Infectious Disease Forecasting: Using Ensemble Models to Improve Predictions

---

Master Thesis in Biostatistics (STA495)

by

Barras Laura
10-408-797

supervised by

Prof. Dr. Leonhard Held
Johannes Bracher

Zurich, April 30, 2018

Abstract

A common view within the statistical field of time series forecasting is that there is a single model at the origin of the data generating process and that the job of the forecaster is to find it (R. Hyndman, 2018). However, many scientists agree that reality is probably too complex to be described by a single model and growing evidence suggest that the use of ensemble models can improve predictions.
The purpose of this thesis is to implement and assess the use of ensemble models within the HHH4 framework designed for the analysis of spatially and temporally aggregated infectious disease surveillance data. We investigate model averaging (MA) according to AIC and BIC criteria as described by Claeskens and Hjort (2008) and a form of *stacking* inspired by Yao et al. (2017), who developed a procedure for the combination of predictive distributions - as compared to point estimation.
We present the methods with an univariate example, discuss the stability of the hereby introduced *stacking* procedure and assess the impact of changes to the hyperparameters. We also study the reproducibility of our results and extend our analysis to multivariate models.
We find that ensemble models perform relatively well in both univariate and multivariate time series analysis. Although they do not beat the best individual model, they never perform the worse. We find that MA is more straightforward to implement than *stacking* because weights are calculated directly and are not the result of an optimization. The optimization may not always have a clearly defined output meaning that the whole *stacking* procedure is not perfectly stable. We also observe an important variation in the performance of individual and ensemble models depending on both the year under study and the values of the hyperparameters. Although ensemble models show promising results, more research will be needed to better understand how to control these variations to ultimately improve predictions.

Notation

The notation for this thesis closely follows that of Held et al. (2017). Scalar parameters, such as w_j , are represented by lowercase letters. Vectors and matrices are printed in bold, with matrices also represented by uppercase letters as in $\mathbf{\Sigma}_P$.

Acknowledgment

I would like to thank my thesis supervisor Prof. Leonhard Held for providing a thesis topic aligned with my interests in infectious disease modeling and predictions, and his guidance throughout this project. I am also grateful to Johannes Bracher for his time, advice and help. I have gained much insight through their discussions and explanations. I would like to thank all of the people without who the Master Program in Biostatistics would not be running; all the professors and Dr. Eva Furrer in particular. I would like to thank my fellow students. Thank you Tea Isler for the long library sessions and being my study buddy. Thank you also Eleftherios Papagiannoulis, Xinglu Liu and Sandar Lim. The four of you are such great colleagues and friends and made this whole experience a lot more fun and interesting. Finally, the greatest thanks to Nicolas, Elizabeth and my family for their advice, correction support and encouragement.

Contents

Chapter 1

Introduction

Predictions of infectious diseases incidence and quantification of the uncertainty related to those predictions are essential for policymakers and health care specialists to prepare and control upcoming epidemics. Many national surveillance systems routinely collect surveillance data on notifiable diseases in the form of daily, weekly or monthly counts possibly stratified by geographical area and/or age groups. These univariate or multivariate time series are then analysed to provide probabilistic forecasts for the future.

A well developed statistical framework designed to work with such space-time counts of infectious diseases is the "HHH4" model proposed by Held et al. (2005), Held et al. (2006) and Paul et al. (2008) and made available in the `R` package **surveillance** (Meyer et al., 2017). The model was later extended to incorporate non-linear random effects (Paul and Held, 2011), appropriate seasonal variations parameters (Held and Paul, 2012), power-law decay of spatial interaction (Meyer and Held, 2014), an age-structured social contact matrix (Meyer and Held, 2017), and, most recently, distributed lags (Bracher and Held, 2017). Additionally, Held et al. (2017) provide the analytical derivations for the first and second moments of long-term multivariate forecast.

Although much effort has been invested into improving specific models, the impact of combining different models within the "HHH4" framework has not yet been investigated. So far, the best model is selected based on a selection criterion (e.g, AIC, Log-Score, RPS or DSS score). However, it is known that model selection is often unstable and may ignore model uncertainty which in turns leads to over-confident inferences and unnecessary high variability in the final prediction (Zou and Yang, 2004). A solution consists in combining *many* possible models. Since the seminal paper of Newbold and Granger (1974), substantial and ever growing evidence supports that combining multiple forecasts often improves accuracy (see Clemen 1989 for an annotated bibliography). Variations of model combination (also called "ensemble techniques", "ensemble learning", or "meta-learning") are used in fields such as economics (Geweke and Amisano, 2012), weather forecasting (Raftery et al. 2005,Sloughter et al. 2010) and disease forecasting (Yamana et al., 2016). The organizers of the M3-competition Makridakis and Hibon (2000) conclude that "the accuracy when various methods are being combined outperforms, on average, the individual methods and does very well in comparison to other methods". In the field of machine learning, many popular methods such as bagging, boosting and random

forest are ensemble methods and it is common knowledge that top-performers in machine learning contests such as Kaggle use ensemble methods to win competitions.

In its most efficient form, ensemble methods are able to capture the best aspect of each model which is the reason why they perform better than the base models individually. Base models should therefore be very diverse, ideally from different techniques so that they seize the data from different angles. When they are very similar, model combination rarely performs better than the best single model. As stated by Newbold and Granger (1974) "If [the critics] are saying that combination is not a valid proposition if one of the individual forecasts does not differ significantly from the optimum, we must of course agree". However we share the view of Zou and Yang (2004) who state that combining forecasts from very similar models is also important. Model combination avoids model uncertainty and hence combining has the great potential to reduce the variability that arises in the forced action of selecting a single model. The forecasting accuracy can be improved relative to the use of a selection criterion.

There are many different methods to combine models, potential approaches range from simple averaging to more complex schemes designed to give optimal combination of weights. In the Bayesian framework, model combination is performed by Bayesian model averaging (BMA) or a variation thereof. BMA weights are posterior probabilities. If we assume equal prior probabilities for each model, BMA weights are proportional to BIC and it is a small leap to consider using AIC and other criteria. Weights can also be estimated to optimize some criterion on a hold-out sample within the training data before being validated on an out-of-sample set. This is the idea behind "stacking" or "stacked generalization" that originated with Wolpert (1992). A simple version of stacking, introduced by the Netflix Grand Prize winners as "blending", splits the data into 3 parts: fitting, weighing, and testing (Töscher and Jahrer, 2009; Koren, 2009). Fitting and weighing form the training dataset and models are validated on the testing set. However, this can be viewed as an inefficient use of the data. What people commonly call stacking always involves some kind of cross-validation to fully exploit the entire data set. Stacking is not widely used in Bayesian model combination because it only works with point estimates, not the entire posterior distribution. Based on the recent work by Yao et al. (2017), we use a form of stacking which gives a probabilistic forecast by minimizing proper scores. The purpose of this thesis is to assess whether three methods of model combination - namely model averaging, blending and stacking - perform better on average than the best base model.

Chapter 2 will describe the HHH4 framework as well as the theory behind model averaging and stacking. We will also introduce two univariate time series and one multivariate time series which will be analysed in further chapters. In Chapter 3, we will walk through an example of long-term and one-step-ahead univariate predictions and compare results across base and ensemble models. Chapter 4 will explain in more detail the optimization process taking place in the stacking and blending procedure and assess its stability and uniqueness of solutions. We will evaluate the impact of various parameters on the findings in Chapter 5. Finally, Chapter 6 will extend findings to the multivariate case.

Chapter 2

Methods

2.1 HHH4 Framework

The HHH4 multivariate time series framework is designed for spatially and temporally aggregated surveillance data. The formulation of the HHH4 framework is built upon an additive decomposition of disease incidence into an endemic and an epidemic component. The epidemic (or autoregressive) component represents an autoregression on the number of cases from the previous week. The endemic component captures the residual variation in incidence which cannot be explained by previous cases, such as seasonal trends.
Let Y_t denote the number of reported infections at week $t = 1,\ldots,T$. Conditional on past observations Y_{t-1}, counts Y_t are assumed to follow a negative binomial distribution

$$Y_t|Y_{t-1} \sim \text{NBin}(\mu_t, \psi) \tag{2.1}$$

with conditional mean

$$\text{E}(Y_t|Y_{t-1}) = \mu_t = \nu_t + \lambda y_{t-1}, \tag{2.2}$$

and overdispersion parameter $\psi > 0$. In this formulation, the unknown endemic and epidemic log-linear predictors are ν_t and λ_t respectively. Note that an identity link is used in place of a log-link to let the counts act directly on the conditional mean. The aforementioned formulation implies that the conditional variance of Y_t can be written as

$$\text{Var}(Y_t|Y_{t-1}) = \mu_t(1 + \psi\mu_t). \tag{2.3}$$

As described in Held and Paul (2012), seasonal variation can be accounted for in both the endemic and epidemic component as follows:

$$\log(\lambda_t) = \alpha^{(\lambda)} + \beta^{(\lambda)}t + \sum_{s=1}^{S}\{\gamma_s^{(\lambda)}\sin(\omega_s t) + \delta_s^{(\lambda)}\cos(\omega_s t)\},\ \omega_s = \frac{2\pi s}{52}, \tag{2.4}$$

$$\log(\nu_t) = \alpha^{(\nu)} + \beta^{(\nu)}t + \sum_{s=1}^{S}\{\gamma_s^{(\nu)}\sin(\omega_s t) + \delta_s^{(\nu)}\cos(\omega_s t)\},\ \omega_s = \frac{2\pi s}{52}, \tag{2.5}$$

where $\alpha^{(\cdot)}$ is an intercept, $\beta^{(\cdot)}$ is a trend parameter and the terms in the curly brackets are used to model seasonal variation with $\gamma_s^{(\cdot)}$ and $\delta_s^{(\cdot)}$ as unknown parameters. S denotes the number of harmonics to include and $w_s = 2\pi s/52$ for weekly data. We obtain more flexibility in describing the seasonal pattern by increasing the number of S. Seasonal terms can also be formulated as:

$$\gamma_s \sin(w_s t) + \delta_s \cos(w_s t) = A_s \sin(w_s t + \kappa_s) \tag{2.6}$$

where the amplitude $A_s = \sqrt{\gamma_s^2 + \delta_s^2}$ and the phase shift $\kappa = \arctan(\delta_s/\gamma_s)$ are directly identifiable.
All unknown parameters ν, λ, δ, γ and ψ are estimated directly by maximizing the negative binomial log-likelihood using numerical optimization routines (see Paul et al. (2008)).
This model was extended further by Bracher and Held (2017) in the package **hhh4addon** to allow for distributed lags. The conditional mean becomes:

$$\mathrm{E}(Y_t|Y_{t-D:t-1}) = \mu_t = \nu_t + \lambda \sum_{d=1}^{D} u_d Y_{t-d}, \tag{2.7}$$

where D is the maximum lag considered (in this thesis $D = 5$) and weights are normalized such that $\sum_{d=1}^{D} u_d = 1$. Currently only an exponentially decaying lag structure is implemented:

$$u_d = p(1-p)^{(d-1)}, \tag{2.8}$$

where p is estimated via a profile likelihood approach.

2.1.1 Age-Structured Multivariate Model Extension

Often, information about the number of cases in different geographical regions or age groups is available. Equations (2.1) – (2.8) can be extended to fit multivariate time series. For simplicity, we only consider data stratified by age-group similar to model 6 in Held et al. (2017). The interested reader is referred to Held et al. (2017) and Meyer and Held (2017) for a more exhaustive formulation of multivariate time series models also including geographical regions.
We assume $g = 1, \ldots, G$ "units" and denote with Y_{gt} the number of cases in unit g at time t. The symbol "$\cdot$" indicates that all units are considered. Y_{gt} still follows a negative binomial distribution with conditional mean

$$\mathrm{E}(Y_{gt}|Y_{\cdot,t-1}) = \mu_{gt} = \nu_{gt} + \phi_{gt} \sum_{g'} \lfloor c_{g'g} \rfloor y_{g',t-1}. \tag{2.9}$$

The contact matrix $\boldsymbol{C} = (c_{g'g})$ quantifies the average number of contacts of an individual of group g' with individuals of group g with each entry $c_{g'g} \geq 0$. The original formulation by Held et al. (2005) would include both autoregressive parameters λ_{gt} and ϕ_{gt} to estimate separate dynamics for the reproduction of the disease within a unit on the one hand, and transmission from other units on the other hand. However, Meyer and Held (2017) argue that the contact matrix offers an appealing

alternative to reflect predominant local autoregression and hence one log-predictor, $\boldsymbol{\phi}$, would suffice.
The autoregressive parameters are themselves modeled in a log-linear fashion:

$$\log(\nu_{gt}) = \alpha_g^{(\nu)} + \beta^{(\nu)} x_t + \gamma_g^{(\nu)} \sin(\omega t) + \delta_g^{(\nu)} \cos(\omega t), \tag{2.10}$$

$$\log(\phi_t) = \alpha_g^{(\phi)} + \gamma^{(\phi)} \sin(\omega t) + \delta^{(\phi)} \cos(\omega t), \tag{2.11}$$

in the simple case where seasonality of the first order is considered and no trend. Here x_t is an Christmas break indicator (calendar weeks 1 and 52 of each year).

2.2 Predictive Model Assessment

For each model mentioned in the previous section, one-step-ahead predictions, where the model is re-fitted after the addition of every new observation, and long-term predictions can be made. All one-step-ahead predictions follow negative binomial distributions, hence their predictive probability mass function is known in closed form. The same cannot be said for long-term prediction. Nevertheless, using iterative expectations Held et al. (2017) derived the first and second moments analytically . The function to derive predictive moments `predictive_moments` is included in the **hhh4addon** package.

The evaluation of predictive performance is based on the paradigm of maximizing the sharpness of the predictive distributions subject to calibration (Gneiting et al., 2007), where sharpness refers to the concentration of the predictive distribution and calibration refers to the statistical consistency between the probabilistic forecasts and the realizations.
The use of strictly proper scoring rules has been advocated to evaluate model performances and rank competing forecast procedures (Gneiting et al., 2007; Gneiting and Raftery, 2007; Czado et al., 2009). Scoring rules provide summary measures by assigning a numerical score $S(P, \mathbf{y}_{\text{obs}})$ based on the predictive distribution P and the later observed value $\mathbf{y}_{\text{obs}}$. They are typically negatively oriented, i.e. the smaller the better. A scoring rule is proper if the expected value of the penalty $S(P, \mathbf{y}_{\text{obs}})$ for an observation drawn from any other distribution G is minimized if $P = G$. It is strictly proper if the minimum is unique. A proper scoring rule is designed such that quoting the true distribution as the forecast distribution is an optimal strategy in expectation. Therefore, proper scoring rules encourage honest and sharp forecasts (Gneiting and Raftery, 2007). Propriety is essential to ensure that a scoring rule simultaneously addresses sharpness and calibration (Winkler et al., 1996).
Czado et al. (2009) provide a description of proper scoring rules suitable for count data. For a more general list of proper scoring rules see Gneiting et al. (2007).
The logarithmic score (LogS) defined as

$$\text{LogS}(P, \mathbf{y}_{\text{obs}}) = -\log(p_{yobs}). \tag{2.12}$$

is a commonly used score where p_{yobs} is the probability mass at the observed realization. The logarithmic score has been linked to Bayes factors. Indeed the log Bayes

factor is the difference between the logarithmic scores of two models.
Another score of interest is the ranked probability score (RPS) defined as

$$\mathrm{RPS}(P, \mathbf{y}_{\mathrm{obs}}) = \sum_{k=0}^{\infty} \big(P_k - 1(\mathbf{y_{obs}} \leq k)\big)^2, \tag{2.13}$$

or

$$\mathrm{RPS}(P, \mathbf{y}_{\mathrm{obs}}) = E_p|Y - \mathbf{y}_{\mathrm{obs}}| - \frac{1}{2}E_p|Y - Y'|. \tag{2.14}$$

where Y and Y' are independent copies of a random variable with distribution P. The RPS has the particularity to provide a direct way of comparing point forecasts and predictive distributions.
Finally, we consider the Dawid Sebastiani score (DSS)

$$\mathrm{DSS}(P, \mathbf{y}_{\mathrm{obs}}) = \Big(\frac{\mathbf{y}_{\mathrm{obs}} - \mu_P}{\sigma_P}\Big)^2 + \log \sigma_P, \tag{2.15}$$

$$\mathrm{DSS}(P, \mathbf{y}_{\mathrm{obs}}) = \log(|\boldsymbol{\Sigma}_P|) + (\mathbf{y}_{\mathrm{obs}} - \boldsymbol{\mu_p})^T \boldsymbol{\Sigma}_P^{-1} (\mathbf{y}_{\mathrm{obs}} - \boldsymbol{\mu_P}), \tag{2.16}$$

where the second equation represents a multivariate extension of the first equation, which is necessary when more than one timepoint is considered. In this analysis will report the scaled DSS

$$\mathrm{sDSS}(P, \mathbf{y}_{\mathrm{obs}}) = \frac{\mathrm{DSS}(P, \mathbf{y}_{\mathrm{obs}})}{2d},$$

where d is the dimension of the forecasting target. The DSS has the advantage of relying solely on the first two moments $\boldsymbol{\mu_p}$ and $\boldsymbol{\Sigma}_P$ of the predictive distribution. Therefore, we can only rely on the DSS to evaluate long-term forecasts, but on all three scores for one-step-ahead forecasts.
As previously mentioned, proper scoring rules take into account both calibration and sharpness. To have a better idea of which element affects a score more substantially, we will provide a measure of sharpness. Gneiting et al. (2008) suggest using the determinant sharpness (DS) which is a generalization of the standard deviation and defined as follow:

$$\mathrm{DS} = (|\boldsymbol{\Sigma}_P|)^{1/(2d)},$$

We will report

$$\log(\mathrm{DS}) = \log(|\boldsymbol{\Sigma}_P|)/(2d),$$

as used by (Meyer et al., 2017) in the **Surveillance** package which has the advantage of avoiding unnecessarily large numbers.

2.3 Model Averaging

Instinctively, it makes sense to aim for and select the model which best approximates the true underlying data generation process. Statistical software packages have conveniently included popular model selection criteria such as AIC and BIC making it straightforward for users to rank competing models and find the best one. Nowadays, the "standard practice" has apparently become to pick a best model and carry out inference, conveniently "forgetting" that a selection process took place. Ignoring model uncertainty leads to over-optimistic testing, confidence intervals, and biased inference. A Bayesian solution to this problem, called Bayesian model averaging, is to compute posterior probabilities that the given model is the data generating process model, conditional on the data and given that one of the models considered is true. Those probabilities are then used as weights, the final solution being a weighted average of all models.
In a broader formulation, the mean and covariance of the ensemble prediction Y of models $M_1, \ldots, M_J$ can be calculated as follows

$$\mathrm{E}(Y) = \sum_{j=1}^{J} w_j \mathrm{E}(Y|M_j) \tag{2.17}$$

$$\mathrm{Var}(Y) = \sum_{j=1}^{J} \{w_j \sigma_j^2 + w_j(\mathrm{E}(Y|M_j) - \mathrm{E}(Y))^2\} \tag{2.18}$$

where w_j are the weights and must thus be positive and sum up to 1, and σ_j is the standard deviation for a variable drawn from model M_j.
The function `MAMeanCov` that we implemented performs model averaging in R. It requires a list of predictions (`listmodel.pred`) and the weights (weights) to assign to each prediction. First, we extract the vector containing the means of each prediction (`mu_vector`) and store them in a list.

```
MAMeanCov <- function(listmodel.pred, weights){

  ### Calculate the ensemble model mean:

  # Extract means
  list.muvec <- lapply(listmodel.pred,
                       FUN = function(pred){unname(unlist(pred$mu_vector))})
```

We then multiply the mean prediction of each base models M by the corresponding weight and obtain J weighted mean vectors. Finally we sum all vectors to obtain the ensemble mean `mean.av`.

```
  # Multiply each element of the list (mu_vectors) by the corresponding weights
  mu.weight <- Map('*', list.muvec, weights)

  # Sum elements of the list element wise.
  mean.av <- Reduce('+',mu.weight)
```

Notice that as soon as we predict more than one timepoint, we work with vectors of predictions, $\mathbf{y}$, and matrices operations must apply. The Equation 2.19 and 2.18 can respectively be rewritten as:

$$\mathrm{E}(\mathbf{y}) = \sum_{j=1}^{J} w_j \mathrm{E}(\mathbf{y}|M_j) \tag{2.19}$$

$$\mathrm{Cov}(\mathbf{y}) = \sum_{j=1}^{J} \left\{ w_j \left[\underbrace{\mathrm{Cov}(\mathbf{y}|M_j)}_{\text{Term 1}} + \underbrace{\Big(\big(\mathrm{E}(\mathbf{y}|M_j) - \mathrm{E}(\mathbf{y})\big) \big(\mathrm{E}(\mathbf{y}|M_j) - \mathrm{E}(\mathbf{y})\big)^{\mathsf{T}} \Big)}_{\text{Term 2}} \right] \right\} \tag{2.20}$$

To calculate the ensemble variance, we first perform the operation inside "Term 2" in Equation 2.20 where we subtract the mean of each model M with the ensemble mean, convert the vector to a matrix and multiply it by its transpose. The resulting matrices for each M models is stored in a list.

```
  ### Calculate the ensemble model variance.

  # substract ensemble mean to model means + transpose
  SubTrans <- function(muvec){
    std <- muvec - mean.av
    mat.std <- as.matrix(std) %*% t(as.matrix(std))
    return(mat.std)
  }

  list.mat <- lapply(list.muvec, SubTrans)
```

We continue by extracting the covariance matrix for each M model ("Term 1") and adding it to "Term 2". Finally we multiply the term in brackets ("Term 1"+"Term 2") by the appropriate weights to obtain the covariance matrix of the ensemble model.

```
  # Extract covariance matrix for each model
  list.Sigma <- lapply(listmodel.pred, FUN= function(x)x$Sigma)

  # Add the covariance matrix to the "transpose" element
  list.Sigma.mu <- Map('+', list.Sigma, list.mat)

  # Multiply each element of the list by the corresponding weight
  list.Sigma.mu.w <- Map('*', list.Sigma.mu, weights)

  # Sum elements of the list element wise.
  cov.av <- Reduce('+',list.Sigma.mu.w)

  MAlist <- list(MAmean= mean.av, MAcov= cov.av)

  return(MAlist)
}
```

The function returns the mean and covariance matrix of the ensemble model.

In Bayesian model averaging, issues arise from setting posterior probabilities as weights. As often with Bayesian statistics, the question of how to define the priors has to be solved and the computation of posterior probabilities forces the user to use computer-intensive methods. Moreover, this procedure would require the knowledge of the full probability distribution, which is not known for long-term forecasts within the HHH4 framework.
Claeskens and Hjort (2008, chapter 3.2) demonstrate that the posterior distribution for equally likely models can be approximated by a "smooth BIC"

$$\omega_j = \frac{\exp(\frac{1}{2}\mathrm{BIC}_j)}{\sum_{i=1}^{J}\exp(\frac{1}{2}\mathrm{BIC}_i)}, \tag{2.21}$$

with BIC defined by Schwarz (1978) as

$$\mathrm{BIC}_j = \dim(M_j)\log(n) - 2\text{log-likelihood}(M_j) \tag{2.22}$$

with $\dim(M_j)$ being the number of parameters estimated in the model, and n the sample size of the data which in our case translates to the length of the time series used to fit the model.
Formulated as here above, the best model has the smallest BIC. For numerical stability reasons, we use

$$\Delta\mathrm{BIC}_j = \mathrm{BIC}_j - \max_i \mathrm{BIC}_i \tag{2.23}$$

instead of BIC_j. The BIC estimator has interesting properties, among them the fact that it is consistent in the sense that if the true model is among the candidates, the probability of selecting the true model approaches 1 as the sample size, n, approaches infinity (Nishii, 1984). However, if the true model does not belong to the sample of models suggested, BIC may fail to find the best model. Since AIC asymptotically selects the model that minimizes mean squared error of a prediction or estimation, Burnham et al. (1994) suggest to use it instead of BIC. Sakamoto et al. (1986) note that "AIC is not a criterion for the estimation of the true order but the one for the best model fit" and Vrieze (2012) shows in a simulation study that the risk of selecting a bad model is minimized with the use of AIC compared to BIC. Buckland et al. (1997) suggest the use of smoothed AIC in model averaging. Weights are calculated as

$$\omega_j = \frac{\exp(\frac{1}{2}\mathrm{AIC}_j)}{\sum_{i=1}^{J}\exp(\frac{1}{2}\mathrm{AIC}_i)}, \tag{2.24}$$

with AIC defined by Sakamoto et al. (1986) as

$$\mathrm{AIC}_j = 2\dim(M_j) - 2\text{log-likelihood}(M_j), \tag{2.25}$$

again, for numerical stability reasons, we use

$$\Delta\mathrm{AIC}_j = \mathrm{AIC}_j - \max_i \mathrm{AIC}_i \tag{2.26}$$

instead of AIC_j. The penalty still increases with increasing model complexity but does not depend on the number n of observations, hence AIC often penalizes model

complexity to a lesser extent than BIC. AIC has been criticized for its tendency to select complex models. The merits of these two approaches are discussed in more detail in Kass and Raftery (1995). In our analysis, we will use and compare both methods.

2.4 Stacking

The basic idea behind "Stacking" as originally described by Wolpert (1992) is to define a pool of base models and use another second level model, the "super-learner" or "meta-learner", to combine their predictions, with the aim of reducing the mean squared forecast error. Popular meta-learners are, but not limited to, linear and logistic regression. The fact that predictions of the base models are used as covariates (or "features") for the meta-learner means that, in theory, the meta-learner is able to highlight each base model where it performs best and discredit each base model where it performs poorly. For this reason stacking is most effective when the base models are somewhat different (Wolpert, 1992). Other covariates, engineered or not such as the performance rank of the base models, can be introduced in the second level combination with the hope of improving the predictions. Moreover, there are numerous ways a user can extend the stacking model to more than two levels, increasing the complexity of the overall ensemble model.
Although the concept of stacking was developed early on, its benefits were only demonstrated empirically. The theoretical guarantees for stacking were not proven until the publication of the "super learner" paper by van der Laan et al. (2007) which shows that the "Super Learner ensemble" represents an asymptotically optimal system for learning.

Most of the early literature limits stacking to averaging point estimates. As a result, the method was quickly adopted by the machine learning community but did not convince other fields where entire predictive distributions or probabilistic forecasts are required. Moreover, the traditional stacking procedure is designed to minimize the mean squared predictive error, but most predictions are not evaluated based on this criterion. In a recent paper, Yao et al. (2017) extend the use of stacking to the combination of predictive distributions. Instead of minimizing the mean square forecast errors, the authors optimize the log-score in examples based on simulated and real data. The authors argue that any proper scoring rule can be used.
This approach fits the need of probabilistic forecasts and gives the possibility to optimize the same criterion as used to validate the results. The authors compute leave-one-out posterior distributions which are not applicable in our time series setting. We will adapt their idea to our conditions and define two different stacking approaches: *blending* and what we will call *stacking*.

2.4.1 Blending & Stacking

A simple form of stacking, called "blending" was used by the winning teams of the Netflix Grand Prize (Töscher and Jahrer, 2009; Koren, 2009). Blending splits the

data into 3 parts as shown in Figure 2.1. The first subset is used to fit the base models. We use these fitted models to draw predictions for a second subset called stack weight subset and for the test data. Finally, the meta-learner assigns weights to the base models based on their performance in predicting the stack weight subset. Logically, the meta-learner assesses the performance using the same criterion as later to evaluate the test set. The appropriately weighted combination of predictions of the test set results in the final forecast for the test set which can be in turns evaluated. When blending, two strategies are possible: (i) weights can be optimized to minimize the score of the overall blending periods (`Blend-Sum`) or (ii) they could optimize each year of the blending period individually and then be combined by simply averaging the weights (`Blend-Ind`). To the best of our knowledge, there is no recommended method. The choice between the two could rely on how heavily bad predictions should be penalized. Poor predictions will not penalize a model too badly if we follow a `Blend-Ind` strategy because a poor score will be diluted in an average, whereas it could remove a model completely from the ensemble if we consider `Blend-Sum`. The decision on which strategy to favor should therefore be taken after considering the purpose of the model and together with healthcare professionals. We will show both scenarios in Chapter 3 but will then focus on `Blend-Sum` in Chapter 5.
Besides its relative simplicity, blending has the advantage of preventing any data leakage (i.e. when the data used to train the models happens to have information from the test data) since each level of the model uses different parts of the data. However, it can be viewed as an inefficient use of the data, especially if the number of levels in the model is high.

