EVERYTHING ON THE TABLE: TABULAR, GRAPHIC, AND INTERACTIVE APPROACHES FOR INTERPRETING AND PRESENTING MONTE CARLO SIMULATION DATA

MATTHEW J. SIGAL

A DISSERTATION SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN DEPARTMENT OF PSYCHOLOGY
YORK UNIVERSITY
TORONTO, ONTARIO

October, 2017

Abstract

Monte Carlo simulation studies (MCSS) form a cornerstone for quantitative methods research. They are frequently used to evaluate and compare the properties of statistical methods and inform both future research and current best practices. However, the presentation of results from MCSS often leaves much to be desired, with findings typically conveyed via a series of elaborate tables from which readers are expected to derive meaning. The goal of this dissertation is to explore, summarize, and describe a framework for the presentation of MCSS, and show how modern computing and visualization techniques improve their interpretability. Chapter One describes this problem by introducing the logic of MCSS, how they are conducted, what findings typically look like, and current practices for their presentation. Chapter Two demonstrates methods for improving the display of static tabular data, specifically via formatting, effects ordering, and rotation. Chapter Three delves into semi-graphic and graphical approaches for aiding the presentation of tabular data via shaded tables, and extensions to the tableplot and the hypothesis-error plot frameworks. Chapter Four describes the use of interactive computing applets to aid the exploration of complex tabular data, and why this is an ideal approach. Throughout this work, emphasis is placed on how such

techniques improve our understanding of a particular dataset or model. Claims are supported with applied demonstrations. Implementation of the ideas from each chapter have been coded within the `R` language for statistical computing and are available for adoption by other researchers in a dedicated package (`SimDisplay`). It is hoped that these ideas might enhance our understanding of how to best present MCSS findings and be drawn upon in both applied and academic environments.

Acknowledgements

This dissertation is the product of an incredibly supportive network of people. My primary supervisor, Dr. Michael Friendly, was invaluable. He provided me with many opportunities and encouragement throughout my studies that I will always be grateful for. I would also like to acknowledge my committee members, Dr. David Flora and Dr. Robert Cribbie, for their feedback. Their suggestions were instrumental in the completion of this work. Further, I owe a substantial amount of gratitude to Dr. Phil Chalmers. From helping me formulate some of the fundamental aspects of this research to proofreading to always being on call for advice, I cannot express enough thanks for your friendship and camaraderie. I would also like to thank some of my family and closest friends. Maria Sigal, Cindy, Dan, James, Tiffany, Liam, and Derek Belliveau, Nathan Sassi, Elizabeth Malette, Adriane Chalastra, Carrie Smith, Shomar Griffiths, and Timothy Chan, you all were present when I needed you most. Finally, I would like to thank my partner, Arlie Belliveau, and our child, Huxley Sigeau. You two are my home. None of this would have been achievable without your patience, love, and support. From idea generation to proofreading to listening to me rehearse my presentation, you were always there and I couldn't have done it without you.

Contents

List of Tables

List of Figures

1 Setting the Table

The primary goal of statistics is to describe and summarize complex empirical data into simpler concepts. This process of extracting information from messy and unwieldy data necessarily involves an aspect of presentation, as the conclusions or inferences drawn need to be expressed in a clear and meaningful way. If we are unsuccessful on this front, all of the effort that went into our analyses is wasted. As such, methodology for the presentation of statistical findings is an important topic, as thoughtful summaries encourage proper understanding of our statistical models and guard against erroneous interpretations (Cleveland, 1994; Cleveland & McGill, 1984; Few, 2012; Tufte, 2001, 2006; Wainer, 1984; Ware, 2004). Unfortunately, although personal computing resources have become more ubiquitous, especially for the evaluation of statistical methods and techniques, methods and theory for communicating results have often lagged behind.

It is common for the results from various statistical methods to be presented in tabular form or even across an array of tables. For example, intercept and slope estimates from a linear regression are often relegated to a table that provides the reader with essential information about the fit and estimates from that particular model. A more elaborate situation, and one that is the focus of the

current research, pertains to the results from Monte Carlo simulation studies (Jones, Maillardet, & Robinson, 2009; Ross, 2013), which are a computational technique for generating data and evaluating the performance of various statistics. Such research is a foundation for quantitative scholarship and provides a structural backbone for applied statistics at large. However, the results from such work are typically presented as a series of cumbersome and extensive tables from which readers are expected to derive meaning or risk simply trusting the author's summarizations.

The goal of this dissertation is to introduce Monte Carlo simulation research methods in a general fashion and then describe a framework for the presentation of their results. As the goal of simulation research is the discovery of interesting phenomena and its subsequent presentation, emphasis will be placed upon how modern computing and visualization techniques can improve on their interpretability and aid in knowledge translation. Information uptake occurs in three stages: early attention, working memory, and long-term memory (Evergreen, 2014). Enhancing design elements can help draw attention, aid viewers in understanding the information presented, and increase their recall of the findings later on. Throughout this endeavor, sample results from simulation studies are used to demonstrate the various techniques, how they improve our understanding of the underlying phenomena, and their implementation within the `R` project for statistical computing (R Core Team, 2016) using a new package, `SimDisplay`.

This chapter will set the stage by recapitulating the problems faced by researchers who wish to present simulation results: first by introducing the logic of simulation studies, how they are conducted, and what the results typically look like, and then by describing current practices for

their presentation. It will also outline the topics that will be covered in the remainder of the dissertation pertaining to tabular, graphic, and interactive methods that can be used to improve the interpretability of complex tables.

1.1 What are Monte Carlo Simulation Studies?

Monte Carlo simulation studies (MCSS) are computer-driven experimental investigations in which certain parameters, such as population means and standard deviations that are known or set a priori, are used to generate random (but plausible) sample data (Mooney, 1997). These generated data are then used to evaluate the sampling behavior of one or more statistics of interest under various circumstances. A typical application might be to determine how well a particular statistical method performs under various conditions or degree of assumption violations.

This process of generating and analyzing data is repeated over many independent replications, and often across many different conditions that are thought to influence the statistic of interest (e.g., through increasing or decreasing sample size, mean differences, or variability). For latent variable models in particular, MCSS offer a viable approach for evaluating estimators and goodness-of-fit statistics under a variety of conditions, model complexity, and model misspecification (Bollen, 1989). As stated by Paxton et al., "...many topics in [latent variable modeling] would benefit from an empirical analysis through Monte Carlo methods" (2001, p. 288).

MCSS are a fundamental cornerstone for work in fields like quantitative psychology, political science, and economics (Zickar, 2005). In the classroom, these methods can benefit statistical

pedagogy (Diez, Barr, & Cetinkaya-Rundel, 2014; Everton, 1984; Hagtvedt, Jones, & Jones, 2007, 2008; Mills, 2002; Raffle & Brooks, 2005; Sigal & Chalmers, 2016), as conducting simulations with students can provide them with a hands-on investigation of often purely theoretical or abstract concepts. Further, they can be used to address many complex problems and are routinely applied in research published in top tier statistical journals, especially when new methods or estimators are proposed and compared. For instance, this framework has allowed researchers to:

- Evaluate the performance or characteristics of various statistical estimators (e.g., Chalmers, 2015; Chalmers & Flora, 2014; Kenny, Kaniskan, & McCoach, 2015; Nordstokke & Zumbo, 2007; Schönbrodt & Perugini, 2013)
- Investigate the performance of a statistic under various assumption violations (e.g., Arnau, Bendayan, Blanca, & Bono, 2013), particularly in the study of coverage and the robustness of different estimators (e.g., Sass, Schmitt, & Marsh, 2014; Tofighi & MacKinnon, 2016)
- Generate and showcase data from a variety of sampling distributions (e.g., Arnholt, 1999)
- Determine power or Type I error rates for various designs and under various conditions (e.g., Brown & Forsythe, 1974; Chalmers, Counsell, & Flora, 2016; Ramsey & Ramsey, 2009)
- Simulate 'realistic' data which addresses hard to sample phenomena, such as estimating the number of pedestrian casualties or to predict teenage pregnancy rates (e.g., Goldman & McKenzie, 2009; Noland, Klein, & Tulach, 2013; Sayegh, Castrucci, Lewis, & Hobbs-Lopez, 2010)

- Determine the behavior of model fit statistics in complex multivariate systems of equations (e.g., Bollen, Harden, Ray, & Zavisca, 2014; Heene, Hilbert, Freudenthaler, & Bühner, 2012)

This list is not exhaustive, but shows the breadth of topics that the approach can address once mastered. Further, the techniques utilized in this research are intimately related to those required for bootstrapping methods, which utilizes resampling to obtain empirical estimates of sampling distributions, confidence intervals, and *p*-values when a parameter's sampling distribution is non-normal or unknown (Hallgren, 2013).

Additionally, the popularity of MCSS research is on the rise. A cursory search for peer-reviewed articles within scholarly journals using the query `all("Monte Carlo Simulation")` on PsycINFO alone reveals an increasing trend (see Figure 1.1). In particular, MCSS are especially prevalent in the pages of *Multivariate Behavioral Research* and *Structural Equation Modeling*, to the degree that the editors of these journals have published comprehensive guidelines for such studies (Boomsma, 2013; Paxton et al., 2001; Skrondal, 2000), which will be referenced throughout this dissertation as appropriate.

1.1.1 The Mechanics of Monte Carlo Simulations

The organization of MCSS generally mirror that of traditional research studies: a sample of data must first be gathered (or in simulation studies, *generated* by some probability distribution function), *analysed* using one or more statistical methods and data operations, and *summarised* for

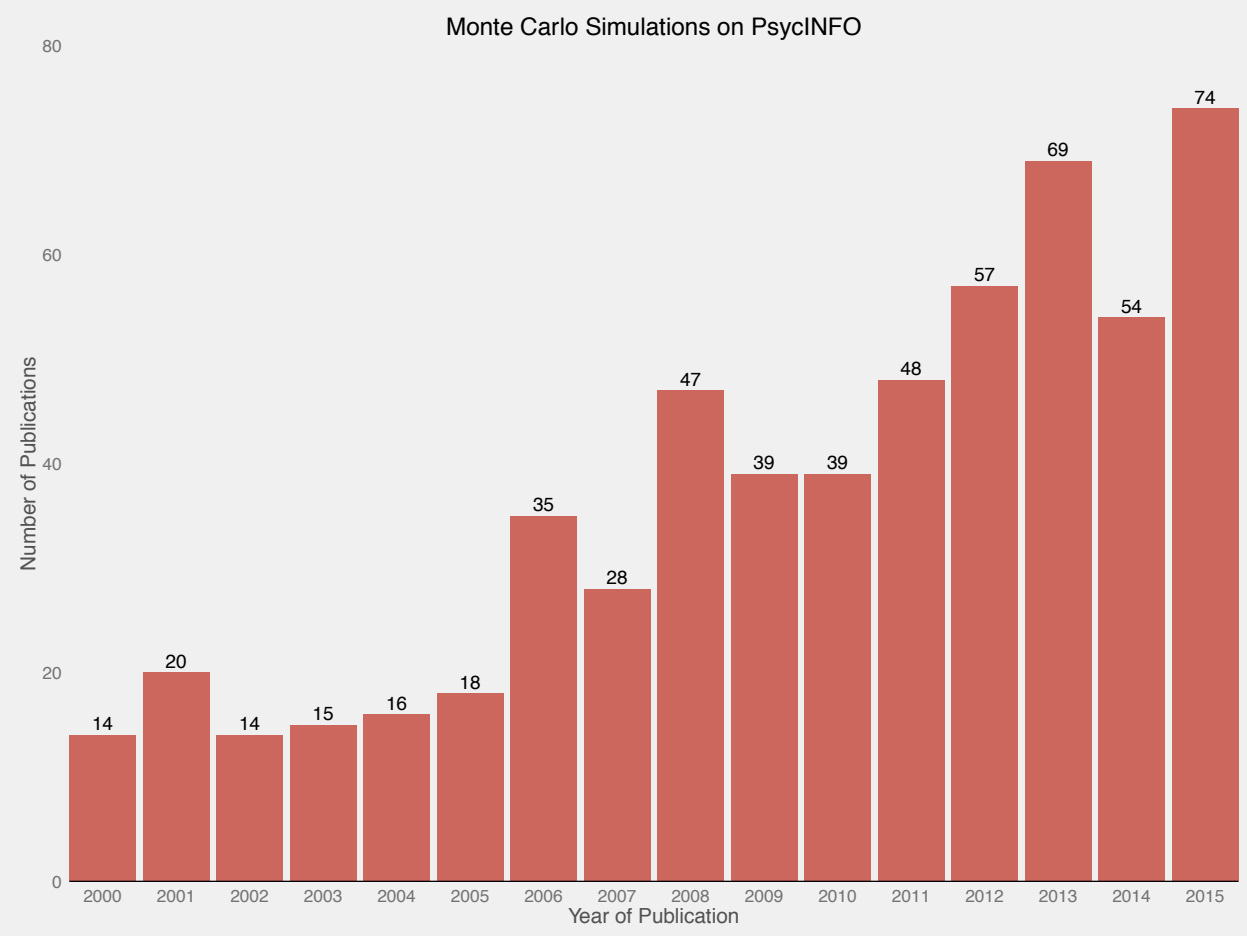


Figure 1.1: Monte Carlo research on PsycINFO by year

interpretation and dissemination. Hence, constructing a simulation study is a matter of organizing a 'generate-analyse-summarise' workflow, such that particular datasets and analysis procedures can be emulated and studied over a large number of random replications.

This schema is extremely flexible primarily because it can use the power of the central limit

theorem (CLT)[1] and the law of large numbers[2] directly.[3] More specifically, given some population parameter of interest ψ, let $\hat{\psi} = f(D)$ be the associated sample estimate that is a function of data input D. If we generalize the usual understanding of the CLT, we could obtain an infinite number of randomly sampled datasets D_i, then the true parameter value could be recovered as the average of all $f(D_i)$ values. However, because drawing an infinite number of random samples is infeasible, a simulation is used instead to generate a *finite* but large number of datasets to obtain a sample estimate of the population parameter. Therefore, instead of obtaining ψ we can use

$$\tilde{\psi} = \frac{f(D_1) + f(D_2) + \cdots + f(D_R)}{R} \tag{1.1}$$

as a suitable approximation, where R is some large number of replications (typically 1,000 or more). The fascinating aspect of Equation 1.1 is that it will hold true for virtually *any* statistic and data generating mechanism that can be imagined (Mooney, 1997). Furthermore, the sampling error of ψ can be approximated by finding the standard deviation of all the generated $f(D_i)$ datasets

$$\text{SE}(\tilde{\psi}) = \sqrt{\frac{[f(D_1) - \tilde{\psi}]^2 + \cdots + [f(D_R) - \tilde{\psi}]^2}{R}}, \tag{1.2}$$

[1]As a refresher, the CLT pertains to the behavior of a statistic under the most idyllic of conditions, and is typically taught as follows: "For any population with mean μ and standard deviation σ, the distribution of sample means for sample size n will have a mean of μ and a standard deviation of $\sigma/\sqrt{n}$ and will approach a normal distribution as n approaches infinity" (Gravetter & Wallnau, 2012, p. 205). This is important because it stresses that samples can exhibit sampling error (the discrepancy between the sample statistic and its parameter), but also points toward the solution – a large number of replications.

[2]This is the principle of probability that states the observed ratio of outcomes will converge on the theoretical, or expected, ratio of outcomes, given enough trials.

[3]While the application of the CLT might not apply uniformly to all MCSS, asymptotic normality for MCSS parameters should hold regardless of the sampling distribution under mild regularity conditions. A key contribution of MCSS is to explore how large sample size needs to be to achieve reasonable convergence to normality for various models. As such, the idea of utilizing the law of large numbers and resampling can be applied fairly generally.

which is interpreted as *the standard deviation of a statistic under a large number of random samples* — a finite definition of the standard error. Since the standard deviation of the collection of R mean estimates is, by definition, an estimate of the standard error of the mean, any questions about its behavior under particular conditions can be addressed through the manipulation of the properties of the generated data.

These concepts presented in isolation may sound daunting but can be illuminated by investigating the properties of the standard error of the mean. If one was interested in how this particular standard error is affected by the generating distribution, sample size, skewness, or any other issue, the MCSS steps required to investigate its behavior are as follows:

1. Generate a dataset with N values according to some probability density function (e.g., normal, log-normal, binomial, or χ^2).

2. Analyse the generated data by finding the mean of the sampled data, and store this value for later use.

3. Repeat steps 1 and 2 R times.

4. Summarise the set of stored values by obtaining the standard deviation statistic from each replication.

The procedure used during this last step depends on the research question at hand and provides the metric for the response variables of interest. Four common methods of summarization are to

calculate bias, root-mean square error (RMSE), relative efficiency (RE), empirical coverage and non-coverage rates (ECR), and empirical detection rates (EDR) .

Bias refers to the difference between the expected value of the parameter and its actual population value, represented as $E(\tilde{\psi} - \psi)$:

$$\text{bias} = \frac{1}{R}\sum_{r=1}^{R}(\hat{\psi}_r - \psi), \tag{1.3}$$

with R still referring to the number of replications, ψ the true population parameter, and $\hat{\psi}_r$ the sample estimate of the parameter for the rth dataset analysed. When applied within a MCSS, a good population estimate should be unbiased. This would indicate that they are not returning values for the parameter that are systematically too high or too low; hence, $E(\tilde{\psi} - \psi)$ and bias values are more preferable the closer they are to zero.

Similarly, good estimators should demonstrate minimal sampling error when used to recover their population values. This is intimately related to the concept of bias, and is customarily reported using the *root-mean square error* (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{R}\sum_{r=1}^{R}(\hat{\psi}_r - \psi)^2}. \tag{1.4}$$

Smaller RMSE values are desirable because they indicate higher precision in estimating the population parameters, and squaring the RMSE yields the mean-squared error (MSE).[4] Because MSE

[4]In relation to the sampling error, it can be demonstrated that $\text{RMSE}^2 = \text{bias}^2 + \text{SE}^2$; therefore, RMSE values will be larger than the associated SE when the estimator is biased.

values are interpretable as variance terms, a method that is often useful for comparing the sampling efficiency between different estimators (distinguished using the subscript i) is the *relative efficiency* (RE) statistic:

$$\text{RE}_i = \text{MSE}_i / \text{MSE}_1. \tag{1.5}$$

In this calculation, one estimator is treated as the reference statistic (in this case, the first *MSE* value, MSE_1) and all other *MSE*'s are evaluated in reference to it. Naturally, when $i = 1$ then $RE \equiv 1$; values greater than 1 indicate less efficiency (i.e., more variability) than the reference statistic, while values less that 1 indicate greater efficiency and therefore greater precision in recovering ψ.

Coverage refers to whether a proposed confidence interval contains the population parameter at some advertised rate equal to $(1 - \alpha)$. This property is generally presented as a proportion or percentage. For instance, if a statistic is calculated with $\alpha = .05$, it should have 95% coverage within each cell of the simulation. Performance is then judged based on if the ECR is substantially higher or lower than .95.

Finally, the empirical detection rate (EDR) is most prevalent in studies that pertain to power and Type I error rates. The EDR is the proportion (sometimes expressed as the percentage) of iterations within a cell of the design matrix that had a p-value less than a selected α value, conventionally set at $\alpha = .05$. For Type I error rate conditions, a good estimator should have an EDR around α; that is, it should not be excessively liberal (have too many rejections) or conservative (too few). In each case, MCSS allows researchers to assess the finite sampling performance of estimators by creat-

ing controlled conditions from which sampling distributions of parameter estimates are produced (Paxton et al., 2001). Knowledge of that sampling distribution, which is typically theoretical and unobserved but artificially produced during simulation, is the key to evaluating the behavior of a statistic.

To apply these concepts to the theoretical simulation about the properties of the standard error of the mean, we would typically be interested in calculating bias, RMSE, or RE rates for different design conditions. For instance, queries regarding sample size can be investigated by changing the size of N; the influence of restriction of range can be investigated by using truncated data in contrast to the original scale; the influence of non-continuous data can be assessed by generating data either via a binomial distribution or by rounding the values generated from other distributions; and so on. The flexibility of MCSS has substantial usefulness when applied to these and other such questions, which would be difficult if not impossible to answer otherwise.

Even though the framework for MCSS is so adaptable, the process of conducting such research is fairly standardized and easily accommodated within the R software environment.[5] Generally speaking, authors of such studies follow eight steps (based upon Paxton et al., 2001):

1. Develop a theoretically derived research question;

2. Choose a valid statistical model and appropriate values for each of the population parameters for assessing the particular question;

[5]For more details and comprehensive examples of how to conduct MCSS in R, refer to Sigal and Chalmers (2016).

3. Design specific experimental conditions that are pertinent to the research question;

4. Choose an appropriate software package;

5. Determine a file storage solution;

6. Execute the simulation repeatedly;

7. Troubleshoot and verify the results; and,

8. Summarize the findings

The first three steps pertain to the particular goals of an individual simulation study, where previous research and appropriate intuition dictate suitable choices. Step three is particularly important, because this choice should be steeped in either genuine curiosity or, more typically, previous research.[6] The main issue to consider is that if too many factors are manipulated, then a simulation can take extreme amounts of time to run, often with diminishing returns in regard to actual illumination. Steps five through seven are handled through the selection of an appropriate software package and careful programming on behalf of the researcher. While many of the ideas presented in this dissertation could be executed from a variety of high-level programming languages or dedicated statistical software applications, the R software environment (R Core Team, 2016) is recommended.

[6]Typical variables used as design factors include sample size, generating distribution of the observed variables, estimation method (e.g., maximum likelihood, generalized least squares, two-stage least squares, asymptotic distribution free, etc.), and extent of model misspecification (Paxton et al., 2001).

1.1.2 Why R?

R (http://cran.r-project.org/) is an open-source and syntax-driven programming language, which means that any code utilized by the application can be viewed, edited, and improved upon by the user-base. The base language allows researchers to incorporate techniques common in other programming languages such as loops, random number generation, conditional (if-then) logic, branching, and the reading and writing of data. However, R was primarily designed for statistical operations, and almost every extant statistical method, especially those on the cutting edge, have implementations in the language. These are generally distributed within a dedicated *package* (a set of functions that are conceptually linked and bundled together; Wickham, 2015). For instance, additional packages provide frameworks for everything from running spatial analyses (Bivand, Pebesma, & Gomez-Rubio, 2008), to text mining (Silge & Robinson, 2017), to the construction of interactive visualizations (D. Cook & Swayne, 2007), and even to the creation of academic presentations and books (Xie, 2016a). One particularly valuable feature of R is the ease with which it allows researchers to produce publication quality graphics (Wickham, 2009; Yau, 2011).

While R fundamentals are outside the scope of the current work, I believe that this language provides a rich set of resources, far beyond those offered by other statistical software packages, and is well worth learning.[7] Further, one particular package is especially relevant for the current

[7]For those not familiar with the language, I recommend Matloff (2011), Wickham (2014a), and Cotton (2013) for introductions to programming in the language, and Fox and Weisberg (2010) and Teetor (2011) for learning about how to conduct basic statistical analyses. It is also recommended that individuals new to R also install RStudio (https://rstudio.com), a graphical user interface that makes working with the language substantially easier.

work: `SimDesign` (Chalmers, 2016; Sigal & Chalmers, 2016). This package provides an elegant template for simulation research, and includes many safe-guards and convenience features. I highly recommend that researchers planning to conduct simulation-based research use this package to organize and execute their work. Most of the content of this dissertation references functions found in a complementary package, `SimDisplay`, which can be installed by entering the following code into an `R` terminal[8]:

```
devtools::install_github("mattsigal/SimDisplay") # Omit if package is already installed
library("SimDisplay") # Load the SimDisplay package
```

Once this package is loaded, a variety of functions are made available to the researcher. This package facilitates the summarization and presentation of results, or step nine of conducting MCSS, which is the primary concern of this dissertation.

1.2 The presentation of Monte Carlo results

Most MCSS designs necessarily produce a voluminous amount of output and results. Sorting through this assemblage is the primary responsibility of the researcher; however, a substantial amount of these data are often reported in the accompanying published work. Entire books written about designing and understanding simulation research exist with no mention of how one might present results from such work (e.g., Jones et al., 2009; Ross, 2013) and, while published guidelines for MCSS research do provide some insight on how to present this form of data, they are fairly

[8]The `devtools` library also needs to be installed. If it is not, run: `install.packages("devtools")`.

lacking in their breadth. For instance, Paxton et al. (2001) state that MCSS results to be published in *Structural Equation Modeling* can be presented descriptively, graphically, and inferentially but provide little detail or guidelines for accomplishing this task.

The shortcomings and lack of theory behind the current methods of presentation will be addressed in the dissertation, as "...reading results from Monte Carlo studies in whatever form should be a revelatory task, not a baffling puzzlement" (Boomsma, 2013, p. 534). Unfortunately, prototypical MCSS findings are generally relegated to very long tables from which it is nearly impossible to decipher overarching patterns. One example of this type of presentation is found in Ramsey and Ramsey (2009), who undertook a fairly straightforward simulation study that compared the performance of 10 pairwise multiple comparison procedures in ANOVA designs under various degrees of heteroskedasticity, numbers of groups, and sample size per group. The primary output from this study was the Type I error rates from true null models. While overall conclusions about the usefulness of the various multiple comparison procedures can be garnered from this paper, the central results were presented in a single table that spanned three pages and still neglected to include many of the sample size comparisons[9] or the number of groups factor.[10]

In light of the above example of how suboptimal MCSS presentations can occur in practice, more thought should be given to their presentation. Recommendations are made based on a thorough literature review of prominent simulation studies and their modes of presentation, the aspects

[9]None of the (many) sample size conditions that pertain to unequal group size ratios were included.

[10]The entire table only pertains to conditions where the number of groups was equal to 4.

of tables that can be manipulated based on information design theory principles, and how such changes can affect our comprehension of results. Further, suggestions are made for both the exploratory analyses typically run by the researcher when first organizing simulation results, as well as the tables and graphics designed for the purpose of presentation and publication. While both activities intimately deal with MCSS results, their goals differ: exploratory analyses provide the researcher with methods for quickly summarizing and highlighting possibly interesting findings, while the goal afterward is to find a way to best present such findings to an external audience.

In terms of organizing this discussion, refer to Figure 1.2 for a conceptual diagram for generating, analyzing, and presenting MCSS results, with elements shaded by chapter. The remainder of this chapter briefly outlines the goals for each chapter of the dissertation, and provides an overview of how the presentation of MCSS results can be improved via tabular, graphical, and interactive displays. While graphical and interactive techniques can be used for exploratory purposes, most of the focus for conducting initial summaries of the data is discussed in Chapter 2. R code is presented throughout the following work to supplement ideas and provide tangible examples.

1.2.1 Tabular Approaches

The focus of Chapter 2 is to determine methods for the improvement of the presentation mechanisms for static tabular data, as such displays can convey substantial amounts of information if they are thoughtfully constructed. This topic deals with the creation of tables for exploratory work as well as those rendered suitable for presentation. In either case, tables should be designed in a

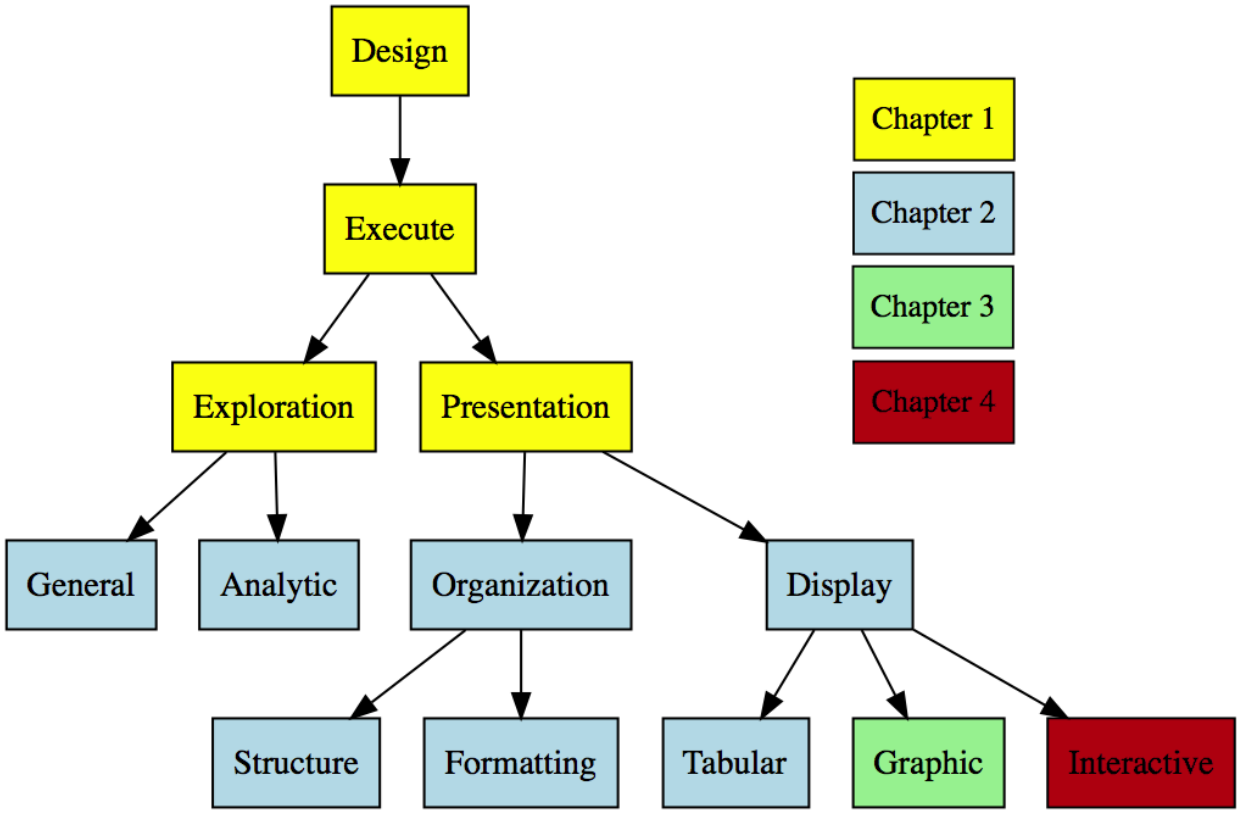


Figure 1.2: The preparation and analysis of a MCSS study: A conceptual diagram for the present work

manner that best presents the relevant results with as little effort or burden on behalf of the reader as possible.

Further, it is customary for a statistician to spend a substantial amount of time making their data amenable to analyses. The goal of data cleaning is to produce what is known as tidy data (Wickham, 2014b): a standard for structuring data, with each variable as a column, each observation as a row, and each type of observational unit a table. It is of interest to this dissertation to in-

vestigate how the concept of tidy data applies to MCSS and how we can utilize that paradigm to sort and store results. This idea has since been extrapolated to tidy *theory* (Robinson, 2014), in which results from statistical analyses are encoded in such a way to make them easily amenable to visualization and summarization. Specifically, this organization can be achieved by tidying results into three categories: statistical components (e.g., slope and intercepts from a linear regression), observations (e.g., fitted values and standard errors for particular data points in a linear regression), and information at the model level (e.g., fit statistics and R^2 values). A goal of the present research is to show how results from MCSS can be understood in this fashion and how this structure enables their scrutiny.

Other methods for improving such tables through formatting and their reorganization are discussed, specifically via the use of effects ordering and rotation. Similarly, aesthetic emphasis, such as bold, italics, and color, can be utilized to help signify important results. I show that through the thoughtful reorganization of the rows and columns of a complex array, patterns that were once hidden may be revealed, and different rotation criteria afford views of the structures in data that would otherwise be difficult to glimpse.

1.2.2 Graphical Approaches

Chapter 3 delves into graphical approaches for aiding the presentation of MCSS data. Such methods involve transforming numerical data through some form of symbolic representation or abstraction (e.g., shapes and color). These summaries can make it easier to see underlying patterns in

simulation results at a glance compared to traditional, unaltered tables.

The most direct approach is a stylized presentation of the raw table. For example, a shaded or semi-graphic table can be used to display the raw values of interest (e.g., empirical detection rates) with numeric values overlaid on a shaded background.[11] The shading allows for easy visual comparison across cells based on a continuous range of values, and the degree of intensity can be manipulated to highlight meaningful differences. It should also be noted that, for designs with three or more dimensions, this approach can utilize marginal tables or facets to examine various relationships.

Considering how large these tables can be, it is useful to think of alternative graphical approaches that make it easier to see what is happening. One alternative is to extend the `tableplot` framework (Kwan, Lu, & Friendly, 2009), which presents a semi-graphic display that uses both numbers and symbols (with sizes, colors, and shapes providing additional information) to summarize complex data. This approach was originally intended to encapsulate the results from exploratory and confirmatory factor analyses, but can be applied to the results from other statistical methods and data types. I show that this structure can be applied to the tables generated from MCSS research.

Additionally, I present how simulation conditions can be treated as ANOVA designs, with the design conditions set as fixed factors. If the researcher applies this methodology, effect plots (Fox, 2003; Fox & Hong, 2009) can be produced to summarize the influence, if any, of the design factors.

[11] This is related to a heatmap display, in which categorical data are summarized and shaded by frequency rather than magnitude.

These plots provide efficient numerical summaries of simulation results and can highlight potential areas of interest.

Finally, I investigate how one might use graphical dimension reduction techniques, specifically hypothesis-error (HE) plots (Friendly, 2007), to organize and present the results from complex tables. Also, I discuss whether these plots are well suited for displaying results from simulation studies. In such a display, each design cell is presented as a point on the graphic, and overall patterns can emerge based on how the points converge.

Graphical approaches can effectively summarize a great deal of information and are underutilized in the presentation of simulation research. Because most simulation studies are conducted with the aim of publication, this chapter is of utmost importance as it details methods available to researchers which are appropriate for both digital and printed venues. Methods and convenience functions for generating such graphics are provided in `SimDisplay` to make their construction as straightforward as possible.

1.2.3 Incorporating Interactivity

With advances in computing technology, the results from complex studies should not be relegated by the restrictions of the printed page. The work outlined in the previous chapters shows static slices of data, and often requires displaying only conditional findings. Such methods can mask important findings, and so it is preferable to allow readers to manipulate results interactively.

In Chapter 4, recent innovations in interactivity are demonstrated which allow for the dynamic

exploration of complex tables and figures via digital presentations. The idea is that tables and graphics can be embedded in published work (either within the online article itself, or as a supplemental website) that contain more information than such a display would typically allow (ideally, all unfiltered data generated by the simulation or the research should be available). Readers could then apply various methods of interaction to view and evaluate both the primary findings detailed by the author, as well as explore the data further for other insights. Good science involves communication and transparency, and these methods provide an ideal template for such research.

In addition, such a framework holds many benefits for exploring tabular results. Ideas for interaction range from allowing changes of the display elements (e.g., permitting the sorting and filtering of results, the addition or configuration of shading, or the ability to search for particular entries), to user interface considerations (e.g., incorporating pop-up text which adds further insight from the author onto the presented data, allowing cells to be selected and provide more information), and so on.

This chapter not only details various approaches to interactivity, but also provides code which allows researchers to incorporate these ideas into their presentations. Such displays only require a modest amount of computing power, and when coupled with an appropriate software package, can be constructed with relative ease. Again, convenience functions will be provided via `SimDisplay` to assist in this regard.

1.3 Overall motivations

Cognitive load refers to the amount of mental activity – perception, memory, problem solving – required to accomplish a goal (Lidwell, Holden, & Butler, 2003). The intention of this work is to outline and provide methods for minimizing such loads from the presentation of MCSS results. As such, throughout this dissertation, emphasis is placed on how these various techniques can improve the understanding of a particular dataset or model. Such claims are supported with applied demonstrations. A classic simulation study has been chosen as an exemplar for use throughout. Specifically, Brown and Forsythe's work on tests for heterogeneity of variance in ANOVA (1974, see Chapter 2) has been replicated by the author and the various presentation techniques are applied to show how they influence interpretation. A secondary example, pertaining to characteristics of the Sobel statistic in mediation analysis, is an expansion on the example study presented in Hallgren (2013), which will also be introduced in Chapter 2 and referenced thereafter.

Further, it is not only important to document these recommendations, but also to provide researchers with an easy to use method for their implementation. As such, an accompanying `R` package, `SimDisplay`, has been developed alongside this dissertation, and provides functions that render the creation of such tables, figures, and interactive applets as trivial as possible. One primary feature of this package is its native integration with `SimDesign` (Chalmers, 2016), an `R` package designed to aid in the design, structure, and implementation of MCSS research, and utilized for each of the presented examples. To encourage reproducibility, relevant and annotated code accom-

panies each approach. Also, the data from the simulation examples used in this work are available from within `SimDisplay` for researchers to explore.[12]

While the primary motivations and aims for this dissertation pertain specifically to MCSS research, many of the findings presented can be extrapolated to any situation where the primary output for presentation is tabular. These techniques and insights should be useful across a variety of applications.

[12]These can be loaded into the global environment by running: `data(Brown1974)` and `data(Hallgren2013)`, respectively.

2 Table Dressing

The structure of results from Monte Carlo simulation studies (MCSS) necessarily takes the form of multi-dimensional arrays. Researchers are often interested in a large combination of design conditions, which might include overall sample size, number of groups, sample size per group, magnitude of effect size, degree of within-group variability, generating distribution for the data, and method of estimation used. When these factor levels are crossed, the full design matrix emerges.

For example, a simple simulation study might only deal with sample size per group (with four levels: 20, 50, 100, and 250), and generating distribution (with three levels: Gaussian, χ^2 with 4 degrees of freedom, and χ^2 with 10 degrees of freedom). Crossing these two factors produces a design matrix with twelve cells: one for each unique combination of sample size and distribution (e.g., the first cell might pertain to a sample size of 20, with data sampled from a normal distribution). With the inclusion of additional factors, the size of this design matrix grows exponentially.[13]

Further, multiple outcomes might be tracked for each cell (or unique combination of design factors), such as Type I error rates, power rates, and degree of bias or size of the root mean square

[13]This is the primary reason why it is of paramount importance for researchers to fully conceptualize their simulation studies before conducting them. Including superfluous conditions can render a feasible experiment so drawn-out that it might just become an exercise in futility.

error (RMSE). Each of these results might be calculated for multiple methods, as is often the case when we wish to compare the performance of various statistics. As such, even results from fairly simple MCSS can produce a voluminous amount of output, especially since a raw simulation study will run hundreds, if not thousands, of iterations of each cell. If the results are not summarized in an appropriate way during the course of the study, it is easy to see how organizing them can become unwieldy (Paxton et al., 2001). As Boomsma (2013, p. 531) observed, summarizing simulation results is "a demanding task given the invariably huge amount of available output that needs efficient reduction to accomplish a sufficient and attractive exhibition".

