A Comparison of Variable Selection Methods for Modeling Human Judgment

A dissertation presented to

the faculty of

the College of Arts and Sciences of Ohio University

In partial fulfillment

of the requirements for the degree

Doctor of Philosophy

Kristina A. Carter

May 2019

This dissertation titled

A Comparison of Variable Selection Methods for Modeling Human Judgment

by

KRISTINA A. CARTER

has been approved for

the Department of Psychology

and the College of Arts and Sciences by

Claudia González-Vallejo

Professor of Psychology

Joseph Shields

Interim Dean, College of Arts and Sciences

ABSTRACT

CARTER, KRISTINA A., Ph.D., May 2019, Experimental Psychology

A Comparison of Variable Selection Methods for Modeling Human Judgment

Director of Dissertation: Claudia González-Vallejo

First introduced by Brunswik (1952), judgment analysis, the statistical modeling of human judgment, has greatly contributed to psychology's understanding of human judgment. Judgment analysis has the potential to more broadly impact psychological and behavioral research yet complexities of its individual-level approach to modeling lead to challenges in its application. One way to increase the accessibility of judgment analysis would be to employ variable selection methods that decrease the predictor pool to allow for research designs that are representative, but do not require large numbers of observations from single individuals. The following research tests three modeling approaches: stepwise regression, all subsets method, and random forest method, to compare each methods approach to variable selection. Study One, which applied each of these methods to empirical judgment analysis data, indicated that the random forest method has lower goodness-of-fit than the all-subsets method, but greater generalizability than both the all subsets and stepwise regression methods. Study Two found that in two out of four cases, random forest was better than the all-subsets method and in three cases better than the stepwise regression approach in including relevant predictors in the final model. Study Three found that the random forest method was less susceptible to multicollinearity, more likely than either other method to exclude irrelevant predictors even when they were correlated with relevant ones. Overall the random forest approach shows great promise and its use may facilitate broader applications of judgment analysis.

DEDICATION

To Alex, with gratitude for all the variables that brought us together, though it may never be possible to accurately model them.

ACKNOWLEDGMENTS

I would like to express my appreciation and gratitude to my advisor, Dr. Claudia González-Vallejo, for her guidance, mentorship, and support throughout the development of this document and my research as whole. I also thank my committee members, Drs. Bruce Carlson, Jeffrey Vancouver, Lonnie Welch, and Yuchun Zhou for their time and invaluable insight which improved the quality of this work.

Special thanks to my partner, Alex Ferguson. Thank you for all the drafts you've read and support you've given me. This dissertation wouldn't exist without you. To Sid Ferguson, thank you for the interest, enthusiasm, and confidence you expressed in my educational pursuits. I miss you.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1: INTRODUCTION

Understanding individuals' judgment and decision-making processes, and the factors affecting them, has long been recognized as essential to predicting a broad set of phenomena such as risky decisions, financial investments, environmental sustainability, and market variability (Fischhoff, 2010; Koehler & Harvey, 2004; Kiker, Bridges, Varghese, Seager & Linkov, 2005). Researchers in judgment and decision-making have adopted several approaches to human judgment, some focused on documenting the characteristics of normative or ideal decision-making, others on explaining the mechanisms through which human judgment falls short of normative or rational decision-making (Baron, 2004). Judgment analysis is an approach to human judgment that conceives of judgment as an interaction between an organism and its environment and focuses on understanding both these aspects of judgment (Goldstein, 2004). In judgment analysis, a model for human judgment is extracted statistically by relating variation in an individual's judgment with variation in the information accessible to the individual. That is, a statistical model is used to extract models that reflect how individuals used information in making judgments. The aim of the current work is to compare the use of different modeling techniques in the context of judgment analysis to provide recommendations to improve implementation and facilitate greater application of judgment analysis.

Advantages to the judgment analysis approach include: avoiding biases associated with self-report and allowing for insight into a process that has been demonstrated to evade introspection (Hartwig & Bond, 2011; Nisbett & Wilson, 1977; Tomassetti, Dalal, & Kaplan, 2016). Judgment analysis has been used to provide insight into areas as

diverse as hiring practices, medical diagnosis, conflict resolution, meteorological forecast, and religious meaning (e.g., Graves & Karren, 1992; Hammond & Adelman, 1976; Pargament, Sullivan, Balzer, Van Haitsma, & Raymark, 1995; Sorum et al., 2002; Stewart, Moninger, Grassia, Brady, & Merrem, 1989). Although there are many advantages to the judgment analysis approach, extracting statistical models to represent individual's judgment processes is not without its challenges. Factors that can complicate the extraction of a model that accurately represents an individual's judgment include factors related to the context in which individuals make their decision. For instance, a judgment context that includes many, potentially correlated, pieces of information complicate the extraction of an accurate model. Additional factors that can make judgment analysis challenging includes the modeling method that is employed. Various methods for extracting individual judgment models have been investigated but most judgment researchers employ a multiple linear regression approach (Goldstein, 2004; Slovic & Lichtenstein, 1971). Linear models have been demonstrated to be helpful in modeling the judgment process in previous research, though there are indications that in many cases human judgment may follow a non-linear process (Gigerenzer, 1996; Kim, Yang, & Kim, 2008; Zelany, 1976). If or when the human judgment process is non-linear, a linear model may be a poor representation of the underlying process, despite its capacity to extract a model (e.g., Gigerenzer, 1996; Kim, Yang, & Kim, 2008).

The current research focused on examining variable selection methods using a judgment analysis framework. Three studies, using simulated and empirical data, applied three methods of variable selection to extract individual-level models of judgments of nutrition of packaged foods. The purpose of this research is to document both the

complexities in modeling empirical judgment data using variable selection techniques and to compare how these techniques shape statistical results and practical implications. This research is expected to contribute to judgment and decision-making research by evaluating the comparative advantages of applying different variable selection techniques in judgment analysis. By identifying effective variable selection methods, the current study can facilitate further applications of judgment analysis by enabling researchers to reduce large predictor pools, thus allowing for the examination of predictor-dense judgment tasks and reducing the number of necessary individual observations, which can often be constraining. This study's findings also have implications for research beyond the specific area of judgment analysis by increasing our understanding of the comparative accuracy, efficiency, and interpretability of results from three variable selection methods: stepwise regression, all subsets variable selection, and random forest variable selection. Based on the findings the author recommends strategies for extending the application of judgment analysis in academic and applied research settings alike.

Challenges of Statistical Modeling

As judgment analysis relies closely on statistical modeling, challenges inherent in statistical modeling are critical to judgment analysis. Regardless of the realm of research, statistical modeling has two primary goals: predicting data and generalizing those predictions (James, Witten, Hastie, & Tibshirani, 2013; Yarkoni & Westfall, 2017). The first goal of predicting data is achieved by extracting a model that accounts for a proportion of variance in the variable of interest. Extracting such a model indicates the researcher has found a relationship between the phenomenon of interest and exogenous factors. Once a model is extracted that explains variation in the available data, the

researcher can determine whether the generated model satisfies the second goal of generalizing to new observations of the phenomenon. For instance, a nutritionist may be interested in determining how an individual uses the nutritional information to form a judgment regarding the nutritional quality of food choices. First, the researcher is interested in extracting the judgment model used by the consumer to determine whether and in which ways nutritional variables impact judgment. Determining a predictive model in this instance would involve determining how the variance in the nutritional information variables is related to the variance in an individual's judgment. If a statistically significant proportion of the variance in an individual's judgment can be predicted by the variance in nutritional information, then the researcher has succeeded in extracting a predictive model.

Whether the extracted predictive model applies to future instances pertains to the second goal of modeling: how well this predictive model generalizes to new instances of judgment formation and expression. If the extracted model correctly represents the underlying relationship between the involved variables, it is anticipated that it will generalize to future instances of the variable of interest (Babyak, 2004; James et al., 2013). For instance, if a model extracted from a consumer's judgments of 25 cereals shows a negative relationship between a consumer's judgment of quality and quantities of calories and sugars, one can expect that, if that model is accurate it will generalize to the judgments of the same consumer on new cereals. That is, among the new cereals, judgment quality should also be negatively related to quantities of calories and sugars.

The ability of a model to generalize to future instances is an important aspect of modeling as it validates the model as useful in contexts where observations of the

phenomenon of interest are unavailable. Such a model provides means of greater knowledge and control over a phenomenon, offering opportunities for intervention.

Take an example of forecasting recidivism. Barnes and Hyatt (2012) described the use of statistical modeling to classify criminal offenders according to their risk for future offenses. The researchers first used archival offense data to extract statistical models for criminal offense and then used these predictive models to estimate the number and severity of future crimes that offenders newly on probation might commit (Barnes & Hyatt, 2012). A test of the models' cross-generalization to a new dataset indicated that, across risk levels, the models all had a classification accuracy of approximately 60%, compared to an average of 64% accuracy estimated within the data from which the model was extracted (Barnes & Hyatt, 2012). Notably, the models were far more accurate at predicting low-risk classification (accuracy ranged from 72.6%-74.6%) than in predicting high risk classification (accuracies ranged from 20.9%-22.3%). This statistical tool's predicted outcomes are currently used by the Philadelphia's Adult Probation and Parole Department to assign probationers to high, moderate, or low risk categories for probation-supervision (Ritter, 2013). Probationer supervision is based on the risk level with probationers in higher risk categories given greater supervision than probationers in lower risk categories (Ritter, 2013).

Model cross-generalization is also important for another, more implicit reason: high cross-generalization indicates that the understanding of the phenomenon extracted from the initial data is likely correct (Shmueli, 2010). That is, if a predictive model can cross-generalize, this indicates that the extracted model of the relationship between the variables of interest is a reasonable depiction of the underlying phenomenological

relationships rather than artifact of a specific dataset that provides no information about the broader world (Babyak, 2004; Shmueli, 2010).

The dual predictive and cross-generalization purpose of statistical modeling requires achieving a balance in model-fitting (Brighton & Gigerenzer, 2015). The suitability of a model is often measured by how well the model fits the data used to derive the model, that is, the model's goodness-of-fit. Goodness-of-fit (GOF) refers to the extent to which a model fits the particular sample of observed data from which the model was generated. GOF is a measure of the predictive value of the model. The discrepancy between the predictions of the model compared to the actual observed data are compared and measured in a variety of ways to provide an estimate of the model's GOF (Pitt & Myung, 2002). The GOF is contrasted with the generalizability of the model, which is the extent to which a model fits all data samples generated by the same underlying phenomenon, as opposed to the one particular sample from which the model was generated.

The GOF of a model can be improved by having a highly flexible model. A complex model is one that can fit many different data patterns; because a complex model can generate many distinct probability distributions by varying many parameters over their entire range, the more complex a model is the greater its flexibility. The higher a model's flexibility, the more likely it will fit a particular dataset and yield a high GOF. However, high flexibility is negatively related to generalizability (James et al., 2013; Pitt & Myung, 2002). Given a dataset, a model can be made flexible enough that it captures every possible data point in the set. Such a complex model would have a high GOF, as

there would be no discrepancy between the predicted and observed outcome, but it would have very low generalizability (Pitt & Myung, 2002).

Such a model, as described above, is sometimes described as "overfitting" the data. That is, in addition to fitting the overall trends in the dataset, the model fits slight variations from the overall trends that are due to random error (Pitt & Myung, 2002). An overfit model contains more parameters than can be justified given the data (James et al., 2013). Furthermore, because an overfit model corresponds so directly to the specific data from which it was estimated, it would vary dramatically if it were estimated using a new dataset. An overfit model is overly sensitive to the idiosyncratic nature of the data from which it is derived, causing it to capture the noise present in the data when an alternative, simpler model is available. When applied to a new sample of data, an overfit model fails to accurately predict the new sample data; its high flexibility results in a failure to cross-generalize to new data that contains different patterns of random error, therefore, not satisfying one of the major goals of statistical modeling. Ideally, a model should be both predictive by having a high GOF to the data sample from which it was generated, and robust such that it resists random error and variation and thus is able to generalize to new data samples (James et al., 2013).

Given that determining a predictive model from an initial sample is the first step in statistical modeling, researchers often overemphasize the goal of finding a predictive model over the goal of finding a generalizable model (Pitt, Kim, & Myung, 2003; Yarkoni & Westfall, 2017). This is particularly true in behavioral research areas, such as psychology, where theoretical foundations are lacking or poorly established (Meehl, 1967; Klein, 2014; Yarkoni & Westfall, 2017). As technology facilitates the collection of

enormous amounts of data, there is an even greater temptation to add all factors that might aid in creating a highly predictive model with the hope of forming a clear understanding of the data (Guyon & Elisseeff, 2003). Such complex models can explain a substantial proportion of the sample variance, but often fail to generalize to new data. Such a failure indicates that the resulting extracted model provides an explanation for the idiosyncrasies of the sample from which it was extracted, but not of the underlying phenomenon the researcher was attempting to explain. This tendency to overemphasize a high GOF, at the cost of making a model that is inextricably tied to the data from which it originated, results in models that appear to be valid but are overfit (Babyak, 2004; Brighton & Gigerenzer, 2015).

One means for guarding against overfitting is to ensure that an equal or comparable GOF, or predictive power, of the model cannot be obtained by a simpler model with fewer predictor variables (Hawkins, 2004). Between two statistical models that perform smiliarly, the simpler model (i.e., which has fewer predictors) is preferable. However, the choice of which predictors variables to include in a final model is not always clear cut. Ideally, variable selection—the process of choosing which predictors to include in a final model—is guided by theoretical reasons. Despite this, there are cases where there is no clear theoretical indication of which variables should be important to a model (Plonsky, Erev, Hazan, & Tenneholtz, 2017). Moreover, variable selection can be prohibitive in areas in which models are constructed on an individual basis, particularly in cases when the sample and predictor pools are large. In such areas, automated variable selection methods—which determine predictor selection based solely on statistical

grounds—can be helpful in reducing the predictor pool and helping researchers extract more appropriate models that have high GOF and generalizability.

Judgment Analysis

Judgment analysis, as previously introduced, in an application of statistical modeling in behavioral sciences that involves extracting a statistical model to explain the relationship between human judgment and information in the environment the judge may use. A typical approach to statistical modeling in behavioral research involves measuring groups of individuals and drawing averages from an aggregated sample. Within mainstream psychology this aggregating approach has been dubbed the nomothetic approach (Lamiell, 1998). Judgment analysis, also known as policy capturing, social judgment theory, and lens model analysis, applies statistical modeling to human judgment behavior on an individual level (Brehmer, 1988; Stewart, 1988). That is, using an idiographic approach[1]. Judgment analysis involves extracting models of judgment of each individual making judgments; that is, constructing statistical models that reflect the process of judgment formation for a single individual. Judgment analysis is not exclusively concerned with the individual though; many studies in judgment analysis also seek to derive generalizable information about human judgment on a population level (Goldstein, 2004). However, judgment analysis is always idiographic, beginning by applying modeling techniques to multiple judgments of a single individual and reserving

[1] As Lamiell (1998) describes, the interpretation of nomothetic and idiographic as science by aggregate and individual respectively is not in keeping with Windelband's 1894 (translated in Windelband, 1998) conception in coining the terms. Nonetheless, as the representation of this terminology has come to be represent this methodological contrast, this paper defines these terms in accordance with their mainstream misconception.

any across-person analyses to be conducted on the resulting individual outcomes (Hammond, McClelland & Mumpower, 1980).

The conceptual foundation of judgment analysis is Brunswik's (1952) probabilistic functionalism, which posits that the relationship between distal variables being judged, and the proximal cues (i.e., variables) used to judge them are, at best, probabilistic. Thus, in making any given judgment, humans cannot expect to find cues that are perfectly dependable. Rather, the relationship between those cues and the variable being judged should be described probabilistically and, therefore, the cognitive system employing the environmental cues should also be described statistically (e.g., Principle of Parallel Concepts, Hammond, Stewart, Brehmer, & Steinmann, 1975; Brehmer, 1988). The statistical relationship between a judgment and criterion variable described by probabilistic functionalism is graphically depicted by the lens model, shown in Figure 1.

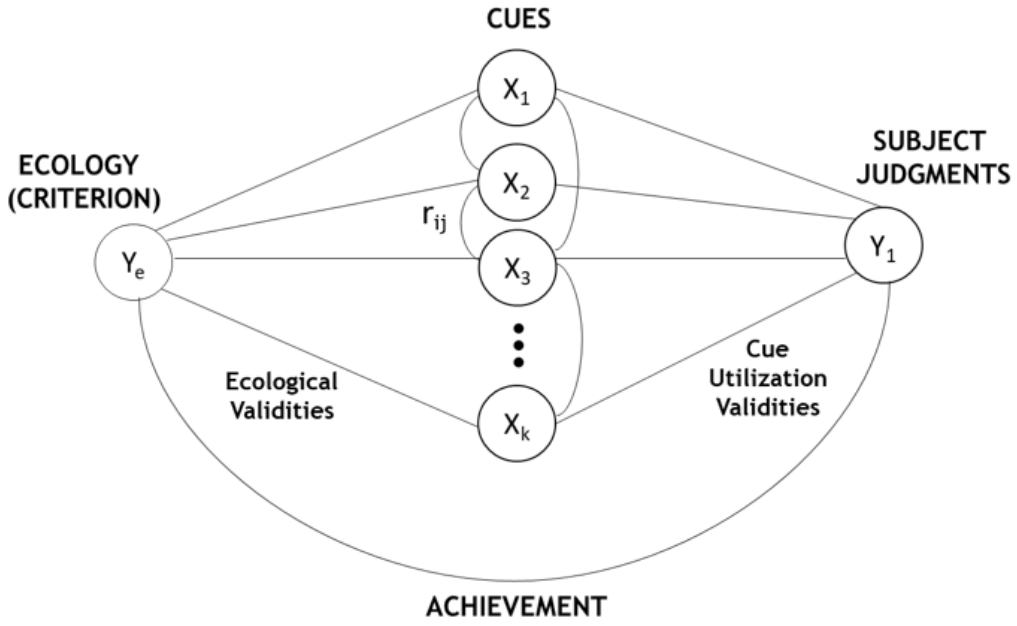


Figure 1. Lens model classic double-system design (Cooksey, 1996, p. 61).

Judgment analysis has multivariate and hierarchical extensions, but it typically involves multiple linear regression to model the relationship between a final judgment and the environmental variables, or cues, relevant to the judgment task or criterion (Cooksey, 1996). This relationship can be seen on the right side of Figure 1. Figure 1 also depicts the relationships between the environmental cues and the criterion which are also statistical. Indeed, two key advantages of the lens model is that it can be used to capture the environmental system as well as the cognitive system, and the same method is used to describe both systems (Brehmer, 1988). This allows for explanatory consistency, which is important because it allows for the comparison between an individual's judgment and that of a "gold standard" criterion, be it an objective measure or an expert's judgment (Dhami & Harries, 2001). When measures for both the individual and the criterion can be obtained, lens model can be further used to describe the relationship between the two. The relationship between human judgment and the criterion, called achievement, is a measure of the individual's judgment accuracy (Stewart, Roebber, & Bosart, 1997).

The conceptualization of the lens model gave rise to statistical indices to quantitatively describe the relationships between the various components of the lens model (Cooksey, 1996). The lens model equation (LME) decomposes the variance achievement—that is, the correlation between the subject's judgment and the criterion—into several components (Hammond, Hursch & Todd, 1964). With the lens model equation, statistical models of the relationships of the criterion (Y_e) and the judgment (Y_1) are first developed.

Equation 1:

$$Y_e = M_{Y_e.X}(X_1, X_2, \ldots, X_n) + E_{Y_e.X}$$

$$Y_1 = M_{Y_1.X}(X_1, X_2, \ldots, X_n) + E_{Y_1.X}$$

In the above equation, Y_e is a *n* x 1 vector of observations of the criterion whereas Y_1 is an *n* x 1 vector of observations of person judgments, X_i are the cues (i = 1 to n) or variables in the environment being judged, $M_{Y_e.X}$ and $M_{Y_1.X}$ represent the weights that describe the relations between the cues and the criterion, as well as the cues and the judgment respectively, and the E's represent the residuals of the models and are not linearly related to the weights.

