Estimating the non-reporting rate of animals used in preclinical research using meta-analytic approaches

Master Thesis in Biostatistics

by

Jonas Füglistaler

13-936-240

supervised by

Prof. Dr. Reinhard Furrer

Version February 28, 2024

Contents

Preface v

Abstract vii

1 Introduction 1
- 1.1 Evidence Aggregation in Preclinical Research . . . 1
- 1.2 Publication Bias in the Context of Animal Research . . . 2
- 1.3 Methods to Estimate the Non-Reporting Rate . . . 3

2 Methods 5
- 2.1 Simulation of Methods to Quantify Publication Bias . . . 5
  - 2.1.1 Simulation Setup . . . 5
  - 2.1.2 Simulate Step . . . 7
  - 2.1.3 Aggregate Step . . . 8
  - 2.1.4 Censor Step . . . 11
  - 2.1.5 Estimate Step . . . 14
  - 2.1.6 Simulation Implementation . . . 19
- 2.2 Tracking of Study Requests to Publication . . . 19
  - 2.2.1 Procedure for Obtaining and Documenting License Applications . . . 19
  - 2.2.2 Overview of Process to Track Study Requests . . . 20
  - 2.2.3 Data Collection and Curation . . . 21
  - 2.2.4 Pubmed Query Definition . . . 22
  - 2.2.5 Keyword Matching . . . 23
  - 2.2.6 Manual Screening . . . 24
  - 2.2.7 Endpoint Calculation . . . 25
  - 2.2.8 Tracking Implementation . . . 25

3 Results 27
- 3.1 Simulation Results . . . 27
  - 3.1.1 Censoring Results . . . 27
  - 3.1.2 Estimation Results . . . 29
  - 3.1.3 Comparison of Estimation Methods . . . 32

3.2 Results of the Study Request Tracking 33
3.2.1 Overview of Tracking Request Results 33
3.2.2 Automatic Query Results 34
3.2.3 Manual Screening Results 36
3.2.4 Tracking Endpoint Results 36

4 Discussion **41**
4.1 Evaluation of Methods to Estimate Non-reporting Rates 41
4.2 Appraisal of the Process for Tracking Study Requests 43

A Additional Figures **47**

B Improvements for Tracking Study Request Process **49**

C Computational Details **51**

D Code **53**

Reference **71**

Preface

The ultimate goal of any scientific discipline is to create knowledge by the systematic examination of their respective environment. Natural scientists try to observe regularities in nature, whereas social scientists try to uncover patterns in the social behavior of humans. Scientific results are made accessible through publications and the generation of knowledge proceeds through the distillation of the reported data. Consolidation starts in review articles and proceeds to textbooks until findings become accepted or refuted. When data remain unpublished which differ substantially from published work, conclusions may not adequately reflect the underlying effects being described. The existence of such "publication bias" is problematic, since the success, reputation and integrity of science rests on the unbiasedness of its results.

The pursuit of knowledge is a subject close to my heart. I consider the recent large-scale authentication of false claims in the Western world as a frightening development. If we cannot agree on the basic facts on which our civilisation is built, any interaction with other-minded members of society will be frustrating. The empirical sciences can make the claim to facts rather than opinions because they provide the corresponding evidence and systematic analysis thereof. Research's findings are therefore the best means we have to understand the surrounding world and should be valued as verifiable facts within the tested paradigm. However, the repeated falsification of results and challenges of replicating data characterize the ongoing "reproducibility crisis". This crisis not only jeopardizes the trust that the public places in science but also squanders valuable resources, including animals. As I near the culmination of my second Master's degree, I am pleased to address these overarching issues within the realm of biomedical research.

A major influence on my change of academic direction had Servan Grüninger. Therefore, I would like to express my sincere gratitude to you first. You not only motivated me to become a better scientist but showed me repeatedly where the necessary limits of research in the societal context are. Thank you for leading me to this thesis and for the continuous suppport therein. Next, I would like to express my profound appreciation to Prof. Dr. Reinhard Furrer for allowing me to carry out this thesis in your group. In addition, I want to convey my sincere esteem for your meticulous corrections made to this report. All persons involved in this project are also grateful to the very helpful staff of the FSVO: Dr. Otto Maissen, Dr. Cristian Berce and Dr. Sara Schärrer. Thank you all for making this research possible, your assistance and the valuable contributions. A special thanks goes to Annouk Petitpierre; you enabled me to have a jump-start into this project by sharing findings of your own thesis. I would also like to thank my employer Genedata, especially my line manager Dr. Matthias Fassler, who made the adoption of each semester's schedule possible next to my employment. To Dominique Lorgé: thank you for being an indispensable cornerstone in my life. "Last but not least, I want to thank me - for believing in me, for doing all this hard work, for having no days off and never quittin" (Snoop Dogg, 2019).

Jonas Füglistaler
December 2023

Abstract

Evidence aggregation plays a crucial role in supporting the translation of preclinical therapies from animal models to human trials. However, the efficacy of meta-analyses is impeded by the prevalent non-reporting of experiments - a consequence of i.a., misguided incentives within the publication process. While prospective countermeasures such as pre-registration of studies exist, their adoption within animal research has been slow. It is therefore essential to also evaluate and develop retrospective valuation methods. In this work, we investigated the performance of statistical methods capable of estimating publication bias, analytically and by simulation: the trim-and-fill, Copas selection, and the truncated normal distribution model. All methods failed to accurately capture the number of omitted studies in scenarios reflecting the publication landscape in animal research, i.e., moderate to heavy censoring. To attain an unbiased and assumption-free estimate of the publication rate in Switzerland, we tracked a sample of 36 animal experiment applications until publication. In this pilot study, we employed an automatic process leveraging the PubMed E-Utilities and a keyword matching algorithm. The automatic process efficiently reduced the pool of publications to 1.42% of the crude hits based on the authorship information on the applications. Endpoint calculations revealed that only a small fraction of requested and used animals are recovered in publications. In conclusion, this work demonstrates the inadequacies of all tested statistical tools to adjust pooled estimates in preclinical research. Moreover, it serves as a starting point towards assessing statistics on animal research in Switzerland, which could provide insights to improve the preclinical scientific rigor.

Chapter 1

Introduction

1.1 Evidence Aggregation in Preclinical Research

Evidence-based medicine requires that medical knowledge is generated on the basis of the best available evidence published in literature. Existing results are often summarized in the form of systematic reviews which encompass an attempt to identify, appraise and aggregate all empirical evidence to answer a specific research question (Jahan et al., 2016). The need to synthesize research arises not only in the context of clinical trials, but is also important for the preceding preclinical studies. Preclinical studies for a specific therapy must include animal experiments due to current limitations of alternative methods, as mandated by law. Alternative methods, including cell culture experiments and computer models, are as of yet not able to fully model the complex interaction of the entire organ, circulatory and nervous system which is needed to accurately predict side-effects of new drug candidates (Garattini and Grignaschi, 2017). At this stage, systematic evidence aggregation is thus important to support translation of the most promising preclinical therapies from animal models into trials on humans (ter Horst et al., 2023).

In the course of conducting a systematic review, studies addressing the research question of interest are selected, summarized and conclusions disseminated (Higgins et al., 2019). A potential question of interest is, if some "*Therapy X*" has a true beneficial treatment effect greater than a physiologically relevant difference on "*Disease Y*" in a mouse model. First, all relevant studies are identified. Next, studies are selected that fulfill the pre-specified eligibility criteria, e.g., only *in vivo* intervention studies with randomized and blinded group allocation should be included, as the most robust research design for the assessment (Laajala et al., 2016). Finally, the summary can be done purely according to a narrative synthesis approach or - additionally - utilizing meta-analytic tools. While narrative synthesis involves the combination of multiple studies using words, meta-analysis is the statistical combination thereof.

Effect measure pooling strategies in meta-analysis allow the quantitative assessment of the cumulative impact of an intervention. Broadly, there are two primary statistical model classes to account for the variability across studies: fixed-effect models and random-effect models. While fixed-effect models assume that the observed effect size variation between studies is purely explained by sampling error, random-effect models portray the variation additionally under the assumption of a genuine difference across studies (Harrer, Mathias and Cuijpers, Pim and Furukawa Toshi A and Ebert, David D, 2021). When performed correctly, a meta-analytic combination of published results improves the precision of estimates compared with single studies and allows an overall conclusion (Sena et al., 2010). For example, if there is a generalizable effect of *Therapy X* on *Disease Y*. In essence, the goal of the systematic review is thus to confirm or disprove a hypothesis already posed in the original articles (Biondi-Zoccai et al., 2011). As with individual confirmatory experiments, meta-analysis answers questions mathematically with

statistical tests. They allow inference on whether the aggregated evidence is sufficient to reject the null hypothesis (H_0), representing the status quo position of a given test statistic (Frick, 1995; Neyman and Pearson, 1933), i.e., of no effect. Contrarily, the alternative hypothesis H_A is the researcher's prediction that *Therapy X* has indeed an effect on *Disease Y*.

In modern clinical practice, systematic reviews including meta-analytic results and hypothesis testing play an important role for health decision making. For instance, reviews of corticosteroids for women at risk of giving birth prematurely (Roberts et al., 2017), treatments for macular degeneration (Moja et al., 2014) and tranexamic acid for trauma patients with bleeding (Coats et al., 2004) have demonstrated the benefits of these life-changing interventions and influenced clinical practice around the world (Higgins and Green, 2023). But also in the preclinical setting have meta-analyses become increasingly common. For example, meta-analyses have been used to assess the overall efficacy of preclinical interventions in neurological diseases. Furthermore, they have been successfully applied to evaluate factors influencing the internal and external validity of methodologies in animal research. In this way, they provided valuable insights into the causes of translational successes and failures and thus enable us to improve the rigor of the scientific conduct also at the preclinical stage (Sena et al., 2010).

1.2 Publication Bias in the Context of Animal Research

Even though meta-analysis has become a powerful tool in biomedical research, statistical analysis of pooled data has some pitfalls, which can lead to a misinterpretation of the available evidence. Among other things, there can be bias of included studies ("garbage in, garbage out") (Page et al., 2018), methodological mistakes, and/or publication bias. Publication bias in a broader sense describes the selective unavailability of research findings and leads to a distortion of the scientific knowledge base. The reasons for this unavailability can be diverse, ranging from low newsworthiness (Auspurg and Hinz, 2011) to research funding difficulties (Dickersin, 1990). In a narrower sense, publication bias describes the selective (non)reporting of studies based on the nature and direction of their results (Nair, 2019). In the narrow definition, publication of positive results that confirm the researcher's preferred outcome is more likely (Dickersin, 1990). This threatens the validity of the pooled, meta-analytic results from a body of literature, since statistical significant results and those in the desired direction are (more likely) to be published. Thus, the available results are overoptimistic, and pooling leads to an inflation of the estimated effect size. In our example: the overall treatment effect of *Therapy X* on *Disease Y* is overestimated in the mouse model, which could physically harm patients in the upcoming clinical trials (Bahnam et al., 2022).

Apart from effect size exaggeration, publication bias can also severely affect the evolution of knowledge, especially if statistical significance is used as a decision rule for publication. Significance is governed by the infamous p-value, which is the probability that an effect would be equal to or more extreme than its observed value under H_0 of a specific model (Wasserstein and Lazar, 2016). In empirical sciences, a 5%-significance threshold is commonly used. Only estimates with a p-value lower than 5% are then designated 'significant'. Now, in a scenario where all studies with a p-values larger than 0.05 are censored, no results in favor of H_0 will be published and theories are confirmed irrespective of truth (Ferguson and Heene, 2012), due to the possibility for false positive results (Type I errors). In addition, science is in this case also inefficient, since sound studies that would add new knowledge to the literature but contain null findings remain unpublished and might get unnecessarily repeated. As a consequence of the latter, substantial loss of time and resources can be side-effects of publication bias. As resources in the context of animal research include living beings, an additional ethical dilemma arises - with respect to the

animals involved in experiments - and the moral responsibility to counteract publication bias is reinforced. Moreover, when knowledge generated from animal experiments is systematically unavailable, the scientific publication process is also in conflict with Swiss legislation. According to Swiss animal testing law, every research request for a project involving animals must demonstrate that the desired scientific insights from the planned study cannot be achieved by any other means than animal experiments. However, any unpublished, scientifically valid result involving animals could theoretically lead to an unnecessary repetition of the experiment. Thus, a procedure that supports this misconduct is - in principle - unlawful.

By now, it should be clear that we need ways to deal with publication bias - from a scientific and ethical perspective. Broadly, there are two method classes on how to deal with it: preventing publication bias for future studies prospectively and evaluating its extent retrospectively. Prospective prevention includes pre-registration of studies to ensure a publication regardless of the nature of the result. It is the act of registering a research protocol before conducting the experiment. Researchers are asked to detail their hypotheses, experimental strategy and analytical plans before the study begins. The aim is to prevent teams from cherry-picking desirable findings and to provide the scientific community with an opportunity to learn about experiments that would otherwise remain unpublished - i.e., to counteract publication bias. Recently, preregistration is rapidly increasing in popularity (Simmons et al., 2021), but is still only required by regulatory authorities for clinical trials (Baker, Monya, 2019). Other proposed solutions for the prevention of selective reporting in the future include to raise researcher's awareness, change the rewarding scheme (to be based on diligence rather than results), limit the rate of publications required, promote replication studies plus transparency, and update the report format in journals (Klein, 2022). There is growing support that these recommendations and the use of trial registries will improve animal research (Baker, Monya, 2019).

1.3 Methods to Estimate the Non-Reporting Rate

Currently, selective reporting is still pervasive and the uptake of publication bias prevention strategies has been slow in animal research (Baker, Monya, 2019). On top of that, some results might always remain hidden and unavailable. Therefore, it is of great importance to also find methods for the retrospective assessment of publication bias. One way to quantify publication bias is to construct an estimate of the non-reporting rate starting from a list of publications. In the literature, various statistical methods to either detect publication bias or adjust pooled estimates are available. Methods to detect publication bias include graphical methods (e.g., funnel-plot based, normal quantile plot adaptations, weighted box plot), regression tests (e.g., Egger's regression), rank correlations tests (e.g., Begg's test), meta-likelihood approaches, truncated sampling, excess significance tests and more, as summarized by Mueller et al. (2016). However, all listed methods focus on the detection of bias in the reported studies and do not provide an estimation of the number of omitted studies. Hence, they cannot deliver an estimate of the non-reporting rate. On the other hand, there are several statistical methods to adjust publication bias, i.e., correct the exaggerated, pooled estimate. They include methods to adjust for funnel plot asymmetry (e.g., trim-and-fill), regression-based methods, selection models (e.g., Copas), p-curve analysis, capture-recapture, publication probability weighted methods, truncated distributions (e.g., normal), fail-safe-N, Monte-Carlo estimations and more. However, there is no consensus which methods should be used under which circumstances (Mueller et al., 2016). Furthermore, many do not correct the overall effect by means of an estimate of omitted studies and are therefore also not useful to quantify publication bias. From the list above, only the classes including the trim-and-fill method, the Copas selection model and the truncated normal distribution method were found to provide a direct estimation of the non-reporting rate. The first major goal of this thesis is thus to explore these methods using both theoretical arguments

and simulations resembling animal research.

The second way to estimate publication bias retrospectively is to track publications from study requests, if they are available. For instance, a study from the Netherlands followed a sample of 67 animal experimentation projects from approval to publication. The result: only 26% of the animals used in the experiments were recovered in the publications while 60% (40/67) of the approved applications resulted in at least one scientific publication (van der Naald et al., 2020). Thus, the true numbers at the animal-level are even worse than the 2012 Dutch survey found, which estimated that 50% of animal experiments are not published (ter Riet et al., 2012). A similar situation was found in Germany, where a sample of 210 archived animal study protocols were tracked for publication. The overall publication rate was 67%, corresponding to a non-reporting rate of 33% of all animal studies (Wieschowski et al., 2019). Studies like these are important, since they theoretically provide an unbiased and assumption-free estimate of the numbers of publications resulting from all conducted animal experiments. In addition, the legal and organizational prerequisites to conduct such a study nationwide would already be in place in Switzerland, as mentioned above. However, there are no such evaluations available yet, to the best of our knowledge. The second major aim of this thesis is thus to propose a process for Switzerland that allows to estimate non-reporting rates of animals by tracking a set of study requests to the corresponding literature list.

Chapter 2

Methods

2.1 Simulation of Methods to Quantify Publication Bias

2.1.1 Simulation Setup

To evaluate the performance of various methods for estimating non-reporting rates resulting from different mechanisms elucidating publication bias (i.e., censoring), Monte Carlo simulation techniques were employed. Monte Carlo simulations enable the manipulation of known parameters in a versatile manner and facilitate the design of conditions that emulate the animal research literature. Furthermore, Monte Carlo simulations allow for the identification of simulated studies that are affected by publication bias and those that are not, enabling an examination of estimator performance. The complete simulation is composed of four sequential steps:

1. The data generating process ("*Simulate*") on observation-level.
2. The aggregation ("*Aggregate*") of observations into study-level metadata.
3. The censoring step ("*Censor*"), i.e., the introduction of publication bias.
4. The estimation ("*Estimate*") of the non-reporting rate and measure of performance.

The flowchart in Figure 2.1 represents the sequential execution of the simulation algorithm's steps. The simulation conditions in the current setup were varied at the *Simulate*, *Censor* and *Estimate* steps, visualized by rhombus shapes in the figure. The *Aggregate* step does not require any input in the current implementation. For each simulation, one data frame is generated and processed, visualized by the stacked document symbols as output of the first three steps. When *Simulate* is initiated, observations are generated representing the raw data of all conducted experiments. In this step, the input parameter β is varied that represents the true underlying effect researchers try to uncover. The *Aggregate* step then imitates the condensing of raw data into summary statistics, i.e., generates all theoretically available study-level results. Next, the *Censor* step introduces publication bias by first selecting a subset of studies based on a censoring mechanism and subsequent "publishing" of a specified proportion thereof, defined by the publication probability π. The true non-reporting rate (Ground Truth, "GT") is then assessed by comparing the published to all theoretically available studies. Finally, the non-reporting rate is estimated during the *Estimate* step by the selected statistical estimation method and the bias is calculated by comparing the estimate to the GT.

The whole simulation framework is modular to enable combinatorial interplay of the four steps. All tested simulation conditions are listed in Table 2.1. The data-generating process of the *Simulate* step will be described in detail in Section 2.1.2. In brief, a linear relationship of the outcome variable Y from the independent variable X is modeled by specifying the input parameter β, representing the true effect size or slope. The random error ϵ and independent variable X are sampled from normal distributions with means zero and fixed variances σ_e^2 and

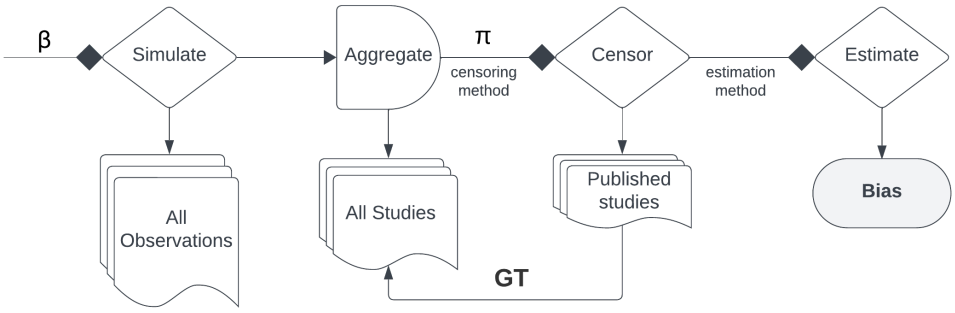

Figure 2.1: Simulation flow chart. Rhombus shapes indicate input-dependent decision points. For each simulation, all theoretically available observations are generated by the *Simulate* step depending on the input parameter β describing the true underlying effect. Observations are condensed to study-level results (All Studies) during *Aggregate*. Which studies are published depends on the selected censoring method and the publication probability π during *Censor*. The bias of the selected estimation method is calculated by comparing the estimated non-reporting rate back to the Ground Truth (GT) proportion of published studies.

σ_x^2, respectively. As a result, β can be transformed into Pearson's correlation coefficient ρ (seen in equation (2.2)) to include the whole range of meaningful effect sizes spanning correlations from -0.7 to 0.7. From this range, 11 values were tested for ρ. In the current setup, the two available censoring methods either select non-significant studies (p-values > 0.05) or studies with negative estimated effect sizes ($\widehat{\beta} < 0$) to be prone for censoring, described in Section 2.1.4. The publication probability π then dictates which proportion of the selected studies is to be published or - conversely - which part is dropped. Also for π, 11 equidistant values were tested spanning the whole range of allowed probability values. Both censorings were combined with three different estimation methods: The trim-and-fill algorithm, Copas selection model and the tuncated normal distribution, described in Section 2.1.5. Conclusively, a grand total of 6 $\times$ 121 conditions were tested per simulation run.

Table 2.1: Overview of simulation conditions. The expression after "R:" indicates the code used to generate the tested parameter space. N assigns the total number of conditions per simulation run for a specific variable.

Variable	**Values**	**Functional Form**	N
Simulate			
True effect β $\rightarrow$ Correlation ρ	R: `seq(-5, 5, by = 1)` $\rightarrow \in$ [-0.7, 0.7]	$Y = \beta X + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma_e^2)$, $X \sim \mathcal{N}(0, \sigma_x^2)$ $\rightarrow \rho = \beta\sigma_x/\sigma_y$	11
Censor			
Censoring method	$\in$ {"*trim*", "*ppr*"}	Selected Studies ("SS"): *trim*: $\widehat{\beta} < 0$ *ppr*: $p > 0.05$	2
Publ. probability π	R: `seq(0, 1, by = 0.1)`	R: `drop = sample(SS, (1-`π`)·length(SS))`	11
Estimate			
Estimation method	$\in$ {"*fill*", "*copas*", "*trunc*"}	*fill*: trim-and-fill algorithm *copas*: Copas selection model *trunc*: Truncated normal distribution	3
		Total:	6×121

2.1.2 Simulate Step

The *Simulate* step generates observations for each study for a given dependency of a hypothetical experiment, inspired by Schneck (2017). An example was selected where an imaginary continuous health outcome variable Y is described by a continuous treatment variable X. Single observations of the dependent variable Y were generated by the linear relationship for each of the s studies:

$$Y_i = \beta X_i + \epsilon_i, \quad i = 1, \ldots, n, \tag{2.1}$$

with the error term $\epsilon \sim \mathcal{N}(0, \sigma_e^2)$, the independent variable $X \sim \mathcal{N}(0, \sigma_x^2)$ and n equal to the number of observations in the study. The resulting outcome distribution is $Y \sim \mathcal{N}(0, \beta^2\sigma_x^2 + \sigma_e^2)$ and conditional outcome distribution $Y|x \sim \mathcal{N}(\beta x, \sigma_e^2)$. The value of the slope coefficient β equals to the expected increase of the dependent variable Y on average when the independent variable X is increased by one unit, i.e., it defines the true underlying treatment effect. Even though fully linear dose-response relationships are scarce in biomedical research due to the inherent saturation of biological systems with increasing effector concentration, they are still valuable for making predictions for specific dose ranges and selected experiments. For example, to predict *in vivo* cancer induction upon radiation at low doses (Laurier et al., 2023). Furthermore, the linear system serves as a great starting point to mathematically describe the mechanism elucidating publication bias. In our simple linear case with the dependencies from equation (2.1), β can be transformed into the Pearson correlation coefficient for a population, derived and calculated by:

$$\rho = \frac{\mathrm{Cov}(X,Y)}{\sigma_x\sigma_y} = \frac{\mathrm{Cov}(X,\beta X+\epsilon)}{\sigma_x\sigma_y} = \frac{\mathrm{Cov}(X,\beta X)+\mathrm{Cov}(X,\epsilon)}{\sigma_x\sigma_y} = \frac{\beta\,\mathrm{Cov}(X,X)}{\sigma_x\sigma_y} = \beta\frac{\sigma_x}{\sigma_y}. \tag{2.2}$$

That way, the desired effect of the treatment to the outcome can be simulated by specifying ρ through β, σ_x and $\sigma_y = \sqrt{\beta^2\sigma_x^2 + \sigma_e^2}$. Due to the Cauchy-Schwartz inequality (Maligranda, 1998), correlation is bounded between -1 and 1. For β approaching plus/minus infinity, correlation will become plus/minus 1. In addition, ρ also approaches plus/minus 1, if σ_e/σ_x strives for zero, as seen by:

$$\beta\frac{\sigma_x}{\sigma_y} = \mathrm{sgn}(\beta)\cdot\left(\sqrt{1+\frac{\sigma_e^2}{\beta^2\sigma_x^2}}\right)^{-1}, \tag{2.3}$$

where, $\mathrm{sgn}(\cdot)$ is the signum function; $\mathrm{sgn}(x) = 0$ if $x = 0$ or $-1/+1$ if x is smaller/larger than zero, respectively. Equation (2.3) shows that for $\sigma_e/(\beta\sigma_x)$ approaching zero, ρ strives to plus/minus 1. As an example, fixed $\beta = 1.5$, $\sigma_x = 2$ and $\sigma_y = 10$ yield $\rho = 0.3$, i.e., as $\rho = \beta \cdot 0.2$. The Pearson correlation coefficient can further be interpreted as representing effect sizes with small, medium and large effect sizes equal the (absolute) example values at $\rho = 0.1$, 0.3, and 0.5, respectively (Cohen, 1992). Therefore, correlations from -0.7 to 0.7 were used to include small to large effect sizes while not approaching perfect (negative) correlation.

Each of the m simulation runs results in a raw data array on observation-level depending on the number of simulated studies s, the true treatment effect β and standard deviations for the error (σ_e) and the treatment (σ_x). In the current setup, m was fixed to 10, s to 100, σ_e to 10 and σ_x to 2 - leaving only β to vary. The number of observations per study was randomly selected with `sample()` for sample sizes between 10 to 50 with replacement. The number of studies was set to 100 although on average the number of trials in a meta-analysis is usually much lower than than (e.g., median 28 studies in the meta-analysis by Elia et al., 2016). The reason to do so is that I expect a more robust behavior over simulations and because the interpretability of the non-reporting rate in percent is facilitated. For each simulation run the function generates an array with the main columns indicating the outcome realizations y assigned to their study.

2.1.3 Aggregate Step

The *Aggregate* step condenses the data sets simulated in the *Simulate* step (Section 2.1.2) on observation-level into study-level results. For each study, it calculates i.a., the empirical mean, standard error and standard deviation of the outcome observations in a meta-analytic array. More importantly, a linear model is fitted by the `lm()` function with the outcome as a function of the treatment. One of the main results is the estimate for the slope coefficient $\widehat{\beta}$ - representing the estimated treatment effect. In addition, the corresponding standard error and p-value of the study are calculated. The p-value corresponds here to the probability of observing the estimated slope coefficient $\widehat{\beta}$ or a more extreme result under H_0, as introduced in Sections 1.1 and 1.2. In other words it quantifies the evidence that there is a linear relationship between the outcome and the treatment, i.e., that the slope is unequal to zero. The standard error of the slope coefficient is a measure of precision and corresponds to the standard deviation of the parameter (Mcbratney, 2004).

