university of groningen / faculty of behavioural and social sciences / research master

Master's thesis

Advancing Replication Study Decision-Making with DISCOURSE: Data-simulation via Iterative Stochastic Combinatorial Optimization Using Reported Summary Estimates

Name and initials:	Lortz S. A. J.
Student number:	S4381319
E-mail address:	s.a.j.lortz@student.rug.nl
First assessor:	Don van Ravenwzaaij
Second assessor:	Marieke Timmerman
Programme:	Research Master Behavioural and Social Sciences
Theme:	Understanding Societal Change
ECs:	30
Date:	29-6-2025
Word count:	9862

Are there deviations of the Master's thesis from the proposed plan?
☒No
☐Yes, please explain below the deviations

Advancing Replication Study Decision-Making with DISCOURSE: Data-simulation via Iterative Stochastic Combinatorial Optimization Using Reported Summary Estimates

Sebastian A. J. Lortz[a]

[a]Faculty of Behavioural and Social Sciences, University of Groningen, Groningen, Netherlands

Abstract

In response to the replication crisis and the frequent unavailability of raw data sets in psychology and the social sciences, I developed DISCOURSE - Data-simulation via Iterative Stochastic Combinatorial Optimization Using Reported Summary Estimates. This algorithmic framework simulates plausible data sets from published summary statistics when access to raw data is restricted. DISCOURSE comprises four modules tailored to various analysis contexts: Descriptives, ANOVA, Multiple Linear Regression, and Linear Mixed-Effects Regression. Each module's workflow initializes a candidate data set and through iterative cycles, the algorithm applies stochastic perturbations, heuristic adjustments, and permutation-based moves to alter values. An objective function evaluates the alignment with reported summary estimates and a simulated-annealing acceptance criterion with temperature schedules ensures robust exploration of the search space with convergence towards global optima. I validate the method on multiple benchmark data sets with known raw data, demonstrating that each module reproduces its target summary measures with very small discrepancies. I showcase the application of the algorithmic framework using published research articles and discuss the method's limitations. DISCOURSE is available as R package and comprehensive ShinyApp, offering researchers a tool for generating synthetic data sets solely from summary estimates.

Keywords: *Data Generation, Summary Statistics, Synthetic Data, Replication, Meta-heuristic Algorithms.*

Introduction

In the past decade, the field of psychology and the social sciences have faced a significant *replication crisis*. Many studies have failed to replicate, calling into question the robustness and reliability of scientific findings. Theoretical perspectives indicate that most published research claims are uninterpretable (Meehl, 1978, 1990) or incorrect (Ioannidis, 2005), and that reported associations are often inflated (Ioannidis, 2008). Empirically, the Open-Science-Collaboration (2015) found that only 36% of replication attempts in psychology resulted in statistically significant effects, compared to 97% in the original studies. Only about half of the initial effect sizes fall between the 95% confidence intervals of the replications. This striking crisis, largely driven by questionable research practices and publication bias (Schimmack, 2020), has fueled a widespread reassessment of research practices.

To address these concerns, systematic replication attempts have become more common and are in line with Open Science (Nosek et al., 2022). Replication studies offer benefits such as corroborating findings, deepening understanding of research fields, and serving as an educational tool (Derksen et al., 2024). Over the past decade, the frequency of replication studies and their associated fundings have grown substantially in the Netherlands and globally, reflecting the commitment to improve research practices (Derksen et al., 2024). Despite these efforts, challenges persist, especially when replication studies fail to confirm original findings.

A *failed replication* study raises important questions about *scientific error* (van Ravenzwaaij et al., 2023). Was the initial finding a true effect, but was the replication study underpowered or conducted under different conditions? Even with true effects, replication studies may fail due to small effect sizes and type II errors, as demonstrated by Schimmack (2020) who estimated a 43% replicability rate in social psychology. Was the initial finding a stochastic false positive, potentially influenced by well-documented problems such as p-hacking (Schimmack, 2020)? These scenarios underscore the difficulty of interpreting *failed replications* and the necessity to further analyse them.

When *raw data* from the original study is available, researchers can attempt to reproduce the results by reanalysing data to assess robustness (Derksen et al., 2024). Methods such as *multiverse* analyses (Steegen et al., 2016), in which various analytical decisions are tested, and *many-analyst* studies (Silberzahn et al., 2018), in which multiple researchers independently analyse the data and report results, help evaluate how analytical flexibility influences findings. These tools can guide decisions about whether further replication attempts are warranted. However, restricted access to *raw data* limits these approaches.

Raw data from initial studies is often unavailable (Wicherts et al., 2006). Miyakawa (2020), chief editor of *Molecular Brain*, reported that 23% of submitted manuscripts lacked *raw data*, and 97% of these did not provide it upon request. In replication research, where data requests occur years later, this hurdle is even higher. Researchers either do not share their data or cannot do so for various reasons. Barriers include ethical concerns (e.g., protecting participant privacy), data loss due to poor management or technical issues, and researchers' unwillingness to share data because of competitive pressures or fear of scrutiny.

Without *raw data*, researchers must rely solely on *summary statistics*, eliminating their ability to conduct reproductions or interrater analyses. This lack of access to plausible raw data complicates replication study decision-making: without detailed information on variability and analytic pipelines, researchers cannot assess the sensitivity of results to different analytic choices. Consequently, planning a replication becomes a blind exercise. Conflicting findings are often resolved through additional replication studies, but these are time-consuming, costly, and burdensome for participants raising ethical concerns. This inefficiency and lack of sustainability in replication research is especially concerning given the recent budget cuts in the Netherlands and abroad. By contrast, preliminary analyses on synthetic data based on reported summary estimates would enable informed judgments about the expected robustness of replication outcomes.

While in fields such as econometrics techniques for simulating population data with and without sample data exist (Lenormand & Deffuant, 2013; Templ et al., 2017), within psychology and related disciplines, methods relying solely on summary statistics have been primarily developed for fraud detection (Bordewijk et al., 2021; Hartgerink et al., 2019). For example, tools such as *statcheck* re-calculate p-values (Epskamp & Nuijten, 2014), the *GRIM* test identifies anomalies in reported means (Brown & Heathers, 2017), the *GRIMMER* test extends these principles to variance measures (Anaya, 2016), and the reference proportions for baseline p-values can be determined (Bolland et al., 2020). Additionally, univariate data distributions for integer variables can be reconstructed with the *SPRITE* algorithm (Heathers et al., 2018). While these techniques all share the principle of anchoring on available summary measures to assess the plausibility of reported measures, no method has been proposed to simulate entire raw data sets.

To address this methodological gap, this research aims to develop a practical tool for simulation of *raw sample data* from *summary statistics* to advance replication study *decision-making*. I pose the following research questions:

1. *Method Development*: How can *raw sample data* be simulated from diverse

summary statistics?

2. *Method Validation*: How close are the reported summary statistics to the summary statistics of the data simulated with the developed method?
3. *Application Development*: How can the method be made user-friendly and accessible?

The project outcome has substantial implications for scientific research and offers several key benefits. Improved *decision-making*: By generating synthetic data that aligns with reported summary estimates, researchers can conduct follow-up analyses to assess consistency across different analytical implementations and perform sensitivity checks to evaluate how variations in modelling choices influence outcomes. Additionally, they can explore other research questions with partly different variable combinations compared to the original study. This ensures that decisions to replicate are grounded in an assessment of robustness, rather than blind assumptions. Resource savings: Simulated data sets could reveal when reported effects are highly contingent on specific analytic decisions or exhibit excessive variability, signalling that a direct replication may be unlikely to yield definitive insights. In such cases, researchers can postpone or redesign replication studies, thereby avoiding unnecessary efforts, conserving financial resources, and reducing participant burden. Enhanced reproducibility: The approach facilitates goals of Open Science by promoting open research practices and enabling analysis even when *raw data* is unavailable.

Taken together, I address a critical need in the scientific community by offering a tool to simulate *raw data* from *summary statistics*. By integrating methodological development with practical utility, the project aims to advance a more efficient and sustainable research process.

Algorithm Framework

I introduce the DISCOURSE framework – **D**ata-simulation via **I**terative **S**tochastic **C**ombinatorial **O**ptimization **U**sing **R**eported **S**ummary **E**stimates. The primary scope of the algorithmic framework is to reconstruct complete data sets using only summary statistics, giving researchers a way - when raw data are unavailable - to inform replication study decision-making. The method is available as R package `discourse` and comprehensive ShinyApp at https://sebastian-lortz.github.io/discourse/.

DISCOURSE is underpinned by three core characteristics. It is *iterative*: The algorithm employs a cyclical process that continuously refines the simulated data. At each iteration, adjustments are made to reduce the discrepancy between the simulated and target summary statistics. It is *stochastic*: The method incorporates random sampling techniques to explore the data space effectively, ensuring a robust search for viable data configurations. It is

combinatorial: By transforming a high dimensional infinite search space into a finite optimization problem through the use of permutations, DISCOURSE efficiently navigates the multitude of potential data arrangements. Together, these features enable the algorithmic framework to simulate and adjust data until the generated summary statistics closely match the reported targets.

Modular Structure

The DISCOURSE framework is composed of four interchangeable optimization modules tailored to different data structures and statistical models, each following a similar high-level workflow. The modules are organized according to the dimensionality of the data to be generated. In the univariate setting, iterative adjustments are applied to a single vector, whereas in the multivariate context, the *moves* operate on an entire matrix containing multiple variables. An overview is presented in Table 1. These modules can operate independently or sequentially, depending on the specific requirements of the optimization context.

Table 1

Overview of Modules and Respective Functions of the R Package

Data Structure	
Univariate	Multivariate
Descriptives (`optim_vec()`)	Linear Regression (`optim_lm()`)
ANOVA (`optim_anova()`)	LME (`optim_lme()`)

Note. The dimensionality of the data structure refers to the data to be simulated.

High-Level Workflow

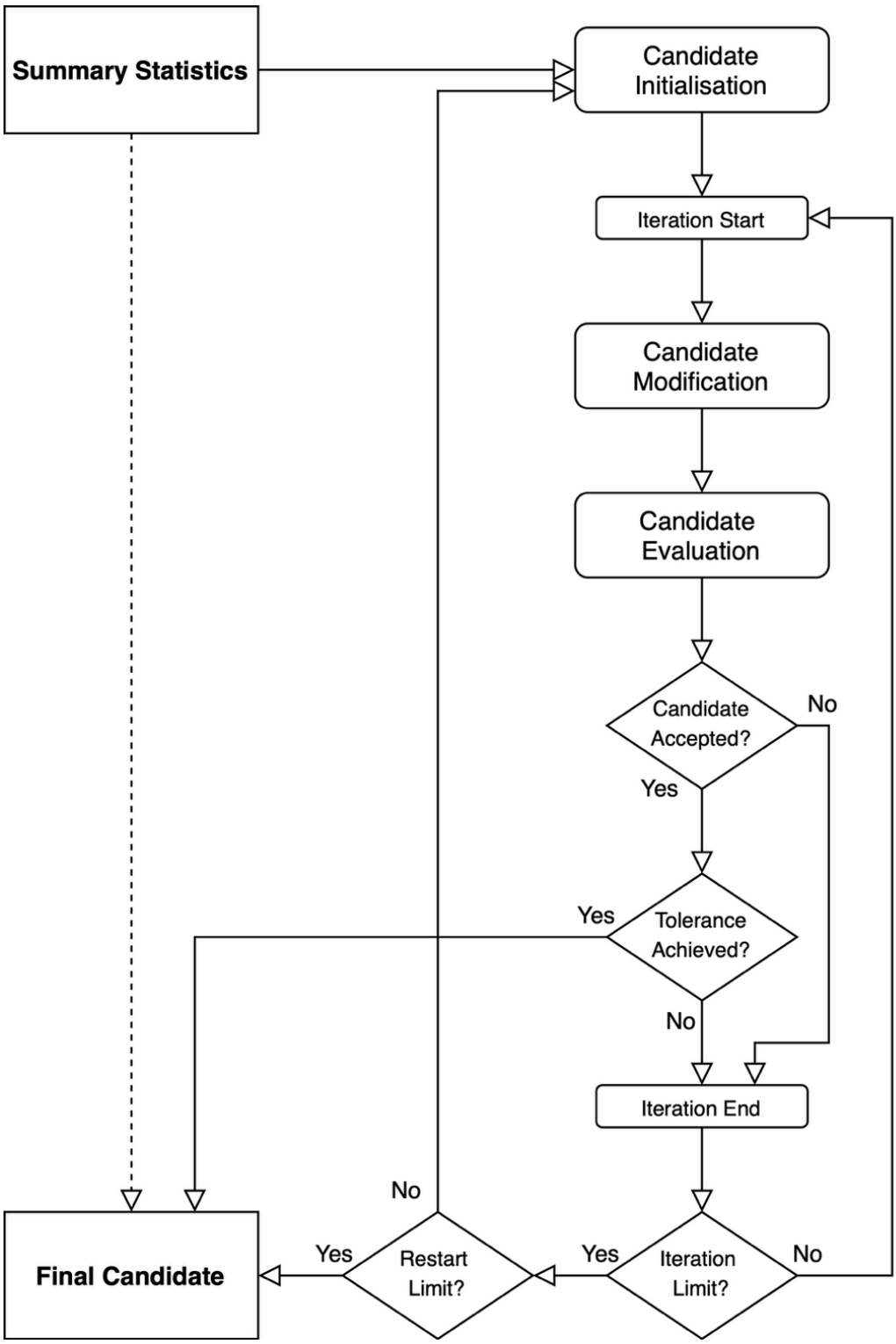
The process (see Figure 1) begins with the *candidate initialization*, thus, the creation of an initial simulated vector or matrix. The algorithm then iteratively refines the candidate by optimizing an objective function that quantifies the discrepancy between the summary statistics of the candidate and the reported targets. At each iteration the following two steps are performed.

Candidate Modification: Modifications to the data are produced through different types of moves (e.g., global and local; heuristic and stochastic) within the search space.

Candidate Evaluation: Each candidate is evaluated by an objective function and accepted based on a stochastic optimization criterion, ensuring that modifications progressively reduce the objective value.

Figure 1

High-Level Workflow of the DISCOURSE Framework



Note. From published summary statistics, DISCOURSE initializes a data set and then iteratively applies modifications and evaluates each candidate against reported summary estimates. The loop continues until convergence or iteration/restart limits are reached, yielding the final simulated data.

Convergence

The algorithm is deemed to have met the convergence criteria as soon as the best objective function score f_{best} falls below the user-specified $tolerance$. If convergence is not reached after $max\ iteration$ steps, the algorithm restarts (up to $max\ starts$ times) from the candidate with f_{best}. Only when all allowed iterations and restarts have been exhausted without achieving the $tolerance$ does the routine stop due to iteration limits rather than error criteria.

Optimization Modules

Univariate Distributions

Descriptives

Overview. Suppose a published article reports the sample $mean$, SD, sample size N, and states that data was collected on Likert scales, implying integer values within a defined $range$. The aim is to reconstruct an underlying data set whose values exactly reproduce these reported summary statistics. Therefore, a vector of length N is reconstructed so that its mean and SD match the prescribed targets, while also enforcing bounds and the requirement of discrete values (or continuous values). This quickly becomes a difficult optimization task. Because all N elements must be optimized at once, the problem resides in an N-dimensional search space (see Appendix A for details). Moreover, the resulting objective surface can contain numerous local minima, where reliable gradient information is difficult to obtain. Under these conditions, the application of standard gradient-based optimizers to find global solutions typically fails (Conn et al., 2009). Hence my choice of meta-heuristic optimization strategies that can explore such high-dimensional landscapes without relying on derivatives (for an overview of meta-heuristic search procedures I refer to Abdel-Basset et al. (2018)).

The descriptives module employs two complementary meta-heuristic algorithms. For integer variables I use a customized simulated-annealing framework (Kirkpatrick et al., 1983) that blends *stochastic* with problem-specific *heuristic moves* to efficiently explore global minima. For continuous variables I leverage particle-swarm optimization (PSO) a population-based algorithm that balances exploration and exploitation by tracking both each particle's own best position and the swarm's global best (Kennedy et al., 1995). Both approaches optimize the same unified objective function - minimizing a weighted error on the sample $mean$ and SD - under a single framework, and each is tuned via algorithm parameters to provide robust, automated optimization without requiring gradients or convexity assumptions. The following sections describe in detail the candidate initialization,

modification strategies, objective evaluation, acceptance rules, convergence criteria and parameter overview (see Table 2).

Candidate Initialization. Candidate initialization proceeds as follows. When dealing with integer variables, the algorithm generates each value of the candidate by drawing integers uniformly from the set of all integers between the specified lower and upper bounds. However, this uniform sampling can be replaced by a discrete distribution: the user may supply a vector of probabilities assigning a probability to each integer in the $range$, and the initialization routine will sample according to those probabilities rather than equal ones.

Continuous candidates are initialized by generating an initial position vector, where each element is drawn uniformly at random within the $range$. That vector seeds the first particle, the remaining particles in the default swarm of size

$$s = \lfloor 10 + 2 \times \sqrt{N} \rfloor$$

are likewise placed uniformly within the same box (Bendtsen, 2022). Initial movement speeds and directions of particles are then seeded in accordance with the SPSO-2007 specification, typically as random vectors whose magnitudes do not exceed the diagonal length of the search region, ensuring the swarm begins with broad but feasible exploration (Bendtsen, 2022; Clerc, 2012).

Hence, the procedure for both integer and continuous vectors does not impose an a priori assumption of normality on the generated data. Nonetheless, the algorithm process may converge toward a Gaussian distribution, if such a plausible data distribution exists given the parameter settings.

Candidate Modification. For integer variables with possible values within allowable bounds the algorithm applies two types of moves: a purely *stochastic move* and a *heuristic move*. The *stochastic move* selects one random observation and alters its value (respecting the variable's bounds), thus changing the $mean$ and SD. The *heuristic move* selects two random observations, increases one by v and decreases the other by v (within allowable bounds), thus preserving the mean while altering the SD (see Appendix B for details). For continuous variables the candidate modification follows a standard PSO routine (see Appendix B)

Candidate Evaluation. Candidate evaluation is formalized by an objective function $f(x)$ that measures the weighted sum of root mean squared normalized errors in the sample mean and standard deviation of a candidate vector x. Let $\bar{x}$ and s denote the candidate's sample mean and standard deviation, each rounded to the same precision (number of decimal places) as the target mean μ and target standard deviation σ. Define the relative deviations

$$\Delta_\mu = \frac{\bar{x}-\mu}{\max(|\mu|,\varepsilon)}, \Delta_\sigma = \frac{s-\sigma}{\max(\sigma,\varepsilon)},$$

where ε is a small constant to avoid division by zero. The objective function is

$$f(x) = \omega_1\sqrt{\Delta_\mu^2} + \omega_2\sqrt{\Delta_\sigma^2},$$

with user-defined weights ω_1 and ω_2 balancing the emphasis on mean versus standard deviation error. Lower values of $f(x)$ correspond to closer agreement with the prescribed targets. This single scalar score both drives the acceptance criterion in the simulated-annealing branch and serves as the fitness measure for personal- and global-best updates in the PSO routine.

Acceptance Criteria. For integer variables, candidate updates follow a simulated-annealing scheme and convergence (see Appendix C). In the continuous variable branch, acceptance as well as convergence is governed by a PSO routing (see Appendix C).

Convergence Criteria. For integer variables, the algorithm is deemed to have met the convergence criteria as soon as the best objective function score f_{best} falls below the user-specified $tolerance$. If convergence is not reached after $max\ iteration$ steps, the algorithm restarts (up to $max\ starts$ times) from the candidate with f_{best}. Only when all allowed iterations and restarts have been exhausted without achieving the $tolerance$ does the routine stop due to iteration limits rather than error criteria.

For continuous variables, the convergence relative to the $tolerance$ is managed by the underlying `psoptim()` call. As particles evolve, f_{gbest} is compared at each iteration to the $abstol$ parameter (equal to the user's $tolerance$). Therefore, if $f_{gbest} < tolerance$, the optimization converges. Otherwise, the algorithm stops when it either (a) reaches the maximum number of iterations ($maxit\ PSO$) or (b) tracks $\frac{maxit\ PSO}{3}$ consecutive iterations without improvement of f_{gbest}.

Table 2

Parameter Overview for the Descriptives Module: Reported Summary Estimates, Algorithm Hyperparameters in the ShinyApp and Package-Only Hyperparameters

Parameter	Description
N	The sample size of the data and thus, the length of the target vectors.
Target Mean	The vector containing reported means of the variables.
Target SD	The vector containing the reported standard deviations of the variables.

