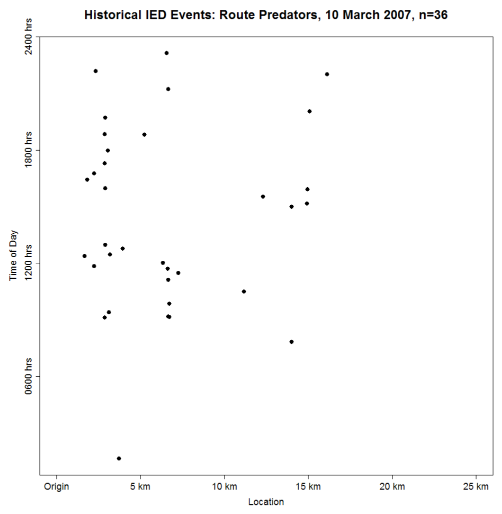


Figure 1.9: IED detonations of 'Route Predator'.

many of these 'out of sector' missions are conducted at the platoon level and not given significant support from battalion or brigade staff.

IED attacks happen almost exclusively on roads. Hence, space is effectively one-dimensional, and we can use time as the second dimension, as illustrated in Figure 1.9.

The data should be used to answer questions like:

- Are there temporal windows that are safer to use?
- Have enemy tactics evolved over time? Especially are there correlations with troop surges?

The data used in the research project (Benigni and Furrer, 2008) was declassified by stripping all information from the IED events. Of course, when working with classified data, the list of questions can be extended and would include the following:

- What are the IED characteristics that cause the most severe casualties?
- Are the IED types clustered in space and time? ♣

In all examples, the data is associated with space or space and time. However, there are fundamental differences. We can obtain measurements at an arbitrary spatial location within our spatial domain in some examples. With additional effort, it would be possible to increase the density of observations. Often, the measurement is representative of a tiny area or volume compared to the entire spatial domain. In other examples, a value represents a fixed geographic region, and it is often impossible to obtain additional readings (at a sub-region scale or for a differently defined region). In this example, the regions are regular. However, the regions are often induced by political boundaries (counties, cantons, ...) or defined historically (US ZIP codes). The last example represents the recording of the spatial location of some sort of "events". Other classical types of such data are the position of a tree in a specific study area, sightings of animals, etc.

We often refer to these three types of data as geostatistical data, lattice data, and spatial point pattern, discussed in detail in this document.

1.2 Random Vectors

A random vector is a (column) vector whose components are random variables, i.e., $\mathbf{Y} = (Y_1, \dots, Y_p)^\top$, and $Y_1, \dots, Y_p$ are random variables.

Definition 1.1. The multidimensional cumulative distribution function (CDF) of $\mathbf{Y}$ is defined as

$$F_{\mathbf{Y}}(\boldsymbol{y}) = P\{\mathbf{Y} \leq \boldsymbol{y}\} = P\{Y_1 \leq y_1, \dots, Y_p \leq y_p\}, \tag{1.1}$$

where the list is understood as the intersection ($\cap$). $\diamondsuit$

Only in the case of independence, the CDF simplifies to $F_{\mathbf{Y}}(\boldsymbol{y}) = \displaystyle\prod_{i=1}^{p} P\{Y_i \leq y_i\}$.

The probability density function for a continuous random vector is defined in a similar manner as for random variables.

Definition 1.2. The probability density function (density function, pdf) $f_{\mathbf{Y}}(\boldsymbol{y})$ of a p-dimensional continuous random vector $\mathbf{Y}$ is defined by

$$\mathrm{P}(\mathbf{Y} \in A) = \int_A f_{\mathbf{Y}}(\boldsymbol{y}) d\boldsymbol{y}, \qquad \text{for all } A \subset \mathbb{R}^p. \tag{1.2}$$

$\diamondsuit$

1.2.1 Basic Properties

The expectation and variance of a random variable determine the location and the spread thereof. For two random variables, the covariance quantifies the linear relationship between the two. More precisely, the covariance between two random variables Y_1 and Y_2 is defined as

$$\operatorname{Cov}(Y_1, Y_2) = \mathrm{E}\big((Y_1 - \mathrm{E}(Y_1))(Y_2 - \mathrm{E}(Y_2)) \big). \tag{1.3}$$

For random variables Y_1, Y_2 and Y_3 with finite second moments, we have the following properties

$$\operatorname{Cov}(Y_1, Y_2) = \mathrm{E}(Y_1 Y_2) - \mathrm{E}(Y_1)\,\mathrm{E}(Y_2) \tag{1.4}$$

$$\operatorname{Cov}(Y_1, Y_2) = \operatorname{Cov}(Y_2, Y_1) \tag{1.5}$$

$$\operatorname{Cov}(Y_1, Y_1) = \operatorname{Var}(Y_1) \tag{1.6}$$

$$\operatorname{Cov}(a + b\,Y_1, c + d\,Y_2) = b\,d\,\operatorname{Cov}(Y_1, Y_2) \tag{1.7}$$

$$\operatorname{Cov}(Y_1, Y_2 + Y_3) = \operatorname{Cov}(Y_1, Y_2) + \operatorname{Cov}(Y_1, Y_3) \tag{1.8}$$

If Y_1 and Y_2 are independent, then the joint density is identical to the product of the densities of Y_1 and Y_2. Thus $\mathrm{E}(Y_1 Y_2) = \mathrm{E}(Y_1)\,\mathrm{E}(Y_2)$ and the covariance is zero. The converse does, in general, not hold.

The correlation between two random variables Y_1 and Y_2 is defined as

$$\operatorname{Corr}(Y_1, Y_2) = \frac{\operatorname{Cov}(Y_1, Y_2)}{\sqrt{\operatorname{Var}(Y_1)\operatorname{Var}(Y_2)}} \tag{1.9}$$

and is a *normed* covariance. For all random variables Y_1 and Y_2, we have $-1 \leq \operatorname{Corr}(Y_1, Y_2) \leq 1$.

Definition 1.3. The expectation of a random vector $\mathbf{Y}$ is defined as

$$\mathrm{E}(\mathbf{Y}) = \begin{pmatrix} \mathrm{E}(Y_1) \\ \vdots \\ \mathrm{E}(Y_p) \end{pmatrix} \tag{1.10}$$

and the variance of a random vector $\mathbf{Y}$ is defined as

$$\mathrm{Var}(\mathbf{Y}) = \mathrm{E}\Big(\big(\mathbf{Y} - \mathrm{E}(\mathbf{Y})\big)\big(\mathbf{Y} - \mathrm{E}(\mathbf{Y})\big)^T\Big) = \begin{pmatrix} \mathrm{Var}(Y_1) & \dots & \mathrm{Cov}(Y_i, Y_j) \\ & \ddots & \\ \mathrm{Cov}(Y_j, Y_i) & \dots & \mathrm{Var}(Y_p) \end{pmatrix}. \tag{1.11}$$

♢

Sometimes one refers to (1.11) as the variance-covariance of $\mathbf{Y}$. Similar to the properties of (scalar) random variables, we have for random vectors the following ones:

Property 1.1. *For arbitrary random p-vectors* $\mathbf{Y}$*, (fixed) vectors* $\boldsymbol{a} \in \mathbb{R}^q$ *and matrices* $\mathbf{B} \in \mathbb{R}^{q \times p}$*,* $p, q > 0$*, we have*

- $\mathrm{Var}(\mathbf{Y}) = \mathrm{E}(\mathbf{Y}\mathbf{Y}^T) - \mathrm{E}(\mathbf{Y})\,\mathrm{E}(\mathbf{Y})^\top$
- $\mathrm{E}(\boldsymbol{a} + \mathbf{B}\mathbf{Y}) = \boldsymbol{a} + \mathbf{B}\,\mathrm{E}(\mathbf{Y})$,
- $\mathrm{Var}(\boldsymbol{a} + \mathbf{B}\mathbf{Y}) = \mathbf{B}\,\mathrm{Var}(\mathbf{Y})\mathbf{B}^\top$

We only consider a particular multivariate distribution, the multivariate Gaussian distribution, introduced in the next section.

1.2.2 Multivariate Gaussian Distributions

Definition 1.4. The random vector $\mathbf{Y} = (Y_1, \dots, Y_p)^\top$ is distributed according a multivariate Gaussian distribution if

$$F_{\mathbf{Y}}(\boldsymbol{y}) = \int_{-\infty}^{y_1} \cdots \int_{-\infty}^{y_p} f_{\mathbf{Y}}(x_1, \dots, x_p) dx_1 \dots dx_p \tag{1.12}$$

with density

$$f_{\mathbf{Y}}(y_1, \dots, y_p) = f_{\mathbf{Y}}(\boldsymbol{y}) = \frac{1}{(2\pi)^{p/2} \det(\boldsymbol{\Sigma})^{1/2}} \exp\Big(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{y} - \boldsymbol{\mu})\Big) \tag{1.13}$$

for all $\boldsymbol{y} \in \mathbb{R}^p$ ($\boldsymbol{\mu} \in \mathbb{R}^p$ and symmetric positive definite matrices $\boldsymbol{\Sigma}$. We denote $\mathbf{Y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. ♢

Property 1.2. *For a multivariate Gaussian distribution, the following properties hold*

$$\mathrm{E}(\mathbf{Y}) = \boldsymbol{\mu}, \qquad \mathrm{Var}(\mathbf{Y}) = \boldsymbol{\Sigma}. \tag{1.14}$$

Property 1.3. *Let* $\boldsymbol{a} \in \mathbb{R}^q$*,* $\mathbf{B} \in \mathbb{R}^{q \times p}$*,* $q \le p$*,* $\mathrm{rank}(\mathbf{B}) = q$ *and* $\mathbf{Y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*. Then*

$$\boldsymbol{a} + \mathbf{B}\mathbf{Y} \sim \mathcal{N}_q\big(\boldsymbol{a} + \mathbf{B}\boldsymbol{\mu}, \mathbf{B}\boldsymbol{\Sigma}\mathbf{B}^\top\big). \tag{1.15}$$

This last property has profound consequences. It also asserts that the one-dimensional marginal distributions are again Gaussian with $Y_i \sim \mathcal{N}\big((\boldsymbol{\mu})_i, (\boldsymbol{\Sigma})_{ii}\big)$, $i = 1, \dots, p$. Similarly, any subset of random variables of $\mathbf{Y}$ is again Gaussian with appropriate mean and covariance matrix selection.

We now discuss how to draw realizations from an arbitrary Gaussian random vector. Let $\mathbf{I} \in \mathbb{R}^{p\times p}$ be the identity matrix, a square matrix that has only ones on the main diagonal and only zeros elsewhere, and let $\mathbf{L} \in \mathbb{R}^{p\times p}$ so that $\mathbf{L}\mathbf{L}^\top = \boldsymbol{\Sigma}$. That means, $\mathbf{L}$ is like a "matrix square root" of $\boldsymbol{\Sigma}$.

To draw a realization $\boldsymbol{y}$ from a p-variate random vector $\mathbf{Y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, one starts with drawing p values from $Z_1, \dots, Z_p \overset{\text{iid}}{\sim} \mathcal{N}(0, 1)$, and sets $\boldsymbol{z} = (z_1, \dots, z_p)^\top$. The vector is then (linearly) transformed with $\boldsymbol{\mu} + \mathbf{L}\boldsymbol{z}$. Since $\mathbf{Z} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$ Property 1.3 asserts that $\mathbf{Y} = \boldsymbol{\mu} + \mathbf{L}\mathbf{Z} \sim \mathcal{N}_p(\boldsymbol{\mu}, \mathbf{L}\mathbf{L}^\top)$.

In practice, the Cholesky decomposition of $\boldsymbol{\Sigma}$ is often used, which decomposes a symmetric positive-definite matrix into the product of a lower triangular matrix $\mathbf{L}$ and its transpose.

Evaluating the log-density (1.13) based on a Cholesky factor $\mathbf{L}$ is very convenient because (1) it holds that $\det(\boldsymbol{\Sigma}) = \det(\mathbf{L})^2 = \prod_{i=1}^p (\mathbf{L})_{ii}^2$, and, (2) the quadratic form is evaluated based on one forward-solve `v <- forwardsolve(L,b)` and `sum(v**2)`.

Example 1.7. This example visualizes the bivariate Gaussian distribution. To simplify the notation, we use X and Y instead of Y_1 and Y_2. The density (1.13) "simplifies" to

$$\begin{aligned} f(x, y) &= f_{X,Y}(x, y) \qquad (1.16)\\ &= \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y}\right]\right), \end{aligned}$$

for all x and y ($\mu_x \in \mathbb{R}$, $\mu_y \in \mathbb{R}$, $\sigma_x > 0$, $\sigma_y > 0$ and $-1 < \rho < 1$). Here, $\operatorname{Corr}(X, Y) = \rho$ and hence $\operatorname{Cov}(X, Y) = \rho\,\sigma_x\,\sigma_y$. If $\rho = 0$, then X and Y are independent and vice versa. This result will be revisited for the general case later.

R-Code 1.4 shows the density of a bivariate Gaussian random vector. The isolines of the density are ellipses (due to the quadratic form in the kernel). The mean vector determines the location, and the covariance matrix the shape. More specifically, the eigenvectors and eigenvalue of the covariance matrix define the principal axes and their relative lengths of the ellipses.

R-Code 1.5 realizations of a bivariate Gaussian distribution for different values of ρ. Note that large sample sizes are required to 'detect' small correlations visually. ♣

R-Code 1.4: Density of a bivariate Gaussian random vector. (See Figure 1.10.)

```r
library(mvtnorm)
library(RColorBrewer)
Sigma <- array(c(1, 2, 2, 5), c(2, 2))
x <- y <- seq(-3, to=3, length=100)
```

```
grid <- expand.grid(x=x, y=y)
densgrid <- dmvnorm(grid, mean=c(0, 0), sigma=Sigma)
density <- array(densgrid, c(100, 100))
col <- colorRampPalette(brewer.pal(9, "Blues"))(100)
image(x, y, density, col=col) # left panel
contour(x, y, density, add=T)
faccol <- col[cut(density[-1,-1], 64)]
persp(x, y, density, col=faccol, border = NA, zlab="", # right panel
    tick="detailed", theta=120, phi=30, r=100, zlim=c(0,0.16))
### To calculate the cdf, we need a lower and upper bound. Passing the grid
cdfgrid <- apply(grid, 1, function(x) {        #  directly is not possible
    pmvnorm( upper=x, mean=c(0, 0), sigma=Sigma) } )
jcdf <- matrix( cdfgrid, 100, 100)
image(x, y, jcdf, zlim=c(0,1), col=col)                     # left panel
contour(x, y, jcdf, add=T)
faccol <- col[cut(jcdf[-1,-1],64)]
persp(x, y, jcdf, col=faccol, border = NA, zlab="",         # right panel
    tick="detailed", theta=12, phi=50, r=100, zlim=c(0,1))
```

R-**Code 1.5** Realisations of a bivariate Gaussian distribution for different values of ρ. The sample size is 500. (See Figure 1.11.)

```
rho <- c(-.25, 0, .1, .25, .75, .9)
lim <- c(-3.3, 3.3)             # common scale for all
for (i in 1:6) {                # cycle over all six correlations
  Sigma <- array(c(1, rho[i], rho[i], 1), c(2,2))
  sample <- rmvnorm(500, sigma=Sigma)
  plot(sample, pch=".", xlab="", ylab="", ylim=lim, xlim=lim)
  legend("topleft", legend=bquote(rho==.(rho[i])), bty="n")
}
```

1.2.3 Conditional Distribution

We consider properties of subsets elements of the random vector $\mathbf{Y}$. To simplify the notation, we reorder the elements of the vector to write

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix}, \quad \mathbf{Y}_1 \in \mathbb{R}^q, \quad \mathbf{Y}_2 \in \mathbb{R}^{p-q}. \tag{1.17}$$

We partition the mean vector and the matrix $\boldsymbol{\Sigma}$ in 2×2 block

$$\mathbf{Y} = \begin{pmatrix} \mathbf{Y}_1 \\ \mathbf{Y}_2 \end{pmatrix} \sim \mathcal{N}_p\left(\begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix} \right) \tag{1.18}$$

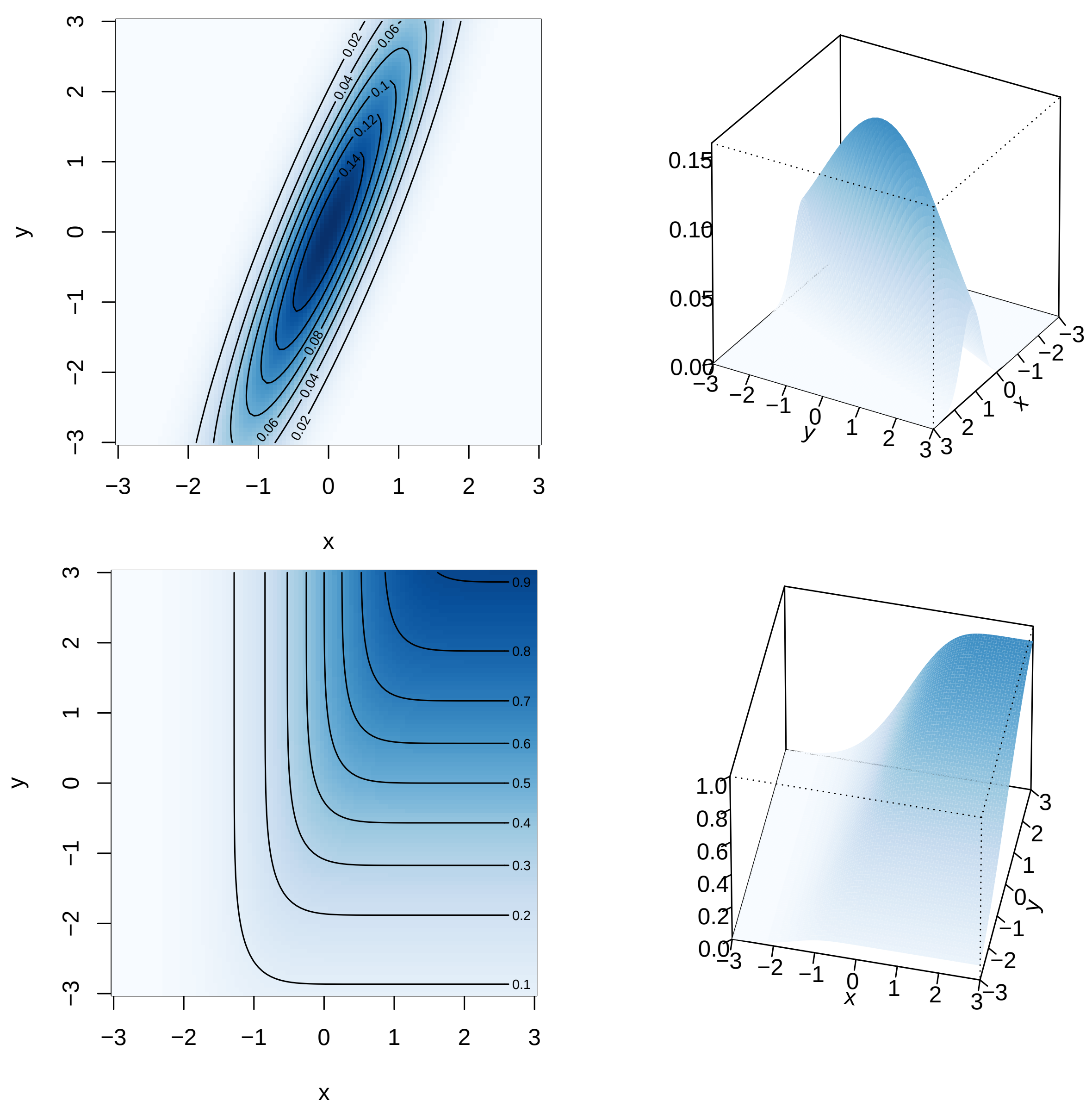


Figure 1.10: Density of a bivariate Gaussian random vector. (See R-Code 1.4.)

The (possibly) multivariate marginal random vectors $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are Gaussian as well with $\mathbf{Y}_1 \sim \mathcal{N}_q(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11})$ and $\mathbf{Y}_2 \sim \mathcal{N}_{p-q}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22})$.

If $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are independent then $\boldsymbol{\Sigma}_{21} = \mathbf{0}$. In the Gaussian case, the converse is true as well.

Much of temporal or spatial statistics can be justified by the following property.

Property 1.4. *If we condition a multivariate Gaussian random vector on a subset of its components, then the resulting conditional distribution is again Gaussian with*

$$\mathbf{Y}_1 \mid \mathbf{Y}_2 = \boldsymbol{y}_2 \sim \mathcal{N}_q\left(\boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{y}_2 - \boldsymbol{\mu}_2), \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\right). \tag{1.19}$$

Notice that the conditional expectation depends linearly on $\boldsymbol{y}_2$. However, the conditional variance is independent of the value $\boldsymbol{y}_2$. The conditional expected value represents an update

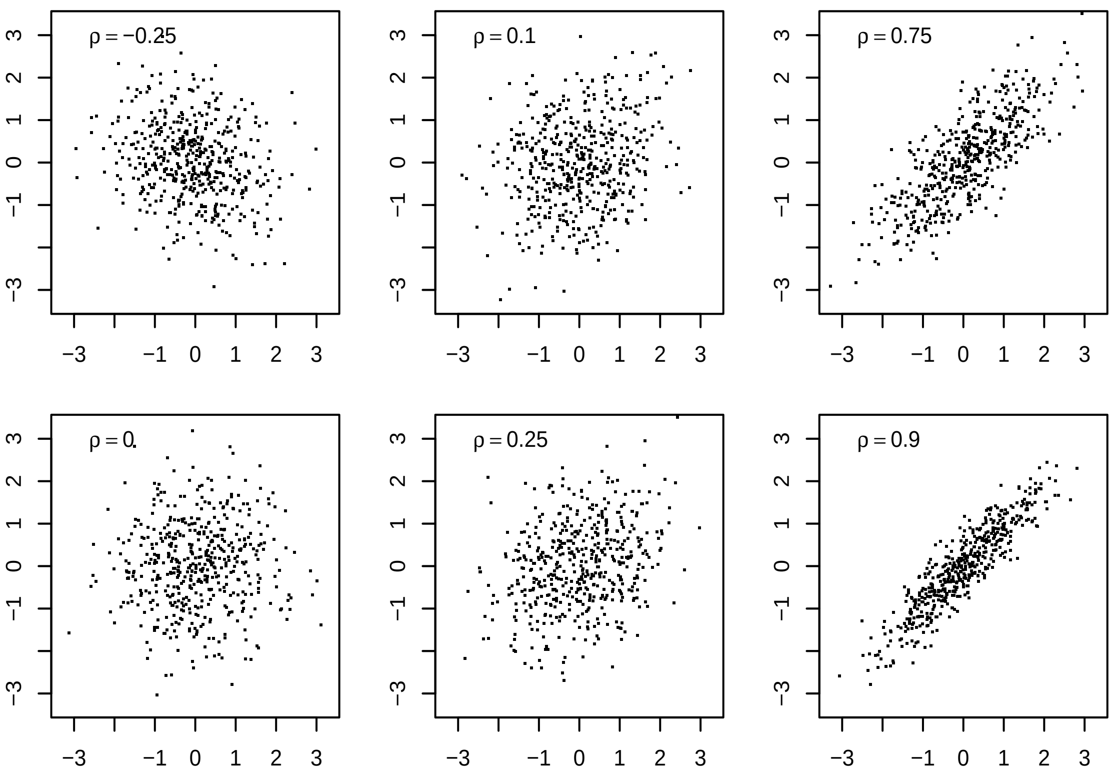


Figure 1.11: Realisations of a bivariate Gaussian distribution for different values of ρ. The sample size is 500. (See R-Code 1.5.)

of $\mathbf{Y}_1$ through $\mathbf{Y}_2 = \boldsymbol{y}_2$: the difference $\boldsymbol{y}_2 - \boldsymbol{\mu}_2$ is normalized by the variance and scaled by the covariance. Notice that for $p = 2$, $\boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1} = \rho\sigma_x/\sigma_y$ and $\boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} = \sigma_x^2(1-\rho^2)$, yielding

$$Y \mid X = x \sim \mathcal{N}\left(\mu_y + \rho\sigma_y\sigma_x^{-1}(x - \mu_x), \sigma_y^2 - \rho^2\sigma_y^2\right). \tag{1.20}$$

This equation is illustrated in Figure (1.12).

1.3 Estimation

The estimators in the multivariate setting are constructed similarly to the univariate case. Let $\boldsymbol{y}_1, \dots, \boldsymbol{y}_n$ be a realization of a random sample $\mathbf{Y}_1, \dots, \mathbf{Y}_n$. We use the following estimators

$$\widehat{\boldsymbol{\mu}} = \overline{\mathbf{Y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{Y}_i \qquad \widehat{\boldsymbol{\Sigma}} = \frac{1}{n-1}\sum_{i=1}^{n}(\mathbf{Y}_i - \overline{\mathbf{Y}})(\mathbf{Y}_i - \overline{\mathbf{Y}})^\top \tag{1.21}$$

and estimates

$$\widehat{\boldsymbol{\mu}} = \overline{\boldsymbol{y}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{y}_i \qquad \widehat{\boldsymbol{\Sigma}} = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{y}_i - \overline{\boldsymbol{y}})(\boldsymbol{y}_i - \overline{\boldsymbol{y}})^\top. \tag{1.22}$$

If the random sample satisfies $\mathbf{Y}_1, \dots, \mathbf{Y}_n \overset{\text{iid}}{\sim} \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, closed form properties of the estimators can be derived.

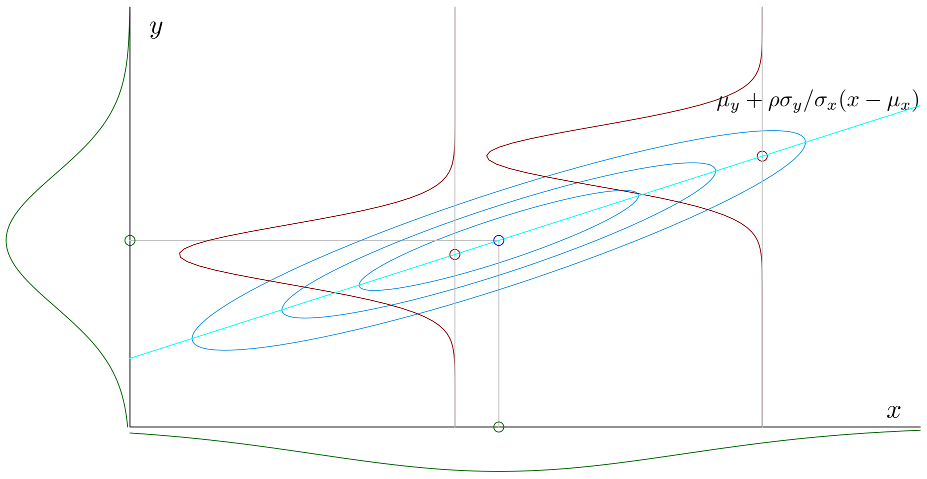


Figure 1.12: Graphical illustration of the conditional distribution of a bivariate normal random vector. Blue: bivariate density with isolines indicating quartiles, green: marginal densities, red: conditional densities. The respective means are indicated with a circle. The height of the univariate densities are exaggerated by a factor of five.

Example 1.8. R-Code 1.6 illustrates graphically that $\widehat{\boldsymbol{\mu}} \to \boldsymbol{\mu}$ and $\widehat{\boldsymbol{\Sigma}} \to \boldsymbol{\Sigma}$ when increasing n in the case of a bivariate Gaussian case. ♣

R-Code 1.6 Realizations of bivariate Gaussian random variables using different sample sizes. (See Figure 1.13.)

```
library(ellipse)
n <- c(10, 100, 500, 1000)
mu <- c(2, 1)
Sigma <- matrix(c(4, 2, 2, 2), 2)
x <- seq(-3, to=7, length=100)
y <- seq(-3, to=5, length=100)
grid <- expand.grid(x=x, y=y)
density <- array(dmvnorm(grid, mean=mu, sigma=Sigma), c(100, 100))
for (i in 1:4) {                        # loop over different sample sizes
  contour(x, y, density, drawlabels=FALSE, col="gray")
  sample <- rmvnorm(n[i], mean=mu, sigma=Sigma)
  points(sample, pch=".", cex=2)
  Sigmahat <- cov(sample)
  muhat <- apply(sample, 2, mean)
  lines(ellipse::ellipse(Sigmahat, cent=muhat, level=.95), col=2, lwd=2)
  lines(ellipse::ellipse(Sigmahat, cent=muhat, level=.5), col=4, lwd=2)
  points(rbind(muhat), col=3, cex=2)
  text(-2, 4, paste("n =", n[i]))
}
```

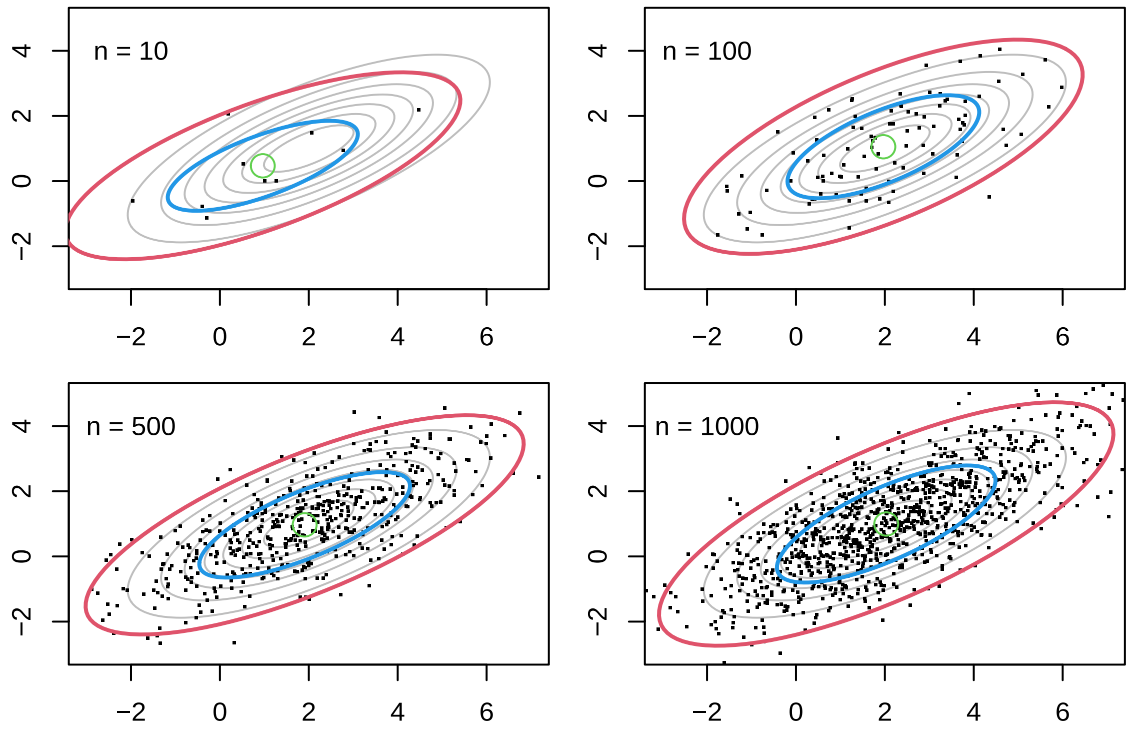


Figure 1.13: Realizations of bivariate Gaussian random variables using different sample sizes. Gray ellipses represent the isolines of the bivariate density. The green dot is the estimated mean. The red and blue ellipses are the estimated 95% and 50% quantile regions. That means that one expects half of all the realizations within the blue ellipse. (See R-Code 1.6.)

1.4 Regression

We often have predictors that explain much of the data's structure. In Example 1.1, the CO_2 increased over time but also depended on the season. A classical approach to "model" this data is by regressing the observations onto these temporal predictors. Hence, this summary section.

We write the multiple linear regression with p predictors

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i, \tag{1.23}$$

$$= \boldsymbol{x}_i^\top \boldsymbol{\beta} + \varepsilon_i \qquad i = 1, \dots, n, \quad n > p \tag{1.24}$$

where

- y_i: dependent variable, observation, data,
- $\boldsymbol{x}_i = (1, x_{i1}, \dots, x_{ip})^\top$: free variables, predictors,
- $\boldsymbol{\beta} = (\beta_0, \dots, \beta_p)^\top$: parameter vector (unknown)
- ε_i: error (unknown), with symmetric distribution and $\mathrm{E}(\varepsilon_i) = 0$.

In the basic framework, we assume that $\mathrm{Var}(\varepsilon_i) = \sigma^2$ and $\mathrm{Cov}(\varepsilon_i, \varepsilon_j) = 0$, $i \neq j$. Here, we additionally assume that the error is Gaussian, i.e., $\varepsilon \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$.

To derive the estimates/estimators etc., it is convenient two write (1.24) using matrix notation

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \tag{1.25}$$

with $\mathbf{X}$ a $n \times (p+1)$-matrix with rows $\boldsymbol{x}_i^\top$. We assume that the (column) rank of $\mathbf{X}$ is $p+1$ ($\text{rank}(\mathbf{X}) = p+1$).

To derive an estimate, we use the least squares principle:

$$\widehat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\text{argmin}}(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) \tag{1.26}$$

$$\Rightarrow \quad \frac{d}{d\boldsymbol{\beta}}(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}) \tag{1.27}$$

$$= \frac{d}{d\boldsymbol{\beta}}(\boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{\beta}^\top\mathbf{X}^\top\boldsymbol{y} + \boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta}) = -2\mathbf{X}^\top\boldsymbol{y} + 2\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} \tag{1.28}$$

$$\Rightarrow \quad \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^\top\boldsymbol{y} \tag{1.29}$$

$$\Rightarrow \quad \widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y} \tag{1.30}$$

Equation (1.29) is called the *normal equation*. Formally, $\widehat{\boldsymbol{\beta}}$ as defined in (1.30) is an estimate, i.e., a "classical" vector with $p+1$ scalars. To derive properties from $\widehat{\boldsymbol{\beta}}$, we have to consider it as an *estimator* in the sense that we "replace" the actual data vector $\boldsymbol{y}$ with the random vector $\mathbf{Y}$, as defined in (1.25).

Starting from equation (1.24) and (1.30) the following estimates (left column) and distributions (right column) can be derived:

$$\mathbf{Y} \sim \mathcal{N}_n(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}) \tag{1.31}$$

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y} \qquad \widehat{\boldsymbol{\beta}} \sim \mathcal{N}_{p+1}\big(\boldsymbol{\beta}, \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}\big) \tag{1.32}$$

$$\widehat{\boldsymbol{y}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y} = \mathbf{H}\boldsymbol{y} \qquad \widehat{\mathbf{Y}} \sim \mathcal{N}_n(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{H}) \tag{1.33}$$

$$\boldsymbol{r} = \boldsymbol{y} - \widehat{\boldsymbol{y}} = (\mathbf{I} - \mathbf{H})\boldsymbol{y} \qquad \mathbf{R} \sim \mathcal{N}_n\big(\mathbf{0}, \sigma^2(\mathbf{I} - \mathbf{H})\big) \tag{1.34}$$

where we term the matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top$ as the hat matrix. The hat matrix is a projection matrix and is symmetric and idempotent ($\mathbf{H}^2 = \mathbf{H}$), and so is $\mathbf{I}-\mathbf{H}$. It also holds that $\text{rank}(\mathbf{H}) = p+1$ and the eigenvalues of $\mathbf{H}$ consist of $p+1$ ones and $n-p-1$ zeros, while the eigenvalues of $\mathbf{I}-\mathbf{H}$ consist of $n-p-1$ ones and $p+1$ zeros (and, thus, $\text{rank}(\mathbf{I}-\mathbf{H}) = n-p-1$).

Further, we have

$$\widehat{\sigma}^2 = \frac{1}{n-p-1}\mathbf{R}^\top\mathbf{R} \sim \chi^2_{n-p-1} \tag{1.35}$$

and thus

$$\frac{\widehat{\beta}_i - \beta_i}{\sqrt{\widehat{\sigma}^2 v_{ii}}} \sim t_{n-p-1} \qquad \text{with } v_{ii} = \big((\mathbf{X}^\top\mathbf{X})^{-1}\big)_{ii}, \quad i = 0, \dots, p. \tag{1.36}$$

Of course, this last pivot is only exact if the assumption of iid errors in (1.24) hold. A generalized least squares approach may be advocated in a classical regression setting. In the context of spatial data, we have additional models that will be discussed in the subsequent chapters.

1.5 Positive-Definite Matrices

Definition 1.5. A matrix $\mathbf{A}$ is nonsingular if it has full column and full row rank.

By definition a nonsingular matrix has to be square. We denote the inverse of a nonsingular matrix $\mathbf{A}$ by $\mathbf{A}^{-1}$.

Definition 1.6. An $n \times n$ matrix $\mathbf{A}$ is positive-definite if $\boldsymbol{x}^\top \mathbf{A}\boldsymbol{x} > 0$, for all $\boldsymbol{x} \neq \mathbf{0}$. If $\mathbf{A} = \mathbf{A}^\top$, the matrix is symmetric positive-definite. ♢

Property 1.5. *A few important properties of symmetric positive-definite matrices* $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{p\times p}$ *are:*

1. $\operatorname{rank}(\mathbf{A}) = p$;
2. $\det(\mathbf{A}) > 0$;
3. *All eigenvalues of* $\mathbf{A}$ *are positive,* $\lambda_i > 0$;
4. $a_{ii} > 0$;
5. $a_{ii}a_{jj} - a_{ij}^2 > 0$, $i \neq j$;
6. $a_{ii} + a_{jj} - 2|a_{ij}| > 0$, $i \neq j$;
7. $\mathbf{A}^{-1}$ *is symmetric positive-definite;*
8. *All principal submatrices of* $\mathbf{A}$ *are symmetric positive-definite;*
9. *There exists a nonsingular lower triangular matrix* $\mathbf{L}$*, such that* $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$.

Property 1.6. *If*

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix} \in \mathbb{R}^{p+q\times p+q} \tag{1.37}$$

with $\mathbf{A}_{11} \in \mathbb{R}^{p\times p}$ *nonsingular, then* $\mathbf{A}$ *is nonsingular if and only if* $\mathbf{C} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}$ *is nonsingular. Further,*

$$\mathbf{A}^{-1} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}_{11}^{-1} - \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}^{-1}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}^{-1} \\ -\mathbf{C}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{C}^{-1} \end{pmatrix}. \tag{1.38}$$

It also holds that $\det(\mathbf{A}) = \det(\mathbf{A}_{11})\det(\mathbf{C})$.

1.6 Bibliographic Remarks

Many textbooks cover random vectors and the multivariate Gaussian distribution. One of my favorites is Mardia *et al.* (1979).

The content of Sections 1.2 to 1.4 are adapted from the script of the lecture *STA121 Statistical Modeling*, available via the URL www.math.uzh.ch/furrer/download/sta121/script_sta121.pdf. More precisely, these sections represent a summary of Chapters 2 and 3. Notice that Chapters 12 and 13 give a gentle introduction to large parts of this document.

The www.stat.berkeley.edu/∼paciorek/research/techVignettes/techVignette6.pdf gives further insight regarding solving efficiently quadratic forms and related quantities (for dense and sparse matrices).

Chapter 2

Time Series: Concepts, Models, Estimation and Forecasting

Observations taken over time often exhibit dependencies at short scales. We introduce simple but effective parametric models for such situations. Inherently, these parameters need to be matched with observations. This process is typically called fitting. A fitted time series model can be used for forecasting a prediction into the future.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter02.R.

2.1 Motivation

A scientific question or hypothesis leads to an experiment (in the larger sense) which may lead to data collected over regular time points. Based on the hypothesis and the data, statistical models for data taken over time are proposed. Here we consider parametric time series models. In this and the next chapter, we discuss approaches (estimation) to obtain "best guesses" (estimates) for these parameters. Depending on these estimates, we need to revise the model (inference) before proceeding to prediction, the statistical task to "foresee" unobserved observations.

Recall the lag-one correlation of the residuals depicted in Figure 1.2. In the first step, we need to relax the condition $\text{Cov}(\varepsilon_i, \varepsilon_{i+1}) = 0$ via time series, the result of extending the concept of a random sample to incorporate such covariances. More formally, we will use the following pragmatic definition of a time series.

Definition 2.1. A set of random variables $Y_1, \dots, Y_n$ are called a time series if the index refers to precise instances in time. We often refer to the series as $\{Y_t\}$ or simply as Y_t. Depending on the context, the series may start at 0 and/or stop at T. $\diamondsuit$

There is no universal agreement on the subscripts for a time series. We choose here an index from 1 to n, such that we have n random variables, n observations, etc. To emphasize the time series nature, we use the subscripts t instead of i.

Note that for theoretical considerations infinite time series are often used: $\{Y_t, t \in \mathbb{Z}\}$.

To simplify the theoretical discussion, we assume that the time series has a mean zero or that the mean is known. In practice, this is rarely the case, and we will discuss the implications of this assumption later.

2.2 Autocovariance Function and Stationarity

We now extend the concept of covariance and correlation to the time series setting.

Definition 2.2. Let $\{Y_t\}$ be a time series. The function

$$\gamma(s,t) = \operatorname{Cov}(Y_s, Y_t) \tag{2.1}$$

is called the autocovariance function of the time series $\{Y_t\}$. The autocorrelation function is obtained by normalizing the autocovariance by $\sqrt{\gamma(s,s)\gamma(t,t)}$, i.e.,

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}}. \tag{2.2}$$

$\diamondsuit$

In time series analysis, it is convenient to restrict the class of the studied time series by considering stationarity. A time series having a constant mean and an autocovariance function that depends on the magnitude $k = |s-t|$ only is called stationary. More precisely,

Definition 2.3. The series $\{Y_t\}$ is (weakly) stationary if

$$\mathrm{E}(Y_t) = \mu \tag{2.3}$$

$$\operatorname{Cov}(Y_s, Y_t) = \operatorname{Cov}(Y_{s+h}, Y_{t+h}) =: \gamma(|s-t|), \tag{2.4}$$

for all $s, t, s+h, t+h \in 1, \dots, n$. $\diamondsuit$

As we subset the times series by $1, \dots, n$, we will consider only integer values of $|s-t|$ which we subscript from now on, $\gamma(|s-t|) = \gamma_{|s-t|} = \gamma_k$. For a stationary time series, the autocorrelations, denoted by ρ_k, simplify to γ_k/γ_0.

Remark 2.1. If the time series at hand is not stationary (see, e.g., the CO_2 concentrations of Example 1.1 in Chapter 1), one needs first to remove the trend and/or seasonal component using, for example, linear regression. Once this task is done, the residual time series with no trend should "look" stationary in order to be able to impose the models presented in the next section. $\heartsuit$

A useful estimate of the autocovariance function is

$$\widehat{\gamma}_h = \frac{1}{n}\sum_{t=1}^{n-h}(y_t - \bar{y})(y_{t+h} - \bar{y}), \tag{2.5}$$

where $\bar{y} = \frac{1}{n}\sum_t y_t$. From this estimate, we derive an estimate of the autocorrelation function:

$$\widehat{\rho}_h = \frac{\widehat{\gamma}_h}{\widehat{\gamma}_0} = \frac{\displaystyle\sum_{t=1}^{n-h}(y_t - \bar{y})(y_{t+h} - \bar{y})}{\displaystyle\sum_{t=1}^{n}(y_t - \bar{y})^2}. \tag{2.6}$$

Such a set of estimates is illustrated in Example 2.1 and Figure 2.1.

Note that the estimator

$$\widehat{\gamma}_h = \frac{1}{n}\sum_{t=1}^{n-h}(Y_t - \bar{Y})(Y_{t+h} - \bar{Y}) \tag{2.7}$$

is (slightly) biased, even when replacing the denominator n by $n-h$. The bias is due to the correlation in the data.

Example 2.1. The data shown in the center panel of Figure 1.2 can be safely assumed to be a zero mean time series. The marginal variance seems to be constant. For the time being, let us assume that the autocorrelation structure does not change over time. R-Code 2.1 and Figure 2.1 illustrate the empirical autocorrelation function $\widehat{\rho}_k$. The dashed lines indicate crude confidence bands for the autocorrelations if an iid series would be used. ♣

R-**Code 2.1** Autocorrelation of the residuals obtained in R-Code 1.2. (See Figure 2.1.)

```r
lm2resid <- resid(lm2)
acf(lm2resid)
```

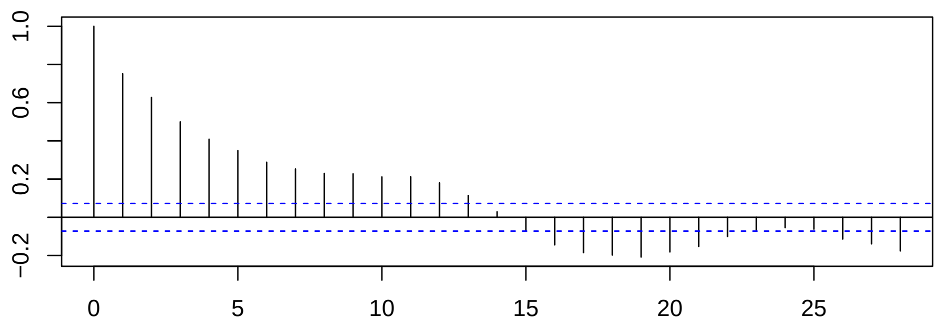


Figure 2.1: Autocorrelation plot of the detrended CO$_2$ time series (i.e., the residuals) shown in the center panel of Figure 1.2. (See R-Code 2.1.)

2.3 ARMA Models

This section sets the stage for a large class of (parametric) time series models.

2.3.1 Autoregressive AR(1) Model

The following introduces an intuitive model for time series, the so-called autoregressive model, where the current observation depends on the previous one:

$$Y_t = \phi Y_{t-1} + \varepsilon_t, \qquad \phi \neq 0 \tag{2.8}$$

with ε_t Gaussian white noise, i.e., $\varepsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$. This model is denoted with AR(1).

In practice, t in (2.8) varies between 1 and n, and some initial condition has to be assumed, for example, $Y_0 = 0$ or $Y_1 \sim \mathcal{N}(0, \sigma^2)$, or $Y_1 \sim \mathcal{N}(0, \sigma^2/(1-\phi^2))$. Further, we assume that $|\phi| < 1$ to ensure stationarity. In the case of $\phi > 1$, it becomes quickly clear that the series explodes. R-Code 2.2 illustrates this behavior. In what follows, we mainly focus on stationary series.

R-Code 2.2 "Manual" AR(1) model simulations. (See Figure 2.2.)

```r
set.seed(15)
n <- 250
epsilont <- rnorm(n, sd=5) # sigma =5
Y1t <- Y2t <- Y3t <- rep(0, times=n)
Y1t[1] <- Y2t[1] <- Y3t[1] <- epsilont[1]   # initial condition

### Case 1: |phi| < 1
for(i in 2:n) {    Y3t[i] <- 0.75 * Y3t[i-1] + epsilont[i]  }
ts.plot(Y3t, ylab=expression(Y[t]))         # stationary
### Case 2: |phi| > 1
for(i in 2:n) {    Y1t[i] <- -1.25 * Y1t[i-1] + epsilont[i] }
ts.plot(Y1t, ylab=expression(Y[t]))         # series explodes
### Case 3: |phi| = 1, boundary case
for(i in 2:n) {    Y2t[i] <- Y2t[i-1] + epsilont[i]          }
ts.plot(Y2t, ylab=expression(Y[t]))         # a so-called "random walk"
```

Assuming $Y_1 \sim \mathcal{N}\big(0, \sigma^2/(1-\phi^2)\big)$, there is a nice connection between an AR(1) model (with $1 \le t \le n$) and a multivariate Gaussian distribution, namely

$$\mathbf{Y} \sim \mathcal{N}_n(\mathbf{0}, \boldsymbol{\Sigma}), \qquad \boldsymbol{\Sigma} = \Big(\frac{\sigma^2}{1-\phi^2}\phi^{|i-j|}\Big)_{ij}. \tag{2.9}$$

Notice that the precision matrix, i.e., the inverse of the covariance matrix, is a tridiagonal matrix, i.e., a sparse matrix.

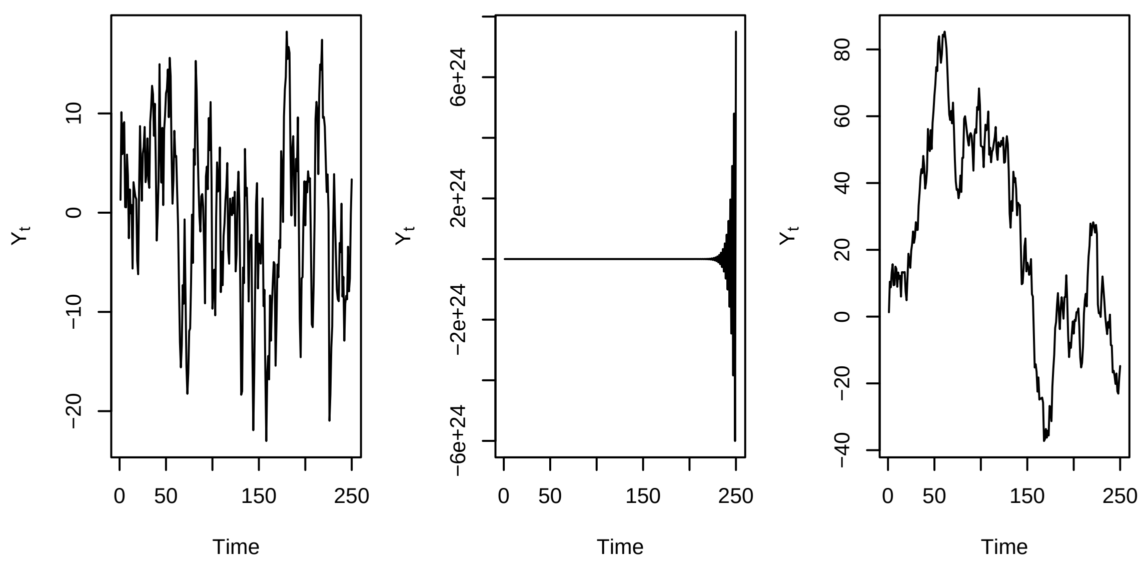


Figure 2.2: Three different AR(1) series with $|\phi| < 1$, $|\phi| > 1$, $\phi = 1$, respectively. Only the first time series is stationary. (See R-Code 2.2.)

2.3.2 AR(p) Models

Model (2.8) can be extended to an AR(p) model, defined as

$$Y_t = \phi_1 Y_{t-1} + \dots \phi_p Y_{t-p} + \varepsilon_t, \qquad \phi_p \neq 0 \tag{2.10}$$

with appropriate assumptions on $\phi_1, \dots, \phi_p$ and "initial" conditions, (e.g., distributions $Y_1, \dots, Y_p$).

The assumptions on $\phi_1, \dots, \phi_p$ guarantee that the series does not explode. Moreover, the series is stationary. When $p = 2$, we require

$$\phi_2 + \phi_1 < 1, \qquad \phi_2 - \phi_1 < 1, \qquad |\phi_2| < 1. \tag{2.11}$$

In case of a non-zero but constant mean, $Y_\bullet$ is replaced by $Y_\bullet - \mu$ in the AR(p) definition. More specifically, we have

$$Y_t = \mu + \phi_1(Y_{t-1} - \mu) + \dots + \phi_p(Y_{t-p} - \mu) + \varepsilon_t, \qquad \phi_p \neq 0. \tag{2.12}$$

Remark 2.2. There is a "notational" shortcut to write autoregressive models based on the concept of the *backshift* operator B defined as

$$BY_t = Y_{t-1}, \tag{2.13}$$

$$B^k Y_t = Y_{t-k}, \qquad k = 1, 2, \dots. \tag{2.14}$$

Another way to write the equation of an AR(p) is thus

$$(1 - \phi_1 B - \dots - \phi_p B^p) Y_t = \varepsilon_t. \tag{2.15}$$

It can be shown that such a time series Y_t is stationary if all the complex roots of the characteristic polynomial $(1 - \phi_1 x - \dots - \phi_p x^p)$ have a modulus strictly larger than 1. When $p = 1$, this condition implies that the solution of $1 - \phi_1 x = 0$ needs to be larger than one in absolute value, which is satisfied if $|\phi_1| < 1$. When $p = 2$, it is satisfied under (2.11). ♡

2.3.3 Moving Average Models

Similarly to autoregressive models, we can define *moving average* models where the present observation depends on a weighted average of white noise components. More precisely, a MA(1) is defined by

$$Y_t = \varepsilon_t + \theta\varepsilon_{t-1}, \qquad \theta \neq 0, \tag{2.16}$$

with, typically, $\varepsilon_t \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2)$, and a MA($q$) is defined by

$$Y_t = \varepsilon_t + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q}, \qquad \theta_q \neq 0. \tag{2.17}$$

Moving average MA(q) time series are always stationary. Additionally, we can add a non-zero mean μ on the right-hand side of (2.16) or (2.17).

Remark 2.3. Moving average time series can also be written using the backshift operator:

$$Y_t = (1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t, \qquad \theta_q \neq 0. \tag{2.18}$$

Note that sometimes, even a shorter notation is used for AR(p) and MA(q) time series, respectively:

$$\phi(B)Y_t = \varepsilon_t, \qquad \text{AR(p) time series,} \tag{2.19}$$

$$Y_t = \theta(B)\varepsilon_t, \qquad \text{MA(q) time series,} \tag{2.20}$$

where $\phi(B) = 1 - \phi_1 B - \cdots - \phi_p B^p$ and $\theta(B) = 1 + \theta_1 B + \cdots + \theta_q B^q$. ♡

2.3.4 ARMA and ARIMA Models

It is possible to combine AR(p) and MA(q) models into so-called ARMA(p, q) models. An ARMA(p, q) time series is defined by

$$Y_t = \phi_1 Y_{t-1} + \cdots + \phi_p Y_{t-p} + \varepsilon_t + \theta_1\varepsilon_{t-1} + \cdots + \theta_q\varepsilon_{t-q}, \qquad \phi_p \neq 0, \theta_q \neq 0, \tag{2.21}$$

Besides subtracting it, a possibility to "circumvent" a constant (non-zero) mean term is to study the time series of the differences of the observations: $\nabla Y_t := Y_t - Y_{t-1}$, where ∇ is the differentiation (or "nabla") operator. This approach leads to so-called "integrated ARMA", or "ARIMA" model. The differentiation can be quite fancy; thus, we have very flexible models.

By definition, a time series $\{Y_t\}$ is an ARIMA($p, 1, q$) time series if $\{\nabla Y_t\}$ is an ARMA(p, q) time series. The differentiation operator may even be applied more than once: a time series $\{Y_t\}$ is an ARIMA(p, d, q) time series if $\{\nabla^d Y_t\}$ is an ARMA(p, q) time series. The three letters (p, d, q) of ARIMA refer to the AR, I, and MA part, respectively. An ARMA($p, 0$) time series is an AR(p), an ARMA($0, q$) time series is a MA(q) time series and, finally, an ARIMA($p, 0, q$) time series is an ARMA(p, q) time series.

A good way to guess which kind of model (AR, MA, ARMA, or even more complicated models) should be chosen to study our data is to have a look at the empirical autocorrelation function (ACF); and compare it to the theoretical ACF of an AR, MA, ARMA, time series. This task will be done in the next section.

Remark 2.4. As for AR or MA models themselves, it is possible to write (2.21) shortly as $\phi(B)Y_t = \theta(B)\varepsilon_t$. Moreover, the backshift operator can also be used instead of the nabla operator $(1-B)Y_t = \nabla Y_t$ and, more generally, $(1-B)^d Y_t = \nabla^d Y_t$. ♡

2.4 Estimation of ARMA Parameters

In Section 2.2, we saw quite intuitive estimates of $\widehat{\rho}_k$. However, we are not primarily interested in the estimation of these ACF, but the parameters $\{\phi_1, \dots, \phi_p\}$ and/or $\{\theta_1, \dots, \theta_q\}$. For simplicity, we focus on autoregressive models, but much of what will be said can be carried over to moving average models as well.

Estimation $\{\phi_1, \dots, \phi_p\}$ of can be carried out based on likelihood estimation, method of moment estimation or Yule–Walker equations. The popular Yule–Walker estimates are based on solving the so-called Yule–Walker equations

$$\gamma_k = \phi_1\gamma_{k-1} + \dots + \phi_p\gamma_{k-p}, \qquad k = 1, \dots, p, \tag{2.22}$$
$$\sigma^2 = \gamma_0 - \phi_1\gamma_1 - \dots - \phi_p\gamma_p. \tag{2.23}$$

In practice, we use (2.5) to estimate the autocovariance and then solve the previous system.

For an AR(1) model, (2.22) yields $\gamma_1 = \phi_1\gamma_0$ and thus $\widehat{\phi}_1 = \widehat{\gamma}_1/\widehat{\gamma}_0$, i.e., the estimated autocorrelation. Further, $\widehat{\sigma}^2 = \widehat{\gamma}_0 - \widehat{\gamma}_1^2/\widehat{\gamma}_0$. Based on these equations, we note that the solutions of Yule–Walker are simpler when expressed in terms of autocorrelations.

For an AR(2) model, we have a 2×2 system and using the symmetry $\gamma_{-k} = \gamma_k$, we have

$$\widehat{\phi}_1 = \frac{\widehat{\rho}_1(1-\widehat{\rho}_2)}{1-\widehat{\rho}_1^2}, \qquad \widehat{\phi}_2 = \frac{\widehat{\rho}_2 - \widehat{\rho}_1^2}{1-\widehat{\rho}_1^2}. \tag{2.24}$$

Example 2.2. We reconsider the detrended CO_2 data and estimate $\widehat{\phi}_k$. R provides several functions for this task, for example `arima` or `ar`. Although slightly more complex for argument specification, we prefer the former as standard errors of the estimates are directly returned as well. The function `arima` estimates the parameters of a specified ARMA model with specified p (`order=c(2,0,0)`. As the series consists of residuals of a linear model fit, there is no need to estimate a mean (`include.mean=FALSE`), as shown in the first line of R-Code 2.3. The function returns (among other things) estimates $\widehat{\phi}_k$ and uncertainties thereof. Of course, we can also assess the uncertainty with a (parametric) bootstrap approach, as shown in the latter part of R-Code 2.3.

Not too surprisingly, the estimates $\widehat{\phi}_k$ are quite correlated $\text{Corr}(\widehat{\phi}_1, \widehat{\phi}_2) = -0.75$, illstrated in Figure 2.3. ♣

Remark 2.5. Yule–Walker estimators are essentially least squares estimators and are for AR(p) models optimal in the sense of the best asymptotic Gaussian distribution. See, e.g., Example 3.32 Shumway and Stoffer (2010) for specific examples. Yule-Walker equations could be set up for arbitrary ARMA(p, q) models. Alternative methods exist, including maximum likelihood approaches and tailored least squares approaches. ♡

R-Code 2.3 Fitting of an AR(2) model and bootstrapping estimation uncertainties. (See Figure 2.3.)

```
(f1 <- arima(lm2resid, order=c(2, 0, 0), include.mean=FALSE))
##
## Call:
## arima(x = lm2resid, order = c(2, 0, 0), include.mean = FALSE)
##
## Coefficients:
##          ar1    ar2
##        0.651  0.142
## s.e.  0.037  0.037
##
## sigma^2 estimated as 0.0941:  log likelihood = -174.01,  aic = 354.03
(phi <- f1$coef)                    # Estimates of phi (as above, of course)
##     ar1     ar2
## 0.65128 0.14243
(se <- sqrt(diag(f1$var.coef)))     # standard errors of phi
##      ar1      ar2
## 0.036783 0.036827
### We now bootstrap from a model with the same estimates and estimate again.
N <- 200                            # Number of bootstrap replicates
bootsam <- array(0, c(N,2))
for (i in 1:N) {
  ab <- arima.sim(n=f1$nobs, list(ar=phi), innov=sample(f1$resid))
  bootsam[i,] <- arima(ab, order=c(2, 0, 0), include.mean=FALSE)$coef
}
### Visualizing the results, bivariate left and marginal right panel:
plot(bootsam, xlab=expression(hat(phi)[1]), ylab=expression(hat(phi)[2]))
abline(v=phi[1], h=phi[2], col=3)
abline(v=phi[1]+se[1]*qnorm(c(.05,.95)),
    h=phi[2]+se[2]*qnorm(c(.05,.95)), col=4, lty=2)
lines(ellipse::ellipse(f1$var.coef, cent=f1$coef, level=.9), col=2)
hist(bootsam[,1], prob=T, xlim=c(0,1), main="", xlab=expression(hat(phi)))
hist(bootsam[,2], prob=T, add=T)
abline(v=phi, col=3)
abline(v=phi[1]+se[1]*qnorm(c(.05,.95)), col=4, lty=2)
abline(v=phi[2]+se[2]*qnorm(c(.05,.95)), col=4, lty=2)
```

The autocorrelation and partial autocorrelation function can be used to infer the “nature” of the ARMA model, as illustrated by the following examples.

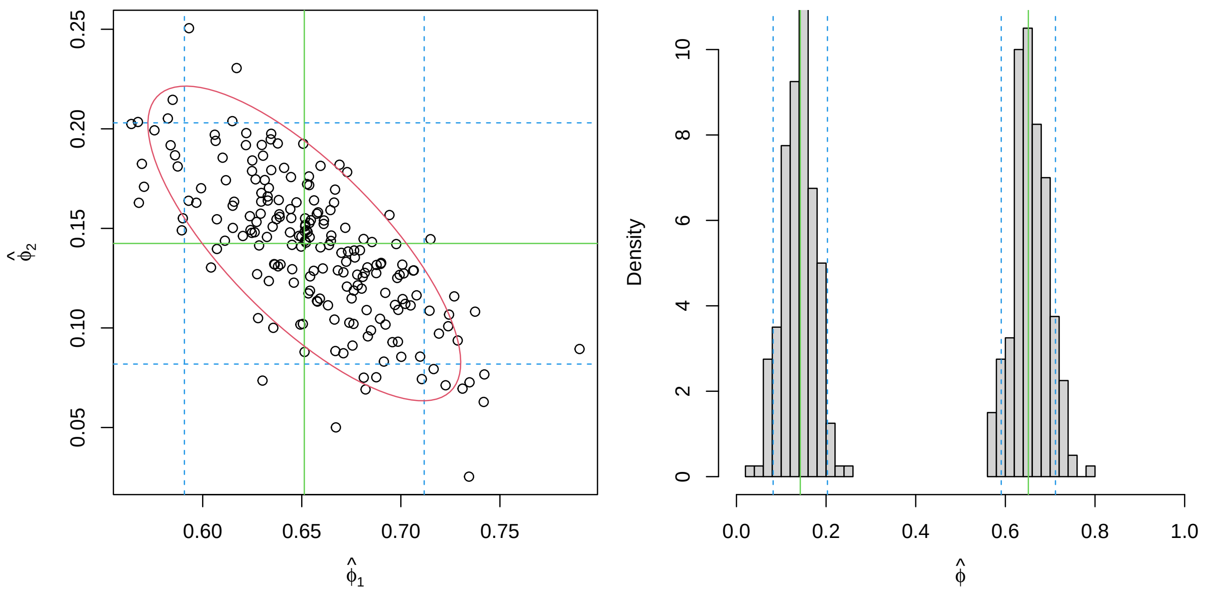


Figure 2.3: Parametric bootstrap of a AR(2) model. Left panel gives 200 bootstrapped estimates of $(\widehat{\phi}_1^{(\ell)}, \widehat{\phi}_2^{(\ell)})$ an AR(2) model. The right panel gives the marginal histograms. Green lines indicate the estimates. Blue dotted lines give univariate 90% confidence intervals. The red ellipse in the left panel is a 90% confidence region based on the original variance estimate. (See R-Code 2.3.)

The ACF of an MA(1) time series $Y_t = \theta\varepsilon_{t-1} + \varepsilon_t$ is obtained from

$$\gamma_h = \operatorname{Cov}(\theta\varepsilon_{t-1} + \varepsilon_t, \theta\varepsilon_{t-1+h} + \varepsilon_{t+h}) = \begin{cases} (\theta^2+1)\sigma^2, & h = 0, \\ \theta\sigma^2, & h = 1, \\ 0, & h \geq 2. \end{cases} \tag{2.25}$$

For an AR(1) time series, we have the autocorrelation function

$$\gamma_h = \sigma^2 \frac{\phi^h}{1-\phi^2} \qquad \text{and thus} \qquad \rho_h = \phi^h. \tag{2.26}$$

Another similar concept is the partial autocorrelation function (PACF). It is essentially the correlation between Y_{t-k} and Y_t "conditional" on knowing $Y_{t-k+1}, \dots, Y_{t-1}$. The PACF is denoted with ϕ_{kk} and by definition $\phi_{11} = \rho_1$.

We have, for example, (i) for an AR(1) time series, we have $\phi_{kk} = 0$ for all $k > 1$; (ii) for an AR(2) time series, we have $\phi_{11} = \rho_1/(1-\rho_2)$, $\phi_{22} = \rho_2$, and $\phi_{kk} = 0$ for all $k > 2$; (iii) for an MA(1) time series, we have $\phi_{22} = -\theta^2/(1+\theta^2+\theta^4) < 0$ and then tails off.

In general, we can show that

1. for an AR(p) model, the PACF cuts off after p lags and the ACF tails off;

2. for a MA(q) model, the ACF cuts off after q lags and the PCAF tails off;

3. for a ARMA(p, q) model, the ACF and the PCAF tail off.

These properties can be used to infer the model type from an ACF or PACF plot. More precisely, starting from data $\{y_t\}$ of a time series, we estimate ACF and PACF and then decide on an AR(p), MA(q) or ARMA(p, q) model.

Example 2.3. Returning once again to our CO_2 data, we see that the estimates $\widehat{\phi}_1$ for an AR(1), AR(2) and AR(3) are similar, especially for the latter two. The models imply certain structures on the ACF and the PACF. The function *ARMAacf* computes the theoretical ACF and the PACF based on estimates $\widehat{\phi}_k$. Because an AR(2) seems to be sufficient, the resulting theoretical differences are only visually pronounced between AR(1) and AR(p), $p > 1$. More specifically, the partial autocorrelation function has two "sufficiently" large values, implying that an AR(2) should be used. For AR(1), we have a single non-zero value, estimates of $\widehat{\phi}_k$, $k > 2$ in higher order autoregressive models are very close to zero. ♣

R-Code 2.4: Fitting of autoregressive models and comparison of ACF plots with theoretical values. (See Figure 2.4.)

```
(phi1 <- arima(lm2resid, order=c(1, 0, 0), include.mean=FALSE)$coef)
##     ar1
## 0.75899
(phi)
##     ar1     ar2
## 0.65128 0.14243
(phi3 <- arima(lm2resid, order=c(3, 0, 0), include.mean=FALSE)$coef)
##       ar1       ar2       ar3
##  0.654461  0.157943 -0.023764
### We now compare the fits with theoretical values:
l <- 28
acf(lm2resid)
points(0:l, ARMAacf(ar=phi,  l=l), col=2, pch=19)
points(0:l, ARMAacf(ar=phi3, l=l), col=3, pch=20)
points(0:l, ARMAacf(ar=phi1, l=l), col=4, cex=1.5, lwd=2)
pacf(lm2resid)
points(1:l, ARMAacf(ar=phi,  l=l, pacf=TRUE), col=2, pch=19)
points(1:l, ARMAacf(ar=phi3, l=l, pacf=TRUE), col=3, pch=20)
points(1:l, ARMAacf(ar=phi1, l=l, pacf=TRUE), col=4, cex=1.5, lwd=2)
### Note that without an order, the "best" model is estimated:
ar( lm2resid)$order # but AIC selects a too complicated model.
## [1] 16
### BIC selects a good one, but the calculation needs to be done manually.
```

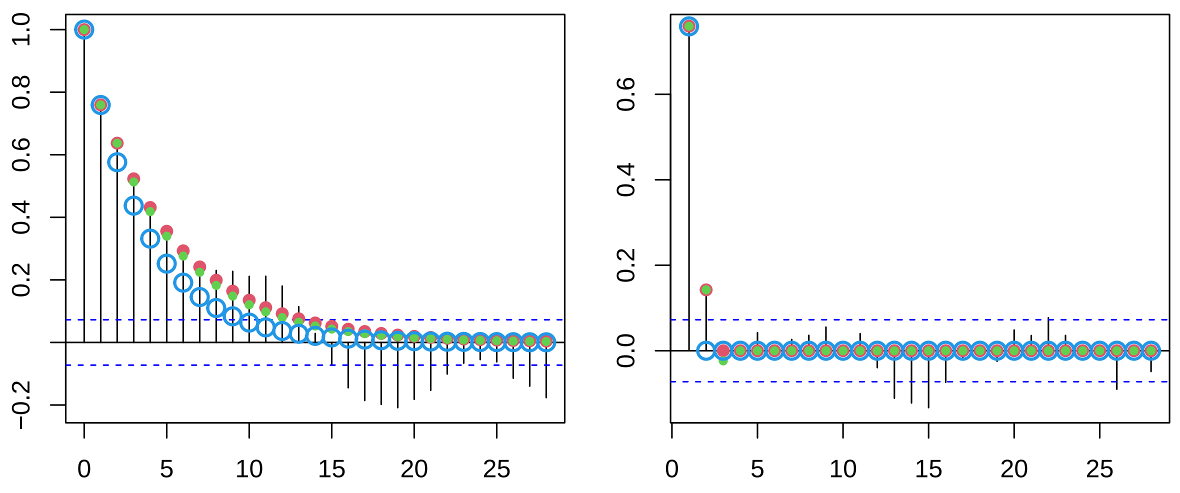


Figure 2.4: Autocorrelation and partial-autocorrelation plot of the detrended time series with superimposed theoretical values based on the estimated AR(p) model ($p = 2$ red, $p = 1$ blue, $p = 3$ green). (See R-Code 2.4.)

2.5 Prediction

2.5.1 Terminology

Here we introduce some definitions:

Definition 2.4. Let $\{Y_t\}$ be a time series and $\{y_t\}$ a realization thereof. We define:

- *prediction*: inferring the value or distribution of a quantity based on observations;
- *forecasting*: inferring the time series at future time points based on $\{y_t\}$;
- *backcasting*: inferring Y_t, $t = 1, \dots, n$ based on $y_1, \dots, y_n$;
- *filtering*: removing from y_t some unwanted (noise) component or feature. $\diamondsuit$

The term *smoothing* is sometimes used for backcasting and implies filtering. Smoothing is often associated with assessing a general trend or a very smooth curve describing the data. In particular sciences, the term *hindcasting* is used for backcasting.

As for ‘estimation’, ‘estimator’, ‘estimate’, and ‘to estimate’, we have the analogies ‘prediction’, ‘predictor’, ‘prediction’, and ‘to predict’.

Differentiating a time series filters out a linear trend and is a simple example of a linear filter. A simple moving average (unweighted or weighted) filters out the noise and is another example of a linear filter.

For prediction, we use the following generic notation:

$$p(\text{ quantity } ; \text{ data }). \tag{2.27}$$

We use the actual data for prediction; we have the corresponding random quantities for a predictor. The following list illustrates the uses of this notation by giving (too simple) examples:

$$p\big(Y_{n+1}; \{y_1, \dots, y_n\}\big) = p\big(Y_{n+1}; \boldsymbol{y}\big) = \frac{1}{n}\sum_{t=1}^{n} y_t : \qquad \text{(one-step head) prediction,} \tag{2.28}$$

$$p\big(Y_{n+2}; \{Y_1, \dots, Y_n\}\big) = p\big(Y_{n+2}; \mathbf{Y}\big) = Y_n : \qquad \text{(two-step head) predictor,} \tag{2.29}$$

where we have classically written the vectors $\mathbf{Y} = (Y_1, \dots, Y_n)^\top$ and $\boldsymbol{y} = (y_1, \dots, y_n)^\top$

2.5.2 Forecasting AR Models

Even though we are forecasting time series, we use the term prediction to introduce the concept, many due to its universality. Suppose we observe a time series $\{Y_t\}$. One way to construct predictors $p(Y_{n+m}; \{Y_1, \dots, Y_n\})$ is based on minimizing some criterion, such a

$$\mathrm{E}\Big(\big(Y_{n+m} - p(Y_{n+m}; \{Y_1, \dots, Y_n\})\big)^2\Big). \tag{2.30}$$

(Here, we minimize the expected squared loss.) For this particular criterion, the *best predictor* (BP) is the conditional expectation.

Remark 2.6. The latter fact can be shown in two steps. For simplicity, consider two random variables X and Y and we attempt to predict Y from X. First, it is shown that the best prediction using a constant c in terms of mean squared error is $c = \mathrm{E}(Y)$, i.e., $c = \mathrm{E}(Y)$ minimizes $\mathrm{MSE} = \mathrm{E}\big((Y - c)^2\big)$. In a second step, we consider predicting Y by some function of X, in our notation $p(Y; X)$, by minimizing $\mathrm{MSE} = \mathrm{E}\big((Y - p(Y; X))^2\big)$. Using the property $\mathrm{E}(Y) = \mathrm{E}\big(\mathrm{E}(Y \mid X)\big)$, we write

$$\mathrm{MSE} = \mathrm{E}\big((Y - p(Y; X))^2\big) = \mathrm{E}\Big(\mathrm{E}\big((Y - p(Y; X))^2 \mid X\big)\Big). \tag{2.31}$$

For every x, the inner expectation is minimized by setting $p(Y; X) = \mathrm{E}(Y \mid X = x)$ (shown in the first step). Thus $p(Y; X) = \mathrm{E}(Y \mid X = x)$. ♡

The conditional expectation in the case of a Gaussian framework is tractable, see equation (1.19), but hardly in any other case. Hence, we restrict the predictor to be a linear function of $Y_1, \dots, Y_n$, that means

$$p\big(Y_{n+m}; \{Y_1, \dots, Y_n\}\big) = \lambda_0 + \sum_{i=1}^{n} \lambda_i Y_{n-1-i} \tag{2.32}$$

In the case of a zero mean, we have $\lambda_0 = 0$, and the other coefficients can be obtained through plugging the linear predictor in (2.30) and expressing the expectations in terms of the autocovariance terms γ_k or by using the property that

$$\mathrm{E}\Big(\big(Y_{n+m} - p(Y_{n+m}; \{Y_1, \dots, Y_n\})\big)Y_t\Big) = 0, \quad t = 1, \dots, n. \tag{2.33}$$

The weights of the *best linear predictor* (BLP) are given by the solution of the following linear system

$$(\gamma_{i-j})_{i,j}(\lambda_i)_i = (\gamma_{m-1+i})_i, \qquad \text{written as} \qquad \boldsymbol{\Gamma}_n \boldsymbol{\lambda}_n = \boldsymbol{\gamma}_n \tag{2.34}$$

and the predictor is

$$p\big(Y_{n+m};\{Y_1,\dots,Y_n\}\big)=\boldsymbol{\gamma}_n^\top\boldsymbol{\Gamma}_n^{-1}(Y_n,\dots,Y_1)^\top. \tag{2.35}$$

The mean squared prediction error is

$$\gamma_0-\boldsymbol{\gamma}_n^\top\boldsymbol{\Gamma}_n^{-1}\boldsymbol{\gamma}_n. \tag{2.36}$$

Notice that for an AR(1) model, the m-step-ahead forecast is:

$$p\big(Y_{n+1};\{y_1,\dots,y_n\}\big)=\phi^m y_n \tag{2.37}$$

with mean squared prediction error

$$\sigma^2\frac{1-\phi^{2m}}{1-\phi^2}. \tag{2.38}$$

In R for classes `Arima` (output of `arima`) and `ar` (output of `ar`, and its particular methods `ar.mle`, etc.) the R-function `predict` can be used.

In the case of a non-constant mean, i.e., the time series is not stationary in the mean, prediction heavily depends on the mean function. Hence, care needs to be taken to model the first moment properly. Similar to regression settings, higher-order polynomials and non-parametric smoothers are unsuitable for extrapolation beyond very few time steps.

Example 2.4 including R-Code 2.5 and resulting figure give some insight in prediction and prediction uncertainties.

Example 2.4. To illustrate prediction in R, we revisit the *annual* CO_2 series of Example 1.1 of Chapter 1. We restrict our knowledge to the 7 previous years for a simpler visualization and better discrimination of different prediction methods. With a standard AR(1) model, we have a clear violation of the stationarity assumption, and thus the resulting prediction uncertainties are huge, black lines in Figure 2.5. The mean prediction levels of the mean of the data and the prediction interval converge to a "half"-width of $\big(\widehat{\sigma}^2/(1-\widehat{\phi}^2)\big)^{1/2}\approx 6.57$, by (2.38).

The R function `arima()` and thus `predict.arima()` provides to mechanism to include additional predictors. When adding a simple linear term, the prediction interval shrinks tremendously (red lines in Figure 2.5). ♣

Remark 2.7. In practice, the matrix $\boldsymbol{\Gamma}$ is not expressed and thus to calculate (2.35) or (2.36) no (possibly large) system is solved. Recursive algorithms exist; the most widely known is the Durbin–Levinson algorithm. The same algorithm additionally gives the means to calculate the PACF of a stationary time series. ♡

R-Code **2.5** AR prediction with and without covariate. (See Figure 2.5.)

```r
n <- 7
m <- 8                          # how many predictions ahead
CO2 <- tail(am, n)$CO2          # last n=7 annual CO2 readings

a1 <- arima(CO2, order=c(1, 0, 0))
a2 <- arima(CO2, order=c(1, 0, 0), xreg=(1:n))
pr1 <- predict(a1, n.ahead=m)
pr2 <- predict(a2, n.ahead=m, newxreg=n+1:m)

plot(CO2, xlim=c(1, n+m), ylim=c(385, 430))
abline(h=mean(CO2), col="gray")
matlines(n+1:m, pr1$pr+cbind(0, 2*pr1$se, -2*pr1$se), col=1, lty=1)
matlines(n+1:m, pr2$pr+cbind(0, 2*pr2$se, -2*pr2$se), col=2, lty=1)
legend("topleft", legend=c("AR( 1 )", "AR( 1 ) + trend"),
    col=c(1, 2), lty=1, bty="n")
```

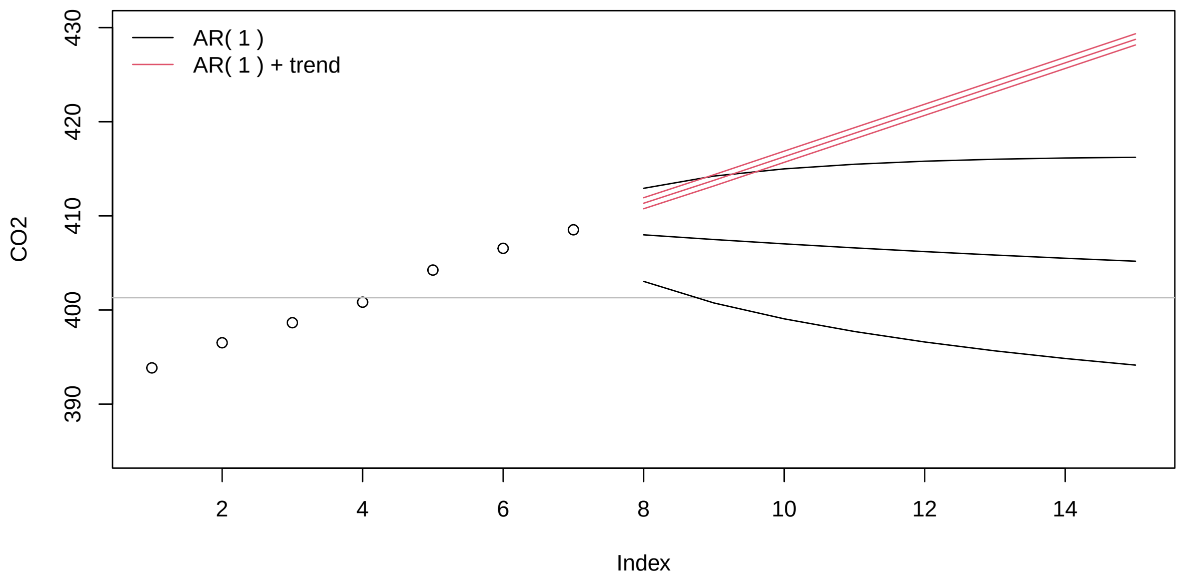


Figure 2.5: AR prediction with and without a covariate. The gray line gives the mean of the seven observations used for estimation and prediction. (See R-Code 2.5.)

2.5.3 Predicting ARIMA Models

Consider a time series $\{Y_t\}$ with a linear trend, that means with mean $\mu_t = \beta_0 + \beta_1 t$. Then the differentiated series has a constant mean because $\nabla Y_t = Y_{t+1} - Y_t$ implies $\mathrm{E}(\nabla Y_t) = \beta_1$.

Such differentiated models lead to the so-called moving average integrated autoregressive (ARIMA) time series. However, there are subtle differences between differentiating and explicitly modeling the trend. Models based on differenced sequences are particularly suitable for series that exhibit one or several shocks. Consider

$$Y_i = \begin{cases} -\mu + \phi(Y_{i-1} + \mu) + \varepsilon_i & \text{if } i = 1, \dots, k \\ +\mu + \phi(Y_{i-1} - \mu) + \varepsilon_i & \text{if } i = k+1, \dots, n. \end{cases} \tag{2.39}$$

That means, between time point k and $k+1$, the mean changes from $-\mu$ to $+\mu$. A constant mean model leads to an estimate close to zero, and the squared residuals are all close to μ^2. When differentiating, there is only one jump (shock) of size 2μ.

The difference between differentiating and explicitly modeling a linear trend is especially recognizable when predicting, as illustrated in the following example.

Example 2.5. We retake the last seven years of annual Mouna Loa CO_2 reading and predict with ARI models in R-Code 2.6. Including the differentiation, we can mimic the linear trend to a certain degree. The uncertainties, however, tremendously increase for predictions in the "far" future. Adding a linear trend to the model with differentiation reduces again the uncertainties compared to without the trend but increases them compared without differentiation. There is a slight shrinkage effect with the model with differentiation when comparing both models with a linear trend. The prediction uncertainties of the models with additional regressors (`a2`, `a4`) are smaller than the ones without. Overall, `a2` performs best in terms of AIC, but the model may be too optimistic with respect to the prediction uncertainties. ♣

R-Code 2.6 ARIMA prediction. (See Figure 2.6.)

```r
a3 <- arima(CO2, order=c(1, 1, 0))
a4 <- arima(CO2, order=c(1, 1, 0), xreg=(1:n))
pr3 <- predict(a3, n.ahead=m)
pr4 <- predict(a4, n.ahead=m, newxreg=n+1:m)

plot(CO2, xlim=c(1,n+m), ylim=c(385, 430))
matlines(n+1:m, pr3$pr+cbind(0, 2*pr3$se, -2*pr3$se), col=3, lty=1)
matlines(n+1:m, pr4$pr+cbind(0, 2*pr4$se, -2*pr4$se), col=4, lty=1)
matlines(n+1:m, pr1$pr+cbind(0, 2*pr1$se, -2*pr1$se), col=1, lty=2)
matlines(n+1:m, pr2$pr+cbind(0, 2*pr2$se, -2*pr2$se), col=2, lty=2)
legend("topleft", legend=c("AR( 1 )", "AR( 1 ) + trend", "ARI( 1, 1 )",
    "ARI( 1, 1 ) + trend"), col=1:4, lty=c(2, 2, 1, 1), bty="n")
```

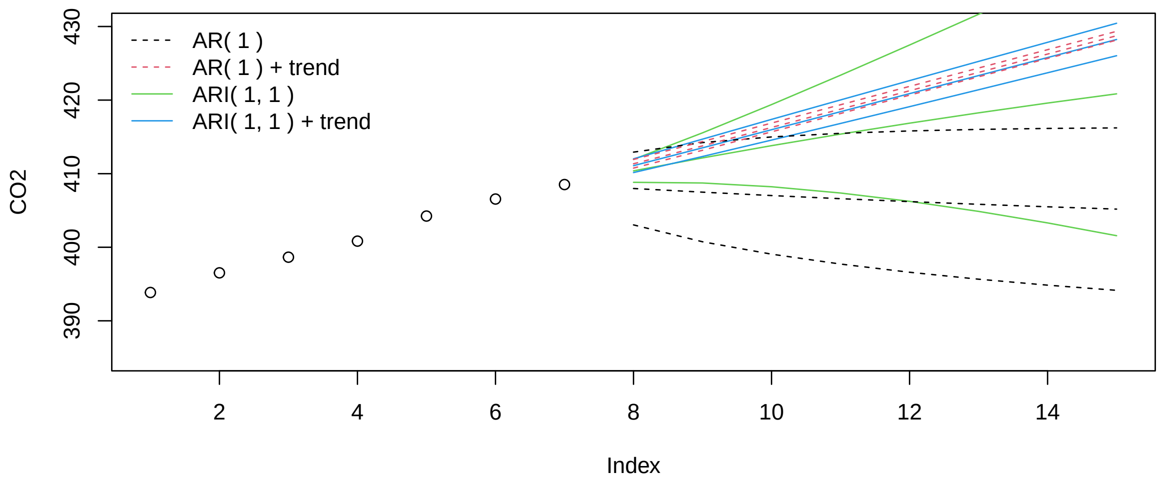


Figure 2.6: ARIMA prediction. The dashed lines are identical to the ones of Figure 2.5. (See R-Code 2.6.)

Similar to the regression setting, the residuals of the fitted model should exhibit Gaussian behavior. Often a square root or logarithmic transformation of the data may improve the model.

In all the prediction/forecast formulas, the BLP requires knowledge of the model parameters. These are, in practice, not known and need to be estimated first. One often resorts to plug-in prediction, and the BLP is only approximately the BLP.

Chapter 3

Time Series: Spectral Methods and Examples

There is an alternative way to analyze time series data called the spectral approach. Spectral analysis is based on a sinus/cosine basis.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter03.R.

3.1 Motivation

We often have a cyclic or "repetitive" structure in time series. A typical example is the annual fluctuation in the CO_2 observations. For the discussion of this chapter, we assume that we have oscillating observations.

A simple model for a zero-mean oscillating time series $\{Y_t\}$ is

$$Y_t = a \cdot \cos(2\pi\,\omega\,t + \phi) + \varepsilon_t, \tag{3.1}$$

where a is the amplitude, ϕ is the phase shift (or simply phase), ω is the frequency, and ε_t is the classical noise (for simplicity, assumed iid). Classical trigonometric identities allow a rewriting

$$Y_t = u_1 \cdot \cos(2\pi\,\omega\,t) + u_2 \cdot \sin(2\pi\,\omega\,t) + \varepsilon_t, \tag{3.2}$$

where $u_1 = a\cos(\phi)$, $u_2 = -a\sin(\phi)$. Using the inverse transformations $a = \sqrt{u_1^2 + u_2^2}$ and $\phi = \operatorname{atan}2(u_2, u_1)$ we have a one-to-one representation, of course.

