Statistical methods for optimal design and information preservation in pharmacokinetics and data squashing with missing values

By

Ryan T. Jarrett

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Biostatistics

August 31, 2021

Nashville, Tennessee

Approved:

Bryan Shepherd, Ph.D.

Matthew S. Shotwell, Ph.D.

Jonathan Schildcrout, Ph.D.

Leena Choi, Ph.D.

André Diedrich, Ph.D.

ACKNOWLEDGEMENTS

There are many people that I would like to thank for the support, advice, and mentorship that they provided me with during my time at Vanderbilt. First, my dissertation advisor Dr. Matt Shotwell has been has been a tremendous source of advice and mentorship. He has encouraged me to pursue my own interests, while also introducing me to new statistical problems, new methods for computation and estimation, and novel perspectives on scientific research. Much of my development as a statistician is thanks to his mentorship.

Second, I would like to thank all those on my research assistantship and, in particular, my RA advisor, Dr. Rameela Raman. I began my RA on the first day that I arrived at Vanderbilt. Since that time, Rameela has been a superb mentor. She has always been an excellent source of statistical and professional advice, while also seeking opportunities for me to publish research papers, attend conferences, and receive awards. Thank you, Rameela, for your unwavering support.

My dissertation committee has also offered valuable advice throughout the progression of this thesis. Thank you Drs. Bryan Shepherd, Jonathan Schildcrout, Leena Choi, and André Diedrich for your insightful contributions and feedback. I would also like to thank the faculty, staff, and students of the Vanderbilt Department of Biostatistics. It has been a pleasure and an honor to be a member of our department. This is, in large part, due to the welcoming and stimulating environment that you have created.

Finally, I would like to thank my family and friends. Many of you have helped me directly by proofreading my papers or offering feedback on presentations. All of you have helped me become the person that I am today. This would not have been possible without you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BIS	Bispectral Index
BIS50	Closed-loop induction strategy targeting BIS=50
CLC	Closed-loop control
DoH	Depth of Hypnosis
E-M	Expectation-maximization procedure
FIML	Full-information maximum likelihood
MAP	Maximum a posteriori
MAR	Missing at random
MCAR	Missing completely at random
MI	Multiple Imputation
MNAR	Missing not at random
OpenLoop	Open-loop induction strategy with weight- and age-adjusted doses
PD	Pharmacodynamic
PK	Pharmacokinetic
RF	Reference function
RFe	RF with exponential functional form
RFs	RF with sigmoidal functional form
SET	Stable entry time criterion
TCI	Target-controlled infusion
TD	Total dose criterion
TZ	Target zone (BIS = [40,60])
WOS	Weighted overshoot criterion

Chapter 1

Introduction

This dissertation aims to extend statistical methodological research in two distinct directions. The first concerns the automation of intravenous drug delivery during anesthesia through "closed-loop" systems. Systems are termed closed-loop when a response signal (i.e., a measurement of a patient's depth of hypnosis) is fed directly back to the mechanism that controls the input delivery (i.e., the drug infusion rate), thereby "closing" the feedback loop. As of the writing of this dissertation, closed-loop systems are still largely experimental in anesthesiology; however, automation has been a long-standing goal within anesthesiology.(Absalom et al., 2011) The primary interest in automation is in its potential to more precisely respond to the substantial inter- and intra-patient variability in pharmacokinetics (PK, i.e., how a drug is processed by a body) and pharamacodynamics (PD, i.e., how a person responds to a specific amount of drug).(Kuck and Johnson, 2017) This variability confounds the ability to reliably sedate patients at the desired level: if the effective dose is too small, a patient may experience interoperative awareness; if too large, a patient may experience undesired cardiovascular events or delirium. During the induction of anesthesia, the goal is to rapidly and reliably transition a patient from an awake state to a stable state of anesthesia. At the time of the initial induction dose, however, one cannot know the precise dosage required due to PK-PD variability.

Chapter 2 introduces a framework for optimizing closed-loop control of anesthetic delivery within a population during the induction of anesthesia according to a user-specified criterion. This criterion could reflect any number of quantities of clinical interest. In Chapter 2, we consider minimizing a combination of over- and under-shoot of the target depth of hypnosis (DOH), time to stably reach the target DOH, and the amount of drug (i.e. propofol) delivered. The problem of unobserved inter-patient variability is considered from an optimal design of experiments perspective. When measurements of patients' DOH are continuously sampled at regular intervals, as they are within this context, the primary experimental conditions that can be modified to elicit a desired response are the infusion rates themselves. To this end, we propose the specification of a "reference function" to be used by the closed-loop controller. This reference function determines the value of the response variable that is targeted by the controller as a varying function over time and, in doing so, effectively implements a protocol for determining infusion rates. By optimizing the parameters of the reference function according to a population-level criterion, we can identify a protocol that is optimal for the criterion within the population of patients.

Chapter 3 continues in the same direction by introducing an open-source software package in the R programming language, `tci`, that flexibly implements open-loop target-controlled infusion (TCI) algorithms, as well as closed-loop control systems, for compartmental PK and PK-PD models. Briefly, TCI systems allow the user (i.e., the anesthesiologist) to target specific drug concentrations for a patient, rather than specifying infusion rates directly. This is discussed in more depth in Chapters 2 and 3. The TCI system then calculates the infusion rates required to achieve and maintain the concentration based on a population PK model that is adjusted for relevant patient covariates, such as weight, sex, and age. Though commonly termed "open-loop" systems, in practice a clinician will adjust the target concentration based on patient responses, thereby "closing" the

feedback loop, albeit indirectly.

TCI systems are much more established in anesthesiology than closed-loop control though, interestingly, are still unapproved for use in the United States by the FDA.(Dryden, 2016) The `tci` software package aims to fill a distinct void: there currently are no packages in R, or any other commonly-used statistical programming languages, that will implement TCI. Further, the only software that will implement TCI is either proprietary or written in low-level programming languages such that it is not readily accessible to, or modifiable by researchers interested in the subject. Consequently, they are not designed for research into or experimentation with TCI algorithms. Chapter 3 provides a more-complete description of the software currently available. `tci`, by contrast, is freely-avaliable, written in a commonly-used statistical programming language, and is easily customizable by the user.

Chapter 4 of this dissertation proceeds in a distinctly different direction. Where Chapters 2 and 3 aim to contribute to the rapidly developing forefront of research in anesthesia and applied control systems, Chapter 4 aims to extend a promising but largely unknown technique for statistical data compression termed "data squashing" or "squashing." First introduced by DuMouchel et al. (1999), the goal of squashing is to summarize a large data set with a much smaller version that contains synthetic "pseudo-data" on the same variables and a corresponding set of frequency weights, such that each pseudo-data point typically represents many points in the original data set. The squashed data set is constructed in such a way as to preserve the statistical information with respect to an arbitrary likelihood function applied to the original data set. Consequently, a model fit to the squashed data set will closely replicate the results that would have been observed had it been fit to the original data set. Since the squashed data set is substantially smaller, however, analyses that were computationally demanding or prohibitive on the original data set may be feasible with the squashed data set. The fact that squashing replaces real data with pseudo-data points provides an ancillary benefit: patient privacy and proprietary claims associated with the original data are protected. This facilitates direct sharing of data and analyses among researchers where it would otherwise not be possible.

Despite the highly promising results of DuMouchel et al. (1999), surprisingly little effort to extend squashing has been made in the intervening years. In Chapter 4 we describe these prior efforts and suggest reasons why, we believe, squashing has not been more widely adopted. One reason for this is that the original squashing method did not provide any mechanism for handling missing data. To address this potential barrier to adoption of data squashing, we propose two different methods to handling missing data within data squashing, which we term "propagation squashing" or "p-squashing", and "expectation squashing" or "e-squashing." In the first, missingness from the original data set is propagated on to the p-squashed data set while the information required for maximum likelihood-based missing data techniques is preserved in the p-squashed data set. In e-squashing the squashed data set is constructed to preserve the expectation of an arbitrary log-likelihood. This approach is motivated by the E-M algorithm, which is commonly used as an estimation procedure for models with missing data. E-squashing preserves the likelihood that results from a single E-step of the E-M algorithm. In doing so, it relies on additional approximations, but also results in a fully-observed data set. We conduct simulations to evaluate the performance of each squashing method and apply them to a real data set consisting of approximately 2.2 million workers' compensation claims in New York state from January 2000 to June 2021.

Chapter 2

Optimal BIS reference functions for closed-loop induction of anesthesia with propofol

2.1 Introduction

2.1.1 Background

During the intravenous induction of anesthesia, the hypnotic agent propofol is frequently administered to a patient to induce a loss of consciousness and amnesia. The advantages of propofol are its rapid onset, short duration of action, and low incidence of postoperative nausea and vomiting.(Sahinovic et al., 2018) For many patients, a rapid administration of propofol may be safe, reduce patient pain and discomfort on injection, and assist with patient intubation upon cessation of breathing.(Bibian et al., 2006; Hajat et al., 2017) Consequently, it is common in clinical practice to deliver an initial bolus or rapid infusion to bring the patient to a concentration that elicits the desired depth of hypnosis, followed by a slower infusion to maintain the effect.(Morton, 2009) Despite a favorable pharmacokinetic (PK) and pharmacodynamic (PD) profile, however, propofol is not without risks to patients. Higher doses of propofol have been associated with postoperative nausea and delirium, slower recoveries from anesthesia, and dose-dependent intraoperative hypotension. In some patients, such as the elderly or physically compromised, this may increase the risk of acute kidney and myocardial injury.(Bibian et al., 2006; Marik, 2005; Phillips et al., 2015; Wesselink et al., 2018; Sahinovic et al., 2018) For these reasons, it may be important in some circumstances to use the lowest effective dose.

The ability to manage these benefits and risks is confounded by a high degree of interpatient PK-PD variability that makes it difficult to anticipate the required effective dose for a particular patient.(Mandel and Sarraf, 2012) Target-controlled infusion (TCI) systems control the administration of medication by using measured patient covariates (e.g., age, weight, sex) and an underlying PK or PK-PD model to estimate the infusion rate required to reach and maintain a target drug concentration or clinical effect. A clinician may then manually adjust the target in response to patient feedback. Since patient responses are not directly incorporated into the calculation of infusion rates, however, TCI systems are termed "open-loop."(Dumont and Ansermino, 2013; Struys et al., 2016)

Closed-loop controllers (CLC), by contrast, collect patient data in real time and adjust infusion rates in order to maintain a specified reference value. The most commonly used reference value indicating a patient's depth of hypnosis is the Bispectral Index (BIS), an electroencephalography-derived measurement with values between zero and 100. Lower values indicate greater degrees of sedation and values between 40 and 60 are typically considered sufficient for general anesthesia.(Brogi et al., 2017) By adapting in real time to patient responses, CLC provide a mechanism for precisely determining patient depth of hypnosis even in the presence of substantial inter- and intra-patient PK-PD variability. More precise control of anesthesia will enable greater predictability in patient responses. This, in turn, may allow for more optimal care and increase patient safety during operations involving anesthesia.(Dumont and Ansermino, 2013)

A number of closed-loop controllers have been proposed for the induction of anesthesia that function in different ways. For example, proportional-integral-derivative (PID) controllers (West et al., 2013; Padula et al., 2017) adjust infusion rates according to the proportion, integral, and

derivative of the difference between the controlled variable and the reference value. Parameters may be tuned such that the PID controller achieves an acceptable level of performance within a target population and is therefore "robust" to uncertainty in population PK-PD. Bayesian controllers, by contrast, update a patient-specific model as measurements are gathered and deliver an infusion designed to reach the reference value as predicted by the model. These may also be tuned to achieve robust performance within a population, as in De Smet et al. (2007). They also have the benefit of incorporating patient covariates by using a TCI system. Work by Schiavo et al. (2021) aims to identify an optimal bolus dose for patients using PID control to emulate the model-based dose traditionally calculated by TCI systems.

The common element of these controllers is the input of a reference signal, about which the controller aims to maintain the patient's response. Most commonly, the reference signal is held constant at BIS=50 and the controller aims to titrate infusion rates so as to achieve and maintain this value.(West et al., 2013; Nascu et al., 2015; Padula et al., 2017; Brogi et al., 2017) There are some exceptions to this. Padula et al. (2016) and Schiavo et al. (2021) define time-varying reference signals for proportional-integral and PID controllers, respectively, achieved by applying an external "feed-forward" signal that directs infusion rates without dependency upon patient feedback.

2.1.2 Contributions in this chapter

In this chapter, we propose a framework for replacing the standard fixed reference value with a time-varying reference function (RF). The goal of this framework is to improve the ability of closed-loop controllers to achieve clinical outcomes in the presence of substantial unobserved interpatient PK-PD variability. Motivated by principles of optimal design of experiments, the RF is "designed" according to a user-specified criterion that incorporates variability explained by patient covariates and unexplained PK-PD variability within the patient population. For example, one criterion considered in this chapter is the average time required for a patient to stably achieve a BIS value between 40 and 60. The full set of RF are detailed in Section 2.3.

The values of the RF are used as inputs to a controller that will translate the target values into corresponding infusion rates. The process by which this happens will vary depending on the controller. In this chapter, we implement a Bayesian controller that estimates a patient-specific PK-PD model as data are collected. At each dosing time, the value of the RF is translated into a target effect-site concentration using the patient-specific PD model. The target concentration is then converted into an infusion rate by applying a TCI algorithm at the patient-specific PK model. Patient PK-PD model parameters are updated at regular intervals such that infusion rates become more accurate over time. This approach is compatible with any controller and can be complementary to existing efforts to make controllers robust in target populations (e.g., robust PID). Further, because the RF is optimized offline, it can be put in place prior to the administration of any medication.

Each RF defines a dosing protocol that will be applied to a population. To evaluate the performance of a specific RF, we simulate patient responses following the protocol and calculate quantities of interest, such as how quickly a patient entered the target zone of BIS = [40,60], or how much propofol was administered. For our simulations, we create a population of synthetic patients based on the Eleveld population PK-PD model.(Eleveld et al., 2018) Using the code and data provided by Eleveld et al. (2018), we identify the maximum *a posteriori* (MAP) PK-PD parameter estimates for the $N = 122$ patients whose data were used to develop both PK and PD components. These MAP

estimates represent the posterior mode of the PK-PD parameter distribution for each person and, therefore, represent each person's most-likely parameters estimates based on the structure of the Eleveld model and the persons' individual data. For our simulations, the MAP estimates are used to define a set of "true" parameters that dictate a patient's BIS response but are unknown to the CLC. The CLC operates based on only the information that would be known in a real induction: it begins with a model based on observable patient covariates and "learns" a more accurate patient-specific model as BIS observations are made available.

Inductions are simulated for a random subset of 50 patients in order to identify optimal RF parameters. The remaining 72 patients are used to evaluate the performance of each optimized RF. Responses for these 72 patients are also simulated using a strategy in which the RF is fixed at BIS=50 (BIS50). Finally, we also simulate results for each patient under an open-loop strategy, in which patient receive doses adjusted to their weight and age in accordance with propofol guidelines (OpenLoop).

The remainder of the chapter is structured as follows: Section 2.2 describes the PK-PD model, the closed-loop control mechanism, and the construction of the prior distribution on PK-PD parameters that is used by the CLC. Section 2.3 describes the specification of a RF and an optimality criterion and presents the set of optimized RF based on the 50 patients in the training set. Section 2.4 discusses the performance of the optimized RF on the 72 test-set patients and compares to BIS50 and OpenLoop strategies. Section 2.5 concludes and discusses strengths and limitations of the optimal reference function methodology.

2.2 Preliminaries

2.2.1 Pharmacokinetic model

Patient PK-PD are simulated using the Eleveld population model. Though full details regarding the final population model are described in Eleveld et al. (2018) and development of the PK component in Eleveld et al. (2014), we will briefly describe the relevant portions here. The Eleveld model is a nonlinear mixed-effects model in which the PK of propofol are described by a three-compartment model. Propofol is infused into a central compartment before circulating between two peripheral compartments or transferred to an effect-site compartment. Heuristically, the central compartment represents the plasma, the two peripheral compartments represent tissues that equilibrate rapidly (e.g., muscle) and slowly (e.g., fatty tissues).(Al-Rifai and Mulvey, 2016) Propofol is eliminated from the central compartment or transferred to an effect-site compartment (i.e., the brain) that links to a PD model. Figure 2.1 provides a diagram of the three-compartment model. The parameters of the three-compartment model describe the volume of each compartment and the rates at which propofol is transferred between compartments and eliminated from the central compartment.

Each of the parameters of the three-compartment model are, in turn, modeled as functions of a fixed effect term describing the average population value that is scaled by a function of observed patient covariates (e.g., weight, age) and a random effect term describing variability between patients conditional on covariate values. Consequently, each patient has two sets of PK parameter values: 1) "predicted" PK values calculated by the fixed effects at the patient's covariates, and 2) "true" parameter values that incorporate both the patient's covariates and the patient's random variation from the typical value. The former can be calculated for a patient before administration of any infusions while the latter are unobservable and cannot be known, except in a simulated setting. In

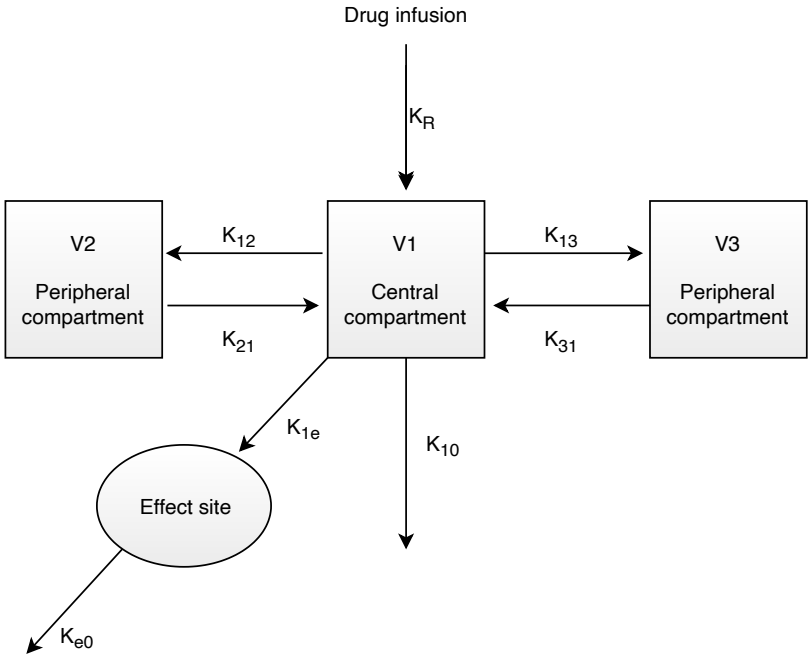

Figure 2.1: Diagram of a 3-compartment pharmacokinetic model with an effect-site linking to the pharmacodynamic model. Drug is administered to the central compartment, from which it circulates between two additional compartments and is eliminated according to rate constants $k_{12}, k_{21}, k_{13}, k_{31}$ and k_{10}, respectively. Absorption from the central compartment to the effect-site compartment, described by k_{1e}, k_{e0}, allows for a time delay between drug administration and pharmacodynamic effect.

our simulations, the empirical Bayes estimates are used as the patients' "true" parameters. The MAP estimates plausibly describe patient PK-PD and incorporate variability both explained and unexplained by patient covariates. Using these parameters, patient PK are simulated according to the following set of differential equations.

$$A = \begin{pmatrix} -(k_{10}+k_{12}+k_{13}) & \frac{V_2}{V_1}k_{21} & \frac{V_3}{V_1}k_{31} & 0 \\ \frac{V_1}{V_2}k_{12} & -k_{21} & 0 & 0 \\ \frac{V_1}{V_3}k_{13} & 0 & -k_{31} & 0 \\ \frac{V_1}{V_e}k_{1e} & 0 & 0 & -k_{e0} \end{pmatrix} \tag{2.1}$$

$$B = [1,0,0,0]^T$$

$$\frac{dC(t)}{dt} = A \cdot C(t) + B \cdot u(t)$$

These equations describe the change in compartmental concentrations over time, $\frac{dC(t)}{dt}$, as a function of concentration, $C(t)$, and infusion rates, $u(t)$. The parameters $[V_1, V_2, V_3]$ represent the volumes within each compartment and the parameters $k_{ij},\ i,j \in \{1,2,3,e\}$ are rate constants describing the transfer of propofol from compartment i to compartment j. To ensure that the fourth effect-site compartment does not disrupt the dynamics of the three-compartment system, the volume of the effect-site is arbitrarily set at $V_e = V_1/10,000$. Additionally, the rate of absorption in the effect site compartment is set equal to the elimination rate, such that $k_{1e} = k_{e0}/10,000$.(Shafer and Gregg 1992)

2.2.2 Pharmacodynamic model

As with the PK model, each patient has a set of true PD model parameters that are used to simulate the patient's BIS response to the effect-site concentration. Specifically, a sigmoidal Emax model is used.

$$\begin{aligned} \text{BIS}(t) &= \text{BIS}_{\text{baseline}} \cdot \left(1 - \frac{C_{e50}^{\gamma}}{C_{e50}^{\gamma} + C_e^{\gamma}}\right) + \epsilon, \quad \epsilon \sim N(0, \sigma^2) \\ \text{BIS}_{\text{delay}}(S) &= 15 + \exp(0.0517 \cdot AGE) \end{aligned} \tag{2.2}$$

Only the concentration at 50% effect, C_{e50}, and the residual error variance, σ^2, include random effects in the Eleveld model. Consequently, the predicted and true values are identical for $\text{BIS}_{\text{baseline}}$ and γ. The PD model also incorporates a time-lag between the underlying BIS response and its observation to describe the BIS signal processing delay. This time lag (in seconds) increases predictably with patient age from a minimum of 15 seconds. To implement this, BIS observations that are simulated at time t are not observed by the controller until time $t + \text{BIS}_{\text{delay}}$.

2.2.3 Prior construction and patient parameters

While the true patient PK-PD parameters describe the data generating process, the predicted parameters are used to formulate a prior distribution on the patient's PK-PD parameters. These are readily calculated by evaluating the Eleveld model equations. Taking the $V2$ parameter as an example, the formula given in the Eleveld model is

$$\begin{aligned} V2_i &= 25.5 \cdot \frac{WGT_i}{70} \cdot \exp(-0.0156(AGE_i - 35)) \cdot \exp(\eta_{V2}) \\ \eta_{V2} &\sim N(0, 0.565) \end{aligned} \tag{2.3}$$

where the values 70 and 35 are the weight (kg) and age (years) for the reference individual. The value 25.5 is the typical $V2$ parameter value in the population, and -0.0156 is the estimated amount that $V2$ decreases with age. The patient's estimated V2 parameter, $\hat{V}2$, is calculated by simply evaluating the function at the patient's age and weight while ignoring the final term, $\exp(\eta_{V2})$. The formulas for the PK-PD parameters are too elaborate to reproduce them all here; however, each of the varying PK-PD parameters in the Eleveld model are of the same form:

$$\theta_{i,\text{true}} = \theta_{\text{typical}} \cdot g(X_i) \cdot \exp(\eta_i) \tag{2.4}$$

where $g(X_i)$ is a function scaling the typical parameter value according to patient covariates. The prior mean for each parameter is given by $\hat{\theta}_i = \theta_{\text{typical}} \cdot g(X_i)$. Since some PK parameters are correlated with each other (e.g., $V2$ is used in the calculation of $Q2$), we empirically estimate each patient's prior variance-covariance matrix through Monte Carlo simulation. For each patient, 10,000 vectors of random effects, H, are drawn from a multivariate normal distribution:

$$\begin{aligned} H &\sim \text{MVN}(0, \Omega) \\ \Omega &= \text{diag}(\omega_{PK}, \omega_{PD}) \\ \omega_{PK} &= (0.610, 0.565, 0.597, 0.265, 0.345, 0.209) \\ \omega_{PD} &= (0.242, 0.702, 0.230) \end{aligned} \tag{2.5}$$

where ω_{PK} are the variances describing interpatient variability for parameters $(V1, V2, V3, CL, Q2, Q3)$ and ω_{PD} are the variances for $(C_{e50}, k_{e0}, \sigma^2)$. The matrix H has dimensions 10,000 $\times$ with the jth column vector, $\boldsymbol{\eta_j}$ providing the random effects for the jth PK-PD parameter. A vector of "true" values for each PK-PD parameter is calculated by adding each vector of random effects to the corresponding logged predicted parameter value. Letting $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}} = [\hat{\theta}_{i1}, ..., \hat{\theta}_{i9}]^T$ be the vector of predicted PK-PD parameters for patient i, the jth vector of "true" parameter values is

$$\log(\hat{\boldsymbol{\theta}}_{\boldsymbol{ij},\text{true}}) = \log(\hat{\theta}_{ij}) + \boldsymbol{\eta_j} \tag{2.6}$$

where $\hat{\boldsymbol{\theta}}_{\boldsymbol{ij},\text{true}}$ is a vector with length 10,000. A 10,000 $\times$ 9 matrix of "true" PK-PD parameters for the ith patient, $\hat{\boldsymbol{\Theta}}_{i,\text{true}}$, is constructed with $\hat{\boldsymbol{\theta}}_{\boldsymbol{ij},\text{true}}$ as the jth column. The prior variance-covariance matrix is given by the sample covariance of this matrix.

$$\hat{\Sigma}_i = \text{cov}(\log(\hat{\boldsymbol{\Theta}}_{i,\text{true}})) \tag{2.7}$$

The vector of unknown patient-specific PK-PD parameters, $\theta_{i,\text{true}}$, are then modeled with a log-normal prior distribution.

$$\log(\boldsymbol{\theta}_{\boldsymbol{i},\text{true}}) \sim \text{MVN}(\log(\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}), \hat{\Sigma}_i) \tag{2.8}$$

2.2.4 Closed-loop control mechanism

For each simulated induction, BIS observations are generated according to each patient's true PK-PD model at a rate of one observation per 10 seconds. This is selected to reflect the default smoothing rate of BIS VISTA monitoring systems under Monitor Mode III, in which the BIS signal is smoothed in 10-second increments to calculate a time-averaged trend that is responsive to state changes during induction.(Aspect Medical Systems, 2005) As observations become available (delayed by the individual's time-lag), the posterior distribution is estimated using a Laplace approximation at minutes 1, 2, 4, 8, 10, 12, 15, and 20. Following each update to the posterior distribution, the target BIS values obtained from the reference function are translated into target effect-site concentrations by inverting the PD model at the posterior parameter estimates. Effect-site concentrations are then converted into infusion rates using the Shafer-Gregg TCI algorithm.(Shafer and Gregg, 1992) Using each person's posterior PK estimates, the Shafer-Gregg algorithm calculates a 10-second infusion designed to bring a patient's effect-site concentration to the target level as quickly as possible without overshoot.

2.3 Reference function

Within each simulation, BIS targets are specified by a parametric RF. The goal of the RF is to provide a simple but flexible mechanism for identifying BIS targets. We consider two possible RF specifications: a three-parameter exponential reference function (RFe), and a four-parameter sigmoidal reference function (RFs).

$$
\begin{aligned}
RFe(t; B_0, B_f, \lambda) &= (B_0 - B_f)e^{-\lambda t} + B_f \\
RFs(t; B_0, B_f, \beta, t_{50}) &= B_0 - (B_0 - B_f) * \frac{t^\beta}{t_{50}^\beta + t^\beta}
\end{aligned}
\tag{2.9}
$$

In both functions, B_0 defines the target BIS value at time $t = 0$ and B_f is the BIS value desired by the end of induction. In the exponential function, λ is the exponential decay constant. In the sigmoidal function, t_{50} is the time at which the target is set to the midpoint between B_0 and B_f, and β is the "Hill parameter" that defines slope of the curve at t_{50}. For simplicity, we refer to the set of RF parameters for either function as Λ. While a number of alternative RF could have been selected, exponential and sigmoidal functions monotonically decrease from B_0 before asymptoting at B_f, similar to the expected induction path for a patient. Further, since both functions can vary in the starting target, B_0, they both allow induction strategies that include an initial bolus dose, followed by slower infusions, as is typical in inductions. Finally, by setting $B_0 = B_f = 50$, both RF include a fixed reference point of BIS = 50 as a special case.

To simplify the RF parameterization, we require both RF to asymptote at BIS = 50 by fixing $B_f = 50$. To ensure that the induction concludes in a reasonable amount of time we also add the constraint that the RF must pass through the point $BIS = B_f + \epsilon$ at time $t = t_\epsilon$. Here, ϵ is some small distance from the final induction target (BIS=50), and t_ϵ is the time at which the RF must equal $B_f + \epsilon$. We set $\epsilon = 1$ and require $t_\epsilon \leq 10$. This allows us to reparameterize λ and t_{50} in RFe and RFs, respectively, as follows.

$$
\begin{aligned}
\lambda &= \left(log(B_0 - B_f) - log(\epsilon)\right)/t_\epsilon \\
t_{50} &= t_\epsilon \left(\frac{\epsilon}{B_0 - B_f - \epsilon}\right)^{\frac{1}{\beta}}
\end{aligned}
\tag{2.10}
$$

Consequently, the parameter sets are $\Lambda_{RFe} = (B_0, t_\epsilon)$, $\Lambda_{RFs} = (B_0, \gamma, t_\epsilon)$.

2.3.1 Reference function optimization

The reference function effectively defines a dosing protocol to be carried out within a population, though the protocol is carried out differently for each patient (i.e., the same RF values result in different infusion rates due to an individual's PK-PD model). To evaluate how well the protocol works for any specific patient, we can define an individual-level objective function that describes a clinical criterion in terms of the patient's BIS-time curve when the patient targets the RF. Due to population PK-PD variability, any single RF will result in a different BIS-time curve for each patient. Consequently, there will also be a distribution of corresponding individual-level criterion values. To optimize a RF for the population, a population-level criterion is defined that is a function (e.g., the mean) of the individual-level values.

This optimization process is similar to that of optimal experimental design methodologies for "robust" design criteria such as the ED- or minimax-optimal designs.(Dodds et al., 2003; Zhou, 2008) In these cases, the experimenter defines a set of experimental conditions at which to take measurements so as to maximize the information gathered about a set of parameters. During induction of anesthesia, there is little need to determine optimal sampling times as BIS observations are being gathered continuously. Instead, the experimenter must determine the set of infusion rates

to be administered or, more specifically, the parameters for a protocol that will determine how the infusion rates are be identified.

As with the prediction variance criteria, we consider criteria that are a function of the BIS-time response curve; however, we select them to reflect quantities of clinical significance. Specifically, we consider 1) the weighted over and undershoot (WOS) of the region $BIS = [40, 60]$ (hereafter the "target-zone" or TZ), 2) the "stable entry time" (SET), defined as the time until the patient reaches the TZ without subsequently overshooting, and 3) the total dose (TD) of propofol administered until the SET. The three individual-level optimization criteria are described by the equations below in which $f(t)$ is the patient's BIS response at time t.

$$\begin{aligned} \phi_{WOS} &= \alpha \int (f(t) - 60)_+ \, dt + (1 - \alpha) \int (40 - f(t))_+ \, dt \\ \phi_{SET} &= \min(t \mid \forall \; t^* \geq t, f(t^*) \in [40, 60]) \\ \phi_{TD} &= \int_0^{\phi_{SET}} k_R(t) dt \end{aligned} \tag{2.11}$$

The tuning parameter α in ϕ_{WOS} describes the weight placed on TZ undershoot relative to overshoot and the $+$ subscript indicates that only positive values are evaluated. $k_R(t)$ is the propofol infusion rate in milligrams per minute at time t.