Estimated Slope Coefficient

The estimated slope coefficient is calculated by the unbiased ordinary-least squared (OLS) estimator in `lm()`. The key assumptions for OLS regression are given as follows. Linearity: The relationship between the independent variables and the parameters is assumed to be linear. Independence: The observations are assumed to be independent of each other. Homoscedasticity: The error term has constant variance across all levels of the independent variables. Symmetry: the error distribution is symmetric around zero. No multicollinearity: The independent variables are not perfectly correlated with each other. Zero mean assumption: The expected value of the error terms should be zero (Poole and O'Farrell, 1971). Under these assumptions for OLS, and assuming normality of the error term, the estimators of the slope coefficient $\widehat{\beta}$ and the intercept $\widehat{\beta}_0$ are distributed as follows (adapted from Furrer, 2018), conditional on the independent variable:

$$\widehat{\boldsymbol{\beta}} = \begin{pmatrix} \widehat{\beta}_0 \\ \widehat{\beta} \end{pmatrix} \sim \mathcal{N}_2(\boldsymbol{\beta}, \sigma_e^2 \cdot (\mathbf{X}^T\mathbf{X})^{-1}), \tag{2.4}$$

with $\mathbf{X}$ being the design matrix with column one filled with ones and column two corresponding to all observations of the treatment variable X. The parameter σ_e^2 is the random error's variance. For the slope coefficient of interest of an arbitrary study, this translates to the marginal estimator distribution of:

$$\widehat{\beta} \sim \mathcal{N}(\beta, \sigma_e^2 \cdot ((\mathbf{X}^T\mathbf{X})^{-1})_{1,1}) = \mathcal{N}\left(\beta, \frac{n\sigma_e^2}{n\sum_{i=1}^n x_i^2 - (\sum_{i=1}^n x_i)^2}\right), \tag{2.5}$$

when indexing starts at zero for the intercept and with n as the number of observations per study. For the second expression in equation (2.5), the inverse of the nonsingular square matrix was solved by dividing the adjoint by its determinant. To derive an a priori approximation of the $\widehat{\beta}$ distribution that does not depend on knowing the realizations of X, we write equation (2.4) again in joint notation:

$$\widehat{\boldsymbol{\beta}} = \begin{pmatrix} \widehat{\beta}_0 \\ \widehat{\beta} \end{pmatrix} \sim \mathcal{N}_2\left(\begin{pmatrix} \beta_0 \\ \beta \end{pmatrix}, \sigma_e^2 \left((\mathbf{1}\boldsymbol{x})^\top(\mathbf{1}\boldsymbol{x})\right)^{-1}\right),$$

with $\boldsymbol{x}$ as the vector with all realizations of X. Since we sample the realizations of the treatment variable for some fixed parameter values of the distribution $X \sim \mathcal{N}(\mu, \sigma_x^2)$, the variance matrix

of $\widehat{\boldsymbol{\beta}}$ can be approximated using a delta method approach for the general case by:

$$\sigma_e^2 \left((\mathbf{1}\boldsymbol{x})^\top (\mathbf{1}\boldsymbol{x}) \right)^{-1} \approx \sigma_e^2 \begin{pmatrix} n & \mathrm{E}\left(\sum_{i=1}^n X_i\right) \\ \mathrm{E}\left(\sum_{i=1}^n X_i\right) & \mathrm{E}\left(\sum_{i=1}^n X_i^2\right) \end{pmatrix}^{-1}$$
$$= \sigma_e^2 \begin{pmatrix} n & n\mu \\ n\mu & n(\mu^2 + \sigma_x^2) \end{pmatrix}^{-1} = \frac{\sigma_e^2}{n\sigma_x^2} \begin{pmatrix} \mu^2 + \sigma_x^2 & -\mu \\ -\mu & 1 \end{pmatrix}.$$

Here, first $\mathrm{E}(X^2) = \mathrm{Var}(X) + \mathrm{E}(X)^2$ is used. On average, the variance of the slope estimate is thus equal to $\sigma_e^2/(n\sigma_x^2)$. Therefore, the expected variance of the slope coefficient is small for large n and vice versa; while being scaled by the ratio of the two variances. For instance, the values $\mu = 0$, $n = 25$, $\sigma_e^2 = 100$ and $\sigma_x^2 = 4$ result in the following approximated variance matrix:

$$\sigma^2(\widehat{\boldsymbol{\beta}}) \approx \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix}.$$

More generically, the estimator of the slope coefficient is thus approximately distributed as follows:

$$\widehat{\beta} \mathrel{\overset{\approx}{\sim}} \mathcal{N}\left(\beta, \frac{\sigma_e^2}{n\sigma_x^2}\right), \tag{2.6}$$

which is a simplification of equation (2.5) with which we can now estimate the proportion of estimated (e.g.,) negative treatment effects as:

$$P\left(\widehat{\beta} \leq 0\right) = \Phi\left(\frac{-\beta}{\sqrt{\mathrm{Var}(\widehat{\beta})}}\right) \approx \Phi\left(-\frac{\sqrt{n}\beta\sigma_x}{\sigma_e}\right), \tag{2.7}$$

without knowing the realizations of the treatment variable X. Here, $\Phi(\cdot)$ corresponds to the cumulative distribution function of the standard normal distribution. Furthermore, we can use equation (2.6) to derive an (approximate) distribution of the estimated standard error - also purely based on the input parameters, as outlined next.

Standard Error of Slope Coefficient

From equation (2.5) we know the population standard deviation of the estimated slope coefficient, which is equal to the true theoretical standard error (SE) of $\widehat{\beta}$. However, this expression does not allow to predict the distribution of $\mathrm{SE}(\widehat{\beta})$ without knowing the respective realizations of the treatment X. To draw a priori conclusions about the estimator of SE, we use the sample variance as a consistent (and even unbiased) estimator for the variance of a normal distributed variable (Held, Leonhard and Sabanés-Bové, Daniel Sabanés, 2020) together with the approximate expression from equation (2.6). For a general case with a random sample $G_1, G_2, \ldots, G_n$ of of size n from the normal distribution $\mathcal{N}(\mu, \sigma^2)$, $\bar{G} = 1/n \sum_{i=1}^n G_i$ and sample variance $S^2 = 1/(n-1) \sum_{i=1}^n (G_i - \bar{G})^2$, it can be shown that (Pennstate, 2023b):

$$\frac{(n-1)S^2}{\sigma^2} = \frac{\sum_{i=1}^n (G_i - \bar{G})^2}{\sigma^2} \sim \chi^2_{n-1}.$$

Since the distribution of $\widehat{\beta}$ is normal and its population variance is known to be approximately $\sigma_e^2/n\sigma_x^2$, its adapted sample variance $S^2(\widehat{\beta})$ can also be described by a χ^2_{n-1} distribution:

$$\frac{(n-1)S^2(\widehat{\beta})}{\sigma^2(\widehat{\beta})} \approx \frac{(n-1)S^2(\widehat{\beta})}{\frac{\sigma_e^2}{n\sigma_x^2}} \mathrel{\overset{\approx}{\sim}} \chi^2_{n-1}. \tag{2.8}$$

The probability density function of some variable $V = S^2(\widehat{\beta})$ with n degrees of freedom from equation (2.8) is thus:

$$f_V(x) = \frac{1}{2^{n/2}\Gamma(n/2)} x^{n/2-1} e^{-x/2} \quad x > 0.$$

To derive the density of a possible estimator $\widehat{\text{SE}(\widehat{\beta})}$, we use $S(\widehat{\beta})$ as a consistent estimator of $\sqrt{\text{Var}(\widehat{\beta})}$ and follow the approach from Pennstate (2023a) below. To keep notation simple(r), I will use "SE" from now on not only for the true underlying standard deviation of the coefficient but also the realizations thereof and - most importantly - its estimator. The transformation $R(y) = g(V) = \sqrt{V(x)}$ is a one-to-one transformation from $V = \{x \mid x > 0\}$ to $R = \{y \mid y > 0\}$ with inverse $V = g^{-1}(R) = R^2$ and Jacobian $dV/dR = 2R$. Therefore, the probability density function of R is derived by applying the change of variables formula:

$$\begin{aligned} f_R(y) &= f_V\left(g^{-1}(y)\right) \left|\frac{dx}{dy}\right| \\ &= \frac{1}{2^{n/2}\Gamma(n/2)} \left(y^2\right)^{n/2-1} e^{-y^2/2} |2y| \\ &= \frac{1}{2^{n/2-1}\Gamma(n/2)} y^{n-1} e^{-y^2/2} \quad y > 0. \end{aligned} \tag{2.9}$$

The probability density function in equation (2.9) is a χ-distribution with n degrees of freedom. Now, the behavior of SE can be approximated analogously to equation (2.8):

$$\frac{\sqrt{(n-1)}S(\widehat{\beta})}{\sigma(\widehat{\beta})} \approx \frac{\sqrt{(n-1)}\,\text{SE}(\widehat{\beta})}{\frac{\sigma_e}{\sqrt{n}\sigma_x}} \overset{\approx}{\sim} \chi_{n-1},$$

but following a $\chi_{(n-1)}$ distribution. Since $k = \sqrt{n(n-1)}\sigma_x/\sigma_e$ is fixed in the simulation, the distribution of $\text{SE}(\widehat{\beta})$ can thus be approximated in this way by setting $n = 30$. The latter representing the expected number of observations when sampling from a uniform distribution between 10 to 50.

The p-Value of the Observed Slope Coefficient

To quantify the evidence against the hypothesis of no relationship between X and Y, i.e., to calculate a ***p*-value** for the observed realizations, a test statistic is necessary. Since σ_e is known in the simulation, the p-value for $H_0 : \beta = 0$ and $H_A : \beta \neq 0$ of each study can be calculated as $p = \Pr(|Z| \geq |z| \mid H_0)$ with the test statistic:

$$Z = \frac{\widehat{\beta} - \beta_{H_0}}{\sqrt{\text{Var}(\widehat{\beta})}} = \frac{\widehat{\beta}}{\sigma_e\sqrt{v_{11}}} \quad \text{with } v_{11} = \left(\left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\right)_{11}.$$

From this equation we see that $Z \overset{\approx}{\sim} \mathcal{N}(0,1)$ and the p-value thus equals approximately to:

$$p \approx 2 \times (1 - \Phi(z)). \tag{2.10}$$

However, in the simulated research scenario the error of the effect is unknown. Therefore, the `lm()` function calculates the error's standard deviation with the unbiased estimator:

$$\widehat{\sigma}_e^2 = \frac{1}{n-2}\mathbf{r}^\top\mathbf{r},$$

where $\mathbf{r}$ are the residuals. The estimator $\widehat{\sigma}_e^2$ is thus itself a random variable which renders the distribution of the test statistic variable T non-normal:

$$T = \frac{\widehat{\beta} - \beta_{H_0}}{\sqrt{\widehat{\sigma}_e{}^2 v_{11}}} \sim t_{n-2}.$$

Now, p-values can be calculated with equation (2.10) replacing Z with T that follows a central t-distribution with $n-2$ degrees of freedom under H_0. When the term $\sqrt{\widehat{\sigma}_e{}^2 v_{ii}}$ is replaced with SE we can approximate values for T a priori:

$$T = \frac{\widehat{\beta} - \beta_{H_0}}{\mathrm{SE}(\widehat{\beta})} \approx t_{n-2;ncp}, \quad \text{with } ncp \approx \frac{\beta_{H_0}\sqrt{n}\sigma_x}{\sigma_e}. \tag{2.11}$$

The second parameter ncp of the distribution is the non-centrality parameter. Under H_0, the t-distribution is central, i.e., it is centered around zero and the ncp is zero. Under H_A, the values of T follow a non-central t-distribution with non-centrality parameter equal to $(\beta_{H_A} - \beta_{H_0})/\sigma(\widehat{\beta})$ which is approximately $(\beta\sqrt{n}\sigma_x)/\sigma_e$, seen in equation (2.6). Now it is easy to calculate the proportion of expected significant slope coefficients in the simulation, by power estimation:

$$\begin{aligned} \Pr(|T| > |t_{crit}|) &= F_{t:n-2;ncp}(-t_{crit}) + (1 - F_{t:n-2;ncp}(|t_{crit}|)) && (2.12)\\ &\approx (1 - F_{t:n-2;ncp}(|t_{crit}|)). && (2.13) \end{aligned}$$

Here, $F_{t:n-2}(t)$ is the cumulative distribution function of the t-distribution with $n-2$ degrees of freedom. The critical value t_{crit} is equal to the $(1-\alpha/2)$-quantile of T_{H_0}; or, equivalently, the Type II error quantile of T_{H_A}. The approximation in equation (2.13) is explained by the increasingly smaller integral from $-\infty$ to $-t_{crit}$ for increasing positive treatment effects or non-centrality parameters, correspondingly. Since the equation (2.12) captures both instances - the null and alternative hypothesis - we can calculate the expected proportions of significant slopes for both. For $H_0 : \beta = 0$, the resulting proportion of expected significant slopes is equal to the expected significance level α: $F_{t:n-2;ncp=0}(-t_{crit}) + (1 - F_{t:n-2;ncp=0}(t_{crit})) = 5\%$. For a large $\beta = 3$ under H_A, the approximation can be used: $1 - F_{t:n-2;ncp=3.6}(t_{crit}) \approx 98\%$. Thus, with increasing β, more significant results are expected. In a more generic setting, any increase of the number of observations n or the decrease of the residual standard deviation would lead to an increase in expected significant results, i.e., the power of the test.

The output of the function within the *Aggregate* step is a study-level meta-analysis data frame per simulation run. The function returns an array containing all studies ($s = 100$) with main columns: estimated slope coefficient $\widehat{\beta}$, (estimated) standard error of the slope coefficient $\mathrm{SE}(\widehat{\beta})$ and corresponding p-value of the study.

2.1.4 Censor Step

The *Censor* step drops a proportion of selected studies in the meta-analysis array generated in the *Aggregate* step (Section 2.1.3), depending on the publication probability π. Studies are selected based on a mechanism explaining the non-reporting. Methods select either negative estimates ("*trim*") or non-significant studies ("*ppr*"). It is helpful to mentally visualize the selection step in a funnel plot. The funnel plot is a scatterplot of the study's precision (usually $1/\mathrm{SE}(\widehat{\beta})$) against the estimated treatment effect ($\widehat{\beta}$), and was introduced by Light and Pillemer (1984). It can be used to check the existence of publication bias based on its symmetry. The assumption is that without publication bias, the funnel will display a symmetric, inverted funnel shape, whereas an asymmetry indicates selective reporting. Interpretation of funnel plots is often facilitated by the inclusion of diagonal lines representing the, e.g., 95% confidence limits around the estimated treatment effect. Because these lines are not strict 95% limits, they are referred to as "pseudo

95% confidence limits" (Sterne and Harbord, 2004). An example of a contour-enhanced funnel plot with potential publication bias is given in Figure 2.2. Contour-enhanced funnel plots provide contours of statistical significance (under H_0) to aid in interpreting the plot by classifying studies in terms of their respective p-value. We can see in the example that three small studies (low precision) are significant ($p < 0.01$), but only one study exists with similar precision that is not significant. For large scale studies (high precision), the distribution is less lopsided and many of the studies are close to the significance threshold. Therefore, the asymmetry present in the sample together with the biased distribution in terms of the p-value regions allow the assumption that publication bias is indeed present.

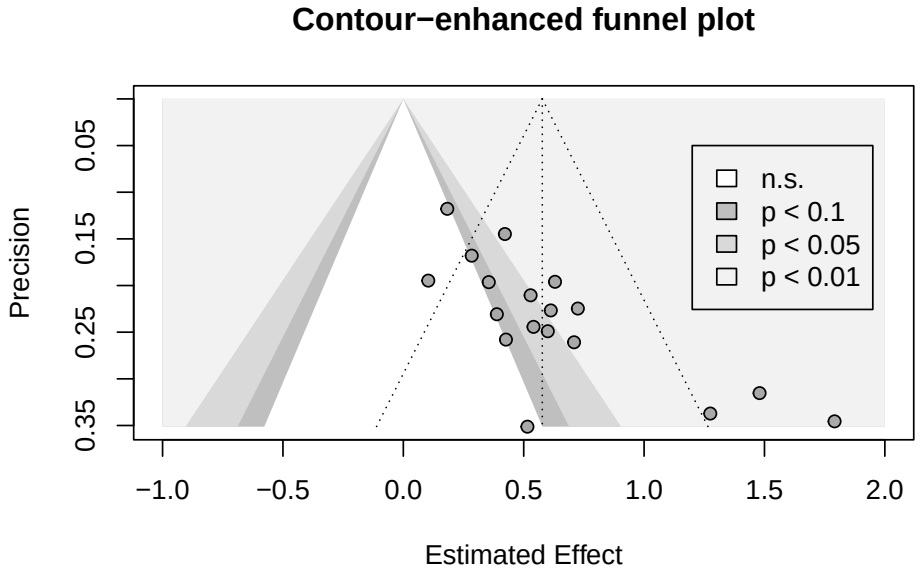

Figure 2.2: Example of a contour-enhanced funnel plot. Pseudo confidence limits of the estimated effect are indicated with dashed lines. Shaded areas represent the contours of statistical significance, i.e., the p-values for H_0 : Effect $= 0$ of the studies therein. The region containing studies with less than weak evidence according to significance testing (p-value > 0.1) is indicated in white (n.s.).

The ppr Censoring Method

The *ppr* method drops non-significant studies (p-value > 0.05) with a probability of $1 - \pi$. Graphically spoken, a linear decision boundary in the effect-versus-precision plane is introduced, whether a study is prone for censoring or not. If you draw a line starting from the origin with a slope equal to the critical value t_{crit} to a scatter plot of study-specific $\widehat{\beta}$s against their SEs - where the ratio of the axis dimensions defines the t-value, all points (studies) falling below this line will be non-significant and are eventually censored. This corresponds to a horizontal translation of the pseudo confidence interval lines in a funnel plot, until they intersect with $\widehat{\beta} = 0$ for $\mathrm{SE}(\widehat{\beta}) = 0$, i.e., number of studies approaches infinity. In the contour-enhanced funnel plot in Figure 2.2, this is identical to the contour boundaries between $p < 0.05$ and $p < 0.1$. Here, studies falling outside of this contour (farther away from the null effect) will be significant. In other words, the introduced boundary at t_{crit} is identical to a p-value cutoff at 0.05. How likely it is that the selected, significant studies are kept ("published") is defined by the publication

probability π.

Thus, the *ppr* method depends on two things: whether a particular test statistic S corresponds to a significant finding or not, and the respective publication probabilities of the dichotomized p-value space. Let $Y_1, \ldots, Y_n$ be a sample of the outcome variable Y for study j with corresponding test statistic S_j. Assuming that π is constant, a method to calculate the probability of keeping an individual study with S_j in the metadata array, i.e., "publish" it, is:

$$P(\text{Publish } S_j) = \pi + (1 - \pi) \cdot \delta(S_j(Y_1, \ldots, Y_n)) \tag{2.14}$$

$$\Rightarrow P(\text{Censor } S_j) = (1 - \pi) \cdot (1 - \delta(S_j(Y_1, \ldots, Y_n))), \tag{2.15}$$

where $\delta(\cdot)$ is any decision function taking S_j as input and returns one if "true" (adapted from Grüninger, 2019). The censor probability in equation (2.15) is the composite of the publishing probability, which can take the values:

$$P(\text{Censor } S_j) = \begin{cases} 1 - \pi, & \text{if } \delta(S_j) = 0; \\ 0, & \text{if } \delta(S_j) = 1. \end{cases}$$

In our case, we are most interested in the expected censoring proportion ("P(Censoring)") - i.e., the overall censoring probability for the whole meta-array. For *ppr* censoring, the study-specific test statistic is T_j. Thus, $\delta(\cdot)$ returns 1 if $T_j > t_{crit}$ and 0 otherwise. In addition, π_k is fixed in the simulation for a certain condition k. Leveraging the other input parameters, the distribution T_k is now also known from the equation (2.11). Therefore, we can estimate the proportion of censored studies in one meta-array without knowing a study-specific T_j for a given distribution of T_k as:

$$P(\text{Censoring} \mid T_k, \pi_k) = \sum_{j=1}^{s} P(\text{Censor } T_j)/\text{s} \tag{2.16}$$

$$\approx (1 - \pi_k) \cdot \left(1 - \sum_{j=1}^{s} \frac{\delta(T_k)}{s}\right) \tag{2.17}$$

$$\approx (1 - \pi_k) \cdot P(|T_k| \leq |t_{crit,k}|) \tag{2.18}$$

$$= (1 - \pi_k) \cdot (F_{t:n-2;ncp,k}(|t_{crit,k}|) - F_{t:n-2;ncp,k}(-t_{crit,k})) \tag{2.19}$$

$$\approx (1 - \pi_k) \cdot F_{t:n-2;ncp,k}(|t_{crit,k}|). \tag{2.20}$$

The average over all "publishing" probabilities of s studies in equation (2.16) is an expression of the proportion of non-significant studies. Plugging in equation (2.15) we arrive at the expression (2.17). Here, the left term equals to the composite of the publication probability, i.e., the "non-publication probability". The term on the right is the overall proportion of non-significant studies, i.e., of such that have a t-value smaller than $|t_{crit,k}|$ given the parameters under condition k. This proportion can be estimated by the composite of the power from equation (2.12). Conveniently, the resulting equation (2.18) now only depends on $t_{crit,k} = F^{-1}_{t:n-2}(-\alpha/2)$ and the condition-specific non-centrality parameter $ncp, k = (\beta_{H_0,k}\sqrt{n}\sigma_x)/\sigma_e$, as seen in equation (2.19). The approximation in equation (2.20) is valid for large positive values of β. In essence, the overall expected censoring rate is the product of the non-publication probability and the censoring selection probability.

For s studies, there are z significant studies ($p < 0.05$), that get published with a 100% probability, and $s - z$ non-significant studies ($p \geq 0.05$) that get published with a probability π. In this simulation, p-values were calculated with equation (2.10) replacing Z with T that follows a central t-distribution with $n - 2$ degrees of freedom. Therefore, for one specific condition k, the *ppr* censoring drops exactly $s^* = (s - z_k) \cdot (1 - \pi_k)$ studies. This process is based on relative frequencies,

which was proposed by Anouk Petitpierre, and not following the "truncated Bernoulli model" on study-level from equation (2.14). The *Censor* step under *ppr* outputs a reduced meta-analysis array with $s - s^*$ published studies with persisted values for the main columns: estimated slope coefficient $\widehat{\beta}$, (estimated) standard error of the slope coefficient $\text{SE}(\widehat{\beta})$ and corresponding p-value.

The trim Censoring Method

The *trim* method drops negative effect estimates ($\widehat{\beta} < 0$) with a probability of $1 - \pi$. Graphically spoken, a vertical, linear decision boundary whether the study is selected or not is introduced at $\widehat{\beta} = 0$ in the effect-versus-precision plane. In the funnel plot in Figure 2.2 this would be a vertical line at the null effect. How likely it is that selected studies are kept ("published") is again defined by the publication probability π. The process on study-level thus follows the one described in equation (2.14) while the decision function $\delta(\cdot)$ now only considers the estimated treatment effects $\widehat{\beta}$. Let us again derive an expression to predict the overall censoring for one β_k purely based on the input parameters:

$$P(\text{Censoring} \mid \widehat{\beta}_k, \pi_k) = \sum_{j=1}^{s} P(\text{Censor } \widehat{\beta}_j)/\text{s} \tag{2.21}$$

$$\approx (1 - \pi_k) \cdot \left(1 - \sum_{j=1}^{s} \frac{\delta(\widehat{\beta}_k)}{s}\right) \tag{2.22}$$

$$\approx (1 - \pi_k) \cdot P(\widehat{\beta}_k < 0) \tag{2.23}$$

$$\approx (1 - \pi_k) \cdot \Phi\left(-\frac{\sqrt{n}\beta\sigma_x}{\sigma_e}\right). \tag{2.24}$$

The calculations from equations (2.21) to (2.23) follow the same procedure described in equations (2.16) to (2.18). In equation (2.24) we can now simply plug in equation (2.7) to have an approximate expression for the overall censoring under the *trim* mechanism. In addition, also the *trim* method applies a relative frequency for all studies in the meta-analysis array rather than selecting studies based on their individual probabilities. Thus, exact censoring rates can be defined. For s studies, there are k negative studies ($\widehat{\beta} < 0$) for a specific condition that get published with a probability of π, and $s - k$ positive studies ($\widehat{\beta} \geq 0$) that get published with a probability of 100%. Therefore, exactly $s^{**} = (s - k)(1 - \pi_k)$ studies get dropped in each dataset. The *Censor* step under *ppr* outputs a reduced meta-analysis array with s^{**} published studies with persisted main columns: estimated slope coefficient $\widehat{\beta}$, (estimated) standard error of the slope coefficient $\text{SE}(\widehat{\beta})$ and corresponding p-value.

2.1.5 Estimate Step

In the literature, various statistical methods to detect publication bias are available, as introduced in Section 1.3. Of those, only such that adjust funnel plot asymmetry, selection models and estimations via truncated probabilities were found to also quantify publication bias directly. Of those, the trim-and-fill method, Copas selection model and the truncated normal distribution were implemented. The *Estimate* step estimates the non-reporting rate (NRR) based on the censored data frame generated by the *Censor* step (Section 2.1.4). The NRR estimate is given by:

$$\widehat{NRR} = \frac{\widehat{n}_o}{n + \widehat{n}_o},$$

where n is the number of observed studies in the censored data frame and $\widehat{n}_o$ is the estimated number of omitted studies by the selected method. In addition, the bias is calculated by comparing the estimate to the Ground Truth NRR, which is the known proportion of censored studies to all (previously) available studies. The bias is thus calculated as:

$$\text{bias} = NRR - \widehat{NRR}.$$

For all methods, the average of the bias was calculated over all simulations for a specific condition. In addition, the empirical mean squared error (MSE) was calculated for the bias estimator for an overall comparison of the methods over all simulation conditions i as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(NRR_i - \widehat{NRR}_i\right)^2, \tag{2.25}$$

which is identical to $1/n\sum_{i=1}^{n}(\text{bias}_i)^2$ in our case. For the simulation runs that resulted in only two or less available studies, all estimation methods were programmed to return no available value ("NA").

The Trim-and-Fill Estimation Method

The trim-and-fill method from Duval and Tweedie (2000) assumes that the most negative effect estimates are most likely to be censored. This is equivalent to the graphical observation that the (lower) left part of the funnel plot is missing. As such, it is a formalization of the qualitative approach using the funnel plot. The trim-and-fill method trims the asymmetric part of the funnel after estimating how many studies were actually in the asymmetric part, i.e., estimates the number of omitted studies. The symmetric remainder is then used to estimate the true center and to replace the omitted studies - as the missing counterpart of the trimmed studies - symmetrically. The final (corrected) estimate of the meta-analysis data frame is then derived from the filled plot.

In detail, the method assumes that in addition to the n observed studies, there are an additional k_0 studies that are not observed due to publication bias. The method is nonparametric and further assumes that the suppression has taken place on the most extreme left-most (negative) values. It uses the ranks of the absolute values of the observed effect sizes $\widehat{\beta}_i$ and the signs of those effects around the global effect estimate β. If β is assumed to be known, individual $\widehat{\beta}_i - \beta$ are written as U_i, and the ranks of $|U_i|$ are denoted as r_i. These ranks run from one to n observed studies. Now, the Wilcoxon rank test statistic is calculated as $T_n = \sum_{i:U_i>0}^{n} r_i$. Subsequently, k_0 can be estimated with three different estimators (Duval and Tweedie, 2000). The one we used in this thesis is known as the linear L_0 estimator, given by:

$$\widehat{k}_0 = \frac{4T_n - n(n+1)}{2n-1}. \tag{2.26}$$

Note that in practice the global effect estimate β is not known and is iteratively estimated in the trim-and-fill method. In brief, β is estimated by a fixed-effect model (in our case). With this value, k_0 is estimated with equation (2.26). Next, the funnel plot's most outlying $\widehat{k}_0$ studies are trimmed off and β is re-estimated. Estimation and trimming are repeated for a number of iterations allowing to stabilize the estimation, i.e., achieving constant values for consecutive estimates of the two parameters. With the final estimates $\widehat{\beta}$ and $\widehat{k}_0$, the missing studies are imputed by reflecting the values of the rightmost $\widehat{k}_0$ studies symmetrically around $\widehat{\beta}$; resulting in a total number of $n + \widehat{k}_0$ studies.

Now, the non-reporting rate NRR can be calculated from $\widehat{k}_0$, estimated by the L_0 estimator defined in equation (2.26), by:

$$\widehat{NRR} = \frac{\widehat{k}_0}{n + \widehat{k}_0},$$

with $n + \widehat{k}_0$ as the total number of studies in the filled funnel plot after imputation.

The Copas Selection Estimation Model

The Copas selection model described by Copas and Shi (2000, 2001) and Copas and Malley (2008) assumes that studies have individual selection probabilities based on their effect sizes and variances. The model combines two ideas:

- *Study population:* There is a study population with estimate $\widehat{\beta}$ and standard deviation σ for each study, following a random-effects model.
- *Study selection:* Selection is based on a study-specific probability p, and bias arises when p is correlated with $\widehat{\beta}$.