As seen in Equation 2, the partitioning of the judgment and the criterion is used to derive a two-component formulation (Stewart, 1976).

Equation 2:

$$r_{Y_1 Y_e} = R_{Y_e.X} G R_{Y_1.X} + C\sqrt{1 - R^2_{Y_e.X}}\sqrt{1 - R^2_{Y_1.X}}$$

$$r_{YO} = R_{O.X}\, G\, R_{Y.X} + C\sqrt{1 - R^2_{O.X}}\,\sqrt{1 - R^2_{Y.X}}$$

In Equation 2, $R_{Y_e.X}$, a measure of environmental predictability, is the correlation between Y_e and $M_{Y_e.X}$; G is a measure of the agreement between the criterion weights of the cues and the judgment weights of the cues, determined by the correlation between $M_{Y_e.X}$ and $M_{Y_1.X}$; $R^2_{Y_1.X}$, a measure of the consistency of the weighting of the cues by the judge, is the correlation between Y_1 and $M_{Y_1.X}$; C is a measure of the correlation between the variation of the target and the judge that is unaccounted for by the models, it is calculated as the correlation between $E_{Y_1.X}$ and $E_{Y_e.X}$. If the weighted cues capture the shared systematic variance, then C should be near zero. That is, the errors of Y_1 and Y_e

should not be linearly related to each other if the weighted cues capture the systematic variation in Y_l and Y_e. Significant C's indicate that there is shared variance that is not accounted for by the cues, meaning there is systematic variation in Y_l and Y_e that is not accounted for by the weighted cues. High values of C indicate the presence of effects not described by the judgment policies, however, low values of C do not indicate the absence of systematic variation in the residuals (Stewart, 1988). Rather, low C values indicate the absence of *shared* systematic variation between Y_l and Y_e. Low C values could be low because both Y_l and Y_e are error variation, or because one or both contain systematic nonlinear components which are uncorrelated (Stewart, 1988).

Statistical Challenges in Judgment Analysis

Due to the widespread use of regression methods in applying judgment analysis, judgment analysis applications share the characteristics of aggregate research designs that rely on regression such as assumptions of linearity and underlying additivity, sensitivity to multicollinearity, and challenges in producing stable statistical estimates (James et al., 2013; Karelaia & Hogarth, 2008). However, some of those characteristics are uniquely manifested due to the individual nature of the lens model (Karelaia & Hogarth, 2008). Judgment analysis faces unique challenges in two areas related to how model predictors affect the stability of statistical estimates: predictor inter-correlation and sample size, or number of observations.

In many studies, the impact of predictor inter-correlation may be reduced by limiting the included variables based on theoretical grounds or the primary research interest. In judgment analysis, the inter-correlation of predictors is seen as a key characteristic of judgment formation. This is because in a real environment many cues

may be related to a single criterion in similar ways, allowing individuals multiple means of attaining the same accurate end, a characteristic termed vicarious functioning by Brunswik (Brunswik, 1952). That is, cue inter-correlation is a key aspect of judgment formation as it allows humans using different cues to arrive at the same conclusion (Brunswik, 1952). This vicarious functioning capacity depends upon both the ecological validity of the cues and the inter-correlation among them (Stewart, 2001). Vicarious functioning recognizes that humans may base the same judgments on various reasonable cues according to what is allowed by the environment rather than on certain, ideal cues. Importantly, judgment analysis is interested in observing, not controlling, the vicarious functioning present in human judgment behavior making the presence of inter-correlated cues a necessary feature of judgment analysis (Cooksey, 1996).

Though the inter-correlation between cues is key to understanding judgment formation in real-world environments, it presents challenges in interpreting the relationship between cues and judgment. When cues are highly correlated, the linear regression measures helpful for understanding the relative importance of the different cues are subject to estimation errors. Furthermore, as the effects of one cue are inseparable from those that covary with it, high cue inter-correlation may make the meaning of the cue weights impossible to clearly interpret (Stewart, 1988).

In judgment analysis, as in any statistical analysis, the number of observed cases must be large enough to produce stable statistical estimates. Judgment analysis has the added challenge of having enough cases that the judgment task is representative of the actual task environment, but not too many cases that a participant is unable to reasonably complete the task. Sampling for idiographic modeling involves multiple observations of a

single individual and diverse sampling of the environment of the judgment task. For instance, to enable the extraction of an idiographic model for the relationship between nutrient information and nutritional judgment, a single individual must judge multiple food products. This results in a burden, not solely on the researcher to obtain a large sample of judgments, but on the participants, whose individual judgments make up the sample. A complex judgment task that contains many cues may ultimately require too many observations for an individual to reasonably complete without debilitating fatigue (Cooksey, 1996). Although completing the task over multiple time points may be an option, such a design factor would increase the likelihood of attrition (Girden, 1992). Thus, obtaining a sufficiently large number of observations is necessary to enable the extraction of an accurate judgment model, but methods for increasing observations may result in observations that do not adequately reflect individual's judgments.

The task environment under consideration is key in determining both the required number of cases per predictor and the cue inter-correlations among those cases. Although researchers may have some freedom as to the scope of relevant cues they wish to examine, these predictors are largely determined by the task environment itself. For instance, a judgment task involving packaged food products may include variables as diverse as the color of the product’s packaging and its shelf placement in a store. In addition, the nutrient use of a consumer may, in principle, include all nutrients listed on the package plus ingredients. Importantly, relevant cues in a representative environment are not freely determined and should reflect the actual judgment task (Gibson & Hobson, 1983).

Implementing Variable Selection Methods in Judgment Analysis

Adequate theoretical reasons for excluding potential predictors from a judgment analysis task are often lacking in judgment analysis, particularly when the task is novel or when little research has investigated it systematically. In lieu of theoretical reasons for reducing a predictor pool, variable selection methods within an analysis can be utilized. Thus, a common method for addressing an excess of predictors compared to observations is to reduce the predictors that are retained in a final statistical model of judgment based on some reasonable standard for inclusion. Such variable selection methods have been used in empirical applications of judgment analysis, but their use has not been systematically examined nor have different methods of variable selection been formally compared. Commonly, judgment analysis applications incorporate full regression models that contain all possible cues. There are two notable characteristics of judgment analysis applications that may explain why variable selection methods have been so far overlooked in formal and empirical development of the lens model: cue number and overall sample size. These characteristics are outlined in greater detail below.

Cue Number

A limited number of cues in lens model studies may be one reason that variable selection methods in this area has not been explored. In many studies, judgment analysis tasks include few predictor variables, which eliminates the need for variable selection methods. In a review of 249 individual studies across 86 articles, Karelaia and Hogarth (2008) reported 149 task environments (60%) as having two or three cues, and 99 as having more than three. Kaufmann, Reips, and Whittmann (2013) examination of lens model studies reported more specific cue number data. They found that out of 44

individual task environments for which the cue number was reported, 24 tasks had fewer than 10 cues, 18 had 10 or more cues but fewer than 25, and 2 had more than 25, one of which was a definite outlier with 64 cues. Given that many lens model applications have a small number of cues, many observations relative to predictors would not be difficult to obtain. That is, with few cues, sufficiently large numbers of observations can be obtained that modeling techniques will produce stable estimates. As this has traditionally been the case, variable selections methods to decrease the cue pool, though still useful, have not been necessary, which could explain the lack of research done in this area.

Sample Size

A second possible explanation for the lack of attention given to variable selection methods in judgment analysis is the number of individuals common to judgment analysis applications. Often, judgment analysis is used descriptively; the final goal is to describe and understand the judgment process of specific individuals (Karren & Barringer, 2002). In other cases, judgment analysis is used to reconcile the cue use of different individuals or groups of individuals, or to generate a transparent and reliable weighting system for a decision-maker (Goldstein, 2004; Hammond & Adelman, 1976). These objectives, coupled with the individual-level of analysis inherent in the lens model method, often results in sample sizes that can be individually examined.

Judgment analysis studies with a small number of participants are common. Reported sample sizes in meta-analyses of judgment analysis studies indicate a typical sample size contains about 20 participants with the number of judgments per participant, unsurprisingly, varying according to the number of included cues. Karelaia and Hogarth (2008) found that on average, studies having two cues had 24 participants and 48

judgments; studies with three cues had 20 participants who completed 49 judgments; and studies with more than three cues had 18 participants who completed 73 judgments. In a 2013 meta-analysis that reported sample size for 46 studies, 34 had fewer than 30 participants, 10 of which had fewer than 10 participants (Kaufmann, Reips, & Whittmann, 2013). These sample sizes indicate that in many cases, judgment analysis applications have participant sample sizes small enough that extracted judgment models can be individually examined and described, reducing the usefulness of automated variable selection and ranking methods.

Neither of the abovementioned characteristics—that of a small predictor pool or a small number of individuals—is necessarily problematic, but both contribute to limitations in a more widespread application of judgment analysis. For instance, many task environments contain large numbers of cues that may be related to the judgment processes. Limiting the number of cues explored in judgment tasks is not surprising given the previously discussed challenges of designing judgment analysis studies that contain a sufficiently high judgment to cue ratio. Nonetheless, the Brunswikian method is uniquely invested in accurately describing the environment within which the judge operates (Brunswik, 1952; Gibson & Hobson, 1983; Hammond, 1954). Thus, the low number of predictors commonly employed points to a possible tendency for researchers to err on the side of practicality and thus give insufficient attention to the role of ecological representativeness. Additionally, tasks that contain many cues in the actual environment may be artificially limited or not explored at all.

Similarly, a tendency towards small samples of individuals in judgment analysis studies is unnecessarily limiting. Examining only a small sample of individuals using

judgment analysis may be perfectly appropriate if the objectives of the study are constrained to individual-level analysis; however, judgment analysis has the potential to be more broadly applied. Importantly, much of previous research using judgment analysis has focused on this individual level, thus the relatively small (i.e., $N < 50$) samples common among published studies are likely appropriate (Karren & Barringer, 2002). Nonetheless, the lens model's utility at the individual level of analysis has for too long served as a constraint that has disproportionately restricted it to this level. In addition to its utility in modeling judgment for idiographic studies designed to examine variability at the individual level, judgment analysis can be taken a step further to examine idiographic models nomothetically. That is, judgment analysis should be applied more broadly in idiothetic designs—research that combines idiographic and nomothetic approaches.

An example of an idiothetic approach employing judgment analysis can be found in Carter and González-Vallejo (2018). In that study, the researchers combined judgment analysis with an experimental design. A judgment analysis approach was used to extract judgment models for each individual participant. Modeling individual judgment was not the final goal; rather, characteristics of the extracted individual models were then statistically compared across experimental conditions in order to determine whether participant judgment varied as a function of information conditions. In that study, the nomothetic nature of the experimental condition comparisons was improved by the idiographic method used to understand the individuals' judgment processes. For broader applications of judgment analysis such as this one, a larger sample of participants is necessary. Indeed, for their three-condition experimental design, Carter and González-Vallejo (2018) needed 288 participants to detect a .25 effect size with 80% power. A

sample size of several hundred participants is exponentially larger than most judgment studies and yet this exemplifies a design element necessary to allow for the application of judgment analysis to research studies interested in accounting for between-person differences in judgment.

Variable Selection

Variable selection methods are key in enabling judgment analysis to be more broadly applied in both idiographic and idiothetic designs. In idiographic approaches, even where between-participant statistical tests are not of interest, employing variable selection methods will allow researchers to apply judgment analysis to environments where cue number is high without rendering the number of judgment tasks implausibly high. Furthermore, the implementation of variable selection methods in judgment analysis will allow judgment analysis to be applied more broadly, including in idiothetic approaches where research goals involve submitting extracted models to further statistical tests.

In considering the implementation of variable selection methods for improving model extraction and reducing the potential for overfitting models in judgment analysis, the first necessary distinction is between automatic and directive variable selection. Purely automatic variable selection is determined by a computational calculation that is conducted by software and includes no theoretical insight beyond the initial inclusion of cues. A purely directive approach relies entirely on the expertise of the researcher, is based on prior research, and involves a theoretical or practical conception of which variables are relevant or interesting. Previous research has argued that substantive expertise should be included in predictive modeling and that such modeling has suffered

due to excluding expert, research-informed knowledge (Plonsky, Erev, Hazan, & Tenneholtz, 2017). Although the interaction of expert and algorithmic modeling is an important topic, this study is focused on the comparative utility of purely automatic procedures. As the use of automatic procedures does not exclude expert input, a focus on automatic predictor selection has the potential to improve both purely automatic approaches, that are sometimes deemed necessary, and a combined directive-automatic approach that has been recommended as optimal (Plonsky, Erev, Hazan, & Tenneholtz, 2017).

Automated methods for decreasing the predictor pool for tasks with many cues can enable judgment analysis to be used in areas that have not been previously examined with individual-level modeling or areas where examinations have artificially restricted the cue pool. These methods are particularly important when many initial cues are necessary to accurately represent the judgment environment. In many areas, sufficient observations may not be practical or even possible, making models vulnerable to overfitting. Automated variable selection methods will also facilitate the use of the judgment analysis designs with group-level objectives, allowing for the development of reduced individual-level models that can be compared across experimental conditions or in other broader applications.

Stepwise Regression Variable Selection

Judgment analysis is typically implemented using linear regression. For this reason, a natural beginning point in considering automated variable selection methods is stepwise linear regression. Linear regression applications of judgment analysis enjoy a simplicity and ubiquity that increase their reproducibility and replicability. Multiple

linear regression models use several predictors to predict a single criterion variable as shown in Equation 3:

Equation 3:

$$Y = X_1\beta_1 + X_2\beta_2 + \varepsilon$$

in this equation, Y is a n x 1 vector of observations of the criterion, X_1 and X_2 are predictor variables, β_1 and β_2 represent the weights that describe the relationship between the predictors and the criterion, and ε is the error associated with the model which is not linearly related to the weights. In a judgment analysis application, Y is a y x 1 vector of a single person's multiple judgments (y) of a y x 1 criterion vector.

In addition to the prevalence of linear regression in judgment analysis, multiple linear regression is also widely used to extract explanatory models across research domains including: business and consumer research, social and behavioral sciences, biological sciences, and medical research (Kutner, Nachtsheim, Neter, & Li, 2005). The prevalence of linear regression in academic instruction and its subsequent ubiquity in research has been linked to the easy availability and accessibility of tools for conducting linear regression. For example, Microsoft Suite's Excel program provides user-friendly, point-and-click options for conducting linear regression analyses, enabling even the novice statistician to conduct such analyses without the need for expert training or complex software. Thus, a combination of the linear regression implementation of judgment analysis along with the use of the automated variable selection methods within linear regression would likely facilitate the ease of promoting a judgment analysis approach in a diverse number of research domains.

Multiple linear regression can automatically select between variables in several ways. The most commonly utilized are step procedures due to their ability to minimize computational requirements when potential predictors are many. For simplicity's sake, this study refers to step procedures as forward selection, backward deletion, and stepwise procedures. All three of these selection procedures operate based on the sequence of regression models and altering predictors at each step. Whether predictors are added or removed from the model is determined by the extent to which their inclusion or exclusion impacts the sum of squares error (SSE).

One such step procedure, forward selection, selects the best model by sequentially adding predictors to a model. This procedure begins with no predictors in the model and performs a partial *F* test of the unique variation explained by each potential predictor. The partial *F* test can be seen in Equation 4 where: df_r and SSE_r are the degrees of freedom and sum of square error, respectively, of the reduced model; df_f and SSE_f are the degrees of freedom and sum of square error, respectively, of the full model; and MSE_f is the mean squared error of the full model.

Equation 4:

$$F(df_r - df_f) = \frac{(SSE_r - SSE_f)/(df_r - df_f)}{MSE_f}$$

In a forward selection procedure, predictors are added into the model sequentially according to the unique variation they explain. After the first added predictor, each subsequent potential addition is tested in the presence of all previously added predictors. When no further potential predictors can explain unique variation above what the model predicts, the model is complete.

In contrast, the backward deletion procedure selects the best model by sequentially discarding predictors. This procedure begins with a full model containing all potential predictors and identifies the least predictive variable by performing a partial *F* test of the unique variation explained by the predictors while accounting for all variables in the model. The predictive variable with the largest *p*-value is compared to a predetermined threshold (e.g., $p > .05$) and if it surpasses that threshold, the predictor associated with it is removed from the model. After each elimination, the updated model is reconsidered, and the largest *p* value is again compared to the threshold until the final model contains only those predictors whose *p* values are within the desired threshold. The backwards deletion procedure has been argued to be preferable when the predictor pool is small and when a full model containing all predictor variables would provide a more complete understanding of the impact of predictors in the presence of all potential predictors (Kutner, Nachtsheim, Neter, & Li, 2005).

Both forward selection and backward deletion procedures can be problematic due to their sensitivity to multicollinearity. Multicollinearity is an issue because predictor variables are usually highly associated with one another; therefore, the order in which they are included in the model may impact the degree to which they are recognized by the model as reducing overall SSE. In forward stepwise regression, a variable *A* that is highly correlated with variables *B* and *C* may be selected due to the large amount of variation it accounts for. Subsequently, the model may include variables *B* and *C* due to their unique explanatory power. As can be seen in Figure 2, the resulting model containing predictors *A*, *B*, and *C* will contain a non-significant predictor, namely *A*. Lastly, forward selection procedures cannot account for multicollinearity that occurs after the addition of variables

to the model, and therefore they risk a final model that includes non-significant predictors. The opposite problem of forward selection models occurs with backward elimination models because this procedure only allows for the elimination of predictor variables, hence, final models might not be the best possible model. For instance, a highly correlated variable *A* might be eliminated early on due to its lack of unique variance while subsequent eliminations might result in a final model that could be improved by the addition of variable *A*.

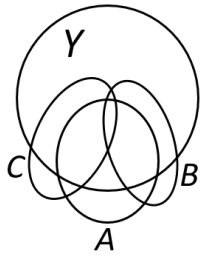


Figure 2. A depiction of multicollinearity. An instance in which forward selection would result in a final model including a non-significant predictor.

A strategy for minimizing the effects of multicollinearity is using a bidirectional procedure that combines both backward and forward regression into a stepwise procedure. In the stepwise regression procedure, an initially empty model is added according to the results of a partial *F*-test of the unique variation explained by each predictor. As with forward selection, predictors following the first are determined according to a partial *F* test of their unique variation in the presence of all entered variables. Unlike the forward selection procedure, the stepwise procedure continues to a backwards step procedure once none of the remaining predictors are significant; that is, the backwards step begins with the forward step's final model, eliminating non-

significant predictors from the model. In the stepwise procedure, the model is considered final when no further predictors can be added or eliminated from the model.

The stepwise procedure method has advantages over forward selection and backward deletion methods as it reduces the impact of multicollinearity, decreasing the likelihood that high predictor correlations will unduly affect the final model. However, there are disadvantages to the stepwise approach. Stepwise procedures are not guaranteed to result in "best case" models; that is, model variance is high indicating that their predictive power for new data is low (Thompson, 1995). In fact, several good models may be available and regression selection methods may not always result in selection of the same model (e.g., Derksen & Keselman, 1992; Greenland, 1989; Whittingham, Stephens, Bradbury, & Freckleton, 2006). For example, in one study using epidemiological data, forward, backward, and stepwise methods were found to lack sufficient stability; the final models they selected were more likely to vary across multiple bootstrapped datasets than alternative variable selection methods (Morozova, Levina, Uusküla, & Heimer, 2015). Moreover, the methods were found to produce biased coefficient estimates, raising the prospect that in addition to having high variance, these relatively simple strategies are ill-equipped to produce even strong explanatory models for the datasets from which they are extracted (Morozova, Levina, Uusküla, & Heimer, 2015).