Provided the frequency ω is known, we can estimate the coefficients u_k via a linear model. If the frequency ω is not known, we include several different frequencies $\omega_1, \dots, \omega_K$

$$Y_t = \sum_{k=1}^{K} u_{k1} \cdot \cos(2\pi\,\omega_k t) + u_{k2} \cdot \sin(2\pi\,\omega_k t) + \varepsilon_t \tag{3.3}$$

and later "choose the best" frequency(ies), for example based on "significant" estimates $\widehat{u}_{k1}$ and $\widehat{u}_{k2}$.

Example 3.1. We now illustrate this "frequency" representation using artificial data. To further simplify the discussion, we assume we have a time series with three frequencies but no noise component. R-Code 3.1 first constructs the data based on (3.3) with $K = 3$ (also shown in Figure 3.1) and then uses least squares to estimate $\{u_{k1}, u_{k2}\}$, $k = 1, \dots, 4$, with the three correct frequencies and one supplementary (i.e., unnecessary) frequency.

As the frequencies from the construction and fitting coincide, we can exactly recover the initial parameters, of course. ♣

R-Code 3.1: Decomposition of artificial data (See Figure 3.1.)

```
mts <- 1:(12*15)            # Mimic monthly data for 15 years
n <- length(mts)            # i.e., 180 observations
a <- c(1, .5, .2)           # Three amplitudes
phi <- c(0, pi/3, pi/2)     # and three phase shifts.
w <- c(1/12, 1/4, 1/(12*5)) # Here we have n*w cycles to observe
Mean <- 1.1
annual   <- a[1] * cos(2*pi*w[1]* mts + phi[1])
seasonal <- a[2] * cos(2*pi*w[2]* mts + phi[2])
slow     <- a[3] * cos(2*pi*w[3]* mts + phi[3])
y <- Mean + annual + seasonal + slow   # no additional noise for the moment

plot(y, type = "l", ylim = c(-1.5, 2.5), xlab = "t", ylab = "")
matlines(cbind(Mean, annual, seasonal, slow), lty=1, col="gray")
### The coefficients can be determined by classical regression:
out <- lm(y ~ cos(2*pi*w[1]* mts) + sin(2*pi*w[1]* mts) +
              cos(2*pi*w[2]* mts) + sin(2*pi*w[2]* mts) +
              cos(2*pi*w[3]* mts) + sin(2*pi*w[3]* mts) +
              cos(2*pi*1/24* mts) + sin(2*pi*1/24* mts))
print(round(unname(out$coef), 4)) # suppress long names
## [1]  1.100  1.000  0.000  0.250 -0.433  0.000 -0.200  0.000  0.000
all.equal(out$coef[c(2,4,6,3,5,7)], c(a*cos(phi),-a*sin(phi)),
          check.attributes=FALSE)
## [1] TRUE
beta <- out$coef[-1]        # Back transformation to amplitude and phase
dim(beta) <- c(2, 4)
round(rbind(a=sqrt(colSums(beta^2)), phi=atan2(beta[2,], beta[1,])), 5)
##      [,1]    [,2]    [,3]    [,4]
## a       1  0.5000  0.2000  0.0000
## phi     0 -1.0472 -1.5708 -1.1946
### Note the result of numerical instabilities of the last phase!
```

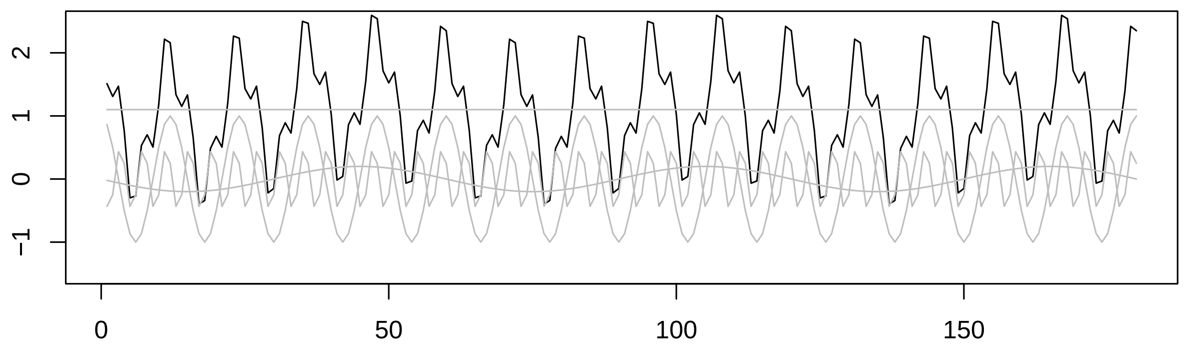


Figure 3.1: Decomposition of an artificial sample. The black line is the sum of the four gray lines. (See R-Code 3.1.)

Models (3.1) to (3.3) are in the classical setting of signal–noise (additive) decomposition. Thus a regression framework to estimate the "signal" is fully legitimate. We now extend that view and assume that the sine/cosine elements have a stochastic amplitude. To simplify, we start with assuming noise-free decomposition

$$Y_t = \sum_{k=1}^{K} U_{k1} \cdot \cos(2\pi\, \omega_k\, t) + U_{k2} \cdot \sin(2\pi\, \omega_k\, t), \qquad t = 1, \dots, n, \tag{3.4}$$

where $\{U_{k1}, U_{k2}\}$ are zero-mean independent random variables with variances σ_k. As often, we assume a Gaussian distribution for these.

With trigonometric identities, we can show that for a time series Y_t as given (3.4), we have

$$\gamma(h) = \sum_{k=1}^{K} \sigma_k^2 \cos(2\pi\, \omega_k\, h), \tag{3.5}$$

establishing a link to the last chapter.

As a regression approach may indicate which frequencies in (3.4) are substantial, a so-called periodogram summarizes the main frequencies (a more formal definition follows). For simplicity, assume that n is odd ($n = (K-1)/2$) the observed time series y_t can be *exactly* represented as

$$y_t = u_0 + \sum_{k=1}^{K} u_{k1} \cdot \cos(2\pi\, k/n\, t) + u_{k2} \cdot \cos(2\pi\, k/n\, t), \qquad t = 1, \dots, n, \tag{3.6}$$

(this is essentially a change of an orthogonal basis) and the periodogram at frequencies $\omega_k = k/n$ is given by

$$P(0) = u_0^2, \qquad P(k/n) = u_{k1}^2 + u_{k2}^2, \qquad k = 1, \dots, K = (n-1)/2. \tag{3.7}$$

Hence, the periodogram indicates which components in (3.4) are large (in magnitude) and which are not.

Example 3.2. The periodogram for Example 3.1 is

$$P(0) = 1.1, \quad P(1/12) = 1, \quad P(1/4) = 0.5, \quad P(1/60) = 0.2, \tag{3.8}$$

(compare row `a` at the end of R-Code 3.1) and zero for all other frequencies. ♣

Remark 3.1.

1. It can be shown that if U_1 and U_2 are iid Gaussian, then the corresponding frequency A and phase Φ are distributed according to a chi-squared with two degrees of freedom and an independent uniform $(-1/2, 1/2)$. An vice versa.
2. If n is even, an additional cosine term is added, $u_{n/2} \cos(2\pi\,(n/2)/n\,t) = u_{n/2}(-1)^t$.
3. The periodogram can also be derived from a decomposition (3.3), in which case it can be seen as a measure of the squared correlation of the data with sine/cosine functions at the corresponding frequencies. ♡

Instead of classically "regressing" the coefficients $\{u_{ki}\}$, very efficient algorithms exist. We introduce some ideas now.

3.2 Fourier Transform

A generalization of a sine/cosine basis is based on Fourier representation using the classical identity $\mathrm{e}^{\imath x} = \cos(x) + \imath \sin(x)$ with $\imath$ the unit imaginary number. The classical definition of a Fourier transform of a (sufficiently well-behaved) function $f(t)$ is

$$F(\xi) = \int_{-\infty}^{\infty} f(t)\, \mathrm{e}^{-2\pi\imath\xi t}\, \mathsf{d}t, \tag{3.9}$$

where $\imath$ is again such that $\imath^2 = -1$. The argument ξ represents the frequency in Hertz for t in seconds. There are many different flavors of the precise definition of the Fourier transform; normalizing constants and arguments often vary depending on the scientific domain. Often a transformation is indicated with a hat, a choice that does not work well in statistical terms.

Remark 3.2.

1. For an angular frequency $\omega = 2\pi\xi$, the inverse transform has the normalizing factor $1/(2\pi)$.
2. The Fourier transform of a real function is, in general, complex. Therefore one often defines the Fourier transform for complex functions $\mathcal{F} : \mathbb{C} \to \mathbb{C}$. With the notation used above, the inverse transform $\mathcal{F}^{-1} : \mathbb{C} \to \mathbb{C}$ takes the simple form

$$f(t) = \mathcal{F}^{-1}\big(\mathcal{F}(f)\big) = \int_{-\infty}^{\infty} F(\xi)\, \mathrm{e}^{2\pi\imath t\xi}\, \mathsf{d}\xi. \tag{3.10}$$

3. If the function f is periodic with fundamental period T, then the Fourier transform is an infinite sum of impulses where the impulse amplitudes are proportional to the Fourier coefficients of the function. ♡

We are interested in working with discrete times series and thus consider a discredited version of equation (3.9) for a finite number of values of the function.

Definition 3.1. For observed time series $y_1, \dots, y_n$, we define the discrete Fourier transform

$$F_j = \sum_{t=1}^{n} y_t \, \mathrm{e}^{-2\pi\imath\omega_j(t-1)}, \tag{3.11}$$

where $\omega_j = j/n$, $j = 0, \dots, n-1$ are the fundamental frequencies. The sequence $\{F_j\}$ is the discrete Fourier transform (DFT) of the sequence $\{y_k\}$. $\diamondsuit$

The frequencies should be seen in terms of a periodic function or sequence, observed at n points with period T. Often (including the definition above), we take $T = 1$ unit-less, and then ω_1 is in one cycle per sequence, and $2\pi\omega_1$ is in one radian per sequence.

Remark 3.3. If the concept of Dirac delta functions is known, the derivation of (3.11) is straightforward using the following representation. We write $f(t)$ in (3.9) as

$$f(t) = f_1\delta(t-1) + f_2\delta(t-2) + \cdots + f_n\delta(t-n), \tag{3.12}$$

where $f_1, \dots, f_n$ is are the observed values (say the time series y_t), and $\delta(x)$ is the Dirac delta function, 1 if $x = 0$ and zero otherwise. $\heartsuit$

The DFT can be written in matrix form $\mathbf{F} = \mathbf{W}\boldsymbol{f}$, where $\mathbf{F} = (F_1, \dots, F_n)^\top$ contains the DFT sequence of $\boldsymbol{f} = (f_1, \dots, f_n)^\top$ and the matrix $\mathbf{W} = (W_{ij})$ contains the elements $W_{ij} = \exp(-2\pi\imath(i-1)(j-1)/n)$. In other words

$$\begin{pmatrix} F_1 \\ \vdots \\ F_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & w & w^2 & w^3 & \dots & w^{n-1} \\ 1 & w^2 & w^4 & w^6 & \dots & w^{2(n-2)} \\ \vdots & & & & & \\ 1 & w^{n-1} & w^{2(n-1)} & w^{3(n-1)} & \dots & w^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}, \tag{3.13}$$

where $w = \exp(-2\pi\imath/n)$.

Remark 3.4. The matrix $\mathbf{W}$ and its elements have many properties that we summarize below (most are straightforward to show).

1. The second column of the matrix, $w^0, \dots, w^{n-1}$, are the n roots of unity. Hence, we often write w_n.

2. The powers of the entries of $\mathbf{W}$ can be simplified using the identity $\exp(-2\pi\imath \cdot j/n) = \exp(-2\pi\imath \cdot (j+kn)/n)$ for all $k \in \mathbb{Z}$. More specifically $w_n^2 = -1$, $w_n = w_n^2$, $w_n^n = 1$ and for n even $w_{n/2} = w_n^2$.

3. $\mathbf{W}$ is the Vandermonde matrix of the n roots of unity.

4. $\mathbf{W}^{-1}$ exists (property of a Vandermonde matrix) and thus $\boldsymbol{f} = \mathbf{W}^{-1}\mathbf{F}$. Further, the elements of $\mathbf{W}^{-1}$ are $\exp(2\pi\imath(i-1)(j-1)/n)/n$, i.e., a similarly defined Vandermonde matrix based on $1/w$ instead of w.

5. The matrix $\mathbf{U} = \mathbf{W}/\sqrt{n}$ is unitary, i.e., $\mathbf{U}^*\mathbf{U} = \mathbf{I}$, where * denotes the conjugate transpose. Therefore, the DFT is often defined based on $\mathbf{U}$.

6. For a real sequence $\boldsymbol{f}$, the Fourier sequence satisfies $F_{n-k} = \overline{F}_k$ and thus reduces the degrees of freedom of the DFT sequence to n.

7. Many books use a "C-language" indexing approach where the indices of vectors run from 0 to $n-1$. In such a setting, $F_r = \sum_{s=0}^{n-1} w^{rs} f_s$, $r = 0, \dots, n-1$. ♡

From equation (3.13), it seems that calculating a DFT sequence requires n^2 operations (matrix-vector multiplication). However, there are so-called divide-and-conquer approaches resulting in efficient algorithms. A divide-and-conquer strategy consists of three steps: (1) Divide the problem into subproblems of smaller size (typically into two subproblems). (2) Solve the small subproblems directly or solve each subproblem recursively by the same algorithm. (3) Combine the subproblems' individual solutions to get the original problem's solution.

In our setting, the second step consists of performing the DFT as two DFTs of length $n/2$, reducing the load from n^2 to $2 \times (n/2)^2 = n^2/2$. The operation count $T(n)$ for a general divide-and-conquer strategy of a problem of size n can be written by the recurrence relation

$$T(n) = aT(n/b) + O(n), \tag{3.14}$$

where a denotes the number of subproblems, b determines the fraction of subproblem size, and $O(n)$ is the operation count for the third step above. Typically, $T(1)$ is constant and $a = b$. The total operation count $T(n)$ depends on the recombination cost. Often, $O(n)$ has order $n^{\log_b a}$ in which the total count is reduced to $n^{\log_b a} \log(n)$. (If the recombination step is very costly, it will ultimately dominate the decomposition, and the algorithm has its operation count).

To introduce the so-called fast Fourier transform (FFT), we assume that n is a power of two. We start by reordering the even indexed and the odd indexed series and write the vector-vector multiplications in (3.13) as the sum of two vector-vector multiplications of half of the size. Each element W_{ij} can be rewritten in terms of a power of the principal $n/2$ root of unity (Property 2 of Remark 3.4). That means two FFTs of half the size. The recombination is

$$F_i = F_i^{\text{even}} + aF_i^{\text{odd}}, \qquad F_{i+n/2} = F_i^{\text{even}} - aF_i^{\text{odd}}, \tag{3.15}$$

where F_i^{even} and F_i^{odd} are the DFTs of the even and odd indexed sequence and where a is a power of w_n. To calculate F_i^{even} and F_i^{odd}, we restart the algorithm with $n/$ as the new problem size.

This FFT algorithm is most often termed the Cooley–Tukey FFT. Instead of grouping the original series, it is also possible to group the frequencies F_i in terms of even and odd indices. A similar divide-and-conquer approach exists. In both cases, $a = b = 2$ and the combination in the final step is $O(n) = 5n$ (one complex multiplication and complex addition), and thus, the algorithm has operation count $n \log_2 n$.

If case n is not a power of two, the operation count remains essentially the same, but the derivation gets more cumbersome. In practice, one often adds pseudo observations with zeros until reaching the next power.

In the time domain, the autocovariance determines the correlation structure over time. The spectral density introduced below is the corresponding quantity in the spectral domain.

Definition 3.2. If the autocovariance function of a time series is absolutely summable, then the spectral density is given by

$$f(\omega) = \sum_{h=-\infty}^{\infty} \gamma_h \, \mathrm{e}^{-2\pi \imath \omega h} = \gamma_0 + 2\sum_{h=1}^{\infty} \gamma_h \cos(2\pi \, \omega \, h), \qquad -1/2 \leq \omega \leq 1/2. \tag{3.16}$$

and the inverse transform is given by

$$\gamma_h = \int_{-1/2}^{1/2} f(\omega) \, \mathrm{e}^{2\pi \imath \omega h} \, \mathrm{d}\omega, \qquad h = 0, \pm 1, \pm 2, \dots \tag{3.17}$$

♢

Notice that $f(\omega) \geq 0$, $f(\omega) = f(-\omega)$. We would also have $f(\omega) = 0$, for $\omega < -1/2$ or $1/2 < \omega$. The spectral density is somewhat like a "un-normalized" probability density.

Example 3.3. For white noise, the autocovariance function is $\gamma(h) = \sigma^2$ for $h = 0$ and zero otherwise. Hence the spectral density is $f(\omega) = \sigma^2$ for $-1/2 \leq \omega \leq 1/2$. ♣

As Fourier pairs are one-to-one, the autocovariance function $\gamma(h)$ and the spectral density function $f(\omega)$ contain the same information. From a signal, we can thus analyze the ACF (expressing information in terms of lags) or the spectral density (expressing information in terms of cycles). Depending on the question, it is easier to handle the problem in the time domain or in the spectral domain. For example, analyzing the spectrum is much more adequate for identifying cycles or periodicities.

The spectral density can be estimated using estimates of the autocovariance function in 3.16. In practice, better approaches exist by slightly smoothing the time series.

Definition 3.3. The periodogram of a time series y_t is given by

$$I_k = |F_k|^2, \tag{3.18}$$

where F_k is the DFT of the series y_t. ♢

In the above definition, $|F_k|^2 = F_k\overline{F_k}$ and thus I_k is a real. Further, $I_0 = n\bar{y}$. The periodogram is a real, discrete "representation" the spectral density

$$I_k = F_k\overline{F_k} = \sum_{r=1}^{n} y_r \, \mathrm{e}^{-2\pi \imath rk/n} \sum_{s=1}^{n} y_s \, \mathrm{e}^{2\pi \imath sk/n} = \sum_{r=1}^{n} (y_r - \bar{y}) \, \mathrm{e}^{-2\pi \imath rk/n} \sum_{s=1}^{n} (y_s - \bar{y}) \, \mathrm{e}^{2\pi \imath sk/n} \tag{3.19}$$

$$= \sum_{r=1}^{n} \sum_{s=1}^{n} (y_r - \bar{y})(y_s - \bar{y}) \, \mathrm{e}^{-2\pi \imath (r-s)k/n} \tag{3.20}$$

$$= \sum_{h=-(n-1)}^{n-1} \sum_{t=1}^{n-|h|} (y_{t+|h|} - \bar{y})(y_t - \bar{y}) \, \mathrm{e}^{-2\pi \imath hk/n} = n \sum_{h=-(n-1)}^{n-1} \widehat{\gamma}_h \, \mathrm{e}^{-2\pi \imath hk/n}, \tag{3.21}$$

where in the last equality of (3.19), we used the fact that the sum of the n roots of unity sum to one.

Further, it is possible to show that the spectral density is the long-term average of the periodogram.

Example 3.4. R-Code 3.2 illustrates the links between (1) fitting some sine/cosine terms to observations, (2) estimating the spectrum, (3) DFT of the series.

Note that the function *spec.freq()* defines the spectrum scaling *1/frequency()*, defining the spectral density a density over the range from *-frequency()/2* to *frequency()/2*. The input is a simple vector *y*, and thus *frequency(x)* is by default one. ♣

R-**Code 3.2:** Complete example of spectral analysis for artificial data. (See Figure 3.2.)

```
### Suppose we do not know the frequencies at hand. We then have to
### estimate these for all possible 2*pi/n * j, j=0,1,...
sp <- spec.pgram(y, main="", sub="", taper=0, detrend=FALSE)
      # three large peaks (significant!)
      # Plots automatically the spectrum, see stats:::plot.spec

### Spectrum is evaluated for fundamental frequencies: (0:(n-1))/n
### but 0 is not included. Additionally, we have symmetry:
all.equal(1:(n/2)/n, sp$freq)
## [1] TRUE
### Determine the most important frequencies:
1/sp$freq[head(order(sp$spec, decreasing=TRUE))]
## [1] 12.0000  4.0000 60.0000  3.9130  4.0909 11.2500
### 12 = annual, 4 = seasonal and 60 = slowly varying cycle

### verify manually:
rbind(frequency=sp$freq[c(15, 45, 3)], spectrum=sp$spec[c(15, 45, 3)])
##                 [,1]  [,2]     [,3]
## frequency  0.083333  0.25 0.016667
## spectrum  45.000000 11.25 1.800000
### Note that the spectrum is plotted on a log scale. Here we have
### three distinct waves in our signal, and hence, we have a textbook case.
plot(sp$spec, type="h")
### Notice that the mean is filtered:
all.equal(spec.pgram(y,   plot=FALSE, taper=0, detrend=FALSE)$spec,
          spec.pgram(y+3, plot=FALSE, taper=0, detrend=FALSE)$spec )
## [1] TRUE
### We can do everything from scratch:
fftManual <- function(z) {     # From the help of fft (without "inverse")
  n <- length(z)
  k <- 0:(n-1)
  ff <- -2*pi * 1i * k/n
  vapply(1:n, function(h) sum(z * exp(ff*(h-1))), complex(1))
  # the last argument is to return a complex value.
}
### notice that exp(2*pi * 1i * k/n) = 1 for k=0
###             exp(2*pi * 1i * j/n) = exp(2*pi * 1i * (j+k*n)/n)

all.equal(fft(y), fftManual(y))    # Compare fftManual() against fft()
## [1] TRUE
F <- fftManual(y)
### There is redundancy because of the real entries.
### We observe the symmetry:
all.equal(F[1+(1:(n/2))], Conj(F[n+1-(1:(n/2))]))
## [1] TRUE
all.equal(as.double(F[1]/n), mean(y)) # first component carries the mean.
## [1] TRUE
FFc <- Re(F * Conj(F))/n
all.equal(FFc[1:(n/2)+1], sp$spec)
## [1] TRUE
### Here, we do have exactly the signal:
all.equal(FFc[c(15,45, 3)+1], colSums(beta^2)*n/4)
## [1] "Numeric: lengths (3, 2) differ"
### Putting things together, we get:
specManual <- function(xfft){
  N <- length(xfft)
  Nspec <- floor(N/2)
  freq <- seq.int(from = 1/N, by = 1/N, length.out = Nspec)
  spec <- (Re(xfft * Conj(xfft))/N)[2:(Nspec+1)]
  list(freq=freq, spec=spec)
}
all.equal(specManual(F),
    spec.pgram(y, plot=FALSE, taper=0, detrend=FALSE)[1:2])
## [1] TRUE
```

Example 3.5. We now add white noise to the signal of the artificial data constructed in Example 3.1. R-Code 3.3 illustrates that a random component with $\sigma = 0.3$ ("amplitude" stronger than seasonal and slowly varying component) still allows us to identify the three cycles.

The white noise shifts the spectrum, which is because white noise is a random signal with a constant power spectral density (see Example 3.3). Note that the spectrum (without the peaks) is not a good estimate of the variance of the errors. The reason is a slight smoothing when calculating the spectrum. To eliminate the later, use the argument `taper=0` in `spectrum()`. ♣

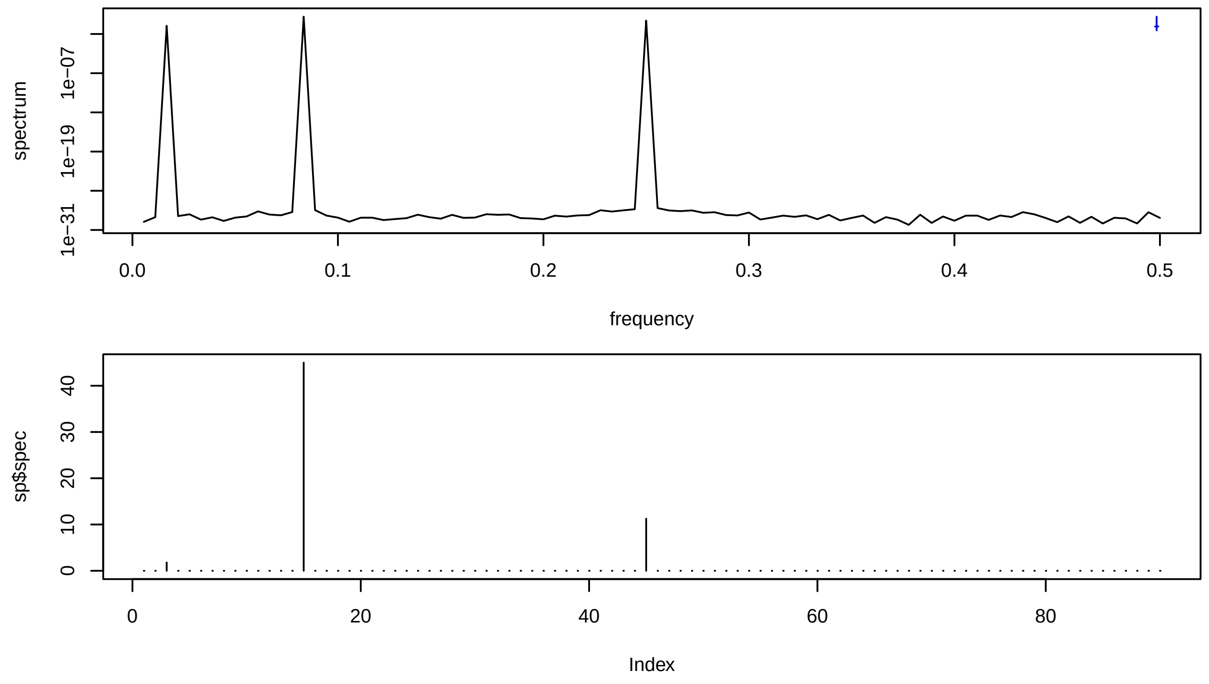


Figure 3.2: Spectral analysis. (See R-Code 3.2.)

R-**Code 3.3:** Spectral analysis for artificial data with noise. (See Figure 3.3.)

```
# we use the same signal but add a noise component
set.seed(14)
sd.noise <- 1/3
z <- y + rnorm(n, sd=sd.noise)
plot(z, type="l")                           # hard to identify the cycles!
spz <- spectrum(z, main="", sub="")   # two large peaks, a third weak one
plot(spz$spec, type="h")
c(estimate=mean(spz$spec[-c(3,15,45)]), exact=sd.noise^2) # bad estimate!

## estimate    exact
##  0.15622  0.11111
```

3.3 Filters

The literature often refers to low-pass and high-pass filters. A low-pass filter is such that low frequencies are passed, and high frequencies are attenuated. A high-pass filter is such that high frequencies are passed, and low frequencies are attenuated. Recall: high frequencies correspond to short wavelengths. In other words, a low-pass signal removes the high-frequency fluctuations and smooths the data.

We now present a few simple filters. Denote x_i be the input signal and y_i the filtered (output) signal.

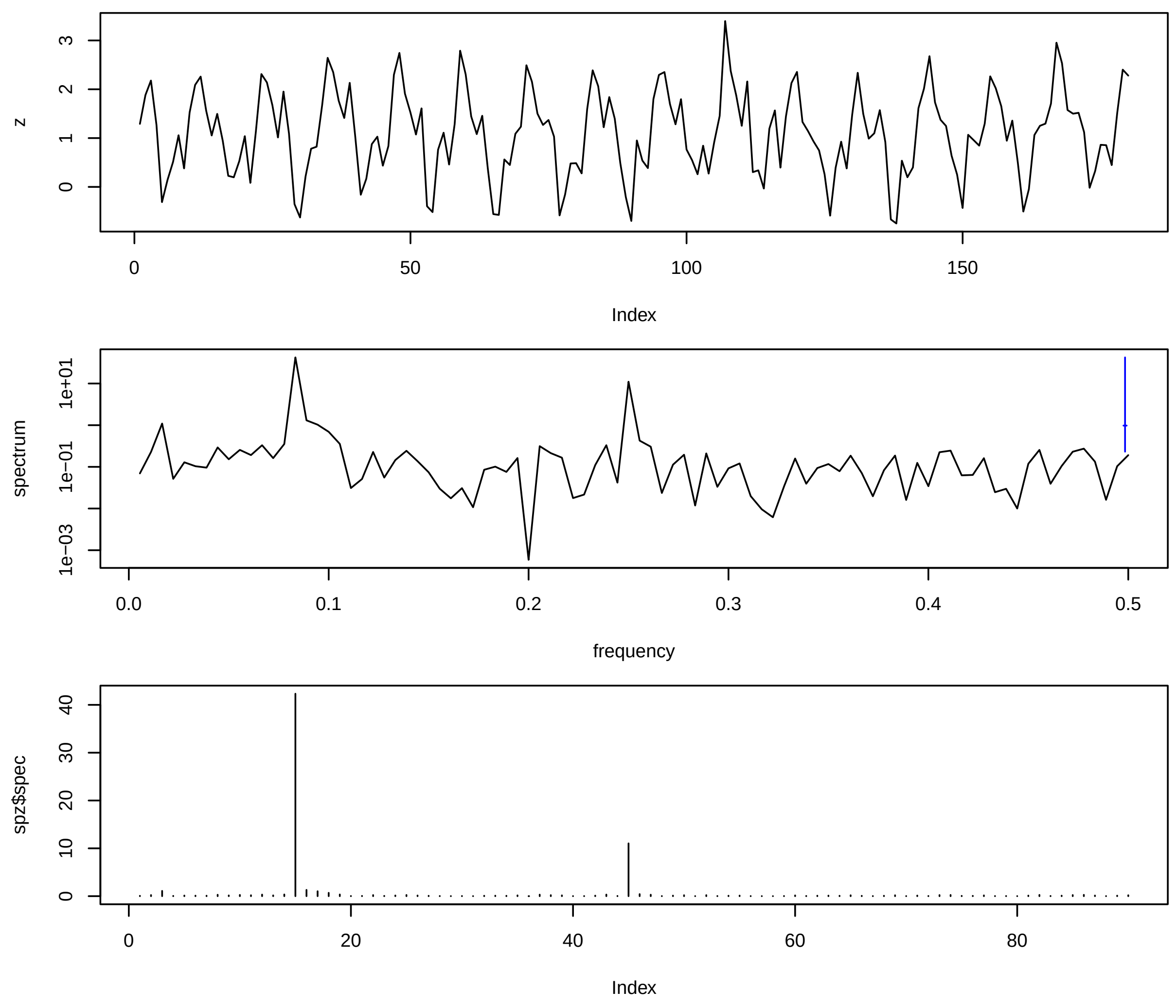


Figure 3.3: Spectral analysis of a signal with noise. (See R-Code 3.3.)

The classical weighted moving average filters are

$$y_i = w_1 x_i + w_2 x_{i-1} + \cdots + w_K x_{i+1-K}, \tag{3.22}$$

for suitably chosen weights $w_1, \dots, w_K$.

A different low-pass filter, the *exponentially weighted moving average* filter that is given by

$$y_i = \alpha x_i + (1-\alpha) y_{i-1}, \qquad 0 \leq \alpha \leq 1 \tag{3.23}$$

if $i > 1$ and $y_1 = x_1$ otherwise. The name stems from recursively expressing the definition to

$$y_i = \alpha x_i + \alpha(1-\alpha)x_{i-1} + \alpha(1-\alpha)^2 x_{i-2} + \dots. \tag{3.24}$$

A high-pass filter is

$$y_i = \alpha(x_i - x_{i-1}) + \alpha y_{i-1}. \tag{3.25}$$

Remark 3.5. In general, a linear filter is

$$y_t = \sum_{j=-\infty}^{\infty} a_j x_{t-j}, \tag{3.26}$$

for absolutely sumable coefficients $\{a_j\}$. In such a setting, it is straightforward to show a one-to-one relationship between the spectral densities of the input and output series. This relationship can also be used to derive the spectral densities of ARMA(p, q) processes. ♡

3.4 Additional Examples

3.4.1 CO$_2$ data

We revisit the annual CO$_2$ seen in Section 2.4. Plotting the spectrum of the raw time series indicates three peaks at zero, one, and two. The peak at zero is due to the (cubic) trend. The remaining two peaks are the annual cycle and climatic differences in both hemispheres.

Removing the annual trend with a regression (cubic polynomial with interaction for Pinatubo) from the series yields a surprisingly similar spectrum for all but the seven first frequencies.

R-**Code 3.4** Spectral analysis of CO$_2$ data. (See Figure 3.4.)

```
sp1 <- spectrum(co2, main="", sub="")
lm3 <- lm(craw ~ indi * (poly(time, 3))) # detrend
co2res <- ts(lm3$resid, start=c(1958, 3), frequency=12)
sp2 <- spectrum(co2res, plot=FALSE)
lines(sp2$freq, sp2$spec, col=3)
1/sp2$freq[ head(order(sp2$spec, decreasing=TRUE))]
```

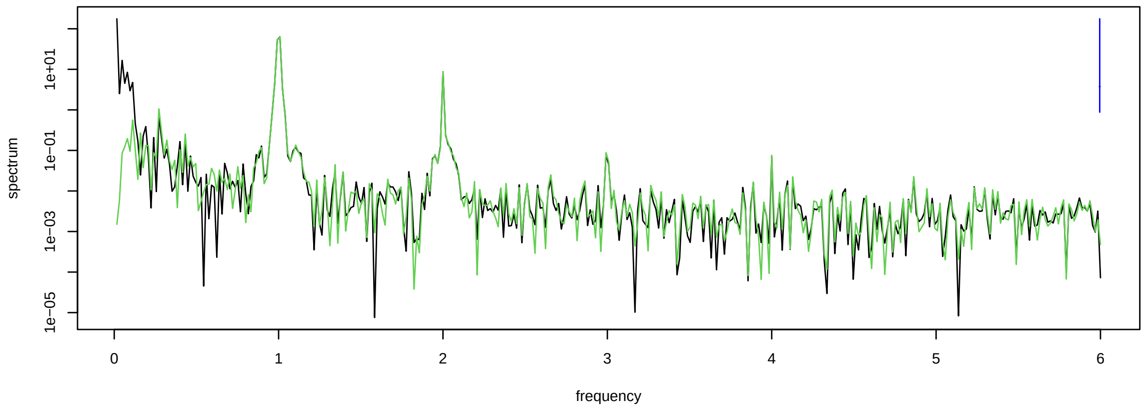


Figure 3.4: Spectrum of raw (black) and detrended (green) CO$_2$ time series. (See R-Code 3.4.)

3.4.2 Seat Belt Data

We now consider the R dataset *UKDriverDeaths*, a time series giving the monthly totals of car drivers in Great Britain killed or seriously injured from January 1969 to December 1984 ($n = 192$). The original data is from Harvey and Durbin (1986).

Compulsory wearing of seat belts was introduced on 31 Jan 1983, and the dataset is often used to query or study the effect of seat belt legislation on casualties.

Here we want to point out that a spectral analysis may not be the ubiquitous tool for all time series.

The time series exhibits a strong seasonal component and a (possibly) smooth trend, as shown in Figure 3.5. Instead of using a parametric approach, we use an autoregressive seasonal component. The fitted component is `sar1`, in models `f1` and `f2` significant. The prediction using a covariate reduces the prediction mean and prediction uncertainty.

R-Code 3.5: Time series of the `UKDriverDeaths` dataset. (See Figure 3.5.)

```
data(UKDriverDeaths)        # Notice that there are additional covariables
                            # available in "Seatbelts" (datasets package)
n <- length(UKDriverDeaths)          # n=192
belt <- c(rep(0,169), rep(1,23))
incr <- c(1:60, rep(0, n-60))
plot(UKDriverDeaths, xlim=c(1969,1986))
### Redefine the variable to play with different versions thereof:
UKDD <- UKDriverDeaths
# UKDD <- log(UKDriverDeaths)
# UKDD <- lm(UKDriverDeaths ~ belt )$resid
# UKDD <- lm(log(UKDD) ~ belt+incr )$resid
# UKDD <- lm(log(UKDD) ~ belt+incr+factor(rep(1:12,length=n)) )$resid
acf(UKDD)
spz <- spectrum(UKDD, main="", sub="") # two large peaks, a third weak one
  # log="no", type="h")

### There is a possibility to play with the order...
f1 <- arima(UKDD, order=c(2,0,0),season=c(1,0,0), method="ML")
f2 <- arima(UKDD, order=c(2,0,0),season=c(1,0,0), method="ML", xreg=belt)
f1
##
## Call:
## arima(x = UKDD, order = c(2, 0, 0), seasonal = c(1, 0, 0), method = "ML")
##
## Coefficients:
##          ar1    ar2   sar1  intercept
##        0.435  0.203  0.667   1643.095
## s.e.   0.075  0.073  0.055     83.893
##
## sigma^2 estimated as 24942:  log likelihood = -1248.1,  aic = 2506.3
```

```
f2

##
## Call:
## arima(x = UKDD, order = c(2, 0, 0), seasonal = c(1, 0, 0), xreg = belt, method = "ML")
##
## Coefficients:
##          ar1     ar2    sar1   intercept       belt
##        0.347   0.197   0.689    1714.813   -365.063
## s.e.   0.075   0.074   0.053      69.382     80.698
##
## sigma^2 estimated as 22720:  log likelihood = -1239.4,  aic = 2490.8
```

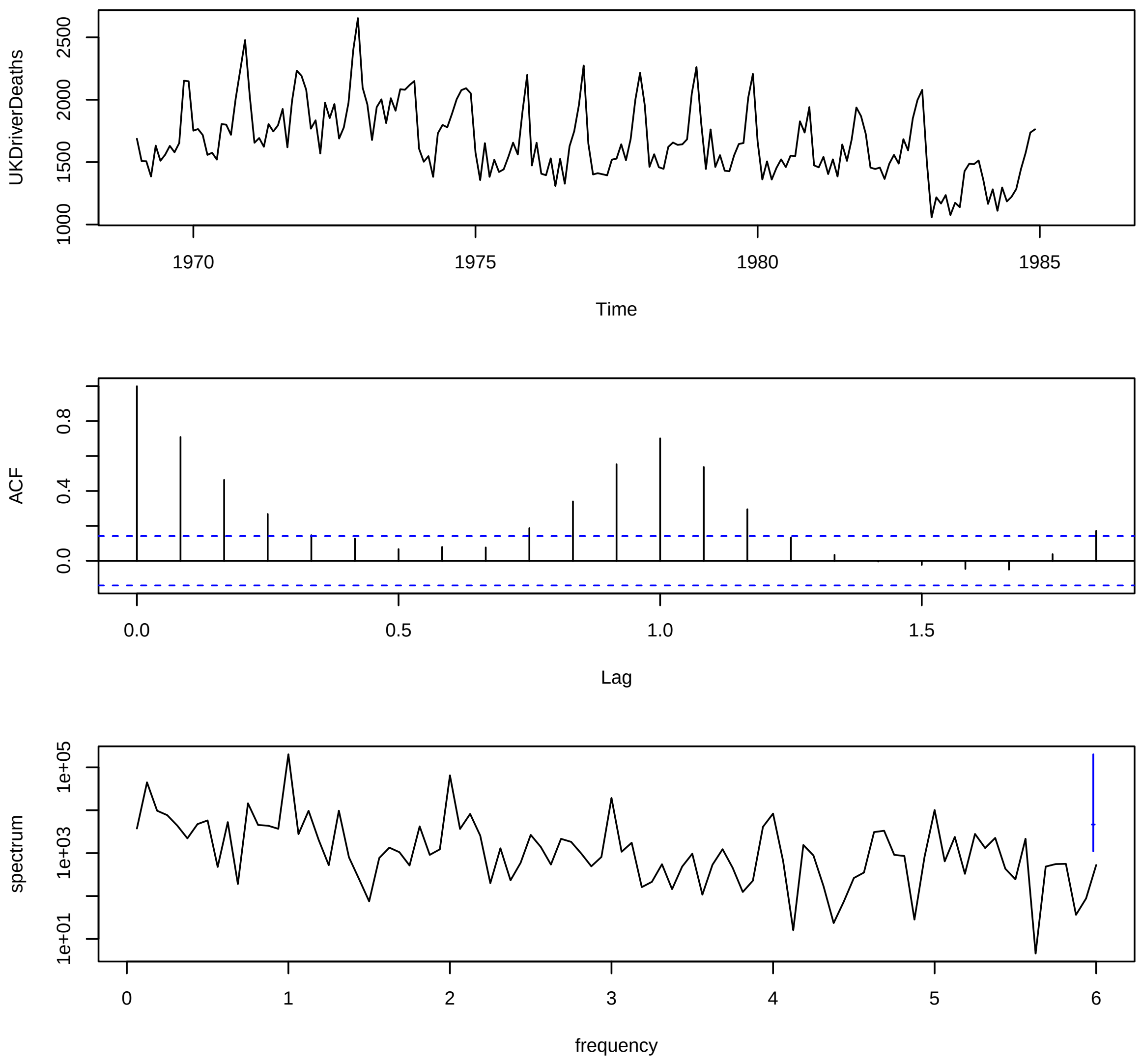


Figure 3.5: *UKDriverDeaths* data. (See R-Code 3.5.)

It is interesting to note that an integrated model does not improve the fit compared to the inclusion of the belt covariate. An AIC comparison would issue a warning as not all models are fitted on to the same number of observations due to the differentiating in an integrated model.

The raw data represents counts, and a priori, a "Gaussian" model may not be adequate. A comparison of the residuals of `f2` and a corresponding model with square root transformed input `sqrt(c(UKDriverDeaths))` is not very decisive.

3.5 Bibliographic Remarks for Time Series

There are plenty of good and valuable time series books, like Box *et al.* (2008); Brockwell and Davis (2010); Shumway and Stoffer (2010). They are all quite formal with notation. For the latter, there is a public pdf version available at www.stat.pitt.edu/stoffer/tsa3/tsa3.pdf, as well as an "easy version" www.stat.pitt.edu/stoffer/tsa3/tsa3EZ.pdf, that covers essentially the topics covered here. Both versions have an accompanying R package `astsa`.

Brockwell and Davis (1991) is another classic, at the upper end, with respect to technicalities.

There exist many lecture notes for time series. A notable one is faculty.smu.edu/tfomby/eco6375/BJ Notes/Forecast Profiles.pdf which specifies explicitly many forecast functions (including for ARI(1,1))

The chapter www.robots.ox.ac.uk/ sjrob/Teaching/SP/l7.pdf gives an excellent and accessible overview of discrete Fourier transform. The algorithmic details of the FFT are nicely illustrated in Chapter 3 'The Divide-and-Conquer Paradigm and Two Basic FFT Algorithms' of Chu and George (1999).

When working with the DFT matrix, the following identities are helpful:

$$\sum_{k=a}^{b-1} \exp(\imath sk) = \frac{\exp(\imath sb) - \exp(\imath sa)}{\exp(\imath s) - 1}, \tag{3.27}$$

$$\prod_{k=0}^{b-1} \exp(\imath sk) = \exp(\imath sb(b-1)/2), \tag{3.28}$$

for $s \in \mathbb{R}$ and integers $a < b$.

Chapter 4

Lattice Data: Concepts

We introduce spatial models for data on a specific grid. These models can also be used for modeling data over a set of areas. R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter04.R.

4.1 Introduction

In the last chapter, we looked at temporal data. That means we have introduced models for dependent data that were observed at regular (one-dimensional) locations. Extending this concept to two dimensions would lead to locations on a regular grid.

There is, however, another significant difference between time series data and lattice data. In time series, we have a temporal "direction" and forecasting (prediction) is a very natural concept. In the case of lattice data, forecasting in the sense of interpolation or extrapolation is only used in rare cases. Tasks are much more centered around the concept of "smoothing" (i.e., filtering in the time series context). Smoothing can be used to separate the signal from the noise or to fill in missing values.

Lattice data is also called areal data in many books, based on the idea that the area under investigation is inherently discretized into a finite number of blocks or sub-areas. The areas may be very regular, matrix, or image-like (Example 1.5 discusses such a case) or irregular, as shown in Figure 1.3 of Example 1.2. The set of a representative location of the block defines a lattice.

The lattice may have a varying size of areas, having a possible effect on the spatial variability ("smoothness") of the data. Further, the lattice may have varying shapes of the individual areas and thus a different structure of neighbor areas. An example illustrating both aspects is the partition of the lower 48 US states into counties. Figure 4.1 shows how the sizes of the counties vary as well as the number of counties with a sharing boundary (i.e., number of neighbors. A pure geographic distance between the counties does not necessarily make sense.

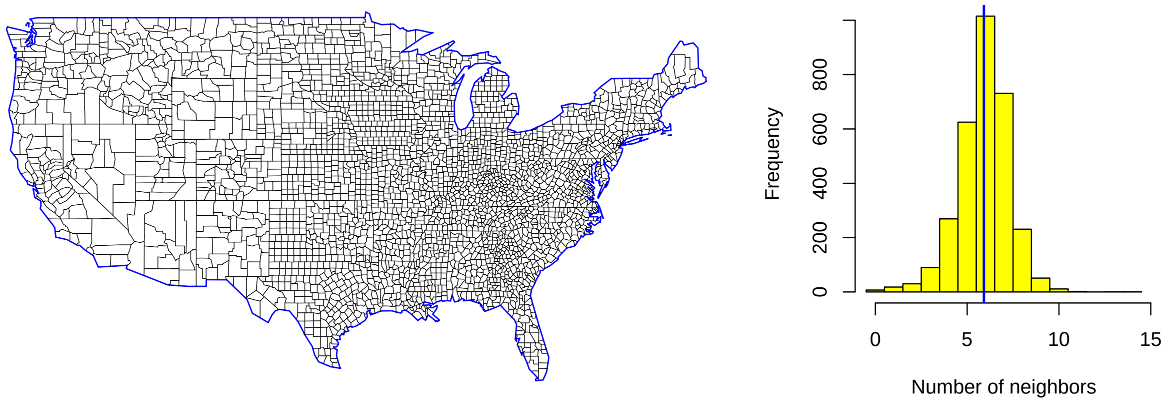


Figure 4.1: The 3082 counties of the lower 48 US states (left panel) and the histograms of the number of neighbors (right). The vertical line represents the average of 5.9. Seven counties do not share any boundaries with others. (Data from packages *maps* and *spam*.)

Example 4.1. The prime example of a lattice dataset is the sudden infant death syndrome (SIDS) cases in North Carolina, e.g., Cressie (1993), further details to references are given in the vignette of the package *spdep*, i.e., `vignette("sids", package="spdep")`. R-Code 4.1 loads the data and represents the counties.

To represent the data, formal 'shape'-files can be processed. There exist many ways to read in a shape-file. We recommend using tools provided by the packages *sf* and *terra*. Here, the *maps* package can be used as well: `map("county", region="North Carolina", ...)`. Care is needed when assigning colors to individual counties. ♣

R-**Code 4.1** Crude SIDS rates for the period 1974 to 1979. (See Figure 4.2.)

```
library(sf)      # General handling of spatial data objects
library(spdep)   # General description of the SIDS data
library(spData) # shape-file for North Carolina counties
### the raw data itself is available through
# data(nc.sids, package="spdep")
nc <- st_read(system.file("shapes/sids.shp", package="spData")[1], quiet=TRUE)
# st_crs(nc) <- "+proj=longlat +datum=NAD27"
# row.names(nc) <- as.character(nc$FIPSNO)

nc$rates <- nc$SID74 / nc$BIR74
plot(nc["rates"], pal=tim.colors(10))
```

In this chapter, we will use the generic $\pi(\cdot)$ expression to denote densities. The vast Most textbooks, articles, and suchlike about Bayesian statistics also use this notation. We also use the notation of "negative" indices. Let $\boldsymbol{x}$ be an arbitrary n-vector. Then $\boldsymbol{x}_{-i}$ is the vector

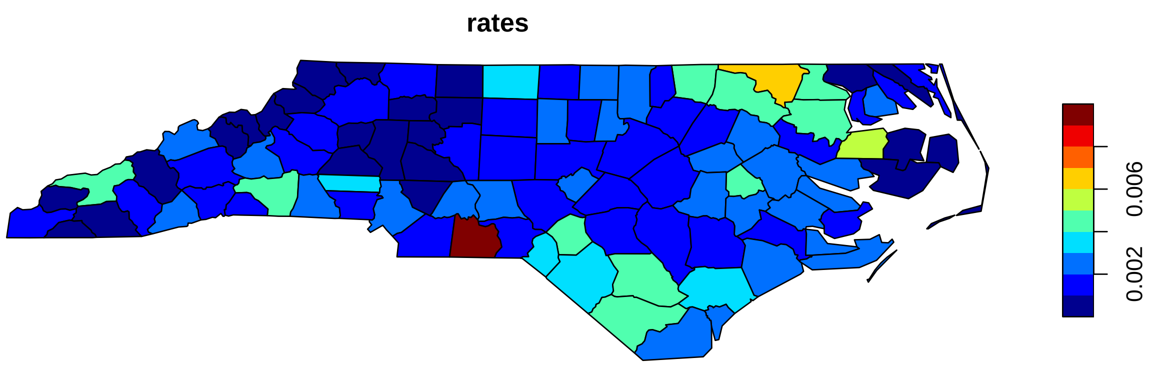


Figure 4.2: Visualization of crude SIDS rates for the period 1974 to 1979. (See R-Code 4.1.)

$(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n)^\top$, that means the vector $(n-1)$-vector obtained by deleting the ith component of $\boldsymbol{x}$.

4.2 Conditionally Autoregressive Models

For simplicity, take a regular grid as shown in the panels of Figure 4.3. The basic idea behind conditionally autoregressive (CAR) models is that the mean of the distribution of a location i depends on the other grid locations. Conditionally, the "red" location depends (conditionally) on the "blue" ones, possibly on the "blue" and "green" ones etc. In general terms, a simple model is given by

$$Y_i \mid \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\sum_{j, j\neq i} b_{ij} y_j, \tau_i^2\Big), \qquad i = 1, \dots, n. \tag{4.1}$$

There are some additional conditions (mainly on $\{b_{ij}\}$ and $\{\tau_i^2\}$) but *Brook's lemma* allows us to conclude that the joint distribution of $\mathbf{Y}$ is also Gaussian

$$\mathbf{Y} \sim \mathcal{N}_n(\mathbf{0}, (\mathbf{I} - \mathbf{B})^{-1}\mathbf{T}), \tag{4.2}$$

where $\mathbf{B} = (b_{ij})$ and $b_{ii} = 0$, $\mathbf{T} = \text{diag}(\tau_i^2)$.

Through the joint distribution, we can directly derive a few conditions on the model. Because a covariance matrix is symmetric, it is required that

$$\frac{b_{ij}}{\tau_i^2} = \frac{b_{ji}}{\tau_j^2} \tag{4.3}$$

Further, the matrix $\mathbf{I} - \mathbf{B}$ has to be positive definite. As typically done, a low-dimensional parameterization will be used (see Section 5.3).

The CAR model approach specifies the joint density through n full conditional densities $\{\pi(y_i \mid \boldsymbol{y}_{-i})\}$ has been pioneered by Besag (1974). For non-Gaussian distributions, a few more theoretical results are necessary but are receiving more and more attention.

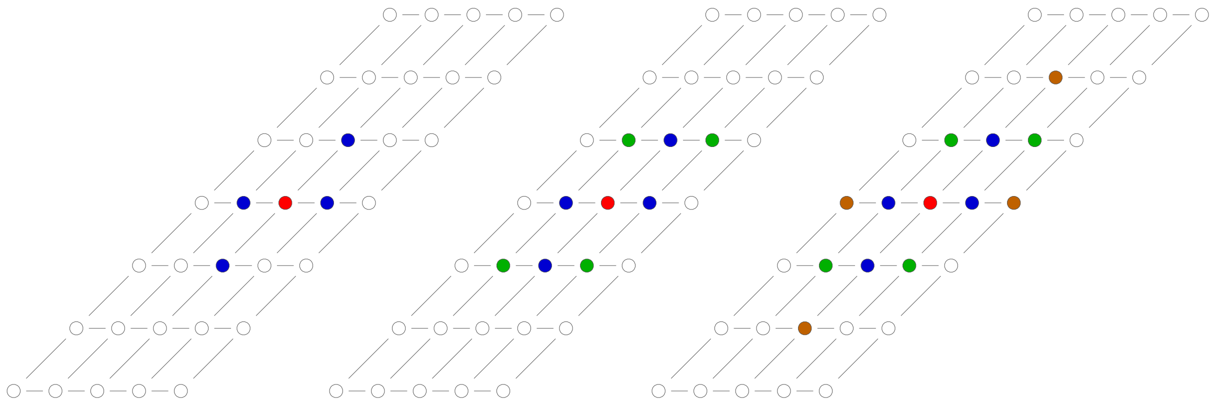


Figure 4.3: Grid locations. For a specific location, indicated in red, first-order neighbors (left), eight neighbors (middle), first- and second-order neighbors (right).

4.3 Gaussian Markov Random Fields

We now impose a Markov property on the conditional dependence structure and assume that we only have conditional dependence on a few neighbor sites. In a temporal setting, *neighbor* is interpreted as the previous and next value; in lattice models, *neighbor* typically signifies sharing a common edge or boundary. We denote the neighbor relation with $i \sim j$ for sites $i \neq j$. The relation is symmetric, i.e., if $i \sim j$, then $j \sim i$. In the middle panel of Figure 4.3 the blue locations are *first order* neighbors of the red location. In terms of (4.1), only four of the $\{b_{ij}\}$ are non-zero. In the center panel of Figure 4.3, the four green neighbors of the red location have been added, implying four more non-zero $\{b_{ij}\}$s.

Neighbors of first-order neighbors are often called *second-order* neighbors. Additional neighborhoods or contiguity structures exist for regular lattices, which are referred to differently in the literature. The first-order neighborhood is also called 4-neighborhood, von Neumann, or rook neighborhood (Figure 4.3, left panel). The center panel of Figure 4.3 shows a Moore or Queen neighborhood. The green neighbors represent a Bishop's structure. The right panel of Figure 4.3 illustrates a von Neumann neighborhood of range 2 (all locations within Manhattan distance 2).

It is convenient to represent the dependence structure with an undirected, labeled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes in a graph and $\mathcal{E}$ the set of edges $\{i, j\}$, $i \neq j \in \mathcal{V}$. As example, Figure 4.4 shows the 5 counties of Rhodes Island ($\mathcal{V} = \{1, 2, 3, 4, 5\}$) and the first-order neighbor structure ($\mathcal{E} = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}\}$) and the associated *adjacency matrix* $\mathbf{A}$, with $a_{ij} = a_{ji} = 1$ if $\{i, j\} \in \mathcal{E}$, $\forall i \neq j \in \mathcal{V}$ and $a_{ij} = a_{ji} = 0$ otherwise.

Definition 4.1. The precision of a random variable is the inverse of the variance.

In the case of random vectors, the inverse of a covariance matrix is called the precision matrix. $\diamondsuit$

The precision and the diagonal precision matrix is an intuitive concept: a high precision (low variance) implies a lot of knowledge about the variable.

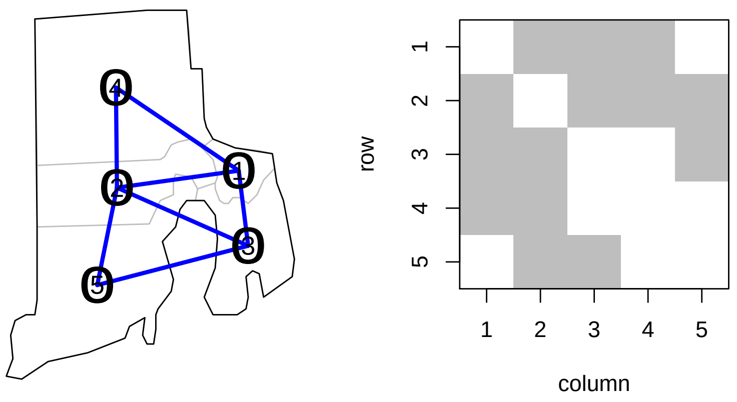


Figure 4.4: The 5 counties of Rhodes Island (left panel) and the associated adjacency matrix with non-zero values represented in gray (right panel).

Definition 4.2. A random vector $\mathbf{Y} = (Y_1, \dots, Y_n)^\top$ is a GMRF with respect to a labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with mean $\boldsymbol{\mu}$ and (spd) precision matrix $\mathbf{Q}$ if its density is given by

$$\pi(\boldsymbol{y}) = (2\pi)^{-n/2} \det(\mathbf{Q})^{1/2} \exp\Big(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})^\top \mathbf{Q} (\boldsymbol{y} - \boldsymbol{\mu}) \Big) \tag{4.4}$$

and $Q_{ij} \neq 0 \iff \{i, j\} \in \mathcal{E}$, $\forall i \neq j$. $\diamondsuit$

We will denote independence between two random variables X and Y by $X \perp Y$ and conditional independence between X and Y given $Z = z$ by $X \perp Y \mid Z = z$.

Property 4.1. *Let* $\mathbf{Y}$ *be a GMRF with respect to* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *with mean* $\boldsymbol{\mu}$ *and (spd) precision matrix* $\mathbf{Q}$*. Then the following hold.*

1. $Y_i \perp Y_j \mid \mathbf{Y}_{-ij} = \boldsymbol{y}_{-ij} \iff Q_{ij} = 0.$
2. $\mathrm{E}[Y_i \mid \mathbf{Y}_{-i} = \boldsymbol{y}_{-i}] = \mu_i - \dfrac{1}{Q_{ii}} \sum_{j:j\sim i} Q_{ij}(y_j - \mu_j),$
3. $\mathrm{Prec}(Y_i \mid \mathbf{Y}_{-i} = \boldsymbol{y}_{-i}) = Q_{ii},$
4. $\mathrm{Corr}(Y_i, Y_j \mid \mathbf{Y}_{-ij} = \boldsymbol{y}_{-ij}) = -\dfrac{Q_{ij}}{\sqrt{Q_{ii}Q_{jj}}},\ i \neq j.$

4.4 Simultaneous Autoregressive Models and Other Extensions

As an alternative to a CAR specification, we now look at an autoregressive approach that is closer to the time series one:

$$Y_i = \sum_{j, j\neq i} b_{ij} Y_j + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma_i^2), \qquad i = 1, \dots, n. \tag{4.5}$$

To see the link to the time series, suppose a univariate grid and set $b_{i,i-1} = \phi$ and zero for all others.

Model (4.5) is called a simultaneous autoregressive (SAR) model. The condition that the sum is over $i \neq j$ implies that $b_{ii} \equiv 0$ for all i. Further conditions on $\{b_{ij}\}$ exist, and for the moment, we assume that $\mathbf{I} - \mathbf{B}$ is invertible. Then

$$\mathbf{Y} \sim \mathcal{N}_n\big(\mathbf{0}, (\mathbf{I}-\mathbf{B})^{-1}\mathbf{V}((\mathbf{I}-\mathbf{B})^{-1})^\top\big), \tag{4.6}$$

where $\mathbf{V} = \operatorname{diag}(\sigma_i^2)$. We refer to the next chapter for a parameterized version of this model.

There is a link between CAR and SAR models. To see the link, we write both models as

$$\mathbf{Y} = \mathbf{BY} + \boldsymbol{\varepsilon}. \tag{4.7}$$

The models and the (dis)similarities are summarized in Table 4.1. However, every SAR model can be uniquely written as a CAR, not vice-versa. See also Ver Hoef *et al.* (2018).

Table 4.1: (Dis)similarities between a CAR and a SAR model.

CAR	SAR
$\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \mathbf{T}(\mathbf{I}-\mathbf{B})^\top)$	$\boldsymbol{\varepsilon} \sim \mathcal{N}_n(\mathbf{0}, \mathbf{V})$
$\mathbf{Y} \sim \mathcal{N}_n(\mathbf{0}, (\mathbf{I}-\mathbf{B})^{-1}\mathbf{T})$	$\mathbf{Y} \sim \mathcal{N}_n\big(\mathbf{0}, (\mathbf{I}-\mathbf{B})^{-1}\mathbf{V}((\mathbf{I}-\mathbf{B})^{-1})^\top\big)$
$\operatorname{Cov}(\mathbf{Y}, \boldsymbol{\varepsilon}) = \mathbf{T}$	$\operatorname{Cov}(\mathbf{Y}, \boldsymbol{\varepsilon}) = (\mathbf{I}-\mathbf{B})^{-1}\mathbf{V}$

SAR models are typically introduced on the residuals of $\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}$, i.e., a regression mean is added to the model.

SAR models have been extended to a spatio-temporal setting, leading to the so-called STAR models. As a side note, a multivariate extension is typically done on the CAR setting, leading to the so-called MCAR models.

4.5 Intrinsic Gaussian Markov Random Fields

A density that does not integrate to one is called an improper density. Such densities are typically used as priors in a Bayesian framework because the resulting posterior may be proper.

Here, we will look at two different types of "improper" GMRF resulting from a natural constructive approach. These GMRFs will be later used as building blocks in our examples.

4.5.1 Random Walk Models

In many modeling approaches, we assume that a series consisting of $Y_1, \dots, Y_n$ has a constant mean μ or a mean that is parameterized (in a regression setting). This is often very restrictive, and it is natural to relax this condition by considering that $Y_{i+1} - Y_i \sim \mathcal{N}(0, \kappa^{-1})$.

More specifically, assume that the locations of n random variables are $i = 1, \dots, n$, e.g., referred to as equispaced observations on the transect. In such a case, we also often consider i as the time. We define $\Delta Y_i = Y_{i+1} - Y_i$, $i = 1, \dots, n-1$. Assuming that the ΔY_i are iid $\mathcal{N}(0, \kappa^{-1})$, we have

$$\pi(\Delta \boldsymbol{y}) \propto \kappa^{(n-1)/2} \exp\Big(-\frac{\kappa}{2}\sum_{i=1}^{n-1}(\Delta y_i)^2\Big) = \kappa^{(n-1)/2} \exp\Big(-\frac{1}{2}\boldsymbol{y}^\top \mathbf{Q}\boldsymbol{y}\Big), \tag{4.8}$$

where the $n \times n$ precision matrix is given by

$$\mathbf{Q} = \kappa \begin{pmatrix} 1 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 1 \end{pmatrix} = \kappa \mathbf{D}^\top \mathbf{D}, \quad \mathbf{D} = \begin{pmatrix} -1 & 1 & 0 & \dots \\ 0 & \ddots & \ddots & \ddots \\ \vdots & \ddots & -1 & 1 \end{pmatrix} \in \mathbb{R}^{n-1\times n}. \tag{4.9}$$

Note that (4.8) is not a proper density because $\mathbf{Q1} = \mathbf{0}$. In other words, the precision $\mathbf{Q}$ has rank $n-1$, thus, is rank deficient (here symmetric positive-semidefinite) and does not fit in our "classical" framework for multivariate normal variables.

Definition 4.3. Let $\mathbf{Q}$ be an $n \times n$ symmetric positive-semidefinite matrix of rank $n-k > 0$. $\mathbf{Y}$ is an improper GMRF of rank $n-k$ and parameters $(\boldsymbol{\mu}, \mathbf{Q})$ with respect to the labelled graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if its density is

$$\pi^*(\boldsymbol{y}) = (2\pi)^{-(n-k)/2} \det{}^*(\mathbf{Q})^{1/2} \exp\Big(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{y}-\boldsymbol{\mu})\Big) \tag{4.10}$$

and $Q_{ij} \neq 0 \iff \{i,j\} \in \mathcal{E}$, $\forall i \neq j$. ♢

The term $\det^*(\mathbf{Q})$ is the generalized determinant, i.e., the product of all the non-zero eigenvalues of $\mathbf{Q}$. The parameters $(\boldsymbol{\mu}, \mathbf{Q})$ no longer represent the mean and the precision since they no longer exist formally.

Definition 4.4. An intrinsic GMRF (IGMRF) of first-order (or order $n-1$) is an improper GMRF of rank $n-1$ where $\mathbf{Q1} = \mathbf{0}$. ♢

The model given by (4.8) is thus an IGMRF of first-order, also referred to as *Random Walk* of first-order, RW1.

Prediction in an RW1 model is a meaningful concept considering space as time. We have, for example, the following results:

$$Y_i \mid \mathbf{Y}_{-i} = \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\frac{1}{2}(y_{i+1} + y_{i-1}), \frac{1}{2\kappa}\Big), \tag{4.11}$$

$$Y_{i+p} \mid Y_i = y_i, Y_{i-1} = y_{i-1}, \dots \sim \mathcal{N}\Big(y_i, \frac{p}{\kappa}\Big), \qquad 0 < i < i+p \leq n. \tag{4.12}$$

Notice that there is no shrinkage towards the mean, as we have seen in the case of an AR(p). Here, we are in a similar framework as an ARIMA($p,1,q$) setting.

Similar to a RW1 model, we can define a RW2 by defining $\Delta^2 Y_i = \Delta Y_{i+1} - \Delta Y_i = Y_{i+2} - 2Y_{i+1} + Y_i$, $i = 1, \dots, n-2$. We assume again that the $\Delta^2 Y_i$ are iid $\mathcal{N}(0, \kappa^{-1})$.

In a similar spirit, we can construct models for seasonal variation. For example, with period $m = 12$, we assume that

$$Y_{i+0} + Y_{i+1} + \dots + Y_{i+m-1} \overset{\text{iid}}{\sim} \mathcal{N}(0, \kappa^{-1}), \qquad i = 1, n-m+1. \tag{4.13}$$

4.5.2 GMRF Under a Linear Constraint

Assume that $\mathbf{X}$ is a GMRF with mean $\boldsymbol{\mu} = \mathbf{0}$ and precision matrix $\mathbf{Q}$. We decompose $\mathbf{Q}$ into its eigenvalues and eigenvectors

$$\mathbf{Q} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^\top = \sum_{i=1}^{n} \lambda_i \boldsymbol{e}_i \boldsymbol{e}_i{}^\top \tag{4.14}$$

with $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$, $\mathbf{V} = (\boldsymbol{e}_1, \dots, \boldsymbol{e}_n)$ with $\mathbf{V}^\top\mathbf{V} = \mathbf{I}_n$. Write for $0 < p < n$, $\mathbf{A}^\top = (\boldsymbol{e}_1, \dots, \boldsymbol{e}_p)$ and set $\mathbf{Y} = \mathbf{V}^\top\mathbf{X}$. We are interested in the conditional density $\pi(\boldsymbol{y} \mid \mathbf{A}\boldsymbol{x} = \boldsymbol{a})$ for some p-vector $\boldsymbol{a} = (a_1, \dots, a_p)^\top$. It is straightforward to show that

$$\pi(\boldsymbol{y} \mid \mathbf{A}\boldsymbol{x} = \boldsymbol{a}) = 1_{[\boldsymbol{y}_{1:p} = \boldsymbol{a}]} \prod_{i=p+1}^{n} \pi(y_i). \tag{4.15}$$

The following statements hold

$$\mathrm{E}[\mathbf{Y} \mid \mathbf{A}\mathbf{X} = \boldsymbol{a}] = \begin{pmatrix} \boldsymbol{a} \\ \mathbf{0} \end{pmatrix}, \tag{4.16}$$

$$\operatorname{Prec}(\mathbf{Y} \mid \mathbf{A}\mathbf{X} = \boldsymbol{a}) = \begin{pmatrix} 0 & \dots & & & & 0 \\ \vdots & \ddots & & & & \\ & & 0 & & & \\ & & & \lambda_{p+1} & & \vdots \\ & & & & \ddots & 0 \\ 0 & & & \dots & 0 & \lambda_n \end{pmatrix}, \tag{4.17}$$

$$\mathrm{E}[\mathbf{X} \mid \mathbf{A}\mathbf{X} = \boldsymbol{a}] = \mathbf{V} \begin{pmatrix} \boldsymbol{a} \\ \mathbf{0} \end{pmatrix} = a_1\boldsymbol{e}_1 + \dots + a_p\boldsymbol{e}_p, \tag{4.18}$$

$$\operatorname{Prec}(\mathbf{X} \mid \mathbf{A}\mathbf{X} = \boldsymbol{a}) = \mathbf{V} \begin{pmatrix} 0 & & & 0 \\ & \lambda_{p+1} & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{pmatrix} \mathbf{V}^\top := \widetilde{\mathbf{Q}}. \tag{4.19}$$

$$\tag{4.20}$$

Hence, we also have

$$\log \pi(\boldsymbol{x} \mid \mathbf{A}\boldsymbol{x} = \boldsymbol{a}) = -\frac{n-p}{2}\log(2\pi) + \frac{1}{2}\sum_{i=p+1}^{n} \lambda_i - \frac{1}{2}\boldsymbol{x}\widetilde{\mathbf{Q}}\boldsymbol{x}. \tag{4.21}$$

In other words, we can separate the vector $\mathbf{X}$ into two components, $\mathbf{X} = \mathbf{X}^{\parallel} + \mathbf{X}^{\perp}$ and in view of (4.10), $\pi^*(\boldsymbol{x}) = \pi(\boldsymbol{x}^{\perp})$. Hence, to simulate realizations from (4.10), we draw Y_i, $i = p+1, \dots, n$, iid from $\mathcal{N}(0, \kappa^{-1})$ and put $\mathbf{X} = Y_{p+1}\boldsymbol{e}_{p+1} + \dots + Y_n\boldsymbol{e}_n$.

Remark 4.1. In case of an arbitrary condition $\mathbf{B}\mathbf{X} = \boldsymbol{b}$, we express the constraints in terms of $\mathbf{A}\mathbf{X} = \boldsymbol{a}$, where $\mathbf{A}^\top$ span the nullspace of $\operatorname{Cov}(\mathbf{X} \mid \mathbf{B}\mathbf{X})$. ♡

4.5.3 Specific Example

Notice that a constrained GMRF and an intrinsic GMRF induce two different types of singularities in the covariance matrix. There is a "variance collapse" for the first and an "variance inflation" for the second.

We illustrate the concept of these improper GMRFs with a simple 2×2 example in R-Code 4.2. We use the same seed for all simulations, so the "projections" will become clear.

In terms of the covariance matrix, the constrained simulation results in a rank-deficient covariance matrix, i.e., one eigenvalue is zero. The same holds for the precision matrix.

In the case of a RW1, we simulate from $\mathbf{Q}(\gamma) = \mathbf{Q} + \gamma\mathbf{1}\mathbf{1}^\top$ for smaller and smaller γ. In an RW1 model, we do not specify any "marginal" distributions; thus, no information about the individual means is known, which can be spread over the entire real line. The eigenvalues of $\mathbf{Q}(\gamma)$ are 2 and $2(0.1+\gamma)$. Thus, the eigenvalues of Sigma are $0.5/\gamma$ and 0.5 and, hence, variance inflation as soon as $\gamma \to 0$.

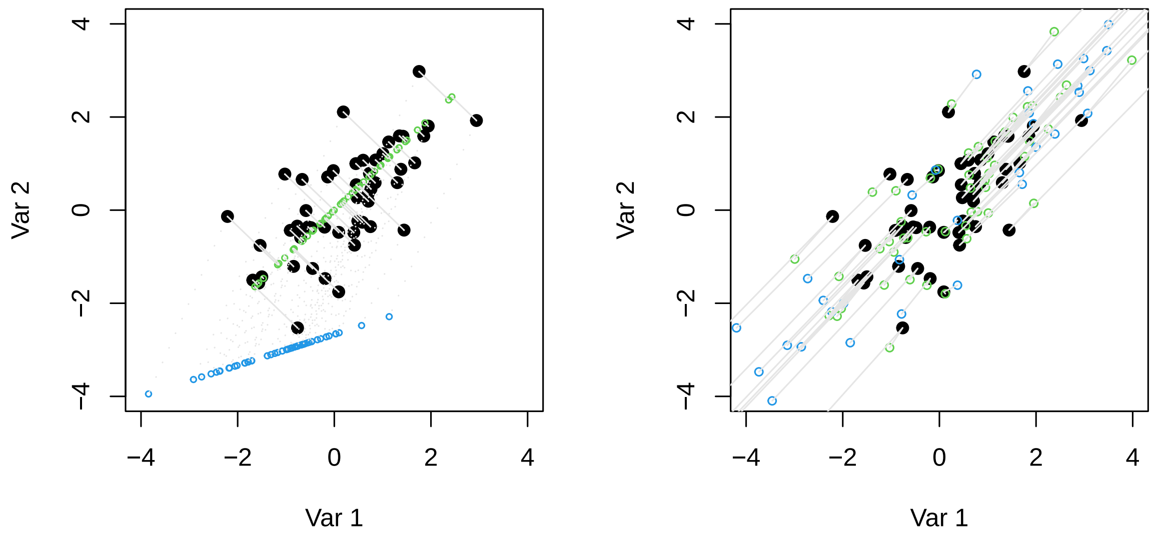


Figure 4.5: Sampling from "improper" GMRFs. Left for constrained sampling with $(1,-1)\mathbf{X} = 0$ (green) and $(1,-3)\mathbf{X} = 8$ (blue); right for RW1, mimicked with $\mathbf{Q}(\gamma)$ for $\gamma = 0.1$ (green) and $\gamma = 0.01$ (blue). (See R-Code 4.2.)

R-**Code 4.2** Sampling from "improper" GMRFs. (See Figure 4.5.)

```
n <- 50                                             # sample size
### Left panel:
Q <- as.spam(matrix(c(1.2, -.8, -.8, 1.2), 2, 2)) # gamma = 0.2
solve(Q)                              # this is the covariance matrix
##      [,1] [,2]
## [1,]  1.5  1.0
## [2,]  1.0  1.5
set.seed(1); sa0 <- rmvnorm.prec(n, Q=Q)
set.seed(1)
sa1 <- as.matrix(rmvnorm.prec.const(n, Q=Q, A=cbind(1,-1)))
set.seed(1)
sa2 <- as.matrix(rmvnorm.prec.const(n, Q=Q, A=cbind(1,-3), a=8))

plot(sa0, ylim=c(-4,4), xlim=c(-4,4), xlab="Var 1", ylab="Var 2", pch=19)
points(sa1, col=3, cex=.5)
points(sa2, col=4, cex=.5)
segments(sa0[,1], sa0[,2], sa1[,1], sa1[,2], col="gray90")
segments(sa0[,1], sa0[,2], sa2[,1], sa2[,2], col="gray90", lty="19")
var(sa2)
##         [,1]    [,2]
## [1,] 0.93187 0.31062
## [2,] 0.31062 0.10354
### Right panel:
Q <- as.spam(matrix(c(1.1, -.9, -.9, 1.1), 2, 2))     # gamma = 0.1
set.seed(1); sa3 <- rmvnorm.prec(n, Q=Q)

Q <- as.spam(matrix(c(1.01, -.99, -.99, 1.01), 2, 2)) # gamma = 0.01
set.seed(1); sa4 <- rmvnorm.prec(n, Q=Q)

plot(sa0, ylim=c(-4, 4), xlim=c(-4, 4), xlab="Var 1", ylab="Var 2", pch=19)
points(sa3, col=3, cex=.7)
points(sa4, col=4, cex=.7)
segments(sa0[,1], sa0[,2], sa3[,1], sa3[,2], col="gray90")
segments(sa0[,1], sa0[,2], sa4[,1], sa4[,2], col="gray90")
var(sa4)
##        [,1]   [,2]
## [1,] 19.687 19.588
## [2,] 19.588 20.312
```

4.6 Bibliographic Remarks

Virtually all books about spatial data have at least one chapter dedicated to lattice data or areal data. We typically recommend Cressie (1993); Schabenberger and Gotway (2005); Banerjee *et al.* (2003). Handling the data with appropriate packages is extensively discussed in Bivand *et al.* (2008).

The SIDS dataset is extensively discussed in CRAN.R-project.org/web/packages/spdep/vignettes/sids.html. See r-spatial.github.io/sf/articles/sf1.html for a gentle introduction to the package ***sf***, see also `https://rspatial.org/` for details about the ***terra*** package.

The CAR models approach was pioneered by Besag (1974). However, it took a couple of decades until its full power was recognized and through the nowadays computational capabilities exploited.

Rue and Held (2005) extend the RW1 to random walk models for irregular locations, on lattices, on irregular lattices (Section 3.3) and to random walk models of higher order (Section 3.4), and more. See also www.stat.berkeley.edu/∼paciorek/research/techVignettes/tech Vignette5.pdf for more insights of intrinsic fields.

Chapter 5

Lattice Data: Simulation and Estimation

Based on existing spatial R packages, we assess spatial dependency and fit simple GMRF models to data. Emphasis on computational aspects is given as well.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter05.R.

5.1 Spatial Objects in R

In the last chapter, we have (statistically) introduced GMRF for regular and irregular lattices. The latter are more present in real-world applications. Unfortunately, much of the time in any data analysis is spent visualizing the data and gathering and assembling each area's boundary information. Before the actual modeling, we look at some indispensable software components when visualizing areal data.

For spatial data analysis, we need at least a list of (named) polygons from a database and a list of neighbors for each of these. Ideally, we have direct access to plotting methods for the polygons. The neighborhood structure is often given as a list or an adjacency matrix and may often be constructed based the polygons themselves.

The packages ***sf***, ***spdep***, and ***spatialreg*** provide a framework for handling, analyzing, and plotting spatial data. Often it is intimidating to get acquainted with the various R functions and classes. The maintainers of the above packages tightly collaborate and are aware of the technical overhead that might distract from a statistical analysis. For more details, a good start is Chapter 2 of Bivand *et al.* (2013). The basic idea of the packages is to store all objects within a family of classes, define many methods for these classes, and provide a useful number of helping functions. Thus many situations, we do not need to worry about the underlying technical details. For didactic purposes, we often illustrate the formal and the "manual" handling of the data.

In the R-Code below, we illustrate two approaches, a manual and a "formal" one. The former one is based on simple lists containing the x and y coordinates of the polygons, bounding box, and polygon names. Thus the plotting can be done manually. The latter one relies on the formal *sf* class. Thus the objects are more complicated but plotting is easier.

R-Code 5.1: Handling spatial objects in R.

```
nc <- st_read(system.file("shapes/sids.shp", package="spData")[1], quiet=TRUE)
nc$rates <- nc$SID74 / nc$BIR74
class(nc)          # classes "sf" and "data.frame"
## [1] "sf"          "data.frame"
dim(nc)            # for each county a lot of information
## [1] 100  24
str(nc$geometry)   # spatial info in one list element
## sfc_MULTIPOLYGON of length 100; first list element: List of 1
##  $ :List of 1
##   ..$ : num [1:27, 1:2] -81.5 -81.5 -81.6 -81.6 -81.7 ...
##  - attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
### see methods(class="sf")
### The following for an even more formal analysis
# st_crs(nc) <- "+proj=longlat +datum=NAD27"
# row.names(nc) <- as.character(nc$FIPSNO)
### eliminate the outlier:
index <- which.max(nc$rates)   # "remove" outlier:
as.character(nc$NAME[index])
## [1] "Anson"
ncNo <- nc[-index,]
fitbreaks <- seq(min(nc$rates), max(nc$rates), by=diff(range(nc$rates))/10)
plot(nc["rates"], reset=FALSE, breaks=fitbreaks, key.pos=NULL)
plot(ncNo["rates"], breaks=fitbreaks, key.length=1)
```

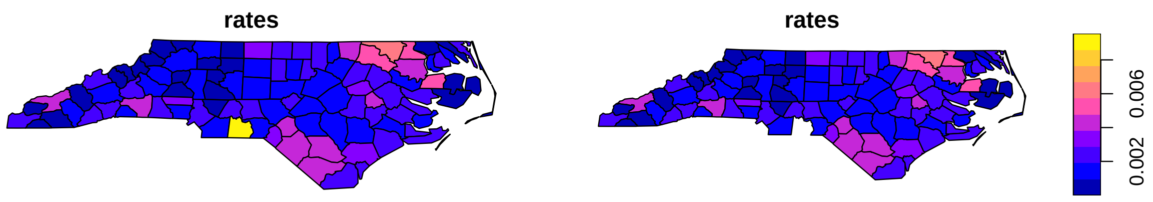


Figure 5.1: SIDS rates for all 100 counties and without Anison county. (See R-Code 5.1.)

5.2 Assessing Spatial Dependency

Correlation in time series is based on the evaluation of lagged values, e.g., analyzing $Y_t - Y_{t-1}$, where Y_{t-1} is the lagged value of Y_t. With lattice data, we can construct a spatial lag by considering all first-order neighbors. That means for a zero-mean process, the spatial lag of Y_i is

$$\sum_{j=1}^{n} w_{ij} Y_j, \tag{5.1}$$

where w_{ij} are the spatial weights, often such that rows sum to one or binary values induce from the adjaceny matrix. In practice, $\sum_{j=1}^{n} w_{ij}(y_j - \bar{y})$ is used.

In the case of (arbitrary) lattice data, *Moran's I* and *Geary's C* are often used metrics to assess spatial dependencies. Let $\mathbf{Y}$ be a multivariate random n-vector and $\mathbf{W} = (w_{ij})$ a matrix of spatial weights, often encoding a first-order neighborhood structure. Moran's I (as estimate) is defined as

$$I = \frac{n}{\sum_{i,j} w_{ij}} \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(y_j - \overline{y})(y_i - \overline{y})}{\sum_{i}(y_i - \bar{y})^2}. \tag{5.2}$$

Positive (negative) values of the observed statistic indicate positive (negative) spatial autocorrelation. With weights summing to one, $I \in [-1, 1]$. Note that under no spatial dependency, Moran's I written as estimator satisfies $\mathrm{E}(I) = -1/(n-1)$. Closed-form expressions for the variance exist.

Geary's C (as estimate) is defined as

$$C = \frac{n-1}{2\sum_{i,j} w_{ij}} \frac{\sum_{i,j} w_{ij}(y_i - y_j)^2}{\sum_{i}(y_i - \overline{y})^2} \in [0, 2]. \tag{5.3}$$

Smaller values indicate stronger positive spatial dependency, no spatial dependency is reflected by values around 1. With weights summing to one, $I \in [0, 2]$. Hence, for interpretability and comparability with Moran's I, it would make more sense to consider $1-C$ instead of C. Moran's I and Geary's C are measures of global spatial autocorrelation. However, Geary's C is more sensitive to local spatial autocorrelation.

Example 5.1. R-Code 5.2 illustrates the Moran's I and Geary's C for the *SIDS* data. The construction of the spatial weight matrix $\mathbf{W} = (w_{ij})$ is based on neighbors, i.e., if two counties i and j are neighbors, $w_{ij} = 1$ and zero otherwise. Here we start with `nc` from Example 5.1, a `sf` object, and construct the matrix using first the function `poly2nb()`. The argument `queen=FALSE` implies that more than one shared boundary point is necessary. Typically, this means one shared boundary segment. The neighbor structure is transformed to a spatial weight matrix with `nb2listw()`. The argument `style="W"` enforces a row-sum of one for the weights w_{ij}. ♣

R-Code 5.2: Tests for spatial autocorrelation for the SIDS dataset.

```
(ncnb <- poly2nb(nc, queen=FALSE))     # polygons to neighbors

## Neighbour list object:
## Number of regions: 100
## Number of nonzero links: 462
## Percentage nonzero weights: 4.62
## Average number of links: 4.62
ncW <- nb2listw(ncnb,  style="W")      # neighbors to weightmatrix
moran.test(nc$rates, ncW)              # testing spatial dependencies 1
##
##  Moran I test under randomisation
##
## data:  nc$rates
## weights: ncW
##
## Moran I statistic standard deviate = 3.94, p-value = 4e-05
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##          0.247725         -0.010101          0.004276
ncWNo <- nb2listw(poly2nb(ncNo, queen=FALSE))  # style='W' is default
moran.test(ncNo$rates, ncWNo)[c(1,3)]
## $statistic
## Moran I statistic standard deviate
##                               5.28
##
## $estimate
## Moran I statistic       Expectation          Variance
##         0.3456117        -0.0102041         0.0045413
t2 <- geary.test(ncNo$rates, ncWNo)    # testing spatial dependencies 2
1 - t2$estimate[1]                     # to better compare both
## Geary C statistic
##            0.3565
```

Moran's I and Geary's C measure the dependency "globally", i.e., one single value. It is possible to assess the dependency locally. For each individual area i, the local version of Moran's I is given by

$$I_i = \frac{(y_i - \bar{y})}{\sum_{k=1}^{n}(y_k - \bar{y})^2/(n-1)} \sum_{j=1}^{n} w_{ij}(y_j - \bar{y}), \tag{5.4}$$

where again, different books or software implementations use some variations. The global version is a standardized sum of the local ones (again slight differences in the normalization exist).

The local spatial autocorrelation can further be exploited with functions *`moran.plot()`* and *`localmoran()`* (Bivand *et al.*, 2013, Section 9.3.2). The output of these functions are not straightforward to interpret. Additionally, one needs to be careful not to over-interpret single p-values from the many involved tests.

Example 5.2. R-Code 5.3 illustrates local Moran's I for the `SIDS` data. The visualization is based on a function adapted from https://github.com/gisUTM/spatialplots and included in the chapter's R script. The interpretation is not straightforward, as there seems to be a missing symmetry. The lower panel of Figure 5.2 plots the lagged rates versus the rates. Alignment along the diagonal line indicates spatial dependency. The plot also marks outlying values from the regression `lm(y~Wy)` where `Wy=W %*% y` are the lagged variables. ♣

R-Code 5.3: Tests for spatial autocorrelation for the SIDS dataset.

```r
localI <- localmoran(nc$rates, ncW)
plot.localmoran(nc, "rates", local.moran=localI, weights=ncW)
localINo <- localmoran(ncNo$rates, ncWNo)
plot.localmoran(ncNo, "rates", local.moran=localINo, weights=ncWNo)

moran.plot(ncNo$rates, ncWNo, xlab="SIDS rates", ylab="Lagged variable",
    labels=ncNo$NAME, ylim=c(0, .01), xlim=c(0, 0.01))
abline(c(0,1), col="gray")
```

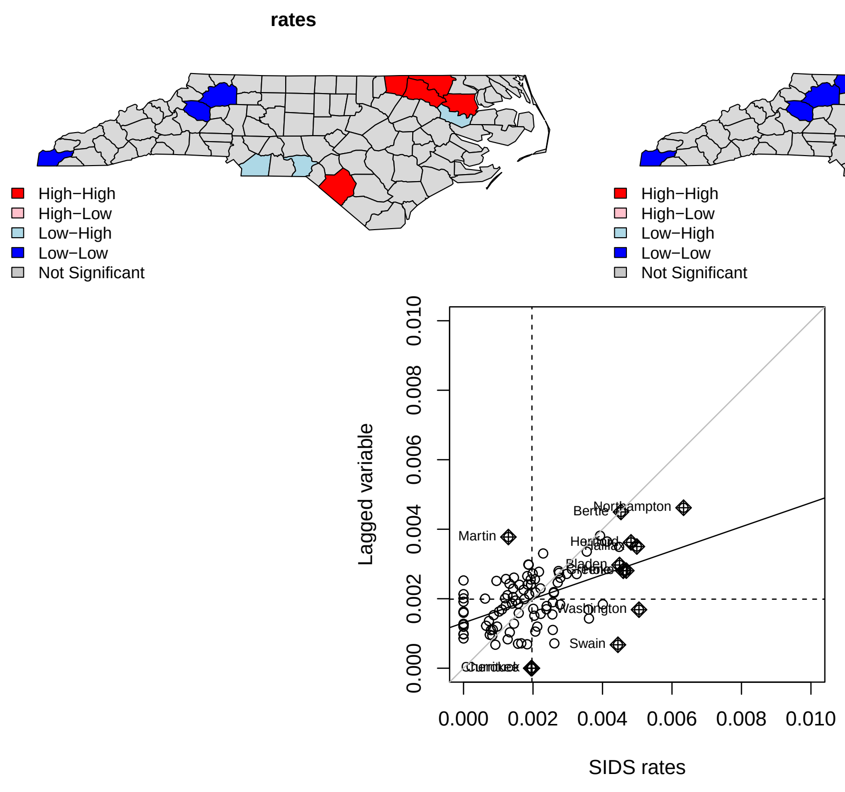


Figure 5.2: Visualization of local Moran's I. (See R-Code 5.3.)