To overcome these drawbacks, the commonly called "stacking" procedure always involves some form of cross-validation (CV), usually a K-fold CV. In our application however, the use of K-fold CV is somehow limited by the autocorrelated nature of the data and the intuition that we should not use the future to predict the past. When dealing with time series, practitioners usually resort to out-of-sample evaluation (which would result in blending) or some time series cross-validation techniques. In the paper by Bergmeir et al. (2017), authors demonstrate that for purely autoregressive models, the use of standard K-fold CV is possible provided the models considered have uncorrelated errors. They use simulation studies to show that the performance of the K-fold cross-validation outperforms other forms of validation in time series. However, because the **surveillance** package was not designed for K-fold CV it would be very tedious to adapt all the necessary functions and is beyond the scope of this thesis.
Instead, we will use a procedure inspired by the "evaluation on a rolling forecasting origin" as described by R. Hyndman (2016) in his blog, which we will more simply call "rolling CV" (Figure 2.2). Adapted to stacking, the idea is to use multiple series of fitting and stack weight sets, each one containing one more time unit than the previous one until we reach the test year. Please note that in our illustration a time unit is defined as a full disease cycle equivalent to a year. We hence re-fit the models with an extra full year in the following series and predict an entire year. In the case of one-step-ahead predictions we still use the concept of stacking years to stack weights, but re-fit the models before every single observation. The concept of

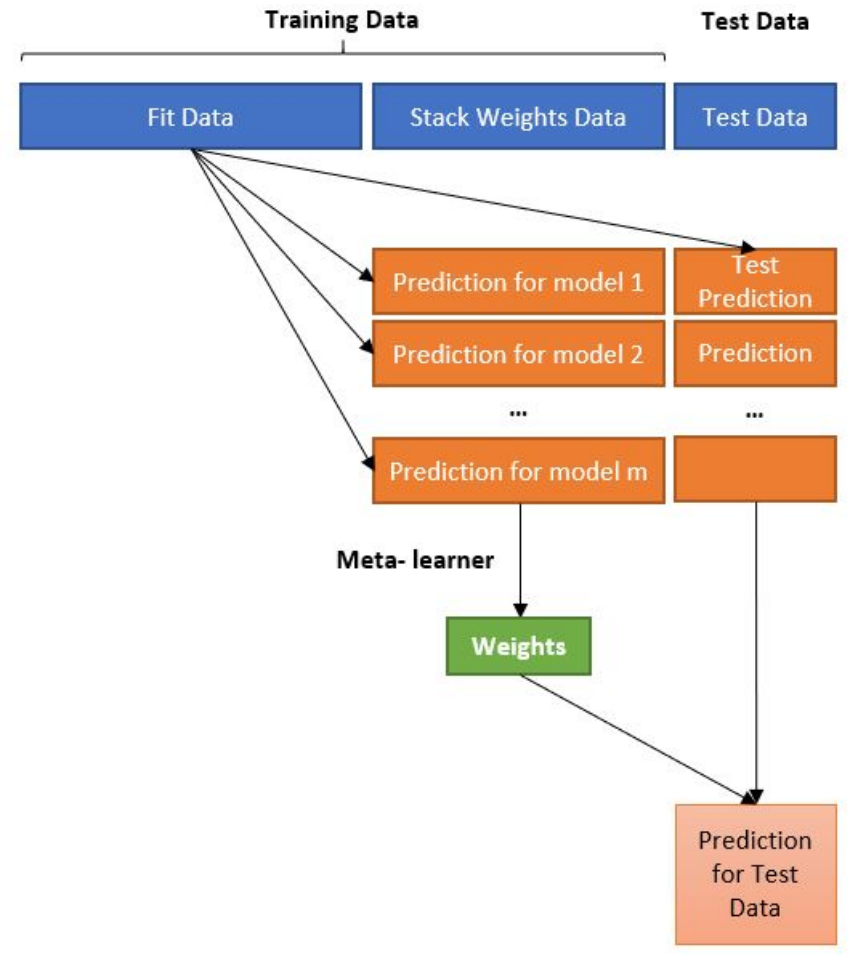

Figure 2.1: **Schematic Representation of Blending.** Blending relies on the splitting of the data into 3 parts: a fit, stack weights and test data. Base models are fitted on the fit data. Predictions are made for the stack weights data and the test data. Note that the models are not re-fitted to the entire training data. A meta-learner evaluates the predictions made on the stack weights data to assign weights to each base model. The weighted combination of the predictions for the test data results in the final forecast.

blending is thus ill fitted for this use and we will only discuss of stacking for one-step-ahead predictions. In both cases, the training window can be extendable (Figure 2.2, panel A) or of fixed length (Figure 2.2, panel B) so that early observations are progressively dropped. In the HHH4 framework many years are required to build a stable model, we will thus not implement such a "moving window" approach in this thesis. However, it is not an uncommon practice, the reason being that the model can forget old data and adapt faster to new conditions. We can easily imagine different ways of using the predictions to stack weights as shown in (Figure 2.2, panel C). The options would be to use all the data available, only the season of interest, or the most recent observations. There are arguments in favor of each of them and we believe it is up to the researcher, based on his best judgement, to decide what is more appropriate. Regardless of whether the training data is fixed or extendable, the length of the stacking/blending period are open parameters and likely open to discussion.
Figure 2.3 shows a global view of the stacking process. The rolling CV is used in the first stage of the process to build a set of predictions within the training set. Similar to blending, a meta-learner uses predictions and realizations of the training data set to estimate weights for each model. However, due to the nature of the rolling-CV procedure we now consider each prediction individually and average the weights obtained from each optimization. Finally, all models are fitted on the entire training data set and predictions are made for the test set. The predictions from each model are appropriately weighted to give the final predictions which can then be evaluated by proper scoring rules against the realizations of the test data.

Top Kagglers talk about the "art" rather than the "science" of stacking. With experience and much trial and error, top-performers get an intuition of which models to use and how to structure their stacking.
The aim of this work is not to deliver the best predictions or the best stacking model, but rather a proof-of-concept that ensemble methods work within the HHH4 framework. We will introduce this procedure with real data examples and discuss its limitations and complexity.

2.5 Data

This section describes the three different datasets used in this thesis. Two univariate time series are studied in Chapters 3, and 5 and Chapter 6 will focus on a multivariate case.

2.5.1 Univariate Time Series

The first data set (`CHILI`) consists of weekly counts of influenza like illness (ILI) cases in Switzerland from 2000/01 to 2016/01 [1], estimated from the Swiss Sentinella Reporting System and integrated in the **HIDDA.forecasting** package.
ILI is a medical diagnosis of possible influenza or other illness causing a set of

[1] year/week

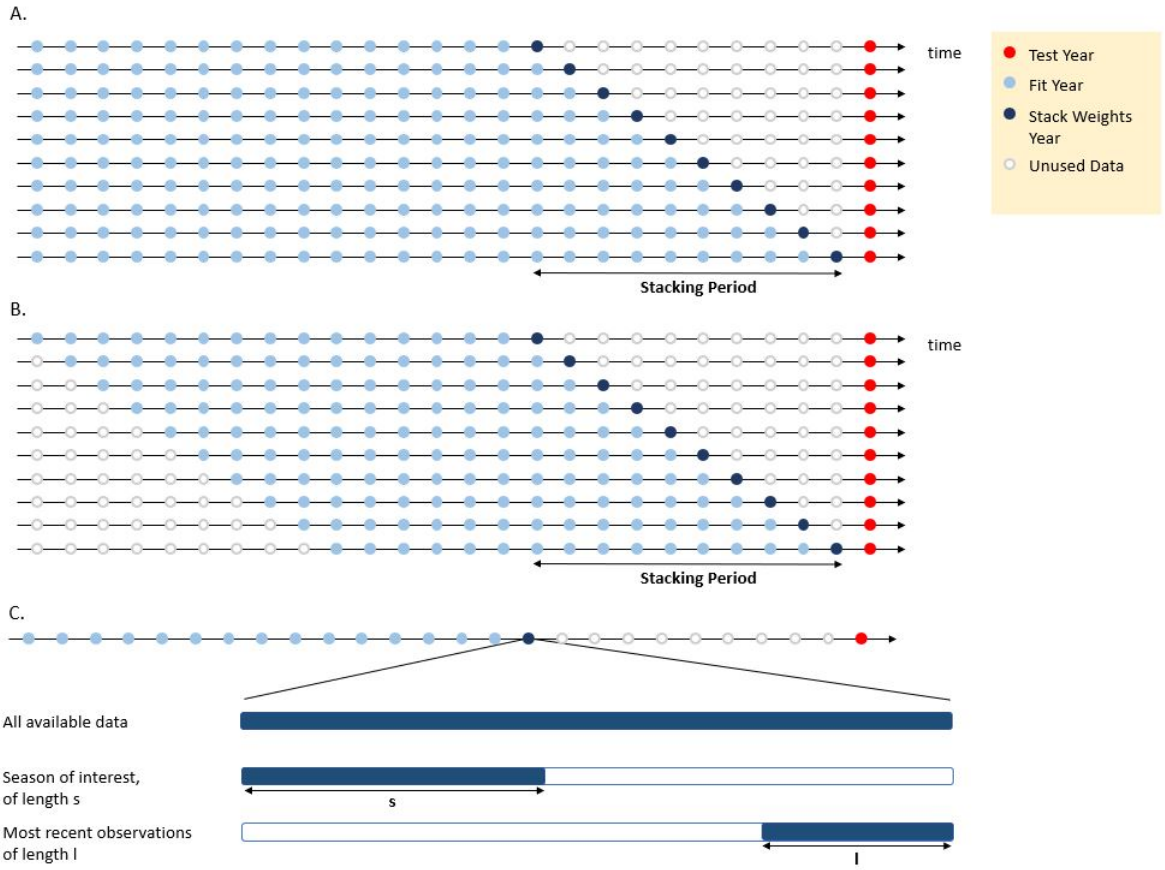

Figure 2.2: **Schematic representation of time series cross-validation technique "rolling CV" in the context of stacking.** In order to make a prediction for the test year with the stacking procedure, a number of years prior to the test year are selected as the stacking period. For each year of the stacking period, base models are fitted on previous data and a prediction is made. The predictions for the stacking period are then used to define weights for each base model. Panel **A** shows an adaptive rolling CV procedure where the number of years to fit the base models increases with each year of the stacking period. In panel **B** the number of fit years stays constant. Panel **C** shows what information can be used within each stacking year.

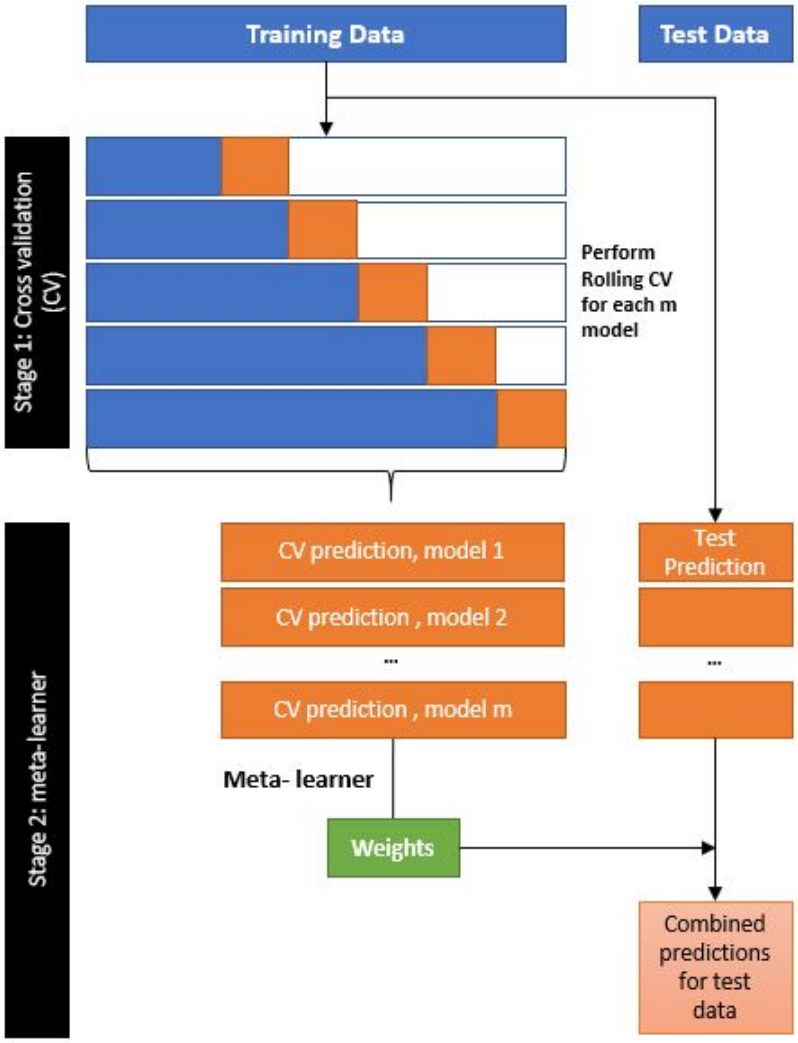

Figure 2.3: **Schematic representation of Stacking.** Stacking splits the data in only two parts and relies on cross-validation. The base models are successively fitted an predictions are made within the training data according to a rolling CV. A meta-learner assign weights to each base model by evaluating the predictions made on the training data individually, averaging the results of each unit. The base models are finally re-fitted on the entire training dataset and used to make predictions on the test set and the weighted combination of these predictions results in the final forecast.

common symptoms, namely fever and coughs within the last 10 days. ILI can be caused by a variety of microbial agents other than influenza viruses, and the range of symptoms observed with influenza virus infections is nonspecific and resembles the clinical picture of a variety of other pathogens (Monto et al., 2000). This uncertainty poses challenges when diagnosing influenza and for influenza surveillance. However, the case definition for ILI is not necessarily intended to capture all cases but to describe trends over time (WHO, 2007). Cases follow a seasonal cycle, peaking in winter in the northern hemisphere as can be seen in Figure 2.4. The prediction targets are highlighted in the figure. Chapter 3 will use the last year as test data whereas chapter 5 will require the last 5 years to assess reproducibility. We consider the season of interest for ILI to be 30 weeks long starting on week 1 of each year. We have defined the ILI season in this manner so as to predict the period with high incidence, the 30^{th} week having on average the lowest incidence, as can be seen in Figure 2.5.

The second time series is a weekly count of reported cases of dengue fever in San Juan, Puerto Rico from 1990/18 to 2013/35 as provided by the Puerto Rico Department of Health and Centers for Disease Control and Prevention (US Department of Commerce, n.d.). The data was part of a forecasting competition but complete data can be found at `https://predict.phiresearchlab.org/`
Dengue is a viral mosquito-borne disease which is widespread in tropical and subtropical countries. Symptoms may include a high fever, headaches, vomiting, muscle and joint pains, and a characteristic skin rash. There is no specific treatment for dengue nor vaccine. The best prevention remains effective vector control (WHO, 2017). Incidence of dengue fever usually reaches a peak during the wet season in the summer months as seen in Figure 2.6. The prediction targets are highlighted in the figure and will be used similarly as for the ILI dataset. Regarding prediction length and onset for dengue, we have decided to follow the forecasting competition guidelines and define a dengue season as 52 weeks starting at week 18 of each year. Historically this has been the week of the year with lowest dengue incidence.

Both of these time series exhibit fairly regular seasonal trends. However, within these general trends there is variation in the timing and severity of the disease seasons, with more variability across different seasons for dengue than for ILI. Studying both time series should allow us to observe whether models' performances vary according to the regularity of the time series.

2.5.2 Multivariate Time Series

We use `BNV`, an age-stratified norovirus surveillance dataset from Berlin , Germany, together with an age-structured social contact matrix from the POLYMOD survey (Mossong et al., 2008), as provided in the `R` package **hhh4contacts**. This data has been analyzed by Held et al. (2017) and Meyer and Held (2017). The code we used to display this dataset is largely inspired by the yet to be published book chapter by Held and Meyer (n.d., chapter 6) which also analyses this dataset.

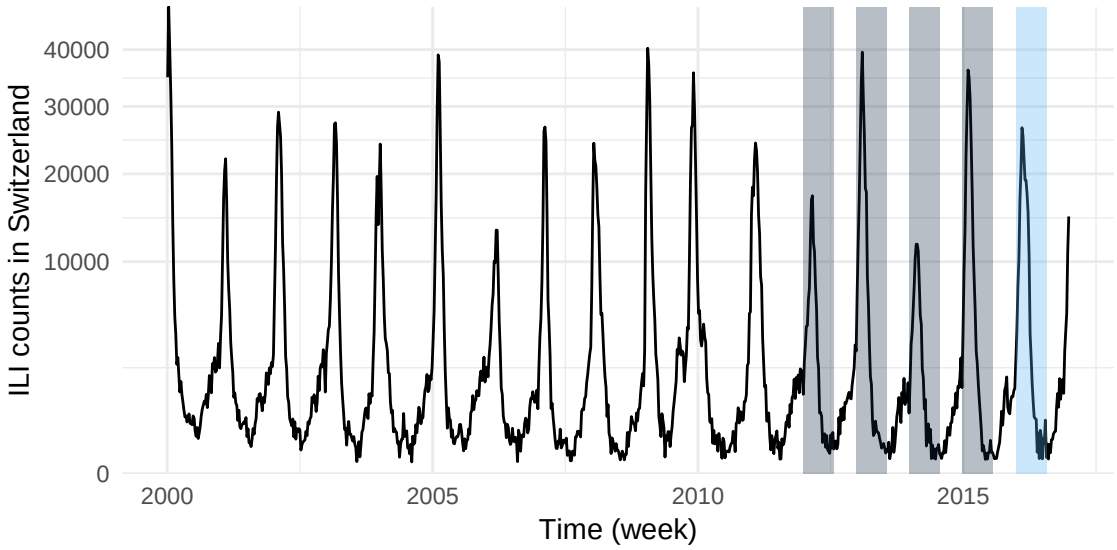

Figure 2.4: **ILI time series counts in Switzerland from 2000/1 to 2016/52.** All plots use the same $\sqrt{}$-scale. The last 5 years are highlighted and will be used as test data in chapter 5. The last year will be used as an example in chapter 3.

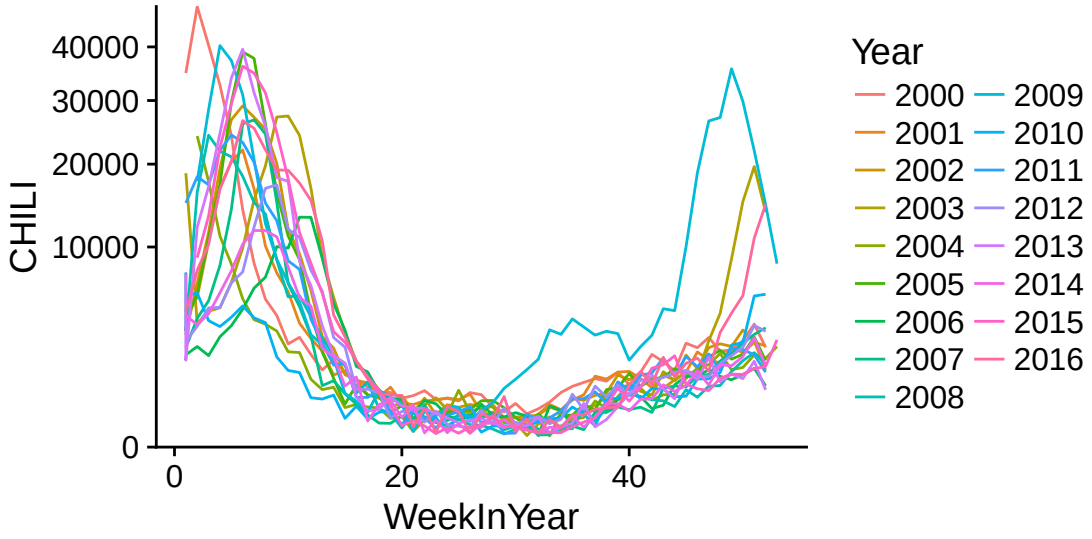

Figure 2.5: **Overlaid CHILI time series.** All plots use the same $\sqrt{}$-scale

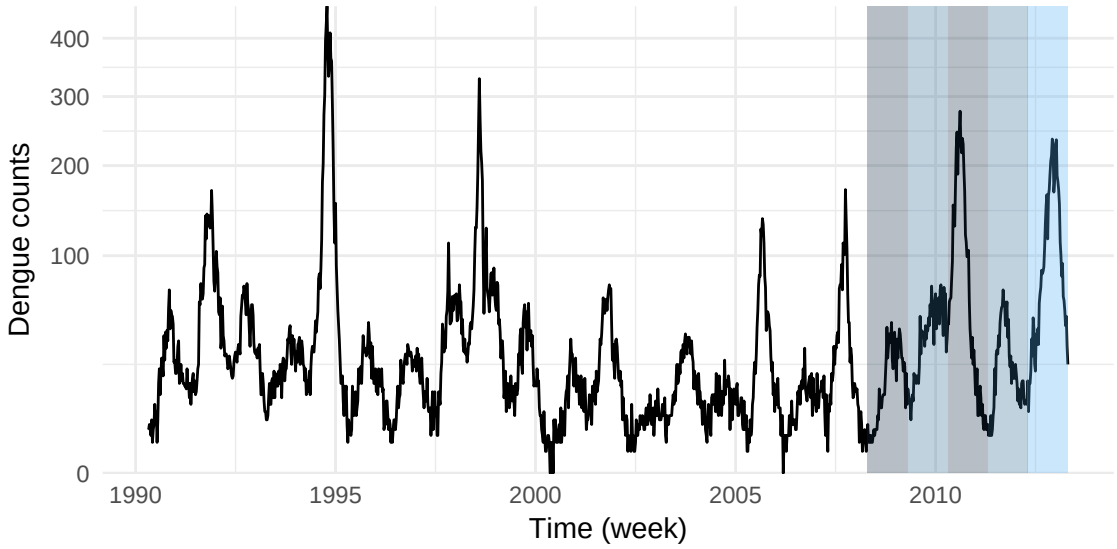

Figure 2.6: **Dengue time series counts in Puerto Rico from 1990/18 to 2013/13.** All plots use the same $\sqrt{}$-scale. The last 5 years are highlighted and will be used as test data in chapter 5. The last year will be used as an example in chapter 3.

These counts cover five norovirus seasons, from week 27, 2011 to week 26, 2016, and are stratified by the 12 city districts and six age groups: 0–4, 5–14, 15–24, 25–44, 45–64, and 65+ years of age. Here we only consider models for spatially aggregated counts, i.e. stratified by age-group only. The last year (2015/27 to 2016/26) has been used to assess model predictions (test data). Figure 2.7 shows the age-specific time series.
Noroviruses are the most common cause of acute gastroenteritis, which is characterized by vomiting, diarrhea and stomach pain. The virus is transmitted directly from person to person but also via contaminated surfaces or food. In the northern hemisphere, norovirus incidence consistently peaks during wintertime (Ahmed et al., 2013).
The reported incidence in our data is higher in pre-school children and the retired population than in the other age groups. Comparing seasonality between the age groups, the peak incidence in pre-school children seems to precede the peak in the highest age group. This phenomenon has previously been observed (Bernard et al., 2014). This underlines the potential of incorporating social information to better model the incidence of this pathogen.
On a more general note, this time series can be considered as fairly regular.

We will consider four different contact matrices to quantify the relation between the different age groups which are displayed in Figure 2.8.
The `Homogeneous` contact matrix has identical rows which implies all groups mix with each other equally. Since ϕ_{gt} will incorporate any group specific effects (in Equation 2.9 we sum over g' not g, any group effect will thus be extracted from the sum into ϕ), a matrix of ones ($\mathbf{C} = \mathbf{1}$) will induce the same contact structure.

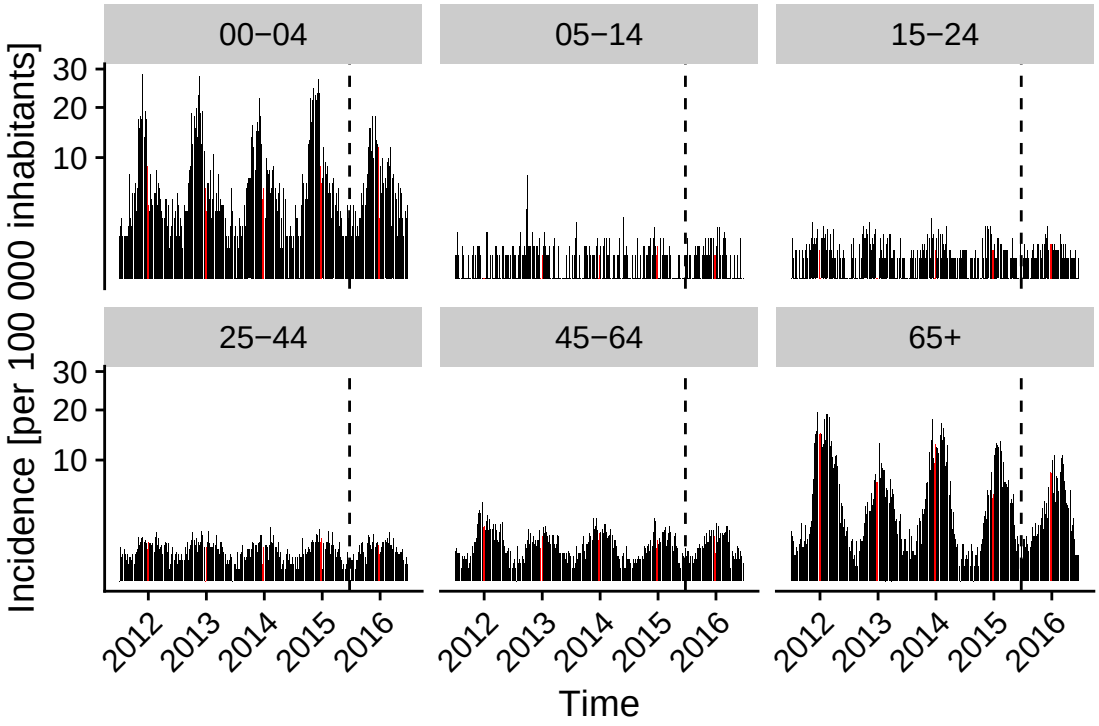

Figure 2.7: **Age-stratified time series of norovirus gastroenteritis incidence (per 100 000 inhabitants) in Berlin, 2011-W27 to 2015-W26.** All plots use the same $\sqrt{}$-scale. The Christmas break in calendar weeks 52 and 1 is highlighted in red. The vertical dotted line separates training from testing data.

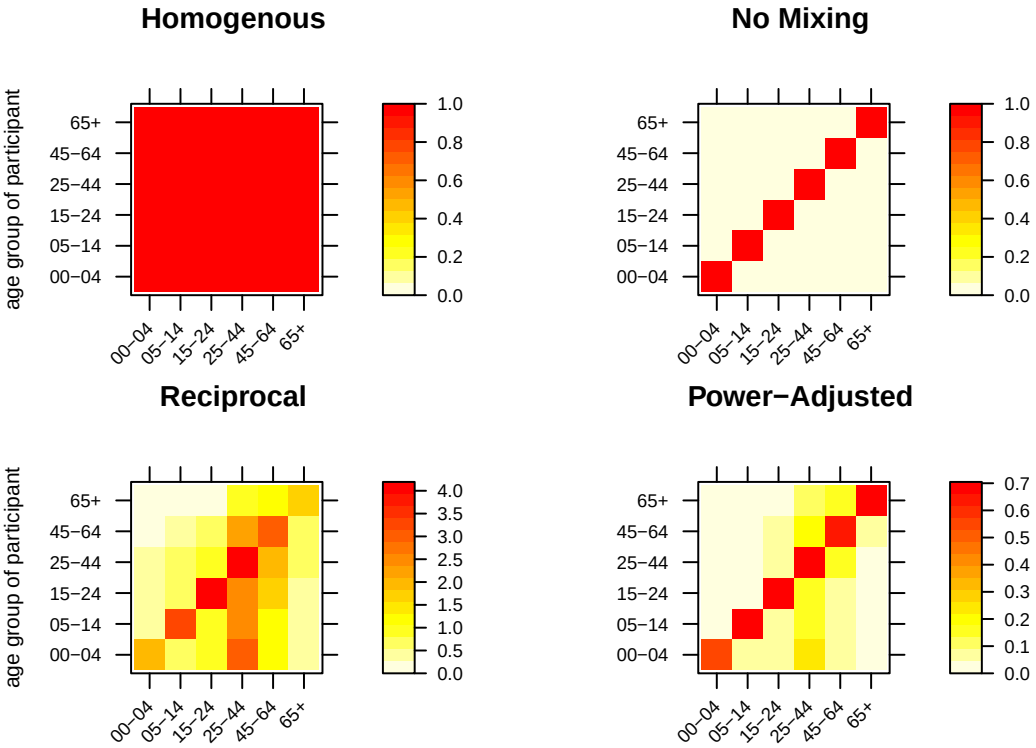

Figure 2.8: **Age-structured contact matrices.** The entries refer to the mean number of contact persons per participant per day.

The `No mixing` contact matrix is a diagonal matrix ($\mathbf{C} = \mathbf{I}$). In this case there is no mixing between the different age groups. These are two special cases of contact structures. A more plausible contact matrix is the contact matrix estimated from the POLYMOD survey in Germany (Mossong et al., 2008) and used in Held et al. (2017) and Meyer and Held (2017). The original contact matrix was aggregated to match the age groups of the surveillance data and further improved to ensure reciprocity on the population level (Wallinga et al., 2006). The result is the `Reciprocal` contact matrix. The last matrix, `Power-Adjusted`, is specified using a power transformation for the contact matrix:

$$\mathbf{C}^k := \mathbf{E}\boldsymbol{\Lambda}^k\mathbf{E}^{-1}, \tag{2.27}$$

where Λ is the diagonal matrix of eigenvalues and $\mathbf{E}$ is the corresponding matrix of eigenvectors of $\mathbf{C}$. The exponent k adjusts the amount of transmission between the age groups and is to be estimated as part of the epidemic model component via a profile likelihood approach.
These different models will be used in chapter 6.

Chapter 3

Univariate Analysis: an Example

In this chapter, we apply the various ensemble methods to make long-term and one-step-ahead predictions on the CHILI dataset. We study five different ensembles models for long-term forecasts: model averaging according to AIC and BIC scores (`MA-AIC` & `MA-BIC`), two types of blending (`Blend-Ind` & `Blend-Sum`) and stacking (`Stack`). For one-step-ahead predictions we will look at model averaging (`MA`) and stacking (`Stack`).
We use the last year (2016) as the test data and all previous information (1990-2015) as training data.

We consider the following base models:

1. No Seasonality (`No S`),
2. Seasonality in endemic component (`End`),
3. Seasonality in autoregressive component (`AR`),
4. Seasonality in endemic and autoregressive component (`End+AR`),
5. Seasonality in endemic and autoregressive component + trend parameter (`End+AR+t`),
6. Seasonality of second order in endemic and autoregressive component + trend parameter(`S2 + t`),
7. Seasonality of third order in endemic and autoregressive component + trend parameter(`S3 + t`),
8. Seasonality of first order in endemic and autoregressive component + exponentially decaying lags (`Exp`),
9. Seasonality of first order in endemic and autoregressive component + exponentially decaying lags + trend parameter (`Exp+t`),
10. Seasonality of second order in endemic and autoregressive component + exponentially decaying lags + trend parameter (`Exp+t+S2`),
11. Seasonality of third order in endemic and autoregressive component + exponentially decaying lags + trend parameter (`Exp+t+S3`).