The use of a stepwise regression approach to reducing predictor pools in judgment analysis has several benefits. Broadly, linear regression and the variable selection methods, such as stepwise regression associated with it, are well-known and used by many researchers across diverse research fields where increased application of

judgment analysis may be beneficial. In addition to being widely accessible and computationally simple, linear regression methods have traditionally been used for implementing judgment analysis; adding the use of stepwise regression for variable selection in this area would seem natural.

Despite these benefits, stepwise regression may not be optimal for selection variables in judgment analysis. Stepwise regression does allow researchers to develop explanatory models with fewer irrelevant predictors, thus reducing the risk for overfitting the model. However, previous research indicates that the resulting models may still have poor predictive performance; they may also be biased. Stepwise regression may not result in the best explanatory model. That is, the final model extracted by stepwise regression may not accurately represent the underlying data. Moreover, stepwise regression does not inform the researcher of the extent to which the selected predictors improve the model above other predictor, resulting in a lack of clarity on the comparative validity of the final model.

All Possible Subsets Variable Selection

A long-recommended alternative to stepwise linear regression is the all possible subsets regression (e.g., Hocking, 1976). All-subsets regression examines all possible combinations of predictor variables and extracts the best model based on a comparison across all possible models. All-subsets regression is easily implemented in linear regression cases, thus its application to judgment analysis is not challenging on a theoretical level. However, even with current computing power, the all-subsets method can easily become computationally challenging. Therefore, although the all-subsets method may be helpful for developing an explanatory model and indeed increasing the

predictive power of the developed models, widespread implementation of this method is currently unlikely.

The predictor criticality approach proposed by Azen, Budescu, and Reiser (2001), addresses possible inconsistencies in traditional regression approaches by combining all-subsets regression and bootstrapping procedures to compare all possible models. For simplicities sake, this approach is referred to throughout as the all-subsets method, but its implementation varies from a simple all-subsets regression approach. In the all-subsets method described by Azen et al. (2001), resampling with replacement is used to obtain a large number of samples from an original data set; the specific resampling method utilized is dependent on the type of regression model under consideration. If model predictors are variable (i.e. can take on any value within their distribution), which is generally the case in behavioral studies, case resampling is recommended (Azen et al., 2001). In case resampling, a random sample of *n* observations is drawn, with replacement, from the original dataset of size *n.* Multiple replicate datasets are drawn in this way, generally 1,000 or 10,000 (e.g., Azen et al., 2001). In the context of judgment analysis, models are developed on the individual level, therefore, replicate datasets would be drawn from the group of observations derived from each individual.

Once the desirable number of samples have been drawn, all possible regression models are fit to all bootstrapped datasets. The number of possible regression models can be designated $2^p - 1$, where p is equal to the number of predictors under consideration. For each of the bootstrapped datasets, a single one of the total $2^p - 1$ models is selected as the best-fitting-model by some predetermined criterion; traditionally, that criterion is a GOF measure such as adjusted R^2 (denoted R^2_{ADJ} hereafter), AIC, or Mallow's C_p. In

principle, any criterion can be established as the determinant of the best model, but a measure that adjusts for the number of predictors in the model is necessary. Some possible candidates for criterion (e.g., the R^2) can only increase as a function of additional predictors and are therefore inadequate representations of the value placed on model parsimony.

As an example of this all-subsets method, suppose there is a judgment analysis dataset of 50 observations per individual which the researcher is interested in modeling using four predictors ($X_{1\text{-}4}$). From the initial 50-observation dataset, 1,000 datasets are bootstrapped. For each bootstrapped dataset, the following fourteen models would be fit: X_1; X_2; X_3; X_4; X_1, X_2; X_1, X_3; X_1, X_4; X_2, X_3; X_2, X_4; X_3, X_4; X_1, X_2, X_3; X_1, X_2, X_4; X_2, X_3, X_4; X_1, X_2, X_3, X_4. Using one of the aforementioned GOF criterions, the models fit within each sample would be compared and the best-fitting model would be determined. Additionally, the frequency with which each model is the best-fitting model across all bootstrapped samples would be computed and the model which is most frequently the best-fitting model is selected as the final model. In a judgment analysis context this process would be repeated for every participant's set of 50 observations.

Azen, Budescu, and Reiser (2001) proposed that this all-subsets method offers the further advantage of determining and ranking potential predictors according to a new dimension they call predictor criticality. They define predictor criticality as the probability that a potential predictor is included in the best subset model. Unlike the traditional linear regression selection methods previously discussed, the predictor criticality perspective does not assign importance ranks to predictors according to the unique variance they explain. Rather, *criticality* is assigned to predictor variables based

on whether they are highly predictive across multiple models. In traditional linear regression methods, the importance of a predictor would be determined by the size of its β-weight; in contrast, with the bootstrapping technique importance it is based upon the necessity of a predictor in enabling the identification of the best model. Thus, unlike traditional methods of determining variable importance, predictor criticality connects the ranking of predictor variables to the distribution of best-fitting models (Azen et al., 2001).

There are key differences between interpretations of predictor criticality and variable importance measures such as β-weights. Azen et al. (2001) conducted simulations to examine the effect of multicollinearity on predictor criticality. In their first group of simulations, they examined the effect of varying correlations among six pairs of predictors while the pattern of correlations between the predictors and the criterion were kept constant. When the four predictors were highly related to each other ($\rho = 0.75$), their ranking in importance measured were varied. In contrast, their criticality measures were equally and maximally critical. This equal and maximal criticality ranking is understandable when the measurement of criticality is recalled. Criticality is measured by the probability that a given predictor is included in the best-fitting model. In the simulated case where the correlations among all predictors are high ($\rho = 0.75$), the full model with all predictors was identified as the best-fitting model in every case. Therefore, removing any of the predictors would result in misidentification of the best-fitting model and thus all predictors are equally and maximally critical even though they are not equally correlated with the criterion (Azen et al., 2001).

Azen et al. (2001) found that in the simulated case in which there is no correlation between predictors (no multicollinearity), the criticality of the predictors is directly related to their correlation with the criterion. This finding mirrored their results regarding variable importance measures β, partial correlations, and semi partial correlations, which correctly ranked predictors when there was no intercorrelations between them ($\rho = 0$). The researchers also found that a predictor's correlation with the criterion and other predictors had an interactive effect on criticality. Specifically, a second set of simulations showed that an increase in the relationship between a predictor with no relationship with the criterion ($\rho = 0$) and a predictor with a moderately high relationship with the criterion ($\rho = 0.6$) resulted in an increase in the criticality of the non-related predictor (Azen et al., 2001). This effect was also found for traditional measures of importance, although the interpretation of these measures would not be equivalent to that of criticality (Azen et al., 2001).

The all-subsets method for extracting a model has several advantages over stepwise regression. First, the reliability of the final explanatory model in the all-subsets method is greater. This technique compares all possible models over resampling thus the best-fitting model selected is one that has been shown to be the best fitting across multiple samples. Secondly, the predictor criticality measure provides a clearer standard for comparing predictors. Azen et al. (2001) note that most measures of variable importance are dependent on the choice of a particular model. In these methods, as in that of stepwise regression, a model is chosen based upon some overall GOF measure and predictor importance is determined according to the proportion of the overall GOF (e.g., R^2) a predictor accounts for when that GOF is partitioned across all variables. Predictor

criticality, unlike these traditional measures of predictor importance, is not model-dependent (Azen et al., 2001). That is, because predictor criticality relies on an all subsets approach, all possible models are considered. Thus, criticalities can be computed for all predictors rather than only those that are included in the final "best model" (Azen et al., 2001). By being expressed in terms of the conditional probability distribution of best-fitting models, predictor criticality provides intuitive clarity on the need to include a given predictor. That is, the predictor criticality provides the probability that the model would be misspecified if a predictor was not included in the model. As this probability can be calculated for all predictors, not only the subset of predictors that are included in the final model, the comparison of all available predictors is facilitated (Azen et al., 2001).

In addition to these important benefits, the conceptualization of the all-subsets method is easily related to linear regression, therefore increasing its accessibility to researchers in a wide variety of fields. Like the stepwise regression method, all subsets could be applied to the current prevalent use of linear regression in judgment analysis improving judgment analysis' applicability to new areas and designs without adding challenges to its conceptualization.

The primary deterrent to implementing the all-subsets method in judgment analysis is the computational cost of applying it. The all-subsets method is computationally daunting in any circumstance when the number of potential predictors is high, but individual level modeling poses an even higher level of computational cost. In judgment analysis, a model is extracted for each individual participant from the multiple judgments they formed of the environment and the predictors associated with their

judgment task. Even for judgment analysis studies in which the number of participants is very small (e.g., 3 participants as in Dougherty, Ebert, Callender, 1986), the all-subsets method would have to fit $2^p - 1$ models across many bootstrapped samples *for each participant.* When more than a few predictors are evaluated, the computational cost of fitting $2^p - 1$ across even one set of 50 bootstrapped samples may become prohibitive; the cost of doing so multiple times, as is necessary for judgment analysis, is even more computationally costly (Johnson & LeBreton 2004; Strobl, Malley, & Tutz, 2009).

Despite the computational deterrent to implementing the all-subsets method in judgment analysis, previous researchers have used it in this context. González-Vallejo, Lavins, and Carter (2016) examined the nutritional judgments of 196 individuals using an all-subsets method. Each individual had evaluated either 40 cereals or 40 snacks and a set of 12 nutrients were selected to be included in the models predicting their judgments. For each participant, the researchers generated a set of 10,000 bootstrapped samples and all models were compared using R^2_{ADJ} in order to derive the best model for each individual's judgment process.

González-Vallejo et al.'s (2016) study provides a case example for the application of the all-subsets method to judgment analysis; however, the extent to which the all-subsets method is computationally feasible and statistically advantageous for implementation in judgment analysis has not been documented. Of particular interest, is whether the all-subsets method will be more reliable in the context of extracting judgment models than the less computationally demanding stepwise approach. Moreover, an investigation is needed to determine whether the statistical advantages of the all-subsets method, that previous researchers have documented, are sufficiently impactful in

the judgment analysis context to warrant recommendation of this method over the more broadly accessible stepwise method.

Random Forest Variable Selection

An additional method for implementing variable selection that could be applied to judgment analysis is a random forest variable selection method. There are indications that random forest variable selection may be a more reliable and practical method than the stepwise and all-subsets methods respectively. Sandri and Zuccolotto (2006) used simulation tests and empirical data to compare the random forest variable selection to stepwise procedure in classification modeling. Their research found that the random forest method identified a smaller number of relevant predictors than the logistic stepwise procedure, allowing for a more parsimonious model with similar predictive performance. Moreover, unlike the all-subsets method, the random forest method does not fit all possible models on bootstrapped samples. Rather, random forest fits a tree model to bootstrapped samples only once and estimations are found by averaging over the many trees across the bootstrapped samples. Random forests then determine the most relevant variables by ranking predictor variables according to how much their inclusion decreases mean squared error (MSE). Although the all-subsets method of selecting variables is more intuitively clear, random forests' variable selection is less computationally costly (Strobl, Malley, & Tutz, 2009).

Random Forest Process Description

Random forests are developed by constructing and aggregating decision trees across multiple bootstrapped samples (Breiman, 2001). As with the all-subsets method, random forests begin by using sampling with replacement to draw multiple samples from

a single original sample. In random forest, however, decision trees are constructed for each sample. Constructing an initial regression tree—a decision tree for a real-valued number rather than a classification—begins with splitting the data at multiple points. The size of the regression tree is determined by the required minimum number of observations for each node. A large regression tree will have a low minimum number and will have the advantage of having low bias but the disadvantage of overfitting the data.

The problem of overfitting regression is best addressed by developing regression trees across multiple bootstrapped datasets and then aggregating the resulting trees over the repeated samples (for recommended number of trees, see Oshiro, Perez, & Baranauskas, 2012). This bootstrap aggregation method—called bagging—reduces the variance of the final estimated values by averaging across the low bias, high variance trees. Although individual trees have high variance, aggregating values across ensembles of trees allows for an impressive reduction in estimate variance, thereby increasing the predictive accuracy of the model (Breiman, 2001).

In usual bagging methods, a regression tree is grown by determining split points based on which point, out of all predictor variables, will most greatly minimize the sum of squared error (Breiman, 1996). Random forest bagging methods determine a split point by considering a random sample (without replacement) of predictors from the full set of predictors. The number of predictors randomly sampled for consideration as split candidates is predetermined by setting *m* predictors. Thus, traditional bagging procedures are random forest procedures in which *m* is equal to the number of predictors. Once a random subset of predictor variables has been selected, the random forest method follows the procedure of selecting a split point according to which predictor will allow for a split

point that will most minimize the sum of squared error within the region. The split value resulting from this process is a node in the subsequent regression tree. This process is then repeated for each subsequent split until the tree is grown. Function 1 describes the first step in this recursive process.

Function 1:

$$\sum_{y_i:x_i>b} (y_i - \overline{y})^2 + \sum_{y_i:x_i\leq b} (y_i - \overline{y})^2$$

In the above equation, y_i is defined as a real-valued case of a criterion *y* that is a function of x_i, real-valued predictor variables 1-k, and some error. Some specific predictor *i* and a value *b* of that x_i predictor are chosen such that for all cases of *y*, in each region of x_i, the sum of the squared difference of y_i and mean *y* is minimized.

For an example of this random forest regression tree method, assume that *m* has been set to three. Then, for three randomly selected predictor variables $X_{1\text{-}3}$, the entirety of data space, designated A is split into A_1 and A_2 based upon which variable $X_{1\text{-}3}$ and associated value *b* will minimize error within the initial region A (see Figure 3). Subsequent split points are determined in the same fashion, beginning by taking a fresh random sample of *m* predictors, and selecting a predictor variable and value which minimize the error in the subset region being considered (e.g., in Figure 3, region A_2). The number of splits allowed is determined by a preset parameter setting the minimum required observations within a final region as a result of splitting. In addition to allowing for a stopping point for recursive splitting, such an *a priori* determination ensures that outlier data points will not fully determine split points.

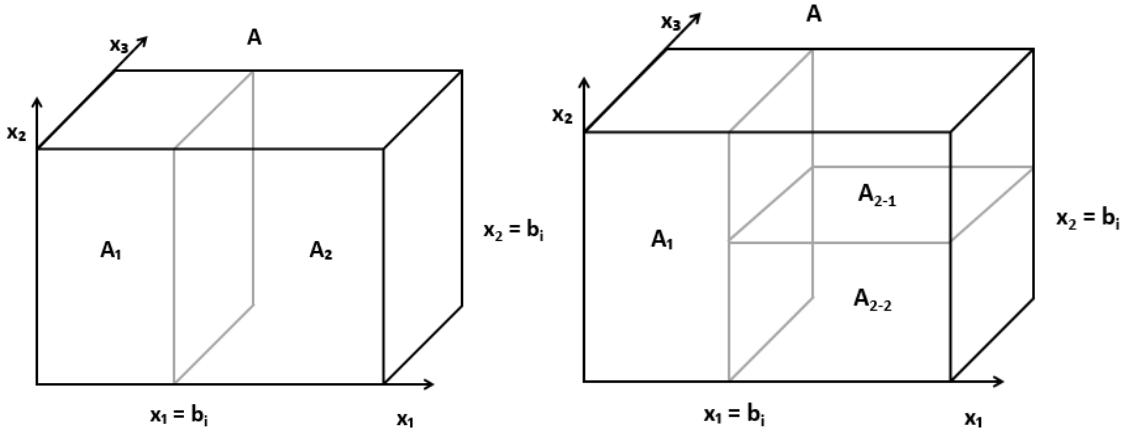


Figure 3. Random forest splitting process. Subset regions A_1 and A_2 are determined by selecting split points on $x_i = b_i$ such that error within both regions is minimized. A value of x_2 that minimizes error in further subset regions $A_{2\text{-}1}$ and $A_{2\text{-}2}$ is selected as the subsequent split.

The key factor in random forest regression tress is the determination of number *m* of variables less than the total number of predictors (*p*). By selecting an ideal initial split point out of a randomly selected sample of candidate variables, the random forest method reduces the correlation among trees. For example, consider a participant in a judgment analysis task who is evaluating the nutritional quality of multiple food products. Calories, a nutritional variable that is positively related to many of the other nutrients in a food, may be highly related to the participant's nutritional rating. If a traditional bagging approach were used, calories would be included in most or all the trees developed for this participant. As a result, the bagged trees would be quite similar and their predictions highly correlated resulting in a loss of variance reduction despite bagging. By introducing an element of randomness in the selection of the variables to be considered, the resulting trees are less correlated, and their averages have less variance and are therefore more reliable.

Following the construction of the many regression trees that make up the "random forest," a validation technique called permutation importance is used to provide a comparative measure of “predictor importance” for the predictor variables. A by-product of bagging is that any given bootstrap training set does not use all observations available in the original dataset. In fact, previous testing has demonstrated that approximately a third of observations in the original dataset do not appear in a given bootstrapped dataset (Breiman, 2001). Permutation importance relies on these observations, called "out-of-bag" (OOB) cases, to assess the relative importance of predictor variables (Breiman, 2001). To determine variable importance, the value of a particular predictor, X_1, is permuted in every tree. That is, rather than reflecting the value dictated by the model, every occurrence of X_1 is assigned a randomly selected value from the possible values for X_1 and the resulting tree is used with the OOB cases. Subsequently, the MSE is computed for every tree and the difference between the permuted trees' MSE values and the non-permuted trees' MSE values is taken and these differences are then averaged across all trees to determine the importance of predictor X_1. This process is repeated for all remaining predictor variables $X_{2\text{-}k}$. The average difference between the MSE error for the actual prediction and the MSE for the permuted prediction (referred to hereafter as ΔMSE_{OOB}) is the variable importance ranking for each predictor, which can then be compared to determine relative importance. This method presumes that permuting important predictor variables will have a greater impact on the MSE than permuting unimportant variables (Genuer, Poggi, & Tuleau-Malot, 2015).

In random forest, the OOB cases are also used to compute a “percent variance explained” metric that is a measure of cross-generalization rather than goodness-of-fit.

This measure (denoted R^2_{OOB} hereafter) is derived from the $MSE_{OOB,}$ which is the aggregated sum of squared differences between the actual judgment values and the judgment values predicted for the OOB cases that were not included in the bootstrapped samples used to extract the model. Percent variance explained is defined as

Equation 5:

$$R^2_{OOB} = 1 - \frac{MSE_{OOB}}{\hat{\sigma}^2_y}$$

where $\hat{\sigma}^2_y$ is computed with *n* rather than *n* – 1 as the divisor (Liaw & Wiener, 2002). Because R^2_{OOB} is a proxy for a cross-generalization measure it does not have direct linear regression or all subsets counterpart.

A goodness-of-fit metric analogous to that of an R^2_{ADJ} can be computed for random forest using the following equation.

Equation 6:

$$R^2_{Adj.RF} = 1 - \frac{\frac{SSE}{n-p-1}}{\frac{SST}{n-1}}$$

SSE is the sum of squared error, determined by the sum of the squared differences between actual observed values and the predicted values averaged across all regression trees (James et al., 2013). The number of observations is denoted by *n*; the number of predictors included in the model is represented by *p*. SST is the total sum of squares, determined by the sum of the squared differences between the observed values and the overall mean of the observed values.

The random forest method includes the selection of several a priori determinations, or hyperparameters that can impact the analysis. The first of these

hyperparameters is the number of trees which should be included in the ensemble of trees; second, the number of predictors that are randomly selected to be considered at each spilt; third, node size, or the minimum number of observations in a terminal node. Lastly, when reducing model predictors is of interest, some variable importance threshold is necessary to determine which variables from the full model are included in the final model.