The population model for the i^{th} study can be formulated as:

$$\widehat{\beta}_i \sim \mathcal{N}(\mu_i, \sigma_i^2) \quad \text{with} \quad \mu_i \sim \mathcal{N}(\mu, \tau^2), \tag{2.27}$$

with μ as the overall mean effect, σ^2 as the within study variance and τ^2 the between study variance. The corresponding selection model describes the expected proportion of selected studies given the reported study standard deviation s, equal to $\mathrm{SE}(\widehat{\beta})$, and can be formulated as:

$$P(\text{select}|\,\mathrm{SE}(\widehat{\beta})) = \Phi\left(a + \frac{b}{\mathrm{SE}(\widehat{\beta})}\right), \tag{2.28}$$

where a is controlling the overall proportion of selected studies and $b \geq 0$ is influencing the speed of the proportion increase while $\mathrm{SE}(\widehat{\beta})$ gets smaller. In other words, it controls how fast this proportion increases as the standard error becomes smaller, i.e., the study size increases. From Equation (2.28) we know that a study is only expected to be selected if $a + b/\,\mathrm{SE}(\widehat{\beta}) > 0$. In other words, for a positive value of z, defined by:

$$z = a + \frac{b}{\mathrm{SE}(\widehat{\beta})} + \delta \quad \text{with} \quad \delta \sim \mathcal{N}(0, 1).$$

Here, $\delta \sim \mathcal{N}(0, 1)$ is linearly transformed to $z \sim \mathcal{N}(a + b/\mathrm{SE}(\widehat{\beta}), 1)$. We can thus interpret z as the propensity for selection, i.e., a quantity increasing together with the conditional probability of the study to be selected. Combining these two models, we get for one study:

$$\begin{aligned}
\widehat{\beta}_i &= \mu_i + \sigma_i \epsilon_i \\
\mu_i &= \mathcal{N}(\mu, \tau^2) \\
z_i &= a + \frac{b}{\mathrm{SE}(\widehat{\beta}_i)} + \delta_i,
\end{aligned}$$

where (ϵ, δ_i) are jointly normal with correlation $\rho = \mathrm{corr}(\widehat{\beta}_i, z_i)$. We can now see, that if a study is selected ($z > 0$), it is likely that it has positive δ_i and thus also ϵ_i if the effect is positively correlated with the standard error. As a consequence: when ρ is positive, the true

mean μ is prone to be overestimated. Thus, the model reflects publication bias. To formulate the probability depending also on $\widehat{\beta}$, let us use the following notation:

$$u = a + \frac{b}{\mathrm{SE}(\widehat{\beta})}, \quad \tilde{\rho} = \frac{\sigma}{\sqrt{\tau^2 + \sigma^2}}\rho.$$

Then, for a typical study in the population it can be shown that:

$$P(\text{select} \mid \mathrm{SE}(\widehat{\beta}), \widehat{\beta}) = p(a, b, \mathrm{SE}(\widehat{\beta}), \widehat{\beta}) = \Phi\left(\frac{u + \tilde{\rho}(\widehat{\beta} - \mu)/\sqrt{\tau^2 + \sigma^2}}{\sqrt{1 - \tilde{\rho}^2}}\right),$$

with $p(\cdot)$ as the conditional distribution of selection given $\mathrm{SE}(\widehat{\beta})$ and $\widehat{\beta}$. By Bayes theorem, the likelihood is the product of the marginal distribution of $\widehat{\beta}_i$ times the conditional distribution of selection p, divided by the marginal probability of selection given by $\Phi(u)$. As described in detail by Copas and Shi (2001) the log-likelihood function $l(\cdot)$ of one study in the the meta-analysis is thus:

$$l(a, b, \mathrm{SE}(\widehat{\beta}_i), \widehat{\beta}_i) = -\frac{1}{2}\log\left(\sigma_i^2 + \tau^2\right) - \frac{\left(\widehat{\beta}_i - \mu\right)^2}{2\left(\sigma_i^2 + \tau^2\right)} + \log p\left(a, b, \mathrm{SE}(\widehat{\beta}_i), \widehat{\beta}_i\right) - \log\Phi\left(u_i\right). \tag{2.29}$$

Pairs of (a, b) are imputed and for each given pair, the likelihood function is maximized to give estimates of μ, τ and ρ. In detail, the algorithm for the Copas selection model in the implementation used from Schwarzer et al. (2010) performs the following steps:

1. The Copas selection model is fitted, i.e., equation (2.29) maximized, repeatedly over a grid of (a, b) values for increasing degree of selection.
2. The p-value for residual selection bias ($H_0 : \mu = 0$) is monitored.
3. The first estimated treatment effect which results in a p-value for residual selection bias below 10% is selected. Additionally, the corresponding standard error and estimate of the number of missing studies m is returned.

The last step estimates the number of unpublished studies m from the number of observed studies n for a specified pair (a, b) and estimate of μ by:

$$m = n - \sum_{i=1}^{n}\left(P(\text{select}|\,\mathrm{SE}(\widehat{\beta}), \widehat{\beta})\right)^{-1}. \tag{2.30}$$

Taken together, the NRR is estimated as:

$$NRR = \frac{m}{m + n}.$$

The Truncated Normal Distribution Estimation Method

The truncated normal distribution method from Zhu and Carriere (2018) assumes that the estimated effect is normally distributed as:

$$\widehat{\beta}_i \sim \mathcal{N}(\mu, \sigma^2),$$

with μ as the common true effect size and σ^2 as the within-study variance. Now, if studies are (completely) omitted due to small effect size, only studies with $\{\widehat{\beta}_i | \widehat{\beta}_i \geq k\}$ are observed; k being some unknown cutoff value for the (estimated) effect variable. Similarly, when studies

are omitted due to large p-values only studies with $\{\widehat{\beta}_i|\widehat{\beta}_i/s_i \geq t\}$ are included, t being some unknown value for the cutoff on the t-value scale. If the effect is (completely) truncated by effect size, study i is not included in the meta-analysis when $\widehat{\beta}_i < k$. Then, the probability of truncating the i^{th} study is:

$$p_i = P\left(\widehat{\beta}_i < k\right) = P\left(\frac{\widehat{\beta}_i - \mu}{\sigma_i} < \frac{k-\mu}{\sigma_i}\right) = \Phi\left(\frac{k-\mu}{\sigma_i}\right),$$

for a given k, μ. Conversely, the likelihood of including study i in the meta-analysis is:

$$L_i(\mu, k) = P\left(\widehat{\beta}_i \mid \widehat{\beta}_i > k\right) = \frac{P\left(\widehat{\beta}_i\right) P\left(\widehat{\beta}_i > k \mid \widehat{\beta}_i\right)}{P\left(\widehat{\beta}_i > k\right)} = \frac{\phi\left(\frac{\widehat{\beta}_i-\mu}{\sigma_i}\right)}{\left[1-\Phi\left(\frac{k-\mu}{\sigma_i}\right)\right]} I\left(\widehat{\beta}_i > k\right), \tag{2.31}$$

with $P(\widehat{\beta}_i)$ as the density of $\widehat{\beta}_i$. Note that the expression for the density in equation (2.31) from Zhu and Carriere (2018) is wrong. Correctly, the density $P(\widehat{\beta}_i) = \phi((\widehat{\beta}_i - \mu)/\sigma_i) \times 1/\sigma_i$ to ensure integration to one. However, I have used the incorrect formula shown here to test the published procedure. Here, Bayes theorem is applied first and the indicator function $I(\cdot)$ maps all elements $\widehat{\beta}_i > k$ to one. As a result, the likelihood of observing the n studies is:

$$L(\mu, k) = \prod_{i=1}^{n} L_i = \prod_{i=1}^{n} \frac{\phi\left(\frac{\widehat{\beta}_i-\mu}{\sigma_i}\right)}{\left[1-\Phi\left(\frac{k-\mu}{\sigma_i}\right)\right]} I\left(\widehat{\beta}_i > k\right).$$

Therefore, the log-likelihood is given by:

$$l(\mu, k) = \sum_{i=1}^{n} \log \phi\left(\frac{\widehat{\beta}_i - \mu}{\mathrm{SE}(\widehat{\beta}_i)}\right) - \sum_{i=1}^{n} \log\left[1 - \Phi\left(\frac{k-\mu}{\mathrm{SE}(\widehat{\beta}_i)}\right)\right], \tag{2.32}$$

with σ_i being replaced by the estimated $\mathrm{SE}(\widehat{\beta}_i)$ and constraint that $\widehat{\beta}_i > k, \forall i$. Due to this constraint, the estimate of k cannot be achieved by normal maximization of the likelihood, as described in detail by Zhu and Carriere (2018). Instead, the estimate of k is derived by the consistent estimator:

$$\widehat{k} = \min_i(\widehat{\beta}_i). \tag{2.33}$$

Now, we obtain $\widehat{\mu}$ by maximizing the log-likelihood function in equation (2.32) and replacing k with $\widehat{k}$, estimated in equation (2.33) to estimate the proportion of truncation p. Since $\widehat{p}_i = P(\widehat{\beta}_i < \widehat{k})$ is the probability of study i being truncated, we can sum all truncation probabilities over the n observed studies to get an estimated number of studies which were truncated. The resulting estimated number of omitted studies $\widehat{n}_{\widehat{\mu},\widehat{k}}$ is thus:

$$\widehat{n}_{\widehat{\mu},\widehat{k}} = \sum_{i=1}^{n} \widehat{p}_i = \sum_{i=1}^{n} P\left(\widehat{\beta}_i < \widehat{k}\right) = \sum_{i=1}^{n} \Phi\left(\frac{\widehat{k} - \widehat{\mu}}{\mathrm{SE}(\widehat{\beta}_i)}\right), \tag{2.34}$$

for a given $\widehat{k}$ and $\widehat{\mu}$. The NRR can now be calculated as follows:

$$NRR = \frac{\widehat{n}_{\widehat{\mu},\widehat{k}}}{\widehat{n}_{\widehat{\mu},\widehat{k}} + n}.$$

In the thesis at hand, we applied one method for the truncation based on $\widehat{\beta}$ and one on the corresponding p-value. The process of the latter is identical to the one described above, except that k and $\widehat{\beta}_i$ are replaced by t and $\widehat{\beta}_i/\,\mathrm{SE}(\widehat{\beta}_i)$, respectively.

2.1.6 Simulation Implementation

All analyses were performed in the R programming language (R Core Team, 2022) using base packages and the following analysis-specific packages: `tidyverse` for complex data preparation (Wickham et al., 2019), `ggpubr` for complex ggplot formatting (Kassambara, 2023), `metafor` to fit meta-analytic models for "*ppr*" censoring, to carry out trim-and-fill analysis (Viechtbauer, 2010), `metafor` and `meta` to fit meta-analytic models for "*trim*" censoring and contour-enhanced funnel plots (Balduzzi et al., 2019), `foreach` and `doParallel` for parallel computing (Microsoft and Weston, 2022; Corporation and Weston, 2022).

2.2 Tracking of Study Requests to Publication

2.2.1 Procedure for Obtaining and Documenting License Applications

The basic procedure for obtaining an animal experiment license in Switzerland is as follows: first, a scientist of an institute writes an application. The cantonal veterinary office (CVO) then carries out a preliminary examination of this application. In addition, the cantonal animal experimentation commission also assesses the application and makes a recommendation to the CVO as to whether or not the permit should be granted. Taking this into consideration, the CVO then makes the final decision. In Switzerland, the Federal Food Safety and Veterinary Office (FSVO) produces annual public reports with national statistics on animal experimentation. This presupposes firstly that the researcher reports on all his experimental activities with animals every year. Secondly, the FSVO thus needs access to all cantonal applications and annual reports. All forms required for processing and documenting animal experiment applications are managed by the web application "animex.ch". The two most important document types for this thesis are: the "Application for licence to perform animal experiment" (Form A) and the "Annual report for animal experiment" (Form AC). Form A contains a specific application ID (ID), the cantonal and national license application number, the address of the applicant and the cantonal authority, information about the experiment (title, purpose, number and species of animals, degree of severity, duration, etc.) and more. The national license numbering is automatically done by animex, whereas the cantonal number is assigned by the authorities. Importantly, an attachment of Form A contains a list of all involved persons for one request.

After the approval of a Form A, the experimental activity of the corresponding ID is reported each year in Form AC. Thus, there can be several Forms AC, for one Form A or - equivalently - one ID. If the experiment was completed in the reporting year, the report type of Form AC changes from "Annual Report" to "Last Annual Report". Form AC always contains the information of the ID, contact information and total number of animals used in the calendar year. The animal numbers are stratified i.a., to the degree of severity. In addition, the numbers of approved and requested animals are listed. Furthermore, researchers are asked if the results were achieved, if there are improvements and potential for 3Rs (Replace, Reduce, Refine) and whether there were any unforeseen events in the reporting year. In addition, researchers should provide a list of publications that have already appeared or are planned under this ID. Importantly, there is a second version of the Form AC, designated as "Public Report". The Public Report i.a., contains the number of animals per study that are published by the FSVO. These numbers differ to the ones found in the AC reports if mistakes were made by the researcher or the CVO. In particular, the researcher's entries are checked by the CVO, whose entries are in turn corrected again by the FSVO, if necessary. Mistakes can include manual transcription errors, such as filling in the wrong fields in terms of species type or degree of severity. Another reason for errors may be that animal counts are carried over to the next year and are thus counted in two consecutive years (FSVO, personal communication). We therefore assumed that figures in the public reports which were checked twice best reflect the actual experimental activity.

2.2.2 Overview of Process to Track Study Requests

One of our main objectives in tracking license applications (requests) is to determine the proportion of approved requests that resulted in one or more publications. This requires a procedure to identify publications that can be traced back to one specific ID. In addition, comprehensive information on the requests is needed for identifying associated projects in the literature. For the latter, we received several data base export files from the FSVO, containing information from multiple merged form types that were introduced in Section 2.2.1. More details about the files and their structure is given in Section 2.2.3. In brief, there are three relevant files, coined as follows: the "*Application*", "*Personnel*" and "*Public Report*" file. They correspond to merged documents of either Form As, attachments of Form As or Public reports, respectively. My proposed process to track study requests (applications) consists of four main steps: the PubMed query, keyword matching, manual screening and endpoint calculation. The flowchart in Figure 2.3 represents the sequential execution of this process. First, IDs are selected that exist in the *Public Report* file. Then, all names assigned to individual IDs are extracted from the *Personnel* file. This ID-specific list of potential author names is collapsed to a string and is subsequently used as the most basic element of one PubMed query. In addition, the date from which the application was valid from ("*Valid Date*") is extracted from the *Application* file and appended together with other elements to the query, as explained in Section 2.2.4. Now, the query is triggered programmatically for each request. The result of one query is, i.a., a raw list of abstracts ("Abstracts raw"). Abstracts from the raw list are only kept if they contain at least one keyword extracted from the application title. The keyword matching algorithm is described in detail in Section 2.2.5. The kept abstracts are then manually screened for additional in- or exclusion criteria, described in Section 2.2.6. The final selection of publications is then used to calculate different endpoints, described in Section 2.2.7. In brief, we calculated the proportion of studies that resulted in at least one publication. In addition, the non-reporting rate of animals and discrepancies between requested and published animals were assessed.

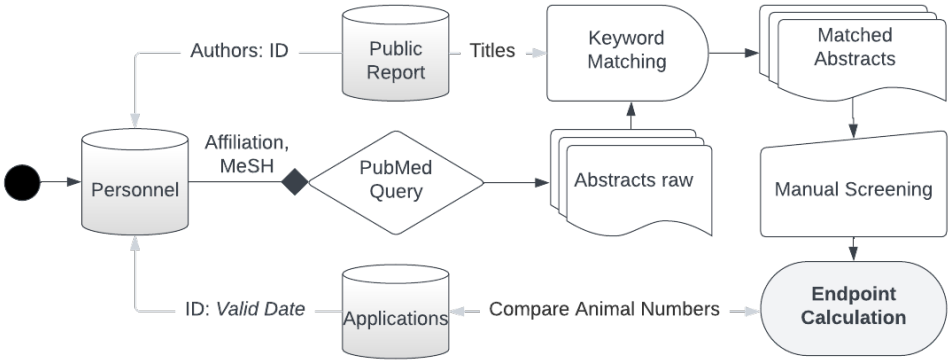

Figure 2.3: Tracking of Study request flow chart. Data base shapes with grey color gradients represent received documents from the FSVO. The rhombus shape indicates a potential input-dependent decision point. First, the author lists in the *Personnel* file are selected for IDs present in the *Public Report* file, and merged with the corresponding date from which an application is valid (*Valid Date*) from the *Application* file. The PubMed query is then performed with or without specifying affiliation and MeSH (Medical Subject Headings) search fields. The result is a list of abstracts which undergo a keyword matching algorithm. Keywords are derived from the application titles listed in the *Public Report* file. Only publications are kept which contain at least one of the keywords. These are then manually screened and the remaining are used to calculate endpoints, which are compared back to the *Application* file.

2.2.3 Data Collection and Curation

The *Personnel*, *Application* and *Public Report* files were kindly provided by the FSVO as UTF-16 encoded .txt files. The FSVO generated these files with a Structured Query Language (SQL) export from their data base. We had problems to parse the first versions of the .txt files into R with `read.table()`, since certain free-text columns contained special characters and line breaks. Therefore, the FSVO repeated the export and omitted the following column names per document type:

- *Personnel*: "AREA_OF_RESPONSIBILITY".
- *Application*: "TITLE", "OD_OTHER_DISEASE", "SPECIAL_PROVISIONS", "COMMISSION_DECISION", "PROPOSAL_COMMENTS", "Experiment_Title", "SUBTITLE" and "REQUIREMENTS".
- *Public Report*: "RESULTS_ACHIEVED", "TOXICOLOGICAL_STUDIES", "SUITABILITY_SPECIES_NUMBERS", "SPECIAL_EVENTS" and "PLANNED_REPORTS_PUBLICATIONS".

Apart from these three files we also received the "*AC*" file. The *AC* file contains information from multiple AC forms, completed by the researchers. However, we did not use this file as the *Public Report* file contains the columns we are interested in too, but reviewed and condensed by the FSVO. For all applications I asserted that the status was "TERMINATED" or "EXPIRED" in the *Application* document. In other words, I considered only projects that do not longer perform animal experiments under this application. Since the files were exclusively handled on a secure server of the Department of Mathematics of the UZH, no dynamic exploratory data analysis of the files is possible.

The filtered files contained information on a total of 36 applications approved by the CVO

of Zurich in the field of neurology. The vast majority of requests were for 2016, while three were from 2021. The *Personnel* file is a data frame with the main columns: "APPLICATION_ID" (ID), "FIRST_NAME" and "LAST_NAME". Each row contains information about one person, i.e., there are as many rows per ID as there are potential authors. In other words, the total number of rows of the *Personnel* file equals to the grand total of staff names in all requests. This total was 286 and the number of potential authors per ID ranged between 3 and 38. Also in the *Application* document, there are several rows present for one ID. There is one row for each animal category, application type (e.g., new or renewal), status of the application, etc. Here, the number of rows per ID ranged from 1 to 5. There was a substantial amount of row duplicates in the *Application* document, which were removed with `unique()`. In addition, one application was duplicated with two different IDs. It is likely that the ID was included twice, one time with the old ID naming convention from the web application preceding animex and one time with the new notation. The ID with the old naming convention was thus manually removed from the analysis. The processed *Application* file contains 75 rows and the main columns: ID, "DECISION_DATE", "TOTAL_ANIMAL_REQUESTED" and "TOTAL_ANIMAL_APPROVED". In the *Public Report* file the multiple rows per ID represent the individual reporting year and animal category. Thus, the number rows per ID equals to the product of the number of animal categories and the number of reported years. For instance, for a project with a duration of two years and three animal categories, there are $2 \times 3 = 6$ rows for this ID. Here, numbers of rows per ID span from 2 to 16. The main columns of this file were "TotalPublished", representing the total number of published animals per animal category, and the columns summing up the used animals per degree of severity, e.g., ANIMALS_USAGE_DS_ONE for the degree of severity (DS) one.

2.2.4 Pubmed Query Definition

First, a data frame containing a list with all potential authors was generated from the *Personnel* file with the `aggregate()` function while collapsing the "FIRST_NAME" and "LAST_NAME" column. After collecting all potential author names in a list, a query string is defined per request. The string is generated with standard PubMed syntax. Included are all author names appended with the author search field tag [au], separated by the Boolean operator "OR". Next, ID-specific publication dates are defined by specifying the earliest date with the information from the column "APPLICATION_VALID_FROM" (*Valid Date*) from the *Application* file. To each query, the term "APPLICATION_VALID_FROM:NOW" is appended together with the publication date tag [dp]. "NOW" indicates the current date which is detected automatically with `Sys.Date()` in the implementation. However, the results from the thesis at hand were generated for publication dates only until November 13, 2023. The rational to include the *Valid Date* is that it represents the earliest possible date of a publication originating from the respective request. Therefore, this restriction was added to filter out irrelevant dates, i.e., such that were published before the approval of the experiment. In addition, the publication type was specified to "Journal Article" for all queries with the [pt] tag. Beyond this, several publication types were additionally excluded with the "NOT" operator. The reason to do so is that I observed that some publications have multiple publication types assigned to the [pt] search field tag, e.g., "Review" as well as "Journal article".

The current implementation allows to exclude certain elements from the query string in order to examine the influence of the individual terms on the number of hits retrieved. These terms are coined as "*Variable*" in the following, while elements that were always included are referred to as the "*Core*" of the query. For instance, the institutional name was a *Variable* part of the query and added with the "AND" operator to strings. I will refer to this search field and tag with "*Inst*" in the following. Next, all searches had the further option to be limited to strings specifying the affiliations fields with "Switzerland" and the affiliation tag [ad]. I will refer to this

element as "*CH*". Furthermore, a Medical Subject Heading field (MeSH) was optionally used as *Variable* to only include publications including the label for animal experiments, by the addition of: "Animals[Mesh]". To this element I refer to as "*MSH*' in the following. Since PubMed processes searches in a left-to-right sequence, parentheses were not required for the most part of the query string. Parentheses nest concepts that should be processed as a unit and incorporated as such into the overall search. The only nested term in the query structure, for which parentheses are thus necessary, is the specification of the university name. Here, all but one request belonged to the university of Zurich and the search was thus appended with "(UZH[ad] OR Institute XY[ad])", where "Institute XY" is the institute name corresponding to the sole request stemming from another university. I will refer to this nested search field and tag with "*UZH*". To summarize: there are *Variable* and *Core* search field tags. The *Core* fields are present in every query and contain the author list, publication date and -type. I decided to never remove these fields, since they represent direct exclusion points if they are not present in a publication. The *Variable* fields include *MSH* and the three specifying the affiliation field. Furthermore, a distinction can be made between "*Static*" and "*Dynamic*" query terms. *Static* means that these fields are identical across request - if included - and are: *CH*, *UZH*, *MSH* and all those specifying the publication type. The *Dynamic* terms are ID-specific, i.e., change with every ID, and include: author names, institution name and publication date. Table 2.2 summarizes the architecture of the query. To illustrate, an example (truncated) query string: "Dominique Lorgé[au] OR Reinhard Furrer[au] OR Servan Grüninger[au] AND InstituteName AND Switzerland[ad] AND 2021/24/04:2023/11/05[dp] AND Animals[MeSH] AND Journal Article[pt] NOT Review[pt]".

Table 2.2: PubMed query architecture. All search fields belonged either to the query *Core* or were *Variable*. The *Core* of the string was present in every query performed, whereas the implementation allows the *Variable* fields to be removed. *Static* search fields are constant terms across requests while *Dynamic* ones contain request-specific information.

	Core	*Variable*
Static	publication type	*CH, UZH, MSH*
Dynamic	authors, publication date	*Inst*

After generating the query string, the PubMed database was programmatically queried via the National Center for Biotechnology Information (NCBI) Entrez E-utilities interface. In essence, the Entrez E-utilities interface is an Application Programming Interface (API) with a set of functions that allow to assess PubMed records in an automated manner. In particular, the `easyPubMed` package was used, which is an R interface to the Entrez E-utilities. The functions of the package take the query as an argument and return i.a., a list containing all matching records in .txt or .xml format directly in R. To increase the limit of queries per second, I obtained an API key from NCBI. The result of this step is thus a list of abstracts (Abstracts Raw) which: matched one of the author names, the time period defined by the decision date, the affiliation and MeSH terms, the publication type and university name specifications. Note that also all *Variable* search fields were used in the query generating the abstract list for the manual screening and upcoming sections. The implementation, which makes it possible to optionally exclude the *Variable* elements, was set up for analysis purposes only.

2.2.5 Keyword Matching

Each retrieved abstract in Section 2.2.4 is then checked for keywords automatically, to exclude further irrelevant hits. The ID-specific list of keywords was extracted from the application titles

in the *Public Report* file. To identify relevant keywords, I used the `udpipe` package which contains functions for natural language processing tasks and which provides an English language model. In particular, lemmatization and Part-of-speech (POS) tagging was applied on the title strings. Lemmatization is the process of reducing words to their base or root form. For example "walking" is reduced to "walk". POS analyzes words in a sentence and labels them with categories, such as noun, adjective or verb. All words from the POS categories nouns, adjectives and proper nouns are extracted from the title as keywords after lemmatization. Proper nouns are specific names for individual people and other unique entities. Nouns were selected as POS categories for keywords, since they represent objects that contain substantial information from the titles. Adjectives are selected, since technical attributes can be important in distinguishing fields of research. For instance, "neural development" conveys much more information than just "development". Proper nouns are extracted to account for e.g., gene names and the like. Verbs are intentionally dropped, since they are often of a rather generic nature in titles of scientific projects, e.g., "analyzing" and so on. Obviously, connecting words, prepositions, pronouns, auxiliary verbs and other not content-carrying words are neither extracted. From the example string: "These biostatisticians are analyzing some very interesting CRISPR/Cas9 data", only the keywords "biostatistician", "interesting", "CRISPR/Cas9" and "data" would be extracted. Note the lemmatization from "biostatisticians" to "biostatistician". In addition, there were two titles in German, which were manually translated to English before the keyword extraction. Finally, the presence of at least one of the extracted keywords from a specific request title was checked for the corresponding abstract strings using the base function `grepl()`. The result of this step is a narrowed down list with relevant publications for the full-text screening.

2.2.6 Manual Screening

Publications were considered as origination from a specific ID, if it could be confirmed manually. In particular, I performed a review on the selected abstracts and/or full-texts to check additional in- and exclusion criteria. First, I quickly check the abstracts content to rule out obvious candidates right away. The first thing I checked in detail was if the cantonal authorization number (ZHXXX/XX) or the national one is available and matches the one on from the request. These two numbers were considered as the sole unique identifiers (UIDs) of publications for a given ID. Therefore, if e.g., a cantonal number was mentioned that matched the one on the application, the publication was directly included for the downstream analysis. Contrarily, if a non-matching cantonal or national number was present, the publication was directly excluded. If no UID was available, an attempt was made to assign the publications to the application using other information. For instance, it was ensured that the canton of the responsible veterinary office was Zurich. In addition, the author names were double-checked, since PubMed truncates names to account for varying initials. Next, the matching author(s)' institutional affiliation as well as company (university) name was examined and compared to the applicant's information on the *Application* document. When authors and requester names matched, the content of the abstract and/or full-text was compared to the title in the *Public Report* document and further information present in the *Application* file. For instance, animal species, specific field of study and related human disease were reviewed in the full-text and checked if they are in accordance with the information on the application files. Finally, it was assumed that one PMID only corresponded to one ID. To semi-automate this process, I have implemented an interactive function with the `readline()` function. The function runs through all abstracts from a given ID and prompts the user to decide on a case-by-case basis whether the queried abstracts actually originate from the corresponding request. In addition, comments about the why a study is in- or excluded are stored by the function. The main result of this step is a list with publications that are considered originating from the queried request information. This list is then used to assess the publication rate and other endpoints, described in Section 2.2.7.

2.2.7 Endpoint Calculation

Three main endpoints were determined. First, the non-reporting rate on request-level was assessed as the proportion of applications which led to at least one publication. Second, the total number of published animals per request were identified in a full-text search on the corresponding articles. This allows to calculate the NRR on animal-level. To do so, an initial animal count extraction was conducted on the full-texts leveraging on ChatGPT-4, an advanced Large Language Model (LLM). This approach aimed to provide a preliminary idea and estimation of the numerical data about animal numbers present in the literature. In addition, the LLM's efficacy could be evaluated. The .pdf files were acquired directly from e.g., PubMed Central (PMC) and subsequently uploaded without alterations to ChatGPT-4. The exact message to the ChatGPT-4 chat bot was: "Please extract and calculate the total number of animals used in this research article, including any explicit statements (like 'n = x') and other descriptions of the number of animals in each experimental group. It's crucial to ensure that all relevant sections of the document are thoroughly reviewed and all instances of animal numbers are accounted for. Conduct a thorough review to extract all mentions of animal numbers. For the output as the total number, execute a simple addition using the extracted numbers. Also show your calculation, i.e., the number of the individual animals per groups." Subsequently, I performed a manual extraction by reviewing the full-text content of each article in .pdf format. This involved an examination wherein all mentions related to animal counts were highlighted within the articles. When publications lacked details on the exact number of animals used for a specific experiment, I used the biggest number mentioned, e.g., when a group size was mentioned as a range from 3 to 5, I noted 5 animals. The identified counts were then manually summed up. Finally, the number of requested and approved animals from the application files, and the observed counts in the literature were compared.