Appendix A, Section A.2.1 shows the code used to build these base models. There are, of course, many other possible models we could include within the HHH4 framework, but we consider this list to be a good mix between very simple models (1-4)

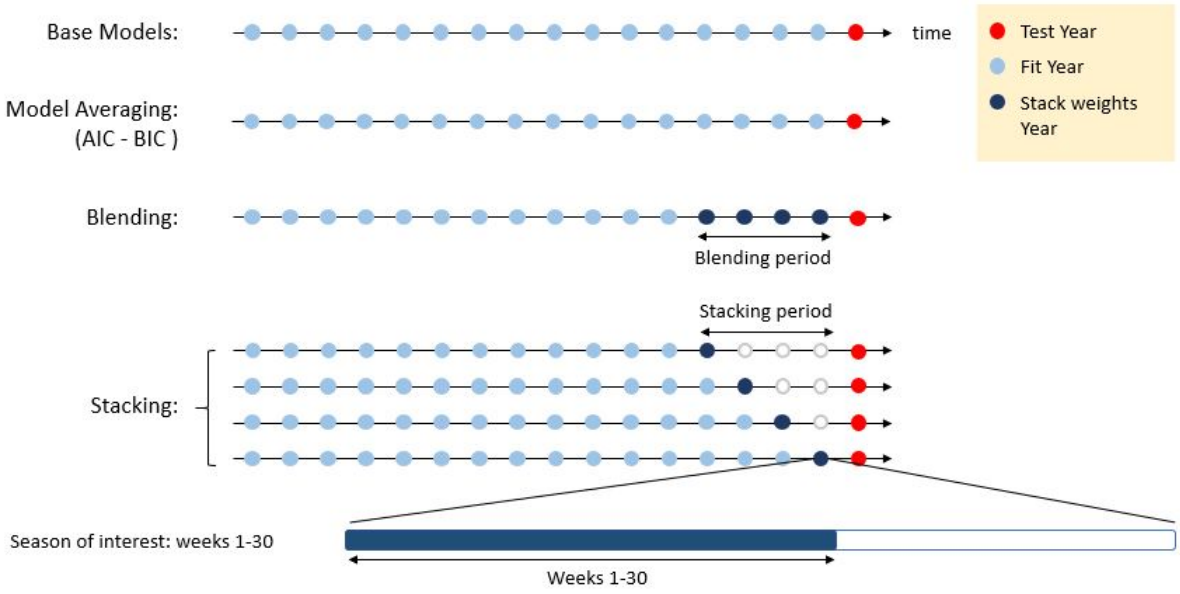

Figure 3.1: **Data analysis representation.** The CHILI dataset contains 17 years of influenza-like-illness counts in Switzerland (1990-2016) represented by 17 dots in the illustration. Years 1990-2015 are used as training data to predict year 2016. For base models and model averaging the entire training data is used to fit the models, no extra data is required to set weights. When blending, a period of 4 years (2012-2015) is used as a hold-out set to stack weights. Base models are never fitted on this hold-out set at all as opposed to stacking which gradually extends the fitting data.

and more complex models involving higher orders of seasonality (6, 7), exponentially decaying lags (8-9) or both (10, 11).
We performed the analysis twice, first on a pool of nine models (1-9) and then adding models 10 and 11 to the pool for reasons we will disclose further.
We also included a baseline model in the score analysis (but not in the pool of base models). The baseline model uses the mean and variance of the corresponding weeks in previous years. For example, to predict week 1 of 2016, this model will average the values of the weeks 1 in years 1990-2015 and use the variance of these counts.

3.1 CHILI Analysis

Figure 3.1 shows how the data is split for each type of model. For base models and model averaging the entire training data is used to fit the models, no extra data is required to set weights. When blending, a period of 4 years (2012-2015) is used as a hold-out set to blend weights. The models will not be fitted on this hold-out set at all as opposed to stacking which gradually extends the fitting data. There are two strategies possible to calculate the weights assigned to each base model: `Blend-Sum` and `Blend-Ind`. In this chapter, we show both. When stacking, the weights are optimized for each roll and then averaged (simple average).

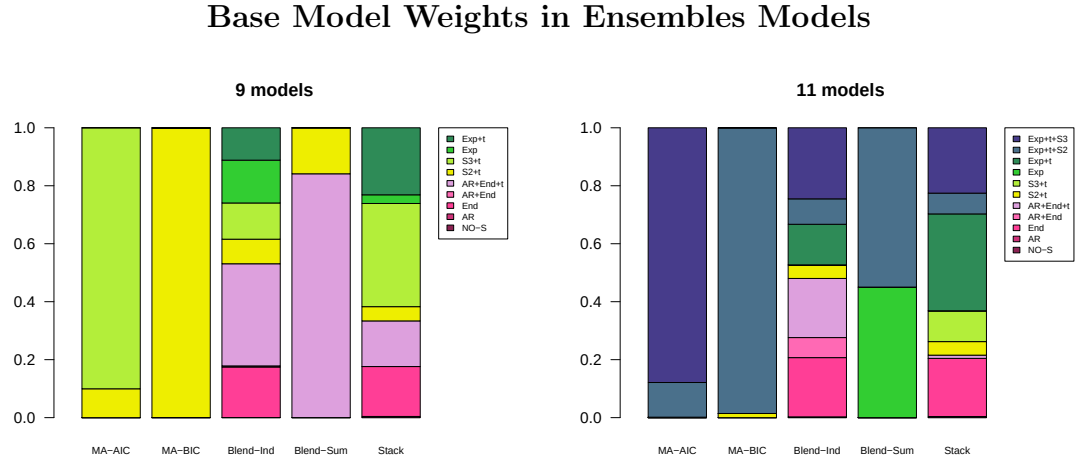

Figure 3.2: **Base model weights** in ensemble models issue from a pool of 9 (left panel) and 11 (right panel) base models.

3.1.1 Long-Term Predictions

Table 3.1: Base model weights in ensembles, 9 models.

	MA-AIC	MA-BIC	Blend-Ind	Blend-Sum	Stack
NO-S	0.00	0.00	0.00	0.00	0.00
AR	0.00	0.00	0.00	0.00	0.00
End	0.00	0.00	0.17	0.00	0.17
AR+End	0.00	0.00	0.00	0.00	0.00
AR+End+t	0.00	0.00	**0.35**	**0.84**	0.16
S2+t	0.10	**1.00**	0.08	0.16	0.05
S3+t	**0.90**	0.00	0.12	0.00	**0.36**
Exp	0.00	0.00	0.15	0.00	0.03
Exp+t	0.00	0.00	0.11	0.00	0.23

We start by analyzing the weights of individual base models in each ensemble model for the pool of 9 and 11 base models as shown in Figure 3.2 and detailed in Tables 3.1 and 3.2.
When 9 base models are present, `MA-AIC` favors the `S3+t` but also includes the `S2+t` model whereas the `MA-BIC` model exclusively takes after the `S2+t` model. This is not surprising as AIC based selection can favor models with greater complexity. `Blend-Ind` and `Stack` models have a wider distribution of weights and include simpler base models.
Both methods are closely related as the only difference between the two is that `Stack` refits the base models after predicting a stacking period whereas `Blend-Ind` does not. This can be seen in the weight distribution which is relatively similar between both methods yet distinct, indicating that including an extra year at each roll to refit the base models plays a role. Finally, `Blend-Sum` only include base models `Ar+End+t` and `S2+t`. It is likely that the base models present in `Blend-Ind` but not in `Blend-Sum` were less efficient in one blending period and were more heavily

Table 3.2: Base Model Weights in Ensembles, 11 models

	MA-AIC	MA-BIC	Blend-Ind	Blend-Sum	Stack
NO-S	0.00	0.00	0.00	0.00	0.00
AR	0.00	0.00	0.00	0.00	0.00
End	0.00	0.00	0.20	0.00	0.20
AR+End	0.00	0.00	0.07	0.00	0.00
AR+End+t	0.00	0.00	0.20	0.00	0.01
S2+t	0.00	0.01	0.05	0.00	0.05
S3+t	0.00	0.00	0.00	0.00	0.11
Exp	0.00	0.00	0.00	0.45	0.00
Exp+t	0.00	0.00	0.14	0.00	**0.33**
Exp+t+S2	0.12	**0.99**	0.09	**0.55**	0.07
Exp+t+S3	**0.88**	0.00	**0.25**	0.00	0.23

penalized.
Noteworthy is the fact that exponentially decaying models contribute very little to the ensemble models. Based on preliminary work on other datasets (not included), we expected considerable weight on models including exponentially decaying lags. As this first analysis indicated that seasonality may be of more importance than structured lags, we repeated the analysis including two extra base models with both components: `Exp+t+S2` and `Exp+t+S3`.
The weights of base models in ensemble models when using the pool of 11 base models is shown in the right panel of Figure 3.2. We observe a similar weight pattern in model averages but now clearly favoring models `Exp+t+S3` and `Exp+t+S2`. In blending and stacking we also see a similar pattern as with the pool of 9 models, except that more complex models are now present in the mix. We note that simple models are still present however it seems that models including seasonality without structured lags are less present.
In conclusion, this indicates that for this dataset, seasonality plays a more important role in model performance than exponentially decaying lags. We chose to further analyse the test year only considering the pool of 11 models.

3.1.2 Test Year & Scores

We move to the analysis of the performance of each model on the test year. Figure 3.3 shows the predicted mean counts and standard deviation for the test period. We observe that the predicted counts are correlated with the variance, models predicting higher counts having a larger variance. This is a direct consequence of the model formulation with a negative binomial distribution.
There is a lot of variance between the predictions of each model. Ensemble models do not predict the most extreme values. We also observe that all predictions underestimate the disease incidence. Interestingly, the baseline models seems to perform relatively well in this situation.
Figure(3.4) shows the fitted counts for the last four years of the training data and the predictive distribution for the test year for `Exp+t+S3` base model (See Appendix A, Section A.3 for all base models). This Figure illustrates how the model is closely

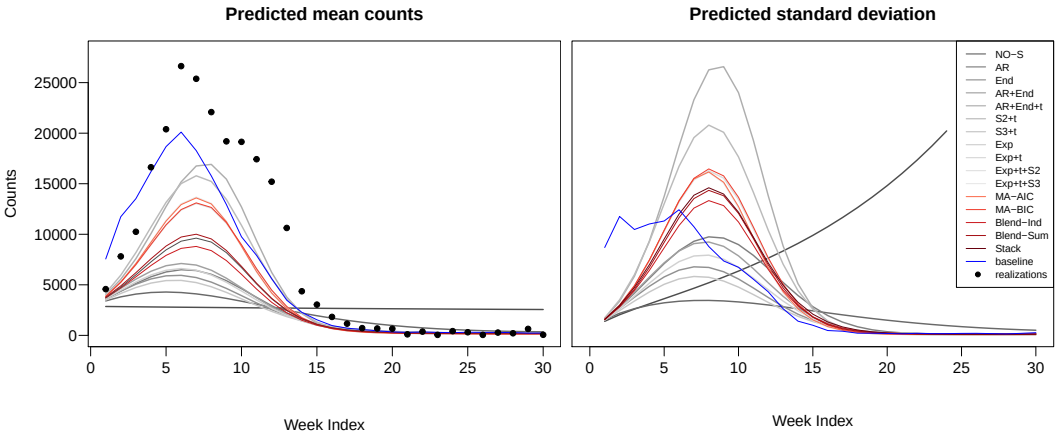

Figure 3.3: **Predicted mean counts and standard deviation** for the test year 2016 of the CHILI dataset for all 11 base models and 5 ensemble models.

fitted to the training data to then make predictions. We observe that the endemic component does not contribute much to the model compared to the autoregressive part for the CHILI dataset. We notice that the mean prediction for the test year is particularly low, lower than any year in the test data. This is likely due to a low conditioning value. This issue is discussed in more detail by Reeve (2017).
Figure 3.5 shows the predictive distribution of counts for the ensemble models (See Appendix A, Section A.4 for all base models). We observe that `MA-AIC` and `MA-BIC` seem to have mean predictions closer to actual observations but they also have higher variability as the prediction interval is wider.
Table 3.3 shows the sDSS and log(DS) for each model. We first notice that the ensemble models perform neither best nor worst. The best ensemble model is `Blend-Sum` whereas model averages perform worse. Sharper does not mean better in terms of sDSS. Actually the best model is one of the least sharp, which makes sense if since all models are "off target". We notice that our simple baseline model, although not sharp, performs surprisingly well. However, we would not draw too many conclusions on only one test data since the impact of the test year (high or low incidence year) probably plays an important role. In Chapter 5 we analyse more than one test year to assess the robustness of these results.

3.1.3 One-Step-Ahead Predictions

In this section we perform one-step-ahead predictions on the same test data (first 30 observation of year 2016).
Again, we start by looking at base model weights in ensemble models (Figure 3.6). We observe a similar pattern as for long-term forecasts, that is, model averages mainly favor one single model (`Exp+t+S3` for `MA-AIC` and `Exp+t+S2` for `MA-BIC`) whereas `Stack` shows a little more diversity. It is surprising that the addition of one observation to the stacking period induces such changes in weights composition but not impossible. This could be an indication that model `Stack` can adapt faster to new trends in the data.

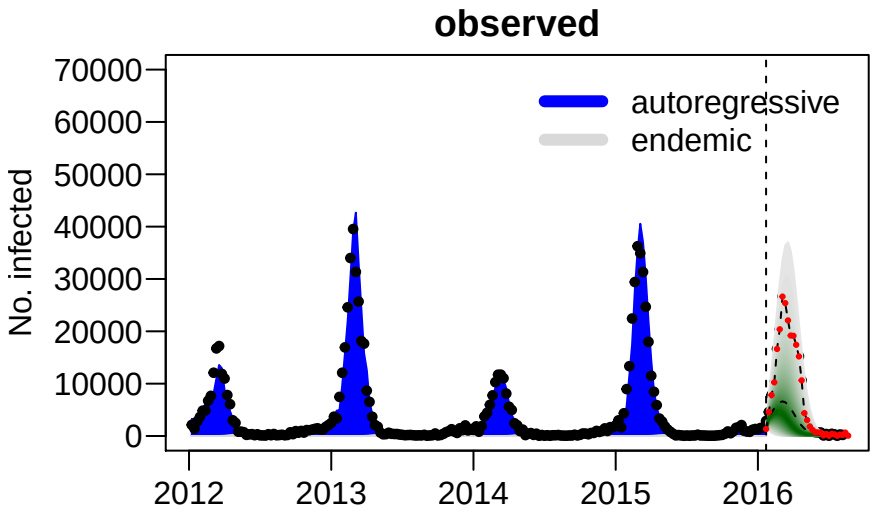

Figure 3.4: **Fitted values and predictions for base model `Exp+t+S3`.** Fitted values for the last 4 years of training data of the CHILI dataset and predicted values for the test data (year 2016).

Figure 3.7 shows the predicted mean counts and standard deviation for the test year 2016 of the CHILI dataset for all 11 base models and 5 ensemble models. Compared to long-term forecasts, the predictions are closer to the realizations which is expected, as we condition our forecast on $t-1$. We also notice that the variance is a lot smaller and increases with the value of the forecast which is normal behavior.

With one-step-ahead forecasts, we could evaluate the calibration of the forecast using probability integral transform (PIT). PIT methods for count data are described in Czado et al. (2009). In short, if the data generating process is the studied model, the PIT has a standard uniform distribution. U-shaped histograms indicate underdispersion, inverse-U shaped histograms point at overdispersion, and skewed histograms occur when central tendencies are biased. The PIT are shown in Appendix A Section A.5 but are not very informative since we only have a small sample of 30 forecasted timepoints.
Figure 3.8 shows the sDSS for each model in function of the week index of the test year. Globally the scores are very close to each other and we observe some severe fluctuations from week 20 onwards. We see in Figure 3.7 that the incidence between week 20 and 30 is very low and so is the associated standard deviation. The bad scores in this period could be due to wrong predictions which are heavily penalized by the small variance. However, this begs the question should these data points be included in the analysis or not? Let's remember that we chose a period of interest of 30 days so as to include the high season. In this context, the choice of a model should rely solely on its performance during periods of high incidence therefore discarding timepoints 20 to 30 would make sense. Moreover, Figure 2.5 in Chapter 2 shows

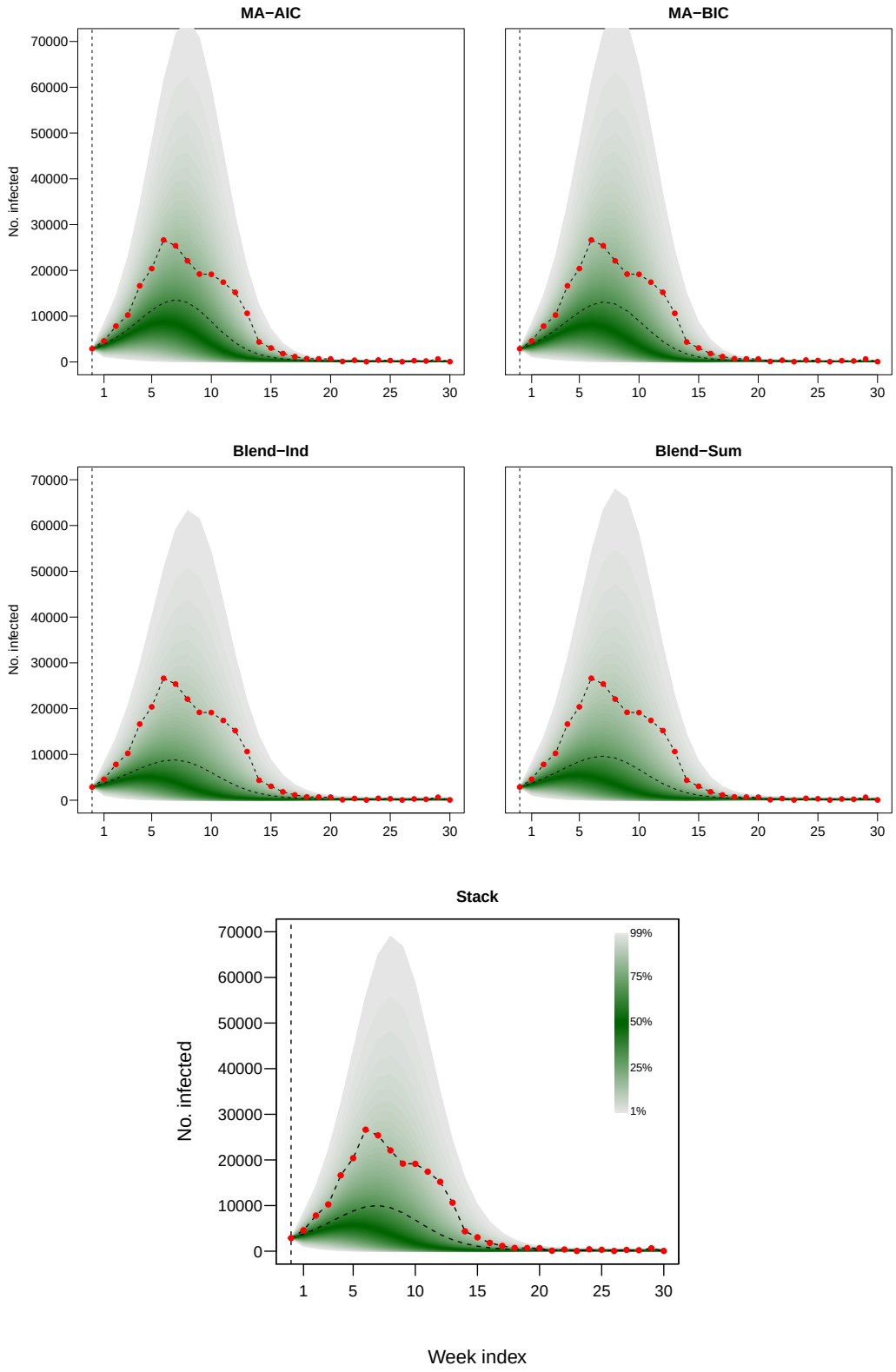

Figure 3.5: **Ensemble models predictions** for test year 2016 of the CHILI dataset.

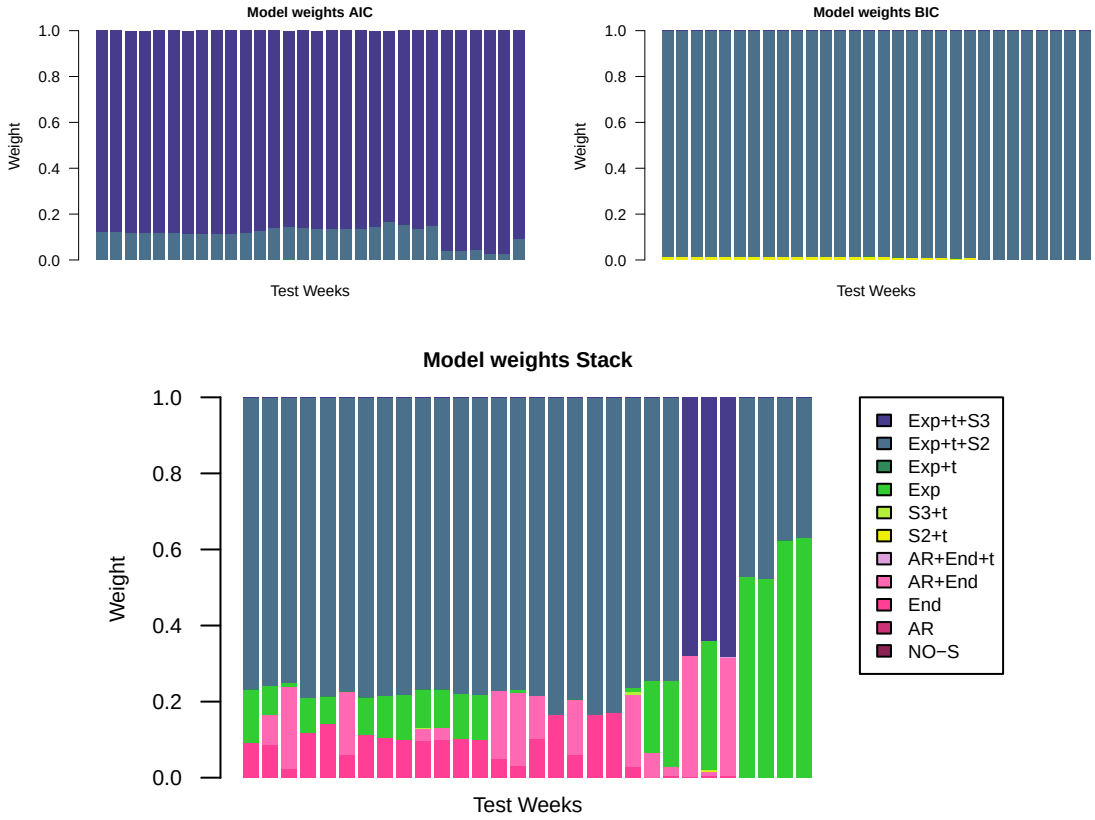

Figure 3.6: **Base Model weights** in ensemble models for one-step-ahead predictions.

Table 3.3: **DSS and log(DS) values.** sDSS and log(DS) values on test data are shown along with the ranks for log(DS) values. Models are ranked in order of increasing sDSS.

	End	AR+End	Baseline	Exp	Blend-Sum	Blend-Ind	AR	Stack
DSS	7.560	7.734	7.767	7.815	7.887	8.039	8.126	8.253
DS	6.596	6.385	7.044	6.317	6.503	6.468	6.899	6.509
DS rank	14	6	16	3	10	9	15	11

	AR+End+t	Exp+t	Exp+t+S2	MA-BIC	NO-S	S2+t	MA-AIC	Exp+t+S3
DSS	8.817	8.904	8.994	8.998	9.119	9.293	9.335	9.399
DS	6.288	6.223	6.402	6.407	8.431	6.582	6.346	6.335
DS rank	2	1	7	8	17	13	5	4

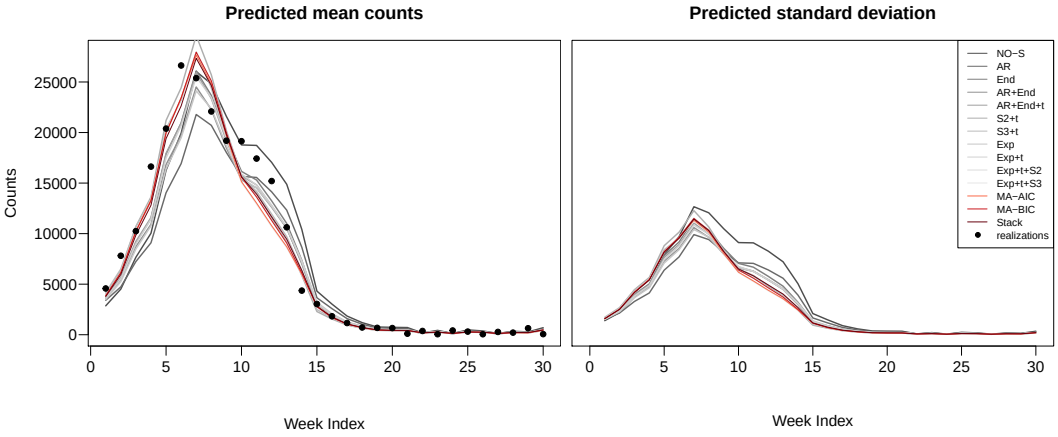

Figure 3.7: **Predicted mean counts and standard deviation** for one-step-ahead predictions for the test year 2016 of the CHILI dataset for all 11 base models and 5 ensemble models.

that the seasons are fairly regular with no high incidence in weeks 20 to 30 for the CHILI dataset.

Figure 3.9 shows the sDSS of all 30 weeks according to the model as well as the mean score for each model when considering 30 and 20 timepoints and the minimum mean score. We observe that the high scores are all linked meaning that in this case two specific weeks (weeks 24 and 29) are responsible for these bad results as was also observed in Figure 3.8. It is interesting to see that apart from these two weeks, all models have a fairly similar score distribution. If we look at the mean score of all 30 predictions, we notice some differences with models `End`, `AR+End` and `Exp` performing better than the others.

If we narrow down the set of predictions to 20 timepoints and calculate the mean, we observe that the differences in mean scores are less obvious.

Table 3.4 shows the mean sDSS when all 30 or only the first 20 timepoints are considered. The best performing models are different in both cases. With all 30 timepoints, the best model is `End` and the best ensemble model, `Stack` is ranked fourth. With only 20 timepoints, `Exp+t+S2` is the best model, the second best is

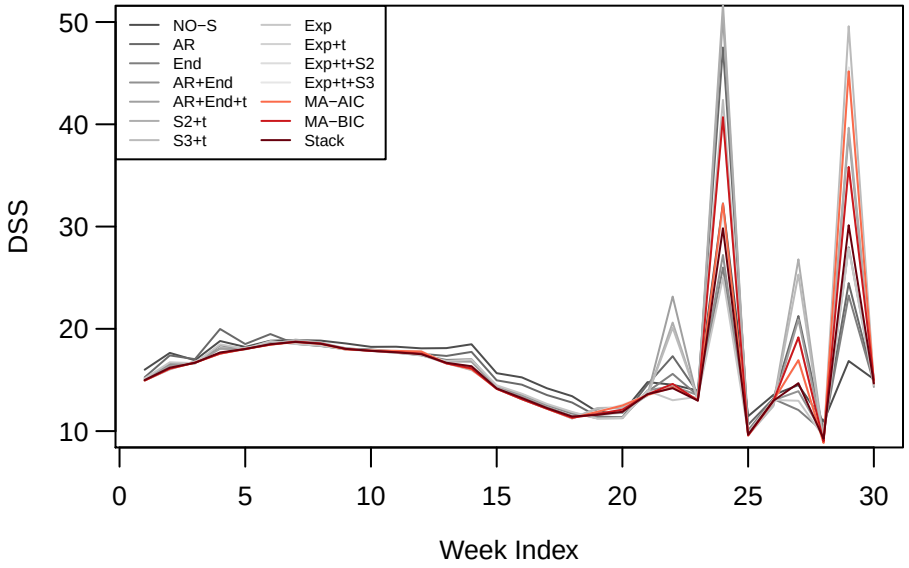

Figure 3.8: **DSS per Week** for 30 one-step-ahead predictions on test year 2016.

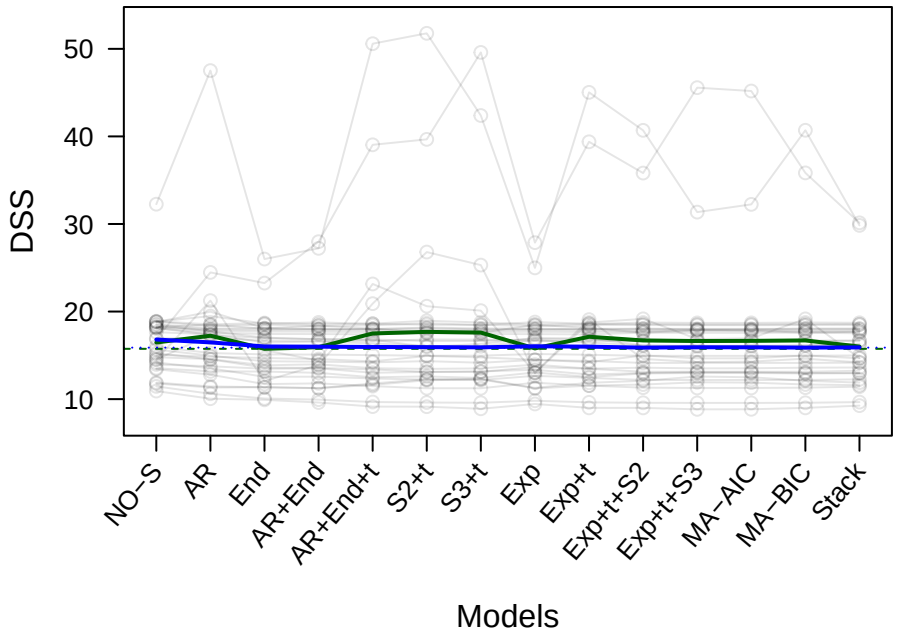

Figure 3.9: **DSS per Model** for 30 one-step-ahead predictions on test year 2016. The predictions for the same week are linked. The mean sDSS of 30 predictions is indicated by full green line and the minimum score is shown as a dashed green line. The mean sDSS of 20 predictions is indicated in full blue line and the minimum score is shown as a dashed line.