Previous research, on the number of trees to include in a random forest, has indicated that fewer bootstrapped samples are necessary when the outcome variable is numerical (i.e., for regression forests) compared to when there are many classes (i.e., for classification forests) (Breiman, 1996). A 2018 empirical study that conducted random forest analysis on 193 classification datasets and 113 regression datasets confirmed this numeric-class distinction (Probst & Boulesteix, 2018). Theoretical results by Probst and Boulesteix indicated that increasing the number of trees is—in most cases, and in all regression forest cases—monotonically related to decreased average out-of-bag error rates. However, both theoretical and empirical indicated that the greatest improvement in performance is attained by the earlier increases in tree number. For regression, Probst and Boulesteix (2018) found that increasing the number of trees from 11 trees to 2000 trees resulted in a 0.1249 improvement in the OOB R^2. However, 0.121 of that improvement was attained by using 250 trees, with the increase from 250 to 2000 trees improving the cross-generalization of the model by 0.0039 OOB R^2.

The findings of Probst and Boulesteix (2018) concur with those of earlier researchers examining how many trees are sufficient before improvements in model fit and cross-generalization become negligible. Oshiro et al. (2012) compared the

performance of random forest classification among 29 datasets using 12, exponentially increasing tree size conditions. For each tree size condition, Oshiro et al. used the average Area Under the Curve (AUC) between the true positive rate and false positive rate over ten repetitions of a ten-fold cross-validation of the classification random forest for each dataset as a measure of performance. They examined the effect of the number of trees on the average AUC across all datasets and concluded that adding more trees passed a certain threshold (in their cases, 128 trees) resulted in minimal gains in AUC performance.

The number of predictors randomly considered at each split (denoted *m*) is another random forest hyperparameter that can be "tuned" to increase the cross-generalizability of the model (Genuer, Poggi, & Tuleau-Malot, 2010). Lower values of *m* lead to individual trees that are less correlated, yielding greater forest stability and thus greater cross-generalizability (Breiman, 2001; Grömping, 2009). Breiman (2001) recommended $m = \sqrt{p}$ for classification forests but noted that the negative relationship between *m* and cross-generalizability was weaker in regression forests. Based on these findings, the default *m* for regression forests is *p*/3 (Grömping, 2009; Liaw & Wiener, 2002). Previous studies have indicated that these default values generally provide optimal decreases in cross-generalization error (e.g., Díaz-Uriarte & De Andres, 2006; Grömping, 2009; Li, Tran, & Siwabessy, 2016), although, increasing *m* can lead to increases in performance when there are many predictors and a high proportion of irrelevant predictors (Díaz-Uriarte & De Andres, 2006).

The minimum number of observations allowed in a terminal node determines the depth to which a tree is grown. A lower node size allows for more splits to be performed

and thus a "deeper" tree than is allowed by a larger node size (Strobl, Malley, & Tutz, 2009). The default node sizes in implementing random forest are 1 for classification and 5 for regression (Liaw & Wiener, 2002). In general, growing large trees is seen as imperative to the random forest approach (e.g., Breiman, 2000; Lin & Jeon, 2006). This is particularly true when the ratio of observations to predictors is small, however, in cases when the ratio of observations to predictors is large, larger minimum node sizes, and thus growing smaller trees, have been shown to be optimal (Lin & Jeon, 2006).

Grömping (2009) found that although regression forest variable importance rankings were not dependent on *m*, the combination of large minimum node sizes and small *m* impacted variable importance. Simulation results showed that when minimum node sizes were large, a small *m* resulted in a substantial decrease in importance allocation to the predictor with the largest coefficient in favor of weaker predictors. Notably, this was the case even when those predictors were uncorrelated. Variable importance for predictors that had no real effect on the criterion did not increase (Grömping, 2009).

Variable Selection Using Random Forest

Substantial attention has been given to the development of methods for selecting variables with random forest. Most approaches utilize the variable importance measures that rank predictors according to their impact on OOB error, but the methods in which these measures are used vary widely (Hapfelmeier & Ulm, 2013). Hapfelmeier and Ulm (2013) classify recent approaches to variable selection as "performance-based" and "test-based." Methods belonging to the former utilize recursive refitting and select the predictor set that results in the lowest OOB model error (e.g., Díaz-Uriarte & De Andres,

2006; Ben Ishak, 2016; Genuer et al. 2010). The latter methods involve the use of significance tests for individual predictors and rely on *p*-values thresholds (e.g., < 0.05) to determine which predictors are selected (e.g., Altmann, Tolosi, Sander, & Lengauer, 2010; Breiman & Cutler, 2008). Test-based approaches are problematic as the power of this approach has been demonstrated to depend on the size of the ensemble included in the random forest rather than the number of observations (Strobl & Zeileis, 2008). As a result, it is unclear what hypothesis is actually being tested by test-based approaches and their conclusions regarding predictor importance are likely inappropriate (Hapfelmeier & Ulm, 2013; Strobl & Zeileis, 2008).

These approaches to variable selection vary in complexity and sophistication but all operate on determining thresholds for elimination or selection. Liaw and Wiener (2002), in their early demonstration of random forest for regression and classification, exclude predictors simply based on their relative variable importance ranking, keeping only predictors that are high in variable importance. Díaz-Uriarte & De Andres (2006) recommend iteratively eliminating the 20% of predictors with the smallest variable importance and building a new forest with the remaining variables, selecting a final set of predictors according to the variables that minimize the OOB error rate over multiple forests. In the following study, a performance-based variable selection method proposed by Genuer et al. (2010) and developed in the VSURF R package (Genuer, Poggi, & Tuleau-Malot, 2018) is used.

Genuer et al.'s (2010) variable selection strategy has two main advantages. First, its thresholds are derived from the data, rather than relying on arbitrary parameters (e.g., proportion or number to remove) data-driven thresholds. Second, it distinguishes between

two different researcher goals, interpretation of the data or prediction of new data (Genuer et al. 2010). According to their strategy, researchers interested in interpretation will prioritize finding all predictor variables highly related to the criterion, even if the final selection includes variables that are redundant (Genuer et al., 2015). In contrast, a researcher interested solely in predicting the criterion variable would be more interested in finding the smallest number of variables sufficient to allow for good prediction of the criterion (Genuer et al., 2015). Genuer et al.'s method uses a multi-step process to select variables depending on the objective of interest. First, the multiple forests of trees are generated to determine variable importance. Variables are ranked according to their variable importance and their averaged variable importance values; additionally, corresponding standard deviations are used to estimate a threshold value. This threshold value is computed according to the minimum prediction value given by a random forest model, where the variable importance values are used to predict the variable importance standard deviations (Genuer et al., 2015). Only variables with an average variable importance higher than this threshold are retained.

In the second step, all possible random forest models with the selected variables are estimated in multiple forests of trees and the models OOB error rate is used to select the final model. Specifically, the method selects the smallest model that has an average OOB error rate less than the sum of the smallest OOB error and its standard deviation. The inclusion of the error's standard deviation allows the selection to account for the stability of the model (i.e., models with a low average OOB error but a wide range of OOB error values are less likely to be selected). For researchers interested solely in developing a model to predict new criterion, a third step begins with the variables

retained for interpretation to construct an ascending sequence of multiple random forest models using a stepwise method that adds and removes variables according to their impact on OOB error (Genuer et al., 2015).

The random forest variable selection method has an advantage over stepwise regression methods due to the variance reduction technique built into it by bagging. Unlike the all-subsets method, the random forest's computational complexity is not exponentially increasing. In fact, it can be held relatively stable despite increases in predictor variables. Random forest's reduction in variance and its reduced computational complexity make it advantageous for implementation in judgment analysis over stepwise and all-subsets methods, respectively.

Despite the statistical and computational advantages to a random forest variable selection method, its implementation in judgment analysis is not without challenges. The greater complexity of the random forest technique both computationally and in terms of interpretation have contributed to it being less prevalent in behavioral research. Notably, because the random forest method involves the aggregation of tree models that partition the data space in a non-additive fashion, interpretation of the predictors' directional impact is much less clear-cut than it is in an additive, linear approach that relies on directional coefficient estimates (James et al., 2013; Strobl, Malley, & Tutz, 2009). However, the computational complexity of implementing random forest models has been reduced due to the increasing accessibility and prevalence of more powerful and flexible statistical tools within the field of psychology. Thus, there is an opportunity for judgment analysis researchers to adopt variable selection methods with this technique.

As was noted in an earlier section, the use of a linear regression model is not required by judgment analysis and indeed at least one previous empirical study used decision trees for judgment analysis showing them to outperform the traditional linear approach. Lafond, Vallières, Vachon, St-Louis, and Tremblay (2015) had 60 participants use five cues in a naval air-defense judgment task to classify over 300 radar contacts as hostile, uncertain, or non-hostile. The researchers generated two models for the judgment process of individual participant, a logistic linear regression and a decision tree model and found that across all participants, that decision trees had higher goodness-of-fit and higher cross-validation accuracy than logistic regressions (Lafond et al., 2015). The average goodness-of-fit across participants for decision tress and logistic regressions were 96% and 89.10% respectively, while the cross-validation accuracy of each method was 95.28% and 88.47% respectively (Lafond et al., 2015). Notably, the decision tree models generated in this task were not bagged nor did they use random subsets of predictors, both of which help improve the predictive accuracy of decision trees in the random forest approach. Other researchers have also noted that decision trees are likely a more accurate model of the human judgment process than a linear weighting process (e.g., Dhami & Harries, 2001; James et al., 2013). The overwhelming use of regression modeling in judgment analysis applications is largely due to convenience.

The question of whether a decision tree approach improves the validity of an extracted model is an empirical question. Additionally, the impact of any improvement, in accuracy or efficiency, should be documented in order to facilitate researcher decision-making. It is possible that an all-subsets method results in substantially different outcomes than a random forest approach. If this is the case, then researchers should be

urged to consider the more computational costly technique if they are to develop reliable models. Conversely, if the accuracy of the random forest method is better or equivalent to that of the all-subsets method, the reduced computational cost of the former will make its use advantageous. Similarly, if the random forest method offers significant improvement in accuracy over a stepwise regression method than adoption of this method should be recommended. If, however, comparisons indicate that stepwise regression performs comparably well, the computationally inexpensive and widespread prevalence of stepwise regression would make its implementation preferable.

Summary

Judgment analysis could be more broadly applied in behavioral research if variable selection methods could be implemented to reduce overfitting under conditions in which many possible predictors might be included in the model. Automated methods for decreasing the predictor pool can enable judgment analysis to be used in areas that have not been previously examined with individual-level modeling and facilitate its application in designs with group-level objectives.

By using variable selection methods in judgment analysis researchers can improve the parsimony of explanatory models of human judgment. Developing more parsimonious models of the judgment process has two advantages. First, an emphasis on determining the importance of factors related to a judgment task will increase the clarity of explanatory models, providing a better understanding of the judgment process. Second, by reducing the number of predictor variables, the likelihood of overfitting explanatory models will be decreased, thus facilitating the development of models that can predict future data as well as explain the data from which they are extracted.

The three variable selection methods discussed in this introduction have been previously demonstrated to have different advantages and disadvantages. Stepwise regression variable selection is computationally inexpensive and enjoys established prevalence in behavioral research. However, previous research has indicated that stepwise regression variable selection can lead to biased and high-variance estimates, reducing its potential for interpretable models and accurate prediction. All subsets variable selection provides intuitive interpretations of a best final model and allows for clear comparisons of predictor usefulness. Nonetheless, the clarity allowed for by the comprehensiveness of all subsets comes at a computational cost that may be too high, particularly for designs with many predictors and many subjects, which would most greatly benefit from variable selection methods. Random forest variable selection does not involve the same computational cost as all subsets and involves bias and variance reduction techniques that may prove advantageous over stepwise regression. Despite this, the random forest method is more computationally challenging than a stepwise regression approach and involves a decision tree approach—which is currently uncommon in judgment analysis. Random forest variable selection also lacks the theoretical clarity present in the all-subsets method and is comparably novel and less well-understood.

Current Research

The current research will use three variable selection methods in applying the lens model to investigate the utility of variable selection methods and compare three specific methods: stepwise regression, all possible subsets, and random forest. In keeping with the Brunswikian approach, these methods are contrasted in describing the judgment environment used to different degrees in previous work by González-Vallejo et al. (2016)

and Carter and González-Vallejo (2018). Subsequently, across three studies, the researcher contrasts the accuracy and efficiency of the traditional variable selection method of stepwise regression, the comprehensive all-subsets method, and the more novel random forest method. In the first study, the results of these methods are compared using empirical data, the judgments of 298 individuals from the archival data from Carter and González-Vallejo's (2018) study. In the second study, the methods are compared using simulated data that were based on a nutritional judgment task of the archival data examined in the first study. In a final study, data with different correlational structures are simulated in order to investigate the effect of collinearity on the three methods' selection and ranking performance.

Not only do these studies describe the results of the respective variable selection methods, they also seek to document the differences in approach and process each method requires. This provides future researchers, particularly those interested in human judgment and decision-making, with a concrete picture of the relative costs and benefits of diversifying the methodologies they employ.

CHAPTER 2: DESCRIBING THE TASK ENVIRONMENT

A key aspect of Brunswik's (1952) approach, and that of the judgment analysis method derived from it, is a focus on understanding the judgment environment. Brunswik argued that the entire study of judgment is that of the relationship between a judge and the environment; an examination of the judge alone is insufficient to derive an appropriate understanding of the judgment process (Brunswik, 1952). The characteristics of the environment, or the judgment task that a judge undertakes, are important for two primary reasons. First, the characteristics of the judgment task can help or hinder a judge, impacting both the subjective ease of judgment and its objective accuracy (Stewart et al., 1997). Second, the characteristics of the judgment task describe the context in which the judgment process was learned; the judgment task itself can be crucial in determining the cognitive approach adopted by the judge so understanding the task is fundamental to understanding the judgment process (Hammond et al., 1987; Stewart et al., 1997).

A classic study exemplifying the necessity of understanding the task environment is that Stewart et al.'s (1997) examination of human forecasting. In that study, Stewart et al. compared human forecasting of two weather conditions, temperature and precipitation. Analysis of the two judgment tasks found differences in the two tasks number of cues, cue redundancy, linearity, and predictability, leading the researchers to predict that accuracy, agreement, and consistency of the forecasters' judgments would be lower for precipitation than for temperature. Results from the study found that this was the case. Temperature forecasts were more accurate than precipitation forecasts and agreement among the forecasts was lower for precipitation with indirect evidence suggesting that reliability was also lower than in the temperature task (Stewart et al.,

1997). In contrast, in a study of human judgment in another task environment—forecasting hail—the forecasting accuracy, agreement, and consistency of forecaster judgment was lower than in the temperature and precipitation tasks (Stewart et al., 1989). Analysis of the hail task environment indicates that a good statistical model for this task is difficult to derive, indicating that the task itself has low predictability. A high level of unpredictability in the task environment limits the accuracy of human judgment, increasing the likelihood that judgment performance in that task environment will be low (Stewart 1990, Stewart et al. 1997).

Original Judgment Task

In the original study by Carter and González-Vallejo (2018), participants were presented with 74 cereals and asked to judge their nutritional quality on a scale from 1-100. The products used in the study were actual cereals available in the market and in this way were representative of the actual judgment task of judging cereals. Cereal images were shown in three different display conditions (see Appendix for an example), but in all cases, an image of the front of the cereal box was provided along with the cereal's Nutrition Facts Panel (NFP), and ingredient list. The authors used the NuVal® score (referred to as NuVal hereafter) as an expert criterion of nutritional quality (Carter & González-Vallejo, 2018). NuVal, a nutrition scoring system developed by medical and nutritional experts, summarizes the overall nutrition of a food on a scale from 1 to 100 with higher scores indicating higher nutritional quality (NuVal LLC, 2015). Further discussion of this system as a nutritional criterion can be found in Carter & González-Vallejo's (2018) study, as well as previous studies utilizing NuVal (González-Vallejo & Lavins, 2015; González-Vallejo, Lavins, & Carter 2016).

Statistical Analysis of the Environment

To draw conclusions about the environment, the three methods of analysis—stepwise linear regression, all subsets, and random forest—were each applied to examining the environment. As the expert criterion used for the environment was the NuVal score, the statistical analysis of the environment is based on modeling the relationship between the nutrient predictors in the environment and the NuVal. The cereals' NFP labels contained nutrient information for 16 nutrients.[2] calories, sodium, sugars, dietary fiber, as well as calories from fat, total fat, saturated fat, polyunsaturated fat, monounsaturated fat, potassium, insoluble fiber, soluble fiber, total carbohydrates, other carbohydrates, protein, and number of vitamins and minerals. Prior to the analysis, the predictor pool was reduced to include only nutrients known to be included in the NuVal algorithm. These included twelve nutrient values: calories, total fat, saturated fat, polyunsaturated fat, monounsaturated fat, sodium, potassium, total carbohydrates, dietary fiber, sugar, protein, and a subset number of vitamins and minerals.[3] Additionally, due to an error in survey design, a single cereal was dropped from the analysis, resulting in a total of 73 unique cereals.

Statistics describing the correlations between NuVal and the nutrients, as well as the inter-correlations of the nutrients for the 73 cereals and the variables' central tendencies are shown in Table 1. As can be seen from Table 1, many of the nutrients are

---

[2] Two other nutrient variables, trans fat and cholesterol, were also included on the labels and their values are coded in the database of cereals but as they had constant values (= 0) across all cerals they were excluded from the initial predictor set and had no impact on subsequent analyses.

[3] NuVal's algorithm includes a subset of vitamins and minerals compared to those that appear on the cereal's NFP. The vitamins and minerals included in NuVal include: folate (used interchagably on the NFP with folic acid), vitamin A, vitamin C, vitamin D, vitamin E, vitamin B12, vitamin B6, calcium, zinc, magnesium, and iron.

related to each other. Out of the sixteen nutrients included in the NFP, ten had significant correlations with at least half of the total nutrients. Calories was the most inter-correlated predictor; it was significantly correlated (at the $p < .05$ level) with twelve other nutrients. Sodium was the least intercorrelated, significantly correlating (at the $p < .05$ level) with only three other nutrients.

When only the twelve nutrients variables included in the NuVal are considered, intercorrelations among the predictors remained high. Calorie value is positively related to the four nutrient variables excluded by the NuVal, calories from fat, insoluble fiber, soluble fiber, and other carbohydrates, so excluding these variables brings the number of nutrients that calories is significantly related to down to eight. Potassium and protein are also each significantly related to eight nutrients. In all, seven of the twelve nutrient variables included in the NuVal were significantly correlated (at the $p < .05$ level) with at least half of the twelve total nutrients.