Each individual-level criterion, ϕ, is evaluated across the distribution of PK-PD variability in the population. The distribution of MAP parameters incorporates both explained variability from differing covariate values between patients, and unexplained variability due to unobserved patient characteristics. From the population of 122 patients, a random sample of 50 were selected as a training set on which to optimize the RF, while the remaining 72 were used as a hold-out testing set. A summary of patient covariates and MAP PK-PD parameter values, stratified by testing set, is provided in Table 1.

A population-level criterion, Φ, is calculated by applying a function to the distribution of individual-level criterion values, ϕ. Letting θ_0 be the distribution of "true" patient parameters, the population criterion for the mean and variance can be expressed as

$$\Phi(\Lambda) = \int g(\phi(\Lambda, \theta_0)) f(\theta_0) d\theta_0 \tag{2.12}$$

where $f(\theta_0)$ is the probability density function for θ_0 and $g(X) = X$ for the population mean and $g(X) = (X - E[X])^2$ for the population variance. In this chapter, the integral in the above equation is approximated by summing over the 50 patients included in the training data set. The optimal set of RF parameters, Λ^* is given by minimizing $\Phi(\Lambda)$ with respect to Λ: $\Lambda^* = \arg\min_{\Lambda} \Phi(\Lambda)$. Depending on the individual-level criterion selected, either minimizing the population mean or the population variance may be of clinical interest. In the case of the stable entry time, ϕ_{SET}, we hypothesized that both may be of interest: minimizing the average SET may result in reliably rapid inductions, while minimizing the variance in SET between patients may make inductions more predictable and assist in operation planning.

Table 2.1: Summary of covariates and PK-PD parameters for N=122 patients from Eleveld population PK-PD model (used to develop PD component). Prior PK-PD parameters were calculated using patients' covariate values. "True" parameter values are given by maximum *a posteriori* (MAP) estimates for each patient. Parameters with standard deviations of zero are assumed by the Eleveld model to be fixed within the population.

	Train (N=50)	Test (N=72)	Total (N=122)
Covariates (observed)			
Age (years)			
Mean (SD)	36.82 (20.99)	34.49 (20.35)	35.44 (20.56)
Range	3.00 - 74.00	3.00 - 73.00	3.00 - 74.00
Weight (kg)			
Mean (SD)	70.81 (27.38)	69.13 (30.31)	69.82 (29.04)
Range	15.00 - 124.00	15.00 - 141.00	15.00 - 141.00
Height (cm)			
Mean (SD)	161.62 (24.02)	157.33 (22.92)	159.09 (23.38)
Range	100.00 - 190.00	99.00 - 196.00	99.00 - 196.00
Sex			
Female	23 (46.00%)	43 (59.72%)	66 (54.10%)
Male	27 (54.00%)	29 (40.28%)	56 (45.90%)
Opiates coadmin			
No	45 (90.00%)	58 (80.56%)	103 (84.43%)
Yes	5 (10.00%)	14 (19.44%)	19 (15.57%)
PK Parameters (MAP)			
K10			
Mean (SD)	0.40 (0.48)	0.32 (0.21)	0.35 (0.35)
Range	0.05 - 2.55	0.03 - 0.86	0.03 - 2.55
K12			
Mean (SD)	0.30 (0.32)	0.27 (0.26)	0.28 (0.29)
Range	0.06 - 1.33	0.04 - 1.32	0.04 - 1.33
K21			
Mean (SD)	0.08 (0.03)	0.08 (0.03)	0.08 (0.03)
Range	0.03 - 0.20	0.04 - 0.18	0.03 - 0.20
K13			
Mean (SD)	0.18 (0.21)	0.14 (0.10)	0.16 (0.16)
Range	0.02 - 1.19	0.02 - 0.52	0.02 - 1.19
K31			
Mean (SD)	0.0043 (0.0005)	0.0044 (0.0004)	0.0043 (0.0005)
Range	0.0034 - 0.0055	0.0033 - 0.0052	0.0033 - 0.0055
PD parameters (MAP)			
Ke0			
Mean (SD)	0.53 (1.20)	0.86 (1.94)	0.73 (1.68)
Range	0.06 - 6.90	0.04 - 11.69	0.04 - 11.69
E50			
Mean (SD)	3.31 (0.91)	3.10 (0.77)	3.18 (0.84)
Range	1.68 - 5.28	1.68 - 5.45	1.68 - 5.45
Emax			
Mean (SD)	92.98 (0)	92.98 (0)	92.98 (0)
Gamma (Ce $\geq$ C50)			
Mean (SD)	1.47 (0)	1.47 (0)	1.47 (0)
Gamma (Ce $\leq$ C50)			
Mean (SD)	1.89 (0)	1.89 (0)	1.89 (0)
Residual error			
Mean (SD)	8.09 (2.02)	8.24 (1.80)	8.18 (1.88)
Range	4.76 - 14.04	4.86 - 12.43	4.76 - 14.04
Lag time (seconds)			
Mean (SD)	26.36 (11.38)	24.76 (9.44)	25.42 (10.26)
Range	16.17 - 61.01	16.17 - 58.69	16.17 - 61.01

2.3.2 Identification of optimal reference functions from training set

We consider the performance of both exponential and sigmoidal reference functions at five combinations of ϕ and $g(\phi)$ for a total of 10 specifications. Two are the expected value of the weighted overshoot, ϕ_1, at $\alpha = 0.05, 0.2$. This corresponds to weighting the cost of remaining above the region BIS=[40,60] relative to overshooting it at rates of 20:1 and 5:1, respectively. Two more are the expected value and standard deviation of the SET, ϕ_2. The final criterion is the expected value of amount of propofol administered until SET, ϕ_3. Optimal RF parameter values were identified via a grid search with five uniformly distributed values per parameter. For both RFe and RFs, values of t_ϵ range from 0.1 to 10 and values of B_0 range from 55 to 92.982 (the value of $\text{BIS}_{\text{baseline}}$ in the Eleveld model). For RFs, γ ranges from 2 to 5.

A grid approach to optimization was used primarily to minimize the computational time required. Simulating patient responses at each set of TF parameters requires substantial time. Once simulated, however, the values of all optimality criteria can be calculated on the same set of responses. This is in contrast to direct optimization routines, where each optimality criteria would require its own set of simulated responses. Further, the simulation of closed-loop control is a stochastic process: BIS observations are generated randomly, and the evolution of the posterior distribution depends upon the values observed. Consequently, conventional routines may have difficulty converging to a minimum. To remove the stochasticity, we generated a set of residual errors in BIS measurements that were applied to each simulation, such that a patient simulated twice using the same RF would have identical inductions. To account for the Monte Carlo error in the simulated residual errors, each patient in the training set had responses simulated using three sets of errors. Optimal parameters for each RF were identified as the set of parameters that minimized the population-level criterion averaged over these three repetitions.

Reference functions are illustrated in Figure 2.2, colored by performance on each criterion in the training set and with the optimal value bolded. The color gradients show that more rapid inductions perform better for the WOS ($\alpha = 0.2$) and average SET criteria, which both identify the same RFe and RFs curves as optimal. As α decreases from 0.2 to 0.05 for the WOS criterion, the cost of overshooting is up-weighted, resulting in a less-steep RF. SdSET and TD criteria similarly identify more gradual curves as optimal, indicating that less rapid inductions may result in less variability between patients and less propofol administered.

In addition to the five optimality criteria described, we also evaluate, but do not optimize, the average rise time in the population. The rise time is defined as the amount of time between the first infusion and the time at which BIS drops below 60. This provides an indicator of how quickly patients will approach a level of hypnosis at least as extreme as the target level. Rise time was not included as an optimality criterion because it can always be decreased by increasing the amount of propofol administered. Consequently, a more rapid induction will always be favored. Among the induction strategies considered (and, indeed, among all RF restricted to values of 50 or greater), BIS50 will minimize RT. While this serves as a proxy for the time until anesthetized, it is important to note that not all patients will experience a loss of consciousness at BIS = 60. Patients frequently lose consciousness within the first 20-30 seconds of an induction, though it may take longer to reach the target level of anesthesia.(Ilyas et al., 2017)

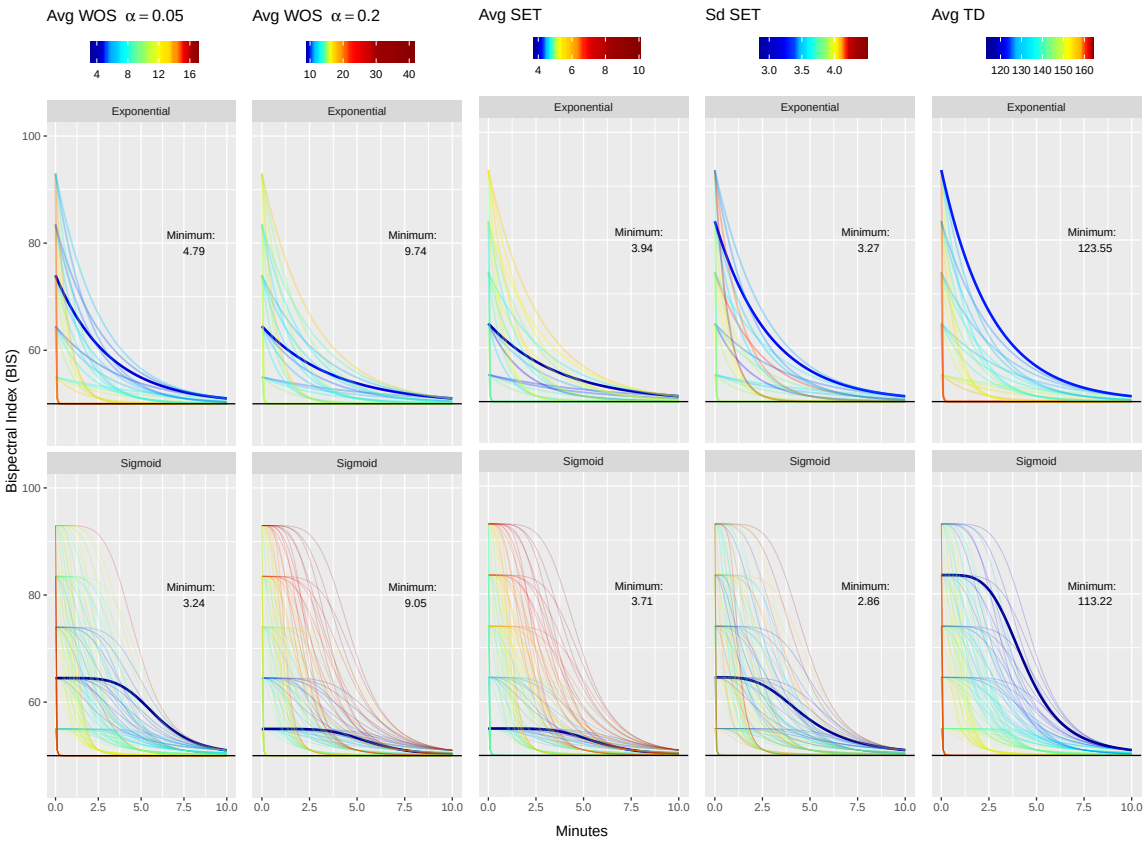

Figure 2.2: Visualization of optimized reference functions. Columns represent each of the five population criterion considered. Within each column, curves are colored according to the value of the population criterion calculated from the average performance of N=50 patients across three iterations. The optimal exponential and sigmoidal reference function for each criterion are highlighted blue. WOS = 'Weighted Overshoot', SET = 'Stable Entry Time', TD = 'Total Dose'.

Table 2.2: Performance of optimized reference functions in testing set of N=72 patients. 'True' parameters are taken to be the empirical Bayes estimates for each of the patients. WOS = 'Weighted Overshoot,' SET = 'Stable Entry Time,' TD = 'Total Dose,' RT = 'Rise Time.'

	Optimal parameters				Test set criterion values					
Reference Function	BIS_0	λ	t_{50}	γ	$\text{WOS}_{\alpha 05}$	$\text{WOS}_{\alpha 20}$	AvgSET	SdSET	TD	RT
BIS50	-	-	-	-	12.67	13.65	4.42	3.68	148.18	**1.16**
OpenLoop	-	-	-	-	18.5	38.7	14.87	5.86	178.76	9.79
RFe-$\text{WOS}_{\alpha 05}$	73.99	0.32	-	-	**3.84**	9.75	4.13	2.89	117.32	2.79
RFe-$\text{WOS}_{\alpha 20}$	64.5	0.27	-	-	3.9	**8.16**	**3.72**	3.42	119.84	2.3
RFe-AvgSET	64.5	0.27	-	-	3.9	**8.16**	**3.72**	3.42	119.84	2.3
RFe-SdSET	83.49	0.35	-	-	4.59	12.47	4.9	**2.88**	117.24	3.5
RFe-TD	92.98	0.38	-	-	5.34	15.59	5.22	2.99	**116.63**	4.1
RFs-$\text{WOS}_{\alpha 05}$	64.5	-	5.94	5	**3.04**	**8.8**	5.15	3.04	116.08	3.55
RFs-$\text{WOS}_{\alpha 20}$	55	-	5.7	5	6.05	8.81	**3.3**	3.14	127.1	1.61
RFs-AvgSET	55	-	5.7	5	6.05	8.81	**3.3**	3.14	127.1	1.61
RFs-SdSET	64.5	-	4.75	3.5	3.18	8.49	4.14	3.11	114.88	3.01
RFs-TD	83.49	-	4.41	4.25	6.29	21.69	6.56	**2.32**	**113.82**	5.91

2.4 Performance of optimized reference functions

2.4.1 Comparison approaches

Each optimized RF was compared to two "standard" clinical approaches to induction: BIS50 and OpenLoop. Under BIS50, the reference value is set to a fixed value of BIS=50 for the duration of the induction. This exemplifies the way in which closed-loop controllers are typically implemented, that is, with a fixed, rather than a time-varying, reference signal. The same Bayesian closed-loop controller is used to evaluate BIS50 as each of the optimized RF.

The OpenLoop approach, by contrast, does not use the closed-loop controller or implement any stopping rule. Under OpenLoop, patients receive infusion rates based on recommended doses of propofol, as per the Diprivan (propofol) package insert.(Fresenius Kabi USA LLC, 2014) Adults (17-54 years) received induction doses of 2.25 mg/kg at a rate of 40mg/10sec followed by a maintenance dose of 10 mg/kg/h. Adults 55 years or older received an induction dose of 1.25 mg/kg at a rate of 20mg/10sec, followed by a maintenance dose of 4 mg/kg/h. Children 16 years and younger received an induction dose of 3 mg/kg over 20 seconds, followed by a maintenance dose of 15 mg/kg/h. These infusion rates are not further modified by TCI control or by patient responses. This is not, however, a realistic portrayal of manual administration in a clinical setting. In practice, the anesthesiologist would modify infusion rates based on patient indicators and behavior (e.g., blood pressure, movement). This action by an anesthesiologist, however, is not easily simulated. OpenLoop is included primarily to illustrate the degree of PK-PD variability in the population under standard dosing guidelines.

2.4.2 Simulation results

To distinguish between the many reference functions under evaluation, each RF is labeled according to its functional form (RFe = exponential, RFs = sigmoidal) and the criterion that it optimized in the training set. For example, RFe-$\text{WOS}_{\alpha 20}$ represents the reference function with an exponential (e) functional form that was optimal for the average WOS criterion when $\alpha = 0.20$. RFs-SdSET is the sigmoidal reference function that minimized the standard deviation of SET.

Each optimized RF and the two comparison approaches are evaluated by simulating inductions for the testing set of $N = 72$ patients. Table 2.2 displays the parameters associated with each RF

and its performance on the testing set. Using the test data, all RFe achieved the minimum value in the criterion for which it was optimized. Among the RFs, RFs-WOS$_{\alpha 05}$ performed equally well as RFs-WOS$_{\alpha 20}$ on the WOS$_{\alpha 20}$ criterion while the RFs-SdSET was outperformed by several on the SdSET criterion. BIS50 was optimal at minimizing rise time, as expected, but did poorly across several other criteria. OpenLoop was the worst-performing method across all criteria.

The reference functions that minimized the average SET in the training sample favored a rapid induction and visually were quite similar to a fixed reference point of BIS = 50. Nonetheless, the small changes translated to a noticeable change in performance in the test set. RFe-SET and RFs-SET had mean (SD) SET values of 3.72 (3.42) and 3.3 (3.14) minutes in the test set compared to 4.42 (3.68) minutes for BIS50. This translates to reductions in average SET of 42 seconds and 67.2 seconds for RFe-SET and RFs-SET, respectively. Figure 2.3 displays selected simulation results for the RFs and RFe optimized for average SET compared to BIS50 and OpenLoop.

The trade-off of achieving a faster entry time by is a slight increase in rise time. As expected, BIS50 minimized rise time with an average of 1.16 (0.75) minutes until patients dropped below BIS = 60. The RFs optimized for average SET was the next fastest with an average rise time that was 27 seconds slower at 1.61 (1.27) minutes. The third fastest was the RFe optimized for average SET was 68.4 with a rise time of 2.3 (2.49) minutes. Panels A and B of Figure 2.4 shows the proportion of patients stably within the target zone and the proportion of patients under BIS = 50, respectively, for the first ten minutes using these three methods. At each minute mark, more patients are within the target zone under RFs-SET than BIS50, with a maximum difference of 49 patients (68.1%) in the target zone at three minutes versus 32 (44.4%) under BIS50 and 43 patients (59.7%) under RFe-SET. At the same time, however, 69 (95.8%) of patients were below BIS = 60 at three minutes under BIS50, versus 67 (93.1%) under RFs-SET and 60 (83.3%) under RFe-SET.

Figure 2.5 displays the BIS-time profiles of the test set patients for the RFe and RFs optimized for $WOS_{\alpha 05}$, SdSET, and TD. For conciseness, direct comparisons to BIS50 and OpenLoop are not shown for each of these, though the BIS-time curves without the appropriate coloring can be referenced in Figure 2.3. Performance of RF optimized for $WOS_{\alpha 20}$ are identical to those optimized for AvgSET and are also not displayed.

Among all reference functions, the standard deviation of SET was minimized by the RFs-TD, which had a value of 2.32 min. The RF optimized for SdSET had values of 3.42 min and 3.11 for RFe-SdSET and RFs-SdSET, respectively. The SdSET criterion was intended to measure variability between patients' responses. To that end, it may function well, as the RFs-TD curve displayed in Figure 2.5 clearly appears to minimize variability between patients' BIS-time curves. Nonetheless, the absence of a clear gradient in the SdSET column of Figure 2.2 suggests that the parameter surface was less-smooth than the other objective functions. Consequently, more than three Monte Carlo iterations may be necessary to reliably identify an optimally performing set of parameters.

Panel C of Figure 2.3 compares the distribution of SET under BIS50 and the two best-performing reference functions for this criterion: RFs-TD and RFe-SdSET. Though the average SET increases with both, the variability in SET decreases substantially, with the middle 80% of patients entering between 0.72 min and 9.66 min (difference 8.94 min) under BIS50 versus 5.17 min and 8.28 min (difference 3.11 min) for RFs-TD and 2.22 min and 8.36 min (difference 6.13 min) for RFe-SdSET.

For total dose (TD) criterion, BIS50 and OpenLoop averaged (SD) 148.18 (68.02) mg and 178.76 (113.72) mg propofol, respectively, to reach the zone BIS = [40,60]. By contrast, RFe-TD reduced TD

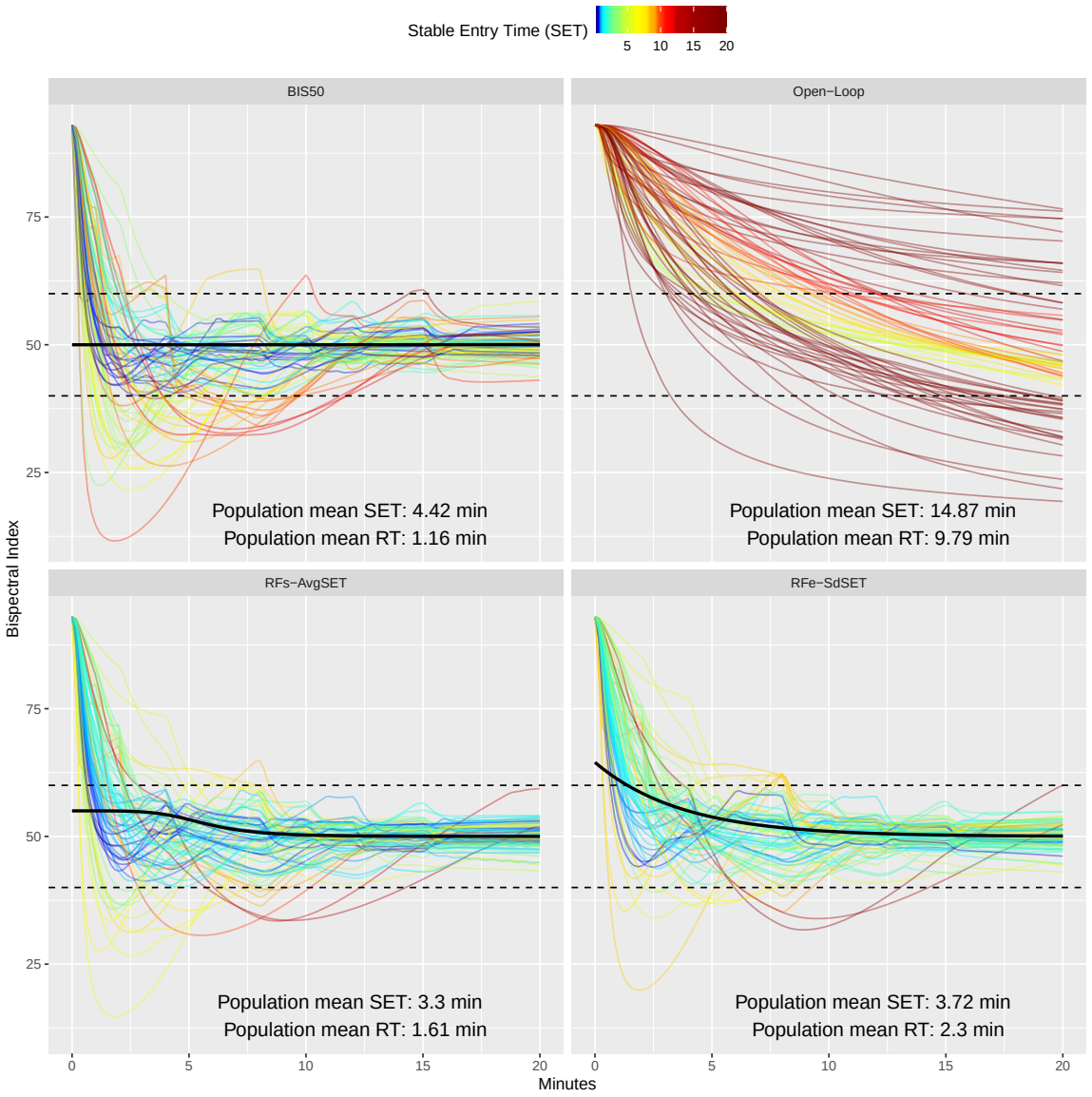

Figure 2.3: Simulated inductions for N = 72 test set patients. Panels display inductions using a fixed reference point of BIS=50 (top left), an open-loop induction using recommend infusion rates based on patient age and weight (top right), a sigmoid reference function optimized to minimize average stable entry time (bottom left), and an exponential reference function optimized to minimize average stable entry time (bottom right). SET = 'Stable Entry Time,' RT = 'Rise Time.'

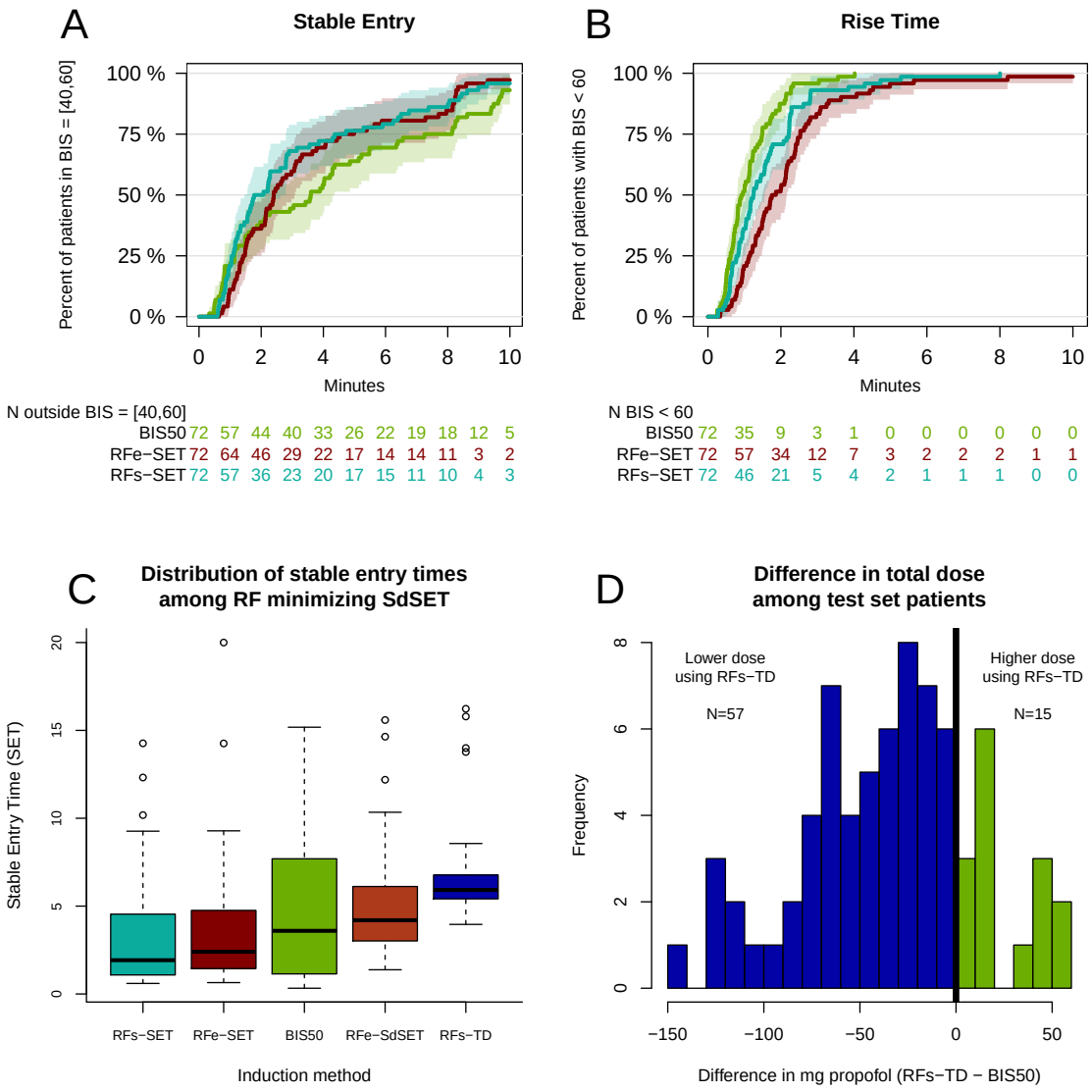

Figure 2.4: Comparison individual-level criteria within the testing group. Panel A shows the percentage of the patients in target zone of BIS = [40,60] over the first 10 minutes of the induction for BIS50 and the two RF optimized to minimize average SET: RFe-SET and RFs-SET. Panel B shows the percentage of patients below BIS = 60 at any point. Panel C shows the distribution of SET for BIS50 and the two RF that minimized the standard deviation of SET: RFe-SdSET, RFs-TD. Panel D shows the difference in propofol doses required for patients to reach the target zone under BIS50 and RFs-TD, with negative values indicating that lower doses were used for RFs-TD. RF = 'Target Function', RFe = 'RF-exponential', RFs = 'RF-sigmoid', SET = 'Stable Entry Time', TD = 'Total Dose.'

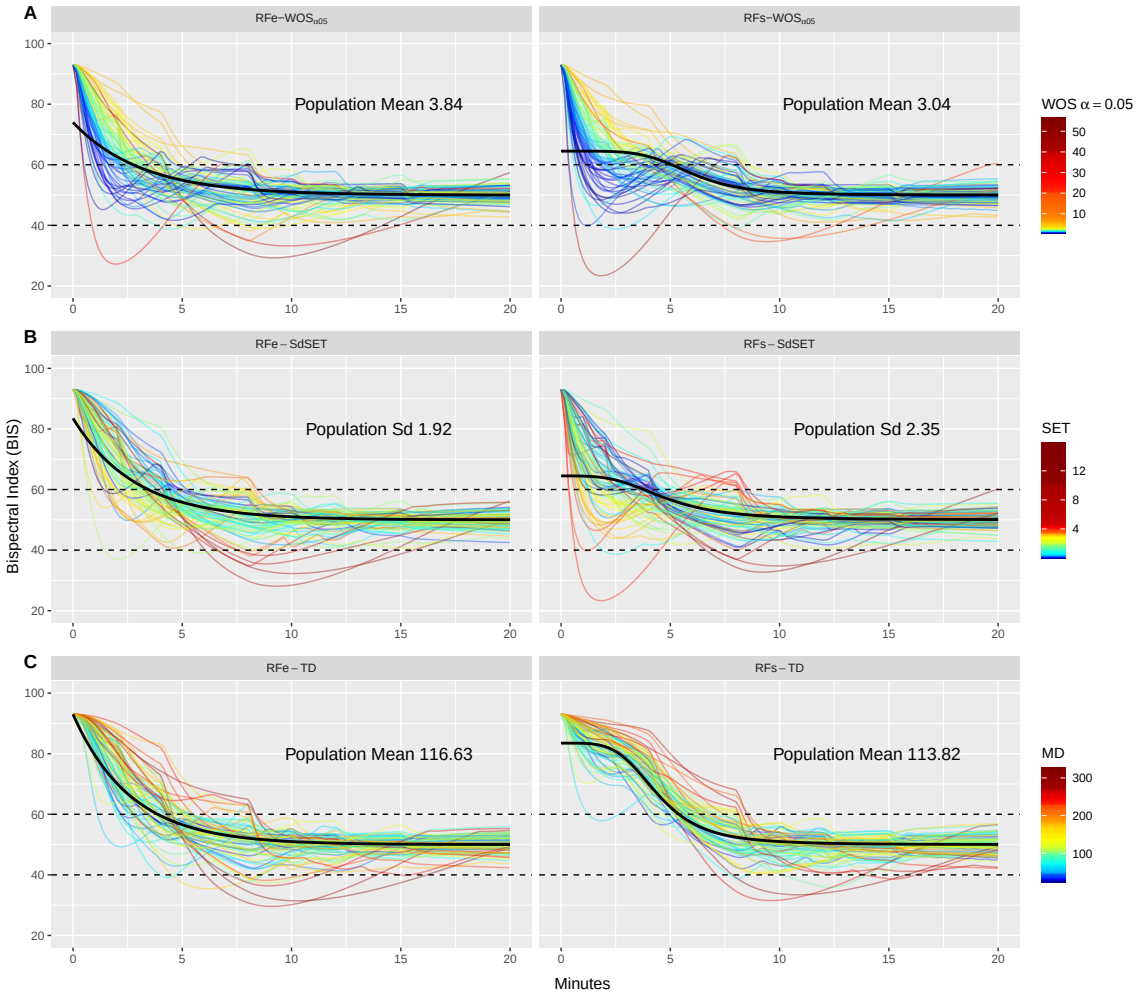

Figure 2.5: Simulated BIS-time curves for N = 72 test set patients at optimized reference functions. First and second columns display RFe and RFs, respectively. Panels A, B, and C show inductions optimized for WOS at $\alpha = 0.05$, standard deviation of SET, and MD, respectively. RFe = 'RF-exponential', RFs = 'RF-sigmoid,' WOS = 'Weighted Overshoot,' SET = 'Stable Entry Time', TD = 'Total Dose.' 'Population Mean' and 'Population Sd' refer to the mean and SD of the criterion values for the 72 patients.

to 116.63 (69.97) mg (a 21.3% decrease relative to BIS50) and RFs-TD reduced TD to 113.82 (65.36) mg (a 23.2% decrease). Panel D of Figure 2.3 displays the distribution of differences in amount of propofol received per patient between BIS50 and RFs-TD. Of the 72 patients, 57 (79.17%) received less propofol under RFs-TD than under BIS50 with an average decrease of 50.2 mg and a maximum decrease of 144.29 mg. A total of 15 (20.83%) patients received more propofol under RFs-TD with an average increase of 25.82 mg and a maximum increase of 59.97 mg.