Range	The matrix containing minimum (first row) and maximum (second row) for each variable.
Integer	The vector containing a logical statement for each variable; TRUE: the variable has integer values, FALSE: the variable has continuous values.
Tolerance	The threshold for the weighted objective function value below which the optimization will stop.
Max Iterations	The maximum number of iterations the algorithm will run each time it restarts and for each integer variable.
Maxit PSO	The maximum number of iterations the algorithm will run for each continuous variable.
Temperature	The starting temperature for the simulated annealing, which sets the initial likelihood of accepting worse solutions in the first start.
Cooling Rate	The factor by which the temperature is multiplied after each iteration, governing how quickly the algorithm reduces its acceptance of worse solutions.
Max Starts	The maximum number of times the optimization algorithm will restart from the current best solution using reduced initial temperatures.
Parallel	Enable the algorithm to execute on a parallel backend for improved performance.
Weights	The weights multiplied with the mean and standard deviation term in the objective function; adjusting these values can steer the optimization toward preferred trade-offs and improve performance. In the ShinyApp the R function `weights_vec()` can automatically compute quasi-optimal objective function weights from each term's initial contribution, using a Monte-Carlo routine.
Prob Heuristic	The probability of applying a heuristic vs. a stochastic search move.
Init Probs	The sampling probabilities for integer moves to use prior expectations.
Progress Bar	Enable the text progress bar during optimization.
EPS	The small constant to avoid division by zero in the objective function.
Check Grim	Perform a GRIM check on target mean for integer variables.
Min Decimals	Minimum decimal places for target values' trailing zeros.

Note. **Bold** indicates reported summary estimates; *italic* indicates hyperparameters adjustable in the ShinyApp; plain text indicates hyperparameters only configurable via the R package.

Example. Robin knows the data ($N = 5$) come from a Likert scale ($range\ 1-7$) with reported $mean\ \mu = 4$ and $SD\ \sigma = 2$. They set the algorithm's parameters to

$Weights: \omega_1 = 1,\ \omega_2 = 1$

$Tolerance = 0.001$

$Temperature = 1$

$Cooling\ Rate = .9$

All other parameters $= default$.

DISCOURSE then optimizes the data vector $X^{(i)}$, where i is the current iteration, to minimize the objective function f and match the reported summary estimates.

Candidate Initialization. The algorithm initializes the first data

$$X^{(0)} = [1, 7, 3, 5, 2] = X_{best}\ ,$$

which results in

$$mean = 3.6\ , SD = 2.4\ ,$$

$$f^{(0)} = 1 \times \sqrt{\left(\frac{3.6-4}{4}\right)^2} + 1 \times \sqrt{\left(\frac{2.4-2}{2}\right)^2} = 0.30 = f_{best} = f_{current}\ ,$$

Iteration 1.

Modification. A *stochastic move* is randomly chosen. Index 2 (value 7) is selected, and a new value is drawn from the within-range values $(1, \ldots, 6) \rightarrow 4$.

The data is updated accordingly

$$X^{(1)} = (1, 4, 3, 5, 2)\ .$$

Candidate Evaluation. The $mean$ and SD is computed

$$mean = 3.0\ , SD = 1.6\ ,$$

and the value of the objective function is

$$f^{(1)} = 1 \times \sqrt{\left(\frac{3.0-4}{4}\right)^2} + 1 \times \sqrt{\left(\frac{1.6-2}{2}\right)^2} = 0.45\ .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(1)} - f_{current} = +0.15\ .$$

Since the new candidate data fits worse, it is accepted with probability

$$P = e^{(-\Delta f / T^{(0)})} = 0.86\ .$$

Here the draw succeeds, so the candidate is accepted

$$X_{current} = X^{(1)} \text{ and } f_{current} = f^{(1)}\ ,$$

While the best candidate remains

$$X_{best} = X^{(0)} \text{ and } f_{best} = f^{(0)}\ .$$

Cooling Schedule. The new temperature T is

$$T^{(1)} = 1 \times 0.9 = 0.9 \, .$$

Iteration 2.

Candidate Modification. A *heuristic move* is randomly chosen. Index 2 (value 4) and index 4 (value 5) are selected and the SD needs to be increased. To decrease index 2 and increase index 4, $\delta = 2$ is randomly sampled from the possible values $(1,2)$, and the data is updated

$$X^{(2)} = (1, 2, 3, 7, 2) \, .$$

Candidate Evaluation. The $mean$ and SD is computed

$$mean \; = \; 3.0 \, , SD \; = \; 2.3 \, ,$$

and the objective is

$$f^{(2)} = 1 \times \sqrt{\left(\frac{3.0-4}{4}\right)^2} + 1 \times \sqrt{\left(\frac{2.3-2}{2}\right)^2} = 0.40 \, .$$

Note how the $mean$ remains unchanged while the SD is pushed closer to the specified target.

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(2)}\text{-}f_{current} = -0.05 \, ,$$

Since the new candidate data fits better, it is unconditionally accepted

$$X_{current} = X^{(2)} \text{ and } f_{current} = f^{(2)} \, ,$$

While the best candidate remains

$$X_{best} = X^{(0)} \text{ and } \; f_{best} = f^{(0)} \, .$$

Cooling Schedule. The new temperature is

$$T^{(2)} = 0.9 \times 0.9 = 0.81 \, .$$

Subsequent Iterations.

Subsequent iterations continue the same until either $tolerance$ or $max\ iterations$ are reached**.**

Final Candidate. In this example, further updates eventually yield

$$X^{(*)} = (2, 2, 4, 6, 6) = X_{best}$$

and the algorithm converges with an exact match

$$mean \; = \; 4.0 \, , SD \; = \; 2.0 \, ,$$

$$f^{(*)} = 1 \times \sqrt{\left(\frac{4.0-4}{4}\right)^2} + 1 \times \sqrt{\left(\frac{2.0-2}{2}\right)^2} = 0 \, .$$

Finally, the data X_{best} is returned to Robin.

ANOVA

Overview. Suppose a published article reports the design and results of an ANOVA. It states the number of factors (between, within) with $levels$, F statistics, $group\ means$, sample size N, and that the data was collected on Likert scales, implying integer values within a defined $range$. The aim is to reconstruct an underlying data set whose values exactly reproduce these reported summary estimates. In the ANOVA module, the data include one or more categorical factors and a single dependent variable (outcome). The grouping structure is represented in the design matrix M, in which a column is assigned to each factor and filled with the respective $levels$ (e.g., 1, 2, 3, …) for each observation. I can reconstruct M directly from reported summary statistics whenever the $subgroup\ sizes\ n_j$ are reported. If only the total sample size N is provided, a balanced design is assumed. Consequently, the single data vector to be simulated and optimized is the outcome variable Y, and the goal is to determine Y (in conjunction with M) so that the resulting ANOVA F statistics match the predefined target values (for parameter overview see Table 3). This optimization is challenging because it occurs in an N-dimensional search space (see Appendix A for details). Notably, rearranging the values of Y inside a group does not affect F, but swapping values between groups does. As in the descriptive case, the resulting objective surface is rugged and lacks reliable gradient information. For these reasons, I employ derivative-free meta-heuristic methods capable of navigating such complex, high-dimensional landscapes (Abdel-Basset et al., 2018; Conn et al., 2009).

The ANOVA module employs a bespoke simulated-annealing framework (Kirkpatrick et al., 1983) with a search move that is both *heuristic* and *stochastic* to locate global minima. In contrast to the descriptive module's PSO application, it provides two specialized moves for integer and continuous outcomes, each respecting the fixed grouping structure that standard PSO updates cannot accommodate. The algorithm minimizes f defined as the error between the computed and target ANOVA F statistics and relies on a set of tuning parameters to deliver robust, automated optimization without any need for gradient information or convexity assumptions.

Candidate Initialization. To begin with, the design matrix M is created based on N, $levels$, $subgroup\ size\ n\_j$, and $factor\ type$ so that it mirrors the reported design. Please note that M is constructed for ANOVA (each column encodes a categorical factor with multiple levels) rather than as a dummy-coded predictor matrix for ordinary least-squares (OLS) regression. Once M is fixed, the candidate vector Y is initialized to reproduce each

supplied target $group\ mean\ tMean_j$ exactly. For continuous data, the algorithm simply repeats $tMean_j$ exactly n_j times; for integer data, it assigns the two nearest integer values in proportions that yield the exact mean (see Appendix E for details).

Hence, the procedure for both integer and continuous vectors does not impose an a priori assumption of normality on the generated data. Nonetheless, the algorithm process may converge toward a Gaussian distribution, if such a plausible data distribution exists given parameter settings.

Candidate Modification. Candidate modifications in the ANOVA module employ a single heuristic-stochastic move that leaves every group's mean exactly intact but perturbs individual values within groups to adjust their variability. While being conceptually similar to the descriptives module, it differs from it, as it fully preserves the grouping structure defined by M, optimizes only a single objective (F values), employs more stochastic elements, and applies to integer and continuous data. At each iteration, one group is selected at random and two distinct observations within that group are sampled. Denoting their current values as x_i and x_j and the allowable $range$ as $[L, U]$, feasible adjustment bounds

$$\delta_{min} = \max\big(L - x_i,\ x_j - U,\ -\varepsilon\big)\ , \quad \delta_{max} = \min\big(U - x_i,\ x_j - L, \varepsilon\big)\ ,$$

are computed so that adding δ to x_i and subtracting δ from x_j will keep both values within $[L, U]$. ε governs the magnitude of modification and is implied by the user's setting of the $max\ step$ parameter

$$\varepsilon = (U - L) \times Max\ Step\ ,$$

to prevent early convergence due to tremendous modification of single values (this would result in a distribution with many values identical to the $group\ mean$ and a few extreme values, with nothing in between). Subsequently, δ is sampled uniformly from the continuous or integer space $[\delta_{min}, \delta_{max}]$ and added to or subtracted from x_i or x_j, respectively. If $\delta_{min} > \delta_{max}$ no valid modification is possible, and the candidate is left unchanged. Because each move conserves the sum of the two selected values, the $group\ mean$ remains exact. In contrast to the descriptive module, where unit increments ($\delta = 1$) are applied across the entire vector to nudge the SD towards its target, this ANOVA search move allows variable step sizes up to ε, confines perturbations to within-group pairs, and stochastically modifies integer and continuous data. Thus, repeated applications enable effective exploration of the constrained search space without the need of explicitly optimizing for $group\ means$ themselves.

Candidate Evaluation. Candidate evaluation is governed by an objective function that computes the root mean square error (RMSE) between the target F values F^{target} and the ANOVA F statistics $\hat{F}$ for a candidate outcome vector, after rounding the computed F statistics to the same precision as the targets to prevent spurious inaccuracy. Because there is only a single component to match (F values), no additional normalization or weighting is required, unlike in the descriptive module. For any design that is unbalanced, involves specified contrasts, or includes within-subject factors, I assemble the full data frame (including subject identifiers when required), estimate the relevant $\hat{F}$ statistics using ANOVA, and compute

$$f(x) = \sqrt{\frac{1}{E}\sum_{e=1}^{E}\left(\hat{F}_e - F_e^{target}\right)^2},$$

Where E is the number of effects e of interest. In the special case of a balanced, between-subjects design with no contrasts I can derive a computational less demanding f. From these designs it follows

$$F_e = \frac{MS_e}{MSE}, \qquad \text{and} \qquad SS_I = SS_{II} = SS_{III},$$

where MS_e is the mean square of the effect e, MSE the mean square error of the model, and SS the sum of squares, I precompute the target MSE^{target} from the sequential decomposition by fitting a series of weighted OLS regressions that add one factor at a time to an intercept-only model. The reduction in weighted residual sum of squares gives the SS_e of the effect and it is possible to calculate

$$MSE^{target} = \frac{SS_e/df_e}{F_e^{target}},$$

and I define the objective function

$$f(x) = \sqrt{\left(\frac{\sum_{j=1}^{G}(n_j-1)s_j^2}{\sum_{j=1}^{G}(n_j-1)} - MSE^{target}\right)^2},$$

where s is the standard deviation in group j.

Because each evaluation either minimizes the RMSE of rounded F values or the difference in MSE, the optimizer directly targets the quantities of interest for all ANOVA designs without relying on gradient information or convexity.

Acceptance Criteria. For integer and continuous data, candidate updates follow a simulated-annealing scheme (see Appendix B for details).

Table 3

Parameter Overview for the ANOVA Module: Reported Summary Estimates, Algorithm Hyperparameters in the ShinyApp and Package-Only Hyperparameters

Parameter	Description
N	The sample size of the data and thus, the number of subjects.
Levels	The vector containing the number of levels for each factor.
Factor Type	The vector containing if a factor is 'between' or 'within' subjects.
Subgroup Size	Optional sizes of each between-subjects group for unbalanced designs.
Group Means	The vector containing reported means of all subgroups.
Target F List	List with: *F*, target F statistics vector; *Effect*, character vector of effect names.
DF Effects	The vector with degrees of freedom for each target effect.
Range	The vector containing minimum and maximum of the outcome.
Integer	Weather the outcome is an integer variable.
Type SS	The type 2 or type 3 sum of squares to be used
Formula	The model formula of the ANOVA.
Tolerance	The threshold for the objective function value below which the optimization will stop.
Max Iterations	The maximum number of iterations the algorithm will run each time it restarts.
Temperature	The starting temperature for the simulated annealing, which sets the initial likelihood of accepting worse solutions in the first start.
Cooling Rate	The factor by which the temperature is multiplied after each iteration, governing how quickly the algorithm reduces its acceptance of worse solutions.
Max Starts	The maximum number of times the optimization algorithm will restart from the current best solution using reduced initial temperatures.
Max Step	The proportion of the *range* governing the maximum magnitude of modification in an iteration.
Parallel Start	The number of independent runs (parallel or sequential) to simulate multiple data sets.
Best Solution	Return the best matching data set only (otherwise: return all data sets).

Progress Bar	Enable the text progress bar during optimization.
Check Grim	Perform a GRIM check on target mean for integer variables.
Target F List	List with: *Contrast*, optional contrast formula; *Contrast Method*, optional contrast method name.
Min Decimals	Minimum decimal places for target values' trailing zeros.

Note. **Bold** indicates reported summary estimates; *italic* indicates hyperparameters adjustable in the ShinyApp; plain text indicates hyperparameters only configurable via the R package.

Example. Robin knows the data come from a Likert scale ($range\ 1-7$) with reported $group\ means\ \mu_1 = 2, \mu_1 = 3, \mu_1 = 4$ and $subgroup\ size\ n\ =\ 3$ ($N = 9$). The fixed-effect ANOVA results are reported with $F^{target}(2{,}6) = 3$. They set the algorithm's parameters to

$Tolerance = 0.001$

$Temperature\ =\ 1$

$Cooling\ Rate\ =\ 0.9$

All other parameters $= default$.

DISCOURSE then optimizes the data vector $Y^{(i)}$, where i is the current iteration, to minimize the objective function f and match the reported summary estimates. The target MSE is computed for the optimization

$$MSE^{target} = \frac{SS_e/df_e}{F_e^{target}} = \frac{3(2-3)^2+3(3-3)^2+3(4-3)^2}{2\times3} = 1\ .$$

Candidate Initialization. The algorithm generates the factor structure

$$M = (1,1,1,2,2,2,3,3,3)\ ,$$

and initializes the first data

$$Y^{(0)} = (2,2,2,3,3,3,4,4,4) = Y_{best}\ ,$$

which results in

$$\mu_1 = 2, \mu_1 = 3, \mu_1 = 4\ \ ,$$

$$f^{(0)} = \sqrt{(0-1)^2} = 1 = f_{best} = f_{current}\ .$$

Iteration 1.

Candidate Modification. Group 1 is randomly picked and index 1 (value 2) and index 3 (value 2) are selected. The bounds of possible values to increment index 1 and decrement index 2 are determined $\delta_{min} = -1$ and $\delta_{max} = 1$. $\delta = 1$ is sampled and added to index 1 and subtracted from index 2. The data is updated accordingly

$$Y^{(1)} = (3, 2, 1, 3, 3, 3, 4, 4, 4) .$$

Candidate Evaluation. The value of the objective function is computed

$$f^{(1)} = \sqrt{\left(\frac{\sum_{j=1}^{G}(n_j-1)s_j^2}{\sum_{j=1}^{G}(n_j-1)} - MSE^{target}\right)^2} = \sqrt{\left(\frac{(3-1)\times1+(3-1)\times0+(3-1)\times0}{(3-1)+(3-1)+(3-1)} - 1\right)^2} = 0.67 .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(1)}\text{-}f_{current} = -0.33 .$$

Since the new candidate data fits better, it is unconditionally accepted

$$Y_{current} = Y^{(1)} \text{ and } f_{current} = f^{(1)} ,$$

and the best data is updated

$$Y_{best} = Y_{current} \text{ and } f_{best} = f_{current} .$$

Cooling Schedule. The new temperature T is

$$T^{(1)} = 1 \times 0.9 = 0.9 .$$

Iteration 2.

Candidate Modification. Group 2 is randomly picked and index 4 (value 3) and index 6 (value 3) are selected. The bounds of possible values to increment index 1 and decrement index 2 are determined $\delta_{min} = -2$ and $\delta_{max} = 2$. $\delta = 1$ is randomly sampled and added to index 1 and subtracted from index 2. The data is updated accordingly

$$Y^{(2)} = (3, 2, 1, 4, 3, 2, 4, 4, 4) .$$

Candidate Evaluation. The value of the objective function is computed

$$f^{(2)} = \sqrt{\left(\frac{(3-1)\times1+(3-1)\times1+(3-1)\times0}{(3-1)\times3} - 1\right)^2} = 0.33 .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(2)}\text{-}f_{current} = -0.34 ,$$

Since the new candidate data fits better, it is unconditionally accepted

$$X_{current} = X^{(2)} \text{ and } f_{current} = f^{(2)} ,$$

and the best data is updated

$$Y_{best} = Y_{current} \text{ and } f_{best} = f_{current} .$$

Cooling Schedule. The new temperature is

$$T^{(2)} = 0.9 \times 0.9 = 0.81 .$$

Iteration 3.

Candidate Modification. Group 3 is randomly picked and index 8 (value 4) and index 9 (value 4) are selected. The bounds of possible values to increment index 8

and decrement index 9 are determined $\delta_{min} = -3$ and $\delta_{max} = 3$. $\delta = -1$ is randomly sampled and added to index 8 and subtracted from index 9. The data is updated accordingly

$$Y^{(3)} = (3, 2, 1, 4, 3, 2, 4, 3, 5)\ .$$

Final Candidate. In this example, the $tolerance$ is reached, and the algorithm converges with an exact match

$$f^{(3)} = \sqrt{\left(\frac{(3-1)\times1+(3-1)\times1+(3-1)\times1}{(3-1)\times3} - 1\right)^2} = 0\ ,$$

$$Y^{(3)} = Y_{best}\ .$$

Finally, the data Y_{best} is returned to Robin.

Multivariate Distributions

Multiple Linear Regression

Overview. Suppose a published article reports the results of a multiple linear regression. Besides the descriptive statistics, it states the bivariate $correlations$, unstandardized $regression\ coefficients$ with standard errors (SE), and the regression $equation$ or model. The aim is to reconstruct an underlying data set whose values exactly reproduce these reported summary estimates. Thus, the multiple linear regression (LM) module reconstructs the full joint data matrix $W = (X, Y)$, where the design matrix X $(N \times p)$ may include p continuous predictors, dummy-coded variables, and/or interaction terms, and the outcome vector Y $(N \times 1)$ contains the criterion values. The data simulation needs to be performed so that the estimated $means$, SD, $regression\ coefficient$ (b) with their SE, and all pairwise $correlations$ (r) exactly match the supplied target values (for parameter overview see Table 4). Beyond the curse of high dimensionality (see Appendix A for details), the problem is inherently multi-objective: a modification that improves the match with one target (e.g., a predictor $mean$) can simultaneously worsen fit on another (e.g., a $regression\ coefficient$ or $correlation$), producing a highly nonconvex objective surface with deep local minima and lack of gradient information. As a result, the optimization techniques that sufficed for the descriptive and one-way ANOVA modules cannot converge here, and even advanced meta-heuristic algorithms routinely fail to locate a global solution that satisfies all objectives simultaneously.

To overcome these challenges, the LM module transforms the original mixed-numeric optimization into a sequential, two-stage procedure. Because each column's sample $mean$ and SD in the joint data matrix W are invariant under any permutation of its entries, whereas

b, SE, and r, are not, I first apply the descriptive module to simulate each column j of W ($j = p + 1$) so that $mean$ and SD of the predictor(s) and criterion match the target values. With these marginal distributions fixed, I hold Y constant and restate the optimization problem as finding within-column permutations π_j of each X_j that minimizes the deviation of the resulting b, SE, and r from their supplied summary estimates. Thus, the criterion values are set while predictor values are permuted. This makes the meta-heuristic simulated annealing framework (Kirkpatrick et al., 1983) an ideal choice: it can efficiently approximate the global optimum in high-dimensional permutation problems (Gutjahr, 2011; Korte & Vygen, 2018).

Candidate Initialization. In a first step, the user needs to apply the descriptive module to predictor(s) and criterion variable to produce an optimized data matrix W which matches the $target\ means$ and SD. This matrix is subsequently handed off to the LM module, where candidate initialization simply splits W into Y (the criterion) and X (the initial candidate). By anchoring on W, I eliminate the need to re-optimize descriptive statistics and smoothly transition into the LM module, and thus, the combinatorial optimization.

Candidate Modification. Candidates are modified by stochastic search moves that permute X either globally across all rows or locally between two rows in a single column. While *global moves* enhance the algorithm's ability to quickly explore diverse regions of the solution space and escape suboptimal configurations effectively, *local moves* generate fine-grained adjustments that progress slowly towards optima. At each iteration t, the algorithm decides whether to perform a *global* or a *local move* according to a fixed probability $prob\ global$ of choosing a *global move* (default is $.05$). The algorithm performs a *global move* by choosing a random permutation π of $\{1, \dots, N\}$ and setting

$$X^{(t+1)} = \pi\left(X^{(t)}\right),$$

which reorders all N rows at once while keeping Y constant. Otherwise, a *local move* is executed: a predictor column j and two distinct row indices $i \neq k$ are selected and the swap

$$X_{i,j}^{(t+1)} = X_{k,j}^{(t)}, \qquad X_{k,j}^{(t+1)} = X_{i,j}^{(t)},$$

leaves every other entry unchanged. As both move types only permute entries within each column, they exactly preserve the estimated $means$ and SD while combining global jumps with fine local refinements.