Table 1

Environment Descriptives: Pearson Correlations and Descriptive Statistics of NuVal and Cereal Nutrient Variables (N = 73)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1 NuVal	.																	
2 Calories	.07																	
3 Calories (Fat)	.12	.46*																
4 Total Fat (g)	.14	.44*	.95*															
5 Saturated Fat (g)	.02	.14	.53*	.53*														
6 Polyunsaturated Fat (g)	.31*	.42*	.76*	.80*	.40*													
7 Monounsaturated Fat (g)	-.02	.43*	.80*	.80*	.36*	.58*												
8 Sodium (mg)	-.54*	.03	-.11	-.15	-.10	-.20	-.06											
9 Potassium (mg)	.30*	.72*	.34*	.34*	.02	.46*	.24*	-.08										
10 Total Carbs (g)	.05	.93*	.19	.18	-.02	.18	.20	.00	.68*									
11 Dietary Fiber (g)	.38*	.53*	.26*	.27*	.01	.28*	.13	-.23	.68*	.59*								
12 Soluble Fiber (g)	.13	.24*	.28*	.26*	.01	.23	.30*	-.22	.38*	.23*	.57*							
13 Insoluble Fiber (g)	.27*	.36*	.05	.06	-.04	.18	-.06	-.40*	.51*	.47*	.67*	.42*						
14 Sugar (g)	-.45*	.52*	.21	.19	.07	.11	.20	.27*	.33*	.53*	.15	.05	.02					
15 Other Carbs (g)	-.15	.27*	.21	.26*	.06	.19	.46*	.20	.06	.20	-.06	-.08	-.13	.04				
16 Protein (g)	.32*	.65*	.32*	.33*	-.01	.44*	.24*	-.11	.74*	.56*	.68*	.51*	.54*	.10	-.04			
17 Vitamins & Minerals (#)	-.19	.12	.05	.06	-.06	-.02	.18	.36*	.01	.08	-.22	-.24*	-.28*	.24*	.47*	-.18		
18 NuVal Vitamins & Minerals (#)	-.22	.01	.05	.06	-.02	-.02	.16	.39*	-.06	-.03	-.26*	-.25*	-.34*	.24*	.42*	-.24*	.97*	.
Mean	29	145	15	1.62	0.13	0.45	0.41	139	118	31	3.65	0.40	1.11	8.89	7.05	3.52	9.34	6.26
Median	26	120	10	1.00	0.00	0.00	0.00	140	85	26	3.00	0.00	0.00	9.00	0.00	2.00	11.0	7.00
S.D.	16	44	11	1.28	0.30	0.68	0.67	73.5	90.5	9.55	2.82	0.96	2.44	4.11	10.1	2.35	4.70	3.00

*Correlation is significant at the $p < 0.05$ level (2-tailed).

The intercorrelations between nutrient variables complicates the extraction of a reduced model for a criterion variable. As was outlined in the introduction, high intercorrelations pose challenges to statistical tests and can result in models that are impossible to correctly interpret (e.g., Stewart, 1988). Nonetheless, for judgment analysis, an examination of the environment mandates a representative design, one that involves including relevant predictors even if those predictors redundantly relate to the environment (Brunswik, 1952; Cooksey, 1996). The current study is interested in comparing the performance of modeling methods in a task environment that accurately represents a real-world judgment environment, that is, that includes human interaction with cue redundancy. Thus, despite the inter-correlations amongst the twelve nutrients known to be related to NuVal, no reduction of this predictor pool was made prior to the application of the three analytic methods. The three analysis methods—stepwise regression, all-subsets method, and random forest—and their respective results are described below.

Stepwise Linear Regression

Stepwise linear regression was utilized to extract a reduced model of the environment of cereal products. The full initial model contained the twelve nutrient variables known to be included in NuVal as earlier stated: calories, total fat, saturated fat, polyunsaturated fat, monounsaturated fat, sodium, potassium, dietary fiber, sugars, total carbohydrates, protein, and the number of NuVal vitamins and minerals listed on the NFP. The stepwise process resulted in a reduced model containing three variables: sugars (β = -.50, $p < .001$), potassium (β = .44, p = .01), and sodium (β = -.37, $p < .001$). This

three-variable model significantly predicted the expert nutritional measure NuVal, $F(3, 69) = 28.83$, $p < .001$, with $R^2_{ADJ} = 0.537$.

All-Subsets Method

The environment was also analyzed using the all-subsets method and the predictor criticality analysis developed by Azen et al. (2001). The original sample of 73 cereals along with their NuVal score ratings and the nutrient predictors (p) associated with them were bootstrapped (resampled with replacement) to generate 100,000 samples of size 73. For each sample, 2^p -1 subset regression models—all possible models but the null model— were fitted. A 'best-fit' model for each sample was determined according to the model with the highest adjusted r-squared value. Subsequently, the model with the highest frequency of being the 'best-fit' across the entire distribution of 100,000 samples was selected as the overall best-fitting model. In modeling the environment, a reduced model including eleven predictors had the highest probability of being the best fit across the 100,000 bootstrapped samples. This eleven-predictor model had the highest fit amongst all models for 3,757 of the bootstrapped samples (3.76%). The model included all predictors but saturated fat, resulting in a final model that included the following variables: calories, dietary fiber, monounsaturated fat, polyunsaturated fat, potassium, protein, sodium, sugars, total fat, total carbohydrates, and number of NuVal-included vitamins and minerals. Adjusted R^2 values for this best-fitting model ranged from .380 to .910 ($M = 0.691$, $MDN = 0.695$, $SD = 0.075$).

The criticality of the predictors was determined based on the probability of a given predictors inclusion in the distribution of best fit model. Unlike traditional measures of variable importance that can generate rankings only for variables included in

a final model (such as stepwise regression's beta weights), predictor criticality analysis ranks all possible variables (Azen et al. 2001). Thus, predictor criticalities (*C*) are generated for all variables, not only those included in the model determined to be the best fit. Among the eleven predictors included in the final best-fitting model, criticalities ranged from .307 to .995: sodium (*C* = .995), sugars (*C* = .990), dietary fiber (*C* = .915), calories (*C* = .849), number of NuVal included vitamins and minerals (*C* = .773), total carbohydrates (*C* = .737), monounsaturated fat (*C*= .709), potassium (*C* = .645), protein (*C* = .611), polyunsaturated fat (*C* = .588), and total fat (*C* = .580). The one predictor that was not included in the final, best-fitting model was also the predictor with the lowest predictor criticality, saturated fat (*C* = .307).

Random Forest Method

The random forest method also began with the initial inclusion of all nutrient predictors. The variable selection method used in this study was the approach of the VSURF package developed by Genuer et al. (2015) and previously discussed in the Random Forest section. In accordance with Genuer et al. (2015) and Behnamian et al.'s (2017) research indicating that a higher number of trees to a certain point improves the stability of the variable importance estimates and ranking, the default VSURF ensemble number of 2000 trees per forest was used. That is, for each forest in the variable selection process of the environment analysis, NuVal scores were resampled with replacement to generate 2000 bootstrapped datasets of 73 scores each. The number of predictors that would be randomly selected to be considered at each split was set at four in accordance with the VSURF default of selecting number of predictors (*p*) according to rounding down *p*/3, where *p* is the total number of predictors included in the model. To determine

the minimum threshold for maintaining variables, the VSURF default of 50 forests were generated, and the default of 25 forests for the interpretation step was used.

Because random forest is an aggregated tree process in which predictors can be used repeatedly in recursive splitting, random forests have no equivalent to the clear directional coefficients that are estimated in linear regression (Strobl, Malley & Tutz, 2009). Instead, the primary result of interest is the extent to which each predictor variable contributes to the model's predictive capacity. In modeling NuVal Scores, the thresholding step eliminated two variables, saturated fats ($\Delta MSE_{OOB} = -0.101$) and monounsaturated fats ($\Delta MSE_{OOB} = 0.170$), from the initial pool of 12 nutrient variable. The 10 predictors selected by the thresholding step included sugars ($\Delta MSE_{OOB} = 107.47$), sodium ($\Delta MSE_{OOB} = 70.78$), dietary fiber ($\Delta MSE_{OOB} = 23.60$), total carbohydrates ($\Delta MSE_{OOB} = 22.74$), potassium ($\Delta MSE_{OOB} = 16.77$), calories ($\Delta MSE_{OOB} = 8.36$), protein ($\Delta MSE_{OOB} = 7.48$), number of NuVal-included vitamins and minerals ($\Delta MSE_{OOB} = 4.46$), total fat ($\Delta MSE_{OOB} = 3.00$), and polyunsaturated fats ($\Delta MSE_{OOB} = 1.44$). Percent reduction in MSE refers to the predictor's ranked mean importance in the across the thresholding forests. That is, the MSE values above describe the percent increase in MSE_{OOB} when a given predictor is permuted.

At the interpretation step the predictor set was further reduced, resulting in a model that included the three nutrient variables with the highest variable importance values: sugar, sodium, and dietary fiber. Using the $R^2_{Adj.RF}$ measure found in Equation 6, this three-predictor random forest model was found to have an $R^2_{Adj.RF}$ of .879.

Method Comparisons

The three methods had clear differences in size. The all-subsets method yielded the largest final model, one that excluded only a single nutrient, saturated fats. Both the stepwise and random forest methods resulted in models with three nutrients. The stepwise regression method's final model included sugars, sodium, and potassium. The random forest method's final model included sugars, dietary fiber, and sodium. As can be seen in Table 1, dietary fiber has no relationship with sugars ($r = .15$, $p > .05$) or sodium ($r = -.23$, $p > .05$). In contrast, potassium is significantly related to sugars ($r = .33$, $p < .05$), but not to sodium ($r = -.08$, $p > .05$). Sugar and sodium have a positive relationship ($r = .27$, $p < .05$).

Table 2 depicts the variable importance results for the three methods as well as the results of a regression model containing all nutrient variables. A comparison of the findings depicted in Table 2 shows that a stepwise regression method results in the selection of variables other than those deemed significant in a full regression model. A full regression model containing all 12 nutrients indicates that calories, dietary fiber, sodium, and sugars are statistically significant (at the $p < .05$ level) in predicting NuVal. Using the size of the standardized coefficients to rank the importance of the variables would indicate that calories ($\beta = 1.309$, $p < .05$) is the most important variable, followed, in descending order, by total carbohydrates ($\beta = -1.091$, $p > .05$), sugars ($\beta = -0.494$, $p < .05$), dietary fiber ($\beta = 0.438$, $p < .05$), sodium ($\beta = -0.436$, $p < .05$), monounsaturated fat ($\beta = -0.277$, $p > .05$), total fat ($\beta = -0.208$, $p > .05$), potassium ($\beta = 0.191$, $p > .05$), polyunsaturated fat ($\beta = 0.186$, $p > .05$), number of NuVal-included vitamins and

minerals (β = 0.135, p > .05), saturated fat (β = -0.066, p > .05), and protein (β = -0.270, p > .05).

The stepwise regression method resulted in a final model that included three nutrient variables: sugars, potassium, and sodium. Here, standardized coefficients indicated that sugars (β = -0.499, p < .05) was the most important variable, followed by potassium (β = 0.436, p < .05) and sodium (β = -0.369, p < .05).

Table 2

Predicting NuVal from NFP Nutrients (N=73). Values in bold indicate predictors' inclusion in the final model.

	Full Model Regression	Stepwise Method	All-subsets method	Random Forest Method
	β	β	C	ΔMSE_{OOB}
Calories[a]	**1.309***	--	**0.849**	8.365
Total Fat (g)	**-0.208**	--	**0.580**	2.999
Saturated Fat (g)	**-0.066**	--	0.307	-0.101
Monounsaturated Fat (g)	**-0.277**	--	**0.709**	0.170
Polyunsaturated Fat (g)	**0.186**	--	**0.588**	1.440
Dietary Fiber (g)	**0.438***	--	**0.915**	**23.600**
Potassium	**0.191***	**0.436***	**0.645**	16.767
Protein	**-0.270**	--	**0.611**	7.480
Sodium (mg)	**-0.436***	**-0.369***	**0.995**	**70.785**
Sugars (g)	**-0.494***	**-0.499***	**0.990**	**107.473**
Total Carbohydrates (g)[a]	**-1.091**	--	**0.737**	22.739
NuVal Vitamins & Minerals (#)	**0.135**	--	**0.773**	4.457
R^2_{ADJ}	.571	.537	M = .691 (SD = .075)[b]	.879

*$p \leq 0.05$, two-tailed. [a]Beta value > 1 indicative of high multicollinearity. [b]Mean and standard deviation of adj. R^2 across 3,757 all subsets bootstrapped samples in which this model was found to be the best fitting.

The all-subsets method predictor criticality method included eleven of the twelve nutrients in its final best-fitting model, excluding only saturated fats. Predictor criticality

rankings of the nutrients' importance in predicting NuVal varied from the importance implied by the full model regression and the selections of the stepwise regression approach. The highest ranked predictors by the all-subsets method were sodium (C = 0.995), sugars (C = 0.990), and dietary fiber (C = 0.915), followed by calories (C = 0.849). High rankings of these variables were congruent with the nutrients that were statistically significant in the full regression model. Although in the full regression model potassium was statistically significant, it was ranked eighth in terms of both criticality and beta weight size.

The three nutrient variables included in the random forest's final model were also partially congruent with the nutrients stastically significant in the regression model. Sugars (ΔMSE_{OOB} = 107.473) was ranked the highest, followed by sodium (ΔMSE_{OOB} = 70.785) and dietary fiber (ΔMSE_{OOB} = 23.600). Potassium, which was statistically significant in the full and stepwise regression models, was ranked fifth by the random forest metric (ΔMSE_{OOB} = 16.767).

For both the all-subsets method and the random forest method, the inclusion of final variables was consistent with their metric for variable importance. This is not surprising given that in both these methods the process of variable selection and importance ranking are related. However, the results in Table 2 do provide insight into how the all subsets predictor criticality and the random forest variable importance measures vary from the beta weights in stepwise regression. Unlike the two other methods, the beta weights size in the full model are not necessarily indicative of their significance in a first model or their inclusion in a final stepwise model.

Several other conclusions can be drawn from comparing the results of the three methods. First, the size of the all subsets final model is dramatically larger than the other two methods. Despite the use of a random forest selection procedure that aims to retain all relevant predictors (i.e., it results in a final model that includes more nutrients than would a procedure purely concerned with prediction), the random forest final model includes only three nutrients. This final model size is congruent with that of the stepwise procedure, which also included three predictors. This indicates that given the current task environment, the stepwise regression and random forest methods may result in more succinct judgment models than the all-subsets method.

A second line of comparison is the goodness-of-fit of each method. Compared to the stepwise regression method, the all subsets average adjusted r-squared ($\overline{R^2_{ADJ}}$ = .691) was higher than that of the stepwise regression method (R^2_{ADJ} = .537) indicating that the inclusion of additional variables in the model improved the fit of the model in predicting NuVal even after a penalization for added variables. The three-variable random forest model (R^2_{ADJ} = .879), however, had a higher fit than either of the alternative methods. These results indicate that in the given task environment, the random forest method may result in the most parsimonious models compared to the all subsets and stepwise methods.

Descriptive Analysis of the Environment

The importance of understanding the task environment is highlighted by cognitive continuum theory (CCT). In contrast to the traditional dichotomization of intuitive and analytic thinking, CCT proposes that both cognitive processes and task conditions lie on a continuum from intuition to analysis (Hammond et al., 1987). Hammond et al.

distinguished intuition and analysis according to the degree of cognitive control, rate of data processing, conscious awareness, organizing principle, errors, and confidence level involved in judgment. In CCT, intuition is described as a cognitive process that is low in cognitive control and occurs at a rapid rate of data processing. Intuition is posited to be associated with little conscious awareness and its organizing principle is proposed to be that of weighting averages with normally distributed errors. The intuitive process leaves judges with high confidence in their answers, but low confidence in the method with which their answers were attained (Hammond et al., 1987). In contrast, CCT describes the analytic process as high in cognitive control, with a slow rate of data processing, and with high conscious awareness of the cognitive process. The organizing principle of the analysis process is task specific and errors in the process tend to be few but, when they do occur, they are large. The analytic process leaves judges with low confidence in their answers, but high confidence in the method they employed to reach them (Hammond et al., 1987). Importantly, CCT proposes that quasi-rationality lies between analytic and intuitive cognition and includes properties from both types of cognition (Hammond et al., 1987).

The properties of a task greatly impact an individual's decision-making (Hammond et al., 1987). Hammond et al. outline specific task characteristics that induce intuitive or analytic cognitive processes with a mix of both characteristics inducing a quasi-rational process between the two. The task characteristics outlined by the authors include: the number of cues, their measurement, distribution, the redundancy among cues, the decomposition of the task, the degree of certainty in the task, the relation between the cues and the criterion, the weighing of cues in the environmental model, the

availability of an organizing principle, the display of cues, and the time period of the task (Hammond et al., 1987). The following states of those characteristics are likely to induce intuition: a large number of cues (> 5); perceptual measurement of cues; continuous, highly variable distribution of cues; high redundancy among cues; low decomposition of the task; low certainty in the task; a linear relation between cues and criterion; equal weighting of cues in an environmental model of the judgment task; lack of availability of an organizing principle; simultaneous display of cues; and a brief time period for the task. In contrast, the following task characteristic conditions are likely to induce analytic judgment: a small number of cues (< 5); objectively and reliably measured cues; cues with an unknown distribution or cues that are dichotomous and judgment values that are discrete; low redundancy among cues; high decomposition of the judgment task; high certainty in the judgment task; a nonlinear relationship between the cues and the criterion; unequal weighting of cues in the environmental model of the judgment task; an available organizing principle; cues that are displayed sequentially; and a long time period for the task (Hammond et al., 1987).

The task characteristics that induce intuitive and analytic thinking can be further distinguished by the extent to which knowledge about them is available to the judge (Hammond et al., 1987). Surface characteristics refer to properties of the task that are overtly present in the display of the task variables to the judge. Depth characteristics refer to the underlying relationships among the variables within the task that are not obvious to the judge. A task's surface and depth characteristics may be congruent with each other, or they may conflict. Examples of surface-depth congruence and conflict are provided in the later description of the current study's task characteristics.

Hammond et al. (1987) developed the concept of a task continuum to parallel that of a cognitive continuum. Specifically, the authors proposed eight task properties and provided insight on the measurement of these properties to compare tasks. Below we apply the Hammond et al. (1987) approach to the nutritional judgment task of this study to systematically describe the judgment environment.

The first subindex of the Task Continuum Index (TCI) developed by Hammond et al. (1987) is the number of cues presented in the task. Here, the distinction between surface and depth conditions becomes helpful. On the surface, many possible cues are available to the judge; the ingredients present or lacking in the cereal, the nutritional values in the entire cereal or per serving, the name of the product, or even colors or other details of the product image. The depth condition, in contrast, includes only the general information relevant to the criterion of nutritional quality. As the NuVal criterion relies on nutrient information to calculate a products nutritional score, the values of 16 nutrient variables included on the Nutrition Facts Panel were considered by the researchers as the relevant cues.

The second subindex of the TCI is the redundancy among cues. On the surface, information about cue redundancy can be acquired through examination of the Nutrition Fact Panel or may be available to participants through personal knowledge. For instance, on the Nutrition Fact Panel, total fat is subdivided into different categories of fat. These different categories of fat are to some degree redundant with that of total fat. A knowledgeable consumer might also be aware that a food's caloric density is supplied entirely by its carbohydrates, fats, and protein, and thus there is some redundancy among these three cues and a product's calories cue. Nonetheless, the depth condition of cue

redundancy, the average extent to which cue redundancy exists among the cues is not accessible to participants. Cue redundancy is measured by the mean intercorrelation among all relevant cues. For the cereals used in the study, cue intercorrelations ranged from -0.40 to 0.95 with a median of 0.22.

The third subindex is the reliability of cue measurement which is measured by the mean intercorrelation among the cue judgments of several experts. As the expert standard used in this study was algorithmically derived rather than elicited from multiple experts, mean intercorrelation among cue judgments could not be obtained.

The degree of task decomposition, the fourth subindex, is a qualitative measure of the extent to which the presentation of the task decomposes, and thus facilitates, the task for the subject. On a surface level, the judgment task in the experimental task was greatly decomposed, as cereals were presented one at a time with all their relevant information included. High decomposition of the task is associated with analysis-inducing states. However, the display of cues for each cereal were simultaneous, rather than sequential. Hammond et al. (1987) suggest that simultaneous cues are likely to induce intuition. Ease of use of cue display may also have varied according to experimental condition. As Carter and Gonzalez-Vallejo's (2018) original conditions included three display conditions, two of which highlighted particular nutrient values, it is possible the task appeared more decomposed in the highlighted conditions. Indeed, the expectation that judgment would be impacted by this nutrient highlighting was a key hypothesis (Carter & González-Vallejo, 2018). This hypothesis was not confirmed, but it is possible the highlighted label resulted in perceived procedural ease without influencing judgment accuracy.