In certain situations, some areas/polygons do not have neighbors. In such situations, most approaches may handle such polygons differently (e.g., the resulting value is zero or `NA`). The argument `zero.policy` specifies the choices.

If data is available on a regular grid, autocovariances along the dimensions can be calculated similarly to time series. Under the assumption of stationarity, it is possible to estimate an autocovariance function based on the distance between the grid points. This approach is much more natural in the framework of geostatistics, and we discuss it extensively in Chapter 10.

5.3 Specific Models for GMRF

We now introduce a low-dimensional parametrization of a GMRFs by reducing the number of parameters $\{b_{ij}\}$ and τ_i^2 under the symmetry constraint and positive definiteness of the precision matrix.

We often assume that $\tau_i = \tau$, for all i. That means that the conditional precision is constant. Another simplification is that the coefficients b_{ij} do not depend on j, i.e., $b_{ij} = b_i$, no preference is given to "neighboring" information. We might even further simplify to $b_{ij} = b = \theta$ for all $j \sim i$, leading to

$$Y_i \mid \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\sum_{j,j\sim i} \theta y_j, \tau^2\Big), \quad \tau > 0, \qquad i = 1, \dots, n, \tag{5.5}$$

where $j \sim i$ indicates a first-order neighbor.

The neighbor structure can be encoded in a matrix, often denoted with $\mathbf{A} = (a_{ij})$ (adjacency matrix) or $\mathbf{W}$ (spatial weight matrix). In the former case, we have a $a_{ij} = 1$ if $i \sim j$ and zero otherwise. In the latter case, we may have $\mathbf{W} = \mathbf{A}$, or w_{ij} is proportional to the number of its neighbors or similar. To link with the last chapter and literature elsewhere, we often write $\mathbf{B} = \lambda\mathbf{W}$, for some λ.

Example 5.3. R-Code 5.4 illustrates the construction of the weight matrix $\mathbf{W}$ using the five counties from the US state Rhode Island. The upper bound of λ is numerically determined. The lower bound can be determined similarly. Note that there might be several zeros, and finding a lower bound for the interval with `uniroot()` needs some care.

Constant conditional precision does not imply constant (marginal) variances, as illustrated, i.e., we have a non-stationary model. ♣

R-Code 5.4: Construction of the weight matrices, valid parameter space, and resulting covariance matrix.

```
ri <- map("county","Rhode Island", fill=TRUE, plot=FALSE)
str(ri, strict.width="cut")
## List of 4
##  $ x     : num [1:144] -71.3 -71.3 -71.2 -71.2 -71.2 ...
##  $ y     : num [1:144] 41.8 41.8 41.8 41.7 41.7 ...
##  $ range: num [1:4] -71.9 -71.1 41.3 42
##  $ names: chr [1:5] "rhode island,bristol" "rhode island,kent" "rhode i"..
##  - attr(*, "class")= chr "map"
```

```
id <- sapply(strsplit(ri$names, ","), function(x) x[2])
ri.poly <- st_as_sf(ri, IDs=id)           # Convert to sf-class
ri.nb <- poly2nb(ri.poly)                 # Convert sf to nb object

(ri.matB <- nb2mat(ri.nb, style="B"))     # only 0-1 entries
##   [,1] [,2] [,3] [,4] [,5]
## 1    0    1    1    1    0
## 2    1    0    1    1    1
## 3    1    1    0    0    1
## 4    1    1    0    0    0
## 5    0    1    1    0    0
## attr(,"call")
## nb2mat(neighbours = ri.nb, style = "B")
(ri.matW <- nb2mat(ri.nb, style="W"))     # row sums to one
##      [,1]    [,2]    [,3]    [,4]    [,5]
## 1 0.00000 0.33333 0.33333 0.33333 0.00000
## 2 0.25000 0.00000 0.25000 0.25000 0.25000
## 3 0.33333 0.33333 0.00000 0.00000 0.33333
## 4 0.50000 0.50000 0.00000 0.00000 0.00000
## 5 0.00000 0.50000 0.50000 0.00000 0.00000
## attr(,"call")
## nb2mat(neighbours = ri.nb, style = "W")
### NOT symmetric (there are also styles "C" or "U")
### Valid parameter range for largest lambda:
f <- function(x, mat) det(diag(5)-x*mat)  # needs to be positive
c(uniroot(f, c(0, 1), mat=ri.matB)$root,  # gets upper bound for both cases
  uniroot(f, c(0, 1.5), mat=ri.matW)$root)
## [1] 0.34066 1.00000
ll <- seq(-2, to=1.5, l=500)              # plot is not shown in the script
plot(ll, sapply(ll, f, mat=ri.matB), type="l")
abline(h=0)
lines(ll, sapply(ll, f, mat=ri.matW), col=4)
### Example of resulting CAR-type covariance matrix. We assume N(0,1)-structure
###   for the errors (see Table 4.1). We fix lambda at 1/2 of possible range.
### We display only diagonal terms:
diag(solve(diag(5) - 0.17 * ri.matB))
## [1] 1.1354 1.1856 1.1354 1.0841 1.0841
diag(solve(diag(5) - 0.5 * ri.matW))
## [1] 1.1368 1.1692 1.1368 1.1031 1.1031
```

We refer to Example 5.8 and corresponding R-Code 5.11 for the discussion of a more complex parameterization, where the coefficients b_{ij} depend on the degree of neighbor, $b_{ij} = b_1$ for adjacent cells (first order neighbors) and $b_{ij} = b_2$ for adjacent cells of adjacent cells (second-order neighbors).

The vignette cran.r-project.org/web/packages/spdep/vignettes/nb.pdf gives further insights in building neighborhood structures (`vignette("nb", package="spdep")`) .

Example 5.4. R-Code 5.5 fits a simple CAR model using the function `spautolm()`. For illustration, we use a small setting based on artificial data. The function's output is similar to a classical `lm()` output. We revisit the output of the function in Example 5.7. ♣

R-Code 5.5: Using the `spautolm()` function to fit a simple model.

```
library("spatialreg")
y <- c(-0.58, -1.22, 1.68, 0.98, 0.44)     # artificial data for RI!
ri.B <- nb2listw(ri.nb, style="B")         # neighbors to weight matrix
carfit <- spautolm(y ~ 1, listw=ri.B, family="CAR") # binary weight matix
summary(carfit, adj.se=FALSE)
##
## Call: spautolm(formula = y ~ 1, listw = ri.B, family = "CAR")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.41115 -0.25837 -0.16130  0.37442  0.45640
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.15434    0.11912  1.2956   0.1951
##
## Lambda: -0.58653 LR test value: 4.8371 p-value: 0.027854
## Numerical Hessian standard error of lambda: 0.050678
##
## Log likelihood: -4.8976
## ML residual variance (sigma squared): 0.18747, (sigma: 0.43298)
## Number of observations: 5
## Number of parameters estimated: 3
## AIC: 15.795
rbind(yhat=fitted(carfit), resid=resid(carfit))
##             1        2      3        4        5
## yhat  -0.4187 -0.96163 1.2236  1.39115 0.065581
## resid -0.1613 -0.25837 0.4564 -0.41115 0.374419
```

Remark 5.1. The function *spautolm()* with argument *family="SAR"* from package *spatialreg* and the function *errorsarlm()* from the same package are essentially identical and thus deliver the same results. ♡

Example 5.5. As a more realistic example, we investigate the spatial dependency for the SIDS dataset. R-Code 5.6 fits a CAR model (5.5) using (i) the neighbor structure used in the literature and (ii) a first-order neighbor structure. As seen in Figure 4.2, the data contains a possible outlier. The code illustrates the influence on the estimates and fit of this single value. For further details, see cran.r-project.org/web/packages/spdep/vignettes/sids.pdf.

Depending on the data source, a slightly different neighborhood structure of the dataset is used, resulting in minor differences. ♣

R-**Code 5.6:** SIDS again. See text for further explanations. (See Figure 5.3.)

```
ncWB <- nb2listw(poly2nb(nc), style="B")
# summary(ncW)
library(spatialreg)    # `spautolm()` was formerly part of package spdep
carfit1 <- spautolm(rates ~ 1, data=nc, listw=ncWB, family="CAR")
# summary(carfit1)     #  we should look at
nc$ratesNO <- nc$rates
nc$ratesNO[index] <- nc$rates[which.max(nc$rates)]/10
carfit2 <- spautolm(ratesNO ~ 1, data=nc, listw=ncWB, family="CAR")
# summary(carfit2)
ncWBNo <- nb2listw(poly2nb(nc[-which.max(nc$rates),]), style="B")
carfit3 <- spautolm(rates ~ 1, data=ncNo, listw=ncWBNo, family="CAR")
nc$fit1 <- fitted(carfit1)     # with outlier
nc$fit2 <- fitted(carfit2)     # "without" outlier
ncNo$fit3 <- fitted(carfit3)   # without county
nc$resid1 <- resid(carfit1)
nc$resid2 <- resid(carfit2)
ncNo$resid3 <- resid(carfit3)
nc$diff1to2 <- fitted(carfit1) - fitted(carfit2)
ncNo$diff3to2 <- fitted(carfit3) - fitted(carfit2)[-which.max(nc$rates)]
rbind(M1=c(coef(carfit1), s2=carfit1$fit$s2, resid=range(nc$resid1)),
      M2=c(coef(carfit2), s2=carfit2$fit$s2,   range(nc$resid2)),
      M2=c(coef(carfit3), s2=carfit3$fit$s2,   range(resid(carfit3))))
##    (Intercept)  lambda          s2     resid1    resid2
## M1   0.0020020 0.13062 2.1205e-06 -0.0021387 0.0076864
## M2   0.0018462 0.15057 1.4774e-06 -0.0022982 0.0033259
## M2   0.0018554 0.15124 1.4835e-06 -0.0023068 0.0033242
```

```r
both <- c(nc$fit1, nc$fit2, ncNo$fit3)
fitbreaks <- seq(0, max(both), by=max(both)/10)
plot(nc[ "fit1"], reset=FALSE, breaks=fitbreaks,key.pos=NULL)
plot(nc[ "fit2"], reset=FALSE, breaks=fitbreaks,key.pos=NULL)
plot(ncNo[ "fit3"], breaks=fitbreaks, key.length=1)
residbreaks <- seq(min(resid(carfit3)), max(resid(carfit1)), len=10)
plot(nc[ "resid1"], reset=FALSE, breaks=residbreaks,key.pos=NULL)
plot(nc[ "resid2"], reset=FALSE, breaks=residbreaks,key.pos=NULL)
plot(ncNo[ "resid3"], breaks=residbreaks, key.length=1)
diffbreaks <- seq(min(nc$diff1to2,ncNo$diff3to2),
                  max(nc$diff1to2,ncNo$diff3to2), len=10)
plot(nc[ c("diff1to2")], breaks=diffbreaks, key.length=1)
plot(ncNo[ c("diff3to2")], breaks=diffbreaks, key.length=1)
```

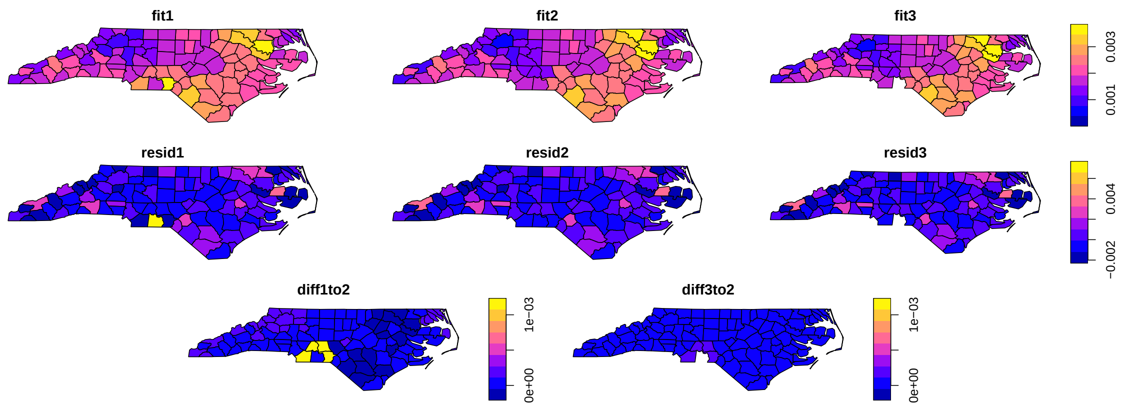


Figure 5.3: Model fits (top row), residuals (middle row), and differences in the fits for SIDS rates (bottom). For the first fit, the original data is used; for the second fit, the value of Anson county is divided by 10; and for the final one, the Anson county is omitted. (See R-Code 5.6.)

The function `spautolm(.~1, listw=., ..., family="CAR")` fits the model $\mathbf{Y} \sim \mathcal{N}_n(\mathbf{1}\beta, (\mathbf{I} - \lambda\mathbf{W})^{-1}\tau^2)$, where the symmetric matrix $\mathbf{W}$ is given by the argument `listw`. It is possible to specify more complex mean functions, in which case the mean of $\mathbf{Y}$ is adapted accordingly. The fitting is done via likelihood estimation. The fitted values consist of the sum of the fitted trend $\mathbf{1}\widehat{\beta}$ and the fitted stochastic signal $\widehat{\lambda}\mathbf{W}(\boldsymbol{y} - \mathbf{1}\widehat{\beta})$ (by the CAR/GMRF property (4.1)), i.e.,

$$\widehat{\boldsymbol{y}} = \mathbf{1}\widehat{\beta} + \widehat{\lambda}\mathbf{W}(\boldsymbol{y} - \mathbf{1}\widehat{\beta}), \tag{5.6}$$

where $\widehat{\beta}$ is the generalized least squares estimate of β.

The argument `family="SAR"` would fit a SAR model, allowing row standardized spatial weight matrices (i.e., `style="W"`). Further fitting functions are `errorsarlm()` and `lagsarlm()` by the package `spatialreg`. The following example illustrates the manual implementation of `spautolm()`.

Example 5.6. Once a model has been specified for specific data (here function *loglikelihood()*), parameter estimation can be carried out with, e.g., the *optim()* function. R Code 5.7 shows how to fit a CAR model based on (5.5) (and data as in Example 5.4). Note that the simple implementation may not work well for arbitrary data sets. We will improve on it later in this chapter. ♣

R-Code 5.7: Defining a likelihood function of a CAR model, "manually" optimizing it and comparing it with the output of the *spautolm()* function.

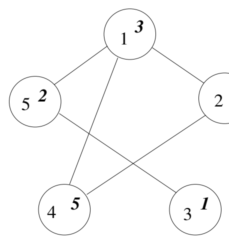
```
loglikelihood <- function(theta) {
    Q <- (diag(n) - theta[2] * W)
    cholQ <- try(chol(Q), silent=TRUE)
    if(inherits(cholQ, "try-error"))  return(1e-6)   # not valid
    resid <- ncNo$rates - theta[1]
    c(- n * log(2*pi * theta[3])/2 + log(det(cholQ)) -
        sum(resid * (Q %*% resid))/(2*theta[3] ))
  }
n <- length(ncNo$rates)
W <- nb2mat(poly2nb(nc[-which.max(nc$rates),]), style="B")

theta0 <- c(0,.15,.15e-6)  # Starting values, similar numbers as fit above
tm1 <- optim(theta0, loglikelihood, control=list(fnscale=-1))
# print(tm1)
rbind(spautolm3=c(coef(carfit3), sigma2=carfit3$fit$s2, logLik=carfit3$LL),
    manual=c(tm1[[1]],tm1[[2]]))
##           (Intercept)  lambda     sigma2 logLik
## spautolm3   0.0018554 0.15124 1.4835e-06 519.48
## manual      0.0018362 0.15738 1.4662e-06 519.42
```

5.4 Exploiting the Sparsity Structure

In the case of GMRF, the off-diagonal non-zero elements of the precision matrix $\mathbf{Q}$ are associated with the conditional dependence structure. As by the Markovian property, the number of neighbors is small, the precision matrix $\mathbf{Q}$ is *sparse*, i.e., contains only $\mathcal{O}(n)$ non-zero elements compared to $\mathcal{O}(n^2)$ for a regular, *full* matrix. To take advantage of the few non-zero elements, special structures to represent the matrix are required, i.e., only the positions of the non-zeros and their values are kept in memory. Because of these special structures, tailored algorithms are required to fully exploit the sparsity structure. The package `spam` provides this functionality; see Furrer and Sain (2009) for a detailed exposition.

The sparsity structure is very important for the calculation based on the precision matrix $\mathbf{Q}$. It determines the (conditional) dependency structure and drives the computational cost. Hence, the sparsity structure is often represented using a graph with edges representing the non-zero elements or a "pixel" image of the zero/non-zero structure. Figure 5.4 gives such an illustration for an "arbitrary" 5×5 matrix **A**.

$$A = \begin{pmatrix} 1 & 0.5 & 0 & 0.5 & 0.5 \\ 0.5 & 1 & 0 & 0.5 & 0 \\ 0 & 0 & 1 & 0 & 0.5 \\ 0.5 & 0.5 & 0 & 1 & 0 \\ 0.5 & 0 & 0.5 & 0 & 1 \end{pmatrix}$$



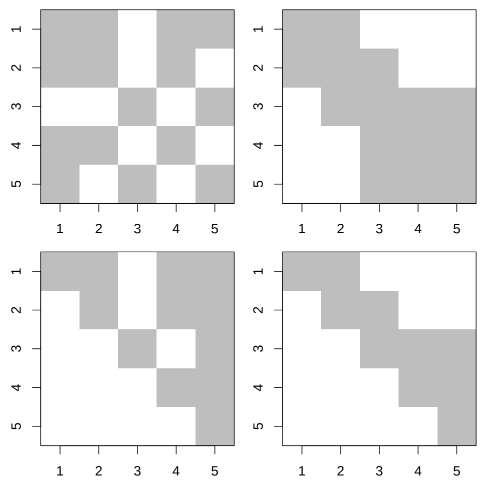


Figure 5.4: The symmetric positive-definite $n = 5$ matrix $\mathbf{A}$ and the sparsity structure of $\mathbf{A}$ and $\mathbf{P}^\top\mathbf{A}\mathbf{P}$ (top row). The graph associated with the matrix $\mathbf{A}$ and the Cholesky factors $\mathbf{R}$ and $\mathbf{U}$ of $\mathbf{A}$ and $\mathbf{P}^\top\mathbf{A}\mathbf{P}$ respectively are given in the bottom row. The nodes of the graph are labeled according to $\mathbf{A}$ (upright) and $\mathbf{P}^\top\mathbf{A}\mathbf{P}$ (italics).

It is important to note that the labeling (order of the variables) influences the structures that result from relevant computations. Reordering is performed through a so-called permutation. In matrix notation, a permutation is a matrix having exactly one element 1 per row and column. All other elements are zero.

5.4.1 The *spam* Package

There are several R packages available to handle sparse matrices: *Matrix*, *SparseM*, *spam*. We use the last one, which provides an extensive set of functions for sparse matrix algebra. Major differences with *Matrix* are: (1) *spam* only supports (essentially) one sparse matrix format, (2) it is based on transparent and simple structure(s), (3) it is tailored for MCMC calculations within GMRF and (4) S3 and S4 like-"compatible" ... and it is fast. R-Code 5.8 gives a quick overview and shows that the handling of sparse matrices is straightforward.

R-Code 5.8: The shortest possible illustration of the package *spam*. The second part contains code for Figure 5.4.

```
library(spam)
mat <- spam(sample(c(0,1), size=18, replace=T, prob=c(.8,.2)), 2, 9)
mat

##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9]
## [1,]    0    0    0    0    0    0    0    0    0
## [2,]    1    0    0    1    0    0    0    0    1
## Class 'spam' (32-bit)
class(mat)
## [1] "spam"
## attr(,"package")
## [1] "spam"
str(mat)
## Formal class 'spam' [package "spam"] with 4 slots
##   ..@ entries    : num [1:3] 1 1 1
##   ..@ colindices : int [1:3] 1 4 9
##   ..@ rowpointers: int [1:3] 1 1 4
##   ..@ dimension  : int [1:2] 2 9
diag(mat) <- 4
solve(mat %*% t(mat), c(1:2))
## [1] 0.038194 0.097222
### Lets construct a second sparse matrix `A`
A <- 0.5 * diag.spam(5)
i <- c(2, 4, 4, 5, 5)
j <- c(1, 1, 2, 1, 3)
A[cbind(i, j)] <- rep(.5, length(i))
A <- t(A) + A       # this is the matrix as in Figure 5.1
summary(A)
## Matrix object of class 'spam' of dimension 5x5,
##     with 15 (row-wise) nonzero elements.
##     Density of the matrix is 60%.
## Class 'spam' (32-bit)
U <- chol(A)
class(U)            # Special class, you do not really need to work with.
## [1] "spam.chol.NgPeyton"
## attr(,"package")
## [1] "spam"
(pivot <- U@pivot)  # The permutation is found automatically!
## [1] 3 5 1 2 4
U@invpivot          # Inverse of the permutation,  see also `?permutation`
## [1] 3 4 1 5 2
P <- diag.spam(5)[U@invpivot,]
norm(A[ pivot, pivot] - t(P) %*% A %*% P ) # sum of squared differences.
## [1] 0
```

```
spam::display(A)
spam::display(U)
```

5.4.2 Never Calculate the Actual Inverse of a SPD Matrix

Covariance matrices (and thus precision matrices) are symmetric and positive definite matrices. When calculating or maximizing multivariate normal log-likelihoods, we need to calculate determinants ($\det(\boldsymbol{\Sigma})$) and quadratic forms ($\boldsymbol{y}^\top \boldsymbol{\Sigma}^{-1} \boldsymbol{y}$).

To be more specific, assume we need to calculate $\mathbf{A}^{-1}\boldsymbol{b}$ with $\mathbf{A}$ a symmetric positive-definite matrix featuring some sparsity structure, which is usually accomplished by solving $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}$. We proceed by factorizing $\mathbf{A}$ into $\mathbf{R}^\top\mathbf{R}$, where $\mathbf{R}$ is an upper triangular matrix, called the Cholesky factor or Cholesky triangle of $\mathbf{A}$, followed by solving $\mathbf{R}^\top\boldsymbol{y} = \boldsymbol{b}$ and $\mathbf{R}\boldsymbol{x} = \boldsymbol{y}$, called forwardsolve and backsolve, respectively. Note that the exposition could be done with the lower triangular matrix $\mathbf{L} = \mathbf{R}^\top$.

From a computational point of view, there is a huge difference between `solve(A)%*% b` and `solve(A, b)`!

Calculating the determinant can be done through various paths. For symmetric positive definite matrices, the best approach is to perform a Cholesky factorization and use the property

$$\det(\mathbf{A}) = \det(\mathbf{R}^\top\mathbf{R}) = \det(\mathbf{R})^2 = \prod_i r_{ii}^2, \tag{5.7}$$

where r_{ii} are the diagonal entries of $\mathbf{R}$.

Notice that the Cholesky factor can be seen as a *matrix square root* and is thus used when drawing multivariate standard random variables as well; see Section 1.2.2.

5.4.3 Solving Linear Systems

The Cholesky factor of a banded matrix is again a banded matrix. However, arbitrary sparse matrices may produce full Cholesky factors. To reduce this so-called *fill-in* of the Cholesky factor $\mathbf{R}$, we permute the columns and rows of $\mathbf{A}$ according to a (cleverly chosen) permutation $\mathbf{P}$, i.e., $\mathbf{U}^\top\mathbf{U} = \mathbf{P}^\top\mathbf{A}\mathbf{P}$, with $\mathbf{U}$ an upper triangular matrix. Many different algorithms exist to find permutations that are optimal for specific matrices or at least close to optimal with respect to different criteria. The cost of finding a good permutation matrix $\mathbf{P}$ is at least of order $\mathcal{O}(n^{3/2})$ (for lattices in two dimensions).

Note that $\mathbf{R}$ and $\mathbf{U}$ cannot be linked through $\mathbf{P}$ alone. Figure 5.4 illustrates the factorization with and without permutation. Two triangular solves are performed after the factorization for solving a linear system. The determinant of $\mathbf{A}$ is the squared product of the diagonal elements of its Cholesky factor $\mathbf{R}$. Hence the same factorization can be used to calculate determinants (a necessary and computational bottleneck in the computation of the log-likelihood of a Gaussian model), illustrating that it is crucial to have a very efficient integration (with respect to calculation time and storage capacity) of the Cholesky factorization.

A typical Cholesky factorization of a sparse matrix consists of the steps illustrated in the following pseudo-code algorithm.

```
[1]   Determine permutation and permute the input matrix $\mathbf{A}$ to obtain $\mathbf{P}^\top\mathbf{A}\mathbf{P}$
[2]   Symbolic factorization, where the sparsity structure of $\mathbf{U}$ is constructed
[3]   Numeric factorization, where the elements of $\mathbf{U}$ are computed
```

When factorizing matrices with the same sparsity structure, Steps 1 and 2 do not need to be repeated. In MCMC algorithms, this is commonly the case, and exploiting this shortcut leads to very considerable gains in computational efficiency (we revisit this in the coming chapters).

As for Step 1, there are many different algorithms to find a permutation, for example, the multiple minimum degree (MMD) algorithm, (Liu, 1985), and the reverse Cuthill-McKee (RCM) algorithm, (George, 1971). The resulting sparsity structure in the permuted matrix determines the sparsity structure of the Cholesky factor. As an illustration, R-Code 5.9 and Figure 5.5 illustrate the sparsity structure of the Cholesky factor resulting from an MMD, an RCM, and no permutation of a precision matrix induced by a second-order neighbor structure of the US counties.

R-Code 5.9: Illustrating the sparsity structure of the Cholesky factor using different permutation schemes implemented in *spam*. (See Figure 5.5.)

```r
In <- diag.spam(nrow(UScounties.storder))
Q <- In + .1 * UScounties.storder + .1 * UScounties.ndorder
summary(Q)
## Matrix object of class 'spam' of dimension 3082x3082,
##     with 59978 (row-wise) nonzero elements.
##     Density of the matrix is 0.631%.
## Class 'spam' (32-bit)
struct <- chol(Q)
spam::display(Q, ylab="", xlab="", cex=1)  # Without cex, a warning is issued
spam::display (struct, ylab="", xlab="", cex=1)
summary(struct)
## (Upper) Cholesky factor of class 'spam.chol.NgPeyton' of dimension 3082x
## 3082 with 146735 (row-wise) nonzero elements.
##     Density of the factor is 1.54%.
##     Fill-in ratio is 4.65
##     (Optimal argument for 'chol' is 'memory=list(nnzR=146735)'.)
## Class 'spam.chol.NgPeyton'
(nnzMMD <- sum(struct@entries > .Machine$double.eps ))
## [1] 81345
```

```
struct <- chol(Q, pivot="RCM")
spam::display(struct, ylab="", xlab="", cex=1)
summary(struct)
## (Upper) Cholesky factor of class 'spam.chol.NgPeyton' of dimension 3082x
## 3082 with 256198 (row-wise) nonzero elements.
##     Density of the factor is 2.7%.
##     Fill-in ratio is 8.13
##     (Optimal argument for 'chol' is 'memory=list(nnzR=256198)'.)
## Class 'spam.chol.NgPeyton'
(nnzRCM <- sum(struct@entries > .Machine$double.eps ))
## [1] 135457
struct <- chol(Q, pivot=FALSE)
spam::display (struct, ylab="", xlab="", cex=1)
summary(struct)
## (Upper) Cholesky factor of class 'spam.chol.NgPeyton' of dimension 3082x
## 3082 with 689615 (row-wise) nonzero elements.
##     Density of the factor is 7.26%.
##     Fill-in ratio is 21.9
##     (Optimal argument for 'chol' is 'memory=list(nnzR=689615)'.)
## Class 'spam.chol.NgPeyton'
(nnzNONE <- sum(struct@entries > .Machine$double.eps ))
## [1] 270140
```

How much fill-in with zeros is present depends on the permutation algorithm. In the example of Figure 5.5 there are 146 735, 256 198 and 689 615 non-zero elements in the Cholesky factors with MMD, RCM, and no permutation, respectively. Note that the actual number of non-zero elements of the Cholesky factor may be smaller than what the constructed sparsity structure indicates, Here, there are 81345, 135457 and 270140 zero elements (up to machine precision) that are not exploited.

We finish this section with examples illustrating further the functionality of *spam* in the context of GMRFs.

Example 5.7. We manually implemented the *spautolm()* using sparse matrix functionality. Once a model has been specified for specific data (here function *mle.CAR()*), parameter estimation is carried out with the *optim()* function. R Code 5.10 is similar to R Code 5.7 and due to the better implementation also more stable and thus closer to the *spautolm()* results. Both approaches estimate the uncertainties differently and thus slight differences.

The function *mle.CAR()* is very similar to the function *spam::mle.nomean()*, both represent a rudimentary approach to estimate parameters in a multivariate normal setting. ♣

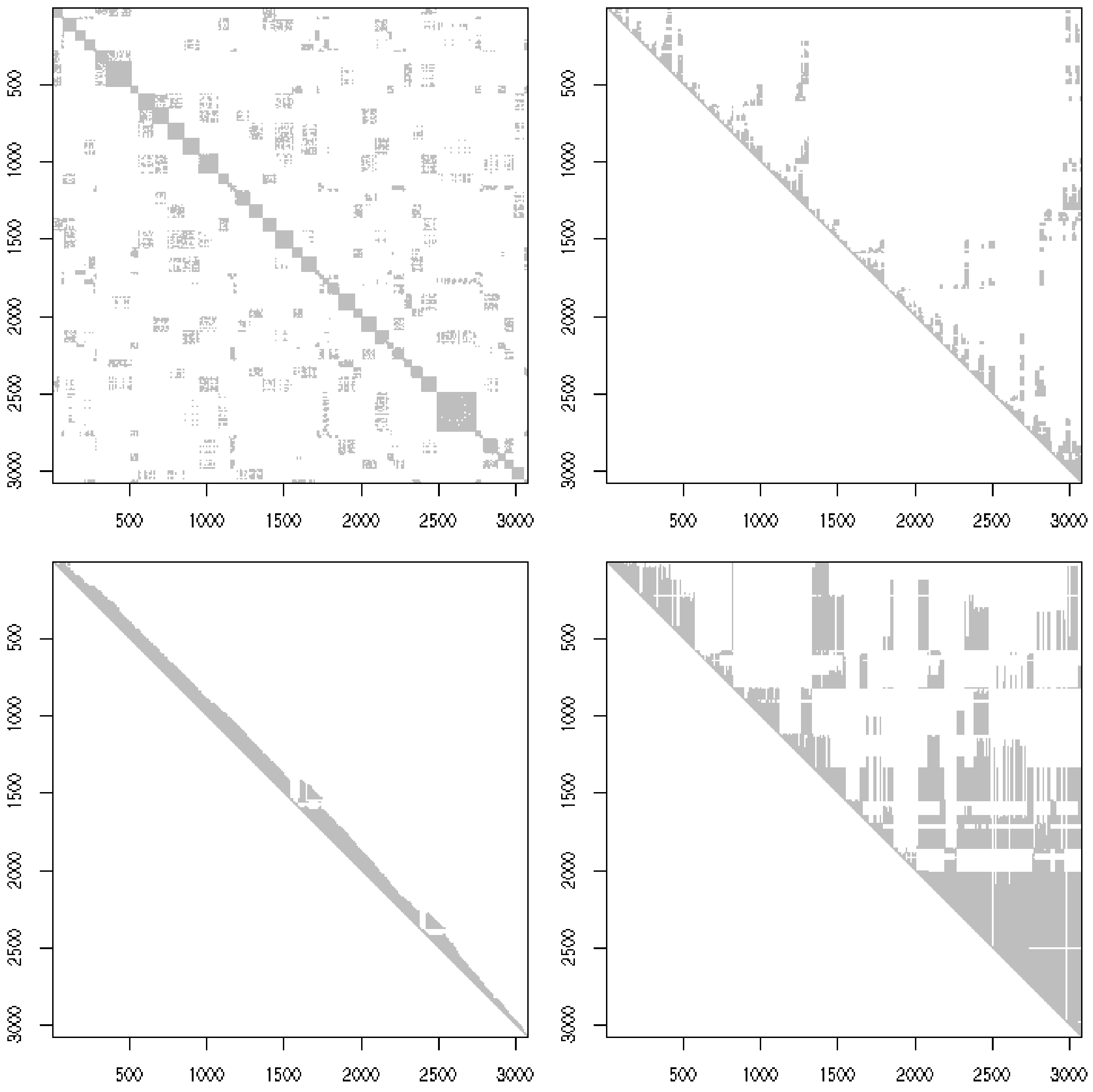


Figure 5.5: Top left represents the sparsity structure of a precision matrix induced by a second-order neighbor structure of the US counties. Sparsity structure of the Cholesky factor with MMD (top right), RCM (bottom left), and no permutation of the precision matrix. (See R-Code 5.9.)

R-Code 5.10: Defining a likelihood function of a CAR model, "manually" optimizing it and comparing it with the output of the `spautolm` function.

```
options(spam.cholupdatesingular="null")
mle.CAR <- function (y, W, theta)  {
  n <- length(y)
  In <- diag.spam(n)
  Qstruct <-  chol(In - 0.0001 * W)
  neg2loglikelihood <- function(theta) {
    Q <-  (In - theta[2] * W)
    cholQ <- update(Qstruct, Q)
    if (is.null(cholQ)) return(1e6)
    resid <- y - theta[1]
    return(n * log(2*pi * theta[3]) - 2*c(determinant(cholQ)$modulus) +
        sum(resid * (Q %*% resid))/theta[3] )
  }
  out <- optim(theta, neg2loglikelihood, method="L-BFGS-B",
               lower=c(-Inf, -Inf, 1e-5), hessian=TRUE)
  if (out$convergence !=0) cat("Convergence issues, please inspect\n")
  return(out)
}

listw <- nb2listw(ri.nb, style="B")
W <- as.spam.listw(listw)
print(tm1 <- mle.CAR(y , W, c(0,-.1,1)))
## $par
## [1]  0.15433 -0.58652  0.18748
##
## $value
## [1] 9.7953
##
## $counts
## function gradient
##       50       50
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
##
## $hessian
##              [,1]     [,2]        [,3]
## [1,] 140.9364476   -9.608   4.8488e-03
## [2,]  -9.6080461 1127.902 -2.2225e+02
## [3,]   0.0048488 -222.249   1.4229e+02
rbind(spamautolm=c(coef(carfit), sigma2=carfit$fit$s2, logLik=carfit$LL),
    manual=c(tm1[[1]],tm1[[2]]/-2))
##            (Intercept)   lambda  sigma2  logLik
## spamautolm     0.15434 -0.58653 0.18747 -4.8976
## manual         0.15433 -0.58652 0.18748 -4.8976
### Slight differences in the uncertainty estimates:
rbind(spamautolm=c(se.Inter=summary(carfit)$Coef[,2], se.lambda=
    carfit$lambda.se), manual=c(sqrt(solve(tm1$hessian/2)[c(1,5)])))
##            se.Inter se.lambda
## spamautolm  0.11912  0.050678
## manual      0.11918  0.050633
```

Example 5.8. Illustration of numerically determine the valid parameter space. Consider a CAR model with a first- and a second-order neighbor structure. More specifically, we consider

$$Y_i \mid \boldsymbol{y}_{-i} \sim \mathcal{N}\Big(\sum_{j,j\sim i} \theta_1 y_j + \sum_{j,j\approx i} \theta_2 y_j, \tau^2\Big), \quad \tau > 0, \qquad i = 1, \dots, n, \tag{5.8}$$

where $j \sim i$ and $j \approx i$ indicate first- and second-order neighbors, respectively. That means, the resulting precision matrix $\mathbf{Q}$ has the structure $\mathbf{Q} = \tau^{-2}(\mathbf{I} - \theta_1\mathbf{W}_1 - \theta_2\mathbf{W}_2)$ where $\mathbf{W}_i$ are (binary) spatial weight matrices. We have to impose constraints on (θ_1, θ_2), such that $\mathbf{Q}$ is positive definite, i.e., we need to determine the valid parameter space $\boldsymbol{\Theta} \subset \mathbb{R}^2$. Here, we do not have a closed form description of the parameter space (recall Example 5.3).

We first construct an (arbitrary but valid) precision matrix based on the first- and second-order structure to exploit the `spam` options. To ensure validity, we choose very small values of (θ_1, θ_2), as we know that $(0, 0) \in \boldsymbol{\Theta}$. Then, we cycle over a specified fine grid of `theta1` and `theta2` and verify if the precision matrix is positive definite. If the matrix passed to `update()` is not symmetric positive definite, which means that the value of `tmp` is `NULL`, the pair (`theta1[i]`, `theta2[2]`) lies not within $\boldsymbol{\Theta}$. Figure 5.6 shows $\boldsymbol{\Theta}$. The valid range is color-coded according to the value of $\log(\det \mathbf{Q})$. Notice the domain's asymmetry and the determinant's very small values. The maximum (being zero) is at the origin, see also Remark 5.2.

Based on the fine grid used, the loop takes several minutes. Naturally, the convexity of the domain could easily be exploited. Of course, in practice, the precise space $\boldsymbol{\Theta}$ is not necessary for optimization in a maximum likelihood estimation context. Similarly, in Bayesian settings, it would be possible to place a prior over $\boldsymbol{\Theta}$ without knowing it explicitly. ♣

Remark 5.2. In the case of independence, the precision/covariance matrix is the identity. For simplicity, assume $\sigma^2 = 1$, then the determinant is equal to one, as all eigenvalues are 1. Adding spatial structure will decrease the determinant, as some of the eigenvalues will be larger than one, others smaller. However, since the sum of the eigenvalues remains constant (here n), the product of these decreases. ♡

R-**Code 5.11** Determining the valid parameter space **Θ** through a numerical assessment of the precision matrix. (See Figure 5.6.)

```
In <- diag.spam(nrow(UScounties.storder))
struct <- chol(In - .1 * UScounties.storder - .01 * UScounties.ndorder)
###    which is a valid, but (arbitrary) precision matrix.
options(spam.cholupdatesingular="null")      # We want to avoid errors.
len1 <- 300      # even
len2 <- 150
theta1 <- seq(-.515, .225, len=len1)
theta2 <- seq(-.19, .095, len=len2)
grid <- array(NA, c(len1, len2))
for (i in 1:len1) {
  for(j in 1:len2) {
    tmp <- update(struct, In - theta1[i]*UScounties.storder
                              - theta2[j]* UScounties.ndorder)
    if(!is.null(tmp))   grid[i,j] <- determinant(tmp)$modulus
  }
}
image.plot(theta1, theta2, grid, xlab=expression(theta[1]),
            ylab=expression(theta[2]), xlim=c(-.45, .22), ylim=c(-.2, .1))
abline(v=0, h=0, lty=2)
```

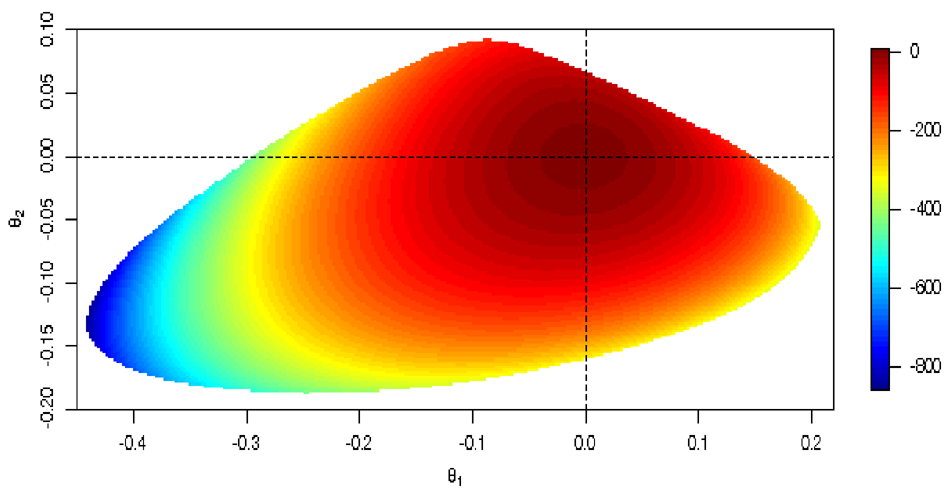


Figure 5.6: The domain **Θ** for the US counties with a second-order neighbor structure. The values represent $\log(\det \mathbf{Q})$ and the white area represents $\mathbf{\Theta}^c$. (See R-Code 5.11.)

5.5 Further Example

We consider here an additional example based on the oral cavity cancer data. We fit a simple CAR model with a binary weight matrix to the standardized mortality rates. The weight matrix is constructed in two different ways for illustrative purposes: once with the build-in function *adjacency.landkreis()* from the package *spam* and second manually based on a file defining the neighbor list. The same package also provides the latter.

The estimated and fitted values of *spautolm()* and the manual approach are very similar, as R-Code 5.12 shows. We also manually reconstruct the fitted values according equation (5.6) with *fitTrend* being where $\widehat{\beta} = \mathbf{1}^\top\widehat{\mathbf{Q}}\boldsymbol{y}/\mathbf{1}^\top\widehat{\mathbf{Q}}\mathbf{1}$, i.e., the generalized least squares estimate of β, $\widehat{\mathbf{Q}} = (\mathbf{I}_n - \widehat{\lambda}\mathbf{W})/\widehat{\tau^2}$ and *fitStochastic* being $\widehat{\lambda}\mathbf{W}(\boldsymbol{y} - \mathbf{1}\widehat{\beta})$. ($\tau^2$ is denoted with *sigma2* in the output of *spautolm()*.)

The range of possible values of λ is approximately from -0.296 to 0.159. The estimated value $\widehat{\lambda} = 0.152$ is quite close to the boundary of the valid range. This often indicates that the proposed CAR model is not sufficiently flexible. This might also be seen as the drawn realizations seem more speckled than the observed data (see Figure 5.7). For a better comparison, we start with the same seed, such that the first $n = 544$ random numbers from *rnorm()* are equivalent. For the GMRF realization, they are further transformed but the features can be seen in both panels (denoted with "White noise" and "Sample 1").

Notice that only the marginal conditional precision is constant, not the marginal variances or marginal standard deviation. Districts that are within another one have a particular low marginal standard deviation.

R-Code 5.12: Oral cavity cancer example. (See Figure 5.7.)

```
library(spam)
data(Oral, package="spam")
hist(Oral$SMR, main="", xlab="SMR")
abline(v=mean(Oral$SMR), col=4, lwd=2)
filename <- system.file("demodata/germany.adjacency", package="spam")
# system(paste("head ", filename), intern=F)  # show content of the file
W <- adjacency.landkreis(filename)
barplot(table(diff(W@rowpointers)), xlab="# of neighbors")
### The following uses as input a classical ASCII format of nb files.
n <- as.numeric(readLines(filename, n=1))
nnodes <- nodes <- numeric(n)
adj <- list()
for (i in 1:n) {
  tmp <- as.numeric(scan(filename, skip=i, nlines=1, quiet=T,
                         what=list(rep("", 13)))[[1]])
  nodes[i] <- tmp[1]
  nnodes[i] <- tmp[2]
  adj[[i]] <- as.integer(tmp[-c(1:2)]+1)
}
adj <- adj[order(nodes)]
attr(adj, "region.id") <- germany.info$id
attr(adj, "sym") <- TRUE
class(adj) <- "nb"
Dlistw <- nb2listw(adj, style="B")
W1 <- as.spam.listw(Dlistw)
all.equal.spam(W, W1)

## [1] TRUE

# summary(Dlistw)
# table(unlist(lapply(adj, length)))

### We can now start spatial modeling. For example a simple CAR:
carfit <- spautolm(SMR ~ 1, data=Oral, listw=Dlistw, family="CAR")
###   strictly speaking, we should include an offset of 1 here:
#  lm(SMR ~ offset(rep(1,n))+ 1, data=Oral)
### apparently spautolm() does not incorporate this.
options(spam.cholupdatesingular="null")
tm1 <- mle.CAR(Oral$SMR, W, c(0,.1,2))
### Comparing results of `spautolm()` and manual fitting:
coefs <- c(coef(carfit), sigma2=carfit$fit$s2)
rbind(spamautolm= c(coefs, logLik=carfit$LL),
    manual=c(tm1[[1]], tm1[[2]]/-2))

##            (Intercept)  lambda   sigma2  logLik
## spamautolm       1.001 0.15219 0.088880 -141.58
## manual           1.001 0.15216 0.088894 -141.58

Q1 <- (diag.spam(n)- tm1$par[2]*W1)/tm1$par[3]
fitTrend <- c(rep(1, n) %*% Q1 %*% Oral$SMR)/c(rep(1, n) %*% Q1 %*% rep(1, n))
c(fitTrend - unname(carfit$fit$signal_trend[1]))  # constant trend here

## [1] -2.5552e-05

fitStochastic <-  tm1$par[2]*W1 %*% (Oral$SMR-tm1$par[1])
summary(c(fitStochastic - carfit$fit$signal_stochastic))

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.45e-04  2.66e-06  2.80e-05  2.85e-05  5.91e-05  1.34e-04

###   -> all values (estimates and fits) very similar!

### Construct and visualize valid parameter range for lambda:
lambda <- c(seq(-.2959, to=-.2958, by=0.00005),
            seq(.159, to=.1591, by=0.00005))
```

```
for (i in 1:length(lambda))
  if (class(try(chol(diag.spam(n)-lambda[i]*W),silent=TRUE))!=
                         "spam.chol.NgPeyton") lambda[i] <- NA
lambda   # estimates are quite on the boundaries!!!
## [1]       NA -0.29585 -0.29580  0.15900  0.15905       NA
zl <- c(-0.6, 2.4)
germany.plot(Oral$SMR, main="SMR",border=NA, zlim=zl)
set.seed(14)
germany.plot(rnorm(n, mean=coefs[1], sd=sqrt(coefs[3])),
             main="White noise", border=NA, zlim=zl)
N <- 1000     # We simulate many realizations with similar parameters
Q <- (diag.spam(n)- coefs[2]*W)/coefs[3]
set.seed(14)
ex <- rmvnorm.prec(N, mu=coefs[1], Q=Q)
lambdahat <- numeric(4)  # We only look at four
for (i in 1:4) {
  germany.plot(ex[i,], main=paste("Sample",i),border=NA, zlim=zl)
#   mle.CAR(ex[i,], W, c(1,.12,09))  # leads often to convergence issues!!
  lambdahat[i] <- spautolm(ex[i,] ~ 1, listw=Dlistw, family="CAR")$lambda
}
lambdahat
## [1] 0.15437 0.15669 0.14492 0.15754
germany.plot(colMeans(ex), main="Means", border=NA)
germany.plot(apply(ex, 2, sd), main="SD", border=NA)
germany.plot(diag(solve(cov(ex))), main="Precision", border=NA)
```

We extend our model to two parameters, much in the spirit of Equation (5.8). As a first step, we define in R-Code 5.13 a spatial weight matrix with second-order neighbors (here `W2`). Similarly, as in R-Code 5.11, we determine the valid parameter domain, shown in Figure 5.8.

R-Code 5.13: Oral cavity cancer example: valid parameter domain. (See Figure 5.8.)

```
options(spam.cholupdatesingular="null")
listw2 <- nblag(adj,2)[[2]]   # constructs higher order neighbors
W2 <- as.spam.listw(nb2listw(listw2, style="B"))
In <- diag.spam(nrow(W2))
W1[4,1:20]     # 3 first order
##  [1] 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0
W2[4,1:20]     # 8 second order neighbors
```

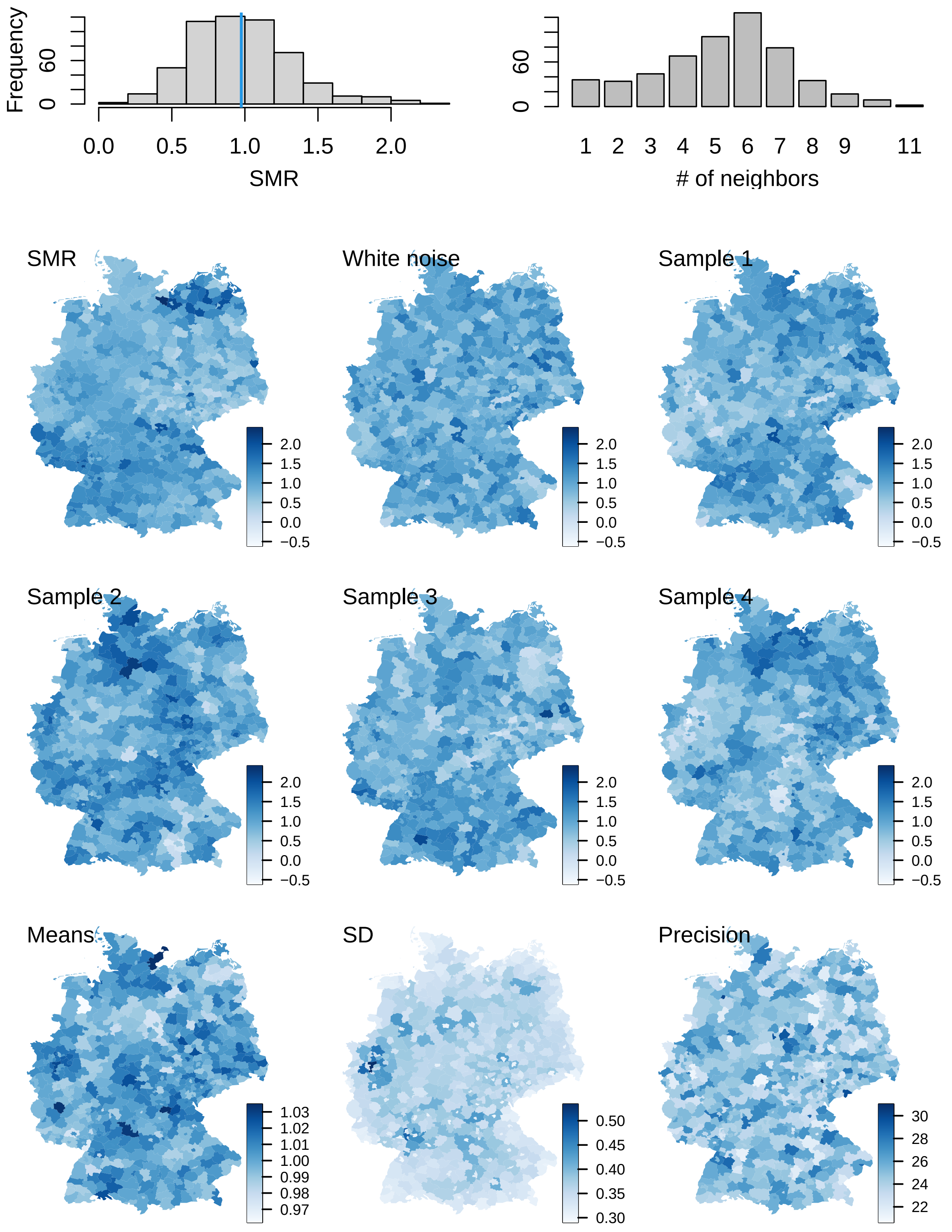


Figure 5.7: Top row: histogram of SMR of oral cavity cancer (left) and the number of neighbors of each of the districts (right). Bottom panels: SMR, white noise comparison, four samples having the same mean and precision matrix (unconditional simulation), means, marginal standard deviation, and precision of 1000 samples. (See R-Code 5.12.)

```
##  [1] 0 1 0 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 0
len1 <- 60
len2 <- 56
theta1 <- seq(-.46, .25, len=len1)
theta2 <- seq(-.21, .12, len=len2)
grid <- array(NA, c(len1, len2))
Qstruct <-  chol(In - 0.0001 * W1 - 0.0001 * W2)
for (i in 1:len1) {
  for(j in 1:len2) {
    grid[i,j] <- !is.null(update(Qstruct, In - theta1[i]*W1 - theta2[j]* W2))
} }
image(theta1, theta2, grid, xlab=expression(theta[1]),
             ylab=expression(theta[2]), col=c(0,"gray"))
abline(v=0, h=0, lty=2)
points(tm1[[1]][2], 0, col=4, pch=19)  # optimum one parameter case
points(0.0334, 0.0567, col=2, pch=19)  # fit which we will obtain later ;-)
```

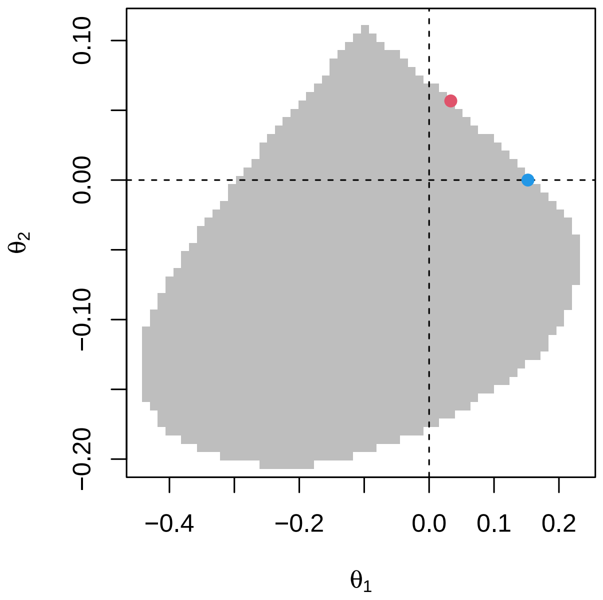


Figure 5.8: Valid parameter domain. The blue and red dots indicate the one- and two-parameter model estimates, respectively. The image consists of a regular 60×56 grid. (See R-Code 5.13.)

In the second step, we define in R-Code 5.14 the corresponding likelihood function, now optimizing over four parameters (intercept β, λ_1, λ_2 and τ^2). We extend the function `mle.CAR()` of R-Code 5.10 to incorporate the additional neighbor structure (`mle.CAR2()`). The estimation is very fast (less than a couple of seconds, although there are many likelihood evaluations, see `tm2$counts[1]`).

R-**Code 5.14:** Oral cavity cancer example: second order neighbor structure. (See Figure 5.9.)

```
options(spam.cholupdatesingular="null")
mle.CAR2 <- function (y, W1, W2, theta)  {
  n <- length(y)
  In <- diag.spam(n)
  Qstruct <-  chol(In - 0.0001 * W1 - 0.0001 * W2)
  neg2loglikelihood <- function(theta) {
    Q <- (In - theta[2] * W1 - theta[3] * W2)
    cholQ <- update(Qstruct, Q)
    if (is.null(cholQ)) return(1e6)
    resid <- y - theta[1]
    return(n * log(2*pi * theta[4]) - 2*c(determinant(cholQ)$modulus) +
                sum(resid * (Q %*% resid))/theta[4] )
  }
  out <- optim(theta, neg2loglikelihood, method="L-BFGS-B",
                 lower=c(-Inf, -Inf, -Inf, 1e-5), hessian=TRUE)
  if (out$convergence !=0) cat("Convergence issues, please inspect\n")
  return(out)
}

listw2 <- nblag(adj,2)[[2]]   # constructs higher order neighbors
W2 <- as.spam.listw(nb2listw(listw2, style="B"))
tm2 <- mle.CAR2(Oral$SMR, W1, W2, c(1,-0.0,-0.0,.1)) # W1 from above
## Convergence issues, please inspect
print(unlist(tm2[c(2,1)]))    # Hessian and other stuff not relevant here
##      value       par1       par2       par3       par4
## 274.491868   0.994615   0.033386   0.056566   0.105442
print(unlist(tm1[c(2,1)]))    # quite a few function calls
##      value       par1       par2       par3
## 283.165444   1.001000   0.152156   0.088894
Q2 <- (diag.spam(n)- tm2$par[2]*W1 - tm2$par[3]*W2)/tm2$par[4]

fitW1 <- tm1$par[2]*W1
fitW2 <- tm2$par[2]*W1 + tm2$par[3]*W2
set.seed(14)
ex2 <- rmvnorm.prec(1, mu=tm2$par[1], Q=Q2)

germany.plot(ex[1,], main="Sample 1", border=NA, zlim=zl)
germany.plot(ex2[1,], main="Sample 1, 2 pars",border=NA, zlim=zl)
germany.plot(ex2[1,]-ex[1,], main="Difference",border=NA)
```

```
fitStochastic1 <- fitW1 %*% (Oral$SMR-tm1$par[1])
fitStochastic2 <- fitW2 %*% (Oral$SMR-tm2$par[1])
zS <- range(fitStochastic1, fitStochastic2)
germany.plot(fitStochastic1, main="Stochastic fit, 1 par", border=NA, zlim=zS)
germany.plot(fitStochastic2, main="Stochastic fit, 2 par" ,border=NA, zlim=zS)
germany.plot(fitStochastic1-fitStochastic2, main="Difference", border=NA)
resid1 <- Oral$SMR - (tm1$par[1] + fitStochastic1) # tm1$par[1] is fitTrend
resid2 <- Oral$SMR - (tm2$par[1] + fitStochastic2) #
c(norm(resid1), norm(resid2))
## [1] 126.42 123.69
zR <- range(resid1, resid2)
germany.plot( resid1, main="Residuals, 1 par", border=NA, zlim=zR)
germany.plot( resid2, main="Residuals, 2 par", border=NA, zlim=zR)
germany.plot( resid1-resid2, main="Difference", border=NA)
```

For the two-parameter case, the optimal fitted value remains very close to the boundary (Figure 5.8). The negative-two-loglikelihood is decreased from 283.2 to 274.5. However, the residual standard deviation increased from 0.304 to 0.297. Samples drawn from the estimated precision matrix are somewhat smoother than with one parameter only, as shown by Figure 5.9 for one specific example. The same holds for the fitted values. However the additional smoothness comes at a price of less flexibility. The fitted surface represents a poorer fit.

As shown with the figures and R-chunks above, a one and two-parameter model is not optimal. We will revisit this example with a different modeling strategy.

5.6 *Details of Spatial Classes

All spatial objects (data, locations, etc.) are linked to a bounding box (defining the spatial domain, slot `bbox`) and a coordinate reference system (defining map projections and transformations to references projections, slot `proj4string`, itself of class `CRS`) (Bivand *et al.*, 2013, Section 4.1). For many years the package `sp` provided many different classes for spatial objects. The core class for the spatial objects is `Spatial` with many subclasses, e.g., `SpatialPoints` (extending with a `coords` slot), `SpatialLines` (extending with a `lines` slot), `SpatialPolygons`, etc. Working with the package `sf` is much more intutitive as shown below.

R-Code 5.15: `Spatial` class of package `sp`.

```
library(sp)
# getClass("Spatial") # structure of the class and its subclasses
# vignette("intro_sp", package="sp")
getSlots("SpatialPolygons")
##    polygons    plotOrder        bbox proj4string
```

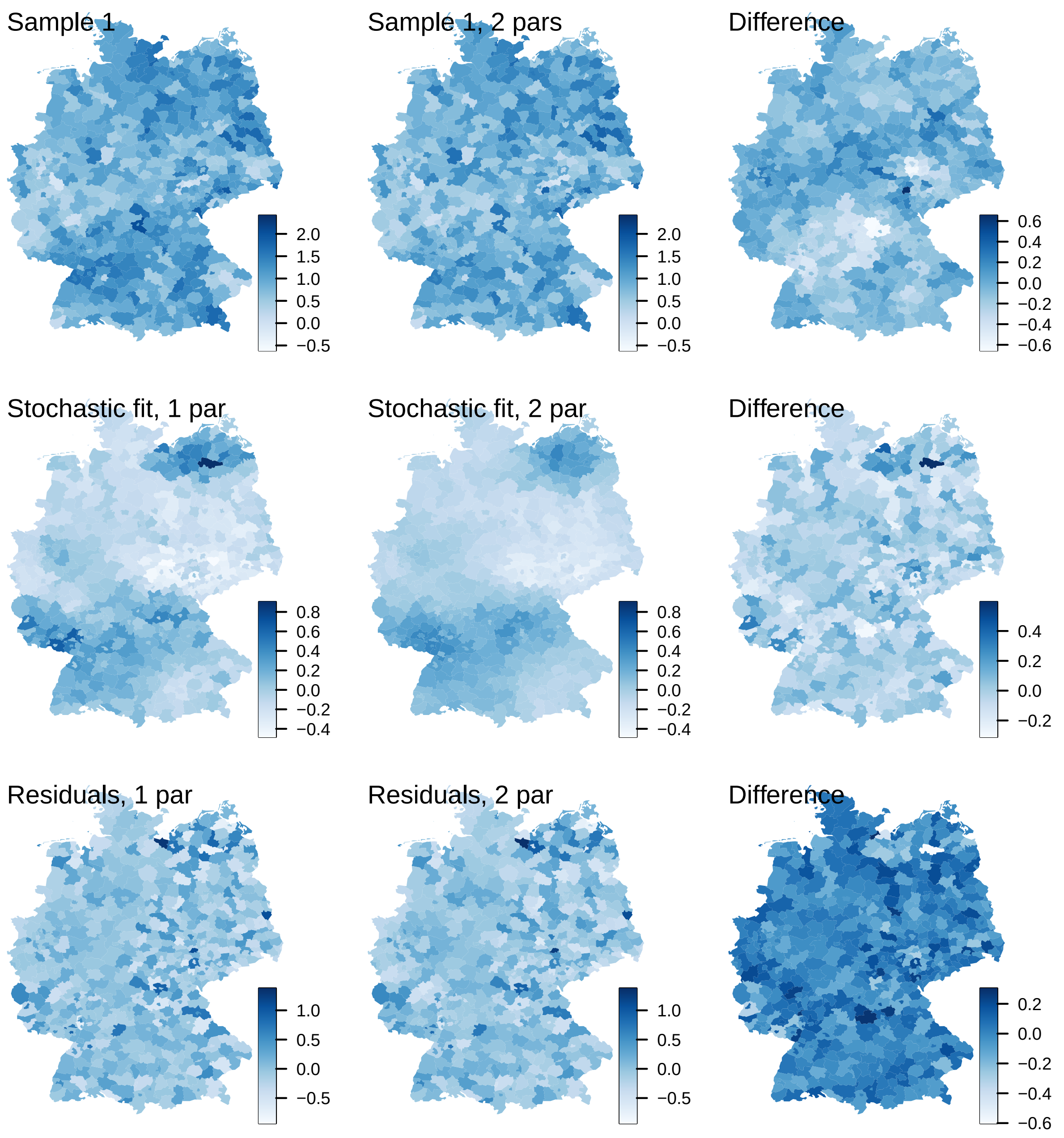


Figure 5.9: Comparison of a realization, fitted values and residuals with a one- and two-parameter models. The top left panel is identical to Figure 5.7. The right column is the difference between the first two. (See R-Code 5.14.)

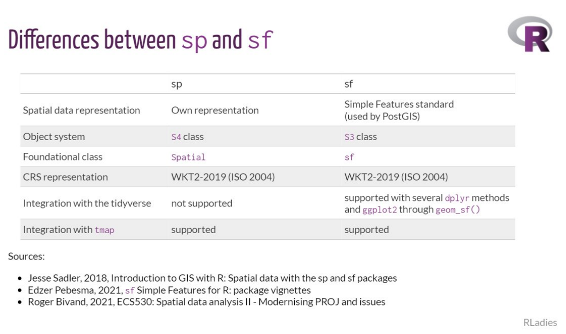
```
##       "list"    "integer"     "matrix"        "CRS"
```

For example, the manual construction of a *SpatialPolygons* is very tedious and hardly done in practice. Often the relevant objects are created by querying databases or other spatial objects such as extracting the information from the *maps* or *mapdata* packages, for example.

R-Code 5.16: Handling spatial objects in R.

```
### First example (simple version):
library(maps)          # simple database
ncMaps <- map("county", region="North Carolina", fill=TRUE, plot=FALSE)
###   "fill=TRUE" is very important! We need regular polygons.
str(ncMaps, strict.width="cut") # no formal class, just a particular list.

## List of 4
##  $ x     : num [1:3771] -79.5 -79.5 -79.5 -79.5 -79.5 ...
##  $ y     : num [1:3771] 35.8 35.9 35.9 36 36.2 ...
##  $ range: num [1:4] -84.3 -75.5 33.9 36.6
##  $ names: chr [1:102] "north carolina,alamance" "north carolina,alexand"..
##  - attr(*, "class")= chr "map"

### 100 Counties, "currituck" consisting of 3 polygons. Polygons are separated
###   with separated NAs.
sum(is.na(ncMaps$x))

## [1] 101

### Plotting is done with "plot=TRUE" (default) in the function `map()` and
###   color specification. Alternatively:
plot(ncMaps$x, ncMaps$y, type="n")
polygon(ncMaps$x, ncMaps$y, col=sample(1:16))
```

Moreover, the entire framework has been shifted more towards 'simple features' provided by the package *sf*. The mitigation is not entirely completed yet but it is generally recommended. See also Figure 5.10.



Differences between sp and sf

	sp	sf
Spatial data representation	Own representation	Simple Features standard (used by PostGIS)
Object system	S4 class	S3 class
Foundational class	Spatial	sf
CRS representation	WKT2-2019 (ISO 2004)	WKT2-2019 (ISO 2004)
Integration with the tidyverse	not supported	supported with several dplyr methods and ggplot2 through geom_sf()
Integration with tmap	supported	supported

Sources:

- Jesse Sadler, 2018, Introduction to GIS with R: Spatial data with the sp and sf packages
- Edzer Pebesma, 2021, sf Simple Features for R: package vignettes
- Roger Bivand, 2021, ECS530: Spatial data analysis II - Modernising PROJ and issues

RLadies

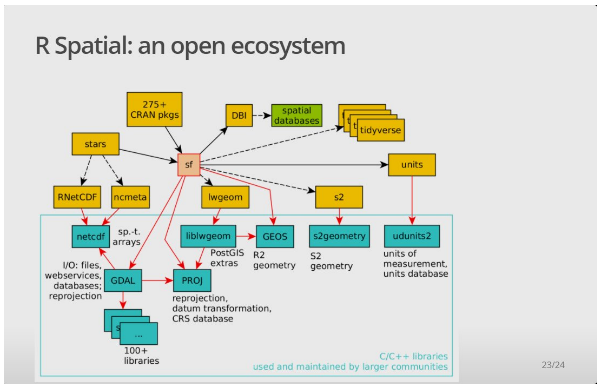

R Spatial: an open ecosystem
275+ CRAN pkgs
DBI
spatial databases
tidyverse
stars
sf
units
RNetCDF
ncmeta
lwgeom
s2
netcdf
sp.-t. arrays
liblwgeom
GEOS
s2geometry
udunits2
PostGIS extras
R2 geometry
S2 geometry
units of measurement, units database
I/O: files, webservices, databases; reprojection
GDAL
PROJ
reprojection, datum transformation, CRS database
...
100+ libraries
C/C++ libraries used and maintained by larger communities
23/24

Figure 5.10: Similarities and differences between the packages *sp* and *sf* (source left) and the R spatial ecosystem (source right).

Chapter 6

Hierarchical Models: Bayesian Modeling

In statistics, there exist two different philosophical approaches to inference: frequentist and Bayesian inference. Here, we introduce the Bayesian approach, where we consider the parameter as a random variable with a suitable distribution chosen a priori, i.e., before the data is collected and analyzed. The goal is to update this prior knowledge after observation of the data in order to conclude.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter06.R.

Chapters 4 and 5 introduced a family of spatial models for areal data (lattice data). Similarly as in the chapters about time series, much more emphasis has been given to modeling the second-order structure (the spatial dependency), whereas the mean has not been addressed with sufficient details. We will move towards more realistic models in the "Hierarchical Models" chapters.

In order to get there, we have to take several steps. More specifically, hierarchical models are typically tackled through a Bayesian modeling framework (this chapter). The following two chapters introduce two general hierarchical models and approaches to "fit" these.

6.1 Introduction and Simple Example

Example 6.1. Suppose that the CO_2 measurements are observed with some imprecision, i.e., some (measurement) error. We typically denote the "true" but unobserved quantity as the state or latent variable. This idea leads to a decomposition along the lines

$$Y_t = \mu_t + \varepsilon_t, \tag{6.1}$$

$$\mu_t = \boldsymbol{x}_t^\top \boldsymbol{\beta}. \tag{6.2}$$

The last two equations represent a very simple form of hierarchical modeling. At this point, the advantage of decomposing the model in two equations is not clear yet. ♣

Let us assume we have observations with measurement error and some knowledge about the structure of the observations. We then observe data and update our knowledge. In terms of conditional probabilities, we start from

$$\left.\begin{array}{c}[\text{ parameters }]\\ \text{model for the data}\end{array}\right\} \Rightarrow [\text{ parameters } \mid \text{ data }], \tag{6.3}$$

where the $[\cdot]$ and $[\cdot \mid \cdot]$ notation indicates the density of the argument and the conditional density of the first argument given the second, respectively. The model for the data is typically written as $[\text{ data} \mid \text{parameters }]$ thus

$$\left.\begin{array}{c}[\text{ parameters }]\\ [\text{ data } \mid \text{ parameters }]\end{array}\right\} \Rightarrow [\text{ parameters } \mid \text{ data }]. \tag{6.4}$$

Based on classical probability, we have

$$[\text{ parameters } \mid \text{ data }] = \frac{[\text{ data } \mid \text{ parameters }]\cdot[\text{ parameters }]}{[\text{ data }]}, \tag{6.5}$$

which can be seen as the first example of a hierarchical model.

In recent years, Bayesian (hierarchical) modeling approaches flourished due to the flexible modeling approach and the ability to simulate large samples from complicated posterior densities. The basic idea behind a hierarchical Bayesian approach is to break the (statistical) model down into different sub-models. These sub-models are typically a model for the data (as in equation (6.1)), a model for the process (as in equation (6.2)), and priors for the parameters. Equation (6.5) links the former two through Bayes' theorem. Gamerman and Lopes (2006) give a concise and accessible overview.

We now see two more concepts of (Bayesian) hierarchical modeling. Given the context here, it is not surprising that GMRFs have a recurrence.

6.1.1 Latent GMRF Modeling

We assume that the observations $\boldsymbol{y}$ are conditionally independent given latent parameters $\boldsymbol{\eta}$ and additional parameters $\boldsymbol{\theta}_{\boldsymbol{y}}$

$$\pi(\boldsymbol{y} \mid \boldsymbol{\eta}, \boldsymbol{\theta}_{\boldsymbol{y}}) = \prod_{i=1}^{n} \pi(y_i \mid \eta_i, \boldsymbol{\theta}_{\boldsymbol{y}}), \tag{6.6}$$

where $\pi(\cdot \mid \cdot)$ denotes the conditional density of the the first argument given the second argument.

For example, $\boldsymbol{\eta}$ may represent $\mathbf{X}\boldsymbol{\beta}$ in Example 6.1 or SIDS rates that we observe with some noise. As $\boldsymbol{\eta}$ is not observed, we talk of a *latent* parameter/variable/field here.

For computational reasons, the latent parameters $\boldsymbol{\eta}$ are often part of a larger latent random field $\boldsymbol{x}$, which is modeled as a GMRF with mean $\boldsymbol{\mu}$ and precision matrix $\mathbf{Q}$, both depending on parameters $\boldsymbol{\theta}_{\boldsymbol{x}}$; that is,

$$\pi(\boldsymbol{x} \mid \boldsymbol{\theta}_{\boldsymbol{x}}) \propto \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{x}-\boldsymbol{\mu})\Big). \tag{6.7}$$

This approach has been successfully implemented using iteratively nested Laplace approximations (INLA) by Rue *et al.* (2009) and subsequent work. We come back to such examples in upcoming chapters.

6.1.2 General State-Space Modeling Formulation

A particular modeling aspect is the so-called state-space approach, where we have an underlying (unobserved or latent) process described by the state equation. The state is then observed through some measurement operator with a measurement error. For example, biodiversity (state) is observed through remotely sensed normalized difference vegetation index (NDVI) (observations). More precisely, suppose that S_i is the random variable for the state process at grid node (area) j ($j = 1, \dots, J$), for example, modeled by $S_j = \boldsymbol{x}_j^\top \boldsymbol{\beta} + Y_j$, where Y_j is a zero-mean GMRF. Then an observation (at an arbitrary location) i is modeled by

$$Z_i = \mathcal{H}_i\big(S_1, \dots, S_I\big) + \varepsilon_i, \tag{6.8}$$

where $\mathcal{H}_i$ is the measurement operator, mapping the state space to the observation space and ε_i the measurement error.

This modeling approach is handy when the state equation is described by some dynamic process or modeled on a high-resolution grid. The measurement operator is often linear, for example, a bilinear interpolator between the grid points of the state equation. The model can be extended naturally to incorporate several different measurement processes, each observed at different locations with different precision. The measurement operator also allows to incorporate change of support; for example, one measurement is based (virtually) on a point source, the other on some aggregated measure.

6.2 Bayes Terminology and Examples

In statistics, two different philosophical approaches exist: frequentist and Bayesian inference. In this section, we give a very brief introduction to the Bayesian paradigm. We now consider the parameter as a random variable with a suitable distribution, chosen a priori, i.e., before the data is collected and analyzed. The goal is to update this prior knowledge after observation of the data in order to draw conclusions (with the help of the so-called posterior distribution).

Bayesian statistics is often introduced by recalling the so-called *Bayes theorem*, which states for two events A and B

$$\mathrm{P}(A \mid B) = \frac{\mathrm{P}(B \mid A)\ \mathrm{P}(A)}{\mathrm{P}(B)}, \qquad \text{for } \mathrm{P}(B) \neq 0, \tag{6.9}$$

and is shown by using twice $\mathrm{P}(A \mid B) = \mathrm{P}(A \cap B)/\,\mathrm{P}(B)$. Bayes theorem is often used in probability theory to calculate probabilities along an event tree, as illustrated in the arch-example below.

Example 6.2. A patient sees a doctor and gets a test for a (relatively) rare disease. The prevalence of this disease is 0.5%. As typical, the screening test is not perfect and has a sensitivity of 99%, i.e., true positive rate; correctly identified the disease in a sick patient, and a specificity of 98%, i.e., true negative rate; a healthy person is correctly identified disease free. What is the probability that the patient has the disease, provided the test is positive?

Denoting the events D = ‘Patient has disease’ and $+$ = ‘test is positive’ we have using Equation (6.9)

$$\begin{aligned} \mathrm{P}(D \mid +) &= \frac{\mathrm{P}(+ \mid D)\ \mathrm{P}(D)}{\mathrm{P}(+)} = \frac{\mathrm{P}(+ \mid D)\ \mathrm{P}(D)}{\mathrm{P}(+ \mid D)\ \mathrm{P}(D) + \mathrm{P}(+ \mid D^c)\ \mathrm{P}(D^c)} \\ &= \frac{99\% \cdot 0.5\%}{99\% \cdot 0.5\% + 2\% \cdot 99.5\%} = 20\%. \end{aligned} \tag{6.10}$$

Note that for the denominator, we have used the so-called *law of total probability* to get an expression for $\mathrm{P}(+)$. ♣

An interpretation of the previous example from a frequentist view is in terms of the proportion of outcomes (in a repeated sampling framework). In the Bayesian approach, we view the probabilities as “degree of belief”, where we have some proposition (event D in Example 6.2) and new evidence (event $+$ in Example 6.2). More specifically, $\mathrm{P}(D)$ represents the prior believe of our proposition, $\mathrm{P}(+ \mid D)/\,\mathrm{P}(+)$ is the support of the evidence for the proposition and $\mathrm{P}(D \mid +)$ is the posterior believe of the proposition after having accounted for the new evidence $+$.

Extending Bayes’ theorem to the setting of two continuous random variables X and Y, we have

$$f_{X\mid Y=y}(x \mid y) = \frac{f_{Y\mid X=x}(y \mid x)\, f_X(x)}{f_Y(y)} \quad \text{for all } y \text{ s.t. } f_Y(y) > 0. \tag{6.11}$$

In the context of Bayesian inference, the random variable X will now be a parameter, typically of the distribution of Y:

$$f_{\Theta\mid Y=y}(\theta \mid y) = \frac{f_{Y\mid\Theta=\theta}(y \mid \theta)\, f_\Theta(\theta)}{f_Y(y)}, \quad \text{for all } y \text{ s.t. } f_Y(y) > 0. \tag{6.12}$$

Hence, current knowledge about the parameter is expressed by a probability distribution on the parameter: the prior distribution. The model for our observations is called the likelihood. We use our observed data to update the prior distribution and thus obtain the posterior distribution.

In the next section, we discuss examples where the parameter is the success probability of a trial and the mean in a normal distribution.

Notice that $\mathrm{P}(B)$ in (6.9), $\mathrm{P}(+)$ in (6.10), or $f_Y(y)$ in (6.11) and (6.12) serves as a normalizing constant, i.e., it is independent of A, D, x or the parameter θ, respectively. Thus, we often write the posterior without this normalizing constant

$$f_{\Theta\mid Y=y}(\theta \mid y) \propto f_{Y\mid\Theta=\theta}(y \mid \theta) \times f_\Theta(\theta), \tag{6.13}$$

(or in short form $f(\theta \mid y) \propto f(y \mid \theta) f(\theta)$ if the context is clear). The symbol “$\propto$” means “proportional to”. For simplicity, we will omit the additional constraint that $f(y) > 0$.

Finally, we can summarize the most important result in Bayesian inference as the posterior density is proportional to the likelihood multiplied by the prior density, i.e.,

$$\text{Posterior density} \ \propto \ \text{Likelihood} \ \times \ \text{Prior density} \tag{6.14}$$

In a nutshell, the advantages of using a Bayesian framework are:

- formal way to incorporate prior knowledge;
- intuitive interpretation of the posterior;
- much easier to model complex systems;
- no n-dependency of 'significance' and the p-value.

As nothing comes for free, there are also some disadvantages:

- more 'elements' have to be specified for a statistical model;
- in virtually all cases, a Bayesian approach is computationally more demanding.

Until recently, there were clear fronts between frequentists and Bayesians. Luckily these differences have vanished.

6.2.1 Bayesian Inference

We illustrate the concept of Bayesian inference with two typical examples that are tractable.

Example 6.3 (beta-binomial model)**.** Suppose we observe y successes (out of n), a frequentist setting assumes $Y \sim \mathcal{Bin}(n,p)$ and uses as estimate $\widehat{p} = y/n$. In the Bayesian framework, we assume the success probability p as a random variable and thus an associated distribution. We require the support of the associated density to be the interval $(0,1)$. One example is the uniform distribution $\mathcal{U}(0,1)$ or the so-called Beta distribution. The density of a Beta random variable is given by

$$f(p) = c \cdot p^{\alpha-1}(1-p)^{\beta-1}, \qquad p \in [0,1], \alpha > 0, \beta > 0, \tag{6.15}$$

with normalization constant c. We write $P \sim \mathcal{Beta}(\alpha,\beta)$. If we investigate the probability of observing heads with a "regular" coin, it is highly unlikely that $p < 0.1$ or $p > 0.9$. This additional knowledge about the parameter p would be reflected using a prior $P \sim \mathcal{Beta}(5,5)$, for example.

The posterior density is then proportional to

$$\propto \binom{n}{y} p^y (1-p)^{n-y} \times c \cdot p^{\alpha-1}(1-p)^{\beta-1} \tag{6.16}$$

$$\propto p^y p^{\alpha-1}(1-p)^{n-y}(1-p)^{\beta-1} = p^{y+\alpha-1}(1-p)^{n-y+\beta-1}, \tag{6.17}$$

which can be recognized as a beta distribution $\mathcal{Beta}(y+\alpha, n-y+\beta)$.

Figure 6.1 illustrates the case of $y = 10$, $n = 13$ with prior $\mathcal{Beta}(5,5)$. Posterior mode is now between the prior one (0.5) and the frequentist estimate $\widehat{p}$. The expected value of a beta distributed random variable $\mathcal{Beta}(\alpha,\beta)$ is $\alpha/(\alpha+\beta)$ (here, the prior distribution). The posterior expected value is thus

$$\mathrm{E}(P \mid Y = y) = \frac{y+\alpha}{n+\alpha+\beta}. \tag{6.18}$$

Specifically, the mean changed from 0.5 to $(10+5)/(13+5+5) \approx 0.65$. ♣

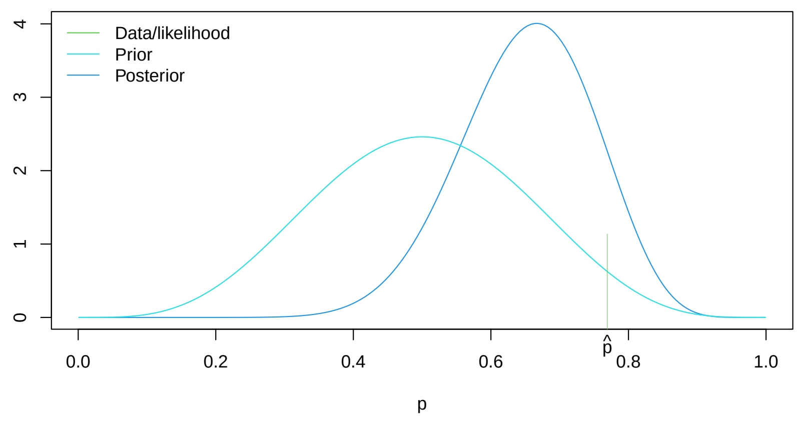


Figure 6.1: Beta-binomial model with prior density (cyan), data/likelihood (green), and posterior density (blue).

Table 6.1: Updates of point estimates for the beta-binomial model.

Prior	posterior
$\mathrm{E}(P) = \dfrac{\alpha}{\alpha+\beta}$	$\mathrm{E}(P \mid Y = y) = \dfrac{y+\alpha}{n+\alpha+\beta}$
$\mathrm{Mod}(P) = \dfrac{\alpha-1}{\alpha+\beta-2}$	$\mathrm{Mod}(P \mid Y = y) = \dfrac{y+\alpha-1}{n+\alpha+\beta-2}$
$\mathrm{Med}(P) \approx \dfrac{\alpha-1/3}{\alpha+\beta-2/3}$	$\mathrm{Med}(P \mid Y = y) \approx \dfrac{y+\alpha-1/3}{n+\alpha+\beta-2/3}$

The updates of specific point estimates for the beta-binomial model are given in Table 6.1.

In the previous example, we use $P \sim \mathcal{B}eta(\alpha, \beta)$ and fix α and β during model specification, thus called hyper-parameters.

The beta distribution $\mathcal{B}eta(1, 1)$, i.e., $\alpha = 1$, $\beta = 1$, is equivalent to a uniform distribution $\mathcal{U}(0, 1)$. However, the uniform distribution for the probability p does not mean "information-free". As a result of Equation (6.18), a uniform distribution as prior is "equivalent" to two experiments, of which one is a success. That means we can see the prior as two pseudo-observations.

In the following example, we have both continuous data and a parameter.

Example 6.4 (normal-normal model)**.** Let $Y_1, \dots, Y_n \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2)$. We assume σ is known. The mean μ is the only parameter of interest, for which we assume the prior $\mathcal{N}(\eta, \tau^2)$. Thus, we have the Bayesian model:

$$Y_i \mid \mu \overset{\text{iid}}{\sim} \mathcal{N}(\mu, \sigma^2), \qquad i = 1, \dots, n, \tag{6.19}$$

$$\mu \sim \mathcal{N}(\eta, \tau^2). \tag{6.20}$$

where σ^2, η and τ^2 are considered as hyper-parameters. Notice that we have again slightly abused the notation by using μ as the realization in (6.19) and as the random variable in (6.20). Since the context determines the meaning, we use this simplification for the parameters in the

Bayesian context. The posterior density is then

$$f(\mu \mid y_1, \dots, y_n) \propto f(y_1, \dots, y_n \mid \mu) \times f(\mu) = \prod_{i=1}^{n} f(y_i \mid \mu) \times f(\mu) \tag{6.21}$$

$$\propto \prod_{i=1}^{n} \exp\Big(-\frac{1}{2}\frac{(y_i-\mu)^2}{\sigma^2}\Big) \exp\Big(-\frac{1}{2}\frac{(\mu-\eta)^2}{\tau^2}\Big) \tag{6.22}$$

$$\propto \exp\Big(-\frac{1}{2}\sum_{i=1}^{n}\frac{(y_i-\mu)^2}{\sigma^2} - \frac{1}{2}\frac{(\mu-\eta)^2}{\tau^2}\Big), \tag{6.23}$$

where the constants $(2\pi\sigma^2)^{-1/2}$ and $(2\pi\tau^2)^{-1/2}$ do not need to be considered. Through further manipulation (of the square in μ), one obtains

$$\propto \exp\left(-\frac{1}{2}\left(\frac{n}{\sigma^2}+\frac{1}{\tau^2}\right)\left(\mu - \left(\frac{n}{\sigma^2}+\frac{1}{\tau^2}\right)^{-1}\left(\frac{n\bar{y}}{\sigma^2}+\frac{\eta}{\tau^2}\right)\right)^2\right) \tag{6.24}$$

and thus, the posterior distribution is

$$\mathcal{N}\left(\left(\frac{n}{\sigma^2}+\frac{1}{\tau^2}\right)^{-1}\left(\frac{n\bar{y}}{\sigma^2}+\frac{\eta}{\tau^2}\right), \left(\frac{n}{\sigma^2}+\frac{1}{\tau^2}\right)^{-1}\right). \tag{6.25}$$

In other words, the posterior expected value

$$\mathrm{E}(\mu \mid y_1, \dots, y_n) = \eta\,\frac{\sigma^2}{n\tau^2+\sigma^2} + \bar{y}\,\frac{n\tau^2}{n\tau^2+\sigma^2} = \eta\,\omega + \bar{y}(1-\omega) \tag{6.26}$$

is a weighted mean of the prior mean η and the mean of the likelihood $\bar{y}$. The weights $\omega = \sigma^2/(n\tau^2+\sigma^2)$ depend on the variance parameters and on n. With a smaller prior variance τ, more weight is given to the prior, for example. The larger n is, the less weight there is on the prior mean, since $\sigma^2/(n\tau^2+\sigma^2) \to 0$ for $n \to \infty$. Typically, the prior is fixed, but if more data is collected, the posterior mean will be closer to the mean of the data, and the prior has a weaker "influence" on the posterior. Figure 6.2 (based on R-Code 6.1) illustrates the setting of this example with $\bar{y} = 2.1$, $n = 4$ and the hyper-parameters $\sigma^2 = 1$, $\eta = 0$ and $\tau^2 = 2$. Here, the likelihood is with respect to $\overline{Y}$, i.e., the likelihood is a function of the parameter μ, given by the density of $\overline{Y}$, a Gaussian with mean $\bar{y}$ and variance σ^2/n. ♣

The posterior mode is often used as a summary statistic of the posterior distribution. Naturally, the posterior median and posterior mean (i.e., the expectation of the posterior distribution) are intuitive alternatives. In the case of the previous example, the posterior mode is the same as the posterior mean.