2.5 Discussion

In this chapter we propose a method of defining and optimizing reference functions for BIS controllers when inducting patients into general anesthesia with propofol. These reference functions implement optimal dosing protocols for a specified population and criterion. Simulations from the Eleveld et al. population PK-PD model indicate RF use can improve performance across the several criteria relative to a traditional fixed reference point signal and an open-loop approach. The one exception to this is in minimizing the rise time, for which more rapid infusions will always result in faster sedation. However, a minimal rise time may come at the cost of additional overshoot, longer time until stabilization at the target level of sedation, greater variability between patient responses, and more medication administered. Four of the reference functions, RFe-$WOS_{\alpha 05}$, RFe-AvgSET, RFs-AvgSET, and RFs-SdSET, outperformed BIS50 in each of these criteria. If rapid sedation is not the only criterion of interest, or if there are concerns of harm due to inadvertently overdosing patients, then a reference function approach may provide superior results relative to targeting a BIS value of 50 directly.

The objective functions in this chapter were intended to reflect quantities of clinical interest. Depending on the context, a clinician may reasonably prefer one criterion over another or may wish to select their own. Additionally, the clinical significance of any gains made may depend upon the context. For example, a reduction in the variability between responses may hold less significance in a population of young and healthy patients than among elderly or frail patients, the latter for whom an unexpectedly high effective dose may carry a higher risk.

2.5.1 Limitations

The primary limitation of the RF methodology is that it may be suboptimal for populations that are not adequately described by the PK-PD model. This is true for any model-based method; however, it is an important consideration when applying the methodology to patients that are substantially different than those for whom the model was developed. The diversity of the population in Eleveld PK-PD model mitigates this concern, as it includes data from neonates, the elderly, and high-BMI individuals. Nonetheless, Eleveld et al. caution that BIS data for children under 12 came from only one study, were absent for adolescents 12-19 years old, and were limited to three individuals older than 67 years. Consequently, the PD component has limited support within these populations.

A related limitation is that the optimization process does not account for sources of variability or error beyond that described by the population PK-PD model. It has been noted that the PK of propofol, and IV agents more broadly, depend upon the method of administration (Bibian et al., 2006; Struys et al., 2007; Masui et al., 2009), with slower infusions increasing the accuracy of effect-site model predictions.(**?**) This is believed to occur due to the violation of the instantaneous mixing

assumption made by traditional PK models. Similarly, it has been suggested that the accuracy of BIS measurements may also decline when tracking rapid changes.(Hajat et al., 2017) While this suggests that reducing infusion rates via implementation of a reference function may increase the accuracy of model predictions, this consideration has not been incorporated into our analysis.

Finally, while the proposed methodology can be readily implemented by alternate BIS controllers, the performance of a specific reference function may vary when an alternate controller is used. This can be addressed by identifying optimal RF with the new controller; however, the optimization process is more computationally intensive than the evaluation. The Bayesian controller implemented in our simulations required the computationally intensive step of maximizing a patient-specific posterior distribution following each data collection period. This step is only required during the optimization process and would not affect implementation of a RF within a clinical setting. Nonetheless, controllers that do not require posterior estimation (e.g., PID controllers) may substantially increase the computational speed relative to Bayesian controllers. Code to simulate open- and closed-loop control for intravenous compartment PK/PK-PD models is freely available in the R package tci and can be readily modified to incorporate user-specified population PK-PD models, TCI algorithms, or closed-loop controllers.

Chapter 3

R statistical software package: tci

3.1 Introduction

3.1.1 Background

Target-controlled infusion (TCI) systems automate the delivery of medication by administering intravenous infusions that are calculated to reach a target concentration in a tissue or compartment of interest. Traditionally, intravenous drugs are delivered manually by a clinician either by bolus dose or via an infusion pump. Due to the accumulation of a drug within a patient, however, the precise dose-concentration relationship is difficult to predict.(Struys et al., 2016) Further, this relationship may depend strongly upon observable patient characteristics, such as age and weight, as well as unobserved differences between patients.

Effectively, TCI devices are computerized syringes equipped with a microprocessor and a user interface.(Al-Rifai and Mulvey, 2016) Using the interface, a clinician will select a pharmacokinetic (PK) model believed to represent the patient and a target concentration. The TCI device will then apply an algorithm that calculates a series of infusion rates designed to achieve and maintain the target concentration. Due to the complex nonlinear dynamics of drug absorption, circulation, and excretion, maintaining a constant concentration requires continuously modulating infusion rates based on the patient's underlying PK and the amount of drug given. TCI systems facilitate this process by maintaining a history of all previously administered doses as well as an underlying model that describes the patient's predicted PK based on observable characteristics that may influence PK, such as weight, age, and sex.(Schnider et al., 2016) Based on the patient's response and feedback acquired from various sensors, a clinician will subsequently raise or lower the target passed to the TCI system.

TCI systems have been used in anesthesiology for over 20 years and have become a routine part of care in many countries. (Absalom et al., 2016) The use of TCI is intuitive to anesthesiologists and has been found at least as safe as alternate modes of delivery.(Struys et al., 2016; Schnider et al., 2016) With the coupling of a pharmacodynamic (PD) model, TCI systems can be extended to target a desired patient response rather than a concentration. The PD model is inverted to provide the target effect-site concentration, which is then converted into a dose by the TCI algorithm. Beyond routine clinical use, TCI has been an essential research tool, allowing researchers to characterize the PK-PD of various drugs, understand interactions between combinations of drugs, and understand drug behavior in animal models.(Absalom et al., 2016)

3.1.2 Existing software and need for this package

A variety of R packages exist dedicated to the design and analysis of PK data and can be readily found at the CRAN task view for pharmacokinetics (https://cran.r-project.org/web/views/Pharmacokinetics.html). While several of these packages facilitate simulation from PK models, there currently are no packages available in the R programming language that implement TCI algorithms or simulate patient responses under TCI control.

At the moment, TCI devices have not yet received regulatory approval by the FDA. Conse-

quently, there is no regulatory framework for operating a TCI device within the United States. (Absalom et al., 2016; Dryden, 2016) Most research involving TCI has been conducted by academic research groups using software-only TCI platforms. A complete list of academic TCI systems can be found in Absalom et al. (2016). The majority of these platforms have either been commercialized or not made their source code available. The exceptions are those provided through the "Open TCI" initiative (http://opentci.org/). These include STANPUMP, developed by Steven L. Shafer at Stanford University, STELPUMP, developed by Johan Coetzee and Ralph Pina at the University of Stellenbosch in South Africa and IVA-SIM developed by Jurgen Schuttler and Siegbert Kloos of the Department of Anesthesiology at the University of Bonn. STANPUMP is written in C while STELPUMP and IVA-SIM are available with source code as MS-DOS executable files. While each of these are freely available for download and contain a several commonly-used PK models, there is a high barrier to entry for researchers unfamiliar with these programs. Additionally, they are each intended to be compatible with mechanical pumps and are not easily modifiable by a casual user. Within the Python programming language, the library PyTCI implements TCI applied to several population PK models commonly used for propofol, remifentanil, alfentanil, and dexmedetomidine. The package, however, appears to still be under development.

This package, `tci`, provides R users with the ability to simulate PK-PD dynamics for a variety of models under TCI control. It is intended strictly for simulation purposes and should not be used with physical devices to administer medication to patients as it does not contain the necessary safety protocols embedded within it. The main strengths of `tci` are its ease of use for users unfamiliar with PK-PD modeling or TCI algorithms and its modular format that allows users to specify alternate PK-PD models or TCI algorithms. It also extends TCI control to incorporate a PD response, it offers a variety of convenience functions for visualizing PK-PD responses, and it allows for simulation in open- and closed-loop settings.

`tci` incorporates closed-form solutions for IV dosing in 1-, 2-, and 3-compartment PK models, as well as 3-compartment models with a fourth metabolite/effect-site compartment. PK model code is based on solutions provided by Abuhelwa et al. (2015). This component is not unique among R packages, with several packages providing implementations of PK or PK-PD models. The R package **linpk**(Rich, 2021) similarly provides solutions for compartmental PK models defined by linear systems of ordinary differential equations (ODE) with a convenient syntax and allows for flexible dosing schedules (e.g. combinations of oral doses, boluses, and infusions). The packages **mrgsolve** (Baron, 2021), PKPDsim, and **RxODE** (Wang et al., 2015) incorporate ODE solvers for user-defined models, allowing for more flexibility in model specification than in **tci**. Nonetheless, custom PK models created through these packages can be used with `tci` by a process illustrated in the examples.

3.1.3 Propofol example

Throughout this chapter, several functions are illustrated for the intravenous hypnotic drug propofol. Propofol is commonly used for the induction and maintenance of anesthesia with PK typically described by a three-compartment model. Several population PK models have been proposed for propofol. Thorough descriptions of these models can be found in Sahinovic et al. (2018). For illustration, we use the Eleveld model described in 2. The Eleveld model includes an effect-site compartment that is linked to an Emax pharmacodynamic model that describes the patient's Bispectral

Index (BIS) value.

$$BIS(t) = E_0 - E_{max}\left(\frac{C_e(t)^\gamma}{C_e(t)^\gamma + C_{e50}^\gamma}\right) \tag{3.1}$$

In the above equation, E_0 describes the BIS response with no drug infused, E_{max} is the maximal effect, C_{e50} is the concentration required for 50% effect, and γ is the slope of the dose-response curve at $C_e = C_{e50}$. The Emax model can be inverted to describe the target effect-site concentration required for a specific effect.

3.2 Theory

In this chapter, we illustrate the 'tci' package using a three-compartment model equipped with an additional fourth effect-site compartment that can be linked to a pharmacodynamic model. A description of the three-compartment model can be found in Section 2.2.

3.2.1 Notation

The following notation is used throughout this chapter.

- t: Time, either a scalar or a vector of values
- C_p, $C_p(t)$: Scalar or vector of plasma concentration(s) at time t. The explicit use of t may be dropped for convenience.
- C_e, $C_e(t)$: Scalar or vector of effect-site concentration(s) at time(s) t.
- C_T, $C_T(t)$: Scalar or vector of target concentration(s) (plasma or effect-site) at time(s) t.
- C_0: Scalar or vector of initial concentrations.
- $\mathbf{C}$: Matrix of concentrations for a set of time values.
- θ_{PK}, θ_{PD}: Vector of patient PK and PD parameters that are used by a PK or PD model.
- $\theta_{PK}^{(0)}$, $\theta_{PD}^{(0)}$: Vector of "true" patient PK and PD parameters. These will never be known in practice, but can be used to simulate model misspecification.
- k_R, $k_R(t)$: Scalar or vector of infusion rates (e.g. mg/min). Units will depend upon the units of the PK model parameters.

3.2.2 TCI Algorithms

Within the `tci` package, we implement two TCI algorithms: the Jacobs Algorithm for plasma-targeting and the Shafer-Gregg Algorithm for effect-site targeting.(Jacobs, 1990; Shafer and Gregg, 1992)

3.2.2.1 Jacobs' algorithm for plasma targeting

The Jacobs algorithm is based on the observation that, within sufficiently short time periods, the relationship between infusion rates and plasma concentrations is approximately linear. By calculating the plasma concentration at two arbitrary points, one can estimate the linear relationship between infusion rate and plasma concentration and extrapolate to find the infusion rate needed to reach the target concentration.

$C_T(t)$ and $C_p(t)$ are the target and predicted plasma concentrations at time t, respectively. Let Δt be the interval of time between target and/or infusion rate updates. At update time t^*, the Jacobs algorithm calculates the infusion rate, $k_R(t^*)$, of duration Δt, that is required to reach the target plasma concentration, such that $C_p(t^* + \Delta t) = C_T(t^*)$. It does this by selecting two arbitrary infusion rates, (x_1, x_2), and calculating the predicted plasma concentration as though each infusion were administered during the next interval, given prior concentrations. Let $C_p^{(x_1)}$, $C_p^{(x_2)}$ be the predicted plasma concentrations at time $t^* + \Delta t$ if infusions x_1 and x_2 are administered, respectively.

Assuming a linear relationship between the infusion rate and the plasma concentration, the slope, m, and intercept, b, of the infusion-plasma line are

$$m = \left(C_p^{(x_2)} - C_p^{(x_1)}\right) / (x_2 - x_1) \tag{3.2}$$

$$b = C_p^{(x_1)} - mx_1 = C_p^{(x_2)} - mx_2 \tag{3.3}$$

The infusion rate required to reach target $C_T(t^*)$ at time $t^* + \Delta t$ is

$$k_R(t^*) = \left(C_T(t^*) - b\right) / m \tag{3.4}$$

Since the calculation of $\left(C_p^{(x_1)}, C_p^{(x_2)}\right)$ requires evaluation of the PK model given the current concentrations at time t^*, a set of state variables representing compartment amounts or concentrations must be maintained and updated following each interval, Δt. The Jacobs' algorithm allows target concentrations to be modified as frequently as desired and does not require that intervals be constant. Larger intervals will result in less precise control but greater computational speed due to fewer updates. If the intervals are made too long the linearity assumption may not hold as well. This may result in less-accurate infusion rates if overall drug clearance from the central compartment is rapid relative to the interval duration. By default, Δt is set to 10-seconds but can be modified to larger or smaller values.

3.2.2.2 Shafer-Gregg algorithm for effect-site targeting

For many drugs, a hysteresis is observed between the plasma concentration and the clinical effect due to the time required to reach equilibrium between the plasma and the site of drug action.(Absalom et al., 2009) While targeting plasma concentrations will eventually achieve the target concentration in the effect-site, this process will not provide the most precise control over the drug effect on the patient. Instead, effect-site concentrations can be achieved more quickly by overshooting the target in the central compartment with a bolus dose or rapid infusion that is then stopped while the drug is transferred from the central to the effect-site compartment.

The Shafer-Gregg algorithm identifies the dose required to attain the target effect-site concen-

tration in the shortest amount of time without overshoot. Let $C_T(t^*)$ and $C_e(t^*)$ be the target and predicted effect-site concentrations at update time t^*. After administering an infusion rate of $k_R(t^*)$ for time Δt that is then discontinued, the plasma concentration will peak at time $t^* + \Delta t$, while the peak effect-site concentration will occur at a later time, $t_{peak} > t^* + \Delta t$. The Shafer-Gregg algorithm calculates k_R so that the effect-site concentration peaks exactly at the target: $C_e(t_{peak}) = C_T(t^*)$. The effect-site concentration at any time can be calculated as the superposition of the concentrations resulting from all previously administered doses. At time t^*, the effect of all prior doses is described by the effect-time course if $k_R(t^*) = 0$. The contribution of an additional infusion, $k_R(t^*) > 0$, to the resulting effect-site concentration is given by the predicted effect-site concentrations when all initial concentrations are set to zero.

Let $B(t) = C_e(t|k_R(t^*) = 0)$ represent the effect-site concentration for $t \geq t^*$ when no additional drug is given. The effect-site concentrations resulting from an infusion of rate I can be represented as the product of I and the effect-site concentrations resulting from an infusion of rate one: $C_e(t|k_R(t^*) = I) = I \times C_e(t|k_R(t^*) = 1) \equiv I \times E(t)$. The effect-site concentration following an infusion of I is the sum of these two concentrations: $C_e(t) = B(t) + I \times E(t)$. By adding the restriction that $C_e(t_{peak}) = C_T(t^*)$, we can solve for the infusion rate required to reach $C_T(t^*)$ as a function of t_{peak}.

$$I = \frac{C_T(t^*) - B(t_{peak})}{E(t_{peak})} \tag{3.5}$$

If all compartments have concentrations of zero, $B(t_{peak}) = 0$ and I is given by the ratio of the target effect-site concentration to the maximum effect-site concentration following a unit infusion of duration Δt. If prior concentrations are non-zero, the effect-site concentration will peak earlier, such that $t_{peak} < t_{peak,0}$, where $t_{peak,0}$ is the peak concentration time of $E(t)$. When this is the case, the Shafer-Gregg algorithm implements an iterative search for I and t_{peak} on the range $[0, t_{peak,0})$. The pseudo-code for this algorithm is provided in Algorithm 1.

Algorithm 1 Iterative search algorithm for t_{peak}, I

```
Define t_search ≫ t_peak
tms = sequence(0, t_search)
t_peak,1 = arg max(E(tms))
Initialize value t_peak,0 < t_peak,1
while t_peak,0 ≠ t_peak,1 do
  t_peak,0 = t_peak,1
  I_0 = (C_T(t*) − B(t_peak,0))/E(t_peak,0))
  Ĉ_{E,I_0} = B(tms) + E(tms) × I_0
  t_peak,1 = max(Ĉ_{E,I_0})
end while
I = (C_T(t*) − B(t_peak,1))/E(t_peak,1))
```

3.3 Examples

To illustrate the functions within this package, we consider the case of preparing a dosing schedule for a 50 year old male who weights 60 kg and is 163 cm tall with coadministration of opiates using the Eleveld PK-PD model. The process would be the same for any other patient or PK model,

though the units of measurement may change for different drugs or PK models.

3.3.1 Pharmacokinetic model

The `marsh_poppk()`, `schnider_poppk()`, and `eleveld_poppk()` functions implement the published population PK models and can be used to evaluate the respective models at a set of patient covariates to identify the point estimates for each patient. Alternately, the functions can be used to randomly sample a new set of parameter values based on the unexplained interpatient variability (i.e. not accounted for by covariates) within the models' respective populations.

Table 3.1 displays the predicted and true PK-PD parameter values for the example patient.

Table 3.1: Predicted and true PK-PD parameters for an example 50 year old, 163 cm tall, male patient weighing 60 kg and with coadministration of opiates using the Eleveld PK-PD model. 'Predicted' values, θ, are expected parameter values at patient covariates. 'True' values, θ_0, are equal to predicted values with random effect terms drawn from a multivariate normal distribution describing interpatient variability. L = Liters, S = Seconds, BIS = Bispectral Index.

	PK parameters (Units)						
	V_1 (L)	V_2 (L)	V_3 (L)	CL (L/min)	Q_2 (L/min)	Q_3 (L/min)	K_{e0} (min^{-1})
θ	8.99	17.3	120.96	1.38	0.92	0.61	1.29
θ_0	5.51	19.86	63.42	3.14	1.24	0.26	2.09

	PD parameters (Units)						
		Fixed parameters					
	C_{e50} (ng/L)	E_0 (BIS)	E_{max} (BIS)	γ_1	γ_2	Delay (S)	σ (BIS)
θ	2.8	93	93	1.47	1.89	28.26	8.03
θ_0	4.03	93	93	1.47	1.89	28.26	6.94

The structural three-compartment effect-site model is implemented in the function `pkmod3cptm()`, which takes a vector of times, "tm," a singular infusion rate, "kR," and a vector of named PK parameters, "pars," as required arguments. It returns a matrix of concentrations associated with the times under a constant infusion rate of "kR." Optional arguments include initial concentrations, "init," infusion starting time (if different from zero), "inittm," and an option of which compartment concentrations to return, "returncpt." One, two, and three-compartment models are provided by functions `pkmod1cpt()`, `pkmod2cpt()`, and `pkmod3cpt()`, respectively. PK models are assigned S3 class "pkmod" and are equipped with `plot.pkmod()` and `predict.pkmod()` methods.

3.3.2 TCI dosing schedules

Given a patient model and a set of parameters, the next step is to identify an appropriate dosing schedule. The `tci` package requires dosing schedules to be specified as a matrix with column names "infrt", "begin", and "end" designating to infusion rates and corresponding begin and end times. The user may specify this matrix directly if they wish; however, for TCI applications, the dosing schedule will be determined by the TCI algorithm to meet the designated targets.

TCI algorithms in the package are structured to return a single infusion rate associated with a single target and an infusion duration. he Jacobs and Shafer-Gregg algorithms are implemented in the functions `tci_plasma()` and `tci_effect()`, respectively, and require a target concentration, infusion duration, a PK model, and a set of parameters as arguments.

```
theta <- c(v1 = 8.995, v2 = 17.297, v3 = 120.963, cl = 1.382,
           q2 = 0.919, q3 = 0.609, ke0 = 1.289, c50 = 2.8, gamma = 1.47,
           gamma2 = 1.89, e0 = 93, emx = 93, sigma = 8.03, bis_delay = 28.263)
kR_Cp <- tci_plasma(Cpt = 2, dtm = 1, pkmod = pkmod3cptm, pars = theta[1:7])
kR_Ce <- tci_effect(Cet = 2, dtm = 1, pkmod = pkmod3cptm, pars = theta[1:7])
print(round(c(kR_Cp = kR_Cp, kR_Ce = kR_Ce),2))
```

```
## kR_Cp kR_Ce
## 21.04 28.98
```

In practice, we are typically interested in either reaching and maintaining a set target, or reaching a series of targets. The function `tci()` is used to iterate TCI algorithms over a set of targets and a corresponding times at which the target should be set. To illustrate, we consider calculating the dosing schedule required to reach a target of 1 mg/L of propofol for the first minute, followed by 2 mg/L for the next five, followed by 2.5 mg/L for five more minutes. The inputs to `tci()` are a set of target concentrations, "Ct", a set of times at which to begin targeting those concentrations, "tms", a PK model function, "pkmod", such as `pkmod3cptm()`, a set of PK parameters, "pars", and the type of TCI algorithm to be used, "tci_alg".

```
inf_plasma <- tci(Ct = c(1,2,2.5,2.5), tms = c(0,1,6,11), pkmod = pkmod3cptm,
                  pars = theta, tci_alg = "plasma")
inf_effect <- tci(Ct = c(1,2,2.5,2.5), tms = c(0,1,6,11), pkmod = pkmod3cptm,
                  pars = theta, tci_alg = "effect")
```

The output of `tci()` is a matrix with a set of infusion rates, corresponding begin and end times, target concentrations, "Ct," and concentrations within each of the compartments at the beginning and end of each infusion.

```
head(inf_plasma,3)
```

```
##          infrt     begin       end       dtm Ct c1_start    c2_start
## [1,] 55.436656 0.0000000 0.1666667 0.1666667  1        0 0.000000000
## [2,]  2.901432 0.1666667 0.3333333 0.1666667  1        1 0.004454062
## [3,]  2.892892 0.3333333 0.5000000 0.1666667  1        1 0.013230688
##          c3_start  c4_start c1_end      c2_end       c3_end    c4_end
## [1,] 0.0000000000 0.0000000      1 0.004454062 0.0004231955 0.1009869
## [2,] 0.0004231955 0.1009869      1 0.013230688 0.0012615770 0.2747853
## [3,] 0.0012615770 0.2747853      1 0.021929939 0.0020992555 0.4149843
```

```
head(inf_effect,3)
```

```
##         infrt     begin       end       dtm Ct c1_start    c2_start
## [1,] 85.55203 0.0000000 0.1666667 0.1666667  1 0.000000 0.000000000
## [2,]  0.00000 0.1666667 0.3333333 0.1666667  1 1.543239 0.006873685
## [3,]  0.00000 0.3333333 0.5000000 0.1666667  1 1.462470 0.020058365
```

```
##          c3_start  c4_start   c1_end      c2_end       c3_end    c4_end
## [1,] 0.0000000000 0.0000000 1.543239 0.006873685 0.0006530919 0.1558470
## [2,] 0.0006530919 0.1558470 1.462470 0.020058365 0.0019127340 0.4159028
## [3,] 0.0019127340 0.4159028 1.386165 0.032434641 0.0031054647 0.6105199
```

By default, infusion rates are calculated per 1/6th unit of time (i.e. 10 seconds if infusion rates are in terms of minutes), however, this can be adjusted as needed if more granular control is needed. The output from `tci()` has class "tciinf" for which a plotting method is illustrated in Figure 3.1. Plots are created with the **ggplot2** library arranged using the `gridExtra::arrangeGrob()` function. The base plots can be modified (e.g. to add appropriate units) using **ggplot2** or **gridExtra** functions.

In this example, we observe that the plasma-targeting algorithm reaches the targets almost instantaneously within the central compartment and then adjusts infusions to maintain a constant concentration. In time, the effect-site (i.e. "Cmpt4") equalizes with the central compartment to reach the target concentration. The effect-site algorithm, by contrast, overshoots the target concentration in the central compartment and then switches off the pump in order to cause the effect-site concentration to rise as quickly as possible without overshooting the target.

3.3.3 TCI dosing schedules with a PD model

The **tci** package can extend TCI algorithms to PD models by specifying a PD function and its inverse within the function `tci_pd()`. The Emax model given in Equation 3.1 has the inverse

$$C_e(t) = \left(\frac{(BIS(t) - E_0) C_{e50}^{\gamma}}{E_{max}\left(1 - \frac{BIS(t)-E_0}{E_{max}}\right)} \right)^{1/\gamma} \tag{3.6}$$

The function `tci_pd()` extends the `tci()` function to PD models using the same syntax, but with additional PD and PD inverse functions as arguments. As with `tci()`, it returns an object with class "tciinf" for which the `plot.tciinf()` method can be used. We illustrate this by targeting BIS values of 40, 50, and 60 in five-minute increments for a 15-minute dosing period.

```
tci_bis <- tci_pd(pdresp = c(40,50,60,60), tms = c(0,5,10,15), pdinv = inv_emax,
                  pdmod = emax, pkmod = pkmod3cptm, pars_pk = theta[1:7],
                  pars_pd = theta[8:12])
```

3.3.4 Simulation functions

The functions defined so far will identify a TCI dosing schedule for a patient based on their PK or PK-PD model. In many instances we may be interested in simulating patient responses to medications administered under TCI control and evaluating how these change with model misspecification. The function `gen_data()` can be used to simulate such responses.

```
##       time    c1    c2    c3    c4 timeobs    pd0   pdobs
## [1,] 0.000 0.000 0.000 0.000 0.000  28.263 93.000 100.000
## [2,] 0.167 8.181 0.043 0.003 1.300  28.430 78.168  80.872
```

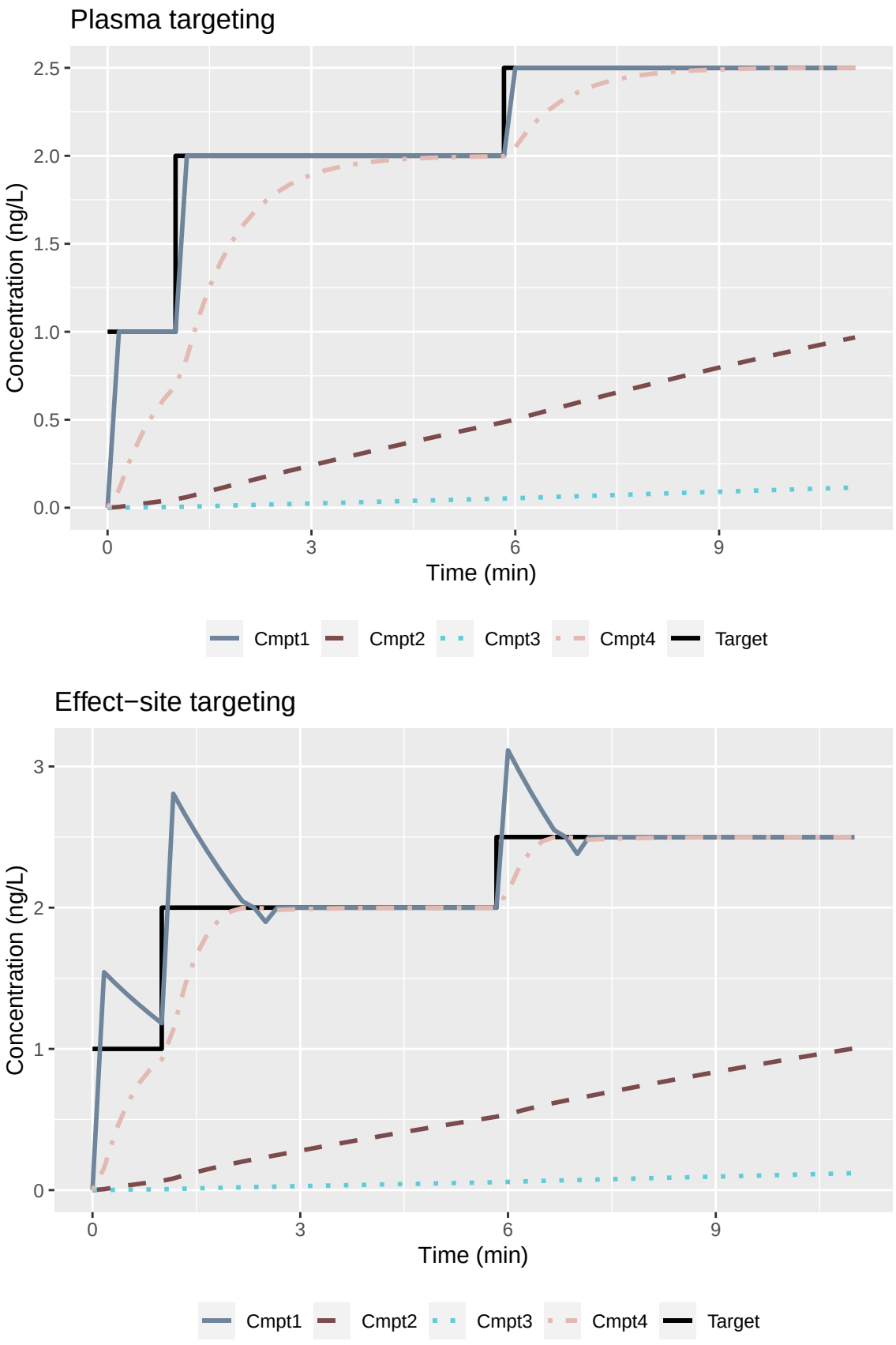

Figure 3.1: Predicted patient responses to infusion schedule defined by 'tci' function.