Table 3.4: **One-step-ahead prediction sDSS.** Scores when all 30 or only the first 20 timepoints are considered are shown along with the ranks for sDSS(20). Models are ranked in order of increasing sDSS(30).

	End	Exp	AR+End	Stack	NO-S	Exp+t+S3	MA-AIC
Mean sDSS (30)	15.744	15.790	15.931	16.001	16.467	16.636	16.653
Mean sDSS (20)	16.012	16.038	15.989	15.903	16.806	15.922	15.915
Rank Mean sDSS 20	11	12	10	3	14	6	5

	Exp+t+S2	MA-BIC	Exp+t	AR	AR+End+t	S3+t	S2+t
Mean sDSS (30)	16.699	16.701	17.123	17.238	17.499	17.600	17.680
Mean sDSS (20)	15.881	15.882	15.989	16.479	15.956	15.914	15.937
Rank Mean sDSS 20	1	2	9	13	8	4	7

ensemble model `MA-BIC` followed by `Stack`.
Figure 3.10 shows the distribution of the ranks over 30 predictions for the three ensemble models and `End`. Models `End`, `MA-AIC` and `MA-BIC` have a relatively flat distribution with fewer bad ranks for `End` and quite a few ranks, between 2 and 4, for `MA-AIC`.
`Stack` has a narrower distribution with fewer very good ranks but less very bad ranks too.

Conclusion This chapter showed how to analyse a univariate time series with ensemble model techniques. We have performed long-term and one-step-ahead forecasts of one test year (2016) on the CHILI dataset. We have seen that each ensemble model combines base models differently and that the composition of the pool of base models has a great influence on the weight composition. Moreover, we have observed that `Blend` performs relatively well as does `Stack` but not `MA-AIC` and `MA-BIC`. Ensemble models do not perform better than the best base model.
Regarding one-step-ahead predictions, we noticed that the results, including the performance of ensemble models, depend greatly on the number of timepoints considered in the test set.
It is very delicate to draw any general conclusions. The most obvious next step would be to repeat this analysis over more test years which will be covered in Chapter 5. We will also analyse the dengue dataset which is less regular. Moreover, we have seen that parameters such as the composition of the pool of base models and the length of the season considered can influence our finding. It is likely that ensemble hyperparameters such as the length of the stacking or blending period plays a role. We will also investigate this aspect in Chapter 5.
First however, it is critical that we make sure that the weight optimization process used in our functions to stack and blend is stable, i.e. that we indeed find the weights that optimize the sDSS in the stacking/blending periods. This will be discussed in the next chapter.

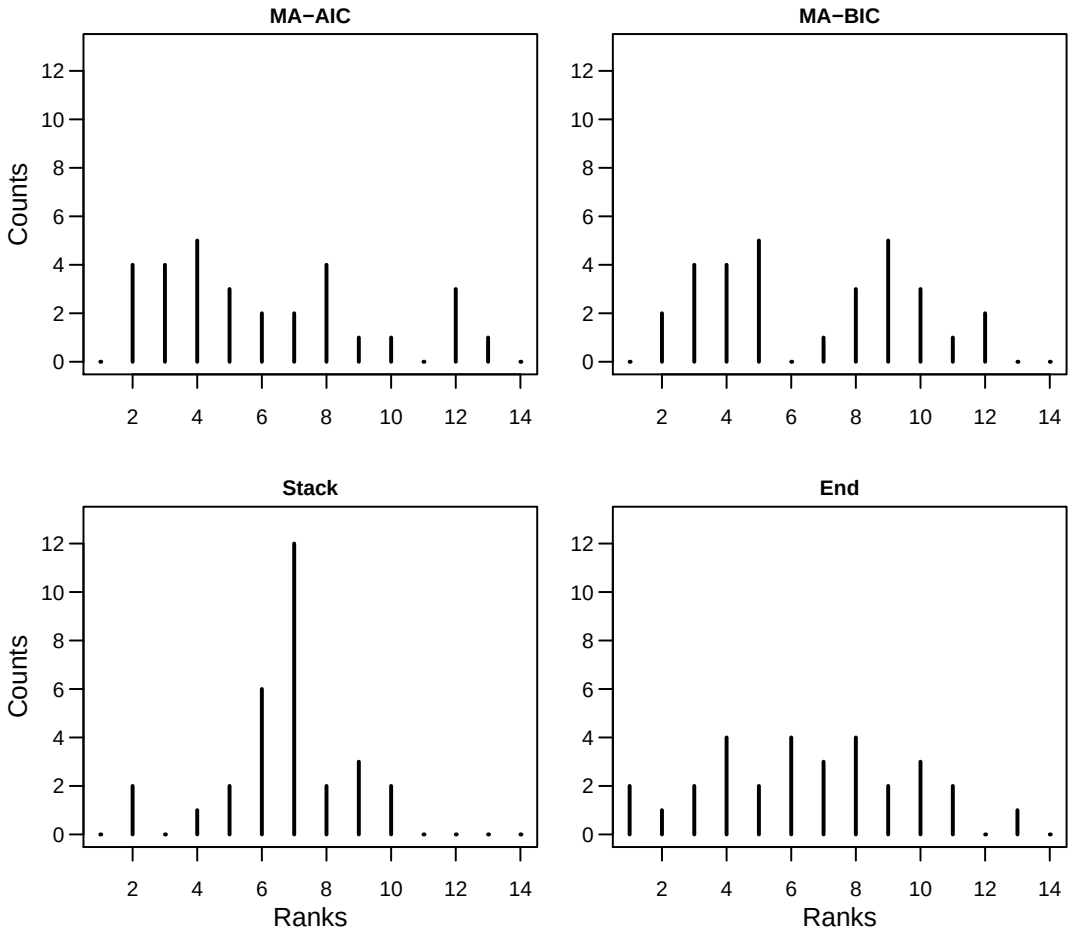

Figure 3.10: **Distribution of ranks.** The distribution of 30 performance ranks (measured by sDSS) is displayed for the three ensemble models and the best model `End`.

Chapter 4

Weight Optimization: Stability

In this chapter, we test the stability of the blending and stacking procedure i.e. the stability of the weight optimization process. We use the function `optimr` from the package **optimr** with default settings. This package is a replacement and extension of the best known `optim` function. According to its authors, all the optimizers included in the package are local minimizers, meaning that they attempt to find a local minimum. Global optimization is a much bigger problem, especially in non-convex spaces, but one that we must address if we want our functions to be stable and to be able to trust our results. This chapter will give a little more insight into the optimization code and will suggest some strategies to systematically obtain solutions as close as possible to the global solution without having to explore the entire search-space. These strategies will be tested on multiple years of the CHILI dataset.

4.1 The Optimizer

An optimizer aims to maximize or minimize an objective function. In our case, we aim to find the best combination of 11 different models as measured by the sDSS of the training data. The default settings of `optimr` cause an appropriate set of optimization methods to be automatically used depending on the presence or absence of bounds on the parameters. For more details on which methods are used under which conditions see the vignette and manual of this package which are available on `CRAN` at `https://cran.r-project.org/web/packages/optimr/index.html`.

As we are dealing with weights, we have two constraints on the initial set of parameters and on the solutions: (i) the weights cannot be negative and (ii) they have to sum up to 1.
Considering the non-quadratic nature of our function, imposing these two constraints is impossible with `optimr` and any other optimization package/function to the best of our knowledge. An elegant way around this is to feed real numbers to the optimizer and re-parametrize the weights inside the objective function as the output of the softmax function (see Equation 4.1). Let x_1, x_2, ..., x_J be any real numbers and w_1, w_2, ..., w_J denote the weights satisfying both conditions mentioned above

where J is the number of base models. We define

$$w_j = \exp(x_j)/(\sum_{j=1}^{J} \exp(x_j)). \tag{4.1}$$

As the weights have to sum up to 1, this means that we only need to solve for `nmod-1` parameters. We do this by setting $x_j = 0$ (the softmax function normalizes the weights).
The objective function is likely to be non-convex and include several local minima. Ideally, we would like to cover the whole search-space by performing the optimization with many different initial parameters and then pick the best solution (weight combination) to predict the test year. However, there is a trade-off between the number of initial combination tested and the time required.

In order to assess the quality and stability of the optimization, we repeat the optimization procedure for each test year 100 times, each time starting with a different set of random initial parameters. We also assess the influence of the number of initial parameters on the results by limiting the number of optimizations to respectively 10 and 50 runs. We also test another method consisting in setting the initial parameters in an informed way.

This is a summary of the three strategies:

1. `Run 10`: Randomly sample initial weights 10 times and pick the solution which performs the best on the training data.
2. `Run 50`: Similar but with 50 runs.
3. `Best base`: Set the initial parameters corresponding to using only the best base model, i.e. weight of the best base model close to 1 and weights of all other base models close to 0.

The last algorithm should run faster than the others since it only performs the optimization once. The first and second strategies should be more robust but slower.

We perform the stability analysis for the blending procedure (`Blend-Sum`). Any findings should be directly applicable to stacking. We now show in more detail the code for the function `Weightoptim` related to weight optimization.
The `Weightoptim` function has the following arguments:

- `listmodel.pred` is the list of base model predictions for the blending period.
- `OSA` is a length-one logical vector indicating whether we are performing a one-step-ahead analysis.
- `score.type` indicates which score should be minimized, in long-term predictions this argument is set to `'dss'`.
- If `total.stack` is TRUE, we are performing `Blend-Sum` otherwise `Blend-Ind`.
- `optim.type` can be set to `best.base` or `best.run` according to the strategy we use to optimize the weights.

- `nrun` indicates the number of times we perform the optimization when `optim.type` is set to `best.run`.
- `best.base.ind` is a number between 1 and J indicating the best performing base model when `optim.type` is set to `best.base` .

We also need to load the `optimr` library and source the function to optimize `MinDssSum`. Depending on `OSA` and `total.stack` the function to be minimized will change slightly (not shown). `nmod` is the equivalent to J and indicates the number of base models.

```
WeightOptim<- function(listmodel.pred,
                       OSA = FALSE,
                       score.type = NULL,
                       stack.length = NULL,
                       total.stack = NULL,
                       optim.type = NULL,
                       nrun= NULL,
                       seed= NULL,
                       best.base.ind = NULL){

  ############  inital settings
  library(optimr)
  source("functions/MinDssSum.R")
  nmod <- length(listmodel.pred)

  namemod <- vector()
  for (i in 1:nmod){
    namemod[i] <- paste ("model", i, sep="")
  }
```

In the next lines the initial parameters are set according to `optim.type`. Remember that we feed real numbers which will then be reparameterized to weights according to Equation 4.1.
In the `Best base` case we want to assign a weight close to 1 to the best model. This is achieved by setting the initial parameter x to 0 or 10 for the best model and -10 or 0 for the other models depending on whether the last model is the best model or not respectively.
In the `Run 10` or `Run 50` case we sample initial parameters from a uniform distribution between 0 and 1.
Whether these are the most appropriate initial conditions is debatable. For instance we could argue that values too close to 1 for `Best base` would be likely to get us stuck in a local minimum or that sampling only between 0 and 1 will seldom give heavy weights to one particular model. We are aware of these issues but consider this topic to be beyond the scope of this thesis and regard these conditions as acceptable.

```
  if(optim.type=="best.base"){
    if(best.base.ind == nmod){
      parini <- rep(-10, nmod-1)
    } else {
      parini <- rep(0, nmod-1)
      parini[best.base.ind] <- 10
```

```
    }
    parini.mat <- matrix(parini, nrow=1)
  }

  if(optim.type=="best.run"){
    set.seed(seed)
    parini.mat <- matrix(runif( n = nrun * (nmod - 1)), ncol = nmod-1)
  }
```

The following code chunk shows the `optimr` set-up within the `Stab` function and how we convert the output back to weights according to Equation 4.1.

```
  ########### optimr function
  Stab <- function(parini){
    x.dss.r <- optimr(par = parini,
                      fn = MinDssSum,
                      listmodel.pred = listmodel.pred,
                      nmod = nmod,
                      univar=univar,
                      score.type = score.type)

    # convert x to weights
    x <- x.dss.r$par # extract the parameters
    xfull <- c(x, 0) # we are missing the last value which was set to 0.
    w.dss.r <- exp(xfull)/(sum(exp(xfull))) # convert to weights
    dssr <- x.dss.r$value # extract the minimized score
    names(w.dss.r) <- namemod

    listw <- list(score.optim = dssr,
                  weights = vec.r)
    return(listw)
  }
```

short, the function `MinDssSum` converts the parameters x to weights, combines the base models according to those weights with `MAMeanCov`, and then calculates the score with `Dssval` (which is an extension of the `ds_score_hhh4` from the **surveillance** package).

```
MinDssSum <- function(x, listmodel.pred,
                      nmod) {

  source("functions/Dssval.R")
  source("functions/MAMeanCov.R")

  # Empty list and vector
  pred <- list()
  vec.score <- vector()

  ### calculate Score for each blending period.
  xfull <- c(x, 0)
  weights <- exp(xfull)/(sum(exp(xfull)))

  for (i in 1: length(listmodel.pred)){
    ### Calculate MA mean and covariance
```

```
    meancov.av <- MAMeanCov(listmodel.pred[[i]], weights)
    pred$mu_vector <- meancov.av[["MAmean"]]
    pred$Sigma <-  meancov.av[["MAcov"]]

    vec.score[i]<- Dssval(pred = pred,
                          realizations = listmodel.pred[[i]][[1]]$realizations,
                          detailed = FALSE,
                          scaled = FALSE,
                          univar= univar)
  }
  return(sum(vec.score))# not scaled
}
```

We now apply the `Stab` function to the matrix containing the single or multiple initial parameters. In `Run 10` or `Run 50` we then extract the weights that gave the best solution.

```
  ############ optimr execution

  listw <- apply(parini.mat, 1, FUN = Stab)
  w <- t(sapply(listw, FUN = function(x) x$weights))
  s <- t(sapply(listw, FUN = function(x) x$score.optim))

  list.return <- list(weights = w,
                      score.optim = s)

  if(optim.type=="best.run"){
    best.index<- which.min(s)
    list.return <- list(weights = w[best.index, ],
                        score.optim = s[best.index])
    }
  return(list.return)
}
```

4.2 Stability Analysis

We perform blending five times, using one of the last 5 years of the CHILI dataset (2012-2016) for test data each time. As previously, we set the blending period to 4 years and adapt the number of fitting years accordingly so as to always use all available data (see Figure 4.1).
Figure 4.2 shows the weights and score distribution for all 100 runs for blending period 2008-2011 ordered by best sDSS in the training data. We can observe some clusters of identical scores and weight distributions but also many different combinations and results. Interestingly, it seems that some identical weight combinations result in different training scores. For example the "green" combination where base model `Exp` has a weight close to 1 and appears in the 8.02 and 8.05 score sections. The reason is that the difference in weight combination is so small that it is not visible by eye but still impacts the scores.
Figure 4.3 shows the sDSS for the test year according to the sDSS in training data.

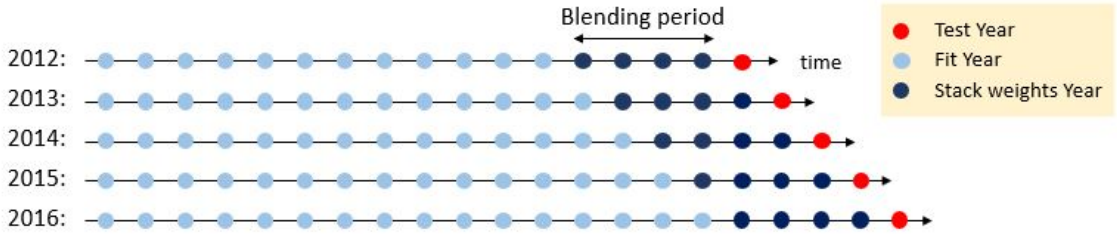

Figure 4.1: **Stability analysis representation.** We use the last 5 years of the CHILI dataset as test years (2012-2016) and adapt the number of training years so as to always use all available data and 4 years as blending period.

Figure 4.3 panel A shows the results for the 100 runs, the three optimized algorithms and the base models. Figure 4.3 panel B shows an enlargement of the bottom left box in Figure 4.3 panel A.
First, we observe that the results from the base models are a lot more scattered than the results from the blending procedure regarding both scores from the training and testing data. Second, the results from blending are located in the bottom left area of the plot indicating both better training and testing scores than other base models. In this case there is a correlation between the scores in the training data and those in the test data which is encouraging because it justifies the use of blending to improve the performance of the predictions.
Looking at Figure 4.3B we observe that the correlation between the results in the training data and test data is less obvious but this can be expected since the difference in scores is reduced. `Run 10` and `Run 50` yield the same results indicating that 10 runs may be sufficient. Regarding the training scores, `Run 10` and `Run 50` perform better than `Best base` although all three algorithms perform relatively well when considering all 100 runs and base models. `Best base` performs better than the best model which is a good indication that the optimization is working well as we would expect `Best base` to perform equally well or better than the best base model.

Figure 4.4 shows the weight distribution and sDSS for four blending periods. As we can see, blending period 2010-2013 has a simpler search-space with three major combinations, two of them resulting in identical scores. However, this seems to be an exception rather than the rule.
We could be concerned by the multiplicity of the model combinations observed when changing initial parameter values. In our opinion, this reflects the fact that many combinations achieve good training scores but could also arise due to the fact that the models are quite similar to each other by nature meaning that replacing one model by another in the ensemble does not change the predictions fundamentally. In our opinion this could be of concern if the resulting training and testing scores were very different from one another. However, Figure 4.5 shows that the distribution of the blending scores for training and testing are systematically narrower than that of the base models. Moreover the majority of the scores in the training data are

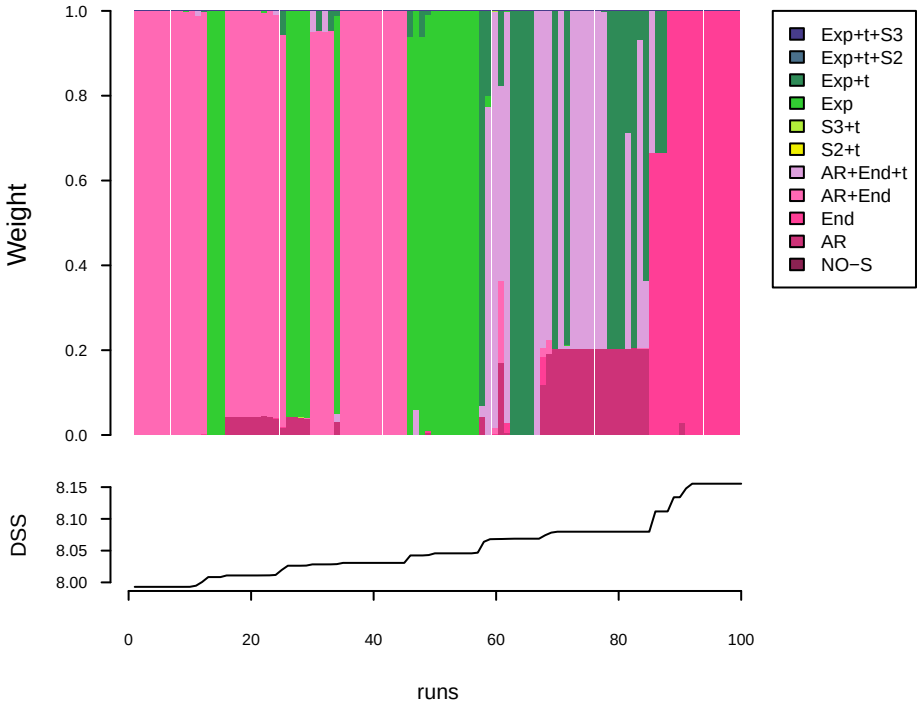

Figure 4.2: **Weights and sDSS for blending period 2008-2011.** Weights and sDSS are represented for 100 optimization runs on training years 2008-2011. They are ordered by increasing DSS. These are the possible weight combinations coming from blending to predict year 2012.

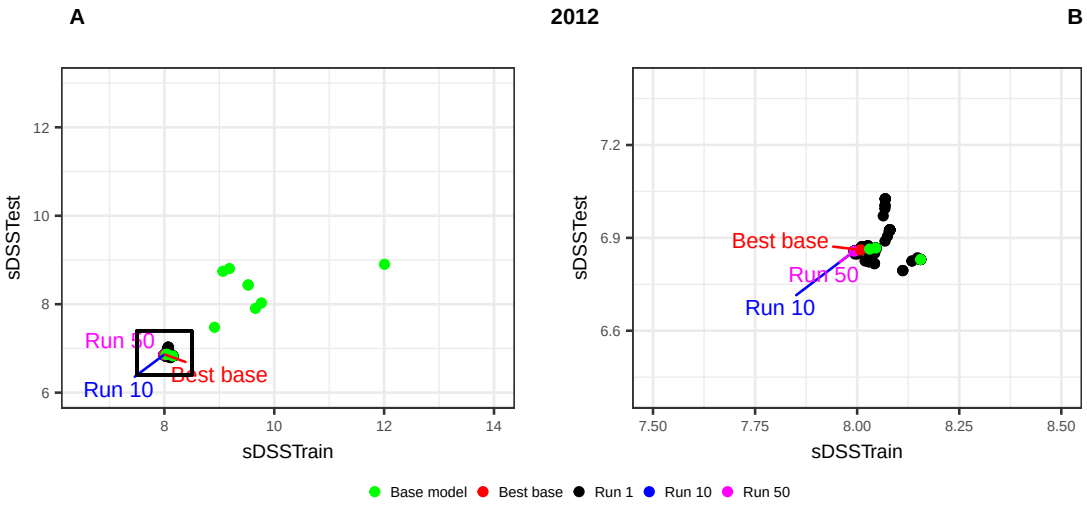

Figure 4.3: **Correlation between train and test scores for blending period 2008-2011, test year 2012. A** shows the sDSS in test data in function of the sDSS in training data for 100 runs, the 3 algorithms and base models. **B** shows the same relation for an enlargement of the boxed area in A.

better or as good as the best base model and more importantly this is true for the three algorithms tested. However, this does not automatically translate to the best score for the test data. The variation between the test score for the best and worst blending solution varies according to the tested year and is particularly bad for test year 2016.
The results of the three tested algorithms are always very similar with no clear best option in terms of test score. However, `Best base` is much faster compared to `Run 10` and `Run 50`, the full algorithms taking on average 50, 140 and 600 seconds to run respectively.

Conclusion Even though the blending procedure - and by extension the stacking procedure - are not completely stable due to the non-convex search-space spanned by 11 base models, the results of our stability test show that by using one of the three strategies to deal with instability we obtain, on average, good results. We constantly obtain the best optimized value in the training data which often translates to better scores in the test data and justifies the use of this procedure.
Setting initial optimization parameters by finding the best base models and assigning a weight close to 1 to the respective initial parameter as we did in the `Best base` procedure is the fastest option and the one chosen to run all the analyses in this thesis.

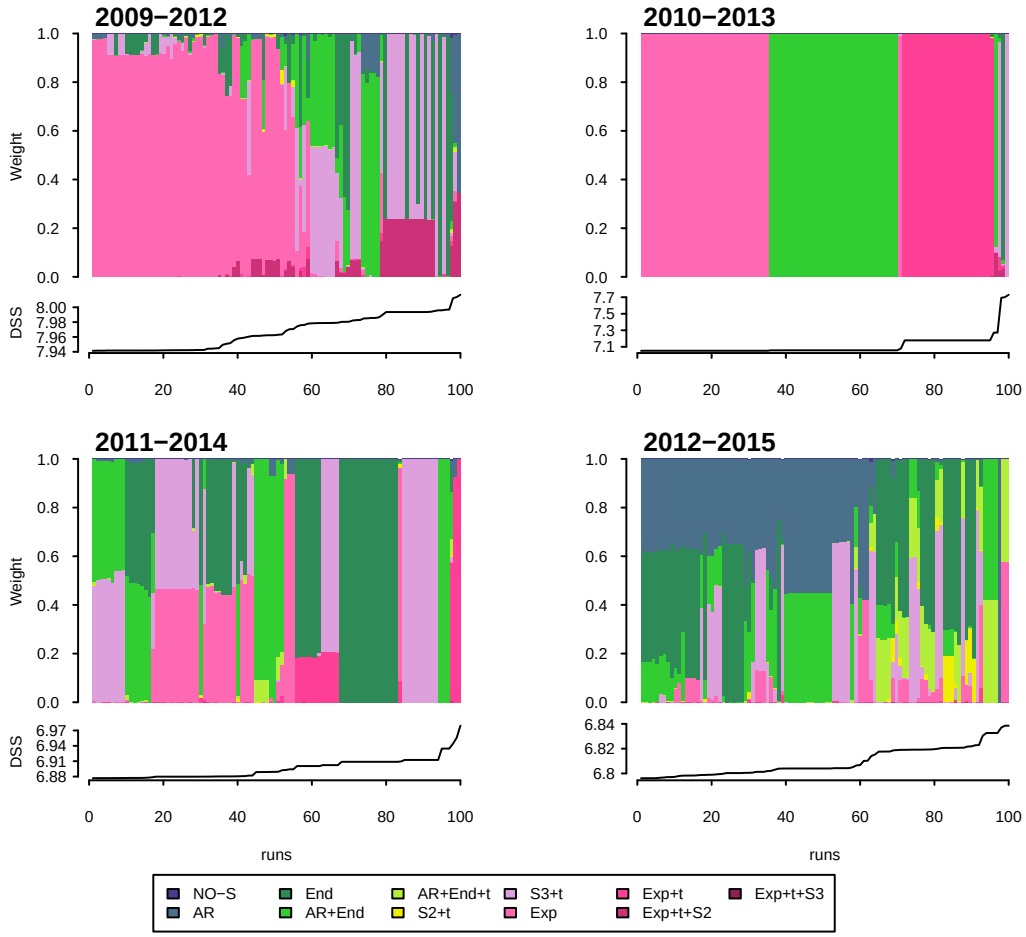

Figure 4.4: **Weights and sDSS for different blending periods.** Weights and sDSS are represented for 100 optimization runs on training years 2009-2012, 2010-2013, 2011-2014 and 2012-2015. They are ordered by increasing DSS. These are the possible weight combinations coming from blending to predict years 2013, 2014, 2015 and 2016 respectively.

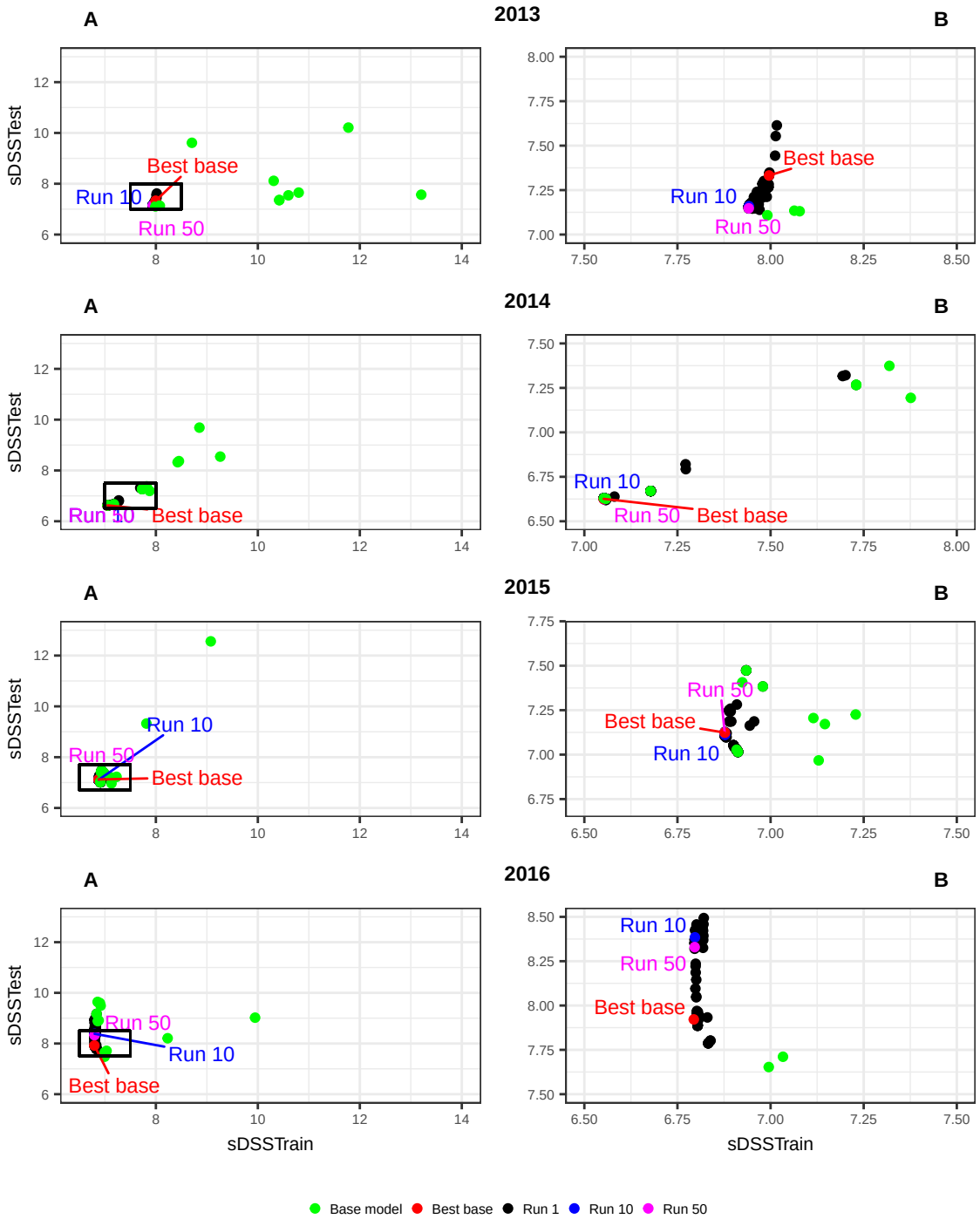

Figure 4.5: **Correlation between train and test scores for test years 2013-2016.** Respective panels **A** shows the sDSS in test data in function of the sDSS in training data for 100 runs, the 3 algorithms and base models. Respective panels **B** shows the same relation for an enlargement of the boxed area in A.