6.2.2 Bayesian Confidence intervals

Interval estimation in the frequentist approach results in confidence intervals. But sample confidence intervals need to be interpreted carefully in the context of repeated sampling. A sample $(1-\alpha)\%$ confidence interval $[b_u, b_o]$ contains the true parameter with a frequency of $(1-\alpha)\%$ in infinite repetitions of the experiment. With a Bayesian approach, we can now make statements about the parameter with probabilities. In Example 6.4, based on Equation (6.25)

$$\mathrm{P}\left(v^{-1}m - z_{1-\alpha/2}v^{-1/2} \le \mu \le v^{-1}m + z_{1-\alpha/2}v^{-1/2}\right) = 1-\alpha, \tag{6.27}$$

R-**Code 6.1** Normal-normal model. (See Figure 6.2.)

```r
## Information about data:
ybar <- 2.1;   n <- 4;   sigma2 <- 1
## information about prior:
priormean <- 0;   priorvar <- 2
## Calculating the posterior variance and mean:
postvar <- 1/( n/sigma2 + 1/priorvar)
postmean <- postvar*( ybar*n/sigma2 + priormean/priorvar )
## Plotting follows:
y <- seq(-2, to=4, length=500)
plot( y, dnorm( y, postmean, sqrt( postvar)), type="l", col=4,
    ylab="Density", xlab=bquote(mu))
lines( y, dnorm( y, ybar, sqrt( sigma2/n)), col=3)
lines( y, dnorm( y, priormean, sqrt( priorvar)), col=5)
legend( "topleft", legend=c("Data/likelihood", "Prior", "Posterior"),
    col=c(3, 5, 4), bty="n", lty=1)
```

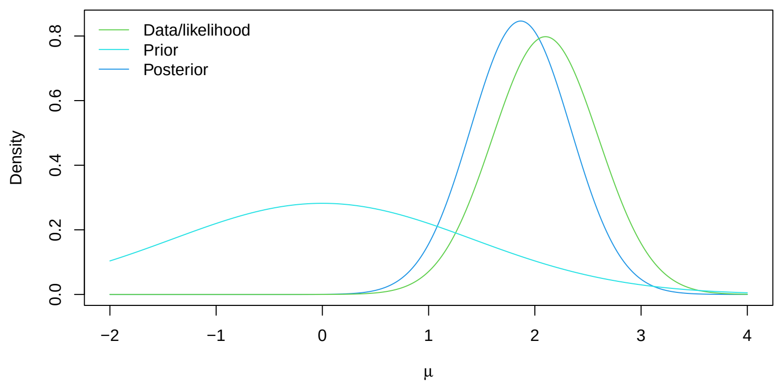


Figure 6.2: Normal-normal model with prior (cyan), data/likelihood (green) , and posterior (blue). (See R-Code 6.1.)

with $v = n/\sigma^2 + 1/\tau^2$ and $m = n\bar{y}/\sigma^2 + \eta/\tau^2$. That means that the bounds $v^{-1}m \pm z_{1-\alpha/2}v^{-1/2}$ can be used to construct a Bayesian counterpart to a confidence interval.

Definition 6.1. The interval R with

$$\int_R f(\theta \mid y_1, \dots, y_n)\, \mathrm{d}\theta = 1 - \alpha \tag{6.28}$$

is called a $(1-\alpha)\%$ credible interval for θ with respect to the posterior density $f(\theta \mid y_1, \dots, y_n)$ and $1-\alpha$ is the credible level of the interval. ♢

The definition states that the parameter θ, now seen as a random variable whose posterior density is given by $f(\theta \mid y_1, \dots, y_n)$ is contained in the $(1-\alpha)\%$ credible interval with probability $(1-\alpha)$.

Example 6.5 (continuation of Example 6.4)**.** The interval $[\,0.94, 2.79\,]$ is a 95% credible interval for the parameter μ. ♣

Since the credible interval for a fixed α is not unique, the "narrowest" is often used. This interval is the so-called HPD interval (highest posterior density interval). HPD intervals and credible intervals, in general, are often determined numerically.

Example 6.6 (continuation of Example 6.3)**.** The 2.5% and 97.5% quantiles of the posterior (6.17) are 0.45 and 0.83, respectively. A HPD is given by the bounds 0.46 and 0.84. The differences are not pronounced, as the posterior density is relatively symmetric. Hence, the widths of both are almost identical: 0.377 and 0.375.

The Wilson frequentist sample 95% CI is $[0.5, 0.92]$, with width 0.42. ♣

6.2.3 Predictive Distribution

In the classical regression framework, the estimated regression line represented the mean of an unobserved new location. To fully assess the uncertainty of the prediction, we had to take into account the uncertainty of the estimates and argued that the prediction is given by a t-distribution.

In the Bayesian setting, the likelihood $f(y_\text{new} \mid \theta)$ can be seen as the predictive distribution's density. That means the distribution of an unobserved new observation y_new. As the classical regression framework, using $f(y_\text{new} \mid \widehat{\theta}\,)$, with $\widehat{\theta}$ some Bayesian estimate of the parameter (e.g., posterior mean or posterior mode). The better approach is based on the posterior predictive distribution, defined as follows.

Definition 6.2. The *posterior predictive distribution* of a Bayesian model with likelihood $f(y \mid \theta)$ and prior $f(\theta)$ is

$$f(y_\text{new} \mid y_1, \dots, y_n) = \int f(y_\text{new} \mid \theta) f(\theta \mid y_1, \dots, y_n)\, \mathsf{d}\theta. \qquad \diamondsuit \tag{6.29}$$

In the previous equation, $f(y_\text{new} \mid \theta, y_1, \dots, y_n)$ represents the likelihood, and thus there is no dependency on the data. Hence, $f(y_\text{new} \mid \theta, y_1, \dots, y_n) = f(y_\text{new} \mid \theta)$.

Example 6.7 (continuation of Example 6.3)**.** In the context of the beta-binomial model, the posterior predictive distribution is constructed based on the single observation y only

$$\begin{aligned} f(y_\text{new} \mid y) &= \int_0^1 f(y_\text{new} \mid p) \times f(p \mid y)\, \mathsf{d}p \\ &= \binom{n}{y_\text{new}} c \int_0^1 p^{y_\text{new}} (1-p)^{n-y_\text{new}} \times p^{y+\alpha-1} (1-p)^{n-y+\beta-1}\, \mathsf{d}p, \end{aligned} \tag{6.30}$$

where c is the normalizing constant for the posterior. The integral itself gives us the normalizing constant of a $\mathcal{Beta}(y_\text{new} + y + \alpha, 2n - y_\text{new} - y + \beta)$ distribution. We do not recognize this distribution per se. As an illustration, Figure 6.3 shows the posterior predictive distribution based on the observation $y = 10$ and prior $\mathcal{Beta}(5, 5)$. The prior implies that the posterior predictive distribution is much more centered compared to the likelihood with plugin parameter $\widehat{p} = 10/13$ (i.e., the binomial density $\mathcal{Bin}(13, 10/13)$). ♣

R-**Code 6.2** Predictive distribution with the beta-binomial model. (See Figure 6.3.)

```
library(LearnBayes)
n <- 13
y <- 0:n
pred.probs <- pbetap(c( 10+5, 13-10+5), n, y) # prior Beta(5,5)
plot(y, pred.probs, type="h", ylim=c(0,.27), col=2, ylab="")
lines( y+0.07, dbinom(y, size=n, prob=10/13), type="h")
legend("topleft", legend=c("Predictive posterior", "Likelihood plugin"),
    col=2:1, lty=1, bty="n")
```

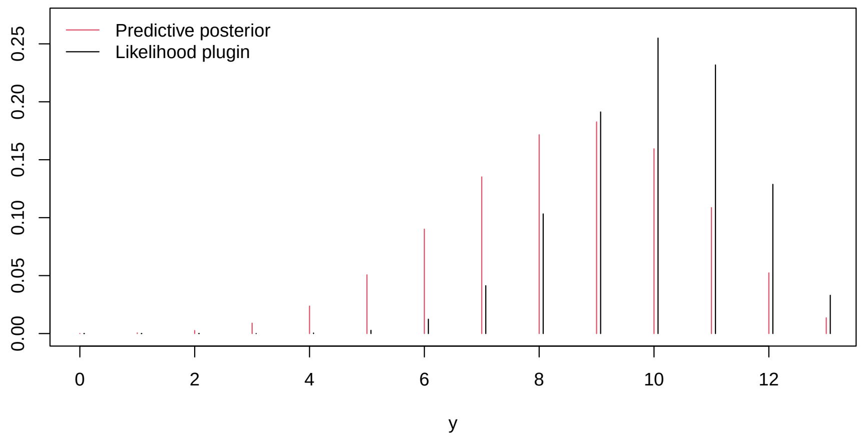


Figure 6.3: Predictive posterior distribution for the beta-binomial model (red) and likelihood with plugin parameter $\widehat{p} = 10/13$ (black).

Even in the simple case of the beta-binomial model, it is not straightforward to derive the predictive posterior distribution. Quite often, more detailed integration knowledge is required. In the case of the normal-normal model, as introduced in Example 6.4, it is possible to show that the posterior predictive distribution is again normal $\mathcal{N}(\mu_{\text{post}}, \sigma^2 + \sigma^2_{\text{post}})$, where μ_{post}, σ^2_{post} are the posterior mean and posterior variance as given in (6.25).

6.2.4 Bayes Factors

The Bayesian counterpart to hypothesis testing is done through a comparison of posterior probabilities. For example, consider two specific models specified by two hypotheses H_0 and H_1. By Bayes theorem,

$$\underbrace{\frac{\text{P}(H_0 \mid y_1, \dots, y_n)}{\text{P}(H_1 \mid y_1, \dots, y_n)}}_{\text{Posterior odds}} = \underbrace{\frac{\text{P}(y_1, \dots, y_n \mid H_0)}{\text{P}(y_1, \dots, y_n \mid H_1)}}_{\text{Bayes factor } (\text{BF}_{01})} \times \underbrace{\frac{\text{P}(H_0)}{\text{P}(H_1)}}_{\text{Prior odds}}, \tag{6.31}$$

that means that the Bayes factor BF_{01} summarizes the evidence of the data for the hypothesis H_0 versus the hypothesis H_1. The Bayes factor is any positive number. However, it has to be mentioned that a Bayes factor needs to exceed 3 to talk about substantial evidence for H_0. For strong evidence, we typically require Bayes factors larger than 10. More precisely, Jeffreys (1983) differentiates

$$1 < \begin{array}{l}\text{barely worth}\\ \text{mentioning}\end{array} < 3 < \text{substantial} < 10 < \text{strong} < 30 < \begin{array}{l}\text{very}\\ \text{strong}\end{array} < 100 < \text{decisive}$$

For values smaller than one, we would favor H_1 and the situation is similar by inverting the ratio, as also illustrated in the following example.

Example 6.8. We consider the setup of Example 6.3 and compare the models with $p = 1/2$ and $p = 0.8$ when observing 10 successes among the 13 trials. To calculate the Bayes factor, we need to calculate $\text{P}(Y = 10 \mid p)$ for $p = 1/2$ and $p = 0.8$. Hence, the Bayes factor is

$$\text{BF}_{01} = \frac{\binom{13}{10}0.5^{10}(1-0.5)^3}{\binom{13}{10}0.8^{10}(1-0.2)^3} = \frac{0.0349}{0.2457} = 0.1421, \tag{6.32}$$

which is somewhat substantial ($1/0.1421 \approx 7$) in favor of H_1. This result is not surprising, as the observed proportion is $\widehat{p} = 10/13 = 0.77$ close to $p = 0.8$ corresponding to H_1. ♣

In the example above, the hypotheses H_0 and H_1 are understood in the sense of $H_0 : \theta = \theta_0$ and $H_1 : \theta = \theta_1$. The situation for an unspecified alternative $H_1 : \theta \neq \theta_0$ is much more interesting and relies on using the prior $f(\theta)$ and integrating out the parameter θ:

$$f(y_1, \dots, y_n \mid H_1 : \theta \neq \theta_0) = \int f(y_1, \dots, y_n \mid \theta) f(\theta)\, \mathsf{d}\theta, \tag{6.33}$$

illustrated as follows.

Example 6.9 (continuation of Example 6.8)**.** For the situation $H_1 : p \neq 0.5$ using the prior $\mathcal{B}eta(5, 5)$, we have

$$\begin{aligned} \text{P}(Y = 13 \mid H_1) &= \int_0^1 \text{P}(Y = 13 \mid p) f(p)\, \mathsf{d}p \\ &= \int_0^1 \binom{13}{10} p^{10}(1-p)^3 \cdot c\, p^4(1-p)^4\, \mathsf{d}p = 0.0704, \end{aligned} \tag{6.34}$$

where we used `integrate( function(p) dbinom(10,13,prob=p)*dbeta(p, 5,5),0,1)`. Thus, $\text{BF}_{01} = 0.0349/0.0704 = 0.4957$. Hence, the Bayes factor is approximately 2 in favor of H_1, barely worth calculating the value. Under a uniform prior, the support for H_1 only marginally increases (from 2.017 to 2.046). ♣

Example 6.10. We now look at a Bayesian extension of the frequentist t-test. For simplicity, we assume the one sample setting without deriving the explicit formulas. The package `BayesFactor` provides functionality to calculate Bayes factors for different settings.

R-Code 6.3 shows that the Bayesfactor comparing the null model $\mu = 1$ against the alternative $\mu \neq 1$ is approximately 2.5. Here, we have used the standard parameter setting, which includes the prior and prior variance specifications. The prior variance can be specified with the argument `rscale` with default $0.707 = \sqrt{2}$. Increasing this variance leads to a flatter prior and, thus, to a smaller Bayes factor. Default priors are typically very reasonable, and we return to the priors' choice in the next section. ♣

R-Code 6.3 Bayes factor within the normal-normal model.

```
library(BayesFactor)
set.seed(12)
dat <- rnorm( 20, mean=1.4, sd=.5)
t.test( dat, mu=1)
##
##  One Sample t-test
##
## data:  dat
## t = 2.42, df = 19, p-value = 0.026
## alternative hypothesis: true mean is not equal to 1
## 95 percent confidence interval:
##  1.0315 1.4373
## sample estimates:
## mean of x
##    1.2344
ttestBF( dat, mu=1)
## Bayes factor analysis
## --------------
## [1] Alt., r=0.707 : 2.3525 ±0%
##
## Against denominator:
##   Null, mu = 1
## ---
## Bayes factor type: BFoneSample, JZS
```

Bayes factors are popular because they are linked to the BIC (Bayesian Information Criterion) and thus automatically penalize model complexity. Further, they also work for non-nested models.

6.3 Choice of the Prior Distribution

The choice of the prior distribution is part of the model specification, and a statistician should not be afraid of specifying priors. In many cases, the assumptions on the data (likelihood) are more restrictive and questionable than the choice of the prior. Naturally, the prior should be fixed before the data has been collected.

The examples in the last section were such that the posterior and prior distributions belonged to the same class. Naturally, that is no coincidence. In these examples, we have chosen so-called *conjugate prior* distributions.

In many cases, the posterior does not have a form from which we can sample directly, and sampling approaches are needed. Such routines will be discussed in Sections 7.1.2 and 7.1.3 and for more complicated settings in the subsequent two chapters.

Next to conjugate priors, there are many more classes that are typically discussed in a full Bayesian lecture. We prefer to classify the effect of the prior instead. Although not universal and quite ambiguous, we differentiate between *informative*, *weakly informative*, and *uninformative* priors. The first describes a prior that is specific to the data at hand, and with different data, the prior typically chances. The prior has potentially a substantial effect on the posterior. Weakly informative priors do not strongly influence the posterior but may substantially in situations when the model is ill-posed. Finally, an uninformative prior is such that the likelihood relates essentially to the posterior in terms of some criterion like posterior mean.

Uninformative priors are not classical prior distributions. In Example 6.3, we would require a "beta density" with $\alpha = \beta = 0$ in order to have a posterior mean that is equivalent to the likelihood estimate (see Equation (6.18)). However, for $\alpha = \beta = 0$ the normalizing constant of (6.15) is not finite as $\int_0^1 p^{-1}(1-p)^{-1}\,\mathsf{d}p$ diverges. Similarly, in Example 6.4, in order that $\mathrm{E}(\mu \mid y_1, \dots, y_n) = \bar{y}$, we need $\tau \to \infty$, that means, the prior of μ is "completely constant" (see Equation (6.26)). As we do not have a bounded range for μ, we have again not a "proper density", i.e., a so-called *improper prior*. Without going into details, it is possible that the posterior is a legitimate density for specific improper priors. However, improper priors should not be used unless typical examples are treated, as the posterior may not be proper either.

In the Bayesian literature, many articles are about soliciting specific priors (e.g., reference priors, Jeffrey's prior, ...). We advocate a purely pragmatic approach. If conjugate priors are available, use these. If not, choose priors that imply a more straightforward, stable implementation. Again, one should not be afraid of choosing informative priors.

For large n, the difference between a Bayesian and likelihood estimate is not pronounced. It is possible to show that the posterior mode converges to the likelihood estimate as the number of observations increases.

Example 6.11. We consider the normal-normal model again and compare the posterior density for various n with the likelihood. We keep $\bar{y} = 2.1$, independent of n. As shown in Figure 6.4, the maximum likelihood estimate does not depend on n ($\bar{y}$ is kept constant by design). The uncertainty decreases, however (standard error is $\sigma/\sqrt{n}$). For increasing n, the posterior "approach" the likelihood density. There is no difference between the posterior and the likelihood in the limit. The R-Code follows closely R-Code 6.1. ♣

6.4 Regression in a Bayesian Framework

In this section, we introduce a Bayesian approach to simple linear regression and logistic regression. More complex models are deferred to the next chapter. We will discuss the conceptual ideas and use software tools as a black-box approach. The underlying computational ideas will be discussed in the last chapter.

The simplest Bayesian regression model is as follows

$$Y_i \mid \boldsymbol{\beta}, \sigma^2 \overset{\text{indep}}{\sim} \mathcal{N}(\boldsymbol{x}_i\boldsymbol{\beta}, \sigma^2), \qquad i = 1, \dots, n, \tag{6.35}$$

$$\boldsymbol{\beta} \sim \mathcal{N}_{p+1}(\boldsymbol{\eta}, \sigma^2\mathbf{T}), \tag{6.36}$$

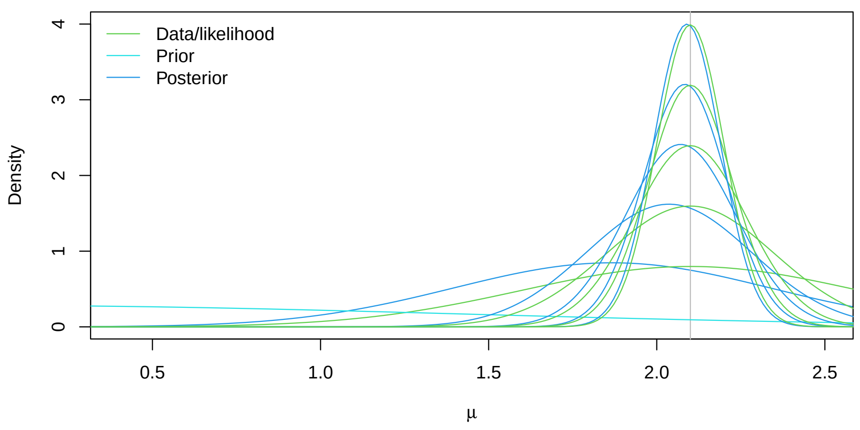


Figure 6.4: Normal-normal model with prior (cyan), data/likelihood (green), and posterior (blue) for increasing n ($n = 4, 36, 64, 100$). The prior is $\mathcal{N}(0, 2)$ and $\bar{y} = 2.1$ in all cases.

where σ^2, $\boldsymbol{\eta}$ and $\mathbf{T}$ are hyper-parameters. The model is quite similar to (6.19) and (6.20) with the exception that we have a multivariate prior distribution for $\boldsymbol{\beta}$. Instead of the parameter τ^2, we use $\sigma^2\mathbf{T}$, where $\mathbf{T}$ is a $(p+1)\times(p+1)$ symmetric positive definite matrix. The special form will allow us to factor σ^2 and simplify the posterior. With a few steps, it is possible to show that

$$\boldsymbol{\beta} \mid \boldsymbol{y} \sim \mathcal{N}_{p+1}(\mathbf{V}^{-1}\boldsymbol{m}, \sigma^2\mathbf{V}^{-1}) \tag{6.37}$$

with $\mathbf{V}^{-1} = \mathbf{T}^{-1} + \mathbf{X}^\top\mathbf{X}$ and $\boldsymbol{m} = \mathbf{T}^{-1}\boldsymbol{\eta} + \mathbf{X}^\top\boldsymbol{y}$.

It is possible to show that the posterior is a weighted average of the prior mean $\boldsymbol{\eta}$ and the classical least squares estimate $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}\boldsymbol{y}$.

The function `bayesglm()` from the R package `arm` implements an accessible way for simple linear regression and logistic regression. It is simple in the sense that it returns the posterior modes of the estimates in a framework that is similar to a frequentist approach. We need to specify the priors for the regression coefficients (separately for the intercept and the remaining coefficients).

Example 6.12 (Bayesian approach to `orings` data)**.** In January 1986, the space shuttle Challenger exploded shortly after taking off, killing all seven crew members aboard. Part of the problem was with the booster rockets' rubber seals, the so-called O-rings. Due to low ambient temperature, the seals started to leak, causing the catastrophe. The data set `data(orings, package="faraway")` contains the number of defects in the six seals in 23 previous launches (Figure 6.5). The question we ask here is whether the probability of a defect for an arbitrary seal can be predicted for an air temperature of 31°F (as in January 1986). See Dalal *et al.* (1989) for a detailed statistical account or simply https://en.wikipedia.org/wiki/Space_Shuttle_Challenger_disaster.

The variable of interest is a probability (failure of a rubber seal) that we estimate based on binomial data (failures of o-rings). However, a linear model cannot guarantee $\widehat{p}_i \in [0, 1]$. In this and similar cases, logistic regression is appropriate. The logistic regression models the probability of a defect as

$$p = \mathrm{P}(\text{defect}) = \frac{1}{1+\exp(-\beta_0 - \beta_1 x)}\,, \tag{6.38}$$

where x is the air temperature. Through inversion, one obtains a linear model for the log odds

$$g(p) = \log\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x, \tag{6.39}$$

where $g(\cdot)$ is generally called the link function. In this special case, the function $g^{-1}(\cdot)$ is called the logistic function. Such a model is a particular case of a *generalized linear model*.

R-Code 6.4 fits a logistic regression with the function `glm(..., family=binomial)`. The fitted and predicted values are shown in blue in Figure 6.5.

A Bayesian logistic model is fitted to the data in the second step. In the first `bayesglm()` call, we set the prior variances to infinity, resulting in uninformative priors. The posterior mode is identical to the result of a classical `glm()` model fit.

In the second call, we use Gaussian priors for both parameters with mean zero and variance 9. This choice is set by `prior.df=Inf` (i.e., a t-distribution with infinite degrees of freedom), by the default `prior.mean=0`, and by `prior.scale=3`, and similarly for the intercept parameter. The slope parameter is hardly affected by the prior. The intercept is, however: with its rather informative choice of the prior variance, the posterior mode is shrunk towards zero. Note that `summary(baye)` should not be used, as the printed p-values are irrelevant in the Bayesian context. The function `display()` is the preferred way. Interpreting a multivariate posterior distribution is not always straightforward, especially if the parameters are correlated. Figure 6.5 illustrates resulting fitted curves for which the parameters have been sampled from the posterior distribution (with `sim(bayes2)`). ♣

R-Code 6.4: `orings` data and estimated probability of defect dependent on air temperature. (See Figure 6.5.)

```
data( orings, package="faraway")
library(arm)
str(orings)
## 'data.frame': 23 obs. of  2 variables:
##  $ temp  : num  53 57 58 63 66 67 67 67 68 69 ...
##  $ damage: num  5 1 1 1 0 0 0 0 0 0 ...
plot( damage/6~temp, xlim=c(21,80), ylim=c(0,1), data=orings, pch="+",
     xlab="Temperature [F]", ylab="Probability of damage", cex=1.5) # data
abline( v=31, col="gray", lty=2)          # actual temp. at start

## frequentist approach:
glm1 <- glm( cbind(damage,6-damage)~temp, family=binomial, data=orings)
arm::display( glm1) # or similarly `summary( glm1)`
```

```
## glm(formula = cbind(damage, 6 - damage) ~ temp, family = binomial,
##     data = orings)
##             coef.est coef.se
## (Intercept) 11.66     3.30
## temp        -0.22     0.05
## ---
##   n = 23, k = 2
##   residual deviance = 16.9, null deviance = 38.9 (difference = 22.0)
```

```
points( orings$temp, glm1$fitted, col=4, pch=19, cex=1.5)  # fitted values
ct <- seq(20, to=85, length=100)                  # vector to predict
p.out <- predict( glm1, new=data.frame(temp=ct), type="response")
lines(ct, p.out, lwd=3, col=4)

## Bayesian approach:
bayes1 <- bayesglm( cbind(damage,6-damage)~temp, family=binomial, data=orings,
            prior.scale=Inf, prior.scale.for.intercept=Inf) #
coef(bayes1)   # result is "similar" to `coef(glm1)`
```

```
## (Intercept)        temp
##    11.66299    -0.21623
```

```
bayes2 <- bayesglm( cbind(damage,6-damage)~temp, family=binomial, data=orings,
            prior.df = Inf, prior.scale=3,
            prior.df.for.intercept=Inf, prior.scale.for.intercept=3) #
arm::display(bayes2)
```

```
## bayesglm(formula = cbind(damage, 6 - damage) ~ temp, family = binomial,
##     data = orings, prior.scale = 3, prior.df = Inf, prior.scale.for.intercept = 3,
##     prior.df.for.intercept = Inf)
##             coef.est coef.se
## (Intercept) 10.54     3.03
## temp        -0.20     0.05
## ---
## n = 23, k = 2
## residual deviance = 17.1, null deviance = 38.9 (difference = 21.8)
```

```
scoefs <- coef(sim(bayes2)) # simulation of coefficients...
for (i in 1:100) {
   lines( ct, invlogit( scoefs[i,1]+scoefs[i,2]*ct), col=rgb(.8,.8,.8,.2))
}
```

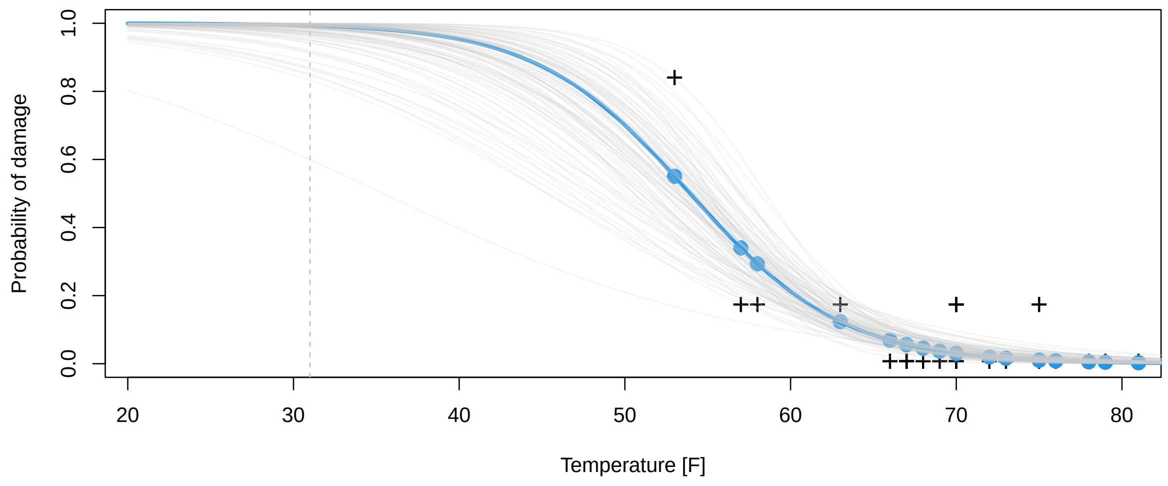


Figure 6.5: `orings` data (proportion of damaged orings, black crosses) and estimated probability of defect (blue dots) dependent on air temperature. The dotted vertical line is the ambient launch temperature at the time of launch. The blue line is the glm fit. Gray lines are based on draws from the posterior distribution. (See R-Code 6.4.)

6.5 Bibliographic Remarks

An accessible discussion of the Bayesian approach can be found in Held and Sabanés Bové (2014), including a discussion about the choice of prior distribution. A classic is Bernardo and Smith (1994).

The online open access book "An Introduction to Bayesian Thinking" available at `https://statswithr.github.io/book/` nicely melds theory and R source code.

Chapter 7

Hierarchical Models: Gibbs Sampling

In a Bayesian framework, the posterior distribution summarizes the update of the prior based on data. In realistic settings, the posterior distribution does not have a closed form, and inference is based on a sample thereof. This chapter discusses a straightforward approach to draw a sample from the posterior.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter07.R.

The (principal) goal of a hierarchical modeling approach is to obtain the posterior distribution of the process (and potentially of parameters) given the model observations and to derive relevant statistical quantities thereof. The posterior density can be derived via Bayes' theorem, synthesized as

$$\begin{aligned}&[\text{ process} \mid \text{data, parameters }]\\ &\qquad\propto [\text{ data} \mid \text{process, parameters }] \cdot [\text{ process} \mid \text{parameters }] \cdot [\text{ parameters }].\end{aligned} \tag{7.1}$$

The densities on the right-hand side of (7.1) are given by the three levels of the hierarchical model. The joint posterior is often a complicated distribution that has no closed form or from which it is impossible to draw directly. However, the posterior can be sampled using Markov chain Monte Carlo (MCMC) procedures, e.g., Geman and Geman (1984); Gelfand and Smith (1990). The essence of the MCMC approach is to simulate joint probability distributions by sampling from a Markov chain with a stationary (and ergodic) distribution that is identical to the posterior distribution (see also Gilks *et al.*, 1998; Robert and Casella, 1999).

7.1 General Remarks about Generating Random Numbers

There are many ways to sample from a distribution and examples are given in the following sections.

7.1.1 Ask R

Besides the classical distributions implemented in R, a plenitude of packages provides additional distributions. The CRAN task view 'Distributions' cran.r-project.org/web/views/Distributions.html gives a summary of the most important packages.

7.1.2 Transformation based on the inverse cdf

Let X be a random variable with CDF $F_X(x)$. If the inverse CDF F_X^{-1} has a closed form, then sampling from X can be done by sampling uniforms $U_1, \dots, U_n$ and transform it to $X_1 = F_X^{-1}(U_1), \dots, X_n = F_X^{-1}(U_n)$.

$$P\{X \leq x\} = P\{F_X^{-1}(U) \leq x\} = P\{U \leq F_X(x)\} = F_X(x). \tag{7.2}$$

There are many examples nice examples of transformation-based sampling, e.g., $X \sim \mathcal{E}xp(\lambda)$.

Remark 7.1. Interestingly, R uses a seemingly more complex algorithm to sample from $X \sim \mathcal{E}xp(\lambda)$. The algorithm is, however, fast and does not require a lot of memory (Ahrens and Dieter, 1972); properties that were historically very important. The actual code is in github.com/statslabs/rmath/blob/master/src/sexp.c.

7.1.3 Rejection sampling

Rejection sampling is a simple way to sample from a virtually arbitrary density $f_X(x)$. The idea is to sample a random variable with a known density $g_Y(y)$ and select suitable realizations.

More formally, we require $M < \infty$ such that $f_X(x) \leq M g_Y(x)$, for all x. We draw $\tilde{x}$ as the realization of Y and u from a standard uniform distribution. If $u \leq f_X(\tilde{x})/(M g_Y(\tilde{x}))$ we accept $\tilde{x}$ as a realization of X otherwise we reject $\tilde{x}$. Example 7.1 illustrates the rejection sampling using a straightforward example.

Naturally, the choice of g and M determines the sampler's efficiency.

Example 7.1. Using a uniform, we want to draw realizations from a $Beta(6, 3)$ distribution. More precisely, $f_X(x) = x^{6-1}(1-x)^{3-1}/\beta(6,3)$ and $g_Y(x) = 1_{0 \leq x \leq 1}(x)$. Choosing $M = 3$ satisfies $f_X(x) \leq M g_Y(x)$. R-Code 7.1 gives a straightforward implementation of the algorithm and a vectorized version. The resulting densities are shown in Figure 7.1. ♣

R-Code 7.1: Rejection sampling (See Figure 7.1.)

```
set.seed(16)
n.sim <- 1000
M <- 3
f <- function(x) x^(6-1) * (1-x)^(3-1) / beta(6,3)
g <- function(x) ifelse(x >= 0 & x <= 1, 1, 0)
result <- sample <- rep(NA, n.sim)
for (i in 1:n.sim){
  sample[i] <- runif(1)
  u <- runif(1)
  if(u <= f(sample[i]) / (M * g(sample[i]))) # if accepted ...
    result[i] <- sample[i]
}
mean(!is.na(result)) # proportion of accepted samples
## [1] 0.348
result <- result[!is.na(result)]

### Alternative implmentation:
sample <- runif(n.sim)
u <- runif(n.sim)
result <- sample[ u <= f(sample) / (M * g(sample)) ]

### Constructing the figures:
hist(sample, xlab="x", main="", col="lightblue")
hist(result, add=TRUE, col=4)
curve(dbeta(x, 6, 3), frame =FALSE, ylab="", yaxt="n")
lines(density(result), lty=2, col=4)
legend("topleft", legend=c("truth", "empirical"), lty=1:2, col=c(1,4))
```

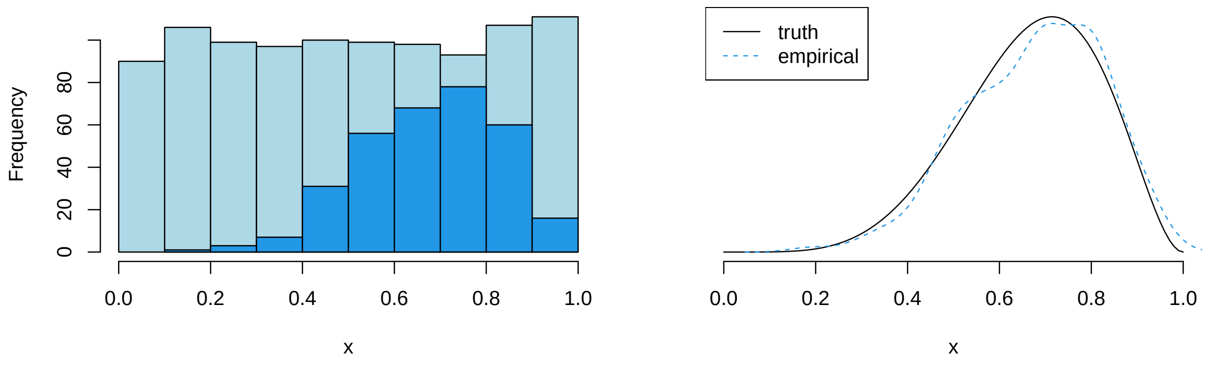


Figure 7.1: Left: histogram of simulated values from Y, representing $g_Y(x)$ (light blue) and from $f_X(x)$ (blue). The histograms are not scaled. Right: empirical and true density. (See R-Code 7.1.)

7.2 The Gibbs Sampler

A straightforward MCMC algorithm is the Gibbs sampler which aims to provide a sample from distributions that do not have closed-form quantile functions or ready-made R functions. Essentially, the Gibbs sampler works as follows in the general setting. First, for each parameter in the model, its distribution, conditional on all the other random quantities in the model, is identified. Such distributions are called *full conditionals* because only the parameter of interest is allowed to be random, and the remaining part of the model is fixed (or conditioned upon). Second, the Monte Carlo algorithm cycles among the parameters by simulating a new value for each parameter based on the full conditional distribution and the current values of the other parameters. Under weak assumptions the sequence converges to the intended distribution, i.e. the sample represents a valid realization of the distribution.

We now consider the Bayesian setting, where we would like to construct a sample from a posterior $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ where $\boldsymbol{\theta}$ is a p-valued parameter and $\boldsymbol{y}$ indicates the data vector. We first derive its p full conditional densities $\pi(\theta_i \mid \boldsymbol{\theta}_{-i}, \boldsymbol{y})$, $i = 1, \dots, p$, and in the second step we sample thereof. This procedure is particularly useful, if the full conditionals are such that we can directly sample thereof, i.e., if $\pi(\theta_i \mid \boldsymbol{\theta}_{-i}, \boldsymbol{y})$ has a ready built-in sampling procedure in R.

For the case $p = 3$, i.e., $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)^\top$, the algorithm to generate a sample of size M writes as follows (we write samples from iteration i with superscript $^{(i)}$):

(1) Set $\theta_2^{(0)}$ and $\theta_3^{(0)}$ to some admissible value and set $i = 1$.

(2) Sample $\theta_1^{(i)}$ from $\pi(\theta_1 \mid \theta_2^{(i-1)}, \theta_3^{(i-1)}, \boldsymbol{y})$;
sample $\theta_2^{(i)}$ from $\pi(\theta_2 \mid \theta_1^{(i)},\ \theta_3^{(i-1)}, \boldsymbol{y})$;
sample $\theta_3^{(i)}$ from $\pi(\theta_3 \mid \theta_1^{(i)},\ \theta_2^{(i)},\ \boldsymbol{y})$.

(3) If $i < M$, $i = i + 1$ and go to (2) else stop.

The sampler converges under very weak conditions (the full conditionals need to be valid distributions, and the support of $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$ can be written as a product set of the supports of $\pi(\theta_i \mid \boldsymbol{\theta}_{-i}, \boldsymbol{y})$, $i = 1, \dots, p$). For a more thorough discussion, we refer to Roberts and Smith (1994).

Example 7.2. Suppose we want to draw from a bivariate Gaussian distribution with zero mean and unit marginal variances, ρ known with no explicit dependence on $\boldsymbol{y}$, i.e., $\pi(\theta_1, \theta_2)$. Based on (1.19) the full conditionals $\pi(\theta_1 \mid \theta_2)$ and $\pi(\theta_2 \mid \theta_1)$ are

$$\theta_i \mid \theta_j \sim \mathcal{N}(\rho\theta_j, 1 - \rho^2), \qquad i \neq j. \tag{7.3}$$

R-Code 7.2 illustrates the updating. Figure 7.2 shows the sample and the first 10 draws by highlighting the consecutive updating of θ_i and θ_j. The empirical values are (naturally) quite close to the true values. The sample does exhibit a strong correlation, close to the actual value of ρ^2. Of course, we might draw all $2M = 2 \cdot 1000$ Gaussian realizations of variance $1 - \rho^2$ and just adjust the means. ♣

R-**Code 7.2:** Illustration of a Gibbs sampler for a bivariate Gaussian distribution. (See Figure 7.2.)

```
set.seed(19)
rho <- 0.9
M <- 1000
```

```
sam <- matrix(0, nrow=M, ncol=2)
for (i in 2:M) {
  sam[i, 1] <- rnorm(1, rho*sam[i-1, 2], sd=sqrt(1-rho^2))
  sam[i, 2] <- rnorm(1, rho*sam[i,   1], sd=sqrt(1-rho^2))
}
### Sampling done, lets visualize it:
plot(sam, xlab=bquote(theta[1]), ylab=bquote(theta[2]), pch=20, cex=.5)
points(sam[1,1], sam[1,2], col=2, pch=19, cex=1.5)
for (i in 1:10) {
  arrows(sam[i,1], sam[i,2], sam[i+1,1], sam[i,2], col=7, len=.1, lwd=2)
  arrows(sam[i+1,1], sam[i,2], sam[i+1,1], sam[i+1,2], col=8, len=.1, lwd=2)
} # can be written without loops by replacing `i' with `1:10'
### quick sanity check... qqplots etc should be added as well...
summary(sam)
##        V1                 V2
##  Min.   :-4.3314   Min.   :-3.8076
##  1st Qu.:-0.7256   1st Qu.:-0.7400
##  Median :-0.0423   Median :-0.0616
##  Mean   :-0.0551   Mean   :-0.0560
##  3rd Qu.: 0.6351   3rd Qu.: 0.6538
##  Max.   : 2.6988   Max.   : 3.3040
cov(sam)                # truth is [ [1, 0.9], [0.9, 1]]
##         [,1]    [,2]
## [1,] 0.99508 0.89426
## [2,] 0.89426 0.98498
cor(sam)[2]             # truth is 0.9
## [1] 0.90327
c(cor(sam[-1,1], sam[-M,1]), cor(sam[-1,2], sam[-M,2])) # will be .81
## [1] 0.81767 0.81174
```

For higher dimensional vectors $\boldsymbol{\theta} \in \mathbb{R}^p$, Step (2) in the Gibbs sampler algorithm is replaced by a loop over the components θ_k. That means

(2') For $k = 1, \dots, p$: sample $\theta_k^{(i)}$ from $\pi(\theta_k \mid \theta_1^{(i)}, \dots, \theta_{k-1}^{(i)},\ \theta_{k+1}^{(i-1)}, \dots, \theta_p^{(i-1)}, \boldsymbol{y})$.

Provided that starting values are correctly chosen, the sample $\boldsymbol{\theta}^{(1)}, \dots, \boldsymbol{\theta}^{(M)}$ is a legitimate sample from the joint distribution $\pi(\boldsymbol{\theta} \mid \boldsymbol{y})$. However, typically not the entire sample is kept: (i) every kth element is kept (termed *thinning*) to keep storage low, and (ii) the first $m < M$ elements of the sample are discarded (termed *eliminating burn-in*) to ensure that the sampler has "forgotten" the starting values and converged. Notice that thinning also reduces the sample's autocorrelation (serial correlation). For practical purposes, it is not required to have uncorrelated samples.

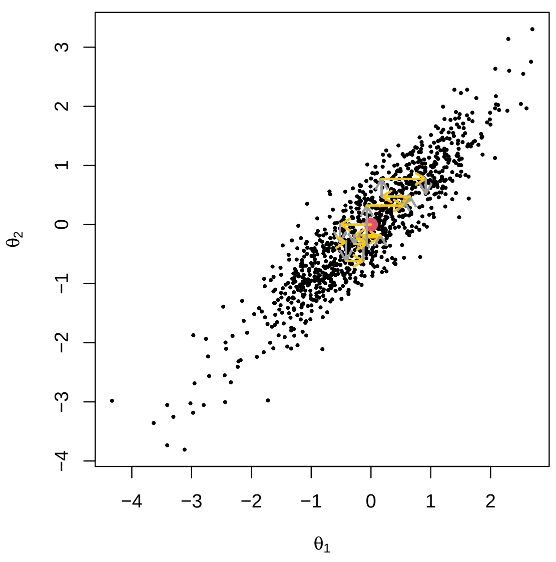


Figure 7.2: Illustration of a Gibbs sampler for a bivariate Gaussian distribution. The initial draw is indicated in red, and the next ten updates are drawn with arrows. (See R-Code 7.2.)

The illustration given in R-Code 7.2 intuitively extends to arbitrary multivariate Gaussian distributions (provided the mean vector and covariance matrix are known). However, for practically relevant cases, the theoretical justification of the Gibbs sampler is not trivial. As an intuitive justification, Brook's lemma can be used.

Lemma 7.1. *Under mild assumptions, we have*

$$\frac{\pi(\boldsymbol{x})}{\pi(\boldsymbol{x}')} = \prod_i \frac{\pi(x_i \mid x_1, \dots, x_{i-1}, x'_{i+1}, \dots x'_n)}{\pi(x'_i \mid x_1, \dots, x_{i-1}, x'_{i+1}, \dots x'_n)} \tag{7.4}$$

The lemma can be shown by starting with the equality

$$\pi(\boldsymbol{x}) = \frac{\pi(x_n \mid x_1, \dots, x_{n-1})}{\pi(x'_n \mid x_1, \dots, x_{n-1})} \pi(x_1, \dots, x_{n-1}, x'_n), \tag{7.5}$$

then recursively using

$$\pi(x_1, \dots, x_{n-1}, x'_n) = \frac{\pi(x_{n-1} \mid x_1, \dots, x_{n-1}, x'_n)}{\pi(x'_{n-1} \mid x_1, \dots, x_{n-1}, x'_n)} \pi(x_1, \dots, x_{n-2}, x'_{n-1}, x'_n). \tag{7.6}$$

For our intuitive justification, we take $\boldsymbol{x}'$ as the value of the previous draw.

7.3 Convergence Diagnostics

Every sample from an MCMC algorithm needs to be checked for *convergence*, i.e., does the sample indicate any evidence that convergence of the Markov chain has not (yet) reached.

One should always plot the (thinned) sample versus the indices, called the trace plots. These plots indicate how the samples fluctuate around the mean. Slow meandering is an indication of slow or poor mixing. Changing the initial value helps to assess how quickly a chain reaches its equilibrium. Auto-correlation plots (`acf()`-plots) assess the serial dependence in the sample. The parameters are correlated in many realistic model settings even when thinned is performed. It is worthwhile plotting scatter plots and cross-correlation plots as well. Irregular patterns or strong (asymmetric) dependencies should be investigated.

Comparing several long MCMC chains leads to analyzing a huge number of data points. To summarize the information, several traditional methods, such as summary tables, trace plots, Gelman plots, and Geweke plots, are helpful. Most of these functions are provided by the R package *`coda`* and are further illustrated in Section 8.2.

7.4 Example: Normal Response Model

This section illustrates a Gibbs sampler with a "realistic" example. It is clear that there are many technical details involved, but the emphasis should be put on the conceptual ideas behind.

We reconsider the R dataset *`UKDriverDeaths`* seen in Section 3.4.2, constituting a time series of the monthly totals of car drivers in Great Britain killed or seriously injured from January 1969 to December 1984 (i.e., 192 months or 16 years). The raw data is shown in the top panel of Figure 3.5. The following two sections illustrate a Gibbs sampler first for the annual and then for the monthly time series. Although the annually aggregated data is not particularly complex, it allows for presenting the framework first and adding additional modeling elements to the framework.

7.4.1 Annual Data

We assume that the square root of the monthly averages of cases per year is normal (hence the section title) and that these response variables are conditionally (on its mean) independent:

$$Y_i = t_i + \varepsilon_i, \qquad \varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \qquad i = 1, \dots, n. \tag{7.7}$$

Here, $n = 16$ and t_i indicate the mean. We will choose a non-parametric approach to model the smooth trend, much in the spirit of Section 4.5.1. When setting $\sigma^2 = 1/\kappa_{\boldsymbol{y}}$, the joint density of the responses can be written as

$$\pi(\boldsymbol{y} \mid \boldsymbol{t}, \kappa_{\boldsymbol{y}}) \propto \kappa_{\boldsymbol{y}}^{\frac{n}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{y}}}{2} \sum_{i=1}^{n} (y_i - t_i)^2 \Big). \tag{7.8}$$

This last equation defines the *likelihood.*

We now define the *process model*, which defines distributions for $t_1, \dots, t_n$. To model the smooth trend we assume that $t_i - 2t_{i+1} + t_{i+2}$, $i = 1, \dots, n-2$, are independent normal with mean zero and precision $\kappa_{\boldsymbol{t}}$ (an intrinsic second-order random walk model), as seen in Section 4.5.1. Hence, with $\boldsymbol{t} = (t_1, \dots, t_n)$

$$\pi(\boldsymbol{t} \mid \kappa_{\boldsymbol{t}}) \propto \kappa_{\boldsymbol{t}}^{\frac{n-2}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{t}}}{2} \boldsymbol{t}^\top \mathbf{Q}_{\boldsymbol{t}} \boldsymbol{t} \Big), \tag{7.9}$$

where the precision matrix $\mathbf{Q}_{\boldsymbol{t}}$ has non-zero entries on the five diagonals only. More specifically, a typical row is given by $(\dots, 0, 1, -4, 6, -4, 1, 0, \dots)$, see R-Code further below.

Finally, we need to specify the priors. We propose to use independent Gamma priors for the precisions, $\pi(\kappa_{\boldsymbol{t}}) \propto \kappa_{\boldsymbol{t}}^{\alpha_{\boldsymbol{t}}-1} \exp\big(-\kappa_{\boldsymbol{t}}\beta_{\boldsymbol{t}}\big)$ and $\pi(\kappa_{\boldsymbol{y}}) \propto \kappa_{\boldsymbol{y}}^{\alpha_{\boldsymbol{y}}-1} \exp\big(-\kappa_{\boldsymbol{y}}\beta_{\boldsymbol{y}}\big)$, being conjugate for the Gaussian components. The set $\alpha_{\boldsymbol{t}}, \beta_{\boldsymbol{t}}, \alpha_{\boldsymbol{y}}, \beta_{\boldsymbol{y}}$ constitute the hyper-parameters we need to specify as part of the Bayesian modeling approach. We choose $\alpha_{\boldsymbol{y}} = 4$, $\beta_{\boldsymbol{y}} = 4$, $\alpha_{\boldsymbol{t}} = 1$, $\beta_{\boldsymbol{t}} = 0.0005$; representing weakly informative priors. See also Remark 7.2 at the end of this section. (A Gamma distribution with scale and rate (or inverse scale) parameters α and β has an expectation α/β and variance α/β^2.)

Figure 7.3 illustrates the hierarchical model in a graph-like structure.

Graph $\mathcal{G}$

$$\begin{tikzcd}
\boldsymbol{y} \mid \boldsymbol{t}, & \sigma^2 \\
\boldsymbol{t} \mid \kappa_{\boldsymbol{t}} \arrow[u] & \\
\kappa_{\boldsymbol{t}} \mid \alpha_{\boldsymbol{t}}, \beta_{\boldsymbol{t}} \arrow[u] & \kappa_{\boldsymbol{y}} = \frac{1}{\sigma^2} \mid \alpha_{\boldsymbol{y}}, \beta_{\boldsymbol{y}} \arrow[uu]
\end{tikzcd}$$

Level	Distribution
likelihood	*iid*–Normal
process model	IGMRF (2nd order RW)
prior	Gamma

Figure 7.3: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

It is now possible to specify the full joint density

$$\pi(\boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\kappa}) = \pi(\boldsymbol{y} \mid \boldsymbol{t}, \kappa_{\boldsymbol{y}})\, \pi(\boldsymbol{t} \mid \kappa_{\boldsymbol{t}})\, \pi(\kappa_{\boldsymbol{t}})\, \pi(\kappa_{\boldsymbol{y}}). \tag{7.10}$$

We will now simplify this expression by collecting and arranging the terms in $\boldsymbol{t}$, $\boldsymbol{y}$ and the precision parameters. First, we write the (negative-2)-quadratic term of (7.18) as $\kappa_{\boldsymbol{y}}(\boldsymbol{y}-\boldsymbol{t})^\top(\boldsymbol{y}-\boldsymbol{t})$. Collecting all their terms (individual precisions parameters and the quadratic forms) leads to

$$\begin{aligned}\pi(\boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\kappa}) &\propto \kappa_{\boldsymbol{t}}^{\alpha_{\boldsymbol{t}}+\frac{n-2}{2}-1} \kappa_{\boldsymbol{y}}^{\alpha_{\boldsymbol{y}}+\frac{n}{2}-1} \exp\big(-\kappa_{\boldsymbol{t}}\beta_{\boldsymbol{t}} - \kappa_{\boldsymbol{y}}\beta_{\boldsymbol{y}}\big) \\ &\quad\times \exp\left(-\frac{1}{2}(\boldsymbol{t}^\top, \boldsymbol{y}^\top)\begin{pmatrix} \mathbf{Q}_{\boldsymbol{tt}} & \mathbf{Q}_{\boldsymbol{ty}} \\ \mathbf{Q}_{\boldsymbol{yt}} & \mathbf{Q}_{\boldsymbol{yy}} \end{pmatrix}\begin{pmatrix} \boldsymbol{t} \\ \boldsymbol{y} \end{pmatrix}\right),\end{aligned} \tag{7.11}$$

where the individual block precisions are, $\mathbf{Q}_{\boldsymbol{tt}} = \kappa_{\boldsymbol{t}}\mathbf{Q}_{\boldsymbol{t}} + \kappa_{\boldsymbol{y}}\mathbf{I}_n$, $\mathbf{Q}_{\boldsymbol{yy}} = \kappa_{\boldsymbol{y}}\mathbf{I}_n$, $\mathbf{Q}_{\boldsymbol{ty}} = \mathbf{Q}_{\boldsymbol{yt}} = -\kappa_{\boldsymbol{y}}\mathbf{I}_n$, with $\mathbf{Q}_{\boldsymbol{t}}$ defined above. The block precision $\mathbf{Q}_{\boldsymbol{tt}}$ contains two components, one from the prior and one from the likelihood. The block precision $\mathbf{Q}_{\boldsymbol{ty}}$ is due to the cross terms of the likelihood.

We further have

$$\pi(\boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y}) \propto \pi(\boldsymbol{y}, \boldsymbol{t}, \boldsymbol{\kappa}) \tag{7.12}$$

$$\pi(\boldsymbol{\kappa} \mid \boldsymbol{t}, \boldsymbol{y}) = \pi(\kappa_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{y})\pi(\kappa_{\boldsymbol{y}} \mid \boldsymbol{t}, \boldsymbol{y}). \tag{7.13}$$

More specifically,

$$\boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y} \sim \mathcal{N}_n\Big(\mathbf{Q}_{\boldsymbol{tt}}^{-1}\mathbf{Q}_{\boldsymbol{ty}}\boldsymbol{y}, \mathbf{Q}_{\boldsymbol{tt}}^{-1}\Big) \tag{7.14}$$

and

$$\pi(\kappa_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{y}) \propto \kappa_{\boldsymbol{t}}^{\alpha_{\boldsymbol{t}}+\frac{n-2}{2}-1} \exp\Big(-\kappa_{\boldsymbol{t}}\Big(\beta_{\boldsymbol{t}} + \frac{1}{2}\boldsymbol{t}^\top(\mathbf{Q}_{\boldsymbol{t}}\boldsymbol{t})\Big)\Big), \tag{7.15}$$

$$\pi(\kappa_{\boldsymbol{y}} \mid \boldsymbol{t}, \boldsymbol{y}) \propto \kappa_{\boldsymbol{y}}^{\alpha_{\boldsymbol{y}}+\frac{n}{2}-1} \exp\Big(-\kappa_{\boldsymbol{y}}\Big(\beta_{\boldsymbol{y}} + \frac{1}{2}(\boldsymbol{y}-\boldsymbol{t})^\top(\boldsymbol{y}-\boldsymbol{t})\Big)\Big). \tag{7.16}$$

Drawing from (7.14) can also be done with the R function *rmvnorm.canonical()* from the package *spam* with arguments $\mathbf{Q}_{\boldsymbol{ty}}\boldsymbol{y}$ and $\mathbf{Q}_{\boldsymbol{tt}}^{-1}$. Furthermore, drawing from (7.15) and (7.16) can be achieved in a single call to *rgamma()* with *shape* and *rate* arguments of length two.

It is now straightforward to implement a Gibbs sampler based on the full conditionals $\pi(\boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y})$, $\pi(\kappa_{\boldsymbol{t}} \mid \boldsymbol{t}, \boldsymbol{y})$ and $\pi(\kappa_{\boldsymbol{y}} \mid \boldsymbol{t}, \boldsymbol{y})$ as illustrated in R Code 7.3. The code essentially (i) loads the data and specifies the hyper-parameters of the prior for $\boldsymbol{\kappa} = (\kappa_{\boldsymbol{y}}, \kappa_{\boldsymbol{t}})^\top$ (ii) builds the precision matrices of equation (7.11) (based on unit precision, i.e., without κ (iii) sets up sampler specific variables, e.g., initialize the arrays containing the posterior samples and starting values for $\boldsymbol{\kappa}$, (iv) runs the Gibbs sampler.

R-**Code 7.3:** Manual implementation of a Gibbs sampler in the case of a normal response.

```
### (i) transforming the data and setting hyper-parameters:
library(spam)                         # UKDriverDeaths comes with package datasets

y <- rowMeans(matrix(sqrt(c(UKDriverDeaths)), 16, 12))
n <- length(y)                        # n=16 years
priorshape <- c(4, 1)                 # hyper-parameters for priors: alpha's,
priorrate <- c(4, 0.0005)             # beta's, some values

### (ii) constructing individual block precisions:
Qt <- precmat.RW2(n)
print(Qt[1:4,1:10])
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    1   -2    1    0    0    0    0    0    0     0
## [2,]   -2    5   -4    1    0    0    0    0    0     0
## [3,]    1   -4    6   -4    1    0    0    0    0     0
## [4,]    0    1   -4    6   -4    1    0    0    0     0
## Class 'spam' (32-bit)
In <- diag(n)                         # or equivalently diag.spam(n)

### (iii) setup sampler variables:
totalg <- 750                         # maybe too small in practice as well
```

```
set.seed(14)

tpost <- array(0, c(totalg, n))    # prepare arrays
kpost <- array(0, c(totalg, 2))
kpost[1,] <- c(1, 100)             # some starting values
postshape <- priorshape + c(n/2, (n-2)/2) # is fixed throughout the loop

### (iv) Gibbs loop:
for (i in 2:totalg) {              # start loop
  Qtt <- kpost[i-1,2]*Qt + kpost[i-1,1]*In       # pre-calculate precision
  tpost[i,] <- rmvnorm.canonical(1, c(kpost[i-1,1]*In %*% y), Qtt)   #
  postrate <- priorrate +          # prior contribution
      c(sum((y-tpost[i,])^2)/2,              # for kappa_y
      t(tpost[i,]) %*% (Qt %*% tpost[i,])/2) # for kappa_t
  kpost[i,] <- rgamma(2, postshape, postrate)     # update kappa
}                                  # end loop
kpost <- kpost[c(250:totalg),]     # eliminate burn-in.
tpost <- tpost[c(250:totalg),]
```

The loop takes a couple of seconds to run. R-Code 7.4 shows the post-processing. Then it is possible to look at the posterior density and posterior summaries, e.g., `plot(density(kpost[-1, 1]))`, `summary(kpost)`.

Figure 7.4 (top left) shows the trace plots of the log precisions. By eliminating the first few values we eliminate the apparent effect of the starting values. The trace plot of κ_t does not look good, there is further evidence in the autocorrelation plot of slow mixing. The precisions are not too strongly correlated.

The bottom left panel of Figure 7.4 shows the individual trends as gray lines, with the first ten posterior draws shown in black. (The first row of `tpost` is zero, corresponding to the initial values.) Additionally, the posterior median and 2.5% and 97.5% quantiles of the trend are given. Note that the latter are calculated pointwise with `quantile()` and do not correspond to actually observed trends.

R-Code 7.4: Postprocessing, summaries, some visualization of the sampler output, and predictive distribution of the observations. (See Figure 7.4.)

```
### Construct summary of precisions:
allkappas <- rbind(apply(kpost, 2, median), apply(1/kpost, 2, median))
colnames(allkappas) <- c("kappa_y", "kappa_t")
rownames(allkappas) <- c("Prec (median)", "Var (median)")
allkappas

##                 kappa_y    kappa_t
## Prec (median) 1.35277 1.0985e+03
## Var (median)  0.73922 9.1034e-04
### Construct posterior quantiles of mean terms:
postquant <- apply(tpost, 2, quantile,c(.025,.5,.975))

### Plotting:
matplot(log(kpost), lty=1, type="l", xlab="Sample index", col=c(1,3))
abline(h=log(allkappas[1,]), col=3)
acf(kpost[,2], ylab=expression(ACF~of~kappa[t]), main="")
plot(kpost, ylab=bquote(kappa[t]), xlab=bquote(kappa[y]), pch=20, cex=.5)
abline(h=allkappas[1,2], v=allkappas[1,1],col=3)

matplot(t(tpost), type="l", ylab="Trend", ylim=c(39,44), col="gray", lty=1)
matlines(t(tpost[2:11,]), col=1, lty=1 )           # highlight 10 observations
matlines(t(postquant), col=c(4,2,4), lty=1, lwd=2) # posterior quantiles
points(y, pch=20)                                  # add observations
legend("topright", bty="n", col=c(2,4,"gray"), legend=c("Posterior median",
    "Quantiles of posterior sample", "individual trends"), lty=1)

ypred <- rnorm(totalg*n, tpost, sd=rep(1/sqrt(kpost[,1]), n))
dim(ypred) <- c(totalg, n)
postpredquant <- apply(ypred, 2, quantile, c(.025,.975))

plot(y^2, ylab="Counts", pch=20, ylim=c(1400,2100)) # actual counts
matlines(t(postquant)^2, col=c(4,2,4), lty=1)       # quantiles of the trend
matlines(t(postpredquant)^2, col=3, lty=1)          # quantiles of pred dist
legend("topright", bty="n", col=c(2,4,3), lty=1, legend=c("Posterior median",
    "Quantiles of posterior sample", "Quantiles of predictive distribution"))
```

The predictive distribution of $\boldsymbol{y}$ is obtained by adding zero mean normal random variables with precision *`kpost[i,1]`* to the mean *`tpost[i,]`*. The bottom right panel of Figure 7.4 gives the posterior median, the posterior quantiles, and the quantiles of the predictive distribution, again calculated pointwise.

In this simple example, we could have also worked with total annual cases with similar conclusions. The monthly averages simplify the extension to individual monthly data as outlined in the next section.

Remark 7.2. The chosen priors may appear somewhat arbitrary. Here are some additional insights. The annual data example is a simplified case compared to the monthly data, which is discussed in Sections 4.2.1 of Rue and Held, 2005. To ensure better comparability, we aligned the values of the prior hyperparameters. In this example, the priors have a notable influence on the posterior, particularly due to the small sample size.

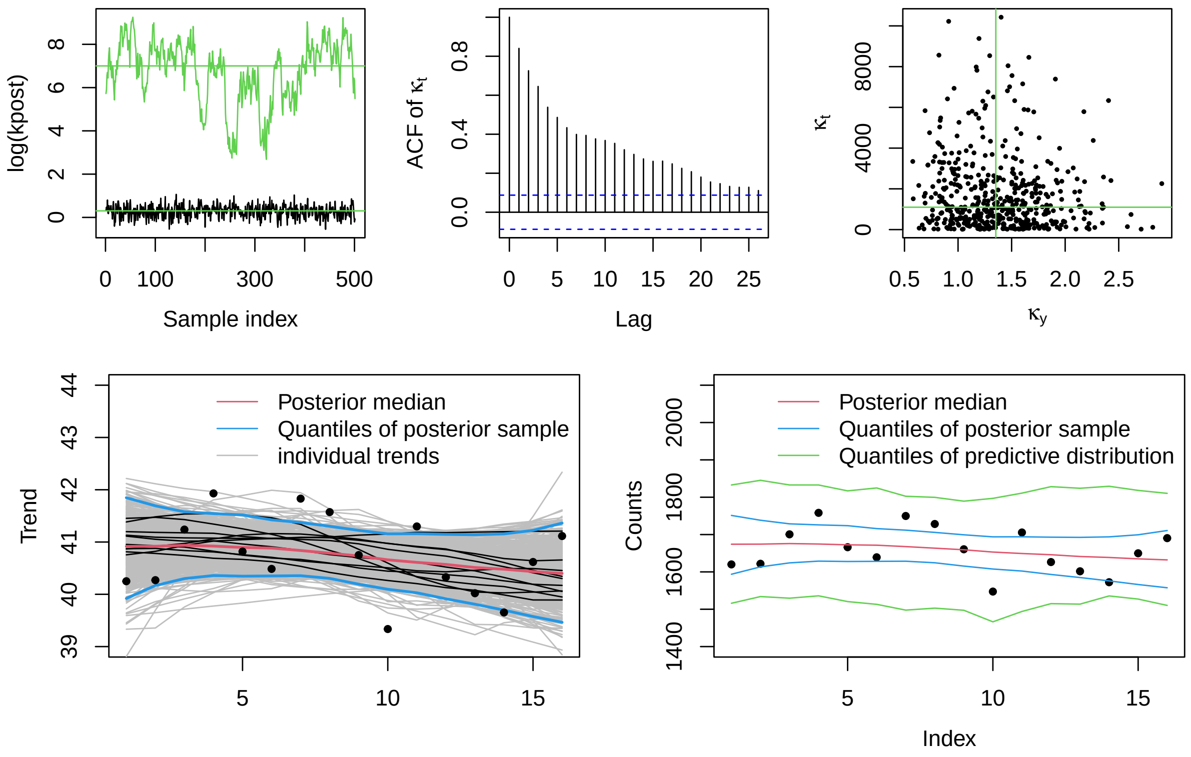


Figure 7.4: Top row: trace plots of the log precision, autocorrelation plot of κ_t and scatter plot of κ_t against κ_y. Solid green lines indicate posterior medians. Bottom left: posterior median and 2.5% and 97.5% quantiles of the trend. The individual trends are given in gray lines, the first ten in black. Bottom right: observed counts, the posterior median, its quantiles, and the quantiles of the predictive distribution. (See R-Code 7.4.)

R-Code 7.5 and Figure 7.5 illustrate this dependence. We define a set of different hyperparameters, run a Gibbs sampler (similar to R-Code 7.5, but retaining only the precision sample), and visualize the posterior densities in Figure 7.5. The prior of κ_θ has a stronger effect on the posterior compared to the prior of κ_γ. It is worth noting the posterior sample variability, as indicated by the two black densities in Figure 7.5 (one from R-Code 7.3 and one from here). ♡

R-Code 7.5: Visualizing the effect of the prior.

```
### Defining a set of hyperpriors
hyperseq <- cbind(shapey=c(4,1,1,1,1,1), ratey=c(4,1,1,1,1,.5),
                  shapet=1, ratet=c(0.0005, .005, .05, .5, .01, .01))
npss <- dim(hyperseq)[1]
mkpost <- array(0, c(npss, totalg, 2))        # prepare array
mkpost[,1,] <- rep(c(1, 100), each=npss)     # with some starting values

for (jj in 1:npss){                # loop over all hyperparameter configurations
  for (i in 2:totalg) {              # start Gipps loop
    Qtt <- mkpost[jj, i-1, 2]*Qt + mkpost[jj, i-1, 1]*In
    tpostvec <- c(rmvnorm.canonical(1, c(mkpost[jj,i-1,1]*In %*% y), Qtt) )
    mkpost[jj,i,] <- rgamma(2, shape=hyperseq[jj, c(1,3)] + c(n/2, (n-2)/2),
                       rate=hyperseq[jj, c(2,4)] + c(sum((y-tpostvec)^2)/2,
                                          t(tpostvec) %*% (Qt %*% tpostvec)/2 ))
}  }                                      # end both loop
mkpost <- mkpost[,250:totalg,]            # eliminate burn-in
### Below is only plotting:
out <- density(log(kpost[,1]), from=min(log(mkpost[,,1])),  # density estimate
               to=max(log(mkpost[,,1])))
plot(out$x, out$y, type="l", ylim=c(0, 1.6), ylab="", xlab="log(kappa_y)")
for (jj in 1:npss) {     # densities for posteriors and priors of kappa_y
  lines(density(log(mkpost[jj,,1])), col=jj)
  lines(out$x, dgamma(exp(out$x), hyperseq[jj,1], hyperseq[jj,2]),
        col=jj, lty=2)
}
legend("topleft", paste(hyperseq[,1],hyperseq[,2], sep=", "), bty="n",
       lty=1, col=1:npss)
out <- density(log(kpost[,2]), from=min(log(mkpost[,,2])),
               to=max(log(mkpost[,,2])))
plot(out$x, out$y, type="l", ylim=c(0,.9), ylab="", xlab="log(kappa_t)")
for (jj in 1:npss) {
  lines(density(log(mkpost[jj,,2])), col=jj)
  lines(out$x, dgamma(exp(out$x), hyperseq[jj,3], hyperseq[jj,4]),
        col=jj, lty=2)
}
legend("topleft", paste(hyperseq[,3],hyperseq[,4], sep=", "), bty="n",
       lty=1, col=1:npss)
```

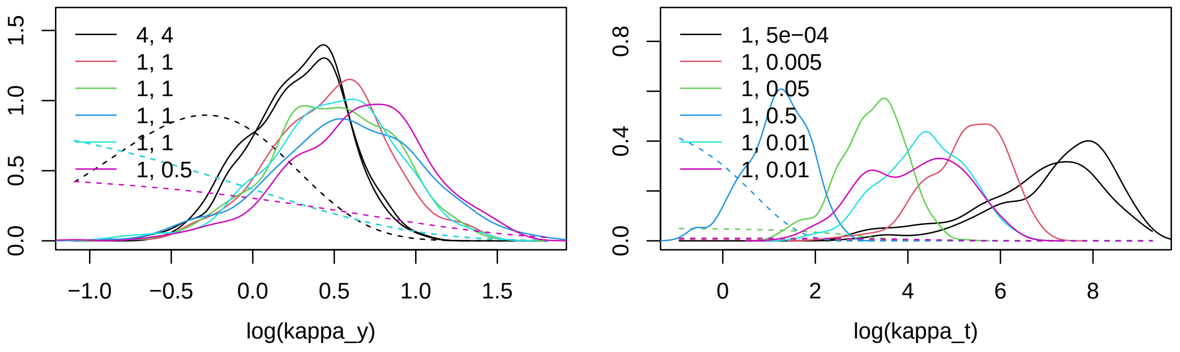


Figure 7.5: Prior (dashed) and posterior (solid) densities of the precision parameters κ_y (left) and κ_t (right) on log-scale. (See R-Code 7.5.)

7.4.2 Monthly data

We now look at monthly data and thus need to add a seasonal component. Additionally, we will extend the previous section with the following elements: (i) prediction of the trend for which we do not have observations; (ii) exploiting the sparsity structure of the precision matrix. Detailed source code is available by `demo("article-jss-example1", package="spam")`. The example is also discussed in Sections 4.2.1 of Rue and Held, 2005.

The series exhibits a strong seasonal component (denoted by s_i) and a smooth trend (denoted by t_i). Here, we want to predict the pattern $\eta_i = s_i + t_i$ for the additional $m = 12$ months. We assume again that the square root responses are normal and are conditionally (on η_i) independent:

$$Y_i + \eta_i + \varepsilon_i = s_i + t_i + \varepsilon_i, \qquad \varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2), \qquad i = 1, \dots, n, \tag{7.17}$$

which is, when setting $\sigma^2 = 1/\kappa_{\boldsymbol{y}}$, in a Bayesian nomenclature equivalent to

$$\pi(\boldsymbol{y} \mid \boldsymbol{\eta}, \kappa_{\boldsymbol{y}}) = \pi(\boldsymbol{y} \mid \boldsymbol{s}, \boldsymbol{t}, \kappa_{\boldsymbol{y}}) \propto \kappa_{\boldsymbol{y}}^{\frac{n}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{y}}}{2} \sum_{i=1}^{n} (y_i - t_i - s_i)^2 \Big). \tag{7.18}$$

This last equation defines the *likelihood*.

We now define the *process model* and foresee prediction beyond the n observations. Therefore, the process layer defines distributions for $\eta_1, \dots, \eta_{n+m}$. To model the seasonal component, we assume that $\sum_{j=0}^{11} s_{i+j}$, $i = 1, \dots, n+1$, are independent normal random variables with mean zero and precision $\kappa_{\boldsymbol{s}}$ (an intrinsic GMRF model for seasonal variation, e.g., Rue and Held, 2005, page 122). Similarly, to model the smooth trend we assume that $t_i - 2t_{i+1} + t_{i+2}$, $i = 1, \dots, n+m-2$, are independent normal with mean zero and precision $\kappa_{\boldsymbol{t}}$ (an intrinsic second-order random walk model). Both intrinsic GMRF models are direct extensions of what we have seen in Section 4.5.1. Hence, with $\boldsymbol{s} = (s_1, \dots, s_{n+m})$ $\boldsymbol{t} = (t_1, \dots, t_{n+m})$

$$\pi(\boldsymbol{s} \mid \kappa_{\boldsymbol{s}}) \propto \kappa_{\boldsymbol{s}}^{\frac{n+1}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{s}}}{2} \boldsymbol{s}^\top \mathbf{Q}_{\boldsymbol{s}} \boldsymbol{s} \Big), \tag{7.19}$$

$$\pi(\boldsymbol{t} \mid \kappa_{\boldsymbol{t}}) \propto \kappa_{\boldsymbol{t}}^{\frac{n+m-2}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{t}}}{2} \boldsymbol{t}^\top \mathbf{Q}_{\boldsymbol{t}} \boldsymbol{t} \Big), \tag{7.20}$$

where the precision matrices $\mathbf{Q}_{\boldsymbol{s}}$ and $\mathbf{Q}_{\boldsymbol{t}}$ are given by analogues of equations (3.59) and (3.40) of Rue and Held (2005). We do not need to explicate the precision matrices as these will be provided by calls to the functions `precmat.season()` and `precmat.RW2()` provided by the package `spam`.

Finally, we need to specify the priors. We propose to use independent Gamma priors for the three precisions, e.g., $\pi(\kappa_{\boldsymbol{s}}) \propto \kappa_{\boldsymbol{s}}^{\alpha_{\boldsymbol{s}}-1} \exp\big(-\kappa_{\boldsymbol{s}}\beta_{\boldsymbol{s}}\big)$, being conjugate for the Gaussian components. Figure 7.6 illustrates the hierarchical model in a graph-like structure.

It is now possible to specify the full joint density

$$\pi(\boldsymbol{y}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\kappa}) = \pi(\boldsymbol{y} \mid \boldsymbol{s}, \boldsymbol{t}, \kappa_{\boldsymbol{y}})\, \pi(\boldsymbol{s} \mid \kappa_{\boldsymbol{s}})\, \pi(\boldsymbol{t} \mid \kappa_{\boldsymbol{t}})\, \pi(\kappa_{\boldsymbol{s}})\, \pi(\kappa_{\boldsymbol{t}})\, \pi(\kappa_{\boldsymbol{y}}). \tag{7.21}$$

We will now simply this expression by collecting and arranging the terms in $\boldsymbol{s}$, $\boldsymbol{t}$, $\boldsymbol{y}$ and the precision parameters. First, we write the (negative-2)-quadratic term of (7.18) as $\kappa_{\boldsymbol{y}}(\boldsymbol{y} - \boldsymbol{s}_o - \boldsymbol{t}_o)^\top(\boldsymbol{y} - \boldsymbol{s}_o - \boldsymbol{t}_o)$, where the subscript "$_o$" indicates the truncated version of the vector, e.g.,

Graph $\mathcal{G}$

$$\begin{tikzcd}
\boldsymbol{y} \mid \boldsymbol{s}, & \boldsymbol{t}, & \sigma^2 \\
\boldsymbol{s} \mid \kappa_{\boldsymbol{s}} \arrow[u] & \boldsymbol{t} \mid \kappa_{\boldsymbol{t}} \arrow[u] & \\
\kappa_{\boldsymbol{s}} \mid \alpha_{\boldsymbol{s}}, \beta_{\boldsymbol{s}} \arrow[u] & \kappa_{\boldsymbol{t}} \mid \alpha_{\boldsymbol{t}}, \beta_{\boldsymbol{t}} \arrow[u] & \kappa_{\boldsymbol{y}} = \frac{1}{\sigma^2} \mid \alpha_{\boldsymbol{y}}, \beta_{\boldsymbol{y}} \arrow[uu]
\end{tikzcd}$$

Level	Distribution
likelihood	*iid*–Normal
process model	IGMRF (seasonal) IGMRF (2nd order RW)
prior	Gamma

Figure 7.6: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

$\boldsymbol{s}_o = (s_1, \dots, s_n)^\top$. With $\mathbf{D} = (\mathbf{I}_n, \mathbf{0})$ we have $\boldsymbol{s}_o = \mathbf{D}\boldsymbol{s}$ and similarly for $\boldsymbol{t}$. Collecting all their terms (individual precisions parameters and the quadratic forms) leads to

$$\pi(\boldsymbol{y}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\kappa}) \propto \kappa_{\boldsymbol{s}}^{\alpha_{\boldsymbol{s}} + \frac{n+1}{2} - 1} \kappa_{\boldsymbol{t}}^{\alpha_{\boldsymbol{t}} + \frac{n+m-2}{2} - 1} \kappa_{\boldsymbol{y}}^{\alpha_{\boldsymbol{y}} + \frac{n}{2} - 1} \exp\big(- \kappa_{\boldsymbol{s}}\beta_{\boldsymbol{s}} - \kappa_{\boldsymbol{t}}\beta_{\boldsymbol{t}} - \kappa_{\boldsymbol{y}}\beta_{\boldsymbol{y}} \big) \tag{7.22}$$

$$\times \exp\left(-\frac{1}{2}(\boldsymbol{s}^\top, \boldsymbol{t}^\top, \boldsymbol{y}^\top) \begin{pmatrix} \mathbf{Q}_{\boldsymbol{ss}} & \mathbf{Q}_{\boldsymbol{st}} & \mathbf{Q}_{\boldsymbol{sy}} \\ \mathbf{Q}_{\boldsymbol{ts}} & \mathbf{Q}_{\boldsymbol{tt}} & \mathbf{Q}_{\boldsymbol{ty}} \\ \mathbf{Q}_{\boldsymbol{ys}} & \mathbf{Q}_{\boldsymbol{yt}} & \mathbf{Q}_{\boldsymbol{yy}} \end{pmatrix} \begin{pmatrix} \boldsymbol{s} \\ \boldsymbol{t} \\ \boldsymbol{y} \end{pmatrix} \right), \tag{7.23}$$

where the individual block precisions are, $\mathbf{Q}_{\boldsymbol{ss}} = \kappa_{\boldsymbol{s}}\mathbf{Q}_{\boldsymbol{s}} + \kappa_{\boldsymbol{y}}\mathbf{D}^\top\mathbf{D}$, $\mathbf{Q}_{\boldsymbol{tt}} = \kappa_{\boldsymbol{t}}\mathbf{Q}_{\boldsymbol{t}} + \kappa_{\boldsymbol{y}}\mathbf{D}^\top\mathbf{D}$, $\mathbf{Q}_{\boldsymbol{yy}} = \kappa_{\boldsymbol{y}}\mathbf{I}_n$, $\mathbf{Q}_{\boldsymbol{st}} = \kappa_{\boldsymbol{y}}\mathbf{D}^\top\mathbf{D}$, $\mathbf{Q}_{\boldsymbol{sy}} = \mathbf{Q}_{\boldsymbol{ty}} = -\kappa_{\boldsymbol{y}}\mathbf{D}^\top$ and symmetric counterparts. The block precision $\mathbf{Q}_{\boldsymbol{tt}}$ contains two components, one from the prior and one from the likelihood. However, in the likelihood, only the first n elements of $\boldsymbol{t}$ are present, thus the $\mathbf{D}^\top\mathbf{D}$ contribution. The off-diagonal block precisions are due to the cross terms of the likelihood again only on the first n terms.

We further have

$$\pi(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\kappa} \mid \boldsymbol{y}) \propto \pi(\boldsymbol{y}, \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{\kappa}) \tag{7.24}$$

$$\pi(\boldsymbol{\kappa} \mid \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{y}) = \pi(\kappa_{\boldsymbol{s}} \mid \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{y})\pi(\kappa_{\boldsymbol{t}} \mid \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{y})\pi(\kappa_{\boldsymbol{y}} \mid \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{y}), \tag{7.25}$$

where each density on the right-hand side of the last equation is a Gamma density. The former equation implies that

$$\boldsymbol{s}, \boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y} \sim \mathcal{N}_{2(n+m)}\left(\begin{pmatrix} \mathbf{Q}_{\boldsymbol{ss}} & \mathbf{Q}_{\boldsymbol{st}} \\ \mathbf{Q}_{\boldsymbol{ts}} & \mathbf{Q}_{\boldsymbol{tt}} \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{Q}_{\boldsymbol{sy}} \\ \mathbf{Q}_{\boldsymbol{ty}} \end{pmatrix} \boldsymbol{y}, \begin{pmatrix} \mathbf{Q}_{\boldsymbol{ss}} & \mathbf{Q}_{\boldsymbol{st}} \\ \mathbf{Q}_{\boldsymbol{ts}} & \mathbf{Q}_{\boldsymbol{tt}} \end{pmatrix}^{-1} \right). \tag{7.26}$$

It is now straightforward to implement a Gibbs sampler based on the full conditionals $\pi(\boldsymbol{s}, \boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y})$ and $\pi(\boldsymbol{\kappa} \mid \boldsymbol{s}, \boldsymbol{t}, \boldsymbol{y})$ as illustrated in R Code 7.6. The code essentially (i) loads the data and specifies the hyper-parameters of the prior for $\boldsymbol{\kappa} = (\kappa_{\boldsymbol{y}}, \kappa_{\boldsymbol{s}}, \kappa_{\boldsymbol{t}})^\top$ (ii) builds the precision matrices of equation (7.23) (based on unit precision, i.e., without the individual components of $\boldsymbol{\kappa}$, and constructing "template" matrices for a fast implementation (iii) sets up sampler-specific variables, e.g., initialize the arrays containing the posterior samples and starting values for $\boldsymbol{\kappa}$, (iv) runs the Gibbs sampler.

R-**Code 7.6:** Manual implementation of a Gibbs sampler in the case of a normal response.

```r
# (i) loading the data and setting hyper-parameters:
library(spam)                      # provides the data
y <- sqrt(c(UKDriverDeaths))       # square root of raw counts
n <- length(y)                     # n=192 months of data
m <- 12                            # We want to predict for one season.

priorshape <- c(4,1,1)             # hyper-parameters for priors: alpha's,
priorinvscale <- c(4, 0.1, 0.0005) # beta's, as in Rue & Held (2005)

### (ii) constructing individual block precisions and template matrix:
Qs <- precmat.season(n=n+m, season=12)
Qt <- precmat.RW2(n+m)
DTD <- DT <- In <- diag.spam(n)
pad(DT) <- c(n+m, n)               # used in Qsy = -kappa_y D^T
pad(DTD) <- c(n+m, n+m)            # used in Qss = kappa_s Q_s + kappa_y D^T D

Qst_yk <- rbind(cbind(Qs + DTD, DTD), # Precision for s,t|kappa,y
cbind(DTD, Qt + DTD))                 # where kappas will be used below
struct <- chol(Qst_yk)                # precalculate Cholesky structure

### (iii) setup sampler variables:
burnin <- 10                          # > 0, maybe too small in practice
ngibbs <- 5000                        # maybe too small in practice as well
totalg <- ngibbs + burnin             # total number of Gibbs iterations
set.seed(14)

spost <- tpost <- array(0, c(totalg, n+m))           # prepare arrays
kpost <- array(0, c(totalg, 3))
kpost[1,] <- c(.5, 28, 500)                          # some starting values

postshape <- priorshape + c(n/2, (n+1)/2, (n+m-2)/2) # fixed throughout

### (iv) Gibbs loop:
for (i in 2:totalg) {                                # start loop
  Q <- rbind(cbind(kpost[i-1,2]*Qs + kpost[i-1,1]*DTD, kpost[i-1,1]*DTD),
  cbind(kpost[i-1,1]*DTD, kpost[i-1,3]*Qt + kpost[i-1,1]*DTD))
  b <- c(kpost[i-1,1]*DT %*% y, kpost[i-1,1]*DT %*% y)
  tmp <- rmvnorm.canonical(1, b, Q, Rstruct=struct)  # draw from s,t|kappa,y
  spost[i,] <- tmp[1:(n+m)]                          # separate sample into s
  tpost[i,] <- tmp[1:(n+m)+(n+m)]                    # and t
```

```
    tmp <- y-spost[i,1:n]-tpost[i,1:n]                # "residuals"
    postinvscale <- priorinvscale +                   # prior contribution for
    c(sum(tmp^2)/2,                                   # kappa_y, Qyy=kappa_y In
    t(spost[i,]) %*% (Qs %*% spost[i,])/2,           # kappa_s
    t(tpost[i,]) %*% (Qt %*% tpost[i,])/2)           # kappa_t
    kpost[i,] <- rgamma(3, postshape, postinvscale)   # and finally the draw
  } # end loop
```

Recall that the template precision matrix of the GMRF characterized by $\pi(\boldsymbol{s}, \boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y})$ is to obtain the structure of the Cholesky factor (`struct`). This structure is then used to efficiently factorize the precision matrix (passed as argument `Rstruct=struct` to `rmvnorm.canonical()`). The sparsity structure of the precision matrix and its Cholesky factor is shown in Figure 7.7. Besides this, most of the code does not differ for sparse and non-sparse input matrices.

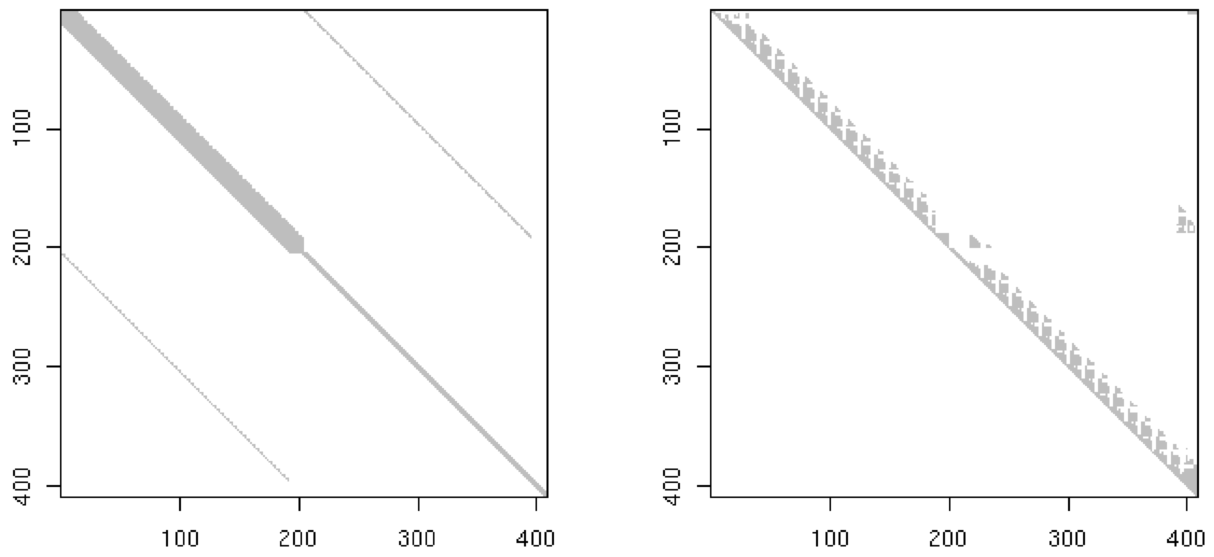


Figure 7.7: The sparsity structure of the precision matrix of $\pi(\boldsymbol{s}, \boldsymbol{t} \mid \boldsymbol{\kappa}, \boldsymbol{y})$ and of its Cholesky factor. Note that the band in the off-diagonal block is "interrupted", marking the start of the null matrix in $\mathbf{D} = (\mathbf{I}, \mathbf{0})$ and $\mathbf{D}^\top$ for the top right and bottom left block, respectively.