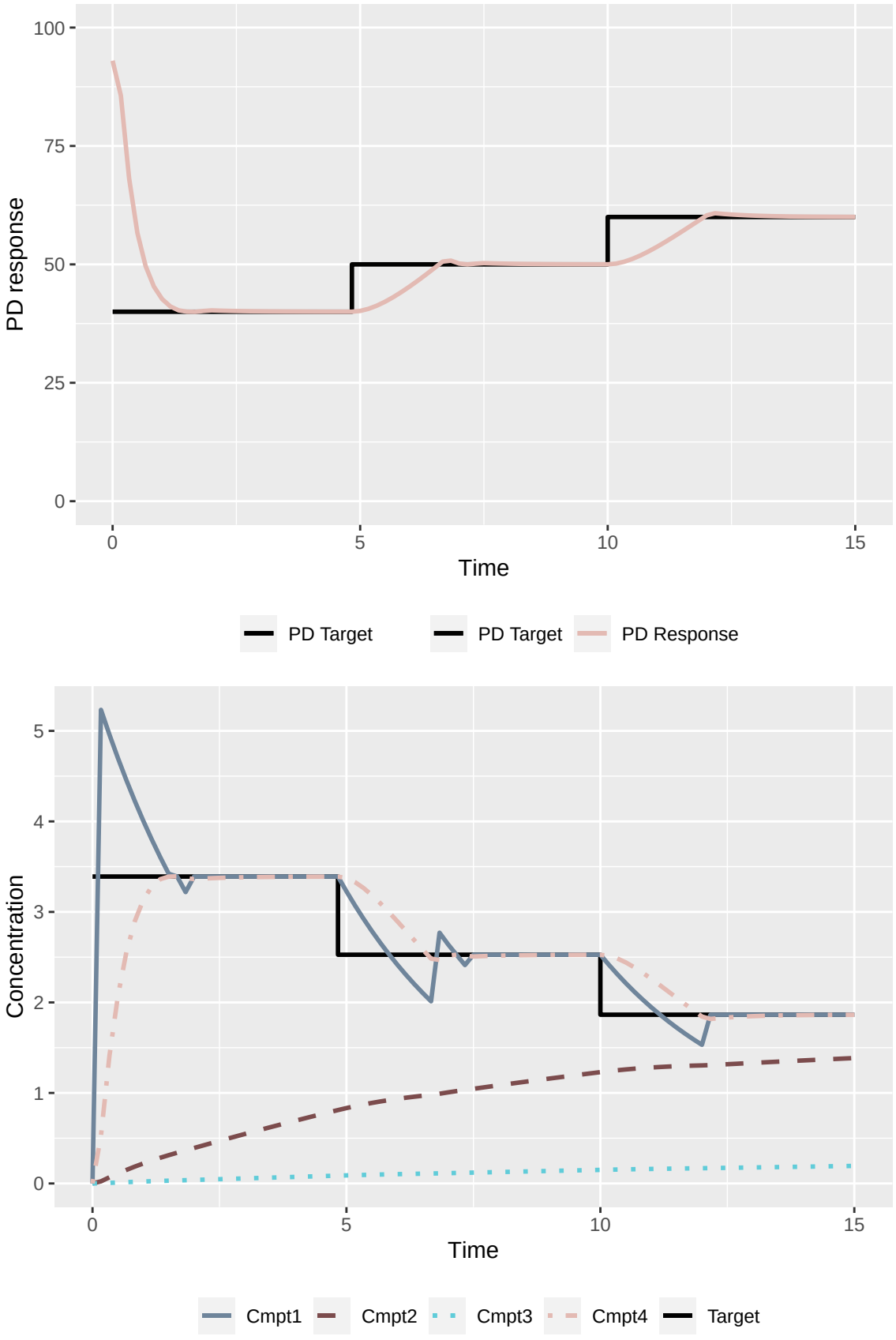

Figure 3.2: Predicted patient BIS responses to infusion schedule defined by 'tci_pd' function.

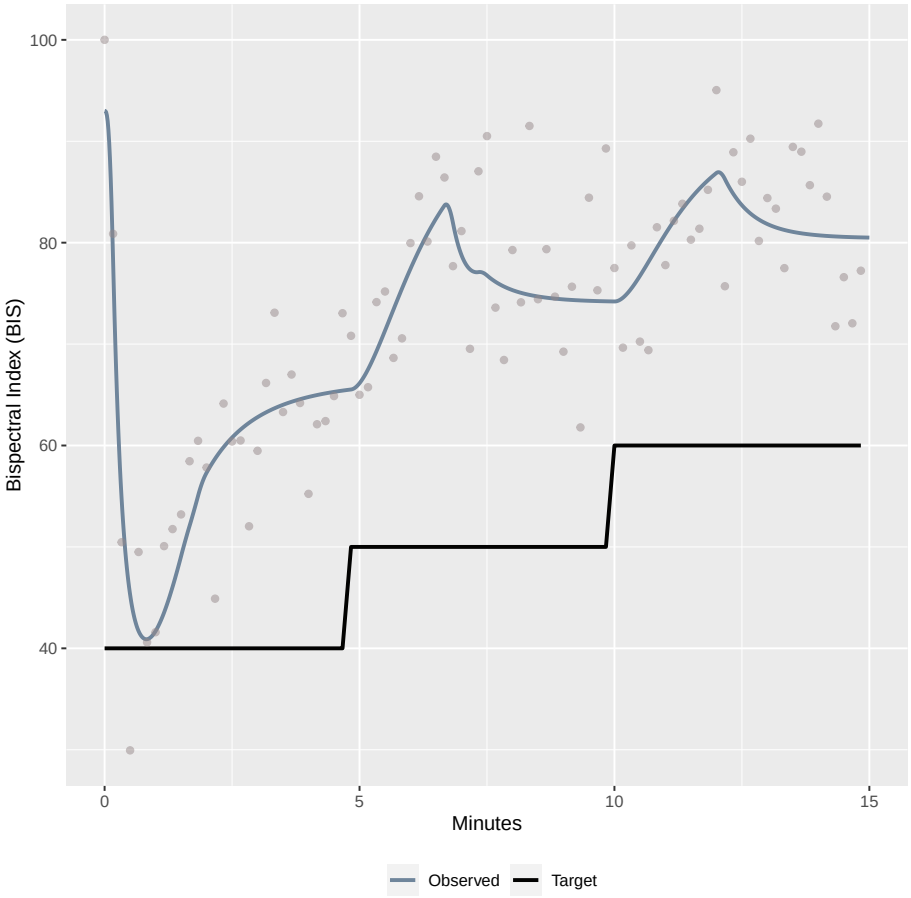

Figure 3.3: Simulated patient data under PK-PD model misspecification.

```
## [3,] 0.333 7.114 0.122 0.008 3.152  28.596 54.771  50.462
```

The output of `gen_data` is a list with class "datasim" that can be plotted, as demonstrated in Figure 3.3.

We observe that, due to the PK-PD parameter misspecification, the true BIS values vary substantially from those predicted for the patient at their covariate values. Consequently, the actual level of sedation is less than the target level and the targets would need to be adjusted to achieve the desired result. In practice, a clinician would adjust the target as needed, thereby "closing" what would otherwise be an "open-loop" system.

As discussed in chapter 2, in a closed-loop control system, patient responses are fed back to a controller that adjusts infusion rates. Closed-loop control can be implemented within the `tci` package through Bayesian updates to the patient-specific PK-PD model using the function `bayes_control()`. `bayes_control()` requires four components as arguments: 1) a data frame specifying a set of targets and corresponding times, 2) a data frame specifying update times, 3) a set of prior PK-PD estimates, and 4) a set of true PK-PD values. The prior PK-PD estimates refer to the values predicted for each patient based on their covariate values alone. The true PK-PD values are the parameter values that describe the patients underlying response. In practice, these are never known, but can be assigned

to evaluate the impact of model misspecification with simulations.

Using the example patient, the response targets and update times are easily set up.

```
targets <- data.frame(time = c(0,5,10,15), target = c(40,50,60,60))
update_times <- data.frame(time = seq(1,15,1),
                           full_data = rep(TRUE,15))
```

The "full_data" argument in the "update_times" data frame specifies whether or not all of the data should be used to update the PK-PD parameters at each time. If set to FALSE, only the data gathered since the prior update time will be used and the prior point estimates and variance covariance will be replaced with the current estimates. Use of only the most recent data increases the computational speed of the closed-loop controller, but may cause a loss of accuracy. A compromise between the two would be to use full updates early on to ensure that an accurate patient-specific model is identified, followed by partial updates at later times, when more data have been collected and the computational cost of full updates is higher.

The prior distribution of PK-PD parameter values describes the variability between patients due to unobserved characteristics. That is, it characterizes how much parameters vary among patients with identical covariate values. The Eleveld PK-PD model assumes that all parameters that vary within the population are log-normally distributed. Consequently, we formulate a multivariate normal prior distribution centered at the logged parameter point estimates for the example patient. The Eleveld model also provides variance estimates for the inter-individual random effect terms, which are assumed to be uncorrelated. Despite this, correlation between PK-PD parameters is nonetheless induced due to common terms in the PK parameter calculations. For example, patient parameters Q_2 and V_2 are correlated, as are Q_3 and V_3, because V_2 and V_3 randomly vary within the population and are included within the fixed-effects portions of Q_2 and Q_3, respectively. Consequently, we estimate the variance-covariance matrix for the example patient by drawing 1,000 Monte Carlo "true" PK-PD parameter samples.

```
rand_vars <- c("V1","V2","V3","CL","Q2","Q3","KE0","CE50","SIGMA")
set.seed(1)
theta_samples <- replicate(1e3, unlist(eleveld_poppk(patient_covariates,
                                                     rand = TRUE)[,rand_vars]))
prior_pars <- list(pars_pkpd = theta[1:12], sig = cov(t(log(theta_samples))),
                   pk_ix = 1:7, pd_ix = 8:12, fixed_ix = 9:12,
                   err = theta["sigma"], delay = theta["bis_delay"]/60)
```

Additional components that need to be specified in both the list of prior patient values and the list of true patient values are the parameter indices that correspond to the PK parameters, the PD parameters, the parameters that are fixed within the population and, therefore, are not updated, the true residual error term, and the time delay between effect and measurement, if present.

```
true_pars <- list(pars_pkpd = theta0[1:12], pk_ix = 1:7, pd_ix = 8:12,
                  fixed_ix = 9:12, err = theta0["sigma"],
                  delay = theta0["bis_delay"]/60)
```

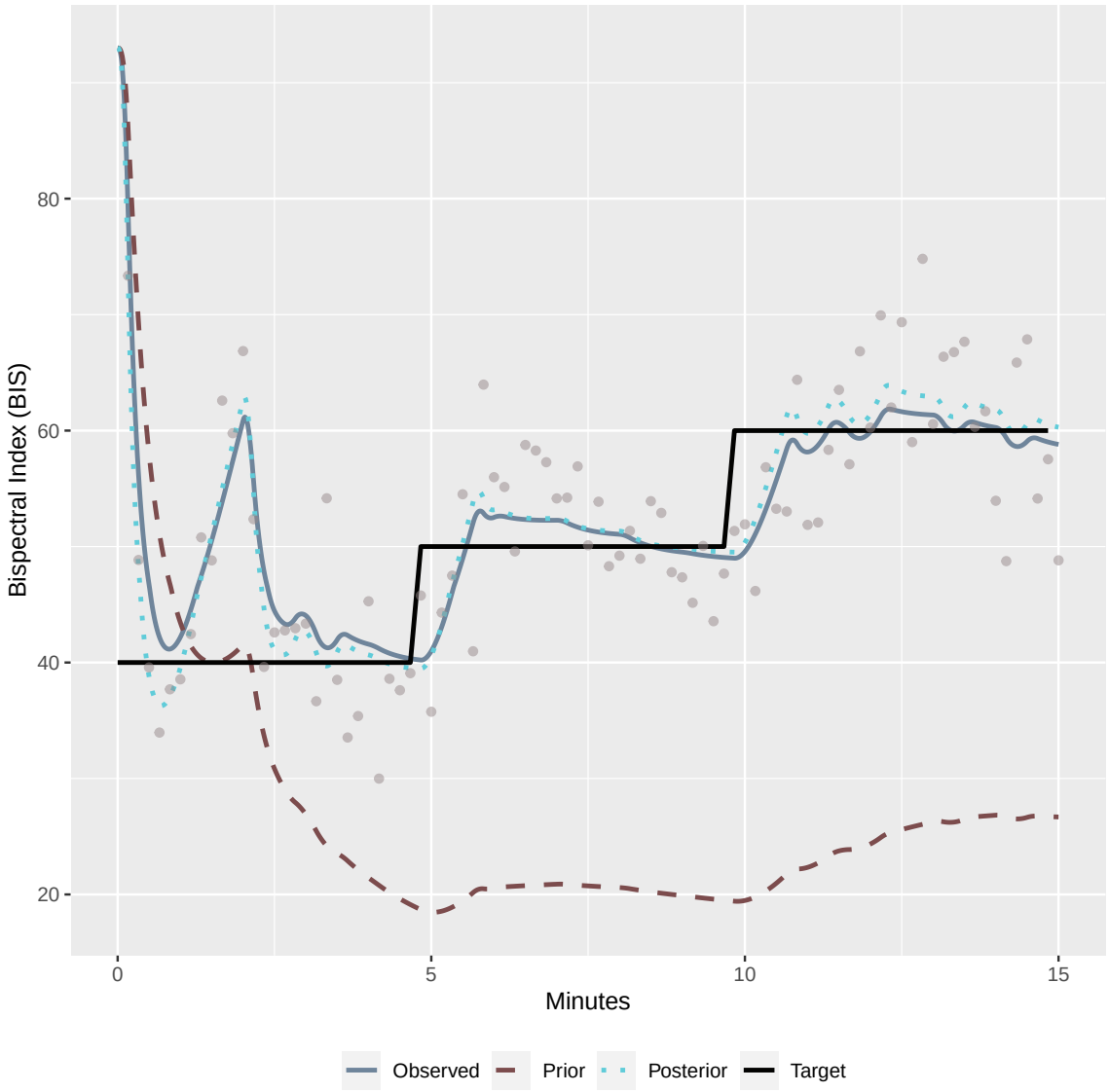

Figure 3.4: Simulated response of example patient under Bayesian closed-loop control.

The `bayes_control` function uses these four components to simulate data under closed-loop control. The simulated results for the example patient are displayed in Figure 3.4. During this simulation, the same targets were used as previously for open-loop control in Figure 3.3

```
bayes_sim <- bayes_control(targets = targets,
                           updates = update_times,
                           prior = prior_pars,
                           true_pars = true_pars)
plot(bayes_sim) + ylab("Bispectral Index (BIS)") + xlab("Minutes")
```

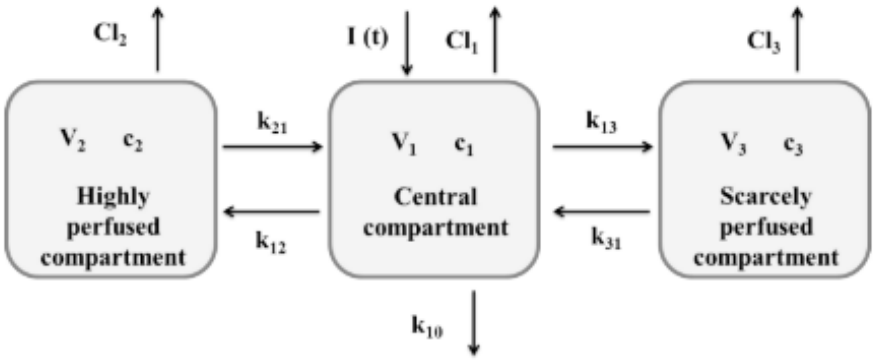

Fig 2. A schematic of the three-compartmental model.

Figure 3.5: Reproduction Figure 2 from Cascone et al. (2013)

3.4 User-defined functions

3.4.1 Custom PK models

If a user needs to simulate responses from a PK model outside of the library, such as one based on a set of ODE, user-defined functions can be specified. To illustrate this functionality, we consider the PK of the drug remifentanil. Remifentanil is an opioid derivative that is often administered intravenously to induce analgesia alongside propofol. Here, we consider the three-compartment PK model proposed by Cascone et al. (2013). The structural model is similar to that of the Eleveld propofol model in that remifentanil is infused into a central compartment, representing the blood supply, and then circulated to two peripheral compartments, representing highly-perfused and scarcely-perfused organs and tissues. Unlike the propofol model, however, remifentanil is removed from all three-compartments with separate clearance rates, rather than just the central one. A diagram of the three-compartment model, reproduced from Cascone et al. (2013), is displayed in figure 3.5. The differential equations describing the remifentanyl model are given in equations 3.7.

$$\begin{aligned} V_1 \cdot \frac{dC_p}{dt} &= -Cl_1 \cdot C_1 + k_{21} \cdot V_2 \cdot C_2 + k_{31} \cdot C_3 \cdot V_3 + -[(k_{12} + k_{13} + k_{10}) \cdot C_1] \cdot V_1 + k_R(t) \\ V_2 \cdot \frac{dC_2}{dt} &= k_{12} \cdot C_1 \cdot V_1 - k_{21} \cdot C_2 \cdot V_2 - Cl_2 \cdot C_2 \\ V_3 \cdot \frac{dC_3}{dt} &= k_{13} \cdot C_1 \cdot V_1 - k_{31} \cdot C_3 \cdot V_3 - Cl_3 \cdot C_3 \end{aligned} \tag{3.7}$$

Since the closed-form solution for this model does not exist in the **tci** package, we can instead program the structural model using the differential equation solver provided by the **mrgsolve** R package. The differential equations are implemented as follows.

```
library(mrgsolve)

form <- '
$PARAM V1=1, V2=2, V3=3, CL1=1.1, CL2=1.2, CL3=1.3, k12=0.4, k21=0.6,
k13=0.6, k31=0.3, k10=0.8
$CMT A1 A2 A3
$ODE
dxdt_A1 = -CL1*A1/V1 + k21*A2 + k31*A3 - (k12+k13+k10)*A1;
dxdt_A2 = k12*A1-k21*A2 - CL2*A2/V2;
dxdt_A3 = k13*A1-k31*A3 - CL3*A3/V3;
'
mod_remif <- mcode("remifentanil", form)
```

Note that the differentials are defined in terms of the amounts of remifentanil infused, rather than concentrations. The `mcode()` function is a **mrgsolve** function used to compile the code.

To implement this model within **tci**, we need to define a wrapper function in terms of specific quantities. The function should be defined to take in as arguments 1) a vector of time points, "tm," 2) a scalar infusion rate, "kR," 3) a vector of named parameters, "pars," 4) initial concentrations (not amounts), "init," and 5) a scalar value indicating the starting time of the infusions. The function should additionally be assigned S3 class "pkmod" for the `predict.pkmod()` and `plot.pkmod()` methods to be compatible. The output of the function should be a vector or matrix of concentrations predicted under a constant infusion rate.

```
pk_remif <- function(tm, kR, pars, init = c(0,0,0), inittm = 0){

  # begin times at 0 and end at last time evaluated
  tm <- tm - inittm
  end_inf <- max(tm)

  # pass parameters as list
  pars <- sapply(pars, as.list)
  vols <- unlist(pars[c("V1","V2","V3")])
  A0 <- init*vols # initial amounts
  names(A0) <- c("A1","A2","A3") # names required by mrgsolve

  # update parameters and initial values (as amounts)
  mod_remif <- update(mod_remif, param = pars, init = A0)

  # dosing regimen - mrgsolve function in terms of amount infused
  event <- ev(amt =  kR*end_inf, time = 0, tinf = end_inf)

  # simulate responses (skip tm=0 unless specified)
  dat <- mrgsim_q(x = mod_remif, # pk model
                  data = event, # dosing event
```

```
                    stime = tm) # evaluation times

  # skip tm=0 unless specified in tm
  dat <- dat@data[-1,]

  # return concentrations with compartments in rows and times in columns
  cons <- t(dat[,c("A1","A2","A3")]) / vols

  rownames(cons) <- colnames(cons) <- NULL
  return(cons)
}
class(pk_remif) <- "pkmod"
```

We can now evaluate the remifentanil PK model as we would any of the internal PK functions. The optimized parameter values identified by Cascone et al. (2013), which we will use as an example, are reproduced in Table 3.2.

Table 3.2: Reproduction of Table 1 from Cascone et al. (2013): Values and dimensions of the three-compartmental model parameters.

Parameter	Optimized value	Parameter	Optimized value
V_1	7.88 L	k_{10}	0.172/min
V_2	23.9 L	k_{12}	0.373/min
V_3	13.8 L	k_{21}	0.103/min
CL_1	2.08 L/min	k_{13}	0.0367/min
CL_2	0.828 L/min	k_{31}	0.0124/min
CL_3	0.0784 L/min		

We observe in Table 3.2 that clearance parameters are given in terms of L/minute. Consequently, dosage amounts measured in μg will result in infusion rates in terms of μg/min and concentrations of μg/L or, equivalently, ng/mL.

As in Cascone et al. (2013), we consider our example patient's predicted response when administered a bolus dose (rapid infusion over 1 minute) of 1μg/kg and an constant infusion of 1μg/kg/min for 20 min for our example patient, who weights 60 kg.

```
pars_remif <- c(V1 = 7.88, V2=23.9, V3=13.8, CL1=2.08, CL2=0.828, CL3=0.0784,
                k10=0.172, k12=0.373, k21=0.103, k13=0.0367, k31=0.0124)
dtm = 1
dose = 1
inf <- cbind(begin = c(0,20), end = c(20,200),
             infrt = c(dose*60,0))
bolus <- cbind(begin = c(0,dtm), end = c(dtm,200),
             infrt = c(dose*60/dtm,0))
pb <- plot(pk_remif, pars = pars_remif, inf = bolus, title = "Bolus") +
   scale_y_log10() + ylab("Concentration (ng/mL)") + xlab("Minutes")
pi <- plot(pk_remif, pars = pars_remif, inf = inf, title = "Infusion") +
```

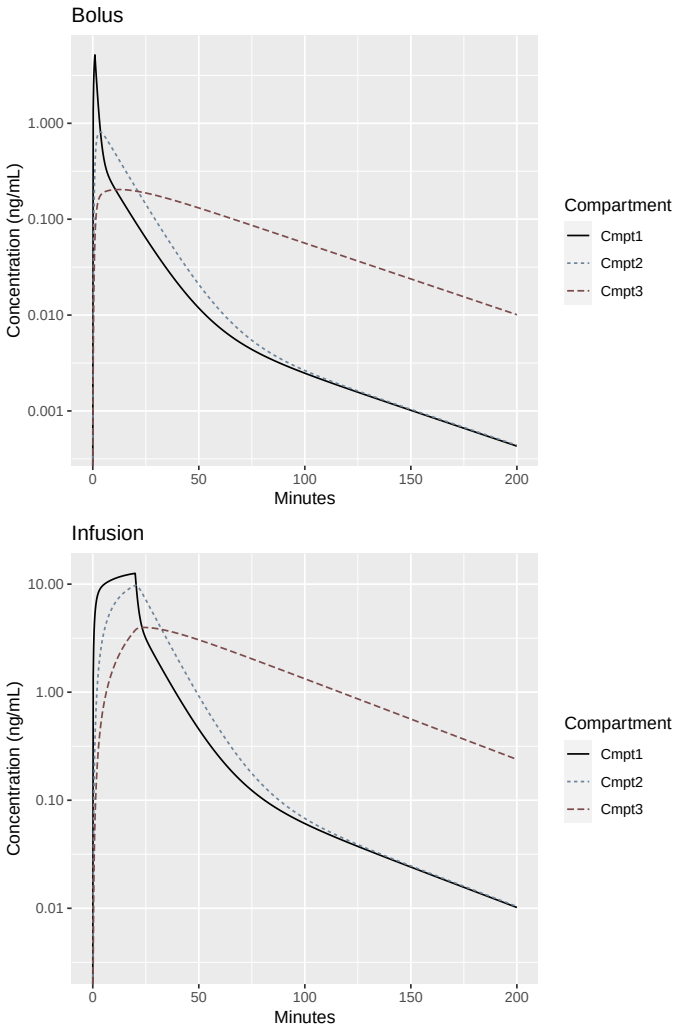

Figure 3.6: Evaluation of user-defined remifentanil model.

```
   scale_y_log10() + ylab("Concentration (ng/mL)") + xlab("Minutes")
grid.arrange(pb,pi,nrow = 2)
```

3.4.2 Custom TCI algorithms

Though the Jacobs and Shafer-Gregg TCI algorithms are commonly used for plasma and effect-site targeting, respectively, a user may wish to specify an alternative algorithm. The structure of the `tci` package allows users to do this while retaining use of the package's S3 generic functions. We illustrate this process using an example effect-site algorithm that limits the maximum overshoot tolerated within the central compartment. The motivation for such an algorithm is similar to that of an algorithm proposed by Van Poucke et al. (2004), which sets a maximum percentage overshoot for the plasma concentration. As Van Poucke et al. (2004) notes, such an algorithm may be useful in cases where there there may exist a site of toxic effect that equilibrates with the central compartment

faster than the therapeutic effect-site and resulting in toxicity to the patient. Rather than limiting the percentage overshoot, here we implement an algorithm that limits the absolute concentration overshoot.

3.4.2.1 Example effect-site algorithm

In this example algorithm the user specifies a permissible amount of overshoot in the central compartment, `lim_amt`, beyond the nominal target. At each step, the example TCI algorithm defines a maximum plasma concentration to equal the target effect-site concentration plus the permissible overshoot: `Cp_max = Cet + lim_amt`. It then calculates the infusion required to reach or maintain `Cp_max` over the subsequent ten seconds, `pinf`. It then calculates the maximum effect-site concentration if `pinf` is given, `Ce_max`. If `Ce_max` is less than the target concentration, `pinf` can be administered without overshoot. If `Ce_max` is greater than the target concentration, then targeting the effect-site directly will result in a maximum plasma concentration less than `Cp_max` and the effect-site targeting algorithm is applied.

To implement the algorithm, we create a function that takes in a single target concentration as its first argument, a PK model with class `pkmod` as its second, and the duration of the infusion administered, `dtm`, as its third.

```
tci_plasma_lim <- function(Cet, pkmod, pars, init = NULL, dtm = 1/6,
                           lim_amt = 0.5,ecmpt = NULL, tmax_search = 20,
                           cetol = 0.05, cptol = 0.1, maxrt = 1200){

  if(is.null(init))
    init <- eval(formals(pkmod)$init)
  if(is.null(ecmpt))
    ecmpt <- length(init)

  ecmpt_name <- paste0("c", ecmpt)

  # if effect-site concentration is close to target,
  # switch to plasma targeting
  if((Cet - init[ecmpt]) / Cet <  cetol &
     (Cet - init[1])/Cet <= cptol)
    return(tci_plasma(Cet, pkmod = pkmod, dtm = dtm, maxrt = maxrt,
                      init = init, pars = pars))

  Cp_max <- Cet + lim_amt

  # infusion required to reach Cp_max
  pinf <- tci_plasma(Cpt = Cp_max, pkmod = pkmod, dtm = dtm, maxrt = maxrt,
                      init = init, pars = pars)

  # Administer dtm-minute infusion
  unit_inf <- create_intvl(
```

```
    data.frame(time = c(dtm, tmax_search),
               infrt = c(pinf, 0))
    )

  # Calculate maximum effect-site concentration
  CeP <- function(tm) predict(pkmod, inf = unit_inf, pars = pars,
                              init = init, tms = tm)[,ecmpt_name]
  Ce_max <- optimize(CeP, c(0,20), maximum = TRUE)$objective


  # if max Ce < Cet administer infusion to reach maximum target
  if(Ce_max <= Cet + cetol*Cet)
    infrt <- pinf
  else
    infrt <- tci_effect(Cet, pkmod, dtm, ecmpt, init = init, pars = pars)

  return(infrt)
}
```

As with the plasma- and effect-site targeting TCI algorithms, `tci_plasma_lim()` returns a single infusion rate when a single target is provided. Applying the new algorithm to the example patient using the Eleveld propofol model, we target an effect-site concentration of 2 mg/L with a maximum overshoot in the central compartment of 0.25 mg/L.

```
tci_plasma_lim(Cet = 2, pkmod = pkmod3cptm, pars = theta, lim_amt = 0.25)
```

```
## [1] 124.7325
```

Once the base TCI algorithm has been created, it can be passed to `tci()` through the "tci_custom" argument to be iteratively applied to a series of targets.

```
tci_custom_alg <- tci(Ct = c(1,2,2.5,2.5), tms = c(0,1,6,11),
                      pkmod = pkmod3cptm, pars = theta,
                      tci_custom = tci_plasma_lim, lim_amt = 0.25)
plot(tci_custom_alg, title = "Plasma-limiting effect-site TCI algorithm")
```

3.5 Summary

In this chapter we have introduced the **tci** package for implementing target-controlled infusion algorithms for compartmental PK and PK-PD models. Interest in TCI systems has grown considerably in the last two decades as TCI has become a "mature" technology.(Absalom et al., 2016) Extensions to closed-loop control systems remain largely experimental, but have demonstrated significant promise in their ability to maintain targets with less over- and under-shoot and lower doses than manual control.(Loeb and Cannesson, 2017) To date, however, no software exists for the R

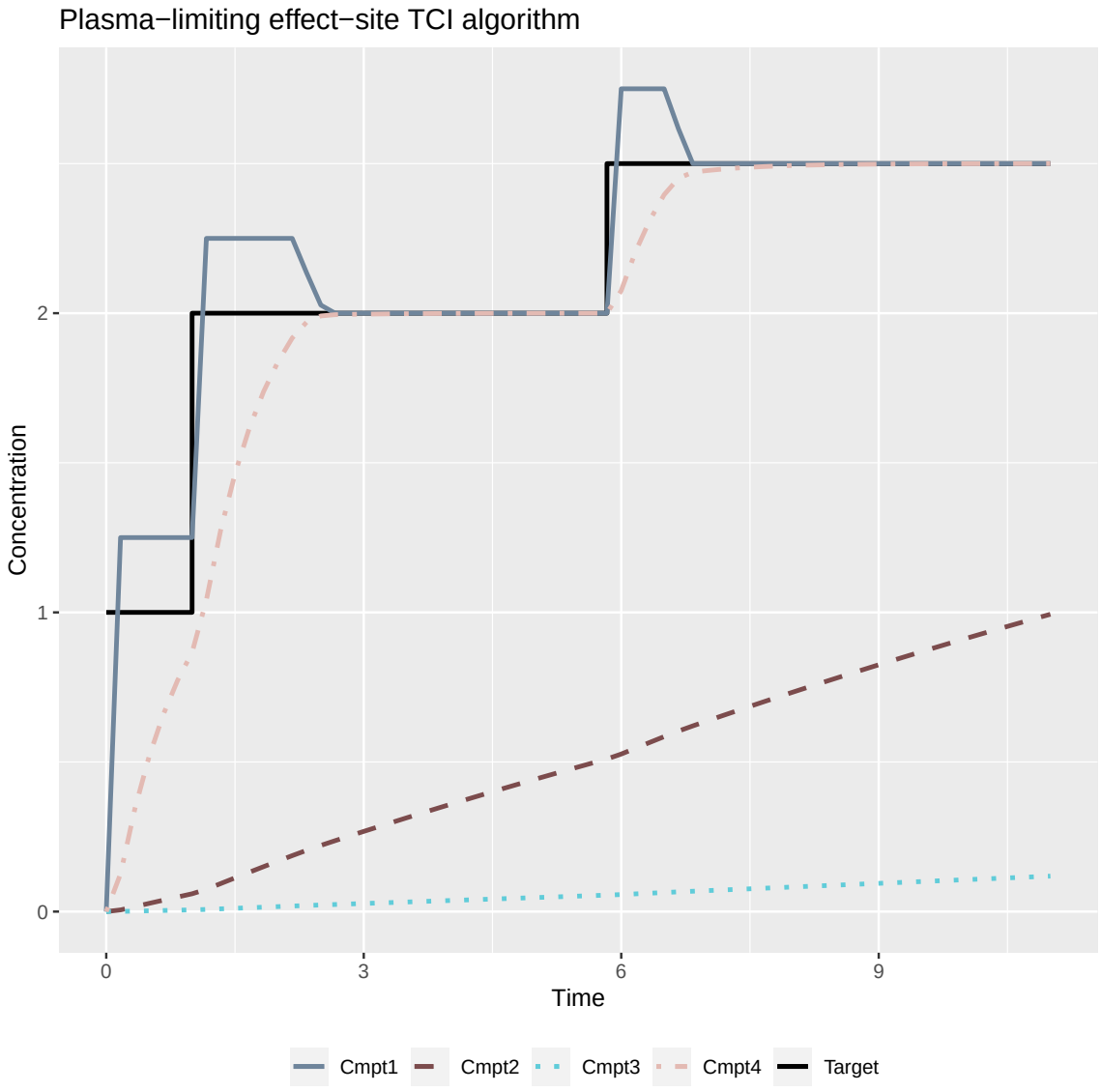

Figure 3.7: Evaluation of user-defined TCI effect-site algorithm.

programming language to implement TCI control, while existing software is either proprietary, less accessible, or not easily modifiable by interested researchers. The goal of this package is to provide a simple and easy to use, but also highly customizable, set of functions that are accessible to any researcher familiar with R and the basic principles of TCI. Future additions to the package are expected to include a wider array of structural and population PK models within the package library, C++ coding of PK models via the `Rcpp` library for faster computation, and extended simulation capabilities for closed-loop control algorithms.