A common method of summarization is the empirical detection rate (EDR, see Chapter 1), which is used in studies that pertain to power and Type I error rates. The results from a MCSS might therefore have one or more columns pertaining to the EDR for *each* outcome measure. For example, Figure 2.1 shows truncated output from Ramsey and Ramsey (2009). This experiment pertained to power and Type I errors rates for comparing multiple group means in the presence of heterogeneous variance. The results here pertain to the first nine Type I error rates for sample sizes of $n = 2$ to $n = 10$ for the four-group case, with equal sample sizes per group, and with c, a variance multiplier, set to 1 (equal variance across groups). Columns in this table designate *ten* different multiple comparison procedures that could be used for detecting mean differences.

Figuring out how to present the results from such studies in a manner that highlights important findings is very important. It is common for researchers to present MCSS results in one or more tables. However, due to the size of many MCSS designs, it is customary for authors to simply omit

Table 1. Type I errors for 10 MCPs at $\alpha = .05$, $k = 4$, equal n_j, variance multiplier c and a true full null hypothesis

n	T3	C	GH	GF*	PF*	GH9	GH8	GH7	GH6	GH5
(a) $c = 1$										
2	.0791	.0117	.0445	.0048	.0039	.0368	.0313	.0260	.0197	.0158
3	.0397	.0135	.0512	.0137	.0171	.0439	.0385	.0331	.0272	.0216
4	.0391	.0157	.0514	.0186	.0256	.0450	.0388	.0335	.0283	.0234
5	.0383	.0173	.0499	.0216	.0311	.0446	.0379	.0328	.0265	.0218
6	.0426	.0231	.0536	.0286	.0371	.0472	.0426	.0375	.0309	.0253
7	.0388	.0239	.0497	.0259	.0351	.0434	.0386	.0338	.0295	.0252
8	.0446	.0301	.0534	.0305	.0373	.0479	.0443	.0391	.0342	.0292
9	.0410	.0263	.0511	.0301	.0390	.0460	.0407	.0360	.0290	.0238
10	.0471	.0343	.0562	.0363	.0441	.0518	.0468	.0410	.0353	.0291

Figure 2.1: Truncated results presented in Ramsey and Ramsey (2009).

conditions from published reports.[14] Even so, the published findings often take the form of some *very* long tables – for instance, the full published table presented in Figure 2.1 spans three pages!

That being said, if thoughtfully constructed, tabular displays can be one of the best methods for presenting MCSS results and are able to convey substantial amounts of information. The focus of this chapter is to discuss and demonstrate methods that can be used to improve the display of static tabular data. Organizing simulation output follows a similar structure to that of traditional statistical analyses: Tables must be constructed in a manner that best presents relevant results, while minimizing the amount of effort required on behalf of the reader to detect them. This chapter will touch on some issues that cause particular trouble when working with tables, as well as introduce a motivating example, discuss some theory to help organize simulation results, and explore methods for improving tabular displays, along with details about their implementation in R.

[14]For instance, Ramsey and Ramsey (2009) also investigated the impact of having eight group means instead of four and tested additional group sample size conditions that were not included in the published report.

2.1 Issues with Tables

> The graphic method has considerable superiority for the exposition of statistical facts over the tabular. A heavy bank of [numbers] is grievously wearisome to the eye, and the popular mind is as incapable of drawing any useful lessons from it as of extracting sunbeams from cucumbers (Farquhar & Farquhar, 1891, p. 55).

As expressed in the above quote, a page overrun with numbers can be daunting. Due to the nature of MCSS results, tables often span many rows and columns and are printed in such a way that meaningful findings are not visible at a glance. For instance, the results from Hittner, May, and Silver (2003), a simulation study pertaining to methods for detecting dependent correlations, are conveyed via a series of three tables which account for eight pages (or almost half) of the manuscript. No emphasis is used throughout the presentation, and trends are difficult to discern. A more recent study expressed concern over the daunting amount of output: "With 840 crossed conditions, there would be *more than 100 tables and figures* to report if we include the RMSEs, RDs [relative differences], and test statistics at each *a* [tuning parameter] in the article. Instead, we discuss the results as... *a* changes, and only include those corresponding to the smallest RMSE" (Yuan & Chan, 2015, p. 168, emphasis added). Even with this stipulation, results were presented across eight tables.

The bottom line is that basic tabular displays can be nearly unreadable, except for perhaps looking up a particular combination of factors (Friendly & Kwan, 2003, 2011). Further difficulties arise when one wishes to make meaningful comparisons of entries within a table, or across multiple tables. This process requires one to look up a particular combination, memorize the finding, read

another one, memorize it, and then compare it against the first value (Cairo, 2013; Few, 2012). This is a cumbersome procedure and, without any distinguishing features, long strings of numbers will visually swim together and make patterns difficult to discern, especially at a glance. Finally, for MCSS research in particular, due to the multitude of factors often under study, many comparisons are almost necessarily hidden from view in the published results. Thus, many readers consult the verbal description of simulation results rather than attempt to glean findings directly from the included tables.

The goal of the remainder of the chapter is to express methods and guidelines that aid in the presentation of tabular results. This presentation involves strategies for properly structuring the original data, presenting the table in a way to best promote recall, making appropriate typographic choices, and highlighting important contrasts. As shown in Figure 1.2, a substantial proportion of work pertaining to the organization and presentation of MCSS results deals with table management. Specifically, this chapter highlights differences in tables generated for the researcher (exploratory methods) from those prepared for presentation, and shows how to improve presentation output via structural and formatting improvements. A final goal for this section is to highlight functions found in `SimDisplay` that will aid researchers in this endeavor.

2.2 Motivating Example

As a motivating example, we can replicate the results from Brown and Forsythe (1974). This classic simulation study pertains to tests of heterogeneity of variance in analysis of variance (ANOVA)

designs, specifically in the two group case. The primary issue is that the then commonly used methods for detecting heterogeneity of variance, the F-ratio[15] and Bartlett's test, are very sensitive to the assumption that the underlying populations are from a Gaussian distribution. If these underlying distributions are non-normal, results can be severely compromised. The paper investigated six methods for testing for the equality of variances, specifically the traditional F-ratio, a jackknife procedure, Layard's χ^2 test, an unmodified Levene's statistic (W_0), Levene's statistic with 10% symmetric trimming (W_{10}), and Levene's test based upon the median rather than the mean as the measure of central tendency (W_{50}). Refer to Appendix A for details on how each of these procedures are calculated. This study only had three design variables:

- Generating distribution for the data:
  - Gaussian;
  - χ^2 with four *df* (positively skewed); and
  - Student's t with four *df* (symmetric but long-tailed); and,
  - Cauchy (coincides with $t(1)$ with long-tails)
- Sample sizes per group, n_1/n_2[16]: 40/40, 20/40, 10/10, and 10/20
- Population variance ratios for the two groups: 1:1 (equal population variance); 1:2 (σ_2 is

[15]This simplistic approach looks at the ratio of s_1 to s_2 by performing an F-test on the variances of the two groups with the null hypothesis that the ratio of the variances of the populations from which the two samples were drawn is equal to one.

[16]This study only investigated the two-group case.

twice the size of σ_1), 1:4, 2:1, and 4:1

The full design matrix therefore has $4 \times 4 \times 5 = 80$ cells for *each* of the six measures.[17] 1,000 iterations were run, and the number of rejections made at both the nominal five-percent and one-percent levels of significance were recorded ($EDR_{\alpha=.05}$ and $EDR_{\alpha=.01}$, respectively). Overall, this is a fairly straightforward simulation. Other examples in the literature complicate matters by adding additional design elements or outcome measures. However, this case study brings out the essential points I aim to make.

In the results section, Brown and Forsythe (1974) only included findings that pertain to the Gaussian, Student's t, and χ^2 distributions, and only for the 5% level of significance.[18] The findings that are included are presented in one full-page table (see Figure 2.2), and described in-text. In brief, when the group variances were homogeneous, the results for the Gaussian distribution indicated that W_{50} was conservative (failed to reject the null less frequently than the specified alpha rate) for small sample sizes. The results for the other test statistics could be seen as resulting from sampling error. The power of these statistics did not differ greatly when the differences in their Type I error rates are taken into account.

When the distribution was long-tailed (and especially with unequal group sizes), there were problems with the F-test, the jackknife, and the Layard χ^2. Of the Levene-based procedures, W_{10}

[17]This would be the number of cells included in the simulation for a completely crossed design. For this study, some condition combinations are redundant or not applicable and are removed from the analysis. For example, with equal sample sizes per group, the 1:4 and the 4:1 variance ratio conditions are redundant.

[18]Results from the Cauchy distribution were omitted due to redundancy.

Empirical Size and Power ($\alpha = 5\%$)

n_1, n_2	$\sigma_1^2:\sigma_2^2$	F	Jackknife	Layard χ^2	Levene (W_0)	W_{10}	W_{50}
			A. Gaussian distribution				
40, 40	1:1	6.3	5.8	6.5	6.4	6.1	5.1
	2:1	57.5	54.2	56.3	51.1	50.8	48.4
	4:1	98.2	98.1	98.2	97.1	96.9	96.7
10, 10	1:1	5.4	4.6	6.3	5.5	4.9	2.9
	2:1	16.7	13.9	19.5	15.8	14.7	9.8
	4:1	51.3	42.1	51.1	44.3	41.4	31.7
20, 40	1:1	5.8	5.7	6.2	5.8	5.2	4.5
	2:1	43.4	41.3	37.8	38.8	37.5	32.9
	4:1	92.0	89.4	88.4	88.5	88.0	85.7
	1:2	36.6	38.0	42.9	33.2	32.9	31.1
	1:4	92.0	90.5	93.3	86.3	85.3	83.9
10, 20	1:1	4.8	5.2	6.6	5.7	5.2	4.0
	2:1	24.3	19.7	17.7	21.5	20.4	15.7
	4:1	71.7	63.0	59.8	62.2	59.8	52.7
	1:2	16.1	18.4	24.3	15.8	14.8	12.1
	1:4	57.0	57.4	66.0	49.9	48.6	43.0
			B. Student's t on 4 df				
40, 40	1:1	24.1	6.3	4.8	5.0	4.8	4.4
	2:1		33.7	31.2	36.4	34.9	33.2
	4:1		74.3	79.3	87.2	86.4	85.8
10, 10	1:1	15.9	6.8	8.4	5.9	5.3	3.5
	2:1		13.2		12.2	10.3	7.1
	4:1		31.9		32.1	28.6	22.4
20, 40	1:1	21.9	8.2	6.0	5.0	4.7	4.4
	2:1			19.4	27.5	25.7	22.9
	4:1			60.9	75.9	74.2	71.3

Figure 2.2: Truncated results presented in Brown and Forsythe (1974).

was the most robust, as it varied less than W_0 and was nearer to the nominal rate than W_{50}. When the generating distribution was positively skewed, only the W_{50} maintained a nominal rate while all of the other tests rejected too frequently. Overall, Brown and Forsythe (1974) found that the equality of variances is best tested for data generated by long-tailed distributions with the W_{10}, and if the data was derived from an asymmetric distribution, one should use the W_{50} instead.

One particularly notable aspect of this presentation is that there are many cells (found in the non-Gaussian conditions) that appear blank. These conditions were analyzed during the simulation, as Brown and Forsythe (1974) remark, "to avoid distracting the reader, the power results are not shown for tests whose empirical sizes [EDRs] are greater than eight percent and are parenthesized for tests whose sizes are between seven and eight percent" (p. 365). This statement informs the reader that there were many conditions whose results were too liberal[19], and to be aware of them when investigating the power results. This is a fairly common method for highlighting the intersection between Type I error rates and power; however it does yield a large amount of whitespace and it might not be clear at a glance to a casual reader why cells appear missing.

Overall, this paper provides a nice example for our present purposes because a) it's goals are easily understandable; b) the results are not overly cumbersome; c) it is easily replicable; and, d) the method of presenting results is typical for Monte Carlo research. Refer to Appendix B for sample `R` code to replicate this simulation in full, and the results from this study can be loaded from `SimDisplay`:

[19]This conclusion is indicated by an EDR for the Type I error rate condition that is larger than the nominal α value.

```
devtools::install_github("mattsigal/SimDisplay") # Omit if package is already installed
library("SimDisplay") # Load the SimDisplay package
data("Brown1974") # Load the simulation results into the workspace
```

2.3 Standard Tables

Tables have been used to record and present data since the 2nd century common era, and consist of two primary components: data components and support components (Few, 2012). The former category refers to both the quantitative values[20] that appear within a cell, as well as any textual information used to structure those values (e.g., titles, row and column labels, notes, etc.), while the latter pertains to how the table is arranged and how certain components are emphasized. When discussing tabular elements and how they relate to data presentation, it is useful to outline some basic terminology.

Tables are made up of *rows* (horizontal elements) and *columns* (vertical elements). Each unique combination of a row and column is a *cell*, and when taken together the cells form the *body* of the table. Rows that summarize information from previous rows are called *footers* (or *group footers*, if they pertain to a subset of rows), and columns that provide categorical labels to the left of the the table body are called *row headers*, as each cell provides information relevant to a particular row. Similarly, rows above the table that provide labels for particular columns are called *column headers*, and if a cell spans multiple columns it is a *spanner header* (or *decked header*, if it also

[20]Tables can also be used to present qualitative data; however, this is never the case for MCSS and so will not be discussed here.

spans multiple rows). *Rules* for tables refer to horizontal or vertical lines, often used to visually separate elements (e.g., a "header rule" would refer to a vertical line separating the header row from the rest of the table). Finally, if a row partitions or facets a table into multiple subtables (usually by a categorical variable) it is called a *table spanner*.

There is a large literature pertaining to recommendations for formatting tabular displays. As discussed in Section 2.5, these are often linked to issues of design and typography for the purpose of enhancing legibility and highlighting comparisons. An underlying theme of these discussions is the Gestalt of the display and how it interacts with the viewer. This refers to seeing all the elements of the display as a singular experience that can be adapted or modified to assist readers in seeing patterns in the data.[21] Our goal is to encourage such interactions and revelations.

For publication in psychology, the American Psychological Association (APA) provides a brief discussion on their rules for formatting tabular displays in the most recent edition of the *Publication Manual* (2009). The section is prefaced with a warning to researchers that over-reliance on tables and figures can cause difficulties for the reader, the publisher, and effective communication. If a table is necessary, careful thought must be given to what data to present so that it will (a) aid readers in understanding the discussion and (b) provide a sufficient set of statistics to support the inferential methods used in the study.

In all cases, table layout should be done logically, with values meant to be compared placed in relative proximity to each other. As discussed in Section 2.5.3, consideration of how the data are

[21] Particular aspects of this are discussed in Section 3.1.1.

sorted is imperative. For instance, different trends can be emphasized to the reader depending on if the data are sorted or faceted by one or more particular factors.

While the APA guidelines do not explicitly distinguish between structural and data components, they do present guidelines for both aspects. Numerous standards are presented regarding the structural makeup of a table. For instance, if there are cells that are not applicable, they should be left blank, but if the data are missing because it was not obtained for some reason, a dash should be used instead. Further, every column must have a heading and it should not be substantially longer in length than its widest data entry. Decked heads and table spanners allow for the presentation of headers in such a way as to avoid repetition (by stacking content across multiple rows or tables, respectively). Overall design formatting is kept to a minimum: rules are limited to only those that are necessary for clarity. For large tables, a horizontal rule or white space can be inserted after every fourth or fifth entry. Tables can be submitted either single- or double-spaced. However, no vertical rules can be used.

Regarding data content, the APA recommends that decimal values be used to the degree of precision that the measurement justifies and with all entries presented with the same number of decimal places. If confidence intervals are relevant, they should be presented either in brackets or in separate columns for lower and upper limits, with the confidence level clearly indicated. Some element delineation is allowed, specifically the special use of italics, parentheses, dashes, boldface, and special symbols; however, if any of these elements are used, they must be clearly indicated in the table note pertaining to why they have been used.

See Figure 2.3 for an APA style presentation of the Type I error rate results from the motivating example that has been annotated with some of the more esoteric structural element labels.[22] These guidelines are meant for broad application within the discipline, and the *Publication Manual* includes numerous examples of "canonical [tabular] forms" for some of the more typical output found in psychological research. However, none of these forms are perfectly conducive for presenting MCSS results, as will be discussed in the following sections.

2.4 Tidy Data and Tidy Theory

The standards from the APA largely center around a table's visual structure. However, it is also important to think about how to manipulate data to ensure that its presentation is appropriate and straightforward. It is customary for a statistician to spend a substantial amount of time making their data amenable to analysis (often referred to as data munging or data wrangling), and this is equally relevant to MCSS. To help conceptualize MCSS results, it is useful to think of them within the *tidy* framework. The overarching goal of data munging is to produce what is known as *tidy data* (Wickham, 2014b), which is a standard for structuring data, where each variable is represented as a column, each observation as a row, and each type of observational unit can be understood as a table.

This concept of tidy data can also be applied to MCSS, providing insight on how we can utilize

[22]This table also demonstrates how a factor variable (in this case, data generating distribution) can be used to organize a table into specific subsets to show particular comparisons more easily. This idea is discussed further in Section 2.5.5.

Table X: Type I Error Rates in an APA Formatted Table

			Test Statistic					
Stub Head	n_1	n_2	*F*	*Jackknife*	χ^2	W_0	W_{10}	W_{50}
			Gaussian					
Stub or Stub Columns	40	40	4.4	4.9	5.7	5.6	5.2	4.3
	10	10	4.5	4.7	7.0	6.8	5.8	3.9
	20	40	4.3	4.2	5.1	5.1	4.8	4.2
	10	20	3.6	4.6	6.1	5.0	4.6	3.2
			Student's *t* on 4 *df*					
	40	40	26.0	8.3	6.6	7.2	6.3	6.0
	10	10	16.2	6.5	8.5	5.9	5.3	3.7
	20	40	23.4	10.0	6.5	5.7	5.0	4.7
	10	20	16.6	8.5	6.9	5.5	4.6	3.5
			χ^2 on 4 *df*					
	40	40	19.6	6.9	6.8	11.1	7.7	4.9
	10	10	13.5	7.2	9.9	9.7	7.3	3.9
	20	40	19.7	9.6	9.7	11.9	7.9	5.8
	10	20	13.6	8.1	9.0	9.3	7.2	4.3

Table Note — *Note.* χ^2= Layard's χ^2; W_0 = unmodified Levene's Test; W_{10}= Levene's with 10% trim; W_{50}= Levene's with median

Figure 2.3: Sample table presented in recommended APA style.

that paradigm to sort and store our results. Further, this paradigm has been extrapolated to tidy *theory* (Robinson, 2014), in which results from statistical analyses are encoded in such a way to make them easily amenable to visualization and summarization. Specifically, this organization can be achieved by tidying results into three categories: statistical components (e.g., slope and intercepts from a linear regression), observations (e.g., fitted values and standard errors for particular data points in a linear regression), and information at the model level (e.g., fit statistics and R^2 values). A goal of the present research is to show how results from MCSS can be understood in this fashion, and how this structure enables their scrutiny.

MCSS tables presented in journals are often *messy*. Specifically, condition variables are found in both columns and rows, and even used to partition results through table spanners. This practice is done to conserve space, which is at a premium. However, MCSS *data* should be tidy. Results should not span multiple files. Columns should pertain to design factors and output, with appropriate naming conventions. Rows should pertain to singular combinations of design factors. This process also requires collapsing all of the individual replications into appropriate summarizations (thankfully, a properly structured simulation study should do this organization automatically).

In Figure 2.4, we see a tidied version of the results from Brown and Forsythe (1974). Each row pertains to a unique combination of factor variables (the first three columns), and the simulation results are each given their own variable (columns four through nine), having been summarized by the EDR. The final three columns are meta-variables and contain details about the execution of the

	distribution	sample_size	var_ratio	F	Jacknife	Layard	Levene	W10	W50	REPLICATIONS	SIM_TIME	COMPLETED
1	Gaussian	40/40	1	0.038	0.044	0.044	0.040	0.039	0.037	1000	16.476	Tue Apr 25 13:58:44 2017
2	Gaussian	40/40	2	0.578	0.548	0.566	0.529	0.522	0.498	1000	11.693	Tue Apr 25 13:58:56 2017
3	Gaussian	40/40	4	0.995	0.990	0.995	0.983	0.981	0.976	1000	12.042	Tue Apr 25 13:59:08 2017
4	Gaussian	20/40	1	0.042	0.048	0.051	0.051	0.047	0.039	1000	11.398	Tue Apr 25 13:59:19 2017
5	Gaussian	20/40	0.5	0.374	0.398	0.442	0.348	0.344	0.320	1000	13.239	Tue Apr 25 13:59:33 2017
6	Gaussian	20/40	0.25	0.919	0.892	0.914	0.854	0.852	0.832	1000	14.643	Tue Apr 25 13:59:47 2017
7	Gaussian	20/40	2	0.437	0.384	0.393	0.391	0.385	0.345	1000	18.236	Tue Apr 25 14:00:05 2017
8	Gaussian	20/40	4	0.949	0.920	0.898	0.901	0.898	0.873	1000	17.241	Tue Apr 25 14:00:23 2017
9	Gaussian	10/10	1	0.056	0.050	0.065	0.051	0.047	0.034	1000	12.416	Tue Apr 25 14:00:35 2017
10	Gaussian	10/10	2	0.175	0.148	0.199	0.152	0.147	0.112	1000	12.956	Tue Apr 25 14:00:48 2017

Figure 2.4: Tidied results from a replication of Brown and Forsythe (1974), with additional meta variables.

simulation itself.[23] This data structure provides a standardized basis for further visualization and exploration, and will be implicitly utilized by functions within `SimDisplay` to allow convenient summarization and display of results.

2.5 Improving Tabular Displays

Beyond the guidelines from the APA, there are many ways to improve tabular displays, especially for MCSS. Broadly speaking, this discussion targets specific techniques that enable easier comparisons, improve the memorability of results, reinforce the structural components of a table, and those that enhance the presentation of the data components themselves.

[23]By default, the number of replications, simulation runtime, and date completed appear. However, this can be customized and may show other meta-level results, such as the number of times a model failed to converge within a cell or counts of the error messages generated over the course of a simulation.

2.5.1 Promoting comparisons

A primary goal of any display is to promote comparisons across elements. For instance, a simple boxplot may utilize a grouping variable on the x-axis to highlight how the groups differ (in terms of their median and variability) on a particular outcome variable. Likewise, for designs with multiple factors, plots or tables can be separated across the levels of a particular factor. This would place all of the data from one level of a factor variable in one panel of the display and produce separate panels for each level of that variable. In the motivating example, different panels might pertain to the levels of the generating distribution. This was accomplished in Figure 2.3 using table spanners; in graphical displays, this technique is referred to as faceting the display.

Placing certain data elements beside each other (and separating others) in this manner allows for multiple perspectives of a multi-dimensional dataset. However, there is the caveat that it is easier to compare the data within the same panel than it is to do so across different panels. As such, when preparing a static display, the choice of which variable to use to facet the data is very important as it will promote particular comparisons but minimize others. Despite this criticism, this is a powerful technique and often appropriate. Additionally, this can be implemented in an interactive fashion to allow the reader to navigate between faceting variables (see Chapter 4), which circumvents this issue.

2.5.2 Strategies to encourage recall

Few (2012) discusses how data components and support components of tables work in tandem to convey the desired information. The most important realization is that *every* aspect of a table exhibits visual attributes, and that the design of a table involves "not only choosing components that are used to construct the table and display the information, but also determining the visual attributes of each" (Few, 2012, p. 157). To promote recall and understanding, the most memorable visualizations have the following key characteristics (Borkin et al., 2016):

- Should be striking "at-a-glance"
- Have meaningful titles and text
- May include pictograms, or some form of non-data-based graphical element
- Should not shy away from some redundancy, e.g. numerically labeling as well as shading content to demonstrate degree of intensity

These characteristics skew toward the creation of improving infographics, but are useful to keep in mind here as they hint at the idea that researchers should go beyond traditional displays to convey their findings. If the presentation does not provoke attention in some way, it is not as successful as it should be.

For example, Tufte (2006) recommended the use of sparklines, small graphs that provide a quick summary of numerical or statistical information, to accompany textual or tabular presenta-

Group	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	Sparkline
G1	100	150	50	125	75	100	100	100	100	100	
G2	100	100	100	100	100	100	150	50	125	75	
G3	100	100	100	100	150	50	125	75	100	100	
G4	30	40	50	60	70	80	90	100	110	120	
G5	120	110	100	90	80	70	80	90	100	110	
G6	30	50	70	90	110	130	130	130	150	130	

Figure 2.5: Sparkline example using a fictional dataset.

tions. Sparklines are line graphs with no labeling on either axis and are traditionally used to show patterns over time (a natural variable for the horizontal axis). For example, Figure 2.5 presents (contrived) data from six different groups across ten time points. The sparklines in the rightmost column succinctly highlight how the group trends differ.

For MCSS results, such a display might have a separate sparkline for each statistical method under investigation; for example, with the EDR on the y-axis and an important condition variable (such as sample size) on the x-axis.[24] Looking across the set of sparklines, a reader can efficiently spot different patterns of results, as depicted in the following table. In this case, the sparklines show that each of the methods produced Type I error rates around the nominal rate with the exception of the F-statistic when used with non-Gaussian data. This trend could be further highlighted by using a different maximum value for the y-axis.

[24]The sparklines in the following table were generated using the `SimDisplay` function `sparkline()`, which is demonstrated in Section 2.6.

Table 2.1: Type I error rate sparklines

Method	Gaussian	t4	Chi4	Sparkline
F	0.04	0.21	0.17	
Jackknife	0.05	0.08	0.08	
Layard	0.06	0.07	0.09	
Levene	0.06	0.06	0.10	
W10	0.05	0.05	0.07	
W50	0.04	0.04	0.05	

A general guideline is to think of tables as a visual component of a research presentation, and to critically evaluate each of the attributes that make up such a display. Incorporating distinctive elements and making reasonable design decisions about the structural components of a table can go a long way to make it both memorable and easier to decipher.

2.5.3 Improving tabular structural components

When constructing a table, there are many structural components that can be manipulated, with the overarching goal of clearly delineating content across rows and columns. For instance, there is the degree and use of white (or blank) space, the inclusion of horizontal and vertical rules, and the use of color. Care must be made while deciding which of these aspects to include, as too much visual flair causes distraction, while too little causes numbers to perceptually converge.

One recommendation made for tabular displays in particular is to treat these visual aspects hierarchically (Few, 2012). First, rely on white space to separate elements whenever possible. If this does not provide adequate delineation, then incorporate subtle fill colors, and, if additional separation is required, use subtle rules. For MCSS, color can serve multiple purposes: it can be mapped to reinforce information (e.g., values that deviate substantially from the nominal EDR can be highlighted) or to display alternative information. For instance, numerical values might indicate power rates, but shading could highlight Type I error rates for those conditions, which would highlight statistics that may have better power, but at the cost of being more liberal.

Beyond deciding what to color, it is imperative that the palette used is colorblind friendly[25], and the scale must be reasonable. For instance, EDR values are generally "best" when they are relatively equal to their nominal α value. The fill should take that into consideration, and bring into focus both cells that substantially surpass that value and those that underestimate it. Further, depending on the scale chosen for the fill, overlaid text values may blend into the shading. In such cases, researchers must make sure that both the actual number and the color-encoded background are readable. This can be done using a logical statement (e.g., if a particular value is greater than the darkly shaded 0.8, print it in white instead of black) and is incorporated into the `SimDisplay` function for generating semi-graphic tables.

Further, alignment of the cellular elements can aid in readability (Ström, 2016). A general recommendation is to right align numerical data, which properly aligns the digits regardless of the

[25]Such a palette can be generated using the following website: `http://colorbrewer2.org`. Further, a recent `R` package, `viridis` (Garnier, 2017), could also be used to generate the range of colors.

Consider the following vector of coefficients from a fitted model:

```
##                    effect
## (Intercept)  1.135000e+02
## A           -1.350000e+01
## B            4.500000e+00
## C            2.450000e+01
## C1           6.927792e-14
## C2          -1.750000e+00
## D            1.650000e+01
```

Without scientific notation:

```
##             effect
## (Intercept) 113.5
## A           -13.5
## B             4.5
## C            24.5
## C1            0.0
## C2           -1.75
## D            16.5
```

Figure 2.6: Scientific notation comparison.

number of significant figures, and to left align textual data, which aligns the first letter of each entry. The header of each column should mimic the alignment of the content, which provides consistency and context for the display. Finally, tabular elements should never be center aligned, as this can produce "raggedness" if the entries are of differing lengths.

2.5.4 Improving tabular data components

Beyond the structure of the table, there are also decisions to be made about the content of the cells. For instance, how can we display numerical output to be best understood and compared? And, to what degree is rounding beneficial? Numerical output from software is often not in an optimal human readable format (e.g., R often prints numbers using scientific notation). This makes comparisons within a table tricky (see Figure 2.6). When glancing across these values, it is hard to determine which number is actually largest.

One approach to ensure numerical output is easier to compare is to print "lucidly" (Wright, 2016). The goal is find a way to present an arbitrary series of numbers in such a way that the deci-

mal place lines up vertically for all elements. This approach converts numeric output to character vectors, with appropriate white-space to maintain column/digit consistency. As MCSS results are often presented as decimals or percentages, this organization can help if there is substantial variation among methods. For example, the following chunk demonstrates the use of `roundSim()` to do such a conversion, in contrast to the base R output:

```
set.seed(10)
dat <- rnorm(n = 5, mean = 0, sd = 150)

cat(dat, sep="\n") # Use cat() to print each element on a new line

## 2.811926
## -27.63788
## -205.6996
## -89.87516
## 44.18177

roundSim(dat, digits = 3) # In regular vector form

## [1] "   2.812" " -27.638" "-205.700" " -89.875" "  44.182"

cat(roundSim(dat, digits = 3), sep="\n")

##    2.812
##  -27.638
## -205.700
##  -89.875
##   44.182
```

In the output from `roundSim()`, white space is added to the left of the first element to ensure that it is the same width as other values that are larger in magnitude, and the trailing zeroes are kept on the third element to ensure each value is expressed to the same precision. This utility function is best used before or while generating presentation displays, and is used during later examples.

Another feature of `roundSim()` is that it also allows entries to be easily converted into percentages. For instance, using the Brown and Forsythe dataset:

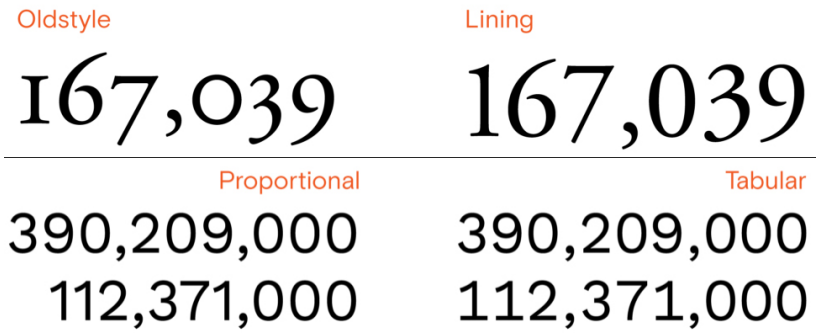


Figure 2.7: Characteristics of typographic structure.

```
data(Brown1974)
class(Brown1974) # Many SimDisplay functions check data is class 'SimDesign'

## [1] "SimDesign"  "data.frame"

roundSim(Brown1974[,4:9], percent = TRUE)[1:5,] # Display the first 5 rows.

##      F Jackknife Layard Levene  W10  W50
## 1  4.4       4.9    5.7    5.6  5.2  4.3
## 2 59.1      56.6   57.8   51.5 51.0 49.0
## 3 99.3      98.6   98.9   97.8 97.6 97.5
## 4  4.5       4.7    7.0    6.8  5.8  3.9
## 5 17.0      14.7   19.1   15.3 14.2  8.2
```

Further, the choice of typography, or the style and appearance of the text on the table, matters. Ström (2016) described characteristics of typefaces that can influence the reader's ability to compare numbers within a data table. These features are: lining versus old-style, and proportional versus tabular (see Figure 2.7).

Lining figures refers to typefaces for which numerical values approximate capital letters in that they are uniform in height, and generally align with top (cap-height) or bottom (baseline) of the other characters. Numerical elements from old-style typefaces often extend above or below these rules. For tabular displays, it is recommended that the typeface have a lining structure, as it

promotes vertical height consistency across the rows.

A typeface is referred to as proportional if the numerical elements have variable spacing (e.g., the display of the value "1" might use less horizontal space on the display than the value "2"), and as tabular if each numeric element is designed to be the same width. Proportional figures are recommended for use in-text, while tabular figures are essential for tables – as the uniform spacing allows for the various columns of digits to align vertically, so long as a consistent number of decimal places is maintained.

Accordingly, the body of a table is best presented using a typeface that is designed as lining-tabular. Two open-source and freely available examples of such a typeface are Open Sans[26] and Lato[27], both from Google. While it is often trivial to change the typeface of text output using basic functions in the document processor, doing so in R graphical output is less straightforward. How to accomplish this will be discussed in Chapter 3.

Overall, an informed selection of typeface results in a table where the numerical elements properly align, and it ensures that both horizontal and vertical comparisons are as natural as possible. However, for MCSS, we often also want to highlight particular entries. In such cases, it is recommended to use emphasis to signify important results (such as bold, italics, or the inclusion of special symbols). If such conditional formatting is used, the rules governing their inclusion should be clearly described in the table note.

[26] `https://fonts.google.com/specimen/Open+Sans`

[27] `https://fonts.google.com/specimen/Lato`

2.5.5 Organizing principles for tabular displays

After considering both the structural and data components of a table, thought should be given to their interaction, specifically what the cells represent and how the table is structured to present those particular values. A table is more than its rows, columns, and formatting – it is a narrative that should tell a story. This is related to the idea of faceting (see Section 2.5.1) and is the primary motivation behind effects ordering (Friendly & Kwan, 2003). This concept draws attention to the fact that all data presentations are necessarily ordered (e.g., alphabetically, by a grouping factor, by time), although the order is often chosen by convention or, worse, by the software that generated the display.

Effects ordering pertains to general principles for ordering factors and variables in tables and graphs. It can be accomplished by sorting by marginal statistics, effect sizes, or more mathematically complex techniques.[28] The approach chosen by the researcher demonstrates how given data values can be rendered in different ways for different purposes.

To this end, rows and columns of a tabular display should be presented in a meaningful manner. Miller (2007) presented a straightforward framework for organizing tables for ease of use depending on whether they were designed to accompany a prose description or to be used for reference. As the findings from MCSS are typically presented within the pages of journal articles, on academic posters, or on slides at a conference, they are typically accompanied by some form of description. As such, Miller proposes a few different methods to consider for their presentation:

[28] Such as through eigenvalue or singular-value decompositions or other algorithms discussed later in this chapter.

1. Organize rows and columns to best highlight some form of empirical ordering (based on value or frequency; e.g., organizing rows by their average EDR)

2. Organize rows and columns by some form of theoretical grouping (to create conceptually related sets, such as grouping a set of analyses by one or more of the condition variables)

3. Use multiple organizing criteria, for instance:

    (a) group cells theoretically then arrange within groups via some empirical consideration

    (b) arrange cells alphabetically within conceptual or empirical groupings

No matter how a table is organized, a description should be included in-text that clearly indicates the organizing principle used. This choice largely depends on the type of variables in question; however, a general guideline is to organize the data in the order that will be used when describing them in the results.

These strategies are useful for most typical tables. However, those from MCSS studies often have many rows and columns, such that approaches that rely on manual rearrangement often become unwieldy. As such, it is helpful to conceptualize the results as a matrix that can be manipulated to best highlight various trends through rotation (e.g., manipulating rows using clustering algorithms; Hurley, 2004). This is an approach popularized by Bertin (1981) as the *reorderable matrix*.

2.5.5.1 Reorderable matrices

The concept of the reorderable matrix was popularized by Bertin (1967; 1981, English translation), and refers to a particular form of presentation of tabular data (Siirtola & Mäkinen, 2005). This process entails transforming a multidimensional data set into a two-dimensional (2D) interactive graphic. The graphic mimics a tabular display, as it maintains the common row and column structure of the original matrix. However, these values are subsequently modified through two processes:

- *Construction*: where data values are replaced with symbols (e.g., circles or rectangles), which are given a size that is relative to the actual data value; and,
- *Reconstruction*: where the rows and columns of the matrix are manually permuted, one at a time, which allows different views of the data set.

This process is visually represented in Figure 2.8. No apparent trends appear in the original data (the table in the top left of the display); however, through the reconstruction process, a clear separation between urban and rural locations becomes evident in the final step.

Bertin's (1967; 1981) original method relied heavily on visual judgment and manual manipulation (Perin, Dragicevic, & Fekete, 2014), and is very impractical for large datasets. See Figure 2.9 for an example of what a physical representation of the reorderable matrix might look like.