Chapter 5

Reproducibility & Hyper Parameter Tuning

In this chapter, we extend the long-term analysis done previously on the CHILI dataset to five test years (2011-2016). We do this in the same manner as in Chapter 4 by adding an extra year to the training data for each additional test year. We also perform the same analysis on the Dengue dataset. Finally, we modify hyperparameters of ensemble models such as the length of the blending and stacking period to assess the impact of this "fine-tuning" on our results.

Globally, this chapter helps us understand how confident we can be when assessing model performances.

5.1 CHILI: Five Test Years (2012-2016)

As a reminder, results for year 2016 are identical to those of Chapter 3. We do not aim to do an in depth analysis of each individual test year, but rather to explore possible general trends amongst the results and to consider the global performance of each model.
We start by looking at the weights of the base models composing `Blend-Sum` and `Stacking` for long-term predictions. Table 5.1 shows that base models `NO-S` and `AR` are never included and could be removed from the pool of base models to make the process faster. Weights vary according to the year tested with no clear patterns.

Table 5.2 shows the score for each model including `Baseline` for each year along with the mean over five years (2012-2016) and the mean over four years (2012-2015). We also included three more "Baseline results": `Best.AIC`, `Best.BIC` and `Best.Stack`. They show the sDSS of the base model with respectively the best AIC, BIC or sDSS score [1] on the training data. The idea behind this is that a possible strategy when making predictions would be to always consider the model that performs the best on the training data. The performance can be evaluated by these three criteria.
All years considered, we observe that two "simple" models `End` and `AR+End` perform best followed by the best ensemble model `Blend-Sum`. The second best ensemble model is `Stack` followed by `MA-BIC` and `MA-AIC`. It is also worth noting that nearly

[1] as calculated in the stacking procedure

all models perform on average better than the baseline model, which is reassuring. Both simple models perform particularly well in 2016. we wondered whether the results would be greatly changed if we repeated the analysis excluding year 2016. Indeed, the best model is then ensemble model `MA-BIC`. The fact that an ensemble model can perform better than base models on average is encouraging, however, one must be very cautious when interpreting results of selected data. For instance, ensemble model `Stack` goes from rank 5 to 12. Thus, we cannot say that 2016 was a "bad" year for all ensemble models.

If we compare the results of the ensemble models to `Best.AIC`, `Best.BIC` and `Best.Stack`, we observe that performance of the ensemble models (except `MA-AIC`) are very encouraging as they are ranked 6^{th}, 5^{th} and 3^{rd} as opposed to 13^{th}, 11^{th} and 9^{th}. The results are completely different if we do not consider the results of test year 2016.

These results are a little disappointing as we had hoped that ensemble models could improve forecasts more systematically. The fact that removing a year from the dataset causes such changes in the ranks is disturbing. A solution to this would be to study more test years and thus longer time series or average the results over many different time series.

The second important factor is that comparing the results with sDSS in general is complicated. The scores allow us to easily rank the models but deciding whether differences in scores are actually relevant is problematic. A method to do so would be helpful. It is thus not straightforward to evaluate whether the difference in score between the models ranking first and last is important or not.

Moreover, it is likely that some other parameters play a role in ensemble model performances such as the values of the hyperparameters for `Stack` and `Blend` as we discuss later in this chapter. First, though, we were curious to see whether we would obtain different results on another, less regular dataset: Dengue.

Table 5.1: CHILI: base model weights in ensemble models over test periods 2012-2016. Yearly maxima are highlighted in bold.

	MA-AIC					MA-BIC				
	2012	2013	2014	2015	2016	2012	2013	2014	2015	2016
NO-S	0	0	0	0	0	0	0	0	0	0
AR	0	0	0	0	0	0	0	0	0	0
End	0	0	0	0	0	**0.58**	**0.69**	0.04	0	0
AR+End	0	0	0	0	0	0.14	0.13	0.03	0	0
AR+End+t	0	0	0	0	0	0.04	0.01	0	0	0
S2+t	0.06	0.01	0.02	0	0	0.16	0.05	0.28	0.03	0.01
S3+t	0.35	0.04	0.06	0	0	0	0	0	0	0
Exp	0	0	0	0	0	0.01	0.01	0	0	0
Exp+t	0	0	0	0	0	0	0	0	0	0
Exp+t+S2	0.21	0.22	**0.54**	**0.51**	0.12	0.06	0.12	**0.64**	**0.97**	**0.99**
Exp+t+S3	**0.37**	**0.73**	0.37	0.48	**0.88**	0	0	0	0	0

	Blend					Stack				
	2012	2013	2014	2015	2016	2012	2013	2014	2015	2016
NO-S	0	0	0	0	0	0	0	0	0	0
AR	0	0	0	0	0	0	0	0	0	0
End	0	0	0	0	0	0	0.2	0.2	0.2	0.2
AR+End	**1**	**0.92**	**1**	0	0	**0.25**	0.25	**0.25**	0	0
AR+End+t	0	0.07	0	0	0	0	0	0	0	0.01
S2+t	0	0	0	0	0	**0.25**	**0.29**	0.05	0.05	0.05
S3+t	0	0	0	0.01	0	0	0	0.11	0.11	0.11
Exp	0	0	0	**0.49**	0.45	**0.25**	0.25	**0.25**	0.25	0
Exp+t	0	0	0	**0.49**	0	**0.25**	0	0.14	**0.32**	**0.33**
Exp+t+S2	0	0	0	0	**0.55**	0	0	0	0.07	0.07
Exp+t+S3	0	0	0	0	0	0	0	0	0	0.23

5.2 Dengue: Five Test Years (2008/2009 - 2012/2013)

We perform a similar analysis on the Dengue dataset. We use the last five years as test data, increasing the training data accordingly. The blending and stacking periods are set to 4 years as previously. As the dengue dataset is 23 years long versus 17 years long for the CHILI dataset, the number of fitting years is greater. Note that a forecast year overlaps over two historical year. To make tables clearer, test years are only indicated by the last historical year. For example test year 2008/2009 will be indicated as 2009.
We start by looking at the weights of the base models composing ensemble models. This time Table 5.3 shows that base models `S2+t` and `S3+t` are never included. Globally, unlike for CHILI, seasonality does not seems to play an important role in the modelling of the Dengue dataset, exponentially decaying lags on the other hand appear essential.

Table 5.4 shows the score for each model including `Baseline`, `Best.AIC`, `Best.BIC` and `Best.Stack` for each year along with the mean over five years and excluding 2009/2010.
To our surprise, we observe that, all years considered, the simplest model `NO-S` performs best, followed by ensemble model `Stack`. 2009/2010 is particular in the sense that the conditioning observation is high but the peak is not. Most models thus overestimated the future disease incidence giving an advantage to model `NO-S` as its estimations are consistently low.
If we take that year out, ensemble model `Stack` performs best. Again we must be very cautious with such interpretations but we will note that this time the rank of ensemble models is not greatly disrupted by this modification. This could be an indication that ensemble models are beneficial in more unpredictable environments but again, we remain cautious. More data would be necessary to confirm this hypothesis.
If we compare ensemble models, we notice again that `MA-AIC` does not perform very well. Compared to `Best.AIC`, `Best.BIC` and `Best.Stack`, ensemble models (except `MA-AIC`) perform better, ranking at the 8^{th}, 5^{th} and 2^{nd} places as opposed to the 12^{th}, 6^{th} and 15^{th}. Again, these results are very different if we exclude year 2010 from the analysis.
In addition we note that the `Baseline` model performs very poorly. Indeed it shows a disastrous score for test year 2011. This is due to the fact that disease incidence in 2011 started exceptionally high, and that it was on average a year with a high number of cases. HHH4 models pick this up with the conditioning value but not the baseline model.
The same limitations as for the CHILI dataset apply here namely the need of more data and the issue when it comes to comparing absolute score values.

Table 5.2: CHILI: sDSS over test periods 2012-2016. Yearly minima are highlighted in bold (`Best.AIC`, `Best.BIC` and `Best.Stack` not included).

						Mean		Rank	
	2012	2013	2014	2015	2016	All	-2016	All	-2016
NO-S	8.67	9.97	8.52	12.16	9.12	9.69	9.83	19	19
AR	7.41	9.08	7.07	9.18	8.13	8.17	8.19	18	18
End	6.84	7.1	6.66	7.25	**7.56**	**7.08**	6.96	**1**	15
AR+End	6.91	7.05	6.6	7.26	7.73	7.11	6.96	2	14
AR+End+t	6.95	6.9	6.58	7.07	8.82	7.26	6.87	8	7
S2+t	7	7	6.67	6.94	9.29	7.38	6.9	15	10
S3+t	7.03	**6.88**	6.71	6.89	9.55	7.41	6.88	16	8
Exp	6.94	7.09	6.59	7.37	7.82	7.16	7	4	16
Exp+t	6.97	6.92	**6.57**	7.13	8.9	7.3	6.9	10	9
Exp+t+S2	6.92	6.94	6.59	6.87	8.99	7.26	6.83	7	2
Exp+t+S3	6.93	6.93	6.63	**6.86**	9.4	7.35	6.84	14	4
MA-AIC	6.96	6.93	6.61	**6.86**	9.33	7.34	6.84	12	5
MA-BIC	**6.78**	7.04	6.61	6.87	9	7.26	**6.82**	6	**1**
Blend-Sum	6.86	7.16	6.63	7.12	7.89	7.13	6.94	3	13
Stack	6.81	7.3	6.58	7.02	8.25	7.19	6.93	5	12
Baseline	7.41	7.42	7.4	7.72	7.77	7.54	7.49	17	17
Best.AIC	6.93	6.93	6.59	6.87	9.4	7.34	6.83	13	3
Best.BIC	6.84	7.1	6.59	6.87	8.99	7.28	6.85	9	6
Best.Stack	6.91	7	6.6	7.13	8.9	7.31	6.91	11	11

Table 5.3: Dengue: Base model weights in ensemble models. Yearly maxima are highlighted in bold.

	MA-AIC					MA-BIC				
	2009	2010	2011	2012	2013	2009	2010	2011	2012	2013
NO-S	0	0	0	0	0	0	0	0	0	0
AR	0	0	0	0	0	0	0	0	0	0
End	0	0	0	0	0	0	0	0	0	0
AR+End	0	0	0	0	0	0	0	0	0	0
AR+End+t	0	0	0	0	0	0	0	0	0	0
S2+t	0	0	0	0	0	0	0	0	0	0
S3+t	0	0	0	0	0	0	0	0	0	0
Exp	0	0.02	0.01	0.04	0.03	0.07	**0.85**	**0.86**	**0.95**	**0.97**
Exp+t	**0.63**	**0.52**	0.24	0.27	0.14	**0.93**	0.15	0.14	0.05	0.03
Exp+t+S2	0.16	0.11	**0.44**	**0.45**	**0.56**	0	0	0	0	0
Exp+t+S3	0.2	0.35	0.3	0.24	0.27	0	0	0	0	0

	Blend					Stack				
	2009	2010	2011	2012	2013	2009	2010	2011	2012	2013
NO-S	0	0	0	0.28	**0.44**	0.02	0.07	0.3	**0.27**	**0.27**
AR	0	0	0.09	0	0	0	0	0	0	0
End	0	0	0.03	0	0	**0.25**	0.25	0.25	0.25	0
AR+End	**0.79**	**0.57**	0.19	0.13	0	**0.25**	0.25	0	0.07	0.07
AR+End+t	0	0	0	0	0	0	0	0	0	0
S2+t	0	0	0	0	0	0	0	0	0	0
S3+t	0	0	0	0	0	0	0	0	0	0
Exp	0	0.43	**0.69**	**0.59**	0.37	0.23	**0.29**	**0.31**	**0.27**	**0.27**
Exp+t	0.21	0	0	0	0.19	**0.25**	0	0	0	0
Exp+t+S2	0	0	0	0	0	0	0.14	0.14	0.14	0.14
Exp+t+S3	0	0	0	0	0	0	0	0	0	0.25

5.3 Fine-Tuning Influence

In this section, we modify the hyperparameters "length of blending period" and "length of stacking period" to assess their impact on the respective ensemble models. We study the last five years of the CHILI dataset and vary this parameter from 1 to 6 years adapting the fitting period accordingly. "Fine-tuned" models are named `B-1` to `B-6` and `S-1` to `S-6` for blending and stacking respectively. Note that in this context, `B-1` and `S-1` are the same model.
Table 5.5 and 5.6 show the weight of base models according to the fine-tuned model. We can say that in general, the fine-tuning impacts blending weight more than stacking weights, which could be explained by the "averaged" nature of stacking weights compared to the "sum" of blending.
This tendency is confirmed by sDSS results in Tables 5.8 and 5.7. There is more variance in the scores between the different fine-tuned models in blending than stacking. Indeed, the difference between the best and worse score for blending and stacking is 0.22, 0.17, 0.08, 0.57, 1.26 and 0.14, 0.27, 0.06, 0.21, 1.18 for years 2012 to 2016 respectively.

This is very troubling because it is of the same order as the differences between models.

Table 5.4: Dengue: sDSS over test periods 2008-2012. Yearly minima are highlighted in bold (`Best.AIC`, `Best.BIC` and `Best.Stack` not included).

						Mean		Rank	
	2009	2010	2011	2012	2013	All	-2010	All	-2010
NO-S	2.47	**3.02**	3.86	2.61	4.04	**3.2**	3.25	**1**	8
AR	2.45	3.12	4.36	2.57	3.9	3.28	3.32	4	14
End	2.6	3.84	4.09	2.59	**3.57**	3.34	3.21	7	5
AR+End	2.57	4.06	3.71	2.49	4.26	3.42	3.26	9	9
AR+End+t	2.91	6.79	3.77	2.41	5.52	4.28	3.65	16	16
S2+t	2.87	6.79	4.55	2.42	5.48	4.42	3.83	17	18
S3+t	2.89	7.01	4.31	2.41	5.53	4.43	3.78	18	17
Exp	**2.39**	3.64	**3.64**	2.37	4.31	3.27	3.18	3	2
Exp+t	3.13	5.73	3.8	2.33	4.45	3.89	3.43	14	15
Exp+t+S2	2.85	6.03	3.91	2.32	3.99	3.82	3.27	13	11
Exp+t+S3	2.88	5.98	3.81	**2.31**	3.83	3.76	3.21	11	4
MA-AIC	2.98	5.6	3.83	2.32	3.95	3.73	3.27	10	12
MA-BIC	2.79	3.74	3.66	2.36	4.31	3.37	3.28	8	13
Blend-Sum	2.49	3.64	3.93	2.46	4.09	3.32	3.24	5	7
Stack	2.48	3.58	3.74	2.52	3.81	3.23	**3.14**	2	**1**
Baseline	3.74	4.23	63.63	5.23	10.29	17.43	20.73	19	19
Best.AIC	2.88	5.98	3.91	2.32	3.83	3.78	3.23	12	6
Best.BIC	2.6	3.84	3.91	2.32	3.99	3.33	3.2	6	3
Best.Stack	2.57	6.79	3.71	2.33	4.45	3.97	3.26	15	10

Table 5.5: Fine-tuning CHILI: base model weights in variations of ensemble model `Blend`. Yearly maxima are highlighted in bold.

	2012						2013						2014					
	B-1	B-2	B-3	B-4	B-5	B-6	B-1	B-2	B-3	B-4	B-5	B-6	B-1	B-2	B-3	B-4	B-5	B-6
NO-S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
End	0	0	0	0	0	0.01	**0.8**	0	0	0	0	0	0	0	0	0	0	0
AR+End	0	**1**	**1**	**1**	0	0.22	0	**0.92**	**1**	**0.92**	**1**	0	0	0	0	**1**	**0.98**	**1**
AR+End+t	0	0	0	0	0	0	0	0	0	0.07	0	0	0	**0.86**	**0.51**	0	0	0
S2+t	0	0	0	0	0	0.2	0.18	0	0	0	0	0.01	0	0.14	0	0	0	0
S3+t	0	0	0	0	0	0	0	0.08	0	0	0	0	0.42	0	0	0	0	0
Exp	**1**	0	0	0	**1**	0	0	0	0	0	0	**0.99**	0	0	0.48	0	0	0
Exp+t	0	0	0	0	0	0	0	0	0	0	0	0	**0.58**	0	0	0	0	0
Exp+t+S2	0	0	0	0	0	**0.57**	0.01	0	0	0	0	0	0	0	0.01	0	0.02	0
Exp+t+S3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	2015						2016					
	B-1	B-2	B-3	B-4	B-5	B-6	B-1	B-2	B-3	B-4	B-5	B-6
NO-S	0	0	0	0	0	0	0	0	0	0	0	0
AR	0.01	0	0	0	0	0	0	0	0	0	0	0
End	0	0	0	0	0	0	0	0	0	0	0	0
AR+End	0	0	0.01	0	**1**	**0.98**	0	0	0	0	0	**1**
AR+End+t	0	**0.86**	**0.66**	0	0	0	0.04	0	0	0	0	0
S2+t	0	0	0.09	0	0	0	0	0	0	0	0	0
S3+t	0	0	0	0.01	0	0	0	0	0	0	0	0
Exp	0	0	0.24	**0.49**	0	0	0	0	0.14	0.45	0.24	0
Exp+t	**0.71**	0	0	**0.49**	0	0	0.04	0.26	0.23	0	**0.76**	0
Exp+t+S2	0.27	0.14	0	0	0	0.02	0	**0.74**	0.12	**0.55**	0	0
Exp+t+S3	0	0	0	0	0	0	**0.9**	0	**0.51**	0	0	0

Table 5.6: Fine-tuning CHILI: base model weights in variations of ensemble model `Stack`. Yearly maxima are highlighted in bold.

	2012						2013						2014					
	S-1	S-2	S-3	S-4	S-5	S-6	S-1	S-2	S-3	S-4	S-5	S-6	S-1	S-2	S-3	S-4	S-5	S-6
NO-S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
End	0	0	0	0	0	0.13	**0.8**	0.4	0.27	0.2	0.16	0.13	0	**0.4**	0.27	0.2	0.16	0.13
AR+End	0	**0.5**	**0.33**	**0.25**	**0.2**	**0.17**	0	0	**0.33**	0.25	0.2	0.17	0	0	0	**0.25**	0.2	0.17
AR+End+t	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S2+t	0	0	**0.33**	**0.25**	**0.2**	**0.17**	0.18	0.09	0.06	**0.29**	**0.24**	**0.2**	0	0.09	0.06	0.05	**0.24**	0.2
S3+t	0	0	0	0	0	0.03	0	0	0	0	0	0	0.42	0.21	0.14	0.11	0.08	0.07
Exp	**1**	**0.5**	**0.33**	**0.25**	**0.2**	**0.17**	0	**0.5**	**0.33**	0.25	0.2	0.17	0	0	**0.33**	**0.25**	0.2	0.17
Exp+t	0	0	0	**0.25**	**0.2**	**0.17**	0	0	0	0	0.2	0.17	**0.58**	0.29	0.19	0.14	0.12	**0.26**
Exp+t+S2	0	0	0	0	**0.2**	**0.17**	0.01	0.01	0	0	0	0.17	0	0.01	0	0	0	0
Exp+t+S3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	2015						2016					
	S-1	S-2	S-3	S-4	S-5	S-6	S-1	S-2	S-3	S-4	S-5	S-6
NO-S	0	0	0	0	0	0	0	0	0	0	0	0
AR	0.01	0.01	0	0	0	0	0	0.01	0.01	0	0	0
End	0	0	0.27	0.2	0.16	0.13	0	0	0	0.2	0.16	0.13
AR+End	0	0	0	0	0.2	0.17	0	0	0	0	0	0.17
AR+End+t	0	0	0	0	0	0	0.04	0.02	0.01	0.01	0.01	0.01
S2+t	0	0	0.06	0.05	0.04	0.2	0	0	0	0.05	0.04	0.03
S3+t	0	0.21	0.14	0.11	0.08	0.07	0	0	0.14	0.11	0.08	0.07
Exp	0	0	0	0.25	0.2	0.17	0	0	0	0	0.2	0.17
Exp+t	**0.71**	**0.65**	**0.43**	**0.32**	**0.26**	**0.22**	0.04	0.38	**0.45**	**0.33**	**0.27**	**0.22**
Exp+t+S2	0.27	0.14	0.1	0.07	0.06	0.05	0	0.14	0.09	0.07	0.06	0.05
Exp+t+S3	0	0	0	0	0	0	**0.9**	**0.45**	0.3	0.23	0.18	0.15

Table 5.7: Fine-tuning CHILI: `Blend` sDSS over test periods 2012-2016. Yearly minima are highlighted in bold.

	2012	2013	2014	2015	2016	Mean	Mean Rank
Blend-1	6.94	**7.04**	6.64	**6.97**	8.63	7.24	5
Blend-2	6.89	**7.04**	6.61	7.00	8.26	7.16	4
Blend-3	6.87	7.10	**6.56**	7.02	8.22	7.16	3
Blend-4	**6.86**	7.16	6.63	7.12	7.89	7.13	2
Blend-5	6.89	7.11	6.59	7.54	8.37	7.30	6
Blend-6	7.08	7.21	6.62	7.32	**7.38**	**7.12**	**1**

Table 5.8: Fine-tuning CHILI: `Stack` sDSS over test periods 2012-2016. Yearly minima are highlighted in bold.

	2012	2013	2014	2015	2016	Mean	Mean Rank
Stack-1	6.94	7.06	6.63	6.95	9.14	7.34	6
Stack-2	6.92	7.04	6.62	**6.92**	8.77	7.25	4
Stack-3	6.82	**7.03**	**6.58**	6.97	8.86	7.26	5
Stack-4	6.81	7.30	**6.58**	7.02	8.25	7.19	3
Stack-5	6.81	7.26	6.64	7.05	8.05	7.16	2
Stack-6	**6.80**	7.23	6.63	7.13	**7.96**	**7.15**	**1**

Conclusion We obtain results both in favor of and against the use of ensemble models for infectious disease predictions. On the bright side, analysis over multiple test years show that ensemble models perform well and sometimes better than base models on average. They perform better than the strategies only considering the base model that performs best on the training data according to the AIC, BIC or sDSS [2]. Globally, `MA-AIC` is the worst ensemble model followed by `MA-BIC` but it is not always obvious which ensemble models performs best. It would be interesting to repeat the exercise on longer time series or on many other short time series and average the results. That way we may have a better picture.

A major disadvantage of our methods consists in the high impact of the tuning of hyperparameters on the predictions. Indeed, results depend on the number of blending or stacking years and also what information is used within each of these years (for a reminder, see Figure 2.2). Part of this issue could be avoided with a K-fold cross-validation rather than a rolling-CV. This approach might be worth the try since we have witnessed a considerable impact when changing the blending or stacking period length. Another suggestion would be to discuss with a health specialist what kind of stacking period would make sense. However, we feel that a systematic approach is easier to justify.

To conclude, we would argue that ensemble models may have some potential but that their use for forecasting predictive distributions of time series is not as evident as we thought. Further studies will be necessary help assert the usefulness of ensemble models in the field of disease forecasting.

[2] as calculated in the stacking procedure

Chapter 6

Multivariate Analysis

In this final chapter, we demonstrate that our ensemble model methods also extend to multivariate time series defined in Section 2.5.2. We study the BNV dataset stratified by age group with different contact matrices described in Chapter 2, Subsection 2.5.2. As a reminder, we analyse four different contact matrices (and base models): `Homogeneous`, `No mixing`, `Reciprocal` and `Power-Adjusted`. The code used to generate these models is shown in Appendix A, Section A.2.2 and most of the code used for graphical illustrations are largely inspired by Held and Meyer (n.d., chapter 6).

The BNV dataset contains only 5 years of data (2011-2015), meaning a maximum of 4 years of training data and one test year. We are therefore quite restricted with the different length possibilities of the blending and stacking periods. Moreover, the `Power-Adjusted` model requires at least 3 years of data to be fitted. Hence, only 1 year can be used to stack weights. In this case, there is no difference between `Blend-Sum` and `Blend-Ind`. We will therefore refer to the process simply as `Blend`. Furthermore, the only difference between `Blend` and `Stack` is that `Blend` is fitted on 3 years to make predictions on the 5^{th} test year, whereas `Stack` is fitted on all 4 years. Whether this is reasonable is beyond the point of this chapter as its aim is only to demonstrate the applicability of the ensemble functions.

6.1 Long-Term Predictions

The weights of the four base models in each ensemble model are displayed in Table 6.1. The `Power-Adjusted` model has a weight of exactly 1 except in `MA-BIC` where it is 0.995.

Table 6.1: : Base Model weights in ensemble models.

	MA-AIC	MA-BIC	Blend	Stack
Reciprocal	0	0.004	0	0
Homogeneous	0	0.000	0	0
No mixing	0	0.001	0	0
Power-Adjusted	**1**	**0.995**	**1**	**1**

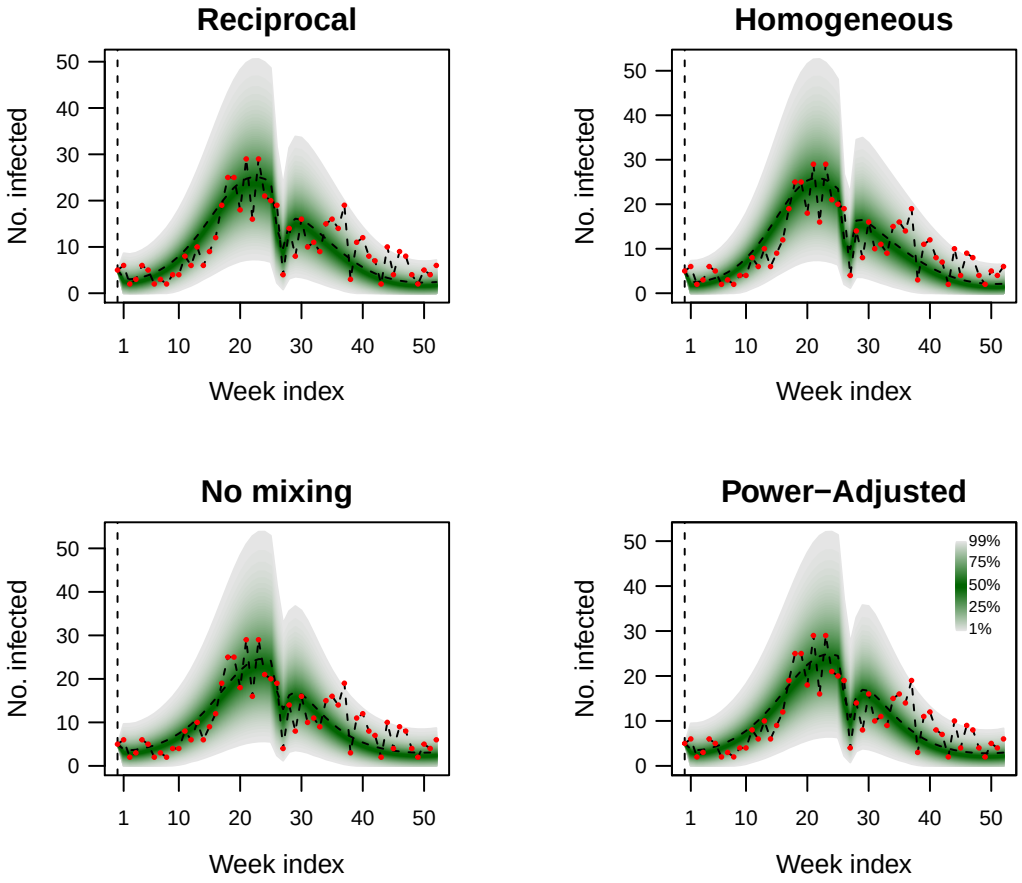

Figure 6.1: **Long-term predictions** of norovirus for age group 00–04.

Figure A.12 shows the predictions for the 4 base models for age group `00-04`. Ensemble models would be indistinguishable from model `Power-Adjusted`. The sudden drop is characteristic of the Christmas period which suffers from under reporting. The plot for all age groups is available in Appendix A, Section A.6 . The predictions differ mainly for age groups `05-14` and `65+`.

Table 6.2: SS for test year and related ranks.

	Reci	Homo	No mixing	Power-Adj	MA-AIC	MA-BIC	Blend	Stack
DSS	1.539	1.564	1.521	1.527	1.527	1.527	1.547	1.527
DS	1.067	1.096	1.078	1.065	1.065	1.065	1.088	1.065
Score Rank	6	8	1	2	2	2	7	2

Table 6.2 shows the sDSS for the predictions of the test year. The model with the best performance is `No mixing` followed by `Power-Adjusted` and all the ensemble models. We suspect that the choice of the test year is of great influence. Indeed, the fact that `Blend` and `Stack` are entirely composed by `Power-Adjusted` indicates that this model performed the best on year 4 (the blending/stacking periods).

6.2 One-Step-Ahead Predictions

This section shows the results of 52 one-step-ahead predictions for the same dataset and test year.
Figure A.5 shows the probability integral transform (PIT) for $6 * 52 = 312$ one-step-ahead forecasts (6 groups and 52 timepoints). We could argue that the data is slightly shaped as an inverse-U indicating that the data is overdispersed although nothing alarming.
Figure 6.4 and 6.5 shows the predictive distribution and the corresponding scores on test year for base model `No mixing` and ensemble model `Stack`. Similar plots for all models are found in Appendix A.7. We observe that the predictions and the scores are very similar between the two models which is not surprising since `Stack` is mostly made up by the base model `No mixing`.
The mean sDSS and the model score rank are shown in Table 6.3. `No mixing` model preforms the best followed by `Stack`. Again, it is delicate to draw conclusions on only one validation year, but it seems that stacking may adapt faster than `MA-AIC` and `MA-BIC`. We could try to improve this by shortening the stacking period. As a reminder, the stacking period used was 1 year and increases by one observation after each forecast. Maybe shortening this period would allow the model to adapt faster. We would then enter some fine-tuning analysis which is beyond the scope of this thesis.