2.2.8 Tracking Implementation

Due to the sensitive nature of the information contained in the animal experiment license application forms, all steps including data with personal information were performed on the thinlinc server of the Department of Mathematics at the University of Zurich. Only non-personal and aggregated results were then transferred back to my local machine and included in the report at hand. All scripts for the data loading, processing and analysis are attached in the online version of the report, whereas the data is not provided. All analyses were performed in the R programming language (R Core Team, 2022) using base packages and the following analysis-specific packages: `easyPubMed` for programmatic PubMed record retrieval (Fantini, 2019), `xtable` to convert query result tables in `R` to Latex (Dahl et al., 2019), `udpipe` for lemmatizing and part-of-speech tagging (Wijffels, 2023), `car` for enhanced scatter plot matrices (Fox and Weisberg, 2019) and `MASS` for robust regression lines (Venables and Ripley, 2002). The LLM ChatGPT-4 was used for automatic full-text animal count extraction (OpenAI, 2023).

Chapter 3

Results

3.1 Simulation Results

The goal of the simulation was to compare the performance of statistical methods that are capable of estimating the non-reporting rate across a broad spectrum of conditions. The range of meaningful treatment effects ($\rho = \beta \times 0.2 \in [-0.7, 0.7]$) and publication probabilities ($\pi \in [0, 1]$) was assessed for $m = 10$ simulations and $s = 100$ studies. Heatmaps were then generated with cell values and grayscale mapping according to the censoring rates (Section 3.1.1) or bias of the method (Section 3.1.2); calculated as the average over all simulations.

3.1.1 Censoring Results

To first investigate the censoring behavior of the simulation, the non-reporting rate was visualized without the *Estimate* step (Section 2.1.5). The corresponding heatmap is depicted in Figure 3.1 with grayscale mapping as the mean of the ground truth (GT.mean) non-reporting rate over 10 simulations. The *trim* censoring (left) keeps studies with a negative estimated treatment effect $\widehat{\beta} < 0$ (or correlation) with a publication probability π. There is no censoring for high positive correlation and publication probability of 100%. The non-reporting rate is gradually increasing for decreasing correlation and publication probability, reaching the expected maximum at high negative correlation and publication probability equal to zero. At correlation and publication probability equal to zero, the normal distribution of $\widehat{\beta}$ is basically cut in half at its mean, thus resulting in approximately 50% censoring. The overall pattern is explained by the gradual increase of the proportion of censored studies with decreasing publication probability π and true treatment effect β. This is also seen in equation (2.24) which was derived in Section 2.1.4 and is again presented here:

$$P(\text{Censoring} \mid \widehat{\beta}_k, \pi_k) \approx (1 - \pi_k) \cdot \Phi\left(-\frac{\sqrt{n}\beta\sigma_x}{\sigma_e}\right).$$

The approximation of the non-reporting rate in this equation reflects the *trim* censoring very well, depicted in Figure A.1. Graphically spoken, we are moving the simulated funnel plot from right (positive effect) to left (negative effect) when we consider the lateral movement from top to bottom in Figure 3.1. As soon as the null effect is first crossed, the left-most negative ranks are prone for censoring. Ultimately, when all studies have $\widehat{\beta}_i < 0$ and $\pi = 0$, all studies are censored, even though the vast majority of them portray the true underlying effect.

The *ppr* censoring (Figure 3.1, right) keeps studies with non-significant treatment effects (p-value > 0.05) with a publication probability π. For *ppr* censoring, there is again no censoring for publication probability of 100% and only vanishingly small amounts for high negative and positive correlation. The non-reporting rate gradually increases for decreasing absolute effect

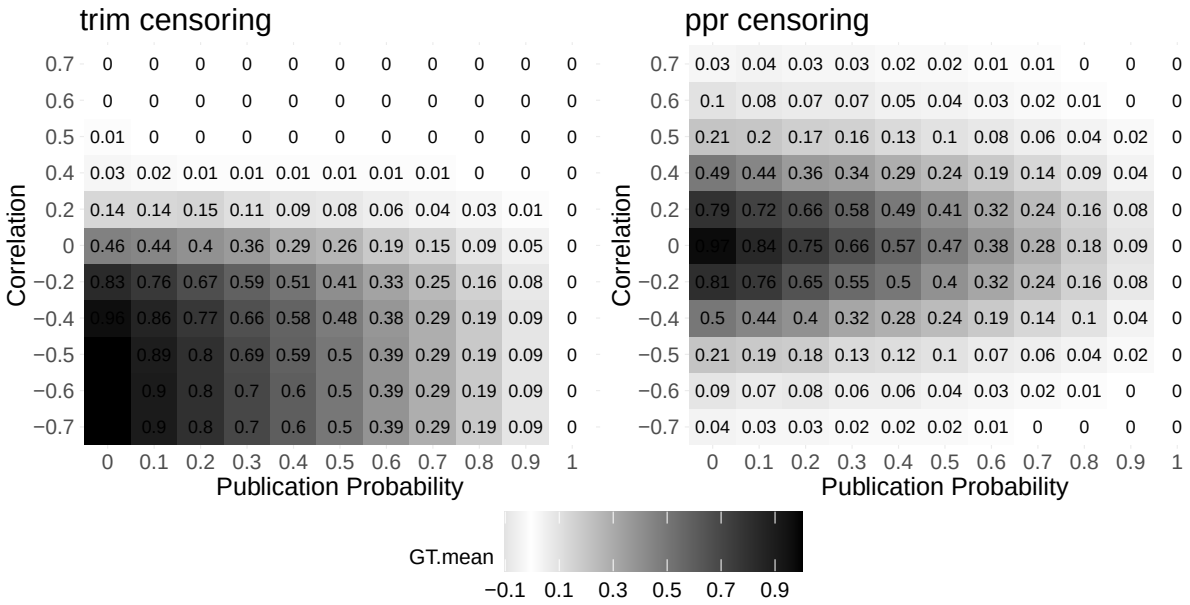

Figure 3.1: Heatmap for the censoring rates for the *trim* (left) and *ppr* censoring (right). A total of 121 conditions are tested for $s = 100$ studies and $m = 10$ simulation runs. Grayscale mapping and cell values indicate the mean of all true non-reporting rates over all simulation runs per conditions, referred to as "Ground Truth" (GT.mean). Correlation equals to Pearson's correlation coefficient ρ, i.e., to $\beta \times 0.2$, as described in equation (2.2). Publication probability π determines the proportion of censored studies from the selected studies.

values and publication probabilities, reaching the maximum at no correlation and publication probability equal to zero. The asymmetric horizontal pattern is again explained by the gradual increase of the proportion of censored studies with decreasing publication probability π. The overall pattern is explained by the p-value censoring in conjunction with the publication probability dependency. This is captured in the equation (2.19) which was derived in Section 2.1.4 and is presented again below:

$$P(\text{Censoring} \mid T_k,\, \pi_k) \approx (1 - \pi_k) \cdot (F_{t:n-2;ncp}(|t_{crit,k}|) - F_{t:n-2;ncp}(-t_{crit,k}))\,.$$

The approximation by this equation depicted in Figure A.1 reflects the *ppr* censoring very well. In addition, it is again helpful to mentally visualize a funnel plot, as introduced in Section 2.1.4. Since the p-value is the probability to observe the effect under $H_0 : \beta = 0$, the corresponding pseudo confidence interval limits are centered around zero. Now, the closer we come to the null effect (input β approaching zero), the more non-significant studies are expected and are prone to be censored. Ultimately, for no correlation ($\beta = 0$) and $\pi = 0$, the proportion of studies that will be censored is 95%. This is exactly the worst-case scenario already stated by Rosenthal (1979) in the context of publication bias, where no true effect exists and the censoring of non-significant findings is completely strict, such that only the 5% of false positive results are available in the literature.

3.1.2 Estimation Results

Results for the Trim-and-Fill Method

The results for the trim-and-fill ("*fill*") estimation can be seen in Figure 3.2. For the *trim* censoring (left), the *fill* estimation method performs best for high positive correlation as well as publication probability. The method does not greatly overestimate the non-reporting rate for low censoring, with negative bias in the single-digit percentages. However, the trim-and-fill method fails to precisely estimate the non-reporting rate for moderate to high censoring. For the *trim* censoring, this translates to a bad performance for negative correlations and low publication probability. Conversely, the method performs best for positive correlations and high publication probabilities π. For the *ppr* censoring (Figure 3.2, right), the *fill* estimation behaves best for high correlation and high publication probability. However, the method also fails to estimate the non-reporting rates for moderate to high censoring here. Over the whole grid, a MSE of the NRR estimation of 0.1 and 0.06 was calculated respectively for the *trim* and *ppr* censorings.

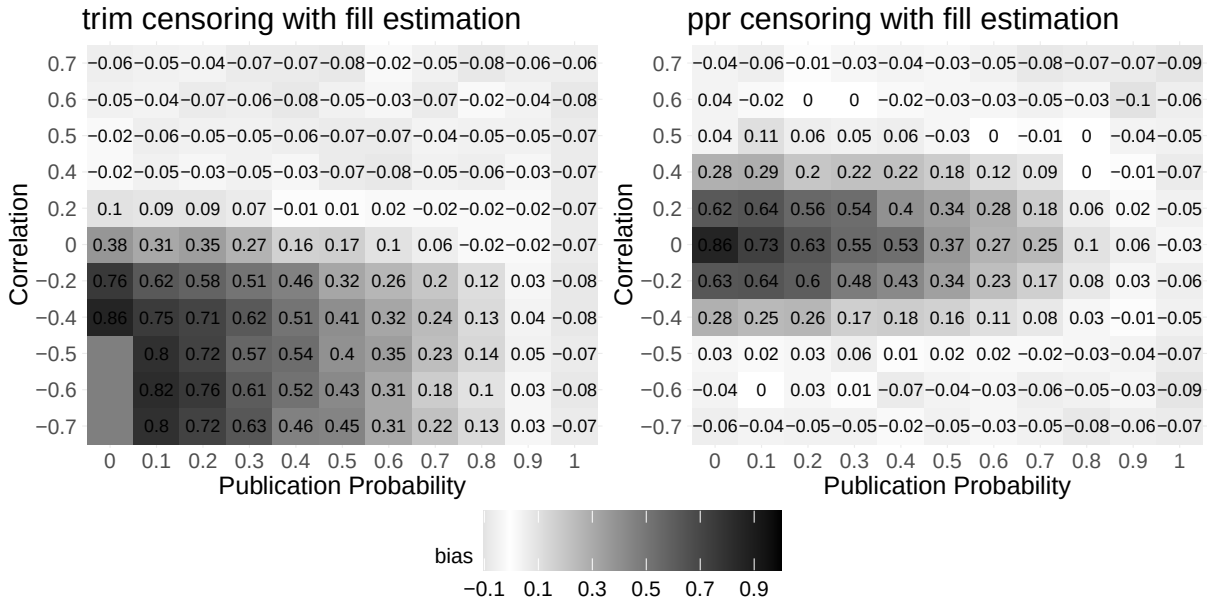

Figure 3.2: Heatmap for the bias of the *fill* estimation with *trim* censoring (left) and *ppr* censoring (right). A total of 121 conditions are tested for $s = 100$ studies and $m = 10$ simulation runs. Grayscale mapping and cell values indicate the mean of all bias over all simulation runs per conditions (bias). Bias is calculated as the true non-reporting rate minus the estimated non-reporting rate. Correlation equals to Pearson's correlation coefficient ρ, which is $\beta \times 0.2$, as described in equation (2.2). Publication probability π determines the proportion of censored studies from the respective censoring mechanism. Cells without values have no available numerical result for the condition.

To analyze the NRR estimation behavior theoretically, let us recap equation (2.26) to estimate the number of omitted studies k_0:

$$\widehat{k}_0 = \frac{4T_n - n(n+1)}{2n-1}.$$

Here, the highest value of $T_n = \sum_{i:X_i>0}^{n} r_i$ is achieved in the hypothetical setting that $X_i = \widehat{\beta}_i - \beta > 0, \forall i$, for a known β. In this case, the statistic is just the sum of all n ranks, i.e.,

$T_n = \sum_{i=1}^{n} i = n(n+1)/2$. We can now plug this into the equation above to derive the maximum possible value for $\widehat{k}_0 = n(n+1)/(2n-1)$. Since the algorithm restricts the estimator to positive values for $\widehat{k}_0$, the bounds for the number of estimated studies that were omitted are given by: $[0, n(n+1)/(2n-1)]$. The upper limit in this expression will increase with increasing n and will always be smaller than the number of observed studies for $n > 2$. For increasingly large n, the maximum value of $\widehat{k}_0$ will approach $n/2$. Using this limit for large n, we can approximate the bounds of the L_0 estimator for k_0 with $[0, n/2)$. This means that the trim-and-fill algorithm can maximally fill half the number of studies that were observed. In other words, the estimated NRR has an approximated upper limit of $1/3$ for large n. This is also reflected in the background calculations of Figure 3.2, where the highest NRR estimation was 0.22 for both censoring methods (NRR estimation values not shown in the Figure).

Results for the Copas Selection Model

The results for the Copas selection model ("*copas*") estimation can be seen in Figure 3.3. For the *trim* censoring (left), the *copas* estimation method performs best for moderate to high positive correlations and a publication probability of 100%. For the *ppr* censoring it behaves best for high positive or negative correlations and publication probability of 100%. The method does also not overestimate the non-reporting rate for low censoring in both regimes, with a better precision than the trim-and-fill method for low censoring. However, also the *copas* method fails to precisely estimate the non-reporting rate for both methods if there is increasing censoring. The method also performs best for positive correlations and high publication probabilities π under *trim*; and low π and correlation under *ppr*. The *copas* method performs similar to the *fill* estimation for increasing non-reporting rates. However, the method performs substantially better than the trim-and-fill method for the *ppr* censoring with $\pi = 0$ and low absolute correlation. Over the whole grid, a MSE of 0.12 and 0.07 was calculated respectively for the *trim* and *ppr* censorings.

To analyze the NRR estimation behavior, let us recap equation (2.30) to estimate the number of omitted studies m from the number of observed studies n for a specified pair (a, b):

$$m = n - \sum_{i=1}^{n} \left(P(\text{select}|\, \text{SE}(\widehat{\beta}_i) \right)^{-1}.$$

In the terminology used for the population model in equation (2.27), the sum in this expression is an estimate of the size of the study population $(n + m)$ from which the observed studies are assumed to be a sample. With no selection (e.g., at $a = 10$ and $b = 0$ in equation (2.28)), $P(\text{select}|\, \text{SE}(\widehat{\beta}_i)) = 1$ and the sum over all observed estimates will thus just be the number of studies n, i.e., m will be equal to zero. If there is substantial selection, this number will be much larger and m increases. As a consequence, the range of values that m can theoretically have is bounded between $(0, n)$. Therefore, the *copas* estimation can maximally fill the number of studies that were actually observed. In other words, the estimated NRR has an upper limit of 50%. This is also reflected in Figure 3.3, where the highest NRR estimation (not shown) was 0.47 from both censoring methods.

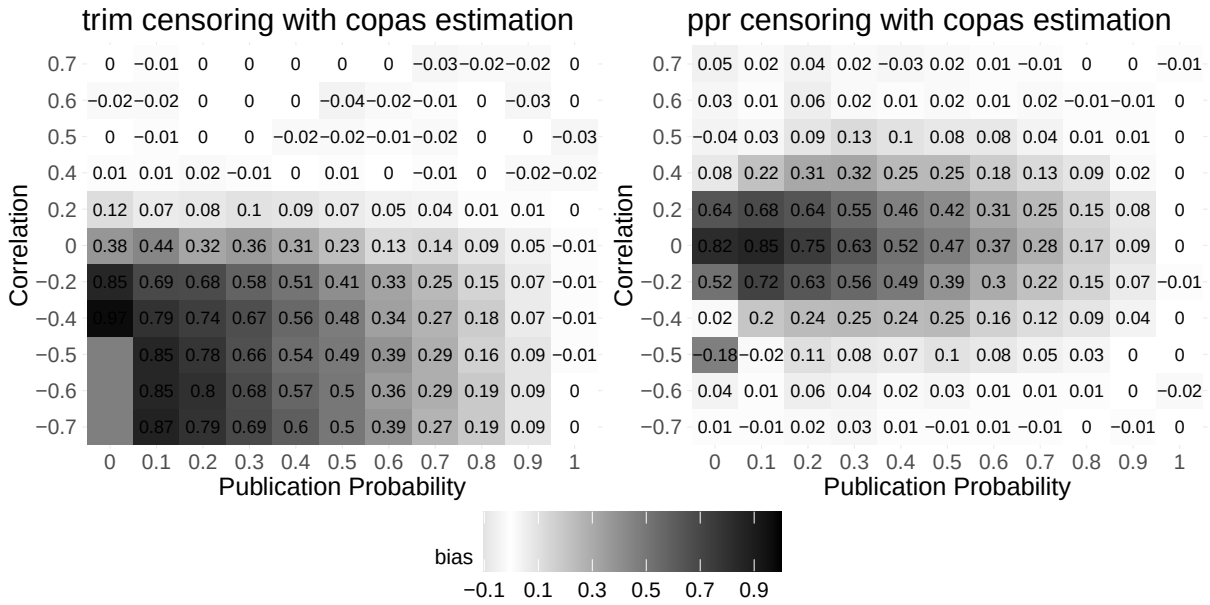

Figure 3.3: Heatmap for the bias of the *copas* estimation with *trim* censoring (left) and *ppr* censoring (right). A total of 121 conditions are tested for $s = 100$ studies and $m = 10$ simulation runs. Grayscale mapping and cell values indicate the mean of all bias over all simulation runs per conditions (bias). Bias is calculated as the true non-reporting rate minus the estimated non-reporting rate. Correlation equals to Pearson's correlation coefficient ρ, i.e., to $\beta \times 0.2$, as described in equation (2.2). Publication probability π determines the proportion of censored studies from the respective censoring mechanism. Cells without values have no available numerical result for the condition.

Results for the Truncated Normal Distribution Model

The results for the trunctad normal distribution model ("*trunc*") estimation can be seen in Figure 3.4. For the *trim* and *ppr* censoring, the *trunc* estimation method performs well in the same regions as the *copas* estimation: for low to high positive correlations and a publication probability of 100% under *trim*; and high positive or negative correlations and publication probability of 100% under *ppr*. In addition, the *trunc* method performs best for $\pi = 0$ and high negative correlation under *trim*, however only with a relatively small improvement. However, the method behaves worse than the *copas* method for *ppr* censoring at $\pi = 0$ and low absolute correlation. Over the whole grid, a MSE of 0.11 and 0.08 was calculated respectively for the *trim* and *ppr* censorings.

To investigate the possible boundaries of the NRR estimation for the *trunc* method, let us recap the calculation in equation (2.34) of omitted studies $\widehat{n}_{\widehat{\mu},\widehat{k}}$:

$$\widehat{n}_{\widehat{\mu},\widehat{k}} = \sum_{i=1}^{n} \Phi\left(\frac{\widehat{k} - \widehat{\mu}}{\mathrm{SE}(\widehat{\beta}_i)}\right),$$

for an estimated $\widehat{k}$ and $\widehat{\mu}$. In essence, the equation above sums all truncation probabilities for the observed studies which now only depend on the estimated parameters and their standard error. For instance, for $\widehat{k} = \widehat{\mu}$ every estimated p_i will be 0.5 and the estimated number of omitted studies will be $n/2$. With increasing $\widehat{k} - \widehat{\mu}$, the probabilities approach one. By contrast, with decreasing $\widehat{k} - \widehat{\mu}$ the denominator of the expression in $\Phi(\cdot)$ from the equation above gets

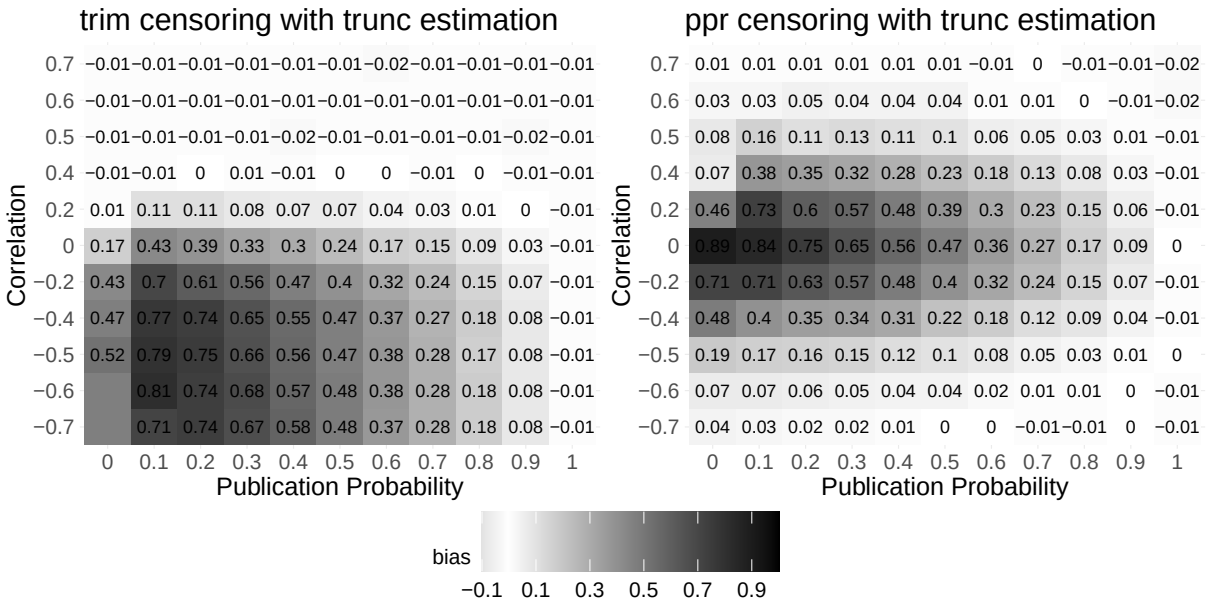

Figure 3.4: Heatmap for the bias of the *trunc* estimation with *trim* censoring (left) and *ppr* censoring (right). A total of 121 conditions are tested for $s = 100$ studies and $m = 10$ simulation runs. Grayscale mapping and cell values indicate the mean of all bias over all simulation runs per conditions (bias). Bias is calculated as the true non-reporting rate minus the estimated non-reporting rate. Correlation equals to Pearson's correlation coefficient ρ, which is $\beta \times 0.2$, as described in equation (2.2). Publication probability π determines the proportion of censored studies from the respective censoring mechanism.

negative and the probabilities approach zero. As such, the estimated number of omitted studies is bounded between $(0, n)$, as for the Copas selection model visible in equation (2.30). Since $\widehat{n}_{\widehat{\mu},\widehat{k}}$ is bounded between $(0, n)$, the estimated NRR has an upper boundary of 50%. This is also reflected in Figure 3.4, where the highest NRR estimation (not shown) was 0.5 from both censoring methods.

3.1.3 Comparison of Estimation Methods

Overall, the behavior of the three tested estimation methods was comparable and none of the methods was able to achieve low bias across the grid. None of the methods overestimated the non-reporting rate for low censoring; while the *copas* and *trunc* method had a higher precision in this region. However, they could not accurately predict the NRR for moderate to high censoring. The within-comparison of the methods is listed in Table 3.1. For the *trim* and *ppr* censoring, the trim-and-fill method achieved the lowest MSE. The theoretical upper bounds for the NRR estimation is 0.25 for the *fill* method and 0.5 for the other two. Here, an approximation was used for large n for the *fill* method. Comparatively spoken, the *copas* method performs best for small correlation ρ and publication probability π; whereas the *trunc* method performs best for the *trim* censoring for small ρ and π.

As described in Section 1.3, the reporting rate of individual animal experiments can be as low as 26% based on previous findings. In addition, our pilot study found a reporting rate of 31% on request-level, described later in Section 3.2. To compare the methods qualitatively under such an extreme condition, we need to evaluate the performance of the estimation methods at reporting

Table 3.1: Overall within-comparison of the three statistical non-reporting rate (NRR) estimation methods (Method). The Mean Squared Error (MSE) is calculated over the whole grid with equation (2.25) for the *trim* and *ppr* censoring, respectively. The NRR bound indicates the theoretical upper proportion bound the method is able to estimate.

Method	MSE (*trim*, *ppr*)	NRR bound	Comment
fill	(0.1, 0.06)	1/3	NRR bound approximated
copas	(0.12, 0.07)	0.5	best for *ppr* with small ρ and π
trunc	(0.11, 0.08)	0.5	best for *trim* with small ρ and π

rates of approximately 30%. This is identical to investigate the methods' bias at non-reporting rates of 70%, i.e., at this underlying censoring rate (GT.mean). In Figure 3.1 we can see that censoring rates closest to these values were simulated for low publication probability in both censoring regimes. For the *trim* censoring, the closest values were found in the left lower part of the heatmap, for pairs of $(\pi \in [0.1, 0.2], \rho = -0.2)$ and $(\pi = 0.3, \rho \in [-0.7, 0.4])$. For the *ppr* censoring, the mid left part of the heatmap best reflected a non-reporting rate of 70%. With the pairs coming closest at $(\pi \in [0.1, 0.2], |\rho| = 0.2)$ and $(\pi = 0.3, \rho = 0)$. Under these conditions, the performance of all estimation methods was poor. It can be appreciated graphically in Figures 3.2, 3.3 and 3.4 that all methods failed to accurately capture the true non-reporting rate there. In addition, MSEs were calculated for all methods at the these pairs (not shown) and the performances were comparable across the estimation methods. In summary, all methods failed to estimate the correct number of omitted studies in a scenario reflecting the publication landscape in animal research.

3.2 Results of the Study Request Tracking

3.2.1 Overview of Tracking Request Results

The aim of tracking study requests through to publications was to examine discrepancies between animal research applications and their resulting articles across different endpoints. We were interested in the actual (non-)reporting rate of the sample of 36 applications. In addition, the proportion of published from used and, e.g., requested animals were calculated. For this purpose, a PubMed query was filed for each of the application IDs, and the remaining articles after a multi-stage selection process were used to calculate associated endpoints. The results of all steps are depicted in the flow chart in Figure 3.5. Briefly, the automatic query efficiently reduced the pool of applications with potential applications featuring animals from (36, 11374) to (23, 162), respectively. The influences of individual parts of the query architecture on the number of retrieved publications and the results from keyword matching are seen in Section 3.2.2. A manual screening was then carried out to check and verify the results of the automatic screening. This step further reduced the application and publication counts to (11, 26), respectively. The detailed outcomes of the manual screening are visible in Section 3.2.3. Ultimately, non-reporting rates on request-level and animal-level of 69% and 79% were calculated, respectively. The latter from manually extracted animal counts. All results of the endpoint calculations are visible in Section 3.2.4.

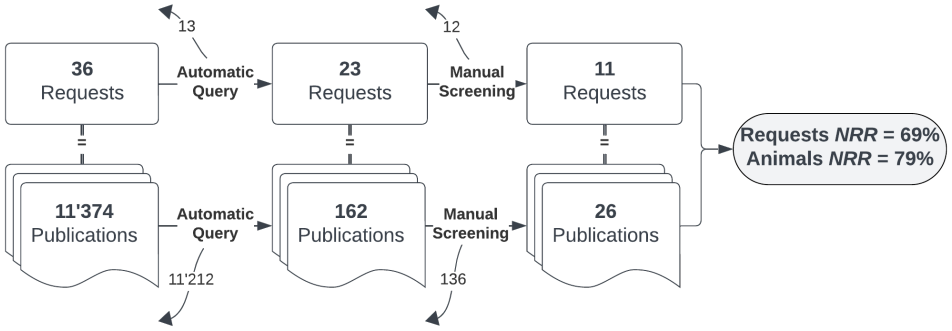

Figure 3.5: Flow chart showing the total number of requests and corresponding publications per sequential step (left to right). Fields connected with an equal sign are from the same stage. Numbers inside the fields indicate the step's remaining count, whereas numbers on arrows pointing outwards indicate removed applications or publications. The Automatic Query step includes the programmatic PubMed query and keyword-matching algorithm. Requests NRR refers to the non-reporting rate on request-level, i.e., 25/36 requests did not lead to any publication. Animal NRR is the proportion of animals found in the literature (published) from the "used" number in all Public Reports.