The feasibility of this approach is also impeded by the fact that researchers must craft a physical reorderable matrix for each data set they wish to analyze, and that the steps taken during recon-

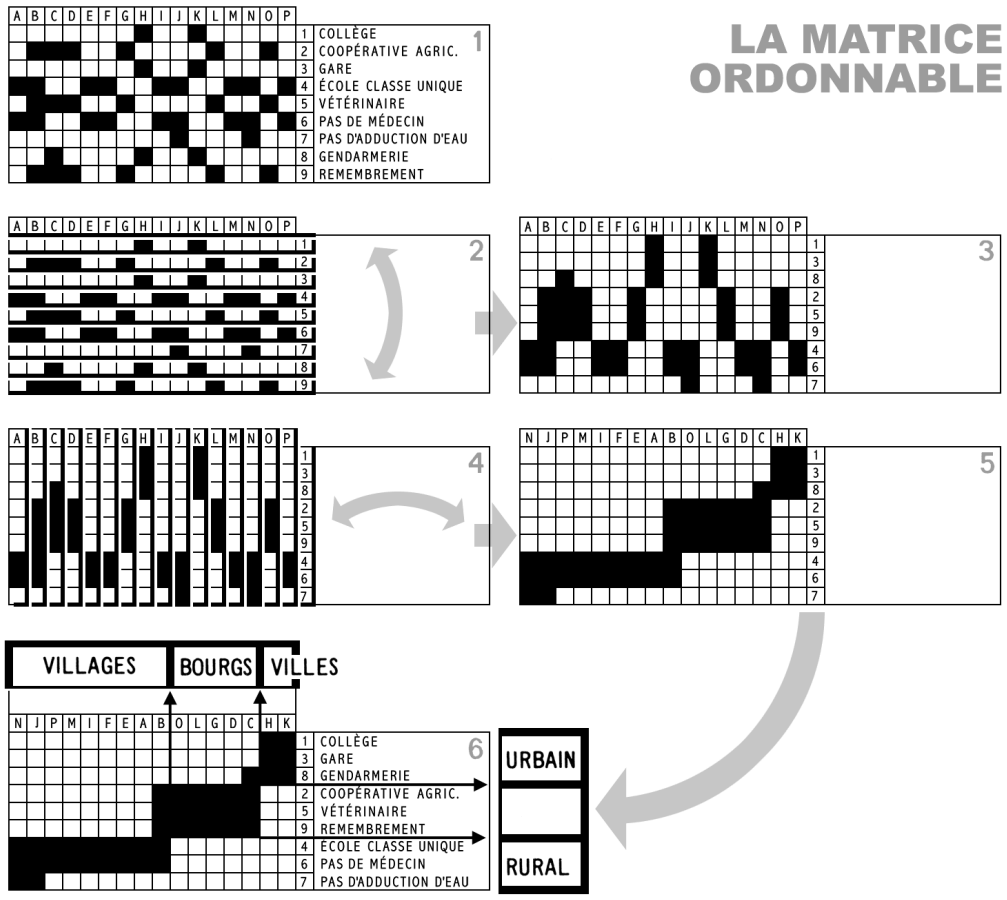


Figure 2.8: Bertin's reorderable matrix (source: `http://rocbo.net/technique/illustr_tech/40.html`)

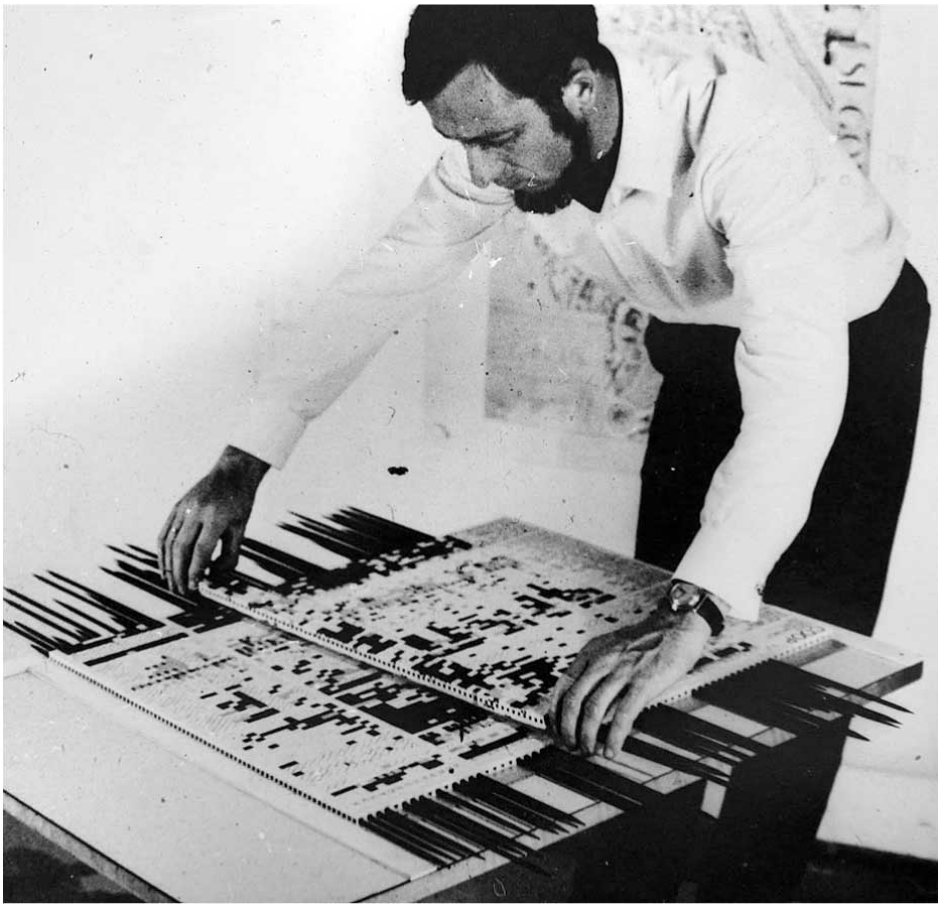


Figure 2.9: Serge Bonin with a physical reorderable matrix (source: http://www.aviz.fr/diyMatrix/)

struction may not necessarily be recorded, which impedes the reproducibility of the results. This idea was given a digital implementation (the "Bertifier") by Perin et al. (2014). This free-to-use web applet[29] has many features, allowing users to upload their own datasets, modify the symbols used to represent their data, and adjust the height, width, and placement of the rows and columns in the matrix.

The Bertifier also introduces a few methods for automatically determining column and row placement. More generally, such algorithms include:

- Cluster analysis and hierarchical clustering;
- Principal components analysis;
- Multidimensional scaling; and,
- The Bond energy algorithm (BEA)

Such methods sort the matrix to fulfill different criteria, and are useful to apply as a starting point when exploring a large set of results. To illustrate such a strategy, the following section demonstrates the use of the BEA on the data used in Figure 2.8.

2.5.5.2 The Bond Energy Algorithm

McCormick, Schweitzer, and White (1972) defined a measure of effectiveness (ME) for an n by m matrix $\mathbf{X} = (x_{ij})$ as:

[29] `http://www.bertifier.com/`

$$M(\mathbf{X}) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \left[x_{i,j+1} + x_{i,j-1} + x_{i+1,j} + x_{i-1,j} \right]$$

This value is maximized if each element in the matrix is as closely related numerically to its four neighboring elements (the two vertical and two horizontal) as possible. The BEA uses the ME to reconstruct a given matrix according to the following steps:

1. Randomly select one column of a data matrix;
2. Place each remaining column at each possible position to the left, right, and between the already placed columns;
3. For each placement, calculate the change in ME;
4. Choose the column and position that results in the largest increase in ME and place that column;
5. Repeat until all columns are placed; and then,
6. Repeat the above process for the rows.

This algorithm is currently implemented in `R` in the `seriation` package (Hahsler, Hornik, & Buchta, 2008), and executed as follows:

```r
install.packages("seriation") # Omit if package is already installed
library("seriation") # Load the seriate package
data(Townships) # Load the Bertin townships dataset
BEA <- seriate(x = Townships, method = "BEA") # Apply the BEA algorithm
bertinplot(x = Townships, order = BEA) # Plot the sorted matrix
```

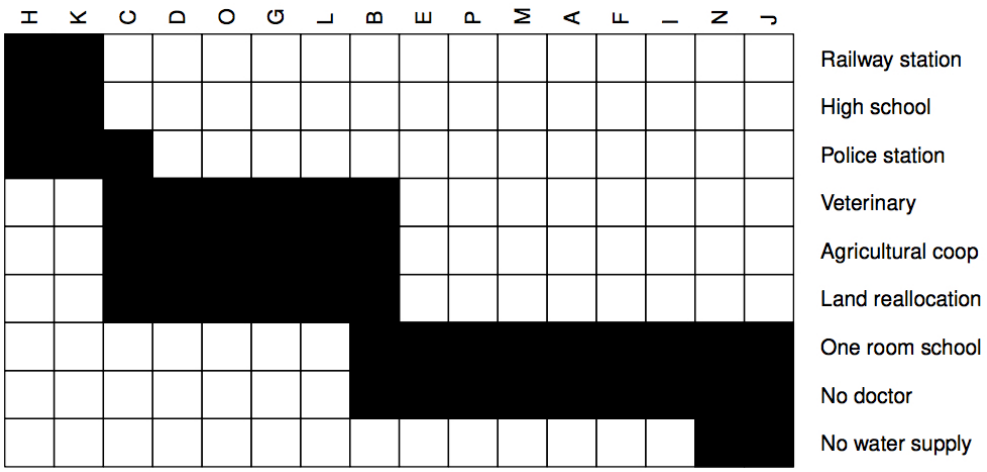


Figure 2.10: BEA Clustering of Bertin's Township Data

This code produces Figure 2.10, which adequately reproduces Bertin's (1967) manual ordering demonstrated above. A similar approach has been applied to sorting the data presented in clustered heatmaps (Wilkinson & Friendly, 2009), and is discussed in Chapter 3.

2.5.6 Summarizing MCSS output via ANOVA

While using clustering algorithms to sort the table may provide certain insights, it is often beneficial to scale down large dimensional tables. Applying a statistical model, such as analysis of variance (ANOVA), allows researchers to examine effects across conditions. This procedure often provides a good starting point for thinking about trends across conditions.

In this approach, the goal is to highlight mean differences across cell combinations (for example, differences in Type I error rates across sample size and variance ratio conditions) by looking at their effect sizes. It should be noted here that power rates for particular tests should only be inter-

preted when their Type I error rates are nominal, as liberal detection rates result in more powerful tests, while conservative detection rates are necessarily linked to lower power.

The first step in such an analysis is to separate the Type I error rates from the power conditions. This call is simulation-dependent, but it can be accomplished with the Brown and Forsythe (1974) dataset by calling `subset()` on the `var_ratio` variable:

```
data(Brown1974)
TypeI <- subset(Brown1974, var_ratio == 1) # View only Type I error conditions
```

Because the aim of the Brown and Forsythe study was to evaluate methods for detecting the homogeneity of variance, the Type I error rate conditions are the cells in which there were no model-specified differences in variance across groups. To get a sense of these effects we can call `simTable()`, which exports a LaTeX table with cell emphasis (this example uses the default upper and lower bounds), proper alignment, custom variable names, and lucid printing, as follows:

```
simTable(TypeI,
         colnames = c("Dist.","Sample Size","Var. Ratio",
                      "F","Jackknife","Layard","Levene","W10","W50"))
```

Table 2.2: Type I error rates summarized with `simTable()`

Dist.	Sample Size	Var. Ratio	F	Jackknife	Layard	Levene	W10	W50
Gaussian	40/40	1	0.04	0.05	0.06	0.06	0.05	0.04
Gaussian	10/10	1	0.04	0.05	0.07	0.07	0.06	0.04
Gaussian	20/40	1	0.04	0.04	0.05	0.05	0.05	0.04
Gaussian	10/20	1	0.04	0.05	0.06	0.05	0.05	0.03
t4	40/40	1	**0.26**	**0.08**	0.07	0.07	0.06	0.06
t4	10/10	1	**0.16**	0.06	**0.08**	0.06	0.05	0.04
t4	20/40	1	**0.23**	**0.10**	0.06	0.06	0.05	0.05
t4	10/20	1	**0.17**	**0.08**	0.07	0.06	0.05	0.04
Chi4	40/40	1	**0.20**	0.07	0.07	**0.11**	**0.08**	0.05
Chi4	10/10	1	**0.14**	0.07	**0.10**	**0.10**	0.07	0.04
Chi4	20/40	1	**0.20**	**0.10**	**0.10**	**0.12**	**0.08**	0.06
Chi4	10/20	1	**0.14**	**0.08**	**0.09**	**0.09**	0.07	0.04

This function can further be used to collapse across a grouping variable and to hide superfluous columns from view:

```
simTable(TypeI, by = 'sample_size', hide = 'var_ratio')
```

Table 2.3: Type I error rates, collapsed over sample size

distribution	F	Jackknife	Layard	Levene	W10	W50
Gaussian	0.04	0.05	0.06	0.06	0.05	0.04
t4	**0.21**	**0.08**	0.07	0.06	0.05	0.04
Chi4	**0.17**	**0.08**	**0.09**	**0.10**	**0.08**	0.05

The descriptive rates seem very favorable for the W_{50} test in that they are all close to the nominal detection rate. However, the other tests appear too liberal in most conditions. Though we could inspect various tables to see why or where this result occurred, sometimes it is easier to use ANOVA to pinpoint these effects.

One important note about this procedure is that to conduct an ANOVA there must be within-group variability. For MCSS, this requirement means that something in the design must be marginalized (or averaged across), preferably based upon a priori knowledge. In this example, we might marginalize sample size because Type I error rates should not be influenced by N and so long as it does not interact with the other design factors.[30] Depending on the simulation, this marginalization can be done for each response variable separately using `SimAnova()`:

```
SimAnova(F ~ distribution, TypeI)

##                 SS df    MS      F  p sig eta.sq eta.sq.part
## distribution 7.213  2 3.607 64.264  0 ***  0.935       0.935
## Residuals    0.505  9 0.056     NA NA      0.065          NA
```

[30]Of course, we could also run `simTable()`, marginalizing across a different variable to see if this statement holds.

However, if there are many response variables, it is more convenient to use a function that will run all of the analyses at once. This can be accomplished using an extension of the previous function, `SimAnovaMV()`, which has an argument for the design factors of interest and for the response variables to include in the output. Both of these parameters are passed as character vectors.

```
SimAnovaMV(data = TypeI, terms = "distribution",
           responses = c("F", "Jackknife", "Layard", "Levene", "W10", "W50"))

##      Method      F     p eta.sq
## 1         F 64.264 0.000  0.935
## 2 Jackknife 21.269 0.000  0.825
## 3    Layard  7.210 0.014  0.616
## 4    Levene 29.789 0.000  0.869
## 5       W10 19.085 0.001  0.809
## 6       W50  0.995 0.407  0.181
```

While most of the above effects are significant, emphasis should primarily placed upon the effect size measures. In this case, there are notable differences in η^2, with W_{50} being the least and the Layard χ^2 the most affected by generating distribution.

For MCSS with more design factors, `SimAnovaMV()` also calculates and reports the interaction terms between the them. For example, this function can be applied to the power results (in which both distribution and variance ratio is of interest) from this study:

```
Power <- subset(Brown1974, var_ratio != 1) # View only power conditions
SimAnovaMV(data = Power, terms = c("distribution", "var_ratio"),
           responses = c("F", "Jackknife", "Layard", "Levene", "W10", "W50"))

## $distribution
##      Method     F     p eta.sq
## 1         F 0.202 0.819  0.008
## 2 Jackknife 3.166 0.060  0.105
## 3    Layard 3.580 0.044  0.104
## 4    Levene 1.144 0.335  0.041
## 5       W10 1.292 0.293  0.048
## 6       W50 1.039 0.369  0.043
```

```
##
## $var_ratio
##      Method      F     p eta.sq
## 1         F  9.014 0.000  0.506
## 2 Jackknife  9.438 0.000  0.467
## 3    Layard 11.935 0.000  0.521
## 4    Levene  9.419 0.000  0.507
## 5       W10  8.761 0.000  0.489
## 6       W50  7.308 0.001  0.450
##
## $`distribution:var_ratio`
##      Method     F     p eta.sq
## 1         F 0.326 0.917  0.037
## 2 Jackknife 0.323 0.918  0.032
## 3    Layard 0.285 0.938  0.025
## 4    Levene 0.198 0.974  0.021
## 5       W10 0.155 0.986  0.017
## 6       W50 0.123 0.993  0.015
```

This approach provides an easy way of examining the pattern of the sum of squares and the effect sizes for a multiple response, multiple factor simulation study. For substantially more complex studies, it is of particular interest to examine the interaction effects between the design factors.[31] Higher order patterns can be assessed at a glance from the output by looking at the decomposition of the sum of squares and effect sizes. Another benefit of this strategy is that such ANOVA models have natural methods for visualization. As discussed in Chapter 3, boxplots, effect plots, and HE plots could be used here to visualize such effects.

2.6 `SimDisplay` Implementation

Thus far, this chapter has focused on introducing methods and techniques that can be used to enhance tabular displays. The remainder shows additional functions and optional arguments from the

[31] In straightforward designs, such as in the motivating example for the Type I error rate conditions, there are not enough design factors to estimate an interaction effect. Similarly, the power conditions for this particular dataset lack the degrees of freedom for estimating an interaction effect.

SimDisplay package that highlight these ideas and how they apply to the results of MCSS. The following demonstrations use the replicated results from Brown and Forsythe (1974), as described in Appendix B, and are available through the SimDisplay package via data(Brown1974). These functions are designed to work with SimDesign objects but can also be applied to generic dataframes if desired.

When a simulation ends, the first task of the researcher is to get a summary of the output. SimDisplay has a generic summary() method for SimDesign objects. The default output is to provide a list, with separate tables for each design variable. This can be used as an exploratory tool to show the researcher what trends occur across the particular factors in the design.

```
library(SimDisplay)
data(Brown1974) # Load the Brown and Forsythe dataset
summary(Brown1974)

## $distribution
##   distribution     F Jackknife Layard Levene   W10   W50
## 1     Gaussian 0.425     0.405  0.428  0.392 0.382 0.349
## 2           t4 0.465     0.301  0.316  0.313 0.294 0.271
## 3         Chi4 0.451     0.312  0.338  0.381 0.335 0.280
##
## $sample_size
##   sample_size     F Jackknife Layard Levene   W10   W50
## 1       40/40 0.556     0.437  0.447  0.476 0.461 0.443
## 2       10/10 0.276     0.190  0.242  0.207 0.181 0.129
## 3       20/40 0.553     0.428  0.436  0.459 0.436 0.408
## 4       10/20 0.378     0.281  0.305  0.290 0.258 0.209
##
## $var_ratio
##   var_ratio     F Jackknife Layard Levene   W10   W50
## 1      0.25 0.731     0.590  0.613  0.658 0.628 0.576
## 2       0.5 0.349     0.252  0.250  0.281 0.253 0.217
## 3         1 0.138     0.070  0.073  0.074 0.060 0.044
## 4         2 0.370     0.256  0.275  0.266 0.241 0.202
## 5         4 0.740     0.610  0.663  0.639 0.608 0.558
```

Depending on the structure of the MCSS, the above summary is somewhat misleading, as it combines the Type I error and the power conditions. A better approach would be to subset the

results beforehand[32]:

```
type1 <- subset(Brown1974, var_ratio == 1)
power <- subset(Brown1974, var_ratio != 1)
summary(type1)

## $distribution
##   distribution     F Jackknife Layard Levene   W10   W50
## 1     Gaussian 0.042     0.046  0.060  0.056 0.051 0.039
## 2           t4 0.206     0.083  0.071  0.061 0.053 0.045
## 3         Chi4 0.166     0.080  0.088  0.105 0.075 0.047
##
## $sample_size
##   sample_size     F Jackknife Layard Levene   W10   W50
## 1       40/40 0.167     0.067  0.064  0.080 0.064 0.051
## 2       10/10 0.114     0.061  0.085  0.075 0.061 0.038
## 3       20/40 0.158     0.079  0.071  0.076 0.059 0.049
## 4       10/20 0.113     0.071  0.073  0.066 0.055 0.037
##
## $var_ratio
##   var_ratio     F Jackknife Layard Levene   W10   W50
## 1         1 0.138     0.070  0.073  0.074 0.060 0.044

summary(power)

## $distribution
##   distribution     F Jackknife Layard Levene   W10   W50
## 1     Gaussian 0.553     0.525  0.551  0.504 0.493 0.452
## 2           t4 0.551     0.373  0.398  0.398 0.375 0.347
## 3         Chi4 0.546     0.389  0.421  0.473 0.422 0.357
##
## $sample_size
##   sample_size     F Jackknife Layard Levene   W10   W50
## 1       40/40 0.750     0.623  0.639  0.675 0.660 0.639
## 2       10/10 0.357     0.254  0.320  0.272 0.240 0.175
## 3       20/40 0.652     0.516  0.527  0.555 0.531 0.498
## 4       10/20 0.444     0.333  0.363  0.346 0.309 0.252
##
## $var_ratio
##   var_ratio     F Jackknife Layard Levene   W10   W50
## 1      0.25 0.731     0.590  0.613  0.658 0.628 0.576
## 2       0.5 0.349     0.252  0.250  0.281 0.253 0.217
## 3         2 0.370     0.256  0.275  0.266 0.241 0.202
## 4         4 0.740     0.610  0.663  0.639 0.608 0.558
```

Tables can be created using `simTable()`, which takes a dataframe object and produces a table

[32]Technical note: The `subset()` function being called here is a generic method for dataframes of class `SimDesign`, and will maintain the additional attributes of those objects.

that is easily embedded in either an `Rmarkdown` report[33] or a LaTeX document. It has the following arguments, and the following chunk demonstrates their use and produces Table 2.4.

- `dat`: A data.frame object of `class('SimDesign')`.
- `by`: A optional character value indicating a factor variable to collapse results across.
- `upper.bound`: An integer value indicating the upper limit for cell emphasis.
- `lower.bound`: An integer value indicating the lower limit for cell emphasis.
- `colnames`: An optional character vector indicating column names to use in the output (to replace those found in the dataset).
- `lucid`: An optional boolean value indicating if results should be printing lucidly (same length, as characters).
- `digits`: If `lucid` is true, this is an integer value indicating the number of digits to round to.
- `caption`: A character value indicating a table caption (used by the document processor).
- `hide`: A character vector indicating columns to hide from view.

```
simTable(dat = type1,
         by = 'sample_size',
         upper.bound = .075,
         lower.bound = .025,
         colnames = c('Distribution','F','Jackknife',
                      'Layard','Levene','W10','W50'),
```

[33] If this function is being used within this context, the chunk option `results='asis'` is needed for proper rendering.

```
        lucid = TRUE,
        digits = 2,
        caption = 'simTable example.',
        hide = 'var_ratio')
```

One positive aspect of SimDesign objects is that they retain a substantial amount of meta data pertaining to the conditions under which the simulation was conducted. For instance, embedded in the dataframe is information on the computer which ran the simulation, the time and date each condition completed, the packages used by the simulation, and so on. However, if MCSS results are going to be passed to a function outside of the SimDesign and SimDisplay framework, it is often useful to simplify the object. This simplification can be accomplished by passing the dataframe to simplifyDf():[34]

```
df <- simplifyDf(Brown1974)
str(df)
```

```
## 'data.frame': 48 obs. of  9 variables:
##  $ distribution: Factor w/ 3 levels "Gaussian","t4",..: 1 1 1 1 1 1 1 1 1 ...
##  $ sample_size : Factor w/ 4 levels "40/40","10/10",..: 1 1 1 2 2 2 3 3 3 ...
##  $ var_ratio   : Ord.factor w/ 5 levels "0.25"<"0.5"<"1"<..: 3 4 5 3 4 5 3 4 5 2 ...
##  $ F           : num  0.044 0.591 0.993 0.045 0.17 0.523 0.043 0.43 0.933 0.383 ...
##  $ Jackknife   : num  0.049 0.566 0.986 0.047 0.147 0.449 0.042 0.394 0.903 0.385 ...
##  $ Layard      : num  0.057 0.578 0.989 0.07 0.191 0.54 0.051 0.36 0.882 0.436 ...
##  $ Levene      : num  0.056 0.515 0.978 0.068 0.153 0.454 0.051 0.374 0.872 0.347 ...
##  $ W10         : num  0.052 0.51 0.976 0.058 0.142 0.426 0.048 0.361 0.863 0.343 ...
##  $ W50         : num  0.043 0.49 0.975 0.039 0.082 0.337 0.042 0.321 0.844 0.325 ...
```

Conversely, if a simulation was conducted outside of SimDesign, a generic dataframe can be converted to one of class SimDesign using convertDf():

```
# Sample design with 3 variables: sample size, standard
# deviation ratio, and degree of kewness
```

[34]While it might seem natural for this function to be a method for as.data.frame(), that conflicts with a function in SimDesign. As the two functions actually perform slightly different operations, a new name was selected here. In the future these functions may be merged.

```
n <- c(15, 30, 50)
sd <- c(0.5, 1, 2)
skew <- c(1, 5, 10)
dat <- expand.grid(N = n, SD = sd, Skew = skew,
                   KEEP.OUT.ATTRS = FALSE)
set.seed(123)
dat$Result1 <- rnorm(27) # Generate example results
dat$Result2 <- rnorm(27)

dat[,1:3] <- lapply(dat[,1:3], as.factor) # Design variables must be set as factors
newdat <- convertDf(dat)
str(newdat)

## Classes 'SimDesign' and 'data.frame': 27 obs. of  5 variables:
##  $ N      : Factor w/ 3 levels "15","30","50": 1 2 3 1 2 3 1 2 3 1 ...
##  $ SD     : Factor w/ 3 levels "0.5","1","2": 1 1 1 2 2 2 3 3 3 1 ...
##  $ Skew   : Factor w/ 3 levels "1","5","10": 1 1 1 1 1 1 1 1 1 2 ...
##  $ Result1: num  -0.5605 -0.2302 1.5587 0.0705 0.1293 ...
##  $ Result2: num  0.153 -1.138 1.254 0.426 -0.295 ...
##  - attr(*, "design_names")=List of 2
##   ..$ design: chr  "N" "SD" "Skew"
##   ..$ sim   : chr  "Result1" "Result2"
```

As demonstrated earlier in the chapter, sparklines can be incorporated into a tabular display to highlight trends, which can be accomplished using `SimDisplay`:

```
Power <- subset(Brown1974, var_ratio != 1) # Separate Type I error rate conditions

power_summary <- summary(Power)
# Create dataframe pertaining to distribution conditions:
distsummary <- data.frame(t(power_summary[[1]])[2:7,])

# Convert all variables to numeric and give
# appropriate variable and row names:
out <- data.frame(lapply(distsummary,
                         function(x) as.numeric(as.character(x))))
colnames(out) <- levels(power_summary$distribution$distribution)
out$Method <- rownames(distsummary)

# Reorder dataframe:
out <- data.frame(Method = out[,4], out[,1:3])

# Generate sparklines and save to 'figures' subdirectory:
for(i in 1:nrow(out)) {
  png(paste0("figures/", sprintf("power-%s.png", i)), height = 300)
  sparkline(out[i, 2:4], min = 0, max = .75)
  dev.off()
}

# Generate LaTeX code to include sparklines in table:
out$Sparkline <- sprintf('\\raisebox{-.4\\totalheight}{
```

```
                        \\includegraphics[width=0.25\\textwidth, height=7.5mm]
                        {figures/power-%s.png}}', seq(1:nrow(out)))
```

```
# Generate table:
newout <- xtable::xtable(out,
                         caption = "Power Rate Sparklines")

# Include table in output:
print(newout, type='latex',
        sanitize.text.function=identity,
        include.rownames=FALSE, table.placement="H",
        caption.placement = "top")
```

Table 2.4: Power Rate Sparklines

Method	Gaussian	t4	Chi4	Sparkline
F	0.55	0.55	0.55	
Jackknife	0.53	0.37	0.39	
Layard	0.55	0.40	0.42	
Levene	0.50	0.40	0.47	
W10	0.49	0.38	0.42	
W50	0.45	0.35	0.36	

Another option for displaying MCSS tables is with a semi-graphic approach that utilizes color to enhance output. These ideas are expanded upon in Chapter 3; however, they demonstrate a semi-graphic use of tabular information. The following code produces Figure 2.11 in which Type I error and power rates pertaining to the EDRs from the Gaussian condition have been shaded using

a colorblind friendly palette.[35]

In this case, the rows pertaining to the largest variance ratio conditions are very noticeable. This code is condensed significantly in Chapter 3 where custom functions are provided to handle the majority of the data management and plotting:

```
# Simplify dataset and create condition variable with all levels:
dat <- subset(Brown1974, distribution == "Gaussian")
dat <- simplifyDf(dat)
dat$condition <- paste0(dat$sample_size, " by ", dat$var_ratio, ":1")
dat <- dat[,c("condition", "F", "Jackknife", "Layard", "Levene", "W10", "W50")]

# Convert dataframe to long format:
dat <- dat[c(3,2,1,6,5,4,9,8,7,10,11,14,13,12,15,16),]
dat.m <- melt(dat, id.vars = "condition")
dat.m$value <- dat.m$value*100 # Display values as percentages
dat.m$condition <- ordered(dat.m$condition,
                           levels=unique(as.character(dat.m$condition)))
dat.m <- dat.m[order(dat.m$condition), ]

# Create ggplot graphic:
p <- ggplot(dat.m, aes(variable, condition,
                       label = paste0(sprintf("%.1f", round(dat.m$value,1)),"%"))) +
  geom_tile(aes(fill = value), colour = "white") +
  scale_fill_gradientn(colours=brewer.pal(n=9, name="BuPu")[1:7]) +
  geom_text(colour = 'black', size = 5, hjust="right",  nudge_x = .4) +
  scale_x_discrete(position = "top")  +  xlab("") + ylab("") +
  ggtitle(bquote(list("Type I Error Rates and Power for Gaussian", alpha==".05"))) +
  labs(fill=bquote(list("Proportion of" ~ italic(p) ~ "<" ~ alpha))) +
  theme(legend.position = "top", plot.title = element_text(hjust = 0.5)) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  ylim(rev(levels(dat.m$condition)))
p
```

The following code and Figure 2.12 present the same information; however, it has been sorted using a principal components analysis clustering method. Even an algorithmic-based sorting method lumps together the rows pertaining to the Type I error rate conditions and moves them

[35]It should be noted that what is highlighted in such plots is very sensitive to the scheme used for shading. MCSS results are always continuous and so require a palette with a range of values. In the following, this is handled by the `scale_fill_gradientn()` call, which requests a gradient be created based upon the `BuPu` (blue-purple) palette. For the `SimDisplay` convenience functions, allowing for control over the palette is an important feature (and one that can be implemented interactively; see Chapter 4).

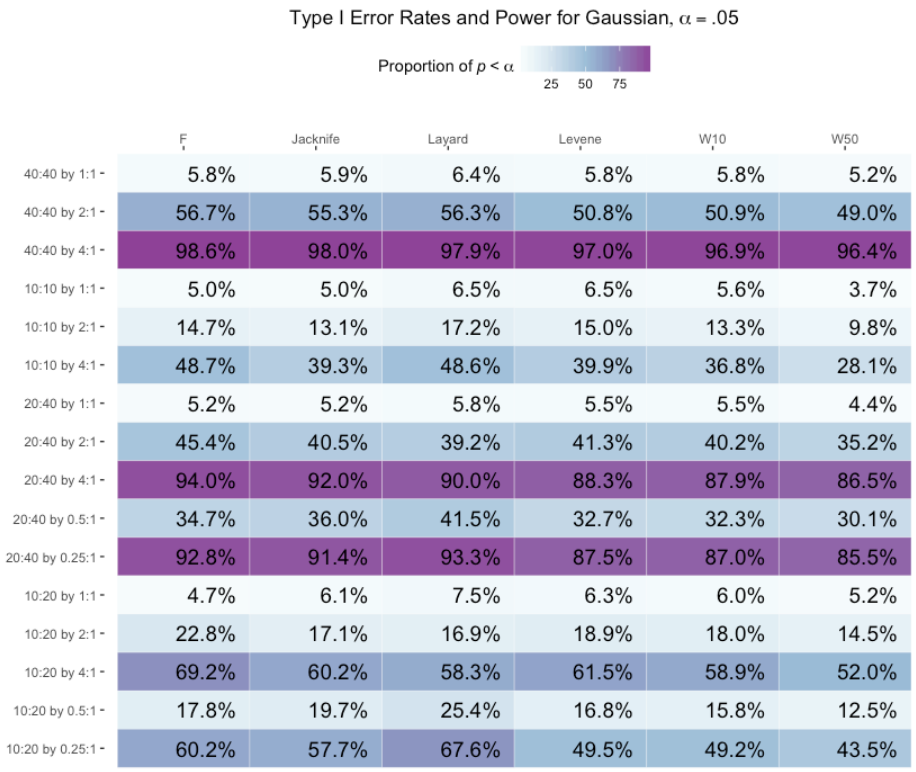


Figure 2.11: Unordered Type I and power rates for the Gaussian distribution

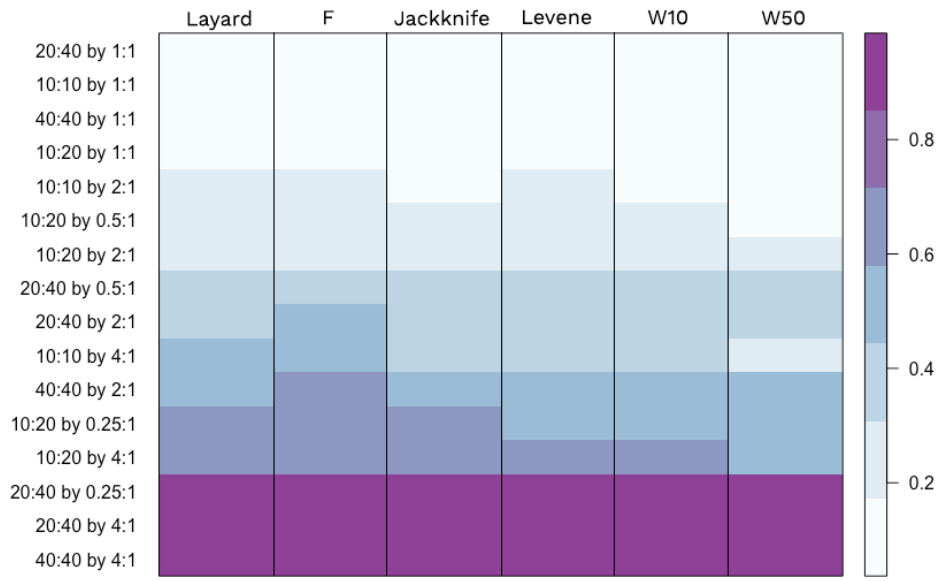


Figure 2.12: Using PCA Clustering with Gaussian EDRs

to the top of the display. Similarly, the largest variance ratio conditions appear at the bottom.

```
library(seriation)
newdat <- as.matrix(dat[,2:7]) # Extract only the data columns from the results
row.names(newdat) <- dat[,1] # Set rownames to ensure information is maintained
order <- seriate(newdat, method = "PCA") # Reorder dataset

pimage(newdat, order, axes = "y", col = brewer.pal(n=9, name="BuPu")[1:7])
```

Finally, to demonstrate a different trend, the following code and Figure 2.13 present the cells pertaining to the $\chi^2(4)$ distribution, in which the liberalness of the F-test is apparent:

```
dat <- subset(Brown1974, distribution == "Chi4")
dat <- simplifyDf(dat)
dat$condition <- paste0(dat$sample_size, " by ", dat$var_ratio, ":1")
dat <- dat[,c("condition", "F", "Jackknife", "Layard", "Levene", "W10", "W50")]

dat.m <- melt(dat, id.vars = "condition")
dat.m$value <- dat.m$value*100
dat.m$condition <- ordered(dat.m$condition, levels=unique(as.character(dat.m$condition)) )
dat.m <- dat.m[order(dat.m$condition), ]

p <- ggplot(dat.m, aes(variable, condition,
                       label = paste0(sprintf("%.1f", round(dat.m$value,1)),"%"))) +
```

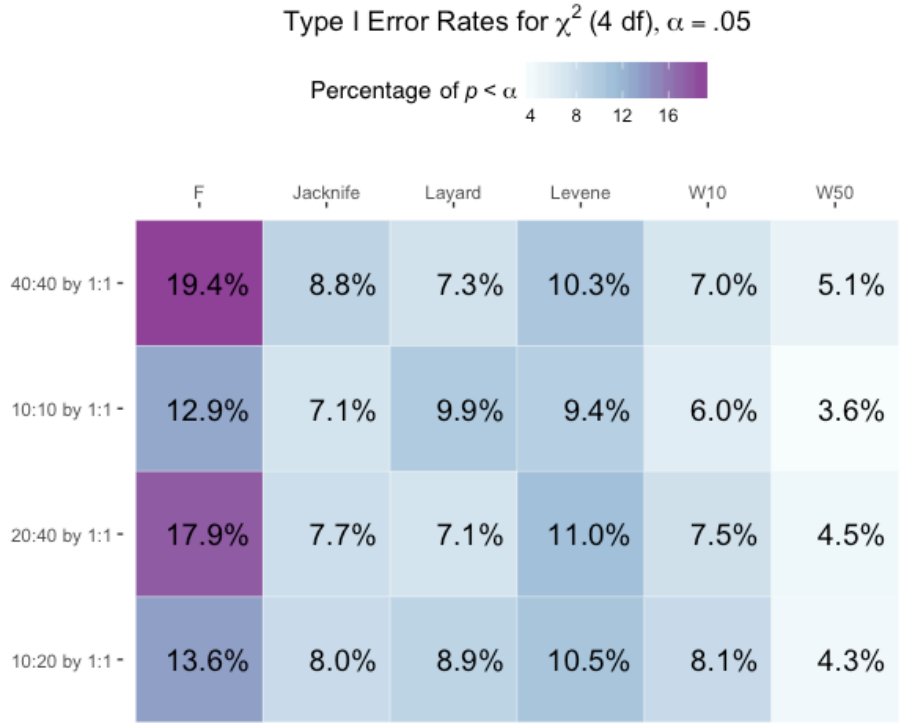


Figure 2.13: Results from the χ^2 distribution with 4 *df*

```
  geom_tile(aes(fill = value), colour = "white") +
  scale_fill_gradientn(colours=brewer.pal(n=9, name="BuPu")[1:7]) +
  geom_text(colour = 'black', size = 5, hjust="right",  nudge_x = .4) +
  scale_x_discrete(position = "top")  +  xlab("") + ylab("") +
  ggtitle(bquote(list("Type I Error Rates for" ~ chi^2 ~ "(4 df)", alpha==".05"))) +
  labs(fill=bquote(list("Proportion of" ~ italic(p) ~ "<" ~ alpha))) +
  theme(legend.position = "top", plot.title = element_text(hjust = 0.5)) +
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  ylim(rev(levels(dat.m$condition)))
p
```

2.7 Supplementary Example

To further illustrate the methods discussed in this dissertation, results from another simulation study will also be displayed. The skeleton of this experiment is based upon the first example presented in Hallgren (2013). In brief, it investigates the behavior of the Sobel test statistic in mediation analysis, and is expanded here through the inclusion of additional design factor levels.

Mediation analysis is a statistical technique that utilizes ordinary least squares regression for testing for the presence of "mediators" (M) or variables that "mediate" a relationship between two other variables: a predictor variable, X, and a response variable, Y (Mackinnon, Fairchild, & Fritz, 2007; Tofighi & MacKinnon, 2016). These variables are said to be mediated by M when their association is conditional upon the inclusion of M. As such, mediation requires the presence of four relationships:

1. A significant relationship between X and M (regression coefficient a);
2. A significant relationship between M and Y (regression coefficient b);
3. A significant relationship between X and Y (regression coefficient c); and,
4. When X and M are entered in the model simultaneously, the relationship between X and Y (c') is substantially reduced (Baron & Kenny, 1986).

In other words, when mediation is present, the degree to which X is associated with Y (regression coefficient c), is changed when M is added to the model (c'). This is visualized in Figure 2.14.

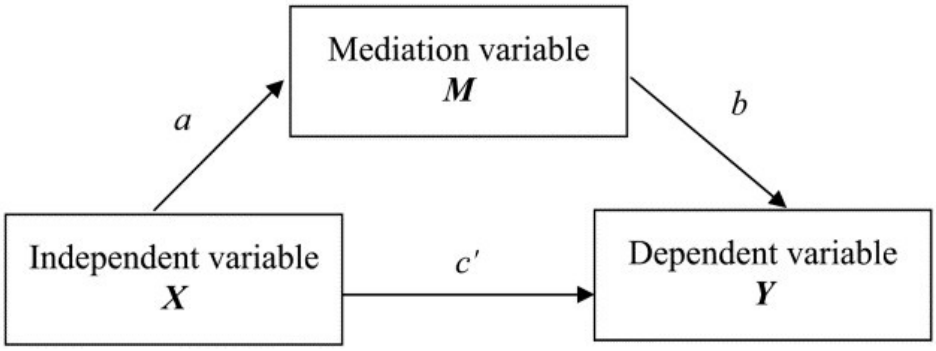


Figure 2.14: Regression paths in a mediation analysis

Testing these relationships can result in either full mediation (where a significant association between X and Y disappears after including M as a predictor), partial mediation (where a significant c relationship is substantially reduced but still significant after including M), or no mediation.