Conclusion We have shown through this simple example that our methods can be extended to multivariate time series analysis. The methods are subject to the same limitations as univariate time series analysis namely: dependency to the base model pool and to fine tuning of ensemble model open parameters. Moreover, results are very likely subject to the test year, the observation we condition our long-term forecast on and the length of the period under study.

Table 6.3: sDSS for test year and related ranks based on 6*52 one-step-ahead forecasts.

	Reci	Homo	No mixing	Power-Adj	MA-AIC	MA-BIC	Stack
DSS	3.031	3.093	3.003	3.011	3.014	3.015	3.006
Score Rank	6	7	1	3	4	5	2

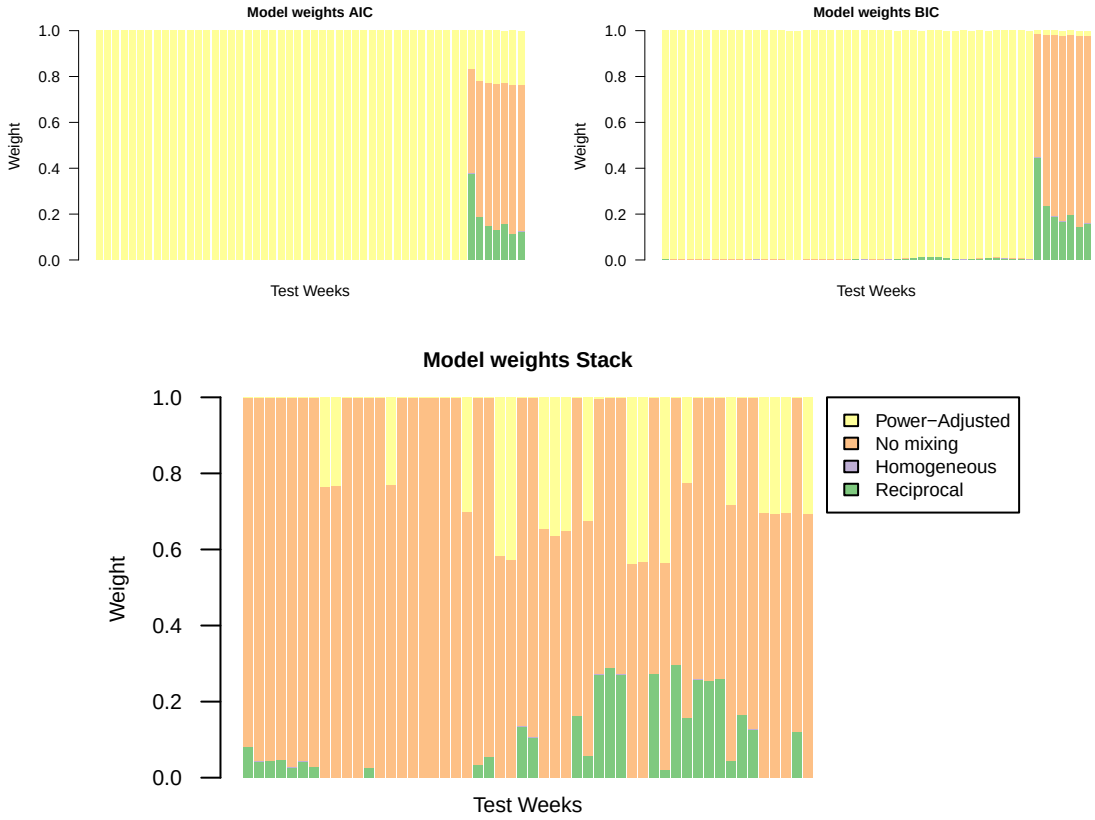

Figure 6.2: **Base Model weights** in ensemble models for one-step-ahead predictions.

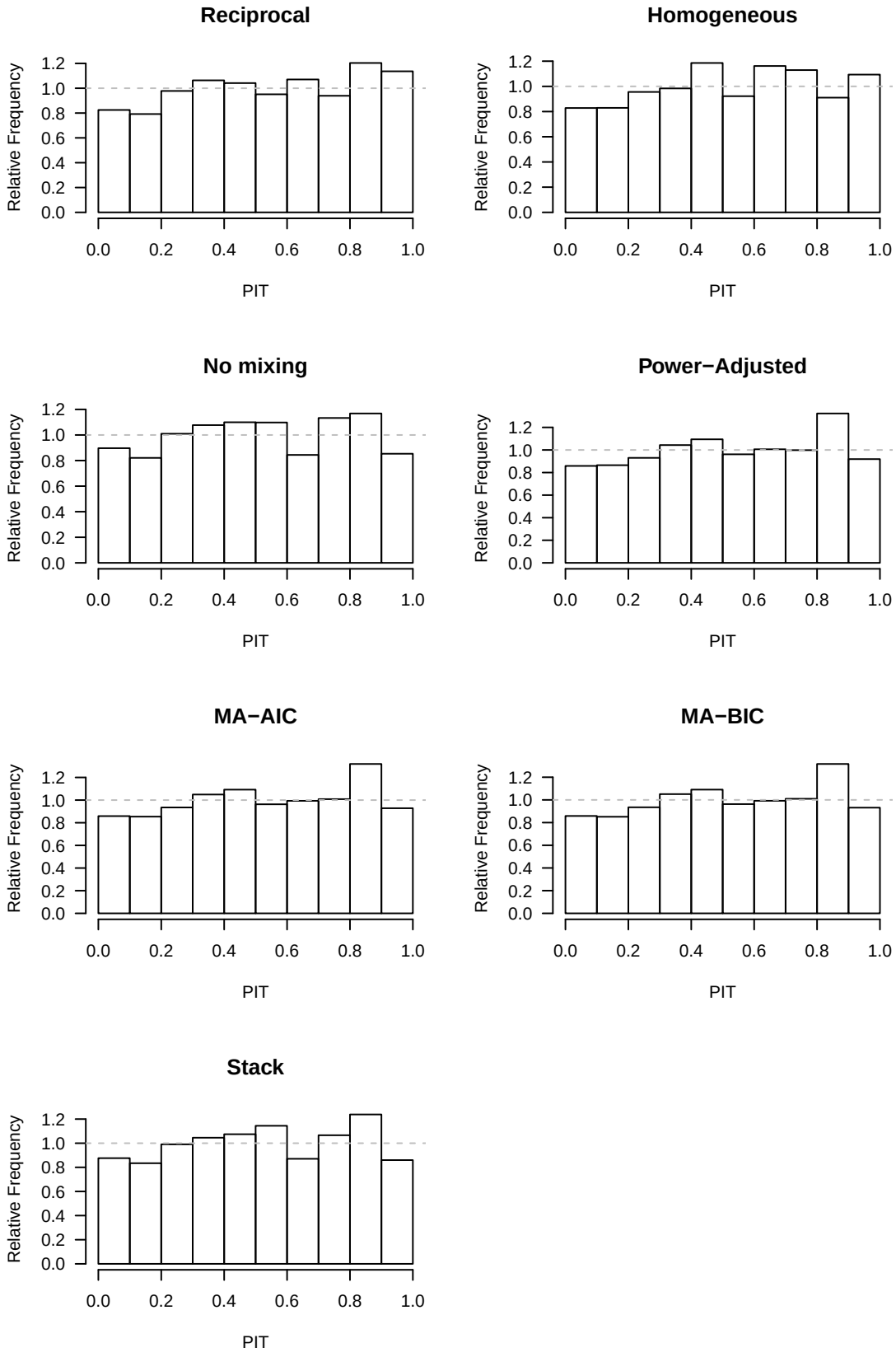

Figure 6.3: **Probability Integral Transform (PIT)** of 6*52 one-step-ahead forecasts.

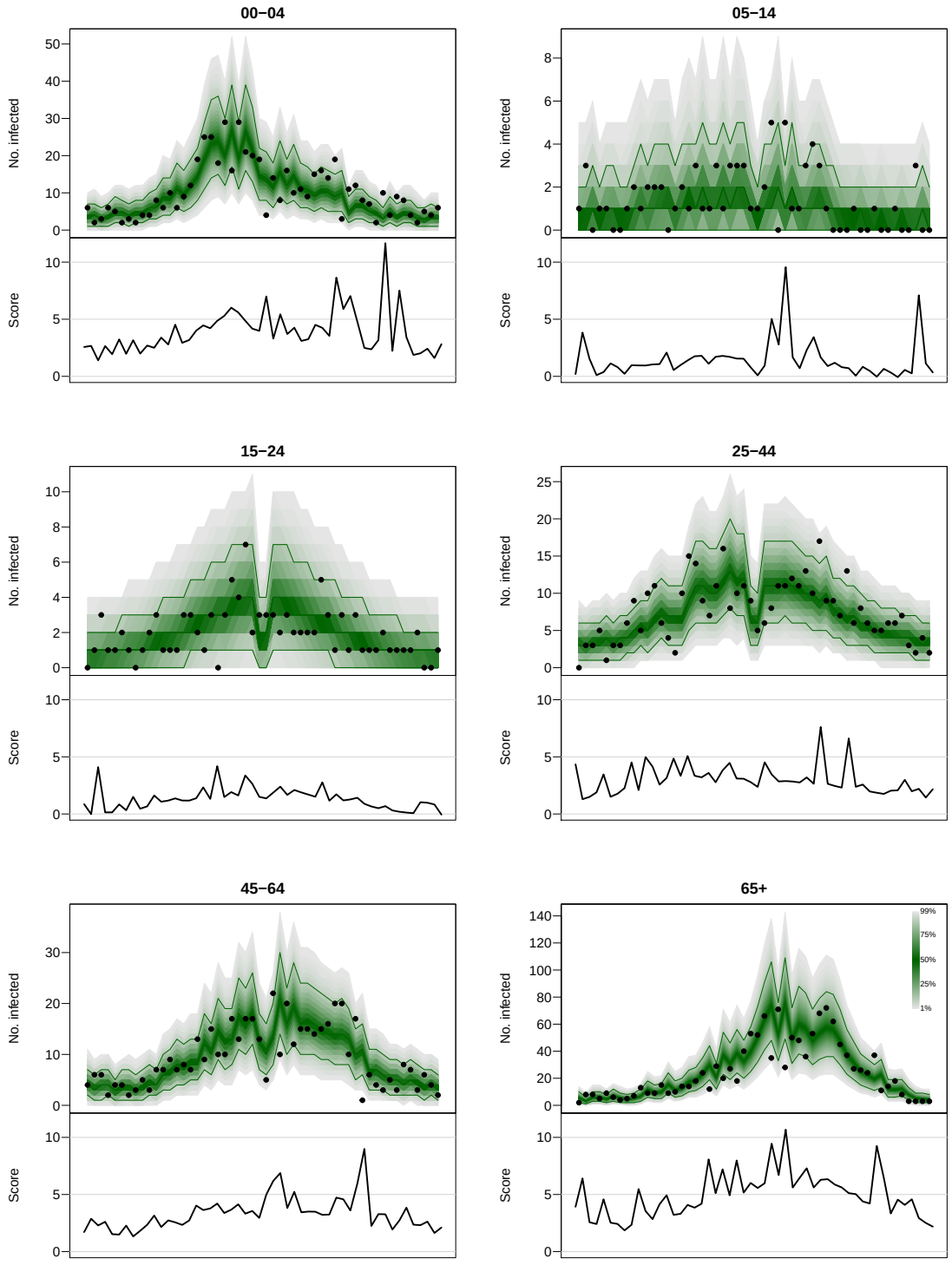

Figure 6.4: **One-step-ahead predictive distribution and sDSS for `No Mixing` base model** for test year 2015 of BNV dataset.

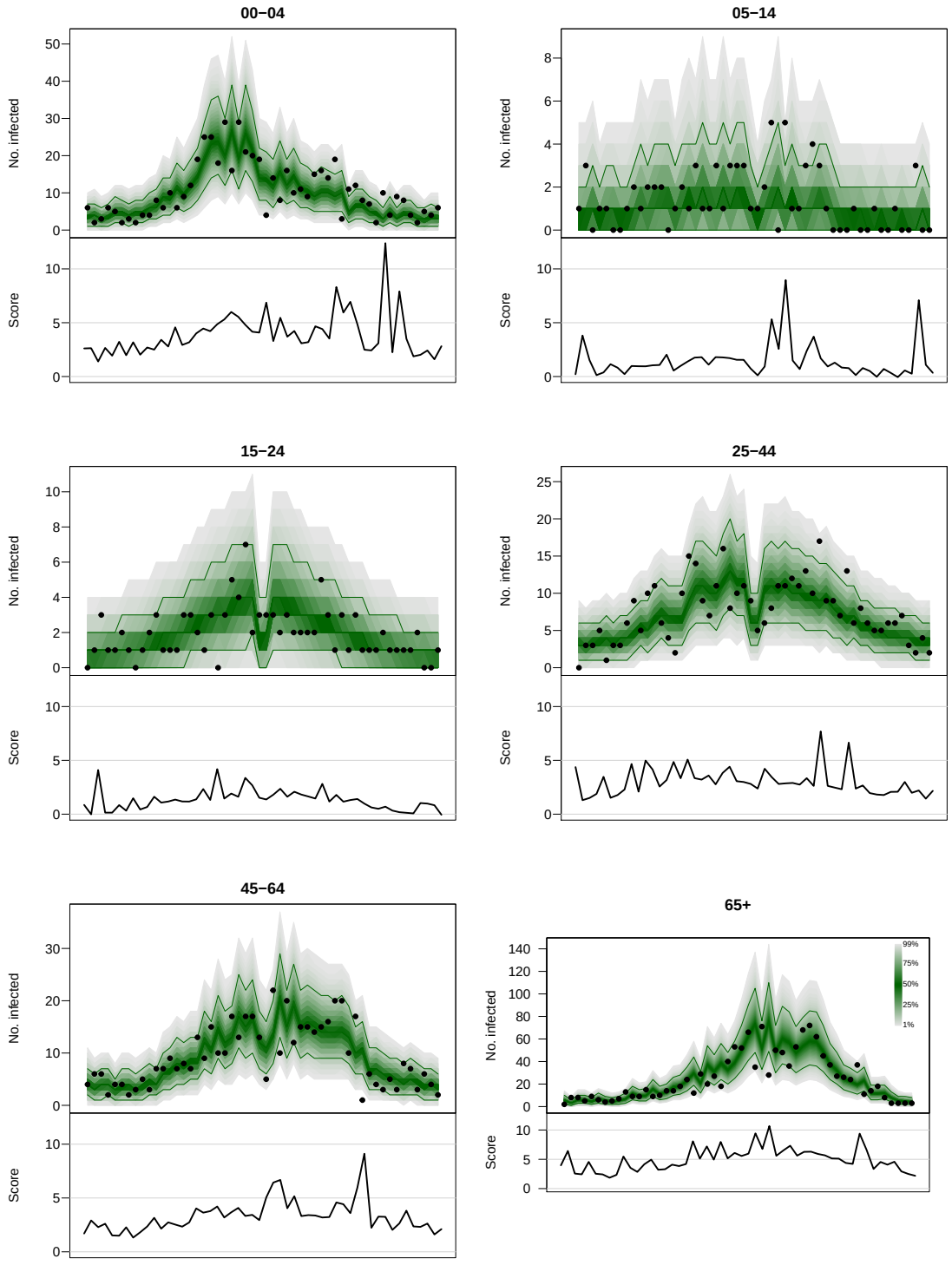

Figure 6.5: **One-step-ahead predictive distribution and sDSS for `Stack` base model** for test year 2016 of BNV dataset.

Chapter 7

Discussion

Nowadays, theory and practice indicate that ensemble models improve predictions on average. For a long time restricted to point forecasts, this idea is spreading outside of the field of machine learning as methods are adapted to predictive distributions. In this thesis, we have implemented ensemble models techniques - specifically, model averaging according to the AIC or BIC criteria, blending and stacking - within the HHH4 framework. We adapted the stacking procedure described by Yao et al. (2017) to a time series settings which led to the distinction between *blending* and what we called *stacking* depending on the use or not of a *rolling cross-validation*.
Tested over 5 years on both the CHILI and Dengue dataset, ensemble methods showed promising results, performing well on average but more importantly, performing better than the strategies `Best.AIC`, `Best.BIC` and `Best.Stack` which consider the model that performs the best on the training data according to the AIC, BIC or sDSS scores[1] respectively. Globally, `Blend` and `Stack` perform better than `MA-AIC` and often better than `MA-BIC`.
However, the benefit of ensemble models is not straightforward to ascertain for two reasons.
First, we evaluate the performance of the models according the to sDSS. This enables us to easily rank and compare models but does not inform us whether the difference in scores is relevant or not. Therefore, is it possible that many models with similar scores but far apart in ranks are actually equally valuable.
The second reason is that many parameters influence the results. Some of these parameters affect all predictive models and others affect only the ensemble models presented here.
We raised the issue of the influence of tested year and the length of the tested period on the results of both the ensemble models but also the base models. In her master thesis Reeve (2017) shows that the week on which we condition the forecast also plays an important role. These three variables influence the forecasts. This issue probably concerns predictions from all types of models - also outside of the HHH4 framework - and will be difficult to solve. An approach to reduce the variability induced by the test year under study and give a better picture of the benefit of ensemble models would be to analyze more data: longer or more time series. In the meantime, we also believe that better communication with healthcare specialists

[1] as calculated in the stacking procedure

may help create some guidelines for specific diseases and eventually help statistical analyses better target specific needs. For instance, with ILI diseases, we can imagine that due to the time needed to perform the biological measurements and report the number of weekly cases, it would not be possible to condition on the first week of the year and already make predictions for the second week. Furthermore, it is likely that predictions of an entire year are not necessary for dengue. Although this does not solve the main issue of the variation resulting from the above mentioned parameters, it could help narrow down what is feasible in practice and what is not.
Regrading our ensemble model procedures specifically, this thesis revealed two main limitations. First, we have noticed that hyperparameters specific to ensemble models - namely the length of the blending or stacking period - also highly influence the results. It is likely that other hyperparameters such as the type of information used to stack weights (reminder: Figure 2.2, panel C) play an important role as well. Regarding the latter, we see no other option than testing multiple options and see what works best in each case. With respect to the length of the blending and stacking period, we believe that a K-fold cross validation would be a good approach to solve this matter. Regardless of this issue, we would recommend to test a K-fold cross-validation based on the paper from Bergmeir et al. (2017) who showed that the performance of the K-fold cross-validation outperforms other forms of validation in time series.
Second, the stability of the stacking procedure is not optimal. The output of the optimization procedure (i.e. the weights) varies depending on the initial parameters and many different weight combinations yield very similar sDSS scores. It is possible that our optimization suffers from *multicollinearity* which arises when one (or more) models may be expressed as a linear combination of the other models and is known to produce unreliable estimates. Because all our models are very similar to one another (they are all built within the HHH4 framework and some are nested) this could be an issue and our procedure may benefit from adding completely different models. In the point estimation literature, Merz and Pazzani (1999) describes a method based on principal component regression to address the issue of multicollinearity while identifying the unique contributions of each base model. Yao et al. (2017) state that when combining predictive distributions, there is no need to worry about multicollinearity except in degenerate cases and advice to use stronger priors on the weights when the dimension of model space is high. We have shown that by setting the initial parameters corresponding to using only the best base model in training, we obtain good solutions. An extensive review of the literature will be necessary to find whether there is a more robust way to improve the stability of our optimization procedure.
Finally, we would like to point out that all the one-step-ahead analyses were evaluated with sDSS but are not limited to this score. As explained in the methods, other proper scoring rules can be used in this case. Inferences about ensemble models would benefit greatly from another "point of view". In theory, our code allows to calculate and minimize the Log-Score and the RPS in addition to the sDSS. In practice, changing the score type considerably slows our code down. Due to time restrictions, we have not been able to address this issue but it is an important element to consider for further research.

Appendix A

Appendix

A.1 Software

R version and packages used to generate this report:

R (version: R version 3.3.1 (2016-06-21)), A foundation for statistical computing, Vienna, Austria available at `http://www.R-project.org/`.

Base packages stats, graphics, grDevices, utils, datasets, methods, base

Other packages

nlme; Version 3.1.131, plotrix; Version 3.7, biostatUZH; Version 1.7.0, survival; Version 2.41.2, ggrepel; Version 0.7.0, zoo; Version 1.8.1, gridExtra; Version 2.2.1, latticeExtra; Version 0.6.28, cowplot; Version 0.9.2, kableExtra; Version 0.8.0, lattice; Version 0.20.34, tidyr; Version 0.6.1, RColorBrewer; Version 1.1.2, colorspace; Version 1.3.2, reshape2; Version 1.4.1, ggplot2; Version 2.2.1, gplots; Version 3.0.1, HIDDA.forecasting; Version 0.5.1, hhh4addon; Version 0.0.0.0.9004, polyCub; Version 0.6.1, hhh4contacts; Version 0.13.0, surveillance; Version 1.16.0, xtable; Version 1.8.2, sp; Version 1.2.7, knitr; Version 1.20

This document was generated on 30.04.2018 at 17:33.

A.2 Base Model Code

A.2.1 `CHILI` & `Dengue`

We upload both datasets. The `CHILI` dataset is contained in the **HIDDA.forecasting** package whereas the `Dengue` dataset was downloaded from `https://predict.phiresearchlab.org/` and converted to a .csv file.
We then show the code for the model controls (not the fitted objects) which are necessary for our functions. Note that the controls for the models does not contain a `subset=` argument in the list. This is not necessary when using our ensemble model functions. The subset will be calculated automatically.

```
library(surveillance)
library(HIDDA.forecasting)
```

```r
### Data

# CHILI
chili <- CHILI
chili <- sts(observed = chili,
             epoch = as.integer(index(CHILI)), epochAsDate = TRUE)
# Dengue
dengue.csv <- read.csv("data/san_juan_testing_data.csv")
dengue <- sts(observed = dengue.csv$total_cases,
              start= c(1990, 18),
              frequency = 52,
              population = NULL)

### Model Controls:

# Model 1: NO S
set.m1.lt <- list(
  end = list(f = ~ 1),
  ar = list(f = ~ 1),
  family = "NegBinM")

# Model 2: End
set.m2.lt <- list(
  end = list(f = addSeason2formula(~ 1,  S=1)),
  ar = list(f =~ 1),
  family = "NegBinM")

# Model 3: AR
set.m3.lt <- list(
  end = list(f =~ 1),
  ar = list(f = addSeason2formula(~ 1,  S=1)),
  family = "NegBinM")

# Model 4: AR + End
set.m4.lt <- list(
end = list(f = addSeason2formula(~ 1,  S=1)),
ar = list(f = addSeason2formula(~ 1,  S=1)),
family = "NegBinM")

# Model 5: AR + End + t
set.m5.lt <- list(
  end = list(f = addSeason2formula(~ 1 +I(t),  S=1)),
  ar = list(f = addSeason2formula(~ 1+ I(t),  S=1)),
  family = "NegBinM")

# Model 6: S2 + t
set.m6.lt <- list(
  end = list(f = addSeason2formula(~ 1 +I(t),  S=2)),
  ar = list(f = addSeason2formula(~ 1+ I(t),  S=2)),
  family = "NegBinM")

# Model 7: S3 + t
set.m7.lt <- list(
  end = list(f = addSeason2formula(~ 1 +I(t),  S=3)),
  ar = list(f = addSeason2formula(~ 1+ I(t),  S=3)),
```

```
  family = "NegBinM")

# Model 8: Exp
set.m8.lt <- list(
end = list(f = addSeason2formula(~ 1,  S=1)),
ar = list(f = addSeason2formula(~ 1,  S=1),
          use_distr_lag = TRUE,
          max_lag = 5),
family = "NegBinM")

# Model 9: Exp + t
set.m9.lt <- list(
end = list(f = addSeason2formula(~ 1+ I(t),  S=1)),
ar = list(f = addSeason2formula(~ 1+ I(t),  S=1),
          use_distr_lag = TRUE,
          max_lag = 5),
family = "NegBinM")

# Model 10: Exp + t + S2
set.m10.lt <- list(
  end = list(f = addSeason2formula(~ 1+ I(t),  S=2)),
  ar = list(f = addSeason2formula(~ 1+ I(t),  S=2),
            use_distr_lag = TRUE,
            max_lag = 5),
  family = "NegBinM")

# Model 11: Exp + t + S3
set.m11.lt <- list(
  end = list(f = addSeason2formula(~ 1+ I(t),  S=3)),
  ar = list(f = addSeason2formula(~ 1+ I(t),  S=3),
            use_distr_lag = TRUE,
            max_lag = 5),
  family = "NegBinM")
```

A.2.2 Base Model Code: BNV

The `BNV` data is available in the hhh4contacts package by calling `noroBE` and stratifying by age group. Note that we again need the model controls rather than the fitted objects. However, this is not possible for the `Power-Adjusted` model which is build on a fitted object. In our ensemble functions, we can specify if a `Power-Adjusted` is required and define which model should be used to build it and which contact matrix. We show you here how such a model is fitted outside our functions.

```
library(surveillance)
library(hhh4contacts)

### Data
BNV <- noroBE(by = "agegroup", agegroups = c(1, 2, 2, 4, 4, 2),
              timeRange = c("2011-w27", "2016-w26"))

# Other settings
NGROUPS <- ncol(BNV)
TRAIN <- 2:(4*52) # The training data is 4 years long
DATAt <- list(t = epoch(BNV) - 1,
```

```
              christmas = as.integer(epochInYear(BNV) %in% c(52, 1)))

# Reciprocal contact matrix
C_reci <- contactmatrix(which = "reciprocal", grouping = c(1, 2, 2, 4, 4, 2))

### Model controls:

# Homogenous
set_Chom <- list(
end = list(f = addSeason2formula(~0 + fe(1, unitSpecific = TRUE) + christmas,
                                 S = rep(1, NGROUPS))),
ne = list(f = addSeason2formula(~0 + fe(1, unitSpecific = TRUE)),
          weights = matrix(1, NGROUPS, NGROUPS),
          scale = NULL, normalize = TRUE),
family = "NegBinM", data = DATAt, subset = TRAIN)

# No Mixing
set_Cdiag <- list(
end = list(f = addSeason2formula(~0 + fe(1, unitSpecific = TRUE) + christmas,
                                 S = rep(1, NGROUPS))),
ne = list(f = addSeason2formula(~0 + fe(1, unitSpecific = TRUE)),
          weights = matrix(1, NGROUPS, NGROUPS),
          scale = diag(NGROUPS), normalize = TRUE),
family = "NegBinM", data = DATAt, subset = TRAIN)

# Reciprocal
set_Creci <-list(
end = list(f = addSeason2formula(~0 + fe(1, unitSpecific = TRUE) + christmas,
                                 S = rep(1, NGROUPS))),
ne = list(f = addSeason2formula(~0 + fe(1, unitSpecific = TRUE)),
          weights = matrix(1, NGROUPS, NGROUPS),
          scale = C_reci, normalize = TRUE),
family = "NegBinM", data = DATAt, subset = TRAIN)

### Fitted object: Power-Adjusted

# We first need to fit the Reciprocal model
fit_Creci<- hhh4(BNV, set_Creci)

# We then fit the Power-Adjusted model
fit_Cpower <- fitC(fit_Creci, C_reci, normalize = TRUE, truncate = TRUE)
```

A.3 Fitted and Long-Term Predictions of Test Year 2016 (CHILI)

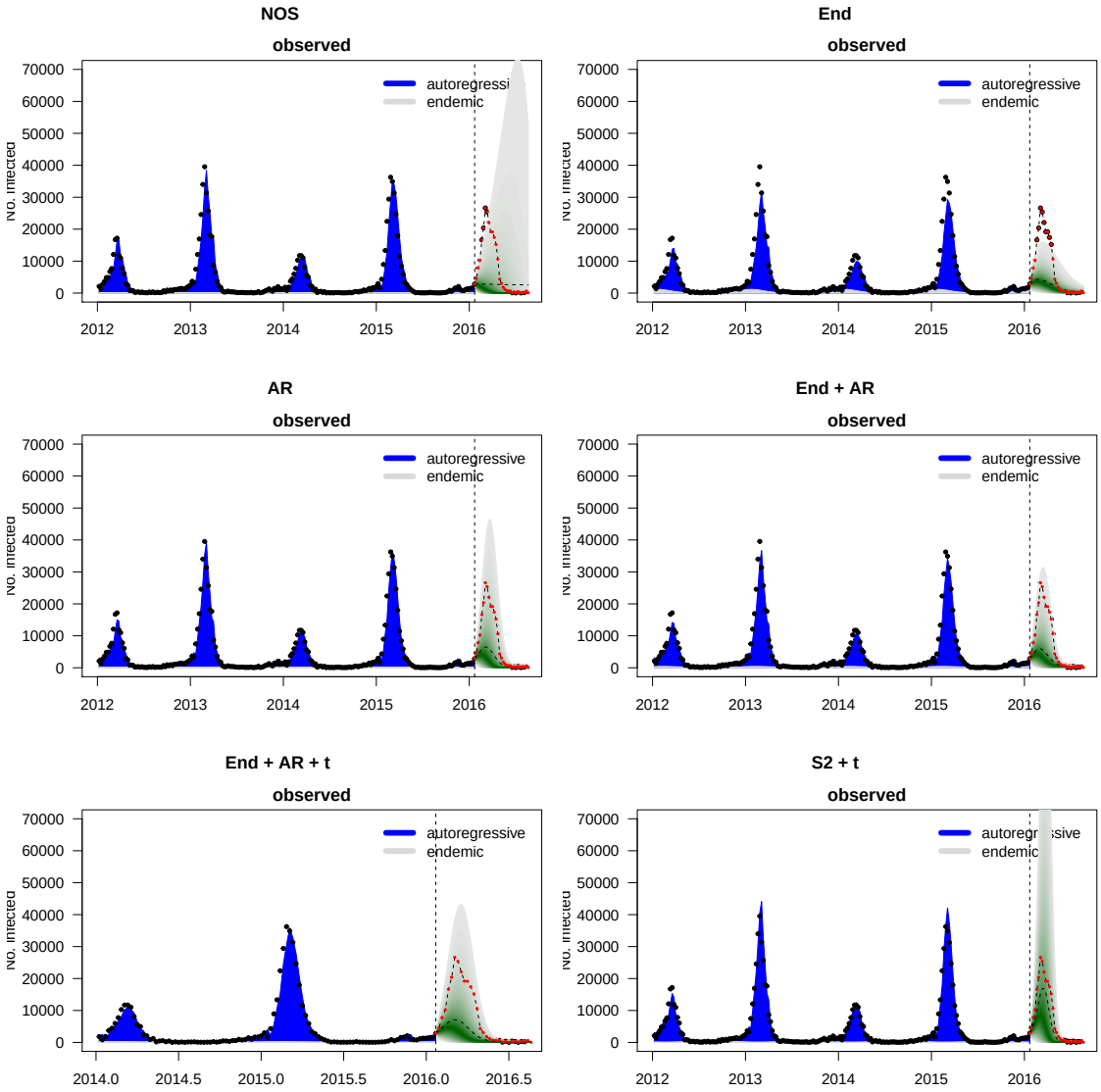

Figure A.1: **Fitted values and predictions for base model `NO S`, `End`, `AR`, `End + AR`, `End + AR + t` and `S2+ t`.** Fitted values for the last 4 years of training data of the CHILI dataset and predicted values for the test data (year 2016).