Chapter 4

Data squashing with missing values

4.1 Introduction

4.1.1 Background

In 1999, DuMouchel et al. (1999) introduced the method of "data squashing," by which a large data set with N independent observations is approximated by a smaller, "squashed," data set. The squashed data set has the same variables as the original but has $M \ll N$ rows of pseudo-data values and corresponding frequency weights. The goal of squashing is to identify pseudo-data points and weights so that the statistical information of the original data set is preserved in the squashed sample. Consequently, statistical models that are fit to the squashed data set will have nearly identical results as when fit to the original data set.

Through this process, squashing aims to scale-down the original data set such that conventional statistical and machine learning (ML) methods can be used where they may have otherwise been prohibitively computationally expensive if applied to the original data set. A secondary benefit of the squashing procedure is that the original data set cannot be reclaimed from the squashed data alone, making it a "lossy" procedure. This may facilitate data sharing between researchers where it otherwise would have been impossible due to proprietary or identifiable information.

4.1.2 Prior work

At the time of publication, DuMouchel et al. (1999) predicted that data squashing had "the potential for revolutionizing the field." Over twenty years later, however, very little has been done to implement or extend the original methodology. In the years immediately following its introduction, some efforts were made to extend squashing, develop similar approaches, or apply squashing to machine learning tasks.

In 2003, DuMouchel and Agarwal (2003) introduced an alternate squashing method that was intended to permit squashing of higher-dimensional data sets and those with more categorical levels than the original method. This method applies a "fractional factorial sampling" method whereby categorical variables are binarized before the data are iteratively partitioned into regions based on principal components. From each region, data points are sampled and reweighted to approximate the moments of the original data set. The authors, however, found this method to have disappointing performance relative to the original.

Owen (2003) proposed a method of "empirical likelihood squashing," by which a simple random sample is reweighted to match selected global moments. Like DuMouchel and Agarwal (2003), however, this method had worse performance than the original squashing method and had diminishing utility relative to random sampling as the size of the original data set increased.

In 2002, Madigan et al. (2002) consider an extension to data squashing called "likelihood-based data squashing" (LDS). In LDS, a set of pseudo data points is similarly constructed, rather than sampled, with the goal of approximating a specific likelihood function and prior distribution. This approach outperforms the original squashing method within some of the examples considered but comes at the cost that the user must prespecify the model that is to be fit.

Around the same period, other authors make use of squashing methods for machine learning applications. Pavlov et al. (2000) use LDS in order to train support vector machines, while Choki and Suzuki (2002) propose an iterative squashing-boosting algorithm that uses data squashing to compress the reweighted data set returned by the boosting algorithm at each step. Following 2003, however, there seems to be no work aimed at extending or applying data squashing. Occasional mentions are made only in passing by Xi (2009), who notes that no asymptotic guarantees have yet been identified for squashing, and Ng (2017), who raises the criticisms that 1) the squashed data points do not carry any specific interpretation since they don't correspond to a specific point in the original data set, and 2) that squashing (and parametric modeling) is not practical if the data set is too large. Similarities also exist between data squashing and "instance selection" methods within the data science and machine learning literature. Instance selection aims to remove noisy or redundant values from a training set such that a model achieves a similar accuracy on the subset of data as on the original data set.(Olvera-López et al., 2010) This shares more in common, however, with the work derived from data squashing than it does with the original method. Like the approaches of DuMouchel and Agarwal (2003) and Owen (2003), specific values from the data are selected rather than constructing new pseudo instances. Also, as in the LDS approach proposed by Madigan et al. (2002), instance selection methods focus on classification of a pre-specified variable, rather than the approximation of an arbitrary likelihood function.

4.1.3 Contributions in this chapter

Given the apparent promise of data squashing, it is worth asking why it not only has not "revolutionized" statistical analysis as predicted but remains largely unknown and unused today. We believe there are several reasons why this may be the case. First, and most practically, to this day there does not exist any published software that will perform data squashing. Though the focus of this chapter is on the extension of the squashing methodology, an ancillary benefit is the construction of open-source software in R that will make it easier for other researchers to further refine data squashing. Methodologically, the primary limitations of data squashing concern the types of data sets that are amenable to the procedure. As we will discuss later on, data sets with a large number of categorical variables can be squashed but may not derive many benefits from the process. This is the primary limitation that DuMouchel and Agarwal (2003) sought to overcome in their earlier work, though they were largely unsuccessful in doing so. Data sets in which the number of columns exceeds the number of rows are also not good candidates for data squashing. For these cases, however, the extensive literature on variable selection can offer guidance.

A final limitation that, we believe, has hindered the implementation and development of data squashing is its inability to handle data sets with missing values. Missing values are ubiquitous in real-world data sets and cannot be handled by many popular machine learning algorithms. These include support vector machines, lasso and elastic-net regression, and neural networks.(Kuhn and Johnson, 2019) For sufficiently large data sets with missing values, more sophisticated methods that adequately account for the variability in missing values, such as multiple imputation, may be discarded in favor of simpler methods, such as mean or median imputation or outright deletion of missing instances.

In this chapter, we describe two different approaches to handling missing data within the original squashing framework proposed by DuMouchel et al. (1999). We begin by describing data squashing

and evaluating the performance characteristics of squashing as an estimation procedure. We then show how the original methodology can be extended to allow for data sets with missing values by propagating missingness through to the squashed data set. Using this process of propagation- or p-squashing, the squashed data set preserves that statistical information of the original data set such that maximum likelihood-based methods for handling missing data can be equivalently used on the parent and squashed data sets. We then evaluate a second extension wherein the squashed data set is constructed to preserve the expectation of the log-likelihood (e-squashing) in the original data, thereby creating a squashed data set without any missing values.

4.2 Data squashing

4.2.1 Theory

Assume we have a large data set with N independent rows that are drawn from a probability model, f, with parameters θ and C categorical variables and Q quantitative variables. Let i index the rows of the data set and $A_1, ..., A_C$ and $X_1, ..., X_Q$ represent the categorical and quantitative variables, respectively.

The log-likelihood for θ is defined as

$$\log(L(\theta|X, A)) = \sum_{i=1}^{N} \log(f(X_{i1}, ..., X_{iQ}, A_{i1}, ..., A_{iC}|\theta)) \tag{4.1}$$

The goal of the squashing procedure is to construct a set of pseudo data points and corresponding frequency weights that use the same set of variables, but with $M \ll N$ rows. Let B_c and Y_j represent the categorical and numeric variables in the squashed data set, respectively, for $j = 1, ..., Q$, $c = 1, ..., C$, and let w_i represent the frequency weights for $i = 1, ..., M$. For the squashed data set to reproduce the log-likelihood in equation 4.1, the two are set to be equal.

$$\sum_{i=1}^{N} \log(f(X_{i1}, ..., X_{iQ}, A_{i1}, ..., A_{iC}|\theta)) = \sum_{i=1}^{M} w_i \log(L(\theta|Y, B)) \tag{4.2}$$

For data squashing to work, the primary assumption is that, within regions defined by fixed values of A, the log-likelihood for θ is sufficiently smooth as to be well-described by a Taylor series expansion with K terms about the point $x = (x_1, ..., x_Q)$. Given this assumption, the log-likelihood within such regions can be written as

$$\log(f(X_{i1}, ..., X_{iQ}|A_{i1} = a_1, ..., A_{iC} = a_c, \theta)) \approx \sum_{k=1}^{K} g_k(a, x, \theta) \prod_{j=1}^{Q} (X_{ij} - x_j)^{p_{kj}} \tag{4.3}$$

The first K coefficients of the Taylor series, $g_k(a, x, \theta)$, $k = 1, ..., K$, depend on the values of the categorical variables in the region, $a = (a_1, ..., a_c)$, as well the expansion point, $x = (x_1, ..., x_Q)$, and the unknown parameters, θ. The product term, $\prod_{j=1}^{Q} (X_j - x_j)^{p_{kj}}$, concisely notes all combinations of products such that the sum of the vector of exponents, $(p_{k1}, ..., p_{kQ})$ is less than a set degree of approximation: $\sum_j p_{kj} \leq d$. For example, letting $Q = d = 2$, the set of power vectors is $\mathbf{p} = \{(0, 0), (1, 0), (0, 1), (1, 1), (2, 0), (0, 2)\}$ such that the six terms of the Taylor approximation are

$$\sum_{k=0}^{5} g_k(a,x,\theta)\prod_{j=1}^{2}(X_{ij}-x_j)^{p_{kj}} = g_0 + g_1(X_{i1}-x_1) + g_2(X_{i2}-x_2) + g_3(X_{i1}-x_1)(X_{i2}-x_2) + g_4(X_{i1}-x_1)^2 + g_5(X_{i2}-x_2)^2 \tag{4.4}$$

If we partition the original data into R regions with N_r rows in the rth region for $r = 1, ..., R$.

$$\sum_{i=1}^{N}\log(f(X,A|\theta)) \approx \sum_{r=1}^{R}\sum_{i=1}^{N_r}\sum_{k=1}^{K} g_k(a(r),x(r),\theta)\prod_{j=1}^{Q}(X_{ij}-x_j(r))^{p_{kj}} \tag{4.5}$$

The notation $a = a(r), x = x(r)$ indicates that the categorical values and Taylor expansion points vary between, but are constant within, each region indexed by r. For each region, we allow the squashed sample to have M_r pseudo data points and set the categorical variable values equal to those of the original data set, such that $A_c = B_c$ for $c = 1, ..., C$. Applying this approximation to both sides of equation 4.2 and swapping the order of summation over K and N_r, we have

$$\sum_{k=1}^{K} g_k(a(r),x(r),\theta)\sum_{i=1}^{N_r}\prod_{j=1}^{Q}(X_{ij}-x_j(r))^{p_{kj}} = \sum_{k=1}^{K} g_k(a(r),x(r),\theta)\sum_{i=1}^{M_r} w_i\prod_{j=1}^{Q}(Y_{ij}-x_j(r))^{p_{kj}} \tag{4.6}$$

The order of summation is swapped to make it more apparent that, by setting $x(r)$ equal the means of X within region r, each side of the equation consists of terms that are proportional to the weighted sample moments. For example, letting $Q = d = 2$, as in equation 4.4, when $k = 3$, $p = (1, 1)$ and the corresponding Taylor series term is $g_3\sum_{i=1}^{N_r}(X_{i1}-\bar{x}_1)(X_{i2}-\bar{x}_2) = g_3(N_r-1)\hat{\sigma}_{12}$, where σ_{12}^2 is the empirical covariance between X_1 and X_2.

Since the Taylor series terms depend upon the values of $a(r), x(r), \theta$, by keeping these constant within each region, the squashed data set that will reproduce the likelihood function of interest is the one in which the sample moments from the original data set are matched to the weighted sample moments of the squashed data set, centered around $x(r)$.

$$\sum_{i=1}^{N_r}\prod_{j=1}^{Q}(X_{ij}-x_j(r))^{p_{kj}} = \sum_{i=1}^{M_r} w_i\prod_{j=1}^{Q}(Y_{ij}-x_j(r))^{p_{kj}} \tag{4.7}$$

The goal of squashing is to find values of Y_{ij} and w_i such that this equation holds within each region. Details of how this is done are provided in the next section.

4.2.2 Implementation

To implement data squashing, a number of choices need to be made. Specifically, one must select the following:

1. A clustering mechanism (if any) for numeric data within categorical levels. Increased numeric clustering will permit more localized squashing, but will decrease the degree of data reduction.

2. The set of moments to match between original and squashed data sets. The set of moments is determined by the degree of the Taylor series approximation within each region. More moments

will result in a better approximation of the likelihood, but will require more squashed data points to do so.

3. The number of pseudo-data points within each region. The number of data points will determine how precisely the original data moments can be reproduced.

Each of these choices may affect the computational speed of the squashing process, the number of points in the squashed data set, and the accuracy to which models fit on the squashed data set reproduce results from the original.

4.2.2.1 Clustering mechanism

Within each region defined by categorical levels, numeric variables can, but do not necessarily need to be, clustered. Any clustering method will generally result in a lower amount of compression than if none is used, but will be better able to faithfully replicate nonlinear relationships within the data. DuMouchel et al. (1999) propose two methods of clustering the data: hyper-rectangles or data spheres. In the hyper-rectangle approach, continuous variables are categorized into three bins at the 25th and 75th percentiles. In the data spheres approach, data are centered and scaled before being split into a pre-defined number of layers based on their distance from the origin. The layers are then further split into pyramids such that, within each grouping, all points will have the same variable with the furthest deviation from the origin.

Within this chapter, we consider only the data spheres approach, as it appeared to perform well within the original simulations and doesn't suffer from the same curse of dimensionality as the hyper-rectangles approach as the number of variables increases.

4.2.2.2 Selection of moments and number of data points

The set of moments matched within each region are the sufficient statistics for the log-likelihood and will determine how precisely it is approximated. Depending on the anticipated complexity of model(s) more or fewer moments may be required. In the simple case of linear regression with linear terms, the set of means, variances, and covariances for each variable are the sufficient statistics. Consequently, matching only the first and second moments would precisely replicate the original data set results (of course, one could save time by just calculating these quantities if these were the only models of interest). DuMouchel et al. (1999) recommend using up to all zero- to fourth-order marginal (e.g. X_i^3) and crossed (e.g. $X_i X_j X_k$) moments, as well as fifth-order marginal moments. In our implementation, we use all of these with the exception of the fifth-order marginal moments.

The precise number of moments matched within each region varies depending on the number pseudo data points in the region. When there are m pseudo data points and Q quantitative variables, there are $df = m(Q+1)$ degrees of freedom in the optimization process (weights included). Consequently, the number of moments, K, is selected such that $K \approx df$, with the lower-order moments included first. The number of pseudo data points created, M_r, is defined by the formula

$$M_r = \max(1, \alpha \log(N_r)) \tag{4.8}$$

The parameter α takes on values $\alpha \geq 0$. As α increases, the number of pseudo data points also increases, allowing for a more precise fit between parent and squashed data sets, but at the cost of a lower amount of reduction in the sample size.

4.2.2.3 Identification of squashed data points and weights

Squashed data points and weights are identified within each region by searching for values that minimize the (weighted) squared distance between the original data moments and the weighted moments in the squashed data set.

$$S(Y, w) = \sum_{k=1}^{K} u_k \left(z_k - \sum_{i=1}^{M_r} w_i \prod_{j=1}^{Q} Y_{ij}^{p_{kj}} \right)^2 \tag{4.9}$$

In the above equation, z_k is the kth moment from the original data set and u_k is a positive constant that indicates the relative importance of the moment. Since lower-order moments (e.g., means and variances) are typically more important than higher-order moments (e.g., 3rd order coskew terms), lower-order moments receive larger values of u_k than higher-order moments.

Rather than leaving it to a search algorithm to find squashed values and weights that minimize the distance between lower order moments, the squashed data set can be initialized such that it matches the original data set on moments of order 0, 1, and pure moments of order 2 (i.e., sample size, means, and variances, respectively). Specifically, we take the following steps:

1. If weights are not present in the original data set, assign each observation a weight of 1.
2. Center and scale the columns of the original data set to have means of 0 and variances of 1.
3. Initialize the squashed sample by sampling values and weights from each column of the original data.
4. Scale squashed weights so that the sum of the squashed weights equals the sum of the original data weights.
5. Center and scale each column of the squashed data set by subtracting the weighted mean and dividing by the weighted standard deviation.

Once these steps are taken, the sample size, means, and variances are assigned high values, arbitrarily set to $\mu_k = 1000$, to dissuade the optimizer from taking a step that will increase the distance between these moments. All other moments receive values of μ_k so that they sum to 1 within each order. For example, for p variables there are $p(p+1)/2 - p$ covariance terms, each of which receives a value $\mu_k = 1/(p(p+1)/2 - p)$. Minimization of Equation 4.9 can be performed by any optimization routine; however, as noted by DuMouchel et al. (1999), each term is a polynomial function of the unknowns, allowing for calculation of the numeric derivatives required for a Newton-Raphson algorithm. In our own implementation, we do this with the R function `nlm` and the numeric gradient.

4.2.3 Simulation performance

In DuMouchel et al. (1999) the performance of data squashing is evaluated by comparing how well regression coefficients from model fit to the squashed data set match those of the same model fit to the full data set. The coefficients from the full data set are taken to be the "true" parameter values and the relative error is calculated as $(\hat{\beta}_{\text{squashed}} - \hat{\beta}_{\text{true}})/SE(\hat{\beta}_{\text{true}})$. While it is important to know this, it is also worth investigating how well data squashing functions as an estimation procedure when the true parameter values are known. To assess this, we consider the following scenario. Two numeric predictors are independently generated from standard normal distributions

and one categorical variable is created to have three categories proportioned approximately 60%, 25%, and 15%, respectively. A numeric outcome is generated based on the model

$$\begin{aligned} Y &= 5 + 2X_1 - X_1^2 + 3A_2 + 5A_3 + 3X_2 + 5X_1A_2 - X_1A_3 - 2X_1^2A_2 + X_1^2A_3 + \epsilon \\ \epsilon &\sim N(0, 4) \end{aligned} \tag{4.10}$$

We then generate 2,000 data sets with $N = 1,000$ observations and squash each of them at tuning parameter values $\alpha = (0.5, 1, 1.5, 2)$ and numeric clustering using the data sphere method with two and three layers. The correctly specified model is fit to both original and squashed data sets, for which we calculate the bias of the coefficients, the root mean squared error (RMSE), and the coverage probability. Results of these simulations are shown in figure 4.1.

At lower levels of the tuning parameter α (i.e., $\alpha = 0.5, 1$), we observe that the squashing procedure results in substantial bias in the parameter estimates, high RMSE, and poor coverage probabilities. As α increases to 1.5 and 2, the squashing performance improves substantially, with estimates at $\alpha = 2$ showing negligible bias and accurate coverage probabilities. At lower values of α, there noticeable improvement in the estimates when three data sphere layers are used rather than two; however, this improvement disappears at $\alpha = 1.5, 2$. This is consistent with expectations. At low levels of α, relatively few pseudo-data points are used to approximate the moments within each region. Consequently, only the lower-order moments will be matched, resulting in an inability to replicate nonlinear relationships. As the number of regions increases (i.e., through 3 layers instead of 2), the nonlinear relationships are better approximated by the piece-wise linear estimates within each region. At higher values of α, more pseudo-data points are used, allowing for a better approximation of nonlinear relationships and reducing the need for additional layers.

The cost of this improvement in performance is a larger squashed sample size. At $\alpha = 0.5$ and two layers, an average of only $M = 38$ pseudo points are used to squash the data set of $N = 1,000$ points. By contrast, when three layers are used at $\alpha = 2$, an average of $M = 350$ pseudo points are used. As noted, the addition of a third data sphere layer is largely unnecessary to preserve the relationships within the data once $\alpha \geq 1.5$. Further, since both the performance and the increase in squashed sample size scales better with α than with the number of layers, squashing performs better in our simulations at $\alpha = 1.5$ and two layers than at $\alpha = 1$ with three layers despite having an average of $M = 140$ points in the former and $M = 170$ in the latter. Finally, it is worth noting that while the reduction factor (N/M) is relatively low for these data sets (e.g. reduction: 1000/140=7.14 at $\alpha = 1.5$, layers = 2), this would be expected to improve with larger sample sizes. Since M_r increases at a rate of $\alpha log_2 N_r$ within each original data set region with size N_r, as the total sample size increases relative to the number of regions, the reduction factor will also increase. With these results in mind, we now consider approaches to handling missing values within the squashing methodology.

4.3 Squashing with missing values

We propose two methods of accommodating missing values in the squashing process. In the first, missing values present in the original data set are propagated onward to the squashed data set. We will refer to this method alternately as "propagation squashing" or "p-squashing." Though this

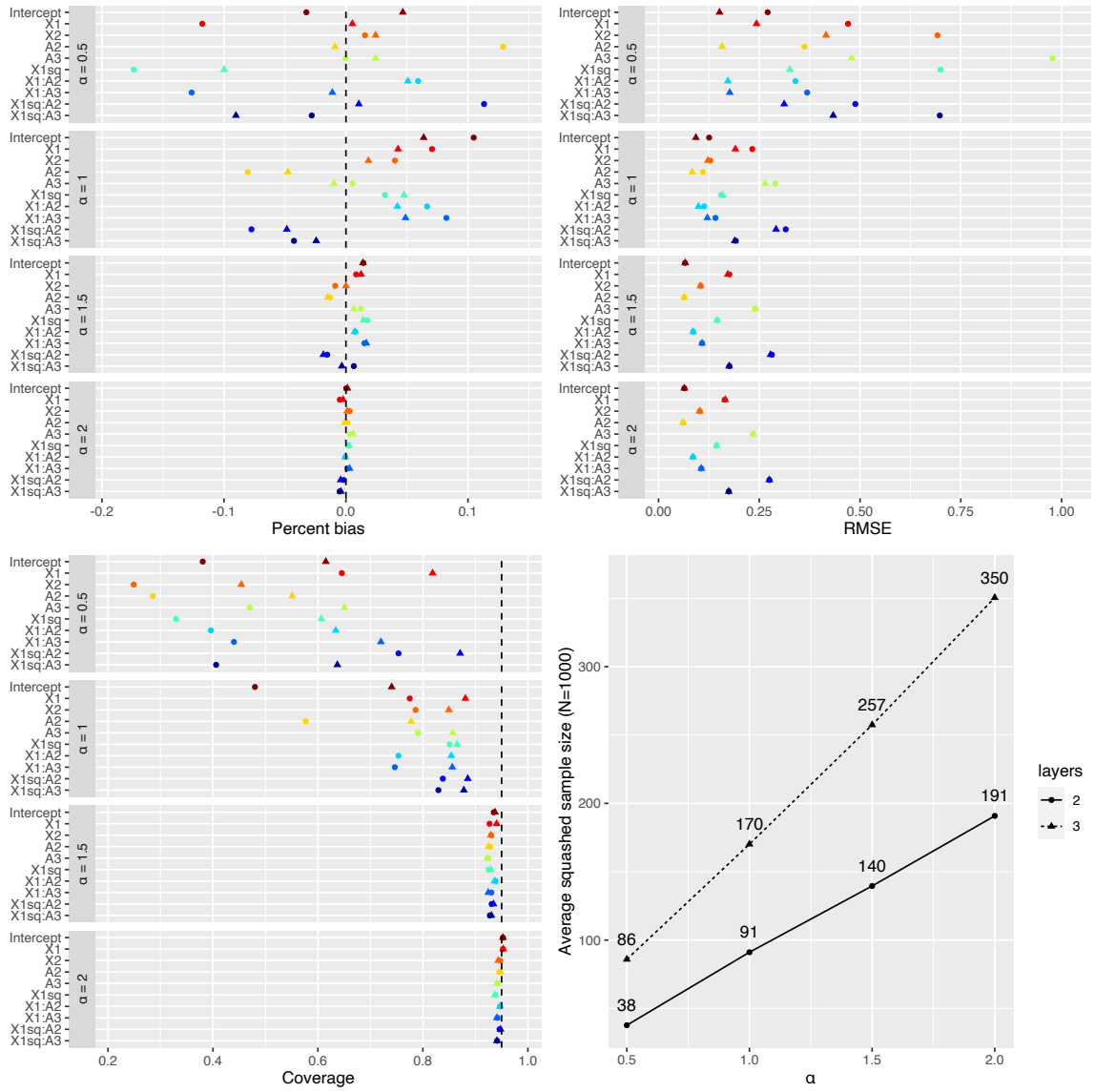

Figure 4.1: Performance of basic squashing method. Results are from 2,000 iterations of data squashing applied to simulated data sets of size N=1,000. As alpha values increase from 0.5 to 2, more pseudo data points are used in the squashed sample allowing for a greater degree of accuracy at the cost of a larger squashed sample size. Either two (circle) or three (triangle) layers are used to cluster numeric values within categorical regions using the data sphere approach.

does not alleviate the problem of missing data, missing data methods such as multiple imputation, maximum likelihood estimation, or Bayesian modeling that were unfeasible due to the size of the original data set may be possible with a squashed data set. Further, we show that p-squashing preserves all of the information required for likelihood-based imputation procedures.

In the second method, rather than squashing by preserving the likelihood function, we preserve the expectation of the log-likelihood function. When models with missing data are fit via maximum likelihood, it is common to use the E-M algorithm as a fitting procedure. Through the E-M algorithm, the replacement of missing observations with their expected values and the maximization of the log-likelihood with respect to the parameter values is iterated until convergence. By squashing the expectation of the log-likelihood, the squashed data set is constructed to resemble a single "E-step" estimate of the model that would result from the E-M procedure. Though this adds a further degree of approximation between the parent and squashed data sets, the result is that the latter has only fully observed variables. We will refer to this method either as "expectation-squashing" or "e-squashing."

4.3.1 Propagation squashing - theory

To show how squashing can be extended to data with missing values, we begin by considering how missing values are handled in maximum likelihood estimation, otherwise known as full-information maximum likelihood (FIML).(Hartley and Hocking, 1971) Without loss of generality, define the set of missingness patterns across categorical and continuous variables as Ξ, such that for each missingness pattern ξ, the first $u = u(\xi)$ numeric and $v = v(\xi)$ categorical variables are completely observed and the remaining variables are missing. Had all variables been observed, the contribution to the likelihood of all observations with missingness pattern ξ would have been

$$L = \prod_{i=1}^{N^{(\xi)}} f(X_{i1}, ..., X_{iQ}, A_{i1}, ..., A_{iC}; \theta) \tag{4.11}$$

where $N^{(\xi)}$ is the total number of observations with missingness pattern ξ. Since missing values are present, however, this likelihood cannot be evaluated. As the name would suggest, FIML aims to use all of the information available in the data in the estimation process. Consequently, for each missingness pattern, a separate marginal likelihood is calculated that is integrated or summed over the distribution of missing values. The contribution to the likelihood for the observations with missingness pattern ξ is

$$\begin{aligned} L^{(\xi)} &= \prod_{i=1}^{N^{(\xi)}} \left[\sum_{A_{v(\xi)+1}} ... \sum_{A_C} \int_{X_{u(\xi)+1}} ... \int_{X_Q} L \; d_{X_{u+1}} ... d_{X_Q} \right] \\ &= \prod_{i=1}^{N^{(\xi)}} f^{(\xi)}(X_{i1}, ..., X_{iu(\xi)}, A_{i1}, ..., A_{iv(\xi)}; \theta) \\ &\equiv \prod_{i=1}^{N^{(\xi)}} f_i^{(\xi)} \end{aligned} \tag{4.12}$$

For all missingness patterns the full FIML likelihood, L^*, is the product of the contributions from each missingness pattern, each of which has its own likelihood function.

$$L^* = \prod_{\xi \in \Xi} \prod_{i=1}^{N^{(\xi)}} f_i^{(\xi)} \tag{4.13}$$

To apply this to data squashing, we need to ensure that missingness patterns are constant within each region. This is easily achieved by creating a categorical variable with levels corresponding to each missingness pattern within all numeric and categorical variables. Since the original squashing methodology separates regions on the basis of categorical levels, the FIML likelihood can be expressed as R regions where the form of the likelihood function is constant within each region, though many regions may have the same missingness pattern.

$$\begin{aligned} \prod_{\xi \in \Xi} \prod_{i=1}^{N^{(\xi)}} f_i^{(\xi)} &= \prod_{r=1}^{R} \prod_{i=1}^{N_r} f^{(\xi)} \left(X_{i1}, ..., X_{iu(\xi)} | A_{i1} = a_1, ..., A_{iv(\xi)} = a_{v(\xi)}, \theta \right) \\ &\equiv \prod_{r=1}^{R} \prod_{i=1}^{N_r} f_i^{\xi(r)} \end{aligned} \tag{4.14}$$

For an arbitrary region r with missingness pattern ξ, the log likelihood can be approximated by a Taylor series about the point $(x_1, ..., x_{u(\xi)})$ using the first $u(\xi)$ numeric variables observed in the region. This can then be set equal to the sum of the weighted mixed moments in the squashed data set as in the original DuMouchel method.

$$\begin{aligned} \log \left(\prod_{r=1}^{R} \prod_{i=1}^{N_r} f_i^{(\xi)} \right) &= \sum_{r=1}^{R} \sum_{i=1}^{N_r} \log \left(f_i^{\xi(r)} \right) \\ &\approx \sum_{r=1}^{R} \sum_{i=1}^{N_r} \sum_{k=1}^{K} g_k \prod_{j=1}^{u(\xi)} (X_{ij} - x_j)^{p_{kj}} \\ &= \sum_{r=1}^{R} \sum_{i=1}^{M_r} w_i \sum_{k=1}^{K} g_k \prod_{j=1}^{u(\xi)} (Y_{ij} - x_j)^{p_{kj}} \end{aligned} \tag{4.15}$$

Since the regions were constructed such that the first $u(\xi)$ numeric variables were fully observed, while the remaining $Q - u(\xi)$ variables were fully missing, the Taylor series approximation is made with respect to only the fully observed numeric variables. Numeric variables that are entirely missing within a region of the original data are also entirely missing within the corresponding region of the squashed data. As with the original method, the values of the categorical variables within the squashed samples are constructed to equal those of the original data set. This remains true when categorical variables are missing observations, with "missing" being treated as a separate category.

These results show that simple modifications of the original squashing procedure can be used to squash data sets with missing values in such a way that the information required for FIML is preserved within the squashed data set. Consequently, using this approach likelihood-based missing data methods can be used equally-well on the squashed data set, where the computation time may be substantially reduced relative to the original data set.