The degree to which the relationship between X and Y changes when the mediator is added or removed (c - c') is called the indirect effect, and represents the amount of change in Y that can be attributed to changes in X operating through M. The Sobel test assesses the statistical significance of this indirect effect, which is calculated as:

$$\text{Sobel } Z = \frac{ab}{\sqrt{b^2 s_a^2 + a^2 s_b^2}},$$

where s_a and s_b are the standard errors of the estimates for the regression coefficients a and b, respectively. The product ab reflects the degree to which the effect of X and Y is mediated through M. The Sobel test provides a Z-like test statistic that assesses whether the indirect effect is significantly different from zero. This technique is relatively popular in psychology and has been

recommended as a method for identifying variables that may cause *Y* as part of a causal sequence where *X* affects *M*, and then *M* affects *Y* (Hallgren, 2013).

The present simulation investigates the consequences of wrongly specifying the variables in the mediation model. Specifically, the goal is to contrast models in which the relationship is properly specified (*X*-*M*-*Y*) against those in which *M* and *Y* have been swapped (*X*-*Y*-*M*). This can happen in applied settings when previous research is not clear about the true nature of the relationship between the variables, and so it is "important to know whether mediation models are likely to produce significant results even when the true causal order of effects is incorrectly specified by investigators" (Hallgren, 2013, p. 48).

2.7.1 Simulation parameters

For each run of this simulation, data is generated for three variables: *X*, *M*, and *Y*, such that *M* mediates the relationship between *X* and *Y*. Within this structure, four design variables will be manipulated:

1. Coefficients for regression path *a* will be manipulated at three levels (*a* = -0.3, 0.0, 0.3);
2. Coefficients for regression path *b* will be manipulated at three levels (*b* = -0.3, 0.0, 0.3);
3. Coefficients for regression path *c′* will be manipulated at three levels (*c'* = -0.2, 0.0, 0.2); and,
4. Sample size (*N*) will be manipulated at 3 levels (*N* = 30, 100, 300).

This produces a 3 (a) x 3 (b) x 3 (c') x 3 (N) design. As in the original paper, one thousand simulated datasets will be generated for each condition and Sobel Z-test p-values will be extracted for two models (X-M-Y and the misspecified X-Y-M). Results are summarized using `EDR()`, which tallies the proportion of runs within a cell for each model with $p < 0.05$. See Appendix C for full details on executing this simulation and can be loaded into an `R` workspace as follows:

```
library(SimDisplay)
data("Hallgren2013")
```

2.7.2 Tabular results

An initial view of the results can be used by the generic `summary()` function for objects of class `SimDesign`. This collapses results by each of the four design factors and, in the following, displays the EDRs as percentages.

```
summary(Hallgren2013, digits = 1, percent = TRUE)
```

```
## $N
##     N XMY.p XYM.p
## 1  30   1.8   1.0
## 2 100  22.3  11.1
## 3 300  45.8  37.7
##
## $a
##      a XMY.p XYM.p
## 1 -0.3  34.5  16.4
## 2    0   1.1  16.6
## 3  0.3  34.4  16.7
##
## $b
##      b XMY.p XYM.p
## 1 -0.3  34.3  24.7
## 2    0   1.2   0.5
## 3  0.3  34.5  24.5
##
## $cp
##     cp XMY.p XYM.p
## 1 -0.2  23.3  22.5
```

```
## 2    0  23.3   5.0
## 3  0.2  23.4  22.2
```

The obvious finding that the power to detect mediation depends on sample size is apparent from the first section of the output, with only 1.8% of simulations with correctly specified models and 1.0% of incorrectly specified models rejecting the null. However, these results are difficult to interpret further as they combine the effects of *a*, *b*, and *c'*, which were each manipulated at three levels. Further, the condition were *a, b*, and *c'* all equal zero represents the Type I error rate for the two models, and should be given special attention separate from the other combinations.

```
# Assign all rows where a, b, and c equal 0 to 'TypeI' dataframe
TypeI <- subset(Hallgren2013, a == 0 & b == 0 & cp == 0)

# Assign all other rows to 'Power' dataframe
Power <- Hallgren2013[-as.numeric(rownames(TypeI)),]
```

As a check, the following summary table presents the Type I error results. In this case, we would expect the Type I error rates to be minimal for both the XYM and the XMY model, since the Sobel test depends on the strength of the regression coefficients.

```
simTable(TypeI, caption = "Type I error rates for Hallgren (2013).", digits = 3)
```

Table 2.5: Type I error rates for Hallgren (2013).

N	a	b	cp	XMY.p	XYM.p
30	0	0	0	0.000	0.000
100	0	0	0	0.001	0.001
300	0	0	0	0.000	0.001

As expected, the rounded EDRs for all three conditions are equal to zero. The following table shows the power rates for the conditions where *a*, *b*, and *c'* are all greater than or equal to zero, sorted by sample size. This is done using `order()` to sort the dataframe and `subset()` to return the desired rows.

```
Power2 <- Power[order(Power$N),]
Power2 <- subset(Power2, a >= 0 & b >=0 & cp >= 0)
simTable(Power2,
         highlight = FALSE,
         caption = "Power rates for Hallgren (2013).")
```

Table 2.6: Power rates for Hallgren (2013).

N	a	b	cp	XMY.p	XYM.p
30	0.3	0	0	0.01	0.00
30	0	0.3	0	0.00	0.00
30	0.3	0.3	0	0.03	0.00
30	0	0	0.2	0.00	0.00
30	0.3	0	0.2	0.00	0.00
30	0	0.3	0.2	0.01	0.01
30	0.3	0.3	0.2	0.04	0.04
100	0.3	0	0	0.01	0.00
100	0	0.3	0	0.02	0.01
100	0.3	0.3	0	0.49	0.04
100	0	0	0.2	0.00	0.00
100	0.3	0	0.2	0.01	0.00
100	0	0.3	0.2	0.02	0.20
100	0.3	0.3	0.2	0.50	0.43
300	0.3	0	0	0.05	0.00
300	0	0.3	0	0.02	0.04
300	0.3	0.3	0	1.00	0.27
300	0	0	0.2	0.00	0.02
300	0.3	0	0.2	0.04	0.02
300	0	0.3	0.2	0.04	0.86
300	0.3	0.3	0.2	1.00	0.99

The following code and Figure 2.15 demonstrates the use of PCA sorting on this dataset. As before, a condition variable is first created that condenses the design variables into one string. The dataframe is then simplified using that variable and the two outcome variables. The `seriate()` function applies the PCA algorithm to the dataframe, and `pimage()` plots the result.

```
dat <- simplifyDf(Power2)
dat$condition <- paste0(dat$N, "/", dat$a,  "/", dat$b, "/", dat$cp)
dat <- dat[,c("condition", "XMY.p", "XYM.p")]

library(seriation)
library(RColorBrewer)
newdat <- as.matrix(dat[,2:3]) # Extract only data columns
row.names(newdat) <- dat[,1] # Set rownames to condition variable
order <- seriate(newdat, method = "PCA") # Reorder dataset
pimage(newdat, order, axes = "y",
       col = brewer.pal(n=9, name="BuPu")[1:7]) # Generate display
```

In these fuller displays, it is again apparent that sample size has a large impact on detecting both proper and improper mediation. When n was 30, the condition with the largest number of rejections only saw 4% of runs to be significant (with a = .3, b = .3, and c' = .2); however, this rate was equivalent for both the XMY and the XYM model. When n = 100, mediation was found 49% of the time for the XMY model when a = .3, b = .3, and c' = .0, and only 4% of the time for the XYM model for that condition. However, when c' increased to .2, the difference in rejection rate between XMY and XYM diminishes to only 7%. This trend is worrisome, and replicated in the n = 300 condition, where when a = .3, b = .3, and c' = .2, both the XMY and the XYM almost always found significant mediation.

The similarities between the results for the XMY and XYM models suggests a practical limitation of using mediation analysis. This simulation suggests that the same datasets can produce

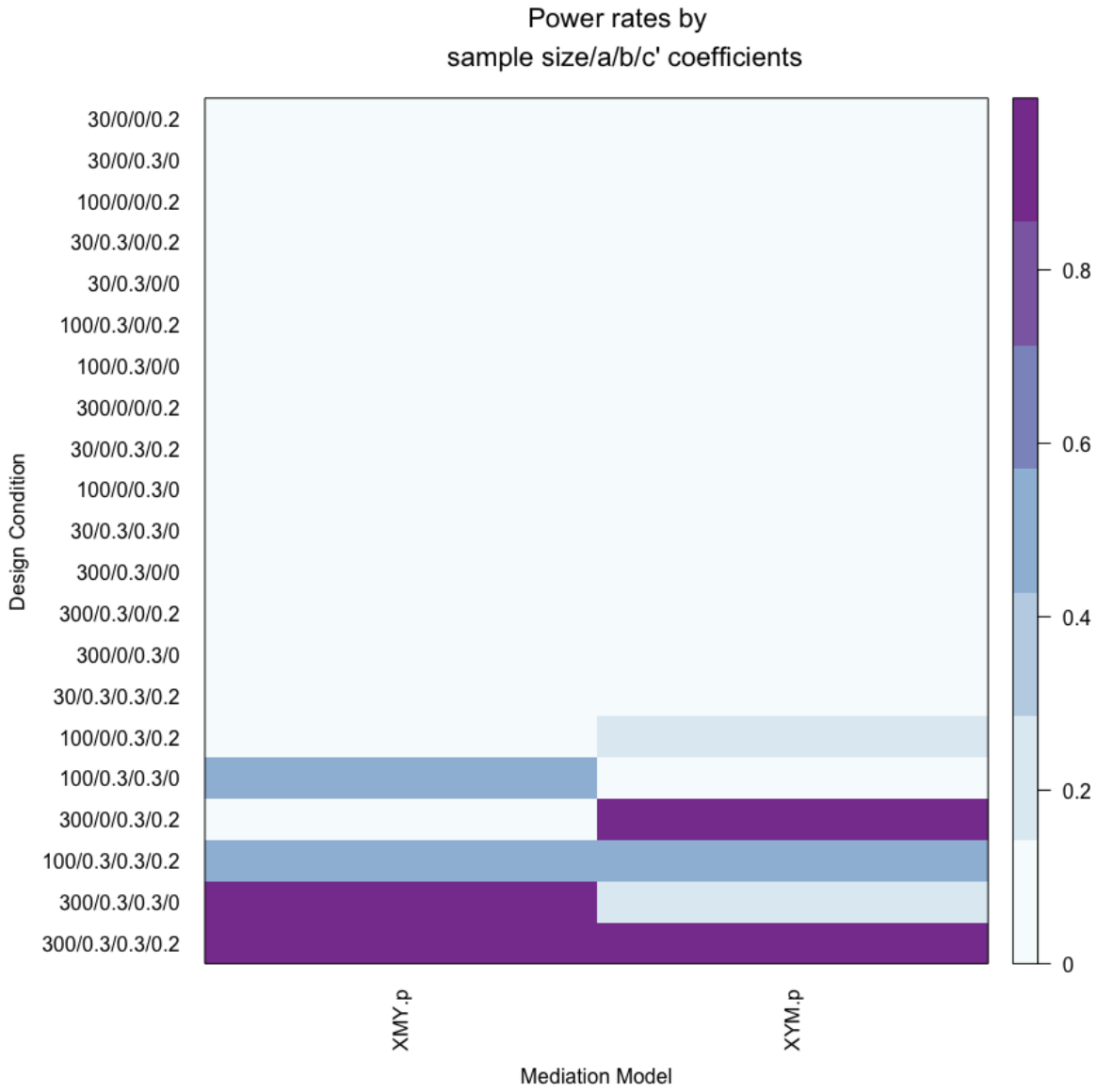


Figure 2.15: Using PCA Clustering on the power results from Hallgren (2013)

significant mediation detection results under a variety of models, which imply different orderings of the variables, e.g. a variable that is truly a mediator might be specified as an outcome and still produce "significant" results and thus mislead practitioners about the true nature of the phenomenon under study. Other methods for displaying these results will be included in Chapters 3 and 4.

2.8 Conclusion

Static tables can convey substantial amounts of information and should still be recognized as a suitable method for presenting MCSS results, if thoughtfully constructed. As tables become more complex, the added value of proper sorting of factor levels, well-chosen color schemes and cut-values, and appropriate typography over a typical numerical table *increases*. These ideas and the functions found in `SimDisplay` package should make it substantially easier to transfer these recommendations into practice.

3 Visualizing the Structure

Tables are the standard method for presenting MCSS results. However, semi-graphic and graphical displays can provide a strong narrative for understanding a large amount of information efficiently. In reference to communicating MCSS results in particular, Boomsma (2013, p. 532) observed that "it is generally recognized that graphs can convey information that might not be noticed otherwise, at least not as easily as in numerical tables, and that the potential impact of the graphs is important." A. R. Cook and Teo (2011) tested this statement – using MCSS tables sourced from the literature and graphical adaptations designed by the authors – by comparing respondent accuracy between the methods of presentation. They found that information could be extracted more efficiently and, for less experienced participants, more accurately from the graphical presentations compared to the original tabular displays.

MCSS generate data that are structured differently than typical research. Generally speaking, studies in psychology produce data that follows a "wide" structure: information from each participant is placed on a row and all of the variables of interest are organized as columns. However, we have seen that a row for MCSS data usually pertain to either one replication or a summary

of replications from a particular combination of design conditions. As such, each combination of design levels can be treated like a participant. Before delving into how to translate such information into a display, a brief discussion of the fundamentals of graphic design is necessary. Such methods involve applying symbolic references based on the numerical output, such as through the use of figures and color. These enhanced summaries make it easier to see underlying patterns in simulation results at a glance compared with traditional, unaltered tables.

Accordingly, this chapter begins with a discussion of the importance of graphical presentation. Information recall is often better when text and images are combined (Lidwell et al., 2003). The chapter will then outline effective strategies to employ in the creation of such displays. This outline is followed by an overview of fundamental graphical terminology, specifically regarding the grammar of graphics framework (Wilkinson, 2005) and its implementation in the `R` package `ggplot2` (Wickham, 2009). With these essentials in hand, the remainder of the chapter will demonstrate how particular graphical techniques can be applied to MCSS results, how they can be used to emphasize particular findings, and how they can be produced using `SimDisplay`. For instance, as introduced in Chapter 2, semi-graphic tables can be used to both aid the researcher in their preliminary explorations of results and for subsequent presentation. Appropriate shading and structure can highlight differences, and make understanding the findings of an experiment substantially more straightforward. Next, it will be shown how tableplots can be used to visualize MCSS data. Further, effect and biplots for assessing group mean differences, and the use of hypothesis-error plots will also be discussed. Overall, the goal of this chapter is to examine and discuss various

graphical approaches for aiding the presentation of MCSS data, as well as showcase convenience functions for generating them in R.

3.1 The importance of graphics

Statistical graphics consist of a statistical component and a visual component (Cleveland, 1993). This interplay between data (what we wish to show) and design (how we show it) is what separates good graphics from ineffective ones. Tukey (1990) wrote that the best graphical displays possess three elements: immediacy, interocularity, and inescapability. More succinctly, this means that their point is made clearly and rapidly 'right between the eyes' and in such a way that one cannot avoid its message. Successful graphics communicate complex ideas with clarity, precision and efficiency (Tufte, 2001), which encourages easy recognition of phenomena. While this goal may seem lofty, it is tenable by first and foremost focusing on *showing* the data, then showing the data *accurately*, and showing the data *clearly* (Gordon & Finch, 2015; Wainer, 1984).

A substantial amount has been written on how to design graphics to best achieve such aims (e.g., Cairo, 2013; Cleveland, 1994; Evergreen, 2014; Few, 2012; Rendgen, 2012; Tufte, 1990, 1997, 2001, 2006; Unwin, 2008; Ware, 2004; Yau, 2011), and on how to best fall short of them (e.g., Wainer, 1984). An overarching point is that good graphical displays work because they reduce "the burden of storing, ordering, and summarizing raw data... [which] frees bandwidth for higher levels of information synthesis and allows observers to note outliers, understand relationships between variables, and form new hypotheses" (VanderPlas & Hofmann, 2016, p. 231). As

such, creating and being presented with graphical displays is often a generative and useful experience (West, 2006).

This observation is especially true if the display has been well-constructed. Gordon and Finch (2015) put forward five principles of graphical excellence:

1. Shows the data clearly (e.g., do not choose a graphical form that obscures the data)
2. Uses simplicity in design (e.g., do not overcomplicate the graphic)
3. Uses good alignment on a common scale for the quantities to be compared (to ensure consistency across plots)
4. Keeps the visual encoding transparent (always label plot elements, such as axes or distinctive symbols, to promote clarity)
5. Uses proper graphical forms that are consistent with the above principles

Keeping these principles in mind will tend to produce a graphic that is clean, comprehensible, and to the point.

3.1.1 Perceptual issues

Limits on attention span and neural information storage mechanisms make it difficult for readers to process numerical information from raw tabular forms effectively (Scaife & Rogers, 1996; Zhang, 1997). A properly designed graphical display instead serves as a form of external cognition in that

a particular ordering and visual summarization of the data has been preprocessed for the viewer. However, once a graphical display is produced, it is of interest to know if it is being perceived in the manner in which it was intended as effectively as possible. Work in this vein has been done to evaluate graphics from a cognitive psychology perspective.

Fundamental research in this area pertains to the ability of participants to judge differences based on visual attributes of a graphical data presentation. For instance, Cleveland and McGill (1984) found that participants were best at differentiating between values when a common scale was used across displays. Accuracy sequentially decreased when judgments were made between displays with non-aligned scales; when they were based on geometric properties, such as length, direction, angle, area, volume, or curvature; and were least accurate when based on shading and color saturation. In contrast, Lewandowsky and Spence (1989) found that participants were best able to differentiate information presented in scatterplots when points were grouped by color, and then by fill and shape. Accuracy of estimation is also affected by more basic elements of the display. For example, when shown two scatterplots of the same data, participants systematically underestimated the correlation coefficient in the second display when the x- and y-axis scales were condensed (Cleveland, Diaconis, & McGill, 1982).

Graphical design involves coordinating the processing of numerous data elements, and attempting to aid that process through informed design decisions. In this way, including a graphic is inviting the reader to experience the data in a curated fashion. VanderPlas and Hofmann (2016) highlighted the interaction between pre-attentive perception (apprehension that occurs after the first

200 ms of exposure to a visual stimuli) and Gestalt heuristics.[36] In this framework, data points are understood as related if they are visually placed in close proximity to each other, if they share common visual aesthetics, and if they demonstrate "good continuation". The researchers found that providing additional but distinct visual heuristics, such as including data summary ellipses (incorporating the proximity heuristic) to a scatterplot that had previously only used color to designate between groups (the point similarity heuristic) helped participants understand the graphic. However, adding different point shapes for the grouping variable to a plot that already had group designed via color aesthetics only increased cluster recognition slightly. The authors hypothesized that this was because the additional element did not provide an additional Gestalt heuristic but instead only emphasized point similarity through two different mechanisms.

Additional research has proposed that graphical and text or tabular output should not be seen as distinct enterprises. For instance, Lane and Sándor (2009) argued that incorporating text onto a graphical display can convey a substantial amount of information. This incorporation can be done through annotations (the labelling of interesting points or areas of a graphic with text), which can be used without losing any of the clarity of the original design. This is also a fundamental point made throughout Yau's *Visualize This* (2011), in which substantial discussion is given to adding narrative elements to graphical displays which allow them to speak for themselves, narratively-speaking.

[36]In this context, Gestalt heuristics refers to seeing the perception of a graphic as a holistic experience, where aesthetic choices (such as the choice of scale and color) are critical in the understanding of said graphic. See also Cairo (2013, chapter 6).

Underlying these discussions are a few fundamental questions: Beyond the basic components, what makes up a graphic? What language can we use to describe their structure? And how can such a language be translated into an applied software environment?

3.2 The Grammar of Graphics

> The grammar of graphics takes us beyond a limited set of charts (words) to an almost unlimited world of graphical forms (statements) (Wilkinson, 2005, p. 1).

Statistical graphics are complex objects, with a substantial number of intricate parts and attributes. Most programs designed for statistical analysis allow the creation of such displays; however, they are often limited by technical constraints - whether they are only coded for particular data structures or feature limited customizability. In 2005, Wilkinson improved the discussion of graphics by replacing chart typographies (where graphical forms are seen as static and distinct entities) with an overarching language, in which all manner of graphics can be understood as being constructed from a set of general concepts. This system is commonly referred to as the grammar of graphics and is broken up into three components**:** *specification* (translating what we expect to happen into a formal language), *assembly* (the coordination of the specified attributes), and *display* (the actual rendering of the graphic onto a display system). Specification is the most formal of these components and pertains to six primary operations that allow for the construction of a graphic:

1. **Algebra**: Operations which combine variables into a dataset to be plotted (referred to as the

"varset")[37] and specify the dimensionality of the graph

2. **Scales**: How the variables will be represented on the measured dimensions

3. **Statistics**: Functions that allow graphs to change appearance and representation schemes through the use of summary statistics (e.g., group averages instead of raw data points)

4. **Geometry**: The creation of geometric objects from variables

5. **Coordinates**: Which coordinate system will be used for plotting (e.g., polar, map projections, Cartesian)

6. **Aesthetics**: Which sensory attributes should be added to the graphic

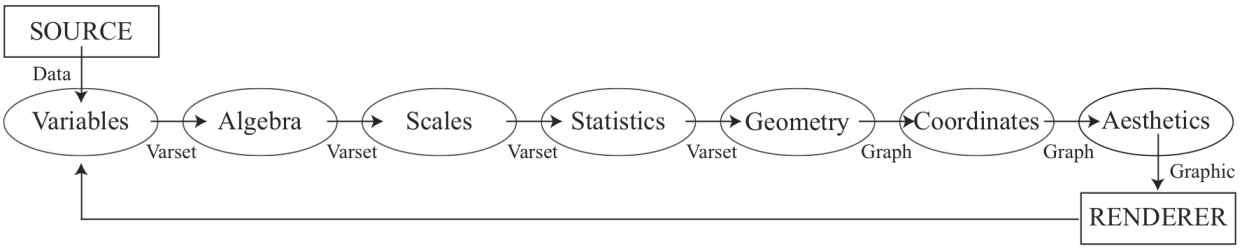
   (a) **Facets and Guides**: Supplemental elements that allow for coordination between graphs and tables, and annotations

These operations provide the fundamental workflow for graphic creation and are shown in Figure 3.1. Traditionally, this process is thought of as being one-way – the creation of a graphic begins with step one and continues through step six. However, this process is iterated, so one is free to subsequently modify one of the above operations, and thus rerun the full workflow, until they are content with the final graphic.

In this framework, any statistical graphic can be expressed during the specification step in terms

[37] This includes manipulations done to the data such as filtering, subsetting, mutating, etc.

Figure 3.1: Grammar of Graphics framework.



of six statements, expressed here using Graphics Production Library syntax (GPL; the programming language created by Wilkinson as an expression of the grammar of graphics):

1. DATA: Expressions that involve the creation of variables to display from datasets

2. TRANS: Statements that apply variable transformations prior to plotting (e.g., ranking the data points)

3. ELEMENT: Statements that define the graphical elements (e.g., points) and their aesthetic attributes (e.g., color) to use in the display

4. SCALE: Expressions that apply scale transformations to the plot (e.g., square root or log)

5. GUIDE: Statements that define the guides incorporated on the plot that aid interpretation (e.g. axes, legends, etc.)

6. COORD: Statements that select the coordinate system for use in the graphic (e.g., Cartesian, polar)

The grammar of graphics paradigm means that, rather than having many different functions, each of which produces a different sort of plot, there is a small set of functions, each of which produces a different sort of plot *component*, and those components can be *combined* in many different ways to produce a huge variety of plots (Murrell, 2011). As a conceptual example, this is how a grouped scatterplot might be expressed using GPL:

```
DATA: x = "SepalLength"
DATA: y = "SepalWidth"
DATA: z = "species"
TRANS: x = x
TRANS: y = y
ELEMENT: point(position(x*y), color(z))
COORD: rect(dim(1,2))
SCALE: linear(dim(1))
SCALE: linear(dim(2))
GUIDE: axis(dim(1), label("Sepal Length"))
GUIDE: axis(dim(2), label("Sepal Width"))
```

In relation to the conceptual framework of the grammar of graphics, the first three lines of the above syntax pertain to choosing the variables to add to the "varset" (see p. 89) and subsequently plot. The TRANS statements could apply algebraic functions to the variables but here simply plots the variables as they are found in the dataset. ELEMENT lines pertain to the creation of the geometric objects used in the display – in this case, the display will use points located at x on the x-axis, y on the y-axis, and grouped by color for the levels of z. COORD and SCALE work in conjunction to define the coordinate system of the plot. In the above, a traditional rectangular Cartesian coordinate system with two linearly scaled dimensions is used.[38] Finally, the GUIDE statements apply the axis labels that annotate the plot.

[38]This statement can be expanded. For instance, a three-dimensional plot could be specified with `COORD: rect(dim(1,2,3))` or the y-axis could be defined using a log scale with a maximum value of 100 with `SCALE: linear(dim(2), max(100))`.

This approach was subsequently adopted by Wickham (2009) and is the basis for the R package ggplot2. In this system, a graphic is produced using four steps:

1. Call the function ggplot() to define the data to plot and create a plot template object
2. Specify the aesthetics of the graphic elements that will be used to represent the data with aes()
3. Specify the geometric objects (geoms) that will be used to view the data and add them with the appropriate function
   (a) For instance, geom_point() to add points to a scatterplot type display or geom_line() to draw lines
4. Print the object to render and display it

This process creates a ggplot2 list object which consists of:

- One or more Layers, which are nested list objects that contain the following elements:
  - Data: The data being passed to the layer
  - Mapping: The aesthetics for the layer
  - Stat: Statistical transformations to apply to the data (e.g., binning or averaging)
  - Geom: The *geometric objects* to be drawn to represent the data
  - Position: Position adjustments for each geom (e.g., jitter)

- `Scale`: Controls mapping between data and aesthetics (variable or constant; color or position)
- `Themes`: Applies visual adjustments to a plot object based upon a pre-existing theme
- `Coord`: The coordinate system (provides axes and gridlines)
- `Facet`: Allows data to be broken up into panels or separate plots

In this framework, the grouped scatterplot described above could be generated as follows:

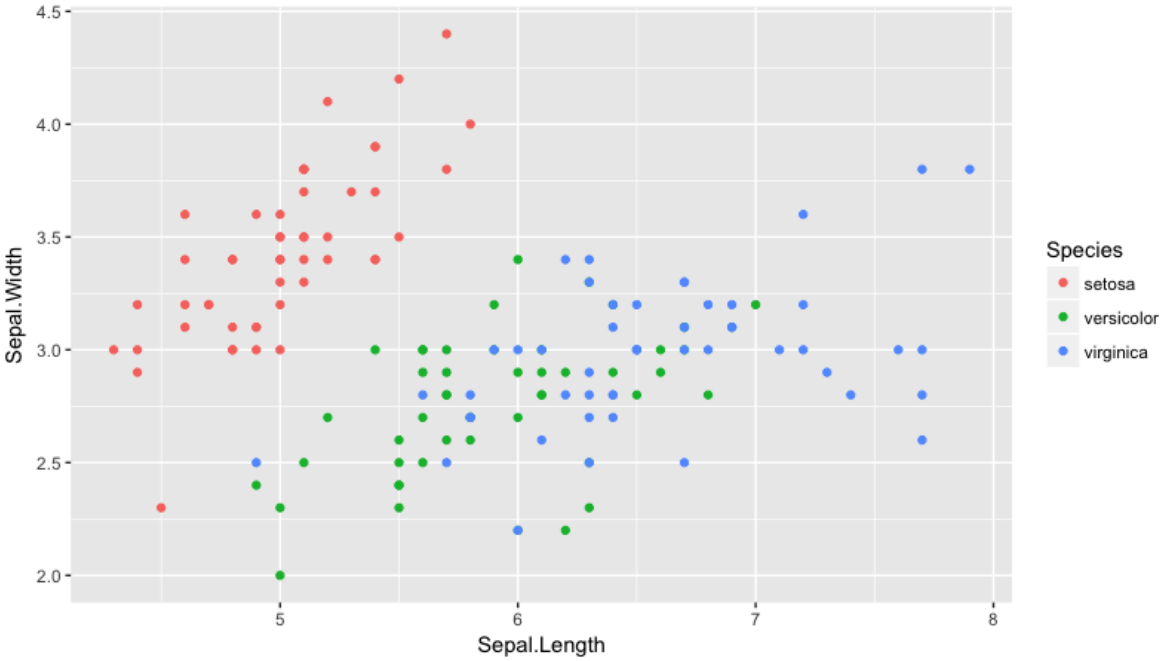
```
p1 <- ggplot(data = iris,
             mapping = aes(x = Sepal.Length,
                           y = Sepal.Width,
                           colour = Species),
                       stat = "identity",
                       position = "identity") +
             geom_point()
print(p1) # render the graphic
```

In the above code, a new object (`p1`) is created by passing the name of the dataframe in the call to `ggplot()`.[39] This command creates the list object that was previously described. The `mapping` lines correspond to defining the scales and properties of the data to be displayed (related to the DATA, TRANS, and ELEMENT statements in GPL). In this case, sepal length is assigned to the x-axis, sepal width to the y-axis, and elements will be colored based on their species. The `stat` argument could be used to apply a statistical transformation to the data, and when this argument is set to the default value (`identity`), no transformation is applied.

The graphical representations of the data on the plot are not added until a `geom_()` function is called. For every geometric object, `ggplot2` provides a convenient wrapper around the layer

[39]The simplest `ggplot2` object could thus be created via `p1 <- ggplot2(data = iris)`.

Figure 3.2: A grouped scatterplot via `ggplot2`.

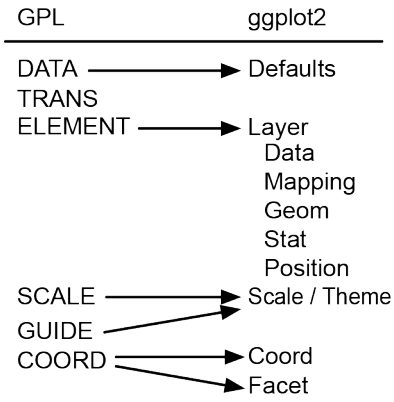


that provides sensible defaults for the mapping of the statistic and position aesthetics (in many cases, both are "identity"). The result is an object (in the above code, `p1`) that has a hierarchical structure and to which layers can be added incrementally.[40] When `print()` is called on that object, Figure 3.2 is produced. As expected, sepal length appears on the x-axis, sepal width on the y-axis, and the data points are colored by the levels of species.

This structure matches fairly cleanly with Wilkinson's framework, as shown in Figure 3.3. The construction of any plot involves the manipulation of the supplied data, and the choice of the geom, mappings, stats, position adjustments, coordinate system, and facets that are appropriate to it. In

[40]This is done by literally using the + operator, which is followed by the code that generates the objects that are to appear on the new layer.

Figure 3.3: The Grammar of Graphics and `ggplot2`.



some ways, the `ggplot2` approach is less restrictive than the original conceptualization of the grammar of graphics (e.g., we can apply aesthetics before coordinates), but is less flexible in terms of complete customization due to how some of the functions are coded.[41]

One notable difference between these two approaches is the inclusion of Algebra in the grammar of graphics framework (as both a specification operation and a step in Figure 3.1). This operation has to do with all the transformations that occur to a raw dataset to produce the components that will appear on a plot. Often, this operation will pertain to conducting various manipulations on the data, such as averaging across groups and calculating summary statistics. This step is an important one that is not dealt with from within `ggplot2` directly. Instead (and as demonstrated in the following section), these procedures take place in a script prior to calling `ggplot()`.[42]

[41] For instance, certain geoms are hard-coded to only work with particular data types.

[42] `ggplot2` is often associated with a set of other packages, typically authored by Hadley Wickham, and collectively known as the 'tidyverse' (Wickham, 2017). The focus of a few of these packages, specifically `dplyr` and `tidyr`, are to perform such manipulations.

MCSS results, especially in their unsummarized form, pertain to thousands of data points across a multitude of dimensions. As when working with tabular displays, the researcher is required to think about how to best summarize this data and the methods of presentation that match those goals. This summary can be as straightforward as incorporating graphical elements into a tabular display (semi-graphic tables and tableplots) or using various dimension reduction techniques to highlight differences across factors. The following sections will demonstrate applied examples of such techniques, using the above discussion to describe and contrast these approaches.

3.3 Semi-Graphical Tables

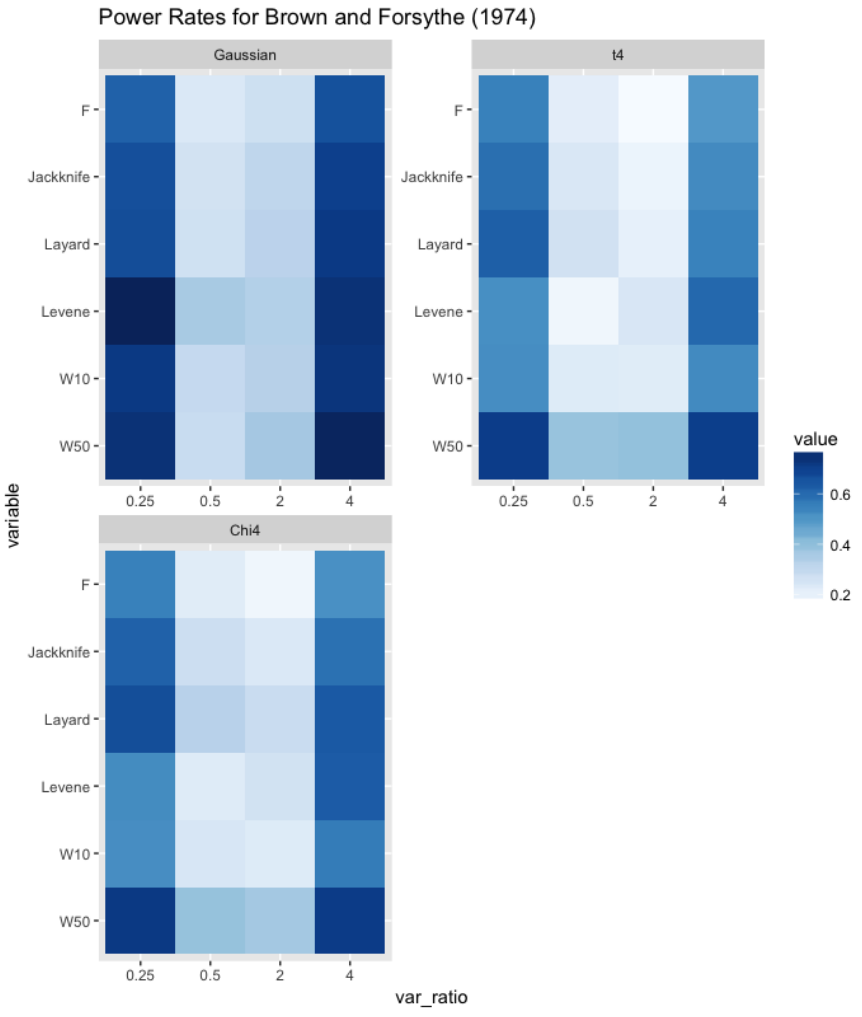
As briefly introduced at the end of Chapter 2, semi-graphic tables present numeric information that has been reinforced using graphical elements. In a sense, this presentation can be understood as a stylized representation of the raw table. This idea is intimately related to a heat map – a graphical display where a matrix of numbers is represented through colors that indicate frequency (Wilkinson & Friendly, 2009). However, instead of frequency, semi-graphic displays are shaded based on magnitude or otherwise visually enhanced. For example, such a display can be used to show the raw values of interest (e.g., the empirical detection rates, or EDRs, in the motivating example), with numeric values overlaid on the shaded background. The shading allows for easy visual comparison across cells based on a continuous range of values, and the degree of intensity can be manipulated for meaningful differences to be highlighted. Further, for designs with three or more dimensions, this approach can utilize marginal tables or facets to examine various relationships.

This style of graphic can be produced using `ggplot2` and is shown in Figure 3.4. To do so, the researcher must begin by converting the dataframe to long format[43], pass the outcome measure values to the `fill` parameter, call `geom_tile()`, and apply a custom scale. As can be seen in the following code, we utilize the grammar of graphics pipeline by loading and preparing the relevant data, passing it to `ggplot2`, applying the appropriate aesthetics (in the `aes()` call), specifying the visualizing method and scale type, and finally, applying custom aesthetics.

```
# Load libraries
library(SimDisplay)
library(ggplot2)
library(RColorBrewer)
library(reshape2)

# Generate a custom palette
myPalette <- colorRampPalette(brewer.pal(9, 'Blues'))

# Convert data to long form:
data(Brown1974)
dat <- subset(Brown1974, var_ratio != 1)
dat <- simplifyDf(dat) # Remove meta variables
dat <- aggregate(. ~ distribution + var_ratio,
                 data = dat, mean) # Average over sample size
dat <- dat[,-3] # Remove the redundant sample size column
dat <- melt(dat) # Convert to long form
levels(dat$variable) <- rev(sort(levels(dat$variable)))

# Generate and plot ggplot2 object:
pdat <- ggplot(dat,
               aes(x = var_ratio,
                   y = variable,
                   fill = value)) +
  facet_wrap(~ distribution,
             nrow = 2, scales = "free") +
  geom_tile() +
  scale_fill_gradientn(colours = myPalette(100)) +
  coord_equal() +
  ggtitle('Power Rates for Brown and Forsythe (1974)')
pdat
```

[43] As mentioned in Chapter 1, MCSS results are typically structured in a wide format where each row in the dataset pertains to a unique combination of levels of the factor variables and has a column for each outcome measure. A "long format" version of such a dataset would have the results from all of the outcome measures on their own rows within the same column (typically through the creation of a column "Variable" that contains the various outcome measures and a column "Value" that has the result for that cell).

Figure 3.4: The construction of a semi-graphic table using `ggplot2`.



There are a few complex statements in that block that deserve attention. First, the formula given in the call to `aggregate()` refers to `. ~ distribution + var_ratio`, and to aggregate the values using `mean()`. This statement requests averages for the levels of all numeric variables (designated by using the period on the left-hand side of the formula) that are not indicated on the right-hand side of the formula. Since `sample_size` is not included, it is the variable that will be averaged over. As such, one row of the resulting dataframe would pertain to the averages across the sample size conditions for the Gaussian distribution at the 1:4 variance ratio for each of the outcome variables.

The resulting dataframe is still in wide form and so `melt()` is then called. This function converts a dataframe from wide to long format through the creation of two new variables: `variable` (which is a factor that corresponds to the six outcome variables in this study) and `value` (the EDRs for each).