The fifth subindex is the availability of an organizing principle to the judge. On the surface, no organizing principle was available to this study's judges. Participants were instructed to answer the questions as best as they could to represent their opinions and, besides being told to use the information available, no guidance was given as to how to rate products. At a depth level, an organizing principle of nutritional quality was likely unavailable. Although individuals may adopt personal organizing principles to facilitate their dietary choices, a single, objectively accurate organizing principle for determining nutritional quality of food products is unlikely to be accessible to participants. A 2017 survey conducted by the International Food Information Council Foundation (IFICF) found that 78% of a 1,002 sample of Americans reported encountering conflicting information on what to eat with 56% of those participants reporting that this conflicting information made them doubt their food choices (IFICF, 2017). Previous research corroborates these findings, indicating that even consumers who are interested in healthy eating are likely to acquire nutrition information from various sources, including many unreliable ones (Cornish & Moraes, 2015). Cornish and Moraes' (2015) interview-based exploration also found that nutritional literacy not only required informational knowledge, but also the capacity to accurately interpret that knowledge. As food products often amalgamate glorified and vilified nutrients, determining the nutritional quality of any one product involves judgments that are far more complex than simply understanding the health impact of individual nutrients (Guthrie, Derby, & Levy, 1999).

The final three subindices of the TCI are informed by the statistical analysis of the environment previously described. As the comparison of analytic methods in extracting

models is a key aspect of the current study, all three methods employed in statistically modeling the environment are used in describing its complexity.

The sixth subindex is the degree of nonlinearity in the optimal organizing principle. Although the task itself cannot be classified as one in which an organizing principle is accessible or even available, the criterion selected for the study provides a measure of the optimal organizing principle for rating nutritional judgment. Degree of nonlinearity in the optimal organizing principle is measured by the difference between the R^2 of the appropriate nonlinear model and the R^2 of the best fit linear model of the environment (Hammond, et al. 1987).

The algorithm for determining NuVal is known to be nonlinear, but it is not publicly available. However, presumably the correct model of the cue nutrients in predicting NuVal would have an R^2 of one (i.e., it is deterministic). Thus, the degree of nonlinearity in the optimal organizing principle can be estimated by subtraction of one minus the r-square of known models. Due to the examination of reduced models, the R^2_{ADJ} can be used to estimate the nonlinearity of the model. The full model R^2_{ADJ} of .571, indicates a nonlinearity of .429; the stepwise model R^2_{ADJ} of .537 results in a nonlinearity equal to .463. As a range of R^2_{ADJ} values are provided for the all subsets approach, the mean R^2_{ADJ} of .691 is reported in Table 2 and was used in calculating an estimated non-linearity of .309. Additional insight into the possible non-linearity of the NuVal algorithm is indicated by the fit of the random forest method—a non-linear approach—compared to the highest R^2_{ADJ} of a linear procedure. As shown in Table 2, the random forest method resulted in a model with an R^2_{ADJ} of .879, a higher fit than that of the all-subsets method ($\overline{R^2_{ADJ}}$ of .691).

The extent to which the cues are weighted equally in the optimal organizing principle is measured using the standard deviation of the beta weights from the linear environment models shown in Table 2. This seventh subindex was determined according to the linear weights of the nutrient variables in modeling NuVal. That is, the standard deviation of beta weights can be calculated for the linear methods as these methods estimate coefficient weights for each nutrient. The standard deviation of weights cannot be calculated for the random forest method as this method is non-linear and does not generate predictor coefficients as linear methods do. For the full model method, the standard deviation for the 12 beta-weights shown in Table 2 was $SD = .587$. Given only three nutrients were included in the stepwise model, the standard deviation of beta weights ($SD = .506$) was calculated using the three coefficients for those included nutrients. For the all-subsets method, beta weights were determined according to the mean beta weights for each nutrient across the 3,757 models for which the eleven-variable model shown in Table 2 was found to be the best-fitting model. From these averaged weights[4], a beta-weights standard deviation of .695 was calculated.

The eighth and final subindex in determining the TCI is the degree of certainty in the task system. This subindex is measured by the R^2 of the environmental model. Assumedly, the included nutrients in the correct model for NuVal score would result in an R^2 of one, but, as previously noted, the algorithm for determining the NuVal is unknown. However, as the NuVal model is known to be nonlinear the goodness-of-fit of

---

[4] The mean beta-weights calculated across the 3,757 models for which the final best-fitting, eleven-model variable model was found to be the best-fit were as follows: calories ($\bar{\beta} = 1.44$), total fat ($\bar{\beta} = -0.428$), monounsaturated fat ($\bar{\beta} = -0.206$), polyunsaturated fat ($\bar{\beta} = 0.286$), dietary fiber ($\bar{\beta} = 0.484$), potassium ($\bar{\beta} =0.279$), protein ($\bar{\beta} = -0.340$), sodium ($\bar{\beta} = -0.454$), sugars ($\bar{\beta} =-0.496$), total carbohydrates ($\bar{\beta} = -1.25$), number of NuVal vitamins and minerals ($\bar{\beta} = 0.175$).

the random forest, a non-linear model, may provide greater insight into the degree of certainty in the environment. The $R^2_{Adj.RF}$ for the random forest reduced model, used as a goodness-of-fit metric for the random forest was .879, indicating a high degree of certainty in the task environment.

Several key conclusions can be drawn from the analysis of the environment. First, the number of cues in this task, at both surface and depth levels, is large and would be expected to induce an intuitive cognitive approach (Hammond et al. 1987). Redundancy among cues is high, indicating that this redundancy would lead the task to be more intuitive than analytic. The task, though decomposed across cereals, presented cues simultaneously rather than sequentially, increasing the likelihood of an intuitive process being induced. According to CCT, the lack of an available organizing principle would increase the likelihood of an intuitive cognitive process. The nonlinearity of the optimal organizing principle, the inequality of weights in the environment, and the high certainty in the task environment, in contrast, would increase the likelihood of an analytic cognitive process.

Conclusions from Examining the Task Environment

Several conclusions can be drawn from the statistical and descriptive analyses of the environment. The statistical analysis of the environment found that the environment of this nutritional judgment task is one in which predictors' inter-correlations are plentiful and often high. In this environment, the random forest method was found to extract a better-fitting model than the two linear methods. It was previously known that the NuVal is non-linear, but this finding confirms that a tree-based decision model is a better representation of the NuVal algorithm than a linear model. The descriptive analysis of the

environment indicates that the nutritional judgment task is unlikely to elicit a fully intuitive or fully analytic cognitive approach. Rather, like most tasks, it lies on the continuum between intuition and analysis and likely elicits a quasi-rational cognitive process that includes elements of both intuition and rationality (Hammond, 1980).

Given this analysis of the environment the following studies examining the nutritional task environment were expected to inform several questions relevant to judgment analysis. First, these studies should clarify the comparative performance of the stepwise, all subsets, and random forest methods for a high-redundancy environment in which the optimal organizing principal is unknown and nonlinear. Second, these studies' results have implications for understanding the underlying judgment process of quasi-rational cognition. Namely, whether a linear model such as the stepwise and all-subsets methods provides a better description of quasi-rational human judgment than a non-linear process, specifically a decision tree model.

CHAPTER 3: STUDY ONE

The first study used archival, empirical data in order to compare the performance of the three modeling techniques: stepwise regression, the all subsets method, and the random forest method. As described in the previous section, the nutritional task environment studied in Carter and González-Vallejo (2018) was one that contained high cue intercorrelation and was expected to elicit a cognitive approach that was neither fully intuitive nor fully analytic. This study was designed to explore key differences in the modeling techniques used to analyze participant judgment.

First, modeling results were expected to vary across participants with the random forest method, on average, resulting in more predictive models (as measured by R^2_{ADJ}) and more generalizable models (measured by R^2_{CV}) than the stepwise regression and all-subsets methods. The all subsets technique was expected to have better predictive and generalizable performance than the stepwise regression approach. Second, methods were expected to vary in terms of the number of predictors that were included in their final models. Across individuals, random forest methods were expected to have the smallest models on average; the all-subsets method was expected to result in best-fit models that are smaller on average than stepwise regression models.

Methods

Participants

Data collected for a research study conducted in 2016 supplied the empirical data for this study. In the original study (Carter & González-Vallejo, 2018), 298 participants of at least 18 years of age who were recruited through the psychology participant pool at Ohio University and received course credit for their participation completed the study's

judgment task. Of the 297 participants from whom demographic data was collected, the majority were female (56.9%), white (88.2%), and in their first year of college (64.3%). Participants' ages ranged from 18 to 25 (M = 19, SD = 1.23) years. Carter and González-Vallejo (2018) used data only from those participants who completed the entire study in their analysis. The current study uses data from 298 participants, including an additional participant who did not complete the study in its entirety but completed the judgment task portion of the study. Among the 298 participants, 206 (69%) did so online, while 92 (31%) did so in the lab. Both versions of the study were identical and administered via Qualtrics.

Design

On beginning the study, participants were randomly assigned to one of the three experimental conditions. In every condition, after being presented with information on the study and consenting to participate, participants were presented with a cereal name, an image of the front of the cereal box, along with the cereal's Nutrition Fact Panel (NFP), and list of ingredients. Depending on the experimental condition, the cereal display showed the FOP1, or FOP2, or only the NFP. Figure 4 depicts an example of the stimuli seen by participants in the FOP1 condition (see Appendix A for an example of all three conditions). Participants in each of the three conditions were presented with 74 cereals, and after viewing each cereal they were asked several questions regarding the product. Following this, participants in each condition completed health behavior, nutrition knowledge, numeracy, and demographic measures.

Figure 4. Example of FOP1. An image of a cereal box is accompanied by its ingredients, NFP label, and a FOP label with highly relevant nutrients.

Measures[5]

Nutrition Rating Cereal Questions

For each cereal, participants responded to a set of nine questions regarding their opinion of the health value and nutritional quality of the product, their familiarity with and consumption of the product, and the likelihood they would purchase it. Questions were answered on a scale from 1-100, except for three questions: the question requesting frequency of consumption (free response), factors impacting purchase (free response), and the question asking participants whether they would add the item to a shopping cart

[5] Additional measures were taken and are described at length in Carter and González-Vallejo (2018). These are omitted from the current document as they are not relevant to the current study's design.

(yes/no). Of particular interest are the judgment variables assessed with the following questions: “How healthy is this cereal?” and “How nutritious is this cereal?” which participants were asked to rate on a scale of 1 (not healthy/nutritious at all) to 100 (extremely healthy/nutritious). Across participants, responses to these questions for each item were highly correlated, (*r*’s ranging from .714 to .932). An average of the participants’ responses to questions concerning the health and nutrition value of each individual cereal was used as the “judgment” variable.

Analysis and Results

To compare the efficiency and results of the three variable selection methods, each of the three analyses, stepwise regression, all subsets method, and random forest method were first used to extract a judgment model for each participant. Prior to the analyses, a group of 40 judgments were randomly sampled from each participant; these judgments were used to extract judgment models for each participant using the nutrients from the NFP. As outlined in the environment section, a total of 16 predictor variables are available for inclusion in the participants’ judgment models. Once the method employed generated a model of judgment, that model was used to predict judgments in each of the respective participants remaining cases (33 in the majority of, but not all, cases) which had not been used to generate the models. That is, the extracted models were cross-validated to determine their generalizability to new data. The squared correlation of the predicted and observed judgment values (the cross-validated R^2, hereafter denoted R^2_{CV}) was computed for each participants’ judgments and used as a measure of model generalizability. All analyses were conducted using statistical packages in R and all code utilized for the analyses is reported in Appendix B.

Stepwise Regression Variable Selection

In the first analysis, stepwise linear regression was applied to extract reduced models for each of the participants' judgments. For each of the 298 participants, stepwise regression was used to generate a linear model relating the 40 randomly selected nutritional judgment ratings to the nutrient information from the NFP. Once a stepwise regression had generated a model for each participant, that model was used to predict the cases that had not been used for model generation.

Stepwise regression successfully extracted linear models for 98.66% ($N = 294$) of participants. For the 294 participants for whom models were successfully extracted, 287 of the models were statistically significant at the $p < .05$ level. Thus, stepwise regression extracted statistically significant linear models for 96.31% of the 298 participants. Among participant for whom models were generated, R^2_{ADJ} values ranged from 0.03 to 0.86 ($M = 0.43$, $Mdn = 0.43$). Among the same 294 participants, values of R^2_{CV} ranged from 0.00 to 0.79 ($M = 0.18$, $Mdn = 0.14$). The number of predictors included in stepwise models ranged from 1 to 10, with a mean number of predictors of 4.55 ($Mdn = 4$).

All Subsets Variable Selection

As with the stepwise regression analysis, the 40 nutritional ratings randomly selected from each participants' set of judgments previously were used in the all-subsets method. For each participant, 100,000 bootstrapped datasets were generated from their randomly selected set of 40 observations. That is, in keeping with the idiographic approach of judgment analysis, bootstrapped datasets of size 40 were drawn for each individual participant and all possible models were subsequently fit to each of the bootstrapped samples. Once all possible models were fit, the goodness-of-fit criterion,

R^2_{ADJ}, was used to determine the best-fit model for each of the 100,000 datasets. The probability distribution of the best-fit model was used to determine the final best model for each participant; the model with the highest probability of being chosen as the best-fit model was chosen as the final judgment model. Lastly, the predictor variables' criticality was also determined according to the probability that it was included in a best-fitting model. Predictor criticality will be used as the all subset method's measure of the relative impact of predictor variables.

Once a model had been estimated for each of the participants, that model was used to predict participants' judgments in the cases that had not been randomly selected for model generation. Predictor weights were determined according to mean coefficient values across the total number of bootstrapped datasets that had generated the best-fit model. For example, for a participant whose best-fit model had the highest R^2_{ADJ} in 500 of the 100,000 bootstrapped datasets, the coefficient of calories in the model predicting their judgments for the remaining cases would be the mean of the calories estimate over the 500 bootstrapped datasets.

The all-subsets methods successfully extracted a linear model for all participants (N = 298). The all-subsets method provides performance results for all bootstrapped samples in which the final model was the best-fit model. Thus, the final model has multiple R^2_{ADJ} values. To summarize these values, a mean R^2_{ADJ} value was computed for each participant. These means ranged in value from 0.14 to 0.97, with a grand mean R^2_{ADJ} of 0.85 (Mdn = 0.87).

Values of R^2_{CV} for the all subsets models ranged from 0.00 to 0.61 (M = 0.12, Mdn = 0.08). The number of predictors included in best all-subsets models ranged from 3 to

16. Unexpectedly, the vast majority of final all-subsets models were not reduced from the original set of 16 variables. The mean number of predictors included in final all-subsets models was 15.47 (*Mdn* = 16), with 269 (90.27%) of the participants having best-fitting models that included the entire initial set of 16 variables.

Random Forest Variable Selection

As with the previous analyses, the 40 nutritional ratings randomly selected from each participants' set of judgments were used to develop random forest models of each participant's judgment. The random forest variable selection method developed by Genuer et al. (2015) was used to develop reduced models for each participant. The default parameters outlined in Genuer et al.'s (2018) VSURF package and described at length in Genuer et al. (2015) were used throughout the analysis. That is, for each participant's forest in the variable selection process of the analysis, judgment ratings were resampled with replacement to generate 2000 bootstrapped datasets of 40 judgments each; five predictors were randomly selected for consideration at each split; 50 forests were used to determine the minimum threshold for maintaining predictors and 25 forests were used at the interpretative step. Once an interpretive model had been generated for each of the participants, each participant's respective model was used to predict their judgments in the excluded cases.

Random forest successfully extracted tree models for all participants ($N = 298$). The goodness-of-fit measure $R^2_{Adj.RF}$ was used as a comparable measure to the R^2_{ADJ} measures employed by the stepwise regression and all-subsets methods. Across random forest models, $R^2_{Adj.RF}$ values ranged from 0.04 to 0.93 ($M = 0.70$, $Mdn = 0.73$). Values of R^2_{CV} for the random forest models ranged from 0.00 to 0.87 ($M = 0.22$, $Mdn = 0.19$). The

number of predictors included in random-forest models ranged from 1 to 10 with a mean number of predictors of 3.58 (*Mdn* = 3).

Method Comparisons

Differences in methods across participants can be seen by an examination of the central tendencies of the three methods. Across participants, methods differed in their model GOF and generalizability, as well as in the number and make-up of model predictors. Averages in model fit, model generalizability, and number of predictors for all participants for whom models could be extracted, are depicted in Table 3.

Table 3

Average Model Characteristics across Participants

		Stepwise Regression ($N = 294$)	All Subsets[a] ($N = 298$)	Random Forest[b] ($N = 298$)
R^2_{ADJ}	Range	0.03 – 0.86	0.14 – 0.97	0.04 – 0.93
	Mean (*SD*)	0.43 (0.18)	0.85 (0.08)	0.70 (0.13)
	Median	0.43	0.87	0.73
	Quartiles 1 & 3	0.30, 0.56	0.83, 0.90	0.65, 0.79
R^2_{CV}	Range	0.00 – 0.79	0.00 – 0.61	0.00 – 0.87
	Mean (*SD*)	0.18 (0.15)	0.12 (0.12)	0.22 (0.18)
	Median	0.14	0.08	0.19
	Quartiles 1 & 3	0.04, 0.28	0.02, 0.20	0.06, 0.36
Predictors Included	Range	1 – 10	3 – 16	1 – 10
	Mean (*SD*)	4.55 (2.08)	15.47 (1.78)	3.58 (1.79)
	Median	4	16	3
	Quartiles 1 & 3	3, 6	3, 16	2, 5

[a]As all subsets results provide a distribution of goodness of fits, R^2_{ADJ} values provided are summary statistics of aggregated adjusted R^2 values across best-fitting bootstrapped samples.
[b]As described in text, R^2_{ADJ} values for random forest models were computed using the $R^2_{Adj.RF}$ measure defined in Equation 6.

Table 3 includes all participants for whom models could be extracted. However, average cross-validity and number of predictors vary when considering only a subset of participants for whom adjusted r-squared values were 0.45 or higher. In that case, the mean and median cross-validity values for the all subsets method (N = 297) remain the same as in all cases, as does median predictor inclusion. For 285 participants with random forest models with adjusted r-squared values equal to or greater than 0.45, cross-validity ranges from 0.00 to 0.87 with M = 0.23 (SD = 0.18) and Mdn = 0.20. Median predictor inclusion for the subset of random forest models remains at 3. Among 139 participants with stepwise regression models that have an adjusted r-square value equal to or greater than 0.45, cross-validity values range from 0.00 to 0.79, M = 0.23 (SD = 0.16) and Mdn = 0.21. The median predictor inclusion for the stepwise method when only those models with adjusted r-squared equal to or higher than 0.45 are considered is 6.

Considering only a subset of models that have an adjusted r-squared equal or above a value of 0.70 decreases the number of participant models substantially. The central tendencies in cross-validity for participants that had all-subsets models' adjusted r-squared values equal or higher than 0.70 (N = 283) were M = 0.12 (SD = 0.12) and Mdn = 0.08. Among the participants with random forest models that had an adjusted r-squared equal to or higher than 0.70 (N = 177), cross-validity was higher on average—M = 0.29 (SD = 0.18) and Mdn = 0.30—than when all participants were considered. Median predictor inclusion for this subset of models was 4. Cross-validity values were also higher among participants with stepwise models that had an adjusted r-squared equal to or higher than 0.70. Although this group was very small (N = 24), cross-validity values were, on average, higher than those of either other method, M = 0.33 (SD = 0.17) and

Mdn = 0.33. Median predictor inclusion (*Mdn* = 6.5) was also higher for this subset of participants. These differences indicate that for higher adjusted r-squared values, the stepwise regression models have greater or approximately equal cross-validity to those of the random forest method, however, stepwise regression is less likely to produce good-fitting models.