4.3.2 Evaluation of propagation squashing in simulations

To evaluate the performance of the first method of data squashing with missing values, we assign missing values to the same 2,000 data sets used in section 4.2.3 and fit the same model using FIML. Missing values are assigned to each data set at rates of both 20% and 50% using the "mice::ampute" R function. The "ampute" function is a multivariate amputation procedure in which the user specifies a set of missingness patterns, corresponding probabilities for each pattern appearing, and weights that indicate the relationship between each variable and the probability of each missingness pattern occurring. By varying the weights, the user can modify the missingness mechanism between missingness completely at random (MCAR), missingness at random (MAR), and missingness not at random (MNAR). In this chapter, we consider only generating missingness under a MAR mechanism such that missingness is related to the values of other variables observed, but not to the underlying value of the missing observation itself. The missingness patterns and the corresponding weights are displayed in Table 4.1.

Table 4.1: Missingness patterns and associated weights for amputation with mice::ampute. For each pattern, variables with zeroes are missing while ones are observed. For each weight, values indicate the contribution of each variable to the weighted sum score that determines if the corresponding pattern will be observed. Zeroes in the same location in patterns and weights denote that a variable does not contribute to its own probability of missingness (i.e., data are MAR, rather than MNAR).

	X1	X2	A1	A2	A3	Y
Pattern 1	0	1	1	1	1	1
Pattern 2	1	0	1	1	1	1
Pattern 3	1	1	1	1	1	0
Weights 1	0	1	1	2	3	1
Weights 2	1	0	1	2	3	1
Weights 3	1	1	1	2	3	0

Zeroes in patterns one through three under columns $X1$, $X2$, and Y indicate that these variables will be missing separately, but never together. Ones in the patterns in columns $A1$, $A2$, and $A3$ indicate that these variables are always observed. The weight values indicate the relative contribution of each variable to a weighted sum score (WSS) that determines the probability that a row within the data set will have the corresponding missingness pattern. Specifically, the "ampute" function randomly designates sets of rows to have each missingness pattern, then calculates the following weighted sum scores for each observation i in the region. The sum scores for each of the three patterns are

$$\begin{aligned} WSS_1(X1_i) &= X2_i + \mathbf{1}_{Ai1} + 2 \cdot \mathbf{1}_{Ai2} + 3 \cdot \mathbf{1}_{Ai3} + Y_i \\ WSS_2(X2_i) &= X1_i + \mathbf{1}_{Ai1} + 2 \cdot \mathbf{1}_{Ai2} + 3 \cdot \mathbf{1}_{Ai3} + Y \\ WSS_3(Y_i) &= X1_i + X2_i + \mathbf{1}_{Ai1} + 2 \cdot \mathbf{1}_{Ai2} + 3 \cdot \mathbf{1}_{Ai3} \end{aligned} \tag{4.16}$$

The WSS score is then used in a logistic function to determine the probability of missingness, with higher WSS scores resulting in a greater probability that a pattern will be observed.

For our simulations, missingness is assigned such that 20% and 50% of the observations have one of the missingness patterns, all of which are equally likely to be represented. Once missing values have been assigned and data sets created with varying levels of missingness, we evaluate the performance of p-squashing at tuning parameters $\alpha = 1, 1.5, 2$. Based on the results of the prior

simulations, the value of $\alpha = 0.5$ is excluded due to its poor performance and only two layers in the data sphere clustering method since the third layer resulted in negligible improvement at these values of α. We additionally fit each model using the full data set for comparison. Results are shown in Figure 4.2 with performance on the full data set is denoted as $\alpha = \infty$.

Our simulation results show substantial improvements in percent bias, RMSE, and coverage probability as α increases from $\alpha = 1$ to $\alpha = 1.5$. The improvements in performance are less noticeable from $\alpha = 1.5$ to $\alpha = 2$, though some gains are observed. At $\alpha = 1.5$ the coverage is slightly anti-conservative, with all 95% confidence intervals having below 95%, but above 90% coverage probabilities. At $\alpha = 2$, some intervals are still slightly anti-conservative, but are practically indistinguishable from those of $\alpha = \infty$, suggesting that no further improvements can be made. Similarly, a small amount of bias is still observed at $\alpha = 2$ that is not present at $\alpha = \infty$. This bias does not noticeably impact RMSE, as the results between $\alpha = 2$ and $\alpha = \infty$ are highly similar.

The squashed sample size is noticeably larger when missing values are present than they are absent, with squashed sample sizes ranging from M=141 at $\alpha = 1$ and 20% missing, to M=336 at $\alpha = 2$ and 50% missing. The primary reason for this is that missingness patterns are treated as a categorical variable. In this simulation, the total number of categorical variable level combinations was increased from the three levels of A with fully observed data to 12 (three levels and four missingness patterns, including the fully observed pattern). Since numeric variables are further clustered within each categorical combination, the total number of clusters increases even further. When data were fully observed, the three categorical levels resulted in a total of 21 clusters. Under 20% and 50% missingness the 12 categorical levels were expanded to 49-51 (mean 50.9) and 48-51 clusters (mean 50.6), respectively.

We further note that at each value of alpha, the squashed sample size is approximately 12% larger under 50% missingness than under 20% missingness. The overall reduction rate is then determined both by the number of regions, as well as the distribution of points within each region. While there were slight differences between the number of clusters at 20% and 50% missingness, this difference in squashed sample sizes is better attributed to differing distributions in the size of each cluster. Since the number of points in each squashed region is set at a rate of $\alpha log_2(N)$, squashing is most efficient in reducing the sample size when applied to regions with a large number of observations. As illustrated in Figure 4.3, under 20% missingness the distribution of cluster sizes was noticeably more right-skewed.

4.3.3 Expectation Squashing - theory

For e-squashing, we wish to express the expectation of the log-likelihood with respect to the missing values in terms so that a Taylor series expansion can be applied within groupings defined by categorical levels. Letting X_{iO}, X_{iM} and A_{iO}, A_{iM} represent the set of numeric and categorical variables, respectively, that are either observed ("O") or are missing ("M") for observation i, the log-likelihood is

$$\log L(\theta|X, A) = \sum_{i=1}^{N} \log(f(X_{iO}, X_{iM}, A_{iO}, A_{iM})) \tag{4.17}$$

As in p-squashing, we would like to partition the data space into regions based on levels of

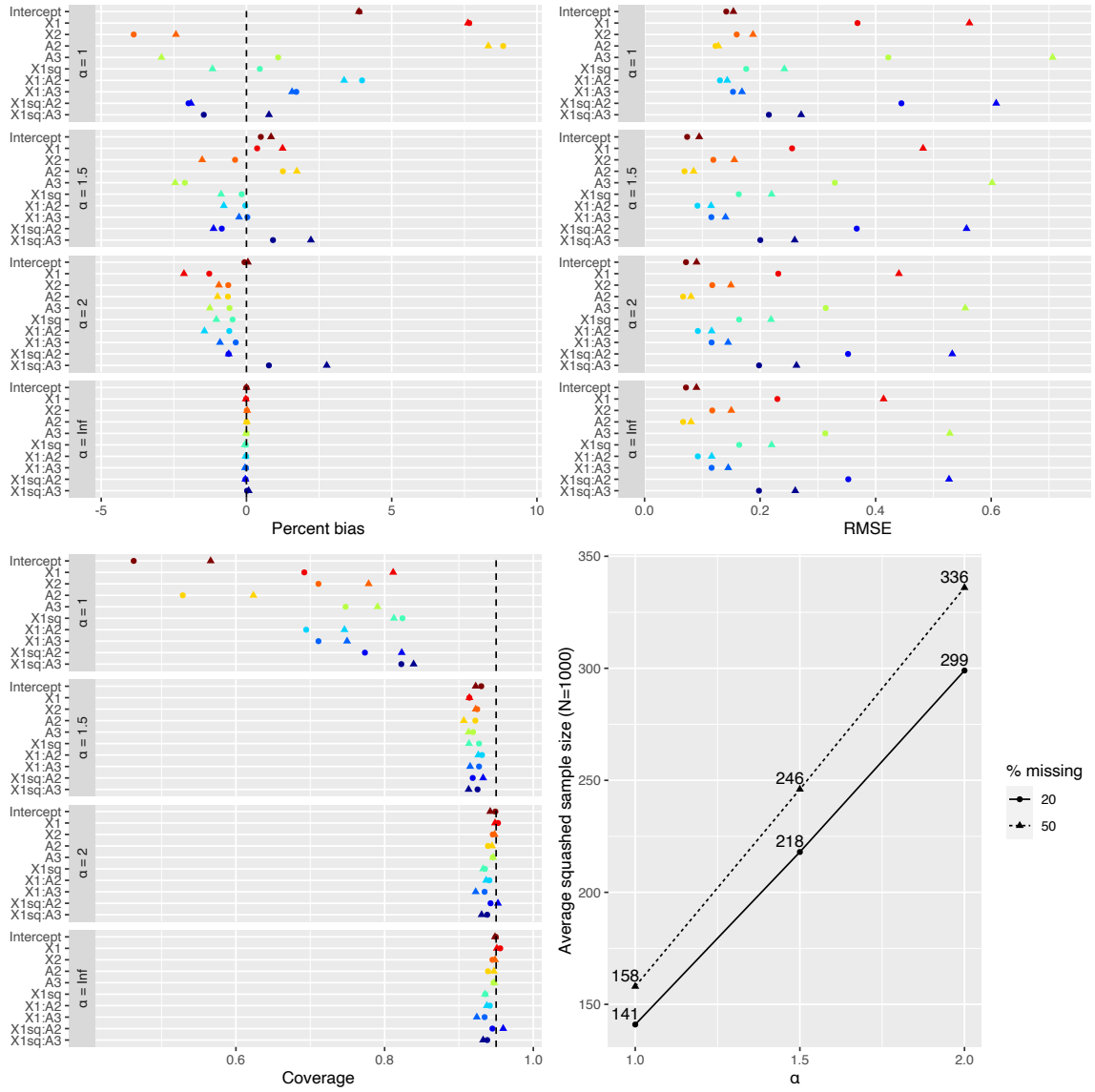

Figure 4.2: Performance of propagation squashing method. Missing values are assigned to 2,000 data sets of size N=1000 and all analyses are conducted using full-information maximum likelihood estimation. Alpha = Inf is used to designate the results of the analysis conducted on the full data set which is included for comparison.

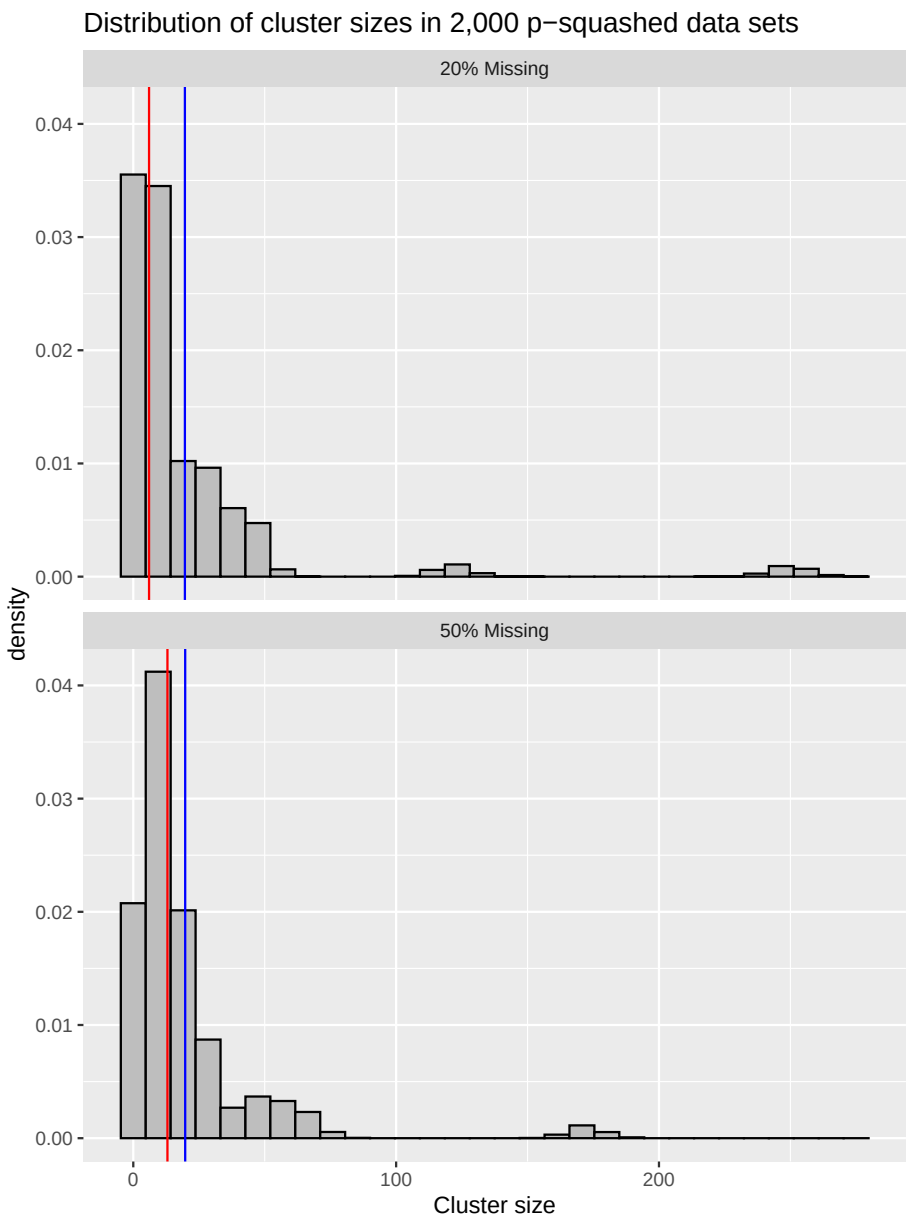

Figure 4.3: Distribution of cluster sizes in p-squashing procedure at 20% and 50% missingness. Vertical blue lines represent means while red lines represent medians. A higher degree of right-skewness is observed when 20% are missing than 50%, resulting in a higher degree of compression and smaller p-squashed sample sizes.

categorical variables and clusters of numeric variables within each categorical stratum. Unlike in p-squashing, however, the appropriate regions cannot be easily identified for observations with missing values in categorical or numeric variables.

Assume that an algorithm ϕ will be applied to each observation that will return a corresponding region of assignment, r for $r = 1, ..., R$.

$$\phi(X_i, A_i) = r \in 1, ..., R \tag{4.18}$$

This algorithm is defined broadly to be a function of both the numeric and the categorical variables. In the original squashing methodology, ϕ is the process of grouping observations first by categorical levels and then further into regions using a clustering algorithm. Additionally, define $Z_{ir}(X, A|\phi) = Z_{ir}$ to be an indicator that the ith observation is assigned to region r:

$$Z_{ir} = \begin{cases} 1 & \text{if } \phi(X_i, A_i) = r \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

Had the regions of assignment been fully observed, the missing categorical values, A_M, would be known for each observation. Consequently, the complete-data log likelihood and its expectation with respect to the missing values is

$$\begin{aligned} \log L(\theta|X, A) &= \sum_{r=1}^{R} \sum_{i=1}^{N_r} Z_{ir} \log(f(X_{iO}, X_{iM}|A = a)) \\ E_{X_M, A_M}[\log L(\theta|X, A)] &= \sum_{r=1}^{R} \sum_{i=1}^{N_r} E_{X_M, A_M}[Z_{ir}] E_{X_M|A_M}[\log(f(X_{iO}, X_{iM}|A = a))] \end{aligned} \tag{4.20}$$

We are now left with two terms: the expectation of an indicator that an observation with missing numeric and/or categorical variables will be assigned to each region, and the expected log likelihood within each region with respect to the missing numeric variables. In both cases, the expectation is conditional upon the observed data values, X_O and A_O, as well as the true model parameters, θ. For the rth region, the first term is equal to the probability that the clustering mechanism will return region r given the set of observed and missing values for observation i. By the definition of conditional probability, this can be separated into the probability that the missing categorical variables will take the value (singular) belonging to region r, multiplied by the probability that the numeric clustering mechanism will cluster values (X_{iO}, X_{iM}) to region r. Using capital letters to denote random variables and lower-case letters to denote observed values, the expectation of Z_{ir} is

$$\begin{aligned} E_{X_M, A_M}[Z_{ir}] &= p(\phi(X_i, A_i) = r) \\ &= p(\phi(X_{iM}, A_{iM}, x_{iO}, a_{iO}) = r|x_{iO}, a_{iO}) \\ &= p(A_{iM} = a_{iM}|x_{iO}, a_{iO}) p(\phi(X_{iM}, a_{iM}, x_{iO}, a_{iO}) = r|x_{iO}, a_{iO}, a_{iM}) \end{aligned} \tag{4.21}$$

The first term is calculated only when missingness exists within categorical variables. Assuming this is the case, we estimate $p(A_{iM} = a_{iM}|x_{iO}, a_{iO})$ by first clustering X_O and then calculating the probability of each level a_m within the region in which $X_O \approx x_O, A_O = a_O$. If we are willing to

assume that A_M is missing completely at random (MCAR), then $p(A_{iM} = a_{iM}|x_{iO}, a_{iO}) = p(A_{iM} = a_{iM})$ and can be calculated from the marginal distribution of A_M. For each observation with missing categorical values, $p(A_{iM} = a_{iM}|x_{iO}, a_{iO})$ will likely be non-zero for more than one a_{iM}. Practically, this means that the original data set must be expanded with imputed values for each possible a_{iM} and with weights given by $p(A_{iM} = a_{iM}|x_{iO}, a_{iO})$.

The second term, $p(\phi(X_{iM}, a_{iM}, x_{iO}, a_{iO}) = r|x_{iO}, a_{iO}, a_{iM})$, is non-zero only for the numeric clusters in which the $A = a_i$. Since numeric variables are clustered within categorical regions, this is a clustering problem with missing values: which cluster does observation i belong to if some of the values of x_i are missing? Depending on the clustering algorithm used, this could be calculated in a number of different ways. The primary criteria for the choice of a clustering algorithm are 1) that it is fast enough to work well in large samples, and 2) that one can estimate the marginal probability of an observation with some missing values belonging to each cluster. We apply the data sphere (DS) technique used by DuMouchel et al. (1999) as follows. For DS, clusters are formed on the basis of the fully observed rows. We then make the assumption that each cluster of points is multivariate-normally distributed and estimate the probability of each observation with missing values belonging to each cluster based on the multivariate normal distribution marginalized over the missing variables.

The final term, $E_{X_M}[\log(f(X_{iO}, X_{iM}|A = a))]$, describes the expected log-likelihood within each region with respect to the missing numeric variables and conditional on the observed data values and model parameters. As in the original squashing method, this can be approximated by a Taylor series expansion about point x.

$$\begin{aligned} E_{X_M}[\log(f(X_{iO}, X_{iM}|A = a))] &\approx E_{X_M}\left[\sum_{k=1}^{K} g_k \prod_{j=1}^{u(\xi)} (X_{ij} - x_j)^{p_{kj}}\right] \\ &= \sum_{k=1}^{K} g_k E_{X_M}\left[\prod_{j=1}^{u(\xi)} (X_{ij} - x_j)^{p_{kj}}\right] \end{aligned} \tag{4.22}$$

Expressing the full likelihood, we have

$$\begin{aligned} E_{X_M, A_M}[\log L(\theta|X, A)] \approx &\sum_{r=1}^{R}\sum_{i=1}^{N_r} p(A_{iM} = a_{iM}|x_{iO}, a_{iO}) p(\phi = r|x_{iO}, a_{iO}, a_{iM}) \\ &\cdot \sum_{k=1}^{K} g_k \cdot E_{X_M}\left[\prod_{j=1}^{u(\xi)} (X_{ij} - x_j)^{p_{kj}}\right] \end{aligned} \tag{4.23}$$

Using this equation, we can calculate the set of expected numeric variable moments on the original data set with the expectation evaluated over the distribution of missing values within each moment. Practically, this is achieved by calculating each moment based on the observed data values. When one of the variables involved in the moment calculation is missing for an observation, that observation's contribution to the moment will be missing and is imputed with the expected value of the non-missing contributions. Each observation's contribution is then weighted by the probability that it belongs to the cluster and summed across all observations within the cluster. All data points with fully observed variables will have only been assigned to one cluster and will have weights of

one.

4.3.3.1 Case when all numeric observations are missing within a cluster

In the event that all observations of a numeric variable are missing with a combination of categorical variables, it is not possible to calculate the term $E_{X_M}\left[\prod_{j=1}^{u(\xi)}(X_{ij}-x_j)^{p_{kj}}|A=a\right]$. In instances of this, we might instead attempt to borrow information from nearby clusters, where "nearby" is defined in terms of the distance between categorical variables. By doing so, we are instead calculating $E_{X_M}\left[\prod_{j=1}^{u(\xi)}(X_{ij}-x_j)^{p_{kj}}|A=a'\right]$ for some a' such that $a'=\arg\min_{a^*} d(a,a^*)$ for some distance metric d.

With only three categorical levels and a maximum of 50% of observations missing values, this does not occur during our evaluation of imputation squashing on the synthetic data sets. During later sections, however, we apply imputation squashing to a real data set in which this does occur. In these instances, values for the missing numeric variables are randomly sampled from the closest cluster(s) according to the "Dice coefficient" (or "Sørensen metric"). The Dice coefficient ranges from zero to one and can be defined in terms of a 2x2 contingency table once all categories have been binarized.

$$DSC=\frac{2TP}{2TP+FP+FN} \tag{4.24}$$

Where TP is the "true positive" rate – the number of categories observed in common between two rows. FP and FN are the "false positive" and "false negative" rates representing the number of categories for which the first row had values of one while the second had zero and vice versa.

4.3.4 Evaluation of expectation-squashing in simulations

E-squashing is similarly evaluated across the same set of 2,000 data sets as used for the original and p-squashing methods with results shown in Figure 4.4. Overall, the performance is notably worse than p-squashing. The simulation results show only modest improvements in bias and RMSE associated with higher values of α and, in general, appear to depend more upon the proportion of the data points missing than on the value of α. At 20% missingness, the bias in most parameter estimates is relatively small, with the exception of the coefficient for the third categorical group, $A3$, which had the fewest observations and approximately 13% relative bias. At 50% missingness, however, the percentage bias was 10% or higher for four parameters at $\alpha=2$.

As α increases, however, improvements are made in coverage probability. At $\alpha=1$, each of the 95% confidence intervals for the estimates are severely anti-conservative with several intervals below 50% coverage at 50% missingness and all intervals below 75% coverage regardless of level of missingness. At $\alpha=1.5$ the coverage is significantly improved, though still very poor. At 20% missing with $\alpha=1.5,2$, the coverage probability increases to approximately 75-92% coverage, though it remains below 60% for the majority of the intervals at 50% missingness.

Notably, the sample sizes are smaller in e-squashing than in p-squashing and are similar to those observed when no observations are missing. The primary reason for this is that clusters of numeric variables are formed using only fully observed data points and observations with missing values are assigned to existing clusters. Consequently, as the degree of missingness increases, the number of numeric clusters decreases. For a fixed sample size, this requires that the size of each clusters must

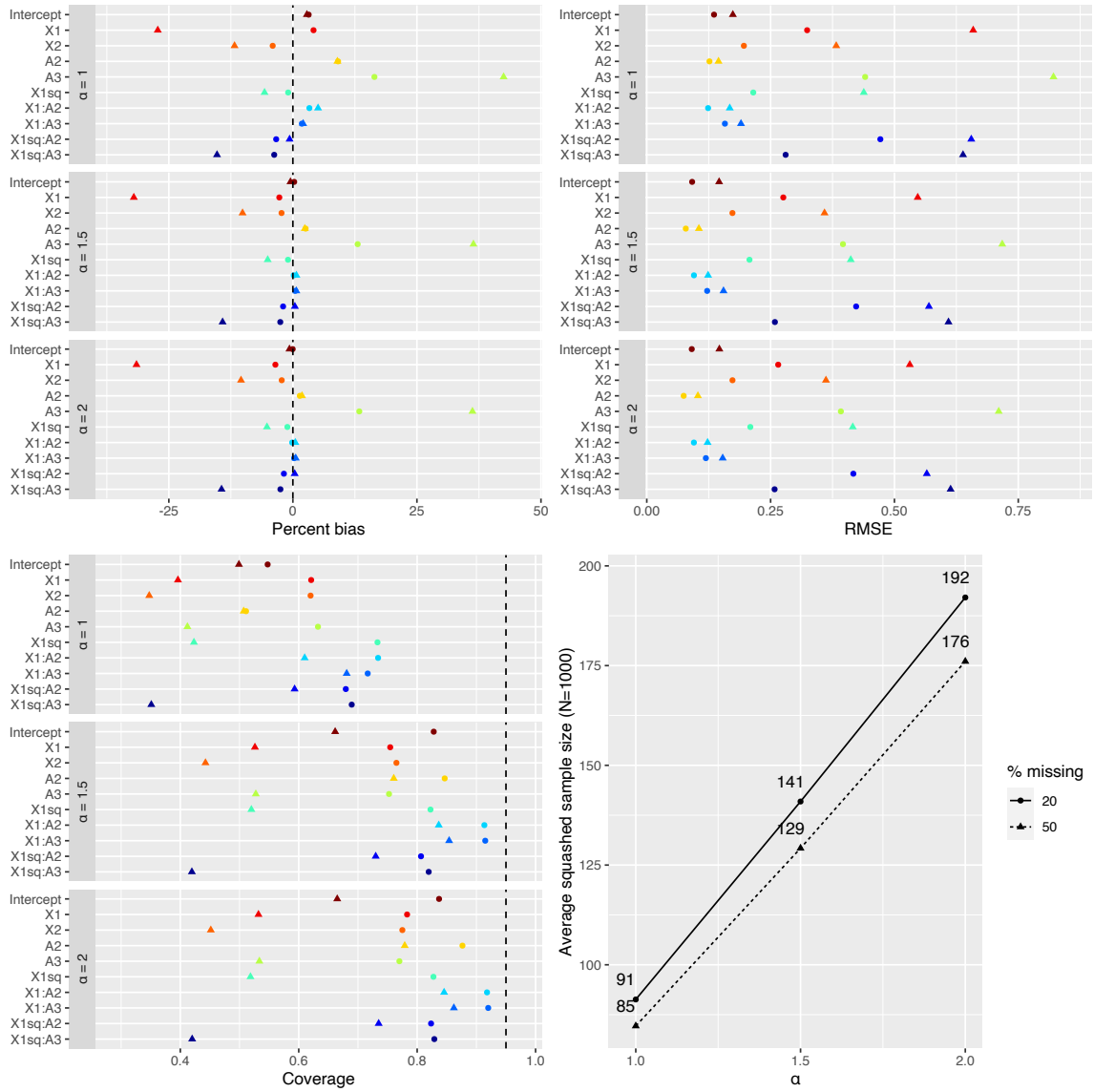

Figure 4.4: Performance of imputation squashing method. Values are missing at random in 2,000 data sets of size N=1000.

also increase, thereby allowing for a greater degree of compression. Table 4.2 displays the number of clusters resulting from the e-squashing procedure at 20% and 50% for each of the 2,000 data sets. Figure 4.5 displays the distribution of cluster sizes, illustrating that both the mean and median cluster size increases at 50% missingness, as does the proportion of clusters with very few points, while the distance between the mean and median values stays relatively constant.

Table 4.2: Distribution of number of clusters resulting from e-squashing procedure. Clusters in numeric variables are formed from fully observed data points. As the proportion of missingness increases, fewer clusters are formed with each cluster having a larger number of observations.

	50% Missing						
20% Missing	16	17	18	19	20	21	Sum
19	0	0	0	1	1	0	2
20	1	3	25	62	71	22	184
21	1	14	136	509	793	361	1814
Sum	2	17	161	572	865	383	2000

4.4 Workers' Compensation Example

4.4.1 Background

We illustrate data squashing on a subset of the New York Assembled Workers' Compensation Claims data set. This data set is publicly available at https://data.ny.gov/Government-Finance/Assembled-Workers-Compensation-Claims-Beginning-20/jshw-gkgu and was downloaded on June 18, 2021. These observations represent all workers' compensation claims made between January 1, 2000, and April 1, 2021 for payment due to illness or injury acquired in the course of employment. The data set provides a number of important variables, including the type of claim (e.g., medical expenditures, disability), the industry in which the worker was employed, several date variables (e.g., date of accident, date at which the claim was assembled), patient characteristics (age at injury and average weekly wage), and, conveniently, an indication of whether or not the claim was COVID-19-related. Both age at injury and workers' industry have non-negligible levels of missingness at 9% and 53% missing, respectively. A more complete description of the data set including pre-processing steps and a data dictionary of the final data set are provided in Section 4.4.2.