Again, the first argument passed to `ggplot()` is the dataframe object, `dat`. This plot requires aesthetics for the x-axis, the y-axis, and fill. In this case, we will place the variance ratio variable on the x-axis, the six statistical methods on the y-axis, and shade the cells by their magnitude. The call to `facet_wrap()` allows for the separation of plots by the levels of a grouping variable. In this case, it makes sense to display the results from the three distributions in separate plots (on either one or two rows) and the `scales` argument is used to place appropriate labels on the x-axis of each plot. Further, `coord_equal()` ensures that each cell is set to equal height and width. This plot can be enhanced by superimposing the actual EDRs onto the cells, as demonstrated in the

following code and shown in Figure 3.5:

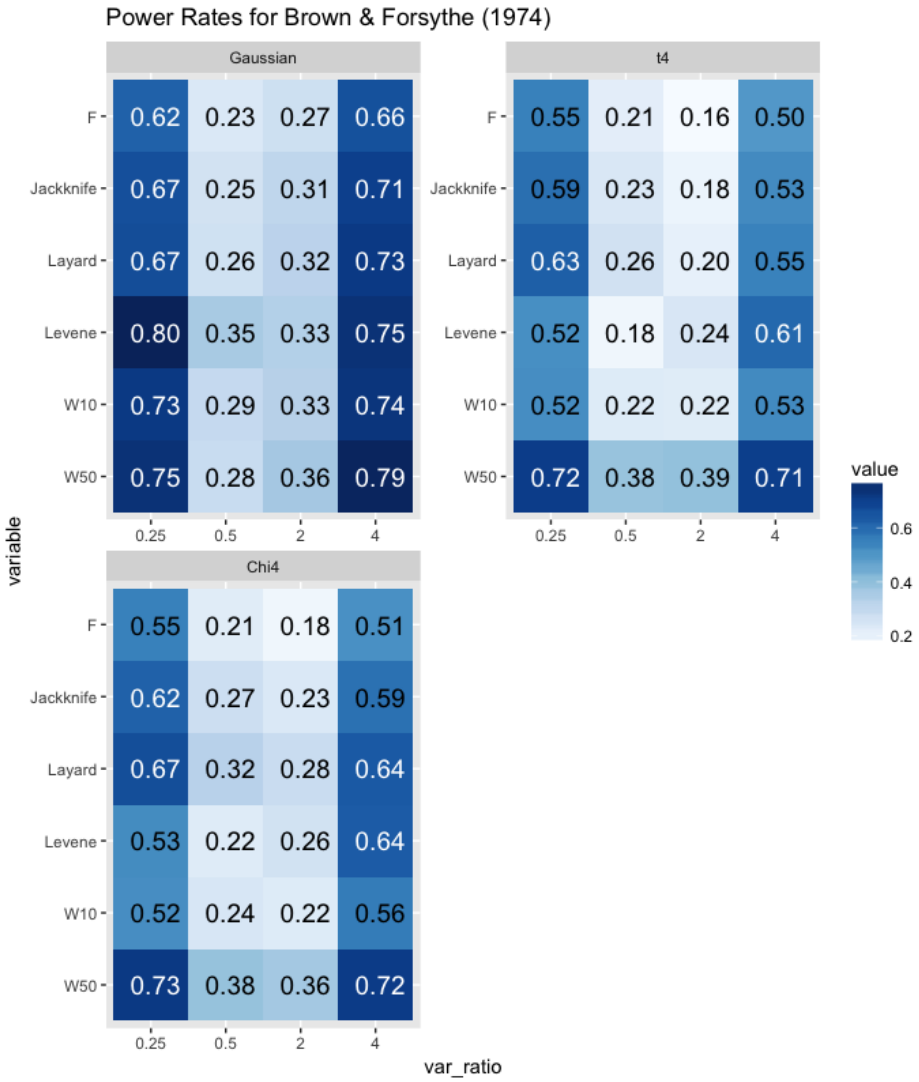
```
library(dplyr)
dat$rval <- roundSim(dat$value, 2) # Lucid labelling
pdat <- ggplot(dat,
               aes(x = var_ratio,
                   y = variable,
                   fill = value)) +
  facet_wrap(~ distribution,
             nrow = 2, scales = "free") +
  geom_tile() +
  scale_fill_gradientn(colours = myPalette(100)) +
  coord_equal() +
  ggtitle("Power Rates for Brown & Forsythe (1974)") +
  geom_text(data = dat %>% group_by(distribution) %>% filter(value < .6),
            aes(label = roundSim(value,  digits = 2)),
            hjust = 'right', nudge_x = .4, size = 5, color = 'black') +
  geom_text(data = dat %>% group_by(distribution) %>% filter(value >= .6),
            aes(label = roundSim(value,  digits = 2)),
            hjust = 'right', nudge_x = .4, size = 5, color = 'white')
pdat
```

One requirement for this type of display is for the text to remain legible. If a cell is shaded too darkly, typical black typefaces will be hard to decipher. In the above code, `geom_text()` was called twice to properly label the cells (using black characters for values less than .6, and white characters for values greater than or equal to .6). Further, care must be taken when converting the data to long form, especially if there are many factor variables in the design. In the above examples, sample size was averaged across cells and a call to `levels()` was required to reverse the ordering of the six outcome measures to have them appear in the expected ordering.

To simplify the production of such displays, `SimDisplay` includes a function called `tableShade()`. This function takes a dataframe object and produces a `ggplot2` graphic. It has the following arguments:

- `dat`: A `data.frame` object of `class('SimDesign')`

Figure 3.5: Annotated semi-graphic table using `ggplot2`.



- `table_vars`: A character vector pertaining to the variables that will be presented on the display. The first element specifies the faceting variable and the second the variable that will appear on the x-axis

- `method`: A character vector of length 1 pertaining to the method to be used to summarize results (the default is to use the mean)

- `ndigit`: An integer indicating the number of decimal places to use in the labelling (defaults to 2)

- `colswitch`: A numeric value indicating where to switch from black text to white (defaults to 0.6)

- `numrow`: An integer value indicating the number of rows to use in the facet grid (defaults to 2)

- `colour`: A character value indicating the colour palette to use (this value takes any valid string typically passed to `brewer.pal()` from the `RColorBrewer` package. See the help file[44] for that function for a list of all possible palettes; the default for this function is '`Blues`').

- `main_title`: A character value for the main title of the plot

- `xlab`: A character value for the x-axis label

[44] After loading `RColorBrewer`, run `?brewer.pal` to access the help file.

- `ylab`: A character value for the y-axis label

The subsequent code demonstrates the default use of `tableShade()` and produces Figure 3.6:

```
data(Brown1974)
dat <- subset(Brown1974, var_ratio != 1)
tableShade(dat, table_vars = c("distribution", "var_ratio"))
```

One problem in the previous figure is that the titles and axes are not appropriately labelled, as by default they use the dataframe's variable names rather than something more useful.[45] To address this limitation and to further highlight some of the other features of the function, Figure 3.7 shows the same plot using the 'Reds' palette and with better labelling. Also, to demonstrate the `numrow` argument, it has been set to one to place the distributions beside each other.

```
tableShade(dat, table_vars = c("distribution", "var_ratio"),
           main_title = "Power Rates by Variance Ratio and Distribution",
           xlab = "Variance Ratio", ylab = "Outcome Variable",
           numrow = 1, ndigit = 1, colswitch = .7, colours = "Reds")
```

Similarly, for the Hallgren replication, a shaded table plot can be produced to see the relationship between sample size and magnitude of the *c'* coefficients (averaging over *a* and *b*) on the EDRs for the two models. See Figure 3.8 for the result. In this display, it is apparent that both the XMY and XYM model specifications generally result in similar rejection rates.

```
data("Hallgren2013")
tableShade(Hallgren2013, table_vars = c("N", "cp"),
           main_title = "Power Rates by sample size and c' coefficient",
           xlab = "c' Coefficient", ylab = "EDRs",
           numrow = 1, ndigit = 2, colswitch = .5)
```

[45] This behavior could be also fixed by changing the names in the dataframe itself prior to running `tableShade()`.

Figure 3.6: Annotated shaded table generated using `tableShade()` defaults.

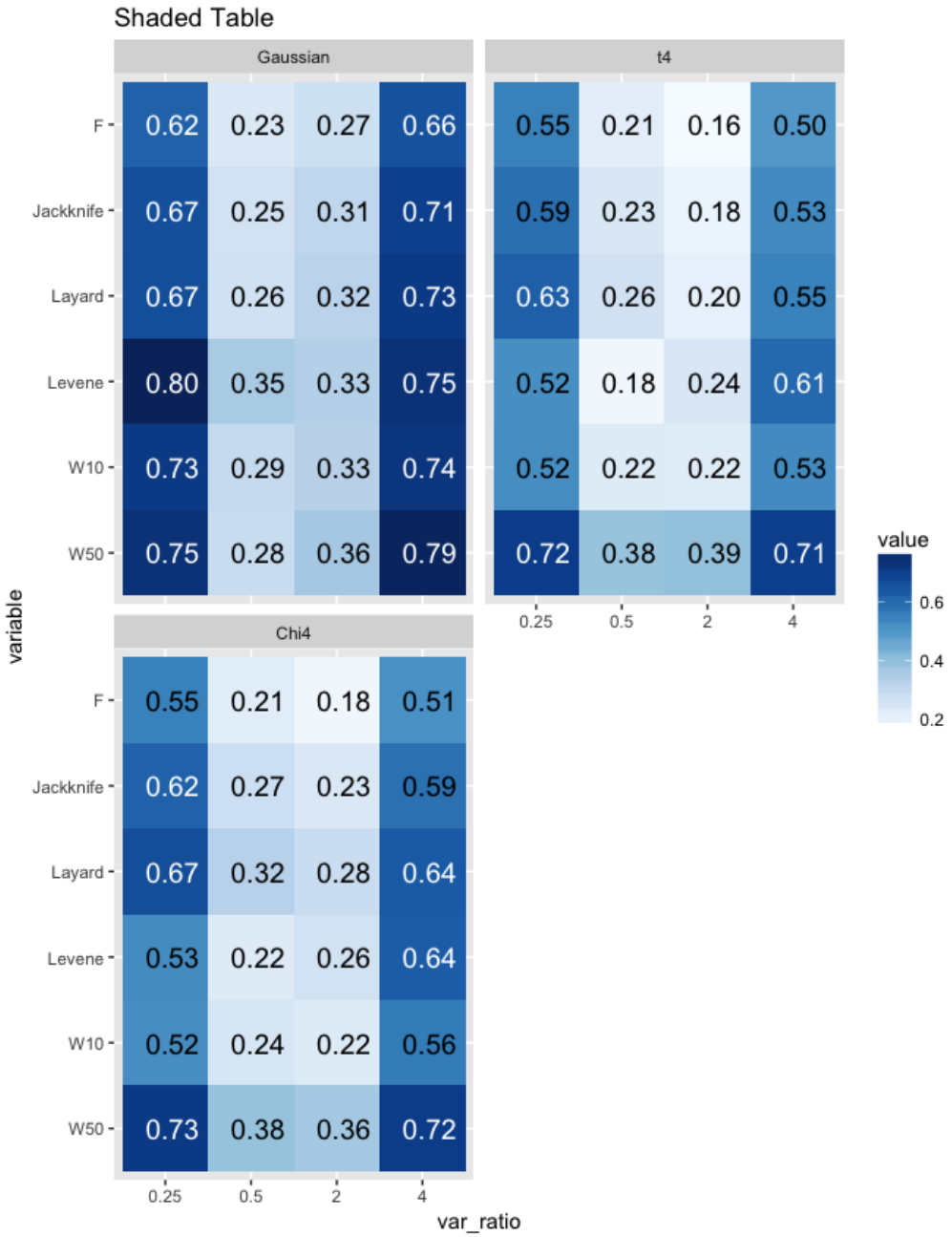


Figure 3.7: Annotated shaded table using `SimDisplay` with custom arguments.

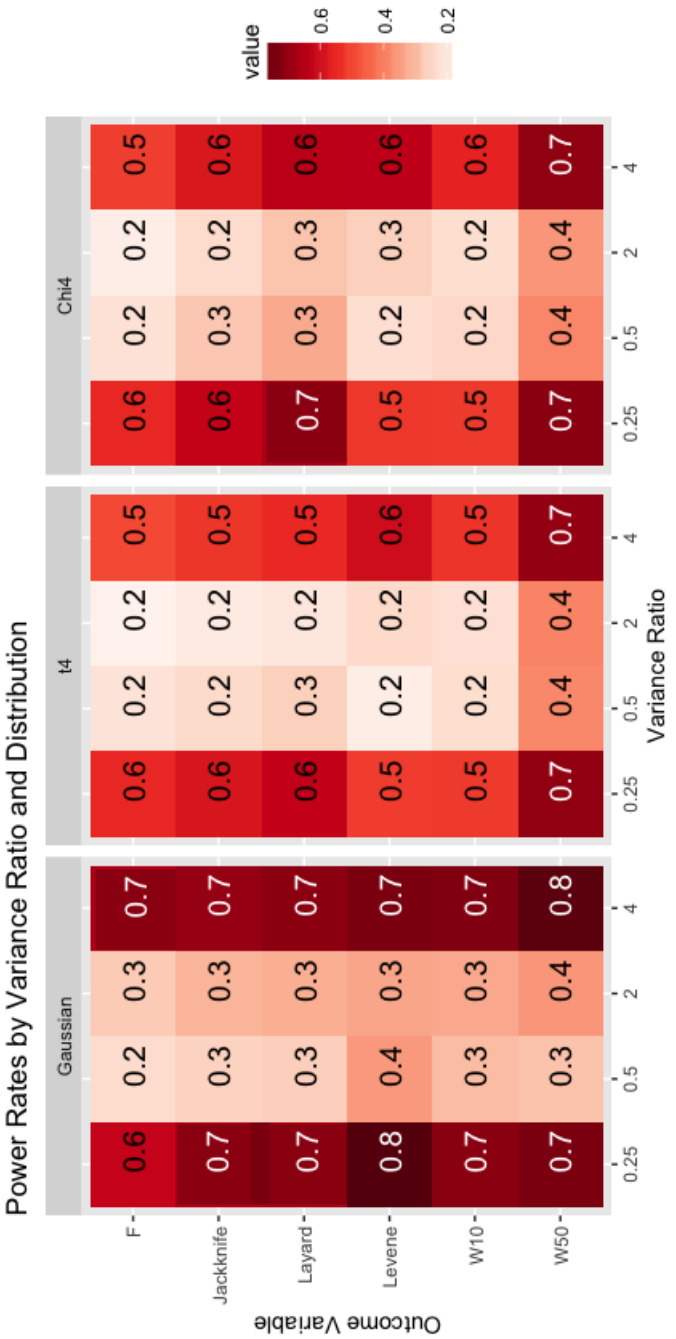
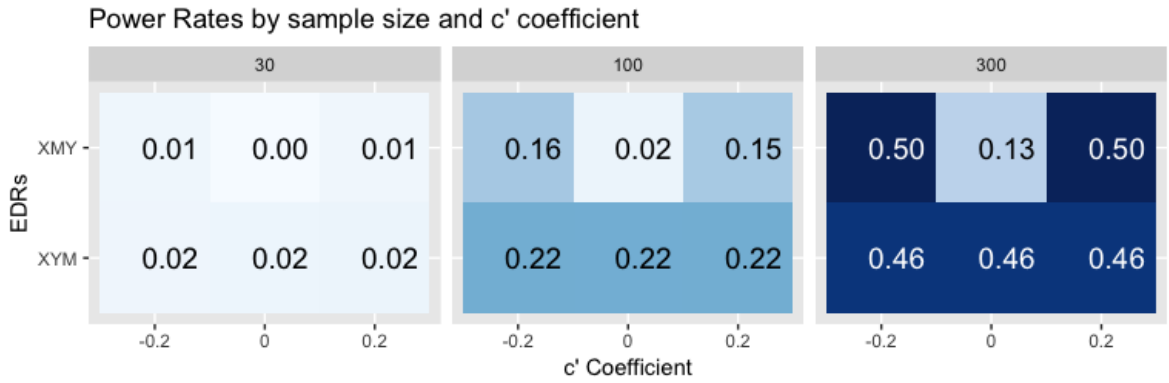


Figure 3.8: Annotated shaded table for Hallgren (2013).



3.4 Tableplots

Considering how large these tables can be, it is useful to think of alternative graphical approaches which make it easier to see what it happening at a glance. One alternative is to use the `tableplot` framework (Kwan et al., 2009), which presents a semi-graphic display that incorporates both numbers and *symbols* (with sizes, colors, shapes providing additional information) to summarize complex data. These visualizations were originally intended to present the results from exploratory and confirmatory factor analyses, but they can be applied to the results from other statistical methods and data types.

In these visualizations, each cell of a table is represented by a symbol proportionate to the value in the respective cell. Negative values are typically displayed in red, while positive values are shown in blue. The size of each shape is scaled in proportion to the maximum possible value. By building the display in this fashion, overall pattern recognition is facilitated. The following code

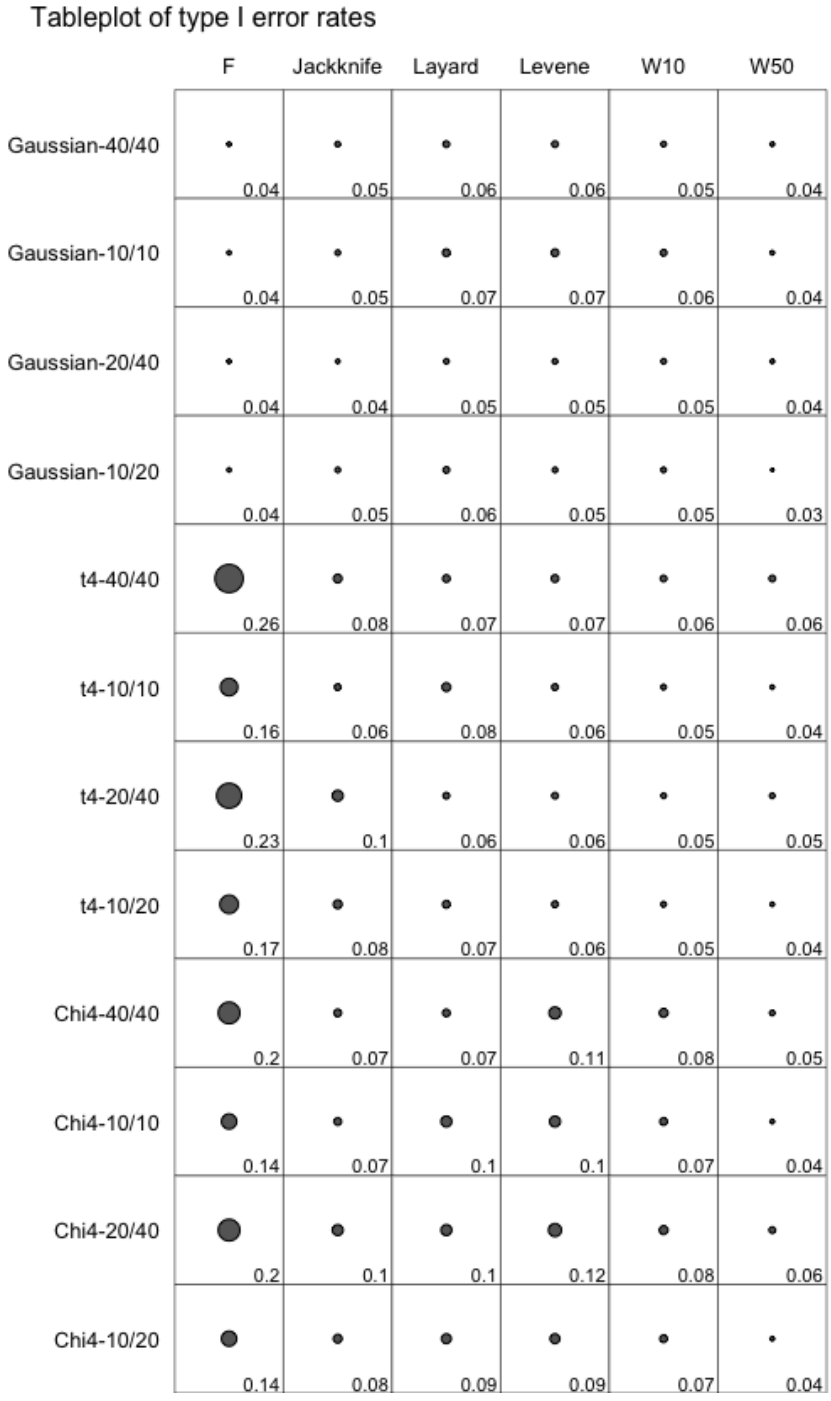
produces a tableplot for the Type I error rates for the motivating example, as shown in Figure 3.9.

```
# install.packages("tableplot") # run if package is not installed
library(tableplot)
data(Brown1974)
TypeI <- subset(Brown1974, var_ratio == 1)

# Simplify the dataframe and provide meaningful rownames:
TypeI <- simplifyDf(TypeI)
tdat <- as.matrix(round(TypeI[,4:9], 2))
colnames(tdat) <- colnames(TypeI)[4:9]
rownames(tdat) <- with(TypeI, paste(distribution, sample_size, sep = "-"))

# Assign the tableplot aesthetics:
specs <- make.specs(
  shape = c(0),
  cell.fill=c('grey40'),
  back.fill="white",
  scale.max=1,
  label=1)

# Generate the plot:
tableplot(tdat, cell.specs = specs,
          left.space = max(nchar(rownames(tdat)))*2,
          title = 'Tableplot of Type I error rates')
```

Two unwieldy aspects of this process are using `make.specs()` to construct the graphical elements for `tableplot()` and to ensure that the row names of the dataframe are handled correctly. As shown above, addressing these issues requires multiple lines of code. Accordingly, a convenience function is included in `SimDisplay` that makes the process substantially more straightforward. For instance, a basic tableplot can be produced in one line: `simTableplot(TypeI)`. For the Type I error rate display, the variance ratio variable is irrelevant (as it only has one factor level). A tableplot without the redundant variance ratio levels indicated and with a custom title applied can be produced as follows:

```
simTableplot(TypeI,
             design_vars = c("distribution", "sample_size"),
             main_title = "Type I error rates by distribution and sample size.")
```

Figure 3.9: Tableplot for the Brown and Forsythe Type I error rates.



In this kind of plot, the variability across conditions and outcome variables can be directly compared. However, with many levels of the design factors, tableplots can become quite long. In such cases, methods for simplifying or condensing MCSS output are useful. The next section outlines such an approach.

3.5 Effect Ordering and Visualizing ANOVA Designs

As discussed in Chapter 2, conducting ANOVAs on MCSS results provide a method for summarizing or distinguishing the effects of the factors (and, potentially, the interactions between them). This enables more succinct methods of display, rather than requiring a large number of raw cell values. In this approach, the design conditions are used as fixed factors and the EDRs are treated as the outcomes. Boxplots, introduced by Tukey (1977), provide important summary information of univariate densities. In these displays, the median is depicted by a bold horizontal line, and the central box is bounded by the first and third sample quartiles, such that the box contains exactly 50% of the data. The "whiskers" above and below the central box extend to the farthest observed values that are within 1.5 times the interquartile range beyond the quartiles and open circles or asterisks are used to depict outliers.

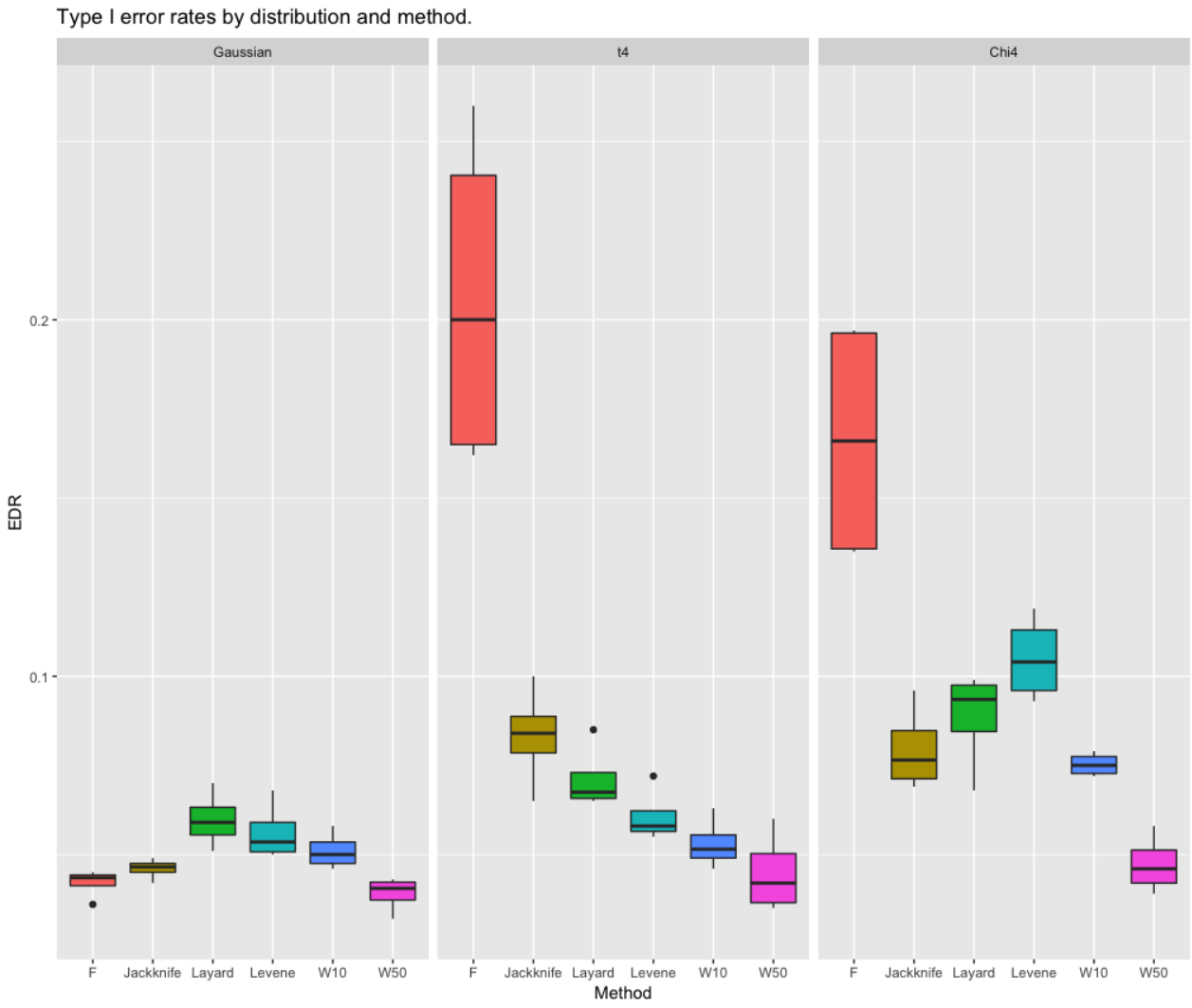
These displays can effectively capture and display MCSS information, especially when paired with exploratory ANOVA analyses. For example, the following code produces Figure 3.10, which shows a set of box plots for the Type I error EDR results for each of the methods for detecting heterogeneity of variance (on the x-axis), facetted by the generating distribution. This display em-

phasizes the high Type I error rates of all procedures but the W50 when the generating distribution is non-Gaussian:

```
# Load libraries
library(SimDisplay)
library(reshape2)
library(ggplot2)

# Prepare data
data(Brown1974)
dat <- subset(Brown1974, var_ratio == 1)
dat <- simplifyDf(Brown1974) # Remove meta variables
dat <- dat[,-c(2,3)] # Remove the redundant columns
dat <- melt(dat) # Convert to long form

# Generate plot
library(ggplot2)
ggplot(dat, aes(variable, value, fill = variable)) +
  geom_boxplot() + facet_wrap(~ distribution) +
  ggtitle("Type I error rates by distribution and method.") +
  theme(legend.position='none') + ylab("EDR") + xlab("Method")
```

Additionally, these displays can be effect ordered (Wainer, 1992), which is a method for sorting data according to some criterion (Friendly & Kwan, 2003). The goal is to reorder the data by the effects that are to be observed and which make notable trends more apparent. As Friendly and Kwan state, reordering has two repercussions: "globally, a more coherent pattern appears, making it easier to spot exceptions; locally, effect-ordering brings similar items together, making them easier to compare" (p. 514). For a MCSS ANOVA design, one approach would be to order the data by the magnitude of the outcome variable means. The following code produces a comparison plot (Figure 3.11) for the Gaussian distribution. In the first plot, the methods are unordered; in the second, they are sorted by the size of their mean. When presented in the latter manner, it is easier to see the methods that are performing the best (with the lowest EDRs) and the worst (with the highest EDRs).

Figure 3.10: Boxplot for the Brown and Forsythe Type I error rates.



```
# Original ordering and plot for Gaussian distribution:
eo_dat1 <- subset(dat, distribution == "Gaussian")
p1 <- ggplot(eo_dat1, aes(variable, value, fill = variable)) +
  geom_boxplot() +
  ggtitle("Type I error rates by method for the Gaussian distribution.") +
  theme(legend.position='none') + ylab("EDR") + xlab("Method")

# Reorder variable by the mean for each design condition:
eo_dat2 <- eo_dat1
eo_dat2$variable <- with(eo_dat2,
                         reorder(x = variable, X = value,
                                 FUN = function(x) mean(x)))

# Plot with effect ordering:
p2 <- ggplot(eo_dat2, aes(variable, value, fill = variable)) +
  geom_boxplot() +
  ggtitle("Effect ordered Type I error rates by method for the Gaussian distribution.") +
  theme(legend.position='none') + ylab("EDR") + xlab("Method")
```

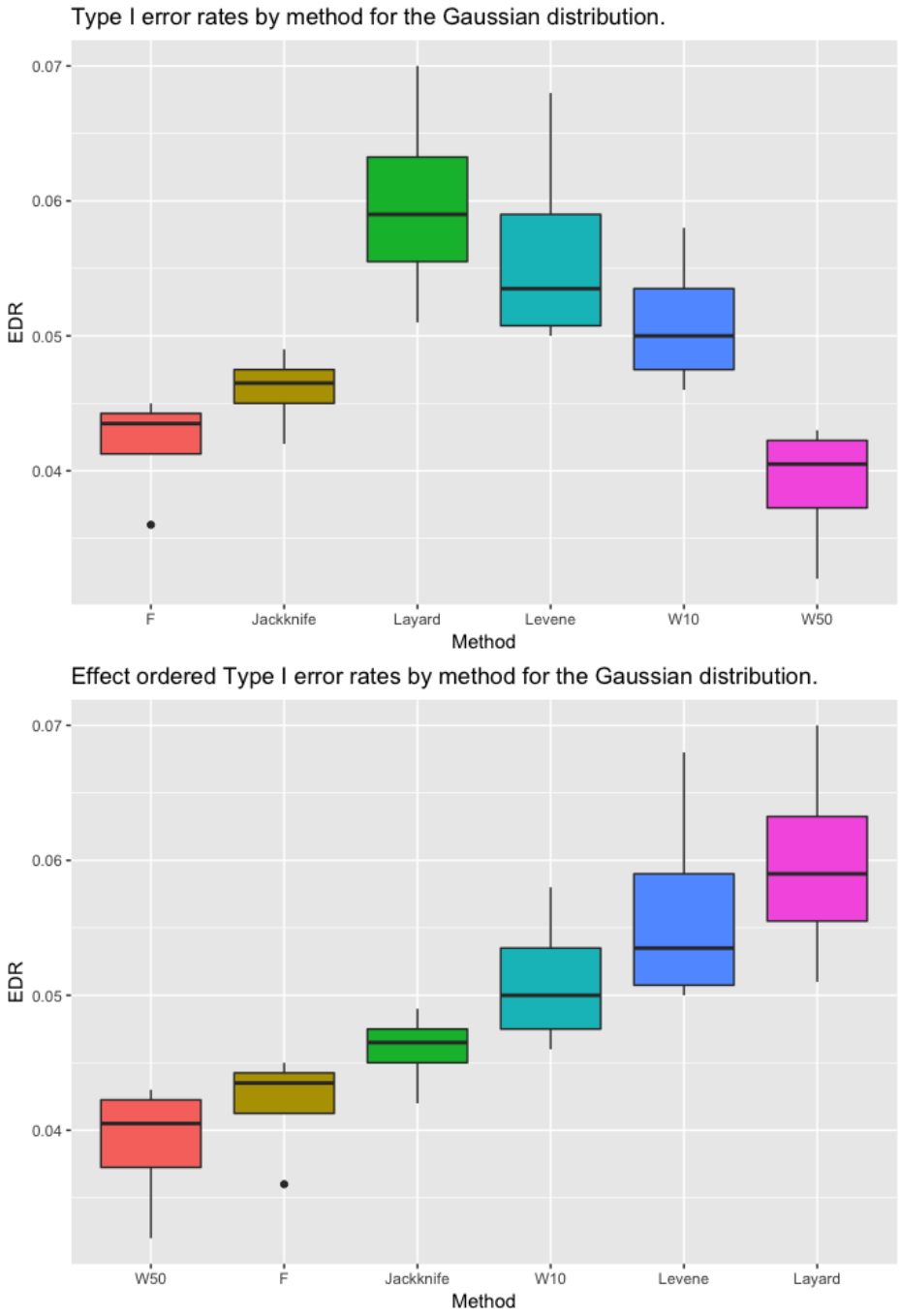
However, for designs with a multitude of experimental conditions, this approach can still be unwieldy. Another option for researchers wishing to apply an ANOVA methodology to their MCSS results is to produce effect plots (Fox, 2003; Fox & Hong, 2009).[46] These line plots connect a series of dot-and-whisker displays based on model coefficients. These plots use dots to represent the estimated effects (means) for one term in a model, averaging over all factors not included in that term, and whiskers convey their respective error bars (confidence intervals).

A core feature of these displays is the ability to simplify interactions between the variables, which appear as non-parallel lines.[47] Effect plots can be produced using the `effects` library (Fox, 2003) in R. The following code demonstrates their generation and use in regards to the Type I error

[46] As in traditional ANOVA-based analyses, each model could be checked for assumption violations. Specifically, one could look at the residual plots for each of the six models.

[47] Interactions can be built into such models by using * instead of + in the `formula()` call. However, due to the simplicity of this particular example, there are no residual degrees of freedom if the interaction is included and so the main effects only models are presented.

Figure 3.11: Unordered versus effect ordering for the Brown and Forsythe Type I error rates.



rates from the motivating example.[48] Figure 3.12 shows the effect plots for the *F* (top) and the W50 (bottom) procedures and visually emphasizes the relationship by distribution and sample size.[49]

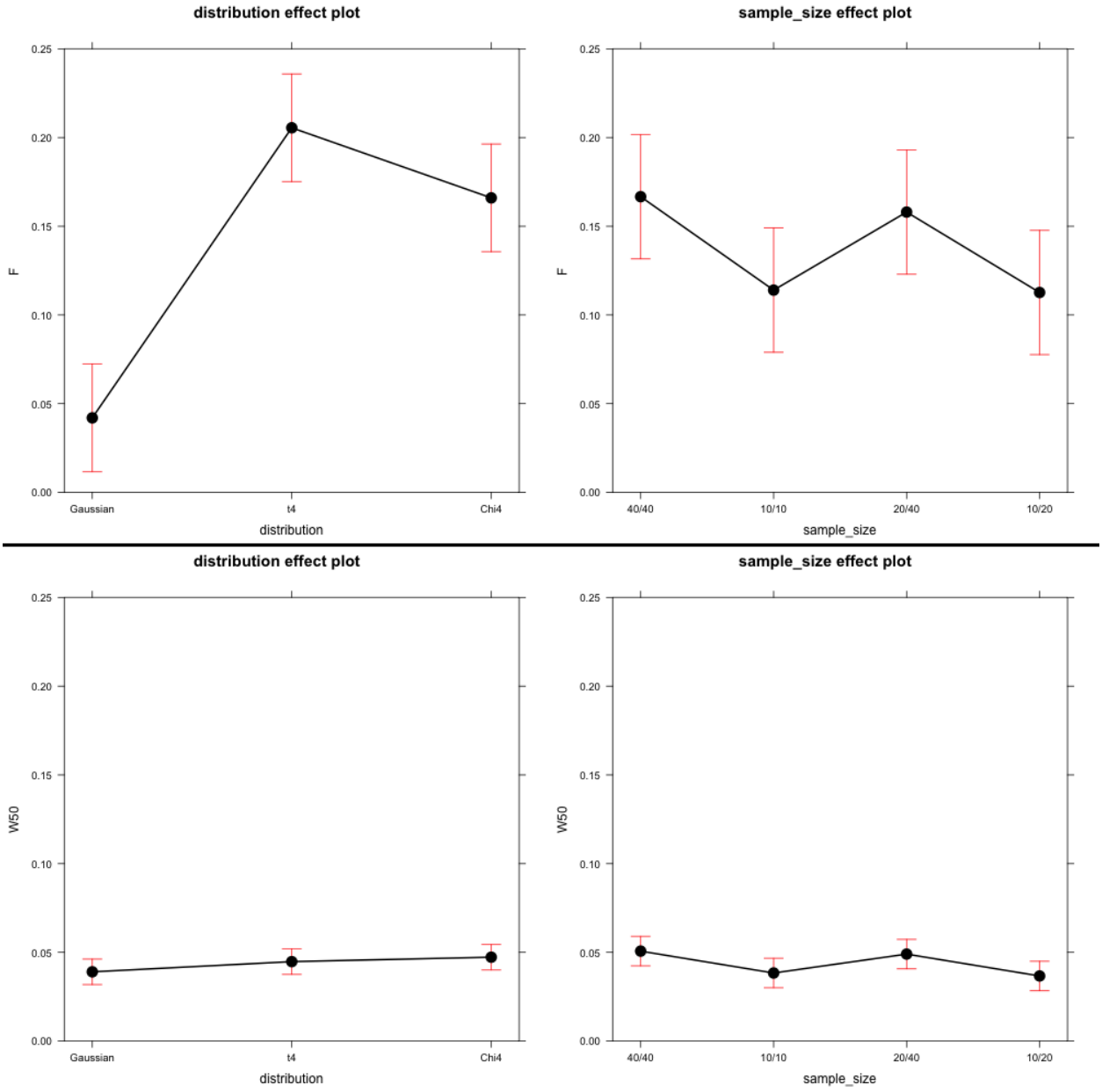
```
data(Brown1974)
dat <- simplifyDf(Brown1974) # Remove meta variables
dat <- subset(dat, var_ratio == 1) # Isolate Type I error rate conditions
dat <- dat[, -c(3)] # Remove the redundant columns

# Create input/output objects:
models <- list() # To store results
dvnames <- names(dat)[3:8] # Outcome variable names
ivnames <- names(dat)[1:2] # Simulation variable names

# Run an ANOVA for each outcome variable:
for (y in dvnames){
  form <- formula(paste0(y, "~",
                         paste0(ivnames, collapse = "+")))
  models[[y]] <- do.call("lm",
                         list(formula = as.formula(form),
                              data = as.name("dat")))
}

# Generate each effect plot and store it locally:
for (y in dvnames){
  png(paste("plot_", y, ".png", sep = ""), width = 1000, height = 500)
  plot(allEffects(models[[y]]), ylim = c(0,.25))
  dev.off()
}
```

Similarly, the following code produces effect plots for the EDRs for the XMY (Figure 3.13) and the misspecified XYM (Figure 3.14) models from the Hallgren dataset. These plots demonstrate the influence of sample size (for both models, the Sobel test is declared significant more frequently with larger sample sizes). Looking across the effect plots highlights a different pattern across the *a* and the *cp* coefficients – for the XMY model, the impact of *a* is consistent across all levels

[48] One technical note about the block is the use of `do.call()`. This function takes the formulas stored in the `form` object, and passes them to `lm()`, which is used to evaluate the linear models. If this call is not used, then if we inspect the individual models (e.g., via `models[[2]]` or `models[["Jackknife"]]`), then the formula in the model output would show the `paste0()` syntax rather than the actual model call.