3.2.2 Automatic Query Results

To examine the performance of the automatic query, the number of retrieved abstracts (hits) was totaled over all requests. In particular, the total number of hits was compared across different query architectures and to the counts after keyword matching. Table 3.2 gives an overview of the total number of hits across conditions. We can clearly see that the complete query drastically reduced the number of Raw Hits compared to the crude author list filing. With the complete query string, only 238 Raw Hits were observed, representing 2.09% of the former. Without specifying any of the *Variable* search fields, the hits were narrowed down from 11374 to 3894, which corresponds to 34.24% of the Author Hits. In addition, I found that including the *UZH* and *Inst* fields lead to a greater reduction in article numbers than the *CH* or *MSH* terms. In particular, the *CH* term did not filter out any publication that had not already been removed by the other search fields. Thus, the query string without the *CH* element resulted in the same number of Raw Hits as the complete query string. By contrast, the *UZH* term filtered out a substantial portion, confining the Raw hits from 571 to 238, or respectively: 5.02 to 2.09% of the Author List.

After the automatic PubMed query, the keyword matching process further refined the selection of abstracts that were kept for the manual screening. The Raw Hits of the complete PubMed query were confined by the keyword matching algorithm from 238 to 162, as visible in Table 3.2. Ultimately, the automatic process thus delivered 162 hits, representing a 70-fold reduction from the Author Hits, or, 1.42% thereof. Note that all upcoming results were created with the complete query. All remaining publications belonged to a total of 23 application IDs. In other words, 13 requests were automatically excluded from the starting pool by the search field information and the keyword-matching algorithm. Figure 3.6 (left) shows the remaining frequency of individual hit counts across all request titles after keyword matching. For instance, 3 requests had 5 remaining publications. The range of counts for publications to be screened manually was between 0 and 20 across requests. On the other hand, the distribution of keywords in the application titles was analyzed. Figure 3.6 (right) shows the top 10 most frequently extracted keywords across all requests can be seen in Figure 3.6. For instance, the lemmatized "brain" is leading the list and was observed 12 times across all application titles. Overall, the

Table 3.2: Total number of retrieved abstracts (hits) from all requests per condition. The Query String either included i) all *Variable* search field tags ("Complete"), ii) removed individual ones indicated after "No" or iii) removed all ("None"). "No MSH" removes "Animals" together with the MeSH tag. "No CH" removes "Switzerland", "No Inst" removes the request-specific institutional name and "No UZH" removes the (nested) search field specifying "UZH", in each case with the affiliation tag. The "Raw Hits" and "Matched Hits" represent the numbers after the query or additionally after keyword matching, respectively. The percentages in brackets indicate the proportion from the 11374 Author Hits. The "Author Hits" are the number of hits when only the author names were used in the query string.

Query String	Raw Hits (%)	Matched Hits (%)
Complete	238 (2.09)	162 (1.42)
No CH	240 (2.11)	163 (1.43)
No MSH	266 (2.34)	176 (1.55)
No Inst	433 (3.81)	311 (2.73)
No UZH	571 (5.02)	402 (3.53)
None	3894 (34.24)	2282 (20.06)

top 10 keywords were counted 277 times over all requests and reflect the neurology research area of the requests.

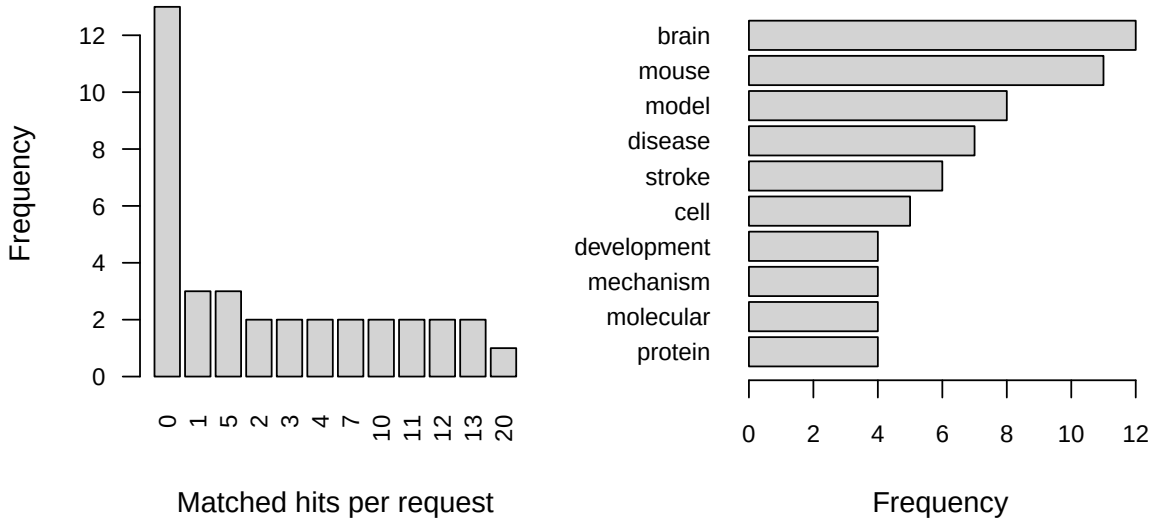

Figure 3.6: Left: Keyword-matched hit counts distribution. The frequencies indicate how many times a specific hit count was observed across all 36 requests after keyword matching. There was a total of 162 matched hits. Right: Top 10 most frequent keywords across request titles. The frequency indicates how many times the keyword was observed across the 36 request titles. All top keywords together were counted 277 times.

3.2.3 Manual Screening Results

The manual screening further reduced the number of publications from 162 to 26. The remaining articles were considered as actually stemming from the subset of 11 application IDs. Table 3.3 gives an overview about the justifications why a publication was in- or excluded during the manual review. A total of 136 publications were excluded during the manual review, corresponding to an exclusion proportion of 84%. The most common reason for an exclusion was that a publication did not match the field of study implied from the application title. The second most frequent reason to remove an article from the downstream analysis was that the cantonal authorization number in the publication did not match the one on the publication. In addition, there were several articles in the Abstracts Raw list that performed animal experiments on another species than indicated in the *Applications* file. On the other hand, a total of 26 publications were included for the downstream analysis, corresponding to an inclusion proportion of 16%. The sole justification for an undisputed inclusion was the presence of one of the two UIDs - the cantonal or the national number. As such, 5 publications were unambiguously identified as stemming from the corresponding ID. However, a greater amount of 21 publications were kept without a unique identification, labeled as miscellaneous. Over all manually screened articles, the cantonal or national license number were only reported in 50 cases of all of the screened publications, representing a license number reporting rate of 30.86%.

Table 3.3: Overview of the justifications for the ex- (left) and inclusion (right) of articles during manual screening. Mismatched veterinary office refers to offices from cantons other than Zurich. Miscellaneous refers to publications that were kept, but without a matching UID from the application information. The total amount of manually screened publications was 162.

Excluded ($n = 136$)	
Mismatched field of study	64
Mismatched cantonal number	45
Mismatched species	21
Mismatched veterinary office	6

Included ($n = 26$)	
Matched cantonal number	4
Matched national number	1
Miscellaneous	21

3.2.4 Tracking Endpoint Results

The remaining 26 publications after the manual screening belonged to a total of 11 application IDs. This allows us to calculate the first endpoint: a non-reporting rate on request-level of 69% from the original 36 applications, most of them filed in 2016. The counts of articles stemming from an application ranged from 1 to 4. Next, we were interested in the trends of the animal counts across publications. Table 3.4 shows an overview over the four consecutive animal count types ("steps"): requested, approved, used and published; as well as their respective proportion to the preceding step. The published counts refer to those manually extracted from the remaining articles. The other three count types come from the information given in the *Public Report* file. Table 3.5 shows the same for the subset of requests that resulted in at least one publication ("*successful*" requests). For instance, a total amount of 2730 animals was published across all *successful* requests seen in Table 3.5. This corresponds to a percentage of 10.71% from the total number of requested animals in the subset of remaining requests. On the other hand, the number of published animal equals to 68.3% from the total used animals from the Public Reports in the *successful* subset.

In order to investigate the performance of an LLM in the manual extraction task, ChatGPT-4 was used. It calculated 948 fewer animals than I did. The inconsistencies arose due to instances

Table 3.4: Overview of total animal counts from all 36 received requests. The used, approved and requested counts stem from the *Public Report* file. Published represents the total number that was extracted manually from 26 publications. The Step percentage is the Count proportion from step $i+1$ to i, while the Requested step is set to 100%. For example the Used from Approved step percentage is calculated from 13316/55789. The Published from Requested percentage amounts to 4.72%.

Step	Type	Count	Step [%]
1	Requested	57799	100.00
2	Approved	55789	96.52
3	Used	13316	23.87
4	Published	2730	20.50

where animals from certain publications belonged to more than one license number. The LLM could not make these distinctions with the current chat text. In addition, several publications did not list their used animal counts directly within the full-text content of the research article. Thus, the LLM was not capable of extracting correct numbers there. In those cases I consulted the supplementary data, or, other data respositories, such as the Gene Expression Omnibus (GEO) of the NCBI. Furthermore, I observed that the LLM was not able to fully account for all animal counts listed in figures. Finally, ChatGPT-4 also missed the total number of animals listed in non-methodological sections sometimes. Overall, "we" agreed only on 2 instances of the calculated number of animals.

Table 3.5: Overview of total animal counts from the 11 requests with at least one publication. The used, approved and requested counts stem from the *Public Report* file of the remaining requests. Published represents the total number that was extracted manually from 26 publications. The Step percentage is the Count proportion from step $i + 1$ to i, while the Requested step is set to 100%. For example the Used from Approved step percentage is calculated from 3995/23655. The Published from Requested percentage amounts to 10.71%.

Step	Type	Count	Step [%]
1	Requested	25480	100.00
2	Approved	23655	92.84
3	Used	3995	16.89
4	Published	2730	68.34

To investigate the trends for the different animal count types, I employed various plots for an exploratory data analysis for the remaining IDs. The scatter plot matrix in Figure 3.7 depicts the relationship among the four animal count types. For instance, the plot illustrates that the pairwise correlations and trends are comparable for all types. All densities are heavily right-skewed, indicating that individual applications with very high animal numbers exist for every count type, comparatively speaking. In addition, it can be seen that the approved and requested counts have a high correlation. Figure 3.8 depicts a combined visualization of the relationship of the four count types on request-level. In the scatter plot on the left, we can observe that most requests have less published animals than used, approved or requested. However, interestingly, I found that the IDs: 1, 8 and 10 have more published animals than e.g., indicated as "used" on the

Public Report file. In addition, we see that many approved counts exactly match the requested amounts. This is also reflected in the normalized bar plot in Figure 3.8 (right), which depicts the proportion of published, used and approved counts from the requested number. Here, it is visible that most requests only published and used a small fraction of animals, compared to the approved or requested numbers. The published count from the Request ID 1 was removed, since it is exceeding 100% from the requested amount.

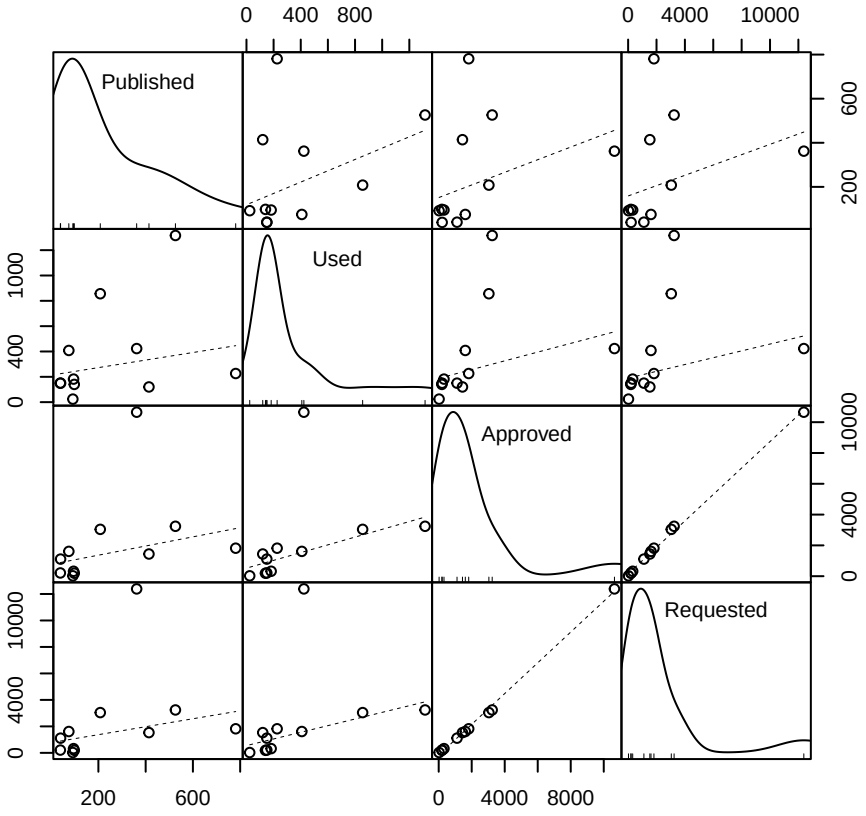

Figure 3.7: Scatter plot matrix of the four animal count types: published, used, approved and requested. The grid represents the matrix of all possible variable combinations, e.g., cell (3,1) plots the published counts on the x-axis and the approved counts on the y-axis. Each off-diagonal plot displays a robust regression line estimated by `MASS::rlm()` of the bivariate scatter plot. The diagonal plots showcase the density of individual variables, as kernel density estimates.

In conlusion, the tracking of 36 requests until publication on PubMed resulted in a total of 26 retrieved studies. Those studies stem from 11 applications, representing the non-reporting rate of 69% as the first endpoint. The comparison of the four animal count types showed that only a minor fraction of the requested animals are published or used. Please note that all these results are provisional, since the current implementation still poses several limitations, as described in detail in the discussion in Section 4.2.

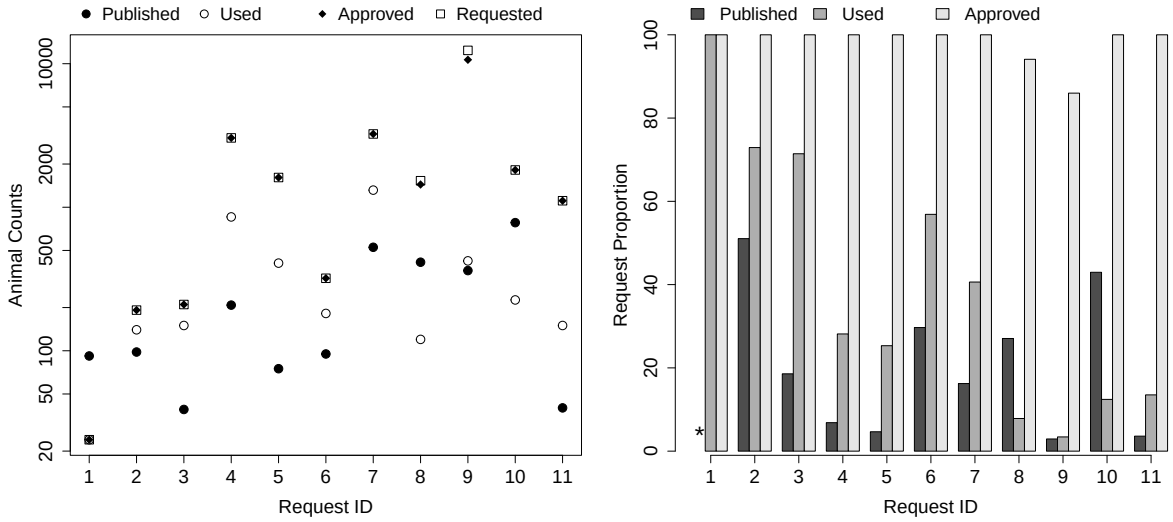

Figure 3.8: Request-specific investigation of animal count composition. The scatter plot (left) illustrates the association of the four different animal counts (logarithmic axis) per anonymized Request ID. The bar plot (right) showcases the proportion of each animal count category to the requested amount per request. The published request proportion of Request ID 1 of 383.3% is removed in the bar plot since it exceeds 100%, indicated with "*".

Chapter 4

Discussion

4.1 Evaluation of Methods to Estimate Non-reporting Rates

In the simulation at hand, the performance of estimating the non-reporting rate of the trim-and-fill algorithm, the Copas selection model and the truncated normal distribution were evaluated. With Monte Carlo simulations, publication probabilities and true underlying effects were varied. In conclusion, all methods failed to quantify publication bias for high censoring conditions. Contrarily, none of the methods did greatly overestimate non-existing or low levels of publication bias. These results were confirmed analytically, as we showed that the trim-and-fill method can only estimate non-reporting rates up to a third with increasing number of studies observed. For the other two methods, we showed mathematically that the applicability of them is limited as soon as the amount of censored studies exceeds the observed ones. By converting the treatment effect to correlation in the simulation, the strength and direction of the linear relationship between the treatment and the outcome is normalized. An advantage of our approach is thus that it allows to span the entire range of meaningful values of effect size and publication probability. Hence, we were able to investigate corner cases, which are usually not evaluated. For instance, in the simulation study performed by Schneck (2017), only correlations corresponding to low or medium positive effect sizes were chosen. Negative and/or strong effect sizes were neglected. Also other simulation studies with the aim to investigate publication bias investigate only subsets of all possible conditions. For example, in the study performed by Moreno et al. (2009), severe publication bias is declared when 25% of non-significant studies are not reported.

Even the original articles introducing the publication bias detection methods often assume that not many studies are censored, proportionally spoken. For instance, Duval and Tweedie (2000) refer to 13% (10/75) as a 'large number of suppressed studies' in one condition of their walk-through example. However, theoretical reporting rates can get much lower for specific conditions, as seen in the simulation in Section 3.1.2. Furthermore, we know from data abroad that reporting rates can reach levels as low as 26% on animal-level or 60% on study request level (van der Naald et al., 2020), as described in Section 1.3. Finally, also our pilot study revealed a non-reporting rate of 31% (Section 3.2), which is also substantially lower than assumed by many methods. Since we investigated the whole realm of relevant values, we were able to evaluate exactly these conditions that reflect meta-analysis in the preclinical animal research context. With reporting rates that low, all the investigated publication bias correction methods are incapable of correctly estimating the non-reporting rate for the most part. We understand that the main goal of these methods is to estimate the true treatment effect rather than the severity of the publication bias or some associated metric, such as the non-reporting rate on animal-level. However, the estimation of the number of omitted studies is a central part of all selected algorithms. Therefore, it is not surprising that the performance to accurately correct the distorted effect largely correlates with the performance for the non-reporting rate, as exemplified in Figure A.2.

Of course, these results are partly explained by the increasingly low number of studies available when e.g., one decreases the publication probability. Nonetheless, it is clear that these methods are of limited use in estimating the extent of publication bias under animal research conditions. We found that all models cannot precisely estimate the severity of publication bias observed in the request tracking in Section 3.2.

Another advantage of the implemented process is its flexibility and extensibility. Since R is a scripting language, other users can write their own functions for any of the four modular steps described in Section 2.1.1. This is in stark contrast to works that used point-and-click software for their simulation, such as SPSS or Stata, as in the studies from Schneck (2017) and Moreno et al. (2009), respectively. With this flexibility, current limitations of the simulation can be overcome. For instance, instead of a linear relationship in the *Simulate* step, a more widely used model could be used: the four-parameter logistic (4PL) model. This model is most famously used to calculate e.g., the half maximal inhibitory concentration (IC_{50}) of novel drug formulations. The 4PL model was already developed by Volund (1978), but is arguably still today the most important regression tool for preclinical bioassays for sigmoidal dose-response curves (ElHarouni et al., 2022). With the current simulation setup, the linear model could easily be exchanged with a 4PL fit by writing your own fitting method or simply using an existing package, such as `dr4pl` (Landis et al., 2021). Furthermore, the switch between data generating models could further be facilitated by implementing the transformation model from Hothorn et al. (2018). Also the other steps in the algorithm benefit from the inherent modular extensibility. For instance, number of studies and/or observations can readily be adapted to suit different meta-analytic research questions or stages thereof. But also more complex functionality could be added, such as the re-run of analyses mimicking p-hacking (Elliott et al., 2022). Finally, the whole simulation was implemented with parallel computing which enables bearable run times on local machines. Thus, there is no need for server-side computation.

It is clear that the heterogeneous conditions with regard to the number of observations and studies, as well as effect sizes, do not allow an absolute comparison and the heatmaps are of rather qualitative than quantitative nature. It is also clear that when input parameters or functions within the modules of the algorithm are changed, the output will also change. However, there is reason to assume that the overall picture will remain similar. Statistical estimation methods will not be able to accurately estimate non-reporting rates for increasingly small numbers of available studies, no matter which regression model is used or how many observations are simulated. To illustrate this point: the selected truncated normal distribution approach is inherently modelling the selection process during censoring as it happens in the simulation. During the Aggregate step the estimated treatment effects become normally distributed, as seen in equation (2.5). For *trim* censoring - graphically spoken - the density of the estimated effect is then vertically cut at zero (at least for publication probability equal to zero) - corresponding to a truncated normal distribution. Despite the correct assumptions about the effect distribution, the *trunc* method is not capable of estimating the non-reporting rates correctly. This is one the one hand explained by the small amount of studies available for severe publication bias conditions. On the other hand, the *trunc* method - as well as the other two - fails to recapitulate the additional publication probability parameter. On top of that, a general downside of any post-hoc method to correct meta-analytic results is that it can only be done based on the nature of the results. In the real world, however, not all correlation between precision and effect size is explained by publication bias associated with results, as explained in Section 1.2. Conclusively, we were able to show that all of the tested methods only work adequately well if a low proportion was missing for the fixed number of tested studies. The limitations of these methods to give a precise estimate of publication bias, reinforces the need for a tracking system of requests to publications. Only such evaluations lead to an unbiased assessment in retrospect and provide insights that can counteract

the waste of animal lives.

4.2 Appraisal of the Process for Tracking Study Requests

With this pilot study we implemented a first process to estimate the publication rate given a set of animal research applications in Switzerland. The process consists of several steps, both in an automatic and a manual part. In conclusion, the automatic process efficiently reduced the number of publications to be screened and thus greatly facilitates the tracking of requests. In addition, this trial run allowed us to identify several limitations and potential improvements for future applications. First, several problems arose with the received data files from the FSVO. We were not able to parse the crude SQL exports (.txt) from the DWH ALVPH data warehouse without excluding many free-text fields manually, prior to the import into R. In addition, as mentioned, I realized that one of the requests was duplicated in our sample. Most likely explained by an import of the file into animex.ch from the preceding application: "e-tierversuche". The preceding web application contained a different application ID naming convention. In addition, the data structure was initially not fully comprehensible, and minor discrepancies between the files received and the FSVO database were identified up to and including the end of this work. For instance, three *Valid Dates* were identified by the FSVO as having been incorrectly propagated to our exports. However, due to time constraints, it was not possible to carry out the queries again. In any case, the three dates did not differ substantially from the used ones and it can safely be assumed that the overall picture would not have changed. Nevertheless, the data transfer between the FSVO and the future data analysts should be improved.

First of all, I would propose to create and transfer a database dump (e.g., a .sql file) rather than a .txt file. Such .sql dumps contain not only the data but also the database structure. These dumps can be readily generated by tools such as pgAdmin for PostgreSQL databases. This approach would ensure the integrity of the database when the .sql file is imported. In other words, the database can be recreated by the data analyst exactly as it was at the time of the dump. Second, it would be beneficial to receive access to the animex.ch web application. This idea was mentioned to me by a staff of the FSVO and it was indicated that a training environment of animex.ch exists. Access would allow the data analyst to better understand the structure, components and functionalities of animex.ch. In addition, corner cases could be investigated directly in the software. Finally, more specific meta data and a codebook would be helpful. Meta data, such as on which basis requests to be analyzed were selected can be important to identify potential errors and for systematic analysis. A codebook is important to support the collaborative data understanding and serves as a reference document. For instance, it was not entirely clear to me what all the individual column variables stood for or how they were calculated. Aside from the structure of the export, the eligibility of the contained requests should be asserted in future applications. For this work, all received study requests were considered as legitimate, since the main goal is to propose a process. For future endeavors, eligibility criteria should be applied to the requests. Criteria could include the non-consent from researcher(s), requests for non-experimental studies, technical failures or not carrying out the experiment.

The programmatic PubMed query poses several advantages over a manual literature search, as done by ter Riet et al. (2012); Wieschowski et al. (2019). Among others, the abstract database is screened in an automatic fashion which renders the process more efficient, consistent and reproducible. However, also the automatic query could be improved going forward. First of all, all potential literature databases and publication types should be screened. So far, only PubMed and journal articles were searched for, since PubMed primarily focuses on providing access to scientific literature. However, there are also further publication types that could stem from the

requests. For example, conference papers should be taken into account and ideally searched in the Embase database, as this has a broader coverage of conference records than PubMed. In addition, patents should also be considered. Here, the Swissreg platform, which contains the Swiss patent register could be screened. To further improve the automatic query, the code should also be tested. It must be ensured that my programmatic approach does not miss a publication due to an implementation error. For instance, unit tests should be applied on artificially created mock data for corner cases or known request-hits relations. In addition, the pool of publications that were excluded due to a specific search field tag should be reviewed to ensure that no articles were incorrectly excluded due to the absence of this tag. I observed the latter point in the past (not for this work) for a general MeSH field, i.e., an article was found about animal experimentation without the "Animals" field and MeSH tag. If such situations would occur regularly for a specific tag, it should be considered to remove the corresponding element from the query. By contrast, it could also be envisioned to extend the query to include more search fields or to be based on full-text content. Finally, author name disambiguity in query search results for PubMed is a known phenomenon. However, there is no easy way to programatically solve this issue without the availability of unique author identifiers (Liu et al., 2014). Unique author identifiers (AUID) would actually be available in PubMed, but in my experience they are not often used.

The keyword matching algorithm further reduced the pool of publications before the manual screening. As such, it further contributed to the efficiency of the process. For me, it seemed as a legitimate assumption that at least one of the titles' keywords can be found in the publication's abstract. However, also the keyword matching could benefit from further refinement and control mechanisms. For example, I noted that many publications were included that did not match the species on the application title. Therefore, it could be an improvement to extend the current keyword matching rule. At the moment, the algorithm checks if *any* of the extracted keywords matches a word in the abstract. However, since the species is decisive, it could be enforced that extracted animal species keywords must match. In any case, a kind of standardized test should also be used here for future applications to ensure that no items are systematically and wrongly excluded. For refinement, the requested animal species from the *Applications* file could be used for the keyword matching. Finally, to include a pattern matching for the license number (if provided) would definitely improve the current process as well.

The manual screening further confined the pool of remaining publications and allowed me to gain valuable insights how to improve the process. However, as mentioned in the beginning of this section, many free-text columns were removed from the data fields during the export at the FSVO. Unfortunately, the information contained therein would have been helpful. The manual in- or exclusion of articles was particularly challenging when there were several requests stemming from the same lab. The research questions were then often quite similar and more information on the applications would have helped to unambiguously attribute the articles to the corresponding request. Next, the animal counts from the remaining articles were extracted manually. Also the manual extraction of animal counts might be prone to inaccuracies due to various factors. Firstly, assumptions about the experimental sets were necessary at times, especially when explicit numbers were not listed, potentially leading to estimations that are imprecise. Additionally, the context of whether the numbers were representing repeated or alone-standing experiments was not always clear to me. It is therefore a prerequisite of future applications of this project to include a second reviewer, as it is usually done for systematic reviews. Finally, ChatGPT-4 was not very consistent in the calculation of animal counts. The submission of the same order twice was observed to result in different numbers at times. However, it can be envisioned that a properly trained LLM would be more accurate and efficient than a human reviewer. With such a tool, a complete end-to-end workflow could theoretically be set up by integrating the automatic query into an API of the LLM. This would theoretically work for the numbers

available in the full-text content. However, such an approach was beyond the scope of this thesis.