In the sections that follow, we evaluate the performance of our two squashing methodologies from the viewpoint of an analyst who is interested in evaluating the effect of COVID-19 on workers' comp claims. In a July, 2020 report by the international accounting firm KPMG noted that, historically, both the frequency and severity of workers' comp claims are affected by economic downturns.(KPMG, 2020) As unemployment rises, more experienced workers may remain, leading to a lower rate of minor injuries, but a higher proportion of severe injuries. Additionally, many employees may be reluctant to file claims for fear of antagonizing their employer, further contributing to a decrease in the quantity of claims. The KPMG report notes that these relationships may have been compounded by the medical nature of the economic crisis: workers may have been reluctant or unable to receive medical attention due to fear of COVID-19 exposure or over-burdened health care systems. Consequently, medical treatments may have been delayed, resulting in a spike in the severity of claims. Finally, closure of courts may have led to back-logs in processing workers compensation claims and delays in reaching settlements. Evidence for both a decrease in workers' comp claims and a back-log in

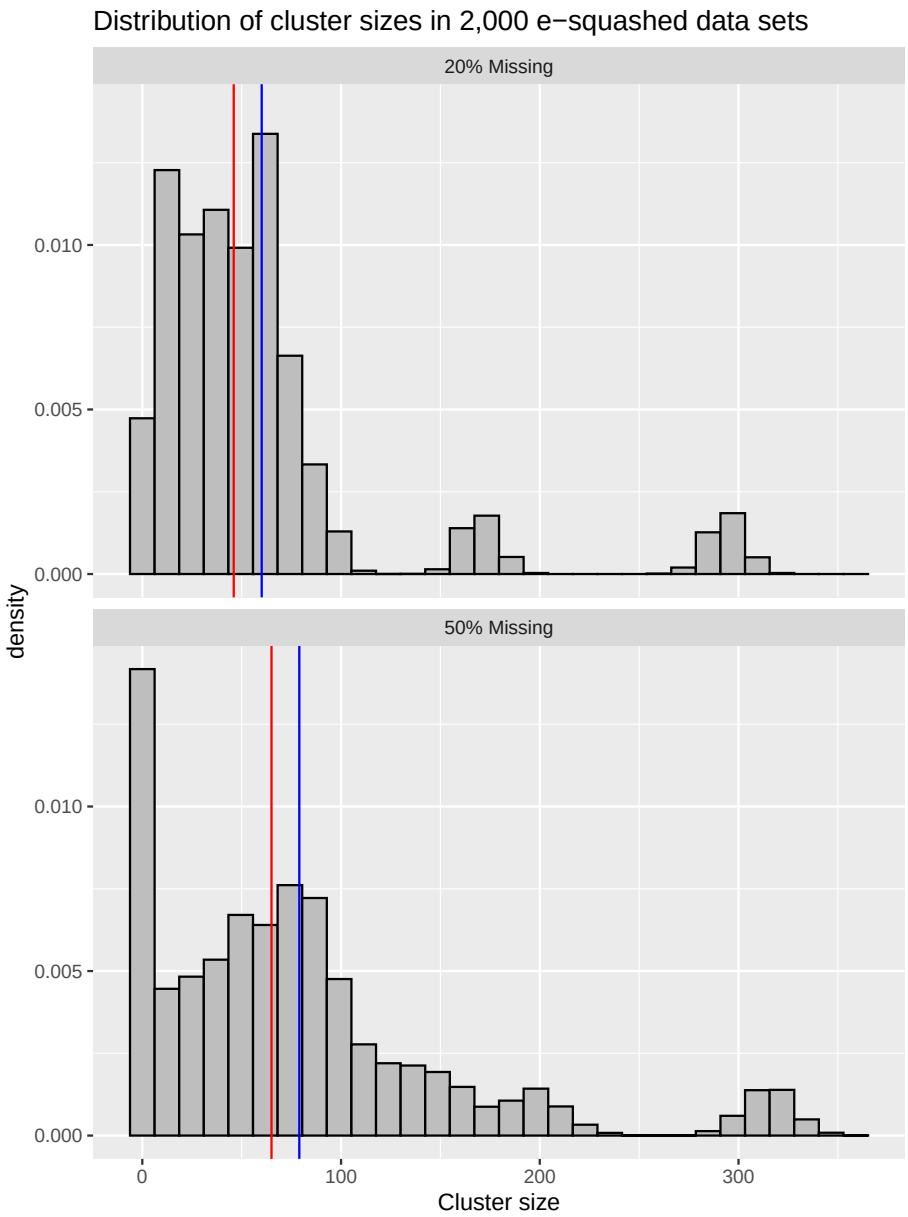

Figure 4.5: Distribution of cluster sizes in e-squashing procedure at 20% and 50% missingness. Vertical blue lines represent means while red lines represent medians. At 50% missingness fewer clusters are formed, causing the size of the clusters to increase. This is illustrated by the blue and red lines shifting right at 50% missing even while the distance between them remains similar.

processing is potentially provided by an April 2021 report issued by the New York Committee for Occupational Safety and Health. This report found that of the quarter-million workers who potentially contracted symptomatic COVID-19 and were eligible for workers' compensation, fewer than 9% had filed claims, while only ten percent of those filed had received final decisions regarding their claims.(Grey, 2021)

There appears to be a strong potential for COVID-19 to have impacted workers' compensation claims in a variety of meaningful ways. Due to the size of the data set and presence of missing data, analyses on the full data set, while perhaps possible, may be exceedingly cumbersome. This motivates the analysis of a subset of the data for data exploration and preliminary analyses steps before analyzing the full data set. Further, the presence of several research questions (and absence of a single outcome measurement) makes approaches such as likelihood-based data squashing less useful. Together, these conditions make the data set ideal for data squashing. Specifically, we consider the following statistical models to assess the corresponding questions:

1. Has COVID-19 caused an increase in longer time between assembly and Accident, Notice and Causal Relationship (ANCR) date? ANCR refers to the establishment that 1) a work-connected accident covered by workers' compensation law occurred, 2) the employer was notified within a sufficient time period, and 3) a causal relationship exists between the accident and injury exists. If COVID-19 has caused delays in the processing of workers' compensation claims, we may expect it to manifest as increased times between the filing and validation of claims. The logged difference (in years) between ANCR and assessment date is modeled as a function of covariates via **multiple linear regression**.

2. Has COVID-19 increased the severity of workers' compensation claims? Each claim indicates the level of disability that the worker is requesting compensation for. These have been grouped into categories "medical-only," "temporary disability," and "permanent disability or death." We model the relationship between claim type severity and COVID-19 using an **cumulative logit regression model** for ordinal data.

The data set includes a variable indicating that a claim is COVID-19-related; however, one may reasonably hypothesize that COVID-19 has had a broader impact on workers' compensation claims beyond those explicitly marked as COVID-19-related. For example, as noted by the KPMG report, workers may have been less willing or able to seek medical attention for non-COVID-related injuries or illnesses due to stay at home orders or a fear of contracting COVID-19 within medical facilities. To capture this possibility, we create an additional "COVID-Period" variable, indicating that the claim was assembled at or following March 1, 2020. This corresponds closely to the specific timeline of COVID-19 in New York, with the first case being observed in New York state on March 1, 2020, and the state entering a state of emergency on March 7, 2020.(Kerr, 2021). For the squashed data sets, the COVID-Period variable was created post-squashing and, therefore, relies on the squashing procedure accurately preserving relationships between each outcome variable and the claim assembly date at a granular level.

4.4.2 Pre-processing of workers' compensation data set

The original data set contained 4,232,321 observations of 54 variables at the time of download. Several pre-processing steps are taken to reduce the size of the data set prior to squashing. A

central problem of data squashing is the rapid proliferation of categories when even a relatively low number of categorical variables are included. As downloaded, 30 of the 54 variables were categorical. While this may seem a relatively small number in relation to the 4.2 million observations, the 30 variables resulted in 3,780,266 unique combinations. After removing the two variables with the highest number of categories: zip code (22,275 categories) and insurance carrier name (2,616 categories), all categorical variables have a maximum of 87 unique values. Nonetheless, the total number of combinations of categories was reduced only to 2,705,291.

This illustrates a serious limitation that many may encounter while implementing squashing on real-life data sets: many, if not most, large data sets have a sufficiently high number of categorical variable combinations so as to undermine the utility of squashing. In these instances, the existing squashing methodology *can* still be applied to create a squashed data set. Without a high degree of data compression, however, squashing is unlikely to offer much practical utility.

While this is a central issue to applying data squashing more broadly, it is secondary to the present analysis, which examines methods of handling missing data. The original squashing paper by DuMouchel et al. (1999) contained only two categorical variables with three levels each, for a maximum of nine categorical-level combinations for a data set with 744,963 records. To make our analysis more comparable to this setting, we take the following pre-processing steps.

- Subset to workers compensation claims only. Volunteer firefighter and volunteer ambulance workers' claims were excluded. (`Claim.Type == "WORKERS COMPENSATION CLAIM"`)
- Subset to claims that used a standard adjudication process, as opposed to those that were external to the workers' compensation board. (`Alternative.Dispute.Resolution == "N"`)
- Subset to claims filed regarding workplace accidents, as opposed to occupational diseases. (`Accident == "Y"`)
- Subset to claims for which compensation was received (`Claim.Injury.Type != "1.  CANCELLED" & Claim.Injury.Type != "2.  NON-COMP"`)
- Consolidate all claims regarding permanent partial disability, permanent total disability, or death into a single category (replace levels `"5.  PPD SCH LOSS","6.  PPD NSL","7.  PTD","8. DEATH"` with `"PD_OR_DEATH"` in variable `Claim.Injury.Type`)
- Consolidate levels of Industry.Code.Description as displayed in Table 4.3

We additionally take the following steps oriented towards "cleaning" the data or transforming highly-skewed variables.

- Replace values of "0" with "NA" in variable `Age.at.Injury`
- Take $\log_{10}$ transformation of `Average.Weekly.Wage`
- Take natural log transformation of `Interval.Assembled.to.ANCR`
- Convert all dates into numeric values indicating the number of years after Jan. 1, 2000

The final data set consists of 2,191,006 observations on 11 variables (excluding the COVID.PERIOD variable) with 35 unique combinations of categorical variables and 84 combinations when including missingness patterns. A data dictionary of the variables retained in the data set is given in Table 4.4, including abbreviated names used in Figures. Figure 4.6 displays the variable distributions.

Table 4.3: Consolidation of levels in industry type variable.

Old levels	New level
INFORMATION PROFESSIONAL, SCIENTIFIC, AND TECHNICAL SERVICES ADMINISTRATIVE AND SUPPORT WASTE MANAGEMENT AND REMEDIAT EDUCATIONAL SERVICES HEALTH CARE AND SOCIAL ASSISTANCE ARTS, ENTERTAINMENT, AND RECREATION OTHER SERVICES (EXCEPT PUBLIC ADMINISTRATION)	SERVICES
CONSTRUCTION MANUFACTURING	CONSTRUCTION_AND_MANUFACTURING
RETAIL TRADE ACCOMMODATION AND FOOD SERVICES WHOLESALE TRADE	TRADE
REAL ESTATE AND RENTAL AND LEASING FINANCE AND INSURANCE MINING MANAGEMENT OF COMPANIES AND ENTERPRISES AGRICULTURE, FORESTRY, FISHING AND HUNTING UTILITIES TRANSPORTATION AND WAREHOUSING	OTHER

Table 4.4: Description of variables selected from Workers Compensation data set.

Variable	Labels	Class	NAs
COVID.19.Indicator	COVID.CLAIM	character	0 (0%)
COVID.19.Period	COVID.PERIOD	integer	0 (0%)
Claim.Injury.Type	A1o	character	0 (0%)
Age.at.Injury	X1m	integer	205736 (9%)
Assembly.Date	X2o	numeric	0 (0%)
Accident.Date	X3m	numeric	3341 (0%)
ANCR.Date	X4m	numeric	3 (0%)
Hearing.Count	X5o	integer	0 (0%)
Closed.Count	X6o	integer	0 (0%)
Industry.Code.Description	A2m	character	1169482 (53%)
Log10.Average.Weekly.Wage	X7o	numeric	0 (0%)
Log.Interval.Assembled.to.ANCR	X8o	numeric	6 (0%)

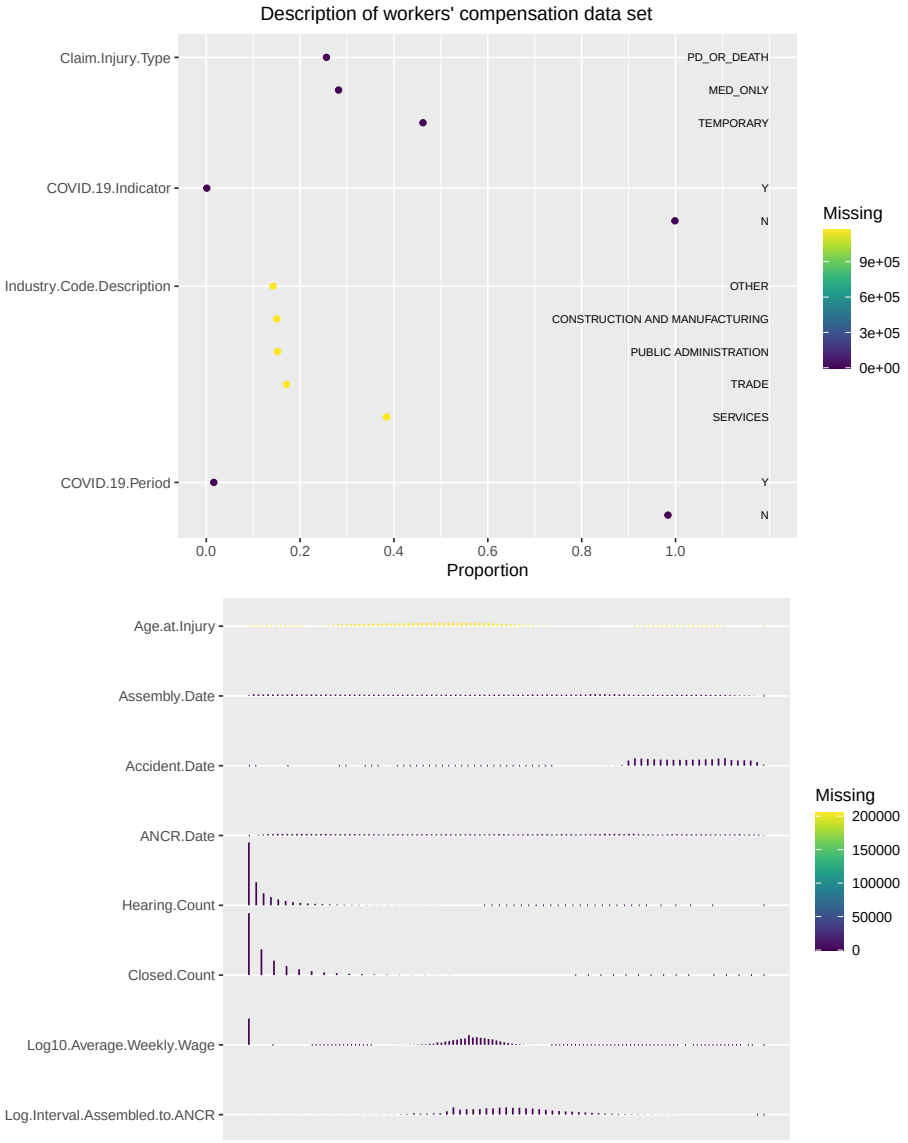

Figure 4.6: Description of variables in workers' compensation data set.

4.4.3 Squashing procedures

The reduced workers compensation data set was squashed using the p-squashing procedure with the tuning parameter set to $\alpha = 1.5$, resulting in a data set of 32,027 weighted pseudo-observations for a reduction factor of 2,191,006/32,027 = 68.4. A total of 4,877 clusters were formed with sample sizes ranging from 1 to 153,575 observations. All missingness patterns present in the original data set were retained within the p-squashed data set.

For the e-squashing procedure we set the tuning parameter $\alpha = 2$ based on its improved performance in prior simulations. This resulted in a data set of 27,660 weighted pseudo-observations (reduction factor of 79.2) that were fully observed across all variables. Expansion of the data set across missingness in categorical variables resulted initially in a data set with 6,862,293 observations. A total of 2,629 clusters were formed with cluster sizes ranging from 1 to 324,716 rows and observation weights within groups ranging from 0.01 to 118,630.24.

The performance of each squashing procedure is compared to the full data set, as well as a simple random sample of $N = 32,000$ observations. Missing values remain present in the random sample and are handled in the same method as the p-squashed data set, described in the following section.

4.4.4 Analyses

4.4.4.1 Handling of missing data

Missingness in the full, p-squashed, and random sample data sets were handled by either FIML or multiple imputation (MI) depending on the analysis. Since the p-squashed data set is created to preserve the likelihood used in FIML, we believed it important to demonstrate its performance when possible. Unfortunately, software capable of implementing FIML is still limited: most software packages capable of implementing FIML can only do so for linear models and cannot be extended to nonlinear models, such as glms, or semi-parametric regression models such as proportional hazards (i.e., Cox) regression. This is true for the R packages `lavaan` and `OpenMx`, as well as the SAS statistical software program (https://www.sas.com/). The exception to this is the program Mplus (http://www.statmodel.com/), which can implement FIML within a much broader array of models but requires that the user purchases a license.

Given this constraint, we use FIML via the `lavaan` package in the first analysis of a multiple linear regression model and MI for the nonlinear ordinal regression model. For each data set (i.e., full, p-squashed, random sample), MI is implemented using predictive mean matching with the R package `mice`. The R package `miceadds` provides functions that extend predictive mean matching to data sets with frequency weights. Analyses are conducted on ten sets of imputations per data set.

4.4.4.2 Analysis 1 - Multiple linear regression with FIML

We begin by estimating the impact of COVID-19 on the logged number of years between claim assembly date and ANCR date adjusted for 1) a linear combination of predictors, and 2) all second-order terms and interactions between covariates. Claims for which the claimant did not receive any compensation were not verified and, therefore, did not have an ANCR date. These observations were omitted from this analysis but were included in the nonlinear model analysis.

The results of this analysis are shown in Figure 4.7. Each method is compared to the results from fitting the model to the full data set. Results are summarized in terms of the relative error

from the full value ($\hat{\beta}_{est} - \hat{\beta}_{full}/SE_{full}$), the unscaled error between the estimate and the full value ($\hat{\beta}_{est} - \hat{\beta}_{full}$), and the ratio of regression coefficient standard errors ($SE_{\hat{\beta}_{est}}/SE_{\hat{\beta}_{full}}$).

We observe very close agreement between the full data model estimates and those of the p-squashed model. For the p-squashed model, the relative difference is limited to approximately ±5, while the error is small enough so as to make inference from the two models practically indistinguishable. Further, the ratio of standard errors between regression coefficients is 1.01 with a minimum and maximum values of 0.96 and 1.05, respectively.

The e-squashed model performs slightly worse than the p-squashed model, although still relatively well. Only a couple of parameter estimates deviate from those of the full model, though not noticeably more so than the estimates of the simple random sample. With the exception of one parameter (`X1`), the standard errors of the e-squashed estimates are practically identical to those of the full data analysis with standard error ratios falling between 1 and 1.05.

The variable `X1` corresponds to the claimant's age at injury which is missing approximately 9% of its observations. While this is not the variable with the highest proportion of missing data (worker's industry is missing on 53% of its observations), it is the highest proportion missing among the numeric variables, which is handled differently by the e-squashing procedure than missingness in categorical variables. For categorical variables, the data set is expanded to place each observation in *all* relevant categories, weighted by the corresponding probability of assignment. Observations with missing numeric variables are similarly assigned to multiple clusters, but within each cluster the moments of the original data set are calculated via mean imputation within each moment. This provides evidence that the e-squashing procedure may be better at reproducing the uncertainty due to missingness when it occurs in categorical variables than when it occurs in numeric variables.

Similar trends are observed when second-order quadratic and interaction terms are added to the model as in the first-order model. Again, the p-squashing returns parameter estimates that are highly similar to the full-data estimates with all estimates within ±10 standard errors and within ±0.05 of the full-data estimates. The standard errors of the full-data estimates are likewise well-reproduced by the p-squashed model, with all standard error ratios falling between 0.95 and 1.15.

The e-squashed model again performs worse than the p-squashed model with relative errors falling between -23 to 18 standard errors, raw errors approximately between ±0.1, and standard error ratios between 0.4 and 1.3. Overall, the relative error and raw errors observed with the e-squashed estimates are comparable to those of the simple random sample. The e-squashed data set, however, has a sample size only 86% that of the random sample and has the added and substantial benefit of not having any missing values.

Returning to our narrative of a hypothetical researcher interested in the effect of COVID-19 on workers' compensation claims, from this analysis we estimate substantially different effects associated with claims being marked as COVID-19-related (`COVID.CLAIM`) and claims that were assembled post March 1, 2020 (`COVID.PERIOD`). Using the second-order model, COVID-19-related claims tended to be associated with longer time durations between claim assembly and claim verification with the following estimated effects [95% CI].

- full data: 0.224 [0.196,0.254]
- p-squash: 0.223 [0.193,0.252]
- e-squash: 0.211 [0.182, 0.241]
- sample: 0.264 [0.049, 0.480]

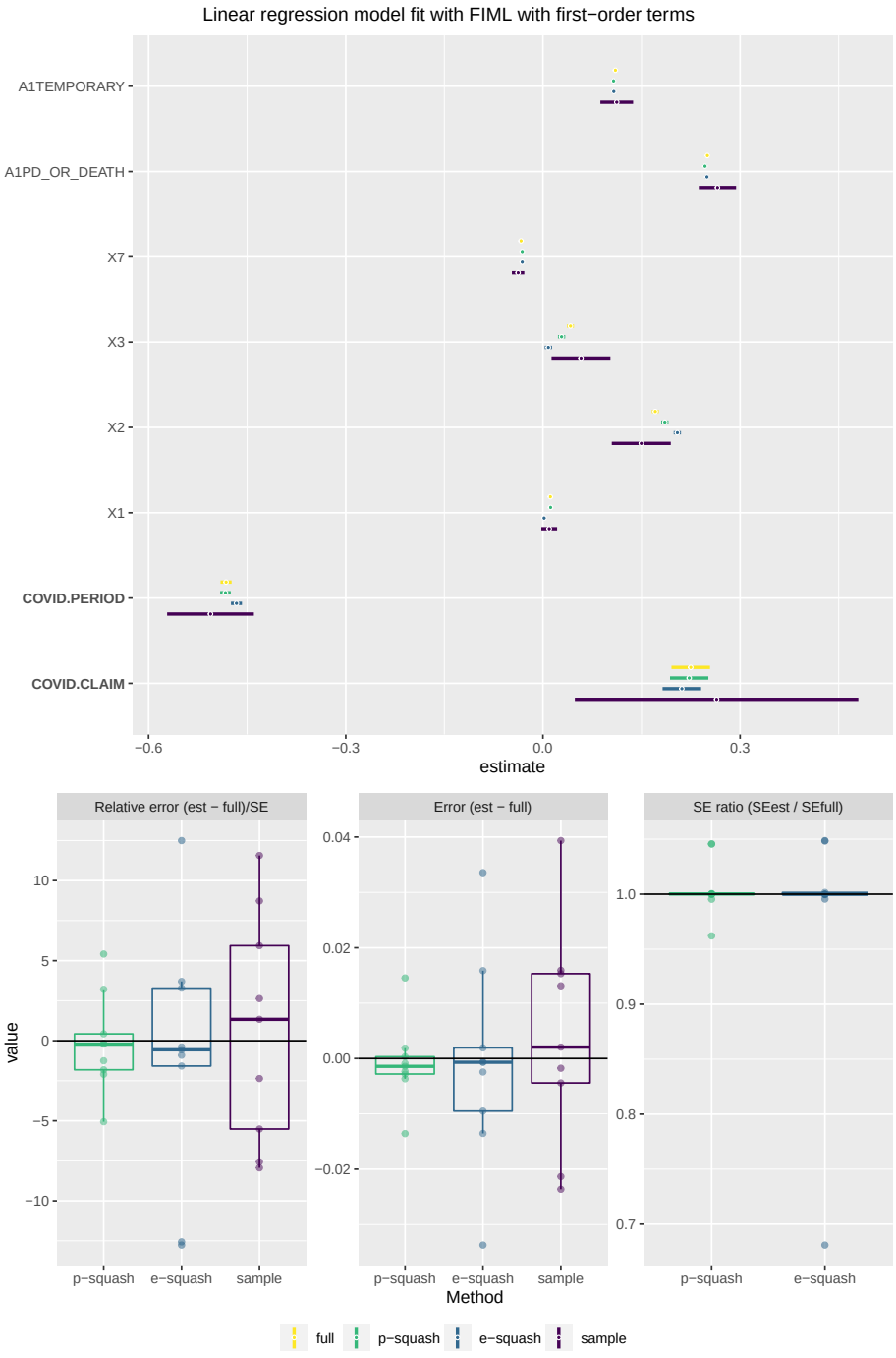

Figure 4.7: Linear regression on logged time until ANCR with first-order terms conducted via FIML. Sample sizes for each data set are as follows: full = 2,191,006; p-squashed = 32,027; e-squashed = 27,660; sample = 32,000.

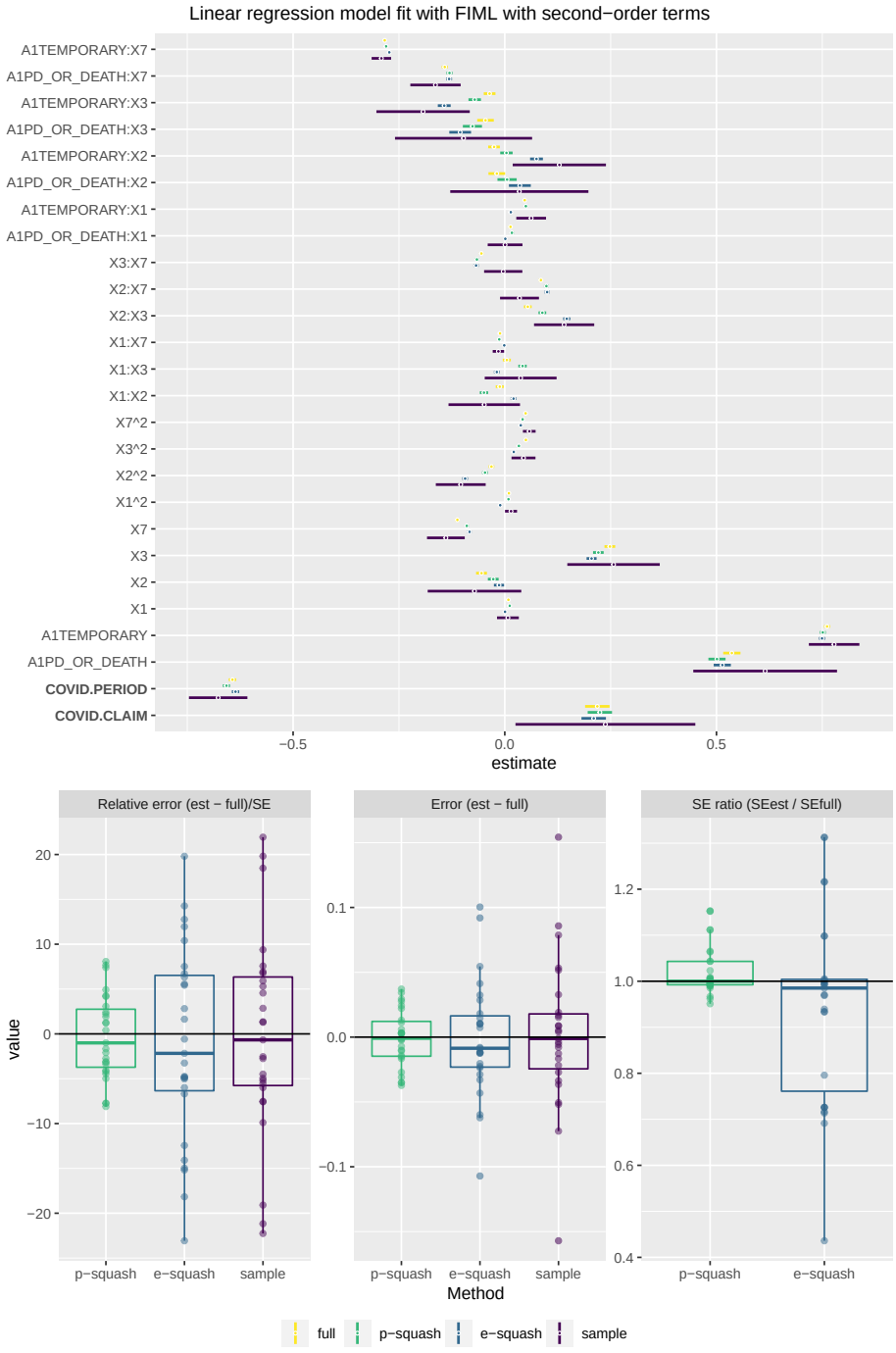

Figure 4.8: Linear regression on logged time until ANCR with second-order terms. Estimation with missing values is conducted via FIML. Sample sizes for each data set are as follows: full = 2,191,006; p-squashed = 32,027; e-squashed = 27,660; sample = 32,000.

By contrast, claims assembled post March 1, 2020 were associated with much faster verification with the following estimates and intervals for the `COVID.PERIOD` variable.

- full data: -0.482 [-0.491, -0.474]
- p-squash : -0.483 [-0.491, -0.475]
- e-squash : -0.466 [-0.475, -0.458]
- sample : -0.506 [-0.572, -0.440]

4.4.4.3 Analysis 2 - Ordinal regression with MI

We next consider a model that a researcher may estimate to evaluate the impact of COVID-19 on the severity of workers' compensation claims. The variable `Claim.Injury.Type` indicates the type of claim that was assembled: medical expenses only, temporary disability, or permanent disability (partial or total) and/or death. Given that this variable has levels that are ordered in terms of severity, we consider the estimation of a cumulative logit (i.e., "proportional odds") model, fit using the R function `clm` from the `ordinal` package. We again consider two models in which only first-order, and all first- and second-order terms are entered as predictors. For these analyses multiple imputation, rather than FIML, was used to handle missing values when present.

Overall, the results from the first-order ordinal model, displayed in Figure 4.9, and the second-order ordinal model, Figure 4.10, are similar to the results of the linear model with FIML. In both models, p-squashing tends to result in low relative and absolute errors, while also closely replicating the standard errors of the full-data model. E-squashing similarly tends to do well on most parameters but has more bias than p-squashing. This bias is particularly pronounced on the `X1m` parameter estimate (relative error of -50.79 in the first-order model), for which the estimated standard error is substantially anti-conservative (SE ratio of 0.71).

Concerning the effect of COVID-19 on the severity of the claims, we estimate that, contrary to the predictions of (KPMG, 2020), we did not observe an association between COVID-19 and the severity of workers' compensation claims. For claims marked as COVID-19-related, we estimate a decreased association with increased claim severity.

- full data: -0.307 [-0.390, -0.223]
- p-squash: -0.280 [-0.364, -0.197]
- e-squash: -0.272 [-0.356, -0.188]
- sample: -0.462 [-1.081, 0.157]

Claims that were assembled post March 1, 2020, were associated with an even more pronounced reduction in claim severity.

- full data: -0.532 [-0.557, -0.507]
- p-squash: -0.562 [-0.586, -0.538]
- e-squash: -0.534 [-0.558, -0.509]
- sample: -0.415 [-0.609, -0.221]

A disclaimer is required for these analyses: the goal of the analyses is to illustrate the degree to which similar inferences are obtained from the full and squashed models, not to fit the best model possible or to accurately assess the impact of COVID-19 on each outcome. We have not performed model checks, such as an examination of the residuals for the multiple linear regression model or

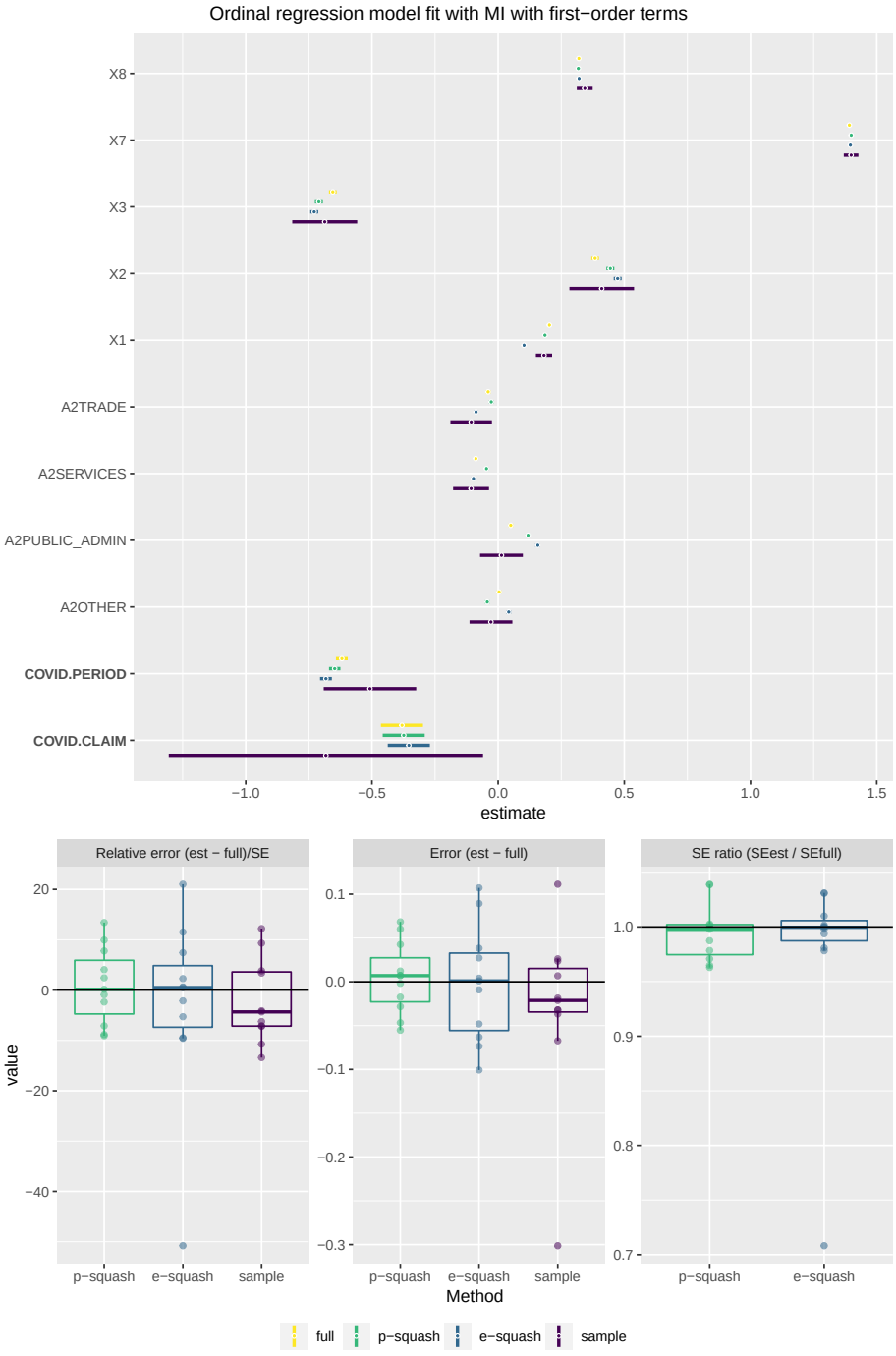

Figure 4.9: Ordinal regression on claim type with first-order terms. Estimation with missing values is conducted via multiple imputation. The imputation model contains all first-order terms only. Sample sizes for each data set are as follows: full = 2,191,006; p-squashed = 32,027; e-squashed = 27,660; sample = 32,000.