Candidate Evaluation. Each candidate is scored by an objective function $f(W)$ that combines the RMSE of the candidate's correlations $\hat{r}$, regressions $\hat{b}$, and SE $\widehat{SE}$, against the target summaries r^{target}, b^{target}, and SE^{target}. First, I compute the candidate correlation matrix from the joint $W(X, Y)$ matrix via linear matrix algebra. I use the upper triangle values

and round it to the same precision as its target, yielding the vector $\hat{r} = (\hat{r}_1, \ldots, \hat{r}_m)$, where m is the number of elements in the upper triangle of the correlation matrix. The correlation RMSE is

$$RMSE_r = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r} - r^{target})^2} ,$$

where missing targets are omitted from the summations. To obtain the estimated coefficient vector $\hat{b}$ and SE vector $\widehat{SE}$ I calculate the OLS solution given W (X, Y) via matrix algebra (see Appendix D for details). To prevent inflated deviance due to spurious precision, $\hat{b}$ and $\widehat{SE}$ are rounded to the same precision as the reported summary estimates. Finally, I compute

$$RMSE_{b,SE} = \sqrt{\frac{1}{q}\sum_{j=1}^{q}\left[\left(\hat{b}_j - b_j^{target}\right)^2 + \left(\widehat{SE}_j - SE_j^{target}\right)^2\right]},$$

omitting missing targets, and I define the objective function

$$f(W) = \omega_r RMSE_r + \omega_{b,SE} RMSE_{b,SE} ,$$

where the weights reflect relative importance of matching $correlations$ versus $regression/SE$ outputs. The value of $f(W)$ thus provides a unified measure of a candidate's deviation from all reported summary statistics, with lower values indicating closer agreement across $correlations$, $regression\ coefficients$, and SEs, without calculating gradients or posing convexity assumptions.

Acceptance Criteria. For integer and continuous data, candidate updates follow a simulated-annealing scheme (see Appendix C for details).

Table 4

Parameter Overview for the LM Module: Reported Summary Estimates, Algorithm Hyperparameters in the ShinyApp and Package-Only Hyperparameters

Parameter	Description
Correlation	The vector containing the upper triangle of the reported bivariate correlation matrix.
Regression-Coefficient	The vector containing the target regression coefficients including the intercept.
SE	The vector containing the target SE of regression coefficients including the intercept.
Equation	The formula of the reported regression model.

Tolerance	The threshold for the objective function value below which the optimization will stop.
Max Iterations	The maximum number of iterations the algorithm will run each time it restarts.
Temperature	The starting temperature for the simulated annealing, which sets the initial likelihood of accepting worse solutions in the first start.
Cooling Rate	The factor by which the temperature is multiplied after each iteration, governing how quickly the algorithm reduces its acceptance of worse solutions.
Max Starts	The maximum number of times the optimization algorithm will restart from the current best solution using reduced initial temperatures.
Hill Climbs	The number of hill climbing iterations for further refinement (see Appendix C for details).
Parallel Start	The number of independent runs (parallel or sequential) to simulate multiple data sets.
Best Solution	Return the best matching data set only (otherwise: return all data sets).
Weights	The weights multiplied with the regression/ SE and correlation term in the objective function; adjusting these values can steer the optimization toward preferred trade-offs and improve performance. In the ShinyApp the R function `weights_est()` can automatically estimate quasi-optimal objective function weights from each term's contribution.
Prob Global	The probability of applying a global vs. a local search move.
Sim Data	While the ShinyApp automatically passes data simulated by the descriptives module to the LM module, users of the R package must supply these simulated data manually.
Progress Bar	Enable the text progress bar during optimization.
Min Decimals	Minimum decimal places for target values' trailing zeros.

Note. **Bold** indicates reported summary estimates; *italic* indicates hyperparameters adjustable in the ShinyApp; plain text indicates hyperparameters only configurable via the R package.

Example. Robin knows the data ($N = 4$) come from a Likert scale ($range\ 1-7$) with reported mean $\mu_X = 2.5$, $\mu_Y = 2.5$ and SD $\sigma_X = 1.29$, $\sigma_Y = 1.29$. The target

$correlation$ is $r^{target} = 0$ and the intercept and slope are ${b_0}^{target} = 2.5$, ${b_1}^{target} = 0$ with ${SE_0}^{target} = 1.94$, ${SE_1}^{target} = 0.71$. They set the algorithm's parameters to

$Weights$: $\omega_r = 1$, $\omega_{b,SE} = 2$

$Tolerance = 0.001$

$Temperature = 1$

$Cooling\ Rate = 0.9$

All other parameters = $default$.

DISCOURSE then optimizes the data $X^{(i)}$, where i is the current iteration, to minimize the objective function f and match the reported summary estimates.

Candidate Initialization. The descriptive module simulates data to match $means$ and SDs

$$X^{(0)} = (1, 2, 3, 4)\ ,\ Y = (1, 2, 3, 4)\ ,$$

which results in

$$r^{(0)} = 1\ ,\ b_0^{(0)} = 2.5\ ,\ b_1^{(0)} = 0\ ,\ SE_0^{(0)} = 1.94\ ,\ SE_1^{(0)} = 0.71\ ,$$

$$f^{(0)} = \omega_r RMSE_r + \omega_{b,SE} RMSE_{b,SE} = 5.80 = f_{best} = f_{current}\ .$$

Iteration 1.

Candidate Modification. A *global move* is randomly chosen. The data is permuted at random

$$X^{(1)} = (4, 3, 2, 1)\ .$$

Candidate Evaluation. The correlation and regression estimates are computed

$$\hat{r}^{(1)} = -1\ ,\ \hat{b}_0^{(1)} = 5\ ,\ \hat{b}_1^{(1)} = -1\ ,\ \widehat{SE}_0^{(1)} = 0\ ,\ \widehat{SE}_1^{(1)} = 0\ ,$$

and the value of the objective function is

$$RMSE_r = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r} - r^{target})^2} = \sqrt{(-1-0)^2} = 1\ ,$$

$$RMSE_{b,SE} = \sqrt{\frac{1}{q}\sum_{j=1}^{q}\left[\left(\hat{b}_j - b_j^{target}\right)^2 + \left(\widehat{SE}_j - SE_j^{target}\right)^2\right]}$$

$$= \sqrt{\frac{(5-2.5)^2+(0-1.94)^2+(-1-0)^2+(0-0.71)^2}{2}} = 2.40\ ,$$

$$f^{(1)} = \omega_r RMSE_r + \omega_{b,SE} RMSE_{b,SE} = 1 \times 1 + 2 \times 2.40 = 5.80\ .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(1)} \text{-} f_{current} = 0\ .$$

Since the new candidate data does not fit worse, it is unconditionally accepted,

$$X_{current} = X^{(1)} \text{ and } f_{current} = f^{(1)}\ ,$$

while the best candidate remains

$$X_{best} = X^{(0)} \text{ and } f_{best} = f^{(0)} .$$

Cooling Schedule. The new temperature T is

$$T^{(1)} = 1 \times 0.9 = 0.9 .$$

Iteration 2.

Candidate Modification. A *local move* is randomly chosen. Index 1 (value 4) and index 4 (value 1) are selected and swapped. The data is updated

$$X^{(2)} = (1, 3, 2, 4) .$$

Candidate Evaluation. The correlation and regression estimates are computed

$$\hat{r}^{(2)} = 0.80 , \hat{b}_0^{(2)} = 0.50 , \hat{b}_1^{(2)} = 0.80 , \widehat{SE}_0^{(2)} = 1.16 , \widehat{SE}_1^{(2)} = 0.42 ,$$

and the value of the objective function is

$$RMSE_r = \sqrt{(0.80 - 0)^2} = 0.80 ,$$

$$RMSE_{b,SE} = \sqrt{\frac{(0.50-2.5)^2+(1.16-1.94)^2+(0.80-0)^2+(0.42-0.71)^2}{2}} = 1.63 ,$$

$$f^{(2)} = 1 \times 0.80 + 2 \times 1.63 = 4.06 .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(2)} - f_{current} = -1.74 .$$

Since the new candidate data fits better, it is unconditionally accepted

$$X_{current} = X^{(2)} \text{ and } f_{current} = f^{(2)} ,$$

and the best data is updated

$$X_{best} = X_{current} \text{ and } f_{best} = f_{current} .$$

Cooling Schedule. The new temperature is

$$Temperature^{(2)} = 0.9 \times 0.9 = 0.81 .$$

Iteration 3.

Subsequent iterations continue the same until either the $tolerance$ or $max\ iterations$ are reached**.**

Final Candidate. In this example, further updates eventually yield

$$X^{(*)} = (3, 1, 4, 2) = X_{best}$$

and the algorithm converges with an exact match

$$\hat{r}^{(*)} = 0 , \hat{b}_0^{(*)} = 2.50 , \hat{b}_1^{(*)} = 0 , \widehat{SE}_0^{(*)} = 1.94 , \widehat{SE}_1^{(*)} = 0.71 ,$$

$$f^{(*)} = 1 \times \sqrt{(0 - 0)^2} + 2 \times \sqrt{\frac{(2.5-2.5)^2+(1.94-1.94)^2+(0-0)^2+(0.71-0.71)^2}{2}} = 0 .$$

Finally, the data $W^{(*)} = (X_{best}, Y)$ is returned to Robin.

Linear Mixed Effects Regression

Overview. Suppose a published article reports the results of a linear mixed-effects (LME) regression. Besides the descriptive statistics (of data in the wide format), it states the bivariate $correlations$ of variables in the long format, unstandardized $regression\ coefficients$ with SE, and the regression $equation$ or model. The aim is to reconstruct an underlying data set whose values exactly reproduce these reported summary estimates. Thus, the LME module simulates a complete repeated-measures data set $W^{long} = (X, Y)$. X is a design matrix containing both, p_b between-subject predictors and p_w within-subject predictors (time-indexed t, measured at m occasions), and Y is the unidimensional outcome vector of length $N \times m$. Each marginal $mean$ and SD of columns of the data in wide format W^{wide} is required to match the targets. I optimize for the fixed-effect $regression\ coefficients$ (b) with their SE as well as the random-intercept SD (τ) and all pairwise $correlations$ (r), so that they align with the specified summary estimates from a mixed model fit (for parameter overview see Table 5). Note that nested multilevel data can be optimized as well, since its statistical modelling framework can be identical to that for repeated measures (Snijders & Bosker, 2012). No random slope input is implemented yet.

To deal with the rugged and high dimensional search space (see Appendix A for details), the LME module adopts a two-step strategy. Analogous to the LM module, I first invoke the descriptive module to generate each column of the data in wide format W^{wide} so that sample $means$ and SD are matching. With these marginals held fixed, I then transform the data in the long format and optimize the data for remaining targets, b, SE, τ, and r as objectives on the joint distribution W^{long}. Each search move preserves column-wise marginals in W^{wide}, while altering mixed-model estimates of W^{long} in an effective way. Thus, I use a simulated-annealing meta-heuristic to find a global best solution in the permutation space (Gutjahr, 2011; Kirkpatrick et al., 1983; Korte & Vygen, 2018).

Candidate Initialization. In a first step, the user needs to apply the descriptive module to simulate each column of W^{wide} so that its sample $means$ and SD match the reported summary estimates for both p_b and p_w as well as the outcome. This matrix is subsequently handed off to the LME module where candidate initialization converts it to long format W^{long}, resulting in $N \times m$ rows with columns for the subject identifier ID, $time$, p_b, p_w, and outcome Y. The resulting long-form table served directly as the initial search candidate, thereby guaranteeing that all univariate moment constraints held a priori. By

anchoring the initialization in these marginals, the subsequent combinatorial optimization could focus exclusively on achieving the target mixed-effects parameters.

Candidate Modification. Candidates in the LME module are transformed to W^{wide} and updated by within-column permutations guided by four types of stochastic and combinatorial search moves: *local move*, *k-cycle*, *tau reorder*, and *residual swap*. Together they ensure both broad exploration and fine-grained refinement of the solution. At each iteration i, the algorithm decides what move to perform. This is governed by adaptive probabilities which linearly shift over iterations from specified start to end values. The *local move* is similar to the one in the LM module, which swaps two values within a column of W^{wide}.

The algorithm performs a *k-cycle* by choosing a column j and drawing a cycle of length k from $(3 \dots, maxK)$ where $maxK = \max\left(3, \left\lfloor \frac{N}{4} \right\rfloor\right)$. Let

$$S \subseteq (1, \dots, N) \ , |S| = k \ ,$$

be the set of k distinct subject indices and let π be a random permutation on S, with corresponding permutation matrix P_π. The candidate update is

$$W_{S,j}^{wide\,(i+1)} = P_\pi \left(W_{S,j}^{wide\,(i)} \right) ,$$

which reorders k rows at once while keeping all other entries of W^{wide} constant. This mid-scale rearrangement is beneficial because it enables the algorithm to escape local minima by exploring larger novel configurations compared to the *local move*.

The next two moves are designed to increase and decrease the random intercept variance, respectively. When the *tau reorder* is selected, a single outcome time column t^* is randomly chosen from the set of repeated-measure time points $t_1, \dots, t_m$. Let

$$\bar{y}_s^{-t^*} = \frac{1}{m-1} \sum\nolimits_{t \neq t^*} y_{s,t}^{(i)}$$

be the mean outcome for subject s across all time points except t^*, and

$$\mathbb{v}^{(i)} = \left(y_{1,t}^{(i)}, \dots, y_{N,t}^{(i)} \right)$$

the vector of outcomes at t^*. Denote by π the permutation (matrix P_π) that sorts $\mathbb{v}^{(i)}$ into ascending order of the ranks of $\left(\bar{y}_s^{-t^*}\right)$. The update is

$$y_{,t^*}^{(i+1)} = P_\pi y_{,t^*}^{(i)} \ ,$$

with all other columns of W^{wide} held fixed. By reassigning the values at t^* to match the ordering of each subject's off-time mean, participant observations become more internally homogenous, thus increasing the random intercept variance τ.

When chosen, the residual swap selects a column j of W^{wide} at random. For each subject s, compute its mean trajectory

$$\mu_s = \frac{1}{m}\sum_{i=1}^{m} W_{s,t}^{wide^{(i)}} ,$$

let the residual be

$$\varepsilon = \mu_s - \bar{\mu} , \quad \bar{\mu} = \frac{1}{N}\sum_{s=1}^{N} \mu_s ,$$

and define the high and low sets of subjects by residuals

$$H = (s : \varepsilon_s > 0) , \qquad L = (s : \varepsilon_s \leq 0) .$$

Sample one subject

$$p \in H \text{ with } Pr(p = s) \propto \varepsilon_s \text{ for } s \in H ,$$

and another subject

$$q \in L \text{ with } Pr(q = s) \propto -\varepsilon_s \text{ for } s \in L .$$

Then swap their values in column j

$$W_{p,j}^{wide^{(i+1)}} = W_{q,j}^{wide^{(i)}} , \qquad W_{q,j}^{wide^{(i+1)}} = W_{p,j}^{wide^{(i)}} ,$$

leaving all other entries of W^{wide} unchanged. By exchanging high- and low-residual observations across two subjects (with regard to repeated measures), participants become more internally heterogeneous, thus decreasing the random intercept variance.

Collectively, these four permutation-based moves guarantee exact preservation of each column-wise distribution (wide data generated by descriptives module) while inducing fine local tweaks to mid-scale cycles and targeted variance adjustments. By balancing broad exploration with focused exploitation, the algorithm ensures thorough mixing of the search space and converges toward solutions that satisfy both the marginal and mixed-model targets.

Candidate Evaluation. Each candidate data matrix W^{long} is scored by an objective function $f(W)$ that combines the RMSE of the candidate correlation's $\hat{r}$, regression coefficients $\hat{b}$ and $\hat{\tau}$, and SE $\widehat{SE}$, against the target summaries r^{target}, b^{target}, τ^{target}, and SE^{target}. First, I compute the candidate correlation matrix from the W^{long} matrix using linear matrix algebra. I extract the upper triangle values and round it to the same precision as its target, yielding the vector $\hat{r} = (\hat{r}_1, \ldots, \hat{r}_m)$, where m is the number of elements in the upper triangle of the correlation matrix. The correlation RMSE is

$$RMSE_r = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r} - r^{target})^2} ,$$

where missing targets are omitted from the summations. Next, I assess how well the candidate reproduces the mixed-effects regression summaries. I fit the specified LME model

to the candidate W^{long} using the *lme4* package (Bates et al., 2015) as the estimation is non-trivial to implement using matrix algebra. $\hat{\tau}^2$ (instead of $\hat{\tau}$) is calculated as method-of-moments estimate via ANOVA to allow for negative values if the likelihood-based estimation gives $\hat{\tau}^2 = 0$. All estimates are rounded to the same precision as the corresponding targets. Let B denote the concatenation of b and τ^2 of length k and SE all standard errors of b of length $j = k - 1$, I compute

$$RMSE_{B,SE} = \sqrt{\frac{1}{k+j}\left[\sum_{i=1}^{k}\left(\hat{B}_i - B_i^{target}\right)^2 + \sum_{i=1}^{j}\left(\widehat{SE}_i - SE_i^{target}\right)^2\right]},$$

omitting missing targets. Finally, these two RMSE components are combined into the overall objective

$$f(W) = \omega_r RMSE_r + \omega_{B,SE} RMSE_{B,SE} ,$$

with specified weights ω_r and $\omega_{B,SE}$ reflecting the relative importance of reproducing the marginal correlations versus mixed-model summaries. Lower values of $f(W)$ indicate closer agreement across both $correlation$ and $regression/SE$, guiding the simulated annealing search towards data that match the target summaries without posing stringent assumptions.

Acceptance Criteria. For integer and continuous data, candidate updates follow a simulated-annealing scheme (see Appendix C for details).

Table 5

Parameter Overview for the LME Module: Reported Summary Estimates, Algorithm Hyperparameters in the ShinyApp and Package-Only Hyperparameters

Parameter	Description
Correlations	The vector containing the upper triangle of the reported bivariate correlation matrix of columns in the wide format.
Regression-Coefficient	The vector containing the target fixed-effect regression coefficients including the fixed-effect intercept and the last element being the SD of the random intercept.
SE	The vector containing the target SE of fixed-effect regression coefficients including the intercept.
Equation	The formula of the reported mixed effects regression model.
Tolerance	The threshold for the objective function value below which the optimization will stop.

Max Iterations	The maximum number of iterations the algorithm will run each time it restarts.
Temperature	The starting temperature for the simulated annealing, which sets the initial likelihood of accepting worse solutions in the first start.
Cooling Rate	The factor by which the temperature is multiplied after each iteration, governing how quickly the algorithm reduces its acceptance of worse solutions.
Max Starts	The maximum number of times the optimization algorithm will restart from the current best solution using reduced initial temperatures.
Hill Climbs	The number of hill climbing iterations for further refinement (see Appendix C for details).
Parallel Start	The number of independent runs (parallel or sequential) to simulate multiple data sets.
Best Solution	Return the best matching data set only (otherwise: return all data sets).
Weights	The weights multiplied with the regression/ SE and correlation term in the objective function; adjusting these values can steer the optimization toward preferred trade-offs and improve performance. In the ShinyApp the R function `weights_est()` can automatically estimate quasi-optimal objective function weights from each term's contribution.
Move Prob	A list with the adaptive probabilities of each move at the 'start' and at the 'end' of the optimization run.
Sim Data	While the ShinyApp automatically passes data simulated by the descriptives module (wide format) to the LME module, users of the R package must supply these simulated data manually.
Progress Bar	Enable the text progress bar during optimization.
Min Decimals	Minimum decimal places for target values' trailing zeros.

Note. **Bold** indicates reported summary estimates; *italic* indicates hyperparameters adjustable in the ShinyApp; plain text indicates hyperparameters only configurable via the R package.

Example. Robin knows three subjects ($N = 3$) were measured at three timepoints $t = (0,1,2)$ on a Likert scale ($range\ 1-7$) with reported $means$, $\mu_y^0 = 4.0$, $\mu_y^1 = 4.3$, $\mu_y^2 = 5.0$ and SD $\sigma_y^0 = 2.0$, $\sigma_y^1 = 1.5$, $\sigma_y^2 = 2.0$. Using a linear function of time t, the target correlation between t and y is $r^{target} = 0.3$ and the intercept and slope are ${b_0}^{target} = 3.9$,

${b_1}^{target} = 0.5$. The random intercept SD is $\tau = 1.8$ ($\tau^2 = 3.3$), and the SE of the intercept and slope are ${SE_0}^{target} = 1.1,\ {SE_1}^{target} = 0.1$. The model is $y \sim t + (1|ID)$. They set the algorithm's parameters to

$Weights$: $\omega_r = 1$, $\omega_{b,SE} = 1$

$Tolerance = 0.001$

$Temperature\ =\ 1$

$Cooling\ Rate\ =\ 0.9$

All other parameters $= default$.