[49] In the call to `plot()`, one can specify the argument `ci.style = "bands"` to request that the confidence intervals be represented using bands instead of whiskers. This style is usually meant for a quantitative x-axis variable, although can still be applied here.

Figure 3.12: Effect plot for the F-statistic and the W50 for the Type I error rates.



(behaving as *cp* does in the XYM model). Such plots provide efficient summaries of simulation results and can highlight potential areas of interest.

```
data("Hallgren2013")
dat <- simplifyDf(Hallgren2013)
models <- list()
dvnames <- names(dat)[5:6]
ivnames <- names(dat)[1:4]

for (y in dvnames){
  form <- formula(paste0(y, "~",
                         paste0(ivnames, collapse = "+")))
  models[[y]] <- do.call("glm",
                         list(formula = as.formula(form),
                              data = as.name("dat")))
}

for (y in dvnames){
  png(paste("Hallgren_power_", y, ".png", sep = ""), width = 1200, height = 750)
  plot(allEffects(models[[y]]), ylim = c(0,1))
  dev.off()
}
```

3.6 The Hypothesis-Error Plot Framework

MCSS results form a complex multi-dimensional table. This structure is additionally complicated when there are multiple outcome measures (such as in the Brown and Forsythe replication). Often this complexity leads researchers to treat each outcome separately and compare the influence of the design factors across model output or visualizations.

However, similar to the dimension reduction techniques discussed in Chapter 2 and the use of ANOVA and effect plots in Chapters 2 and 3, there exists a graphical technique which aims to reduce the impact of the *outcome* dimensionality of an analysis. The hypothesis-error (HE) plot framework presents a particular projection of the data that maximizes mean differences (Friendly,

Figure 3.13: Effect plot for the properly specified XMY model.

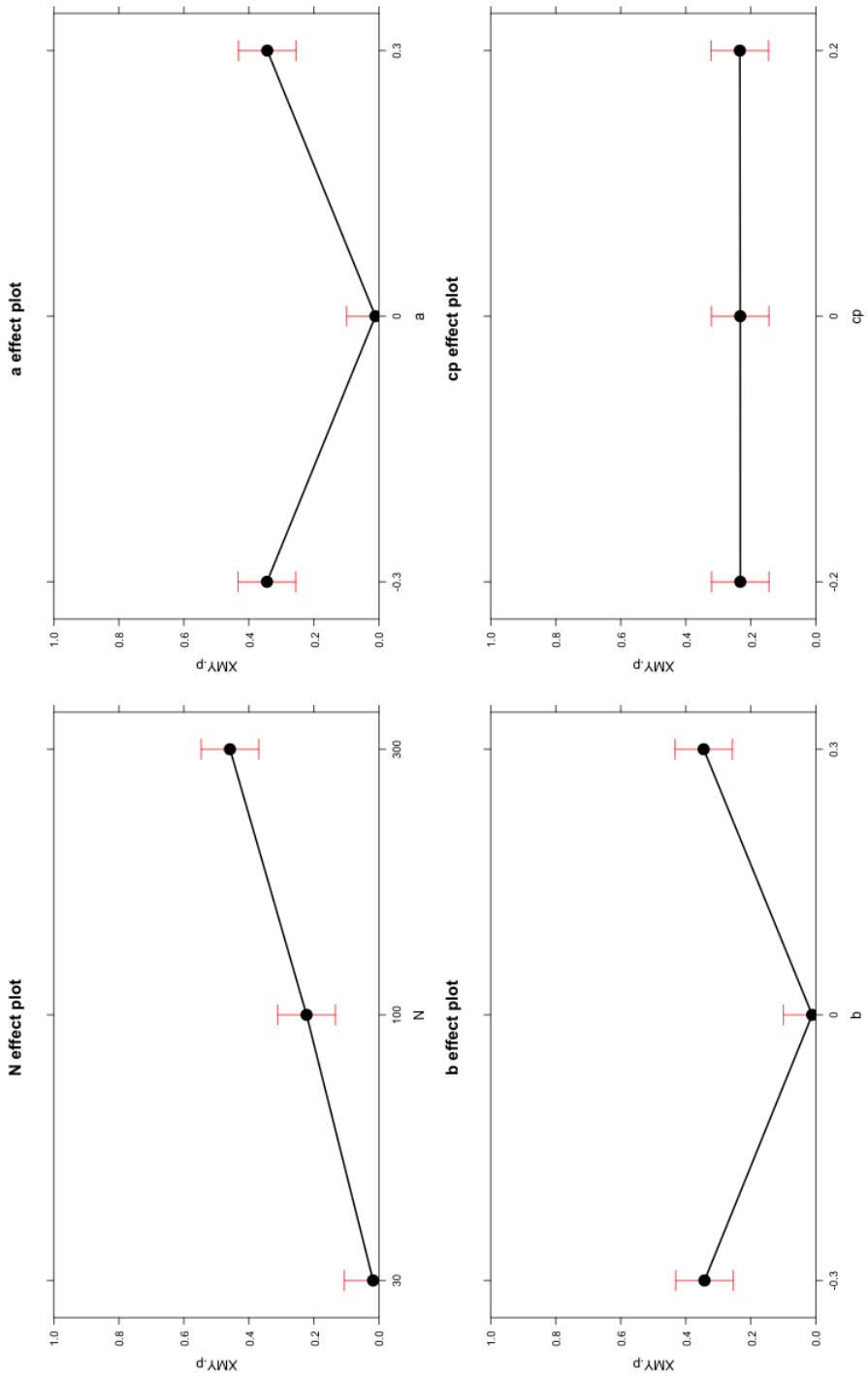
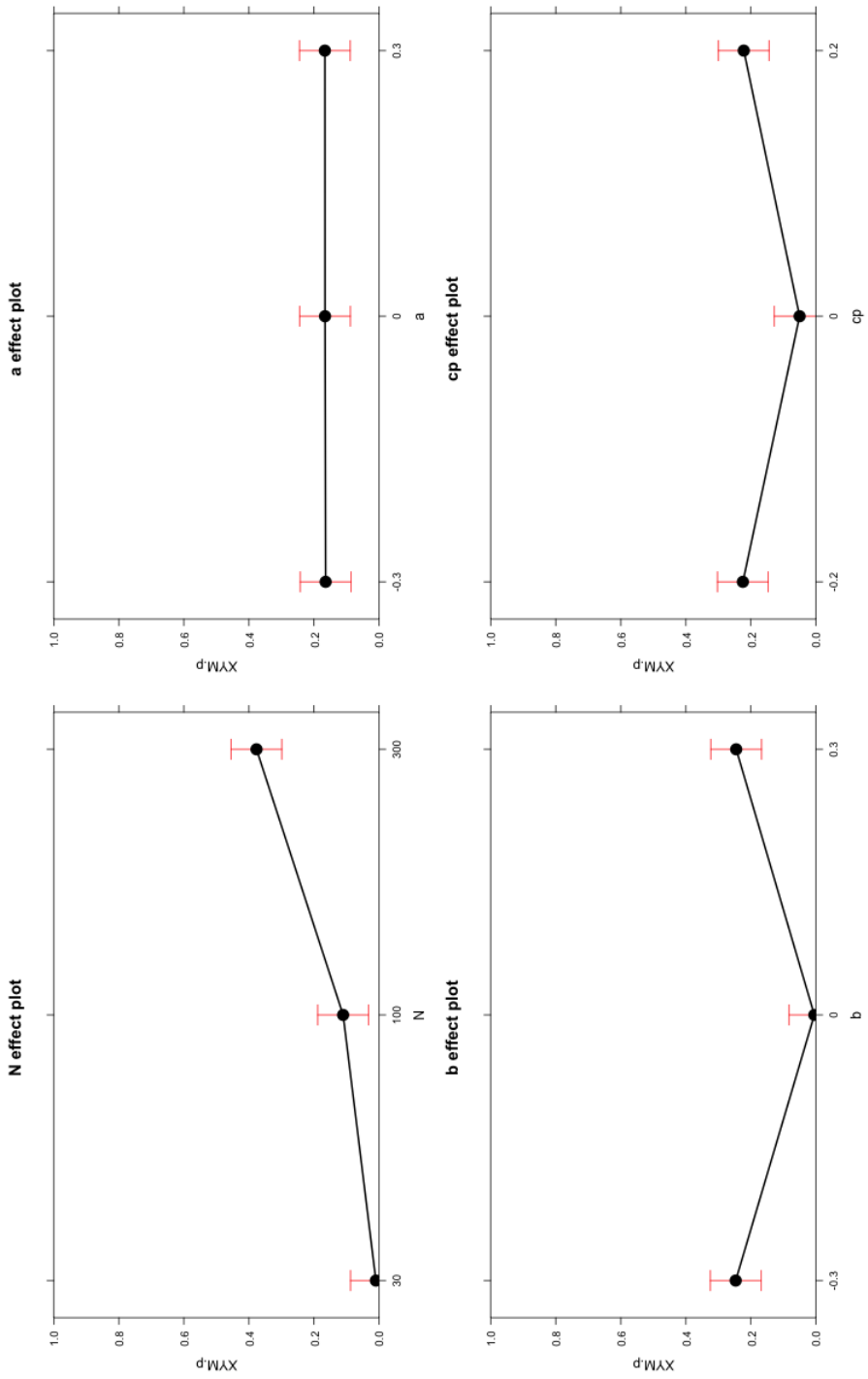


Figure 3.14: Effect plot for the misspecified XYM model.



2007; Friendly, Monette, & Fox, 2013; Friendly & Sigal, 2017) across an arbitrary number of outcome and design variables. While usually applied to more traditional multivariate datasets, this framework can be used to organize, summarize, and present the results from a complex MCSS table. As such, this approach looks at the impact of the design factors on two or more response variables simultaneously and is built on the generalization of the univariate ANOVA to the multivariate linear model (MLM).[50] HE displays typically use data ellipsoids to reduce the complexity of the data, and in the case where there are many outcome variables, the relationship between the predictor and outcome variables can be projected into a low-dimensional space using canonical discriminant analysis.

The core of this approach is summarized in the following points:

1. For any multivariate normal data, the graphical analog of the minimally sufficient statistics (their mean vector,$\boldsymbol{\mu}$, and covariance matrix,$\boldsymbol{\Sigma}$) is a data ellipsoid that has been centered at the overall mean ($\boldsymbol{\mu}$) and whose size and shape is determined by$\boldsymbol{\Sigma}$;

2. Any hypothesis test regarding these relationships can be visualized by superposing an $\boldsymbol{H}$ ellipsoid, which represents variation due to the model hypotheses, against an $\boldsymbol{E}$ ellipse, which represents error variation; and,

3. HE plot methods provide a visual test of significance[51] (based upon Roy's maximum root),

[50]See Friendly and Sigal (2017) for a review of the MLM and how it is applied in the HE plot framework, and Friendly and Sigal (2014)for a more detailed report on the various methods for visualizing MLM data.

[51]While significance tests are not typically a focus when evaluating MCSS results, they can be illuminating and draw attention to interesting findings (as demonstrated in Chapter 2).

which is indicated by the $\boldsymbol{H}$ ellipse protruding outside the $\boldsymbol{E}$ ellipse.[52]

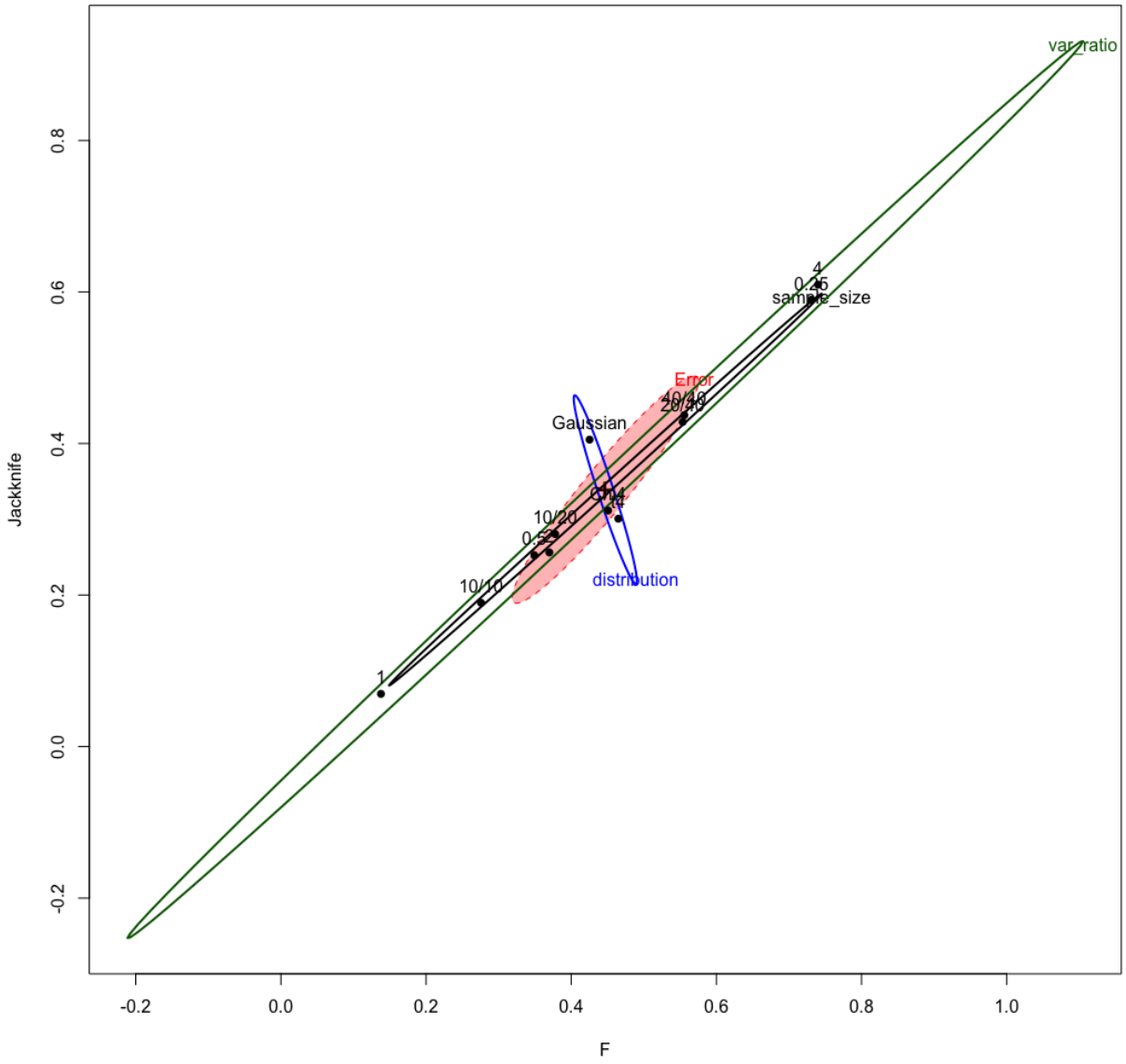
When applied to MCSS results, each design cell is an observation that contributes to the size and shape of the ellipses. These displays are generated using the `heplots` package (Friendly, 2010). The following code demonstrates the generation of an HE plot (Figure 3.15) for the Brown and Forsythe power rates:

```
library("SimDisplay")
library("heplots")

data("Brown1974")
dat <- simplifyDf(Brown1974) # Remove meta variables
dat <- subset(dat, var_ratio != 1) # Isolate power conditions

# Multivariate model:
mod.mlm <- lm(cbind(F, Jackknife, Layard, Levene, W10, W50) ~
                distribution + sample_size + var_ratio,
              data = Brown1974)

# Generate HE plot:
heplot(mod.mlm, variables = 1:2,
       fill=c(T,F)) # Apply shading to the error ellipse
```

In the above code, a multivariate model is produced by listing the outcome variable names via `cbind()` on the left-hand side of the model equation and the HE plot itself is produced by simply passing that model object to `heplot()`. We can interpret this plot as follows: for *F* and the Jackknife, the within-group residuals (E) are positively correlated (the shape and direction of the $\boldsymbol{E}$ ellipse): observations with higher means for *F* tend to also be high for the Jackknife. Further, the variation among the group means ($\boldsymbol{H}$) is in the same direction for the variance ratio and sample size, but the opposite direction for distribution. However, this plot ignores the four other outcome

[52] By default, the `heplot()` function uses significance scaling when constructing the ellipses. If one is more interested in the magnitude of an effect size measure, they can pass `size="effect"` to the function to override this default.

Figure 3.15: HE Plot for the Brown and Forsythe Power Rates.



variables. To see the relations for all variables together, one can generate a set of HE plots in a scatterplot matrix format by running `pairs(mod.mlm)`, although this format can be quite difficult to interpret with more than a few outcome variables.

A better approach would be to distill the relationships into one graphic. To accomplish this, canonical discriminant analysis (CDA) can be used. This plot is constructed using an eigenvalue decomposition that contrasts the size of $\boldsymbol{H}$ relative to $\boldsymbol{E}$.[53] A basic canonical plot shows each observation's position relative to the the canonical scores from the above deconstruction on the first two canonical dimensions. The decomposition is accomplished via the `candisc` package (Friendly & Fox, 2016) by passing the MLM object to `candisc()` and it is displayed by passing that object to `plot()`, as follows:

```
library("candisc") # for Canonical Discrimination Analysis

mod.can <- candisc(mod.mlm) # Run the CDA
plot(mod.can) # Basic CD plot
```

This code generates Figure 3.16, in which observations from each distribution are given separate symbols. The canonical HE plot is an extension of this display, where the observation level data is summarized via $\boldsymbol{H}$ and $\boldsymbol{E}$ ellipses. Figure 3.17 is generated by running `heplot(Brown.can, fill=TRUE, fill.alpha=.1)`.

The interpretation of this plot mirrors that of a regular HE plot: if the hypothesis ellipse extends beyond the error ellipse, then that dimension is significant. Vectors for each predictor are superimposed and show the relation between each and the two canonical dimensions. The relative lengths

[53] Again, for more detail refer to Friendly and Sigal (2017).

Figure 3.16: Basic CDA Plot for the Brown and Forsythe Power Rates.

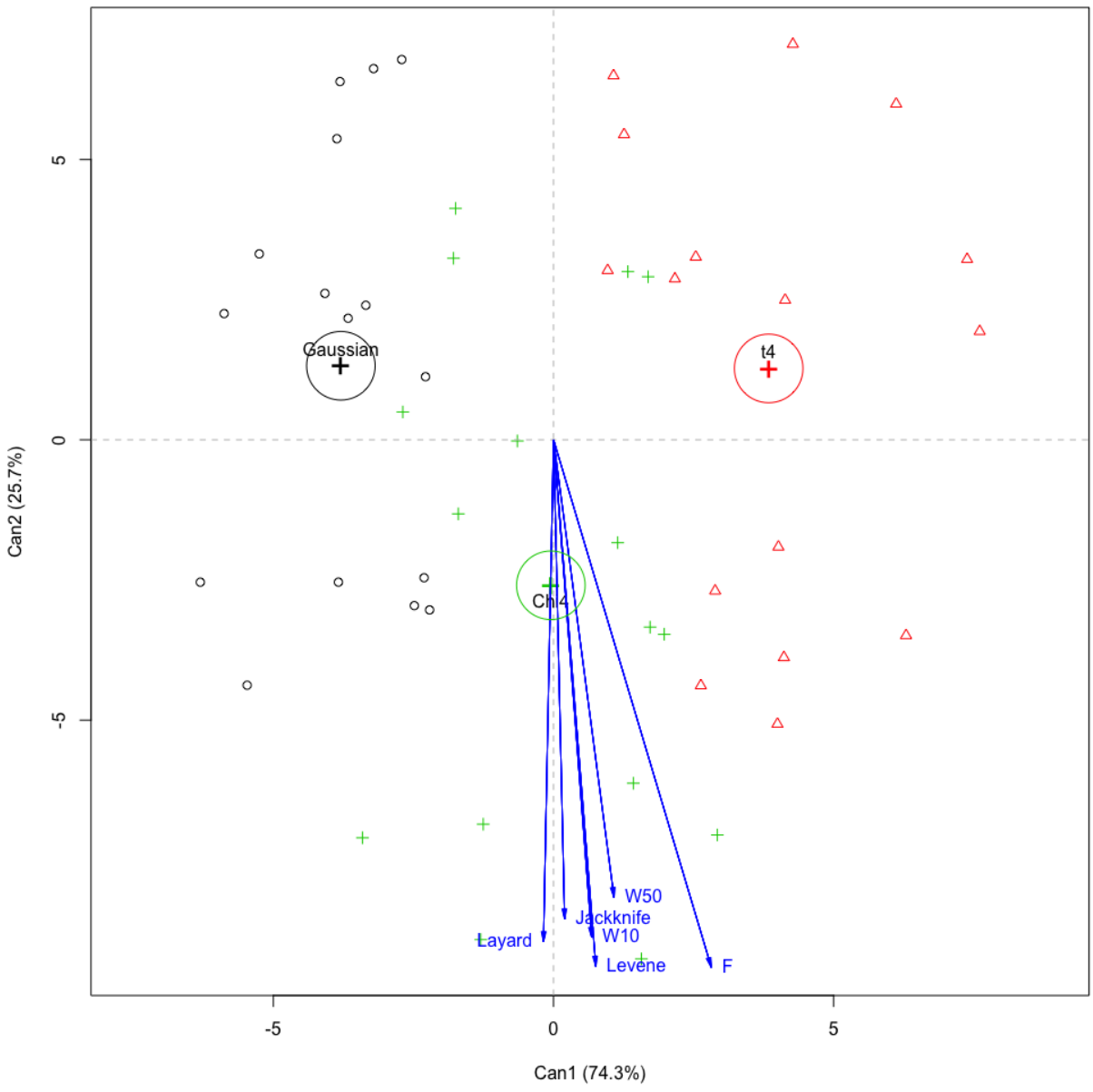
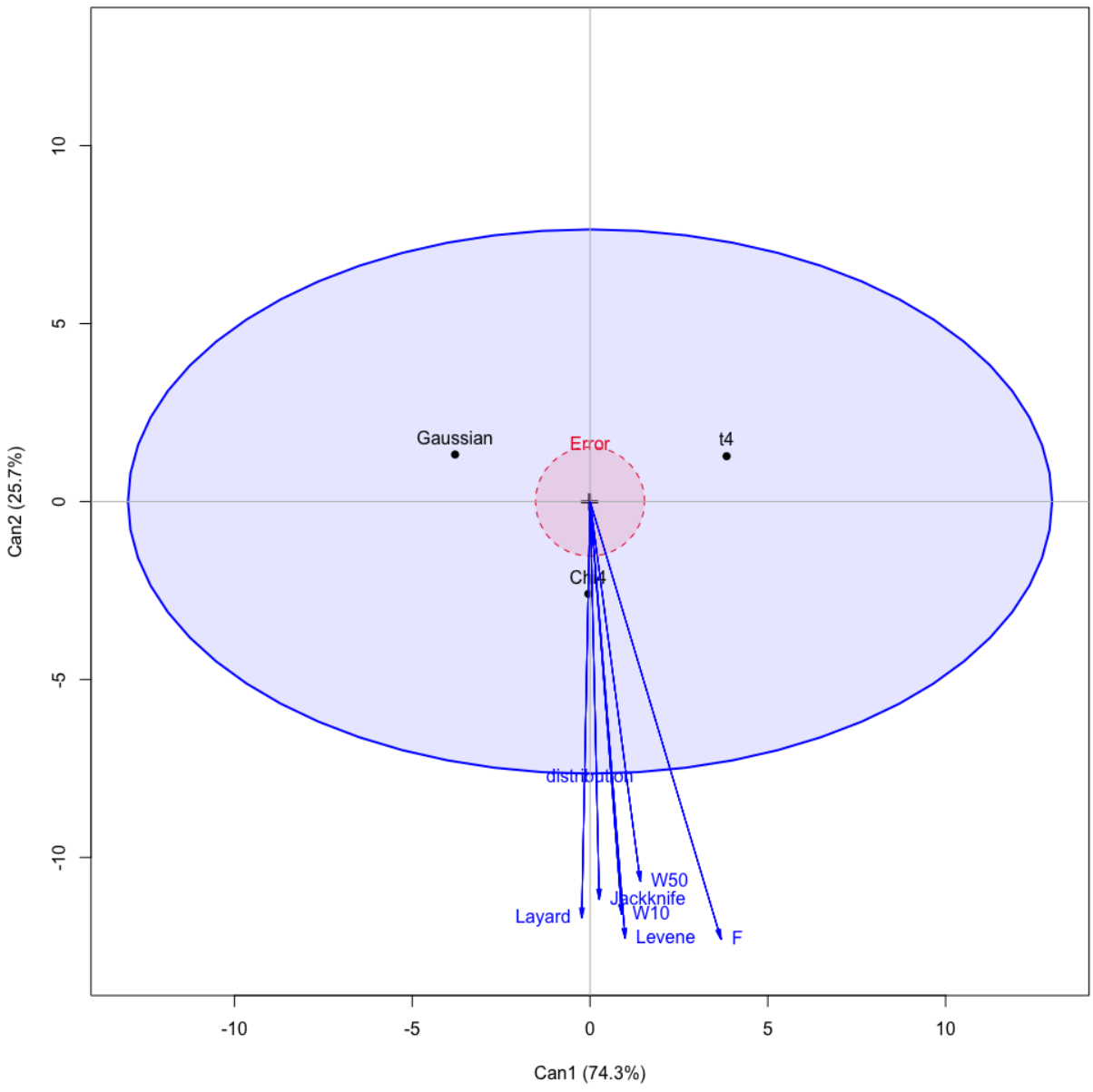


Figure 3.17: Canonical HE Plot for the Brown and Forsythe Power Rates.



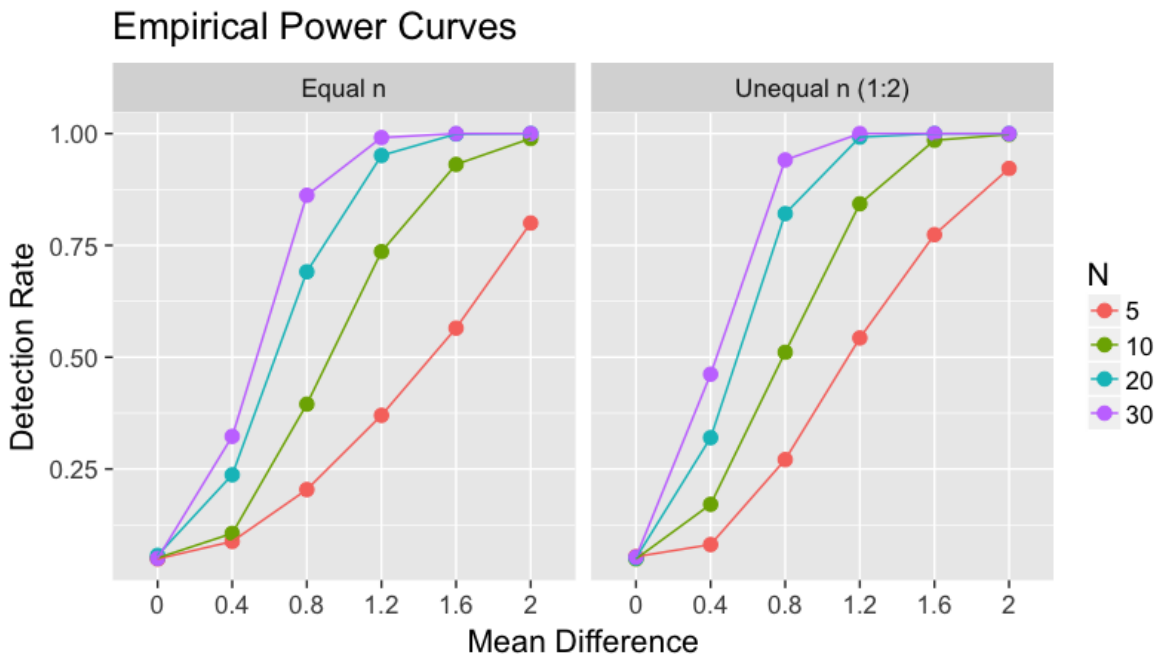
of the variable vectors are proportional to their contributions to group mean differences. As such, in canonical space, variation of the means for the groups is entirely two-dimensional. Dimension 1 (74.3% of the distribution effect) is largely attributable to the distribution effect.

3.7 Power Curves

While not reflected in the motivating examples, many MCSS are performed for assessing the a priori power of a proposed study. These MCSS are generally conducted to provide realistic estimates of pragmatic concerns, such as the minimum required sample size for a given level of power. In such situations, researchers are typically interested in varying population characteristics (e.g., effect size, population variances) and their relationship with sample size.

As a simple example, researchers might be curious about how the power of a study is associated to the magnitude of the population mean difference and the group sample size for a one-way ANOVA design. Such simulations are often straightforward to perform (see Appendix D), and their results can be summarized visually, which is very useful for grant applications. A power curve display is created by running a simulation in which various design factors are manipulated, and then constructing a line plot. On this graphic, power is shown on the y-axis and the design factors (such as sample size, effect size, and so on) are either indicated on the x-axis or via another aesthetic (such as shading, separate lines, etc.). The following code uses the data from Appendix D and creates power curves to highlight the influence of sample size and expected mean differences on the power of a one-way ANOVA design using `ggplot2`.

Figure 3.18: Empirically derived power plot for a one-way ANOVA design.




```
library(ggplot2)
results$ratio <- factor(results$ratio,
                        labels = c("Equal n", "Unequal n (1:2)"))

ggplot(results, aes(x = mean_diff,
                    y = p,
                    colour = sample_size,
                    group = sample_size)) +
  geom_point() +
  geom_line() +
  xlab('Mean Difference') +
  ylab('Detection Rate') +
  ggtitle('Empirical Power Curves') +
  scale_color_discrete('N') +
  facet_wrap(~ ratio, ncol = 2)
```

Figure 3.18 highlights the relationship between mean difference and sample size on the power rates for a one-way ANOVA design. In this plot it can be seen that larger sample sizes (both in terms of sample size per group, indicated by color, and across the design, indicated by the right

plot versus the left) have steeper empirically derived power curves. This skeleton can be expanded to assess the properties of any study or analytic approach through the modification of type of statistical analysis, the design factors under investigation, and the nominal Type I error rate (set at .05 in this example).

3.8 Conclusion

Graphical approaches are extremely useful for summarizing and presenting MCSS results. Faceted semi-shaded tables are particularly efficient, as they do not lose any of the precision compared with typical tables, but have the added benefit of heightened visual impact. Effect ordering can be useful for ensuring presentation is done in the most poignant manner.

Further, various methods exist that take multidimensional data and summarize it in an efficient manner. These methods can pertain to presenting the results from ANOVA summaries (e.g., effect plots and HE plots) or to low-dimensional approximations of the cell-level data (e.g., canonical HE plots). Through illustrating these approaches and providing convenience functions via `SimDisplay`, it is hoped that more researchers will use such graphics in the presentation of their MCSS findings.

4 Providing Power to the Reader

With advances in computing technology, the results from complex studies should not be relegated by the restrictions of the printed page. The work outlined in previous chapters all focused on methods for showing static slices of data, generally through particular projections of the multivariate space. However, "a single graph cannot always reveal everything about the data" (Wilkinson, 1993, p. 357). Each of those displays almost necessarily hides or marginalizes aspects of the design. This problem is exacerbated as more design factors or outcome variables are included in an analysis. Such issues can be circumvented by programming an interactive or dynamic display or applet to explore MCSS results.

In this chapter, the theory of and recent innovations in interactive displays will be showcased that allow the dynamic exploration of complex tables and figures. The reasoning behind such presentations will be discussed, alongside a primer on the terminology that is typically used. Further, convenience functions available in `SimDisplay` for generating such displays will be demonstrated. In brief, ideas for interaction range from allowing changes in display modality (e.g., allowing the sorting and filtering of results, the incorporation of shading, the ability to search for particular en-

tries), to user interface considerations (e.g., providing pop-up text to add further insight from the author onto the presented data, or allow cells to be selected and provide more information), and so on. Such features allow individuals to efficiently see and explore a dataset from a multitude of perspectives, and this capability holds many benefits for exploring complex tabular results, especially those from MCSS research.

4.1 Why interaction?

> Overview first, zoom and filter, then details-on-demand (Shneiderman, 1996, p. 337, the "Visual Information Seeking Mantra").

Tukey (1990) saw dynamic graphics as the future, and their then current implementation primarily hampered by technological restraints. He believed that allowing individuals to rotate a display or alternate between static images of various projections could afford glimpses into the data that otherwise would be very difficult to see. However, such live manipulations required computing resources that were not readily available at the time. Today, such technological barriers have been all but abolished, and displays can be designed for a variety of purposes that are fast, flexible, accessible, and reproducible.

Frameworks for creating interactive displays are becoming more numerous and widely available. Specifically, interactive displays can be made through additional packages available for modern statistical software or general purpose programming languages, such as R (R Core Team, 2016) and Python,[54] as well as via other more directed applications, such as:

[54]Similar to R, Python has a repository of additional packages that users can install for undertaking particular tasks.

- Mondrian (stats.math.uni-augsburg.de/mondrian);
- Eikosogram (sas.uwaterloo.ca/~rwoldfor/software/eikosograms);
- Tableau (tableausoftware.com) ;
- Processing (processing.org) and Processing.js (processingjs.org); and,
- D3.js (Data-Driven Documents; d3js.org/)

Most of the above approaches are either designed for generating general purpose graphics or for producing a very particular type of plot (e.g., the eikosogram is primarily meant for teaching the axioms of probability based on the proportions from a 2x2 contingency table; Oldford & Cherry, 2006). However, a variety of plots and output are useful for interpreting the results of MCSS. Since such research is generally conducted in R, the core of this chapter will pertain to interactive applets that can be created using that language.

No matter the approach, the goal of providing an interactive display is to allow individuals to see the results for themselves (Theus & Urbanek, 2009). This idea goes hand in hand with modern open data policies (Gewin, 2016), where data used in scientific research is made available for other researchers to scrutinize. This availability promotes collaboration and peer monitoring, with the intent that such behaviors will improve reproducibility and the quality of research at large. Further, it is now easier than ever to provide either links to external files or online supplements for published papers. For example, Belliveau (2012) examined the use of film by Frank and Lillian Gilbreth and

For generating interactive applets, such packages include: mpld3, pygal, and Bokeh.

provided links so readers could view and evaluate the films themselves.[55] Similarly, the *Journal of the Royal Statistical Society* (2016) requires authors to submit both computer code and datasets to be hosted online alongside their published article.

In addition to posting the dataset itself, providing an interactive applet promotes three fundamental aspects of research:

- *Justifiability*: The reader can "see it for themselves"
- *Discoverability*: An applet allows access to views and projections of the data that were previously hard to conceptualize
  - This access might even lead individuals to find patterns or relationships that the original authors may have missed!
- *Narrative*: With an applet, the researchers can curate a virtual experience for their readers that conveys a meaningful story
  - This capability can be especially useful to gain access to an audience that might have otherwise been unreachable

The last point is related to the concept of "infographics" (Cairo, 2013; Smiciklas, 2012), in which statistical graphics are paired with relevant discussion in a visually appealing manner. The result is often either an image or webpage that leads readers through a set of analyses and the author's

[55] When one clicks on the link in the digital version of the article, the film is overlaid on the page which allows the reader to view the content without having to navigate to another website.

conclusions. While it may seem unconventional to present MCSS results in such a manner, the preparation of an infographic for a research project can summarize the important findings in a way that is easily sharable and provocative. Such a snapshot shared on social media can then encourage readers to search out the full article for additional details.

Further, interactive applets can provide a structure to teach difficult statistical concepts in a hands-on manner. Entire introductory statistics textbooks, such as Pepper Williams' *Interactive Statistics for the Behavioral Sciences* (2004), have been written using this concept. The digital presentation of this text provides students with animated and interactive figures, prompts with questions about the material, and hyperlinks that allow them to efficiently navigate through the content. Other, more specific applets have been the focus of a variety of publications, such as `s-CorrPlot` (McKenna, Meyer, Gregg, & Gerber, 2016) for exploring a large number of correlation coefficients, `cranvas` (Cheng, Cook, & Hofmann, 2016) for evaluating time series and longitudinal data, or `bertin` (Sawitzki, 2014) for reproducing Bertin's reorderable matrices.

As discussed in Sigal and Chalmers (2016), interactive displays can be a fantastic method for the presentation of MCSS results within a pedagogical setting. For instance, programming a simulation-based interactive applet pertinent to the week's lecture materials can provide additional benefits for students as it allows them to obtain hands-on experience with the concepts.[56] As one example, Raffle and Brooks (2005) designed an applet for conducting a MCSS that provided robustness, power, and sample size analyses for two-, three-, and four-group designs. Students

[56]Additionally, `SimDesign` has a function `SimShiny()` that allows users to test run particular simulation design combinations within an interactive environment and view the results.

were able to compare testing procedures (e.g., overall ANOVA F-statistic versus pooled variance independent group t-tests and separate variance independent groups t-tests), using several user-defined population parameters (such as the central tendency values per group and a choice of generating distribution for the samples). Students reported that the program was quite successful in terms of helping them understand the performance of these statistics under the various conditions.

Overall, interactive applets are a rich tool for communicating results from complex MCSS research. Such applets can be programmed with relative ease using the `shiny` package in `R` (Chang, Cheng, Allaire, Xie, & McPherson, 2017) and can incorporate many elements of interactivity. Previous research has demonstrated how such applets can be coded for pedagogical purposes (e.g., Doi, Potter, Wong, Alcaraz, & Chi, 2016; Ellis & Merdian, 2015). The following section provides an overview of features which such an interactive display might include for presenting MCSS findings and then how such an applet is implemented via `SimDisplay`.