The loop takes a few seconds to run. We eliminate burn-in but do not thin. After eliminating the burn-in, summary statistics can be calculated, e.g., `summary(kpost)`. R-Code 7.4 performs the postprocessing, summarizes precisions, constructs quantiles of the mean terms, and provides the code for some instrumental plots. Figure 7.4 (top left) shows the trace plots of the log precisions; together with the autocorrelation plot, there is some evidence of slow mixing, especially for $\kappa_{\boldsymbol{t}}$. The posterior precision of $\kappa_{\boldsymbol{t}}$ and $\kappa_{\boldsymbol{y}}$ cannot be compared with the ones from the last section as we have a much longer dataset here (192 compared to 16).

The posterior mean can be constructed by superimposing the seasonal signal and the trend. The quantiles can be calculated from the posterior signal $\boldsymbol{\eta}$ or from the posterior trend and season (there is a slight difference between *postquant* and *postquant.t+postquant.s*). The middle row of Figure 7.4 shows that the uncertainties are minimal, and a model with a parametric seasonal component or a polynomial trend would not be as adequate.

The predictive distribution of $\boldsymbol{y}$ is obtained by adding a mean zero normal random variable with precision *kpost[i,1]* to *tspost[i,]+tpost[i,]*. The bottom panel of Figure 7.4 gives the posterior median, the posterior quantiles, and the quantiles of the predictive distribution. The latter quantities are again pointwise.

R-Code 7.7: Postprocessing, summaries, some visualization of the sampler output and predictive distribution of the observations. (See Figure 7.8.)

```
kpost <- kpost[ -c(1:burnin),]          # eliminating burn-in
spost <- spost[ -c(1:burnin),]
tpost <- tpost[ -c(1:burnin),]

### Summary of precisions:
allkappas <- rbind(apply(kpost, 2, median), apply(kpost, 2, mean),
apply(1/kpost, 2, median), apply(1/kpost, 2, mean))
colnames(allkappas) <- c("kappa_y", "kappa_s", "kappa_t")
rownames(allkappas) <- c("Prec (median)", "Prec (mean)",
    "Var (median)", "Var (mean) ")
allkappas

##               kappa_y   kappa_s    kappa_t
## Prec (median) 0.49055 28.189063 5.1535e+02
## Prec (mean)   0.49430 31.330842 6.4139e+02
## Var (median)  2.03854  0.035475 1.9404e-03
## Var (mean)    2.05367  0.038263 2.7083e-03

### Construct posterior mean/quantiles of mean terms:
postmean.t <- apply(tpost, 2, mean)
postmean.s <- apply(spost, 2, mean)

postquant <- apply(spost+tpost, 2, quantile,c(.025,.5,.975))
postquant.t <- apply(tpost, 2, quantile,c(.025,.5,.975))
postquant.s <- apply(spost, 2, quantile,c(.025,.5,.975))

### Plotting:
matplot(log(kpost), lty=1, type="l",xlab="Sample index")
abline(h=log(allkappas[1,]), col=3)
acf(kpost[,3], ylab=expression(ACF~of~kappa[t]), main="")
```

```
plot(kpost[,2:3], ylab=bquote(kappa[t]), xlab=bquote(kappa[s]), pch=20, cex=.5)
abline(h=allkappas[1,3], v=allkappas[1,2],col=3)

### Middle panel: for a better visualization, we use the mai argument:
par(mai=c(.001, .9, .1, .1), mgp=c(2.5, 1, 0))
plot(postmean.t, type="l", ylim=range(y), ylab="Trend", xaxt="n")
matlines(t(postquant.t), col=c(4,2,4), lty=1)
points(y, pch=20)

par(mai=c(.9, .9, .001, .1), mgp=c(2.5, 1, 0))
plot(postmean.s, type="l", ylim=c(-5,7), ylab="Season")
matlines(t(postquant.s), col=c(4,2,4), lty=1)

### Bottom panel: Observed counts and quantiles..
ypred <- rnorm(ngibbs*(n+m), c(spost+tpost), sd=rep(1/sqrt(kpost[,1]), n+m))
dim(ypred) <- c(ngibbs, n+m)

postpredquant <- apply(ypred, 2, quantile, c(.025,.975))

plot(y^2, ylim=c(900,3000), xlim=c(7,n+m-7), ylab="Counts", pch=20)#, cex=.5)
matlines(t(postquant)^2, col=c(4,2,4), lty=1)
matlines(t(postpredquant)^2, col=3, lty=1)
legend("topright", bty="n", col=c(2,4,3), lty=1, legend=c("Posterior median",
    "Quantiles of posterior sample", "Quantiles of predictive distribution"))
```

Remark 7.3. The full conditional distribution of the precision parameters is a gamma distribution. This fact is the appealing aspect of the model. However, the mixing of the parameters is not optimal, and somewhat strong auto-correlations are present (see the top middle and right panel of Figure 7.4). For this example, Knorr-Held and Rue (2002) suggest using a Metropolis–Hastings step (introduced in the next chapter) and updating the precisions with a scaling factor δ having a density $\pi(\delta) \propto 1 + 1/\delta$, for $\delta \in [1/D, D]$, where $D > 1$ is a tuning parameter, see also Rue and Held (2005).

If the dimension of $\boldsymbol{\theta}$ is very large, then mixing might be very poor, and blocking strategies should be used. One straightforward case is combining several components and drawing from the full conditional density $\pi(\boldsymbol{\theta}_i \mid \boldsymbol{\theta}_{-i})$. Further, blocking strategies are discussed in Rue and Held (2005), Section 4.1.2. ♡

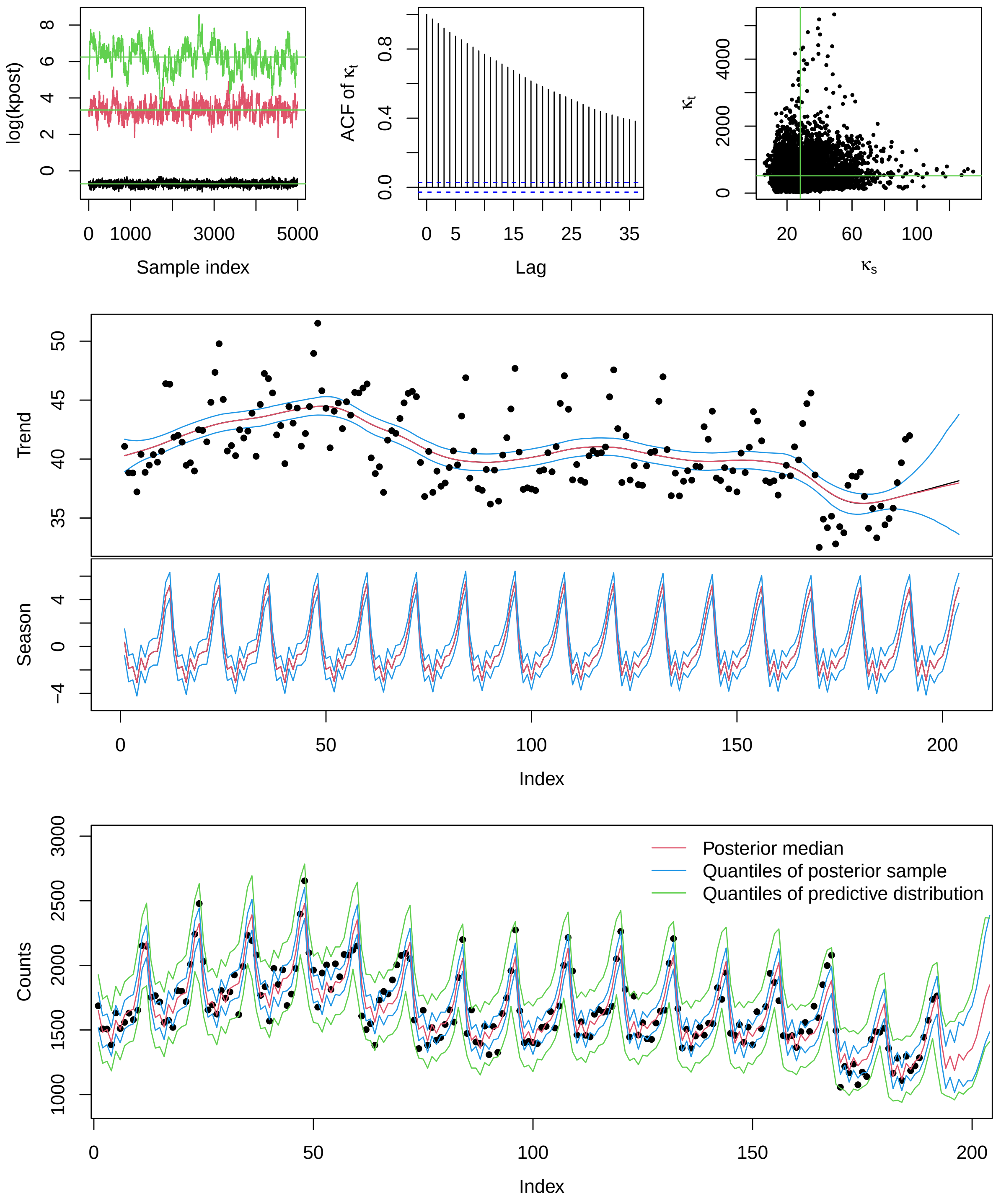


Figure 7.8: Top row: trace plots of the log precision, autocorrelation plot of κ_t and scatter plot of κ_t against κ_s. Solid green lines indicate posterior medians. Middle row: posterior mean of the trend (bottom, with observations) and of the seasonal component (middle) and its 2.5% and 97.5% quantiles. There is visually no difference between the posterior mean and posterior median. Bottom row: observed counts, the pointwise posterior median, quantiles and the pointwise quantiles of the predictive distribution (top panel). (See R-Code 7.7.)

Chapter 8

Hierarchical Models: Metropolis–Hastings Within Gibbs

The Gibbs sampler is applicable only in specific settings. This chapter introduces an alternative sampling scheme. The scheme is more general and can be used in virtually all Bayesian settings.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter08.R.

This chapter illustrates hierarchical modeling by analyzing datasets with a non-normal response with a latent GMRF. Much of this chapter is taken from Gerber and Furrer (2015). The example is also discussed (without documenting code) in Rue and Held (2005), and Section 4.4.2, to which we refer for technical details.

Similarly, as in the final example of the last chapter, we start with a model for the likelihood (distribution for Y_i based on process parameters η_i and possibly some additional ones), a process model for η_i and prior distributions. We relax the normality assumptions of Y_i, and thus a straightforward Gibbs sampler might not work.

We assume that the observations $\boldsymbol{y}$ are a realization of $\mathbf{Y}$ which are conditionally independent given latent parameters $\boldsymbol{\eta}$ and additional parameters $\boldsymbol{\theta}_{\boldsymbol{y}}$

$$\pi(\boldsymbol{y} \mid \boldsymbol{\eta}, \boldsymbol{\theta}_{\boldsymbol{y}}) = \prod_{i=1}^{n} \pi(y_i \mid \eta_i, \boldsymbol{\theta}_{\boldsymbol{y}}), \tag{8.1}$$

where $\pi(\cdot \mid \cdot)$ denotes the conditional density of the first argument given the second argument. The latent parameters $\boldsymbol{\eta}$ are part of a larger latent random field $\boldsymbol{x}$, which is modeled as a GMRF with mean $\boldsymbol{\mu}$ and precision matrix $\mathbf{Q}$, both depending on parameters $\boldsymbol{\theta}_{\boldsymbol{x}}$; that is,

$$\pi(\boldsymbol{x} \mid \boldsymbol{\theta}_{\boldsymbol{x}}) \propto \exp\Big(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \mathbf{Q}(\boldsymbol{x}-\boldsymbol{\mu})\Big). \tag{8.2}$$

The idea of "part of a larger latent random field $\boldsymbol{x}$" was already used in the last chapter: we would set $\boldsymbol{x}^\top = (\boldsymbol{s}^\top, \boldsymbol{t}^\top)$ and $\boldsymbol{\eta} = \boldsymbol{s} + \boldsymbol{t}$.

Depending on the form of $\pi(\boldsymbol{y} \mid \boldsymbol{\eta}, \boldsymbol{\theta}_{\boldsymbol{y}})$ typically closed form expressions for the full conditionals $\pi(\boldsymbol{x} \mid \boldsymbol{y}, \boldsymbol{\theta}_{\boldsymbol{y}})$, $\pi(\boldsymbol{\theta}_{\boldsymbol{y}} \mid \boldsymbol{y}, \boldsymbol{x})$ do not exist. Hence, we cannot directly apply a Gibbs sampler. One typically augments the Gibbs sampler with individual, so-called, Metropolis-Hastings steps. In the following sections, we first introduce the general concept of a Metropolis-Hastings algorithm, discuss convergence diagnostics and implement this algorithm into a Gibbs sampler through different examples.

8.1 The Metropolis–Hastings Algorithm

To simplify the notation, assume that we want to sample from a (target) density $\pi(\theta)$ (for which we do not have a direct sampling approach) and suppress the dependency on all other variables (data, additional parameters). Assume that we have a proposal density $q(\cdot)$ (for which we have a direct and straightforward sampling approach). More specifically, we assume that the proposal density depends on the previous state $\theta^{(i-1)}$, i.e., $q(\theta \mid \theta^{(i-1)})$. The Metropolis-Hastings algorithm is as follows:

(1) Set $\theta^{(0)}$ to some value and set $i = 1$

(2) Sample first a proposal $\theta^\star$ from $q(\theta \mid \theta^{(i-1)})$
then sample u from a uniform $(0, 1)$ random variable.

(3) Set $\alpha = \min\Big(1, \dfrac{\pi(\theta^\star)}{\pi(\theta^{(i-1)})} \dfrac{q(\theta^{(i-1)} \mid \theta^\star)}{q(\theta^\star \mid \theta^{(i-1)})}\Big)$.
If $u \leq \alpha$ set $\theta^{(i)} = \theta^\star$ (accept new one), otherwise set $\theta^{(i)} = \theta^{(i-1)}$ (keep current)

(4) If $i < M$, $i = i + 1$ and go to (2) else stop

The choice of the proposal density q is an art per se. We require that for all θ such that $\pi(\theta) > 0$ we have $q(\theta \mid \cdot)$, i.e., the support of $\pi(\cdot)$ is contained in the support of $q(\cdot)$. The more similar $q(\cdot)$ to $\pi(\cdot)$, the better the algorithm works. There are two important subclasses:

1. The proposal is symmetric with respect to the previous state: $q(\theta \mid \theta^\star) = q(\theta^\star \mid \theta)$. Examples are random-walk proposals, where q is symmetric around θ. Typically, a uniform or normal realization is added to the current value $\theta^{(i-1)}$. Notice that the calculation of α simplifies the likelihood ratio of the target.

2. The proposal does not depend on the current state: $q(\theta \mid \theta^{(i-1)}) = q(\theta)$. The proposal is called an independence proposal.

We defer to, e.g., Roberts and Smith (1994); Gamerman and Lopes (2006) for a thorough discussion on the assumptions of the sampler.

Example 8.1. As a trivial example, we use an MH approach to sample from a chi-squared distribution with four degrees of freedom. As a proposal, we use a Gaussian, centered at the current state with a pre-specified standard deviation τ. The sampler is coded in R-code 8.1 ($\tau = 5$ and starting value 10) with output and summaries illustrated in Figure 8.1. The first panel of the figure shows a trace plot of the entire sample. The acceptance rate levels out to approximately

45%. The autocorrelation plot of the thinned sample does not show any correlation. The smoothed density of the sample is very close to the true density, and the deviations are of stochastic nature only. ♣

R-Code 8.1: MH for a simple case. (See Figure 8.1.)

```
### Defining the generic functions (can be modified)!
alpha.fcn <- function(theta,phi) {
  min(1, pi.fcn(phi)*q.fcn(phi,theta) /(pi.fcn(theta)*q.fcn(theta,phi) )) }
pi.fcn <- function(x) dchisq(x, df=4)     # target density
q.fcn  <- function(x,y) dnorm(y, x, tau) # proposal density
rq.fcn <- function(x) rnorm(1, x, tau)    # samples from the proposal

### Sampler specific parameters and then the sampler loop:
tau <- 5                                  # Tuning parameter. With 5 ~ 43%
burnin <- 1000
every <- 10
N <- 500*every+burnin                     # total number of samples
start <- 10                               # starting value
set.seed(14)
acceptance <- theta <- numeric(N)         # vectors holding the result
theta[1] <- start
for (i in 2:N){
  phi <- rq.fcn(theta[i-1])               # draw proposal
  a <- alpha.fcn(theta[i-1], phi)         # evaluate alpha
  u <- runif(1)
  theta[i] <- ifelse(u < a, phi, theta[i-1]) # accept or keep
  acceptance[i] <- ifelse(u < a, 1, 0)    # mark if accepted
}
acceptance <- cumsum(acceptance)/1:N
### Sampling is now done. Now follows plotting:
plot(theta, type="l")
abline(v=burnin, col=5, lwd=2, lty=2)
theta <- theta[-c(1:burnin)]
theta <- theta[seq(1,to=length(theta),by=every)]
acf(theta)
plot(acceptance, type="l", ylim=c(0,1))
title(paste("Acceptance rate", round(acceptance[N],3)))
hist(theta, prob=TRUE, breaks=20)
lines(density(theta), lwd=2, col=2)
curve(pi.fcn, from=min(theta), to=max(theta), col=3, add=T)
```

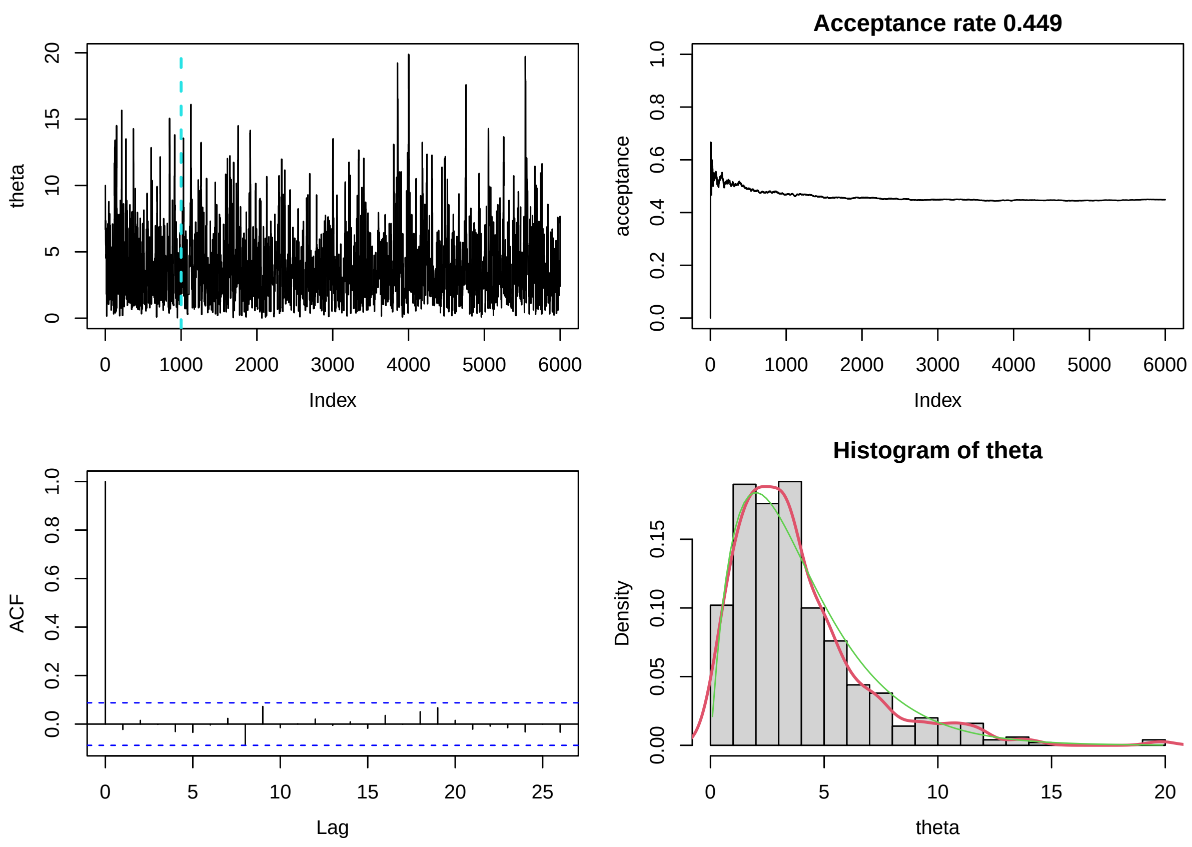


Figure 8.1: Metropolis–Hastings simulation of a chi-squared distribution with four degrees of freedom. Top row: trace plot of the entire sample and acceptance rate. Bottom row: autocorrelation plot of the thinned sample and histogram and smoothed density of the sample. True density is given in green. (See R-Code 8.1.)

Remark 8.1. The Gibbs sampler is a particular case of a component-wise Metropolis-Hastings step, where the acceptance probability α is one in all cases.

More specifically, the proposal function is the full conditional density $\pi(\theta^\star \mid \theta^{(i-1)}) = q(\theta^\star \mid \theta^{(i-1)})$ and thus the acceptance probability of step (3) writes

$$\alpha = \min\Big(1, \frac{\pi(\theta^\star \mid \theta^{(i-1)})}{\pi(\theta^{(i-1)} \mid \theta^\star)} \frac{\pi(\theta^{(i-1)} \mid \theta^\star)}{\pi(\theta^\star \mid \theta^{(i-1)})}\Big) \equiv 1. \tag{8.3}$$

♡

8.2 Convergence Diagnostics

At convergence, the MCMC values represent a sample from the target distribution and should fluctuate around the stable mean value. While it is never possible to formally prove that a sequence has converged, there are several tools that indicate convergence or the lack thereof. One should always plot the (thinned) sample versus the indices, called the trace plots. These plots indicate how quickly they fluctuate around the mean. Slow meandering is an indication of slow or poor mixing. Changing the initial value helps to assess how quickly a chain reaches its equilibrium.

It is suggested that the acceptance probability should be between 20–50% (Gamerman and Lopes, 2006 and references therein). To achieve that acceptance rate, the tuning parameter can be adjusted during the burn-in period.

Example 8.2. We use the same MCMC setup as in Example 8.1 with $\tau = 20$ and $\tau = 0.1$ instead of $\tau = 5$, leading to too large and too small jumps, respectively, (τ is the standard deviation of our proposal). The resulting trace and autocorrelation plots are shown in Figure 8.2 (no thinning and burn-in).

The large jumps often lead to regions of low "likelihood", or even outside the support and thus are rarely accepted. Conversely, too small jumps lead to high acceptance rates. The latter is not bad per se, but this implies that the sampler may not cover the entire support of the random variable. In our example, this is the case as indicated by the range of sample (size of $N = 6000$); see the lower right panel of Figure 8.1) for a comparison. ♣

R-**Code 8.2** Bad mixing due to inappropriate jump sizes. For illustration, we do not thin. (See Figure 8.2.)

```
for (tau in c(20, .1)) {
  theta[1] <- 4
  for (i in 2:N){
    phi <- rq.fcn(theta[i-1])
    a <- alpha.fcn(theta[i-1], phi)
    u <- runif(1)
    theta[i] <- ifelse(u < a, phi, theta[i-1])
    acceptance[i] <- ifelse(u < a, 1, 0)
  }
  plot(theta[1:100], type="l")
  title(paste("Acceptance rate", round(sum(acceptance)/N,3)))
  acf(theta)
  print(range(theta))
}
## [1]  0.083703 23.284970
## [1]  0.17094 10.79693
```

For convergence diagnostics, similar plots and tests as for a Gibbs sampler should be performed. However, there are several additional model diagnostic statistics. We now look at some intuitive and often used ones. But recall that diagnostics cannot guarantee that a chain has converged. Especially in multivariate settings, diagnostics are not well established.

Gelman and Rubin (1992) propose a general approach to monitoring convergence of MCMC output in which $m > 1$ parallel chains are run with starting values that are overdispersed relative to the posterior distribution. Convergence is diagnosed when the chains have 'forgotten' their

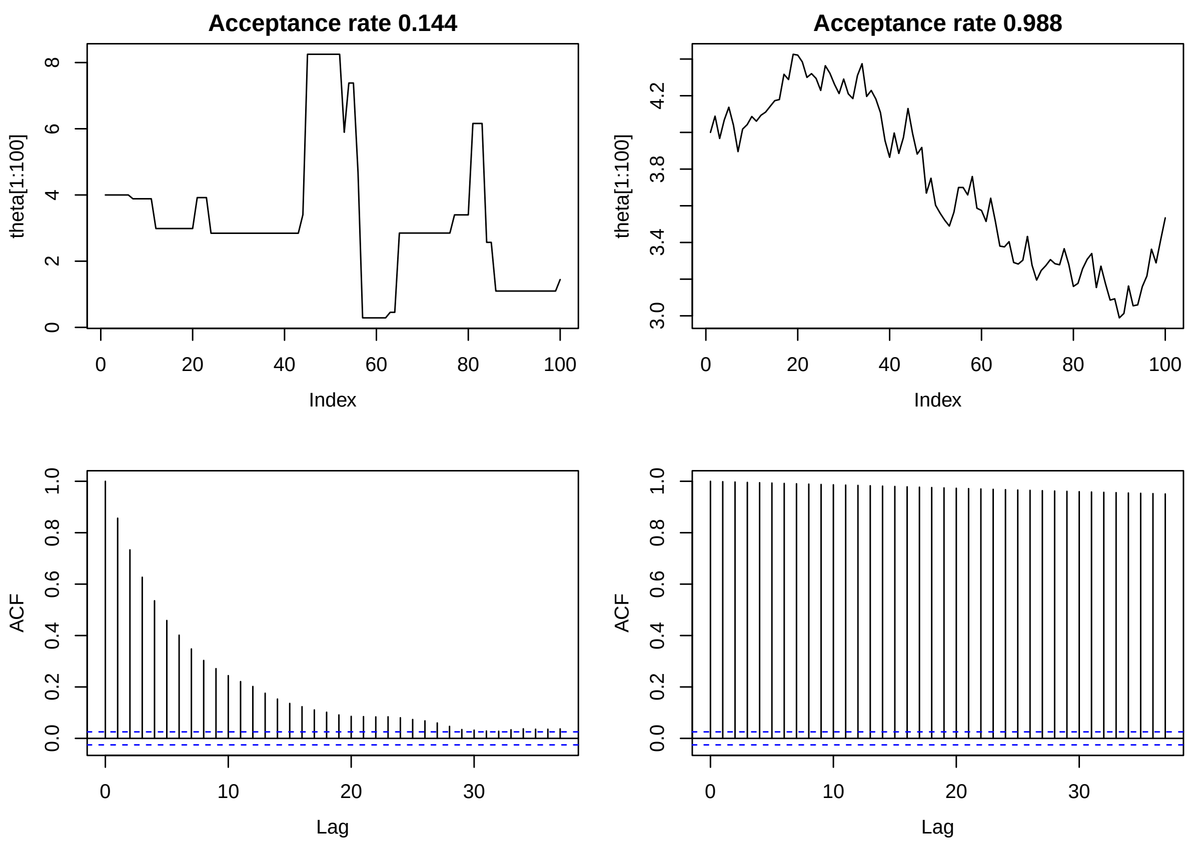


Figure 8.2: Bad mixing, $\tau = 10$ left (too large jumps, low acceptance rate) and $\tau = 0.1$ right (too small jumps, high acceptance rate). Notice the different scales for the trace plots. (See R-Code 8.2.)

initial values, and the output from all chains is indistinguishable. The `gelman.diag()` diagnostic is applied to a single variable from the chain. It is based on a comparison of within-chain and between-chain variances, and is similar to classical analysis of variance.

Geweke (1992) proposed a convergence diagnostic for Markov chains based on a test for equality of the means of the first and last part of a Markov chain (by default, the first 10% and the last 50%). If the samples are drawn from the stationary distribution of the chain, the two means are equal and Geweke's statistic has an asymptotically standard normal distribution (`geweke.plot()`).

The convergence test `heidel.diag()` uses the Cramér–von Mises statistic to test the null hypothesis that the sampled values come from a stationary distribution (Heidelberger and Welch, 1981). The test is successively applied, firstly to the whole chain, then after discarding the first 10%, 20%, of the chain until either the null hypothesis is accepted or 50% of the chain has been discarded. The latter outcome constitutes "failure" of the stationarity test and indicates that a longer MCMC run is needed.

The test `raftery.diag()` calculates the number of iterations required to estimate the quantile q to within an accuracy of $\pm r$ with probability p. Values of dependence factors larger than 5 indicate strong autocorrelation (which may be due to a poor choice of starting value, high posterior correlations, or stickiness of the MCMC algorithm, Raftery and Lewis, 1992).

Example 8.3. We illustrate some of the model diagnostics with the sample used in Example 8.1. Further, we create a second sample using the same sampler with $\tau = 0.1$, proposing too small jumps and not properly exploring the entire sample space. We denote the sample with $\tau = 5$ as *thetagood* and $\tau = 0.1$ as *thetabad*.

Figure 8.3 illustrates the output of *gelman.plot()* and *geweke.plot()*. The former is not always straightforward to interpret, as a statement "close to one" may be slightly subjective.

The output of the other diagnostics clearly differentiates between the two samples. ♣

R-**Code 8.3:** MCMC diagnostics. (See Figure 8.3.)

```
library(coda)
chain1 <- mcmc(thetagood[1:2000])
gelman.plot(mcmc.list(chain1, mcmc(thetagood[2001:4000])), auto.layout=FALSE)
gelman.diag(mcmc.list(chain1, mcmc(thetagood[2001:4000])))
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       1.01       1.01
geweke.plot(chain1, auto.layout=FALSE)
# geweke.diag(chain1)              # Output is harder to interpret
heidel.diag(chain1)                # Note: failure if we take length 1000 only
##
##      Stationarity start     p-value
##      test         iteration
## var1 passed       1         0.742
##
##      Halfwidth Mean Halfwidth
##      test
## var1 passed    3.8  0.283
raftery.diag(chain1)
##
## Quantile (q) = 0.025
## Accuracy (r) = +/- 0.005
## Probability (s) = 0.95
##
## You need a sample size of at least 3746 with these values of q, r and s
chain2 <- mcmc(thetabad[1:2000]) # chain with too slow mixing (tau=0.5)
gelman.plot(mcmc.list(chain2, mcmc(thetabad[2001:4000])), auto.layout=FALSE)
gelman.diag(mcmc.list(chain2, mcmc(thetabad[2001:4000])))
```

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]       5.21         15
### should be much, much smaller!
geweke.plot(chain2, auto.layout=FALSE)
heidel.diag(chain2)            # fails even, if we take the entire sample
##
##      Stationarity start     p-value
##      test         iteration
## var1 passed       401       0.125
##
##      Halfwidth Mean Halfwidth
##      test
## var1 failed    5.88 1.97
```

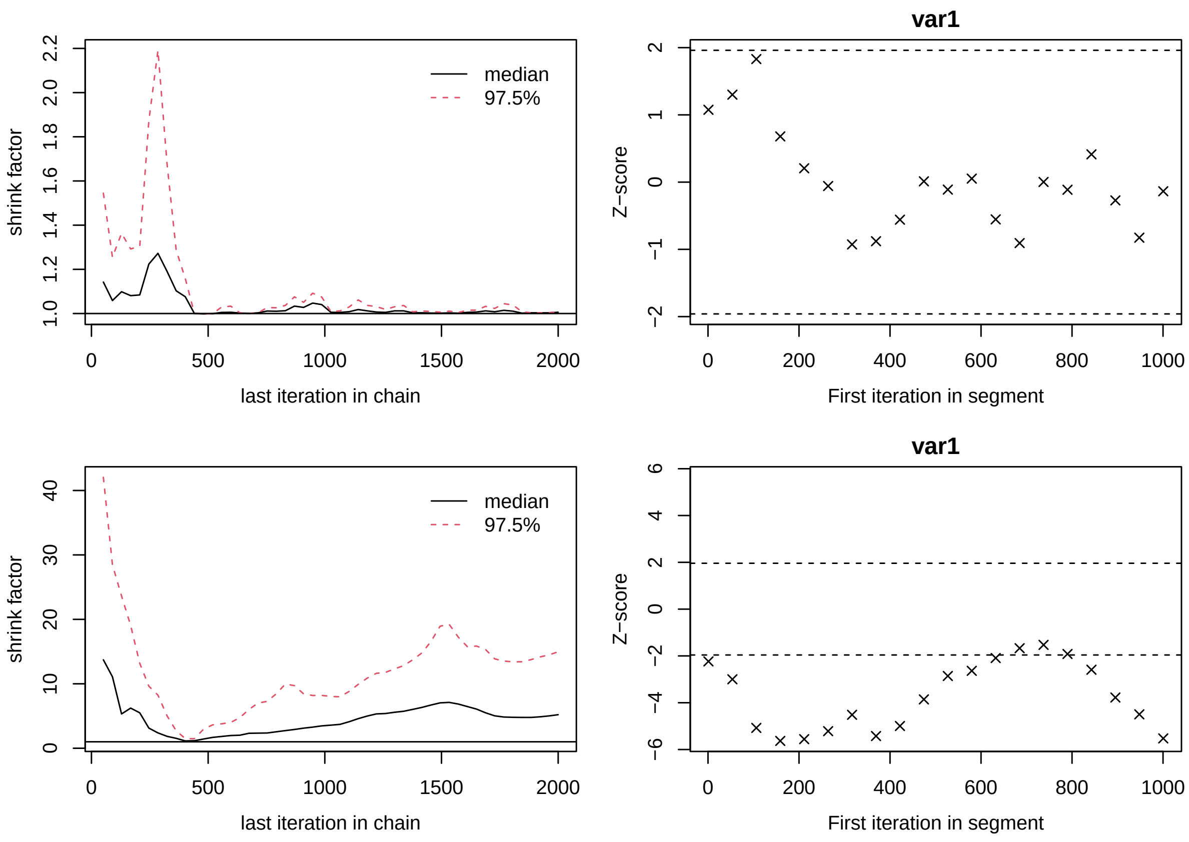


Figure 8.3: Diagnostics based on `gelman.plot()` and `geweke.plot()`. The first chain (top row) is based on R-Code 8.1 and the second (bottom row) on 8.2 with starting value 10 and $\tau = 0.1$. (See R-Code 8.3.)

8.3 Example: Besag–York–Mollié Model

In this example, we consider the number of oral cavity cancer cases for the five years 1986–1990 in the $n = 544$ districts (Landkreise) of Germany (Knorr-Held and Raßer, 2000; Held *et al.*, 2005) and explore the spatial distribution of the relative risk. The expected number of cases e_i was derived using demographical data that allows us to display the standardized mortality ratios y_i/e_i (Figure 8.4, middle map).

The common approach is to assume that the data are conditionally independent Poisson counts, which means $Y_i \mid \eta_i \sim \mathcal{P}ois(\lambda_i)$ with $\lambda_i = e_i \exp(\eta_i)$, where e_i is the expected number of cases in region i, $\exp(\eta_i)$ the relative risk and η_i the log-relative risk. Hence,

$$\pi(y_i \mid \eta_i) \propto \exp\big(y_i\eta_i - e_i \exp(\eta_i)\big), \qquad i = 1, \dots, n, \tag{8.4}$$

where all other terms are constants, i.e., do not depend on η_i. For the log-relative risk, we use $\boldsymbol{\eta} = \boldsymbol{u} + \boldsymbol{v}$, where $\boldsymbol{v}$ is a zero mean white noise with precision $\kappa_{\boldsymbol{v}}$ and $\boldsymbol{u}$ is a spatially structured component, leading to the so-called Besag–York–Mollié model (see, e.g., Besag *et al.*, 1991; Mollié, 1996).

More precisely, $\boldsymbol{u}$ is a first-order intrinsic GMRF with density

$$\pi(\boldsymbol{u} \mid \kappa_{\boldsymbol{u}}) \propto \kappa_{\boldsymbol{u}}^{\frac{n-1}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{u}}}{2}\sum_{i\sim j}(u_i - u_j)^2\Big) = \kappa_{\boldsymbol{u}}^{\frac{n-1}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{u}}}{2}\boldsymbol{u}^\top \mathbf{R}\boldsymbol{u}\Big), \tag{8.5}$$

where $i \sim j$ denotes the set of all unordered pairs of neighbors, i.e., regions sharing a common border, and $\mathbf{R}$ the induced structure matrix. Moreover,

$$\pi(\boldsymbol{v} \mid \kappa_{\boldsymbol{v}}) \propto \kappa_{\boldsymbol{v}}^{\frac{n}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{v}}}{2}\boldsymbol{v}^\top \boldsymbol{v}\Big). \tag{8.6}$$

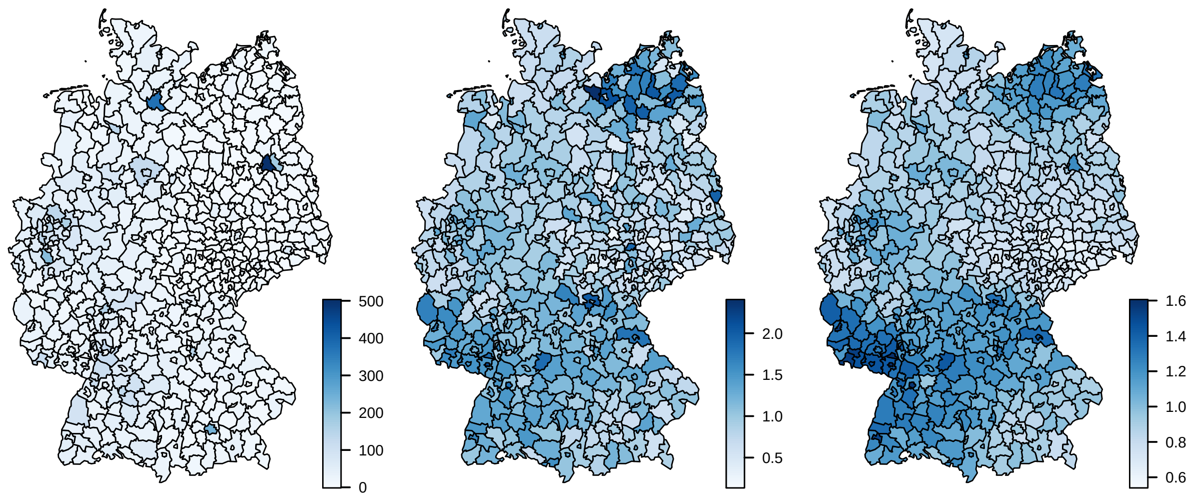


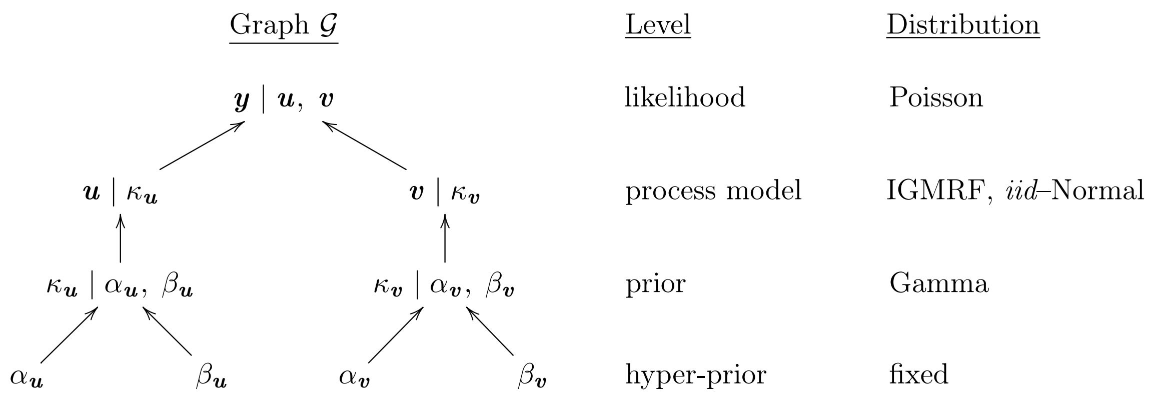
Figure 8.4: Aggregated raw counts (left) and standardized mortality ratios (middle) of oral cavity cancer deaths observed between 1986–1990 in Germany; the posterior medians of the estimated relative risks from a BYM model (right). Notice the slightly different color range in the two rightmost panels.

Thus with Gamma priors for the precision parameters, the posterior density is

$$
\begin{aligned}
\pi(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\kappa} \mid \boldsymbol{y}) \propto\ & \kappa_{\boldsymbol{v}}^{\alpha_{\boldsymbol{v}}+\frac{n}{2}-1} \kappa_{\boldsymbol{u}}^{\alpha_{\boldsymbol{u}}+\frac{n-1}{2}-1} \\
& \times \exp\Big(-\kappa_{\boldsymbol{v}}\beta_{\boldsymbol{v}} - \kappa_{\boldsymbol{u}}\beta_{\boldsymbol{u}} + \sum_{i=1}^{n} \big(y_i\eta_i - e_i \exp(\eta_i)\big) - \frac{\kappa_{\boldsymbol{u}}}{2}\boldsymbol{u}^\top \mathbf{R}\boldsymbol{u} - \frac{\kappa_{\boldsymbol{v}}}{2}\boldsymbol{v}^\top\boldsymbol{v}\Big),
\end{aligned}
\tag{8.7}
$$

with $\eta_i = u_i + v_i$. While $\pi(\boldsymbol{u}, \boldsymbol{v} \mid \boldsymbol{\kappa})$ is a GMRF, $\pi(\boldsymbol{u}, \boldsymbol{v} \mid \boldsymbol{\kappa}, \boldsymbol{y})$ is not, and many details need to be taken into account to implement an MCMC sampler successfully.

Figure 8.5 illustrates the different components of the model in a hierarchy (here, graph structure). The joint density can be constructed by successively writing down the individual densities, albeit some reparametrizations may not be seen from the variable dependency graph. We present three approaches, two build-in ones and a manual one, to obtain posterior samples from (8.7). While the first example is straightforward, the remaining two are of considerable complexity.


Graph $\mathcal{G}$

$$\begin{tikzcd}
 & & & \boldsymbol{y} \mid \boldsymbol{u},\ \boldsymbol{v} & & & \\
 & \boldsymbol{u} \mid \kappa_{\boldsymbol{u}} \arrow[urr] & & & & \boldsymbol{v} \mid \kappa_{\boldsymbol{v}} \arrow[ull] & \\
 & \kappa_{\boldsymbol{u}} \mid \alpha_{\boldsymbol{u}},\ \beta_{\boldsymbol{u}} \arrow[u] & & & & \kappa_{\boldsymbol{v}} \mid \alpha_{\boldsymbol{v}},\ \beta_{\boldsymbol{v}} \arrow[u] & \\
\alpha_{\boldsymbol{u}} \arrow[ur] & & \beta_{\boldsymbol{u}} \arrow[ul] & & \alpha_{\boldsymbol{v}} \arrow[ur] & & \beta_{\boldsymbol{v}} \arrow[ul]
\end{tikzcd}$$

Level	Distribution
likelihood	Poisson
process model	IGMRF, *iid*–Normal
prior	Gamma
hyper-prior	fixed

Figure 8.5: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

8.3.1 CARBayes Implementation

The package *CARBayes* implements a sampler for the BYM model. The function *S.CARbym()* implements the model via a classical formula statement. The function's arguments are quite self-explanatory, as shown in R-Code 8.4. With a burn-in of 1 000 and a thinning of 10, the sampler takes less than half a minute on a decent laptop to generate the 2 000 retained samples (roughly 1 000 iterations per second are achieved).

The trace plots of the log precisions and a scatterplot thereof are shown in Figure 8.6. For easier comparison, we will work with (log) precisions, hence the negative log transformation. The straightforward automated way comes with the price of a sub-optimal mixing and a chain for $\boldsymbol{\eta}$ only (instead of two separate chains for $\boldsymbol{u}$ and $\boldsymbol{v}$). More specifically, $\boldsymbol{\eta}$ includes an intercept and a spatial term. The right panel of Figure 8.4 illustrates the posterior relative risk, i.e., $\exp(\boldsymbol{\eta})$. Due to the spatially structured component, smoothing is present. We refer to Knorr-Held and Best (2001) for an (epidemiological) interpretation of the results.

R-Code 8.4 *CARBayes* BYM model implementation. (See Figure 8.6.)

```
library(spam)         # data is called Oral, in package spam
library(CARBayes)
adj.loc <- system.file("demodata/germany.adjacency", package="spam")
A <- adjacency.landkreis(adj.loc) # loading adjacency matrix, a spam matrix
B.out <- S.CARbym(Y ~ offset(log(E)), data=Oral, family="poisson",
    W=as.matrix(A), burnin=1000, n.sample=21000, thin=10,
    prior.tau2=c(1, 0.5), prior.sigma2=c(1, 0.01))
### Diagnostic plot:
grid_trace2(-log(cbind(B.out$samples$tau2, B.out$samples$sigma2)))
### Right most panel of Figure 8.4
B.u.median <- apply(B.out$samples$psi, 2, median)
B.inter.median <- median(B.out$samples$beta[,1])
germany.plot(exp(B.u.median+B.inter.median))
```

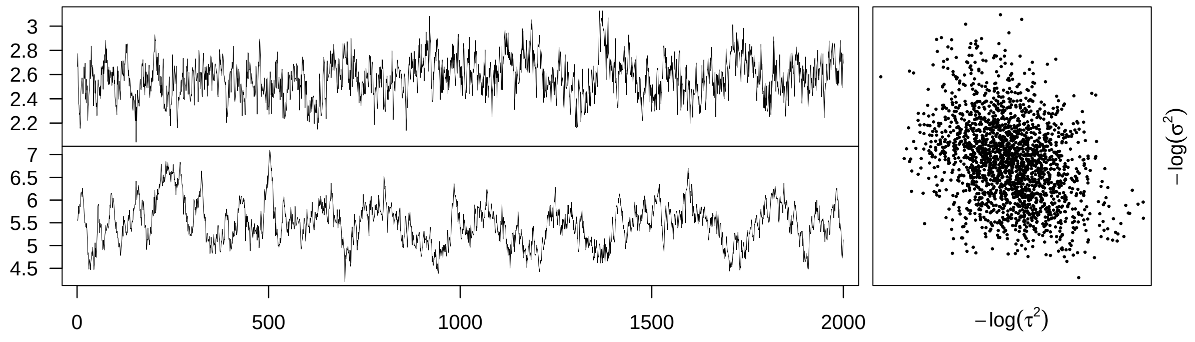


Figure 8.6: Diagnostic plots for the 2000 post burn-in, thinned samples from the *CARBayes* MCMC BYM implementations are shown. Each panel consists of trace plots for $\kappa_u = -\log(\tau^2)$ (upper) and $\kappa_v = -\log(\sigma^2)$ (lower), respectively. The mixing of κ_u and κ_v is shown in a scatter plot (right). The chains were already thinned with a factor of 10.

8.3.2 OpenBUGS Implementation

The most dominant disadvantage of ready-made MCMC implementations, as the one seen in the last section, is the model specification's rigidity. We now look at one alternative without limitations on the model structure and prior specification. The idea is to specify the variable dependency graph, which will then be used in a sampler. Such samplers are software programs typically available outside of R. The BUGS (Bayesian inference Using Gibbs Sampling) family is one of them. Moreover, there are convenient ways to communicate with them via R.

The BUGS/OpenBUGS language is used to specify hierarchical Bayesian models and communicate them to the OpenBUGS engine residing outside R. (To install the OpenBUGS software, visit www.mrc-bsu.cam.ac.uk/software/bugs/openbugs/.) We define the distribution of each level given its parameters nodes using an R like syntax. The model description is declarative, meaning that the order of the node definitions is irrelevant. As usual, the tilde symbol '∼' stands for "is distributed as." In order to specify the spatially structured term $\boldsymbol{u}$, the `car.normal()` distribution from the geoBUGS extension is used. Since the distribution of $\boldsymbol{u}$ is implemented with a sum-to-zero constraint, we add an additional intercept with an improper flat prior. This construct is claimed to be equivalent to a spatially structured term $\boldsymbol{u}$ without constraint (Lunn *et al.*, 2013, p. 264). We save the model to a text file '`model.txt`', part (i) of R-Code 8.5.

The observed and expected counts `Y` and `E`, as well as the neighborhood structure `adj`, are saved in a separate text file '`data.txt`'. The arguments of the `car.normal()` distribution are a sparse adjacency matrix `adj`, the number of regions connected in each row of the adjacency matrix `num[]` (also stored in '`data.txt`'), the precision of `u`, i.e., `kappaU`, and a vector of 1's in `weight`. (We do not weigh the adjacency structure). For another BUGS example of a BYM model, see Bivand *et al.* (2013).

We now use the R function `bugs()` from the R package `R2OpenBUGS`, which provides a convenient user interface to openBUGS (Sturtz *et al.*, 2005). 21000 samples from the posterior distribution are generated with the call. A thinning of 10 and a burn-in of $100 \times 10 = 1000$ is specified, resulting in 2000 actually returned samples per variable.

We manually set initial values for `kappaU` and `kappaV` via the argument `inits`. Further, the argument `parameters` specify variables for which samples are stored and returned to R. We only simulate one chain and set `n.chains=1` for demonstration, but we recommend simulating several chains with different initial values that help to assess convergence. openBUGS automatically selects an appropriate sampling method for each node.

The simulation lasts several minutes using the same number of (total) iterations as in the previous sections. Figure 8.7 shows again diagnostic plots for the samples $\log(\kappa_{\boldsymbol{u}})$ and $\log(\kappa_{\boldsymbol{v}})$ of the Markov chain, and Figure 8.11 given in the next section shows the resulting posterior median fields of the log-relative risks.

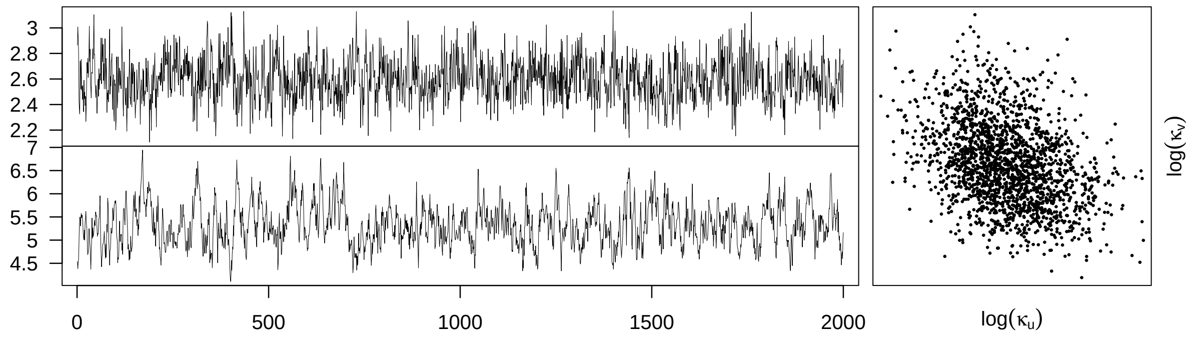


Figure 8.7: Diagnostic plots for the 2000 post burn-in and thinned samples from the `openBUGS` MCMC BYM implementations are shown. Each panel consists of trace plots for $\log(\kappa_{\boldsymbol{u}})$ (upper) and $\log(\kappa_{\boldsymbol{v}})$ (lower), respectively. The mixing of $\log(\kappa_{\boldsymbol{u}})$ and $\log(\kappa_{\boldsymbol{v}})$ is shown in a scatter plot (right). The chains were already thinned with a factor of 10.

R-Code 8.5 OpenBUGS BYM implementation.

```
library(R2OpenBUGS)
### (i) Define the hierarchical model
writeLines(" model{
    for(i in 1:N){
      Y[i] ~ dpois(landa[i])
      log(landa[i]) <- log(E[i]) + u[i] + v[i] }
    for(i in 1:N){ v[i] ~ dnorm(0, kappaV) }
    for(i in 1:N){ u[i] <- uConstr[i] + intercept }
    intercept ~ dflat()
    uConstr[1:N] ~ car.normal(adj[], weights[], num[], kappaU)
    for(k in 1:sumNumNeigh) { weights[k] <- 1 }
    kappaU ~ dgamma(1, 0.5)
    kappaV ~ dgamma(1, 0.01)
    }", con="model.txt")
### (ii) provide data
tmp <- bugs.data(list(Y=Oral$Y, E=Oral$E, N=length(Oral$Y),
    adj=A@colindices, num=diff(A@rowpointers),
    sumNumNeigh=length(A@colindices)))
### (iii) run the model
O.out <- bugs(model.file="model.txt", data="data.txt",
    inits=function() { list(kappaU=10, kappaV=100, intercept=1) },
    parameters=c("kappaU", "u", "kappaV", "v"), n.iter=2100,
    n.burnin=100, n.thin=10, n.chains=1)
### (iv) diagnostics... more to add
grid_trace2(cbind(O.out$sims.array[,1,c("kappaU","kappaV")],
    O.out$sims.array[,1,"kappaV"]))
```

8.3.3 INLA implementation

As opposed to simulation-based inference methods (i.e., full-fledged MCMC methods), the R package *INLA* uses iteratively nested Laplace approximations to estimate model parameters (Rue *et al.*, 2009). The INLA machinery has received substantial attention, and it seems omnipresent. Such a flexible approach often requires many flags, arguments, control options, etc., and thus often implies some "learning-by-doing". The package, documentation, and examples are available on `https://www.r-inla.org/`; installation is done via `https://www.r-inla.org/download-install`.

R-Code 8.6 illustrates the INLA implementation of the BYM model. We first load the R package *INLA* and the oral cancer data. *path* contains the path to a file encoding the corresponding adjacency matrix. Since *INLA* requires an index variable for each of the modeled components $\boldsymbol{u}$ and $\boldsymbol{v}$, we have to duplicate the index column in the data frame, as we would for a "classical" design matrix as well.

Next, we define the model through a formula. The functions *f()* specify the models for $\boldsymbol{u}$

and $\boldsymbol{v}$, respectively. For $\boldsymbol{u}$, a regional structured prior is selected by setting `model= "besag"` and supplying a graph and a hyper-prior. We choose an unconstrained model (`constr=FALSE`), which implies that this random effect absorbs the intercept. Hence, the intercept is not identifiable, and we remove it from the formula through `-1` in the first line of the formula definition. The *iid* random effect $\boldsymbol{v}$ is specified in the second `f()` function. The argument `hyper` sets the hyper-parameters (of the prior parameter `theta`).

R-Code 8.6 INLA BYM implementation.

```r
if (!require(INLA)) install.packages("INLA", repos=c(getOption("repos"),
  INLA="https://inla.r-inla-download.org/R/stable"), dep=TRUE)
library(INLA)
data("Oral")
path <- system.file("demodata/germany.graph", package="INLA")
Oral.inla <- cbind(Oral, region.struct=Oral$region)

formula <- Y ~ - 1 + # specify the model, here without intercept
  f(region.struct, model="besag", graph=path, constr=FALSE,
      hyper=list(theta=list(param=c(1, 0.5)))) +
  f(region, model="iid", hyper=list(theta=list(param=c(1, 0.01))))
I.out <- inla(formula, family="poisson", data=Oral.inla, E=E)
```

Notice that INLA does not generate a chain but uses a set of iterative approximations. Based on these approximations, (marginal) posterior densities are available (see R-Code 8.7 and Figure 8.8). Note that there is a substantial difference between the densities. This difference is essentially due to the fundamentally different model implementations.

R-Code 8.7 Comparing posterior densities of precision parameters. (See Figure 8.8.)

```r
plot(density(1/B.out$samples$tau2), main="", xlab="",
    ylab=expression(paste(pi,"(",kappa[u],"|",y,")")))
lines(I.out$marginals.hyperpar[[1]], type="l", col=4)
tmp <- I.out$summary.hyperpar[,c("mean","0.5quant","mode")]
abline(v=tmp[1,], col=c(1:3))

plot(density(1/B.out$samples$sigma2), main="", ylim=c(0,.005), xlab="",
    ylab=expression(paste(pi,"(",kappa[v],"|",y,")")))
lines(I.out$marginals.hyperpar[[2]], type="l", col=4)
abline(v=tmp[2,], col=c(1:3))
```

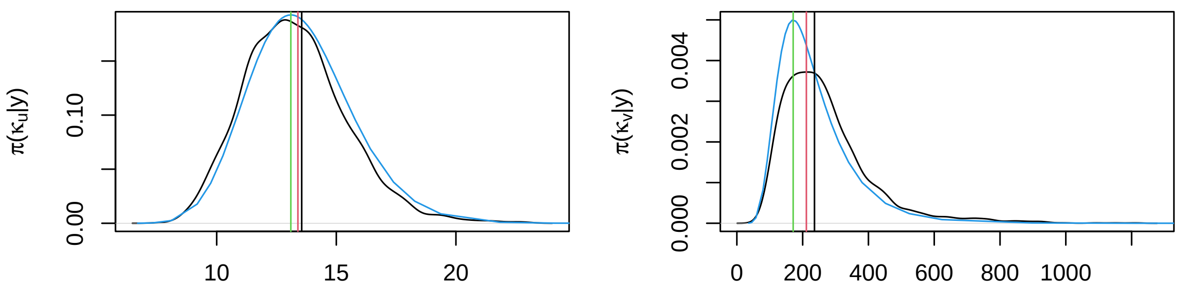


Figure 8.8: Comparison of posterior densities $\kappa_{\boldsymbol{u}}$ (left) and $\kappa_{\boldsymbol{v}}$ (right). `CARBayes()` results in black, `INLA` in blue. Vertical lines indicate mode (green), median (red), and mean (black), respectively. (See R-Code 8.7.)

Remark 8.2. There is a subtle difference in how the priors are specified. Because `S.CARbym()` works with variances, the prior distribution is inverse-gamma (shape, scale). In `R2openBUGS`, it is with inverse Gamma using shape and rate=1/scale. In `INLA`, internally `prior="loggamma"` is used, but the parameters are as for a gamma (shape, rate). See also LeBauer *et al.* (2013) for a link between `openBUGS` and R. ♡

Remark 8.3. As $\boldsymbol{\eta} = \boldsymbol{u} + \boldsymbol{v}$, the observations y_i depend on a sum of elements of the process model $(\boldsymbol{u}^\top, \boldsymbol{v}^\top)$. This is an implementation nuisance and thus `INLA` reparametrizes by setting $\boldsymbol{x}^\top = (\boldsymbol{u}^\top, \boldsymbol{\eta}^\top)$, as is commonly done (Gelfand *et al.*, 1995; Rue and Held, 2005). This reparametrization via $\boldsymbol{\eta} \mid \boldsymbol{u} \sim \mathcal{N}(\boldsymbol{u}, \mathbf{I}/\kappa_{\boldsymbol{v}})$ leads to

$$\pi(\boldsymbol{\eta} \mid \boldsymbol{u}, \kappa_{\boldsymbol{v}}) \propto \kappa_{\boldsymbol{v}}^{\frac{n}{2}} \exp\Big(-\frac{\kappa_{\boldsymbol{v}}}{2}(\boldsymbol{\eta} - \boldsymbol{u})^\top(\boldsymbol{\eta} - \boldsymbol{u})\Big) \tag{8.8}$$

and, instead of equation (8.7), we work with

$$\begin{aligned}\pi(\boldsymbol{x}, \boldsymbol{\kappa} \mid \boldsymbol{y}) &\propto \kappa_{\boldsymbol{v}}^{\alpha_{\boldsymbol{v}}+\frac{n}{2}-1} \kappa_{\boldsymbol{u}}^{\alpha_{\boldsymbol{u}}+\frac{n-1}{2}-1} \\ &\times \exp\Bigg(-\kappa_{\boldsymbol{v}}\beta_{\boldsymbol{v}} - \kappa_{\boldsymbol{u}}\beta_{\boldsymbol{u}} + \sum_{i=1}^{n}\big(y_i\eta_i - e_i\exp(\eta_i)\big) - \frac{1}{2}\boldsymbol{x}^\top \begin{pmatrix} \kappa_{\boldsymbol{u}}\mathbf{R} + \kappa_{\boldsymbol{v}}\mathbf{I} & -\kappa_{\boldsymbol{v}}\mathbf{I} \\ -\kappa_{\boldsymbol{v}}\mathbf{I} & \kappa_{\boldsymbol{v}}\mathbf{I} \end{pmatrix} \boldsymbol{x} \Bigg).\end{aligned} \tag{8.9}$$

As before, $\pi(\boldsymbol{x} \mid \boldsymbol{\kappa})$ is a GMRF, $\pi(\boldsymbol{x} \mid \boldsymbol{\kappa}, \boldsymbol{y})$ is not. ♡

Remark 8.4. The exact meaning of the number of iterations, thinning, and burn-in varies from package to package. We have chosen the parameters to guarantee a burn-in of 1000 (unthinned) and a thinned (by 10) sample of length 2000. ♡

Remark 8.5. Rue and Held (2005) use a slightly different approach for the MCMC steps. Here we discuss a conceptually simpler but computationally tougher version of the Gibbs sampler. ♡

8.4 Leroux Model

An alternative model for areal count data was introduced by Leroux *et al.* (1999). In contrast to the BYM model, it has only one random effect component. Without including an intercept or additional covariates, this random effect simply models the log-relative risk $\boldsymbol{\eta}$. To be consistent with the implementations, we set $\boldsymbol{u} = \boldsymbol{\eta}$. The separation of spatially structured and *iid* variance is controlled by an additional parameter λ. To be more specific, the same likelihood function as in equation (8.4) is used, and $\boldsymbol{u}$ is modeled by the GMRF

$$\pi(\boldsymbol{u} \mid \kappa, \lambda) \propto \det(\mathbf{Q}(\lambda))^{\frac{1}{2}} \exp\Big(-\frac{\kappa}{2}\boldsymbol{u}^\top \mathbf{Q}(\lambda)\boldsymbol{u}\Big), \tag{8.10}$$

where $\kappa > 0$ is a precision parameter, and det denotes the determinant. The parameter $\lambda \in (0, 1)$ defines the degree of the spatial dependency through $\mathbf{Q}(\lambda) = (1-\lambda)\mathbf{I} + \lambda\mathbf{R}$, where $\mathbf{R}$ is the "structure" matrix imposed by (8.5). With appropriate (uninformative) priors for κ and λ, we get the posterior distribution

$$\pi(\boldsymbol{u}, \kappa, \lambda) \propto \kappa^{\frac{n}{2}-1} \det(\mathbf{Q}(\lambda))^{\frac{1}{2}} \exp\Big(\boldsymbol{y}^\top\boldsymbol{u} - \boldsymbol{e}^\top\exp(\boldsymbol{u}) - \frac{\kappa}{2}\boldsymbol{u}^\top\mathbf{Q}(\lambda)\boldsymbol{u}\Big). \tag{8.11}$$

The hierarchical structure is shown in Figure 8.9.

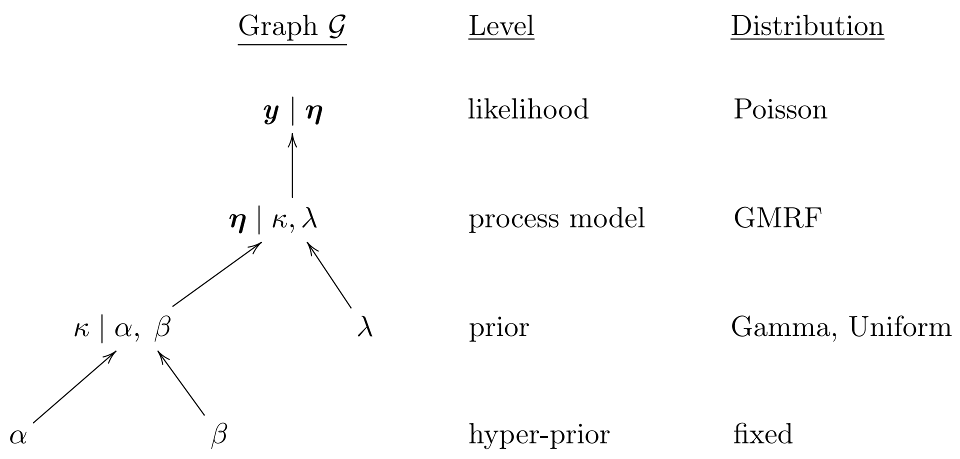


Figure 8.9: The variables (nodes) and their dependency structure are shown in Graph $\mathcal{G}$. The distributions and levels of the nodes in the model hierarchy are also indicated.

The R package *CARBayes* (Lee, 2013) provides the function `S.CARleroux()`, which implements a Gibbs sampler similar to the ones discussed here. Additionally, specific explanatory variables can be defined using a formula interface. The function comes with a mechanism that tunes the acceptance probability automatically. We run the function with the same settings as our implementations. As in other settings, we could specify prior distributions. The default values work nicely here. R-Code 8.8 illustrates the straightforward call to initiate the sampler and a diagnostic plot.

The variance parameters cannot be compared one-to-one to one from the classical BYM model. By disentangling the spatial component, it is possible to get somewhat comparable estimates between the Leroux model and the BYM one (see the end of R-Code 8.8).

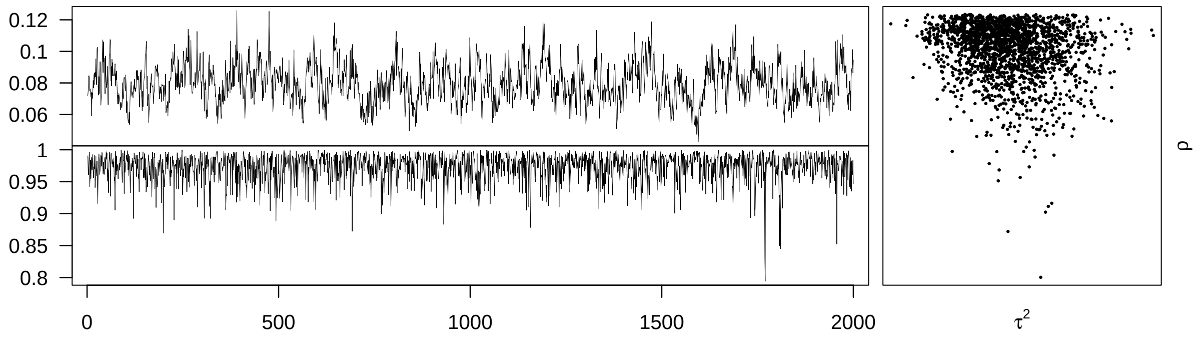


Figure 8.10: Diagnostic plots for the 2000 post burn-in and thinned samples from the `Leroux` MCMC BYM implementations are shown. Each panel contains trace plots for τ^2 (upper) and ρ (lower), respectively. The mixing of τ^2 and ρ is shown in a scatter plot (right). The chains were already thinned with a factor of 10. (See R-Code 8.8.)

R-**Code 8.8** Leroux model fitting and comparison with BYM. (See Figure 8.10.)

```
library(CARBayes)
L.out <- S.CARleroux(Y~offset(log(E)), data=Oral, family="poisson",
    W=as.matrix(A), n.sample=21000, burnin=1000, thin=10)

grid_trace2(cbind(L.out$samples$tau2, L.out$samples$rho))
openBUGS <- O.out$summary[c("kappaU","kappaV"), "mean"] # OpenBUGS, which
tmp <- L.out$summary.results[c("tau2","rho"), "Mean"] # "correspond" to
Leroux <- 1/c(tmp[2]*tmp[1], (1-tmp[2])*tmp[1])

cbind(openBUGS, Leroux)
##        openBUGS Leroux
## kappaU   13.637  12.87
## kappaV  230.148 455.11
```

Figure 8.11 compares the posterior median log-relative risks of the models discussed here. OpenBUGS and INLA have very close posterior medians. CARBayes seems to have minor differences with these latter two. The Leroux model is different. Hence, we should expect differences, as seen in the last two panels of Figure 8.11.

Note that slight variations may exist as packages may change their default arguments. It remains challenging to judge if slight differences are due to minor changes in the code (different sampling strategies, . . .), differences in the model specification (different parameterizations, hyperparameters, . . .), or even bugs in the code. We refer again to Gerber and Furrer (2015) for a more in-depth discussion on that point.

R-**Code 8.9** Comparison of log-relative risks. (See Figure 8.11.)

```
### Assembling the different posterior medians:
B.median <- B.u.median+B.inter.median
O.median <- apply(O.out$sims.array[,1,1:544+1], 2, median)
I.median <- I.out$summary.random$region.struct$`0.5quant`
L.median <- apply(L.out$samples$phi, 2, median) +
    median(L.out$samples$beta[,1])

### Plotting the posterior medians for the four approaches:
library(viridisLite)
col <- viridis(64)   # improved graph readability for readers with
         #  common forms of color blindness and/or color vision deficiency.
zl <- range(B.median, O.median, I.median, L.median)
germany.plot(B.median, col=col, zlim=zlim, main="CARBayes")
germany.plot(O.median, col=col, zlim=zlim, main="OpenBUGS")
germany.plot(I.median, col=col, zlim=zlim, main="INLA")
germany.plot(L.median, col=col, zlim=zlim, main="Leroux")

### Plotting a selection of of differences:
zl <- range(O.median-B.median, O.median-I.median, L.median-B.median,
    L.median-I.median)
germany.plot(O.median-B.median, col=col, zlim=zl, main="OpenBUGS-CARBayes")
germany.plot(O.median-I.median, col=col, zlim=zl, main="OpenBUGS-INLA")
germany.plot(L.median-B.median, col=col, zlim=zl, main="Leroux-CARBayes")
germany.plot(L.median-I.median, col=col, zlim=zl, main="Leroux-INLA")
```

8.5 Bibliographic Remarks

Waller *et al.* (1997); Leroux *et al.* (1999); Waller and Carlin (2010) are classical references to disease rate mapping. Lee (2011) compares conditional autoregressive models used in Bayesian disease mapping. In econometrics, many additional flavors of spatial area models are used; see, e.g., LeSage and Pace (2009).

There exist more sampling engines, e.g., BayesX (Brezger *et al.*, 2005) or ADMB (Fournier *et al.*, 2012), with R interfaces ***BayesX*** (Kneib *et al.*, 2011) and ***R2admb*** (Bolker and Skaug, 2012), or the package ***geoRglm*** (Christensen and Ribeiro, 2002), that can handle BHMs with spatially correlated random effects. Different flavors of spatial models can also be handled with the JAGS engine (Plummer, 2012) or with ***spBayes*** (Finley *et al.*, 2007, 2015). The community is highly active, as indicated by the CRAN task view *Bayesian Inference* (Park, 2014).

On certain Linux flavors, the file `OpenBUGS-3.2.3.tar.gz` cannot be unzipped with `tar zxvf ...` or `unzip ....` Using `7z x OpenBUGS-3.2.3.tar.gz` worked, however. Note that `gcc-multilib` was necessary.

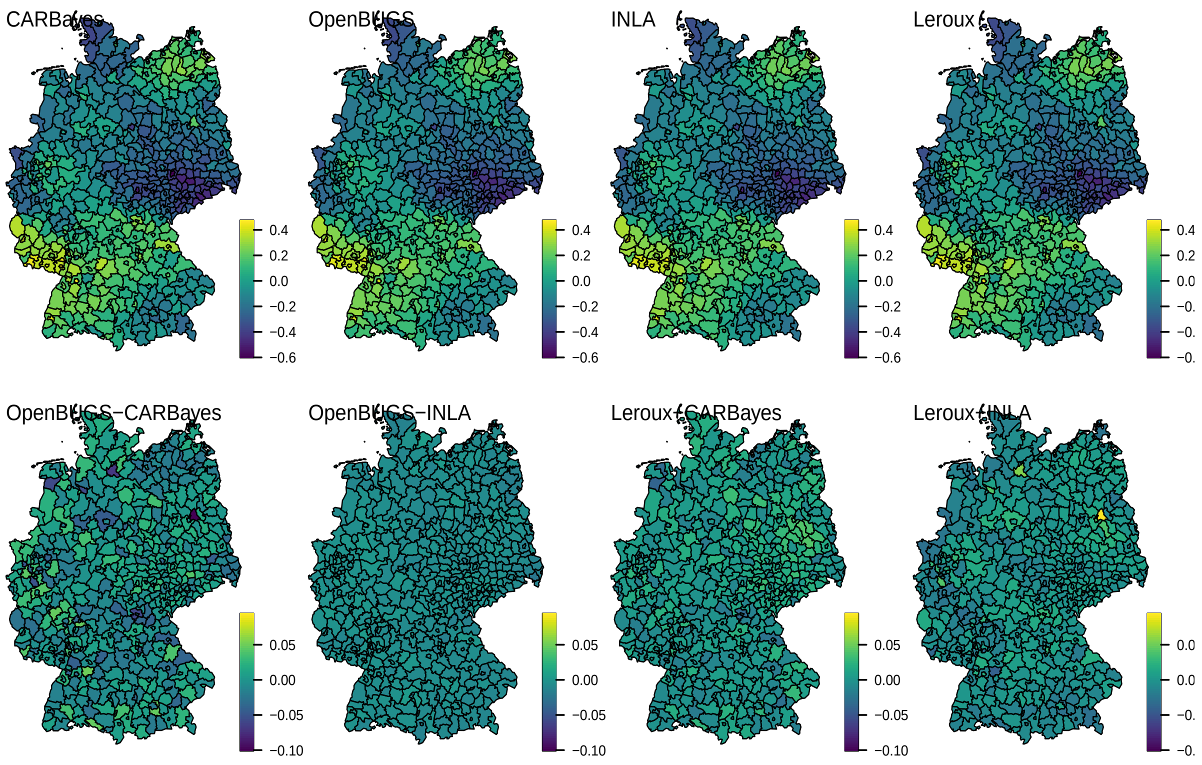


Figure 8.11: Top row: posterior medians of the estimated log-relative risk of the BYM models (CARBayes, OpenBUGS, INLA) and the Leroux model. Bottom row: the difference between the posterior medians of these log risks compared to CARBayes and INLA estimates. (See R-Code 8.9.)

Chapter 9

Spatial Processes: Concepts and ML Approaches

In this chapter, we introduce spatial processes. Such processes serve as a model template for georeferenced data, i.e., for data that can be observed at an arbitrary location within a predefined region. We focus on Gaussian processes, which can be specified by a mean and covariance function.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter09.R.

The set of random variables

$$\big\{Z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D} \subset \mathbb{R}^d,\, d \geq 1\big\} \tag{9.1}$$

is called a stochastic process (or random function, random field, random process or simply process, since we do not consider processes other than random processes) and is often simply denoted by $Z(\cdot)$. The field of modeling and analyzing data from such processes is typically referred to as geostatistics.

The random function $Z(\cdot)$ defines for every $\boldsymbol{s} \in \mathcal{D}$ a random variable $Z(\boldsymbol{s})$. The possible values of the random function $Z(\cdot)$ are often $\mathbb{R}$, $\mathbb{R}_+$, $[0, 1]$, or they are discrete.

The domain $\mathcal{D}$ can be seen as an index set, and it can be finite, discrete, or continuous. If $\mathcal{D} \subset \mathbb{R}^d$, $d \geq 1$ we call $Z(\cdot)$ a spatial process. In most applications, $\mathcal{D}$ has a bounded d-dimensional volume. In geostatistics, d is almost always one, two, or three. If $\mathcal{D}$ is intrinsically discrete and finite, the process is usually analyzed by lattice data methods (see Chapter 4.1). If $\mathcal{D}$ represents the time scale $Z(\cdot)$ is a time series and has to be treated differently to spatial processes, as the time is ordered by past, present, and future. One of the main concerns in analyzing spatial processes is prediction, referred to as interpolation. In time series analysis, the corresponding concepts were extrapolation or smoothing.

The random process (9.1) is often defined through the joint distribution of an arbitrary, finite set of points $\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_m\}$, namely

$$F_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_m}(v_1, \dots, v_m) = \mathrm{P}\big(Z(\boldsymbol{s}_1) \leq v_1, \dots, Z(\boldsymbol{s}_m) \leq v_m\big), \tag{9.2}$$

satisfying Kolmogorov's conditions of symmetry and consistency. These conditions basically impose that

- $F_{\cdot}(\cdot)$ is invariant under simultaneous permutation of $\boldsymbol{s}_i$ and v_i, i.e., let $\tau(\cdot)$ be a permutation of $\{1,\dots,m\}$ then $F_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_m}(v_1,\dots,v_m) = F_{\boldsymbol{s}_{\tau(1)},\dots,\boldsymbol{s}_{\tau(m)}}(v_{\tau(1)},\dots,v_{\tau(m)})$;
- $F_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_m}(v_1,\dots,v_l,\infty,\dots,\infty) = F_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_l}(v_1,\dots,v_l)$, $l < m$.

If the joint distribution (9.2) is a multivariate Gaussian distribution we call (9.1) a Gaussian process or a Gaussian random field (GRF).

A realization of the random function $Z(\cdot)$, denoted by $z(\cdot)$, is often called the sample function, the sample surface, or the sample path. In practice, we observe a set of the following form

$$\big\{z(\boldsymbol{s}_i) : \boldsymbol{s}_1,\dots,\boldsymbol{s}_n \in \mathcal{D}\big\} \tag{9.3}$$

denoting data measured at the locations $\boldsymbol{s}_1,\dots,\boldsymbol{s}_n$ in a domain $\mathcal{D} \subset \mathbb{R}^d$, $d \geq 1$.

We say that the locations $\boldsymbol{s}_i$ are equispaced in $\mathcal{D}$ if $\boldsymbol{s}_i = (\Delta i_1,\dots,\Delta i_d)^\top$, for some integers $i_1,\dots,i_d$ and a $\Delta > 0$. If the equispaced locations form a rectangular shape, we use the term regular grid.

The sample (9.3) can be seen as a subset of a potentially infinite number of measurements $\big\{z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D}\big\}$. The geostatistical modeling spirit is based on the assumption that $\big\{z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D}\big\}$ is a realization of the stochastic process (9.1).

9.1 Stationarity

To infer the statistical process (9.1) from a single set of observations (9.3), we need some restrictions (simplifications) on the process.

Definition 9.1. A process $Z(\cdot)$ is called a strong (or strictly) stationary process, if the equation

$$F_{\boldsymbol{s}_1,\dots,\boldsymbol{s}_m}(v_1,\dots,v_m) = F_{\boldsymbol{s}_1+\boldsymbol{h},\dots,\boldsymbol{s}_m+\boldsymbol{h}}(v_1,\dots,v_m) \tag{9.4}$$

holds for all m, $\boldsymbol{s}_1,\dots,\boldsymbol{s}_m$, $v_1,\dots,v_m$ and all $\boldsymbol{h}$. ♢

A weaker form of stationarity is second-order stationarity.

Definition 9.2. A process $Z(\cdot)$ is a second-order stationary process if the following moments exist and satisfy

$$\begin{aligned} \mathrm{E}\big(Z(\boldsymbol{s}_1)\big) &\equiv \mu, \\ \mathrm{Cov}\big(Z(\boldsymbol{s}_1), Z(\boldsymbol{s}_2)\big) &= c(\boldsymbol{s}_1 - \boldsymbol{s}_2), \quad \boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{D}. \end{aligned} \tag{9.5}$$

If $c(\cdot)$ is a function of $\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ only, then the process is called isotropic; otherwise, it is called anisotropic. We also use the term weak stationarity or wide-sense stationarity.

The function $c(\cdot)$ is called the covariance function. In case of an isotropic process, we write $c^\circ(h) = c(\boldsymbol{h})$, $h = \|\boldsymbol{h}\|$ and we call $c^\circ(\cdot)$ an isotropic covariance function. ♢

Stationarity is an assumption made by the statistician, and it cannot be proved with the data itself — at most, we could show that we cannot reject the hypothesis. In practice, for Gaussian processes, we estimate $c(\boldsymbol{h})$ and verify if the covariance tends to zero with increasing lag.

Remark 9.1. There are other practical concepts of stationarity. Journel and Huijbregts (1978) define quasi-stationarity as local stationarity. In practice, we suppose moving windows inside of which the process can be considered as stationary, and the data are sufficient to make statistical inference possible. Clearly, the precise statistical definition of this concept is vague. ♡

Remark 9.2. To be able to use the common statistical concepts, we do not only suppose that the process is stationary but also that it satisfies the property of ergodicity. This assumption allows, simply said, expectations to be estimated by spatial averages. An exact definition can be found in Adler (1981, Section 6.5). As for Gaussian processes, second-order stationarity and strong stationarity coincides, and a sufficient condition for ergodicity is $c(\boldsymbol{h}) \to 0$ as $\|\boldsymbol{h}\| \to \infty$ (Adler, 1981, Page 145).

According to Cressie (1989a), many statisticians use the ergodicity assumption only to guarantee that mean and covariance estimations are consistent. Processes satisfying this property are called weakly ergodic. We only consider such weakly ergodic processes. The literature on ergodicity in spatial processes is rather sparse, whereas for time series, the results concerning ergodicity are better known and used. ♡

9.1.1 Anisotropy

As seen in Definition 9.2 every non-isotropic process is called anisotropic. Traditionally, we distinguish zonal anisotropy and geometrical anisotropy (Journel and Huijbregts, 1978). It is called geometrical when it can be reduced to isotropy by a linear transformation and any other form of anisotropy as zonal. Geometric anisotropy implies ellipsoid contours of the covariance function. Such processes can be transformed into isotropic processes through a matrix multiplication: if the matrix $\mathbf{O}$ describes the mapping of the corresponding ellipsoids to circles, then the process $Z(\mathbf{O}\boldsymbol{s})$ is isotropic (Borgman and Chao, 1994; Eriksson and Siska, 2000).

9.1.2 Additive Decompositions

When applying statistical procedures, it is often necessary to decompose the process $Z(\cdot)$ into several parts in order to obtain stationary components of the original process. We propose the following decomposition

$$Z(\boldsymbol{s}) = \mu(\boldsymbol{s}) + Y(\boldsymbol{s}) + \varepsilon(\boldsymbol{s}), \tag{9.6}$$

where $\varepsilon(\boldsymbol{s})$ is an iid zero-mean white-noise, $Y(\boldsymbol{s})$ a zero-mean stationary spatial process with continuous covariance function. The non-stochastic term $\mu(\boldsymbol{s})$ is typically considered as the first moment or trend, i.e., $\mathrm{E}\big(Z(\boldsymbol{s})\big) = \mu(\boldsymbol{s})$.

The trend $\mu(\cdot)$ can be modeled via a parametric or a nonparametric approach. The former supposes that $\mu(\cdot)$ is a linear combination of some functions; we typically write the mean in a "regression format", $\mu(\boldsymbol{s}) = \boldsymbol{x}(\boldsymbol{s})^\top\boldsymbol{\beta}$. In a nonparametric setting, one often neglects the spatial dependency in the data and estimates first $\mu(\cdot)$ and then, in a second step, the spatial dependency structure is estimated from $\widetilde{Z}(\cdot) = Z(\cdot) - \widehat{\mu}(\cdot)$.

In any case, estimating $\mu(\cdot)$ is crucial and often involves subjective decisions.

9.1.3 Characterization Using Covariances

For second-order stationarity, we typically use the covariance function (9.5) to describe the spatial dependence structure of the process. Moreover, we will discuss the covariance for $Y(\cdot)$ as the error is a white-noise process.

A valid covariance has to be a positive (semi-)definite function in order to ensure

$$\operatorname{Var}\Big(\sum_{i=1}^n \alpha_i Y(\boldsymbol{s}_i)\Big) = \sum_{i,j=1}^n \alpha_i\alpha_j \operatorname{Cov}\big(Y(\boldsymbol{s}_i), Y(\boldsymbol{s}_j)\big) = \sum_{i,j=1}^n \alpha_i\alpha_j c(\boldsymbol{s}_i, \boldsymbol{s}_j) \geq 0. \tag{9.7}$$

for all non-zero $\boldsymbol{\alpha} \in \mathbb{R}^n$. Hence, we *cannot* pick an arbitrary function $c(\cdot)$ as a covariance function.

We often use the following isotropic covariance functions (for $h \geq 0$):

1. the spherical covariance

$$c^\circ(h; \theta_1, \theta_2) = \begin{cases} \theta_2\Big(1 - \frac{3}{2}\big(h/\theta_1\big) + \frac{1}{2}\big(h/\theta_1\big)^3\Big), & \text{if } 0 \leq h \leq \theta_1, \\ 0, & \text{if } h > \theta_1, \end{cases} \tag{9.8}$$

   with $\theta_1 > 0$, $\theta_2 > 0$.

2. the Wendland$_2$ covariance function

$$c^\circ(h; \theta_1, \theta_2) = \begin{cases} \theta_2(1 - h/\theta_1)^6\big(35(h/\theta_1)^2 + 18h/\theta_1 + 3\big)/3, & \text{if } h \leq \theta_1, \\ 0, & \text{if } h > \theta_1 \end{cases} \tag{9.9}$$

   with $\theta_1 > 0$, $\theta_2 > 0$.

3. the exponential covariance

$$c^\circ(h; \theta_1, \theta_2) = \theta_2 \exp(-h/\theta_1), \tag{9.10}$$

   with $\theta_1 > 0$, $\theta_2 > 0$.

4. the Matérn covariance (with known smoothness)

$$c^\circ(h; \theta_1, \theta_2) = \theta_2(h/\theta_1)^\nu \mathcal{K}_\nu(h/\theta_1), \tag{9.11}$$

   where $\mathcal{K}_\nu$ is the modified Bessel function of the second kind of order $\nu > 0$ (Abramowitz and Stegun, 1970) and with $\theta_1 > 0$, $\theta_2 > 0$. The smoothness parameter translates literally into the "smoothness" of the sample paths (realizations). More precisely, the inducing process $Y(\cdot)$ is m times mean square differentiable if and only if $\nu > m$.

   For certain ν the Matérn covariance function (10.7) has appealing forms. For example, if $\nu = 0.5$, it is an exponential semi-variogram; if $\nu = n + 0.5$ with n an integer, it is the product of an exponential covariance and a polynomial of order n.

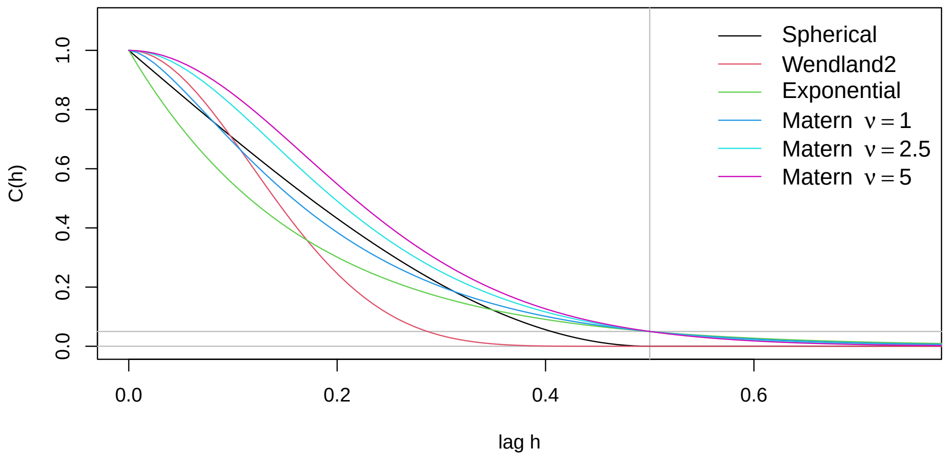


Figure 9.1: Different covariance functions given by equations (9.8) to (9.11) with nugget effect 0, (partial) sill 1 and (practical) range 0.5. (See R-Code 9.1.)

5. the Matérn covariance

$$c^\circ(h;\theta_1,\theta_2,\theta_3) = \theta_2 (h/\theta_1)^{\theta_3}\mathcal{K}_{\theta_3}(h/\theta_1), \tag{9.12}$$

   with $\theta_1 > 0$, $\theta_2 > 0$ and $\theta_3 > 0$.

In the parametrizations above, θ_1 is called the range or range parameter and determines how fast the covariance function decays to zero, i.e., determines the "extend" of the spatial dependency. The parameter θ_2 is called the sill and corresponds to the (marginal) variance of the process: $\theta_2 = c^\circ(0) = \text{Var}\big(Y(\boldsymbol{s})\big)$.

Historically, the squared exponential covariance or Gaussian covariance

$$c^\circ(h;\theta_1,\theta_2) = \theta_2 \exp(-h^2/\theta_1^2), \tag{9.13}$$

has been used quite often. There are several theoretical and practical reasons not to use it. Instead, a Matérn with a moderately high smoothness ($\nu = 2.5$ to $\nu = 4.5$) is typically sufficient.

The spherical and Wendland$_2$ covariance function have finite support $[0,\theta_1]$. For such covariance functions, we often use the term compactly supported. The exponential, Matérn, and squared-exponential covariance functions converge to zero asymptotically, i.e., have an infinite range. In such situations, we often use the term practical range, which is the distance at which the covariance function reaches 5% of its sill. The practical ranges of these covariance functions typically depend on the parameter θ_1. For the exponential covariance function, the practical range is $3\theta_1$.

Figure 9.1 and R-Code 9.1 give examples of the abovementioned covariance functions. For the Matérn covariance function, the range also depends on ν in a non-closed form expression. To overcome this issue, parameterized forms of the Matérn covariance function exist. The function `Matern.cor.to.range()` from the package `fields` can be used to determine the parameter θ_3 given a specific effective range `d` and smoothness `nu`.

R-Code 9.1 Covariance functions. (See Figure 9.1.)