DISCOURSE then optimizes the data $W^{(i)}$, where i is the current iteration, to minimize the objective function f and match the reported summary estimates.

Candidate Initialization. The descriptive module simulates data to match $means$ and SD (excluding ID column)

$$W^{wide(0)} = \begin{matrix} y\_0 & y\_1 & y\_2 \\ 6 & 3 & 7 \\ 4 & 6 & 3 \\ 2 & 4 & 5 \end{matrix},$$

which is converted to

$$W^{long(0)} = \begin{matrix} ID & t & y \\ 1 & 0 & 6 \\ 1 & 1 & 3 \\ 1 & 2 & 7 \\ 2 & 0 & 4 \\ 2 & 1 & 6 \\ 2 & 2 & 3 \\ 3 & 0 & 2 \\ 3 & 1 & 4 \\ 3 & 2 & 5 \end{matrix}$$

and results in

$$\hat{r}^{(0)} = 0.3\ ,\ \hat{b}_0^{(0)} = 3.9\ ,\ \hat{b}_1^{(0)} = 0.5\ ,\ \hat{\tau}^{2\,(0)} = -0.3\ ,\ \widehat{SE}_0^{(0)} = 0.9\ ,\ \widehat{SE}_1^{(0)} = 0.7\ ,$$

$$RMSE_r = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r} - r^{target})^2} = \sqrt{(0.3 - 0.3)^2} = 0\ ,$$

$$RMSE_{B,SE} = \sqrt{\frac{1}{k+j}\left[\sum_{i=1}^{k}\left(\hat{B}_i - B_i^{target}\right)^2 + \sum_{i=1}^{j}\left(\widehat{SE}_i - SE_i^{target}\right)^2\right]},$$

$$= \sqrt{\frac{(3.9-3.9)^2+(0.5-0.5)^2+(-0.3-3.3)^2+(0.9-1.1)^2+(0.7-0.1)^2}{3+2}} = 1.63\ ,$$

$$f^{(0)} = \omega_r RMSE_r + \omega_{b,SE} RMSE_{b,SE} = 1.63 = f_{best}\ .$$

Iteration 1.

Candidate Modification. A *k-cycle* is randomly chosen with $k = 3$ and the second column is randomly selected. The data is permuted

$$W^{wide(1)} = \begin{matrix} y\_0 & y\_1 & y\_2 \\ 6 & 4 & 7 \\ 4 & 3 & 3 \\ 2 & 6 & 5 \end{matrix},$$

and converted to

$$W^{long(1)} = \begin{matrix} ID & t & y \\ 1 & 0 & 6 \\ 1 & 1 & 4 \\ 1 & 2 & 7 \\ 2 & 0 & 4 \\ 2 & 1 & 3 \\ 2 & 2 & 3 \\ 3 & 0 & 2 \\ 3 & 1 & 6 \\ 3 & 2 & 5 \end{matrix}.$$

Candidate Evaluation. The correlation and regression estimates are computed

$$\hat{r}^{(1)} = 0.3\ ,\ \hat{b}_0^{(1)} = 3.9\ ,\ \hat{b}_1^{(1)} = 0.5\ ,\ \hat{\tau}^{2(1)} = 0.6\ ,\ \widehat{SE}_0^{(1)} = 0.9,\ \widehat{SE}_1^{(1)} = 0.6\ ,$$

and the value of the objective function is

$$RMSE_r = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{r} - r^{target})^2} = \sqrt{(0.3 - 0.3)^2} = 0\ ,$$

$$RMSE_{B,SE} = \sqrt{\frac{1}{k+j}\left[\sum_{i=1}^{k}\left(\hat{B}_i - B_i^{target}\right)^2 + \sum_{i=1}^{j}\left(\widehat{SE}_i - SE_i^{target}\right)^2\right]}$$

$$= \sqrt{\frac{(3.9-3.9)^2+(0.5-0.5)^2+(0.6-3.3)^2+(0.9-1.1)^2+(0.6-0.1)^2}{3+2}} = 1.23\ ,$$

$$f^{(1)} = \omega_r RMSE_r + \omega_{b,SE} RMSE_{b,SE} = 1 \times 0 + 1 \times 1.23 = 1.23\ .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(1)} \text{-} f_{current} = -0.40\ .$$

Since the new candidate data does fit better, it is unconditionally accepted,

$$W_{current} = W^{(1)} \text{ and } f_{current} = f^{(1)}\ ,$$

and the best candidate is updated

$$W_{best} = W_{current} \text{ and } f_{best} = f_{current}\ .$$

Cooling Schedule. The new temperature T is

$$T^{(1)} = 1 \times 0.9 = 0.9\ .$$

Iteration 2.

Candidate Modification. A *tau reorder* is randomly chosen and the first repeated-measures column got randomly selected. Row means (and their ranks) are calculated for the remaining columns

$$W^{wide(1)} = \begin{matrix} y\_0 & y\_1 & y\_2 \\ 6 & 4 & 7 \\ 4 & 3 & 3 \\ 2 & 6 & 5 \end{matrix}, \quad \begin{matrix} RowMean \\ 5.5 \\ 3 \\ 5.5 \end{matrix}, \quad \begin{matrix} Rank \\ 2 \\ 1 \\ 3 \end{matrix},$$

and the first column is sorted in ascending order of the ranks

$$W^{wide(2)} = \begin{matrix} y\_0 & y\_1 & y\_2 \\ 4 & 4 & 7 \\ 2 & 3 & 3 \\ 6 & 6 & 5 \end{matrix}$$

and converted to

$$W^{long(2)} = \begin{matrix} ID & t & y \\ 1 & 0 & 4 \\ 1 & 1 & 4 \\ 1 & 2 & 7 \\ 2 & 0 & 2 \\ 2 & 1 & 3 \\ 2 & 2 & 3 \\ 3 & 0 & 6 \\ 3 & 1 & 6 \\ 3 & 2 & 5 \end{matrix}.$$

Candidate Evaluation. The correlation and regression estimates are computed

$$\hat{r}^{(2)} = 0.3\ ,\ \hat{b}_0^{(2)} = 3.9\ ,\ \hat{b}_1^{(2)} = 0.5\ ,\ \hat{\tau}^{2(2)} = 2.1\ ,\ \widehat{SE}_0^{(2)} = 1.0,\ \widehat{SE}_1^{(2)} = 0.4\ ,$$

and the value of the objective function is

$$RMSE_r = \sqrt{(0.3-0.3)^2} = 0\ ,$$

$$RMSE_{B,SE} = \sqrt{\frac{1}{k+j}\left[\sum_{i=1}^{k}\left(\hat{B}_i - B_i^{target}\right)^2 + \sum_{i=1}^{j}\left(\widehat{SE}_i - SE_i^{target}\right)^2\right]}$$

$$= \sqrt{\frac{(3.9-3.9)^2+(0.5-0.5)^2+(2.1-3.3)^2+(1.0-1.1)^2+(0.4-0.1)^2}{3+2}} = 0.55\ ,$$

$$f^{(2)} = 1 \times 0 + 1 \times 0.55 = 0.55\ .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(2)} \text{-} f_{current} = -0.68\ .$$

Since the new candidate data does fit better, it is unconditionally accepted,

$$W_{current} = W^{(2)} \text{ and } f_{current} = f^{(2)}\ ,$$

and the best candidate is updated

$$W_{best} = W_{current} \text{ and } f_{best} = f_{current}\ .$$

Cooling Schedule. The new temperature is

$$T^{(2)} = 0.9 \times 0.9 = 0.81 \ .$$

Iteration 3.

Candidate Modification. A *residual swap* is randomly chosen and the third repeated-measures column got randomly selected. Row means and residuals (compared to the grand row mean) are calculated

$$W^{wide(2)} = \begin{matrix} y\_0 & y\_1 & y\_2 & RowMean & Residual \\ 4 & 4 & 7 & 5 & 0.6 \\ 2 & 3 & 3 & 2.7 & -1.7 \\ 6 & 6 & 5 & 5.7 & 1.3 \end{matrix}, \ \overline{RowMean} = 4.4 \ ,$$

and one row with a residual larger than zero and another with a residual smaller than zero are sampled. Here the second and third row are selected and swapped

$$W^{wide(3)} = \begin{matrix} y\_0 & y\_1 & y\_2 \\ 4 & 4 & 7 \\ 2 & 3 & 5 \\ 6 & 6 & 3 \end{matrix}$$

and converted to

$$W^{long(3)} = \begin{matrix} ID & t & y \\ 1 & 0 & 4 \\ 1 & 1 & 4 \\ 1 & 2 & 7 \\ 2 & 0 & 2 \\ 2 & 1 & 3 \\ 2 & 2 & 5 \\ 3 & 0 & 6 \\ 3 & 1 & 6 \\ 3 & 2 & 3 \end{matrix} .$$

Candidate Evaluation. The correlation and regression estimates are computed

$$\hat{r}^{(3)} = 0.3 \ , \hat{b}_0^{(3)} = 3.9 \ , \hat{b}_1^{(3)} = 0.5 \ , \hat{t}^{2(3)} = 0 \ , \widehat{SE}_0^{(3)} = 0.9, \ \widehat{SE}_1^{(3)} = 0.7 \ ,$$

and the value of the objective function is

$$RMSE_r = \sqrt{(0.3 - 0.3)^2} = 0 \ ,$$

$$RMSE_{B,SE} = \sqrt{\frac{(3.9-3.9)^2+(0.5-0.5)^2+(0-3.3)^2+(0.9-1.1)^2+(0.7-0.1)^2}{3+2}} = 1.49 \ ,$$

$$f^{(3)} = 1 \times 0 + 1 \times 1.49 \ = 1.49 \ .$$

Candidate Acceptance. The change in objective is

$$\Delta f = f^{(3)} \text{-} f_{current} = +0.94 \ .$$

Since the new candidate data fits worse, it is accepted with probability

$$P \ = \ e^{(-\Delta f / T^{(2)})} = 0.31 \ .$$

Here the draw does not succeed, so the candidate is not accepted

$$W_{current} = W_{current} \text{ and } f_{current} = f_{current} ,$$

and the best candidate remains the same.

$$W_{best} = W_{best} \text{ and } f_{best} = f_{best} .$$

Cooling Schedule. The new temperature is

$$T^{(3)} = 0.81 \times 0.9 = 0.73 .$$

Iteration 4.

Subsequent iterations continue until either the $tolerance$ or $max\ iterations$ are reached.

Final Candidate. In this example, further updates eventually yield

$$W^{wide(*)} = \begin{matrix} y\_0 & y\_1 & y\_2 \\ 4 & 4 & 5 \\ 2 & 3 & 3 \\ 6 & 6 & 7 \end{matrix} ,$$

which is converted to

$$W^{long(*)} = \begin{matrix} ID & t & y \\ 1 & 0 & 4 \\ 1 & 1 & 4 \\ 1 & 2 & 5 \\ 2 & 0 & 2 \\ 2 & 1 & 3 \\ 2 & 2 & 3 \\ 3 & 0 & 6 \\ 3 & 1 & 6 \\ 3 & 2 & 7 \end{matrix}$$

and the algorithm converges with an exact match

$$\hat{r}^{(4)} = 0.3 , \hat{b}_0^{(4)} = 3.9 , \hat{b}_1^{(4)} = 0.5 , \hat{\tau}^{2(4)} = 3.3 , \widehat{SE}_0^{(4)} = 1.1, \ \widehat{SE}_1^{(4)} = 0.1 ,$$

$$RMSE_r = \sqrt{(0.3 - 0.3)^2} = 0 ,$$

$$RMSE_{B,SE} = \sqrt{\frac{(3.9-3.9)^2+(0.5-0.5)^2+(3.3-3.3)^2+(1.1-1.1)^2+(0.1-0.1)^2}{3+2}} = 0 ,$$

$$f^{(4)} = 1 \times 0 + 1 \times 0 \ = 0 .$$

Finally, the data $W^{long(*)} = W_{best}$ is returned to Robin.

Performance Enhancement

Performance enhancement in DISCOURSE is achieved through several complementary strategies. First, an optional objective-weights estimation step is offered, whereby the relative importance of each term in the objective function is estimated from the data using Monte-Carlo routines and pilot runs. Second, algorithm hyperparameters such as

temperature schedules and iteration limits can be fine-tuned to the specific characteristics of a given application. Finally, the underlying implementation leverages parallel computation and C++ routines to minimize runtime. All of these techniques are described in detail in Appendix D.

Validation

To validate the DISCOURSE framework, it was applied to benchmark studies for which the original raw data were available (details below). Only the published summary statistics (e.g., $means$, SD, $correlations$, $regression\ coefficients$, SE, F value) were supplied to the algorithm, yielding a fully synthetic data set. Convergence was evaluated and features of the simulated data were compared with those of the original observations across all four modules. I assessed alignment of both topological characteristics and quantitative metrics using histograms, bar charts, and partial regression plots (the R scripts are available on the OSF at https://osf.io/3yjaq/). These results demonstrate that DISCOURSE reliably simulates data that reproduce published summary estimates.

Descriptives

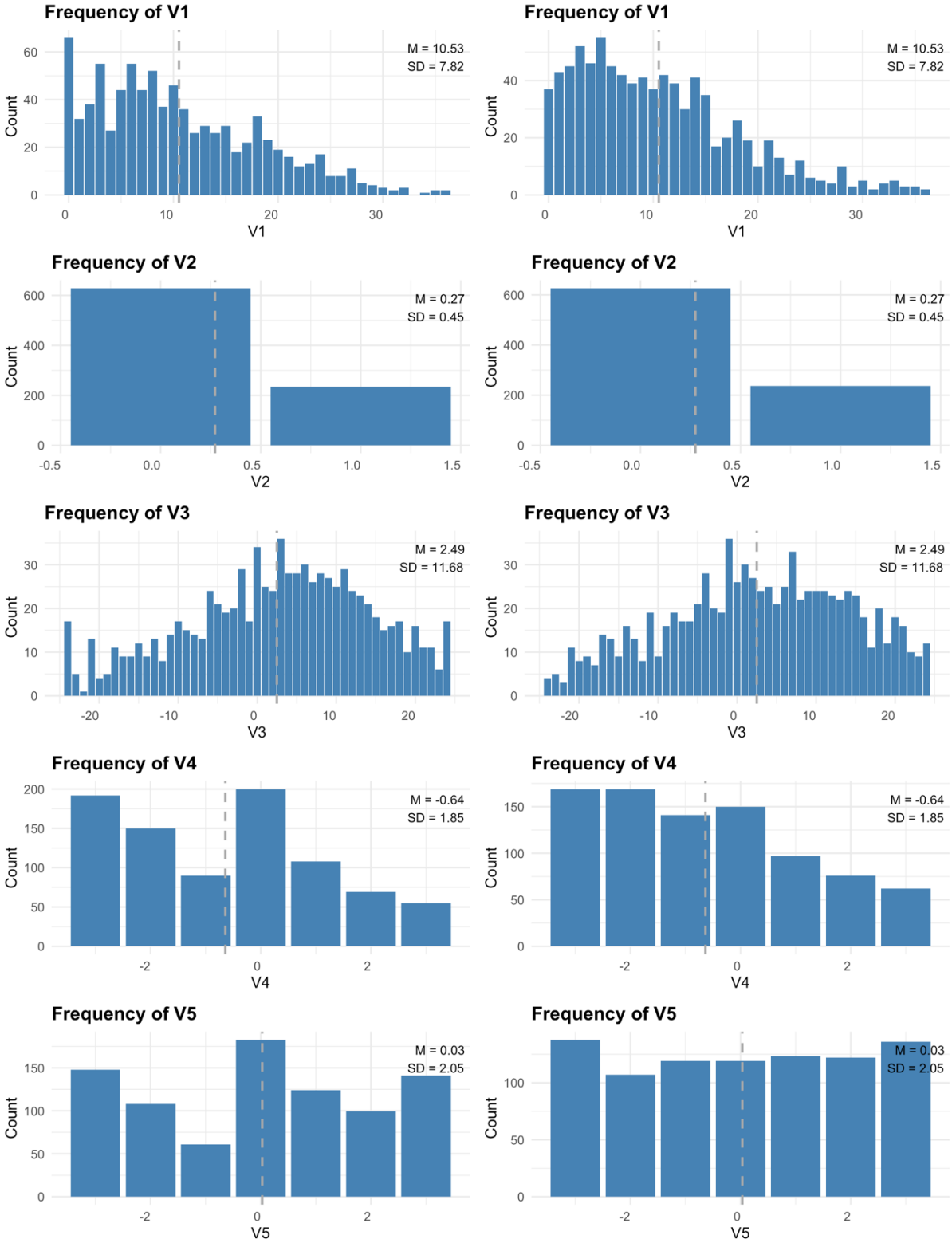
Benchmark data were drawn from the publicly available data set analysed in Le Forestier et al. (2024), data available at https://osf.io/t6d5y. Five variables were arbitrarily selected (named V1, …, V5) to span categorical dummies, integer values, and continuous measures, ensuring a diverse representation of data types. The descriptives module was applied and the optimization converged with $tolerance = e^{-8}$, $RMSE = 0$ for both $means$ and SD. To assess validity, the distribution and frequencies of all five variables in the simulated data were compared to the ones in the original data. Figure 2 illustrates that the distributions produced by the DISCOURSE descriptives module closely match those of the original data set.

ANOVA

The benchmark data set used here was obtained from the publicly available resource analysed by Zhang et al. (2025), data available at https://osf.io/6fhev/. I chose two factors with two groups each, so as to include a between factor (Factor1) and a within factor (Factor2). The outcome variable contained integer values. I computed F values based on the model $Outcome \sim Factor1 + Factor2 + Factor1{:}\,Factor2 + Error(ID/Factor2)$. I then ran the ANOVA module, which converged at a $tolerance = e^{-8}$, yielding an $RMSE = 0$ for the F values. To verify the results, I compared the simulated frequency counts of the outcome variable against the one in the original data set. As shown in Figure 3, the

Figure 2

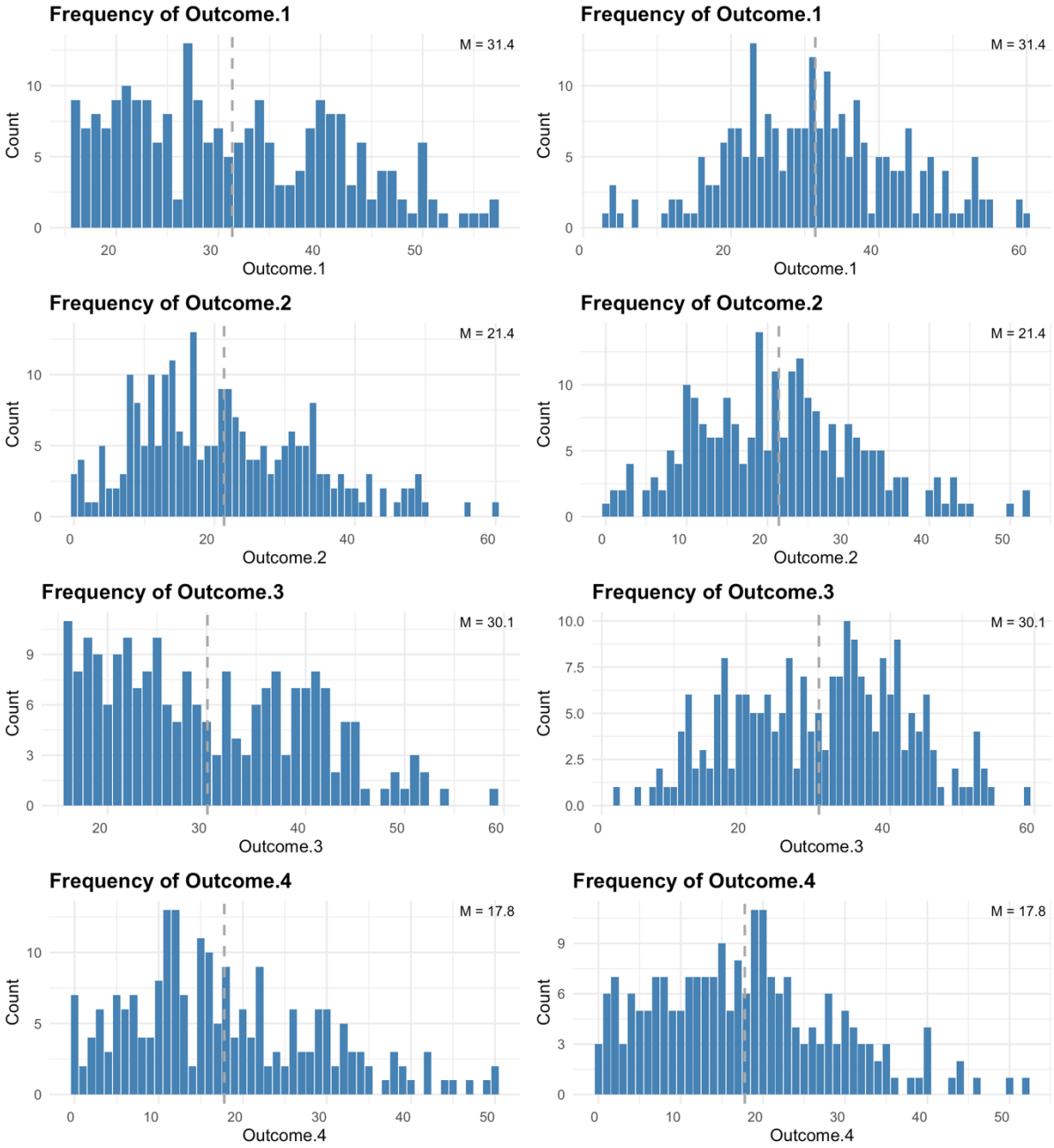
Distribution and Frequency of the Variables based on the Original Data (Left Panel) and the Simulated Data from the Descriptive Module (Right Panel)



distributions generated by the DISCOURSE descriptives module are topologically similar to the original ones, confirming that the module's optimization procedure re-constructs plausible raw data sets.