4.2 Elements of Interactivity

The goal of an interactive display for MCSS results is to allow the dynamic exploration of complex tables and figures. This display must contain:

1. A dataset to explore;

2. Graphical, tabular, or a combination of graphical and tabular displays which are appropriate for visualizing the dataset at hand; and,

3. User controls which allow the manipulation of the display.

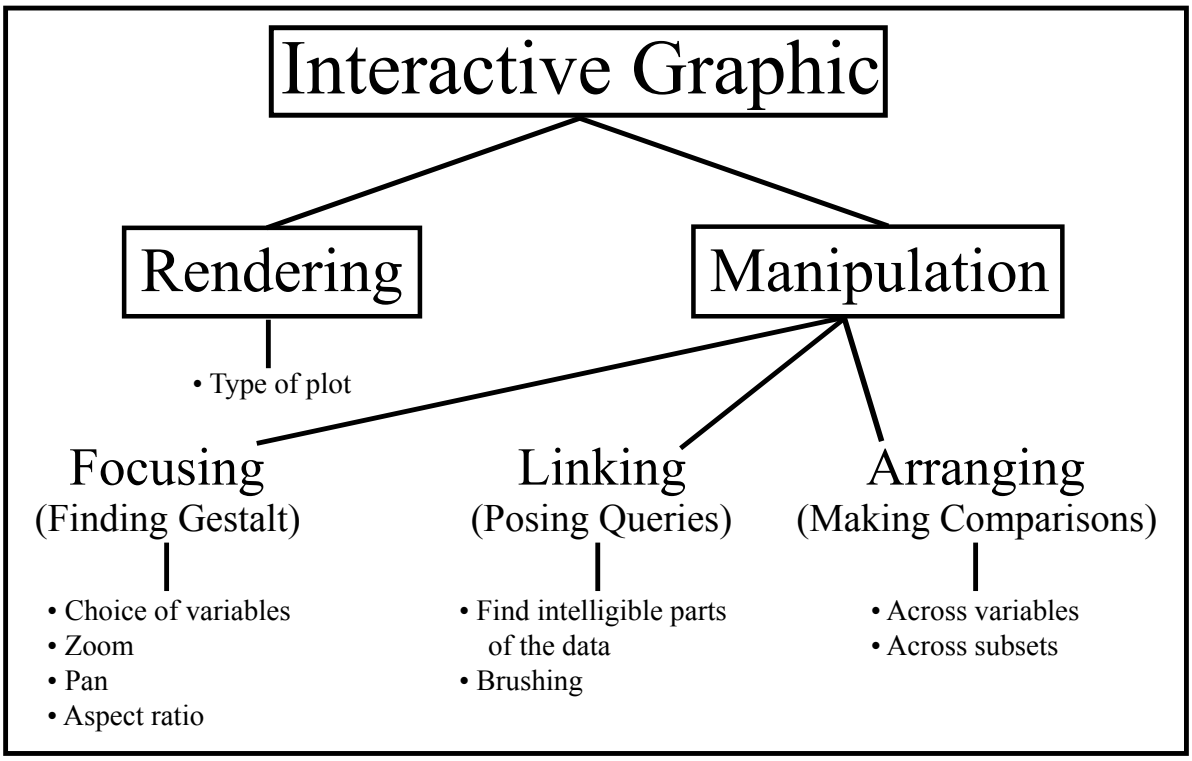
`SimDisplay` contains convenience functions which address each of these components. First, the interactive display call must provide a datafile (specifically of class `SimDesign`), which is to be visualized. Secondly, a variety of options is available to the user to change the type of table or graphic produced, using some of the strategies presented in Chapters 2 and 3. Finally, various toggles appropriate for the chosen display modality will appear and allow for interactive modification of the display. The next section will outline a few of the basic methods for manipulating an interactive display.

4.2.1 Terminology for Interactive Graphics

The creation of an interactive display pertains to the choice of structural elements and manipulations that are made available to the reader. Once this skeleton is conceived, the viewer has the ability to manipulate the parameters of the resulting tables and plots. As Buja, Cook, and Swayne (1996) discussed, once an interactive visualization is live, it can be thought of in terms of rendering (what is currently shown on the plot), and its subsequent manipulation (what is done to it). See Figure 4.1 for an overview of this taxonomy.

The goals of such visualizations in this framework involve assisting viewers in focusing attention (via the Gestalt principles discussed in Chapter 3 to highlight patterns in the display), linking concepts (by posing queries), and arranging the information (in order to make comparisons). Each of the manipulation branches in Figure 4.1 pertains to a set of techniques for accomplishing a spe-

Figure 4.1: Fundamental structure of an interactive display.



cific task (for example, the focusing tasks treat the display as if it were being seen through the lens of a camera that is under the viewer's control to be zoomed, panned, etc.).

Basic interactions allow users to dynamically alter some of the core parameters of a plot. For example, such toggles might allow for the modification of axis limits (increasing or decreasing their range) or the showing and hiding of data labels. Depending on the goals of the application, other features might include:

- *Selection*: the ability of users to dynamically subset the data being visualized using filters
- *Slice*: the ability to facet the data using a selection tool (typically through clicking and dragging on the display via a mouse)
- *Probing*: the generation of "ToolTips" or informative notes for particular data points to gain more insight into the observation (typically shown by clicking on a point itself)
- *Reordering*: allowing users to reorder aspects of the presentation (such as the arrangement of factor variables in a bar plot)
- *Panning* and *Zooming*: the ability to shift the plot in any cardinal direction and to zoom in or out (which allows for the traversal of complex or large datasets)
- *Drill Down*: a method of filtering the display based on a categorical hierarchy (which allows for the hiding of data points to view results for a particular group or subgroup)

- *Modification*: allows users to change the type of analysis, display modality (the form of graphic or table), included variables, etc.

Not all of these characteristics are needed for every visualization, but they provide a fundamental basis for discussing what might be possible to include in such an applet. To actually implement such ideas requires a toolkit of control methods, which is where the `shiny` package shines (Chang et al., 2017).

In the core package,[57] `shiny` allows for a variety of input methods (referred to as "control widgets") that can be rendered in an interactive application. These methods range from simple buttons that users can click on ("action buttons") to a single or group of checkboxes to free-form text inputs. Commonly used widgets also include radio buttons (which could be used for a Likert-type query), select inputs (where users select one or more values from a pre-defined set of choices), and slider inputs (which is a bar that can be configured to have particular minimum, maximum, starting, and step-size values).

For MCSS results in particular, we wish to be able to change between multiple display modalities, to alter basic plot characteristics (such as shading), and to see various projections of the dataset. Such features can make complicated datasets more coherent, present the information from differing perspectives and at various levels of detail, provide support for visual comparisons, and aid in conveying a compelling narrative.

Using these ideas, a straightforward example of an interactive display might pertain to a bar plot

[57] Additional packages, such as `shinyWidgets` (Perrier & Meyer, 2017), provide even more alternatives.

of frequencies or a series of boxplots from an ANOVA design. In either case, most visualizations sort the categories found on the x-axis alphabetically. However, this organization is typically not an ideal method of presentation (see Section 3.5). While methods for data visualization that produce static graphics can be manipulated to reorder these categories in a more meaningful manner (usually by reordering the factor levels of the categorical variable and then recreating the plot), this process can be made more efficient through the implementation of an interactive display. Such an interface would include a dropdown menu with sensible sorting options (such as by the magnitude of the mean or the group variability) which could be applied automatically. Additional features, such as the ability to filter out particular observations or drill down to see results partitioned by certain subgroups, could also be implemented.

Overall, crafting an interactive graphical user interface (GUI) gives power to the reader. Even with only a trivial number of features, it allows them to explore the dataset from a variety of perspectives and methods. The following section will work through some examples of applying such ideas to MCSS output.

4.3 Applied Example: `ShinyMCSS()`

`SimDisplay` features a built-in `shiny` implementation specifically designed for the visualization and exploration of MCSS results: `shinyMCSS()`.[58] This application was written to work seam-

[58] As this interactive display uses functions from a variety of packages, it is highly recommend that users run `install_suggests()` before launching the app. This `SimDisplay` function checks for all of the necessary packages and installs any that are missing.

lessly with dataframe objects of `class(SimDesign)`. There are two methods for launching an interactive session:

```
library(SimDisplay)

# Launch an interactive session with the demo data (Brown and Forsythe replication):
shinyMCSS()
```

or

```
# Launch an interactive session with a dataset found in the current workspace:
data(Brown1974)
shinyMCSS(Brown1974)

data(Hallgren2013) # To load the Hallgren data
shinyMCSS(Hallgren2013)
```

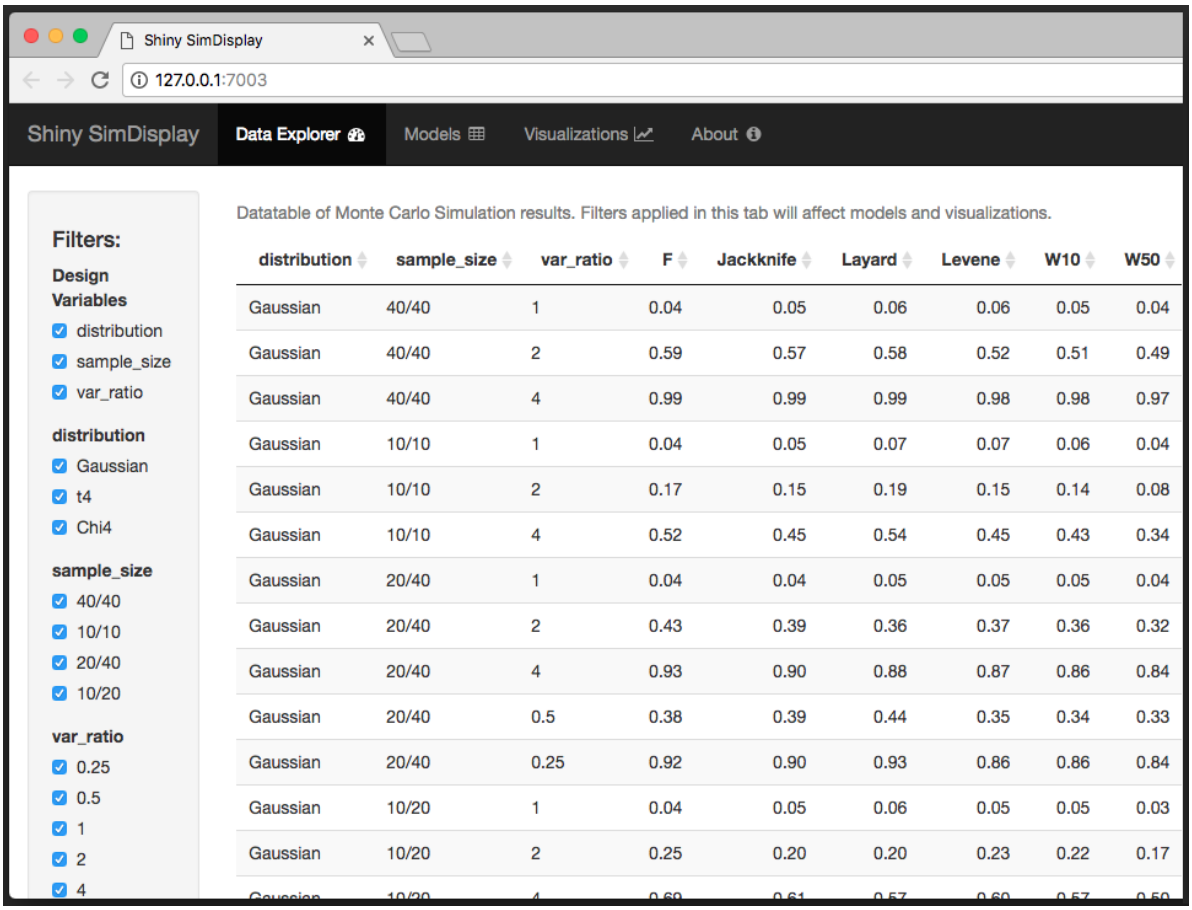
The second approach allows for the interactive data exploration tool to be used on any `SimDesign` dataframe.[59]

4.3.1 Initialization and the Data Explorer View

Upon calling `shinyMCSS()`, a new tab or window will open in the user's default web browser with a local web address (for instance, see the address bar in Figure 4.2). The initial view pertains to the "Data Explorer" tab, in which users are presented with the chosen dataset in the form of a datatable from the `DT` package (Xie, 2016b). A datatable refers to the `R` implementation of the open source and popular jQuery DataTables plug-in for Javascript, which is a library that produces

[59] As a reminder, this procedure can work with simulation data not produced by `SimDesign`. In this case, the dataframe must share a similar structure to one from a `SimDesign` simulation (namely, it should be a dataframe object with design variables specified as factor vectors and outcome variables as numeric vectors). The dataframe can then be passed to the `SimDisplay` function `convertDf()` prior to calling `shinyMCSS()`. See `?convertDf()` for more details.

Figure 4.2: Initial view of Shiny SimDisplay.



web compliant tables.[60] When embedded within an interactive applet, this package allows for the searching and sorting of the data array. By default, the rows and columns of the table are presented in the same order as they appear in the passed dataframe and columns have sensible alignment: numeric columns are always aligned to the right and strings are always aligned to the left.

In addition to the data itself, toggles are provided in the sidebar on the left-hand side of the

[60] See `https://datatables.net/` for more information.

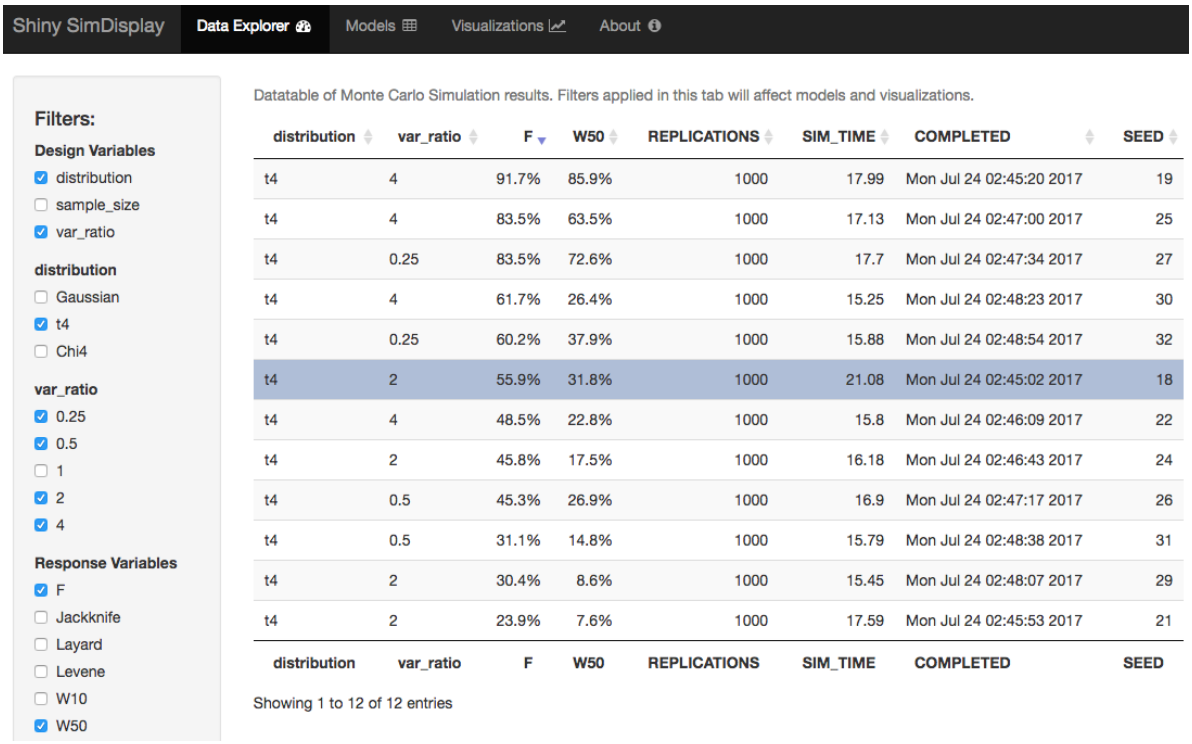
display that allow manipulation of the dataset. Many of these toggles are dynamic, as their values are only set when the application has been initialized and are actively monitored for any changes due to user input. The flexibility afforded by this structure allows the application to be usable for any MCSS dataset.

The toggles currently implemented in the Data Explorer tab pertain to:

- the hiding of design columns;
- the subsetting of the dataframe by particular factor levels of the design variables (useful for separating Type I error and power rate conditions);
- the inclusion or hiding of the simulation result variables;
- a checkbox input to show or hide the meta-level variables that are automatically produced by `SimDesign` (pertaining to the number of replications for that design combination, the amount of time it took to execute, when it was completed, and the seed value, if present);
- a checkbox input to switch between showing the results as percentages or in decimal notation; and,
- a numeric input pertaining to the amount of rounding to apply to the results.

Between these options and some core datatable functionality (such as being able to sort the table in ascending or descending order by any of the columns, the ability to click on one or more rows to highlight them, and having the number of rows currently included based upon the various filter

Figure 4.3: Using the filters of Shiny SimDisplay.



selections at the bottom of the table), one can become very familiar with their dataset. This tab provides an easy to use and understand view of the raw data. See Figure 4.3 for a screenshot of this display where the power conditions for distribution (specifically, the *t*-distribution subset) and variance ratio design variables are included for the *F* and *W50* test statistics. Further, results are shown as percentages alongside their meta variables, sorted in descending order by the empirical detection rate for the *F*-statistic, and one row is highlighted to demonstrate how such visual cues can make for easier viewing.

In addition to providing an interactive way for researchers to explore their MCSS results, the

options in the Design Explorer tab are tracked and applied in subsequent tabs. As such, if the user is interested in the Type I error rate conditions from the motivating example, they could unselect all levels of the `var_ratio` variable except for `var_ratio` = 1. All subsequent analyses will only be done on the twelve rows pertaining to that condition.

4.3.2 Univariate and Multivariate Modeling

Once a particular subset (or the entire dataframe, depending on the structure of the MCSS) is settled upon in the Data Explorer, it is recommended that the researcher proceed to the Models tab. Unlike the Data Explorer, the initial view of this tab only shows a warning: "Please select at least one design and one outcome variable" (see Figure 4.4). The structure of these inputs allows users some flexibility in building an appropriate ANOVA model pertinent to their design.

The toggles in this tab pertain to which design variables (taking into consideration any subsets made in the Data Explorer tab) and outcome variables to include in the model, whether interactions should be incorporated and whether to use logits of the dependent variable.[61] If only one outcome variable is selected, the model is a univariate ANOVA; if more than one is chosen, the model is a MANOVA. In both cases, results are presented using the `Anova()` function from the `car` package (Fox & Weisberg, 2010). This structure allows users to efficiently run a variety of models, as they are free to change the model parameters as well as the underlying data. Figure 4.5 reproduces the ANOVA table found in Section 2.5.6. This display was accomplished by showing

[61] If the response variables consist of rates (e.g., EDRs), taking the logit of the DV will help stabilize the proportion-based summary statistics when computing the parameters.

Figure 4.4: Using the filters of Shiny SimDisplay.

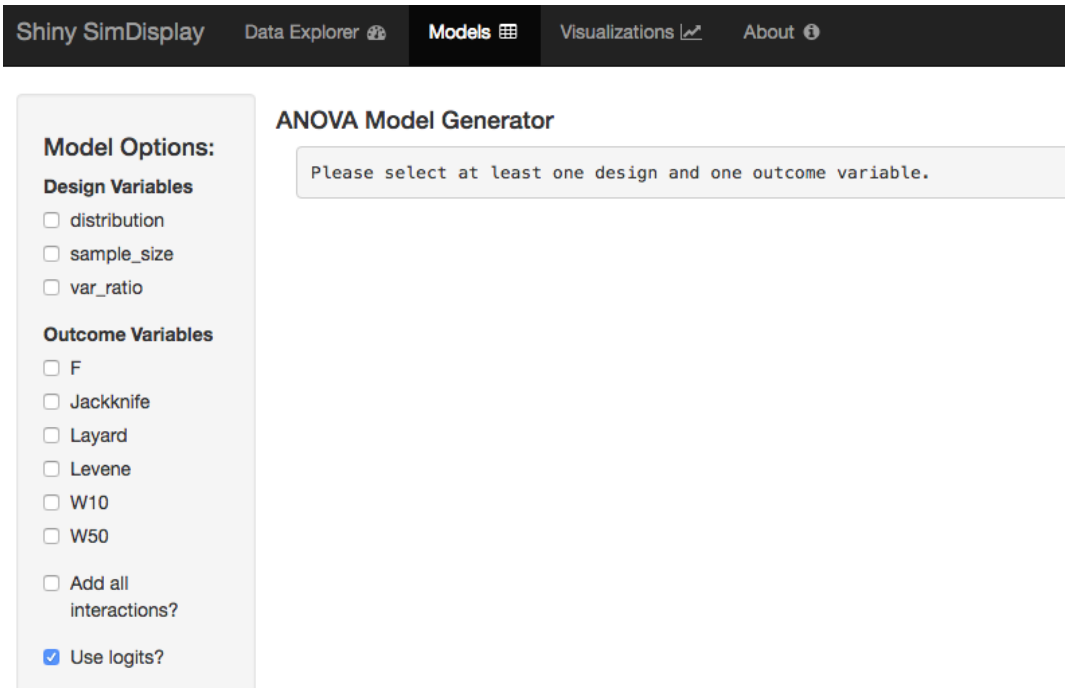
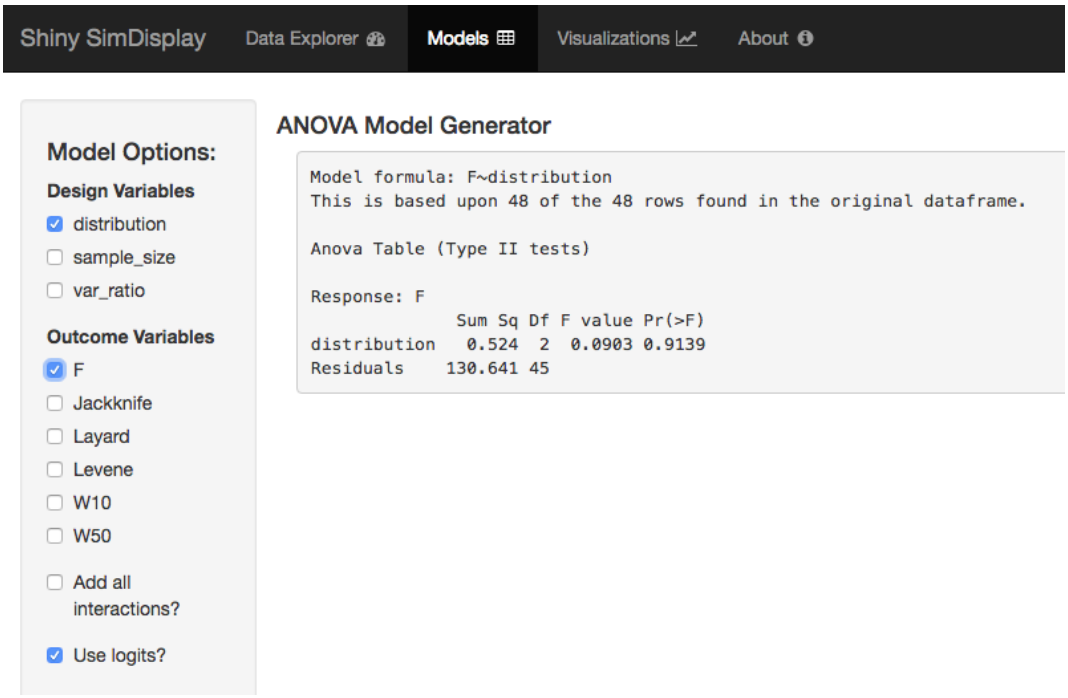


Figure 4.5: ANOVA in Shiny SimDisplay.



only the Type I error rate conditions in the Design Explorer tab and then choosing Distribution as the design variable and F as the outcome under Models.

4.3.3 Data Visualization

The final tab pertains to some of the methods of data visualization discussed in Chapter 3. Depending on the type of graphic selected, a different conditional panel with appropriate customization options will appear in the sidebar on the left-hand side of the screen. Again, the data passed onto each of these functions is subsetted by whatever filters are applied in the Data Explorer.

4.3.3.1 Shaded Tables

The first visualization to appear is a shaded table display. As an example, `SimDisplay` users can reproduce Figure 3.5 by dropping the Type I error condition in the Data Explorer and then switching to the Visualizations tab (see Figure 4.6).

Currently implemented options for shaded tables include: a custom title, the choice of facet variable, the choice of x-axis variable, custom labels for the x- and y-axes, slider inputs for the value at which to use white rather than black text and for determining the number of rows the facets should appear on, and a drop-down menu to switch between the various palettes provided by `RColorBrewer`. Changing any of these options immediately recreates the plot and allows users to efficiently obtain many snapshots of their data from various perspectives.

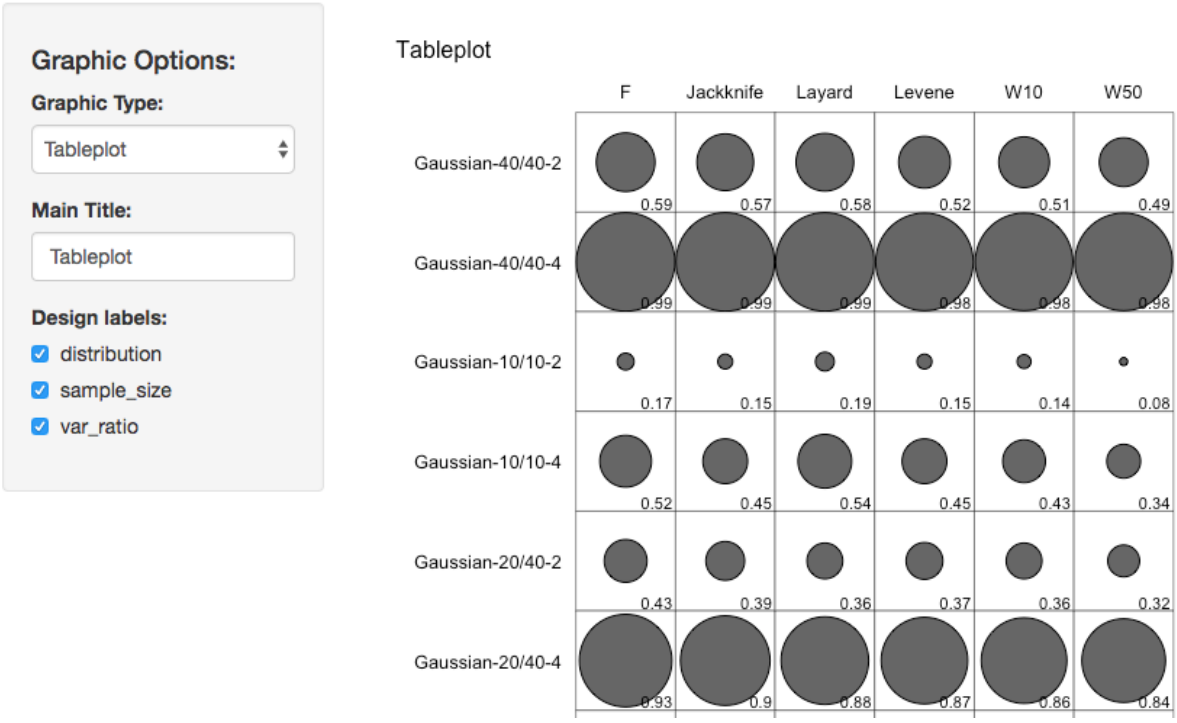
4.3.3.2 Tableplots

The second type of visualization available in the `shinyMCSS()` viewer are tableplots. This tab renders the possibly subsetted dataframe (based on the filters applied in the Data Explorer tab) as a tableplot.

This display currently has parameters to change the title of the graphic as well as which design factor levels to use when naming across rows. The latter control is useful if one of the design factors is redundant; for example, when looking at Type I error rate conditions in the motivating example, the variance ratio variable only takes on the value of 1 across all rows and should be

Figure 4.6: Shaded Table display in Shiny SimDisplay.

Figure 4.7: Tableplot display in Shiny SimDisplay.



removed.

As an example, the tableplot in Figure 4.7 was produced by selecting the Gaussian distribution and the power conditions under variance ratio option in the Data Explorer tab. Further, to reproduce Figure 3.9, only a few adjustments would be needed: the power conditions should be hidden in the Data Explorer tab and `var_ratio` should be removed from the design labels under graphic options.

4.3.3.3 Boxplots

The third type of visualization available in the viewer are boxplots. The boxplot showing Type I error rates for each of the outcomes and facetted by generating distribution (Figure 3.10) can be easily reproduced here by subsetting out the power conditions and navigating to the boxplot submenu (see Figure 4.8). Once the initial plot has been rendered, users can dynamically alter the variable used to facet the plot, the main title, and the axis labels.[62] Further, boxplots can also be reordered based on their group average by checking 'Order by group means' (although this organization tends to make the most sense when the faceting variable has been reduced to only one level, such as in Figure 3.11).

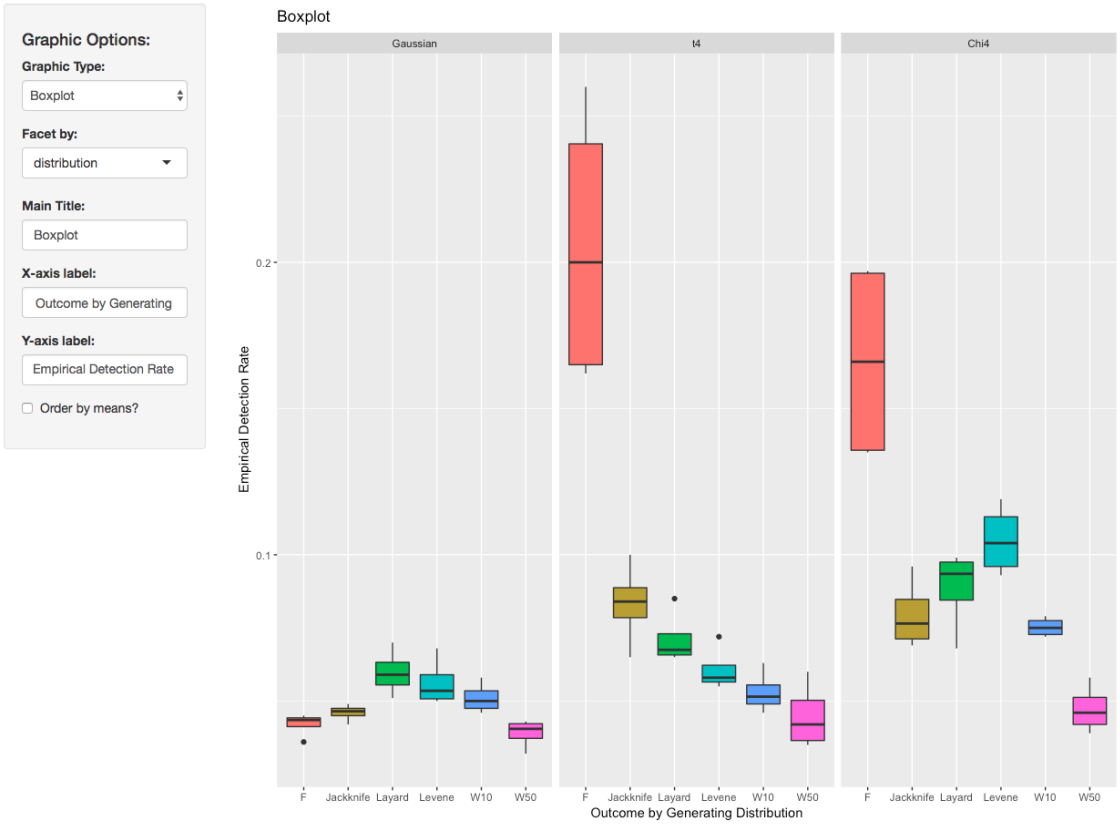
4.3.3.4 HE Plot Framework

The final section in the visualization tab allows for users to produce the three variants of the HE plot, as described in Chapter 3: the regular HE plot (where users can select which variables appear on the x- and y-axes), the traditional canonical discriminant analysis (CDA) plot, and the CDA HE plot (where users can adjust the transparency settings). Each of these visualizations requires a MANOVA to be run in the Models tab prior to rendering, as that model is used as a basis for the graphic.

Figure 4.9 shows an example of the CDA HE plot panel that reproduces Figure 3.17 by check-

[62]The default X and Y labels are 'variable' and 'value', respectively, and are modified in Figure 4.8 to be more meaningful.

Figure 4.8: Boxplot display in Shiny SimDisplay.



ing the fill option and setting the transparency level to 0.1. The HE plot and the CDA plots are generated by changing the plot method selection.

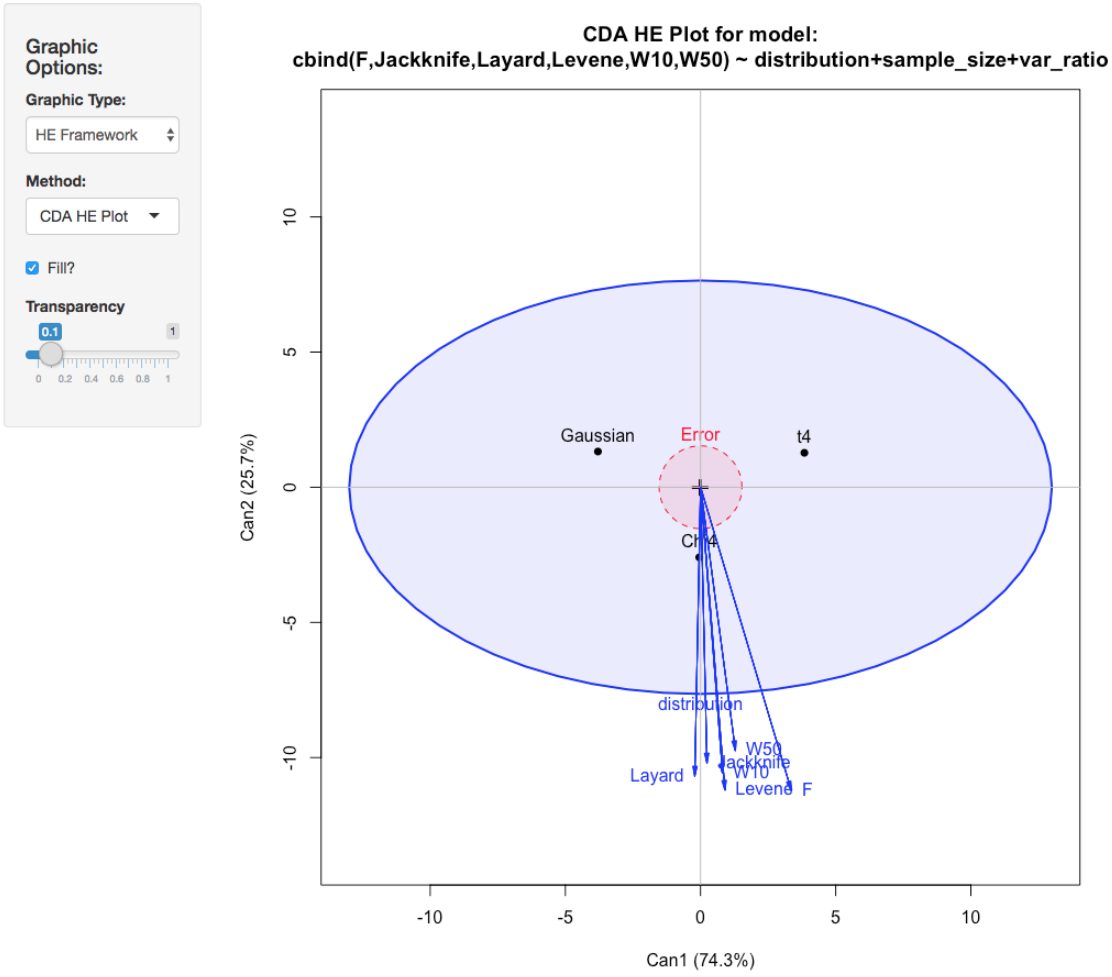
4.4 Conclusion

Interactive graphics are becoming more commonplace in both the academic and public spheres. As languages such as `D3.js`, online data analytic and visualization tools like `plotly`,[63] and `R` packages like `shiny` emerge and expand, it is getting easier to create GUIs that can be hosted online or run locally on a standard personal computer. As another recent example, the `ggvis` package is a restructuring of `ggplot2` that aims to support interactive elements as a core feature and without the need of an additional framework like `shiny` (Chang & Wickham, 2016). The syntax for generating plots with this package differs from that of `ggplot2` and serves as a demonstration of how interaction can be embedded within the grammar of graphics framework. This process can be understood as a loop, during which parameters are able to be changed on the fly and where each change produces an instantaneous re-rendering of the plot. As such, interactive graphics can be understood as a grammar of graphics pipeline that is continuously available for rendering.

The `shinyMCSS()` implementation provides an introduction to such presentations. Being able to rapidly alter the underlying dataset, test various models, and produce a wide array of relevant visualizations can help researchers and their readers get an understanding of what is happening in their results. And, all one needs is a dataset loaded in their `R` workspace to run the application.

[63] `https://plot.ly/`

Figure 4.9: CDA HE plot in Shiny SimDisplay.



Even basic manipulations, such as those described in this chapter, provide a substantial amount of power to the reader. Hosting an accompanying GUI is one way to open up a dataset and its subsequent analysis for evaluation and feedback, and exemplifies the ideals of scientific research. There are many potential applications for such displays, and this chapter touched on how they might be used for and benefit MCSS research.

5 Future Work and Conclusion

The core of this dissertation pertained to the structure of Monte Carlo simulation studies and the presentation of their results, which involved showing how these experiments are conducted and what their typical findings look like. As discussed, MCSS research often produces complex high-dimensional tables. My goal was to present methods for improving the communication of these findings, as well as to provide an `R` package that can assist in this process.

Researchers have many important considerations when preparing a MCSS for publication. As Boomsma (2013) discussed, publications must focus on the main research question, link the interpretation of results to theoretical expectations, present the essential results in a concise and elucidative form, and provide unambiguous conclusions. Taken together, these points demonstrate the importance of *narrative* - that the results from MCSS research are only helpful to the extent that they easily allow other researchers to grasp fundamental findings. Finding and conveying such a story should be easier when it is based on the ideas presented here.

Further, this dissertation represents ongoing work on at least two fronts. Firstly, as an overview of methods for summarizing and exploring Monte Carlo simulation results, it is expected that

future work in the fields of data visualization and statistical theory will suggest novel approaches to apply to such data. As demonstrated throughout this dissertation, there is a multitude of ways we can decompose and inspect a multidimensional tabular display. Having explored some of the techniques currently available, future manuscripts can concentrate on which methods promote the best understanding.

Secondly, the applied component of this dissertation was an `R` package specifically designed to showcase such techniques and render them easy to use for applied research. While the current version of `SimDisplay` serves as a demonstrative implementation of these ideas for the purpose of this dissertation, this version is but a snapshot of a continuing pursuit. As an open source project, the package lives online and users can report issues, request features, and even contribute to the codebase through the package's GitHub repository.[64]

In the near future, I hope to maintain and expand on `SimDisplay` with both new functions and additional features for the currently implemented methods. Resources not referenced in this dissertation on table presentation, such as Ehrenberg (1975), provide additional approaches on how to best display tabular data. In particular, Ehrenberg presents rules for organizing the rows and columns of tables (based upon marginal averages or some other measures of size) and how they relate to various statistical relationships. One particular insight is to put values that are to be compared into columns rather than rows (with larger values on top if possible). Such rules produce tables that are easier to read and should be natively incorporated into `SimDisplay`.