The endpoint calculations revealed a very high publication bias - on request-level and even more so on animal-level. In addition, the sample of applications resulted in large discrepancies between animal counts on the application and the resulting publications. However, I need to emphasize one more time that these results are provisional and the main aim of this thesis was to implement a first version of a process for potential future applications. As mentioned in this section, many parts are still missing to systematically complete this project. In particular, all results with manual results should be taken with a grain of salt. In addition, also the endpoint calculation could be refined, to e.g., calculate numbers of animals stratified to species. Furthermore, sending surveys to the involved researcher after the statistics have been calculated would be a very good quality control instrument to ensure that all publications and results were actually detected and that the process is unbiased. In any case, the thesis at hand serves as an excellent starting point to continue assessing statistics on animal research together with the FSVO. The ten most important points for improving the process are available as a checklist in the Appendix B. I am convinced that such an evaluation in Switzerland will be of great value and could eventually even lead to informed policy decisions to improve the preclinical scientific rigor, as it was the case for the Netherlands after a similar research project (van der Naald et al., 2022).

Let me conclude this thesis with food for thought: in an ideal world in which all scientific data is universally accessible and well-structured, conducting meta-analyses would be basically effortless. No publication bias would exist and summary statistics were readily computed. However, reality presents us with unstructured and incomplete data from scientific articles, hindering this ease. If the data were only unstructured but complete, automated solutions as the presented one could still enable automatic research queries based on available results with levity. However, lack of transparency and neglecting guidelines are known drivers for data unavailability (Kwasnicka et al., 2021). Also this work revealed a lack of compliance with guidelines, notably in the non-disclosure of animal experiment license numbers in the majority of articles, which is not compliant with the ethical ARRIVE guidelines (Item 14) (Percie du Sert et al., 2020). Moreover, sifting through supplementary content for vital data felt burdensome and error prone to me. When scientific data faces such hurdles in accessibility and reusability, it is akin to dealing with data unavailability despite its existence. The issue of data unavailability was already introduced in the beginning of this thesis in Section 1.2 in the context of publication bias and meta-analytic result aggregation. There, I explained how wrong incentives for publications, such as newsworthiness, can lead to a distortion of the knowledge base. Now it is reinforced in a more general meaning: science is ineffective if any data is not readily accessible. The open science movement aims to transform scientific practices for better transparency and reproducibility (Elliott and Resnik, 2019). Embracing this movement means easier access to resources and stricter adherence to data quality standards (Levin et al., 2016). For me personally, the most important demand therein is a rethinking of academia's credit system, moving away from emphasizing publication numbers and citations. Instead, the focus should return to using publications primarily as a means to communicate research. Science with robust experiments following openness and transparency will undoubtedly lead to a faster knowledge evolution and more impactful contributions, while wasting less valuable resources including laboratory animals. It is time to shift from "Publish or Perish" to "Foster and Flourish".

Appendix A

Additional Figures

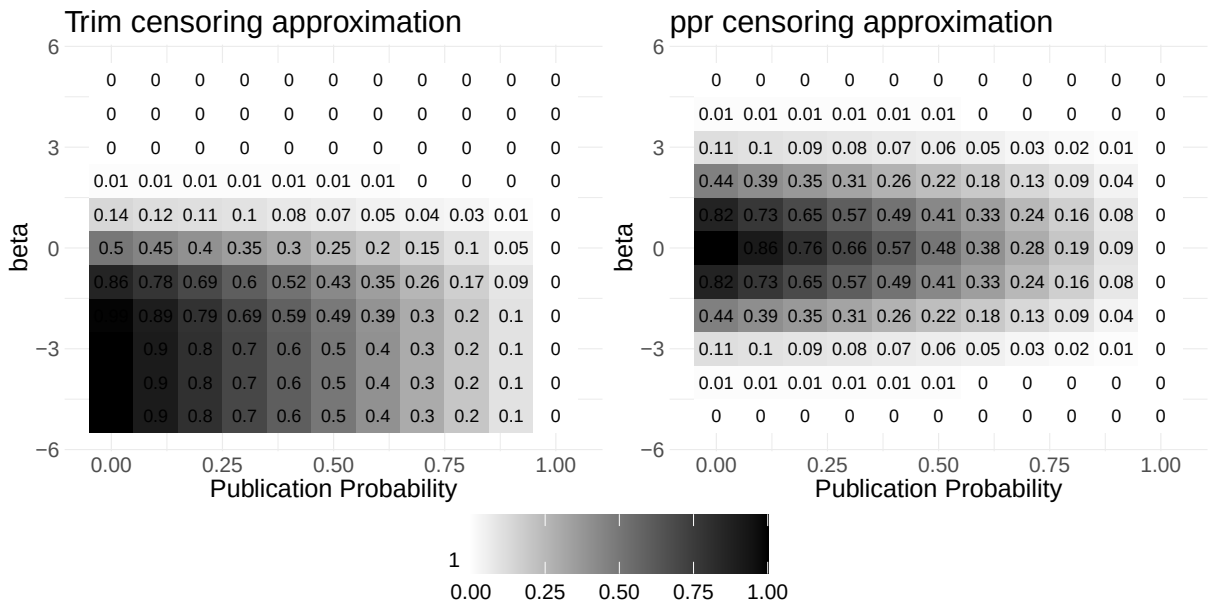

Figure A.1: Approximation of the censoring rates. For trim (left) equation (2.24) and for ppr (right) equation (2.19) were used.

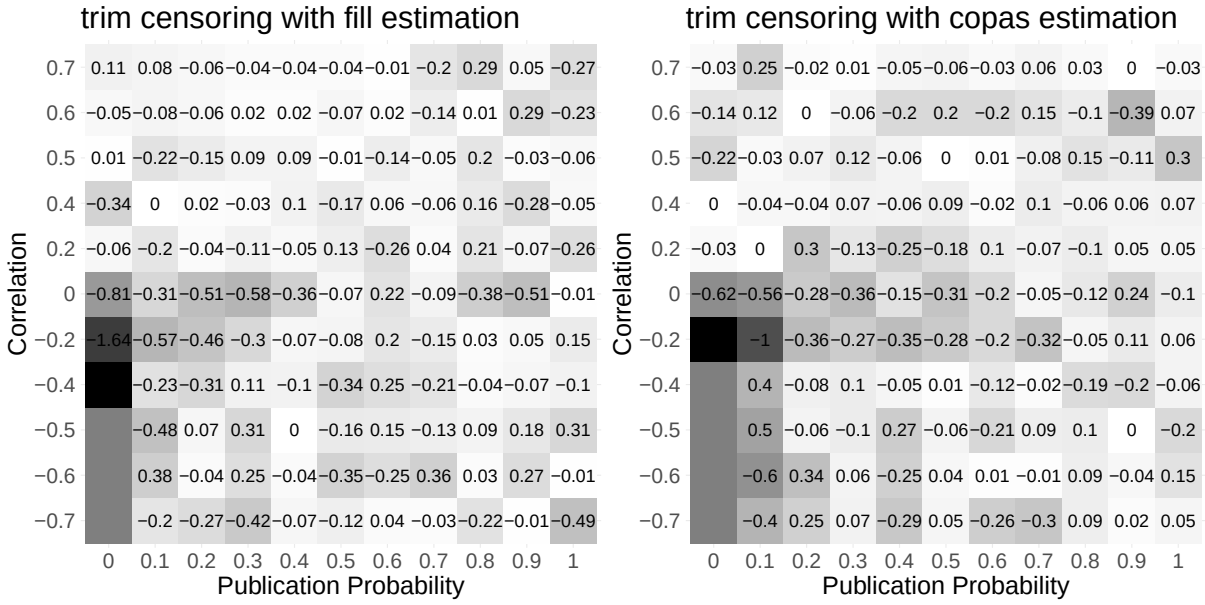

Figure A.2: Bias of the estimated treatment effect $\widehat{\beta}$ from the true simulation input parameter β for the trim- and fill algorithm (left) and the Copas selection model (right) under trim censoring .

Appendix B

Improvements for Tracking Study Request Process

1. Data transfer facilitation between FSVO and data analysts by .sql dump.
2. Receive access to animex-ch training environment for data analysts.
3. Receive codebook for FSVO database (export).
4. Receive meta data for the set of applications.
5. Extend automatic literature search scope (e.g., Embase, patents).
6. Implement unit tests for keyword-matching and automatic PubMed query.
7. Implement pattern matching for license number in keyword-matching.
8. Perform manual publication screenings by more than one reviewer.
9. Stratify endpoint calculation to e.g., species.
10. Send survey to researchers for post-hoc analysis of endpoints.

Appendix C

Computational Details

This document was generated on Dezember 09, 2023 at 17:48. R version and packages used to generate this report:

```
sessionInfo()
## R version 4.2.1 (2022-06-23)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 22.04.3 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.10.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_CH.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=de_CH.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=de_CH.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_CH.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.39
##
## loaded via a namespace (and not attached):
##  [1] compiler_4.2.1  magrittr_2.0.3  cli_3.6.1       tools_4.2.1
##  [5] glue_1.6.2      rstudioapi_0.13 vctrs_0.6.4     stringi_1.7.6
##  [9] highr_0.9       stringr_1.5.0   xfun_0.30       lifecycle_1.0.3
## [13] rlang_1.1.1     evaluate_0.15
```