Figure 3

Frequency of the Outcome Variable in the Subgroups based on the Original Data (Left Panel) and the Simulated Data from the ANOVA Module (Right Panel)



Multiple Linear Regression

The LM module was applied to the same benchmark data set described in the descriptives sub-section (Le Forestier et al., 2024) to assess the module's validity. $Correlations$ were computed and the model $V5 \sim V1 + V2 + V3 + V4 + V1{:}V3 + V2{:}V3$ was estimated to derive $regression\ coefficients$ and their SE. The data simulated by the descriptives module (in descriptives sub-section) was fed into the LM optimization together with these summary estimates. After estimating the $weights$, the algorithm was run and converged with reaching $max\ iterations = 1e^5$ and achieved $RMSE < .001$ for all objectives. To evaluate validity, I compared partial regression plots of all six predictor terms in the simulated data set with those from the original data. As shown in Figure 4, the LM module reproduces the original distributions almost identically.

Linear Mixed-Effects Regression

I utilized another benchmark data set from Spikman et al. (1999), data available at https://osf.io/3yjaq/ (this is a perturbed version of the original data). I selected two between-subject variables (one integer, one dummy), one continuous outcome measured repeatedly at four time points, and I model time as a predictor using an inverse transformation (to ensure a linear relationship with the outcome). $Means$ and SD were calculated based on the data columns in wide format (across subjects), $correlations$ were computed based on the data columns in long format and the random intercept model $V4 \sim V1 + V2 + V3 + age + (1 \mid ID)$ was estimated to derive $regression$ outputs. The LME module was applied as follows: First, the descriptive module given $means$ and SD (moments across subjects and timepoints) converged with $tolerance = e^{-8}$ yielding both $RMSE = 0$ (see Figure 5).

Second, the simulated data was fed into LME's combinatorial optimization together with the bivariate $correlations$, $regression\ coefficients$ including the random intercept variation, and SE. After estimating the $weights$, the algorithm was run in parallel to capitalize on between run variability and converged with reaching $max\ iterations = 5e^3$ ($tolerance = e^{-8}$) and achieved $RMSE_{cor} = 0, RMSE_{reg} < .01$, and $RMSE_{SE} = 0$ for the objectives. Validity was assessed by comparing the partial-regression plots for all six predictors between the simulated and original data sets. As illustrated in Figure 6, the DISCOURSE LME module closely replicates the original patterns.

Figure 4

Partial Regression Plots of the Variables based on the Original Data (Left Panel) and the Simulated Data from the LM Module (Right Panel)

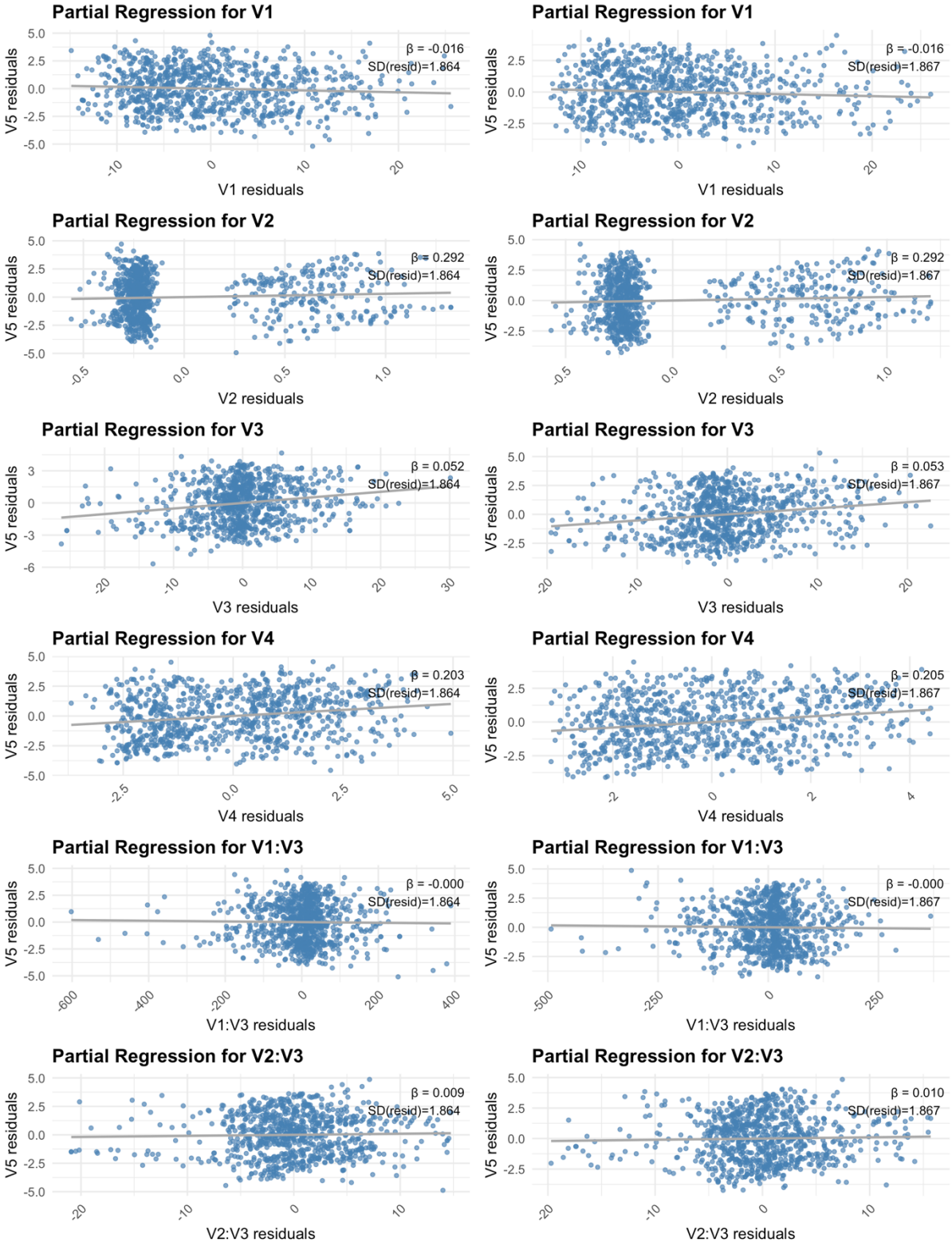


Figure 5

Distribution and Frequency of the Variables based on the Original Data (Left Panel) and the Simulated Data from the Descriptive Module (Right Panel)

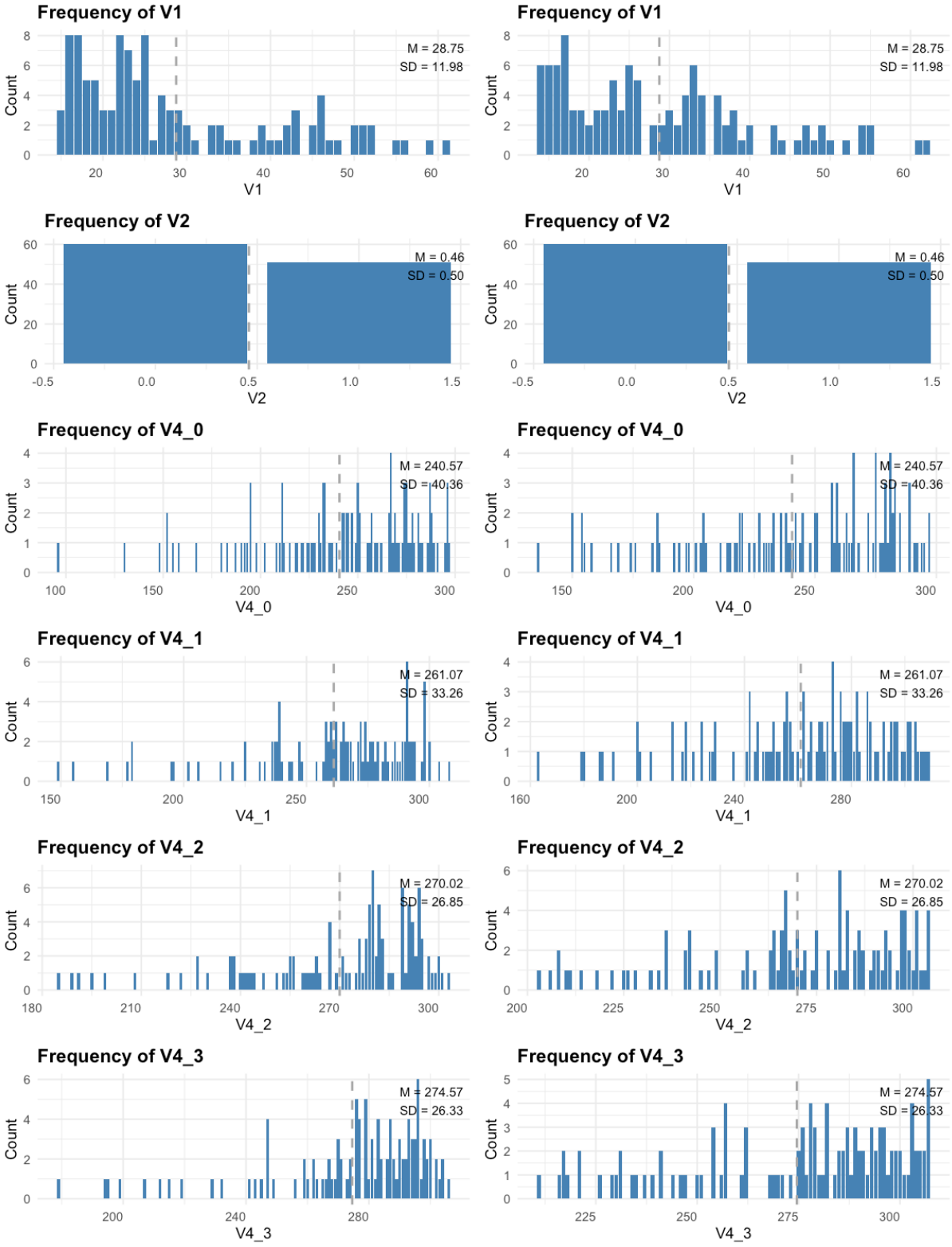
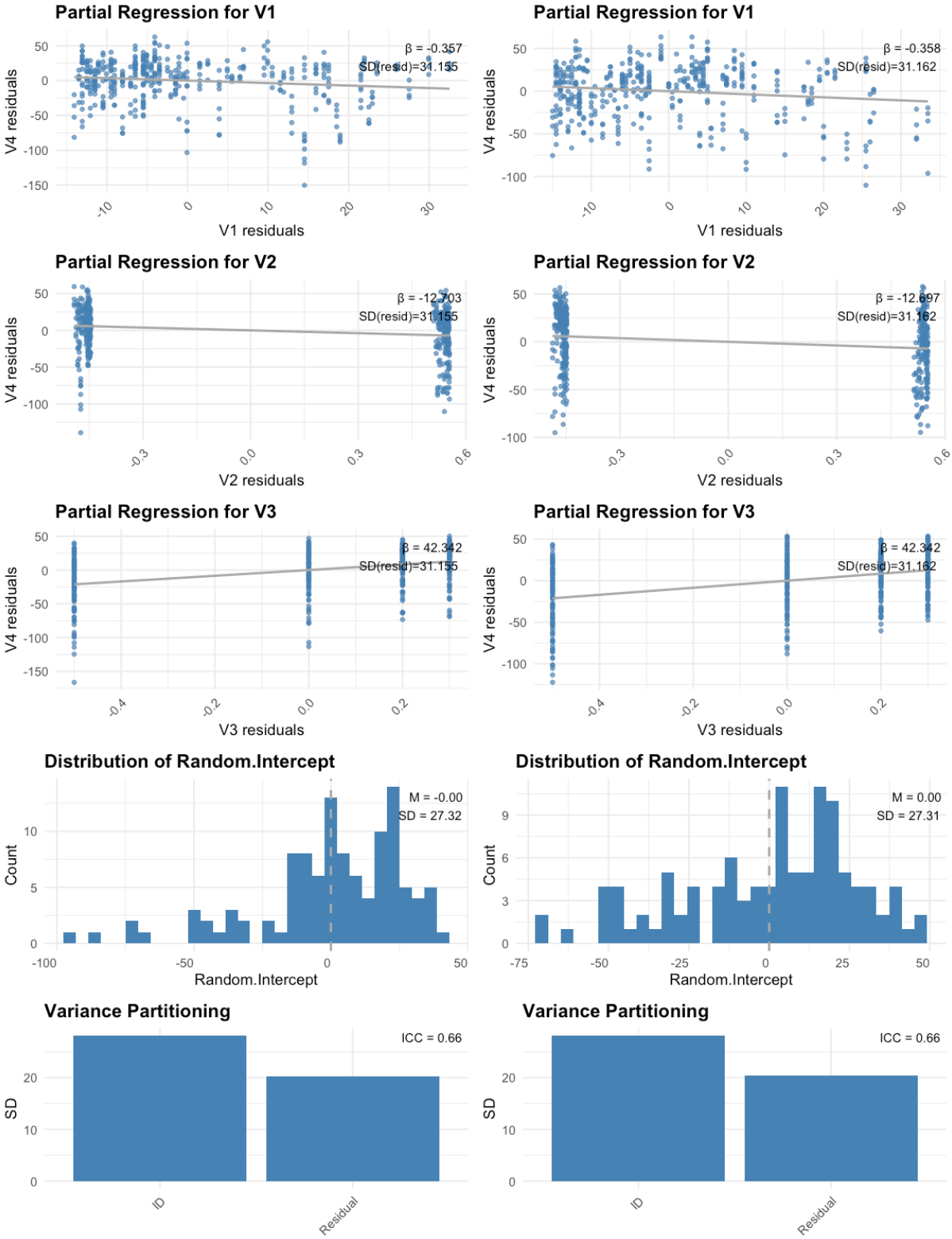


Figure 6

Partial Regression Plots, Random Intercept Distribution and Variance Partitioning based on the Original Data (Left Panel) and the Simulated Data from the LME Module (Right Panel)



Application

As part of the Open Science Collaboration's large-scale effort to estimate the replicability of psychological findings (Open-Science-Collaboration, 2015), many original data sets remain unavailable. In the study by Reynolds and Besner (2008) on contextual effects in reading aloud, participants' response times to exception words and nonwords were measured under predictable switch and stay sequences to probe dynamic pathway control in skilled reading. In the following steps, I applied DISCOURSE to demonstrate how it's ANOVA module can generate a fully synthetic data set matching the reported summary estimates. The R scripts are available on the OSF at https://osf.io/3yjaq/ and a tutorial article can be accessed via https://sebastian-lortz.github.io/discourse/ (for application of the Descriptives, LM, and LME module, see Appendix E).

I began by extracting the relevant parameters from the article.

```
N = 16
levels = c(2,2)
target_group_means <- c(543, 536, 614, 618)
factor_type <- c("within", "within")
formula <- outcome ~ Factor1 * Factor2 + Error(ID / (Factor1 *
Factor2))"
integer <- FALSE
target_f_vec <- list(effect = c("Factor1", "Factor2",
"Factor1:Factor2"),
                          F = c(30.5, 0.0, 0.2))
```

Note that Factor2 and the interaction effect are reported as $F < 1$, thus, I set arbitrary values. I then computed a plausible response time bounds $[L, U]$ from the pooled $MSE = 3070$ using

$$L = \min(target\_group\_means) - \left\lfloor 3 \times \sqrt{MSE} \right\rfloor = 370 ,$$

$$\mathrm{U} = \max(target\_group\_means) + \left\lceil 3 \times \sqrt{MSE} \right\rceil = 785 ,$$

and define the parameter.

```
range <- c(370,785)
```

Next, I ran the ANOVA module with a small $max\ step$ to avoid early convergence given the coarse target precision and otherwise default hyperparameters.

```
result.aov <- optim_aov(N = N,
                        levels = levels,
                        target_group_means = target_group_means,
                        target_f_list = target_f_vec,
                        factor_type = factor_type,
```

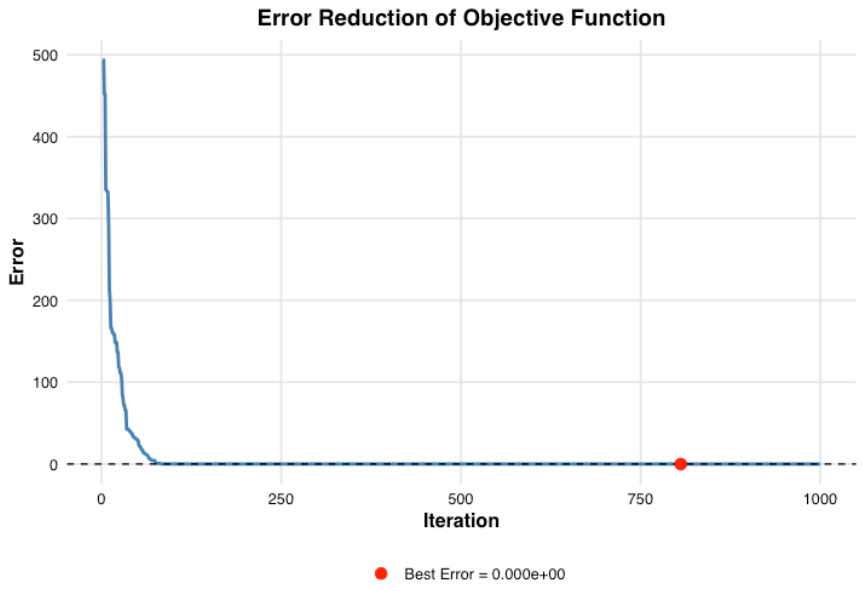
```
                range = range,
                formula = formula,
                integer = integer,
                max_step = .1,
                tolerance = 1e-8,
                max_iter = 1e3,
                init_temp = 1,
                cooling_rate = NULL)
```

The optimization converged exactly. Inspecting

```
summary(result.aov)
      DISCOURSE Object Summary
      ------------------------------------------------

      Achieved Loss of Optimization:  0

      RMSE of F statistics:  0

      Factorial Model:
      outcome ~ Factor1 * Factor2 + Error(ID/(Factor1 * Factor2))

      Group Means:
      [1] 543 536 614 618
```

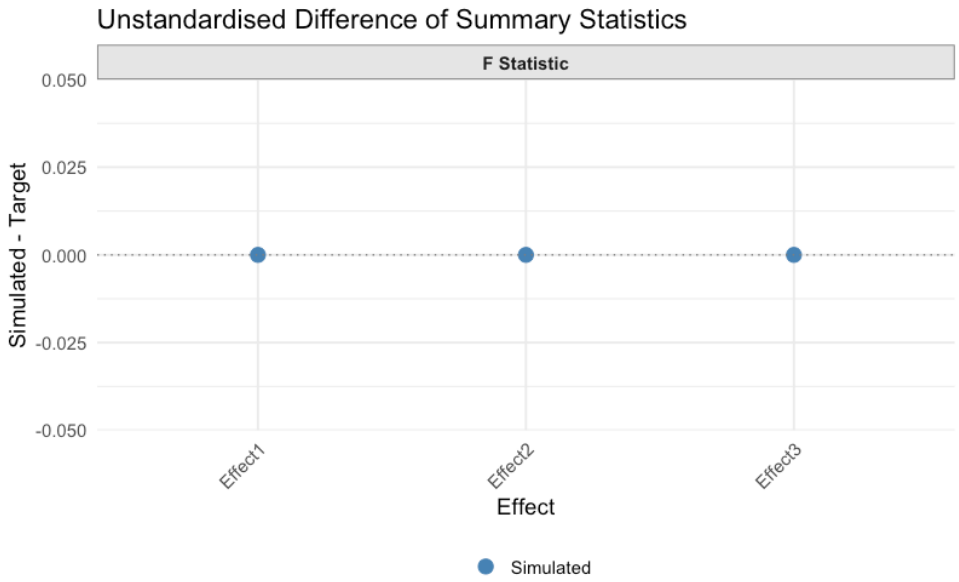
confirms that the simulated data reproduce the published cell means and ANOVA output. I then visualized the error trajectory

```
plot_error(result.aov)
```



and illustrated the estimated versus target effects with

```
plot_summary(result.aov, standardised = FALSE)
```