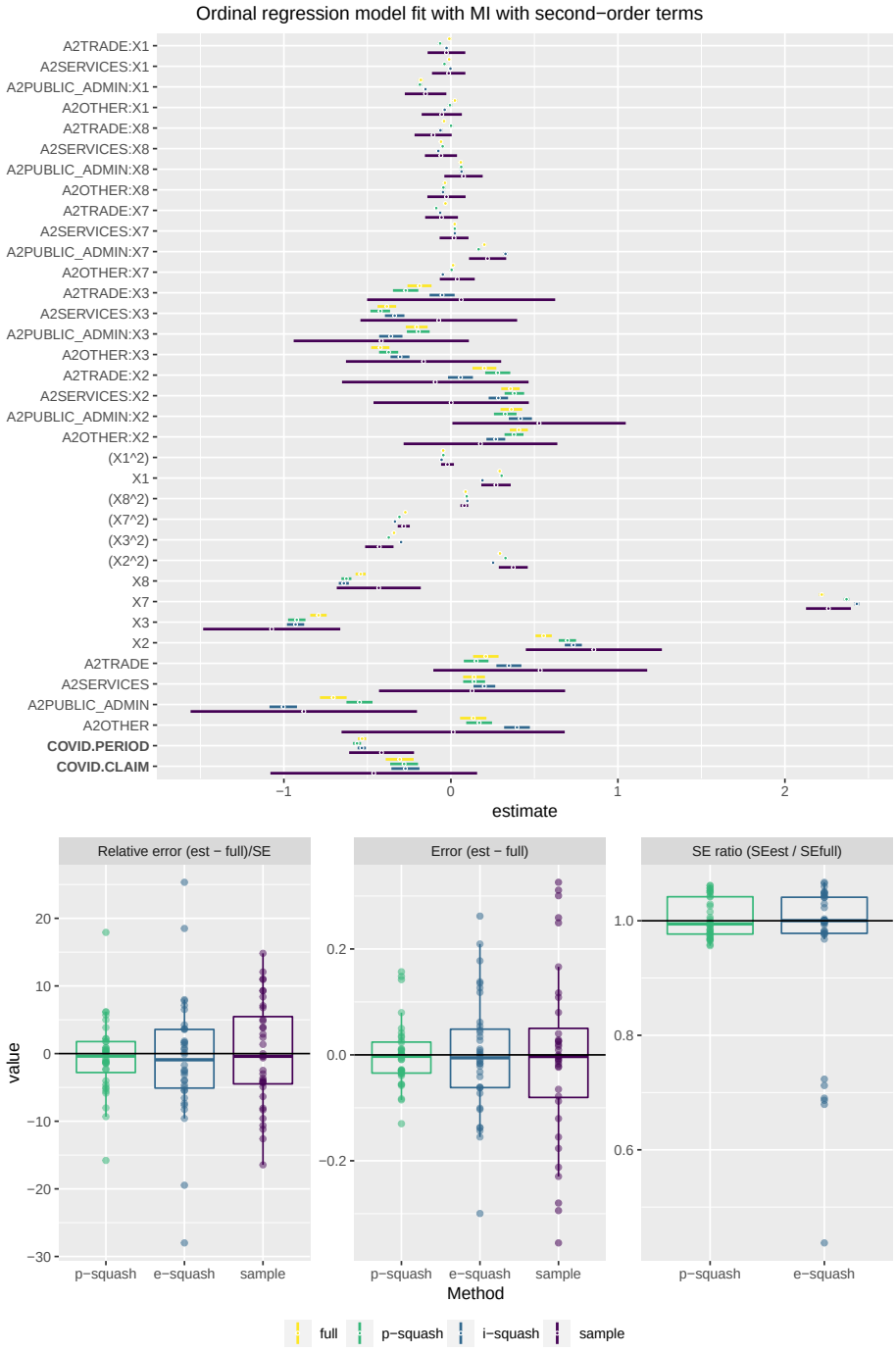

Figure 4.10: Ordinal regression on claim type with all first- and second-order terms. Estimation with missing values is conducted via multiple imputation. The imputation model contains all first-order terms only. Sample sizes for each data set are as follows: full = 2,191,006; p-squashed = 32,027; e-squashed = 27,660; sample = 32,000/

an evaluation of the proportional odds assumption for the cumulative logit model. Similarly, we do not have any particular subject knowledge of which variables should be entered into each model or which nonlinear terms or interactions may be important to capture. These are important steps of an analysis, and a different analysis may result in substantially different conclusions. The squashing procedure, however, aims to preserve any likelihood function and is not dependent upon it being the one that best describes the data. Thus, any errors or biases induced by a poor analysis of a data set will be reproduced in an analysis of a squashed version of the same data set.

4.5 Discussion

Within this chapter we have presented two possible extensions to DuMouchel et al.'s "data squashing" methodology in order to accommodate missing values in the original data set. In the first, termed "propagation-squashing," missingness patterns are treated as a categorical variable such that squashing occurs on the fully observed variables within each pattern, thereby propagating the missingness to the squashed data set. The benefits of this method are that it preserves the structure of the original data as closely as possible and preserves the information required for likelihood-based missing data methods. Though missing values remain in the p-squashed data set, there are many cases when missing values provide information about the data generating process. An example of this was observed in the workers' compensation data set, wherein claims that were rejected for payment did not have an ANCR date because they were never verified. P-squashing demonstrated better performance in matching the full-data estimates than e-squashing or random sampling using both FIML and MI on linear and nonlinear models. Moreover, it was considerably easier and more time-efficient to apply missing data methods to the p-squashed data set than it was to the full data set.

The second method, e-squashing, applied data squashing to the expectation of the log-likelihood, thereby adding an additional degree of approximation in order to produce a squashed data set that was free of missing values. As expected, e-squashing did not perform as well as p-squashing when applied to simulated or real data, though it obtained similar error rates to results of FIML or MI applied to a simple random sample with a larger sample size. The standard error estimates from e-squashing tended to be close to those of the full-data analysis, except for numeric variables in which missingness was observed. For these parameter estimates, the standard errors tended to be anti-conservative.

Depending on the context, inference from the e-squashed data set may be "good enough." When working with a subset of the data set, one should anticipate that the resulting analysis will imperfectly reproduce the full-data estimates. Given this caveat, even as the complexity of the models increased, both squashing methods resulted in inferences similar to the full data set. Due to the convenience of working with a fully observed data set, e-squashing may be the preferred choice when approximately representative results are sufficient. P-squashing, by contrast, would likely be preferred when stricter fidelity to the original data is required or when the missingness structure itself may be informative.

Since both squashing methods result in sets of pseudo-data, a major application of data squashing may be in facilitating data sharing when it otherwise may be difficult due to privacy concerns or proprietary interests. The choice of a squashing method may then be better determined by the desired use of the end-user. If the intent of squashing is to share pseudo-data that is sensitive in its

original form, it is important to ensure that there are no clusters in which the specific combination of categorical variables is informative about the person(s) identity. An obvious case of this is when only one observation occurs with a specific combination of categorical variables. When this occurs, the present squashing procedure returns the exact row of the data with a weight of one. This is easily avoided by identifying and either removing or modifying any squashed observations that have a weight of one.

Significant work remains to improve squashing more broadly, as well as the extensions proposed in this chapter. Nonetheless, the results from our analyses are promising. Missing data is a ubiquitous problem in real data sets and most models and machine learning algorithms cannot accommodate missing values. Due to the size of the data set or the time required to fit models, analysts may opt to either discard observations with missing values or perform single imputation, rather than taking more principled approaches such as FIML or MI. The advances made in this chapter largely mitigate this concern in situations where squashing can be applied. Due to its smaller size, missing data techniques applied to the p-squashed data set required substantially less computational time than to the full data set; for the e-squashed data set the problem was non-existent.

The most important methodological barrier remaining to more widespread implementation of squashing is, in the author's opinion, its ability to be gainfully applied to data sets with a moderate or high number of categorical variables. Our thoughts of how this may be done are detailed as we conclude in the following chapter.

Chapter 5

Conclusion

In this dissertation, we have extended two areas of research. In Chapters 2 and 3, we contributed to efforts to automate the induction of anesthesia through closed-loop control. This is an area of research that has received substantial attention from the anesthesiology and engineering research communities and has been referred to as "the Holy Grail" of anesthesiology.(Absalom et al., 2011) In Chapter 2, we use the tools of optimal experimental design to develop a framework for optimizing the set of targets used by a closed-loop controller in the form of a "reference function." Our simulations suggest that using a reference function to slow down infusions, relative to continuously targeting BIS=50, can reduce 1) over- and under-shoot of the target zone (i.e., BIS = [40,60]) 2) time until stable entry into the target zone,

2) the time until the patient stably enters the range BIS=[40,60], 2) the variability betwe

We show that this approach can be adapted both to different functional forms and a variety of clinical objectives, and that doing so can provide gains across a number of criteria relative to more conventional clinical approaches.

Additional highlights of the reference function methodology are that the reference function can be optimized offline (i.e., prior to giving the patient any medication) and can be applied to any closed-loop controller, as the common feature across the wide variety of controllers developed is the input of a reference value about which to maintain the system.

Chapter 3 furthered this work by introducing an R software package, `tci`, that implements target-controlled infusion algorithms to simulate open- and closed-loop control for compartmental models. Despite the growth of interest in TCI specifically, and automated control of drug administration more broadly, there is an obvious dearth of software available to practicioners or researchers who may wish to explore the topic. No software previously existed in any commonly used statistical programming language for implementing TCI algorithms or simulating patient responses when such algorithms are applied. We hope to fill this gap by introducing the `tci` package, thereby allowing others to more easily conduct their own research into a highly promising technology.

The final chapter of the dissertation attempted to address one of the primary barriers to implementation of data squashing: missing values in the original data set. The two proposed approaches, p-squashing and e-squashing, both performed with reasonable levels of success within simulations and on a real data set comprised of workers' compensation claims. The p-squashing approach represents a relatively straightforward extension to the original squashing method and propagates identical missingness patterns from the original data set to the squashed. This is done, however, so that that the information required for likelihood-based missing data methods is preserved. Application of such missing data methods to the substantially smaller p-squashed data set is far more convenient than to the original data set.

E-squashing, by contrast, modifies the original method more drastically by creating a squashed data set intended to reproduce the expectation of an arbitrary log-likelihood, rather than the likelihood itself. In doing so, it adds an extra degree of approximation in order to remove missing values from the squashed data set. While e-squashing did not perform at the same level as p-squashing, given the significant difficulties associated with missing data, the results were not unsatisfactory.

Depending on the application, the performance of e-squashing may be sufficient and likely could be improved with further work.

5.0.1 Future work

The central difficulty of control systems for drug delivery within anesthesiology, and likely many other fields of medicine, is the substantial variability in pharmacokinetics and pharmacodynamics between patients. In Chapter 2, we approached this problem by trying to identify an optimal dosing protocol that incorporated a measurement of the PK-PD variability within a pre-defined population. There are, however, other potential targets for intervention. Rather than designing the set of targets passed to the TCI algorithm to be robust to inter-patient variability, one could consider modifying the TCI algorithm itself to incorporate the unexplained PK-PD variability for each patient.

For example, in our simulations, the posterior PK-PD distribution for each patient evolved over time as data were collected. At the start of each infusion, the posterior point estimates were used by the TCI algorithm to calculate the infusion rate needed to reach the target. Rather than passing on the point estimates, however, the TCI algorithm could be modified to return an infusion rate based on the full posterior distribution, such as the maximum infusion rate with a 5% probability of overshooting the target. Such an approach would be less broadly applicable across controllers than the reference function methodology of Chapter 2 and would require that all computations be carried out at the time of induction. Nonetheless, it may provide a more personalized approach to mitigating the effect of unexplained inter-patient PK-PD variability. It would also have the capability of handling intra-patient variability, should patient's PK-PD change during the course of a procedure.

Several additions to the `tci` software package are planned or already underway. The pharmacokinetic models, as well as the `predict.pkmod` method that applies them to an infusion schedule, have begun to be translated into the C++ programming language using the R package `Rcpp`. Since these functions are the basis for the TCI algorithms and the simulation functions, this change is expected to substantially improve the computational speed of most functions within the software package. We additionally intend to extend the set of population PK models included within `tci` and improve the functions that interface with them. Many, if not most, users of the `tci` package will be interested in simulating responses from a specific population PK or PK-PD model. While we cannot anticipate every model that others would wish to use, further work can be done to make it easy for users to set up, and simulate from, their own user-specified models. Finally, we hope to extend the closed-loop control functions in future versions of `tci`. Currently, the user can only specify slight modifications to the Bayesian closed-loop controller described in Chapter 2 and the simulation of closed-loop control is closely tied to the use of the specific controller. A more general setup, however, could be implemented, wherein the packages' data simulation functions could be applied more broadly to any user-specified controller.

A number of avenues are open for further work on data squashing. As noted in Chapter 4, there is currently no guidance on how analyses should be conducted using squashed data (e.g., How does it fit into a project work flow? At what stage should variables be created or transformed?). There also are currently no asymptotic guarantees of squashing's performance.

The biggest barrier to the use of squashing, however, is its poor ability to scale in performance as the number of categorical variables increases. This is a difficult issue and substantial modifications to

the original squashing methodology may be required to surmount it. However, the efficiency of data squashing with a medium number of categories may be improved by a relatively simple modification of the way in which numeric variables are grouped within regions.

As discussed within Section 4.2.1, numeric variables are clustered within regions of constant categorical variables to ensure that the likelihood function is smooth enough for a good Taylor-series approximation in the numeric variables. Consequently, the rapid increase in regions associated with the addition of a new categorical variable results not as much from the additional combinations of categories themselves as from the near-exponential increase in regions that occurs when numeric variables are grouped within each combination. Efforts could be made to mitigate this inflation in regions without eliminating categorical combinations by regrouping categorical combinations prior to clustering numeric variables. In order to preserve the requirement that a smooth Taylor expansion of numeric variables is possible, regions should be grouped only if the moments are close to each other. In essence, this approach would empirically estimate regions where the likelihood function is approximately flat with respect to changes in categorical variable combinations. The efficiency of squashing would then improve to the degree that these approximately flat transitions occur.

The largest gains for the missing data extensions proposed in Chapter 4 stand to be made by improving e-squashing. For several parameters, e-squashing resulted in biased point estimates and anti-conservative standard errors. This was particularly apparent for the one numeric variable in the workers' compensation data set with a low-to-moderate amount of missing data. Two stages where these issues may be improved are 1) when numeric clusters are formed, and 2) when moments are calculated for each region to be later matched to the squashed sample.

Considering the first stage, the e-squashing procedure made the assumption that clusters in numeric variables were formed solely from complete observations. Observations with missing values were then assigned to multiple groups based on clustering probabilities. While this approach may work well when there is a high ratio of complete to incomplete observations, its performance will likely decline substantially when the ratio is low. A better approach would allow observations with missing values to contribute to the clustering process to the degree that their values are observed. A better approach still, however, would be to link the clustering and missing data mechanisms such that the moment calculations within each region would be less variable or prone to bias.

The second stage at which e-squashing could be improved is in the estimation of moments within each region. At present, moments for variables with missing data are calculated based on the observed values and then scaled to reflect the full sample size. For example, for a variable with missing values X, $\sum_{i=1}^{N} X_i^2$ is approximated by $\frac{N}{Nobs} E[\sum_{i=1}^{Nobs} X_i^2]$ with $Nobs$ values observed on N total observations. As the proportion of X missing increases, however, under MCAR the probability that $E[\sum_{i=1}^{Nobs} X_i^2] < \sum_{i=1}^{N} X_i^2$ also increases, as shown in Figure 5.1. Consequently, it may be appropriate to scale the expectation of some of the original data moments based on the proportion missing prior to imputing the missing moment values.

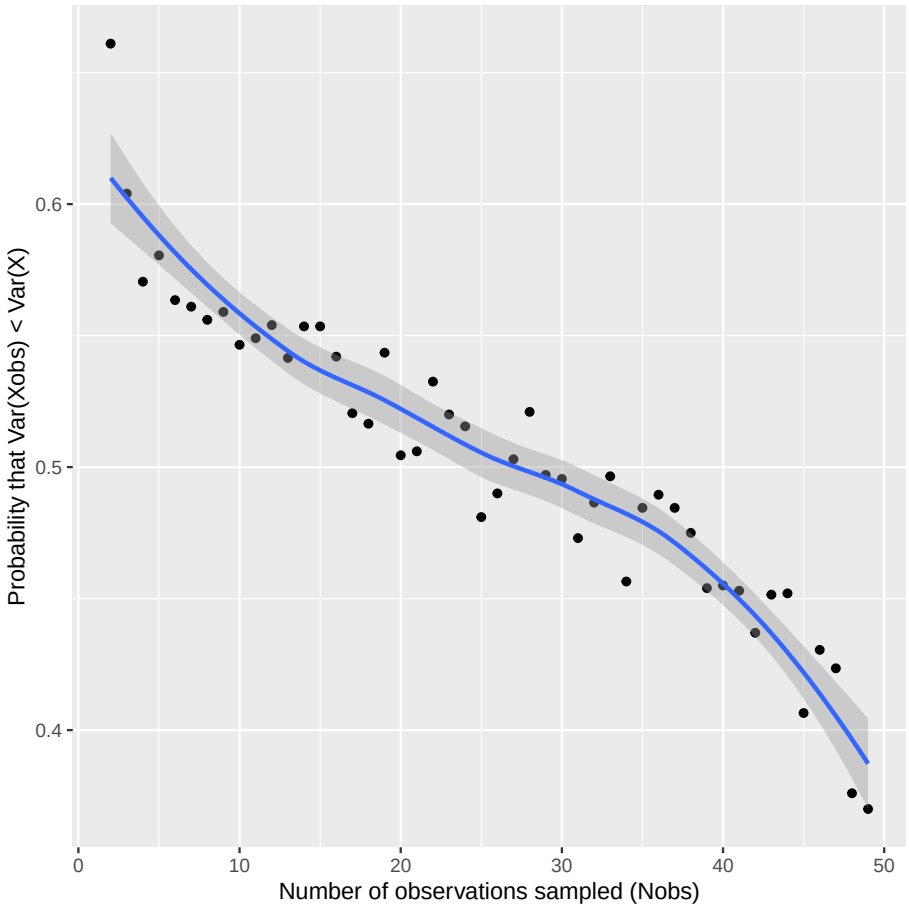

Figure 5.1: Probability that the complete-case variance will underestimate the sample variance as a function of complete case size. At each value along the x-axis, Nobs points drawn randomly from a population of N=50 observations distributed N($\mu = 0, \sigma = 5$). The sample variance of the Nobs points is calculated, and the process is repeated 2,000 times. The proportion of times in which Var(Xobs) is less than the variance of the 50 points is recorded on the y axis.

Absalom, A. R., De Keyser, R., and Struys, M. M. (2011). Closed loop anesthesia: Are we getting close to finding the holy grail? *Anesthesia and Analgesia*, 112(3):516–518.

Absalom, A. R., Glen, J. B., Zwart, G. J. C., Schnider, T. W., and Struys, Michel, M. R. F. (2016). Target-Controlled Infusion: A Mature Technology. *Anesthesia and Analgesia*, 122(1):70–78.

Absalom, A. R., Mani, V., De Smet, T., and Struys, M. M. (2009). Pharmacokinetic models for propofol- Defining and illuminating the devil in the detail. *British Journal of Anaesthesia*, 103(1):26–37.

Abuhelwa, A. Y., Foster, D. J. R., and Upton, R. N. (2015). ADVAN-style analytical solutions for common pharmacokinetic models. *Journal of Pharmacological and Toxicological Methods*, 73:42–48.

Al-Rifai, Z. and Mulvey, D. (2016). Principles of total intravenous anaesthesia: basic pharmacokinetics and model descriptions. *BJA Education*, 16(3):92–97.

Aspect Medical Systems (2005). *BIS VISTA Monitoring System*.

Baron, K. T. (2021). *mrgsolve: Simulate from ODE-Based Models*. R package version 0.11.0.

Bibian, S., Dumont, G. A., Huzmezan, M., and Ries, C. R. (2006). Patient variability and uncertainty quantification in anesthesia: Part II - PKPD uncertainty. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, 6(PART 1):555–560.

Brogi, E., Cyr, S., Kazan, R., Giunta, F., and Hemmerling, T. M. (2017). Clinical performance and safety of closed-loop systems: A systematic review and meta-analysis of randomized controlled trials. *Anesthesia and Analgesia*, 124(2):446–455.

Cascone, S., Lamberti, G., Titomanlio, G., and Piazza, O. (2013). Pharmacokinetics of Remifentanil: a three-compartmental modeling approach. *Translational medicine*, 7(4):18–22.

Choki, Y. and Suzuki, E. (2002). Iterative data squashing for boosting based on a distribution-sensitive distance. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2431 LNAI:86–98.

De Smet, T., Struys, M. M., Greenwald, S., Mortier, E. P., and Shafer, S. L. (2007). Estimation of optimal modeling weights for a bayesian-based closed-loop system for propofol administration using the bispectral index as a controlled variable: A simulation study. *Anesthesia and Analgesia*, 105(6):1629–1638.

Dodds, M. G., Hooker, A. C., and Vicini, P. (2003). Robust population pharmacokinetic experiment design. *Journal of Pharmacokinetics and Pharmacodynamics*, 32(1):33–64.

Dryden, P. E. (2016). Target-Controlled Infusions: Paths to Approval. *Anesthesia and Analgesia*, 122(1):86–89.

Dumont, G. A. and Ansermino, J. M. (2013). Closed-loop control of anesthesia: A primer for anesthesiologists. *Anesthesia and Analgesia*, 117(5):1130–1138.

DuMouchel, W. and Agarwal, D. K. (2003). Applications of sampling and fractional factorial designs to model-free data squashing. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 511–516.

DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. (1999). Squashing Flat Files Flatter. *AT&T Labs-Research*.

Eleveld, D. J., Colin, P., Absalom, A. R., and Struys, M. M. R. F. (2018). Pharmacokinetic-pharmacodynamic model for propofol for broad application in anaesthesia and sedation. *British Journal of Anaesthesia*, 120(5):942–959.

Eleveld, D. J., Proost, J. H., Cortínez, L. I., Absalom, A. R., and Struys, M. M. (2014). A general purpose pharmacokinetic model for propofol. *Anesthesia and Analgesia*, 118(6):1221–1237.

Fresenius Kabi USA LLC (2014). (Propofol) INJECTABLE EMULSION, USP.

Grey, R. E. (2021). System Failure: Essential Workers and COVID-19 in New York State.

Hajat, Z., Ahmad, N., and Andrzejowski, J. (2017). The role and limitations of EEG-based depth of anaesthesia monitoring in theatres and intensive care. *Anaesthesia*, 72:38–47.

Hartley, H. O. and Hocking, R. R. (1971). The Analysis of Incomplete Data. 27(4):783–823.

Ilyas, M., Butt, M. F. U., Bilal, M., Mahmood, K., Khaqan, A., and Ali Riaz, R. (2017). A Review of Modern Control Strategies for Clinical Evaluation of Propofol Anesthesia Administration Employing Hypnosis Level Regulation. *BioMed Research International*, 2017.

Jacobs, J. R. (1990). Algorithm for Optimal Linear Model-Based Control with Application to Pharmacokinetic Model-Driven Drug Delivery. *IEEE Transactions on Biomedical Engineering*, 37(1):107–109.

Kerr, A. (2021). Timeline: How covid-19 took over nyc.

KPMG (2020). The Impact of COVID-19 on Workers' Compensation. pages 1–34.

Kuck, K. and Johnson, K. B. (2017). The three laws of autonomous and closed-loop systems in anesthesia. *Anesthesia and Analgesia*, 124(2):377–380.

Kuhn, M. and Johnson, K. (2019). *Feature engineering and selection: A practical approach for predictive models*.

Loeb, R. G. and Cannesson, M. (2017). Closed-loop anesthesia: Ready for prime time? *Anesthesia and Analgesia*, 124(2):381–382.

Madigan, D., Raghavan, N., Dumouchel, W., Nason, M., Posse, C., and Redgeway, G. (2002). Likelihood-based data squashing: A modeling approach to instance construction. *Data Mining and Knowledge Discovery*, 6(2):173–190.

Mandel, J. E. and Sarraf, E. (2012). The variability of response to propofol is reduced when a clinical observation is incorporated in the control: A simulation study. *Anesthesia and Analgesia*, 114(6):1221–1229.

Marik, P. (2005). Propofol: Therapeutic Indications and Side-Effects. *Current Pharmaceutical Design*, 10(29):3639–3649.

Masui, K., Ph, D., Kira, M., Kazama, T., Ph, D., Hagihira, S., and Ph, D. (2009). Early Phase Pharmacokinetics but Not Pharmacodynamics Are Influenced by Propofol Infusion Rate. (4):805–817.

Morton, N. S. (2009). *Total Intravenous Anesthesia and Target-Controlled Infusion*. Number 4. Elsevier Inc., fourth edi edition.

Nascu, I., Krieger, A., Ionescu, C. M., and Pistikopoulos, E. N. (2015). Advanced Model-Based Control Studies for the Induction and Maintenance of Intravenous Anaesthesia. *IEEE Transactions on Biomedical Engineering*, 62(3):832–841.

Ng, S. (2017). Opportunities and Challenges: Lessons from Analyzing Terabytes of Scanner Data. *Advances in Economics and Econometrics*, pages 1–34.

Olvera-López, J. A., Carrasco-Ochoa, J. A., Martínez-Trinidad, J. F., and Kittler, J. (2010). A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143.

Owen, A. (2003). Data Squashing by Empirical Likelihood. *Data Mining and Knowledge Discovery*, 7(1):101–113.

Padula, F., Ionescu, C., Latronico, N., Paltenghi, M., Visioli, A., and Vivacqua, G. (2016). Inversion-based propofol dosing for intravenous induction of hypnosis. *Communications in Nonlinear Science and Numerical Simulation*, 39:481–494.

Padula, F., Ionescu, C., Latronico, N., Paltenghi, M., Visioli, A., and Vivacqua, G. (2017). Optimized PID control of depth of hypnosis in anesthesia. *Computer Methods and Programs in Biomedicine*, 144:21–35.

Pavlov, D., Chudova, D., and Smyth, P. (2000). Towards Scalable Support Vector Machines using Squashing. *KDD*, pages 295–299.

Phillips, A. T., Deiner, S., Mo Lin, H., Andreopoulos, E., Silverstein, J., and Levin, M. A. (2015). Propofol Use in the Elderly Population: Prevalence of Overdose and Association with 30-Day Mortality. *Clinical Therapeutics*, 37(12):2676–2685.

Rich, B. (2021). *linpk: Generate Concentration-Time Profiles from Linear PK Systems*. R package version 1.1.1.

Sahinovic, M. M., Struys, M. M., and Absalom, A. R. (2018). Clinical Pharmacokinetics and Pharmacodynamics of Propofol. *Clinical Pharmacokinetics*, 57(12):1–20.

Schiavo, M., Consolini, L., Laurini, M., Latronico, N., Paltenghi, M., and Visioli, A. (2021). Optimized feedforward control of propofol for induction of hypnosis in general anesthesia. *Biomedical Signal Processing and Control*, 66(September 2020):102476.

Schnider, T. W., Minto, C. F., Struys, M. M., and Absalom, A. R. (2016). The Safety of Target-Controlled Infusions. *Anesthesia and Analgesia*, 122(1):79–85.

Shafer, S. L. and Gregg, K. M. (1992). Algorithms to rapidly achieve and maintain stable drug concentrations at the site of drug effect with a computer-controlled infusion pump. *Journal of Pharmacokinetics and Biopharmaceutics*, 20(2):147–169.

Struys, M. M., Coppens, M. J., De Neve, N., Mortier, E. P., Doufas, A. G., Van Bocxlaer, J. F., and Shafer, S. L. (2007). Influence of administration rate on propofol plasma-effect site equilibration. *Anesthesiology*, 107(3):386–396.

Struys, M. M., De Smet, T., Glen, J. B., Vereecke, H. E., Absalom, A. R., and Schnider, T. W. (2016). The History of Target-Controlled Infusion. *Anesthesia and Analgesia*, 122(1):56–69.

Van Poucke, G. E., Bravo, L. J. B., and Shafer, S. L. (2004). Target controlled infusions: Targeting the effect site while limiting peak plasma concentration. *IEEE Transactions on Biomedical Engineering*, 51(11):1869–1875.

Wang, W., Hallow, K. M., and James, D. A. (2015). A tutorial on rxode: Simulating differential equation pharmacometric models in r. *CPT: Pharmacometrics & Systems Pharmacology*, 5(1):3–10.

Wesselink, E. M., Kappen, T. H., Torn, H. M., Slooter, A. J., and van Klei, W. A. (2018). Intraoperative hypotension and the risk of postoperative adverse outcomes: a systematic review. *British Journal of Anaesthesia*, 121(4):706–721.

West, N., Dumont, G. A., Van Heusden, K., Petersen, C. L., Khosravi, S., Soltesz, K., Umedaly, A., Reimer, E., and Ansermino, J. M. (2013). Robust closed-loop control of induction and maintenance of propofol anesthesia in children. *Paediatric Anaesthesia*, 23(8):712–719.

Xi, R. (2009). Statistical aggregation: Theory and applications. *ProQuest Dissertations and Theses*, (January):79.

Zhou, J. (2008). D-optimal minimax regression designs on discrete design space. *Journal of Statistical Planning and Inference*, 138(12):4081–4092.