[64] `https://github.com/mattsigal/SimDisplay/issues`

Further, there are many features that would be beneficial to incorporate in the interactive applet. In the Data Explorer view, it would be useful to incorporate a control to allow for the shading of particular cells. This feature would mirror the functionality in the `tableShade()` function, albeit in an interactive setting. Another feature might be to allow the user to define clusters for tables (e.g., pertaining to the separation of the Type I error and power rates) and allow both to appear above or beside each other on the display. It would also be useful to program some functionality that draws on the ability to select rows from a datatable. The highlighted rows are tracked behind the scenes, but at the moment that information is not linked to any visible output.

For the Models tab, there are also some features that could improve the practicality of the app. Since these models are primarily looked at in order to get a sense of the variables, significance testing and *p*-values are not the focus. As such, it would be helpful for the `shinyMCSS()` output to include other information, such as effect sizes. In Chapter 2, `SimAnovaMV()` was used to condense the output of six univariate ANOVAs into one tabular display. It would be handy for the app to provide something similar, as when multiple response variables are selected for the model in the current version, the output from a MANOVA model is presented instead. Also, support should be added to allow more nuanced interactions as at present they are either all incorporated[65] or not included at all. It would be helpful if a particular or a set of interaction terms could be dropped or added interactively through the GUI. One approach that might be implemented that would allow

[65]Interactions are coded via replacing the + in the linear model formulas with an `*`, which in `R` indicates that we would like *all* higher-order interactions between the set of variables. Thus, with four design factors, it would include a four-way interaction, all three- and two-way interactions, as well as the main effects.

the greatest flexibility in terms of incorporating interactions is to allow model formula input in the app itself.

Within the Visualizations tab, there are aspects of the shaded table that I will continue to work on. For instance, it would be useful to allow the filtering of values given some combination of other factors. This filtering was mentioned in Chapter 1, where Brown and Forsythe hid particular values from their power table based on a criterion that assessed if the method's Type I error rates were too liberal. For the tableplot, it would be useful if multiple shapes could be used (e.g., to demarcate values that are too liberal or too conservative). This shading could also be highlighted by incorporating varying colors. For the HE plot framework, this tab should allow users to incorporate additional model and hypothesis parameters, and controls should be created to allow the dynamic adjustment of some of the plot parameters.

Finally, it would be ideal to allow users to export information out of the app. Whether it be a filtered datatable, a set of model output, or a variety of visualizations, a button could be added to each display tab that would allow that view to be saved in a reproducible fashion. This output perhaps could take the form of a text file detailing the particular filters currently applied and the selected output in an appropriate filetype.

Overall, MCSS are a fundamental technique for research in the field of quantitative methods. An impressive variety of hypotheses can be assessed using this methodology and, as it becomes easier to program and run such simulations, I foresee that their popularity will continue to rise in both published research and the pedagogical environment. This dissertation aimed to support that

trend by introducing MCSS research in a way that explains their basic structure and by providing methods for understanding and decomposing their results. These ideas and the functions found in `SimDisplay` should serve researchers well in their future analyses.

Bibliography

American Psychological Association. (2009). *Publication manual* (6th ed.). Washington, DC: Author.

Arnau, J., Bendayan, R., Blanca, M. J., & Bono, R. (2013). The effects of skewness and kurtosis on the robustness of linear mixed models. *Behavioural Research*, *45*(3), 873–879. doi: 10.3758/s13428-012-0306-x

Arnholt, A. T. (1999). Simulating sampling distributions. *Teaching Statistics*, *21*(1), 14–16. doi: 10.1111/j.1467-9639.1999.tb00791.x

Baron, R. M., & Kenny, D. A. (1986). The moderator–mediator variable distinction in social psychological research. *Journal of Personality and Social Psychology*, *51*(6), 1173–1182. doi: 10.1037/0022-3514.51.6.1173

Belliveau, A. (2012, May). Psychology's first forays into film. *Monitor on Psychology*, *43*(5), 24-27.

Bertin, J. (1967). *Sémiologie graphique*. Paris: Gauthier-Villars.

Bertin, J. (1981). *Graphics and graphic information processing*. Berlin: Gruyter & Co.

Bivand, R. S., Pebesma, E., & Gomez-Rubio, V. (2008). *Applied spatial data analysis with R* (2nd ed.). New York, NY: Springer.

Bollen, K. A. (1989). *Structural equations with latent variables* (1st ed.). Hoboken, New Jersey: Wiley-Interscience.

Bollen, K. A., Harden, J. J., Ray, S., & Zavisca, J. (2014). BIC and alternative Bayesian Information Criteria in the selection of Structural Equation Models. *Structural Equation Modeling*, *21*, 1–19. doi: 10.1080/10705511.2014.856691

Boomsma, A. (2013). Reporting Monte Carlo studies in Structural Equation Modeling. *Structural Equation Modeling*, *20*, 518–540. doi: 10.1080/10705511.2013.797839

Borkin, M. A., Bylinskii, Z., Kim, N. W., Bainbridge, C. M., Yeh, C. S., Borkin, D., ... Oliva, A. (2016). Beyond memorability: Visualization recognition and recall. *IEEE Transactions on Visualization and Computer Graphics*, *22*(1), 519–528. doi: 10.1109/tvcg.2015.2467732

Brown, M. B., & Forsythe, A. B. (1974). Robust tests for the equality of variances. *Journal of the American Statistical Association*, *69*(346), 364–367. doi: 10.1080/01621459.1974.10482955

Buja, A., Cook, D., & Swayne, D. F. (1996). Interactive high-dimensional data visualization. *Journal of Computational and Graphical Statistics*, *5*(1), 78. doi: 10.2307/1390754

Cairo, A. (2013). *The functional art*. Berkeley, CA: New Riders.

Chalmers, R. P. (2015). Extended mixed-effects item response models with the MH-RM algorithm. *Journal of Educational Measurement*, *52*(2), 200–222. doi: 10.1111/jedm.12072

Chalmers, R. P. (2016). *SimDesign: Structure for organizing Monte Carlo simulation designs.* Retrieved 11.04.2017, from https://CRAN.R-project.org/package=SimDesign (R package version 1.6)

Chalmers, R. P. (2017). *SimDesign wiki.* Retrieved 11.04.2017, from https://github.com/philchalmers/SimDesign/wiki

Chalmers, R. P., Counsell, A., & Flora, D. B. (2016). It might not make a big DIF. *Educational and Psychological Measurement*, *76*(1), 114–140. doi: 10.1177/0013164415584576

Chalmers, R. P., & Flora, D. B. (2014). Maximum-likelihood estimation of noncompensatory IRT models with the MH-RM algorithm. *Applied Psychological Measurement*, *38*(5), 339–358. doi: 10.1177/0146621614520958

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2017). shiny: Web application framework for R [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=shiny (R package version 1.0.3)

Chang, W., & Wickham, H. (2016). ggvis: Interactive grammar of graphics [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=ggvis (R package version 0.4.3)

Cheng, X., Cook, D., & Hofmann, H. (2016). Enabling interactivity on displays of multivariate time series and longitudinal data. *Journal of Computational and Graphical Statistics*, *25*(4), 1057–1076. doi: 10.1080/10618600.2015.1105749

Cleveland, W. S. (1993). *Visualizing data* (1st ed.). Summit, NJ: Hobart Press.

Cleveland, W. S. (1994). *The elements of graphing data*. Summit, New Jersey: Hobart Press.

Cleveland, W. S., Diaconis, P., & McGill, R. (1982). Variables on scatterplots look more highly correlated when the scales are increased. *Science*, *216*(4550), 1138–1141. doi: 10.1126/science.216.4550.1138

Cleveland, W. S., & McGill, R. (1984). Graphical perception. *Journal of the American Statistical Association*, *79*(387), 531-554. doi: 10.1080/01621459.1984.10478080

Cook, A. R., & Teo, S. W. L. (2011). The communicability of graphical alternatives to tabular displays of statistical simulation studies. *PLoS ONE*, *6*(11), e27974. doi: 10.1371/journal.pone.0027974

Cook, D., & Swayne, D. F. (2007). *Interactive and dynamic graphics for data analysis* (1st ed.). New York, NY: Springer-Verlag. doi: 10.1007/978-0-387-71762-3

Cotton, R. (2013). *Learning R*. Sebastopol, CA: O'Reilly Media.

Diez, D. D., Barr, C. D., & Cetinkaya-Rundel, M. (2014). *Introductory statistics with randomization and simulation* (1st ed.). OpenIntro Publishing. Retrieved from `https://www.openintro.org/download.php?file=isrs1`

Doi, J., Potter, G., Wong, J., Alcaraz, I., & Chi, P. (2016). Web application teaching tools for statistics using R and shiny. *Technology Innovations in Statistics Education*, *9*(1), 1–32.

Ehrenberg, A. S. C. (1975). *Data reduction*. Chichester, West Sussex: Wiley.

Ellis, D. A., & Merdian, H. L. (2015). Thinking outside the box: Developing dynamic data visualizations for psychology with shiny. *Frontiers in Psychology*, *6*. doi: 10.3389/fpsyg

.2015.01782

Evergreen, S. D. H. (2014). *Presenting data effectively*. Thousand Oaks, CA: Sage Publications.

Everton, T. (1984). Probalistic simulation in the classroom. *Teaching Statistics*, *6*(1), 2–5. doi: 10.1111/j.1467-9639.1984.tb00519.x

Farquhar, A. B., & Farquhar, H. (1891). *Economic and industrial delusions*. New York, NY: G. P. Putnam's Sons.

Few, S. (2012). *Show me the numbers* (2nd ed.). Oakland, California: Analytics Press.

Fox, J. (2003). Effect displays in R for generalised linear models. *Journal of Statistical Software*, *8*(15), 1–27. doi: 10.18637/jss.v008.i15

Fox, J., & Hong, J. (2009). Effect displays in R for multinomial and proportional-odds logit models: Extensions to the effects package. *Journal of Statistical Software*, *32*(1), 1–24. doi: 10.18637/jss.v032.i01

Fox, J., & Weisberg, H. S. (2010). *An R companion to Applied Regression* (2nd ed.). Thousand Oaks, CA: Sage Publications.

Friendly, M. (2007). HE plots for multivariate linear models. *Journal of Computational and Graphical Statistics*, *16*(2), 421–444. doi: 10.1198/106186007X208407

Friendly, M. (2010). HE plots for repeated measures designs. *Journal of Statistical Software*, *37*(4), 1–40. Retrieved from `http://www.jstatsoft.org/v37/i04/` doi: 10.18637/jss.v037.i04

Friendly, M., & Fox, J. (2016). candisc: Visualizing generalized canonical discriminant and

canonical correlation analysis [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=candisc (R package version 0.7-2)

Friendly, M., & Kwan, E. (2003). Effect ordering for data displays. *Computational Statistics & Data Analysis*, *43*(4), 509–539. doi: 10.1016/s0167-9473(02)00290-6

Friendly, M., & Kwan, E. (2011). Comment: Graph people versus table people. *Journal of Computational and Graphical Statistics*, *20*(1), 18–27. doi: 10.1198/jcgs.2011.09166b

Friendly, M., Monette, G., & Fox, J. (2013). Elliptical insights. *Statistical Science*, *28*(1), 1–39. doi: {10.1214/12-sts402}

Friendly, M., & Sigal, M. (2014). Recent advances in visualizing multivariate linear models. *Revista Colombiana de Estadistíca*, *37*(2), 261–283. doi: 10.15446/rce.v37n2spe.47934

Friendly, M., & Sigal, M. J. (2017). Graphical methods for multivariate linear models in psychological research: An R tutorial. *The Quantitative Methods for Psychology*, *13*(1), 20–45. doi: 10.20982/tqmp.13.1.p020

Garnier, S. (2017). viridis [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=viridis (R package version 0.4.0)

Gewin, V. (2016). Data sharing. *Nature*, *529*(7584), 117–119. doi: 10.1038/nj7584-117a

Goldman, R. N., & McKenzie, J. D. J. (2009). Creating realistic data sets with specified properties via simulation. *Teaching Statistics*, *31*(1), 7–11. doi: 10.1111/j.1467-9639.2009.00350.x

Gordon, I., & Finch, S. (2015). Statistician heal thyself. *Journal of Computational and Graphical Statistics*, *24*(4), 1210–1229. doi: 10.1080/10618600.2014.989324

Gravetter, F. J., & Wallnau, L. B. (2012). *Statistics for the behavioral sciences* (9th ed.). Belmont, CA: Wadsworth/Cengage Learning.

Hagtvedt, R., Jones, G. T., & Jones, K. (2007). Pedagogical simulation of sampling distributions and the central limit theorem. *Teaching Statistics*, *29*(3), 94–97. doi: 10.1111/j.1467-9639 .2007.00270.x

Hagtvedt, R., Jones, G. T., & Jones, K. (2008). Teaching confidence intervals using simulation. *Teaching Statistics*, *30*(2), 53–56. doi: 10.1111/j.1467-9639.2008.00308.x

Hahsler, M., Hornik, K., & Buchta, C. (2008). Getting things in order. *Journal of Statistical Software*, *25*(3). doi: 10.18637/jss.v025.i03

Hallgren, K. A. (2013). Conducting simulation studies in the R programming environment. *Tutorials in Quantitative Methods for Psychology*, *9*(2), 43–60. doi: 10.20982/tqmp.09.2.p043

Heene, M., Hilbert, S., Freudenthaler, H. H., & Bühner, M. (2012). Sensitivity of SEM fit indexes with respect to violations of uncorrelated errors. *Structural Equation Modeling*, *19*, 36–50. doi: 10.1080/10705511.2012.634710

Hittner, J. B., May, K., & Silver, N. C. (2003). A Monte Carlo evaluation of tests for comparing dependent correlations. *The Journal of General Psychology*, *130*(2), 149–168. doi: 10.1080/ 00221300309601282

Hurley, C. (2004). Clustering visualizations of multidimensional data. *Journal of Computational and Graphical Statistics*, *13*(4), 788–806. doi: 10.1198/106186004x12425

Jones, O., Maillardet, R., & Robinson, A. (2009). *Scientific programming and simulation using R*.

Boca Raton, FL: Chapman and Hall/CRC.

Journal of the Royal Statistical Society. (2016). *Notes on the submission of papers.* http://onlinelibrary.wiley.com/journal/10.1111/(ISSN)1467-985X/homepage/rss-journal-notes-for-authors-2016.pdf.

Kenny, D. A., Kaniskan, B., & McCoach, D. B. (2015). The performance of RMSEA in models with small degrees of freedom. *Sociological Methods & Research*, *44*(3), 486–507. doi: 10.1177/0049124114543236

Kwan, E., Lu, I. R. R., & Friendly, M. (2009). Tableplot: A new tool for assessing precise predictions. *Zeitschrift für Psychologie / Journal of Psychology*, *217*(1), 38–48. doi: 10 .1027/0044-3409.217.1.38

Lane, D. M., & Sándor, A. (2009). Designing better graphs by including distributional information and integrating words, numbers, and images. *Psychological Methods*, *14*(3), 239–257. doi: 10.1037/a0016620

Lewandowsky, S., & Spence, I. (1989). The perception of statistical graphs. *Sociological Methods & Research*, *18*(2-3), 200–242. doi: 10.1177/0049124189018002002

Lidwell, W., Holden, K., & Butler, J. (2003). *Universal principles of design*. Beverly, MA: Rockport Publishers.

Mackinnon, D. P., Fairchild, A. J., & Fritz, M. S. (2007, January). Mediation analysis. *Annual review of psychology*, *58*(1), 593–614. doi: 10.1146/annurev.psych.58.110405.085542

Matloff, N. (2011). *The art of R programming: A tour of statistical software design*. San Francisco,

CA: No Starch Press.

McCormick, W. T., Schweitzer, P. J., & White, T. W. (1972). Problem decomposition and data reorganization by a clustering technique. *Operations Research*, *20*(5), 993–1009. doi: 10 .1287/opre.20.5.993

McKenna, S., Meyer, M., Gregg, C., & Gerber, S. (2016). s-CorrPlot: An interactive scatterplot for exploring correlation. *Journal of Computational and Graphical Statistics*, *25*(2), 445–463. doi: 10.1080/10618600.2015.1021926

Miller, J. E. (2007). Organizing data in tables and charts. *Teaching Statistics*, *29*(3), 98-101. doi: 10.1111/j.1467-9639.2007.00275.x

Mills, J. D. (2002). Using computer simulation methods to teach statistics. *Journal of Statistics Education*, *10*(1), 1–21. Retrieved from `http://www.amstat.org/publications/jse/v10n1/mills.html`

Mooney, C. Z. (1997). *Monte Carlo simulations*. Thousand Oaks, CA: Sage.

Murrell, P. (2011). *R graphics* (2nd ed.). Chapman and Hall/CRC.

Noland, R. B., Klein, N. J., & Tulach, N. K. (2013). Do lower income areas have more pedestrian casualties? *Accident Analysis and Prevention*. doi: 10.1016/j.aap.2013.06.009

Nordstokke, D. W., & Zumbo, B. D. (2007). A cautionary tale about Levene's tests for equal variances. *Journal of Educational Research and Policy Studies*, *7*(1), 1–14.

Oldford, R. W., & Cherry, W. H. (2006). *Picturing probability: The poverty of Venn diagrams, the richness of eikosograms.* Retrieved from `http://www.stats.uwaterloo.ca/`

~rwoldfor/papers/venn/eikosograms/paperpdf.pdf

Paxton, P., Curran, P. J., Bollen, K. A., Kirby, J., & Chen, F. (2001). Monte Carlo experiments. *Structural Equation Modeling*, *8*(2), 287–312. doi: 10.1207/S15328007SEM0802_7

Perin, C., Dragicevic, P., & Fekete, J.-D. (2014). Revisiting Bertin matrices. *IEEE Transactions on Visualization and Computer Graphics*, *20*(12), 2082–2091. doi: 10.1109/tvcg.2014.2346279

Perrier, V., & Meyer, F. (2017). shinywidgets: Custom inputs widgets for Shiny [Computer software manual]. Retrieved from https://CRAN.R-project.org/package=shinyWidgets (R package version 0.3.4)

R Core Team. (2016). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria.

Raffle, H., & Brooks, G. P. (2005). Using Monte Carlo software to teach abstract statistical concepts. *Teaching of Psychology*, *32*(3), 193–195. doi: 10.1207/s15328023top3203_12

Ramsey, P. H., & Ramsey, P. P. (2009). Power and Type I errors for pairwise comparisons of means in the unequal variances case. *British Journal of Mathematical and Statistical Psychology*, *62*, 263–281. doi: 10.1348/000711008X291542

Rendgen, S. (2012). *Information graphics*. Cologne, Germany: Taschen.

Robinson, D. (2014). broom: An R package for converting statistical analysis objects into tidy data frames. *arXiv preprint*. doi: arXiv:1412.3565c2[stat.CO]

Ross, S. (2013). *Simulation* (5th ed.). San Diego, CA: Academic Press.

Sass, D. A., Schmitt, T. A., & Marsh, H. W. (2014). Evaluating model fit with ordered categorical data within a measurement invariance framework. *Structural Equation Modeling*, *21*, 167–180. doi: 10.1080/10705511.2014.882658

Sawitzki, G. (2014). Bertin plots [Computer software manual]. Retrieved from `http://bertin.r-forge.r-project.org/` (R package version 0.1-94)

Sayegh, M. A., Castrucci, B. C., Lewis, K., & Hobbs-Lopez, A. (2010). Teen pregnancy in Texas. *Maternal and Child Health Journal*, *14*(1), 94–101. doi: 10.1007/s10995-008-0436-z

Scaife, M., & Rogers, Y. (1996). External cognition. *International Journal of Human-Computer Studies*, *45*(2 d), 185–213. doi: 10.1006/ijhc.1996.0048

Schönbrodt, F. D., & Perugini, M. (2013). At what sample size do correlations stabilize? *Journal of Research in Personality*, *47*(5), 609–612. doi: 10.1016/j.jrp.2013.05.009

Shneiderman, B. (1996). The eyes have it. In *Proceedings from IEEE symposium on visual languages.* IEEE Comput. Soc. Press. doi: 10.1109/vl.1996.545307

Sigal, M. J., & Chalmers, R. P. (2016). Play it again: Teaching statistics with Monte Carlo simulation. *Journal of Statistics Education*, *24*(3), 136-156. doi: 10.1080/10691898.2016.1246953

Siirtola, H., & Mäkinen, E. (2005). Constructing and reconstructing the reorderable matrix. *Information Visualization*, *4*(1), 32–48. doi: 10.1057/palgrave.ivs.9500086

Silge, J., & Robinson, D. (2017). *Text mining with R* (1st ed.). Sebastopol, CA: O'Reilly Media.

Skrondal, A. (2000). Design and analysis of Monte Carlo experiments. *Multivariate Behavioral*

Research, *35*(2), 137—167. doi: 10.1207/S15327906MBR3502_1

Smiciklas, M. (2012). *The power of infographics*. Seattle, WA: Que Publishing.

Ström, M. (2016). *Design better data tables.* https://planetary.io/blog/design-better-data-tables.

Teetor, P. (2011). *R cookbook*. Sebastopol, CA: O'Reilly Media.

Theus, M., & Urbanek, S. (2009). *Interactive graphics for data analysis*. Boca Raton, FL: Chapman and Hall/CRC.

Tofighi, D., & MacKinnon, D. P. (2016). Monte Carlo confidence intervals for complex functions of indirect effects. *Structural Equation Modeling*, *23*, 194–205. doi: 10.1080/10705511.2015.1057284

Tufte, E. R. (1990). *Envisioning information*. Cheshire, Connecticut: Graphics Press.

Tufte, E. R. (1997). *Visual explanations*. Cheshire, Connecticut: Graphics Press.

Tufte, E. R. (2001). *The visual display of quantitative information*. Cheshire, Connecticut: Graphics Press.

Tufte, E. R. (2006). *Beautiful evidence*. Cheshire, Connecticut: Graphics Press.

Tukey, J. W. (1977). *Exploratory data analysis*. Reading, MA: Addison-Wesley.

Tukey, J. W. (1990). Data-based graphics. *Statistical Science*, *5*(3), 327-339. doi: 10.1214/ss/1177012101

Unwin, A. (2008). Good graphics? In *Handbook of data visualization* (pp. 57–78). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/978-3-540-33037-0_3

VanderPlas, S., & Hofmann, H. (2016). Clusters beat trend!? Testing feature hierarchy in statistical

graphics. *Journal of Computational and Graphical Statistics*, *26*(2), 231–242. doi: 10.1080/10618600.2016.1209116

Wainer, H. (1984). How to display data badly? *American Statistician*, *38*(2), 137–147. doi: 10.1080/00031305.1984.10483186

Wainer, H. (1992, Jun). Understanding graphs and tables. *ETS Research Report Series*, *1992*(1), 4–20. doi: 10.1002/j.2333-8504.1992.tb01443.x

Ware, C. (2004). *Information visualization*. San Francisco, CA: Morgan Kaufmann.

West, S. (2006). Seeing your data. In R. R. Bootzin & P. E. McKnight (Eds.), *Strengthening research methodology* (p. 159-182). Washington, DC: American Psychological Association.

Wickham, H. (2009). *ggplot2* (2nd ed.). New York, NY: Springer.

Wickham, H. (2014a). *Advanced R*. Boca Raton, FL: Chapman and Hall/CRC.

Wickham, H. (2014b). Tidy data. *Journal of Statistical Software*, *59*(10), 1–23. doi: 10.18637/jss.v059.i10

Wickham, H. (2015). *R packages* (1st ed.). Sebastopol, CA: O'Reilly Media.

Wickham, H. (2017). tidyverse: Easily install and load 'tidyverse' packages [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=tidyverse` (R package version 1.1.1)

Wilkinson, L. (1993). Comment: A model for studying display methods of statistical graphics. *Journal of Computational and Graphical Statistics*, *2*(4), 355. doi: 10.2307/1390689

Wilkinson, L. (2005). *The grammar of graphics* (2nd ed.). New York, NY: Springer.

Wilkinson, L., & Friendly, M. (2009). The history of the cluster heat map. *The American Statistician*, *63*(2), 179–184. doi: 10.1198/tas.2009.0033

Williams, P. (2004). *Interactive statistics for the behavioral sciences*. Sunderland, MA: Sinauer Associates.

Wright, K. (2016). lucid: Printing floating point numbers in a human-friendly format [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=lucid` (R package version 1.4)

Xie, Y. (2016a). *bookdown* (1st ed.). Boca Raton, FL: Chapman and Hall/CRC.

Xie, Y. (2016b). DT: A wrapper of the JavaScript library 'DataTables' [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=DT` (R package version 0.2)

Yau, N. (2011). *Visualize this*. Hoboken, New Jersey: Wiley.

Yuan, K.-H., & Chan, W. (2015). Structural equation modeling with unknown population distributions: Ridge generalized least squares. *Structural Equation Modeling: A Multidisciplinary Journal*, *23*(2), 163–179. doi: 10.1080/10705511.2015.1077335

Zhang, J. (1997). The nature of external representations in problem solving. *Cognitive Science*, *21*(2), 179–217. doi: 10.1207/s15516709cog2102_3

Zickar, M. J. (2005). Computational modeling. In F. T. Leong & J. T. Austin (Eds.), *The psychology research handbook* (2nd ed.). Thousand Oaks, CA: Sage Publications.

Appendix A Methods for testing heterogeneity of variance

The study by Brown and Forsythe (1974) is based upon a typical analysis of variance design: Let $x_{ij} = \mu_i + \epsilon_{ij}$ by the jth ($j = i, \ldots, n_i$) observation in the ith group ($i = 1, \ldots, g$) where the means μ_i are neither known nor assumed equal. The ϵ_{ij} are independent and similarly distributed with zero mean and possibly unequal variances. The sample mean ($\bar{x}_i$) and sample variance (s_i^2) are estimated for each group. The researchers then compared the performance of the following statistics for checking for heterogeneous variance:

1. The F-ratio; $F = s_1^2/s_2^2$.

2. Levene's test; let $z_{ij} = |x_{ij} - \bar{x}_i|$. Then perform a one-way ANOVA statistic:

$$W_0 = \frac{\Sigma_i n_i(\bar{z}_i - \bar{z}_{..})^2/(g-1)}{\Sigma_i \Sigma_j (z_{ij} - \bar{z}_{i.})^2/\Sigma_i(n_i - 1)},$$

   where $\bar{z}_i = \Sigma z_{ij}/n_i$ and $\bar{z}_{..} = \Sigma\Sigma z_{ij}/\Sigma n_i$. The critical values of W_0 are obtained from an F-table with$g - 1$ and $\Sigma_i(n_i - 1)$ *df.*

3. A modified Levene's test (W_{50}), where the mean $\bar{x}_i$ is replaced by the median x_1' in forming z_{ij}.

4. A modified Levene's test (W_{10}), where the mean $\bar{x}_i$ is replaced by $\tilde{x}_i$, or the ten percent trimmed mean of the ith group, in forming z_{ij}.

5. Layard's χ^2 test statistic; this is a function of kurtosis. The kurtosis is estimated by pooling

the numerators and denominators of the individual estimates of kurtosis for each group:

$$S' = \Sigma(n_i - 1)\left[\log s_i^2 - \frac{\Sigma(n_i - 1)\log s_i^2}{\Sigma(n_i - 1)}\right]^2 / \hat{\tau^2},$$

where

$$\tau^2 = 2 + \{1 - (g/\Sigma n_i)\}\hat{\gamma},$$

and

$$\hat{\gamma} = \frac{(\Sigma n_i)\Sigma\Sigma(x_{ij} - \bar{x}_i)^4}{\{\Sigma\Sigma(x_{ij} - \bar{x}_i)^2\}^2} - 3$$

is the estimate of the kurtosis. S' is asymptotically distributed as a χ^2 variable with $(g - 1)$ *df*.

6. Miller's jackknife procedure; a one-way analysis of variance procedure based upon:

$$s_{i(j)}^2 = \left(\frac{1}{n_i - 2}\right)\Sigma_{k \neq j}(x_{ik} - \bar{x}_{i(j)})^2,$$

where$\bar{x}_{i(j)} = \left(\frac{1}{n_i - 1}\right)\Sigma_{k \neq j} x_{ik}$ for $(i = 1, \ldots, g)$ and let $U_{ij} = n_i \log s_i^2 - (n_i - 1)\log s_{i(j)}^2$. Then, compute an F-statistic on the U_{ij},,

$$J = \frac{\Sigma n_i(\bar{U}_{i.} - \bar{U}_{..})^2/(g - 1)}{\Sigma\Sigma(U_{ij} - \bar{U}_{i.})^2/\Sigma(n_i - 1)},$$

and test H_0 by approximating the null distribution of J by the $F_{g-1,\Sigma(n_{i-1})}$ distribution.

Appendix B Brown and Forsythe (1974) replication

The following script replicates the main findings from Brown and Forsythe (1974). This is a modified version of code written by Chalmers (2017), and relies on the `SimDesign` package (Chalmers, 2016; Sigal & Chalmers, 2016). Results from this simulation are included in the `SimDisplay` package, and can be loaded using `data(Brown1974)`.

```
# Simulation from the article:
# Robust Tests for the Equality of Variances
# Author(s): Morton B. Brown and Alan B. Forsythe
# Source: Journal of the American Statistical Association,
#         Vol. 69, No. 346 (Jun., 1974), pp. 364-367
# Published by: American Statistical Association

library(SimDesign)

#-----------------------------------------------------------------
# Create and check the simulation design skeleton

# Define the design conditions
distributions <- c('Gaussian', 't4', 'Chi4')
sample_size <- c("40/40", "10/10", "20/40", "10/20")
var_ratio <- c(1, 2, 4, 1/2, 1/4)

# Design Matrix
Design <- expand.grid(var_ratio=var_ratio,
                      sample_size=sample_size,
                      distribution=distributions)
Design <- Design[,c(3,2,1)] # Mimic table structure from article

# Remove redundant rows (variance ratio 1:4 is the same condition as 4:1
# for equal sample size groups)
Design <- subset(Design, !((sample_size == "40/40" & var_ratio < 1) |
                    (sample_size == "10/10" & var_ratio < 1)))

print(Design) # 48 rows, matched to Brown & Forsythe

#-----------------------------------------------------------------
# Custom Functions for simulation
# Brown and Forsythe used two procedures not generally available in R.
# These pertain to the jackknife test and Layard's chi-square.

Jackknife_vartest <- function(DV, group){
  nms <- unique(group)
  g <- length(nms)
  Ns <- s2 <- Ui. <- numeric(g)
  Uij <- vector('list', g)
  for(i in 1:g){
```

```
    pick <- group == nms[i]
    sub <- subset(DV, pick)
    Ns[i] <- length(sub)
    s2[i] <- var(sub)
    sij2 <- numeric(Ns[i])
    for(j in 1:length(sub))
      sij2[j] <- var(sub[-j])
    Uij[[i]] <- Ns[i] * log(s2[i]) - (Ns[i]-1) * log(sij2)
    Ui.[i] <- mean(Uij[[i]])
  }
  U.. <- mean(do.call(c, Uij))
  num <- sum(Ns * (Ui. - U..)^2) / (g-1)
  den <- 0
  for(i in 1:g)
    den <- den + sum((Uij[[i]] - Ui.[i])^2) / sum(Ns-1)
  J <- num/den
  df1 <- g-1; df2 <- sum(Ns-1)
  p.value <- pf(J, df1, df2, lower.tail = FALSE)
  return(list(J=J, df1=df1, df2=df2, p.value=p.value))
}

Layard_vartest <- function(DV, group){
  nms <- unique(group)
  Ns <- table(group)
  N <- sum(Ns)
  g <- length(nms)
  log_s2 <- numeric(g)
  deviation <- numeric(N)
  for(i in 1:g){
    pick <- group == nms[i]
    sub <- subset(DV, pick)
    xbar <- mean(sub)
    deviation[pick] <- sub - xbar
    log_s2[i] <- log(var(sub))
  }
  gamma <- N * sum(deviation^4) / (sum(deviation^2)^2) - 3
  tau2 <- 2 + (1 - g/N) * gamma
  S <- sum( (Ns-1) * (log_s2 - sum((Ns-1) * log_s2)/sum(Ns-1))^2) / tau2
  df <- g - 1
  p.value <- pchisq(S, df, lower.tail=FALSE)
  return(list(S=S, df=df, p.value=p.value))
}

#-----------------------------------------------------------------
# Set-up the simulation

# The following code will generate the appropriate data for each condition.
Generate <- function(condition, fixed_objects = NULL) {
  N <- with(condition, sample_size)   # Sample size per group
  v <- with(condition, var_ratio)     # Variance ratio for condition
  sd1 <- ifelse(v <= 1, 1, sqrt(v))   # Standard deviation for group 1
  sd2 <- ifelse(v <= 1, sqrt(1/v), 1) # Standard deviation for group 2

  if(N == "40/40"){ N1 <- 40; N2 <- 40
  } else if(N == "20/40"){ N1 <- 20; N2 <- 40
```

```
  } else if(N == "10/10"){ N1 <- 10; N2 <- 10
  } else if(N == "10/20"){ N1 <- 10; N2 <- 20
  }

  # Depending on the distribution, utilize a different generating function.
  # Note the second groups variance is minimized/maximized based
  # upon the value of v (standard deviation condition).
  dat <- switch(as.character(condition$distribution),
                Gaussian = c(rnorm(N1, sd = sd1), rnorm(N2, sd = sd2)),
                t4 = c(rt(N1, df = 4), rt(N2, df = 4) * sqrt(v)),
                Chi4 = c(rchisq(N1, df = 4), rchisq(N2, df = 4) * sqrt(v)))
  dat <- data.frame(DV=dat, group=c(rep('G1', N1), rep('G2', N2)))
  dat
}

# The following code will analyze the data for each condition.
Analyse <- function(condition, dat, fixed_objects = NULL) {
  DV <- dat$DV
  group <- dat$group
  # Run the six different variance tests:
  F_test <- var.test(DV ~ group)$p.value
  Jackknife <- Jackknife_vartest(DV, group)$p.value
  Layard <- Layard_vartest(DV, group)$p.value
  Levene <- levene.test(DV, group=group, location = 'mean')$p.value
  W10 <- levene.test(DV, group=group, location = 'trim.mean', trim.alpha = .1)$p.value
  W50 <- levene.test(DV, group=group, location = 'median')$p.value
  # Return p-values from each:
  ret <- c(F=F_test, Jackknife=Jackknife, Layard=Layard,
           Levene=Levene, W10=W10, W50=W50)
  ret
}

# This code will return the EDRs for each cell.
Summarise <- function(condition, results, fixed_objects = NULL) {
  ret <- EDR(results, alpha = .05) # Can be changed to .01 if desired.
  ret
}

#--------------------------------------------------------------------
# Run the MCS

# The following takes the above code and runs the simulation. Output is saved
# to disk. The package lawstat is loaded for the levene.test() function.
# Seed values are set for each cell to ensure reproducibility.
Brown1974 <- runSimulation(design=Design, replications=1000, parallel=TRUE,
                           generate=Generate, analyse=Analyse, summarise=Summarise,
                           packages='lawstat', seed = c(1:48))
View(Brown1974)
save(Brown1974, file = "Brown1974.rda")
```

Appendix C Hallgren (2013) replication

The following script replicates and expands upon the main findings from Hallgren (2013). The original article utilized many nested for-loops to conduct this simulation, and these have been replaced with functions from `SimDesign` (Chalmers, 2016; Sigal & Chalmers, 2016). Results from this simulation are included in the `SimDisplay` package, and can be loaded using `data(Hallgren2013)`.

```
# Simulation from the article: Conducting Simulation Studies in the R Programming Environment
# Author(s): Kevin Hallgren
# Source: Tutorials in Quantitative Methods for Psychology,
#         2013, Vol. 9, No. 2, pp. 43-60.

library(SimDesign)

# Custom fucntion for obtaining p-values from the Sobel test
sobel_test_p <- function(X, M, Y){
    M_X = lm(M ~ X)
    Y_XM = lm(Y ~ X + M)
    a = coefficients(M_X)[2]
    b = coefficients(Y_XM)[3]
    stdera = summary(M_X)$coefficients[2,2]
    stderb = summary(Y_XM)$coefficients[3,2]
    sobelz = unname(a*b / sqrt(b^2 * stdera^2 + a^2 * stderb^2))
    ret <- c(sobelz=sobelz,
             p=pnorm(abs(sobelz), lower.tail=FALSE)*2)
    ret
}

Design <- expand.grid(N = c(30, 100, 300),
                      a = c(-.3, 0, .3),
                      b = c(-.3, 0, .3),
                      cp = c(-.2, 0, .2))

Generate <- function(condition, fixed_objects = NULL) {
    Attach(condition)
    X <- rnorm(N, 0, 1)
    M <- a*X + rnorm(N, 0, 1)
    Y <- cp*X + b*M + rnorm(N, 0, 1)
    dat <- data.frame(X, M, Y)
    dat
}

Analyse <- function(condition, dat, fixed_objects = NULL) {
    Attach(dat)
    XMY <- sobel_test_p(X, M, Y)
    XYM <- sobel_test_p(X, Y, M)
    ret <- c(XMY=XMY, XYM=XYM)
    ret
```

```
}

Summarise <- function(condition, results, fixed_objects = NULL) {
    ret <- EDR(results[,c('XMY.p', 'XYM.p')], alpha = .05)
    ret
}

Hallgren2013 <- runSimulation(design=Design, generate=Generate,
                              analyse=Analyse, summarise=Summarise,
                              replications=1000, parallel = TRUE,
                              seed = c(1:81))
View(Hallgren2013)
save(Hallgren2013, file = "Hallgren2013.rda")
```

Appendix D Example power curve simulation

The following script generates the data used in the power curve display in Section 3.7.

```
library(SimDesign)

Design <- expand.grid(mean_diff = seq(0, 2, by = .4),
                      sample_size = c(5, 10, 20, 30),
                      ratio = c(1, 2))

Generate <- function(condition, fixed_objects = NULL) {
  if (condition$ratio == 1) {
    ret <- with(condition, data.frame(
    value = c(rnorm(sample_size), rnorm(sample_size, mean = mean_diff)),
    group = c(rep("Group1", sample_size), rep("Group2", sample_size))))
  } else {
    ret <- with(condition, data.frame(
      value = c(rnorm(sample_size), rnorm(sample_size*2, mean = mean_diff)),
      group = c(rep("Group1", sample_size), rep("Group2", sample_size*2))))
  }
  ret
}

Analyse <- function(condition, dat, fixed_objects = NULL) {
  mod <- aov(value ~ group, data = dat)
  ret <- c(p = summary(mod)[[1]]$`Pr(>F)`[1])
  ret
}

Summarise <- function(condition, results, fixed_objects = NULL) {
  ret <- EDR(results, alpha = .05)
  ret
}

results <- runSimulation(Design, replications=1000,
                         verbose=FALSE, parallel=TRUE,
                         generate=Generate, analyse=Analyse, summarise=Summarise,
                         edit='none')
```