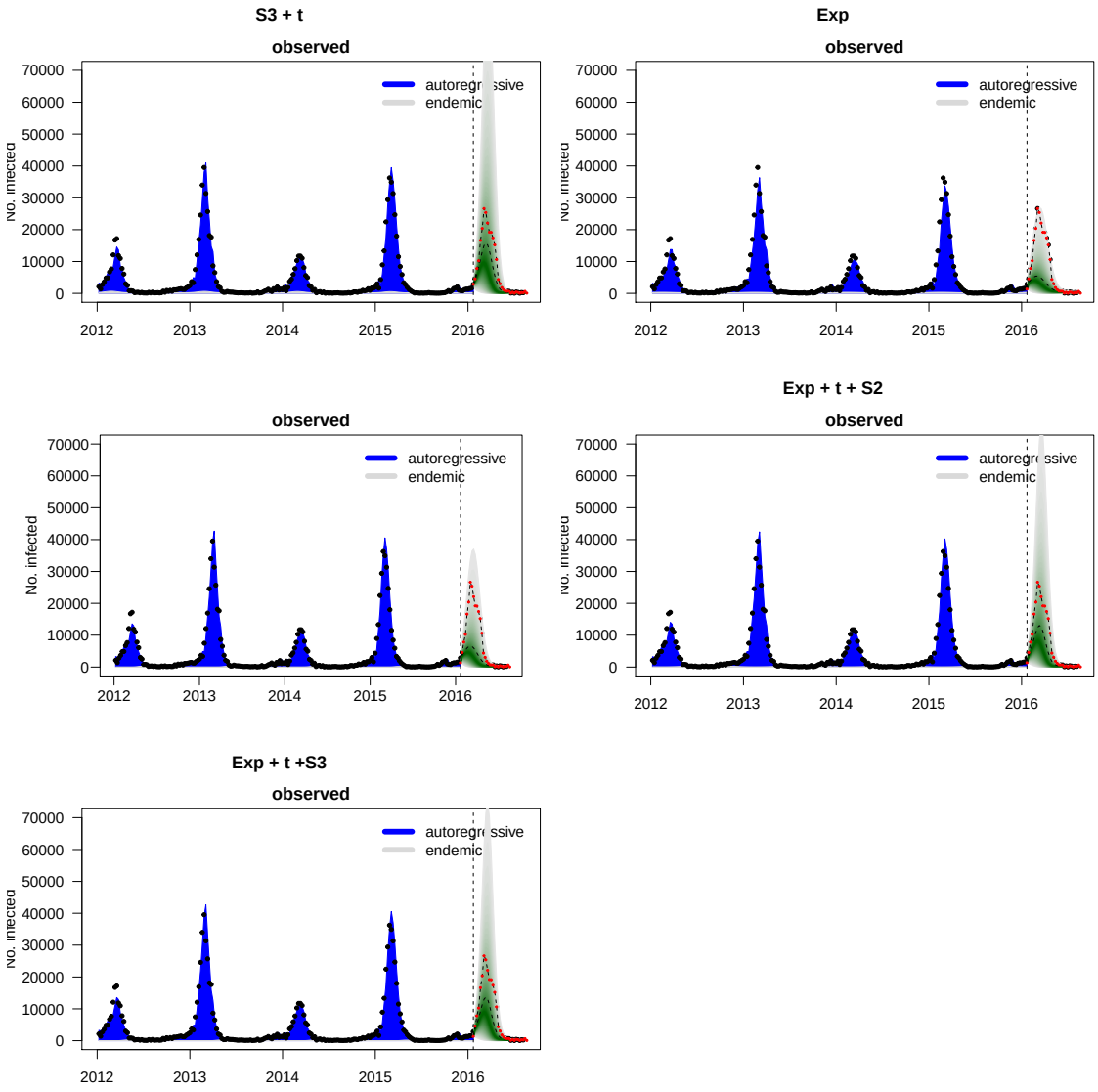

Figure A.2: **Fitted values and predictions for base model `S3 + t`, `Exp`, `Exp + t`, `Exp + t + S2` and `Exp + t + S3`**. Fitted values for the last 4 years of training data of the CHILI dataset and predicted values for the test data (year 2016).

A.4 Long-Term Predictions of Test Year 2016 (CHILI)

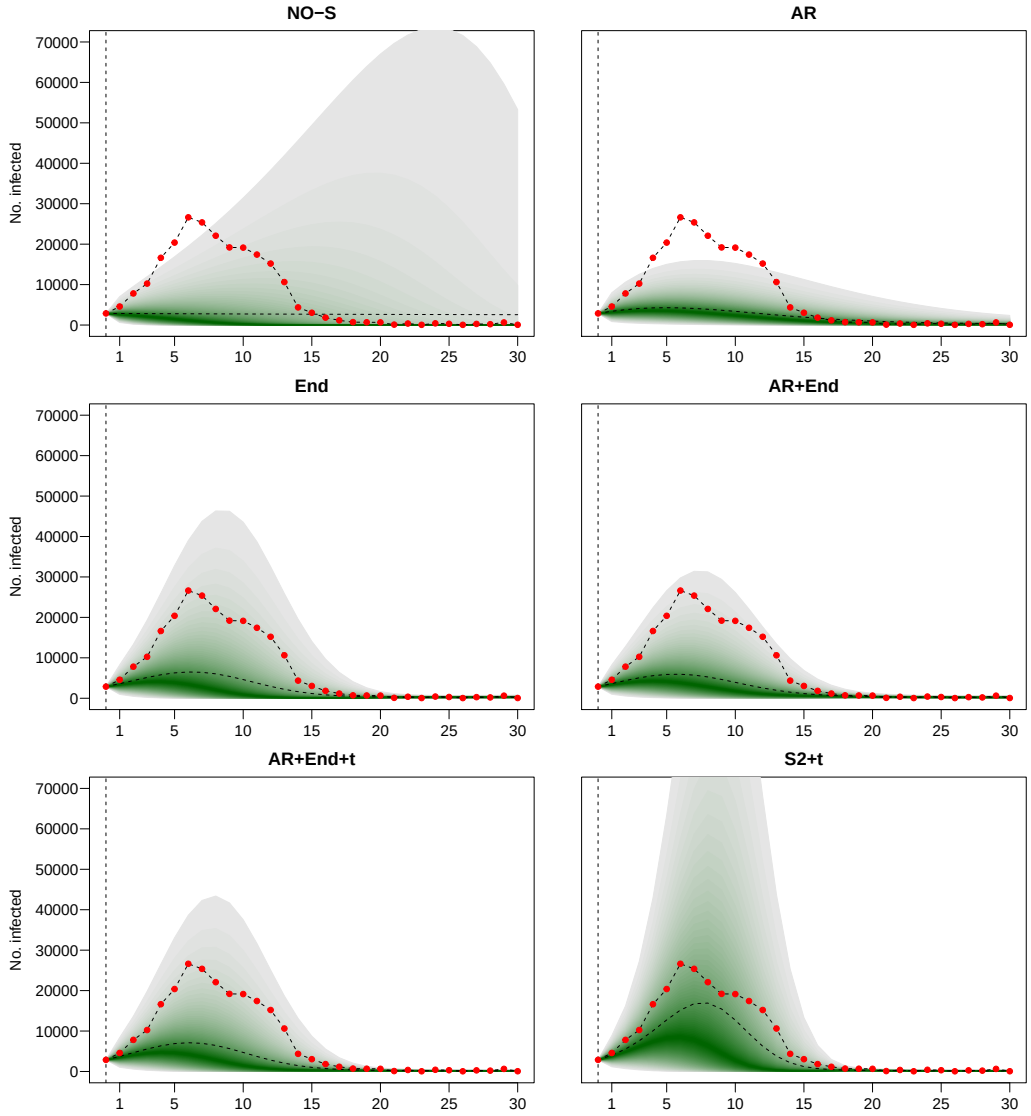

Figure A.3: **Base models predictions** for test year 2016 of the CHILI dataset.

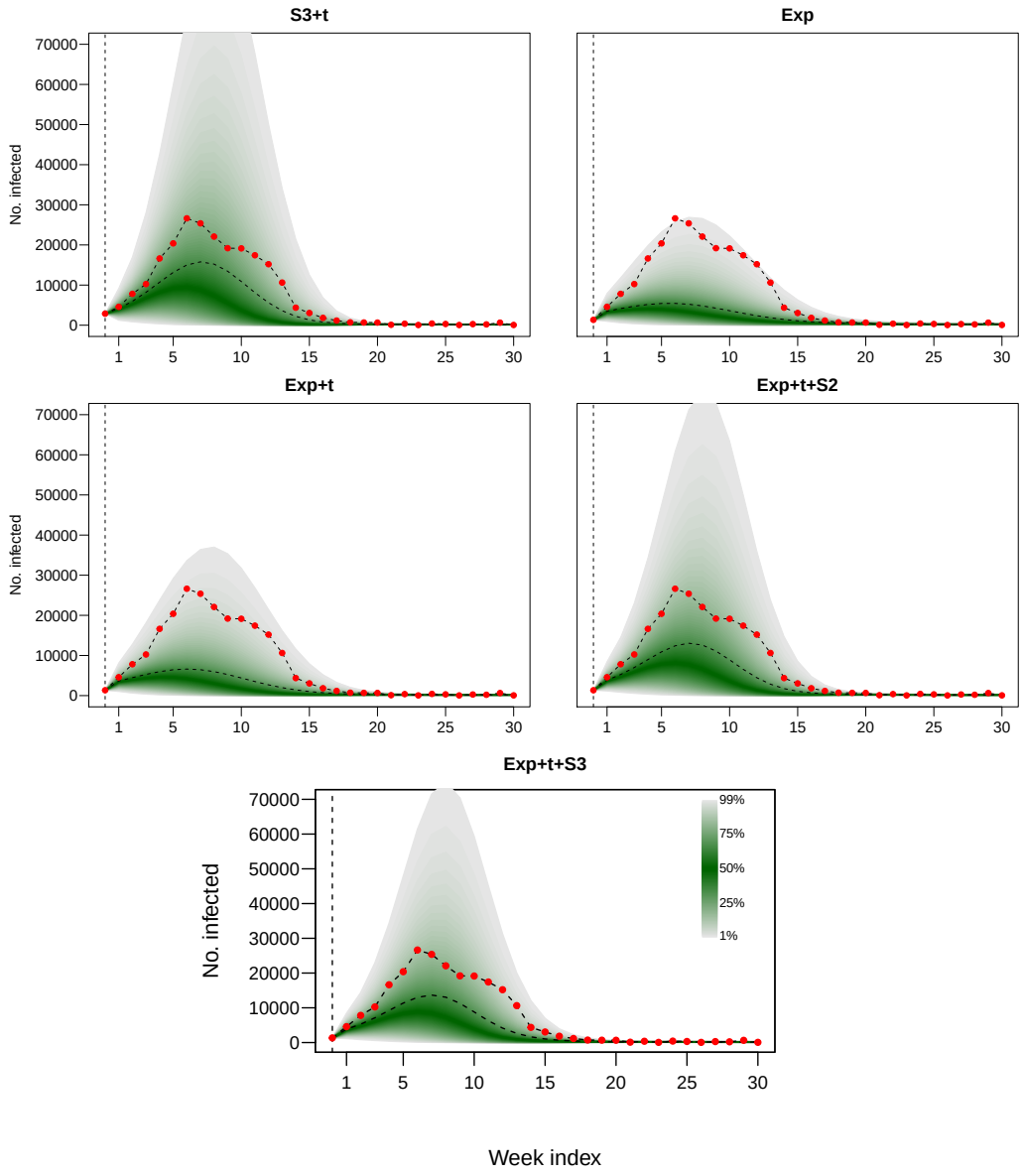

Figure A.4: **Base models predictions** for test year 2016 of the CHILI dataset.

A.5 Probability Integral Transform: 30 Observations of Test Year 2016 (CHILI)

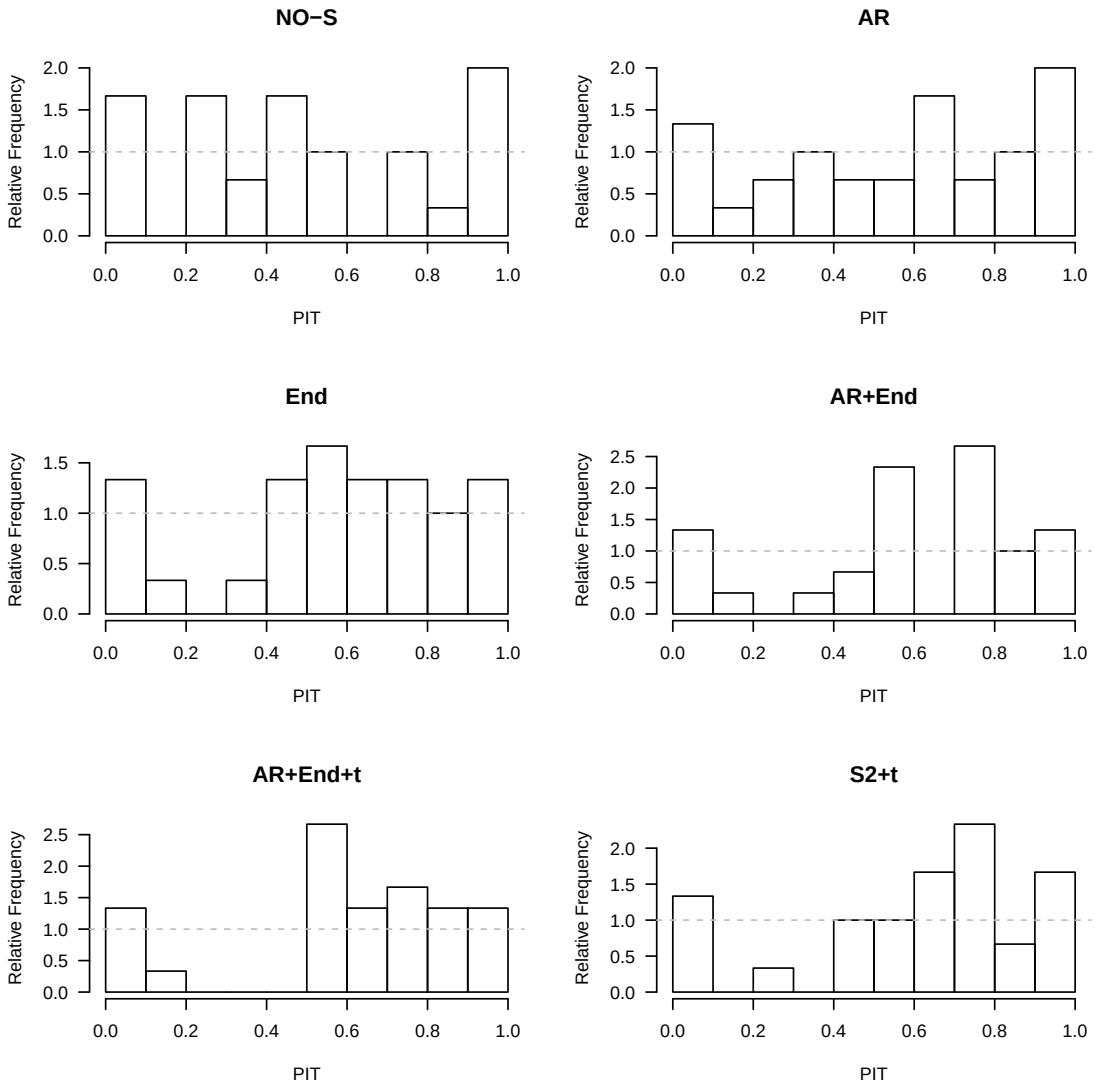

Figure A.5: **Probability Integral Transform (PIT)** of 30 one-step-ahead forecasts of the CHILI test year (2016).

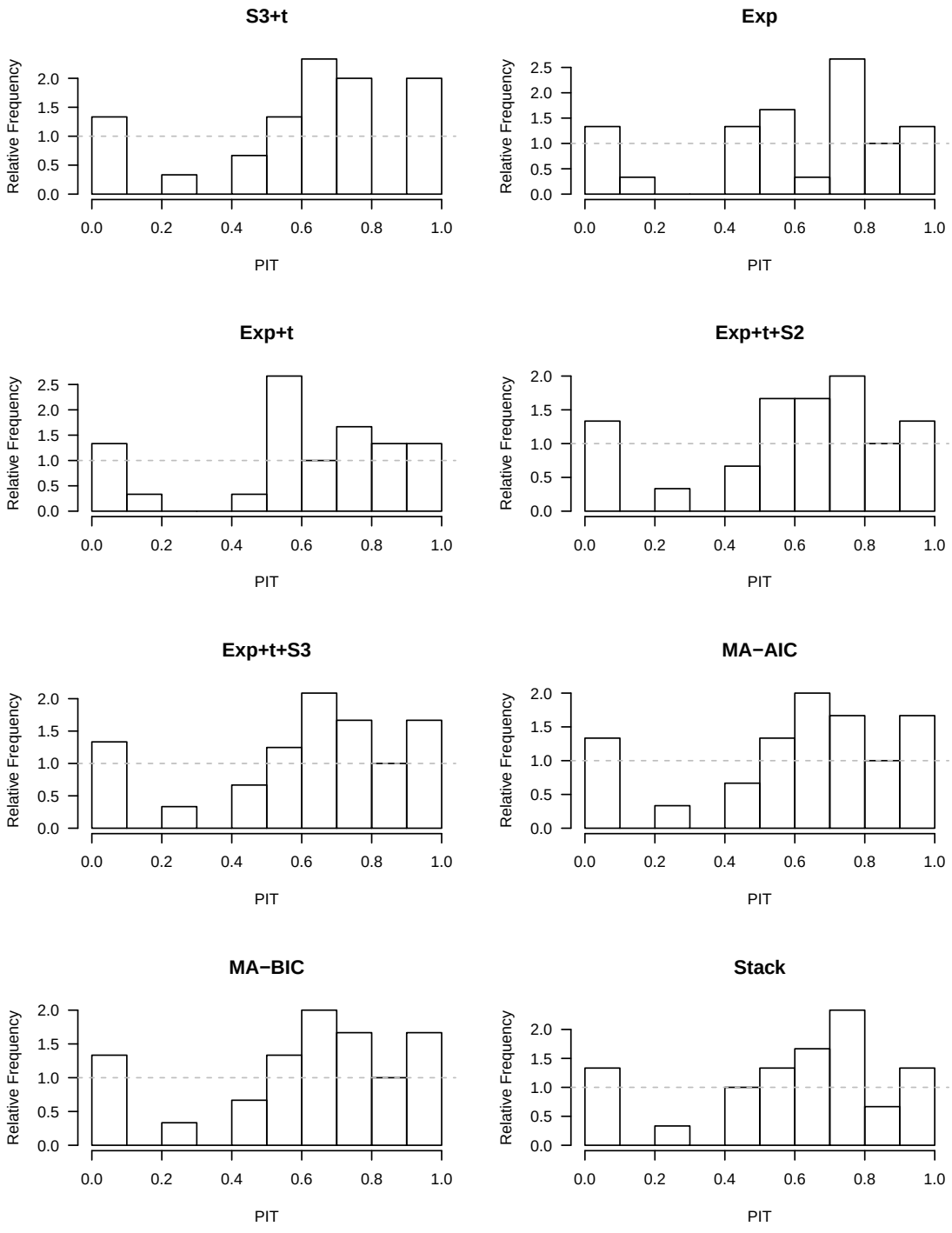

Figure A.6: **Probability Integral Transform (PIT)** of 30 one-step-ahead forecasts of the CHILI test year (2016).

A.6 Long-Term Predictions of Test Year 2016 (BNV)

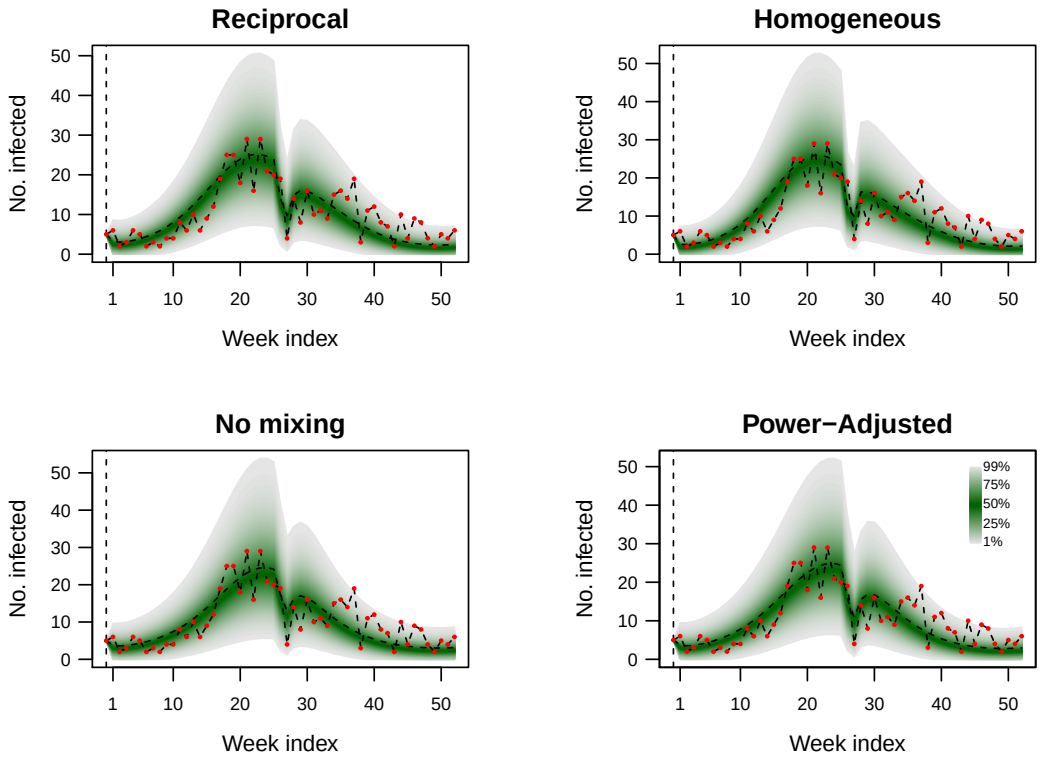

Figure A.7: **Long-term predictions** of norovirus for age group 00–04.

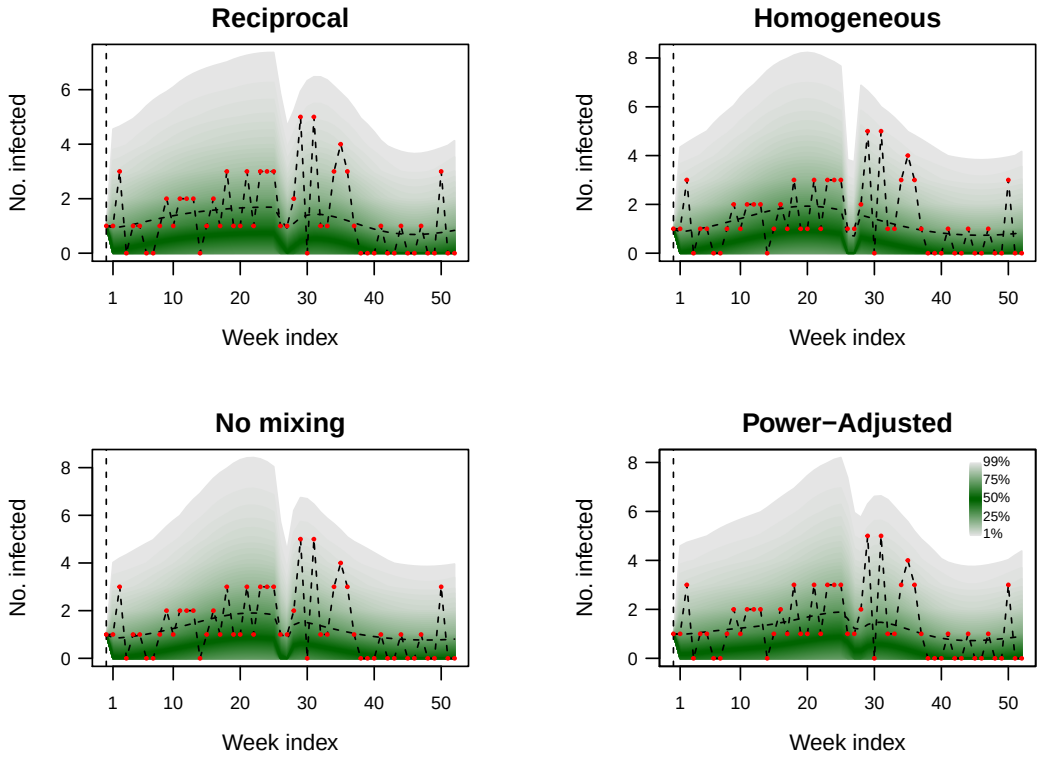

Figure A.8: **Long-term predictions** of norovirus for age group 05–14.

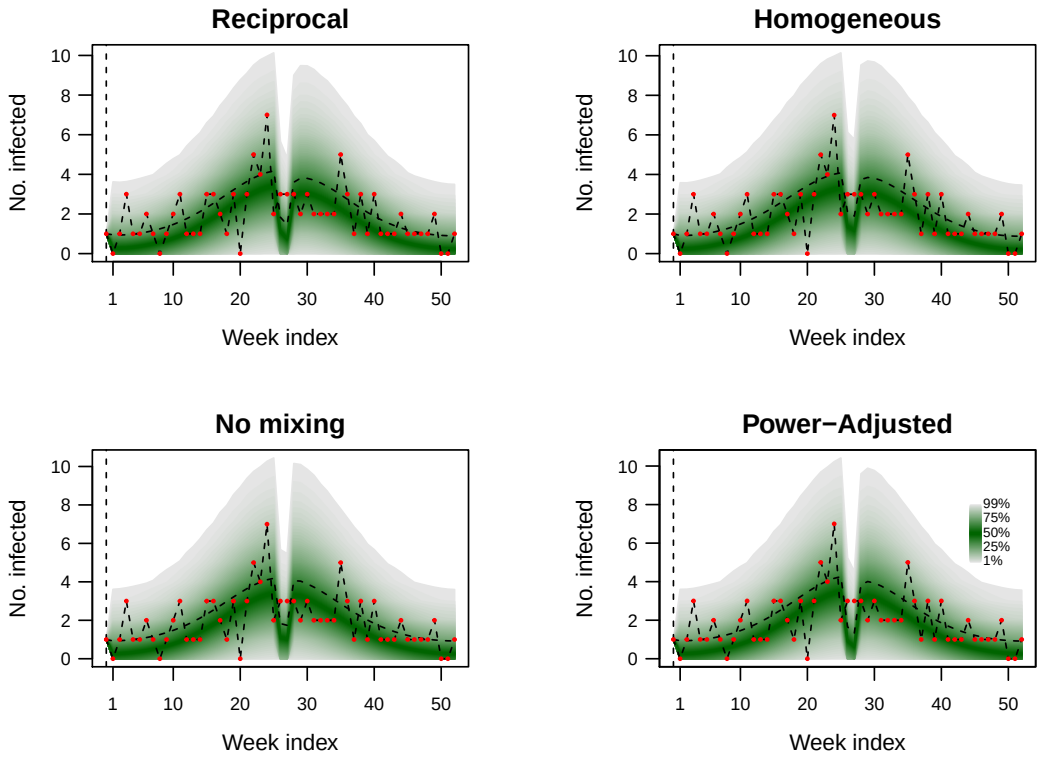

Figure A.9: **Long-term predictions** of norovirus for age group 15–24.