Appendix D

Code

```
##############################################################################
##  Data generating script, containing all four steps of the simulation:
##  1.) simulate.datasets function
##  2.) create.metadata function
##  3.) censor.metadata function
##  4.) estimate function
#######################
## Additionally: definition of utility functions such as SE, corr, beta...
##############################################################################

##############################################################################
#function to generate simulate data sets
##############################################################################

simulate.datasets <- function(m = 1, s = 100, beta = 0,
                              sigma_e = 10, sigma_x = 2, seed = NULL){
  #########
  # outputs a list of m datasets consisting of s studies each
  # sample size n is sampled randomly for each study
  #########
  # m : simulation repetitions
  # s : number of studies per simulation
  # sigma_e : error standard deviation
  # sigma_x : standard deviation of treatment
  #########

    # initialize seed (as in stats:::simulate.lm)
    if (!exists(".Random.seed", envir = .GlobalEnv, inherits = FALSE))
      runif(1)
    if (is.null(seed))
      RNGstate <- get(".Random.seed", envir = .GlobalEnv)
    else {
      R.seed <- get(".Random.seed", envir = .GlobalEnv)
      set.seed(seed)
      RNGstate <- structure(seed, kind = as.list(RNGkind()))
      on.exit(assign(".Random.seed", R.seed, envir = .GlobalEnv))
    }

  #initialize list for m data sets
  lds <- vector(mode = "list", length = m)

  for(k in 1:m){#for all simulations
    #sample sizes (OBSERVATIONS) of each of the s studies
    observations.per.study <- sample(x = 10:50, size = s, replace = T)
    #number of total observations all studies of the meta analysis
    n.observations <- sum(observations.per.study)
    #study indices for all observations
    study <- rep(1:s, observations.per.study)
    #simulate outcomes for all observations as a linear dependency
    treatment <- rnorm(n.observations, 0, sigma_x) #x
    outcome <- beta * treatment + rnorm(n.observations, 0, sigma_e) #y

    #replicate theoretical sd for all observations
    sd_e <- rep(sigma_e, n.observations)
    sd_x <- rep(sigma_x, n.observations)

    #store simulated data sets in dataframe, combined in list
    lds[[k]] <- data.frame(study = as.factor(study),
                           outcome,
                           treatment,
                           sd_e,
                           sd_x,
                           n.obs = rep(NA, n.observations),
                           p = rep(NA, n.observations),
                           p.theoretical = rep(NA, n.observations),
                           ppr = rep(NA, n.observations),
                           beta = rep(beta, n.observations)
                           )
  }
  # attach seed to return value (as in stats:::simulate.lm)
  attr(lds, "seed") <- RNGstate #retrieve via attr(ds, "seed")
  return(lds)
```

```
}

##############################################################################
#function to calc summary statistics for each study and combine in meta df
##############################################################################
create.metadata <- function(ds = ds.raw.test){
  ## input: data sheets (observation level)
  ## output: meta data array () with treatment effect (beta) from lm

  #get studies indices from data sheet
  sorted.studies <- sort(unique(ds$study))

  meta.frame <- c()
  for(j in sorted.studies){#for each study calculate summary statistics
    #indices of all observations of current studies
    ind <- which(ds$study == j)

    #calculate mean of outcome, emp.sd, number of observations per study
    outcome <- mean(ds$outcome[ind])
    sd.emp <- sd(ds$outcome[ind])
    n.obs <- nrow(ds[ind,])
    se <- sd.emp/sqrt(n.obs)

    #propagate theoretical standard deviation used
    sd.theo <- ds$sd_e[ind][1]

    #propagate beta used
    beta <- ds$beta[ind[1]]

    #fit lm model and calculate p-value of slope coefficient
    fit <- lm(outcome ~ treatment, data = ds[ds$study == j,])
    p <- summary(fit)$coefficients["treatment","Pr(>|t|)"] #p for beta.hat
    b.hat <- coefficients(fit)[2]
    se.beta.hat <- summary(fit)$coefficients["treatment", "Std. Error"]

    #get theoretical p-value from normal distribution
    dm <- model.matrix(fit) #design matrix (2x2)
    var.beta <- solve(t(dm) %*% dm)[2,2] #get variance of beta
    sigma_e <- ds$sd_e[ind][1] #true standard deviation
    z <- b.hat/(sigma_e*sqrt(var.beta)) #test stat Z under null hypothesis
    p.theoretical <- 2*(1 - pnorm(abs(z))) #

    #store summary statistics of each study
    sum.stats <- data.frame(study = j,
                        outcome,
                        sd.emp,
                        sd.theo,
                        n.obs,
                        se,
                        p,
                        p.theoretical,
                        beta,
                        b.hat,
                        se.beta.hat)

    #append to meta data frame
    meta.frame <- rbind(meta.frame, sum.stats)
  }
  return(meta.frame)
}

##############################################################################
# function to simulate publication bias
##############################################################################
censor.metadata <- function(mdat = mdat.all.test,
                            pi = 0, alpha = 0.05,
                            method = "trim", beta.threshold = 0){
  # function that publishes studies with Probability = pi
  ##########
  # pi: publication probability of an insignificant finding
  ##########

  if(method == "ppr"){
  # studies to be dropped
  studies.not.significant <- which(mdat$p > alpha) #significant studies
  drop <- sample(studies.not.significant,
                 size = (1 - pi) * length(studies.not.significant))
  censor.method <- "ppr"
  }
  if(method == "trim"){
  ## studies to be dropped
    #get indices for negative studies
  studies.low.beta <- which(mdat$b.hat < beta.threshold)
  drop <- sample(studies.low.beta, size = (1 - pi) * length(studies.low.beta))
  censor.method <- "trim"
  }
  #drop pi studies from original meta frame
  mdat.pubbias <- mdat[(mdat$study %in% drop) == FALSE,]
  if(nrow(mdat.pubbias) != 0){
  mdat.pubbias$ppr <- pi
  mdat.pubbias$corr <- corr(mdat.pubbias$beta)
  mdat.pubbias$censoring <- censor.method
  }
  return(mdat.pubbias)

}
```

```r
##############################################################################
######## Estimate non-reporting rate via trim-and fill and compare to GT
##############################################################################

estimate <- function(metadata.pubbias = mdat.pubbias,
                     metadata.all = mdat.all.test, method = "fill"){

  # get non-reporting rate NRR ground truth
  NRR.GT <- 1- nrow(metadata.pubbias)/nrow(metadata.all)

  # calculate weighted average (of beta)
  weighted_average <- function(data, se) {
    # Calculate the inverse variance weights
    weights <- 1 / (se^2)

    # Calculate the weighted average
    weighted_avg <- sum(data * weights) / sum(weights)
    return(weighted_avg)
  }
  #calculated weighted average of beta as NON-CORRECTED estimate
  beta.weighted.avg <- weighted_average(data = metadata.pubbias$b.hat,
                                        se = metadata.pubbias$se.beta.hat)

  # Assign NA if less than three studies are available in the PB data
  if(nrow(metadata.pubbias) < 3){
    NRR.est <- NA
    beta.est <- NA
    return(output <- list("NRR.est" = NRR.est,
                          "NRR.GT" = NRR.GT, "beta.est" = beta.est,
                          "beta.weighted.avg" = beta.weighted.avg))
  }

  ################### trim and fill ###################
  #####################################################
  if(method == "fill"){
    ### fit FE meta-analysis model and carry out trim-and-fill analysis
    require(metafor)
    require(meta)
    rma.res <- metagen(TE = b.hat,
                       seTE = se.beta.hat, studlab = study,
                       data = metadata.pubbias,
                       control=list(stepadj=0.4, maxiter=100000))
    ##taf analysis
    taf <- metafor::trimfill(rma.res)

    #number of missing studies and all studies inclusive missing ones
    n.missing <- taf$k0
    n.all <- taf$k.all

    #estimated non-reporting rate NRR
    NRR.est <- n.missing/n.all
    beta.est <- as.numeric(taf$beta) #for metafor::rma()
    beta.est <- as.numeric(taf$TE.fixed) #for meta::metagen()
  }

  ################ Copas selection model ################
  #######################################################
  if(method == "copas"){
    require(meta)
    #transform data frame to meta class
    m <- metagen(TE = b.hat, seTE = se.beta.hat,
                 studlab = study, data = metadata.pubbias,
                 control=list(stepadj=0.4, maxiter=100000))

    #fit selection model
    require(metasens)
    copa.m <- copas(m, warn = -1)
    n.input <- nrow(metadata.pubbias)
    n.missing <- copa.m$N.unpubl.adj
    # n.missing <- sum(copa.m$N.unpubl)
    NRR.est <- n.missing/(n.input + n.missing)
    beta.est <- copa.m$TE.adjust
    funnel.plot <- NA
  }

  ################ Truncated normal distribution ################
  ###############################################################
  if(method == "trunc"){ #truncation based on effect
    #assign SE (s), estimated betas (x) and number of obs. studies (n)
    x <- metadata.pubbias$b.hat
    s <- metadata.pubbias$se.beta.hat
    n <- nrow(metadata.pubbias)

    #estimate truncation point (eq 8 in article)
    k.hat <- min(metadata.pubbias$b.hat)

    #define mono-parameteric ll function
    ll<-function(mu){
      sum(log(dnorm((x - mu)/s))) - sum(log(1 - pnorm((k.hat - mu)/s))) #eq 9
    }

    ##Visualize likelihood (optional)
    # mu.grid <- seq(-10, 10, length.out = 100)
    # y <- sapply(mu.grid, ll)
    # plot(mu.grid, y, type = "l", main = "log-likelihood")
```

```
    ##find estimated mean (mu.hat) via maximizing log-likelihood
    opt <- optim(par=0, ll, method = "BFGS", control = list(fnscale = -1))
    mu.hat <- opt$par

    ##calculate truncation proportion (nu.estim) via sum over probs (eq 9-10)
    nu.estim <- sum(pnorm((k.hat - mu.hat)/s))
    NRR.est <- nu.estim/(n + nu.estim) #adapted from eq. 10
    beta.est <- mu.hat
  }

  ################ truncation based on p-value
  ##########################################
  if(method == "trunc.p"){
    x <- metadata.pubbias$b.hat
    s <- metadata.pubbias$se.beta.hat
    t.hat <- min(x/s) #eq 8
    n <- nrow(metadata.pubbias)
    ll<-function(mu){
      sum(log(dnorm((x - mu)/s))) - sum(log(1 - pnorm((t.hat - mu)/s))) #eq 9
    }

    #find parameters as before
    opt <- optim(par=1, ll, method = "BFGS", control = list(fnscale = -1))
    mu.hat <- opt$par
    nu.estim <- sum(pnorm(t.hat - (mu.hat/s)))
    NRR.est <- nu.estim/(n + nu.estim) #adapted from eq. 10
    beta.est <- mu.hat
  }
  if(length(beta.est == 0)){beta.est <- NA}
  output <- list("NRR.est" = NRR.est, "NRR.GT" = NRR.GT,
                 "beta.est" = beta.est,
                 "beta.weighted.avg" = beta.weighted.avg)
  return(output)
}

#############################################################################
######## Utility functions
#############################################################################

## Standard error
se.calc <- function(x){
  se <- sd(x)/sqrt(length(x))
  return(se)
}

## calculate beta (treatment effect) from correlation
beta.corr <- function(corr, sigma_x = 2, sigma_e = 10){
  beta.hat <- (corr*sigma_e)/sigma_x
  return(beta.hat)
}



##calculate correlation (Pearson)
corr <- function(beta, sigma_x = 2, sigma_e = 10){
  corr <- (beta*sigma_x)/(sqrt(beta^2*sigma_x^2 + sigma_e^2))
  stopifnot(corr >= -1)
  stopifnot(corr <= 1)
  return(corr)
}

## Generate test files and check output
ds <- simulate.datasets(m = 10, beta = 3, s = 100)
ds1 <- ds[[1]]
ds2 <- ds[[2]]
ds3 <- ds[[3]]
ms1 <- create.metadata(ds1)
ms2 <- create.metadata(ds2)
cs1 <- censor.metadata(mdat = ms1, method = "trim", pi = 0)
e1 <- estimate(cs1, ms1)
cs2<- censor.metadata(mdat = ms1, method = "trim", pi = 0)
cs3 <- censor.metadata(mdat = ms2, pi = 0.5)
output <- estimate(cs1, ms1)
```

```
#############################################################################
##  Sim script, containing all functions to run combinations in parallel:
##  1.) Create.combinations: generates parameter combinations
##  2.) Simulate.all: Estimates bias (DGP steps) for all combinations
##  3.) run.parallel: run "simulate.all" parallel and return results
#############################################################################


#############################################################################
#Create.combinations
#############################################################################

create.combinations <- function(m = 10, beta = seq(-5, 5, length.out = 11),
                                s = c(100), sigma_e = c(10),
                                pi = seq(0, 1, length.out = 11),
                                censor.method = "trim",
                                estimate.method = "fill"){

  source("Scripts/01_DGP.R")
  ## retreive parameters from input
```

```
  all.paras.list <- list("m" = m, "beta" = beta,
                         "s" = s, "sigma_e" = sigma_e, "pi" = pi,
                         "censor.method" = censor.method,
                         "estimate.method" = estimate.method)

  ##expand all parameter values to input.combination data frame
  n.paras <- 0
  for (para in all.paras.list) {
    n.paras <- n.paras + length(para)
  }
  n.para.values <- n.paras #number of parameter values
  para.names <- names(all.paras.list) #parameter name list
  n.paras.names <- length(names(all.paras.list)) #number of parameter names
  ##expand input parameter combinations to grid
  input.combinations <- expand.grid(all.paras.list)
  input.combinations$correlation <- with(input.combinations, corr(beta = beta))

  return(input.combinations)
}

combins <- create.combinations() #generate combinations

##############################################################################
#Simulate all function
##############################################################################

# define function
simulate.all <- function(n.sim, input.beta,
                         input.s, input.sigma_e, input.pi,
                         censoring, estimation, input.seed = NULL){
  #create m lists with metadata sets
  ds.list <- simulate.datasets(m = n.sim, beta = input.beta,
                               s = input.s, sigma_e = input.sigma_e,
                               seed = input.seed)
  metdat.all.list <- lapply(ds.list, create.metadata)
  metdat.pubbias.list <- lapply(metdat.all.list,
                                censor.metadata, pi = input.pi,
                                method = toString(censoring))

  ##for each simulation, store est and GT in df
  #initialize sim df
  sim.df <- as.data.frame(matrix(data = NA, nrow = n.sim, ncol = 4))
  names(sim.df) <- c("est", "GT")
  for (i in 1:n.sim){
    estimation.output <- estimate(metadata.all = metdat.all.list[[i]],
                                  metadata.pubbias = metdat.pubbias.list[[i]],
                                  method = toString(estimation))
    est <- estimation.output$NRR.est
    GT <- estimation.output$NRR.GT
    beta.est <- estimation.output$beta.est
    beta.weighted.avg <- estimation.output$beta.weighted.avg
    sim.df[i, 1] <- est
    sim.df[i, 2] <- GT
    sim.df[i, 3] <- beta.est
    sim.df[i,4] <- beta.weighted.avg
  }

  #calculate mean of the estimates and standard error thereof
  est.mean <- colMeans(sim.df, na.rm = TRUE)[1]
  est.se <- se.calc(sim.df[,1])
  GT.mean <- colMeans(sim.df, na.rm = TRUE)[2]
  GT.se <- se.calc(sim.df[,2])

  output <- list("est.mean" = est.mean, "est.se" = est.se,
                 "GT.mean" = GT.mean, "GT.se" = GT.se, "sim.df" = sim.df,
                 "beta.est" = beta.est,
                 "beta.weighted.avg" = beta.weighted.avg)
  return(output)
}

##############################################################################
#run parallel function
##############################################################################

run.parallel <- function(input.combinations){##Note: min combinations: 2
  ## apply sim function in parallel for all combinations and store results
  require(foreach)
  require(doParallel)

  source("Scripts/01_DGP.R")
  source("Scripts/02_simulation.R")

  # Set up a cluster with multiple cores
  cl <- makeCluster(detectCores()-1)

  ##cluster export needed functions
  clusterExport(cl, c("censor.metadata", "corr", "create.metadata",
                      "estimate", "se.calc", "simulate.all",
                      "simulate.datasets"))

  # Register the cluster for parallel computation
  registerDoParallel(cl)

  # Apply the function in parallel using foreach
  results <- foreach(i = 1:nrow(input.combinations),
                     .combine = rbind) %dopar% {
```

```
    simulate.all(input.combinations[i, 1], input.combinations[i, 2],
                 input.combinations[i, 3], input.combinations[i, 4],
                 input.combinations[i, 5], input.combinations[i, 6],
                 input.combinations[i, 7])
  }

  # Stop the cluster
  stopCluster(cl)

  ################################### Store in ANALYSIS DATA FRAME

  ##initialize analysis df
  # n.paras <- ncol(input.combinations) - 1 #-1 bc correlation is calculated
  analysis.header <- c(names(input.combinations), "est.mean",
                       "est.se", "GT.mean", "GT.se",  "bias",
                       "beta.est", "beta.weighted.avg")
  #initialize analysis df
  analysis.df <- as.data.frame(matrix(data = NA,
                                      nrow = nrow(input.combinations),
                                      ncol = length(analysis.header)))
  names(analysis.df) <- analysis.header
  #transfer parameter values to analysis df
  analysis.df[,1:(ncol(input.combinations))] <- input.combinations

  #calculate summary statistics
  analysis.df$est.mean <- unlist(results[,1])
  analysis.df$est.se <- unlist(results[,2])
  analysis.df$GT.mean <- unlist(results[,3])
  analysis.df$GT.se <- unlist(results[,4])
  analysis.df$beta.est <- unlist(results[,6])
  analysis.df$beta.weighted.avg <- unlist(results[,7])
  analysis.df$bias <- analysis.df$GT.mean - analysis.df$est.mean #calc bias
  analysis.df$beta.bias <- analysis.df$beta - analysis.df$beta.est

  ################################### write to csv

  # Define base file name with current date and time
  base_name <- "sim/analysis_df"

  ## add censoring and estimation method name
  censoring <- toString(unique(analysis.df$censor.method))
  estimation <- toString(unique(analysis.df$estimate.method))
  base_name <- paste(base_name, censoring, estimation, sep = "_")
  file_name <- paste0(base_name, ".csv")

  # Define file name variable and check if file already exists
  i <- 1
  while (file.exists(file_name)) {
    i <- i + 1
    file_name <- paste0(base_name, "_", i, ".csv")
  }

  # Save data to CSV file with the created file name
  write.csv(analysis.df, file = file_name, row.names = FALSE)
  print("run.parallel complete")
}

##To run:
# head(combins) #check input combinations
# run.parallel(combins) #run from command line after saving this file
```

```
##############################################################################
##  Analysis script, containing all functions for analysis:
##  1.) generate_heatmap: generates heatmap from csv output (02_simulation)
##  2.) make.funnel.plot: generates contour-enhanced funnel plot
##############################################################################

##############################################################################
## generate_heatmap
##############################################################################

generate_heatmap <- function(file.name, result.type = "bias",
                             censoring = "trim",
                             estimation = "trunc", scale = TRUE) {
  #add with path, e.g. file.name <- "sim/analysis_df_trim_fill.csv"
  require(ggplot2)
  require(dplyr)

  # Load the data from the CSV file
  data <- read.csv(file.name)

  #round correlation to one digit
  data$correlation <- round(data$correlation, 1)

  # Perform data transformations
  data <- data %>%
    #round all numerics to 2 digits
    mutate_if(is.numeric, ~ round(., digits = 2)) %>%
    #transform pi and corr to factor
    mutate(pi = as.factor(pi), correlation = as.factor(correlation))

  #calculate MSE
  f.mse <- function(est.input, gt.input){
    tab <- cbind(est.input,gt.input)
```

```
    tab2 <- na.omit(tab)
    est <- tab2[,1]
    gt <- tab2[,2]
    result <- 1/length(est)*sum((est - gt)^2)
    return(result)
  }
  mse <- f.mse(data$est.mean, data$GT.mean)

  if(scale == TRUE){
  #heatmap scale defintion
  Bias.range = c(-0.1, 1)
  Bias.breaks = seq(-0.1,1, by = 0.2)
  }
  else{
    Bias.range = NULL
    Bias.breaks = NULL
  }

  ## Generic heatmap
  if(result.type == "GT.mean"){gg.title <- paste(censoring, "censoring")}
  else{gg.title <- paste(censoring, "censoring with",
                         estimation, "estimation")}
  ggplot <- ggplot(data, aes(x = pi, y = correlation,
                             fill = get(result.type)))+
    ggtitle(gg.title) +
    geom_tile() +
    geom_text(aes(label= get(result.type)), size = 6) +
    # geom_raster(aes(fill = bias), interpolate=TRUE) +
    scale_fill_gradient2(low = "black", high = "black",
                         limits = Bias.range, breaks = Bias.breaks) +
    labs(x = "Publication Probability", y = "Correlation",
         fill = result.type) + theme_minimal()
  return(list("plot" = ggplot, "mse" = mse, "data" = data))
}

###store heatmaps for all conditions:
#only censoring
hm1 <- generate_heatmap("sim/analysis_df_trim_fill_2.csv",
                        result.type = "GT.mean", censoring = "trim",
                        estimation = "fill")
hm2 <- generate_heatmap("sim/analysis_df_ppr_fill_2.csv",
                        result.type = "GT.mean", censoring = "ppr",
                        estimation = "fill")

#with estimation
hm3 <- generate_heatmap("sim/analysis_df_trim_fill_2.csv",
                        result.type = "bias", censoring = "trim",
                        estimation = "fill")
hm4 <- generate_heatmap("sim/analysis_df_ppr_fill_2.csv",
                        result.type = "bias", censoring = "ppr",
                        estimation = "fill")
hm5 <- generate_heatmap("sim/analysis_df_trim_copas_2.csv",
                        result.type = "bias", censoring = "trim",
                        estimation = "copas")
hm6 <- generate_heatmap("sim/analysis_df_ppr_copas_2.csv",
                        result.type = "bias", censoring = "ppr",
                        estimation = "copas")
hm7 <- generate_heatmap("sim/analysis_df_trim_trunc_2.csv",
                        result.type = "bias", censoring = "trim",
                        estimation = "trunc")
hm8 <- generate_heatmap("sim/analysis_df_ppr_trunc.p_2.csv",
                        result.type = "bias", censoring = "ppr",
                        estimation = "trunc")

##### Heatmaps for the slope coefficient bias
hmA1 <- generate_heatmap("sim/analysis_df_trim_fill_2.csv",
                         result.type = "beta.bias", censoring = "trim",
                         estimation = "fill", scale = FALSE)
hmA2 <- generate_heatmap("sim/analysis_df_ppr_fill_2.csv",
                         result.type = "beta.bias", censoring = "ppr",
                         estimation = "fill", scale = FALSE)
hmA3 <- generate_heatmap("sim/analysis_df_trim_copas_2.csv",
                         result.type = "beta.bias", censoring = "trim",
                         estimation = "copas", scale = FALSE)
hmA4 <- generate_heatmap("sim/analysis_df_ppr_copas_2.csv",
                         result.type = "beta.bias", censoring = "ppr",
                         estimation = "copas", scale = FALSE)
hmA5 <- generate_heatmap("sim/analysis_df_trim_trunc_2.csv",
                         result.type = "beta.bias", censoring = "trim",
                         estimation = "trunc", scale = FALSE)
hmA6 <- generate_heatmap("sim/analysis_df_ppr_trunc.p_2.csv",
                         result.type = "beta.bias", censoring = "ppr",
                         estimation = "trunc", scale = FALSE)

## calculate MSE for publication probabilities reflecting animal research
mse.pi <- function(pi = 0.3, hm){
  # pi <- 0.3
  # hm <- hm3
 #initialize pi-indices assignment tab
 pi.axis <- seq(0, 1, by = 0.1)
 hm.indices <- matrix(data = seq(121), ncol = 11, nrow = 11, byrow = T)
 pi.indi.tab <- as.matrix(cbind(as.numeric(pi.axis), hm.indices))

 #select indices for input pi
 ind.sel <- pi.indi.tab[which(pi.indi.tab[,1] ==  toString(pi)),
                        2:ncol(pi.indi.tab)]
```

```
 bias <- na.omit(hm$data$bias[ind.sel]) #bias from selection

 #calculate mse
 mse <- 1/length(bias)*sum(bias^2)
 # mse

 return(mse)
}
##############################################################################
## make.funnel.plot
##############################################################################

make.funnel.plot <- function(){
require(tidyverse)
require(meta)
require(dmetar)

data(ThirdWave)
m.gen <- metagen(TE = TE,
                 seTE = seTE,
                 studlab = Author,
                 data = ThirdWave,
                 sm = "SMD",
                 fixed = FALSE,
                 random = TRUE,
                 method.tau = "REML",
                 hakn = TRUE)

# Define fill colors for contour
col.contour = c("gray75", "gray85", "gray95")

# Generate funnel plot (we do not include study labels here)
funnel.meta(m.gen, xlim = c(-1, 2),
            contour = c(0.9, 0.95, 0.99),
            col.contour = col.contour, ylab = "Precision",
            xlab = "Estimated Effect")

# Add a legend
legend(x = 1.2, y = 0.05,
       legend = c("n.s.","p < 0.1", "p < 0.05", "p < 0.01"),
       fill = c("white", "gray75", "gray85", "gray95"))

# Add a title
title("Contour-enhanced funnel plot")
}

##############################################################################
## Function to test censoring approximation (eq. 2.28 & 2.33)
##############################################################################

#################PPR (eq. 2.28)
ppr.cens.approx <- function(){
ppr <- function(para){
  pi <- para[1]
  beta <- para[2]
  ncp <- beta*sqrt(30)*2/10
  tcrit <- qt(0.975, df = 28, ncp = 0)
  p <- pt(abs(tcrit), df = 28, ncp = ncp) - pt(-tcrit, df = 28, ncp = ncp)
  ppub <- (1-pi)*(p)
  return(ppub)
}

pi <- seq(0,1, by = 0.1)
beta <- seq(-5,5, by = 1)
grid <- as.matrix(expand.grid(pi, beta))
# names(grid) <- c("pi", "beta")
res <- round(matrix(apply(grid, 1, ppr), 121, 1),2)
# res <- apply(grid,1,ppr)
tab <- as.data.frame(cbind(grid, res))
names(tab) <- c("pi", "beta", "res")
# ppub(0.1, -5)

ggplot <- ggplot(tab, aes(x = pi, y = beta, fill = res))+
  ggtitle("ppr censoring approximation") +
  geom_tile() +
  geom_text(aes(label= res), size = 6) +
  # geom_raster(aes(fill = bias), interpolate=TRUE) +
  scale_fill_gradient2(low = "white", high = "black") +
  labs(x = "Publication Probability", y = "beta", fill = res) +
  theme_minimal()
return(ggplot)
}
ppr.cens.fig <- ppr.cens.approx()

################Trim (eq. 2.32)

trim.cens.approx <- function(){
phi <- function(para){
  pi <- para[1]
  beta <- para[2]
  p <- pnorm(-beta*sqrt(30)*2/10)
  ppub <- (1-pi)*p
  return(ppub)
}

pi <- seq(0,1, by = 0.1)
```

```
beta <- seq(-5,5, by = 1)
grid <- as.matrix(expand.grid(pi, beta))
# names(grid) <- c("pi", "beta")
res <- round(matrix(apply(grid, 1, phi), 121, 1),2)
# res <- apply(grid,1,phi)
tab <- as.data.frame(cbind(grid, res))
names(tab) <- c("pi", "beta", "res")
# ppub(0.1, -5)


ggplot <- ggplot(tab, aes(x = pi, y = beta, fill = res))+
  ggtitle("Trim censoring approximation") +
  geom_tile() +
  geom_text(aes(label= res), size = 6) +
  # geom_raster(aes(fill = res), interpolate=TRUE) +
  scale_fill_gradient2(low = "white", high = "black") +
  labs(x = "Publication Probability", y = "beta", fill = res) +
  theme_minimal()
return(ggplot)
}

trim.cens.fig <- trim.cens.approx()
```

```
##############################################################################
##  Analysis script with steps and functions for
##  local request tracking analysis:
##  1) code to load and process data exported from ThinLinc
##  2) Function for report result aggregation: res_table_aggregate()
##  3) Code for Latex table generation
##  4) Code for count barplots from manual screening results
##  5) Endpoint Calculations (+ barplots)
##############################################################################

##############################################################################
# 1) LOAD DATA FROM THINLINC ECPORT FOR REPORT AND ANALYZE
##############################################################################
require(car)

######## KEYWORDS ###########################
res_keywords <- read.csv("Scripts/Thinlinc_export/keywords.csv", header = T)
word_freq <- table(res_keywords)

# Sort the table by frequency
sorted_word_freq <- sort(word_freq, decreasing = TRUE)

# Display the top N words
top_n <- 10
# head(sorted_word_freq, n = top_n)

######## Results from Query ##########################
## table codebook:
## counts_raw = number of pubs only searching for authors
## count = number of pubs with complete query but before keyword matching
## counts_kw = number of pubs with complete query and after keyword matching

#whole query
res_table <- read.csv("Scripts/Thinlinc_export/result_table.csv", header = T)
#no Institution name
res_table_noInst <- read.csv("Scripts/Thinlinc_export/result_table_noInst.csv", header = T)
#no MeSH
res_table_noMesh <- read.csv("Scripts/Thinlinc_export/result_table_noMesh.csv", header = T)
#neither Inst nor Mesh
res_table_noAfilli <- read.csv("Scripts/Thinlinc_export/result_table_noAfilli.csv", header = T)
#No Comp
res_table_noCompany <- read.csv("Scripts/Thinlinc_export/result_table_noCompany.csv", header = T)
#neither Inst nor Mesh
res_table_none <- read.csv("Scripts/Thinlinc_export/result_table_none.csv", header = T)

####get number of IDs that had publicastoins
res_table_noRawHits <- res_table[res_table$count != 0, ]
res_table_noKw <- res_table[res_table$counts_kw != 0, ]

##############################################################################
# 2) Function for report result aggregation
##############################################################################
res_table_aggregate <- function(result_table){
  raw_count <- sum(result_table$counts_raw)
  query_count <- sum(result_table$count)
  kw_count <- sum(result_table$counts_kw)
  query_prop <- round(100*(query_count/raw_count),2)
  q_prop_str <- paste0("(", query_prop, ")")
  kw_prop <- round(100*(kw_count/raw_count),2)
  kw_prop_str <- paste0("(", kw_prop, ")")

  ## paste counts together with percentages
  query_res <- paste(query_count, q_prop_str, sep = " ")
  kw_res <- paste(kw_count, kw_prop_str, sep = " ")

  output <- list("raw_count" = raw_count, "query_res" = query_res,
                 "kw_res" = kw_res, "query_count" = query_count,
                 "kw_count" = kw_count, "query_prop" = query_prop,
                 "kw_prop" = kw_prop)
  return(output)
}
```

```
##############################################################################
# 3) Code for Latex table generation
##############################################################################
## generate table and convert to latex
require(xtable)

res_query_tab <- data.frame(matrix(ncol = 4, nrow = 6))
names(res_query_tab) <- c("Query String", "Author Hits",
                          "Raw Hits (%)", "Matched Hits (%)")
res_query_tab$`Query String` <- c("Complete", "No CH", "No MSH",
                                  "No Inst", "No UZH", "None")
res_query_tab[1,2:ncol(res_query_tab)] <- unname(res_table_aggregate(res_table))[1:3] #complete
res_query_tab[2,2:ncol(res_query_tab)] <- unname(res_table_aggregate(res_table_noAfilli))[1:3] #no Switzerland
res_query_tab[3,2:ncol(res_query_tab)] <- unname(res_table_aggregate(res_table_noMesh))[1:3]
res_query_tab[4,2:ncol(res_query_tab)] <- unname(res_table_aggregate(res_table_noInst))[1:3]
res_query_tab[5,2:ncol(res_query_tab)] <- unname(res_table_aggregate(res_table_noCompany))[1:3] #no UZH
res_query_tab[6,2:ncol(res_query_tab)] <- unname(res_table_aggregate(res_table_none))[1:3]
res_query_tab[] <- lapply(res_query_tab, as.character)

##generate xtable
aggregated_res_table <- res_table_aggregate(res_table)
aggregated_res_table_none <- res_table_aggregate(res_table_none)
aggregated_res_table_noComp <- res_table_aggregate(res_table_noCompany)
res_query_xtable <- xtable(res_query_tab)
res_query_xtable2 <- xtable(res_query_tab[,-which(names(res_query_tab) == "Author Hits")])

##############################################################################
# 4) Code for count barplots from manual screening results
##############################################################################
###after kw-matching
n_zeros <- sum(res_table$counts_kw == 0)
# hist(res_table$counts_kw, breaks = 12, xlab = "Matched Counts",
#      ylim = c(0,25), xlim = c(0,max(res_table$counts_kw)))
count_freq <- table(res_table$counts_kw)

# Sort the table by frequency
sorted_count_freq <- sort(count_freq, decreasing = TRUE)

###after manual screening
res_table_man <- read.csv("Scripts/Thinlinc_export/df_final.csv",
                          header = T) #whole query
res_table_man <- res_table_man[which(res_table_man$ID != "ac8cd4f0-ff27-4290-a2da-5b26430dc6e4"),]


total_sum_of_yes <- sum(res_table_man$man_count, na.rm = T)
prop_yes <- 100*(sum(res_table_man$man_count > 0)/nrow(res_table_man))

res_table_pmids <- read.csv("Scripts/Thinlinc_export/result_table_man_pmid.csv",
                            header = T) #whole query

##add manual counts to dfs
man_df_ex <- as.data.frame(matrix(NA, 1, 4))
names(man_df_ex) <- c("field", "cant", "species", "cvo")
man_df_ex[1,] <- c(65, 45, 21, 6)
yes_total <- sum(man_df_ex)

man_df_in <- as.data.frame(matrix(NA, 1, 3))
names(man_df_in) <- c("cant", "national", "misc")
man_df_in[1,] <- c(6, 1, 18)
no_total <- sum(man_df_in)

man_tot <- sum(man_df_ex) + sum(man_df_in)

n_uid <- man_df_ex$cant + man_df_in$national + man_df_in$cant
prop_uid <- round(100*n_uid/man_tot,2)


##############################################################################
# 5) Endpoint Calculations (+ barplots)
##############################################################################

endpoint_df <- read.csv("Scripts/Thinlinc_export/endpoint_result_table.csv",
                        header = T)
tot_used_GPT <- sum(endpoint_df$ChatGPT)
##tot_used_man
tot_used_man <- sum(endpoint_df$used_manual) #published
GT_tot_used <- sum(endpoint_df$used)
GT_tot_approved <- sum(endpoint_df$approved)
GT_tot_requested <- sum(endpoint_df$requested)
Published_ChatGPT <- sum(endpoint_df$ChatGPT)
tot_animals <- c(tot_used_man, GT_tot_used, GT_tot_approved, GT_tot_requested)
animals_df <- as.data.frame(matrix(tot_animals, ncol = 1, nrow = 4))
animals_df <- cbind(c("Published", "Used", "Approved", "Requested"), animals_df)
names(animals_df) <- c("Type", "Count")
idx_req <- which(animals_df$Type == "Requested")
idx_used <- which(animals_df$Type == "Used")
GPT_prop <- round(100*(Published_ChatGPT/animals_df$Count[idx_req]),1)
prop_pub_used <- round(100*(tot_used_man/animals_df$Count[idx_used]), 1)

animals_df$Percentage <- sapply(animals_df$Count,
                                function(x) round(100*(x/animals_df$Count[idx_req]),1))
animals_xtable <- xtable(animals_df)


bar_data <- matrix(c(endpoint_df$used_manual, endpoint_df$used,
                     endpoint_df$approved, endpoint_df$requested),
```

```
                  ncol = 4, byrow = F)
df_bar <- as.data.frame(bar_data)
names(df_bar) <- c("Published", "Used", "Approved", "Requested")

##barplot
bp_count <- function(){
pch_sel <- c(19, 1, 18, 0)
par(mar = c(5.1, 4.1, 7.1, 2.1), xpd = T)
plot(1:11, endpoint_df$used_manual, log = "y",
     type = "p", pch = pch_sel[1], xlab = "Request ID", ylab = "Animal Counts",
     ylim = range(c(endpoint_df$used_manual,
                    endpoint_df$used, endpoint_df$approved,
                    endpoint_df$requested)), xaxt = "n",
     cex = 1.5, cex.axis = 1.5, cex.lab = 1.5)
axis(1, at=1:11, labels=1:11, cex.axis = 1.5)
points(1:11, endpoint_df$used, type = "p", pch = pch_sel[2], cex = 1.5)
points(1:11, endpoint_df$approved, type = "p", pch = pch_sel[3], cex = 1.5)
points(1:11, endpoint_df$requested, type = "p", pch = pch_sel[4], cex = 1.5)
legend("topleft", legend = c("Published", "Used", "Approved", "Requested"),
       box.lty = 0, pch = pch_sel, cex = 1.4, horiz = T, inset = c(0,-0.1))
}

##barplot2
bp_count_2 <- function(){
par(mar = c(5.1, 4.1, 7.1, 2.1), xpd = T)
# bar_data_normalized <- apply(bar_data, 1, function(x){x*100/sum(x,na.rm=T)})
bar_data_normalized2 <- apply(df_bar, 1, function(x){x[1:3]*100/x[4]})
bar_data_normalized2[1,1] <- NA
bp <- barplot(bar_data_normalized2, xaxt = "n",
              xlab = "Request ID", ylab = "Request Proportion",
              cex.axis = 1.5, cex.lab = 1.5, beside = T)
axis(1, at=colMeans(bp), labels=1:ncol(bar_data_normalized2), cex.axis = 1.5)
text(bp[1,1], 0, "*", pos = 3, cex = 2, col = "black", las = 2)
myCols <- gray.colors(nrow(bar_data_normalized2))
legend("topleft", legend = c("Published", "Used", "Approved"),
       box.lty = 0, fill = myCols, cex = 1.4, horiz = T, inset = c(0,-0.1))
}
```

```
source("TL1_load_data.R")
source("TL2_tracking_requests.R")
source("TL3_manual_screening.R")

################################################################################
##  Thinlinc (TL) algorithm script that executes the tracking request analysis
##  with functions described in the helper scripts:
##  TL_1_load_data.R
##  TL_2_tracking_requests.R
##  TL_3_manual_screening.R
#########
##  Steps:
##  1) Create query input df
##  2) Make query
##  3) Save queries, make results -> save preliminary res
#########
##  4) Manual screening (walk_through function) [start from preliminary res]
################################################################################

#################### 1) Create input df for query ##########################
################################################################################

input_df <- load_data()$input_df
input_test <- load_data()$input_df_test

#################### 2) Perform query with all Variables ###################
################################################################################

q_valid <- make_query(input_df, affili = T, mesh = T)
q_noMesh <- make_query(input_df,mesh = F)
q_noInst <- make_query(input_df,institution = F)
q_noAffili <- make_query(input_df, affili = F)
q_noCompany <- make_query(input_df, company = F)
q_none <- make_query(input_df,mesh = F, institution = F,
                     affili = F, company = F)
qt <- make_query(input_test, affili = T, mesh = T)

#################### 3) Save queries, make results  ########################
################################################################################

##save queries
save_queries(q_valid)
save_queries(q_noMesh)
save_queries(q_noInst)
save_queries(q_noAffili)
save_queries(q_noCompany)
save_queries(q_none)

##make results from queries
result_table <- make_result_table(q_valid)
legacy_result_table <- make_result_table(q_loaded)
result_table_noMesh <- make_result_table(q_noMesh)
result_table_noInst <- make_result_table(q_noInst)
result_table_noAfilli <- make_result_table(q_noAffili)
result_table_noCompany <- make_result_table(q_noCompany)
result_table_none <- make_result_table(q_none)
```

```
##export non-personal data
export_function(result_table)
export_function(result_table_noMesh)
export_function(result_table_noInst)
export_function(result_table_noAfilli)
export_function(result_table_noCompany)
export_function(result_table_none)

################## 4) Add manual results from saved query ###################
#### START HERE AFTER AUTOMATIC QUERY WAS PERFORMED
#############################################################################

##load data
load_data()

##load queries
query_loading() #load q_valid

##make results from loaded query
res_loaded <- make_result_table(q_valid)

## get application IDs from loaded results
hits_IDs <- res_loaded$ID
# Get the index of the first occurrence bigger than zero
idx1 <- which(res_loaded$counts_kw > 0)[1]

#walk through all IDs after idx
# walk_through_strings(q_valid, hits_IDs[[idx1]])

#aggregate man_results
man_res <- aggregate_man_results(path_to_csv = "manuals/old",
                                 res_table = res_loaded)

##export justifications file
total_merged <- export_justifications_file()

#make and export endpoint table
endpoints <- make_endpoint_table() #not all manual counts used here
```

```
#############################################################################
##  Helper Script to TL_Algorithm.R with function to load and process data.
##  Includes:
##  1) load_data()
##  2) make_result_table()
##  3) save_queries()
##  4) export_function()
#############################################################################

######## 1) Function to load and process data to query input format ##########
#############################################################################

load_data <- function(){
  #######
  ## function to load an process data from 1) personell, 2) public report and
  ## 3) applications files.
  #######
  ## Output: 1) loaded data (global env) 2) input df for pubmed query
  #######


### LOADING AND PROCESSING  ##############
##########################################

##Use only Application IDs which are in the Public Report file
dat_PR <- read.table("final/ANIMEX_Grue_PublicReport_311023_2.txt", sep = "~",
                     skipNul = TRUE,
                     fill = TRUE, fileEncoding = "utf-16le", header = T)

##translate titles from german to english (manuallly..)
###Note from JF: censored for report

#save data to global environment
assign("dat_PR", dat_PR, envir = .GlobalEnv)


## Personell file
dat_personell <- read.table("final/ANIMEX_Grue_Personell_011123.txt", sep = "~",
                            skipNul = TRUE,
                            fill = TRUE, fileEncoding = "utf-16le",
                            header = T, quote = "")
dat_personell <- dat_personell[dat_personell$APPLICATION_ID %in% dat_PR$APPLICATION_ID,]
assign("dat_personell", dat_personell, envir = .GlobalEnv)

##get earliest date to include in PubMed query from "DECISION_DATE" of Application forms
dat_Applications_raw <- read.table("final/ANIMEX_Grue_Applications_011123.txt",
                                   sep = "~", skipNul = TRUE,
                                   fill = TRUE, fileEncoding = "utf-16le",
                                   header = T)
dat_Applications <- unique(dat_Applications_raw)
dat_Applications <- dat_Applications[dat_Applications$APPLICATION_ID %in% dat_PR$APPLICATION_ID,]
dat_Applications$INST_adapted <- gsub("[-,-].*", "", dat_Applications$INSTITUTE_NAME)

#save data in global environment
assign("dat_Applications", dat_Applications, envir = .GlobalEnv)
```

```
### Query input df generation #############
##########################################

## Aggregate all names to Application ID
input_df <- aggregate(cbind(authors = paste(FIRST_NAME, LAST_NAME, sep = " ")) ~ APPLICATION_ID,
                      data = dat_personell, FUN = function(x) paste(x, collapse = ", "))

## start merging data from different files for query
input_df <- input_df[input_df$APPLICATION_ID %in% dat_PR$APPLICATION_ID,]
titles_ID_authors <- sapply(input_df$APPLICATION_ID,
                            FUN = function(ID) dat_PR$TITLE_EN[dat_PR$APPLICATION_ID == ID])
input_df$title <- lapply(titles_ID_authors, FUN = function(x) x[1])
input_df$keyw <- sapply(input_df$title, keyword_extraction)

#filter for applications in public report
start_date_unformatted <- sapply(input_df$APPLICATION_ID,
  FUN = function(ID) min(dat_Applications$APPLICATION_VALID_FROM[which(dat_Applications$APPLICATION_ID == ID)]))

# Convert the date-time string to a date-time object
date_time_object <- sapply(start_date_unformatted,
                           strptime, format = "%Y-%m-%d %H:%M:%OS")
start_dates <- sapply(date_time_object, format, "%Y/%m/%d")

# Extract the year from the date-time object
input_df$start_date <- start_dates

##process and add institute name
# input_df$institution <- sapply()
input_df$institution <- NA
for(i in 1:nrow(input_df)){
  ID <- input_df$APPLICATION_ID[i]
  inst <- unique(dat_Applications$INST_adapted[dat_Applications$APPLICATION_ID == ID])
  input_df$institution[i] <- inst
}

##store test input
input_test <- input_df[sample(1:nrow(input_df), 3),]

out_list <- list("input_df" = input_df, "input_df_test" = input_test)
return(out_list)
}

############ 2) Function to Save results and export  ########################
##############################################################################

## function to generate result tables from query
make_result_table <- function(q){
ids <- names(q)
counts <- sapply(q, function(x) x$count)
queries <- sapply(q, function(x) x$query)
result_table <- data.frame(ID = ids, count = counts, query = queries)
##add titles from public reports
titles <- sapply(result_table$ID,
                 FUN = function(ID) dat_PR$TITLE_EN[dat_PR$APPLICATION_ID == ID])
titles <- lapply(titles, function(x) x[1])
result_table$title <- as.character(titles)
##extract keywords from title
# result_table$keyw <- sapply(result_table$title, keyword_extraction)
result_table$counts_kw <- sapply(q, function(x) x$count_kw)
result_table$counts_raw <- sapply(q, function(x) x$count_raw)

##further process
result_table$counts_kw <- as.numeric(result_table$counts_kw)
##order by counts_kw
result_table <- result_table[order(result_table$counts_kw), ]
result_table$title <- as.character(result_table$title)

return(result_table)
}

############ 3) Function to Save queries  ###################################
##############################################################################

save_queries <- function(q_input){
  #get file name
  input_name <- deparse(substitute(q_input))

  # Generate a filename with the current date
  current_date <- format(Sys.Date(), "%Y%m%d")

  file_name <- paste0("queries/", current_date, input_name, ".RData", sep = "")

  # Save the list to the file with the appended date
  save(q_input, file = file_name)

}


############ 4) Function to Export non-sensitive information #################
##############################################################################

## function to export non-sensitive information
export_function <- function(results){
  ##get file name
  file_name <- deparse(substitute(results))
```

```
  #remove query, APPLICATOIN ID and title
  idx_query <- which(names(results) == "query")
  idx_title <- which(names(results) == "title")
  export <- results[, -c(idx_query, idx_title)]
  rownames(export) <- NULL
  # export$ID <- NULL

  #export csv
  write.csv(export, file = paste0(file_name, ".csv"), row.names = F)
}
```

```
##############################################################################
###### TL2_tracking_requests.R
##  Helper Script to TL_Algorithm.R with the automatic query functions.
##  Includes:
##  1) keyword_extraction()
##  2) make_query()
##############################################################################

##############################################################################
# 1) Keyword extraction
##############################################################################

##packages
require(udpipe)
require(textcat) #for checking if text is german

##initialize language models (do outside of function, otherwise slow)
tagger <- udpipe_download_model("english")
tagger_german <- udpipe_download_model("german")
tagger <- udpipe_load_model(tagger$file_model)
tagger_german <- udpipe_load_model(tagger_german$file_model)

#extract keywords (nouns and proper nouns) from titles
#from Universal Part-of-Speech tagging (UPOS),
keyword_extraction <- function(string){
  if(textcat(string) == "german"){ex_tagged <- udpipe_annotate(tagger_german, string)}
  else{ex_tagged <- udpipe_annotate(tagger, string)}
  ex_tagged_df <- as.data.frame(ex_tagged)
  keyw <- ex_tagged_df$lemma[ex_tagged_df$upos %in% c("NOUN", "PROPN", "ADJ")]
  return(keyw)
}


##############################################################################
#Search Pubmed with autor list
# 2) make_query() function definition
##############################################################################

###1) get author list, i.e. all involved persons (Form A attachment)
###   assigned to request number (RN)
###2) get number of animals used per RN -> Form AC
###3) (less important) get number of requested animals per RN -> Form A
## -> aggregate on RN in dictionary (e.g.)

require(easyPubMed)
##Searching and selecting publications for author list (AL)
## example inputs
make_query <- function(ID_authors_tab, affili = TRUE,
                       mesh = TRUE, institution = TRUE,
                       company = TRUE){
  ########
  ### Function queries pubmed for each req's ALt and more specifications.
  ### It downloads and stores abstracts for kw-matching
  ### The output is the query strings, titles, counts (after kw-matching)
  ########

  ##add values to "core" query if true, add nothing otherwise
  if(affili == TRUE){affiliation <- "Switzerland[ad]"}else{affiliation <- NULL}
  if(mesh == TRUE){animals <- "Animals[MeSH]"}else{animals <- NULL}
  if(company == TRUE){##Note from JF: censored for report}

  ##construct the string suffix for publication types
  pt_not1 <- " AND Journal Article[pt] NOT Review[pt] NOT Editorial[pt] NOT Letter[pt]"
  pt_not2 <- "NOT Comment[pt] NOT Case Reports[pt] NOT Clinical Study[pt] NOT Meta-Analysis[pt]"
  pt_not3 <- "NOT Randomized Controlled Trial[pt]"
  pt_not <- paste(pt_not1, pt_not2, pt_not3, comp)

  ## start to loop over all request numbers (RN)
  output <- list()
  for(i in 1:nrow(ID_authors_tab)){
    #get request number
    RN <- ID_authors_tab[i,1]
    #get authors list per RN and process
    authors_list <- ID_authors_tab[i,2]
    authors_list2 <- paste0(authors_list, "[au]")
    authors <- gsub(",", "[au] OR", authors_list2)

    ##get start date
    start_date <- ID_authors_tab$start_date[i]
    ##append with publication date (uncomment for current date)
    # current_date <- Sys.Date()
    # formatted_date <- format(current_date, "%Y/%m/%d")
    # NOW <- paste0(":", formatted_date, "[dp]")
```

```
    NOW <- ":2023/11/13[dp]"
    date <- paste0(start_date, sep = NOW)

    #get Institution name
    if(institution == TRUE){
      inst <- ID_authors_tab$institution[i]
      insti <- paste0(inst, sep = "[ad]")
    }else{insti <- NULL}

    ##convert to dictionary
    keys <- c("authors", "affiliation", "date", "mesh", "insti")
    values <- list(authors, affiliation, date, animals, insti)
    dict <- setNames(values, keys)

    ##generate query string
    my_query_core <- paste(unlist(dict), collapse = " AND ")
    my_query <- paste(my_query_core, pt_not)
    query_raw <- authors #only include authors to compare

    # #geputmedIDs from query, fetch data and write abstracts to txt
    my_entrez_id <- get_pubmed_ids(my_query,
                                   api_key = "56792befa7911c4fc0d4d56ce8b01e76fb09")
    entrez_raw <- get_pubmed_ids(query_raw)
    count <- my_entrez_id$Count #numbers of publications
    count_raw <- entrez_raw$Count
    if(count != "0"){ #if counts = zero, don't try to download data
      #download abstracts as xml to loop through
      my_abstracts_xml <- fetch_pubmed_data(my_entrez_id, format = "xml")
      titles <- custom_grep(my_abstracts_xml, "ArticleTitle", "char")

      ##check for keywords
      keyw_test <- unlist(ID_authors_tab$keyw[i])
      my_PM_list <- articles_to_list(my_abstracts_xml)

      #stroe pmids and abstracts with keywords
      pmids_with_kw <- c()
      abstracts_with_kw <- c()
      count_kw <- 0
      for(j in 1:length(my_PM_list)){#for all abstracts check if cont keyword
        PM_record_temp <- my_PM_list[[j]]
        abstract_temp <- custom_grep(PM_record_temp, tag = "AbstractText",
                                     format = "char")
        ## choose first entry due to bug with ArticleID
        PMID_temp <- custom_grep(PM_record_temp, tag = "PMID",
                                 format = "char")[1]
        abstract_temp_string <- toString(abstract_temp)
        absi <- c(abstract_temp)
        if(length(abstract_temp) == 0){keyw_test_results <- lapply(keyw_test_results,
                                                                   function(x) x <- FALSE)}else{
          keyw_test_results <- sapply(keyw_test, function(x) grepl(x, abstract_temp))
        }
        if(any(keyw_test_results == T)){#store PMIDs that contain at least one keyword
          pmids_with_kw <- c(pmids_with_kw, PMID_temp)
          abstracts_with_kw <- c(abstracts_with_kw, abstract_temp_string)
        }
      }
      #get count of keyword-matched abstracts
      count_kw <- length(pmids_with_kw)
      #download also all abstracts in .txt format
      my_abstracts_txt <- fetch_pubmed_data(my_entrez_id, format = "abstract")}else{
        titles <- NULL; my_abstracts_txt <- NULL; my_abstracts_xml <- NULL
        pmids_with_kw <- NULL; abstracts_with_kw = c(); count_kw <- 0}

    ##summarize results in a (double) list for all request number
    my_list <- list("query" = my_query_core, "query_total" = my_query,
                    "titles" = titles, "abstracts" = my_abstracts_txt,
                    "count" = count, "abstracts_xml" = my_abstracts_xml,
                    "pmids_with_kw" = pmids_with_kw, "count_kw" = count_kw,
                    "count_raw" = count_raw,
                    "abstracts_with_kw" = abstracts_with_kw)
    output[[RN]] <- my_list
  }
  print("query complete")
  return(output)
}
```

```
################################################################################
####### TL3_manual.screenig.R
##  Helper Script to TL_Algorithm.R with the manual screening functions.
##  Includes:
##  1) query_loading()
##  2) walk_through_strings()
################################################################################

################################################################################
# 1) Load Completed (saved) queries
################################################################################
query_loading <- function(){
  load("queries/20231121q.RData")
  assign("q_valid", q_input, envir = .GlobalEnv)

  load("queries/20231113q.RData")
  assign("q_loaded", qx, envir = .GlobalEnv)
```

```
  load("queries/20231124q_noAffili.RData")
  assign("q_noAffili_loaded", q_input, envir = .GlobalEnv)

    load("queries/20231124q_noInst.RData")
  assign("q_noInst_loaded", q_input, envir = .GlobalEnv)

  load("queries/20231124q_noMesh.RData")
  assign("q_noMesh_loaded", q_input, envir = .GlobalEnv)

  load("queries/20231124q_noCompany.RData")
  assign("q_noCompany_loaded", q_input, envir = .GlobalEnv)

  load("queries/20231124q_none.RData")
  assign("q_none_loaded", q_input, envir = .GlobalEnv)

}
# query_loading()


##############################################################################
# 2) # Function to walk through the list and collect user input
##############################################################################

walk_through_strings <- function(q, ID, rep = F) {
  if(rep == T){
    pmids <- non_matching_values[[ID]]
  }else{pmids <- q[[ID]]$pmids_with_kw}
  print(paste("Application ID:", ID))
  print(c("PMIDs", paste(pmids)))
  while(TRUE){
    user_input <- readline(prompt = "ID copied and information open on second screen (write yes)?")
    if(user_input == "yes")
      break
  }
  if(rep == T){string_list <- q[[ID]]$abstracts_with_kw[which(q[[ID]]$pmids_with_kw %in% pmids)]}else{
  string_list <- q[[ID]]$abstracts_with_kw}
  if(is.null(string_list)){return("No Abstracts for this ID, function is terminated")}
  labels <- data.frame(string = character(0), label = character(0), comments = character(0))

  for (string in string_list) {
    print(paste("Current string:", string))

    # Ask for user input and validate the input
    while (TRUE) {
      user_input_label <- readline(prompt = "Label as 'yes' or 'no': ")
      if (user_input_label %in% c("yes", "no")) {
        # Store the label and comment in the data frame
        user_input_comments <- readline(prompt = "Enter your comments: ")
        labels <- rbind(labels, data.frame(string = string,
                                           label = user_input_label,
                                           comments = user_input_comments))
        break
      } else {
        print("Invalid input. Please enter 'yes' or 'no'.")
      }
    }
  }
  print(paste("Manual Screen completed of:", ID))

  #replace abstract string with pmids
  names(labels)[which(names(labels) == "string")] <- "pmid"
  labels$pmid <- pmids

  #combine with ID ID (Application ID)
  labels <- cbind(ID = ID, labels)
  # Get the current date and time
  timestamp <- format(Sys.time(), "%Y%m%d_%H_%M")

  # Save the assessment data frame ame
  filename <- paste0("manuals/", timestamp, "_manual_screen", ".csv", sep = "")
  write.csv(labels, file = filename, row.names = FALSE)
  return(labels)
}



##### Transfer results after manual screening complete ########################
##############################################################################

##sum up all  manual counts to result_table for all requests
##that didn't result in zero matched hit
aggregate_man_results <- function(path_to_csv = "manuals/old", res_table){
csv_files <- list.files(path = path_to_csv,
                        pattern = "\\.csv$", full.names = TRUE)

# Create an empty list to store the data frames
all_data <- list()

# Loop through each CSV file and read it into a data frame
for (file in csv_files) {
  data <- read.csv(file)
  all_data[[file]] <- data
}


##transfer to result_table
```

```
res_table$man_count <- NA
# legacy_result_table$man_count <- NA #

# Loop through each data frame in 'all_data'
for (i in seq_along(all_data)) {
  df <- all_data[[i]]

  # Calculate the sum of "yes" entries in the "labels" column
  sum_of_yes <- sum(df$label == "yes", na.rm = TRUE)

  # Get the corresponding ID
  id <- unique(df$ID)

  # Create a new row with ID and sum_of_yes and append it to the result_df
  res_table$man_count[which(res_table$ID == id)] <- sum_of_yes
  }

#replace NA with zeros
res_table$man_count[is.na(res_table$man_count)] <- 0

##write csvs from result table after manual screening results addition
export_function(res_table)
return(res_table)
}

## also export comments files
export_justifications_file <- function(path_to_csv = "manuals/old"){
csv_files <- list.files(path = path_to_csv, pattern = "\\.csv$", full.names = TRUE)

merged_data <- data.frame()

for (file in csv_files) {
  # print(file)
  current_data <- read.csv(file, stringsAsFactors = FALSE)
  merged_data <- rbind(merged_data, current_data)
}

total_merged <- merged_data #legacy...
total_merged_yes <- total_merged[which(total_merged$label == "yes"),]
##Note from JF: censored manual exclusion of 2 pmids
write.csv(total_merged, "justifications_manual_screening.csv")
return(total_merged)
}


##### Get total used, requestsed and approved ########################
#############################################################################

make_endpoint_table <- function(res = man_res, just = total_merged){
#####get total published
used_animals_DS_sum <- aggregate(cbind(ANIMALS_USAGE_DS_ZERO,
                                       ANIMALS_USAGE_DS_ONE,
                                       ANIMALS_USAGE_DS_TWO,
                                       ANIMALS_USAGE_DS_THREE) ~ APPLICATION_ID,
                                 data = dat_PR, FUN = sum)
total_used_animals <- as.data.frame(cbind(used_animals_DS_sum$APPLICATION_ID,
                                          rowSums(used_animals_DS_sum[,2:ncol(used_animals_DS_sum)])))
names(total_used_animals) <- c("APPLICATION_ID", "used")

##approved and requestsed
approved_animals_sum <- aggregate(TOTAL_ANIMAL_APPROVED ~
                                    APPLICATION_ID, data = dat_Applications,
                                  FUN = sum)
requested_animals_sum <- aggregate(TOTAL_ANIMAL_REQUESTED ~ APPLICATION_ID,
                                   data = dat_Applications, FUN = sum)

#transfer to csv
animal_numbers_overview <- total_used_animals
animal_numbers_overview$approved <- sapply(animal_numbers_overview$APPLICATION_ID,
      FUN = function(ID) approved_animals_sum$TOTAL_ANIMAL_APPROVED[which(approved_animals_sum$APPLICATION_ID == ID)])
animal_numbers_overview$requested <- sapply(animal_numbers_overview$APPLICATION_ID,
      FUN = function(ID) requested_animals_sum$TOTAL_ANIMAL_REQUESTED[which(requested_animals_sum$APPLICATION_ID == ID)])
animal_numbers_overview$NATIONAL_NUMBER <- sapply(animal_numbers_overview$APPLICATION_ID,
      FUN = function(ID) unique(dat_PR$NATIONAL_NUMBER[which(dat_PR$APPLICATION_ID == ID)]))

animal_numbers_overview <- animal_numbers_overview[, c("NATIONAL_NUMBER", "APPLICATION_ID", "used", "approved", "requested")]

write.csv(animal_numbers_overview,
          "20231119_ANIMEX_aggregated_animal_numbers.csv")
animal_numbers_overview_2 <- read.csv("20231119_ANIMEX_aggregated_animal_numbers.csv",
                                      header = T)

##add pmids to endpoint table
aggregated_pmids <- aggregate(pmid ~ ID, just, FUN = function(x) paste(x, collapse = ", "))
pmid <- merge(man_res, aggregated_pmids, by = "ID", all.x = TRUE)
names(pmid)[which(names(pmid) == "ID")] <- "APPLICATION_ID"

endpoint_result_table <- merge(pmid, animal_numbers_overview_2)
###xxx where is man_count?
endpoint_result_table <- endpoint_result_table[-which(endpoint_result_table$man_count == 0),]
export_function(endpoint_result_table)
return(endpoint_result_table)
}
```