In addition to differences in model size, methods also varied in the degree to which predictors were included in individual models. Table 4 shows the proportion of participant judgment models in which each nutrient predictor was included. The most unexpected finding highlighted by Table 4 is the extent to which the final all subsets models include all predictor variables. Inclusion variation for the all-subsets method contrasts sharply with the two other methods which are far more varied in their inclusion rates. Despite this, in none of the methods is a predictor selected in 100% of the models. This is not unexpected and is perhaps even comforting, given that these summaries reflect individual judgment models that would, presumably, vary according to individual.

Table 4

Study One Descriptive Results: Proportion of Models Including Nutrient Predictors

	Stepwise Regression ($N = 294$)	All Subsets ($N = 298$)	Random Forest ($N = 298$)
Calories	53 (18%)	291 (98%)	55 (18%)
Calories from Fat	46 (16%)	292 (98%)	36 (12%)
Total Fat (g)	44 (15%)	287 (96%)	17 (6%)
Saturated Fat (g)	58 (20%)	286 (96%)	11 (4%)
Monounsaturated Fat (g)	67 (23%)	289 (97%)	13 (4%)
Polyunsaturated Fat (g)	57 (19%)	287 (96%)	16 (5%)
Dietary Fiber (g)	98 (33%)	286 (96%)	90 (30%)
Soluble Fiber (g)	68 (23%)	284 (95%)	25 (8%)
Insoluble Fiber (g)	105 (36%)	293 (98%)	22 (7%)
Potassium	104 (35%)	289 (97%)	149 (50%)
Protein	113 (38%)	284 (95%)	178 (60%)
Sodium (mg)	83 (28%)	286 (96%)	72 (24%)
Sugars (g)	169 (57%)	289 (97%)	173 (58%)
Total Carbohydrates (g)	49 (17%)	292 (98%)	77 (26%)
Other Carbohydrates (g)	62 (21%)	281 (94%)	24 (8%)
Vitamins & Minerals (#)	161 (55%)	294 (99%)	109 (37%)

The primary question of this investigation is the level of concurrence of the final models extracted by these three modeling approaches. As models are extracted on an individual-level, this question refers to the concurrency of models on the individual level, that is, the extent to which a model extracted for a single participant by the stepwise regression method matches the model extracted for the same participant by the all-subsets method and the extent to which both aforementioned models match the model extracted for the same participant by the random forest approach. The descriptive analysis outlined

above does not directly address this primary question, but it does indicate that the three modeling methods do not result in equivalent outcomes.

To investigate differences in model fit and generalizability, a multivariate mixed model was conducted using a Bayesian approach. A Bayesian approach was utilized as the dependent variables, R^2_{ADJ} and R^2_{CV} were beta distributed, which is not accounted for by traditional generalized linear mixed models (Bonat, Ribeiro Jr, & Shimakura, 2015). A Bayesian approach to multilevel models can model beta-distributed variables and, in addition, Bayesian analysis has the advantage of being appropriate to the data structure without relying on approximation assumptions made by traditional null hypothesis testing approaches (such as homogeneity of variances and normally distributed errors) (Bonat, Ribeiro Jr, & Shimakura, 2015; Kruschke, 2015).

For each model, four chains of 30,000 iterations were simulated, of which the first 2,000 were discarded as burn-in. No thinning was applied. The convergence of the models was assessed by calculating the multivariate potential scale reduction factor and by assessing trace and density plots (Brooks & Gelman, 1998; Kruschke, 2015). Posterior distributions were summarized using the mean and a 95% highest posterior density interval (HPDI). Highest posterior density intervals (HPDI) are reported rather than confidence intervals (CI) as HPDI are more appropriate for posterior distributions as they account for asymmetrical distributions (Kruschke, 2015). When a posterior distribution is asymmetric, a 95% CI does not include the 95% most likely parameter values whereas a 95% HPDI does. Given a symmetric and single-peaked posterior distribution, the 95% CI and 95% HPDI are identical (Kruschke, 2015). All models were implemented using the brms package (version 2.50) for R, which builds on Stan to perform Markov Chain

Monte Carlo Simulations using an adaptive Hamiltonian Monte Carlo (HMC) sampler known as the no u-turn sampler (NUTS) (Bürkner, 2018). The NUTS enables more efficient HMC sampling than other methods by using a recursive algorithm that automatically stops once it starts to retrace its steps, eliminating the need to input a number of steps parameter (see Hoffman & Gelman, 2014 for greater detail). Weakly informative priors were applied for all models.[6]

Multivariate multilevel models were implemented independently for fixed and random effect parameters and increasingly complex models were fit and tested. At level-two were 298 participants, at level-one were 890 models. The level-1 excluded missing stepwise regression models (i.e., missing R^2_{ADJ} and R^2_{CV} values, $N = 4$). For simplicity, results for each criterion variable are summarized in individual tables (see Tables 5 and 6). Multilevel models were compared using the Watanabe-Akaike Information Criterion (WAIC), a Bayesian approach to model comparison which uses the log pointwise posterior predictive density and adjusts for overfitting by adding a correction for the effective number of parameters in the model (Watanabe, 2010; Gelman, Hwang, & Vehtari, 2013). Lower values of WAIC imply higher predictive accuracy.

Intercept models for each of the criterions of interest were modeled separately in order to determine the variance components for each predicted variable (see Model 0 in Tables 5 and 6). For the GOF measure, R^2_{ADJ}, approximately 1% of the overall variance could be attributed to participant, however, a large proportion of variance in R^2_{CV} could be

---

[6] By default, brms uses weakly informative priors in order to minimize their influence on the results (Bürkner, 2018). Based on the recommendations by Gelman, Jakulin, Pittau, & Su (2008) intercept parameters in the logistic mixed-model analysis are by default given a half student-t prior with 3 degrees of freedom, a mean of 0, and a scale parameter of 10 (Bürkner, 2018). The Beta distribution inverse variance parameter phi is given a gamma prior with both alpha and beta parameter values equal to 0.01.

attributed to participant (ICC = 86%). Model 1 was a multivariate model including both criterion variables in the model. Intercept and variance components for Model 1 are included in the tables for completeness and ease of comparison to later variance terms.

Model 2 added the modeling method used to determine whether method used impacted GOF and generalizability (R^2_{ADJ} and R^2_{CV} respectively). Modeling method significantly improved prediction of model GOF and generalizability (Model 1 WAIC: -2406.9, Model 2 WAIC -3857.7). As expected, models generated by the stepwise method had smaller R^2_{ADJ} than models generated by the random forest method ($\bar{b}$ = -1.23, 95% HPDI = -1.30, -1.16). However, unexpectedly, the all-subsets method's models were associated with greater R^2_{ADJ} values than the random forest method's models ($\bar{b}$ = 0.92, 95% HPDI = 0.84, 1.00). The all-subsets method's models also had greater R^2_{ADJ} than the stepwise regression method's models ($\bar{b}$ = 2.15, 95% HPDI = 2.07, 2.23).

In predicting model generalizability (R^2_{CV}), method effect was again partially consistent with hypothesized outcomes. Consistent with expectations, both stepwise regression ($\bar{b}$ = -0.31, HPDI = -0.42, -0.19) and the all-subsets methods ($\bar{b}$ = -0.77, HPDI = -0.89, -0.65) were associated with lower R^2_{CV} values than the random forest method. Contrary to expectations, the all-subsets method was associated with lower R^2_{CV} than stepwise regression ($\bar{b}$ = -0.47, HPDI = -0.59, -0.34).

Model 3 added the experimental condition that each participant had been randomly assigned to in the original study conducted by Carter and Gonzalez-Vallejo (2018) to test whether experimental condition would affect model GOF and generalizability across participants. A WAIC model comparison of model two [WAIC = -3856.7] and model three [WAIC = -3855.5] indicated that including participant condition

in the model did not improve model fit in predicting R^2_{ADJ} and R^2_{CV}. As can be seen in Tables 5 and 6, 95% HPDIs for the condition coefficients all include zero. Experimental condition was dropped from subsequent models.

Model 4 added the random slopes for the level-1 method predictor. Model 4 [WAIC = -5986.9] was found to improve upon the previous highest model (Model 2, WAIC = -3856.7], indicating that there is between-person variability that explains the relationship between modeling method and the dependent variables R^2_{ADJ} and R^2_{CV}. That is, the effect of method on R^2_{ADJ} and R^2_{CV}, is, to some degree, dependent on participant.

Table 5

Summary of Fit Multilevel Analysis: Fixed and Random Effect Estimates and Highest Posterior Density Intervals (HPDI) for Predicting Goodness of Fit (R^2_{ADJ})

	Fixed Effects									
Parameter	Model 0		Model 1		Model 2		Model 3		Model 4	
	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI
Intercept	0.62	[0.56, 0.68]	0.63	[0.56, 0.69]						
Level 1 (Model)										
Stepwise vs. Random Forest					-1.23	[-1.30, -1.16]	-1.23	[-1.30, -1.16]	-1.30	[-1.38, -1.21]
All Subsets vs. Random Forest					0.92	[0.84, 1.00]	0.92	[0.84, 1.00]	0.96	[0.88, 1.03]
All Subsets vs. Stepwise					2.15	[2.07, 2.23]	2.15	[2.07, 2.23]	2.25	[2.17, 2.33]
Level 2 (Participant)										
NFP-Only vs. FOP1							-0.08	[-0.25, 0.09]		
FOP2 vs. FOP1							-0.03	[-0.19, 0.13]		
NFP-Only vs. FOP2							-0.05	[-0.22, 0.12]		
	Random Effects									
Residual variance[a]	0.26	[0.24, 0.28]	0.23	[0.21, 0.26]	0.04	[0.04, 0.05]	0.04	[0.04, 0.05]	0.00	[0.00, 0.01]
Intercept variance	0.00	[0.00, 0.01]	0.07	[0.04, 0.12]						
Random Slope Stepwise vs. RF									0.50	[0.41, 0.60]
Random Slope AS vs. RF									0.38	[0.32, 0.46]
Random Slope AS vs. Stepwise									0.44	[0.37, 0.53]

[a]Residual variance is measured as the inverse of Beta distribution parameter phi.

Table 6

Summary of Cross-Validity Multilevel Analysis: Fixed and Random Effect Estimates and Highest Posterior Density Intervals (HPDI) for Predicting Generalizability (R^2_{CV})

Parameter	Fixed Effects									
	Model 0		Model 1		Model 2		Model 3		Model 4	
	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI	Posterior Mean	95% HPDI
Intercept	-1.81	[-1.92, -1.69]	-1.81	[-1.93, -1.69]						
Level 1 (Model)										
Stepwise vs. Random Forest					-0.31	[-0.42, -0.19]	-0.31	[-0.42, -0.19]	-0.31	[-0.43, -0.19]
All Subsets vs. Random Forest					-0.77	[-0.89, -0.65]	-0.77	[-0.90, -0.65]	-0.76	[-0.89, -0.63]
All Subsets vs. Stepwise					-0.47	[-0.59, -0.34]	-0.47	[-0.59, -0.34]	-0.45	[-0.58, -0.32]
Level 2 (Participant)										
NFP-Only vs. FOP1							-0.02	[-0.31, 0.28]		
FOP2 vs. FOP1							-0.10	[-0.38, 0.19]		
NFP-Only vs. FOP2							0.08	[-0.21, 0.38]		
	Random Effects									
Residual variance[a]	0.11	[0.10, 0.13]	0.11	[0.10, 0.13]	0.08	[0.07, 0.10]	0.08	[0.07, 0.10]	0.08	[0.07, 0.09]
Intercept variance	0.72	[0.57, 0.91]	0.74	[0.59, 0.92]						
Random Slope Stepwise vs. RF									0.01	[0.00, 0.06]
Random Slope AS vs. RF									0.07	[0.00, 0.26]
Random Slope AS vs. Stepwise									0.08	[0.00, 0.24]

[a]Residual variance is measured as the inverse of Beta distribution parameter phi.

Discussion

The results of Study One indicate that there are differences between the three modeling methods. The three methods varied in the extent to which they included predictors in the final models with the all-subsets method including more predictors, on average, than the stepwise and random forest methods. Mixed model results indicated that there were significant differences in the three models' performance. On average, the all-subsets method had the highest goodness-of-fit ($\overline{R^2_{ADJ}}$ = 0.85), followed by the random forest method ($\overline{R^2_{ADJ}}$ = 0.70). Stepwise regression ($\overline{R^2_{ADJ}}$ = 0.43) had the lowest average goodness-of-fit for the three methods. The mixed model analysis also found significant differences in the cross-validity of the three methods. Among the three methods, random forest had the highest average cross-validity ($\overline{R^2_{CV}}$ = 0.22), followed by stepwise regression ($\overline{R^2_{CV}}$ = 0.18). The all subsets models, on average, had the lowest cross-validity of the three methods ($\overline{R^2_{CV}}$ = 0.12).

A limitation of this study was the strategy employed for cross-validation. Having only a single group for training and testing results can result in the cross-validity having a high variance depending on how the data is split. Additionally, the use of this method resulted in a training dataset with only 40 observations from which to estimate a model. The resulting 40 observations to 16 predictors ratio potentially penalized the performance of the stepwise regression method more heavily to compared to the all subets and random forest methods which utilize bootstrapping.

The results of Study One indicate that the stepwise regression method had worse performance than at least one of the other two methods in both goodness-of-fit and generalizability. The random forest method performed better than the all-subsets method

in terms of generalizability but had worse goodness-of-fit. A possible explanation for the contrasting goodness-of-fit and cross-validation results is that the all-subsets method is overfitting the models. A further indication that this may be the case is the high number of predictors that were included, on average, by the all-subsets method (*Mdn* = 16) compared to the stepwise (*Mdn* = 4) and random forest (*Mdn* = 3) methods.

An additional conclusion that can be drawn from Study One is that judgment in the task environment is better described by a tree model than a linear one. Despite the improved goodness-of-fit of the all-subsets method, the low generalizability of that method, combined with the overall low performance of the stepwise method, suggests that a non-linear tree modeling approach may be a more optimal method for predicting judgment in quasirational task environments such as this one. Although the goodness-of-fit of the random forest method was lower than that of the all-subsets method, it was still relatively high while also having high cross-validity, indicating that the random forest approach provided a better balance between goodness-of-fit and generalizability.

CHAPTER 4: STUDY TWO

Study One examined the difference in the goodness-of-fit and generalizablity of the stepwise regression, all subsets method, and random forest method. Findings indicated that the all-subsets method has the best goodness-of-fit among the three methods, but the lowest generalizability, with the random forest method having the highest generalizability and relatively high goodness-of fit. However, although cross-validation indicates the performance of a given model in explaining new cases, it does not provide definitive confirmation that the model is selecting the actual information that was used in generating the original judgments.

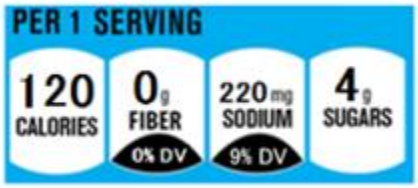
To compare the capacity of the three modeling methods to correctly extract the correct model used to generate judgments, a simulation study was designed in which a known model was used to generate ratings for the cereals in Carter and González-Vallejo (2018). In this study, the goal is to examine the model's performance in accurately extracting models that reflect a simulated judgment policy in a judgment environment that contains high collinearity. As was previously discussed, many real-world judgment environments include cues that contain a high level of redundancy. Thus, the results of this study will provide important indications of the comparative ability of the three methods to determine the environmental cues that are relevant to judgment despite the presence of highly correlated cues.

Methods

In the second study, nutritional judgment policies were simulated based on one of the hypotheses of Carter and González-Vallejo (2018) regarding the use of highlighted information. In that study, the researchers hypothesized that a label known as a nutrient-

specific front-of-package (FOP) label would make consumers more likely to use nutrients highlighted in the label to form nutritional judgments. The researchers found empirically that this was not the case (Carter & González-Vallejo, 2018). In the current study, a judgment policy following the initial hypothesis was simulated and judgment ratings were generated from this known policy. Thus, the study could also provide an indication of the sensitivity of the policy capturing methods to the diagnosis of the hypothesis.

Two judgment policies, consistent with the original hypothesis, but varied in coefficient strength, were simulated. Both were based upon a nutritionally relevant FOP label (FOP1 in the original study) that contained Calories, Sodium, Sugar, and Dietary Fiber. Figure 5 shows the relevant front-of-package label used in the original study. The nutrients for this labeling condition were determined according to findings from previous research, American dietary guidelines, and according to the statistical relationship between 74 cereals in the study and the nutrients they contained. See Carter and González-Vallejo (2018) for a full description of the development of these labels.



FOP1 Label

Figure 5. Front-of-package label. Used as the basis for simulated judgment policies.

Data

Simulation allows for data to reflect an idealized participant who could yield thousands of observations rather than a limited number. However, this study was

concerned with the implementation of variable selection methods to circumstances that are less than ideal. Specifically, this study was concerned with judgment tasks where sufficient observations of a judgment are impractical or impossible because they rely on a single participant's capacity for making multiple judgments or reflect an environment in which the number of possible tasks is limited. Therefore, to reflect the characteristics of the empirical design described above, 73 judgments were simulated (Carter & González-Vallejo, 2018).

Judgment ratings of 73 different, predetermined, breakfast cereals were generated based on the simulated judgment policies. Two judgment policies based nutritional judgments of the 73 cereals on the FOP1 label nutrients (Calories, Sodium, Sugar, and Fiber). These judgment policies were simulated with an emphasis on equal weighting rather than optimal weights, following recommendations regarding the utility of sign coefficient weights (Dawes, 1979; Dawes & Corrigan, 1974). The weight size was a variable condition across the two policies. For both policies, the nutrient variables calories, sodium, sugar, and fiber were used to generate nutritional judgment ratings. As was the case for the empirical data, the 73 nutritional ratings (Y) were bounded at 1 and 100. The intercept was set at 50, the median possible rating value for the gold standard nutritional ratings. Coefficients for the model were set as follows: positive coefficients for calories and fiber, and negative coefficients for sugar and sodium. Each policy varied coefficient size. For the first coefficient condition, absolute coefficient sizes were .01 and for the second they were .25.

Errors were normally distributed with mean of zero and standard deviation that was derived using empirical estimation. An initial standard deviation of the errors of 15

was estimated based upon an overall average of the adjusted average squared residuals of individual linear models extracted in Carter and González-Vallejo (2018). That is, standard deviation of the errors was the average square root of the variances determined by the following equation.

Equation 7:

$$\sigma^2 = \frac{1}{n-2}\sum_{i=1}^{n} e_i^2$$

Where sigma squared is the ratio of the sum of the squared distance between the predicted and actual value (e^2) for each judgment (1-n) over the total number of judgments minus two. However, for the simulation study, this standard deviation of 15 was used only as an initial baseline. For the standard deviation of the errors a value less than half of the initial 15 estimate (i.e., 7) was used in order to ensure that overall error in the model would be representative of an environment in which judgment error was relatively low.

The resulting policies used to determine the 73 ratings were as follows:

Equation 8:

$$\mathrm{Y}_1 = (.01 \times \text{Calories}) - (.01 \times \text{Sodium}) - (.01 \times \text{Sugars}) + (.01 \times \text{Dietary Fiber}) + \varepsilon$$

$$\mathrm{Y}_2 = (.25 \times \text{Calories}) - (.25 \times \text{Sodium}) - (.25 \times \text{Sugars}) + (.25 \times \text{Dietary Fiber}) + \varepsilon$$

where the error term ε is normally distributed with a mean of zero and a standard deviation of seven. No other nutrients were included in generating the simulated ratings. As nutritional rankings from the original study were bounded at 1 and 100, similar bounds were instituted for the simulated data by rounding any values smaller than one to one and rounding any values larger than 100 to 100.