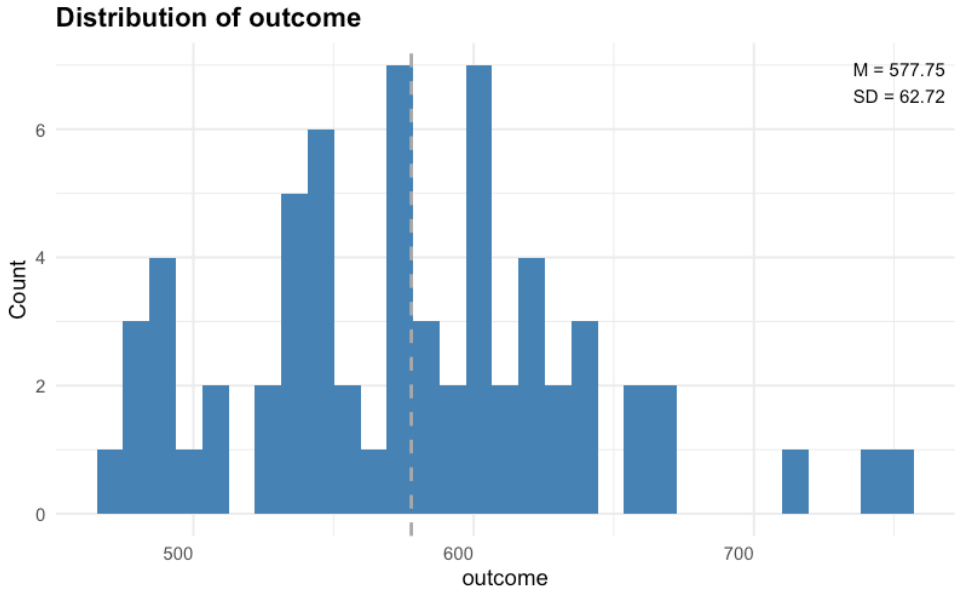
Finally, I saved the RMSE and relevant statistics,

```
get_rmse(result.aov)
      $rmse_F
      [1] 0
get_stats(result.aov)
      $model
      Anova Table (Type 3 tests)

      Response: outcome
                    num Df den Df    MSE       F     ges    Pr(>F)
      Factor1             1     15 3067.7 30.5232 0.37868 5.829e-05
      ***
      Factor2             1     15 1555.0  0.0232 0.00023    0.8811
      Factor1:Factor2     1     15 2111.6  0.2292 0.00314    0.6390
      ---
      Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

      $F_value
      [1] 30.52321490  0.02315103  0.22921165

      $mean
      [1] 543 536 614 618
```

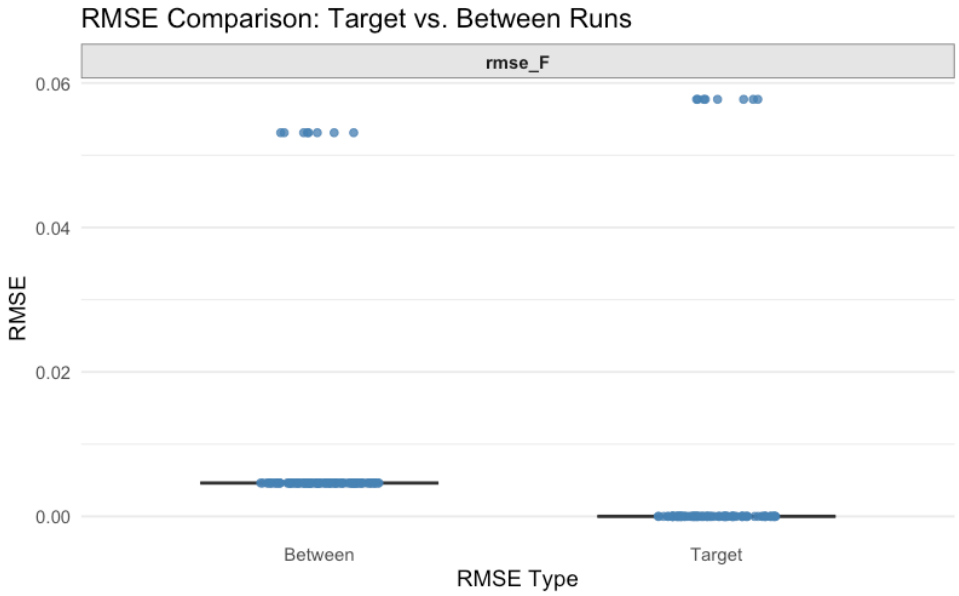
extracted the simulated data set and inspected its distribution.

```
data.aov <- result.aov$data
head(data.aov)
        ID Factor1 Factor2  outcome
```

```
      1   1        1        1 603.3692
      2   1        1        2 538.9177
      3   1        2        1 656.4115
      4   1        2        2 622.3074
      5   2        1        1 569.7279
      6   2        1        2 576.2693
plot_histogram(data.aov[,4, drop = FALSE])
      $outcome
```



I executed the ANOVA module in multiple parallel runs to quantify both the convergence variability of RMSE within each run (compared to the target values) and the variability of RMSE across runs (compared to the average simulated values). I then plotted these RMSE distributions side-by-side to compare within versus between run variation.



Discussion

The present work introduced DISCOURSE - Data-simulation via Iterative Stochastic Combinatorial Optimization Using Reported Summary Estimates - as a practical tool for generating plausible raw data from published summary statistics. By iteratively refining candidate data sets until their descriptive and inferential statistics match target values, the algorithmic framework produces underlying data that could plausibly have produced the original findings. In doing so, the method goes beyond the limited information available in published articles and enables researchers to conduct follow-up analyses such as multiverse and many-analyst comparisons. This would otherwise require access to the original raw data. Importantly, DISCOURSE is distributed both as an R package and via an accessible Shiny application, balancing immediate usability for applied researchers with the flexibility for users to fine-tune optimization parameters and explore advanced customization.

Despite its contributions, some limitations merit consideration. First, the vast heterogeneity of empirical data means that some scenarios may require extensive parameter tuning to achieve satisfactory accuracy, e.g., adjusting iteration limits, temperature schedules, and modification probabilities. Although the Shiny app and R package provide diagnostic outputs to guide these choices, arriving at optimal settings can involve trial and error and substantial computation time. Second, while DISCOURSE reports the achieved accuracy that reflects the differences between summary statistics of the simulated data and their targets, it does not estimate how closely the estimated data mirror the true, unobserved raw data. Thus, low error in reproducing moments and test statistics guarantees only summary-level accuracy, not data-level realism. Users should interpret simulated data sets as useful decision-support tools rather than as definitive reconstructions of original observations. Third, the combinatorial optimization leveraged in the LM module is based on the implicit assumption that some combinations of simulated data values from the descriptives module will yield matching inferential statistics. Therefore, the algorithm may struggle with small sample sizes because there are insufficient permutations available to identify a satisfactory solution. Finally, the LME module requires column-wise means and SD for each measurement occasion (data in wide format), correlations of variables across subjects and measurements (data in long format), and supports only a single random intercept model. Because most multilevel studies include random slopes, crossed effects, and rarely publish correlation structures or the per-occasion descriptives needed for optimization, its direct application remains confined to a small set of well-reported designs. Future research may expand the

LME module to support random slopes and more complex covariance structures and refine its optimization algorithms to rely on fewer reported summary estimates.

Beyond replication-study decision making, DISCOURSE holds promise for a variety of complementary applications. In statistical education, instructors can generate 'textbook-exact' data sets from hypothetical summary scenarios, enabling students to practice analyses without raising privacy or data-access concerns. Educators might also leverage the framework to create unique but equivalent data sets for each student, matching the same summary specifications while preventing academic dishonesty. When privacy concerns preclude sharing raw data and researchers wish to cut all ties to the original data set, they can use DISCOURSE to generate synthetic data that mirror the key summary statistics of their original observations. They can publicly release a fully reproducible data set without compromising individual confidentiality. Finally, the modular design of the algorithmic framework invites ongoing methodological development: future extensions could incorporate additional summary estimates (e.g., effect-sizes, random slopes, or survival curves) or alternative optimization strategies, broadening applicability across diverse research domains.

In conclusion, DISCOURSE advances an efficient, transparent, and sustainable approach to synthetic-data generation when original data sets are unavailable. By integrating robust meta-heuristic optimization routines with accessible interfaces, the framework empowers researchers to perform analyses on simulated data and inform replication study-decision making. Although parameter tuning and the inherent uncertainty of data reconstruction impose natural limits, DISCOURSE represents a substantive step toward more informed and resource-efficient research practices in the behavioural and social sciences.

Transparency

Conflict of Interest Statement

Sebastian A. J. Lortz declares no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

There is no funding information provided for the author.

Ethical Approval

Ethical approval was not required for this study as it did not include real-world data collection.

Data, Materials, and Online Resources

All materials, data, and R scripts are publicly available on the OSF at https://osf.io/3yjaq/. The proposed software is open source at https://sebastian-lortz.github.io/discourse/.

Literature

Abdel-Basset, M., Abdel-Fatah, L., & Sangaiah, A. K. (2018). Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review. In A. K. Sangaiah, M. Sheng, & Z. Zhang (Eds.), *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications* (pp. 185-231). Academic Press. https://doi.org/10.1016/B978-0-12-813314-9.00010-4

Anaya, J. (2016). The GRIMMER test: A method for testing the validity of reported measures of variability. *PeerJ Preprints*, *4*, e2400v2401. https://doi.org/10.7287/peerj.preprints.2400v1

Bardwell, W. A., Ancoli-Israel, S., & Dimsdale, J. E. (2007). Comparison of the effects of depressive symptoms and apnea severity on fatigue in patients with obstructive sleep apnea: A replication study. *Journal of Affective Disorders*, *97*(1), 181-186. https://doi.org/10.1016/j.jad.2006.06.013

Bardwell, W. A., Moore, P., Ancoli-Israel, S., & Dimsdale, J. E. (2003). Fatigue in obstructive sleep apnea: driven by depressive symptoms instead of apnea severity? *Am J Psychiatry*, *160*(2), 350-355. https://doi.org/10.1176/appi.ajp.160.2.350

Bates, D., Mächler, M., Bolker, B., & Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, *67*(1), 1 - 48. https://doi.org/10.18637/jss.v067.i01

Bendtsen, C. (2022). *pso: Particle Swarm Optimization*. In (Version R package version 1.0.4) https://CRAN.R-project.org/package=pso

Bolland, M. J., Gamble, G. D., Grey, A., & Avenell, A. (2020). Empirically generated reference proportions for baseline p values from rounded summary statistics. *Anaesthesia*, *75*(12), 1685-1687. https://doi.org/10.1111/anae.15165

Bordewijk, E. M., Li, W., van Eekelen, R., Wang, R., Showell, M., Mol, B. W., & van Wely, M. (2021). Methods to assess research misconduct in health-related research: A scoping review. *Journal of Clinical Epidemiology*, *136*, 189-202. https://doi.org/10.1016/j.jclinepi.2021.05.012

Brown, N. J. L., & Heathers, J. A. J. (2017). The GRIM Test: A Simple Technique Detects Numerous Anomalies in the Reporting of Results in Psychology. *Social Psychological and Personality Science*, *8*(4), 363-369. https://doi.org/10.1177/1948550616673876

Clerc, M. (2012). *Standard Particle Swarm Optimisation*. https://hal.science/hal-00764996

Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009). *Introduction to derivative-free optimization*. SIAM. https://doi.org/10.1137/1.9780898718768

Delahaye, D., Chaimatanan, S., & Mongeau, M. (2019). Simulated Annealing: From Basics to Applications. In *Handbook of Metaheuristics* (pp. 1). https://doi.org/10.1007/978-3-319-91086-4_1

Derksen, M., Meirmans, S., Brenninkmeijer, J., Pols, J., de Boer, A., van Eyghen, H., Gayet, S., Groenwold, R., Hernaus, D., Huijnen, P., Jonker, N., de Kleijn, R., Kroll, C. F., Krypotos, A.-M., van der Laan, N., Luijken, K., Meijer, E., Pear, R. S. A., Peels, R.,…de Winter, J. (2024). Replication studies in the Netherlands: Lessons learned and recommendations for funders, publishers and editors, and universities. *Accountability in research*, 1-19. https://doi.org/10.1080/08989621.2024.2383349

Eddelbuettel, D., & Balamuta, J. J. (2018). Extending R with C++: a brief introduction to Rcpp. *The American Statistician*, *72*(1), 28-36. https://doi.org/10.1080/00031305.2017.1375990

Eddelbuettel, D., & Francois, R. (2011). Rcpp: Seamless R and C++ Integration. *Journal of Statistical Software*, *40*(8), 1 - 18. https://doi.org/10.18637/jss.v040.i08

Eddelbuettel, D., & Sanderson, C. (2014). RcppArmadillo: Accelerating R with high-performance C++ linear algebra. *Computational statistics & data analysis*, *71*, 1054-1063. https://doi.org/10.1016/j.csda.2013.02.005

Epskamp, S., & Nuijten, M. (2014). statcheck: Extract statistics from articles and recompute p values (R package version 1.0. 0.). https://cran.r-project.org/web/packages/statcheck/index.html

Gutjahr, W. J. (2011). Recent trends in metaheuristics for stochastic combinatorial optimization. *Central European Journal of Computer Science*, *1*(1), 58-66. https://doi.org/10.2478/s13537-011-0003-3

Harrison, R. L. (2010). Introduction To Monte Carlo Simulation. *AIP Conf Proc*, *1204*, 17-21. https://doi.org/10.1063/1.3295638

Hartgerink, C., Voelkel, J. G., Wicherts, J. M., & van Assen, M. (2019). Detection of data fabrication using statistical tools. https://doi.org/10.31234/osf.io/jkws4

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, *57*(1), 97-109. https://doi.org/10.1093/biomet/57.1.97

Heathers, J. A., Anaya, J., van der Zee, T., & Brown, N. J. (2018). Recovering data from summary statistics: Sample parameter reconstruction via iterative techniques (SPRITE). https://doi.org/10.7287/peerj.preprints.26968v1

Ioannidis, J. P. (2008). Why most discovered true associations are inflated. *Epidemiology*, *19*(5), 640-648. https://doi.org/10.1097/EDE.0b013e31818131e7

Ioannidis, J. P. A. (2005). Why Most Published Research Findings Are False. *PLoS medicine*, *2*(8), e124. https://doi.org/10.1371/journal.pmed.0020124

Kennedy, J., Eberhart, R., & Proceedings of, I. I. C. o. N. N. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (pp. 1942-1948 vol.1944). https://doi.org/10.1109/ICNN.1995.488968

Kirkpatrick, S., Gelatt, C. D., Jr., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*(4598), 671-680. https://doi.org/10.1126/science.220.4598.671

Korte, B. H., & Vygen, J. (2018). *Combinatorial optimization : theory and algorithms* (Sixth edition ed.). Springer. https://doi.org/10.1007/978-3-662-56039-6

Le Forestier, J. M., Page-Gould, E., & Chasteen, A. (2024). Identity Concealment May Discourage Health-Seeking Behaviors: Evidence From Sexual-Minority Men During the 2022 Global Mpox Outbreak. *Psychological Science*, *35*(2), 126-136. https://doi.org/10.1177/09567976231217416

Lenormand, M., & Deffuant, G. (2013). Generating a Synthetic Population of Individuals in Households: Sample-Free Vs Sample-Based Methods. *Journal of Artificial Societies and Social Simulation*, *16*(4), 12. https://doi.org/10.18564/jasss.2319

Meehl, P. E. (1978). Theoretical risks and tabular asterisks: Sir Karl, Sir Ronald, and the slow progress of soft psychology. *Applied and Preventive Psychology*, *11*(1), 1-1. https://doi.org/10.1016/j.appsy.2004.02.001

Meehl, P. E. (1990). Why Summaries of Research on Psychological Theories are Often Uninterpretable. *Psychological Reports*, *66*(1), 195-244. https://doi.org/10.2466/pr0.1990.66.1.195

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087-1092. https://doi.org/10.1063/1.1699114

Metropolis, N., & Ulam, S. (1949). The Monte Carlo Method. *Journal of the American Statistical Association*, *44*(247), 335-341. https://doi.org/10.2307/2280232

Miyakawa, T. (2020). No raw data, no science: another possible source of the reproducibility crisis. *Molecular brain*, *13*(1), 24. https://doi.org/10.1186/s13041-020-0552-2

Nosek, B. A., Hardwicke, T. E., Moshontz, H., Allard, A., Corker, K. S., Dreber, A., Fidler, F., Hilgard, J., Kline Struhl, M., Nuijten, M. B., Rohrer, J. M., Romero, F., Scheel, A. M., Scherer, L. D., Schönbrodt, F. D., & Vazire, S. (2022). Replicability, Robustness, and Reproducibility in Psychological Science. *Annual Review of Psychology*, *73*, 719-748. https://doi.org/10.1146/annurev-psych-020821-114157

Open-Science-Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*(6251), aac4716. https://doi.org/doi:10.1126/science.aac4716

Parsopoulos, K. E., & Vrahatis, M. N. (2002). Recent approaches to global optimization problems through particle swarm optimization. *Natural computing*, *1*, 235-306. https://doi.org/10.1023/A:1016568309421

Reynolds, M., & Besner, D. (2008). Contextual effects on reading aloud: Evidence for pathway control. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *34*(1), 50-64. https://doi.org/10.1037/0278-7393.34.1.50

Schimmack, U. (2020). A meta-psychological perspective on the decade of replication failures in social psychology. *Canadian Psychology / Psychologie canadienne*, *61*(4), 364-376. https://doi.org/10.1037/cap0000246

Selman, B., & Gomes, C. P. (2006). Hill-climbing Search. In *Encyclopedia of cognitive science*. https://doi.org/10.1002/0470018860.s00015

Silberzahn, R., Uhlmann, E. L., Martin, D. P., Anselmi, P., Aust, F., Awtrey, E., Bahník, Š., Bai, F., Bannard, C., Bonnier, E., Carlsson, R., Cheung, F., Christensen, G., Clay, R., Craig, M. A., Dalla Rosa, A., Dam, L., Evans, M. H., Flores Cervantes, I.,…Nosek, B. A. (2018). Many Analysts, One Data Set: Making Transparent How Variations in Analytic Choices Affect Results. *Advances in Methods and Practices in Psychological Science*, *1*(3), 337-356. https://doi.org/10.1177/2515245917747646

Skiena, S. S. (2008). *The Algorithm Design Manual*. Springer Publishing Company, Incorporated. https://doi.org/0.1007/978-1-84800-070-4

Snijders, T. A. B., & Bosker, R. J. (2012). *Multilevel analysis : an introduction to basic and advanced multilevel modeling* (2nd edition ed.). Sage.

Spikman, J. M., Timmerman, M. E., Zomeren van, A. H., & Deelman, B. G. (1999). Recovery versus retest effects in attention after closed head injury. *J Clin Exp Neuropsychol*, *21*(5), 585-605. https://doi.org/10.1076/jcen.21.5.585.874

Steegen, S., Tuerlinckx, F., Gelman, A., & Vanpaemel, W. (2016). Increasing Transparency Through a Multiverse Analysis. *Perspectives on Psychological Science*, *11*(5), 702-712. https://doi.org/10.1177/1745691616658637

Templ, M., Meindl, B., Kowarik, A., & Dupriez, O. (2017). Simulation of Synthetic Complex Data: The R Package simPop. *Journal of Statistical Software*, *79*(10), 1 - 38. https://doi.org/10.18637/jss.v079.i10

van Ravenzwaaij, D., Bakker, M., Heesen, R., Romero, F., van Dongen, N., Crüwell, S., Field, S. M., Held, L., Munafò, M. R., Pittelkow, M. M., Tiokhin, L., Traag, V. A., van den Akker, O. R., van 't Veer, A. E., & Wagenmakers, E. J. (2023). Perspectives on scientific error. *R Soc Open Sci*, *10*(7), 230448. https://doi.org/10.1098/rsos.230448

Wang, D., Tan, D., & Liu, L. (2018). Particle swarm optimization algorithm: an overview. *Soft Computing : A Fusion of Foundations, Methodologies and Applications*, *22*(2), 387-408. https://doi.org/10.1007/s00500-016-2474-6

Wicherts, J. M., Borsboom, D., Kats, J., & Molenaar, D. (2006). The poor availability of psychological research data for reanalysis. *Am Psychol*, *61*(7), 726-728. https://doi.org/10.1037/0003-066x.61.7.726

Zhang, Y., Jiang, Q., Luo, Y., & Liu, J. (2025). *Can One Donation a Day Keep Depression Away? Three Randomized Controlled Trials of an Online Micro-Charitable Giving

Intervention. *Psychological Science*, *36*(2), 102-115.
https://doi.org/10.1177/09567976251315679

Appendix A

Search Space Characteristics

Descriptives

If an element in the vector to be optimized can take K discrete values, there are K^N possible ordered vectors, and in the continuous case, the volume of the feasible search space scales with the side length to the Nth power. In either setting, the number of candidate solutions (or the space that must be explored) grows exponentially as N increases. However, all permutations of the same multiset give identical means and standard deviations, thus, the number of distinct multisets of size N drawn from K values is

$$\binom{N+K-1}{N} = \frac{(N+K-1)!}{N!(K-1)!},$$

which still grows very rapidly in N. In practice, the algorithm operates on ordered vectors, so it sees the full discrete K^N (or continuous N-dimensional) search space, while the objective function is partially permutation-invariant.