Reference

Auspurg, K. and Hinz, T. (2011). What fuels publication bias? Theoretical and empirical analyses of risk factors using the Caliper test. *Jahrbücher für Nationalökonomie und Statistik*, 231 636–660. 2

Bahnam, P., Hanzel, J., Ma, C., Zou, L., Narula, N., Singh, S., Kahan, B., and Jairath, V. (2022). Most Placebo-Controlled Trials in Inflammatory Bowel Disease were Underpowered Because of Overestimated Drug Efficacy Rates: Results from a Systematic Review of Induction Studies. *Journal of Crohn's and Colitis*, 17 404–417. 2

Baker, Monya (2019). Animal registries aim to reduce bias. *Nature*, 573 297–298. 3

Balduzzi, S., Rücker, G., and Schwarzer, G. (2019). How to perform a meta-analysis with R: A practical tutorial. *Evidence-Based Mental Health*, 153–160. 19

Biondi-Zoccai, G., Lotrionte, M., Landoni, G., and Modena, M. (2011). The rough guide to systematic reviews and meta-analyses. *HSR proceedings in Intensive Care and Cardiovascular Anesthesia*, 3 161. 1

Coats, T., Roberts, I. G., and Shakur, H. (2004). Antifibrinolytic drugs for acute traumatic injury. *Cochrane Database of Systematic Reviews*. 2

Cohen, J. (1992). Statistical power analysis. *Current Directions in Psychological Science*, 1 98–101. 7

Copas, J. and Shi, J. Q. (2000). Meta-analysis, funnel plots and sensitivity analysis. *Biostatistics*, 1 247–262. 16

Copas, J. and Shi, J. Q. (2001). A sensitivity analysis for publication bias in systematic reviews. *Statistical Methods in Medical Research*, 10 251–265. 16, 17

Copas, J. B. and Malley, P. F. (2008). A robust P-value for treatment effect in meta-analysis with publication bias. *Statistics in Medicine*, 27 4267–4278. 16

Corporation, M. and Weston, S. (2022). *doParallel: Foreach Parallel Adaptor for the 'parallel' Package*. R package version 1.0.17, https://CRAN.R-project.org/package=doParallel. 19

Dahl, D. B., Scott, D., Roosen, C., Magnusson, A., and Swinton, J. (2019). *xtable: Export Tables to LaTeX or HTML*. R package version 1.8-4, https://CRAN.R-project.org/package=xtable. 25

Dickersin, K. (1990). The existence of publication bias and risk factors for its occurrence. *Journal of the American Medical Association*, 263 1385–1389. 2

Duval, S. and Tweedie, R. (2000). A nonparametric "trim and fill" method of accounting for publication bias in meta-analysis. *Journal of the American Statistical Association*, 95 89–98. 15, 41

ElHarouni, D., Berker, Y., Peterziel, H., Gopisetty, A., Turunen, L., Kreth, S., Stainczyk, S. A., Oehme, I., Pietiäinen, V., and Jäger, N. (2022). iTReX: Interactive exploration of mono-and combination therapy dose response profiling data. *Pharmacological Research*, 175 105996. 42

Elia, N., von Elm, E., Chatagner, A., Pöpping, D. M., and Tramèr, M. R. (2016). How do authors of systematic reviews deal with research malpractice and misconduct in original studies? A cross-sectional analysis of systematic reviews and survey of their authors. *BMJ Open*, 6 e010442. 7

Elliott, G., Kudrin, N., and Wüthrich, K. (2022). Detecting p-hacking. *Econometrica*, 90 887–906. 42

Elliott, K. C. and Resnik, D. B. (2019). Making open science work for science and society. *Environmental Health Perspectives*, 127 075002. 45

Fantini, D. (2019). *easyPubMed: Search and Retrieve Scientific Publication Records from PubMed*. R package version 2.13, https://CRAN.R-project.org/package=easyPubMed. 25

Ferguson, C. J. and Heene, M. (2012). A vast graveyard of undead theories: Publication bias and psychological science's aversion to the null. *Perspectives on Psychological Science*, 7 555–561. 2

Fox, J. and Weisberg, S. (2019). *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, third edition. 25

Frick, R. W. (1995). Accepting the null hypothesis. *Memory & Cognition*, 23 132–138. 2

Furrer, R. (2018). STA121: Statistical Modeling - An Excursion through 14 Topics. 28–29. 8

Garattini, S. and Grignaschi, G. (2017). Animal testing is still the best way to find new treatments for patients. *European Journal of Internal Medicine*, 39 32–35. 1

Grüninger, S. (2019). Weights of statistical evidence: Detection and correction of publication bias. Master's thesis, École Polytechnique Fédérale de Lausanne (EPFL). 13

Harrer, Mathias and Cuijpers, Pim and Furukawa Toshi A and Ebert, David D (2021). *Doing Meta-Analysis With R: A Hands-On Guide*. Chapman and Hall-CRC/Press, Boca Raton, FL and London, 1st edition. 1

Held, Leonhard and Sabanés-Bové, Daniel Sabanés (2020). *Likelihood and Bayesian Inference*. Springer. 9

Higgins, J. P. and Green, S. (2023). Cochrane handbook for systematic reviews of interventions. 2

Higgins, J. P., Thomas, J., Chandler, J., Cumpston, M., Li, T., Page, M. J., and Welch, V. A., editors (2019). *Cochrane Handbook for Systematic Reviews of Interventions*. Wiley. 1

Hothorn, T., Möst, L., and Bühlmann, P. (2018). Most likely transformations. *Scandinavian Journal of Statistics*, 45 110–134. 42

Jahan, N., Naveed, S., Zeshan, M., and Tahir, M. A. (2016). How to Conduct a Systematic Review: A Narrative Literature Review. *The Cureus Journal of Medical Science*. 1

Kassambara, A. (2023). *ggpubr: 'ggplot2' Based Publication Ready Plots*. R package version 0.6.0, https://CRAN.R-project.org/package=ggpubr. 19

Klein, J. (2022). Improving the reproducibility of findings by updating research methodology. *Quality & Quantity*, 56 1597–1609. 3

Kwasnicka, D., Ten Hoor, G. A., van Dongen, A., Gruszczyńska, E., Hagger, M. S., Hamilton, K., Hankonen, N., Heino, M. T. J., Kotzur, M., and Noone, C. (2021). Promoting scientific integrity through open science in health psychology: Results of the Synergy Expert Meeting of the European health psychology society. *Health Psychology Review*, 15 333–349. 45

Laajala, T. D., Jumppanen, M., Huhtaniemi, R., Fey, V., Kaur, A., Knuuttila, M., Aho, E., Oksala, R., Westermarck, J., Mäkelä, S., Poutanen, M., and Aittokallio, T. (2016). Optimized design and analysis of preclinical intervention studies in vivo. *Scientific Reports*, 6. 1

Landis, J. T., An, H., Bailey, A. G., Dittmer, D. P., and Marron, J. S. (2021). *dr4pl: Dose Response Data Analysis using the 4 Parameter Logistic (4pl) Model*. R package version 2.0.0, https://CRAN.R-project.org/package=dr4pl. 42

Laurier, D., Billarand, Y., Klokov, D., and Leuraud, K. (2023). The scientific basis for the use of the linear no-threshold (LNT) model at low doses and dose rates in radiological protection. *Journal of Radiological Protection*. 7

Levin, N., Leonelli, S., Weckowska, D., Castle, D., and Dupré, J. (2016). How do scientists define openness? Exploring the relationship between open science policies and research practice. *Bulletin of Science, Technology & Society*, 36 128–141. 45

Light, R. J. and Pillemer, D. B. (1984). *Summing up: The science of reviewing research*. Harvard University Press. 11

Liu, W., Islamaj Doğan, R., Kim, S., Comeau, D. C., Kim, W., Yeganova, L., Lu, Z., and Wilbur, W. J. (2014). Author name disambiguation for PubMed. *Journal of the Association for Information Science and Technology*, 65 765–781. 44

Maligranda, L. (1998). Why Hölder's inequality should be called Rogers' inequality. *Mathematical Inequalities & Applications*, 1 69–83. 7

Mcbratney, A. (2004). The Cambridge Dictionary of Statistics. *Geoderma*, 1 155–156. 8

Microsoft and Weston, S. (2022). *foreach: Provides Foreach Looping Construct*. R package version 1.5.2, https://CRAN.R-project.org/package=foreach. 19

Moja, L., Lucenteforte, E., Kwag, K. H., Bertele, V., Campomori, A., Chakravarthy, U., D'Amico, R., Dickersin, K., Kodjikian, L., and Lindsley, K. (2014). Systemic safety of Bevacizumab versus Ranibizumab for neovascular age-related macular degeneration. *Cochrane Database of Systematic Reviews*. 2

Moreno, S. G., Sutton, A. J., Ades, A., Stanley, T. D., Abrams, K. R., Peters, J. L., and Cooper, N. J. (2009). Assessment of regression-based methods to adjust for publication bias through a comprehensive simulation study. *BMC Medical Research Methodology*, 9 1–17. 41, 42

Mueller, K. F., Meerpohl, J. J., Briel, M., Antes, G., von Elm, E., Lang, B., Motschall, E., Schwarzer, G., and Bassler, D. (2016). Methods for detecting, quantifying, and adjusting for dissemination bias in meta-analysis are described. *Journal of Clinical Epidemiology*, 80 25–33. 3

Nair, A. S. (2019). Publication bias-importance of studies with negative results! *Indian Journal of Anaesthesia*, 63 505. 2

Neyman, J. and Pearson, E. S. (1933). On the Problem of the Most Efficient Tests of Statistical Hypotheses. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 231 289–337. 2

OpenAI (2023). ChatGPT. https://www.openai.com/. Accessed: November 28, 2023. 25

Page, M. J., McKenzie, J. E., and Higgins, J. P. (2018). Tools for assessing risk of reporting biases in studies and syntheses of studies: a systematic review. *BMJ Open*, 8 e019703. 2

Pennstate (2023a). Change-of-Variable Technique. https://online.stat.psu.edu/stat414/lesson/22/22.2. [Accessed: 18-09-2023]. 10

Pennstate (2023b). Sampling Distribution of Sample Variance. https://online.stat.psu.edu/stat414/lesson/26/26.3. [Accessed: September 18, 2023]. 9

Percie du Sert, N., Hurst, V., Ahluwalia, A., Alam, S., Avey, M. T., Baker, M., Browne, W. J., Clark, A., Cuthill, I. C., Dirnagl, U., et al. (2020). The ARRIVE guidelines 2.0: Updated guidelines for reporting animal research. *Journal of Cerebral Blood Flow & Metabolism*, 40 1769–1777. 45

Poole, M. A. and O'Farrell, P. N. (1971). The assumptions of the linear regression model. *Transactions of the Institute of British Geographers*, 145–158. 8

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/. 19, 25

Roberts, D., Brown, J., Medley, N., and Dalziel, S. R. (2017). Antenatal corticosteroids for accelerating fetal lung maturation for women at risk of preterm birth. *Cochrane Database of Systematic Reviews*. 2

Rosenthal, R. (1979). The file drawer problem and tolerance for null results. *Psychological Bulletin*, 86 638. 28

Schneck, A. (2017). Examining publication bias—a simulation-based evaluation of statistical tests on publication bias. *PeerJ*, 5 e4115. 7, 41, 42

Schwarzer, G., Carpenter, J., and Rücker, G. (2010). Empirical evaluation suggests Copas selection model preferable to trim-and-fill method for selection bias in meta-analysis. *Journal of Clinical Epidemiology*, 63 282–288. 17

Sena, E. S., van der Worp, H. B., Bath, P. M. W., Howells, D. W., and Macleod, M. R. (2010). Publication Bias in Reports of Animal Stroke Studies Leads to Major Overstatement of Efficacy. *PLoS Biology*, 8 e1000344. 1, 2

Simmons, J., D Nelson, L., and Simonsohn, U. (2021). Pre-registration: Why and how. *Journal of Consumer Psychology*, 31 151–162. 3

Snoop Dogg (2019). I wanna thank me. azlyrics.com/lyrics/snoopdogg/iwannathankme.html. [Accessed: December 5, 2023]. v

Sterne, J. A. and Harbord, R. M. (2004). Funnel plots in meta-analysis. *The Stata Journal*, 4 127–141. 12

ter Horst, H., Brazda, N., Schira-Heinen, J., Krebbers, J., Müller, H.-W., and Cimiano, P. (2023). Automatic knowledge graph population with model-complete text comprehension for pre-clinical outcomes in the field of spinal cord injury. *Artificial Intelligence in Medicine*, 137 102491. 1

ter Riet, G., Korevaar, D. A., Leenaars, M., Sterk, P. J., Van Noorden, C. J., Bouter, L. M., Lutter, R., Elferink, R. P. O., and Hooft, L. (2012). Publication bias in laboratory animal research: A survey on magnitude, drivers, consequences and potential solutions. *Public Library of Science*. 4, 43

van der Naald, M., Chamuleau, S. A., Menon, J. M., de Leeuw, W., de Haan, J., Duncker, D. J., and Wever, K. E. (2022). Preregistration of animal research protocols: Development and 3-year overview of preclinicaltrials.eu. *BMJ Open Science*, 6. 45

van der Naald, M., Wenker, S., Doevendans, P. A., Wever, K. E., and Chamuleau, S. A. (2020). Publication rate in preclinical research: a plea for preregistration. *BMJ Open Science*, 4. 4, 41

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. 25

Viechtbauer, W. (2010). Conducting meta-analyses in R with the metafor package. *Journal of Statistical Software*, 36 1–48. 19

Volund, A. (1978). Application of the four-parameter logistic model to bioassay: comparison with slope ratio and parallel line models. *Biometrics*, 357–365. 42

Wasserstein, R. L. and Lazar, N. A. (2016). The ASA statement on p-values: context, process, and purpose. *The American Statistician*, 70 129–133. 2

Wickham, H., Averick, M., Bryan, J., Chang, W., McGowan, L. D., Fran**c**cois, R., Grolemund, G., Hayes, A., Henry, L., Hester, J., Kuhn, M., Pedersen, T. L., Miller, E., Bache, S. M., Müller, K., Ooms, J., Robinson, D., Seidel, D. P., Spinu, V., Takahashi, K., Vaughan, D., Wilke, C., Woo, K., and Yutani, H. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, 4 1686. 19

Wieschowski, S., Biernot, S., Deutsch, S., Glage, S., Bleich, A., Tolba, R., and Strech, D. (2019). Publication rates in animal research. Extent and characteristics of published and non-published animal studies followed up at two German university medical centres. *Public Library of Science San Francisco*, 14 e0223758. 4, 43

Wijffels, J. (2023). *udpipe: Tokenization, Parts of Speech Tagging, Lemmatization and Dependency Parsing with the 'UDPipe' 'NLP' Toolkit*. R package version 0.8.11, https://CRAN.R-project.org/package=udpipe. 25

Zhu, Q. and Carriere, K. (2018). Detecting and correcting for publication bias in meta-analysis–A truncated normal distribution approach. *Statistical Methods in Medical Research*, 27 2722–2741. 17, 18