```
library(spam) # for cov.xyz functions
plot(0, xlim=c(0,.75), ylim=c(0,1.1), type="n", xlab="lag h", ylab="C(h)")
abline(v=.5, h=c(.05,0), col="gray")
h <- seq(0, to=.8, length=100) # Lag vector
lines(h, cov.sph(h, c(.5, 1)), col=1) # theta=(range=.5, sill=1)
lines(h, cov.wend2(h, c(.5, 1)), col=2) #
lines(h, cov.exp(h, c(.5/3, 1)), col=3)
# and for the Matern c(range, sill, smoothness)
lines(h, cov.mat(h, c(.125, 1, 1)), col=4) # ranges determined
lines(h, cov.mat(h, c(.0845, 1, 2.5)), col=5) # empirically.
lines(h, cov.mat(h, c(.0618, 1, 5)), col=6)
legend("topright", bty="n",col=1:6, lty=1, cex=1.2,
       legend=c("Spherical", "Wendland2", "Exponential", expression(Matern
          ~~nu==1), expression(Matern~~nu==2.5), expression(Matern~~nu==5)))
# For a better range conversion:
fields::Matern.cor.to.range(d=0.5, nu=5, cor.target=.05)
## [1] 0.061787
```

The covariance function of a white noise process $\varepsilon(\cdot)$ is $c(\boldsymbol{h}) = 0$ for $\boldsymbol{h} \neq \mathbf{0}$ and $c(\mathbf{0}) = \theta_\varepsilon$ otherwise. In this setting, θ_ε is often referred to as the *nugget effect*. Hence, using subscripts to refer to the corresponding sub-process, we have

$$c_Z^\circ(h;\boldsymbol{\theta}_Z) = c_Y^\circ(h;\boldsymbol{\theta}_Y) + c_\varepsilon^\circ(h;\theta_\varepsilon) = \begin{cases} \theta_\varepsilon + c_Y^\circ(0;\boldsymbol{\theta}_Y), & \text{if } h = 0,\\ c_Y^\circ(h;\boldsymbol{\theta}_Y), & \text{if } h > 0, \end{cases} \tag{9.14}$$

for appropriate ${\boldsymbol{\theta}_Z}^\top = ({\boldsymbol{\theta}_Y}^\top, \theta_\varepsilon)$. Further, the sill of $\mathbf{Y}(\cdot)$ (say θ_2) is termed *partial sill* and $\theta_2 + \varepsilon$ is termed the *total sill*.

Example 9.1. Figure 9.2 shows realizations of Gaussian processes over a regular grid of 25×25 locations in the unit square. The figure is constructed using R-Code 9.2 as a template with varying parameters and covariance functions. The left column consists of the first five covariance functions of Figure 9.1. The second column varies the smoothness. The third column has different ranges, and the right-most column one varies the sill. The seed and color range (–3 to 2.7) have been kept the same for all panels for a better comparison.

The smoothness of a Wendland$_2$ is equivalent to a Matérn with smoothness $\nu = 2.5$. Therefore the second and last panels of the first column are comparable concerning smoothness. The range for the former seems to be slightly larger, which is somewhat counterintuitive with the apparent ranges of the red and cyan functions in Figure 9.1.

The larger the range, the smaller the marginal variance of the data. The last panel in the last row is from a spatial process with nominal marginal variance one whereas the variance of the observations (neglecting the spatial dependency is roughly 0.2.

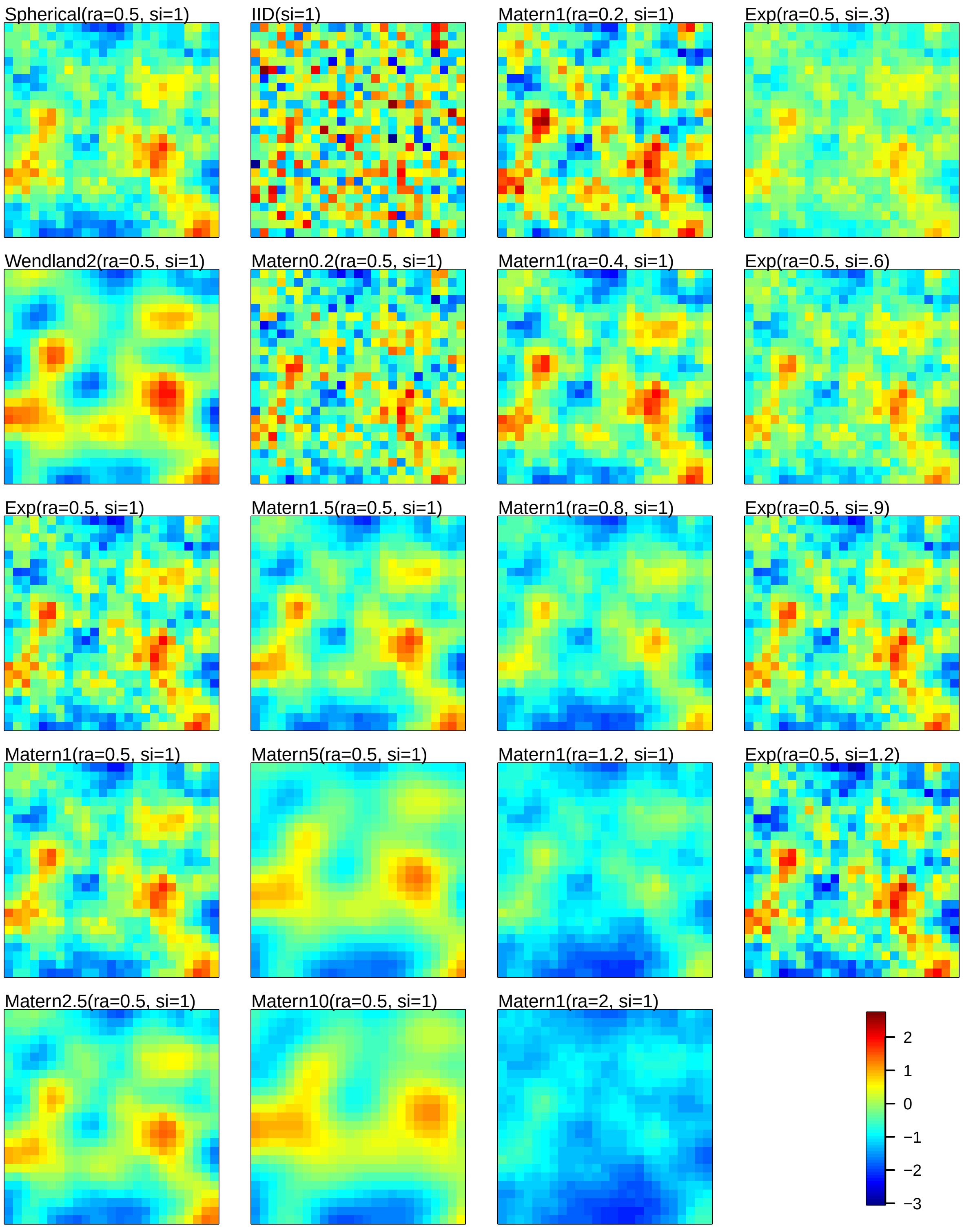


Figure 9.2: Realizations of a Gaussian process for different covariance functions. `Matern2.5` indicates a Matérn covariance function with smoothness $\nu = 2.5$; `ra` and `si` represent effective ranges and sills. The first column uses the first five covariances of Figure 9.1; the remaining vary smoothness, range, and sill parameters. All panels have been created with the same seed and plotted using the same scale. (See R-Code 9.2.)

Note that a covariance function with a practical range of zero or a Matérn with smoothness zero (and arbitrary range) is equivalent to an iid setting. ♣

R-Code 9.2: Gaussian realizations. (See Figure 9.2.)

```
m <- 25
xs <- seq(0, to=1, length=m)
gridp <- expand.grid(xs, xs)            # a fine grid
# vary the covariance matrix below.
set.seed(12)
y <- rmvnorm(1, sigma=cov.mat(rdist(gridp), c(0.105,1,1.5)) )
image.plot(matrix(y, nrow=m))
title("Matern1.5(ra=0.5, si=1)", adj=0, font.main=1, line=.15)
```

A valid isotropic covariance function in $\mathbb{R}^{d_1}$ may not be valid in $\mathbb{R}^{d_2}$ if $d_1 < d_2$, but the inverse always holds. The spherical covariance function is valid only up to $\mathbb{R}^3$. The Matérn covariance function is valid in all dimensions. If $c(\boldsymbol{h}) = \prod_{k=1}^d c_k(h_k)$, we call the covariance separable (which is thus not isotropic). In general, the families of valid covariance functions form convex cones.

Remark 9.3. The so-called Generalized Wendland covariance function gained much attention in the last few years. The function has smoothness parameters ν and κ, variance parameter θ_2 and range parameter θ_1 and is given by For

$$c^\circ(h;\theta_1,\theta_2) = \begin{cases} \dfrac{\theta_2}{\mathrm{B}(2\kappa,\nu+1)} \displaystyle\int_h^{\theta_1} w(w^2-t^2)^{\kappa-1}(\theta_1-w)^\nu dw, & \text{if } h \le \theta_1, \\ 0, & \text{if } h > \theta_1. \end{cases} \tag{9.15}$$

with $\mathrm{B}(\cdot,\cdot)$ the beta function. For technical details about valid parameter values, we refer to Bevilacqua *et al.* (2019).

The Generalized Wendland has many similarities with the Matérn covariance function. It also has closed-form expressions for integer parameter values of the smoothness parameter κ. The Wendland$_2$ covariance model (9.9) is a particular case of the Generalized Wendland with $\kappa = 2$ and $\nu = 4$. Moreover, the Matérn covariance is the limiting case of a particularly parametrized Generalized Wendland model (Bevilacqua *et al.*, 2022).

Finally, in R, the Generalized Wendland covariance function is provided by, e.g., the package ***GeneralizedWendland*** (Fischer *et al.*, 2022). ♡

Remark 9.4. Limiting cases for the covariance functions exist. For example, if the range parameter θ_1 tends to zero, the covariance function tends to the covariance function of a pure white-noise process.

For the Matérn model, $\nu \to 0$ leads to a white-noise process as well. In the limit, as $\nu \to \infty$ and appropriate scaling of θ_1 (as a function of ν) it is the squared exponential covariance (9.13). ♡

9.2 Maximum Likelihood (ML) Methods

We assume now that we have a parametric form of the random field. In typical situations, the underlying distribution is parameterized by some parameter $\boldsymbol{\beta}$ for the first moment, i.e., the mean structure, and by some parameter $\boldsymbol{\theta}$ for the second moment, i.e., the parameters of the covariance function.

Further, we assume Gaussian processes. As all finite-dimensional subsets of the process are Gaussian as well, maximum likelihood (ML) estimation methods can be used to estimate $\boldsymbol{\beta}$ and $\boldsymbol{\theta}$.

To derive the estimates, we assume that we observe at n locations $\boldsymbol{s}_1, \dots, \boldsymbol{s}_n$ a second-order stationary Gaussian random field. We write the n-dimensional distribution as

$$\mathbf{Z} \sim \mathcal{N}_n\big(\boldsymbol{\mu}(\boldsymbol{\beta}), \boldsymbol{\Sigma}(\boldsymbol{\theta})\big), \tag{9.16}$$

where $\boldsymbol{\mu}(\boldsymbol{\beta})$ is the vector with elements $\mathrm{E}\big(Z(\boldsymbol{s}_1)\big)$, and $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ is the covariance matrix with elements $\mathrm{Cov}\big(Z(\boldsymbol{s}_i), Z(\boldsymbol{s}_j)\big)$. The estimation of $(\boldsymbol{\beta}; \boldsymbol{\theta})$ is done by maximizing the likelihood, or, equivalently, by minimizing the negative of twice the log-likelihood

$$-2\ell(\boldsymbol{\beta}, \boldsymbol{\theta}; \boldsymbol{z}) = n\log(2\pi) + \log\det(\boldsymbol{\Sigma}(\boldsymbol{\theta})) + (\boldsymbol{z} - \boldsymbol{\mu}(\boldsymbol{\beta}))^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}(\boldsymbol{z} - \boldsymbol{\mu}(\boldsymbol{\beta})), \tag{9.17}$$

with $\boldsymbol{z}$ the n-vector of observations. Equation (9.17) can now be used to get the estimates $\{\widehat{\beta}_{\mathrm{ml}}, \widehat{\boldsymbol{\theta}}_{\mathrm{ml}}\}$. In R, a call to `optim()` can be used. Alternatively, the package `spam` provides the function `mle()`, `mle.nomean()`, as well as functions for sparse covariance matrices.

However, the joint estimation is not always ideal, and a profiling approach is to be preferred. Suppose we have a constant mean, i.e., $\boldsymbol{\mu}(\boldsymbol{\beta}) = \mu\mathbf{1}$, where $\mathbf{1}$ is a n-vector containing ones. If $\boldsymbol{\theta}$ is known, it is straightforward to show that

$$\widetilde{\mu}(\boldsymbol{\theta}) = \operatorname*{argmin}_{\mu} -2\ell(\mu, \boldsymbol{\theta}) = \big(\mathbf{1}^\top \boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{1}\big)^{-1}\mathbf{1}^\top\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\mathbf{Z}, \tag{9.18}$$

which is generalized least squares estimator. We can substitute $\widetilde{\mu}(\boldsymbol{\theta})$ into (9.17) and minimize $-2\ell\big(\widetilde{\mu}(\boldsymbol{\theta}), \boldsymbol{\theta}; \boldsymbol{z}\big)$ with respect to $\boldsymbol{\theta}$ only. We say that μ has been profiled from the log-likelihood and call the resulting term the profile log-likelihood. The pair

$$\widehat{\boldsymbol{\theta}}_{\mathrm{ml}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \; -2\ell\big(\widetilde{\mu}(\boldsymbol{\theta}), \boldsymbol{\theta}\big) \quad \text{and} \quad \widehat{\mu}_{\mathrm{ml}} = \widetilde{\mu}\big(\widehat{\boldsymbol{\theta}}_{\mathrm{ml}}\big) \tag{9.19}$$

is known as an estimated generalized least squares estimator (EGLSE).

A similar estimate can be derived if $\mathrm{E}(\mathbf{Z}) = \mathbf{X}\boldsymbol{\beta}$.

Remark 9.5. 1. The profile likelihood approach can be elaborated as follows. We write $\boldsymbol{\Sigma}(\boldsymbol{\theta}) = \sigma^2\boldsymbol{\Omega}(\boldsymbol{\vartheta})$, i.e., we write the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{\theta})$ of $\mathbf{Z}$ as a correlation matrix $\boldsymbol{\Omega}(\boldsymbol{\vartheta})$ and the total sill σ^2 and separate $\boldsymbol{\theta}^\top = (\sigma^2, \boldsymbol{\vartheta}^\top)$. For $\boldsymbol{\mu}(\boldsymbol{\beta}) = \mathbf{X}\boldsymbol{\beta}$ and for a given ϑ we have the closed-form solution

$$\widetilde{\beta}(\boldsymbol{\vartheta}) = \big(\mathbf{X}^\top\boldsymbol{\Omega}^{-1}(\boldsymbol{\vartheta})\mathbf{X}\big)^{-1}\mathbf{X}^\top\boldsymbol{\Omega}^{-1}(\boldsymbol{\vartheta})\boldsymbol{z}. \tag{9.20}$$

in analogy to (9.18). We substitute (9.20) into (9.17) and simplify to

$$-2\ell_p(\boldsymbol{\vartheta},\sigma^2;\boldsymbol{z}) = -2\ell(\widetilde{\boldsymbol{\beta}}(\boldsymbol{\vartheta}),\sigma^2,\boldsymbol{\vartheta};\boldsymbol{z}) = n\log(2\pi\sigma^2) + \log\det\boldsymbol{\Omega}(\boldsymbol{\vartheta}) + \frac{1}{\sigma^2}\boldsymbol{z}^\top\mathbf{P}(\boldsymbol{\vartheta})\boldsymbol{z}, \quad (9.21)$$

where $\mathbf{P}(\boldsymbol{\vartheta}) = \boldsymbol{\Omega}^{-1}(\boldsymbol{\vartheta}) - \boldsymbol{\Omega}^{-1}(\boldsymbol{\vartheta})\mathbf{X}(\mathbf{X}^\top\boldsymbol{\Omega}^{-1}(\boldsymbol{\vartheta})\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\Omega}^{-1}(\boldsymbol{\vartheta})$. The function $\ell_p(\boldsymbol{\vartheta},\sigma^2;\boldsymbol{z})$ is known as the profile log-likelihood function of $(\boldsymbol{\vartheta},\sigma^2)$ and can be optimized by numerical algorithms. In fact, the optimization of $\ell_p(\boldsymbol{\vartheta},\sigma^2)$ can be improved further. From (9.21), we get the closed-form solution

$$\widetilde{\sigma}^2(\boldsymbol{\vartheta}) = \frac{1}{n}\boldsymbol{z}^\top\mathbf{P}(\boldsymbol{\vartheta})\boldsymbol{z} \quad (9.22)$$

Hence, we proceed by

$$\begin{aligned}
\widehat{\boldsymbol{\vartheta}}_{\text{ml}} &= \operatorname*{argmin}_{\boldsymbol{\vartheta}} -2\ell\big(\widetilde{\boldsymbol{\beta}}(\boldsymbol{\vartheta}),\widetilde{\sigma}^2(\boldsymbol{\vartheta}),\boldsymbol{\vartheta};\boldsymbol{z}\big) && (9.23)\\
&= \operatorname*{argmin}_{\boldsymbol{\vartheta}} n\log\Big(2\pi\frac{1}{n}\boldsymbol{z}^\top\mathbf{P}(\boldsymbol{\vartheta})\boldsymbol{z}\Big) + \log\det\boldsymbol{\Omega}(\boldsymbol{\vartheta}) + n && (9.24)\\
&= \operatorname*{argmin}_{\boldsymbol{\vartheta}} n\log\big(\boldsymbol{z}^\top\mathbf{P}(\boldsymbol{\vartheta})\boldsymbol{z}\big) + \log\det\boldsymbol{\Omega}(\boldsymbol{\vartheta}), && (9.25)\\
\widehat{\sigma}^2_{\text{ml}} &= \widetilde{\sigma}^2\big(\widehat{\boldsymbol{\vartheta}}_{\text{ml}}\big), \qquad \text{and} \qquad \widehat{\boldsymbol{\beta}}_{\text{ml}} = \widetilde{\boldsymbol{\beta}}\big(\widehat{\boldsymbol{\vartheta}}_{\text{ml}}\big). && (9.26)
\end{aligned}$$

A severe disadvantage of the profiling approach is that it is highly costly to calculate the matrix $\mathbf{P}$ because several linear systems have to be solved.

2. In the ML approach illustrated above, we estimate the mean parameter(s) (here μ) and parameters describing the second-moment structure (here $\boldsymbol{\theta}$) jointly. While the former rarely causes estimation difficulties, the latter is difficult to estimate. The difficulty is due to the high dimensionality of $\boldsymbol{\theta}$ and the intrinsic correlation of its individual elements. A remedy is to use the restricted (or residual) maximum likelihood (REML) approach, in which the mean parameters are "differenced out". More precisely, we estimate $\boldsymbol{\theta}$ from $\mathbf{KZ}$ where the *contrast matrix* $\mathbf{K}$ is such that $\mathrm{E}(\mathbf{KZ}) = \mathbf{0}$. Note that the contrast matrix is not unique. The correlation can be addressed with reparametrizations of the covariance function (see also text before Remark 9.3).

3. Large datasets and large parameter vectors are the major drawbacks of an ML approach. Flavors of ML (REML, quasi-likelihood, composite likelihood, ...) partially mitigate this aspect. However, here we will not see these techniques in detail.

4. Another approach to deal with large datasets is to use sparse covariance matrices, i.e., covariance matrices containing many zeros. The vast number of zeros is exploited computationally with sparse linear algebra algorithms, as implemented in, e.g., *spam*. Typically, the range does not lead to sufficiently sparse matrices, or the covariances do not have compact support. There are essentially two ways to resort: *tapering*, i.e., direct multiplication of the covariance matrix with a compactly supported correlation function, or to *misspecification* of the covariance, i.e., we choose a compactly supported covariance function without estimating the range parameter but set it to some predefined value. While tapering has been the preferred setting in different scenarios, see, e.g., Furrer *et al.* (2006); Kaufman *et al.* (2008); Bachoc *et al.* (2020), direct misspecification is getting more attention now, e.g., Bevilacqua *et al.* (2019). ♡

9.3 Prediction for Gaussian processes

Suppose we wish to make predictions at a location not included in the observed locations, for example, to establish a pollution map. In that case, all measures will have to be taken into account in computing the predicted value. Of course, their contributions will be weighted by the strength of their correlation with the location of interest and should take into account the uncertainty of the observations.

In the case of GRF, we simply use the conditional expectation, as given by equation (1.19). However, the mean and covariance structures are hardly known, and we use plugin-estimates. That means that

$$p\big(Z(\boldsymbol{s}'_1);\boldsymbol{z}\big) = \widehat{\mu}(\boldsymbol{s}'_1) + \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{s}'_1;\boldsymbol{s}_1\dots\boldsymbol{s}_n}\widehat{\boldsymbol{\Sigma}}^{-1}_{\boldsymbol{s}_1\dots\boldsymbol{s}_n;\boldsymbol{s}_1\dots\boldsymbol{s}_n}(\boldsymbol{z}-\widehat{\boldsymbol{\mu}}) \tag{9.27}$$

is a prediction for $Z(\boldsymbol{s}'_1)$ The predictor is more accessible if we make a slight notation abuse, say subscript 'obs' for all observations and 'pred' for all locations we want to predict. For example, $\boldsymbol{\Sigma}_{\text{pred,obs}}$ is the matrix containing the covariances $\text{Cov}(Z(\boldsymbol{s}'_i), Z(\boldsymbol{s}_j))$. Hence, the predictor for locations $\boldsymbol{s}'_1,\dots,\boldsymbol{s}'_m$ is

$$p\big(Z(\boldsymbol{s}'_1),\dots,Z(\boldsymbol{s}'_m);\mathbf{Z}\big) = \widehat{\boldsymbol{\mu}}_{\text{pred}} + \widehat{\boldsymbol{\Sigma}}_{\text{pred,obs}}\widehat{\boldsymbol{\Sigma}}^{-1}_{\text{obs,obs}}(\mathbf{Z}-\widehat{\boldsymbol{\mu}}_{\text{obs}}). \tag{9.28}$$

The predictive distribution with plug-in estimates thereof is Gaussian, but the matrix $\widehat{\boldsymbol{\Sigma}}_{\text{pred,pred}} - \widehat{\boldsymbol{\Sigma}}_{\text{pred,obs}}\widehat{\boldsymbol{\Sigma}}^{-1}_{\text{obs,obs}}\widehat{\boldsymbol{\Sigma}}_{\text{obs,pred}}$ is not precisely the variance of the predictor: as we neglect the uncertainties in the estimates, we have a biased version thereof.

In the next Section, we give several examples of an ML estimation.

9.4 Examples

This section explores the ML approach and illustrates the general approach, culprits, and conditional simulation.

9.4.1 `Chicago03` Dataset

We start by showing that the likelihood surface may be flat for small datasets. We use the `Chicago03` dataset of the package `fields`, reporting the average daily ozone values for 20 Chicago monitoring stations over the period 3.6.1987–30.8.1987. (A similar dataset was formerly known as `ozone`). Note that the help of `Chicago03` states, "The lasting scientific value [of the data] is probably minimal."

R-Code 9.3 calculates the ML estimate for a spatial process with constant mean and a spherical covariance function with initial values `beta0=0` and `theta0=c(2,2)`. Figure 9.3 shows the very shallow likelihood surface. Note that the numerical result suggests a successful optimization. However, choosing a different starting value would lead to a different estimate for the range. This fact indicates that we do not have a robust spatial signal. On the other hand, the initial value for the mean is not crucial.

We work with "relative-negative-2-loglikelihood", which corresponds to the likelihood ratio statistic W and can be directly compared with $q\chi^2(p, 1-\alpha)$ to construct confidence regions of level $1-\alpha$ for the p parameters. Hence, we superimpose the `image.plot()` with a `contour(, add=TRUE)` call and have a direct assessment of the uncertainty of the parameter estimates.

R-**Code 9.3:** ML estimation with the very small dataset `ChicagoO3`. (See Figure 9.3.)

```
library(fields)
y <- ChicagoO3$y              # see help("ChicagoO3", package="fields")
X <- matrix(1, length(ChicagoO3$y), 1)
# quilt.plot(ChicagoO3$lon.lat,ChicagoO3$y, nx=15, ny=15); map("usa", add=T)
c(mean(y), var(y))                          # EDA.
## [1] 39.780 17.638
distmat <- as.matrix(dist(ChicagoO3$x))  # position of station
print(pars <- mle(y, X, distmat, cov.sph, beta0=0, theta0=c(2,2),
           thetalower=c(.01,.01), thetaupper=Inf)[c(1,2,4)])
## $par
## [1] 39.780  2.000 16.756
##
## $value
## [1] 113.13
##
## $convergence
## [1] 0
nr <- 39   # Define the resolution for range and sill. We use a different
ns <- 41   #   value to have a check to properly fill the matrices.
range <- seq(.001, to=11, length=nr)     # sequence of range and sills
sill <- seq(5, 40, length=ns)
grid <- expand.grid(range, sill)         # grid
n2ll <- apply(grid, 1, function(theta)   # fill with neg2log likeihood
    neg2loglikelihood(y, X, distmat, cov.sph, pars$par[1], theta))
reln2ll <- matrix(n2ll - pars$value, nr, ns) # almost as -min(n2ll)
image.plot(range, sill, reln2ll, zlim=c(0,12))
contour(range, sill, reln2ll, levels=qchisq(c(.7,.9), 2),
        labels=c("70%","90%"), labcex=1, add=TRUE)     # Confidence region
points(pars$par[2], pars$par[3], pch=20, col="white") # plot mle
```

9.4.2 Artificial Data with two Parameters

To further illustrate the difficulty in precisely estimating covariance parameters, we continue to work in a low-dimensional setting by assuming a zero mean and no nugget effect. We assume a Matérn model with a smoothness parameter of $\nu = 3/2$ and covariance parameter $\boldsymbol{\theta} = (0.12, 1)^\top$ representing range and sill, respectively. This means that the smoothness is known, and we only need to estimate the range and sill.

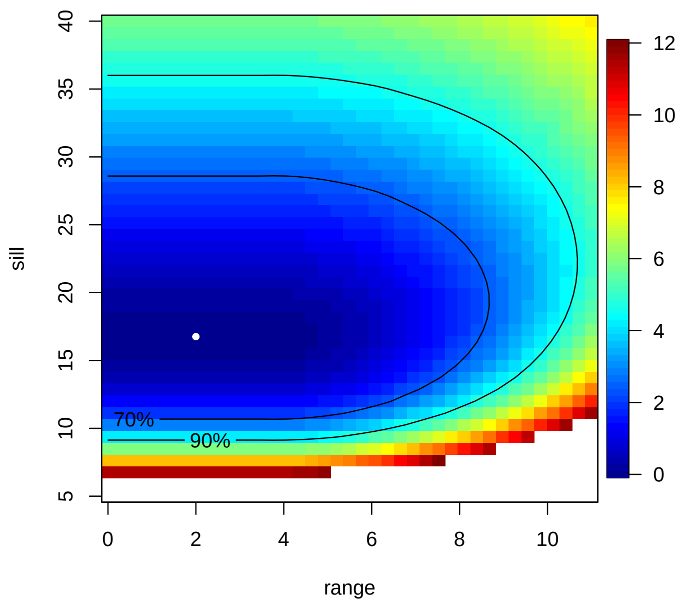


Figure 9.3: Surface plot of the relative (negative 2) log-likelihood. The ML estimate is indicated with the white dot. The contour lines give the 70% and 90% confidence regions. (See R-Code 9.3.)

The (artificial) sample, based on 70 unequally placed locations in the unit square, is shown in the left panel of Figure 9.4. Not surprisingly, changing the seed leads to drastically different estimates.

The following R-Code is very similar to R-Code 9.3 and consists of the following blocks: (i) generating the data, (ii) estimating the parameters, and (iii) visualizing the relative (negative 2) log-likelihood estimates. It is worth noting that the covariance parameters are highly correlated, as indicated by the "banana"-shaped likelihood surface in the right panel of Figure 9.4.

R-**Code 9.4:** Illustration of ML estimation with artificial data. (See Figure 9.4.)

```
library(fields)
library(spam)
n <- 70                                # number of data points
set.seed(16)                           # no particular seed
locs <- cbind(x=runif(n)^2,runif(n)^2) # non-uniform but arbitrary locations
truetheta <- c(.12, 1)                 # True theta=(range,sill)
### (i) We now define a grid, distance matrix, and a sample:
distmat <- rdist(locs) # distance matrix, same as "as.matrix(dist(locs))"
Sigma <- cov.mat32(distmat, truetheta) # true covariance matrix
y <- c(rmvnorm(1, sigma=Sigma))        # construct sample
c(mean(y), var(y))                     # sanity check!!
## [1] -0.10239  0.45857
quilt.plot(locs, y)                    # visualze data
### (ii) Here is the negative-2-loglikelihood and the mle:
```

```
neg2loglikelihood.nomean(y, distmat, cov.mat32, truetheta)
## [1] 51.188
res <- mle.nomean(y, distmat, cov.mat32,
    truetheta, thetalower=c(.02,.12), thetaupper=c(1,3), hessian=TRUE)
print(unlist( res[c(1,2,4)]))          # MLE, convergence ok
##        par1        par2       value convergence
##    0.098521    0.761616   49.477584    0.000000
thetahat <- res$par                    # for later use

### (iii) Construct fine grid to visualize neg2loglik for ranges and sills
nr <- 49
ns <- 45
range <- seq(.05, to=.2, length=nr)
sill <- seq(.16, 2.7, length=ns)
grid1 <- expand.grid(range, sill)      # this is the grid
n2ll1 <- apply(grid1, 1, function(theta)
    neg2loglikelihood.nomean(y, distmat, cov.mat32, theta))
reln2ll1 <- matrix(n2ll1 - res$value, nr, ns)
image(range, sill, reln2ll1, zlim=c(0,50), col=tim.colors())
cont <- contour(range, sill, reln2ll1, levels=qchisq(c(.7,.9), 2), lwd=2,
               labels=c("70%","90%"), labcex=1.2, add=TRUE, col='green')
points(res$par[1], res$par[2], pch=19, col="green")
```

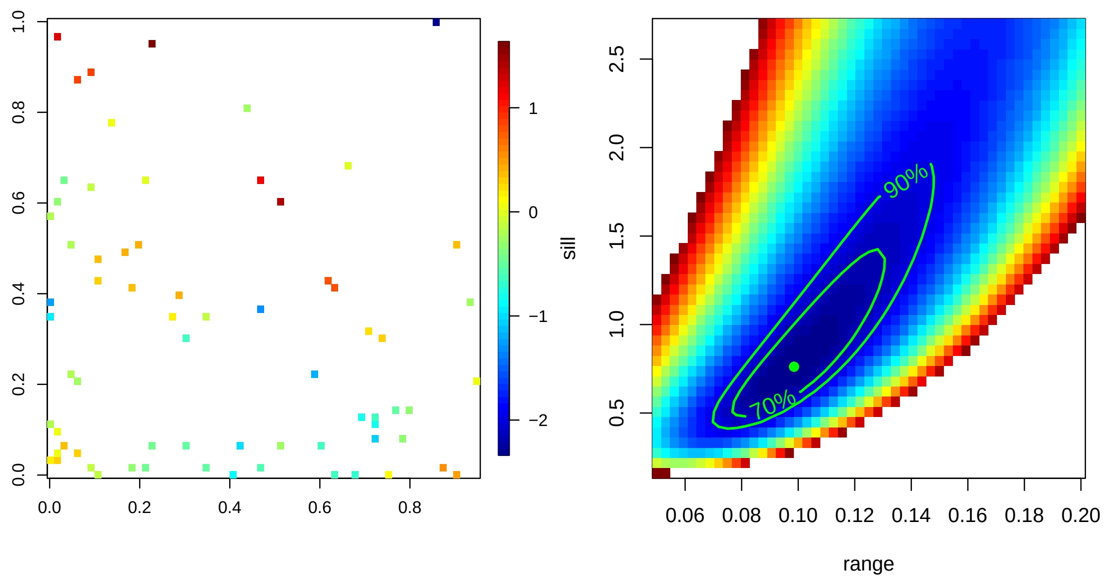


Figure 9.4: Left: artificial data based on 70 locations unequally placed locations in the unit square. Right: relative (negative 2) log-likelihood surface of realization. ML estimate and 70% and 90% confidence regions contour lines are given in green. (See R-Code 9.4.)

We now calculate multiple samples for the same set of locations and estimate the parameters. We demonstrate that the parameter estimates have a distribution similar to the log-likelihood. R-Code 9.5 consists of four blocks. In the first block, we sample 1000 spatial realizations while keeping the locations fixed. For each sample, we estimate the range and sill using the `mle.nomean()` function. The summary of the estimates reveals that none of the estimates are on the boundaries specified for `optim()`. However, a few realizations encounter convergence issues.

The second part plots two panels. In the left panel of Figure 9.5, we add the 1000 estimates to the (negative 2) log-likelihood plot (as shown in the right panel of Figure 9.4). The estimates are slightly shifted to the lower right compared to the confidence regions. This is not surprising since the estimates of the actual sample (0.099, 0.762) are smaller than the true values (0.12, 1). Estimates with convergence issues have been marked with a $+$ symbol. We observe no particular pattern in these points. The right panel of Figure 9.5 displays the contour lines of the bivariate density estimate based on the 1000 samples. To construct empirical confidence regions, a few more lines of code are required. We generate numerous contour lines and select those that correspond to the desired proportion of observations in the region (third block).

In the fourth block of the code, we compare the standard deviation and correlation of the estimates based on the sample with the standard deviation and correlation obtained from the Hessian matrix of the optimization process using the data. Since we are minimizing the negative log-likelihood, half of the Hessian corresponds to the observed information.

R-Code 9.5: Illustration of ML estimation with artificial data. (See Figure 9.5.)

```
### (i) Calculate MLE for many samples:
R <- 1000
set.seed(1)
ymat <- rmvnorm(R, sigma=Sigma)         # construct R samples
out <- t( apply(ymat, 1, function(y) {
    unlist(mle.nomean(y, distmat, cov.mat32,
    truetheta, thetalower=c(.02,.12), thetaupper=c(1,3))[c(1,2,4)]) }
  ))
sum(out[,"convergence"]>0)    # some "ERROR: ABNORMAL_TERMINATION_IN_LNSRCH"
## [1] 17
summary(out)     # nothing on the boundary, mean of estimates close to truth
##       par1                 par2              value         convergence
##  Min.   :0.0674   Min.   :0.271   Min.   : 1.55   Min.   : 0.000
##  1st Qu.:0.1071   1st Qu.:0.760   1st Qu.:33.49   1st Qu.: 0.000
##  Median :0.1190   Median :0.972   Median :41.56   Median : 0.000
##  Mean   :0.1206   Mean   :1.039   Mean   :40.97   Mean   : 0.884
##  3rd Qu.:0.1332   3rd Qu.:1.246   3rd Qu.:49.06   3rd Qu.: 0.000
##  Max.   :0.1944   Max.   :3.000   Max.   :78.78   Max.   :52.000
### (iia) Visualization by adding points
image(range, sill, reln2ll1, zlim=c(0,50), col=tim.colors())
```

```r
cont <- contour(range, sill, reln2ll1, levels=qchisq(c(.7,.9), 2), lwd=2,
                labels=c("70%","90%"), labcex=1.2, add=TRUE, col='green')
points(res$par[1], res$par[2], pch=19, col="green")
points(out[out[,"convergence"]==0,1:2], col='gray', pch=20, cex=.3)
points(out[out[,"convergence"]>0,1:2], col='white', pch="+")
### (iib) Visualize the estimates with empirical bivariate density:
library(MASS)
z <- kde2d(out[,1], out[,2], n=c(3*nr, 3*ns),
           lims=c(range(range), range(sill))) # default bandwith is ok
image.plot(z, col=tim.colors(41), xlab="range", ylab="sill")
points(out[,1:2], col='white', pch=20, cex=.3)
contour(range, sill, reln2ll1, levels=qchisq(c(.7,.9), 2),
        labels=c("70%","90%"), labcex=1, add=TRUE, lwd=2, col=3)
points(res$par[1], res$par[2], pch=19, col="green")

### (iii) long method to determine the approximate empirical confidence region:
library(splancs)
resol <- 100
level <- seq(1, to=30, length=resol)
contPoly <- contourLines(z, levels=level)
npip <- lapply(contPoly, function(x) nrow(pip(out[,1:2], x[2:3])) )
prop <- unlist(npip)/R   # length(prop)  should be of length resol!
getlevel <- sapply(c(.9,.7,.5), function(x) which.min(abs(x-prop)) )
contour(z, levels=level[getlevel], labels=c("90%", "70%", "50%"),
        labcex=1, add=TRUE, lwd=2, col='yellow')
### (iv) Compare Hessian with empirical standard dev and correlations:
iH <- solve(res$hessian/2)      # Inverse of observed Fisher information
### Standard deviations: empirical larger due to skewed data
rbind(FisherSE=sqrt(diag(iH)), empiricalSE=c(sd(out[,1]), sd(out[,2])))
##                 [,1]    [,2]
## FisherSE    0.016731 0.26428
## empiricalSE 0.019761 0.38500
### Correlation: strong dependency between range and smoothness!!!
c(Fisher=cov2cor(iH)[2], empirical=cor(out[,1:2])[2])   # surprisingly close
##    Fisher empirical
##   0.87333   0.88662
```

We conclude with some ideas about prediction and conditional simulation. The latter is an approach to estimate the MSE of prediction (which is more a measure of the sampling design than the uncertainty of the prediction). For very large datasets, the simulation approach is much faster than calculating the exact MSE. We use the same observations and fitted covariance parameters as in R-Code 9.7.

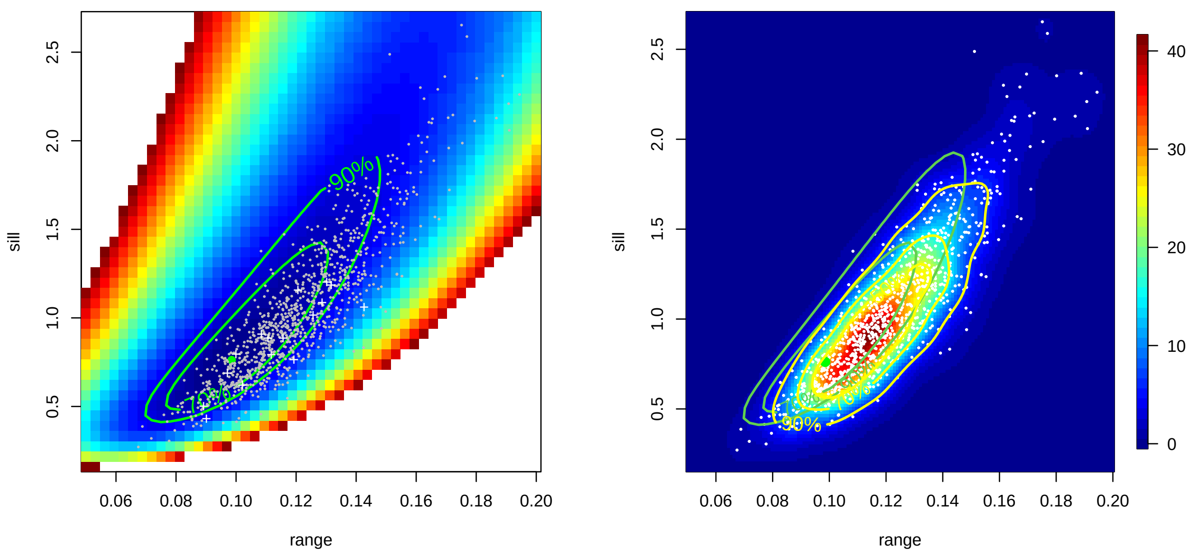


Figure 9.5: Left: Likelihood surface of realization with 1000 estimates. ML estimate and 70% and 90% confidence regions contour lines are given in green. Right: density estimates of the 1000 ML estimates. Empirical 70% and 90% confidence regions contour lines are given in yellow. For reference likelihood contour lines have been added in green.. (See R-Code 9.5.)

R-Code 9.6: Conditional expectation and simulation. (See Figure 9.6.)

```r
m <- 33
xs <- seq(0, to=1, length=m)
gridp <- expand.grid(xs, xs)      # a fine grid
Soo <- cov.mat(rdist(locs), thetahat)
Spp <- cov.mat(rdist(gridp), thetahat)
Sop <- cov.mat(rdist(locs, gridp), thetahat)
tmp <- solve(Soo, Sop)
Sigmacond <- Spp - t(Sop) %*% tmp    # From conditional distribution
mucond <- c(y %*% tmp)               # simplification because of zero mean
condsample <- rmvnorm(n=100, mean=mucond, sigma=Sigmacond)
zlim <- range(condsample)
for (i in 1:12) {
    image(matrix(condsample[i, ], m, m), col=tim.colors(64),
          xaxt="n", yaxt="n", zlim=zlim)
    points(locs, cex=.4, lwd=.5)
}
tmp1 <- matrix(apply(condsample, 2, mean), m, m)
image(tmp1, col=tim.colors(64), xaxt="n", yaxt="n", zlim=zlim,
                         main="Empirical mean", font.main = 1)
points(locs, cex=.4, lwd=.5)
```

```r
image(matrix(mucond, m, m), col=tim.colors(64), xaxt="n", yaxt="n",
                         zlim=zlim, main="True mean", font.main = 1)
points(locs, cex=.4, lwd=.5)
tmp2 <- matrix(apply(condsample, 2, var), m, m)
image(tmp2, col=tim.colors(64), xaxt="n", yaxt="n", zlim=c(0,1.1),
                         main="Empirical variance", font.main = 1)
points(locs, cex=.4, lwd=.5)
image(matrix(diag(Sigmacond), m, m), col=tim.colors(64),
      xaxt="n", yaxt="n", zlim=c(0,1.1), main="True variance", font.main = 1)
points(locs, cex=.4, lwd=.5)

image.plot(tmp1 - mucond, xaxt="n", yaxt="n", zlim=c(-.21,.21))
points(locs, cex=.4, lwd=.5)
image.plot(tmp2 - diag(Sigmacond), xaxt="n", yaxt="n")
points(locs, cex=.4, lwd=.5)
```

9.4.3 Artificial Data with three Parameters

We close this section by extending the previous example with unknown smoothness parameter. That means that we estimate three parameters. More specifically, we assume a Matérn model with parameter $\boldsymbol{\theta} = (0.12, 1, 1.5)^\top$ as range, sill, smoothness. The (artificial) sample is shown in Figure 9.7 left panel. The range and sill estimates are similar as in R-Code 9.4. The smoothness estimate is smaller than the truth, affecting somewhat the other two estimates.

We reiterate: the likelihood estimates are highly correlated, and covariance parameters are hard to estimate (see Figure 9.7 top right panel). The lower panels of Figure 9.7 show the bivariate surface plots of the relative (negative 2) log-likelihood where the third dimension has been fixed at the ML estimate. The R-Code is very much as in R-Code 9.4.

R-Code 9.7: Illustration of ML estimation with artificial data. (See Figure 9.7.)

```r
library(spam)
n <- 70
set.seed(16) # try other seeds, like set.seed(14)/ do not try (12)
locs <- cbind(x=runif(n)^2,runif(n)^2) # non-unform but arbitrary locations
truetheta <- c(.12, 1, 1.5)             # True theta=(range,sill,smoothness)
### We now define a grid, distance matrix, and a sample:
distmat <- rdist(locs) # distance matrix, same as "as.matrix(dist(locs))"
Sigma <- cov.mat(distmat, truetheta)    # true covariance matrix
y <- c(rmvnorm(1, sigma=Sigma))         # construct sample
c(mean(y), var(y))                     #  sanity check!!
## [1] -0.10239  0.45857
quilt.plot(locs, y)
```

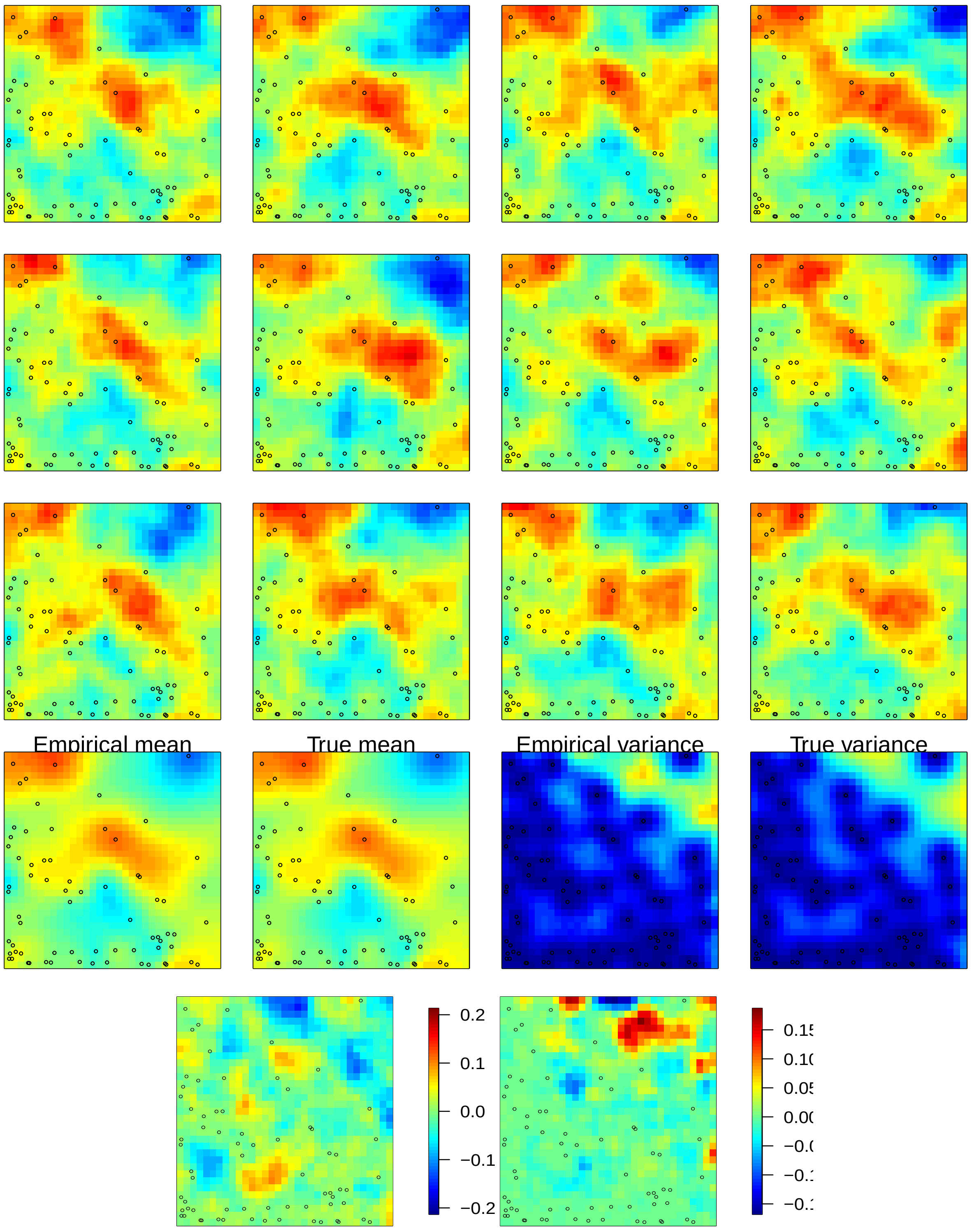


Figure 9.6: Illustration of conditional simulation. The top three rows show the realization of the spatial field, conditional on the observations shown in Figure 9.7. Forth row average shows the conditional mean and conditional variance, calculated based on the 100 samples and based on equation (1.19). The bottom row gives the difference between the empirical and exact of the former row. Note that the last panel uses a different scale. (See R-Code 9.6.)

```
### Here is the negative-2-loglikelihood:
neg2loglikelihood.nomean(y, distmat, cov.mat, truetheta)
## [1] 51.188
res <- mle.nomean(y, distmat, cov.mat,
    truetheta, thetalower=c(.02,.3,.08), thetaupper=c(2,4,3), hessian=TRUE)
print(unlist(res[c(1, 2, 4)]))
##        par1        par2        par3       value convergence
##     0.12196     0.83616     1.31777    49.12325     0.00000
iH <- solve(res$hessian/2)  # we are minimizing neg-2-log
cov2cor(iH)             # strong dependency between range and smoothness!!!
##          [,1]     [,2]     [,3]
## [1,]  1.00000  0.72983 -0.87120
## [2,]  0.72983  1.00000 -0.36641
## [3,] -0.87120 -0.36641  1.00000
thetahat <- res$par
h <- seq(1e-4, to=.8, length=100)
plot(h, cov.mat(h, truetheta), type="l", col=2, ylim=c(0,1.4), ylab="c(h)")

nr <- ns <- nm <- 11
appsd <- sqrt(diag(iH))         # arbitrary quantile, could be refined.
range <- seq(thetahat[1]-appsd[1], to=thetahat[1]+appsd[1], length=nr)
sill <- seq(thetahat[2]-appsd[2], to=thetahat[2]+appsd[2], length=ns)
smoothness <- seq(thetahat[3]-appsd[3], to=thetahat[3]+appsd[3], len=nm)
grid <- expand.grid(r=range, s=sill, m=smoothness)
# str(grid)
n2ll <- apply(grid, 1, function(theta) {
    neg2loglikelihood.nomean(y, distmat, cov.mat, theta) })

# out of the 11^3 possibilities, we illustrate all that have a ...
sel <- (n2ll < min(n2ll)+qchisq(.9, df=3)) # ... n2ll within qchisq(, df=3)
sum(sel)
## [1] 543
for (i in 1:sum(sel))    {
    lines(h, cov.mat(h, unlist((grid[sel,])[i,])), col="gray60")    }
lines(h, cov.mat(h, thetahat), col=4)
lines(h, cov.mat(h, truetheta), col=2)

nr <- 39
ns <- 41
nt <- 37
```

```
range <- seq(.05, to=.3, length=nr)
sill <- seq(.2, 3, length=ns)
smoo <- seq(.6, 2.4, length=nt)
grid1 <- expand.grid(range, sill) # 3 different grids
grid2 <- expand.grid(range, smoo)
grid3 <- expand.grid(sill,smoo)

n2ll1 <- apply(grid1, 1, function(theta)
    neg2loglikelihood.nomean(y, distmat, cov.mat, c(theta, res$par[3])))
reln2ll1 <- matrix(n2ll1 - res$value, nr, ns)
image(range, sill, reln2ll1, zlim=c(0,50), col=tim.colors())
contour(range, sill, reln2ll1, levels=qchisq(c(.7,.9), 3), col=3,
        labels=c("70%","90%"), labcex=1, add=TRUE)
points(res$par[1], res$par[2], pch=20, col="white")
n2ll2 <- apply(grid2, 1, function(theta) neg2loglikelihood.nomean(y,
                    distmat, cov.mat, c(theta[1], res$par[2],theta[2])))
reln2ll2 <- matrix(n2ll2 - res$value, nr, nt)
image(range, smoo, reln2ll2, zlim=c(0,50), col=tim.colors())
contour(range, smoo, reln2ll2, levels=qchisq(c(.7,.9), 3), col=3,
        labels=c("70%","90%"), labcex=1, add=TRUE)
points(res$par[1], res$par[3], pch=20, col="white")
n2ll3 <- apply(grid3, 1, function(theta)
    neg2loglikelihood.nomean(y, distmat, cov.mat, c(res$par[1], theta)))
reln2ll3 <- matrix(n2ll3 - res$value, ns, nt)
image(sill, smoo, reln2ll3, zlim=c(0,50), col=tim.colors())
contour(sill, smoo, reln2ll3, levels=qchisq(c(.7,.9), 3), col=3,
        labels=c("70%","90%"), labcex=1, add=TRUE)
points(res$par[2], res$par[3], pch=20, col="white")
```

R-Code 9.8 contains further visualization ideas for the three-dimensional likelihood. The plots themselves are not included here.

R-**Code 9.8:** Visualizing ML estimates.

```
### The plots are not included in the script!!
### Create a data cube and visualize slices:
an2ll <- array(n2ll, c(nr, ns, nm))
zlim <- range(n2ll) # zlim <- min(n2ll)+c(0,qchisq(.9, df=3))
par(mfrow=c(3,4), mai=c(.1,.1,.1,.1))
for (i in 1:10) {
    image(range, sill, an2ll[,,i], zlim=zlim, col=tim.colors(), xaxt="n",
          yaxt="n")
    points(thetahat[1], thetahat[2], pch=20, col="white")
}
image.plot(range, sill, an2ll[,,i], zlim=zlim, xaxt="n", yaxt="n")
### Alternative with "rgl":
library(rgl)
open3d()
rgl.points(scale(grid[,1]), scale(grid[,2]), scale(grid[,3]),
           col=tim.colors(64)[cut(n2ll, 64)])
sel <- (n2ll < min(n2ll)+qchisq(.9, df=3))
open3d()
rgl.points(scale(grid[sel,1]), scale(grid[sel,2]), scale(grid[sel,3]),
           col=tim.colors(64)[cut(n2ll[sel], 64)])
```

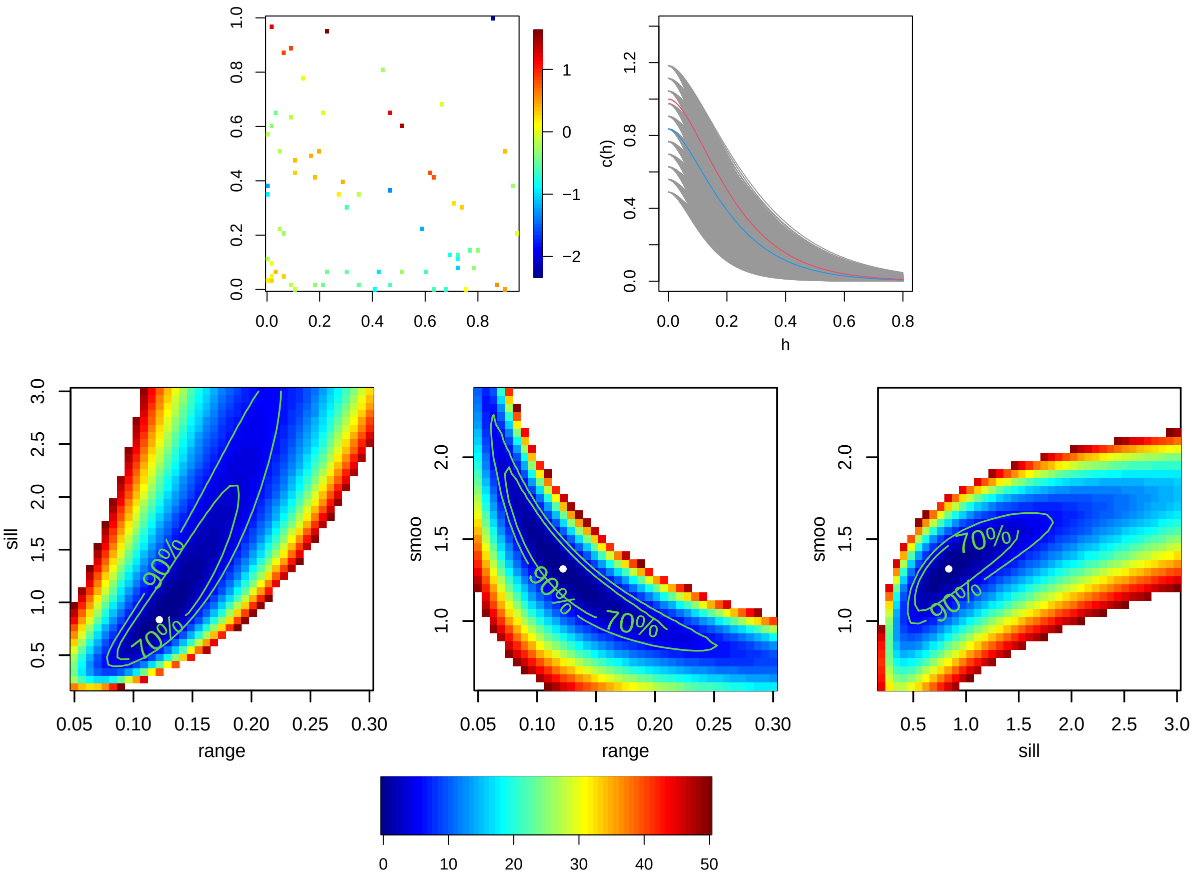


Figure 9.7: Top row: (artificial) data and "plausible" covariance functions. Bottom row: bivariate surface plots of the relative (negative 2) log-likelihood, where the third parameter has been fixed at the ML estimate, which is indicated with the white dot. The contour lines give the 70% and 90% confidence regions. (See R-Code 9.7.)

9.5 *Spatial Dependency as Regularization

We now give some technical detail about the ML estimation if the mean is known. Based on equation (9.17) The estimation of $\boldsymbol{\theta}$ is done by minimizing the (normed) log-likelihood

$$-\frac{2}{n}\ell(\boldsymbol{\theta}) - \log(2\pi) = \frac{1}{n}\log\det(\boldsymbol{\Sigma}(\boldsymbol{\theta})) + \frac{1}{n}\boldsymbol{z}^\top\boldsymbol{\Sigma}(\boldsymbol{\theta})^{-1}\boldsymbol{z} =: \ell_1(\boldsymbol{\theta}) + \ell_2(\boldsymbol{\theta}). \tag{9.29}$$

We look at both terms $\ell_1(\boldsymbol{\theta})$, $\ell_2(\boldsymbol{\theta})$ in (a) a conceptual, theoretical framework and (b) a numerical framework with two explicit realizations and show that the dependency can be interpreted as a regularization of the estimates.

For the first case, we take $c(h;\theta) = \exp(h/\theta)$ for the underlying covariance function with an equally spaced grid in $[0,1]$. The resulting covariance structure has very nice properties, in the sense that it can be associated to an AR(1) model and thus have a tridiagonal inverse (see Section 2.3.1 and take $\theta = -\Delta\log(\phi)$ with Δ the spacing of the adjacent locations). For "arbitrary" locations, if $\theta \to 0$, $\boldsymbol{\Sigma} \to \mathbf{I}$ (complete independence) and if $\theta \to \infty$, $\boldsymbol{\Sigma} \to \mathbf{1}\mathbf{1}^\top$ (complete dependence). The matrix $\mathbf{1}\mathbf{1}^\top$ has rank one and thus is not positive definite. Hence, we propose to work with the approximation $\mathbf{A} = \lambda\mathbf{I} + (1-\lambda)\mathbf{1}\mathbf{1}^\top$, for $\lambda > 0$ very small and possibly depending on n. The inverse of $\mathbf{A}$ exists and is

$$\mathbf{A}^{-1} = \frac{1}{\lambda}\mathbf{I} - \frac{1-\lambda}{\lambda(\lambda+n-\lambda n)}\mathbf{1}\mathbf{1}^\top \tag{9.30}$$

(by the Sherman–Morrison formula). The spectra of $\mathbf{A}$ and $\mathbf{A}^{-1}$ are $\{n-(n-1)\lambda, \lambda, \dots, \lambda\}$ and $\{1/\lambda, \dots, 1/\lambda, 1/(n-(n-1)\lambda)\}$, respectively. Hence, it would be natural to choose $\lambda = 1/n$. Abusing notation, we have for large n

$$\ell_1(0) = \frac{1}{n}\log(1) = 0, \tag{9.31}$$

$$\ell_1(\infty) \approx \frac{1}{n}\log(\lambda^{-(n-1)}/(n-(n-1)\lambda) \approx -\log(\lambda) - \frac{\log(n(1-\lambda))}{n} \quad \underset{n\to\infty}{\overset{\lambda=1/n}{\longrightarrow}} \quad -\log(n), \tag{9.32}$$

$$\ell_2(0) = \frac{1}{n}\boldsymbol{z}^\top\boldsymbol{z} = \widehat{\sigma}^2, \tag{9.33}$$

$$\ell_2(\infty) \approx \frac{1}{n}\boldsymbol{z}\boldsymbol{z}^\top\frac{1}{\lambda} - \frac{1-\lambda}{n\lambda(\lambda+n-\lambda n)}(\mathbf{1}^\top\boldsymbol{z})^2 = \widehat{\sigma}^2\frac{1}{\lambda} \quad \underset{n\to\infty}{\overset{\lambda=1/n}{\longrightarrow}} \quad \widehat{\sigma}^2 n, \tag{9.34}$$

where we used $\widehat{\sigma}^2 = \boldsymbol{z}^\top\boldsymbol{z}/n$ and, as the mean is zero, assumed $\mathbf{1}^\top\boldsymbol{z} = 0$.

Hence, we have an heuristic illustration that $\ell_1(\theta)$ decreases with increasing dependency and $\ell_2(\theta)$ increases with increasing dependency. An optimization based on the quadratic form would lead to an estimate that has no or minimal dependency. The log-determinant of the log-likelihood can be seen as a regularizer, adding a dependency.

We look at both terms for two specific datasets. Figure 9.8 is for a one-dimensional setting (equispaced), and Figure 9.9 is for the dataset of Figure 9.7. For the former, the results are coherent with the theoretical derivation. For the latter, the interpretation is a bit more delicate. The log-det term chooses a model with low variance and high correlation, and the quadratic term is a model with high variance and low correlation. Of course, for a fixed value of the sill (e.g., $\theta_2 = 1$), the "cuts" through the panel surfaces are comparable with Figure 9.8.

To minimize $\boldsymbol{z}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{z}$ for arbitrary positive definite and symmetric matrices, we write $\boldsymbol{\Sigma}^{-1} =$

$\mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a Cholesky factor. Writing $\mathbf{G} = \mathbf{L}^T\boldsymbol{z}$ with elements g_{ij}, we have

$$\boldsymbol{z}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{z} = \operatorname{tr}(\boldsymbol{z}^\top\mathbf{L}\mathbf{L}^T\boldsymbol{z}) = \operatorname{tr}(\mathbf{G}^\top\mathbf{G}) = \sum_{i=1}^{n}\sum_{j=1}^{n} g_{ij}g_{ji} = \sum_{i=1}^{n} g_{ii}^2 + \sum_{i=2}^{n}\sum_{j=1}^{j<i} g_{ij}^2. \tag{9.35}$$

Both square terms have to be minimized, the first via the sill, and the second via the range parameter.

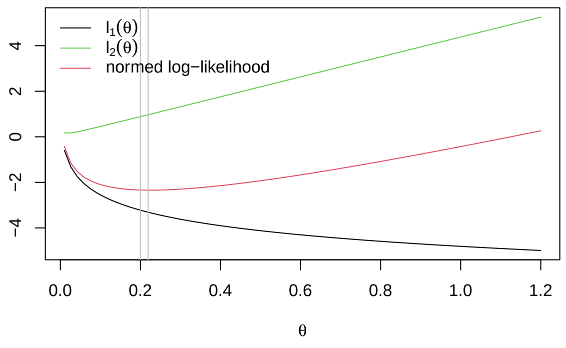


Figure 9.8: Decomposition of the log-likelihood in two components as a function of the range. Equispaced locations on $[0, 1]$ with exponential covariance where a larger range indicates stronger dependency. Vertical lines indicate the estimate and the truth.

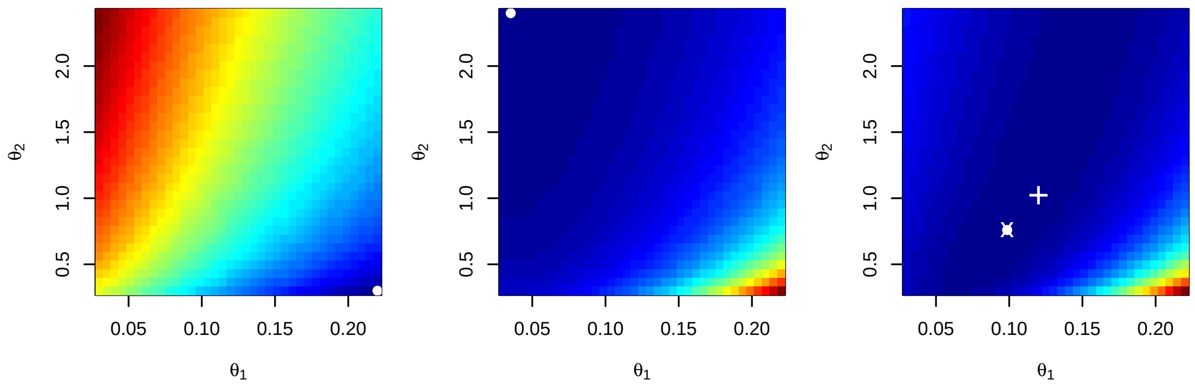


Figure 9.9: Image maps of $\ell_1(\boldsymbol{\theta})$, $\ell_2(\boldsymbol{\theta})$ and their sum (i.e., the normed log-likelihood (9.29)). The x-axes represents the range parameter; the y-axis the sill. We work with the Matérn covariance function with $\nu = 1.5$. The white dot indicates the pixel with the lowest value. The cross and the plus signs indicate the ML estimate and the true parameter value. The panels have different plotting ranges; blue indicates low values and red indicates high ones.

Chapter 10

Spatial Processes: The Classical Approach

Much spatial methodology has been developed outside statistics at times when computing resources were not available. These two aspects led to method-of-moment-based estimation and prediction approaches with somewhat particular terminology. This approach has been the historically preferred one. R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter10.R.

10.1 Intrinsic Stationarity and the Variogram

Chapter 9 considers a likelihood approach for stationary Gaussian spatial processes. We will not only relax the assumption of Gaussianity but also of weak stationarity as the latter may be too restrictive for some applications. We use the following form of stationarity.

Definition 10.1. A process $Z(\cdot)$ satisfying

$$\mathrm{E}\big(Z(\boldsymbol{s}_1)\big) \equiv \mu, \tag{10.1}$$

$$\mathrm{Var}\big(Z(\boldsymbol{s}_1) - Z(\boldsymbol{s}_2)\big) = 2\gamma(\boldsymbol{s}_1 - \boldsymbol{s}_2), \quad \boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{D}, \tag{10.2}$$

is said to be intrinsically stationary (or to satisfy the intrinsic hypothesis). If $\gamma(\cdot)$ is a function of $\|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ only, then the process as well as $\gamma(\cdot)$ is called isotropic, otherwise anisotropic. $\diamondsuit$

The functions $2\gamma(\cdot)$ and $\gamma(\cdot)$ defined by (10.2) are called the variogram and the semi-variogram respectively; a term coined by Matheron (1962). The difference $\boldsymbol{h} = \boldsymbol{s}_1 - \boldsymbol{s}_2$ is called the (spatial) lag, and we let $h = \|\boldsymbol{h}\|$, where $\|\cdot\|$ is some distance, e.g., the Euclidean or great circle distance.

Strong stationarity implies second-order stationarity if the second moment of $Z(\cdot)$ is finite. Second-order stationarity implies intrinsic stationarity. For the inverse to hold, we require that $\gamma(\boldsymbol{h})$ is bounded for all $\boldsymbol{h}$. The intrinsic hypothesis is essentially second-order stationarity for the first-order difference $Z(\cdot + \boldsymbol{h}) - Z(\cdot)$.

For every intrinsically stationary process, the variogram exists and is uniquely defined. However not for every function $\gamma(\cdot)$ there exists a process for which $2\gamma(\cdot)$ is its variogram. For this to hold, the function $\gamma(\cdot)$ has to satisfy certain properties. Suppose locations $\boldsymbol{s}_1, \dots, \boldsymbol{s}_n \in \mathcal{D}$ and scalars α_i, $i = 1, \dots, n$ such that $\sum_i \alpha_i = 0$. If the process $Z(\cdot)$ is intrinsically stationary we have that

$$2\operatorname{Var}\Big(\sum_{i=1}^{n} \alpha_i Z(\boldsymbol{s}_i)\Big) = -\sum_{i,j=1}^{n} \alpha_i \alpha_j 2\gamma(\boldsymbol{s}_i - \boldsymbol{s}_j) \geq 0. \tag{10.3}$$

This means that any continuous and conditionally negative definite function corresponds to an intrinsically (possibly complex-valued) stationary stochastic process and vice versa (Cressie, 1993, Page 87).

If $\gamma(\boldsymbol{h}) \to \theta =: \theta_3 > 0$ as $\boldsymbol{h} \to \boldsymbol{0}$ then θ is called a nugget effect by Matheron (1962). See also Cressie (1993, Page 59 and Paragraph 3.2.1) for a detailed discussion of the nugget effect. The variogram of a process $Z(\cdot)$ is often parametrized with two more parameters: θ_2 called the partial sill (if it exists), where $\theta_2 + \theta_3$ is the maximum value of the semi-variogram; and θ_1 is related to the (practical) range, which is the value where the variogram attains its (practical) maximum. Note that the partial sill θ_2 of a second-order stationary process always exists, whereas the range can be infinite. In this case, we often use the term practical range, which is defined as the distance at which the variogram is at 95% of the practical sill. The terminology of the sill is not unique; some authors use the term sill for $\theta_2 + \theta_3$.

If we knew the process explicitly, we could calculate the underlying variogram (e.g., Armstrong and Diamond, 1984, for some examples). As this is rarely the case, we assume the process has an underlying parameterized variogram.

Consider the following most widely used isotropic semi-variograms, parameterized using range, partial sill (where applicable), and nugget effect parameter ($\theta_i \geq 0$, $i = 1, 2, 3$).

1. If the process is a pure nugget effect, its semi-variogram is defined by

$$\gamma^{\circ}\big(h; \theta\big) = \begin{cases} 0, & \text{if } h = 0, \\ \theta, & \text{otherwise,} \end{cases} \tag{10.4}$$

   where $\theta > 0$. This model is equivalent to a pure white-noise or purely random process and corresponds to a process for which $Z(\boldsymbol{s}_i)$ and $Z(\boldsymbol{s}_j)$ are uncorrelated for all $\boldsymbol{s}_i \neq \boldsymbol{s}_j$.

2. spherical semi-variogram (valid in $\mathbb{R}^d$, $d = 1, 2, 3$):

$$\gamma^{\circ}\big(h; \theta_1, \theta_2, \theta_3\big) = \begin{cases} 0, & \text{if } h = 0, \\ \theta_3 + \theta_2\Big(3/2\big(h/\theta_1\big) - 1/2\big(h/\theta_1\big)^3\Big), & \text{if } 0 < h \leq \theta_1, \\ \theta_3 + \theta_2, & \text{otherwise;} \end{cases} \tag{10.5}$$

3. exponential semi-variogram (valid in $\mathbb{R}^d$, $d > 1$):

$$\gamma^{\circ}\big(h; \theta_1, \theta_2, \theta_3\big) = \begin{cases} 0, & \text{if } h = 0, \\ \theta_3 + \theta_2\big(1 - \exp(-h/\theta_1)\big), & \text{otherwise;} \end{cases} \tag{10.6}$$

4. Matérn semi-variogram with known smoothness (valid in $\mathbb{R}^d$, $d > 1$):

$$\gamma^\circ\big(h;\theta_1,\theta_2,\theta_3\big) = \begin{cases} 0, & \text{if } h = 0, \\ \theta_3 + \theta_2\big(1 - (h/\theta_1)^\nu \mathcal{K}_\nu(h/\theta_1)\big), & \text{otherwise.} \end{cases} \tag{10.7}$$

   where $\mathcal{K}_\nu$ is the modified Bessel function of the second kind of order $\nu > 0$ (Abramowitz and Stegun, 1970).

   Comments made for the Matérn covariance function hold here as well.

The parameters have the same signification as for covariance functions and similar statements about the practical range and the parameter θ_1 can be given. Of course, this list is not exhaustive. All covariance functions seen in Chapter 9 have a variogram counterpart. In fact, in the case of second-order stationary processes, we have $\gamma(\boldsymbol{h}) = c(\boldsymbol{0}) - c(\boldsymbol{h})$ (and $\gamma^\circ(h) = c^\circ(0) - c^\circ(h)$ as well), permitting us to express the second moment in terms of a covariance function or a variogram. This property is used in R-Code 10.1 to plot common variograms shown Figure 10.1.

Further, note that we have directly included the nugget effect in the process. Historically, the stochastic component's additive decomposition 9.6 has not been formulated and exploited.

R-Code 10.1 Semi-variograms for a weakly stationary process. (See Figure 10.1.)

```r
library(spam)
plot(0, 0, xlim=c(0,.75), ylim=c(0, 1.1), pch=20, xlab="lag h",
ylab=expression(gamma(h)))
abline(v=.5, h=c(.95,1), col="gray")
h <- seq(1e-5, to=.8, length=100) # Lag vector

### theta <- c(range, sill, nugget) or c(range, sill, smoothness, nugget)
lines(h, 1-cov.sph(h, c(.5, .9, .1)), col=1)
lines(h, 1-cov.exp(h, c(.5/3,.9, .1)), col=2)
lines(h, 1-cov.mat(h, c(.125, .9, 1, .1)), col=3) # Ranges are approximate
lines(h, 1-cov.mat(h, c(.0845, .9, 2.5,.1)), col=4)
lines(h, 1-cov.mat(h, c(.0618, .9, 5, .1)), col=5)
legend("bottomright", bty="n",col=1:5, lty=1, cex=1.2,
       legend=c("Spherical", "Exponential", expression(Matern~~nu==1),
                expression(Matern~~nu==2.5), expression(Matern~~nu==5)))
```

Example 10.1. We consider a one-dimensional transect with $\mathcal{D} = [0, T]$ and define a process:

1. $Z(0) = 0$
2. $Z(t)$ has stationary, independent increments with $Z(t+h) - Z(t) \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma^2 h)$.
3. $Z(t)$ is continuous in t (with probability one).

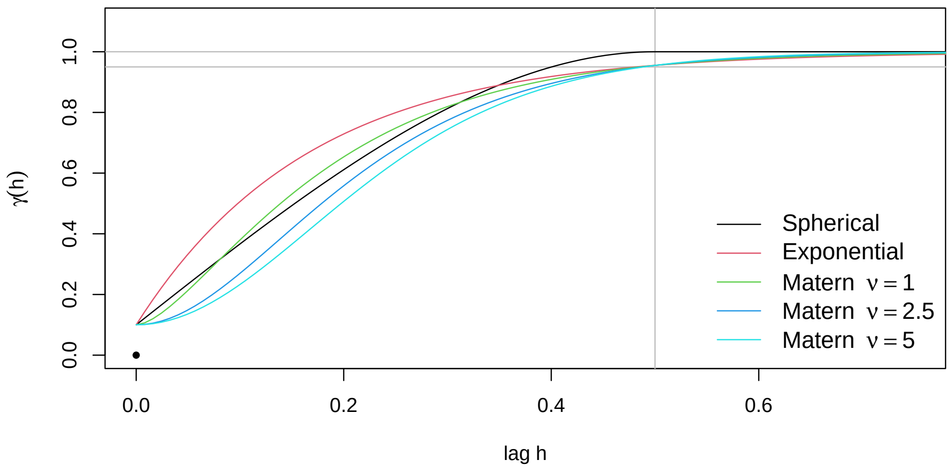


Figure 10.1: Different semi-variograms given by equations (10.5) to (10.7). The semi-variograms have nugget effect 0.1, sill 0.9 and (practical) range 0.5. (See R-Code 10.1.)

This process is also called a Brownian motion or one-dimensional Wiener process.

The process is intrinsically stationary as $\mathrm{E}\big(Z(t)\big) = 0$ and $\mathrm{Var}\big(Z(t+h) - Z(t)\big) = \sigma^2 h$ is a function of h. However, straightforward manipulations show that $\mathrm{Cov}(Z(t+h), Z(t)) = \sigma^2 t$, and thus the process is not second-order stationary. ♣

Remark 10.1. 1. To see equation (10.3), use the sum constraint to write

$$\Big(\sum_{i=1}^{n} \alpha_i Z(\boldsymbol{s}_i)\Big)^2 = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j (Z(\boldsymbol{s}_i) - Z(\boldsymbol{s}_j))^2 \qquad (10.8)$$

and use the definition of the variogram.

2. Useful properties of the variogram are, for example

$$\gamma(\boldsymbol{0}) = 0, \qquad \gamma(\boldsymbol{h}) = \gamma(-\boldsymbol{h}) \geq 0, \qquad \lim_{\|\boldsymbol{h}\|\to\infty} \frac{\gamma(\boldsymbol{h})}{\|\boldsymbol{h}\|^2} = 0. \qquad (10.9)$$

We can verify that any convex combination of conditionally negative definite functions is still conditionally negative definite.

Christakos (1984) establishes different necessary and sufficient conditions for the conditionally negative-definiteness of the function $\gamma(\cdot)$.

3. It can be shown that if $\gamma(\cdot)$ is continuous at the origin, then $Z(\cdot)$ is L_2-continuous (i.e., a process $Z(\cdot)$ for which $\mathrm{E}\big(Z(\boldsymbol{s}+\boldsymbol{h}) - Z(\boldsymbol{s})\big)^2 \to 0$, as $\|\boldsymbol{h}\| \to \boldsymbol{0}$). The converse is also true, if $\gamma(\cdot)$ is not continuous at the origin, then $Z(\cdot)$ is not L_2-continuous (see, e.g., Matheron, 1971, Page 58).

4. A common example of a non-monotone variogram is the so-called hole-effect model; see Journel and Huijbregts (1978, Section III.3, Pages 168–171) or Jones and Ma (2001); Ma and Jones (2001). ♡

10.2 Variogram Estimation and Fitting

We now assume an intrinsically stationary process. This section discusses the variogram estimation and gives some links to covariance functions estimation in case of weak stationarity.

10.2.1 Estimation of the First Moment

For the first moment, we typically assume that $\mu(\boldsymbol{s}) = \boldsymbol{x}(\boldsymbol{s})^\top \boldsymbol{\beta}$, which means $\boldsymbol{x}(\boldsymbol{s})$ is a vector containing the covariates at location $\boldsymbol{s}$.

The classical least squares estimate is $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{z}$, where $\mathbf{X}$ is the so-called design matrix containing $\boldsymbol{x}(\boldsymbol{s}_i)^\top$ as rows and $\boldsymbol{z}$ contains the observed data. The associated estimator is unbiased but is not optimal in the sense that its variance is not appropriate. Let $\mathbf{Z}$ be the random vector associated with $\boldsymbol{z}$, i.e., $\mathbf{Z} = (Z(\boldsymbol{s}_1), \dots, Z(\boldsymbol{s}_n))^\top$ and denote $\boldsymbol{\Sigma}_\mathbf{Z} = \text{Var}(\mathbf{Z})$. Then the best unbiased minimum variance estimator of $\boldsymbol{\beta}$ is

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^\top\boldsymbol{\Sigma}_\mathbf{Z}^{-1}\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{\Sigma}_\mathbf{Z}^{-1}\mathbf{Z}, \tag{10.10}$$

However, $\boldsymbol{\Sigma}_\mathbf{Z}$ is typically unknown and needs to be estimated. But for the estimation of $\boldsymbol{\Sigma}_\mathbf{Z}$, the knowledge of the mean structure is required, as we will see. As a workaround, an iterative approach may be needed.

10.2.2 Estimation of the Variogram

To estimate the spatial variation or spatial dependence structure, we assume an intrinsically stationary process $Z(\cdot)$ and define the process of differences $V(\boldsymbol{h}) = Z(\cdot + \boldsymbol{h}) - Z(\cdot)$. Intrinsic stationarity implies that we can estimate the variogram with mean estimators for the process $V(\cdot)^2$ or with variance estimators for the process $V(\cdot)$. Many different variogram estimators exist, but only a few of them are used in practice. We will discuss three of them in detail.

Matheron's classical estimator of the variogram is defined as (Matheron, 1962)

$$2\widehat{\gamma}(\boldsymbol{h}) = \frac{1}{N_J}\sum_{(i,j)\in J}\big(Z(\boldsymbol{s}_i) - Z(\boldsymbol{s}_j)\big)^2, \tag{10.11}$$

where

$$J = J(\boldsymbol{h}) = \big\{(i,j) : \boldsymbol{s}_i - \boldsymbol{s}_j \in \text{Tol}(\boldsymbol{h})\big\}, \quad N_J = \text{card}\{J\} \tag{10.12}$$

and $\text{Tol}(\boldsymbol{h})$ is some specified tolerance region in $\mathbb{R}^d$. A typical tolerance region is $\text{Tol}(\boldsymbol{h}) = \text{Tol}_{l,a}(\boldsymbol{h}) = \big\{\boldsymbol{k} : \|\boldsymbol{k} - \boldsymbol{h}\| \le l, \arccos\big(\boldsymbol{k}^\top\boldsymbol{h}/(\|\boldsymbol{k}\|\|\boldsymbol{h}\|\big) \le a\big\}$.

The classical estimator is sensitive to outliers but has other appealing properties. If the vector $\boldsymbol{z}$ denotes the observations, we can write (10.11) as $2\widehat{\gamma}(\boldsymbol{h}) = \boldsymbol{z}^\top\mathbf{A}(\boldsymbol{h})\boldsymbol{z}$ for some matrix $\mathbf{A}(\boldsymbol{h})$. If the locations form a regular grid, $\mathbf{A}(\boldsymbol{h})$ takes simple forms (see for example Genton, 1998b, 2000). Under Gaussianity, this fact simplifies investigations on distributional properties, inference etc.; see also Davis and Borgman (1979, 1982) for exact sampling distributions and asymptotic properties.

Cressie and Hawkins (1980) start their development of a variogram estimator by noting that under normality $V(\cdot)^2$ has a chi-squared distribution with one degree of freedom and is highly skewed. They transform the process of differences to obtain an almost symmetric distribution and use the standard mean estimator. Undoing the effect of transformation, they obtain

$$2\widehat{\gamma}(\boldsymbol{h}) = \frac{1}{b(N_J)}\left(\frac{1}{N_J}\sum_{(i,j)\in J}\bigl|Z(\boldsymbol{s}_i) - Z(\boldsymbol{s}_j)\bigr|^{1/2}\right)^4, \tag{10.13}$$

where J and N_J are given by (10.12) and $b(N_J) = 0.457 + 0.494/N_J + (0.045/N_J)^2$ corrects for bias under Gaussianity.

Note that Cressie (1993, equation (2.4.12), Page 75) and most statistical software omit the quadratic correction term in $b(N_J)$. The transformed process is less sensitive to outliers, and the authors consider their estimator as 'robust'. Nevertheless, the estimator has a breakdown point of zero (Hampel, 1971, Page 97), i.e., a single outlier in the data can destroy the estimator.

A genuinely robust estimator is based on the scale estimator Q_n of Rousseeuw and Croux (1992, 1993) and the resulting variogram estimator is

$$2\widehat{\gamma}(\boldsymbol{h}) = \Bigl(2.2191\bigl\{|V_i(\boldsymbol{h}) - V_j(\boldsymbol{h})| : i < j\bigr\}_{(k)}\Bigr)^2 \tag{10.14}$$

with $k = \binom{\lfloor N_J \rfloor + 1}{2}$, termed the Q_J variogram estimator, (Genton, 1998a; Dutter, 1996). The factor 2.2191 guarantees consistency under Gaussianity.

Remark 10.2. The estimate Q_J has $\mathcal{O}\bigl(N_J \log(N_J)\bigr)$ computing time and takes $\mathcal{O}(N_J)$ storage place. The distributional properties of these variogram estimators are complex and nontrivial, implying that inference for the estimated variogram parameters is rarely developed. However, many simulations and case studies exist investigating the effects of the variogram parameters on prediction results.

If the process $Z(\cdot)$ is not intrinsically stationary, for example not correctly detrended, the variogram estimators are biased.

It is not reasonable to take into account large lags $\boldsymbol{h}$. Journel and Huijbregts (1978, Page 194) recommend that variograms should not be estimated for lags bigger than half the diameter of $\mathcal{D}$ (larger structures are modeled through the first moment). Additionally, they recommend that the lags $\boldsymbol{h}$ be chosen, such as $N_J > 30$. For small datasets, this practical rule can only sometimes be applied.

We can estimate the variance along a given axis $\boldsymbol{h}$, i.e., estimate a directional variogram, by using an appropriate tolerance region $\mathrm{Tol}(\boldsymbol{h})$. Directional variograms are an indispensable tool in exploratory variography. Besides the indication of trends and other types of nonstationarity, they allow calculating the transformation matrix $\mathbf{O}$ of geometrical anisotropy. A diagram of ranges in different directions is plotted; the anisotropy ratio is defined as the ratio between the smallest and the biggest range. The corresponding two directions are assumed to be approximately perpendicular to each other (Goovaerts, 1997, Page 90). A ratio of one denotes an isotropic variogram as its range is independent of the direction; see also Paragraph 9.1.1. Directional variograms are also used for a better understanding of zonal anisotropy. Zimmerman (1993) introduces a subtle and clever classification of zonal anisotropy. He distinguishes nugget, sill, range, and slope anisotropy and reveals with several examples and graphics the differences and causes. He also shows that sill anisotropy may be evidence of nonstationarity, non-vanishing spatial correlation, or measurement errors that are correlated or have unequal means.

In case of second-order stationarity, we have $\gamma(\boldsymbol{h}) = c(\boldsymbol{0}) - c(\boldsymbol{h})$ and thus the covariance $c(\boldsymbol{h})$ can be estimated by

$$\widehat{c}(\boldsymbol{h}) = \Big(\lim_{\|\boldsymbol{h}\|\to\infty} \widehat{\gamma}(\boldsymbol{h})\Big) - \widehat{\gamma}(\boldsymbol{h}). \tag{10.15}$$

The advantage of this estimator is that we avoid the $\mathcal{O}(1/n)$ bias of the natural covariance estimator

$$\widehat{c}(\boldsymbol{h}) = \frac{1}{J_N} \sum_{(i,j)\in J} \big(Z(\boldsymbol{s}_i) - \overline{Z}\big)\big(Z(\boldsymbol{s}_j) - \overline{Z}\big), \tag{10.16}$$

where J and N_J are given by (10.12) and $\overline{Z} = \sum_i Z(\boldsymbol{s}_i)/n$. Geostatisticians usually use the variogram approach, whereas statisticians prefer the covariance functions. See Cressie and Grondona (1992) for a detailed variogram and covariance estimation comparison. Note that $\widehat{c}(\cdot)$ as defined by (10.16) may not necessarily be positive definite. Whatever method is used to obtain $\widehat{\gamma}(\cdot)$ and $\widehat{c}(\cdot)$, called the empirical variogram and empirical covariance, we usually need to fit a theoretical family of variograms or covariance functions in order to have a valid second moment structure.

Example 10.2. We reconsider the *Chicago03* dataset from Section 9.4. We use the package *gstat*. The default values of the function *variogram()* are suboptimal as we do not have enough data. Setting manual values for *cutoff* (maximum lag for the variogram) and *width* (bin width), we have a reasonable amount of pairs in each bin (N_J). The estimate of the first bin is very large; changing to a robust estimate does not remedy it. The atypical form of the variogram estimates was also the reason why the likelihood surface was extremely flat. Note that *gstat* uses trellis graphics (via the *lattice* package). To add elements to a variogram plot, we need a slightly atypical approach. ♣

R-Code 10.2: Variogram estimation of *Chicago03* data.