ANOVA

If each entry of the outcome vector D can take K discrete values, there are K^N ordered vectors to explore, and in the continuous case the search region is an N-dimensional hypercube whose volume grows as the Nth power of its side length. Unlike the descriptive module, where permutation invariance of f holds across all observations, the F statistics and thus the objective function f in the ANOVA modules is invariant only under permutations within each group. In other words, rearranging the values of D inside a group does not affect F, but swapping values between groups does. Accordingly, the number of distinct assignments of the D (with repetition allowed) up to this invariance is

$$\prod_{j=1}^{G} \binom{n_j+K-1}{n_j},$$

where G is the number of groups and n_j the size of group j. Even though this reduces the search space relative to K^N, in practice the optimizers operate over full discrete K^N (or continuous N-dimensional) space.

Multiple Linear Regression

The optimization algorithm must find a solution in an ultra-high-dimensional search space with the number of dimensions

$$D = N(p+1).$$

In a purely integer setting with K allowable values per cell, there are $K^{N(p+1)}$ possible data sets, and in the continuous or mixed setting, the feasible region is the hypercube

$$[a, b]^{N(p+1)} ,$$

whose volume grows exponentially in both N and p.

The reformulation into a permutation problem where Y is kept fixed (only predictor values are modified) replaces the continuous hypercube search over $[a, b]^{N(p+1)}$ with a discrete permutation space $(N!)^p$, collapsing an uncountable continuum into a finite graph whose nodes correspond to within-column re-orderings. Moreover, the resulting search landscape, while still nonconvex, is defined over a discrete space where each local optimum is reachable via a sequence of simple transpositions.

Linear Mixed-Effects Regression

The search space to be explored by the optimization process in the LME module is ultra-high-dimensional with the number of dimensions

$$D = N[p_b + (p_w + 1)time] .$$

If the data are solely integer values with K allowable values for each element, there are $K^{N[p_b+(p_w+1)time]}$ possible data sets, and in the continuous or mixed setting, the feasible region is the hypercube

$$[a, b]^{N[p_b+(p_w+1)time]} ,$$

whose volume grows exponentially in N, p_b, p_w, and $time$. By permuting the entries of each predictor column (Y is kept fixed) within each subject and time-group (for between-subject variables, swapping subject profiles), I collapse the infinite continuous search space into a finite permutation space

$$(N!)^{p_b+p_w time} .$$

Moreover, the resulting search landscape, while still nonconvex, is defined over a discrete space where each local optimum is reachable via permutations and thus, combinatorial optimization.

Appendix B

Candidate Handling for Integer Data

Candidate Modification for Integer Data in the Descriptives Module

For integer variables $x = (x_i, \dots, x_N)$ with possible values within allowable bounds $[L, U]$ the algorithm applies two types of moves: a purely *stochastic move* and a *heuristic move*. The *stochastic move* begins with randomly selecting an index i of the candidate vector and determining the allowed integers in the range

$$v_i = (L, L+1, \dots, U) \setminus (x_i) ,$$

if v_i is nonempty, the value x_i is updated with a uniform random draw from v_i. Because this update affects only one component and always picks from the allowable support, it preserves the integer constraints while introducing a purely random perturbation. The move will result in a change in the objective value by affecting both the $mean$ and SD.

The *heuristic move* is inspired by Heathers et al. (2018) SPRITE algorithm and begins by computing the current standard deviation s of the candidate vector and comparing it to the target σ: if $s < \sigma$, the algorithm aims to increase dispersion; if $s > \sigma$, it seeks to reduce or maintain it. It then identifies uniformly at random a decrement index dec from $x_i > L$ (when increasing dispersion, it explicitly excludes the current maximum to avoid shrinking the empirical extremum), and an increment index inc from $x_i < U$ (when reducing dispersion, it further restricts these to $x_i < x_{dec}$). The maximum transferable amount is determined as

$$\delta_{max} = \min(x_{dec} - L, U - x_{inc}) ,$$

and if $\delta_{max} \geq 1$, δ is uniformly drawn at random from $(1, \dots, \delta_{max})$. The values are updated with

$$x_{dec} = x_{dec} - \delta , \quad x_{inc} = x_{inc} + \delta .$$

If no feasible δ exists ($\delta_{max} < 1$), the move is aborted. Because each move subtracts and adds the same mass, the sum of x, thus its mean, is preserved while its dispersion is nudged towards σ.

Candidate Initialization for Integer Data in the ANOVA Module

When dealing with integer data in the ANOVA context, I let

$$total_j = tMean_j \times n_j ,$$

where $tMean_j$ is the target $group\ mean$ and n_j the $subgroup\ size$ of group j. I write

$$total_j = q_j n_j + r_j , \quad q_j = \left\lfloor \frac{total_j}{n_j} \right\rfloor , \; r_j = total_j \bmod n_j ,$$

to construct the vector for subgroups by assigning $q_j + 1$ to exactly r_j observations and q_j to the remaining $n_j - r_j$. Since

$$(q_j + 1)r_j + q_j(n_j - r_j) = total_j \ ,$$

the sample mean for each group is

$$tMean_j = \frac{total_j}{n_j} \ ,$$

and only the two adjacent integer values are used to determine the values for each subgroup. These are combined into the full initial candidate outcome vector D, which results (in conjunction with M) in means equal to the reported $group\ means$.

Appendix C

Optimization Algorithms

Particle Swarm Optimization

Candidate Modification. For continuous variables, during each iteration, every particle's velocity is recomputed as the sum of three contributions: its previous velocity scaled by an inertia weight, a pull toward its own best-seen position scaled by a cognitive weight, and a pull toward the swarm's global best scaled by a social weight (Kennedy et al., 1995; Wang et al., 2018). Independent uniform random terms modulate each pull to preserve exploration. The updated velocity is subsequently added to the particle's current position to propose a new point, after which both velocity and position are clipped to the user's lower and upper bounds to maintain feasibility (Parsopoulos & Vrahatis, 2002). This loop of momentum, personal-best attraction, global-best attraction, and boundary enforcement is applied by using `psoptim()` from the R package *pso* (Bendtsen, 2022).

Acceptance Criteria. In the continuous-variable branch, acceptance simply governs how new positions update each particle's personal and the swarm's global best records. After computing and applying the velocity update, each particle's new position is f_p. If this value improves upon the particle's historical best, $f_p < f_{pbest}$, the new position is unconditionally accepted as the updated personal best. Independently, if it also outperforms the swarm's current global best, $f_p < f_{gbest}$, it replaces that record as well (for formal equations see Wang et al. (2018)). Otherwise, both remain unchanged. This deterministic acceptance rule ensures that only improvements influence the stored best solutions, while the stochastic velocity updates continue to drive exploration (Kennedy et al., 1995).

Convergence Criteria. For continuous variables, the convergence relative to the *tolerance* is managed by the underlying `psoptim()` call. As particles evolve, f_{gbest} is compared at each iteration to the $abstol$ parameter (equal to the user's $tolerance$). Therefore, if $f_{gbest} < tolerance$, the optimization converges. Otherwise, the algorithm stops when it either (a) reaches the maximum number of iterations ($maxit\ PSO$) or (b) tracks $\frac{maxit\ PSO}{3}$ consecutive iterations without improvement of f_{gbest}.

Simulated Annealing

Acceptance Criteria. For integer variables, candidate updates follow a simulated-annealing scheme (Kirkpatrick et al., 1983). Each proposed candidate is first evaluated by the objective function, $f_{candidate}$, and compared to the current state, $f_{current}$. If $f_{candidate} < f_{current}$ the move is accepted unconditionally. Otherwise, it is accepted with probability

$$p = \min\left(1, e^{(f_{current} - f_{candidate})/T}\right)$$

derived from the original Metropolis-algorithm (Metropolis et al., 1953) and where T is the current $temperature$. The exponential term yields a probability between zero and one that decreases both as the deterioration

$$\Delta f = f_{candidate} - f_{current}$$

grows and as the temperature T declines over iterations. T gets multiplied by the $cooling\ rate$ after each iteration. This allows occasional uphill moves, especially at high T, and helps escape local minima, while progressively reducing the acceptance of inferior solutions (for details I refer to Delahaye et al. (2019)). Only accepted improvements $f_{candidate} < f_{current}$ update the recorded best solution; accepted uphill moves merely replace the current state. Therefore, and as the acceptance probability explicitly depends on the nonstationary cooling schedule, the search trajectory does *not* form a Markov chain as in Metropolis-Hastings sampling (Hastings, 1970).

Hill Climb Optimization

Hill-climbing is a greedy local search routine that iteratively refines a candidate solution by exploring its immediate neighbourhood and accepting only those moves that decrease the objective function (Selman & Gomes, 2006; Skiena, 2008). This low-memory strategy converges rapidly to a local optimum but can become trapped without escape mechanisms, making it well suited for refining the best solutions produced by simulated-annealing optimizations.

Candidate Initialization. The hill-climbing phase begins with the best candidate X_{best} obtained from the preceding simulated annealing routine of the LM or LME module. Its associated best objective value is computed and stored as both the current error $f_{current}$ and the best error f_{best}. This initialization anchors the local search on the most promising solution produced by the meta-heuristic optimizer.

Candidate Modification. At each iteration t, a neighbourhood of size m is generated via repeated application of local moves. When refining the solution from the LM module ($lme = FALSE$), the algorithm randomly selects a predictor column j and two distinct row indices $i \neq k$, updates the data with the swap

$$X_{i,j}^{(t+1)} = X_{k,j}^{(t)}, \qquad X_{k,j}^{(t+1)} = X_{i,j}^{(t)},$$

leaving every other entry unchanged. The data in long-format W^{long} from the LME module ($lme = TRUE$) is first converted to wide-format W^{wide}, and subsequently the same two-row swap is applied in one randomly chosen column of W^{wide}. By restricting each proposal

to a simple swap perturbation, the method systematically explores the immediate vicinity of the current solution.

Candidate Evaluation. Every proposed neighbour X is evaluated by the objective function $f(X)$ of the LM or LME module, respectively. Among the m candidates, the current candidate $X_{current}$ with minimal objective value $f_{current}$ is selected for possible acceptance.

Acceptance Criteria. The hill climbing optimization is greedy and thus, only if $f_{current} < f_{best}$ the candidate is updated

$$X^{(t+1)} = X_{best} = X_{current} \text{ and } f^{(t+1)} = f_{best} = f_{current} .$$

Otherwise, the candidate remains unchanged

$$X^{(t+1)} = X^{(t)} \text{ and } f^{(t+1)} = f^{(t)} .$$

This strictly greedy acceptance rule guarantees that the error never increases, and the algorithm moves to a local optimum.

Convergence. After a maximum number of iterations $hill\ climbs$, the routine returns X_{best} and f_{best}. By exploiting these fine grained, purely downhill steps, the hill climbing optimization often achieves further improvements of the solution beyond the reach of simulated annealing, without requiring gradient information or additional tuning.

Appendix D

Performance Enhancement

OLS Solution for Multiple Linear Regression using Matrix Algebra

To obtain the estimated coefficient vector $\hat{b}$ and SE vector $\widehat{SE}$ I calculate the OLS solution given $W(X, Y)$ via matrix algebra. Let X be the design matrix including the intercept with q free parameters and Y the outcome vector, I calculate the OLS solution via matrix algebra to obtain the estimated coefficient vector

$$\hat{b} = (X^T X)^{-1} X^T Y \ .$$

The residual variance is

$$\hat{\sigma}^2 = \frac{\left(Y - X\hat{b}\right)^T \left(Y - X\hat{b}\right)}{N - q}$$

thus, the variance-covariance matrix of $\hat{b}$ is

$$\widehat{Var}\left(\hat{b}\right) = \hat{\sigma}^2 (X^T X)^{-1} \ ,$$

so that each standard error is

$$\widehat{SE}\left(\hat{b}_j\right) = \sqrt{\left[\widehat{Var}\left(\hat{b}\right)\right]_{jj}} \ .$$

Objective Weights Estimation

The algorithm's parameters like $max\ iterations$, $maxit\ PSO$, $temperature$, $cooling\ rate$, and $max\ starts$ mainly control accuracy and runtime, and their default values usually yield reasonable performance on empirical examples. In contrast, the $weights$, ω_1 and ω_2, assigned to the objective function f, directly affect convergence. Poorly chosen $weights$ can prevent the optimization process from converging within practical time limits, whereas well-chosen $weights$ balance the contribution of multiple objective terms effectively.

Univariate Distribution Weights. While the ANOVA module only has a single objective (F values), I automated calibration of the two objective function $weights$ in the descriptive module using a Monte-Carlo routine (Harrison, 2010; Metropolis & Ulam, 1949). That technique requires only input summary estimates, the number of iterations $est\ iter$, an upper bound on weight magnitude $max\ weight$, and a choice of summary $metric$. For a variable, the function generates $est\ iter$ candidate vectors in accordance with the module's random *candidate initialization* and then computes the f error terms in replication i by

$$e_{\mu}^{(i)} = \omega_1 \sqrt{\Delta_{\mu}^2} \ , \qquad e_{\sigma}^{(i)} = \omega_2 \sqrt{\Delta_{\sigma}^2} \ ,$$

where $\omega_1 = \omega_2 = 1$. For a fixed vector, the ratio of the error terms is defined as

$$\vartheta^{(i)} = \frac{e_{\mu}^{(i)}}{e_{\sigma}^{(i)}} \qquad i = 1, \dots, est\ iter\ .$$

These ratios are summarized by

$$\hat{\vartheta} = S\left[\left(\vartheta^{(i)}\right)_{i=1}^{est\ iter}\right],$$

where the $metric$ S is either the mean or median. Clamp $\hat{\vartheta}$ to interval

$$\left[\frac{1}{Max\ Weight}, Max\ Weight\right] \text{ via}$$

$$\bar{\vartheta} = \min\left[\max\left(\hat{\vartheta}, \frac{1}{Max\ Weight}\right), Max\ Weight\right],$$

and finally, normalize so that the smaller of the two $weights$ equals one:

$$(\omega_1, \omega_2) = \frac{(1, \bar{\vartheta})}{\min(1, \bar{\vartheta})}\ .$$

Inserting these data-driven ω_1 and ω_2 in

$$f(x) = \omega_1\sqrt{\Delta_{\mu}^2} + \omega_2\sqrt{\Delta_{\sigma}^2}\ ,$$

ensures that, at the start of optimization, neither optimization target dominates. This largely eliminates manual trial-and-error and significantly improves convergence reliability with different reported summary estimates.

Multivariate Distribution Weights. In the LM and LME optimization modules the two objective $weights$ refer to the $correlation$ ω_1 versus the $regression/SE$ ω_2 matching term. To automate the choice of these $weights$ I developed an estimation routine, which uses pilot simulation run(s) to calibrate their ratio. For a specified number of sequential runs, each iteration calls either the LM or the LME module (if $parallel\ start\ >\ 1$ then `parallel_lm()` and `parallel_lme()` are used in each run), with initial $weights$ set $\omega_1 = \omega_2 = 1$ and records per iteration i the error ratios ϑ,

$$\vartheta^{(i)} = \frac{\omega_r RMSE_r}{\omega_2 RMSE_{b,SE}} \qquad i = 1, \dots, Max\ Iter\ ,$$

from the optimizer's trace. To stabilize this estimate, I extract the most recent $pool\ range$ unique values of $\vartheta^{(i)}$ from each run j and compute their mean, yielding a single summary ratio $\hat{\vartheta}_j$. Across all runs, these per-run estimates are then used to obtain the overall calibration ratio

$$\bar{\vartheta} = \frac{1}{runs}\sum_{j}^{runs}\ \hat{\vartheta}_j\ ,$$

to account for between run variability. Finally, the $weights$ are set

$$(\omega_1, \omega_2) = \frac{(\bar{\vartheta}, 1)}{\min(\bar{\vartheta}, 1)}\ ,$$

so that the smaller of the two $weights$ is normalized to one and the other reflects the estimated scale difference between the two error components near convergence. Inserting these data-driven $weights$ in

$$f(W) = \omega_1 RMSE_r + \omega_2 RMSE_{b,SE} \ ,$$

eliminate manual trial-and-error in $weight$ selection and ensures that neither objective term overwhelms the other and thus, improves convergence reliability.

Algorithm Parameter Tuning

The simulated-annealing framework in each module is governed by four interrelated tuning parameters which jointly determine the algorithm's performance. $Max\ iterations$ determines the total number of candidate generation steps per restart, with larger values allowing for more extensive search but increasing runtime linearly. The initial $temperature$ T_0 sets the starting acceptance probability of uphill moves: higher T_0 encourages broader traversal of the objective surface, reducing the risk of early entrapment in local minima, whereas a lower T_0begins the search more greedily. After each iteration i, the $temperature$ is updated via

$$T_i = \alpha T_{i-1} \qquad \alpha \in (0,1) \ ,$$

where the $cooling\ rate\ \alpha$ controls how quickly the algorithm transitions from exploration to exploitation. A value closer to one slowed the $temperature$ decay, maintaining higher acceptance of uphill moves for longer, while a smaller value accelerates convergence but risks premature trapping. Finally, $max\ starts$ specifies the number of dependent restarts: upon exhaustion of $max\ iterations$ without meeting the convergence $tolerance$, the algorithm resets the temperature to T_0/T_s for starts s and begins again from the best candidate found so far. By tuning these parameters - raising $max\ iterations$ and increasing $max\ starts$ - users can trade additional computation time for greater reliability in locating a global optimum, whereas more aggressive cooling and temperature favour speed when approximating solutions.

In practice, it is most effective to begin by extending the total search effort, either by raising $max\ iterations$ or increasing $max\ starts$ (or both), until the wall-clock runtime remains acceptable. Once sufficient search depth has been ensured, one can then tune the cooling schedule: choose a relatively high initial $temperature$ (e.g., $T_0 \geq .5$) to allow ample uphill moves during early iterations, and only then experiment with more aggressive $cooling\ rates$ to speed convergence. This two-stage approach first guarantees broad exploration via more iterations or restarts and then refines the exploration vs. exploitation

balance by lowering T_0 and/or adjusting the $cooling\ rate$. Ultimately, this helps avoid the premature stagnation that often accompanies low-temperature settings in under-resourced runs.

Computational Efficiency

To accelerate computation, each module supports parallel execution. In the descriptive module, setting the argument `parallel = TRUE` initializes a parallel backend, enabling the `optim_vec()` function to optimize multiple vectors concurrently. Similarly, the specialized functions, `parallel_aov()`, `parallel_lm()`, and `parallel_lme()`, are provided in the ANOVA, linear model, and linear mixed effects modules, respectively. These functions allow users to generate, optimize, and return multiple data sets from summary statistics in parallel. When `return_best_solution = TRUE`, the parallel backend evaluates all simulated data sets internally and returns the candidate with the lowest objective function value, ensuring efficient identification of the optimal solution while considering between run variability.

The repeated evaluation of the objective function is implemented in C++ via the *Rcpp* package (Eddelbuettel & Francois, 2011), which compiles critical routines into native code for execution. Because C++ operates at a lower level than R, providing direct memory management and more efficient looping and arithmetic (e.g., via the C++ Standard Library *cmath*), these core computations run an order of magnitude faster than their R counterparts (Eddelbuettel & Balamuta, 2018). In addition, the linear algebra operations of the closed form OLS solution in the LM module are delegated to *RcppArmadillo* (Eddelbuettel & Sanderson, 2014), a high-performance C++ library optimized for matrix computations. These combined speedups are especially pronounced when iterating thousands of times to simulate and optimize large data sets, yielding substantial reductions in total runtime and thus, a noticeably smoother experience for the user. Although the same C++ infrastructure could be extended to the ANOVA and LME modules, this enhancement is non-trivial and has not yet been implemented.

Appendix E

DISCOURSE Application

Descriptives & LM Module

In the replication attempt by Bardwell et al. (2007) patients with obstructive sleep apnea completed both fatigue and depression scales to examine whether mood symptoms or apnea severity better predict daytime fatigue. The original study's (Bardwell et al., 2003) raw data are not publicly available. Here, I apply the Descriptives and LM module of the DISCOURSE framework to simulate a synthetic data set that reproduces their reported summary estimates. The R scripts are available on the OSF at https://osf.io/3yjaq/ and a tutorial article can be accessed at https://sebastian-lortz.github.io/discourse/.