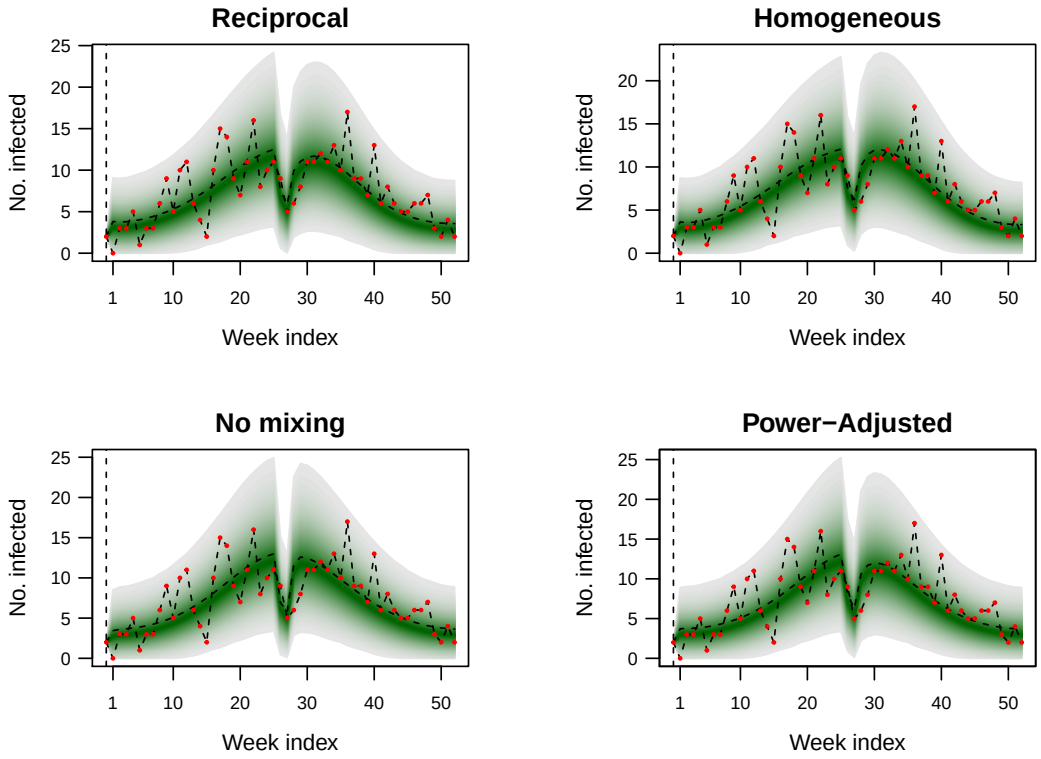

Figure A.10: **Long-term predictions** of norovirus for age group 25–44.

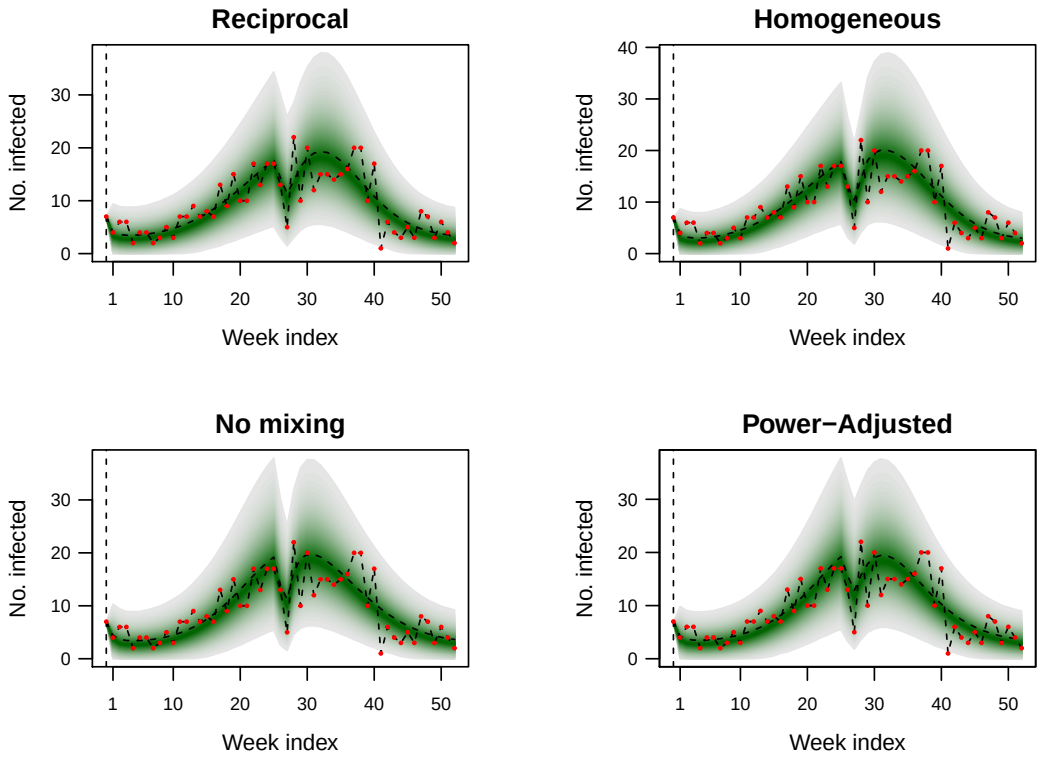

Figure A.11: **Long-term predictions** of norovirus for age group 45–64.

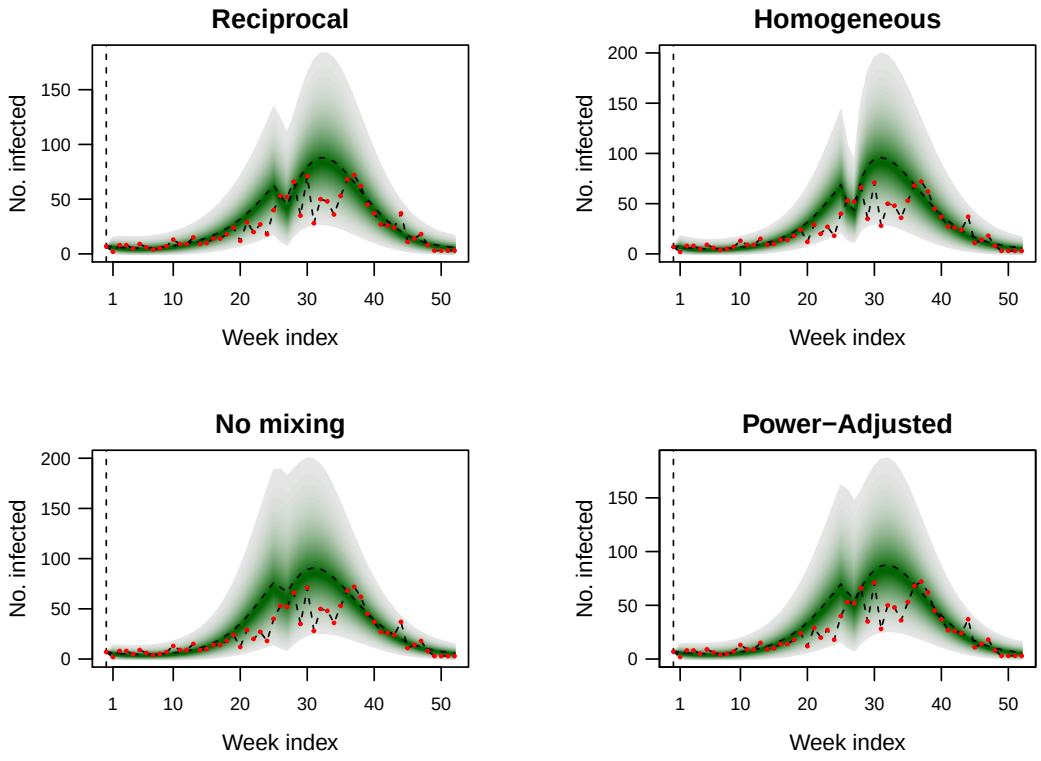

Figure A.12: **Long-term predictions** of norovirus for age group 65+.

A.7 One-Step-Ahead Predictions of Test Year 2016 (BNV)

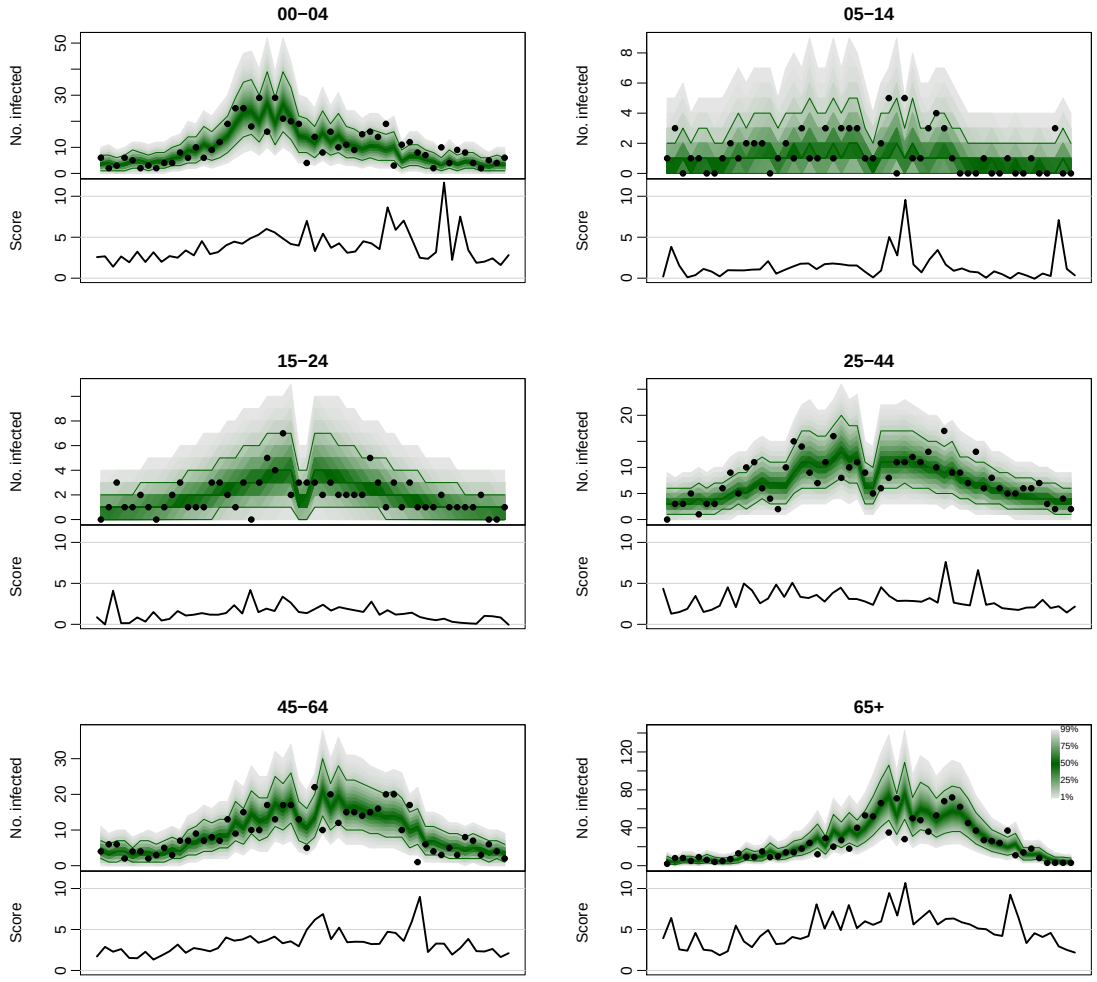

Figure A.13: **Predictive distribution and sDSS for `No Mixing` base model** for one-step-ahead-predictions of test year 2015 of BNV dataset.

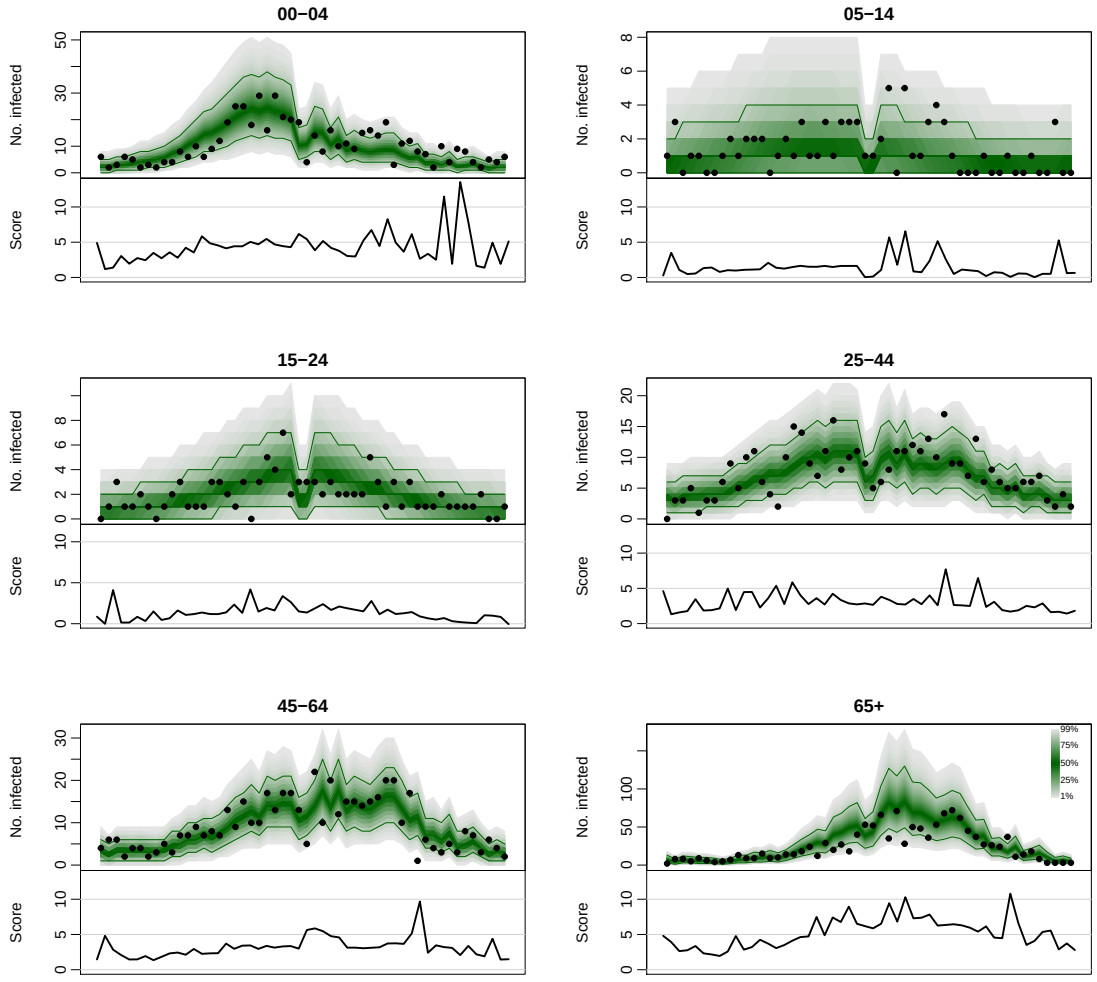

Figure A.14: **Predictive distribution and sDSS for `Homogeneous` base model** for one-step-ahead-predictions of test year 2015 of BNV dataset.

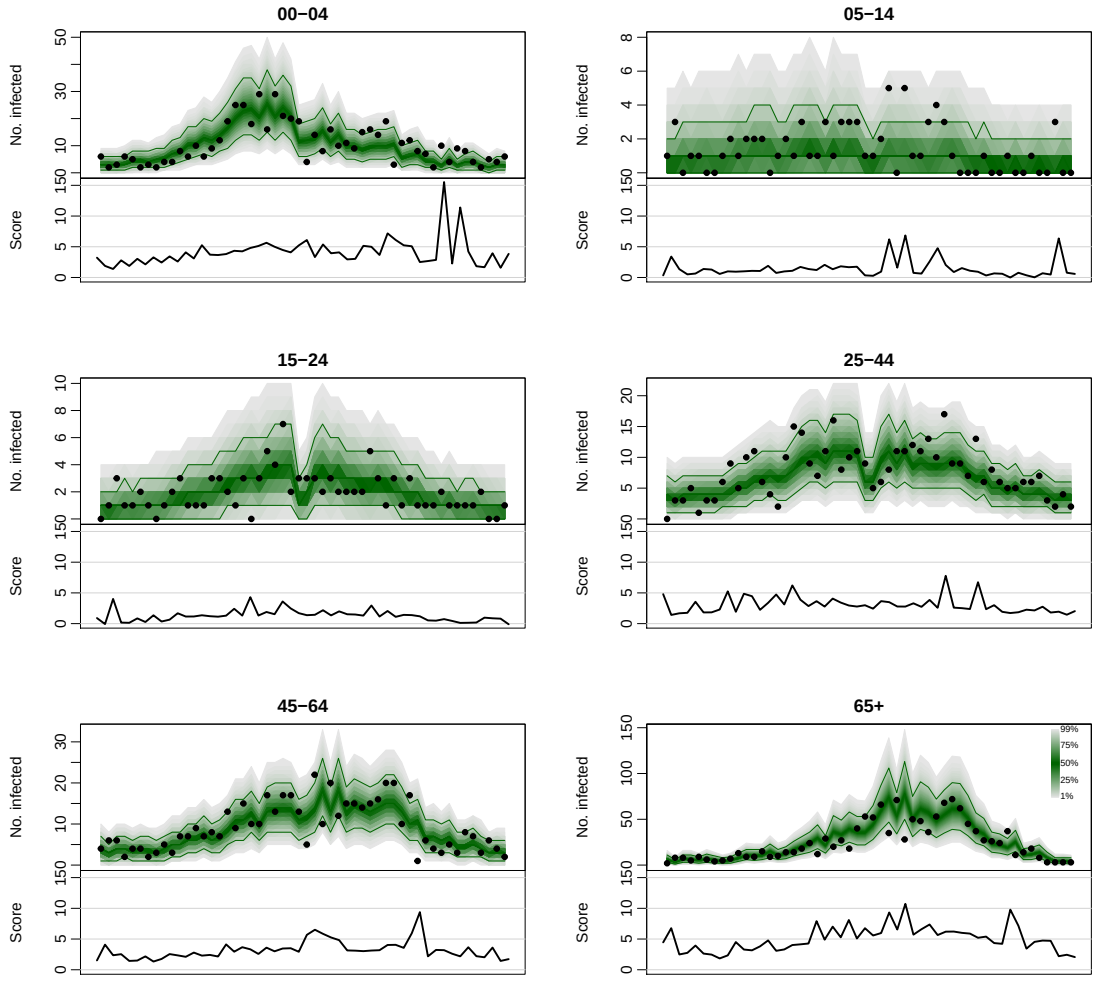

Figure A.15: **Predictive distribution and sDSS for `Reciprocal` base model** for one-step-ahead-predictions of test year 2015 of BNV dataset.

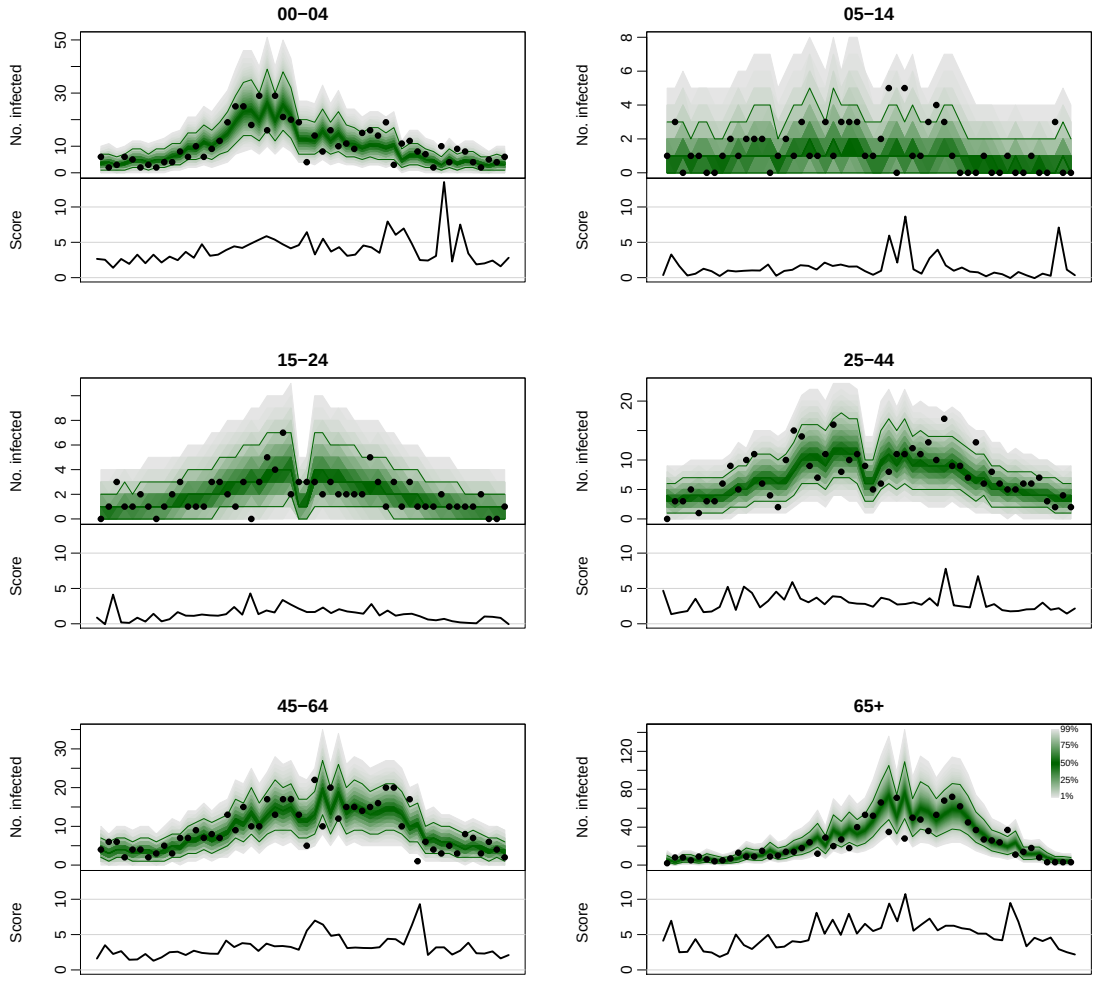

Figure A.16: **Predictive distribution and sDSS for `Power-Adjusted` base model** for one-step-ahead-predictions of test year 2015 of BNV dataset.

Bibliography

Ahmed, S. M., Lopman, B. A. and Levy, K. (2013). A Systematic Review and Meta-Analysis of the Global Seasonality of Norovirus. *PLOS ONE* **8** e75922.

Bergmeir, C., Hyndman, R. and Koo, B. (2017). A note on the validity of cross-validation for evaluating autoregressive time series prediction. *Computational Statistics & Data Analysis* **120**.

Bernard, H., Höhne, M., Niendorf, S., Altmann, D. and Stark, K. (2014). Epidemiology of norovirus gastroenteritis in Germany 2001-2009: eight seasons of routine surveillance. *Epidemiology and Infection* **142** 63–74.

Bracher, J. and Held, L. (2017). Periodically stationary multivariate autoregressive models. *arXiv:1707.04635 [stat]* ArXiv: 1707.04635.

Buckland, S., Burnham, K. and Augustin, N. (1997). Model Selection: An Integral Part of Inference. *Biometrics* **53**.

Burnham, K. P., Anderson, D. R. and White, G. C. (1994). Evaluation of the Kullback-Leibler Discrepancy for Model Selection in Open Population Capture-Recapture Models. *Biometrical Journal* **36** 299–315.

Claeskens, G. and Hjort, N. L. (2008). *Model Selection and Model Averaging*. Cambridge University Press. Google-Books-ID: IJA9nwEACAAJ.

Clemen, R. T. (1989). Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting* **5** 559–583.

Czado, C., Gneiting, T. and Held, L. (2009). Predictive Model Assessment for Count Data. *Biometrics* **65** 1254–1261.

Geweke, J. and Amisano, G. (2012). Prediction with Misspecified Models. *American Economic Review* **102** 482–486.

Gneiting, T., Balabdaoui, F. and Raftery, A. E. (2007). Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **69** 243–268.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association* **102** 359–378.

Gneiting, T., Stanberry, L. I., Grimit, E. P., Held, L. and Johnson, N. A. (2008). Assessing probabilistic forecasts of multivariate quantities, with an application to ensemble predictions of surface winds. *TEST* **17** 211.

Held, L., Höhle, M. and Hofmann, M. (2005). A statistical framework for the analysis of multivariate infectious disease surveillance counts. *Statistical Modelling: An International Journal* **5** 187–199.

Held, L., Hofmann, M., Höhle, M. and Schmid, V. (2006). A two-component model for counts of infectious diseases. *Biostatistics* **7** 422–37.

Held, L. and Meyer, S. (n.d.). *Handbook of Infectious Disease Data Analysis*.

Held, L., Meyer, S. and Bracher, J. (2017). Probabilistic forecasting in infectious disease epidemiology: the 13th Armitage lecture. *Statistics in Medicine* **36** 3443–3460.

Held, L. and Paul, M. (2012). Modeling seasonality in space-time infectious disease surveillance data. *Biometrical Journal* **54** 824–843.

Kass, R. E. and Raftery, A. E. (1995). Bayes Factors. *Journal of the American Statistical Association* **90** 773–795.

Koren, Y. (2009). The BellKor solution to the Netflix Grand Prize .

Makridakis, S. and Hibon, M. (2000). The M3-Competition: results, conclusions and implications. *International Journal of Forecasting* **16** 451–476.

Merz, C. J. and Pazzani, M. J. (1999). A principal components approach to combining regression estimates. *Machine Learning* **36** 9–32.

Meyer, S. and Held, L. (2014). Power-law models for infectious disease spread. *The Annals of Applied Statistics* **8** 1612–1639. ArXiv: 1308.5115.

Meyer, S. and Held, L. (2017). Incorporating social contact data in spatio-temporal models for infectious disease spread. *Biostatistics* **18** 338–351.

Meyer, S., Held, L. and Höhle, M. (2017). Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package surveillance. *Journal of Statistical Software* **77**. ArXiv: 1411.0416.

Monto, A. S., Gravenstein, S., Elliott, M., Colopy, M. and Schweinle, J. (2000). Clinical signs and symptoms predicting influenza infection. *Archives of Internal Medicine* **160** 3243–3247.

Mossong, J., Hens, N., Jit, M., Beutels, P., Auranen, K., Mikolajczyk, R., Massari, M., Salmaso, S., Tomba, G. S., Wallinga, J., Heijne, J., Sadkowska-Todys, M., Rosinska, M. and Edmunds, W. J. (2008). Social Contacts and Mixing Patterns Relevant to the Spread of Infectious Diseases. *PLOS Medicine* **5** e74.

Newbold, P. and Granger, C. W. J. (1974). Experience with Forecasting Univariate Time Series and the Combination of Forecasts. *Journal of the Royal Statistical Society. Series A (General)* **137** 131–165.

Nishii, R. (1984). Asymptotic Properties of Criteria for Selection of Variables in Multiple Regression. *The Annals of Statistics* **12** 758–765.

Paul, M. and Held, L. (2011). Predictive assessment of a non-linear random effects model for multivariate time series of infectious disease counts. *Statistics in Medicine* **30** 1118–1136.

Paul, M., Held, L. and Toschke, A. M. (2008). Multivariate modelling of infectious disease surveillance data. *Statistics in Medicine* **27** 6250–6267.

R. Hyndman, R. (2016). Cross-validation for time series | Rob J Hyndman. Last accessed 2 April 2018.
URL `https://robjhyndman.com/hyndsight/tscv/`

R. Hyndman, R. (2018). A brief history of time series forecasting competitions. Last accessed 23 April 2018.
URL `https://robjhyndman.com/hyndsight/forecasting-competitions/`

Raftery, A. E., Gneiting, T., Balabdaoui, F. and Polakowski, M. (2005). Using Bayesian Model Averaging to Calibrate Forecast Ensembles. *Monthly Weather Review* **133** 1155–1174.

Reeve, K. (2017). *Spatio-temporal forecasting and infectious disease count data*. Master's thesis, University of Zurich.

Sakamoto, Y., Ishiguro, M. and Kitagawa, G. (1986). *Akaike information criterion statistics*. Tokyo : KTK Scientific Publishers ; Dordrecht ; Boston : D. Reidel ; Hingham, MA : Sold and distributed in the U.S.A. and Canada by Kluwer Academic Publishers.

Schwarz, G. (1978). Estimating the Dimension of a Model. *The Annals of Statistics* **6** 461–464.

Sloughter, J. M., Gneiting, T. and Raftery, A. E. (2010). Probabilistic Wind Speed Forecasting Using Ensembles and Bayesian Model Averaging. *Journal of the American Statistical Association* **105** 25–35.

Töscher, A. and Jahrer, M. (2009). The BigChaos Solution to the Netflix Grand Prize .

US Department of Commerce, N. (n.d.). Dengue Forecasting. Last accessed 29 March 2018.
URL `http://dengueforecasting.noaa.gov/`

van der Laan, M., Polley, E. and Hubbard, A. (2007). Super Learner. *U.C. Berkeley Division of Biostatistics Working Paper Series* .

Vrieze, S. I. (2012). Model selection and psychological theory: A discussion of the differences between the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC). *Psychological Methods* **17** 228–243.

Wallinga, J., Teunis, P. and Kretzschmar, M. (2006). Using Data on Social Contacts to Estimate Age-specific Transmission Parameters for Respiratory-spread Infectious Agents. *American Journal of Epidemiology* **164** 936–944.

WHO (2007). WHO | WHO Global Epidemiological Surveillance Standards for Influenza. Last accessed 29 March 2018.
URL `http://www.who.int/influenza/resources/documents/influenza_surveillance_manual/en/`

WHO (2017). WHO | Dengue and severe dengue. Last accessed 29 March 2018.
URL `http://www.who.int/mediacentre/factsheets/fs117/en/`

Winkler, R. L., Muñoz, J., Cervera, J. L., Bernardo, J. M., Blattenberger, G., Kadane, J. B., Lindley, D. V., Murphy, A. H., Oliver, R. M. and Ríos-Insua, D. (1996). Scoring rules and the evaluation of probabilities. *Test* **5** 1–60.

Wolpert, D. H. (1992). Stacked Generalization. *Neural Networks* **5** 241–259.

Yamana, T. K., Kandula, S. and Shaman, J. (2016). Superensemble forecasts of dengue outbreaks. *Journal of The Royal Society Interface* **13** 20160410.

Yao, Y., Vehtari, A., Simpson, D. and Gelman, A. (2017). Using stacking to average Bayesian predictive distributions. *arXiv:1704.02030 [stat]* ArXiv: 1704.02030.

Zou, H. and Yang, Y. (2004). Combining time series models for forecasting. *International Journal of Forecasting* **20** 69–84.