```
library(fields)
O3 <- data.frame(ChicagoO3$x, ChicagoO3$y)
names(O3) <-c("x","y","O3")
library(gstat)
library(sp)                # for function `coordinates()`
coordinates(O3) <- ~x+y
# variogram(O3~1, data=O3 # does not lead to good results
(vg1 <- variogram(O3~1, data=O3, cutoff=40, width=8))
##    np    dist  gamma dir.hor dir.ver   id
## 1 10   5.7473 34.037       0       0 var1
## 2 27  12.0808 10.729       0       0 var1
## 3 32 20.5857 16.142       0       0 var1
## 4 43 27.8581 19.287       0       0 var1
## 5 25 35.2441 21.822       0       0 var1
```

```
(vg2 <- variogram(O3~1, data=O3, cutoff=40, width=8, cressie=TRUE))
```

```
##   np    dist   gamma dir.hor dir.ver   id
## 1 10  5.7473 34.3482       0       0 var1
## 2 27 12.0808  8.7714       0       0 var1
## 3 32 20.5857 15.8362       0       0 var1
## 4 43 27.8581 23.4793       0       0 var1
## 5 25 35.2441 23.7643       0       0 var1
```

```
# g <- plot(vg1)
# g + latticeExtra::layer(panel.points(vg2$dist, vg2$gamma, pch=4, col=2))
```

10.2.3 Variogram Model Fitting

Fitting a variogram model to an empirical variogram consists of choosing an appropriate family of valid variograms $2\gamma(\boldsymbol{h};\boldsymbol{\theta})$ and estimating $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subset \mathbb{R}^L$ within this family to get a good match between the estimated and the theoretical variogram. There are many different approaches to the so-called variogram fitting. We will discuss a few standard techniques which can be classified as parametric fitting.

A broad class of parametric fitting techniques is the least squares method. Suppose we have an empirical variogram $2\widehat{\gamma}(\boldsymbol{h}_k)$, for different lags $\boldsymbol{h}_k$, $k = 1, \dots, K$, and an appropriate family of variograms $2\gamma(\boldsymbol{h};\boldsymbol{\theta})$. The method of ordinary least squares estimates $\boldsymbol{\theta}$ by means of

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{k=1}^{K} \big(\widehat{\gamma}(\boldsymbol{h}_k) - \gamma(\boldsymbol{h}_k;\boldsymbol{\theta})\big)^2. \tag{10.17}$$

However, classical variograms are not linear in $\boldsymbol{\theta}$, and we should rather speak of nonlinear least squares estimates.

To take account of the covariance between different lags, we may use generalized (nonlinear) least squares (GLS), resulting in

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{k,l=1}^{K} \big(\widehat{\gamma}(\boldsymbol{h}_k) - \gamma(\boldsymbol{h}_k;\boldsymbol{\theta})\big) \operatorname{Cov}\big(\widehat{\gamma}(\boldsymbol{h}_k;\boldsymbol{\theta}), \widehat{\gamma}(\boldsymbol{h}_l;\boldsymbol{\theta})\big) \big(\widehat{\gamma}(\boldsymbol{h}_l) - \gamma(\boldsymbol{h}_l;\boldsymbol{\theta})\big). \tag{10.18}$$

Unfortunately the covariance structure in equation (10.18) is in general not known and depends on $\boldsymbol{\theta}$, the parameter to estimate. Often it is sufficient to use weighted (WLS) least squares criterion

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{k=1}^{K} N_{J(\boldsymbol{h}_k)} \Big(\frac{\widehat{\gamma}(\boldsymbol{h}_k)}{\gamma(\boldsymbol{h}_k;\boldsymbol{\theta})} - 1\Big)^2. \tag{10.19}$$

In software implementations, many more weighting schemes are typically implemented. The following example illustrates fitting using the package `gstat`.

Example 10.3. We reconsider the *Chicago03* dataset from Example 9.4. We use the package *gstat* and define a variogram model with the function *vgm()*, with arguments partial sill, model (like spherical *"Sph"*, exponential *"Exp"*, …), range, nugget, etc. The values of the proposed model will be used as starting values. The fitting is performed with *fit.variogram* and the argument *fit.method* specifies the weights (e.g., *=6* for (10.17) or *=2* for (10.19)). The estimate of the first bin is very large and hinders stable optimization. Here, a manual fitting might be adequate and preferable to a numerical optimization (see also Remark 10.4). As an illustration, we omit the first row, now leading to stable estimates. The weights have a minor effect, as illustrated in Figure 10.2.

Of course, the approach to delete one estimate of the varigram is not recommended, but we cite from the help of *Chicago03*: "The lasting scientific value [of the data] is probably minimal." ♣

R-Code 10.3: Variogram estimation of *Chicago03* data. (See Figure 10.2.)

```
vg.model <- vgm(25,"Sph", 30)                    # proposed model. No nugget here!
(fit2 <- fit.variogram(vg1[-1,], vg.model, fit.method=2))  # Fitting with WLS
##   model  psill  range
## 1   Sph 21.362 36.048
(fit6 <- fit.variogram(vg1[-1,], vg.model, fit.method=6))  # with OLS
##   model  psill  range
## 1   Sph 21.669 37.159
plot(gamma~dist, vg2, xlim=c(0, 40), ylim=c(0, 1.05*max(vg2$gamma)),
col=4, pch=19, ylab='semivariance', xlab='distance')
lines(variogramLine(fit6, 42), col=2)       # WLS fit
lines(variogramLine(fit2, 42), col=4)       # OLS fit
```

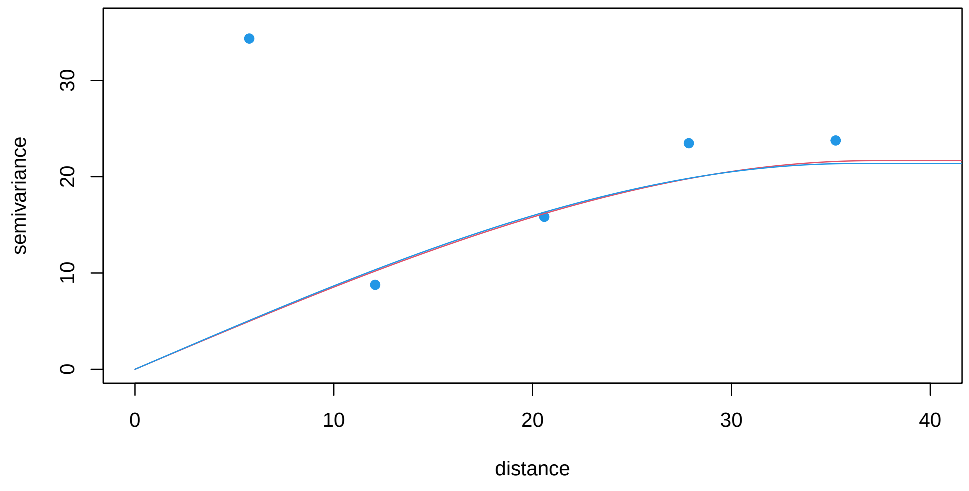


Figure 10.2: Variogram fitting with *gstat*. Note that the variogram estimates have been manipulated to get the OLS fit (blue) and WLS fit (red). (See R-Code 10.3.)

Remark 10.3. Genton (1998b) uses the GLS criterion to approximate the covariance structure of the Q_J variogram estimator for regular grids. As the minimization criterion is highly nonlinear, he proposes an iterative algorithm, referred to as generalized least squares with an explicit formula for the covariance structure (GLSE). He also shows that a good approximation of the covariance structure is achieved by taking into account the explicit formula for the covariance in the independent situation. ♡

Remark 10.4. However, manual fitting is still used in applications, partly because geoscientists use fitting as a part of geologic interpretation (Ma and Jones, 2001). On the other hand, Zimmerman and Zimmerman (1991) show that OLS is usually as good as many of the more complicated and computationally intensive methods. ♡

10.3 Kriging

Suppose we wish to make predictions at a location not included in the observed locations, for example, to establish a pollution map. In that case, all measures must be considered in computing the predicted value. Of course, their contributions will be weighted by the strength of their correlation with the location of interest. We start introducing the most widely used spatial predictors.

Kriging is a minimum mean squared error method of spatial prediction. Matheron (1963) named this method of optimal spatial linear prediction after D. G. Krige, a South African mining engineer who developed empirical methods for determining ore-grade distributions from samples (Krige, 1951). However, the formulation of optimal linear prediction did not come from him. Wold (1938), Kolmogoroff (1941), and Wiener (1949) developed optimal linear prediction, where closer points obtained more weight than distant points. For a more thorough overview of the historical origins of kriging we refer to Cressie (1990b).

We assume an intrinsically stationary process $Z(\cdot)$. As a spatial linear predictor for a location $\boldsymbol{s}_0$, we use

$$\widehat{Z}(\boldsymbol{s}_0) = p\big(Z(\boldsymbol{s}_0); \mathbf{Z}\big) = \sum_{i=1}^{n} \lambda_i Z(\boldsymbol{s}_i) = \boldsymbol{\lambda}^\top \mathbf{Z}, \tag{10.20}$$

where the weights $\boldsymbol{\lambda} = (\lambda_1, \dots, \lambda_n)^\top$ are chosen such that the predictor is unbiased and that the mean squared prediction error

$$\mathrm{E}\Big(\Big(Z(\boldsymbol{s}_0) - \sum_{i=1}^{n} \lambda_i Z(\boldsymbol{s}_i)\Big)^2\Big) \tag{10.21}$$

is minimal. With $\boldsymbol{\lambda}$ chosen under these constraints the predictor (10.20) is called the *ordinary kriging* predictor and is the best linear unbiased predictor (BLUP). The optimal weights satisfy $\mathbf{1}^\top \boldsymbol{\lambda} = 1$ to guarantee unbiasedness and equation (10.21) can be expressed using variograms. We use the notation $2\gamma(\boldsymbol{s}_i - \boldsymbol{s}_j) = \mathrm{Var}\big(Z(\boldsymbol{s}_i) - Z(\boldsymbol{s}_j)\big)$.

Minimizing (10.21) under the constraint $\mathbf{1}^\top \boldsymbol{\lambda} = 1$ by means of Lagrange multipliers leads us to the following system of equations characterizing the kriging weights

$$\begin{cases} \sum_{j=1}^{n} \lambda_j \gamma(\boldsymbol{s}_i - \boldsymbol{s}_j) + \eta = \gamma(\boldsymbol{s}_0 - \boldsymbol{s}_i), \quad i = 1, \dots, n, \\ \sum_{j=1}^{n} \lambda_j = 1, \end{cases} \tag{10.22}$$

or expressed in matrix notation

$$\begin{pmatrix} \boldsymbol{\Gamma} & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \eta \end{pmatrix} = \begin{pmatrix} \boldsymbol{\gamma} \\ 1 \end{pmatrix}, \tag{10.23}$$

where $\boldsymbol{\Gamma} = \big(\gamma(\boldsymbol{s}_i - \boldsymbol{s}_j)\big)$, $\boldsymbol{\gamma} = \big(\gamma(\boldsymbol{s}_0 - \boldsymbol{s}_i)\big)$ and η is the Lagrange multiplier. Solving the previous system we obtain as the solution for the kriging weights

$$\boldsymbol{\lambda} = \boldsymbol{\Gamma}^{-1}\Big(\boldsymbol{\gamma} + \mathbf{1}\frac{1 - \mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma}}{\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1}}\Big). \tag{10.24}$$

The minimized mean squared prediction error (10.21) is often called the kriging variance and is given by

$$\sigma^2_{\text{BLUP}}(\boldsymbol{s}_0) = \boldsymbol{\lambda}^\top\boldsymbol{\gamma} + \eta = \boldsymbol{\gamma}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma} - \frac{\big(1 - \mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma}\big)^2}{\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1}}. \tag{10.25}$$

Many geostatisticians misunderstand the latter equation as it is not an estimation variance in its strict sense but rather some index of data configuration (Srivastava, 1986, Page 144).

Remark 10.5.
1. Formally, the linear predictor should be formulated as $p\big(Z(\boldsymbol{s}_0); \mathbf{Z}\big) = \lambda_0 + \sum_{i=1}^{n} \lambda_i Z(\boldsymbol{s}_i)$. As we work with stationary process, $\mathrm{E}\big(Z(\boldsymbol{s}_i)\big) = \mu$, for $i = 0, \dots, n$, and thus $\lambda_0 = 0$.

2. To derive the kriging equation (10.23) start with the square form of equation (10.21) and expand the square to get

$$\begin{aligned} &\Big(Z(\boldsymbol{s}_0) - \sum_{i=1}^{n} \lambda_i Z(\boldsymbol{s}_i)\Big)^2 \\ &= Z(\boldsymbol{s}_0)^2 - 2Z(\boldsymbol{s}_0)\sum_{i=1}^{n}\lambda_i Z(\boldsymbol{s}_i) + \sum_{i=1}^{n}\lambda_i Z(\boldsymbol{s}_i)^2 - \sum_{i=1}^{n}\lambda_i Z(\boldsymbol{s}_i)^2 + \Big(\sum_{i=1}^{n}\lambda_i Z(\boldsymbol{s}_i)\Big)^2 \end{aligned} \tag{10.26}$$

$$\begin{aligned} &= \sum_{i=1}^{n} \lambda_i \Big(Z(\boldsymbol{s}_0) - 2Z(\boldsymbol{s}_0)Z(\boldsymbol{s}_i) + Z(\boldsymbol{s}_i)^2\Big) \\ &\qquad - \frac{1}{2}\Big(\sum_{i=1}^{n}\lambda_i Z(\boldsymbol{s}_i)^2 + \sum_{j=1}^{n}\lambda_j Z(\boldsymbol{s}_j)^2 - 2\Big(\sum_{i=1}^{n}\lambda_i Z(\boldsymbol{s}_i)\Big)\Big(\sum_{j=1}^{n}\lambda_j Z(\boldsymbol{s}_j)\Big)\Big) \end{aligned} \tag{10.27}$$

$$= \sum_{i=1}^{n} \lambda_i \Big(Z(\boldsymbol{s}_0) - Z(\boldsymbol{s}_i)\Big)^2 - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j\Big(Z(\boldsymbol{s}_i) - Z(\boldsymbol{s}_j)\Big)^2, \tag{10.28}$$

where we have used $\sum_j \lambda_i = 1$ for the second equality. We now use the definition of the variogram (written in terms of the squared expectation) to write equation (10.21)

$$\sum_{i=1}^{n} \lambda_i 2\gamma(\boldsymbol{s}_0 - \boldsymbol{s}_i) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda_i\lambda_j 2\gamma(\boldsymbol{s}_i - \boldsymbol{s}_j) = \boldsymbol{\lambda}^\top 2\boldsymbol{\gamma} - \frac{1}{2}\boldsymbol{\lambda}^\top 2\boldsymbol{\Gamma}\boldsymbol{\lambda}. \tag{10.29}$$

which we have to minimize under the constraint $\boldsymbol{\lambda}^\top \mathbf{1} = 1$. Taking derivatives with respect to λ_i and the Lagrange multiplier 2η (factor 2 for convenience), equations (10.22) and equation (10.23) follow.

3. To solve the system (10.23), we use the classical formula for the inverse of a 2×2 block matrix

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{A}_{11}^{-1} - \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\mathbf{C} \\ -\mathbf{C}\mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{C} \end{pmatrix} \tag{10.30}$$

with $\mathbf{C} = (\mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1}$. (We need to assume that the inverse of $\mathbf{A}_{11}$ and that $\mathbf{C}$ exist.) Hence, using (10.23) we have $\mathbf{A}_{11} = \boldsymbol{\Gamma}$, $\mathbf{A}_{12} = \mathbf{A}_{21}{}^\top = \mathbf{1}$, and $\mathbf{A}_{22} = 0$ and thus $\mathbf{C} = -(\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1})^{-1}$. Further, on the first row, we have $\boldsymbol{\Gamma}^{-1} - \boldsymbol{\Gamma}^{-1}\mathbf{1}(\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}$ and $\boldsymbol{\Gamma}^{-1}\mathbf{1}(\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1})^{-1}$. Hence,

$$\boldsymbol{\lambda} = \boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma} - \boldsymbol{\Gamma}^{-1}\mathbf{1}(\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma} + \boldsymbol{\Gamma}^{-1}\mathbf{1}(\mathbf{1}^\top\boldsymbol{\Gamma}^{-1}\mathbf{1})^{-1}, \tag{10.31}$$

which is exactly (10.24).

4. To derive the formula for the MSPE, we start with (10.29) and plug into $2\boldsymbol{\lambda}^\top\boldsymbol{\gamma} - \boldsymbol{\lambda}^\top\boldsymbol{\Gamma}\boldsymbol{\lambda}$ the expression of $\boldsymbol{\lambda}$ as given in (10.24). Straightforward simplifications lead to (10.25). ♡

The kriging predictor is an "exact predictor". That means that if we do not have a measurement error and we predict at a location for which we have an observation, we have $Z(\boldsymbol{s}_i) = p(\boldsymbol{s}_i, \mathbf{Z})$. The justification is straight forward, $\boldsymbol{\gamma}_i$ is equivalent to the i column of $\boldsymbol{\Gamma}$ and thus $\boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma}_i = \boldsymbol{e}_i$, the ith canonical basis vector. Hence, (10.24) leads to $\boldsymbol{\lambda} = \boldsymbol{e}_i$.

The kriging equations can also be derived in terms of the covariance function. We obtain similar results for the weights: exchange in equation (10.23) $\boldsymbol{\Gamma}$ and $\boldsymbol{\gamma}$ by $\mathbf{C}$ and $\boldsymbol{c}$ respectively ($c(\boldsymbol{s}_i - \boldsymbol{s}_j) = \operatorname{Cov}\big(Z(\boldsymbol{s}_1), Z(\boldsymbol{s}_2)\big)$ and so forth), and the mean squared prediction error (10.21) becomes $\sigma^2_{\text{BLUP}}(\boldsymbol{s}_0) = c(\mathbf{0}) - \boldsymbol{c}^\top\mathbf{C}^{-1}\boldsymbol{c} + (1 - \mathbf{1}^\top\mathbf{C}^{-1}\boldsymbol{c})^2/(\mathbf{1}^\top\mathbf{C}^{-1}\mathbf{1})$.

Example 10.4. In a pure nugget effect model, $\mathbf{C} = \sigma^2\mathbf{I}$ and the ordinary kriging predictor simplifies to

$$p\big(Z(\boldsymbol{s}_0); \mathbf{Z}\big) = \begin{cases} \displaystyle\sum_{i=1}^n Z(\boldsymbol{s}_i)/n, & \text{if } \boldsymbol{s}_0 \notin \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_n\}, \\ Z(\boldsymbol{s}_i), & \text{if } \boldsymbol{s}_0 = \boldsymbol{s}_i, \quad i = 1, \dots, n. \end{cases} \tag{10.32}$$

If the underlying variogram of the process has a nugget effect, the kriging predictor is not continuous. ♣

Remark 10.6. 1. Several approaches to deriving the kriging equations in terms of the covariance function exist. One possibility is to write $\boldsymbol{\Gamma} = c(\mathbf{0})\mathbf{1}\mathbf{1}^\top - \mathbf{C}$ and $\boldsymbol{\gamma} = c(\mathbf{0})\mathbf{1} - \boldsymbol{c}$ and start plugging in these two quantities in (10.23). The $c(\mathbf{0})$ term simplifies and thus we can write (10.23) and (10.24) by replacing $\boldsymbol{\Gamma}$ and $\boldsymbol{\gamma}$ with $\mathbf{C}$ and $\boldsymbol{c}$. For the MSPE, we can start with (10.29) and plug in the corresponding equalities. Here, less terms simplify, but the expression is simpler to interpret.

2. To show that ordinary kriging (based on covariance function) is equivalent to prediction based on the GLS estimate, consider a second-order stationary process $Y(\boldsymbol{s})$. According to equation (1.19) we use $p(Y(\boldsymbol{s}_0), \boldsymbol{y}) = 1\beta + \boldsymbol{c}^\top \mathbf{C}^{-1}(y - \mathbf{1}\beta)$ with covariances $\boldsymbol{\Sigma} = \mathbf{C}$ and $\boldsymbol{c}$. Because of the constant mean assumption, equation (10.10) simplifies to $\widehat{\beta} = (\mathbf{1}^\top \mathbf{C}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top \mathbf{C}^{-1}\boldsymbol{y}$, which we plug into the previous equation to simplify:

$$\begin{aligned} p(Y(\boldsymbol{s}_0), \boldsymbol{y}) &= 1\widehat{\beta} + \boldsymbol{c}^\top \mathbf{C}^{-1}(\boldsymbol{y} - \mathbf{1}\widehat{\beta}) && (10.33)\\ &= 1(\mathbf{1}^\top \mathbf{C}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top \mathbf{C}^{-1}\boldsymbol{y} + \boldsymbol{c}^\top \mathbf{C}^{-1}(\boldsymbol{y} - \mathbf{1}(\mathbf{1}^\top \mathbf{C}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top \mathbf{C}^{-1}\boldsymbol{y}) && (10.34)\\ &= \Big(\boldsymbol{c}^\top + 1(\mathbf{1}^\top \mathbf{C}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top - \boldsymbol{c}^\top \mathbf{C}^{-1}\mathbf{1}(\mathbf{1}^\top \mathbf{C}^{-1}\mathbf{1})^{-1}\mathbf{1}^\top\Big)\mathbf{C}^{-1}\boldsymbol{y} && (10.35)\\ &= \Big(\boldsymbol{c}^\top + \frac{1 - \boldsymbol{c}^\top \mathbf{C}^{-1}\mathbf{1}}{\mathbf{1}^\top \mathbf{C}^{-1}\mathbf{1}}\mathbf{1}^\top\Big)\mathbf{C}^{-1}\boldsymbol{y}, && (10.36) \end{aligned}$$

which is $\boldsymbol{\lambda}^\top \boldsymbol{y}$, where the kriging weights (10.24) are expressed in terms of $\mathbf{C}$ and $\boldsymbol{c}$. ♡

To derive the kriging equations, intrinsic or second-order stationarity is not required. We merely need the existance of the corresponding vectors and matrices $\boldsymbol{\gamma}$, $\boldsymbol{\Gamma}$, $\boldsymbol{c}$ and $\mathbf{C}$.

If the mean $\mu(\cdot)$ and the covariance structure $c(\cdot,\cdot)$ of the process $Z(\cdot)$ are known functions the optimal predictor simplifies to

$$p(\boldsymbol{s}_0, \mathbf{Z}) = \mu(\boldsymbol{s}_0) + \boldsymbol{c}^\top \mathbf{C}^{-1}(\mathbf{Z} - \boldsymbol{\mu}), \tag{10.37}$$

where $\boldsymbol{c} = \big(c(\boldsymbol{s}_0, \boldsymbol{s}_j)\big)$, $\mathbf{C} = \big(c(\boldsymbol{s}_i, \boldsymbol{s}_j)\big)$ and $\boldsymbol{\mu} = \big(\mu(\boldsymbol{s}_i)\big)$. The predictor (10.37) is called the *simple kriging* predictor. For Gaussian processes simple kriging coincides with $\mathrm{E}\big(Z(\boldsymbol{s}_0) \mid Z(\boldsymbol{s}_1), \dots, Z(\boldsymbol{s}_n)\big)$ and is therefore optimal amongst all unbiased predictors.

If the deterministic mean structure can be expressed as

$$\mu(\cdot) = \boldsymbol{x}(\cdot)^\top \boldsymbol{\beta} = \sum_{k=1}^{K} \beta_k x_k(\cdot), \tag{10.38}$$

where $\{x_k(\cdot)\}_{k=1}^{K}$ are known functions spanning a K-dimensional subspace, then ordinary kriging can be generalized to *universal kriging* (Goldberger, 1962; Matheron, 1969). In (10.23), the vector $\mathbf{1}$ is replaced by the design matrix $\mathbf{X}$ induced by the functions $\{x_k(\cdot)\}_{k=1}^{K}$ evaluated at the locations:

$$\begin{pmatrix} \boldsymbol{\Gamma} & \mathbf{X} \\ \mathbf{X}^\top & \mathbf{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\eta} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\gamma} \\ \boldsymbol{x} \end{pmatrix}, \tag{10.39}$$

where $\boldsymbol{x} = \big(x_1(\boldsymbol{s}_0), \dots, x_K(\boldsymbol{s}_0)\big)^\top$ and $\boldsymbol{\eta}$ contains the Lagrange multipliers. With $\mu(\cdot) \equiv \mu$, (10.39) reduces to (10.23).

If we estimate the variogram using the approaches in this chapter, the mean structure has to be known to get an unbiased estimate. Hence, one often reduces the problem to ordinary kriging. (See Armstrong, 1984, Cressie, 1986 or Pardo-Igúquiza and Dowd, 1998 for a discussion of this and other problems with universal kriging.)

Recall that theoretically, kriging is an optimal predictor in the sense that it minimizes the estimation variance under the unbiasedness constraint when the second-moment structure (variogram) is known, and the process is intrinsically stationary. In practice, these conditions are rarely met. Nevertheless, kriging has become a popular interpolator in the geostatistical community. Zimmerman and Cressie (1992) compare the mean squared prediction error with the mean squared prediction error using estimated covariance parameters.

We can use leave-one-out or leave-several-out techniques to assess kriging for one particular dataset. More specifically, cycling over all observations, we omit one observation at a time from the data and predict at the location of the observation. The resulting mean squared prediction error should be close to the kriging variance. Hence, one often reports the resulting root mean squared prediction error (RMSE) and mean squared deviation ratio (MSDR) for the leave-one-out predictions.

The kriging weights and the observations determine the kriging prediction. The kriging weights are determined by a family of (co)variograms and an estimated parameter which also depends on the observations. The stability of the kriging predictor is best judged when studying its sensitivity with respect to the chosen variogram family, the variogram estimation and fitting procedure, and the observations. Since kriging is a linear predictor, small perturbations in the weights λ_i, $i = 1, \dots, n$, do not change the prediction $p(\boldsymbol{s}_0, Z)$ significantly, also discussed by Tukey (1948) in a similar context.

10.4 Example

We use the data introduced in Example 1.3 for a univariate prediction example. We use mercury (Hg) as a variable. We assume a constant trend for simplicity, i.e., we have *ordinary* kriging here. The following sequence of code chunks (i) loads and pre-processes the data: R-Code 10.4, (ii) models the second order structure R-Code 10.5, (iii) constructs a fine grid used for prediction R-Code 10.6, (iv) prediction and validation R-Code 10.7.

R-**Code 10.4:** Loading packages and data and preparing a training dataset. (See Figure 10.3.)

```
load("./data/LacLeman.RData") # see also Example 1.2
library(fields)
library(gstat)
library(sp) # for function `coordinates`

### We have duplicated observations. Creates headache with "gstat"
### ("fields" automatically addresses this).
Hgfull <- leman83[ !duplicated(leman83[,1:2]),]
set.seed(1)
train <- sort(sample(1:dim(Hgfull)[1], 68))
Hg <- Hgfull[train,]
```

```
hist(leman83$Hg, col=7, main="", xlab="Hg [ppm]")
hist(Hg$Hg, add=TRUE, col=4)
uvl <- with(Hg, data.frame(x, y, var1=Hg ) )
# str(uvl)
coordinates(uvl) <- ~x+y
# str(uvl) # !!
```

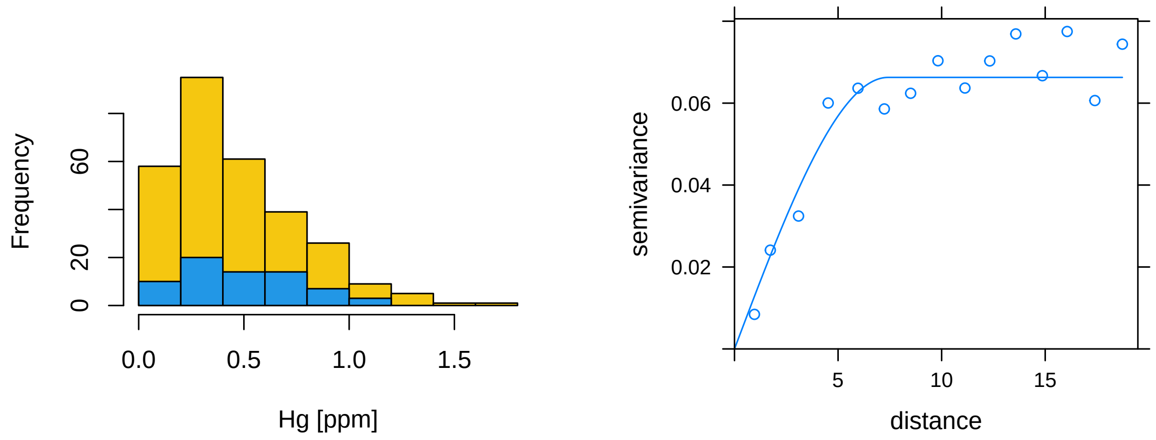


Figure 10.3: Histogram of entire and training mercury dataset (left, R-Code 10.4) and variogram (right, R-Code 10.5).

We now proceed to variogram estimation and fitting. The estimation is done with the function *variogram()*, fitting with the function *fit.variogram()*, where we also need to specify the variogram class. The package *gstat* provides many models through the function *vgm()*, including *"Sph"*, *"Exp"* *"Mat"* and *"Nug"*. See, e.g., *show.vgms(models=c("Sph", "Exp", "Mat", "Nug"))*.

R-Code 10.5: Variogram estimation and fitting. (See Figure 10.3.)

```
eV <- variogram(var1~1, uvl) # empirical
mV <- vgm(.035, model="Sph", 10, .5) # model
## Specification: partial sill, "model", range, nugget
(fV <- fit.variogram(eV, mV)) # fitting
##   model     psill  range
## 1   Nug 0.000000 0.0000
## 2   Sph 0.066285 7.4158
c(class(eV), class(mV), class(fV))
```

```
## [1] "gstatVariogram" "data.frame"     "variogramModel" "data.frame"
## [5] "variogramModel" "data.frame"
plot(eV, model=fV) # plotting both
```

We are now constructing a fine grid within the lake boundaries to predict at these locations. One possible approach is to use the function `pip()` from the package `splancs`.

R-**Code 10.6:** Constructing a fine grid within the lake boundaries.

```
library(splancs)
xr <- seq(min(lake.data[,1]), to=max(lake.data[,1]), l=100)
yr <- seq(min(lake.data[,2]), to=max(lake.data[,2]), by=xr[2]-xr[1])
# xr and yr are fine sequences of points

locs <- data.frame(x=lake.data[,1], y=lake.data[,2])
grid <- expand.grid(x=xr, y=yr) # create a 2-dim grid
pts <- pip(grid, locs, bound=TRUE) # pip points-in-polygon
# we now have a grid...
coordinates(pts) <- ~x+y # ... in the correct structure
```

Finally, we perform kriging with the function `krige()` from the package `gstat`. The function works similarly as a generic `predict()` function and is thus very intuitive. The function includes ordinary, simple, and universal kriging by adapting the `formula` statement.

The function `krige.cv()` performs leave-one-out cross-validation (LOOCV) by predicting the value at that location by leaving out the observed value and cycling over the data points. We report the RMSE and MSDR. Note that an MSDR of 0.974 is very close to one; thus, no indication of model inadequacy.

R-**Code 10.7:** Kriging, visualization and validation. (See Figure 10.4.)

```
okblup <- krige(var1~1, uvl, newdata=pts, model=fV) # kriging
## [using ordinary kriging]
par(mai=c(.1,.1,.1,.1), cex=.8)
lake(okblup$var1.pred, pts)                              # visualization...
par(mai=c(.1,.1,.1,.1), cex=.8)
lake(okblup$var1.var, pts)
points(Hg[,1:2], pch="x", col="white")
### Alternatives to base function `plot`:
# spplot(okblup, "var1.pred")
# quilt.plot(pts@coords, okblup$var1.pred, nx=length(xr)-2, ny=length(yr)-1)
# quilt.plot(pts@coords, okblup$var1.var, nx=length(xr)-2, ny=length(yr)-1)
# points(Hg[,1:2], cex=.3)
```

```
### Validation/verification:
cv <- krige.cv(var1 ~ 1, uvl, model=fV, verbose=FALSE)

summary(cv$residual)
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.49030 -0.07425 -0.00911 -0.00789  0.06608  0.58682
c(RMSE=sqrt(mean(cv$residual^2)),        # the smaller the better
  MSDR=mean(cv$residual^2/cv$var1.var) ) # should be close to one.
##    RMSE    MSDR
## 0.16154 0.97347
```

Notice that the actual kriging calculation is one single call to *krige()*. For the visualization in Figure 10.4 we have used the R-base plotting. Alternatives are given in R-Code 10.7 as well.

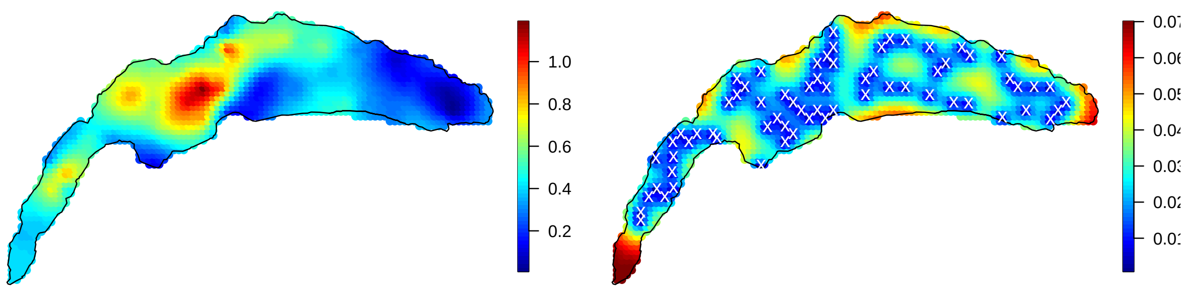


Figure 10.4: Prediction (left) and mean squared prediction error (right). The locations of the training dataset are superimposed. (See R-Code 10.7.)

10.4.1 Alternative Kriging Calls in R

The use of the function *krige()* of the package *gstat* is quite intuitive and universal. An alternative and much more flexible construct in *gstat* is as follows. This approach will also be used when we extend ordinary/universal kriging to more elaborate settings.

R-Code 10.9 predicts with the package *fields* and R-Code 10.10 predicts with the package *geoR*. In the latter one, we compare the predictions with the previous one. As we fixed all parameters, no differences are present. In *fields*, it is straightforward to define further covariance functions, as shown with the function *Spherical()*.

There are several other implementations of kriging beyond the approach mentioned earlier. One such method is to construct a covariance matrix $\mathbf{C}$ using an estimated covariance function, and then solve the linear systems $\mathbf{C}\boldsymbol{v} = \mathbf{1}$ and $\mathbf{C}\boldsymbol{w} = \boldsymbol{y}$. This approach can be highly efficient for evaluating (10.36), requiring only a single factorization of $\mathbf{C}$. However, for large datasets (with n on the order of thousands), it's important to use a compactly supported covariance function

R-**Code 10.8** Alternative use of *gstat* functionalities for kriging.

```
(g1 <- gstat(NULL, id = "var1", formula=var1~1, data=uvl) )
## data:
## var1 : formula = var1`~`1 ; data dim = 68 x 1
(g1 <- gstat(g1, id="var1", model=fV) )
## data:
## var1 : formula = var1`~`1 ; data dim = 68 x 1
## variograms:
##          model    psill  range
## var1[1]    Nug 0.000000 0.0000
## var1[2]    Sph 0.066285 7.4158
okblup.alt <- predict(g1, pts)
## [using ordinary kriging]
colSums((okblup@data- okblup.alt@data)^2)
## var1.pred  var1.var
##         0         0
### to extract the variogram parameters:
vg <- c(psill=g1$model$var1[2,2], range=g1$model$var1[2,3])
nugget <- g1$model$var1[1,2]
```

R-**Code 10.9** Kriging with the package *fields*.

```
library(fields)
Spherical <- function(d, range=1, phi=1){
    if (any(d < 0)) stop("distance argument must be nonnegative")
    d <- d/range
    return(ifelse(d < 1, phi*(1 - 1.5*d + 0.5*d^3), 0))
}

vg <- c(psill=fV[2,2], range=fV[2,3])
nugget <- fV[1,2]

out1 <- mKrig(Hg[,1:2], Hg$Hg, Covariance="Spherical",
theta=vg["range"], lambda=nugget/vg["psill"], m=1)
pout1 <- predict(out1, pts@coords)
```

with a small range relative to the diameter of the domain. This can result in a sparse matrix, which can be handled using the same tools as in Chapter 5.

R-**Code 10.10** Kriging with the package ***geoR*** and comparison.

```
library(geoR)
out2 <- krige.control(cov.pars=vg, cov.model="spherical", nugget=nugget)
pout2 <- krige.conv(coords=Hg[,1:2], data=Hg$Hg,
                    locations=pts@coords, krige=out2)
## krige.conv: model with constant mean
## krige.conv: Kriging performed using global neighbourhood
print(c(out1$d, pout2$beta.est))    # estimate of the mean
##            beta
## 0.41654 0.41654
print(c(norm(pout2$predict - pout1), norm(okblup$var1.pred - pout1)))
## [1] 2.6231e-13 2.1808e-13
```

10.5 *Other Interpolation Approaches

There exist many flavors of the classical kriging predictor. Notably, nearest-neighbor kriging, e.g., ***gstat::krige()*** uses this approach for large datasets, prediction with tapered covariance functions, e.g., implemented in ***fields::mKrig()***. Other examples include: kriging with IRF-k (Matheron, 1973b; Delfiner, 1976), disjunctive kriging (Matheron, 1973a, 1976; Armstrong and Matheron, 1986a,b), robust kriging (Hawkins and Cressie, 1984), median-polish kriging (Cressie, 1986, Section 3.5) or using a robustified kriging predictor (Hawkins and Cressie, 1984). These variants typically involve several algorithmic parameters, and reconstructing exactly published results is often virtually impossible.

Christakos (2000) or Hristopulos and Christakos (2001) discuss prediction in the context of Bayesian maximum entropy (BME). Depending on the BME 'knowledge basis', the prediction reduces to various types of kriging.

Myers (1992) and, in a more detailed way, Nychka (2000) show that the kriging predictor is a type of thin-plate spline and thin-plate splines are kriging predictors for suitably chosen radial basis functions, smoothing parameters and covariance matrix. Therefore, it is not surprising that the package ***fields*** has kriging as well as thin-plate splines implemented (***fields::Tps()***). See also the vivid discussion of Cressie (1989b); Wahba (1990) and Cressie (1990a).

Of course, using non-stochastic interpolation methods for spatial prediction is possible. Notable examples are inverse distance weighted interpolation ***gstat::idw()***, bilinear interpolation ***fields::interp.surface()***, etc. For comparisons of different statistical and geostatistical interpolation approaches see, for example, Englund (1990); Weber and Englund (1992) and Dubois (1998, 2000). Heaton *et al.* (2019) compare many different spatial prediction algorithms in terms of RMSE and computational efficiency. See also Gerber *et al.* (2018) in the context of gapfilling satellite images.

Chapter 11

Spatial Processes: Extensions and More

A natural extension of a spatial process is the extension to multivariate spatial and to spatiotemporal processes.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter11.R.

Dependencies carry information. Hence, exploiting these as much as possible is equivalent to including as much information as possible. In this chapter, we extend the idea of one spatial process to a multivariate setting (Section 11.1) and to a spatiotemporal setting (Section 11.3). Two sections illustrate the approaches with the R-package *gstat*.

We conclude the chapter with some comments about the historical aspects of geostatistics.

11.1 Multivariate Spatial Fields

The concept of spatial processes defined as in (9.1) can be generalized in a straightforward manner to a multivariate framework. The set of random vectors

$$\big\{\mathbf{Z}(\boldsymbol{s}) = \big(Z_1(\boldsymbol{s}), \dots, Z_p(\boldsymbol{s})\big)^\top : \boldsymbol{s} \in \mathcal{D} \subset \mathbb{R}^d,\ d \geq 1\big\} \tag{11.1}$$

is called a multivariate spatial process.

As in the case of a univariate setting, we often work with a specification of the first two moments and in order to reduce the "dimensionality" thereof, we introduce different types of stationarity. For example, if for each component $Z_r(\cdot)$, $r = 1, \dots, p$, of the process, the mean is independent of the location and if, for all $r, s = 1, \dots, p$, the functions

$$c_{rs}(\boldsymbol{s}_i - \boldsymbol{s}_j) = \operatorname{Cov}\big(Z_r(\boldsymbol{s}_i), Z_s(\boldsymbol{s}_j)\big) \tag{11.2}$$

exist, then the process is called second-order stationary. For $r \neq s$, the functions (11.2) are called cross-covariance functions.

In order to generalize the concept of the variogram, two possible approaches exist and are as follows: approaches are

$$2\nu_{rs}(\boldsymbol{s}_i - \boldsymbol{s}_j) = \operatorname{Cov}\big(Z_r(\boldsymbol{s}_i) - Z_r(\boldsymbol{s}_j), Z_s(\boldsymbol{s}_i) - Z_s(\boldsymbol{s}_j)\big), \tag{11.3}$$

$$2\gamma_{rs}(\boldsymbol{s}_i - \boldsymbol{s}_j) = \operatorname{Var}\big(Z_r(\boldsymbol{s}_i) - Z_s(\boldsymbol{s}_j)\big). \tag{11.4}$$

The functions $2\nu_{rs}(\cdot)$ and $2\gamma_{rs}(\cdot)$ are called cross-variogram and pseudo cross-variogram, respectively. These second-moment functions possess slightly different advantages with respect to estimation and interpretability There are several approaches how to estimate Myers, 1991; Papritz *et al.*, 1993; Künsch *et al.*, 1997.

Often we are only interested in the prediction of a particular/primary variable $Z_k(\cdot)$. Similar to Section 10.3 a BLUP can be developed for multivariate processes. Without loss of generality, we assume $k = 1$. The starting point is the linear predictor $p(Z_1(\boldsymbol{s}_0); \mathbf{Z})$ for the primary variable $Z_1(\cdot)$ given by

$$p(Z_1(\boldsymbol{s}_0); \mathbf{Z}) = \sum_{r=1}^{p} \boldsymbol{\lambda}_r^\top \mathbf{Z}_r, \tag{11.5}$$

where $\mathbf{Z}_r = \big(Z_r(\boldsymbol{s}_1), \dots, Z_r(\boldsymbol{s}_n)\big)^\top$. The unbiasedness condition is given by $\mathbf{1}^\top\boldsymbol{\lambda}_1 = 1$ and $\mathbf{1}^\top\boldsymbol{\lambda}_r = 0$, $r = 2, \dots, n$. Minimizing the mean squared prediction error leads to a similar system of equations as (10.23), called the cokriging system. As in the case of univariate kriging approaches (ordinary, universal, . . . kriging) the system of equations can be nicely put in block-matrix notation (e.g., Myers, 1982).

Remark 11.1.

1. Unbiasedness is guaranteed if the weights of the primary variable sum to one and those of the secondary variable sum to zero. Another possibility is to impose only one constraint, i.e., $\sum_{r=1}^{p} \mathbf{1}^\top\boldsymbol{\lambda}_r = 1$. In this case, more weight is given to the secondary variables while reducing the occurrence of negative weights (Goovaerts, 1998).
2. The cokriging system can be expressed with the cross-covariance, cross-variogram, or the pseudo cross-variogram. Cressie and Wikle (1998) discuss the advantages of cross-variogram and pseudo cross-variogram usage with respect to cokriging. ♡

Another very common tool in multivariate geostatistics is coregionalization (e.g. Journel and Huijbregts, 1978). Each component $Z_r(\cdot)$, $r = 1, \dots, p$, of the process is supposed to be a linear combination of m orthogonal random processes $\big\{Y_s(\cdot)\big\}$ such that there exists a $(p \times m)$-matrix $\mathbf{A}$ satisfying $\mathbf{Z}(\cdot) = \mathbf{A}\mathbf{Y}(\cdot)$. The orthogonality of the random functions $\big\{Y_s(\cdot)\big\}$ reduces the multivariate to a univariate setting and simplifies enormously statistical modeling. However, the matrix $\mathbf{A}$ has to be estimated, often iteratively (e.g. Bourgault and Marcotte, 1991).

11.2 Example: Bivariate Spatial Interpolation

We present an example of cokriging based on the software implementation `gstat`. A much more detailed example is given in www.css.cornell.edu/faculty/dgr2/_static/files/R_PDF/CoKrigeR.pdf.

More specifically, we take the same framework shown in Section 10.4 and introduce the additional covariable cadmium (Cd). This two-variable setting is the simplest case of cokriging, and might be called bikriging. The choice of cadmium is somewhat arbitrary, and better variables or transformations might exist. We start by loading the data as in R-Code 10.4 (not shown here). We predict on the same grid we constructed in R-Code 10.6. R-Code 11.1 below loads prepares the covariable (here `Cd`), performs variogram estimation and cokriging. R-Code 11.2 compares the result to the ordinary kriging approach. Notice that we need to take the slightly more complex approach as illustrated in R-Code 10.8.

R-Code 11.1: Cokriging: loading data, variogram estimation, prediction, visualization, and validation. We assume that we have evaluated R-Codes 10.4, 10.5, 10.6 and 10.7 before we start here.

```
### To jointly model Hg and Cd, we need some dependencies:
plot(Hg~Cd, data=leman83) # one outlier!!
### We rename the dataframe, to be more flexible:
Cd <- leman83[ !duplicated(leman83[,1:2]),]
Cd <- Cd[-which.max(Cd$Cd),]
mvl <- with(Cd, data.frame(x, y, var2=Cd) )
coordinates(mvl) <- ~ x+y

### We will fit a LCM and thus have to choose a common range.
mVc <- vgm(.1, "Sph", 12, .005)
### construct the gstat object
g <- gstat(NULL, id = "var1", form = var1 ~ 1, data=uvl)
g <- gstat(g, id = "var2", form = var2 ~ 1, data=mvl)
g <- gstat(g, model=mVc, fill.all=T, set = list(nocheck = 1))
g <- fit.lmc(variogram(g), g,
             fit.method=1,            # some co-located data
             correct.diagonal=1.001)  # to be on the safe side
# plot(variogram(g), model=g$model)   # to visualize a (bad) fit
ckblup <- predict(g, pts)

## Linear Model of Coregionalization found. Good.
## [using ordinary cokriging]

# lake(ckblup$var1.pred, pts)         # visualy similar to OK

### For the script, we suppress many lines of messages:
cv1 <- gstat.cv(g, verbose=FALSE)

summary(cv1$residual)

##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.56339 -0.10515  0.00057 -0.00178  0.08204  0.58185
```

```
c(RMSE=sqrt(mean(cv1$residual^2)),
  MSDR=mean(cv1$residual^2/cv1$var1.var) )  # somewhat elevated
##    RMSE   MSDR
## 0.1780 2.4572
```

We assess now if cokriging performs better than kriging. It turns out that the summary values (RMSE and MSDR) are more favorable for ordinary kriging (end of R-Code 11.1. The LCM fit is probably a bit overly simplistic and thus leads to a too large value of the mean standard deviation ratio (2.457). Hence, “more” is not always “better”. A fully flexible cokriging model would be more appropriate, as the ranges of the variables are very different, and by choosing one common range, the nugget parameter gets inflated, offsetting the additional information from the secondary variable.

We compare differences of absolute error (maybe some ratios would be an alternative) in R-Code 11.2. In Figure 11.1, a reddish color indicates larger cokriging (absolute) errors. The histograms also show that there is not a significant gain in using cokriging. Of course, the MSPE of cokriging is smaller, as seen in the lower right panel of Figure 11.1.

R-Code 11.2: Comparing kriging and cokriging results. (See Figure 11.1.)

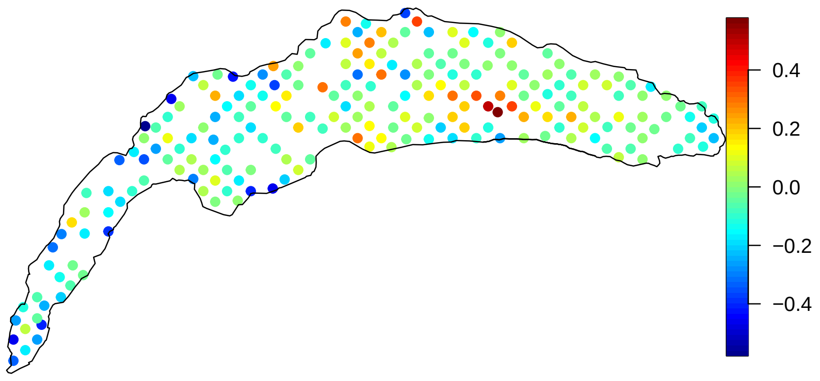
```
cvl <- Hgfull[-train,c("x","y","Hg")]      # predict on validation points
coordinates(cvl) <- ~x+y

ok1 <- predict(g1, cvl)
## [using ordinary kriging]
ck1 <- predict(g, cvl)
## Linear Model of Coregionalization found. Good.
## [using ordinary cokriging]
diff1 <- abs(ck1@data$var1.pred-cvl$Hg) - abs(ok1@data$var1.pred-cvl$Hg)

lake(diff1, cvl, zlim=range(diff1, -diff1))
par(mfcol=c(1,2))
hist(diff1, main="", col=7)
hist(ok1@data$var1.var-ck1@data$var1.var, main="", xlab="Delta MSPE", col=7)
```

11.3 Spatio-Temporal Processes

In the same way, as for spatial processes, we call the set of random variables

$$\big\{Z(\boldsymbol{s},t) : \boldsymbol{s} \in \mathcal{D} \subset \mathbb{R}^d,\, d \geq 1,\, t \in \mathcal{T}\big\} \tag{11.6}$$

a spatiotemporal process. As time is ordered in past, present and future, we cannot properly model a spatiotemporal process as $\big\{Z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D}_* \subset \mathbb{R}^{d+1},\, d \geq 1\big\}$; see also Kyriakidis and



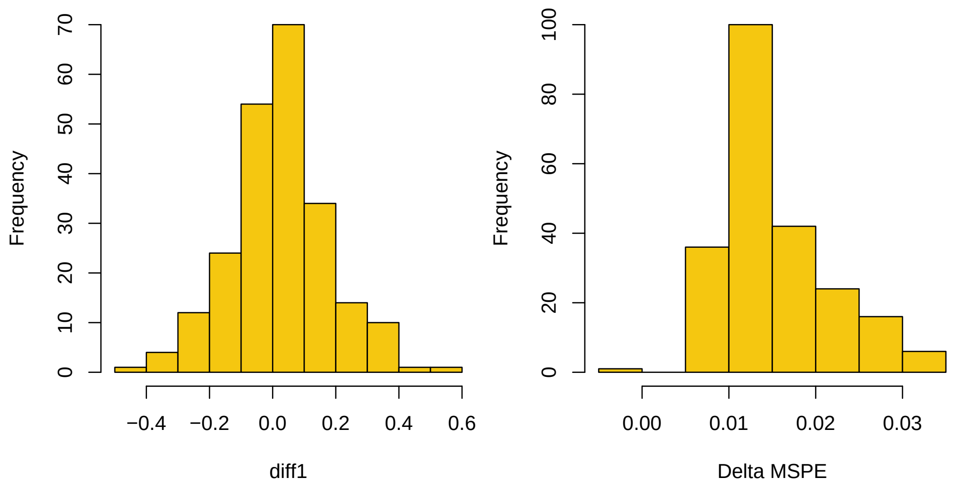


Figure 11.1: Comparing kriging and cokriging results. Differences between the cokriging (absolute) errors and the kriging (absolute) errors (top with spatial location, bottom left as a histogram). The right histogram shows the difference in the mean squared prediction error. (See R-Code 11.2.)

Journel (1999) for a detailed discussion of the differences between the space axes and the time axis. Therefore, the covariance $\text{Cov}\big(Z(\boldsymbol{s}_i, t_r), Z(\boldsymbol{s}_j, t_s)\big)$ is a function of the four arguments $\boldsymbol{s}_i$, $\boldsymbol{s}_j$, t_r and t_s. The concepts of stationarity, isotropy, etc., apply as such. For example, the spatiotemporal process $Z(\cdot,\cdot)$ is second-order stationary in space and time if

$$\text{E}\big(Z(\boldsymbol{s}_i, t_r)\big) \equiv \mu, \tag{11.7}$$

$$c\big(\boldsymbol{s}_i - \boldsymbol{s}_j, t_r - t_s\big) = \text{Cov}\big(Z(\boldsymbol{s}_i, t_r), Z(\boldsymbol{s}_j, t_s)\big), \quad \boldsymbol{s}_i, \boldsymbol{s}_j \in \mathcal{D},\ t_r, t_s \in \mathcal{T}. \tag{11.8}$$

Like spatial covariance, a spatiotemporal covariance has to be positive definite and satisfies, for example, the conditions

$$\big|c(\boldsymbol{h}, u)\big|^2 \le c(\boldsymbol{h}, 0)^2 \le c(\boldsymbol{0}, 0)^2 \quad \text{and} \quad \big|c(\boldsymbol{h}, u)\big|^2 \le c(\boldsymbol{0}, u)^2 \le c(\boldsymbol{0}, 0)^2 \tag{11.9}$$

for all spatial lags $\boldsymbol{h}$ and temporal lags u. It is common to characterize the spatiotemporal processes $Z(\cdot,\cdot)$ with their covariances. There are many different possibilities to construct positive definite covariances $c(\boldsymbol{h}, u)$. We list in what follows only a few of them; further examples are given in Cesare *et al.* (2001).

- The covariogram can be separated into functions of $\boldsymbol{h}$ and u only. For example, the product model, (Rouhani and Myers, 1990),
$$c(\boldsymbol{h}, t) = c_{\boldsymbol{h}}(\boldsymbol{h}) c_u(u), \tag{11.10}$$
or the linear model (Rodríguez-Iturbe and Mjía, 1974),
$$c(\boldsymbol{h}, t) = c_{\boldsymbol{h}}(\boldsymbol{h}) + c_u(u), \tag{11.11}$$
or a combination of the product and the linear model called the product-sum model (Cesare *et al.*, 2001).

Remark 11.2. A few more advanced methods are as follows.

- Cressie and Huang (1999, 2001) establish nonseparable spatiotemporal covariances based on integrated product models. Based on Bochner (1933), a continuous, bounded, symmetric and integrable function $c(\boldsymbol{h}, u)$, defined on $\mathbb{R}^{d+1}$, is a space-time covariance function if and only if
$$c_{\boldsymbol{\omega}}^{\mathcal{F}}(u) = \int \exp(-\imath \boldsymbol{h}^\top \boldsymbol{\omega}) c(\boldsymbol{h}, u) \, \mathrm{d}\boldsymbol{h}, \tag{11.12}$$
where $\imath = \sqrt{-1}$, is a covariance function for almost all $\boldsymbol{\omega} \in \mathbb{R}^d$. The technique consists now of finding covariance functions $c_{\boldsymbol{\omega}}^{\mathcal{F}}(u)$ and use an inverse Fourier transformation to derive the covariance function $c(\boldsymbol{h}, u)$. Cressie and Huang (1999) decompose the covariance function $c_{\boldsymbol{\omega}}^{\mathcal{F}}(u)$ in an autocorrelation function and a factor independent of u, i.e., $c_{\boldsymbol{\omega}}^{\mathcal{F}}(u) = \rho(\boldsymbol{\omega}, u) k(\boldsymbol{\omega})$.

- Gneiting (2001) presents a method based on completely monotone functions. The covariance functions are of the form
$$c(\boldsymbol{h}, u) = \frac{\sigma^2}{\psi\big(|u|^2\big)^{d/2}} \phi\big(\|\boldsymbol{h}\|^2 / \psi\big(|u|^2\big)\big), \tag{11.13}$$
where $\phi(\cdot)$ is a completely monotone function and $\psi(\cdot)$ a positive function with a completely monotone derivative. The main advantage is that this approach does not need Fourier transformations.

- Fuentes and Smith (2002) develop a new class of nonstationary spatial models based on the convolution of local stationary covariance functions. ♡

Remark 11.3. A pure nugget effect is equivalent to white noise. We use the former terminology if we have no spatial dependence and the latter if we have no temporal dependence, i.e., the nugget effect is a spatial nugget effect, and white noise is a temporal nugget effect. ♡

The observations of spatiotemporal processes are often taken at regular points in time. Depending on the application, neglecting the spatial or the temporal component is common, i.e., we interpret the spatiotemporal realization as several spatial or temporal realizations. With this simplification, it is possible to estimate covariance functions that are heterogeneous either in space or in time.

As with spatial or multivariate spatial processes, the corresponding BLUP can be developed (e.g. Bogaert, 1996). Another prediction technique combines kriging and Kalman filtering as developed by Huang and Cressie (1996).

11.4 Example: Spatio-“Temporal” Interpolation

We illustrate the spatiotemporal modeling with the same dataset. We have observations for the years 1978, 1983, and 1988. We assume again that we have constructed our grid within the lake boundaries as done in R-Code 10.6.

R-Code 11.3: Space-time-kriging: visualization and prediction. (See Figure 11.2.)

```r
library(spacetime)   # "sp" is imported
library(gstat)
load("data/LacLeman.RData")
leman83 <- leman83[ !duplicated(leman83[,1:2]),]
leman78 <- leman78[ !duplicated(leman78[,1:2]),]

set.seed(1)                 # as before!
train <- sort(sample(1:dim(leman83)[1], 68))
Hg83 <- leman83[ train, c(1,2,3)]
set.seed(14)                # new set of points
train78 <- sort(sample(1:dim(leman83)[1], 168))
Hg78 <- leman78[train78, c(1,2,4)]

### We eliminate dubious values... see message(leman.info)
pip <- point.in.polygon(leman88$x, leman88$y, lake.data$x, lake.data$y)
Hg88 <- leman88[pip==1, ]   # "==1" very important !!!

space <- SpatialPoints(rbind(Hg78[,1:2], Hg83[,1:2], Hg88[,1:2]))
time <- as.POSIXct(c(rep("1978-01-01", length(Hg78$x)), rep(
         "1978-01-02", length(Hg83$x)), rep("1978-01-03", length(Hg88$x))))
### POSIXct is the signed number of seconds since "the epoch".
### for numerical stability, we cheat with the dates!
st <- STIDF(space, time, data.frame(Hg=c(Hg78$Hg, Hg83$Hg, Hg88$Hg)))
stplot(st, number=3)
estV <- variogramST(Hg~1, st, tunit="days",tlags=0:2, cutoff=10, width=1.4)
# plot(estV, wireframe=T, scales=list(arrows=F), zlab=list(rot=90))

### models are described in vgmST
mstV <- vgmST("productSum", k=1, space=vgm(.2, "Sph", 5, 0),
              time=vgm(.01, "Sph",2, 0))
fstV <- fit.StVariogram(estV, mstV)
attr(fstV, "optim.output")$value  # to compare with other models....
## [1] 4.083e-05
plot(estV, fstV, all=TRUE)        # from gstat:::plot.gstatVariogram
```

```
tgrd <- seq(min(index(st)), max(index(st)), length=5)
pred.grd <- STF(pts, tgrd)
stblup <- krigeST(Hg~1, st, pred.grd, fstV, computeVar=TRUE)
stplot(stblup, number=5)
```

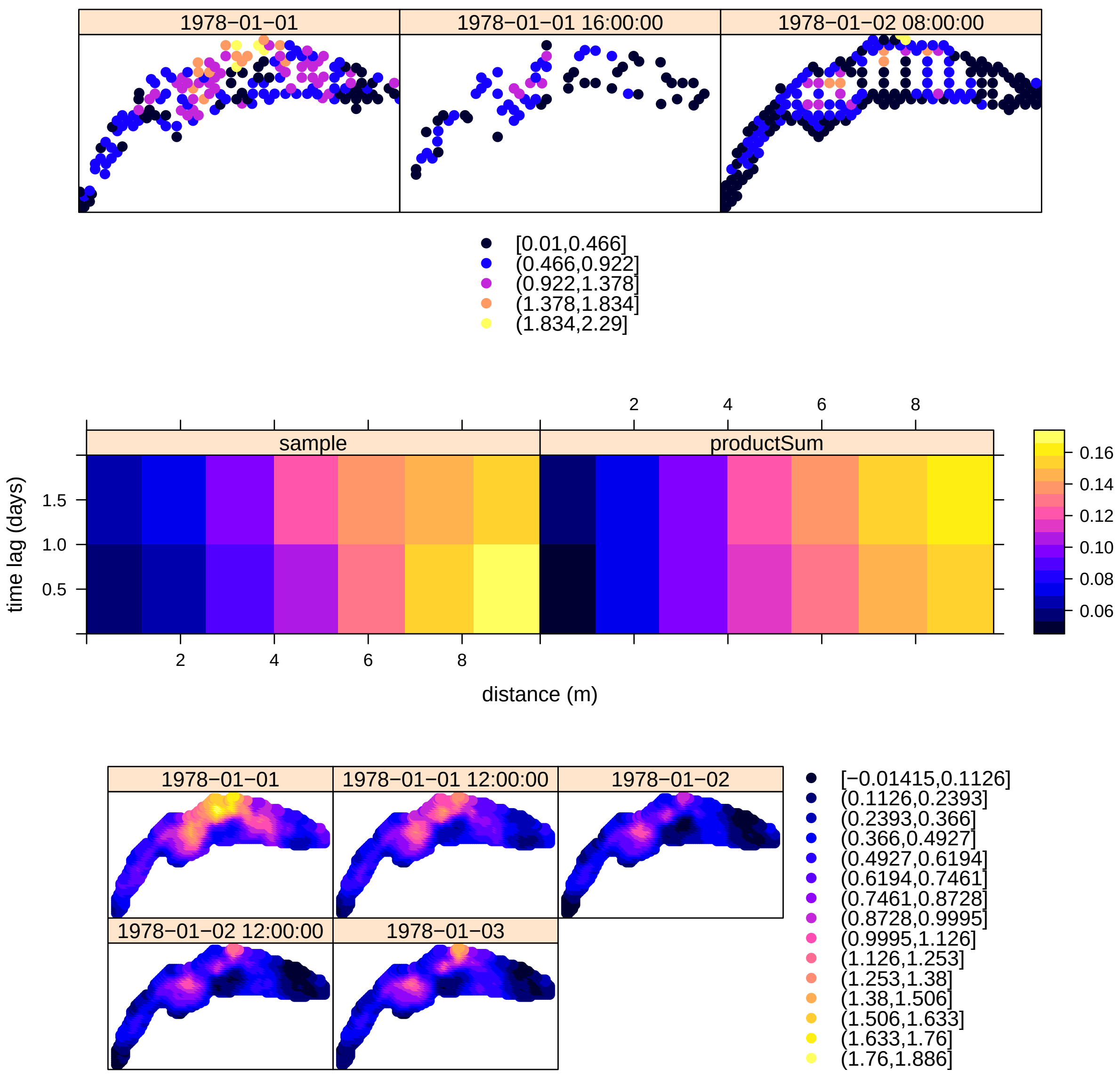


Figure 11.2: Data for three time-points (top), empirical and fitted variograms (middle layer) and predictions for space-time-kriging example (bottom) in the context of the `leman` data. (See R-Code 11.3.)

We compare the predictive performance of cokriging with space-time kriging. Based on different types of data, this comparison has only limited value — somewhat like comparing apples and prunes.

R-**Code 11.4:** Comparison of space-time kriging and ordinary kriging predictions. (See Figure 11.3.)

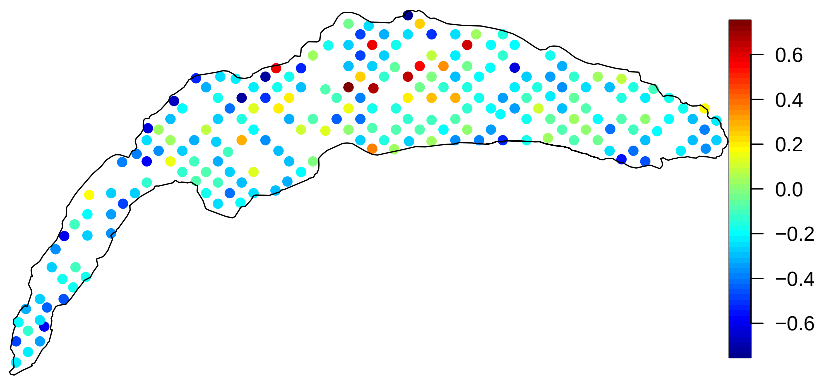
```
tgrd <- seq(min(index(st)), max(index(st)), length=3)
pred.grd <- STF(cvl, tgrd)
st1 <- krigeST(Hg~1, st, pred.grd, fstV, computeVar=TRUE)

tmp <- as.data.frame(st1)
st1 <- tmp[tmp$timeIndex==2, c("var1.pred", "var1.var")]

diff2 <- abs(st1$var1.pred- cvl$Hg) - abs(ok1@data$var1.pred- cvl$Hg)
lake(diff2, cvl, zlim=range(diff2, -diff2))
par(mfcol=c(1,2)) # panels with two superimposed histograms each
hist(diff2, main="", xlab="diffs", col=4, border="blue",
     breaks=seq(-.8,.8,.1), ylim=c(0,80))
hist(diff1, breaks=seq(-.8,.8,.1), col=rgb(.4,.4,.4,.25), add=T) # cokriging

xl <- round(range(ok1@data$var1.var - st1$var1.var,          # get range for
            ok1@data$var1.var - ck1@data$var1.var), 2)      # a nice histogram
hist(ok1@data$var1.var - st1$var1.var, main="", xlab="Delta MSPE", col=4,
     border="blue", breaks=seq(xl[1]-.01, xl[2]+.01, by=.01), ylim=c(0, 140))
hist(ok1@data$var1.var - ck1@data$var1.var, col=rgb(.4,.4,.4,.25),
     breaks=seq(xl[1]-.01, xl[2]+.01, by=.01), add=T)
```

11.5 *A Few Historical Remarks About Geostatistics

Despite its name, the development of geostatistics was not solely driven by the application of statistical tools to problems in geosciences. By the early 1970's, the term started to be a reference to the tools and techniques developed by Georges Matheron, who was strongly influenced by the Russian school of probabilists. Actually, the more modern terminology 'statistics of spatial processes' gives a more adequate description of his work.

Matheron was Professor at the 'École Normale Supérieure des Mines de Paris' and founded the 'Centre de Morphologie Mathématique' in Fontainebleau in the course of a general movement of research units out of the city of Paris. Later on, the center was split into two parts, in which geostatistics was regrouped in the 'Centre de Géostatistique, École des Mines' in Fontainebleau, referred to as 'Center' in what follows. J.-P. Chiles, P. Delfiner, A. Journel, and M. David were part of the initial research group. At this time, the share of applications in Matheron's work became more important, focusing mainly on geophysical problems, like the reserve estimation problem in the mining and petroleum industry or various interpolation problems in hydrology.

From the early 1970's on, geostatistics became moderately well known in the above fields, namely in the mining and petroleum industry, as well as in hydrology. Before, its popularity was limited to France as most publications (for example, Matheron, 1962, 1963, 1969) were in French. Notably, its absence in the statistical literature was striking. This unpopularity was



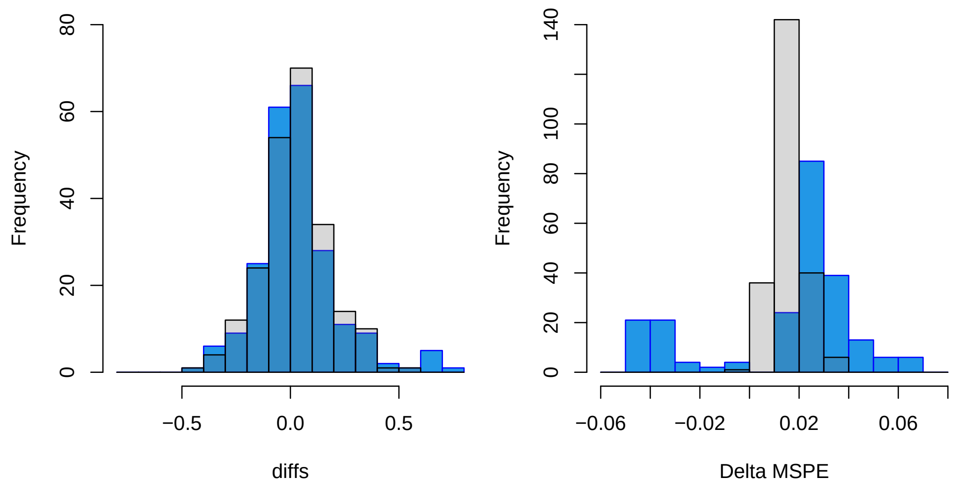


Figure 11.3: Comparison of space-time kriging and ordinary kriging predictions. Lower panels: Gray and blue for cokriging and space-time kriging. (See R-Code 11.4.)

amplified by the fact that two of the other leading researchers in the area initially published in rather 'exotic' languages as well: Matérn (1960) in Swedish and Gandin (1963) in Russian. The translations of these theoretical foundations into English became available only later through Matheron (1971, 1973b); Matérn (1986) and Gandin (1965). With those and the 1975 'NATO Advanced Study Institute' meeting, the popularity of geostatistics in the international scientific community began to grow.

Geostatistics is often associated with the term 'variogram', the variation of the increment of the process. The term is due to Matheron (1962) although there are other terminologies like 'structure function' or 'mean squared difference' that denote the same characteristic but appeared earlier in the literature; see Cressie (1988), for historical notes. This provoked a feeling that some of Matheron's work consisted only in restating known results using different names, which was in part responsible for the long time the lasting difficulty of acceptance of his work in the statistical community. Matheron's propensity to mainly publish his work in 'internal notes' in French at the Center probably contributed to this perception. While it was possible to order copies from the Center there was no generally accessible repository outside of the Center. Armstrong (1982) published a, by now out-dated, index of the internal notes and assisted in publishing some of them in scientific journals.

The applications of the 'founders' of geostatistics were in the mining and petroleum industry, hydrology (Matheron), forestry (Matérn), and meteorological and atmospherical sciences (Gandin). A series of important papers by R. Webster and his colleagues (Burgess and Webster, 1980a,b; Webster and Burgess, 1980 and Burgess *et al.*, 1981) from the 'Rothamstead Experimental Station' launched the applications in soil science. Applications in the environmental sciences and ecology began to appear in the 1980s.

By the mid-1980s, geostatistics had been established within the statistical community, taking its shares in statistical journals and international meetings. In 1968 the 'International Association of Mathematical Geologists' was founded in Prague, publishing a year later its own journal, which is now called 'Mathematical Geosciences' and is one of the leading journals in the field.

For many years the book by Journel and Huijbregts (1978) was the only available textbook providing a broad overview of applied geostatistics. Ripley (1981) contains an early concise statistical account of spatial data analysis. Cressie (1993) provides a large part of his book on geostatistics, combining the theoretical statistical aspects successfully with the mere practice-driven side of geostatistics, compared to Goovaerts (1997), which is more practice-driven. Wackernagel (1995) provides a first approach to multivariate geostatistics, later editions improved significantly (e.g., Wackernagel, 2006). Chilès and Delfiner (1999) is one of the later textbooks. Less attention has been paid to the recent development of the more statistical side of geostatistics. Olea (1999) and Webster and Oliver (2001) provide introductions to geostatistics for a more applied audience. Recently, the textbooks devoted to spatial statistics flourished, as a search on any bookseller illustrates. A short list constitutes Banerjee *et al.* (2003); Rue and Held (2005); Schabenberger and Gotway (2005); Diggle and Ribeiro Jr. (2007); Bivand *et al.* (2008) and many edited volumes, such as Møller (2003). Some treaties are available online, e.g., spatial-analyst.net/book/sites/default/files/Hengl_2009_GEOSTATe2c1w.pdf

Cokriging was extensively studied in the late 1980s and early 1990s (e.g. Davis and Greenes, 1983; Abourfirassi and Marino, 1984 or Carr and McCallister, 1985). As the cokriging system is of order $\mathcal{O}(n^3)$, practitioners often try to reduce the number of equations by taking only into account of a certain amount of the neighbors of the location $\boldsymbol{s}_0$. Myers (1983); Long and Myers (1997) and Furrer and Genton (2011) provide other solutions to reduce the computational burden of cokriging.

Implementations of geostatistical methods began with a software package called BLUEPACK in the late 1970s, developed in cooperation with 'Shell Oil' and the 'Bureau de Recherche Géologie Mathématique'. Later examples are GSLIB, developed at Stanford University from 1985 on (Deutsch and Journel, 1998) and VARIOWIN, (Pannatier, 1995, 1996). R and its libraries (*fields*, *geoR*, *gstat* or *RandomFields*, *spatial*, to name just a few) offer a wide variety of software elements.

Automatic monitoring stations and almost unlimited storage capacities bring classical geostatistics toward new frontiers. Terms like 'data-rich in time and/or space' are frequently used. The massive amount of data allows different statistical methodologies. We have sufficient many data to relax the stationarity hypothesis or to work in spectral domains.

11.6 *Further Technical Details and More Bibliographic Insights

Cressie's additive decomposition The most commonly used and widely accepted decomposition for isotropic processes is based on additive separation according to different scales. Following Cressie (1993, Pages 112–113) we write

$$Z(\boldsymbol{s}) = \mu(\boldsymbol{s}) + U(\boldsymbol{s}) + V(\boldsymbol{s}) + \varepsilon(\boldsymbol{s}), \qquad \boldsymbol{s} \in \mathcal{D}, \tag{11.14}$$

where each term accounts for a variation at a certain scale, namely:

- $\mu(\cdot) = \mathrm{E}\big(Z(\cdot)\big)$ is the deterministic mean structure, called large-scale variation;
- $U(\cdot)$ is a zero-mean, L_2-continuous (i.e., $\mathrm{E}\big(Z(\boldsymbol{s}+\boldsymbol{h})-E(\boldsymbol{s})\big) \to 0$ for $\boldsymbol{h} \to \boldsymbol{0}$) and intrinsically stationary process whose variogram attains its maximum in a (possibly infinite) value larger than $\min\big\{\|\boldsymbol{s}_i-\boldsymbol{s}_j\|\big\}$, $1 \le i < j \le n$. The process $U(\cdot)$ is called smooth small-scale variation;
- $V(\cdot)$ is a zero-mean and intrinsically stationary process, independent of $U(\cdot)$, whose variogram attains its maximum in a value smaller than $\min\big\{\|\boldsymbol{s}_i - \boldsymbol{s}_j\|\big\}$, $1 \le i < j \le n$. The process $V(\cdot)$ is called micro-scale variation;
- $\varepsilon(\cdot)$ is a zero-mean white-noise process, independent of $U(\cdot)$ and $V(\cdot)$ and is considered as measurement error.

Spectral approach to representing covariance functions A continuous function $c(\boldsymbol{h})$ is positive definite if and only if it has a spectral representation

$$c(\boldsymbol{h}) = \int_{\mathbb{R}^d} \cos(\boldsymbol{\omega}^T \boldsymbol{h}) G(\mathsf{d}\boldsymbol{\omega}), \tag{11.15}$$

where $G(\cdot)$ is a bounded symmetric measure on $\mathbb{R}^d$ (Bochner, 1933). For an extensive overview of positive definite functions, see Stewart (1976). We suppose that $c(\boldsymbol{h})$ is integrable, i.e., $\int_{\mathbb{R}^d} c(\boldsymbol{h})\,\mathsf{d}\boldsymbol{h} < \infty$, this implies that the spectral density $g(\boldsymbol{\omega})\,\mathsf{d}\omega/c(\boldsymbol{0})$ exists. Under this point of view, the (normed) covariance function is the Fourier transform of the spectral density, i.e., $c(\boldsymbol{h}) \propto g^{\mathcal{F}}(\boldsymbol{\omega})$. To any positive definite function corresponds a d-dimensional (complex) process (see Yaglom, 1957 for a rigorous treatment). The former approach is also used to construct additional covariance functions.

In time series modeling, spectral approaches are common and widely used, partially due to the fact that the observations are equispaced and fast computational algorithms exist.

Outliers Real datasets often contain outliers, i.e., an outlying observation, that appears to deviate markedly from other members of the sample in which it occurs (Grubbs, 1969; Muñoz-Garcia *et al.*, 1990). While we do not model outliers on a theoretical level, neither do we discuss the origin of the abnormality or the associated identification problem (e.g., Cerioli and Riani, 1999 or Velasco *et al.*, 2000), we are nevertheless concerned about their presence and apply robust methods to minimize the effect of erroneous observations on our estimates.

Stability of kriging It is common to divide the data into a training set and a validation set to compare the accuracy and performance of different prediction methods with Monte Carlo simulations. The parameters are estimated with the training set, and the validation set is used to calculate different error statistics such as the root mean squared error (RMSE), the median absolute deviation (MAD) and the mean absolute error (MAE). Other comparison criteria based on cross-validation are given by Carroll and Cressie (1996) or Carroll and Cressie (1997).

Theoretical results are based on perturbation theory. Depending on the aim of the study, a suitable neighborhood for the variogram is chosen. Recall that the smoothness of the variogram at the origin translates directly into the local smoothness of the process. Consequently, the (co)variograms must have the same behavior at the origin. Diamond and Armstrong (1984) quantifies the robustness of variograms. They define a so-called δ-neighborhood for variograms and show why the simple absolute difference of variograms is an inadequate measure of difference. They prove that the ratio of the kriging weights and its perturbed counterpart is bounded and a function of the condition number of the kriging matrix.

Cressie and Zimmerman (1992) show that, while the kriging predictor is generally stable, the mean squared prediction error depends heavily on the covariance or variogram and is not as stable as the kriging predictor. Note, however, that Brooker (1986) concludes that the kriging variance is robust to most errors likely to be made in variogram model selection. However, if a nugget effect near zero is selected instead of a nonzero value, the kriging variance can be understated significantly.

Asymptotics If the true variogram is known and continuous at the origin, then the mean squared prediction error converges to zero with rate $\mathcal{O}(N^{2/d})$ under infill asymptotics. Yakowitz and Szidarovszky (1985, Theorems 2.1 and 2.3) show that if the true variogram is unknown, the kriging predictor will nevertheless converge under certain circumstances to the true value. Their simulation study shows, that if the kriging hypotheses are met, the kriging predictor performs better than nonparametric regression techniques.

Stein (1988); Stein and Handcock (1989) and Stein (1990) discuss asymptotic properties of kriging when the covariance function is misspecified. They introduce the concept of compatible covariance and show that under the hypotheses of a consistent predictor, of two compatible covariance functions, and of infill asymptotics we have asymptotic efficiency and an asymptotic negligible relative error if the prediction is based on the wrong covariance. Additionally, Stein and Handcock (1989) show that the spherical covariance is asymptotically less efficient and recommend using the Matérn class; see also Stein (1999).

More about estimation Lark (2000) compares variogram estimators with extensive simulations. Further robust location and scale variogram estimators can be found in Armstrong and Delfiner (1980); Dowd (1984) and Dutter (1996).

Standard examples of ML estimation in spatial settings are given in Mardia and Marshall (1984) and Kitanidis and Lane (1985). Warnes and Ripley (1987) show that maximization often bears problems. Recall that the ML estimates of the covariance parameters are biased; see, for instance, Cressie (1993, Paragraph 2.6.3 and the references therein). However, the bias is often of lesser concern due to data sizes. Watkins and Al-Boutiahi (1990) study the effects of model misspecification on ML estimates of parameters of a particular family of covariances.

A completely different approach is nonparametric estimation and modeling. For an accessible overview, see Section 4.6, of Schabenberger and Gotway (2005). Shapiro and Botha (1991) develop a nonparametric fitting method based on the spectral representation of the covariance function; see also Cherry *et al.* (1996) for a numerical case study.

Alternative estimation procedures are based on generalized estimating equations, an application of estimating function theory and quasi-likelihood (see, e.g., Schabenberger and Gotway, 2005, Section 4.5.3.1) or are minimum norm quadratic estimation (Stein, 1987), see also Cressie (1993, Paragraph 2.6.3 and the references therein).

As a final Reference, Olea (1991) gives a detailed glossary of geostatistical terms.

Chapter 12

Spatial Point Processes

This chapter briefly introduces the analysis of data observed in the form of a set of points distributed over some region of space. Such a dataset is typically called a spatial point pattern.

R-Code for this chapter: www.math.uzh.ch/furrer/download/sta330/chapter12.R.

12.1 Introduction

Figure 12.1 shows the location of 2251 trees according to their botanical classification (black oaks, hickories, maples, miscellaneous trees, red oaks, and white oaks). The region is a 924 by 924 feet (19.6 acres) plot in Lansing Woods, Clinton County, Michigan USA, (Gerrard, 1969). The data is available as `lansing` from the package `spatstat`.

We clearly have "spatial data". However, the location of the "observation" is determined, and *defines* the data. This example completely differs from what we have seen in the previous chapters. Loosely speaking, we have a spatial point pattern.

Other classical examples of spatial point patterns are:

- Cholera cases in a city,
- lightning strikes during a storm,
- locations of material defects in an alloy,
- epicenters of earthquakes along a fault line,
- locations of ribosomes in a prokaryotic cell.

Often, the locations may be equipped with an attribute: the earthquake's magnitude in the last example or the botanical classification of a tree in the Lansing Woods dataset.

The principal goal of spatial point process statistics is to analyze the geometrical structure of patterns formed by events that are distributed in space. The events represented by locations can be seen as objects whose (short-range) interactions are of interest.

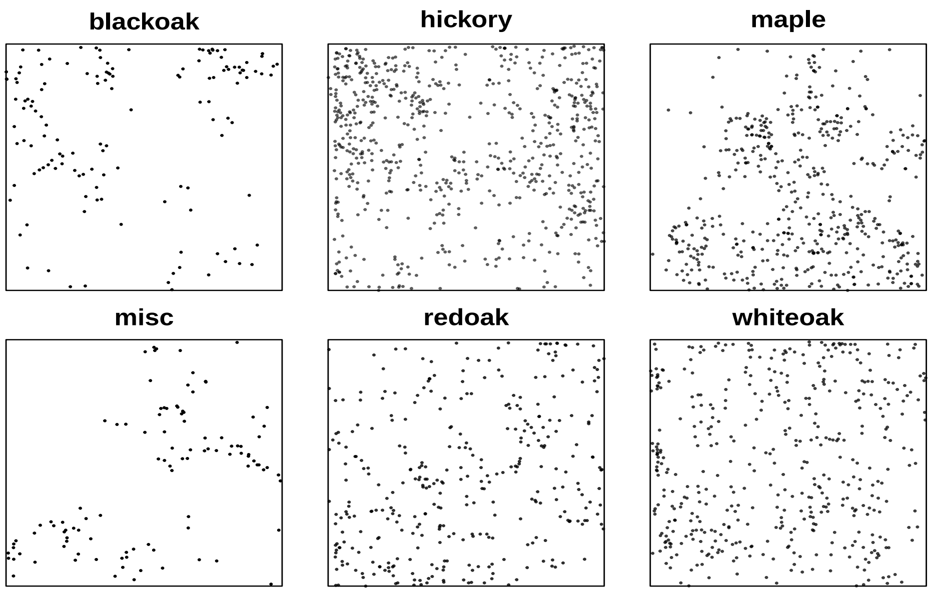


Figure 12.1: The locations of 2251 trees according to their botanical classification in Lansing Woods, Clinton County, Michigan, USA. The plot is essentially `plot(split(spatstat.data::lansing),main="")`.

More formally, a *spatial point pattern* is a set of locations $\{\boldsymbol{s}_1, \boldsymbol{s}_2, \dots, \boldsymbol{s}_n\}$, $\boldsymbol{s}_i \neq \boldsymbol{s}_j$, $\forall i \neq j$, often called events, that are irregularly distributed within a designated region and presumed to have been generated from some form of stochastic mechanism, referred to as a *spatial point process*.

More formally, a spatial point process is a stochastic (locally finite) set $\{\mathbf{S}_1, \mathbf{S}_2, \dots\} \subset \mathbb{R}^d$, $\mathbf{S}_i \neq \mathbf{S}_j$, $\forall i \neq j$, with $d \geq 1$. Locally finite means that in each finite set $\mathcal{B}$, only a finite number of locations $\mathbf{S}_i$ occur. The case of $d = 2$ is the most relevant and will be considered here exclusively. In analogy to the previous chapters and by severely abusing the standard mathematical notation, we could write a spatial point process as

$$\{Z(\boldsymbol{s}) : \boldsymbol{s} \in \mathcal{D} \subset \mathbb{R}, d \geq 1\}, \tag{12.1}$$

with $\mathcal{D}$ being random, a collection of random events whose realization is called the spatial point pattern.

For all sets $\mathcal{B} \subset \mathbb{R}^2$, we define the random variable

$$N(\mathcal{B}) = \sum_i 1_{\mathcal{B}}(\mathbf{S}_i) = \sum_i \{\mathbf{S}_i \in \mathcal{B}\} = \text{number of locations in the set } \mathcal{B}. \tag{12.2}$$

A formal approach to point patterns would be based on all finite-dimensional distributions of $\big(N(\mathcal{B}_1), \dots, N(\mathcal{B}_k)\big)$, $\mathcal{B}_i \subset \mathbb{R}^2$, $k = 1, 2, \dots$.

Definition 12.1. A point process Z is called stationary (isotropic) if all finite-dimensional distributions $\big(N(\mathcal{B}_1), \dots, N(\mathcal{B}_k)\big)$ of the point process are invariant with respect to translation (rotation) of the $\mathcal{B}_i$'s. ♣

While the domain of analysis $\mathcal{D}$ is often (implicitly) given or fixed, there are many different ways to record the events. We distinguish essentially between *intensive mapping* and *sparse sampling*. In the case of the former, all events are recorded with the exact position. In the case of the latter, only summary statistics of the events are available. For example, the number of counts in sub-blocks or the distances to the (few) nearest events are recorded. In any case, replicated sampling is very rare.

The data size of point patterns is usually small to moderate (the Lansing Woods dataset is large for its type) with all types of domain shapes. The domain acts like a censoring device, i.e., only events within the domain are observed. Hence, an essential aspect of point patterns analysis is to take into account edge effects induced by the domain. An extensive literature exists on how to address the edge effects, and most statistical procedures take these into account.

In R, the packages `spatstat` and `splancs` provide a comprehensive set of functionalities to analyze spatial point patterns. Especially the former comes with well-written help pages, cf. `help(spatstat)`.

12.2 Summary Descriptives

A point patterns analysis's first step involves examining summary statistics of the distribution of events, often followed by (non)-parametric modeling (model fitting, model validation, ...).

The baseline model is complete spatial randomness (CSR), defined as

[CSR1] The number of events in any planar region $\mathcal{A}$, is distributed according to a Poisson random variable with mean λ times the area of $\mathcal{A}$.

[CSR2] Given n events $\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_n\}$, they are uniformly distributed in $\mathcal{A}$.

In [CSR1], the constant $\lambda > 0$ is called the *intensity*, the mean number of events per unit area. Further, [CSR2] implies that there are no interactions among the events, i.e., there is no encouragement or inhibition for further events in the neighborhood of an event $\boldsymbol{s}_i$. However, such random patterns often lead to illusory visual impressions of aggregation, as illustrated in Figure 12.2.

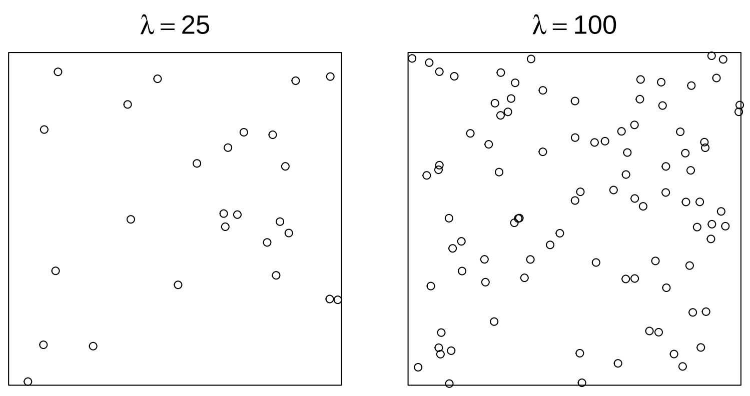


Figure 12.2: Two realizations of CSR based on $\lambda = 25$, yielding $n = 26$ (left) and $\lambda = 100$, yielding $n = 93$ (right) events in the unit square. (See R-Code 12.1.)

R-**Code 12.1** Generating two realizations of CSR based on different intensities. (See Figure 12.2.)