Step 1. I began by extracting the relevant descriptive parameters from the article.

```
N = 60
target_mean <- c(48.8, 17.3, 12.6, 10.8)
names(target_mean) <- c("Apnea.1", "Apnea.2", "Depression",
"Fatigue")
target_sd <- c(27.1, 20.1, 11.3, 7.3)
integer = c(FALSE, FALSE, TRUE, TRUE)
range_matrix <- matrix(c(15, 0, 0, 0,
                         111, 80.9, 49, 28),
                       nrow = 2, byrow = TRUE)
```

I subsequently estimated the weights

```
weight.vec <- weights_vec(
  N = N,
  target_mean =  target_mean,
  target_sd =  target_sd,
  range = range_matrix,
  integer = integer
)
```

and I ran the Descriptives module with default hyperparameters.

```
result.vec <- optim_vec(
  N = N,
  target_mean = target_mean,
  target_sd = target_sd,
  range = range_matrix,
  integer = integer,
  obj_weight = weight.vec,
  tolerance = 1e-8,
  max_iter = 1e5,
```

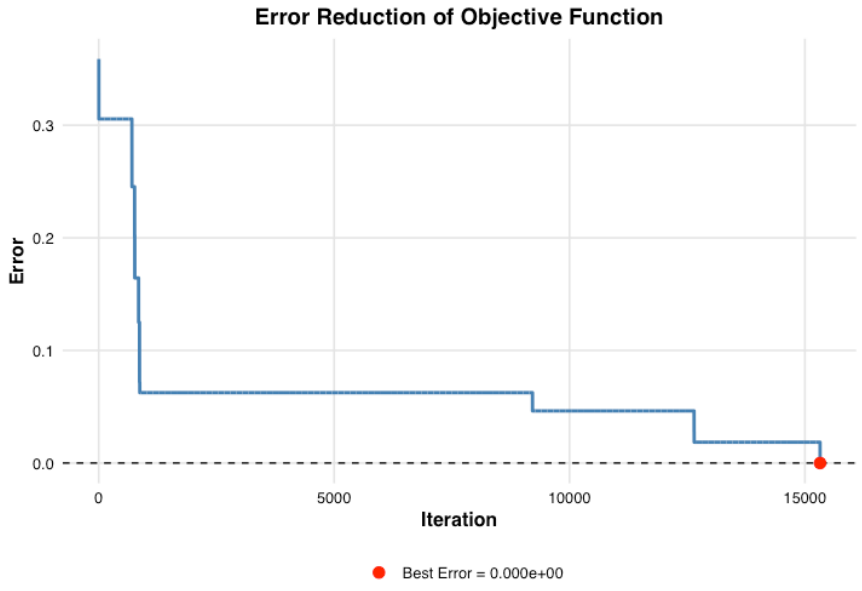
```
  max_starts = 3,
  init_temp = 1,
  cooling_rate = NULL
)
```

The optimization converged exactly. Inspecting

```
summary(result.vec)
      DISCOURSE Object Summary
      ------------------------------------------------

      Achieved Loss of Optimization:
         Apnea.1    Apnea.2 Depression    Fatigue
               0          0          0          0

      RMSE of Summary Statistics
        Means:  0
        SDs:    0

      Means:
         Apnea.1    Apnea.2 Depression    Fatigue
        48.77024   17.26942   12.55000   10.83333

      SDs:
         Apnea.1    Apnea.2 Depression    Fatigue
       27.117733  20.140629  11.329316   7.258348
```

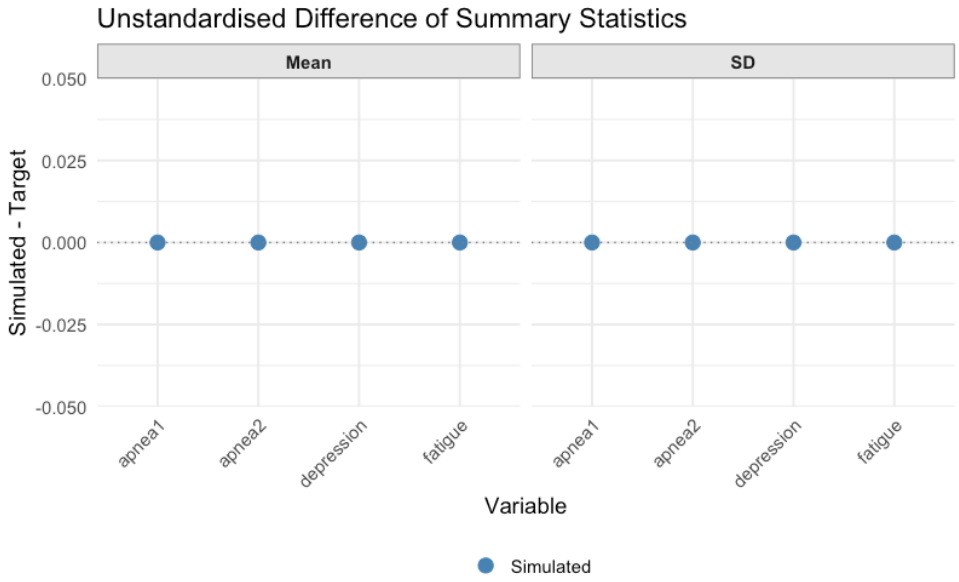
confirms that the simulated data reproduce the published means and SD. I then visualized the error trajectories. For example, the error trajectory of the fatigue variable is:

```
plot_error(result.vec, run = 4)
```



The estimated versus target descriptives are given by

```
plot_summary(result.vec, standardised = FALSE)
```



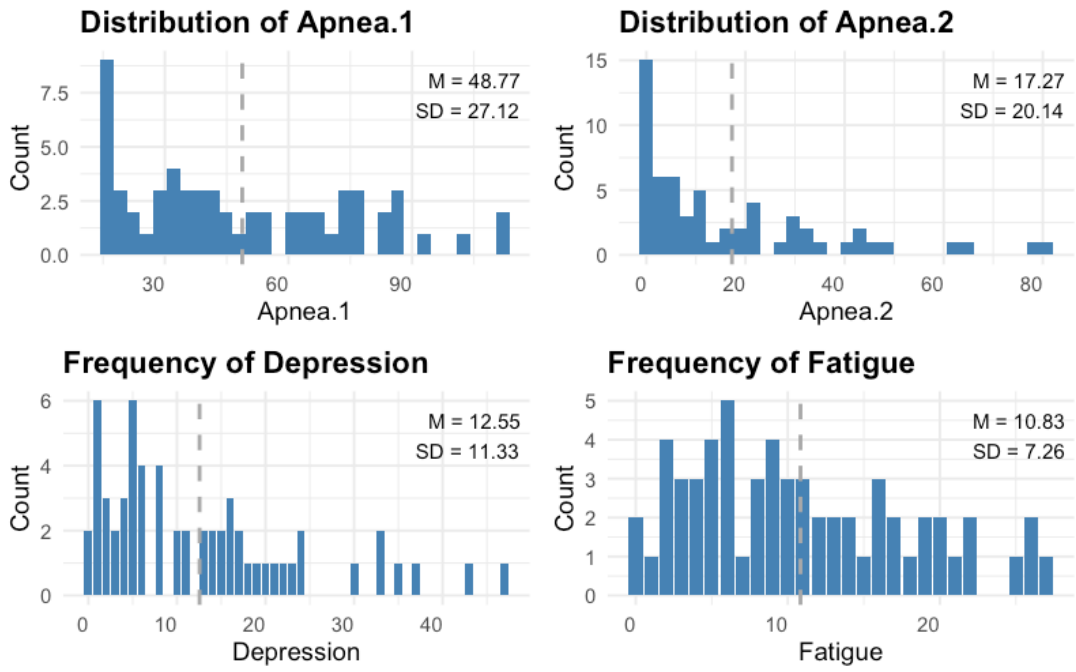
Finally, I saved the RMSE and relevant statistics,

```
get_rmse(result.vec)
      $rmse_mean
      [1] 0

      $rmse_sd
get_stats(result.vec)
      $mean
         Apnea.1    Apnea.2 Depression    Fatigue
        48.77024   17.26942   12.55000   10.83333

      $sd
         Apnea.1    Apnea.2 Depression    Fatigue
       27.117733  20.140629  11.329316   7.258348
```

extracted the simulated data set and inspected its distributions and frequencies.

```
data.vec <- result.vec$data
head(data.vec)
         Apnea.1   Apnea.2 Depression Fatigue
      1 38.98685  9.156641          3       3
      2 40.31700 61.539320          3       2
      3 66.29080  5.880320         10       5
      4 42.25313  0.000000         16      16
      5 32.68454  1.261708         33      11
      6 38.38836 16.411990          8       3
```

```
gridExtra::grid.arrange(grobs = plot_histogram(data.vec), ncol = 2)
```



Step 2. I began by extracting the relevant correlation and regression parameters from the article and handing off the simulated data from the Descriptives module for further use.

```
sim_data <- data.vec
target_reg <- c(4.020, 0.023, 0.008, 0.438)
names(target_reg) <- c("Apnea.1", "Apnea.2", "Depression", "Fatigue")
target_se <- c(NA, 0.034, 0.048, 0.066)
target_cor <-  c(NA, NA, NA, 0.11, 0.20, 0.68)
reg_equation <- "Fatigue ~ Apnea.1 + Apnea.2 + Depression"
```

I subsequently estimated the weights

```
result.weight.lm <- weights_est(
  module = "lm",
  sim_runs = 1,
  sim_data = sim_data,
  reg_equation = reg_equation,
  target_cor = target_cor,
  target_reg = target_reg,
  target_se = target_se,
  tol = 1e-8,
  max_iter = 1e5,
  init_temp = 1,
  cooling_rate = NULL
)
weight.lm <- result.weight.lm$weights
weight.lm
```

```
[1] 1.00 1.14
```

and I ran the LM module with default hyperparameters.

```
result.lm <- optim_lm(
  sim_data = sim_data,
  reg_equation = reg_equation,
  target_cor = target_cor,
  target_reg = target_reg,
  target_se = target_se,
  weight = weight.lm,
  tol = 1e-8,
  max_iter = 1e5,
  max_starts = 1,
  init_temp = 1,
  cooling_rate = NULL
)
```

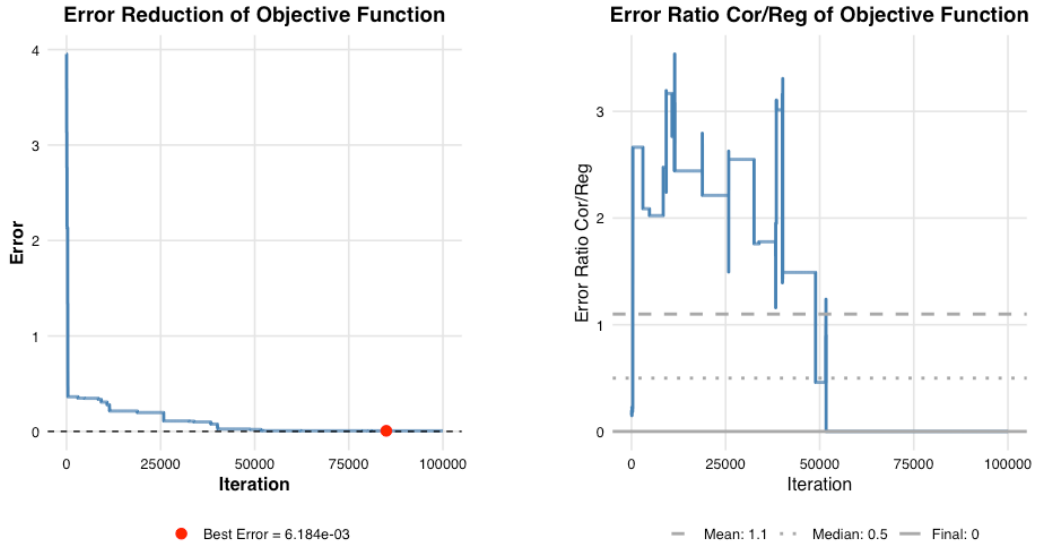
The optimization converged with reaching $Max\ Iterations$ and $RMSE < .01$. Inspecting

```
summary(result.lm)
      DISCOURSE Object Summary
      ------------------------------------------------

      Achieved Loss of Optimization:  0.006184284

      RMSE of Summary Statistics
        Correlations:             0
        Regression Coefficients:  0.002236068
        Standard Errors:          0.007874008

      Regression Model:
      "Fatigue ~ Apnea.1 + Apnea.2 + Depression"

      Simulated Data Summary:
        Coefficients:
      (Intercept)     Apnea.1     Apnea.2  Depression
      4.020274448 0.025432823 0.007570258 0.433622314

        Std. Errors:
      (Intercept)     Apnea.1     Apnea.2  Depression
       1.72534792  0.02613568  0.03671927  0.06469461

        Correlations:
```

```
[1] -0.13491220  0.02074421  0.28631735  0.10622519  0.20197415
0.68481225

  Means:
   Apnea.1    Apnea.2 Depression    Fatigue
  48.77024   17.26942   12.55000   10.83333

  SDs:
   Apnea.1    Apnea.2 Depression    Fatigue
 27.117733  20.140629  11.329316   7.258348
```

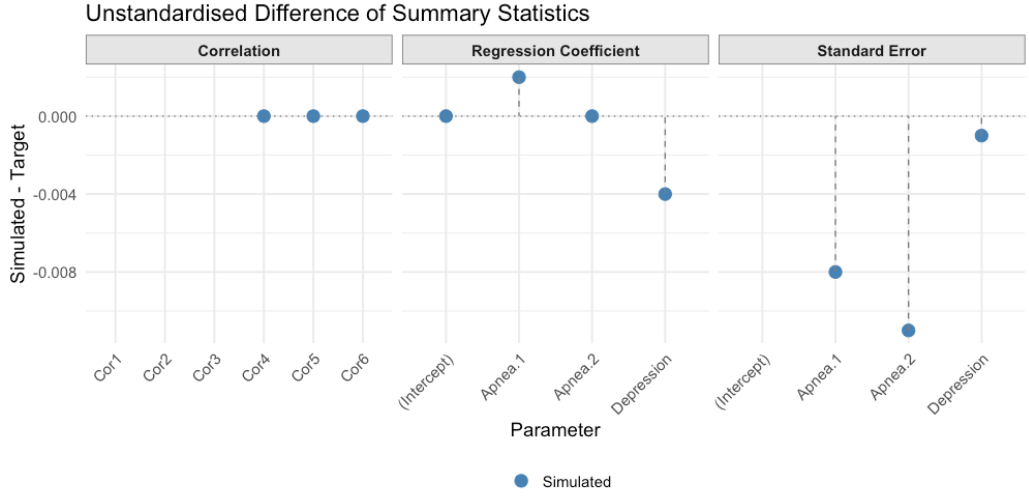
confirms that the simulated data closely reproduce the published regression and correlation parameters. I then visualized the error and error-ratio trajectories

```
plots <- list(plot_error(result.lm), plot_error_ratio(result.lm))
gridExtra::grid.arrange(grobs = plots, ncol = 2)
```



and illustrated the estimated versus target descriptives with

```
plot_summary(result.lm, standardised = FALSE)
```

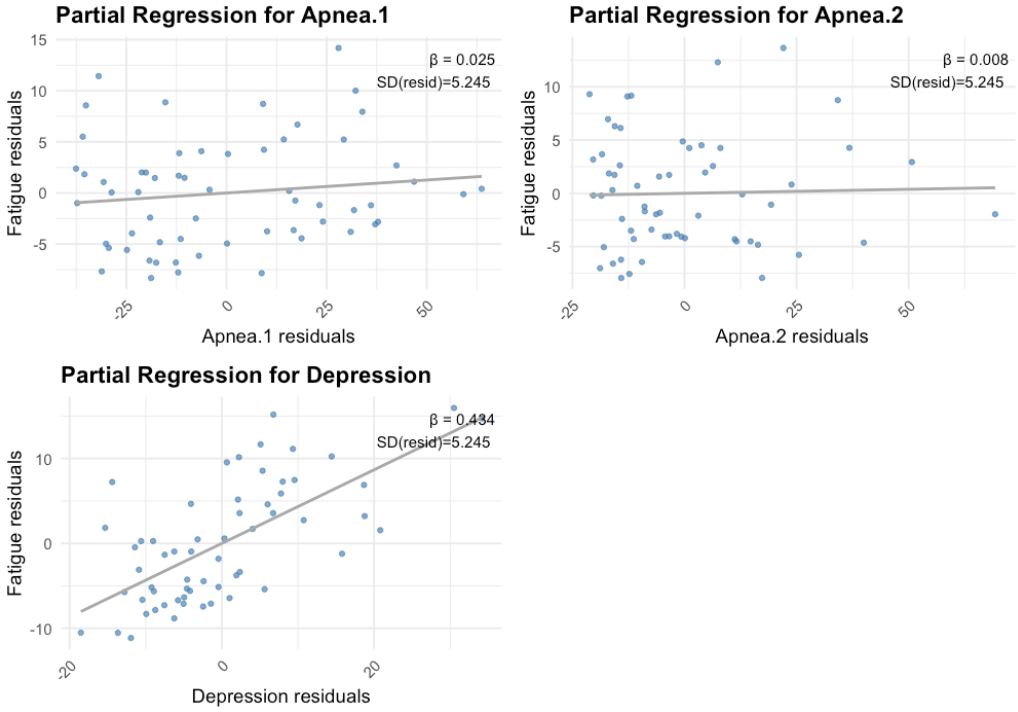


Finally, I saved the RMSE and relevant statistics,

```
get_rmse(result.lm)
      $rmse_cor
      [1] 0

      $rmse_reg
      [1] 0.002236068

      $rmse_se
      [1] 0.007874008
get_stats(result.lm)
      $model

      Call:
      stats::lm(formula = eq, data = data_df)

      Coefficients:
      (Intercept)      Apnea.1      Apnea.2   Depression
          4.02027      0.02543      0.00757      0.43362



      $reg
      (Intercept)     Apnea.1     Apnea.2  Depression
      4.020274448 0.025432823 0.007570258 0.433622314

      $se
      (Intercept)     Apnea.1     Apnea.2  Depression
       1.72534792  0.02613568  0.03671927  0.06469461

      $cor
      [1] -0.13491220  0.02074421  0.28631735  0.10622519  0.20197415
      0.68481225

      $mean
         Apnea.1    Apnea.2 Depression    Fatigue
        48.77024   17.26942   12.55000   10.83333

      $sd
         Apnea.1    Apnea.2 Depression    Fatigue
       27.117733  20.140629  11.329316   7.258348
```

extracted the simulated data set and inspected its partial regression plots.

```
data.lm <- result.lm$data
head(data.lm)
            Apnea.1     Apnea.2 Depression Fatigue
      [1,] 39.09125  3.08544358          5       3
      [2,] 20.79024  0.05366945         10       2
      [3,] 43.52339  0.00000000          5       5
      [4,] 42.25313 20.28673143         15      16
      [5,] 77.31819 12.12986672          1      11
      [6,] 65.55855  9.88781111          3       3
partial.plots <- plot_partial_regression(lm(reg_equation, data.lm))
gridExtra::grid.arrange(grobs = partial.plots, ncol = 2)
```



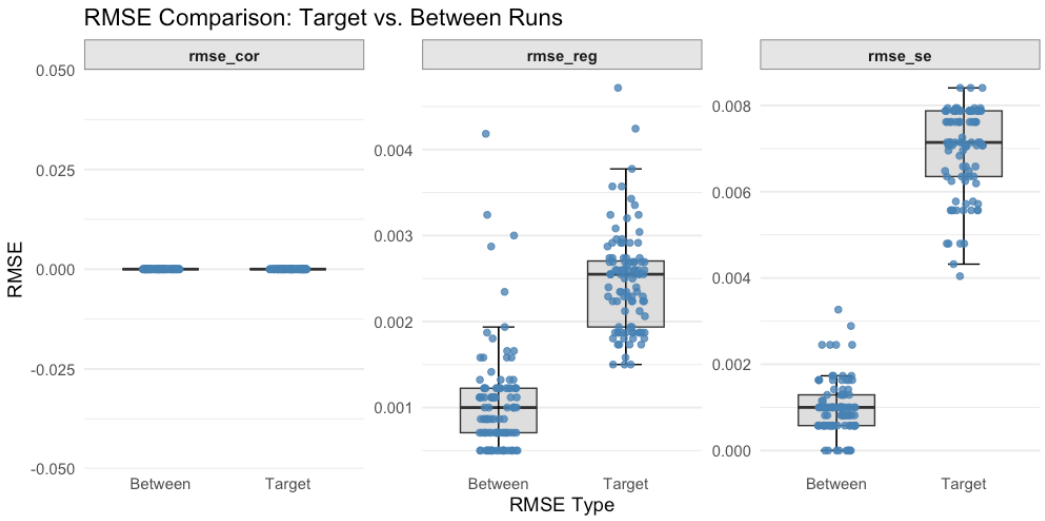
I executed the LM module in multiple parallel runs to quantify both the convergence variability of RMSE within each run (compared to the target values) and the variability of RMSE across runs (compared to the average simulated values).

```
result.parallel.lm <- parallel_lm(
  parallel_start = 100,
  return_best_solution = FALSE,
  sim_data = sim_data,
  reg_equation = reg_equation,
  target_cor = target_cor,
  target_reg = target_reg,
  target_se = target_se,
  weight = weight.lm,
  tol = 1e-8,
```

```
  max_iter = 1e5,
  max_starts = 1)
```

I then plotted these RMSE distributions side-by-side to compare within versus between run variation.

```
plot_rmse(result.parallel.lm)
```



LME Module

The application of DISCOURSE's LME module is necessarily constrained by the complexity of mixed-effects models in practice. Unlike simpler designs (e.g., ANOVA or single-level regression), real-world LME models frequently incorporate crossed and nested random effects, random slopes, and covariance structures that extend beyond a single random intercept. To maintain computational feasibility, DISCOURSE currently supports designs with a single random intercept; all random-slope or more elaborate nesting structures lie outside its current scope. Moreover, the method presumes that the practitioner has access to marginal means and standard deviations at each measurement occasion (columns in wide format). These assumptions are required because, without them, the combinatorial optimization algorithm cannot isolate the marginal- and covariance-level constraints necessary to recover a plausible raw data permutation.

To assess the practical relevance of these restrictions, I checked publications employing and citing the R package *lme4* (Bates et al., 2015) for LME analyses in psychology and the social sciences. Only very few studies used a random intercept model only, and even less articles reported the per-time-point means and standard deviations or correlations of study variables. Consequently, while the LME module can, in principle, reconstruct synthetic data sets for single random-intercept models, its direct application to most published studies remains limited.