```
library(spatstat)
set.seed(15)
CSR <- rpoispp(25)
print(CSR)
## Planar point pattern: 26 points
## window: rectangle = [0, 1] x [0, 1] units
plot(CSR, main=expression(lambda==25))
CSR <- rpoispp(100)
print(CSR)
## Planar point pattern: 93 points
## window: rectangle = [0, 1] x [0, 1] units
plot(CSR, main=expression(lambda==100))
```

While observed point patterns rarely obey the CSR hypothesis, tests for CSR (with confidence bands) often discriminate between aggregated or regular point patterns. As CSR tests notoriously have extremely low power, graphical tests are preferred and help with modeling choices. Three classical tests are based on the following "functions".

***H*-function:** interevent distances.

Let t_{ij} be the distance between the events $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$. Under CSR the theoretical distribution of t_{ij}, say T, is known for simple regions $\mathcal{A}$ and its cdf is denoted as $H(t)$.
A graphical assessment is plotting

$$\widehat{H}_{\text{obs}}(t) = \frac{2}{n(n-1)} \sum_{i \neq j} I(t_{ij} \leq t) \text{ versus } H(t) \tag{12.3}$$

and judging the deviance from the straight line.

However, $\text{Var}(T)$ is not known, and to assess the significance of the null hypothesis CSR, the sampling distribution is determined empirically from the following pseudo-code.

```
[1]   For $i$ in 1 to $N$:
[1a]      Generate a point pattern with $n$ events under CSR
[1b]      Determine $\widehat{H}_i(t)$, $i = 1, \dots, N$, similarly as described above
[2]   $\widehat{H}_{\text{obs}}(t)$ is compared (pointwise) with the quantiles of $\widehat{H}_1(t), \dots, \widehat{H}_N(t)$
```

While the H-function is based on all inter-distances, an assessment of a "nearest" distance is often sufficient.

G-function: nearest neighbor distance.

For n events $\boldsymbol{s}_i$, let $d_i = \min_j \|\boldsymbol{s}_i - \boldsymbol{s}_j\|$ and define

$$\widehat{G}_{\text{obs}}(d) = \frac{1}{n} \sum_i I(d_i \leq d). \tag{12.4}$$

The theoretical distribution of d_i does not exist in closed form, mainly due to edge effects. It needs to be assessed empirically, e.g., by the (pointwise) mean of $\widehat{G}_1(d), \dots, \widehat{G}_N(d)$. The sample is also used to obtain (pointwise) quantiles. As a crude approximation, it is possible to use $G(d) \approx 1 - (1 - \pi d^2/|\mathcal{A}|)^{n-1}$.

F-function: (arbitrary) point-to-nearest-event distance.

For m-sample events $\boldsymbol{s}_j^*$, let $y_j = \min_i \|\boldsymbol{s}_j^* - \boldsymbol{s}_i\|$ and define

$$\widehat{F}_{\text{obs}}(y) = \frac{1}{m} \sum_j I(y_j \leq y). \tag{12.5}$$

Comments similar to the G-function apply here.

The functions *Fest()* and *Gest()* of the package *spatstat* provide the means to estimate the functions, as illustrated in R-Code 12.2. Figure 12.3 shows the observed estimates as well as the lower, upper envelope, and the (approximate) theoretical value of 99 samples under CSR for the point patterns displayed in Figure 12.2 (left column) and the black oak locations of the Lansing Wood data (top left panel of Figure 12.1). Because m can be arbitrarily large, the estimated curves are much smoother.

R-Code 12.2 Observed estimates of F- and G-functions as well as the lower, upper envelope and the (approximate) theoretical value of 99 samples under CSR and the black oak locations of the Lansing Wood data. (See Figure 12.3.)

```r
set.seed(15)
CSR <- rpoispp(25)                    # synthetic data
GCSR <- envelope(CSR, fun=Gest, verbose=FALSE)
FCSR <- envelope(CSR, fun=Fest, verbose=FALSE)
sample <- split(lansing)$blackoak    # data from package spatstat.data
Gsam <- envelope(sample, fun=Gest, verbose=FALSE)
Fsam <- envelope(sample, fun=Fest, verbose=FALSE)
### Now plotting in a 2x2 arrangement:
plot(GCSR, main="CSR", legendargs=list(bty="n"))
tmp <- plot(FCSR, main="CSR", legendargs=list(bty="n"))
tmp <- plot(Gsam, main="Black Oak", ylim=c(0, 1), legendargs=list(bty="n"))
tmp <- plot(Fsam, main="Black Oak", legendargs=list(bty="n"))
```

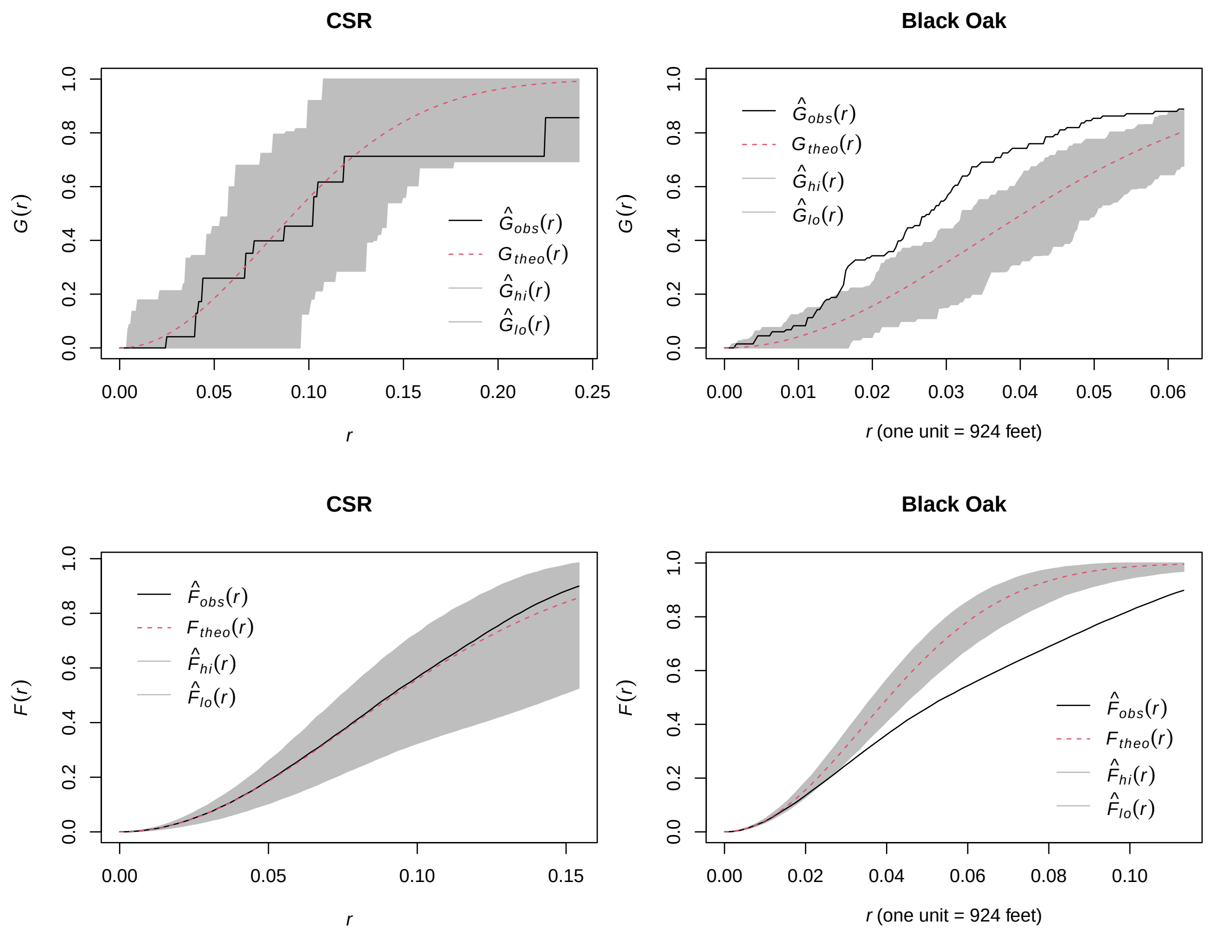


Figure 12.3: Observed estimates of F- and G-functions as well as the lower, upper envelope and the (approximate) theoretical value of 99 samples under CSR for the point patterns displayed in Figure 12.2 (left column) and the black oak locations of the Lansing Wood data (top left panel of Figure 12.1). (See R-Code 12.2.)

12.3 First- and Second-Order Measures

The first-order measure essentially describes how the events in a planar region $\mathcal{A}$ are distributed. We denote with $|\mathcal{A}|$ the area of $\mathcal{A}$ and with $d\boldsymbol{s}$ an infinitesimal region which contains the point $\boldsymbol{s}$.

Definition 12.2. The intensity function of a point process is defined as

$$\lambda(\boldsymbol{s}) = \lim_{|d\boldsymbol{s}|\to 0} \frac{\mathrm{E}\big(N(d\boldsymbol{s})\big)}{|d\boldsymbol{s}|}. \tag{12.6}$$

♣

For a stationary point process, $\lambda(\boldsymbol{s}) \equiv \lambda$ represents the mean number of points per unit area. We now look at the interaction of individual events, characterized by the following definition.

Definition 12.3. The second-order intensity function of a point process is defined as

$$\lambda_2(\boldsymbol{s}, \boldsymbol{y}) = \lim_{|d\boldsymbol{s}|,|d\boldsymbol{y}|\to 0} \frac{\mathrm{E}\big(N(d\boldsymbol{s})N(d\boldsymbol{y})\big)}{|d\boldsymbol{s}||d\boldsymbol{y}|}. \tag{12.7}$$

♣

As in the case of variograms and covariance functions in the context of spatial processes, we have for a stationary process $\lambda_2(\boldsymbol{s}, \boldsymbol{y}) = \lambda_2(\boldsymbol{s} - \boldsymbol{y})$ and for a stationary, isotropic process $\lambda_2(\boldsymbol{s}, \boldsymbol{y}) = \lambda_2(\|\boldsymbol{s} - \boldsymbol{y}\|)$ (note the abuse of notation).

An alternative description of the second-order intensity of a stationary, isotropic process is the so-called K-function.

Definition 12.4. The K-function for a stationary, isotropic process is defined as

$$K(t) = \frac{1}{\lambda} E\big(N_o(t)\big), \tag{12.8}$$

where $N_o(t)$ is the number of events in a disk of radius t centered at an arbitrary event. ♣

Property 12.1. *The following relations between the K-function and the second-order intensity function hold.*

$$\lambda K(t) = 2\pi \frac{1}{\lambda} \int_0^t x \lambda_2(x) \, \mathsf{d}x; \tag{12.9}$$

$$\lambda_2(t) = \frac{\lambda^2}{2\pi t} K'(t). \tag{12.10}$$

From a theoretical point of view, it is often more convenient to work with $\lambda_2(t)$ than with $K(t)$. However, $K(t)$ can be more easily estimated from a set of data.

A natural estimator for the intensity function is

$$\widehat{\lambda}(\boldsymbol{s}) = \frac{N\big(\mathcal{W}(\boldsymbol{s})\big)}{\big|\mathcal{W}(\boldsymbol{s})\big|} \tag{12.11}$$

for some (spatial) window $\mathcal{W}$ that contains the spatial location $\boldsymbol{s}$. While this estimator is intuitive, it is biased (edge effects) and for non-stationary processes, it is nontrivial to balance the bias and variance of the estimator.

Here, we only focus on estimating the K-function in the case of an isotropic point pattern. Notice that from an estimate of the K-function, we can derive the following estimate for the second-order intensity:

$$\widehat{\lambda}_2(t) = \widehat{\lambda}^2 \frac{1}{2\pi t} \cdot \frac{1}{h} \Big(\widehat{K}(t+h) - \widehat{K}(t) \Big). \tag{12.12}$$

To derive an estimator for the K-function, we start by constructing an estimator for $E_o(t) = \mathrm{E}\big(N_o(t)\big)$. If $u_{ij} = \|\boldsymbol{s}_i - \boldsymbol{s}_j\|$, a natural (but negatively biased) estimator is

$$\widehat{E}_o^*(t) = \frac{1}{n} \sum_{i=1}^n \sum_{j \neq i} I(u_{ij} \leq t). \tag{12.13}$$

The bias can be corrected by correctly weighing the individual terms. Ripley (1976) proposes the following unbiased estimator

$$\widehat{E}_o(t) = \frac{1}{n} \sum_{i=1}^n \sum_{j \neq i} \frac{1}{\omega_{ij}} I(u_{ij} \leq t), \tag{12.14}$$

where ω_{ij} is the proportion of the circle's circumference centered at $\boldsymbol{s}_i$ with radius u_{ij}. (Note that $\omega_{ij} \neq \omega_{ji}$.) Finally,

$$\widehat{K}(t) = \frac{|\mathcal{A}|}{n-1}\widehat{E}_o(t). \tag{12.15}$$

Several closed-form approximations for $\operatorname{Var}\big(\widehat{K}(t)\big)$ exist, and we refer again to Diggle (2003) for more details.

12.4 Models for Point Processes

In this section, we provide four essential models for point processes. The models are illustrated with realizations in the unit square and obtained from the package *spatstat*.

12.4.1 Homogeneous Poisson Process (HPP)

The HPP is the most basic (natural) model. It is often called a Poisson process.

[HPP1] For some $\lambda > 0$ and finite planar region $\mathcal{A}$, $N(\mathcal{A}) \sim \mathcal{P}(\lambda \cdot |\mathcal{A}|)$.

[HPP2] Given $N(\mathcal{A}) = n$, the n events in $\mathcal{A}$ are an independent sample from the uniform distribution on $\mathcal{A}$.

Remark 12.1.
- The HPP corresponds to CSR.
- It is straightforward to show that [HPP1] and [HPP2] imply that if $\mathcal{A} \cap \mathcal{B} = \emptyset$, $N(\mathcal{A})$ is independent of $N(\mathcal{B})$.
- $\lambda_2(t) = \lambda^2$, $t > 0$.
  $K(t) = \pi t^2$, $t > 0$.
  $\operatorname{Var}\big(N(\mathcal{A})\big) = \lambda \cdot |\mathcal{A}|$.
  $F(t) = G(t) = \Pr\{N(\pi t^2) > 0\} = 1 - \exp(-\lambda\pi t^2)$, $t > 0$. ♡

Figure 12.4 gives two realizations of an HPP. Notice the apparent clustering. In *spatstat*, it is possible to use *rpoispp()* and *rpoint()* to construct HPP. The former requires λ (points per unit area), used to construct Figure 12.2, and the latter allows to constrain the number of events (based on [HPP2] only) used to construct the panels of Figure 12.4.

R-**Code 12.3** Generating two realizations of an HPP. (See Figure 12.4.)

```
set.seed(15)
HPP <- rpoint(50)
plot(HPP, main=expression(n==50))
HPP <- rpoint(250)
plot(HPP, main=expression(n==250))
```

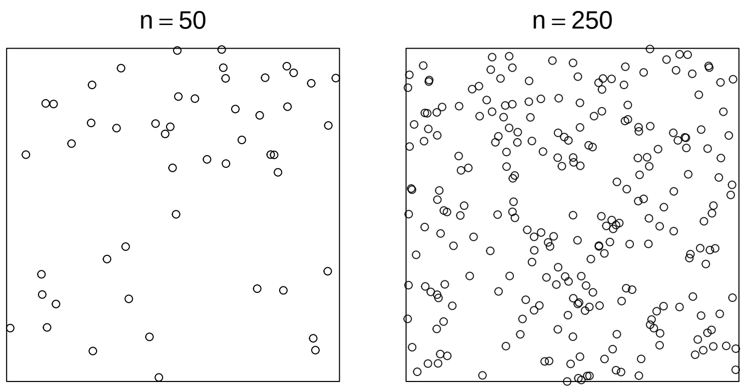


Figure 12.4: Two realizations of a HPP with $n = 50$ (left) and $n = 250$ (right) events in the unit square. (See R-Code 12.3.)

12.4.2 Inhomogeneous Poisson Process (IPP)

Instead of a constant $\lambda > 0$, the IPP assumes an intensity that varies over space $\lambda(\boldsymbol{s}) > 0$, for all $\boldsymbol{s}$.

[IPP1] For some $\lambda(\boldsymbol{s}) > 0$ and finite planar region $\mathcal{A}$, $N(\mathcal{A}) \sim \mathcal{P}\Big(\int_{\mathcal{A}} \lambda(\boldsymbol{s})\,\mathsf{d}\boldsymbol{s}\Big)$.

[IPP2] Given $N(\mathcal{A}) = n$, the n events in $\mathcal{A}$ are an independent sample from the distribution on A with pdf proportional to $\lambda(\boldsymbol{s})$.

The IPP is not stationary but is a natural extension of the HPP.

Figure 12.5 gives two realizations of an IPP with $n = 250$ events. The panels are based on an intensity $\lambda(\boldsymbol{s}) = \lambda(x, y)$ proportional to $x + y$ (left), and $\exp\big(-(x^2 + y^2)/0.3\big)$ (right) in the unit square. Notice that even with moderately large n, the detailed structure of $\lambda(x, y)$ cannot be assessed visually. That is, we see a tendency but cannot differentiate between linear and circular isolines of the intensity.

R-**Code 12.4** Generating two realizations of an IPP. (See Figure 12.5.)

```
set.seed(15)
IPP <- rpoint(250, function(x,y) { x + y})
plot(IPP, main=expression(x+y))
IPP <- rpoint(250, function(x,y) { exp(- (x^2 + y^2)/.3)})
plot(IPP, main=expression(exp(- (x^2 + y^2)/.3)))
```

Remark 12.2. It is possible to introduce covariates $z_1(\boldsymbol{s}), \dots, z_p(\boldsymbol{s})$ into the intensity modeling by setting $\lambda(\boldsymbol{s}) = f\big(z_1(\boldsymbol{s}), \dots, z_p(\boldsymbol{s})\big)$ for some suitable function f. For example, $\lambda(\boldsymbol{s}) = \exp\big(\alpha + \beta z(\boldsymbol{s})\big)$, where $z(\boldsymbol{s})$ is the height above sea-level at location $\boldsymbol{s}$. ♡

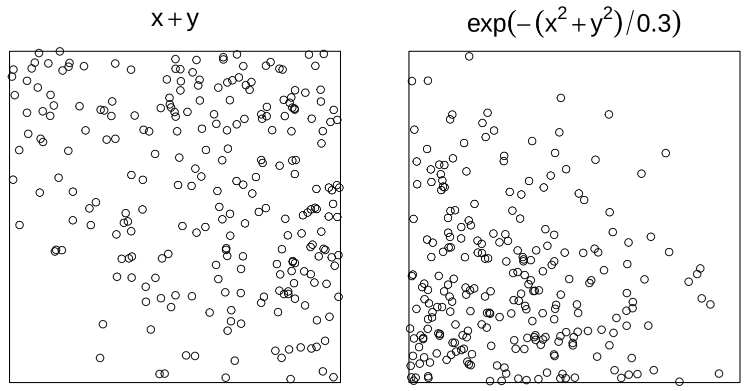


Figure 12.5: Two realizations of an IPP with linear (left) and squared exponential (right) intensity. (See R-Code 12.4.)

12.4.3 Neyman–Scott Process (NSP)

Neymann–Scott processes are clustered point processes and provide a very general model to create clustered processes.

[NSP1] Parent events form an IPP with intensity $\lambda(\boldsymbol{s})$.

[NSP2] Each parent produces a number of offspring iid according to a specified probability mass function.

[NSP3] The positions relative to their parents are iid according to a bivariate pdf.

Figure 12.6 shows two realizations of an NSP based on an HPP. The bivariate pdf to place the offspring is defined via a function passed as an argument to `rNeymanScott()` (package `spatstat`). R-Code 12.5 constructs two NSPs based on an HPP with intensity $\lambda = 9$ (the specific seed, we get 10 parents) with a uniform (left) and a Gaussian kernel (right) for the offspring.

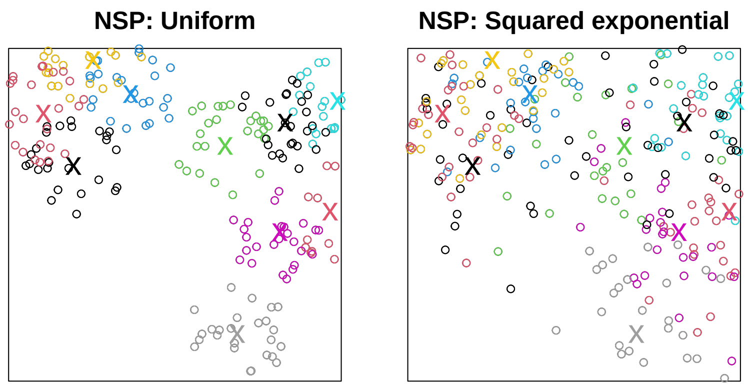


Figure 12.6: Two realizations of an NSP based on an HPP. Each parent produces 25 offspring uniformly in the disc of radius 0.1 (left) and according to a bivariate normal density with $\sigma^2 = 0.1$ (right). The parents and their offsprings are accordingly color coded. For ease of representation, the parent process (indicated with x) was constrained to be in the unit square. (See R-Code 12.5.)

R-Code 12.5 Generating two realizations of NSP. (See Figure 12.6.)

```
set.seed(15)     # Seed for first point pattern
nclust <- function(x0, y0, radius, n) {
  runifdisc(n, radius, centre=c(x0, y0))
}
### Creating 15 Clusters, each with 25 offspring:
NS <- rNeymanScott(9, 0, nclust, radius=0.15, n=25)
### Note that in the plot, not all are displayed due to cropping:
plot(NS, main="NSP: Uniform")
points(NS, col=attr(NS,"parentid"))
points(attr(NS,"parents"), col=1:20, pch="x", cex=2)
set.seed(15)     # Same seed for second point pattern
nclust <- function(x0, y0, radius, n) {
  rpoint(n, function(x,y) { exp(-((x-x0)^2 + (y-y0)^2)/radius)})
}
NS <- rNeymanScott(9, 0, nclust, radius=0.05, n=25)
plot(NS, main="NSP: Squared exponential")
points(NS, col=attr(NS,"parentid"))
points(attr(NS,"parents"), col=1:20, pch="x", cex=2)
```

Neyman–Scott processes are often used in hydrology and weather generation. Each storm parent event represents a storm that is decomposed into individual smaller components represented by the offspring. The offspring would be appended with storm size and intensity.

12.4.4 Simple Inhibition Process (SIP)

In practice, there are often (biological, physical) processes involved in inhibiting the occurrence of events that are close in space, e.g., a manifestation of competition between plants or the territorial behavior of animals. Such point patterns appear regular compared to the aggregated behavior of realizations of HPP. A convenient way of modeling such a simple inhibition point process is to impose a minimum permissible distance, say δ, between two events. Naturally, such a point patterns have an upper bound on the number of events in a domain.

There are two ways of generating events from a SIP: (a) simulate an HPP and apply a thinning procedure or (b) use a sequential approach as follows.

[SIP1] $\mathbf{S}_1$ is uniformly distributed in $\mathcal{D}$.

[SIP2] Given $\{\mathbf{S}_j = \boldsymbol{s}_j, j \leq i\}$, $\mathbf{S}_{i+1}$ is uniformly distributed on $\mathcal{D} \cap \{\boldsymbol{y} : \|\boldsymbol{y} - \boldsymbol{s}_j\| > \delta, j \leq i\}$.

Figure 12.7 shows two realizations of a SIP with $\delta = 0.15$ and $n = 35$ in the left panel and with $\delta = 0.07$ and $n = 100$ in the right panel. With the used seed, placing more realizations is impossible, as indicated by the warning shown in R-Code 12.6.

R-Code **12.6** Generating two realizations of SIP. Note that for $\delta = 0.15$ (as in the left panel), it is impossible to place $n = 38$ events. (See Figure 12.7.)

```
set.seed(15)
SSI <- rSSI(0.15, 32)    # SSI: Simple Sequential Inhibition
plot(SSI, main=expression(delta==0.15))
SSI <- rSSI(0.07, 100)
plot(SSI, main=expression(delta==0.07))
set.seed(15)
SSI <- rSSI(0.15, 38)    # not possible, and thus a `Warning` is issued
## Warning in rSSI(0.15, 38):  Gave up after 1000 attempts with only 36
points placed out of 38
```

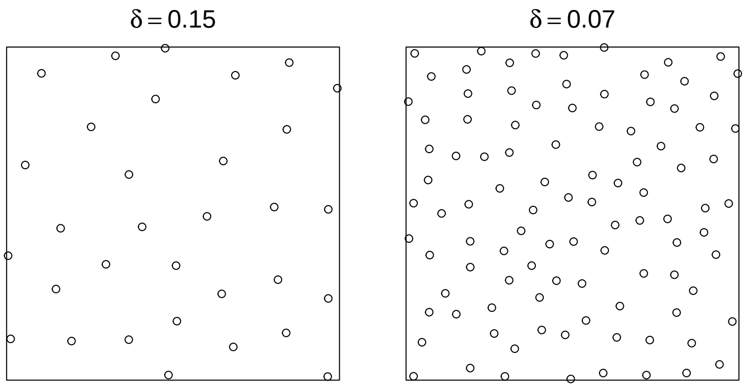


Figure 12.7: Two realizations of a SIP with $\delta = 0.15$, $n = 35$ (left) and $\delta = 0.07$, $n = 100$ (right). (See R-Code 12.6.)

Naturally, SIP can be extended (say to inhomogeneous inhibition processes) by starting from an IPP instead of an HPP. The thinning approach is a so-called Matérn inhibition model 1 and implemented with the function `rMaternI()` in `spatstat`.

12.4.5 Cox Process (CP)

An IPP produces apparent clusters in the region of (relative) high intensity $\lambda(\boldsymbol{s})$. It is natural to consider that the nature of $\lambda(\boldsymbol{s})$ is also stochastic, leading to a "doubly stochastic" approach. The CP provides this framework.

[CP1] $\{\Lambda(\boldsymbol{s}) : \boldsymbol{s} \in \mathbb{R}^2\}$ is a non-negative stochastic process.

[CP2] Conditional on $\{\Lambda(\boldsymbol{s}) = \lambda(\boldsymbol{s})\}$, the events form an IPP with intensity $\lambda(\boldsymbol{s})$.

The CP is stationary (isotropic) if $\Lambda(\boldsymbol{s})$ is stationary (isotropic). First- and second-order properties of a CP are obtained from those of the IPP by taking expectations with respect to $\Lambda(\boldsymbol{s})$. Thus, if $\Lambda(\boldsymbol{s})$ is stationary, $\lambda = \mathrm{E}\big(\Lambda(\boldsymbol{s})\big)$ and $\lambda_2(\boldsymbol{s}, \boldsymbol{y}) = \mathrm{E}\big(\Lambda(\boldsymbol{s})\Lambda(\boldsymbol{y})\big)$. Further, in the case of isotropy, $\lambda_2(t) = \lambda^2 + \mathrm{Cov}(\Lambda(\boldsymbol{s}), \Lambda(\boldsymbol{y}))$ with $t = \|\boldsymbol{s} - \boldsymbol{y}\|$.

Example 12.1. This example has two goals: (i) illustrating a simple way to simulate Cox processes and (ii) assessing the variability between true and estimated intensity functions.

R-Code 12.7 draws realizations from a zero-mean Gaussian process with a Wendland covariance function. We consider a regular grid of 51×51 locations in the unit square which should represent a smooth curve. These realizations are considered as the log-intensity $\log(\lambda(\boldsymbol{s}))$, which is subsequently used to draw an IPP point process with 50 events. By the zero mean assumptions, part of the realized surface is negative, and thus, we exponentiate it to get the intensity $\lambda(\boldsymbol{s})$. Figure 12.8 shows four different (un-scaled) intensity functions $c\lambda(\boldsymbol{s})$, the corresponding IPP (50 events), and their intensity estimate based on `density()` (top and bottom row). As we use unconstrained simulation, the individual intensities represented in the different columns are pretty different. Further, there is a considerable difference between the theoretical and the estimated intensity function, especially in the third column. Increasing the number of points would render the estimated intensities more similar of course.

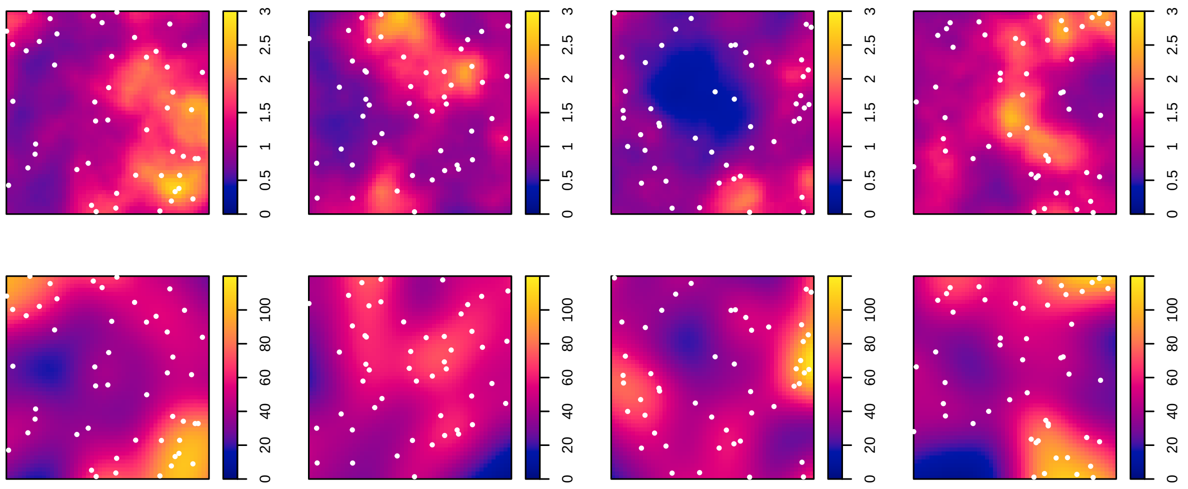


Figure 12.8: Top row: (un-scaled) intensity function $c\lambda(\boldsymbol{s})$ in the unit square. Bottom row: estimated intensities from the obtained point patterns. The white dots are the 50 events of the CP. (See R-Code 12.7.)

The scales of the intensity and estimated intensity panels are different. The intensities are represented as unscaled intensities, denoted as $c\lambda(\mathbf{s})$, as no scaling is performed in the 'as.im()' function. More specifically, we directly work with $\lambda(\mathbf{s}) = \exp(z(\mathbf{s}))$. The densities are designed to integrate to the number of sampled events. For example, for the last column in Figure 12.8, `sum(cox$v)*cox$xstep*cox$ystep` `sum(density(tmp)$v)*cox$xstep*cox$ystep` are 1.343 and 49.63, respectively.

Figure 12.9 shows an intensity function (identical to the second one in Figure 12.8), and eighteen intensity estimates $\widehat{\lambda}_i(\boldsymbol{s})$ from replicated point pattern realizations thereof. The last panel gives the average intensity $\overline{\lambda}(\boldsymbol{s}) = 1/18 \sum_i \widehat{\lambda}_i(\boldsymbol{s})$ based on the intensity estimates displayed.

While the individual estimates are quite variable, the overall mean is very close to the starting intensity $\lambda(\boldsymbol{s})$. Again, 50 points are very few to estimate the intensity. Choosing different parameters for the function *density()* would slightly improve the estimate. ♣

Remark 12.3. The functions *ppm()* and *kppm()* from the package *spatstat* can be used to fit point process data, including inhomogeneous Poisson, Neyman–Scott and Cox processes. ♡

R-Code 12.7 Generate intensity surfaces, draw an IPP thereof and estimate the intensity thereof. (See Figure 12.8.)

```r
library(spam)
set.seed(4)
repli <- 4        # How many replicates
ippn <- 50        # number of points from an IPP (used in function rpoint)
range <- .7       # range to construct the intensity
res <- 51         # Resolution of the Cox surface
ptseq <- seq(0, 1, l=res)                        # sequence
grid <- expand.grid(ptseq, ptseq)                # resulting fine grid
options(spam.nearestdistnnz=c(4926021, 400))     # to avoid a warning in:
distmat <- nearest.dist(grid, delta=range, upper=NULL)  # sparse dist mat
Sigma <- cov.wend1(distmat, theta=c(range, sill=.2))     # Sigma
zz <- rmvnorm.spam(repli, Sigma=Sigma)                   # 4 realizations
### zz[i,] contains a realization of log(Lambda(s))

for(i in 1:repli){
  ### Transform the vector to an image object
  cox <- as.im(cbind(grid, exp(zz[i,])) , win=unit.square())
  tmp <- rpoint(ippn, cox)           # Draw ippn points from intensity lambda

  ### Plotting the original (top) and estimated intensity (bottom):
  plot(cox, main="", ribsep=.07, ribwid=.07, zlim=c(0,3))  # theoretical
  points(tmp$x, tmp$y, col="white", pch=20, cex=.8)        # adding locations
  plot(density(tmp), main="", ribsep=.07, ribwid=.07, zlim=c(0,120)) # empiri
  points(tmp$x, tmp$y, col="white", pch=20, cex=.8)        # adding locations

}
```

R-Code 12.8 Estimating the intensity of a (Cox) process based on several realizations. (See Figure 12.9.)

```
### We (arbitrarily choose the second intensity of Figure 12.8 and plot it:
cox <- as.im(cbind(grid, exp(zz[2,])) , win=unit.square())
plot(cox, main="", ribsep=.07, ribwid=.07, zlim=c(0,3))
total <- 0               # initialization, will contain the average intensity

R <- 18                  # number of intensity functions calculated
set.seed(15)
for(i in 1:R){           # for each realization
  tmp <- rpoint(ippn, cox) # draw 50 points
  dens <- density(tmp)     # estimate the density (here intensity)

  plot(dens, main="", zlim=c(0,127), ribsep=.07, ribwid=.07)
  points(tmp$x, tmp$y, col="white", pch=20, cex=.8) # add locations
  total <- total+dens$v    # summing up the intensities
}
dens$v <- total/R         # Calculate the average and plot it:
plot(dens, main="", zlim=c(0,127), ribsep=.07, ribwid=.07)
```

12.5 Bibliographic Remarks

The exposition is based on Diggle (2003). An excellent treaty is Illian *et al.* (2008). We omit thorough lists of references and refer to these two books or, Diggle (2013), Chapter 8 of Cressie (1993), and Chapter 3 of Schabenberger and Gotway (2005).

The *spatstat* package family has excellent documentation; the summary manual `https://spatstat.org/resources/spatstatManual.pdf` is particularly helpful. See also the accessible online resource `https://www.apps.stat.vt.edu/leman/VTCourses/BaddeleyPointProcesses.pdf`.

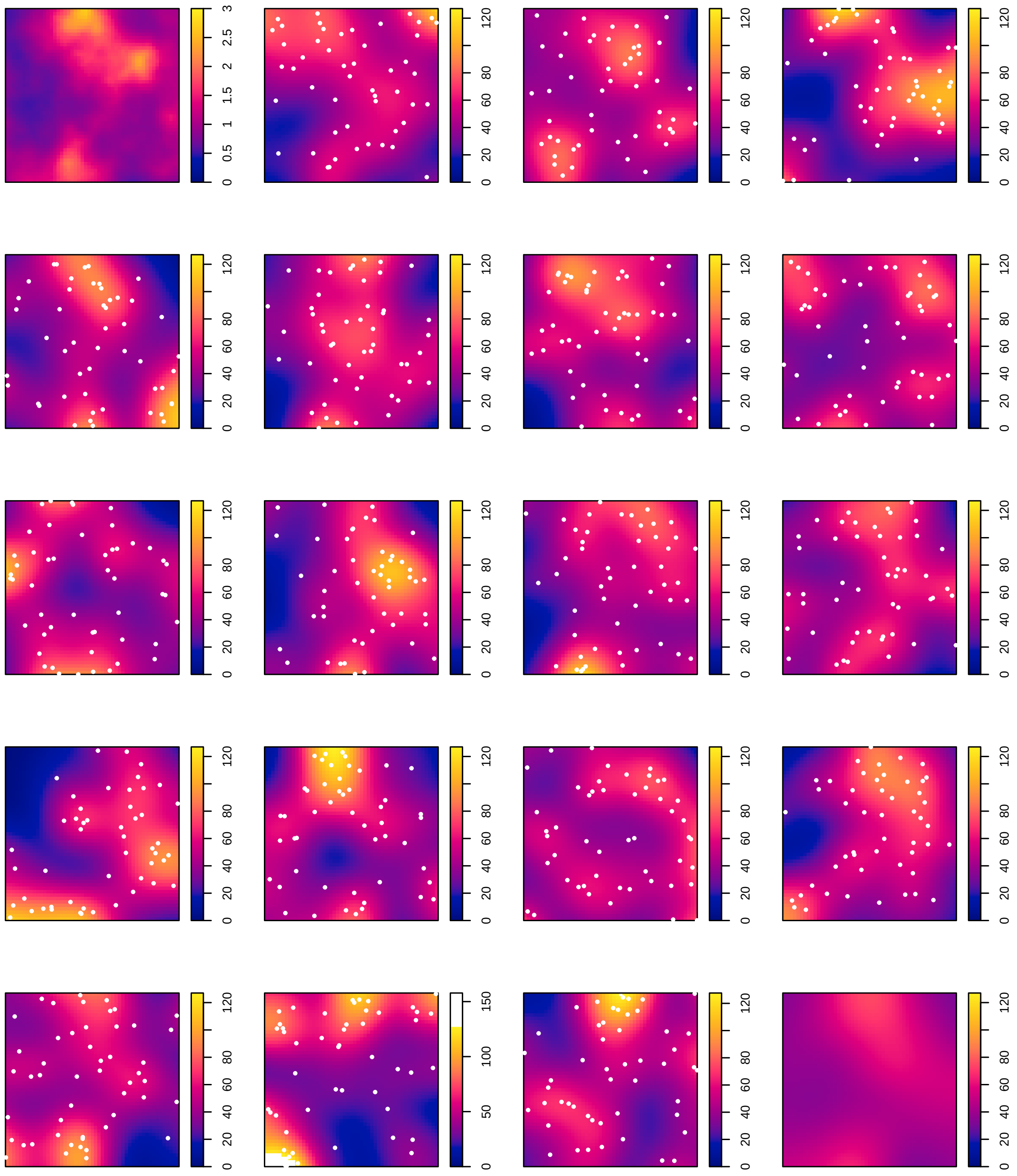


Figure 12.9: Top left panel represents the (un-scaled) intensity function $c\lambda(\boldsymbol{s})$ in the unit square. Followed by 18 intensity estimates from replicates point patterns ($n = 50$) of the intensity function. The lower right panel gives the mean intensity function based on the 18 shown replicates. (Note that one intensity estimate has a different color scale.) (See R-Code 12.8.)

Postscript

The following R script www.math.uzh.ch/furrer/download/sta330/chapterPS.R gives a list of the required packages in each chapter.

The list below gives the attached and loaded packages that were used to compile this document. If evaluating the code results in errors they might be due to outdated R-packages (many of these evolve quickly and regular updates are recommended).

R-Code 12.9: Session info of this document.

```
print(sessionInfo(), locale=FALSE)
## R Under development (unstable) (2023-01-31 r83741)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.2 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/R-devel/lib/R/lib/libRblas.so
## LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3;  LAPACK version 3.10.0
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] spatstat_3.0-3         spatstat.linnet_3.0-4  spatstat.model_3.1-2
##  [4] rpart_4.1.19           spatstat.explore_3.0-6 nlme_3.1-161
##  [7] spatstat.random_3.1-3  spatstat.geom_3.0-6    spatstat.data_3.0-0
## [10] spacetime_1.2-8        splancs_2.01-43        gstat_2.1-0
## [13] geoR_1.9-2             microbenchmark_1.4.9   rgl_1.0.1
## [16] INLA_22.12.16          foreach_1.5.2          R2OpenBUGS_3.2-3.2.1
## [19] CARBayes_5.3           Rcpp_1.0.10            arm_1.13-1
## [22] lme4_1.1-31            MASS_7.3-58.2          BayesFactor_0.9.12-4.4
## [25] coda_0.19-4            LearnBayes_2.15.1      TeachingDemos_2.12
## [28] spatialreg_1.2-6       Matrix_1.5-3           spdep_1.2-7
## [31] spData_2.2.1           sp_1.6-0               sf_1.0-9
## [34] maps_3.4.1             ellipse_0.4.3          RColorBrewer_1.1-3
## [37] mvtnorm_1.1-3          fields_14.1            viridis_0.6.2
## [40] viridisLite_0.4.1      spam_2.9-1             knitr_1.42
##
## loaded via a namespace (and not attached):
##  [1] DBI_1.1.3             deldir_1.0-6          pbapply_1.7-0
##  [4] gridExtra_2.3         tcltk_4.3.0           s2_1.1.2
##  [7] rlang_1.0.6           magrittr_2.0.3        e1071_1.7-12
## [10] compiler_4.3.0        mgcv_1.8-41           vctrs_0.5.2
## [13] quantreg_5.94         stringr_1.5.0         pkgconfig_2.0.3
## [16] wk_0.7.1              fastmap_1.1.0         mcmc_0.9-7
## [19] utf8_1.2.3            nloptr_2.0.3          MatrixModels_0.5-1
## [22] xfun_0.37             jsonlite_1.8.2        goftest_1.2-3
## [25] CARBayesdata_3.0      highr_0.10            reshape_0.8.9
## [28] spatstat.utils_3.0-1  R6_2.5.1              stringi_1.7.12
## [31] GGally_2.1.2          boot_1.3-28.1         iterators_1.0.14
## [34] tensor_1.5            zoo_1.8-11            base64enc_0.1-3
## [37] FNN_1.1.3.1           splines_4.3.0         tidyselect_1.2.0
## [40] abind_1.4-5           codetools_0.2-18      intervals_0.15.2
## [43] lattice_0.20-45       tibble_3.1.8          plyr_1.8.8
## [46] evaluate_0.16         survival_3.5-0        polyclip_1.10-4
## [49] units_0.8-1           proxy_0.4-27          xts_0.12.2
## [52] pillar_1.8.1          KernSmooth_2.23-20    generics_0.1.3
## [55] truncnorm_1.0-8       ggplot2_3.4.0         munsell_0.5.0
## [58] scales_1.2.1          minqa_1.2.5           class_7.3-21
## [61] glue_1.6.2            tools_4.3.0           SparseM_1.81
## [64] dotCall64_1.0-2       grid_4.3.0            MCMCpack_1.6-3
## [67] crosstalk_1.2.0       colorspace_2.1-0      cli_3.6.0
## [70] spatstat.sparse_3.0-0 fansi_1.0.4           expm_0.999-7
## [73] dplyr_1.1.0           gtable_0.3.1          digest_0.6.31
## [76] classInt_0.4-8        htmlwidgets_1.6.1     htmltools_0.5.4
## [79] lifecycle_1.0.3       leaflet_2.1.1
```

Glossary

Throughout the document we tried to be consistent with standard mathematical notation. We write random variables as uppercase letters (X, Y, ...), realizations as lower case letters (x, y, ...), matrices as bold uppercase letters ($\mathbf{\Sigma}$, $\mathbf{X}$, ...), and vectors as bold italics lowercase letters ($\boldsymbol{x}$, $\boldsymbol{\beta}$, ...). (The only slight confusion arises with random vectors and matrices.)

The following glossary contains a non-exhaustive list of the most important notation. Standard operators or products are not repeatedly explained.

$:=$	Define the left hand side by the expression on the other side.
$\clubsuit$, $\diamondsuit$	End of example, end of definition
$\int$, $\sum$, $\prod$	Integration, summation and product symbol. If there is no ambiguity, we omit the domain in inline formulas.
$\cup$, $\cap$	Union, intersection of sets or events.
$\widehat{\theta}$	Estimator or estimate of the parameter θ.
$\bar{x}$	Arithmetic mean of the sample: $\sum_{i=1}^{n} x_i/n$.
$\lvert x \rvert$	Absolute value of the scalar x.
$\lVert \boldsymbol{x} \rVert$	Norm of the vector $\boldsymbol{x}$.
$\mathbf{X}^\top$	Transpose of an matrix $\mathbf{X}$.
$x_{(i)}$	Order statistics of the sample $\{x_i\}$.
$\mathbf{0}$, $\mathbf{1}$	Vector or matrix with components 0 respectively 1.
$\mathrm{Cov}(X, Y)$	Covariance between two random variables X and Y.
$\mathrm{Corr}(X, Y)$	Correlation between two random variables X and Y.
$\frac{d}{dx}$, $'$, $\frac{\partial}{\partial x}$	Derivative and partial derivative with respect to x.
$\mathrm{diag}(\mathbf{A})$	Diagonal entries of an $(n \times n)$-matrix $\mathbf{A}$.
ε, ε_i	Random variable or process, usually measurement error.
$\mathrm{E}(X)$	Expectation of the random variable X.
e, $\exp(\cdot)$	Transcendental number $\mathrm{e} = 2.71828\,18284$, the exponential function.
$\mathbf{I}_n = \mathbf{I}$	Identity matrix, $\mathbf{I} = (\delta_{ij})$.
$I_{\{A\}}$	Indicator function, talking the value one if A is true and zero otherwise.
$\lim$	Limit.
$\log(\cdot)$	Logarithmic function to the base e.

$\max\{A\}$, $\min\{A\}$	Maximum, minimum of the set A.
$\mathbb{N}$, $\mathbb{N}^d$	Space of natural numbers, of d-vectors with natural elements.
$\mathcal{N}(\mu,\sigma^2)$	Normal (Gaussian) distribution with mean μ and variance σ^2.
$\mathcal{N}_p(\boldsymbol{\mu},\boldsymbol{\Sigma})$	Normal p dimensional distribution with mean vector $\boldsymbol{\mu}$ and variance matrix $\boldsymbol{\Sigma}$.
$\varphi(x)$	Gaussian probability densitiy function $\varphi(x) = (2\pi)^{-1/2}\exp(-x^2/2)$.
$\Phi(x)$	Gaussian cumulative distribution function $\Phi(x) = \int_{-\infty}^{x}\varphi(z)\,\mathsf{d}z$.
π	Transzendental number $\pi = 3.14159\,26535$.
$\mathrm{P}(A)$	Probability of the event A.
$\mathbb{R}$, $\mathbb{R}^n$, $\mathbb{R}^{n\times m}$	Space of real numbers, real n-vectors and real $(n \times m)$-matrices.
$\mathrm{rank}(\mathbf{A})$	The rank of a matrix $\mathbf{A}$ is defined as the number of linearly independent rows (or columns) of $\mathbf{A}$.
$\mathrm{tr}(\mathbf{A})$	Trace of an matrix $\mathbf{A}$ defined by the sum of its diagonal elements.
$\mathrm{Var}(X)$	Variance of the random variable X.
$\mathbb{Z}$, $\mathbb{Z}^d$	Space of integers, of d-vectors with integer elements.

The following table contains the abbreviations of the statistical distributions (dof denotes degrees of freedom).

$\mathcal{N}(\mu,\sigma^2)$	Gaussian or normal random variable with parameters μ and σ^2.
$\mathcal{N}(0,1)$, z_p	Standard standard normal random variable, p-quantiles thereof.
$\mathcal{N}_p(\boldsymbol{\mu},\boldsymbol{\Sigma}^2)$	Multivariate Gaussian or normal random vector (of dimension p) with parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^2$.

The following table contains the abbreviations of the statistical methods, properties and quality measures.

EDA	Exploratory data analysis.
dof	Degrees of freedom.
MAD	Median absolute deviation.
MAE	Mean absolute error.
ML	Maximum likelihood (ML estimator or ML estimation).
MM	Method of moments.
MSE	Mean squared error.
OLS, LS	Ordinary least squares.
RMSE	Root mean squared error.
WLS	Weighted least squares.

Bibliography

Abourfirassi, M. and Marino, M. A. (1984). Cokriging of Aquifer Transmissivities From Field Measurements of Transmissivity and Specific Capacity. *Journal of the International Association for Mathematical Geology*, **16**, 19–35. 207

Abramowitz, M. and Stegun, I. A., editors (1970). *Handbook of Mathematical Functions*. Dover. 156, 179

Adler, R. J. (1981). *The Geometry of Random Fields*. John Wiley & Sons Ltd. 155

Ahrens, J. H. and Dieter, U. (1972). Computer methods for sampling from the exponential and normal distributions. *Communications of the ACM*, **15**, 873–882. 114

Armstrong, M. (1982). A Cross-Classification of the Internal Reports on Geostatistics of the Centre de Géostatistique et de Morphologie Mathématique, Fontainebleau, France. *Journal of the International Association for Mathematical Geology*, **14**, 509–537. 206

Armstrong, M. (1984). Problems With Universal Kriging. *Journal of the International Association for Mathematical Geology*, **16**, 101–108. 189

Armstrong, M. and Delfiner, P. (1980). Towards a more robust variogram: A case study on coal. Internal Note N-671. 209

Armstrong, M. and Diamond, P. (1984). Testing Variograms for Positive-definiteness. *Journal of the International Association for Mathematical Geology*, **16**, 407–421. 178

Armstrong, M. and Matheron, G. (1986a). Disjunctive kriging revisited. I. *Mathematical Geology*, **18**, 711–728. 195

Armstrong, M. and Matheron, G. (1986b). Disjunctive kriging revisited. II. *Mathematical Geology*, **18**, 729–742. 195

Bachoc, F., Betancourt, J., Furrer, R., and Klein, T. (2020). Asymptotic properties of the maximum likelihood and cross validation estimators for transformed Gaussian processes. *Electronic Journal of Statistics*, **14**, 1962–2008. 162

Banerjee, S., Carlin, B. P., and Gelfand, A. E. (2003). *Hierarchical Modeling and Analysis for Spatial Data*. Chapman & Hall/CRC, London. 63, 207

Benigni, M. and Furrer, R. (2008). Periodic Spatio-Temporal Improvised Explosive Device Attack Pattern Analysis. Technical Report MCS-08-04, Colorado School of Mines, Golden, USA. 9

Bernardo, J. M. and Smith, A. F. M. (1994). *Bayesian Theory*. John Wiley & Sons Inc., Chichester. 112

Besag, J. (1974). Spatial Interaction and the Statistical Analysis of Lattice Systems (with discussion). *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **36**, 192–225. 55, 63

Besag, J., York, J., and Mollié, A. (1991). Bayesian image restoration, with two applications in spatial statistics (with discussion). *Annals of the Institute of Statistical Mathematics*, **43**, 1–59. 141

Bevilacqua, M., Caamaño-Carrillo, C., and Porcu, E. (2022). Unifying compactly supported and Matérn covariance functions in spatial statistics. *Journal of Multivariate Analysis*, **189**, 104949. 160

Bevilacqua, M., Faouzi, T., Furrer, R., and Porcu, E. (2019). Estimation and prediction using generalized wendland covariance functions under fixed domain asymptotics. *The Annals of Statistics*, **47**, 828–856. 160, 162

Bivand, R. S., Pebesma, E. J., and Gómez-Rubio, V. (2008). *Applied Spatial Data Analysis with R (Use R)*. Springer Verlag. 63, 207

Bivand, R. S., Pebesma, E. J., and Gómez-Rubio, V. (2013). *Applied Spatial Data Analysis with R (Use R)*. Springer Verlag, second edition. 65, 68, 91, 144

Bochner, S. (1933). Monotone Funktionen, Stieltjessche Integrale und harmonische Analyse. *Mathematische Annalen*, **108**, 377–410. 202, 208

Bogaert, P. (1996). Comparison of kriging techniques in a space-time context. *Mathematical Geology*, **28**, 73–86. 202

Bolker, B. and Skaug, H. (2012). *R2admb: ADMB to R Interface Functions*. R package version 0.7.5.3. 150

Borgman, L. E. and Chao, L. (1994). Estimation of a multidimensional covariance function in case of anisotropy. *Mathematical Geology*, **26**, 161–179. 155

Bourgault, G. and Marcotte, D. (1991). Multivariable Variogram and Its Application to the Linear Model of Coregionalization. *Mathematical Geology*, **23**, 899–928. 198

Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (2008). *Time Series Analysis, Forecasting, and Control*. Wiley, fourth edition. 51

Brezger, A., Kneib, T., and Lang, S. (2005). *BayesX*: Analyzing Bayesian Structural Additive Regression Models. *Journal of Statistical Software*, **14**, 1–22. 150

Brockwell, P. J. and Davis, R. A. (1991). *Time Series: Theory and Methods*. Springer-Verlag, second edition. 51

Brockwell, P. J. and Davis, R. A. (2010). *Introduction to time series and forecasting*. Springer texts in statistics. Springer, second edition. 51

Brooker, P. I. (1986). A Parametric Study of Robustness of Kriging Variance As a Function of Range and Relative Nugget Effect for a Spherical Semivariogram. *Mathematical Geology*, **18**, 477–488. 209

Burgess, T. M. and Webster, R. (1980a). Optimal interpolation and isarithmic mapping of soil properties. I. The semivariogram and punctual kriging. *Journal of Soil Science*, **31**, 315–331. 207

Burgess, T. M. and Webster, R. (1980b). Optimal interpolation and isarithmic mapping of soil properties. II. Block kriging. *Journal of Soil Science*, **31**, 333–341. 207

Burgess, T. M., Webster, R., and McBratney, A. B. (1981). Optimal interpolation and isarithmic mapping of soil properties. IV. Sampling strategy. *Journal of Soil Science*, **32**, 643–659. 207

Carr, J. R. and McCallister, P. G. (1985). An Application of Cokriging for Estimation of Tripartite Earthquake Response Spectra. *Journal of the International Association for Mathematical Geology*, **17**, 527–545. 207

Carroll, S. S. and Cressie, N. A. C. (1996). A Comparison of Geostatistical Methodologies Used to Estimate Snow Water Equivalent. *Water Resources Bulletin*, **32**, 267–278. 209

Carroll, S. S. and Cressie, N. A. C. (1997). Spatial modelling of snow water equivalent using covariances estimated from spatial and geomorphic attributes. *Journal of Hydrology*, **190**, 42–59. 209

Cerioli, A. and Riani, M. (1999). The ordering of spatial data and the detection of multiple outliers. *Journal of Computational and Graphical Statistics*, **8**, 239–258. 208

Cesare, L. D., Myers, D. E., and Posa, D. (2001). Estimating and modeling space-time correlation structures. *Statistics & Probability Letters*, **51**, 9–14. 201, 202

Cherry, S., Banfield, J., and Quimby, W. F. (1996). An Evaluation of a Non-parametric Method of Estimating Semi-variograms of Isotropic Spatial Processes. *Journal of Applied Statistics*, **23**, 435–449. 210

Chilès, J.-P. and Delfiner, P. (1999). *Geostatistics: Modeling Spatial Uncertainty*. John Wiley & Sons Inc. 207

Christakos, G. (1984). On the Problem of Permissible Covariance and Variogram Models. *Water Resources Research*, **20**, 251–265. 180

Christakos, G. (2000). *Modern Spatiotemporal Geostatistics*. Oxford University Press. 195

Christensen, O. F. and Ribeiro, P. J. (2002). `geoRglm` – A Package for Generalised Linear Spatial Models. R *News*, **2**, 26–28. 150

Chu, E. and George, A. (1999). *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms.* CRC Press. 51

Cressie, N. A. C. (1986). Kriging Nonstationary Data. *Journal of the American Statistical Association*, **81**, 625–634. 189, 195

Cressie, N. A. C. (1988). Variogram. In Kotz, S., Johnson, N. L., and Read, C. B., editors, *Encyclopedia of Statistical Sciences. Vol. 9.* John Wiley & Sons Inc. 206

Cressie, N. A. C. (1989a). Ergodicity for Time Series and Spatial Processes. *Journal of Statistical Computation and Simulation*, **32**, 61–63. 155

Cressie, N. A. C. (1989b). Geostatistics. *The American Statistician*, **43**, 197–202. 195

Cressie, N. A. C. (1990a). Reply to Comments on "Geostatistics" by Cressie, N. A. C. *The American Statistician*, **44**, 256–258. 195

Cressie, N. A. C. (1990b). The origins of kriging. *Mathematical Geology*, **22**, 239–252. 186

Cressie, N. A. C. (1993). *Statistics for Spatial Data.* Wiley, revised edition. 54, 63, 178, 182, 207, 208, 209, 210, 225

Cressie, N. A. C. and Grondona, M. O. (1992). A comparison of variogram estimation with covariogram estimation. In *The Art of Statistical Science*, 191–208. Wiley. 183

Cressie, N. A. C. and Hawkins, D. M. (1980). Robust estimation of the variogram. I. *Journal of the International Association for Mathematical Geology*, **12**, 115–125. 181

Cressie, N. A. C. and Huang, H.-C. (1999). Classes of nonseparable, spatio-temporal stationary covariance functions. *Journal of the American Statistical Association*, **94**, 1330–1340. 202

Cressie, N. A. C. and Huang, H.-C. (2001). Corrigenda: "Classes of nonseparable, spatio-temporal stationary covariance functions" [Journal of the American Statistical Association (1999), **94**, 1330–1340]. *Journal of the American Statistical Association*, **96**, 784. 202

Cressie, N. A. C. and Wikle, C. K. (1998). The variance-based cross-variogram: you can add apples and oranges. *Mathematical Geology*, **30**, 789–799. 198

Cressie, N. A. C. and Zimmerman, D. L. (1992). On the stability of the geostatistical method. *Mathematical Geology*, **24**, 45–59. 209

Dalal, S. R., Fowlkes, E. B., and Hoadley, B. (1989). Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure. *Journal of the American Statistical Association*, **84**, 945–957. 108

Davis, B. M. and Borgman, L. E. (1979). Some Exact Sampling Distributions for Variogram Estimators. *Journal of the International Association for Mathematical Geology*, **11**, 643–654. 181

Davis, B. M. and Borgman, L. E. (1982). A Note on the Asymptotic Distribution of the Sample Variogram. *Journal of the International Association for Mathematical Geology*, **14**, 189–193. 181

Davis, B. M. and Greenes, K. A. (1983). Estimation Using Spatially Distributed Multivariate Data: An Example With Coal Quality. *Journal of the International Association for Mathematical Geology*, **15**, 291–304. 207

Delfiner, P. (1976). Linear estimation of nonstationary spatial phenomena. In Guarascio, M., David, M., and Huijbregts, C. J., editors, *Advanced Geostatistics in the Mining Industry*, 49–68. D. Reidel Publishing Co. 195

Deutsch, C. V. and Journel, A. G. (1998). *GSLIB: Geostatistical Software Library and User's Guide*. Oxford University Press, second edition. 207

Diamond, P. and Armstrong, M. (1984). Robustness of variograms and conditioning of kriging matrices. *Journal of the International Association for Mathematical Geology*, **16**, 809–822. 209

Diggle, P. J. (2003). *Statistical analysis of spatial point patterns*. Hodder Arnold, second edition. 218, 225

Diggle, P. J. (2013). *Statistical Analysis of Spatial and Spatio-Temporal Point Patterns*. CRC Press, third edition. 225

Diggle, P. J. and Ribeiro Jr., P. J. (2007). *Model-Based Geostatistics*. Springer. 207

Dowd, P. A. (1984). The Variogram and Kriging: Robust and Resistant Estimators. In Verly, G., David, M., Journel, A. G., and Marechal, A., editors, *Geostatistics for Natural Resources Characterization*, 91–106. D. Reidel Publishing Co. 209

Dubois, G. (1998). Foreword and Introduction. *Journal of Geographic Information and Decision Analysis*, **2**, 1–10. 195

Dubois, G. (2000). *Intégration de Systèmes d'Informations Géographiques (SIG) et de méthodes géostatistiques. Applications à la cartographie de pollutions radioactives dans l'environnement.* Ph.D. Thesis, Université de Lausanne, Lausanne, Switzerland. 195

Dutter, R. (1996). On robust estimation of variograms in geostatistics. In *Robust statistics, data analysis, and computer intensive methods (Schloss Thurnau, 1994)*, 153–171. Springer. 182, 209

Englund, E. J. (1990). A Variance of Geostatisticians. *Mathematical Geology*, **22**, 417–455. 195

Eriksson, M. and Siska, P. P. (2000). Understanding Anisotropy Computations. *Mathematical Geology*, **32**, 683–700. 155

Finley, A. O., Banerjee, S., and Carlin, B. P. (2007). `spBayes`: An R Package for Univariate and Multivariate Hierarchical Point-Referenced Spatial Models. *Journal of Statistical Software*, **19**, 1–24. 150

Finley, A. O., Banerjee, S., and Gelfand, A. E. (2015). `spBayes` for Large Univariate and Multivariate Point-Referenced Spatio-Temporal Data Models. *Journal of Statistical Software*, **63**, 1–28. 150

Fischer, T. C., Furrer, R., and Stocker, J. (2022). *GeneralizedWendland: Fully Parameterized Generalized Wendland Covariance Function*. R package version 0.5-2. 160

Fournier, D. A., Skaug, H. J., Ancheta, J., Ianelli, J., Magnusson, A., Maunder, M. N., Nielsen, A., and Sibert, J. (2012). AD Model Builder: Using Automatic Differentiation for Statistical Inference of Highly Parameterized Complex Nonlinear Models. *Optimization Methods and Software*, **27**, 233–249. 150

Fuentes, M. and Smith, R. (2002). A new class of nonstationary spatial models. *Journal of the American Statistical Association*. 202

Furrer, R. and Genton, M. G. (2011). Aggregation-cokriging for highly-multivariate spatial data. *Biometrika*, **98**, 615–631. 207

Furrer, R., Genton, M. G., and Nychka, D. (2006). Covariance tapering for interpolation of large spatial datasets. *Journal of Computational and Graphical Statistics*, **15**, 502–523. 162

Furrer, R. and Sain, S. R. (2009). Spatial Model Fitting for Large Datasets with Applications to Climate and Microarray Problems. *Statistics and Computing*, **19**, 113–128. 75

Gamerman, D. and Lopes, H. F. (2006). *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC. 96, 134, 137

Gandin, L. S. (1963). *Objective Analysis of Meteorological Fields*. Leningrad, Gidrometizdat, (In Russian), second edition. 206

Gelfand, A. and Smith, A. (1990). Sampling Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, **85**, 398–409. 113

Gelfand, A. E., Sahu, S., and Carlin, B. (1995). Efficient Parametrization for Normal Linear Mixed Effects Models. *Biometrika*, **82**, 479–488. 147

Gelman, A. and Rubin, D. B. (1992). Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, **7**, 457–511. 137

Geman, S. and Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**, 721–741. 113

Genton, M. G. (1998a). Highly robust variogram estimation. *Mathematical Geology*, **30**, 213–221. 182

Genton, M. G. (1998b). Variogram fitting by generalized least squares using an explicit formula for the covariance structure. *Mathematical Geology*, **30**, 323–345. 181, 186

Genton, M. G. (2000). The correlation structure of Matheron's classical variogram estimator under elliptically contoured distributions. *Mathematical Geology*, **32**, 127–137. 181

George, J. A. (1971). *Computer implementation of the finite element method.* PhD thesis, Stanford University, Stanford, CA, USA. 79

Gerber, F., de Jong, R., Schaepman, M. E., Schaepman-Strub, G., and Furrer, R. (2018). Predicting missing values in spatio-temporal remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, **56**, 2841–2853. 195

Gerber, F. and Furrer, R. (2015). Pitfalls in the implementation of Bayesian hierarchical modeling of areal count data: An illustration using BYM and Leroux models. *Journal of Statistical Software*, **63**, 1–32. 133, 149

Gerrard, D. J. (1969). Competition quotient: a new measure of the competition affecting individual forest trees. Research Bulletin 20, Agricultural Experiment Station, Michigan State University. 211

Geweke, J. (1992). Evaluating the Accuracy of Sampling-Based Approaches to Calculating Posterior Moments. In Bernardo, J. M., Berger, J., Dawid, A. P., and Smith, J. F. M., editors, *Bayesian Statistics 4*, 169–193. Oxford University Press, Oxford. 138

Gilks, W. R., Richardson, S., and Spiegelhalter, D. J. (1998). *Markov Chain Monte Carlo in Practice*. Chapman & Hall. 113

Gneiting, T. (2001). Nonseparable, Stationary Covariance Functions for Space-Time Data. Technical Report 63, NRCSE. 202

Goldberger, A. S. (1962). Best linear unbiased prediction in the generalized linear regression model. *Journal of the American Statistical Association*, **57**, 369–375. 189

Goovaerts, P. (1997). *Geostatistics for Natural Resources Evaluation*. Oxford University Press. 182, 207

Goovaerts, P. (1998). Ordinary Cokriging Revisited. *Mathematical Geology*, **30**, 21–42. 198

Grubbs, F. E. (1969). Procedures for Detecting Outlying Observations in Samples. *Technometrics*, **11**, 1–21. 208

Hampel, F. R. (1971). A general qualitative definition of robustness. *Annals of Mathematical Statistics*, **42**, 1887–1896. 182

Harvey, A. C. and Durbin, J. (1986). The effects of seat belt legislation on British road casualties: A case study in structural time series modelling. *Journal of the Royal Statistical Society: series B (Statistical Methodology)*, **149**, 187–227. 48

Hawkins, D. M. and Cressie, N. A. C. (1984). Robust kriging — a proposal. *Journal of the International Association for Mathematical Geology*, **16**, 3–18. 195

Heaton, M. J., Datta, A., Finley, A. O., Furrer, R., Guinness, J., Guhaniyogi, R., Gerber, F., Gramacy, R. B., Hammerling, D., Katzfuss, M., Lindgren, F., Nychka, D. W., Sun, F., and Zammit-Mangion, A. (2019). A case study competition among methods for analyzing large spatial data. *Journal of Agricultural, Biological and Environmental Statistics*, **24**, 398–425. 195

Heidelberger, P. and Welch, P. D. (1981). A spectral method for confidence interval generation and run length control in simulations. *Communications ACM*, **24**, 233–245. 138

Held, L., Natario, I., Fenton, S., Rue, H., and Becker, N. (2005). Towards joint disease mapping. *Statistical Methods in Medical Research*, **14**, 61–82. 4, 141

Held, L. and Sabanés Bové, D. (2014). *Applied Statistical Inference*. Springer. 112

Hristopulos, D. T. and Christakos, G. (2001). Practical Calculation of Non-Gaussian Multivariate Moments in Spatiotemporal Bayesian Maximum Entropy Analysis. *Mathematical Geology*, **33**, 543–568. 195

Huang, H.-C. and Cressie, N. A. C. (1996). Spatio-temporal prediction of snow water equivalent using the Kalman filter. *Computational Statistics & Data Analysis*, **22**, 159–175. 202

iCasualties (2009). OIF-Deaths by IED. Retrieved July 17 2009 from http://icasualties.org. 8

Illian, J., Penttinen, P., Stoyan, H., and Stoyan, D. (2008). *Statistical Analysis and Modelling of Spatial Point Patterns*. Wiley. 225

Jeffreys, H. (1983). *Theory of probability*. The Clarendon Press Oxford University Press, third edition. 105

Jones, T. A. and Ma, Y. Z. (2001). Teacher's Aide: Geologic Characteristics of Hole-Effect Variograms Calculated from Lithology-Indicator Variables. *Mathematical Geology*, **33**, 615–629. 180

Journel, A. G. and Huijbregts, C. J. (1978). *Mining Geostatistics*. Academic Press. 155, 180, 182, 198, 207

Kaufman, C. G., Schervish, M. J., and Nychka, D. W. (2008). Covariance tapering for likelihood-based estimation in large spatial data sets. *Journal of the American Statistical Association*, **103**, 1545–1555. 162

Kitanidis, P. K. and Lane, R. W. (1985). Maximum Likelihood Parameter Estimation of Hydrologic Spatial Processes by the Gauss-Newton Method. *Journal of Hydrology*, **79**, 53–71. 209

Kneib, T., Heinzl, F., Brezger, A., and Bové, D. S. (2011). *BayesX: R Utilities Accompanying the Software Package* BayesX. R package version 0.2-5. 150

Knorr-Held, L. and Best, N. G. (2001). A Shared Component Model for Detecting Joint and Selective Clustering of Two Diseases. *Journal of the Royal Statistical Society A*, **164**, 73–85. 142

Knorr-Held, L. and Raßer, G. (2000). Bayesian Detection of Clusters and Discontinuities in Disease Maps. *Biometrics*, **56**, 13–21. 4, 141

Knorr-Held, L. and Rue, H. (2002). On Block Updating in Markov Random. *Scandinavian Journal of Statistics*, **29**, 597–614. 131

Kolmogoroff, A. (1941). Interpolation und Extrapolation von stationären zufälligen Folgen. *Bull. Acad. Sci. URSS Sér. Math. [Izvestia Akad. Nauk. SSSR]*, **5**, 3–14. 186

Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the Chemical, Metallurgical and Mining Socciety of South Africa*, **52**, 119–139. 186

Künsch, H. R., Papritz, A., and Bassi, F. (1997). Generalized cross-covariances and their estimation. *Mathematical Geology*, **29**, 779–799. 198

Kyriakidis, P. C. and Journel, A. G. (1999). Geostatistical space-time models: a review. *Mathematical Geology*, **31**, 651–684. 200

Lark, R. M. (2000). A comparison of som robust estimators of the variogram for use in soil survey. *European Journal of Soil Science*, **51**, 137–157. 209

LeBauer, D. S., Dietze, M. C., and Bolker, B. M. (2013). Translating Probability Density Functions: From R to BUGS and Back Again. *R Journal*, **5**, 207–209. 147

Lee, D. (2011). A Comparison of Conditional Autoregressive Models Used in Bayesian Disease Mapping. *Spatial and Spatio-Temporal Epidemiology*, **2**, 79–89. 150

Lee, D. (2013). *CARBayes*: An R Package for Bayesian Spatial Modeling with Conditional Autoregressive Priors. *Journal of Statistical Software*, **55**, 1–24. 148

Leroux, B. G., Lei, X., and Breslow, N. (1999). *Estimation of Disease Rates in Small Areas: A New Mixed Model for Spatial Dependence*. IMA Volumes in Mathematics and its Applications. US Government Printing Office. 148, 150

LeSage, J. and Pace, R. K. (2009). *Introduction to Spatial Econometrics*. Chapman and Hall/CRC. 150

Liu, J. W. H. (1985). Modification of the Minimum-Degree Algorithm by Multiple Elimination. *ACM Transactions on Mathematical Software*, **11**, 141–153. 79

Long, A. E. and Myers, D. E. (1997). A new form of the cokriging equations. *Mathematical Geology*, **29**, 685–703. 207

Lunn, D., Jackson, C., Best, N., Thomas, A., and Spiegelhalter, D. (2013). *The* BUGS *Book: A Practical Introduction to Bayesian Analysis*. Texts in Statistical Science. Chapman & Hall/CRC. 144

Ma, Y. Z. and Jones, T. A. (2001). Teacher's Aide: Modeling Hole-Effect Variograms of Lithology-Indicator Variables. *Mathematical Geology*, **33**, 631–648. 180, 186

Mardia, K. V., Kent, J. T., and Bibby, J. M. (1979). *Multivariate Analysis*. Academic Press. 20

Mardia, K. V. and Marshall, R. J. (1984). Maximum likelihood estimation of models for residual covariance in spatial regression. *Biometrika*, **71**, 135–146. 209

Matérn, B. (1960). *Spatial Variation*. Meddelanden Fr*r*an Statens Skogsforskningsinstitut, Band 49, Nr 5. 206

Matérn, B. (1986). *Spatial Variation*. Springer-Verlag, second edition. 206

Matheron, G. (1962). Trait? de G?ostatistique Appliqu?e, Tome I. *M?moires du Bureau de Recherches G?ologique et Mini?res*, **No. 14**, Editions Technip, Paris. 177, 178, 181, 205, 206

Matheron, G. (1963). Trait? de G?ostatistique Appliqu?e, Tome II: Le Krigeage. *Mémoires du Bureau de Recherches Géologique et Minières*, **No. 24**, Editions Technip, Paris. 186, 205

Matheron, G. (1969). Le Krigeage Universel. *Cahiers du Centre de Morphologie Math?matique*, **No. 1**, Fontainebleau, France. 189, 205

Matheron, G. (1971). The theory of regionalized variables and its applications. *Cahiers du Centre de Morphologie Math?matiques*, **No. 5**, Fontainebleau, France. 180, 206

Matheron, G. (1973a). Le Krigeage disjonctif. *Internal Note N-360*. 195

Matheron, G. (1973b). The intrinsic random functions and their applications. *Advances in Applied Probability*, **5**, 439–468. 195, 206

Matheron, G. (1976). A simple substitute for conditional expectation: The disjunctive kriging. In Guarascio, M., David, M., and Huijbregts, C. J., editors, *Advanced Geostatistics in the Mining Industry*, 221–236. D. Reidel Publishing Co. 195

Møller, J., editor (2003). *Spatial Statistics and Computational Methods*. Springer Verlag, Lecture Notes in Statistics, 173. 207

Mollié, A. (1996). Bayesian mapping of disease. In Gilks, W. R., Richardson, S., and Spiegelhalter, D. J., editors, *Markov Chain Monte Carlo in Practice*, 359–379. Chapman & Hall, London. 141

Muñoz-Garcia, J., Moreno-Rebollo, J. L., and Pascual-Acosta, A. (1990). Outliers: A Formal Approach. *International Statistical Review*, **58**, 215–226. 208

Myers, D. E. (1982). Matrix formulation of co-kriging. *Journal of the International Association for Mathematical Geology*, **14**, 249–257. 198

Myers, D. E. (1983). Estimation of linear combinations and co-kriging. *Journal of the International Association for Mathematical Geology*, **15**, 633–637. 207

Myers, D. E. (1991). Pseudo-cross variograms, positive-definiteness, and cokriging. *Mathematical Geology*, **23**, 805–816. 198

Myers, D. E. (1992). Kriging, co-kriging, radial basis functions and the role of positive definiteness. *Computers & Mathematics with Applications. An International Journal*, **24**, 139–148. 195

Nychka, D. (2000). Spatial Process Estimates as Smoothers. In Schimek, M. G., editor, *Smoothing and Regression: Approaches, Computation, and Application*, 393–424. John Wiley & Sons Inc. 195

Olea, R. A. (1991). *Geostatistical Glossary and Multilingual Dictionary*. Oxford University Press. 210

Olea, R. A. (1999). *Geostatistics for Engineers and Earth Scientists*. Kluwer Academic Publishers. 207

Pannatier, Y. (1995). *VARIOWIN: Logiciel pour l'analyse spatiale de donn?es en 2D — Etude geologique et geostatistique du gite de phosphates de Taiba (Senegal)*. Ph.D. Thesis, University of Lausanne. 207

Pannatier, Y. (1996). *VARIOWIN: Software for Spatial Data Analysis in 2D*. Springer Verlag. 207

Papritz, A., Künsch, H. R., and Webster, R. (1993). On the pseudo cross-variogram. *Mathematical Geology*, **25**, 1015–1026. 198

Pardo-Igúquiza, E. and Dowd, P. A. (1998). The Second-Order Stationary Universal Kriging Model Revisited. *Mathematical Geology*, **30**, 347–378. 189

Park, J. H. (2014). *CRAN Task View: Bayesian Inference*. Version 2014-12-18. 150

Plummer, M. (2012). `rjags`*: Bayesian Graphical Models Using MCMC*. R package version 3-7. 150

Raftery, A. E. and Lewis, S. M. (1992). One long run with diagnostics: Implementation strategies for Markov chain Monte Carlo. *Statistical Science*, **7**, 493–497. 139

Ripley, B. D. (1976). The second-order analysis of stationary point processes. *Journal of Applied Probability*, **13**, 255–266. 217

Ripley, B. D. (1981). *Spatial Statistics*. John Wiley & Sons Inc. 207

Robert, C. P. and Casella, G. (1999). *Monte Carlo Statistical Methods*. Springer. 113

Roberts, G. and Smith, A. (1994). Simple conditions for the convergence of the gibbs sampler and metropolis-hastings algorithms. *Stochastic Processes and their Applications*, **49**, 207–216. 116, 134

Rodríguez-Iturbe, I. and Mjía, J. M. (1974). The Design of Rainfall Networks in Time and Space. *Water Resources Research*, **10**, 713–728. 202

Rouhani, S. and Myers, D. E. (1990). Problems in Space-time Kriging of Geohydrological Data. *Mathematical Geology*, **22**, 611–623. 202

Rousseeuw, P. J. and Croux, C. (1992). Explicit scale estimators with high breakdown point. In *L_1-statistical analysis and related methods (Neuchâtel, 1992)*, 77–92. North-Holland. 182

Rousseeuw, P. J. and Croux, C. (1993). Alternatives to the median absolute deviation. *Journal of the American Statistical Association*, **88**, 1273–1283. 182

Rue, H. and Held, L. (2005). *Gaussian Markov Random Fields: Theory and Applications*, volume 104 of *Monographs on Statistics and Applied Probability*. Chapman & Hall, London. 63, 123, 126, 131, 133, 147, 207

Rue, H., Martino, S., and Chopin, N. (2009). Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **71**, 319–392. 96, 145

Schabenberger, O. and Gotway, C. A. (2005). *Statistical Methods for Spatial Data Analysis*. Chapman & Hall/CRC. 63, 207, 210, 225

Shapiro, A. and Botha, J. D. (1991). Variogram Fitting With a General Class of Conditionally Nonnegative Definite Functions. *Computational Statistics and Data Analysis*, **11**, 87–96. 210

Shumway, R. H. and Stoffer, D. S. (2010). *Time Series Analysis and Its Applications. With R Examples.* Springer, third edition. 27, 51

Srivastava, R. M. (1986). Philip and Watson — quo vadunt? *Mathematical Geology*, **18**, 141–146. 187

Stein, M. L. (1987). Minimum norm quadratic estimation of spatial variograms. *Journal of the American Statistical Association*, **82**, 765–772. 210

Stein, M. L. (1988). Asymptotically efficient prediction of a random field with a misspecified covariance function. *The Annals of Statistics*, **16**, 55–63. 209

Stein, M. L. (1990). Uniform asymptotic optimality of linear predictions of a random field using an incorrect second-order structure. *The Annals of Statistics*, **18**, 850–872. 209

Stein, M. L. (1999). *Interpolation of Spatial Data*. Springer-Verlag. 209

Stein, M. L. and Handcock, M. S. (1989). Some asymptotic properties of kriging when the covariance function is misspecified. *Mathematical Geology*, **21**, 171–190. 209

Stewart, J. (1976). Positive definite functions and generalizations, an historical survey. *Rocky Mountain Journal of Mathematics*, **6**, 409–434. 208

Sturtz, S., Ligges, U., and Gelman, A. (2005). `R2WinBUGS`: A Package for Running `WinBUGS` from R. *Journal of Statistical Software*, **12**, 1–16. 144

Tukey, J. W. (1948). Approximate weights. *Annals of Mathematical Statistics*, **19**, 91–92. 190

Vanden Brook, T. (2009). Casualties caused by IEDs in Afghanistan on the rise. *USA Today*, retrieved April 2 2009 from http://www.usatoday.com. 8

Velasco, F., Verma, S. P., and Guevara, M. (2000). Comparison of the Performance of Fourteen Statistical Tests for Detection of Outlying Values in Geochemical Reference Material Databases. *Mathematical Geology*, **32**, 439–464. 208

Ver Hoef, J. M., Hanks, E. M., and Hooten, M. B. (2018). On the relationship between conditional (CAR) and simultaneous (SAR) autoregressive models. *Spatial Statistics*, **25**, 68–85. 58

Wackernagel, H. (1995). *Multivariate Geostatistics. An Introduction with Applications*. Springer. 207

Wackernagel, H. (2006). *Multivariate Geostatistics*. Springer-Verlag, New York, third edition. 207

Wahba, G. (1990). *Spline Models for Observational Data*. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM. 195

Waller, L. and Carlin, B. (2010). Disease Mapping. In Gelfand, A. E., Diggle, P. J., Fuentes, M., and Guttorp, P., editors, *Handbook of Spatial Statistics*. Taylor & Francis. Chapter 14. 150

Waller, L., Carlin, B. P., Xia, H., and Gelfand, A. E. (1997). Hierarchical Spatio-Temporal Mapping of Disease Rates. *Journal of the American Statistical Association*, **92**, 607–617. 150

Warnes, J. J. and Ripley, B. D. (1987). Problems with likelihood estimation of covariance functions of spatial Gaussian processes. *Biometrika*, **74**, 640–642. 209

Watkins, A. and Al-Boutiahi, F. (1990). On Maximum Likelihood Estimation of Parameters in Incorrectly Specified Models of Covariance for Spatial Data. *Mathematical Geology*, **22**, 151–173. 209

Weber, D. and Englund, E. J. (1992). Evaluation and Comparison of Spatial Interpolators. *Mathematical Geology*, **24**, 381–391. 195

Webster, R. and Burgess, T. M. (1980). Optimal interpolation and isarithmic mapping of soil properties. III. Changing drift and universal kriging. *Journal of Soil Science*, **31**, 505–524. 207

Webster, R. and Oliver, M. (2001). *Geostatistics for Environmental Scientists*. John Wiley & Sons Inc. 207

Wiener, N. (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series. With Engineering Applications*. The Technology Press of the Massachusetts Institute of Technology, Cambridge. 186

Wold, H. O. A. (1938). *A Study in the Analysis of Stationary Time Series*. Almqvist and Wiksell. 186

Yaglom, A. M. (1957). Some Classes of random fields in n-dimensional space, related to stationary stochastic processes. *Theory of Probability and its Applications*, **2**, 273–320. 208

Yakowitz, S. J. and Szidarovszky, F. (1985). A comparison of Kriging with nonparametric regression methods. *Journal of Multivariate Analysis*, **16**, 21–53. 209

Zimmerman, D. L. (1993). Another Look at Anisotropy in Geostatistics. *Mathematical Geology*, **25**, 453–470. 182

Zimmerman, D. L. and Cressie, N. A. C. (1992). Mean squared prediction error in the spatial linear model with estimated covariance parameters. *Annals of the Institute of Statistical Mathematics*, **44**, 27–43. 190

Zimmerman, D. L. and Zimmerman, M. B. (1991). A Comparison of Spatial Semivariogram Estimators and Corresponding Ordinary Kriging Predictors. *Technometrics*, **33**, 77–91. 186

Dataset Index

Only the first or the most prominent appearance in each chapter is listed.

Command Index

Only the first or the most prominent appearance in each chapter is listed.

Package Index

Only the first or the most prominent appearance in each chapter is listed.