Following the simulation of 73 ratings for each of the two judgment policies, these ratings were bootstrapped in order to generate 150 resamples of each judgment policy. That is, for each of the two judgment policies, $|b|$ = .01 and $|b|$ = .25, a total of 150 samples of 73 ratings each were drawn using resampling with replacement from the original 73 ratings generated using simulation. These 300 datasets of 73 observations generated from known judgment policies were used to compare to the three modeling methods.

Analysis and Results

To compare the three variable selection methods, each of the three analyses, stepwise regression, the all-subsets method, and random forest were used to extract the judgment policy from the simulated data. Unlike the previous study, in this case all 73 judgments for both policies were modeled using the nutrients from the NFP, with no cross-generalization. As the data were simulated based on known policies, this study was primarily interested in the ability of the different techniques to capture the predictors included in the correct model. That is, although only four predictors (calories, sodium, sugars, and dietary fiber) were included in generating the simulated ratings, all 16 nutrient predictors (calories, sodium, sugars, dietary fiber, as well as calories from fat, total fat, saturated fat, polyunsaturated fat, monounsaturated fat, potassium, insoluble fiber, soluble fiber, total carbohydrates, other carbohydrates, protein, and number of vitamins and minerals) were used in extracting models for the ratings. This approach was used to compare differences in the performance of the models in including the relevant nutrients and excluding nutrients that were not present in the generating policy. Analyses

for Study Two were conducted using statistical packages in R. All code utilized for the analyses is reported in Appendix B.

Stepwise Regression Variable Selection

In the first analysis, stepwise linear regression was applied to extract reduced models for each of the policies. For each of the two policies' 150 data sets, a linear model relating the 73 nutritional judgment ratings to the nutrient information from the NFP was developed using stepwise regression to generate models that contained a reduced number of predictors from the original set of 16.

Stepwise regression successfully extracted linear models for 99.67% ($N = 299$) of the 300 bootstrapped judgment sets. The one judgment set for which stepwise regression could not extract models was from the simulated policy with absolute coefficient values of .01. For the 299 judgment sets for which models were successfully extracted, 295 of the models were statistically significant at the $p < .05$ level. Thus, stepwise regression extracted statistically significant linear models for 98.33% of the 300 bootstrapped judgment sets. All four judgment sets for which models were not significant were bootstrapped from the low ($|b| = .01$) weight condition. The number of predictors included in final stepwise models ranged from 1 to 13 with a mean of 4.83 ($Mdn = 4$).

Among the 299 judgment sets for which stepwise models were generated, 168 (56.19%) models included calories in the final predictor set; 209 (69.90%) included sodium; 74 (24.75%) included sugar; and 82 (27.42%) included dietary fiber. Twenty-five models included all four nutrient variables, along with other variables; none of the stepwise models included the key four nutrients alone. Of the twenty-five models that

included all four nutrient variables, a single model was from the low ($|b|$ = .01) weight condition and the remaining twenty-four were from the high ($|b|$ = .25) weight condition.

All Subsets Variable Selection

Prior to executing the all-subsets method, 100,000 bootstrapped datasets were generated from each of the 150 judgments sets of each simulated policy. In keeping with the idiographic approach of judgment analysis, a set of 100,000 bootstrapped datasets was drawn for each set of 73 ratings and all possible models was subsequently fit to each of the bootstrapped samples. Once all possible models were fit, the goodness-of-fit criterion, adjusted R^2 was used to determine the best-fit model for each of the 100,000 datasets. Subsequently, the probability distribution of the best-fit model was used to determine the final best model for each participant. That is, the final judgment model for each participant was chosen according to which model had the highest probability of being chosen as the best-fit model. Lastly, the predictor variables' criticality was determined according to the probability that it was included in a best-fitting model.

The all-subsets method successfully extracted a linear model for all data sets (N = 300). The number of predictors included in best-fitting all-subsets' models ranged from 3 to 16. The mean number of predictors included in final all-subsets' models was 10.81 (*Mdn* = 11). Among the 300 judgment sets for which models were generated, 263 (87.67%) models included calories in the best-fitting model, 251 (83.67%) included sodium, 182 (60.67%) included sugar, and 210 (70.00%) included dietary fiber. Of the 300 models, 120 models included all four nutrient variables, along with other variables; none of the best-fitting models included the key four nutrient variables alone. Of the 120

models that included all four nutrient variables, 50 were from the low ($|b|$ = .01) weight condition and the remaining 70 were from the high ($|b|$ = .25) weight condition.

Random Forest Variable Selection

As in the previous study, the random forest variable selection method used the VSURF approach (Genuer et al., 2015). Following the VSURF default, for each forest in the variable selection process of the analysis, each simulated judge was resampled with replacement to generated 2000 bootstrapped datasets of 73 scores each. Default numbers were also used for the predictors randomly selected for consideration at each split (p = 5), forests for determining the minimum threshold (*threshold forests* = 50) and forests for the interpretative step (*interpret forests* = 25).

The random forest method successfully extracted tree models for all of the bootstrapped judgment sets (N = 300). The number of predictors included in final random forest models ranged from 2 to 15 with a mean of 7.49 (*Mdn* = 7). Among the 300 judgment sets for which forest models were generated, 119 (39.67%) models included calories in the final predictor set, 289 (96.33%) included sodium, 141 (47.00%) included sugar, and 249 (83%) included dietary fiber. Fifty-eight models included all four nutrient variables, along with other variables; none of the random forest final models included the four nutrient variables alone. Of the 58 models that included all four nutrient variables, 16 models were from the low ($|b|$ = .01) weight condition and the remaining 42 were from the high ($|b|$ = .25) weight condition.

Method Comparisons

Of initial interest is which of the three methods extracted models that most closely reflect the actual judgment policies that generated the simulated judgment ratings. An

examination of the proportion of predictor inclusion indicates some differences between the methods. In both weight conditions, the all-subsets method included the key variables in over 50% of models. For the random forest method, the same was true for sodium and dietary fiber, but not for calories and sugars. In contrast, the stepwise regression method included two predictors, calories and sodium, in more than 50% of models in the high weight condition ($|b|$ = .25) but not in the low weight condition. The stepwise regression method included calories in 56% of the 299 models: calories was included in 19% of models in the low weight condition ($|b|$ = .01), and 99% of models in the high weight condition ($|b|$ = 25). Sodium was included in 69.90% of the stepwise regression models; in the low weight condition ($|b|$ = .01), sodium was included in 40% of the stepwise models and in the high weight condition ($|b|$ = 25) it was included in 100% of the stepwise models. Table 7 shows the percentage of final models in each method and weight condition that included the nutrient predictors.

Table 7

Study Two Descriptive Results: Proportion of Final Models Including Nutrient Predictors

	Stepwise Regression ($N = 299$)		All Subsets ($N = 300$)		Random Forest ($N = 300$)	
	$\|b\| = .01$ ($N = 149$)	$\|b\| = .25$ ($N = 150$)	$\|b\| = .01$ ($N = 150$)	$\|b\| = .25$ ($N = 150$)	$\|b\| = .01$ ($N = 150$)	$\|b\| = .25$ ($N = 150$)
Calories*	**19 (13%)**	**149 (99%)**	**121 (81%)**	**142 (95%)**	**24 (16%)**	**95 (63%)**
Calories from Fat	17 (11%)	27 (18%)	104 (69%)	105 (70%)	50 (33%)	51 (34%)
Total Fat (g)	17 (11%)	24 (16%)	79 (53%)	88 (59%)	37 (25%)	80 (53%)
Saturated Fat (g)	73 (49%)	62 (41%)	99 (66%)	97 (65%)	7 (5%)	0 (0%)
Monounsaturated Fat (g)	30 (20%)	28 (19%)	100 (67%)	90 (60%)	9 (6%)	33 (22%)
Polyunsaturated Fat (g)	32 (21%)	35 (23%)	77 (51%)	82 (55%)	30 (20%)	28 (19%)
Dietary Fiber* (g)	**26 (17%)**	**56 (37%)**	**106 (71%)**	**104 (69%)**	**127 (85%)**	**122 (81%)**
Soluble Fiber (g)	43 (29%)	46 (31%)	107 (71%)	106 (71%)	5 (3%)	37 (25%)
Insoluble Fiber (g)	50 (34%)	46 (31%)	105 (70%)	97 (65%)	67 (45%)	115 (77%)
Potassium	12 (8%)	16 (11%)	84 (56%)	73 (49%)	117 (78%)	133 (89%)
Protein	35 (23%)	40 (27%)	117 (78%)	116 (77%)	53 (35%)	112 (75%)
Sodium* (mg)	**59 (40%)**	**150 (100%)**	**101 (67%)**	**150 (100%)**	**139 (93%)**	**150 (100%)**
Sugars* (g)	**20 (13%)**	**54 (36%)**	**84 (56%)**	**98 (65%)**	**95 (63%)**	**46 (31%)**
Total Carbohydrates (g)	18 (12%)	29 (19%)	117 (78%)	101 (67%)	58 (39%)	92 (61%)
Other Carbohydrates (g)	61 (41%)	59 (39%)	113 (75%)	105 (70%)	97 (65%)	17 (11%)
Vitamins & Minerals (#)	56 (38%)	56 (37%)	83 (55%)	93 (62%)	118 (79%)	104 (69%)

Table 8

Methods' Mean Relative Importance Values and Standard Errors

	Stepwise $\bar{\beta}_i$ (N = 299)		All Subsets $\overline{PC_i}$ (N = 300)		Random Forest $\overline{\Delta MSE_{OOB_i}}$ (N = 300)	
Simulated *b*-value	$\|b\| = .01$ (N = 149)	$\|b\| = .25$ (N = 150)	$\|b\| = .01$ (N = 150)	$\|b\| = .25$ (N = 150)	$\|b\| = .01$ (N = 150)	$\|b\| = .25$ (N = 150)
Calories*	**0.92 (0.25)**	**0.59 (0.03)**	**0.75 (0.01)**	**0.92 (0.01)**	**4.3 (0.16)**	**46.8 (1.67)**
Calories from Fat	0.03 (0.17)	0.06 (0.07)	0.67 (0.01)	0.66 (0.01)	5.17 (0.22)	25.3 (0.86)
Total Fat (g)	0.14 (0.18)	-0.11 (0.06)	0.60 (0.01)	0.60 (0.01)	4.72 (0.21)	29.6 (0.92)
Saturated Fat (g)	0.24 (0.01)	0.07 (0.01)	0.62 (0.01)	0.63 (0.01)	1.43 (0.12)	2.78 (0.20)
Monounsaturated Fat (g)	-0.23 (0.07)	-0.11 (0.03)	0.64 (0.01)	0.62 (0.01)	2.42 (0.12)	15.9 (0.59)
Polyunsaturated Fat (g)	0.23 (0.05)	0.07 (0.02)	0.59 (0.01)	0.58 (0.01)	3.23 (0.16)	17.4 (0.67)
Dietary Fiber* (g)	**0.08 (0.09)**	**0.13 (0.02)**	**0.65 (0.01)**	**0.68 (0.01)**	**9.23 (0.38)**	**78.8 (2.85)**
Soluble Fiber (g)	0.26 (0.03)	0.06 (0.01)	0.66 (0.01)	0.65 (0.01)	1.97 (0.11)	22.8 (1.42)
Insoluble Fiber (g)	0.38 (0.03)	0.13 (0.01)	0.72 (0.01)	0.69 (0.01)	7.04 (0.46)	85.6 (3.69)
Potassium	-0.08 (0.12)	-0.08 (0.03)	0.57 (0.01)	0.56 (0.01)	7.85 (0.25)	89.2 (3.61)
Protein	-0.27 (0.06)	-0.14 (0.02)	0.72 (0.01)	0.74 (0.01)	5.09 (0.26)	61.6 (2.87)
Sodium* (mg)	**-0.24 (0.02)**	**-0.78 (0.01)**	**0.62 (0.01)**	**1.00 (0.00)**	**11.8 (0.37)**	**284 (6.07)**
Sugars* (g)	**0.01 (0.06)**	**-0.09 (0.01)**	**0.57 (0.01)**	**0.61 (0.01)**	**7.04 (0.27)**	**22.3 (0.72)**
Total Carbohydrates (g)	-0.37 (0.27)	-0.49 (0.08)	0.70 (0.01)	0.68 (0.01)	5.61 (0.19)	43.0 (1.39)
Other Carbohydrates (g)	-0.34 (0.02)	-0.09 (0.01)	0.72 (0.01)	0.70 (0.02)	8.86 (0.45)	15.9 (0.66)
Vitamins & Minerals (#)	-0.20 (0.02)	-0.10 (0.01)	0.60 (0.01)	0.61 (0.01)	9.47 (0.31)	42.5 (1.35)

Table 8 shows each method's mean importance measure and associated standard error for each predictor in each weight condition. The accuracy of the ranking order of the stepwise and all-subsets methods were most impacted by weight condition. In the low weight condition, only one of the relevant nutrients (calories) was within the five predictors with the largest average beta-weights generated by the stepwise method. In the high weight condition, three of the relevant nutrients, calories, sodium, and dietary fiber, were in that group. For the all-subsets method, the low weight condition criticalities included only calories among the top five most critical predictors. In the high weight condition, sodium, calories, and dietary fiber placed among the top five most critical predictors on average. For both methods, in the high weight condition, dietary fiber held the fifth highest ranking.

The random forest method's ranking of the relevant nutrients seemed less impacted by weight condition. Although the value of the average rankings for all predictors were much larger in the high weight condition than in the low weight condition, in both weight conditions two of the relevant nutrients, sodium and dietary fiber, were ranked amongst the top five most important variables. This indicates that in the low-weight condition random forest's ranking metric was more accurate than the stepwise and all-subsets metrics, but in the high weight condition it was less accurate.

To investigate differences in inclusion of the four key variables across method and coefficient weight, a multivariate logistic mixed model was conducted using a Bayesian approach. All models were implemented using the brms package (version 2.50) for R, which builds on Stan to perform Markov Chain Monte Carlo Simulations using an

adaptive Hamiltonian Monte Carlo (HMC) sampler known as the no u-turn sampler (NUTS) and weakly informative priors were applied for all models (Bürkner, 2018)[7]. For each of the initial models, 4 chains of 30,000 iterations were simulated, of which the first 2,000 were discarded as burn-in. No thinning was applied. The convergence of the models was assessed by calculating the multivariate potential scale reduction factor and by assessing trace and density plots (Brooks & Gelman, 1998; Kruschke, 2015). Posterior distributions were summarized using the mean and a 95% highest posterior density interval (HPDI).

Multivariate multilevel models were implemented independently for fixed parameters and increasingly complex models were fit and tested. The current analysis did not include random effects models as previous research has indicated that both frequentist and Bayesian methods have difficulty accurately estimating random effects for logistic models at samples sizes similar to those in this study (Li, Lingsma, Steyerberg & Lesaffre, 2011). At level-two were 300 participants, at level-one were 899 models. Level-one excluded a single missing stepwise regression model ($N = 1$). For simplicity, results for the four criterion variables are summarized across two tables (see Tables 9 and 10). Multilevel models were compared using the WAIC with lower values of WAIC implying higher predictive accuracy.

---

[7] By default, brms uses weakly informative priors in order to minimize their influence on the results (Bürkner, 2018). Based on the recommendations by Gelman, Jakulin, Pittau, & Su (2008) parameters in the logistic mixed-model analysis are by default given a half student-t prior with 3 degrees of freedom and a scale parameter of 10 (Bürkner, 2018). In addition, results of the Bayesian analysis in this study were compared to those of a frequentist approach of four individual logistic regression analyses conducted using the generalized linear mixed-effect modeling function glmer available through the lme4 package (Bates, Martin, Bolker & Walker, 2015). Estimates from the two results were within rounding error of each other and the conclusions from both methods were exactly the same.

Intercept models for each of the criterions of interest were initially modeled separately in order to determine the variance components for each predicted variable. The extent to which variance in the inclusion rates could be attributed to the individual resampled dataset varied for the four nutrients. Only a very small proportion of the variance in sugar (ICC = 3%) and dietary fiber (ICC = 1%) could be attributed to simulated individual but a substantial proportion of the variance in calories (ICC = 22%) and sodium (ICC = 41%) could be attributed to the dataset, therefore a multivariate mixed model was decided upon as the best method of analysis.

Model 1 was a multivariate model including all four predicted variables in the model. Intercept components for this model are included in the tables for completeness. Model 2 added the modeling method used to determine whether modeling method impacted inclusion of the four variables of interest in the final models. Modeling method significantly improved the overall prediction of predictor inclusion (Model 1 WAIC: 4261.5, *SE* = 49.4, Model 2 WAIC 3500.2, *SE* = 61.5). Because the outcome of interest (inclusion of the relevant nutrients) is common in the overall population (i.e., > 10%), the adjusted odds ratios derived from logistic regression do not provide a good approximation of the difference in the methods' likelihood of including the relevant nutrients (Zhang & Yu, 1998). For this reason, the relative risk and HPDI values reported below are adjusted risk ratios (RR_C) derived using a simple correction formula recommended by Zhang and Yu (1998).[8]

---

[8] This formula developed by Zhang and Yu (1998) is: $RR_C = \frac{OR}{(1-P_0)+(P_0 \times OR)}$ where the OR is the adjusted odds ratio obtained from logistic regression and P_0 is the incidence or the outcome of interest in the nonexposed or comparison group.

From Model 2, conclusions regarding the impact of method on predictor inclusion are not consistent for all four nutrient predictors. The all-subsets method's final models were more likely to include calories than those of the random forest method (RR_C = 2.44, 95% HPDI = 2.37, 2.48) or the stepwise method (RR_C = 1.69, 95% HPDI = 1.63, 1.73). The stepwise method also had higher odds of including calories than the random forest method (RR_C = 1.70, 95% HPDI = 1.43, 1.93). The all-subsets method's final models were also more likely to include sugar than the random forest's final models (RR_C = 1.31, 95% HPDI = 1.13, 1.47) and the stepwise method's final models (RR_C = 2.58, 95% HPDI = 2.20, 2.92). The random forest method was more likely to generate models that included sugar than was the stepwise regression method (RR_C = 1.99, 95% HPDI = 1.62, 2.36).

For sodium and dietary fiber, random forest performance was better than that of the other methods. The random forest method's final models were 1.42 times as likely to include sodium than the stepwise method's final models (RR_C = 1.42, 95% HPDI =1.40, 1.43). The all-subsets method was also more likely to include sodium than the stepwise method (RR_C = 1.31, 95% HPDI = 1.22, 1.36) but less likely to include sodium than the random forest method (RR_C = 0.69, 95% HPDI = 0.45, 0.86). Similarly, the all-subsets method was more likely than the stepwise method (RR_C = 2.79, 95% HPDI = 2.48, 3.05) to include dietary fiber in its final models, but less likely to include dietary fiber than the random forest method (RR_C = 0.81, 95% HPDI = 0.69, 0.92). The random forest method was also more likely to include dietary fiber in its final models than the stepwise method (RR_C = 3.23, 95% HPDI = 3.00, 3.38).

Model 3 added weight condition to test whether differences in the coefficient weight of each simulated model would affect predictor inclusion. A WAIC model comparison of model two (WAIC 3500.2, *SE* = 61.5) and model three (WAIC = 3306.3, *SE* = 62.7) indicated that including weight condition improved model fit. Further examination of the odds ratios in each model, as shown in Tables 9-10 indicated that weight condition impacted some nutrient predictors but not others. As can be seen in Table 10, 95% HPDIs for the weight condition's risk ratios in predicting inclusion of sugar (RR_C = 0.99, 95% HPDI = 0.81, 1.18) and dietary fiber (RR_C = 1.14, 95% HPDI = 0.97, 1.29) include one, indicating that for these predictors there is no difference between the two weight conditions when controlling for modeling method. However, for calories and sodium, the coefficient condition did have an impact on the probability of inclusion. Final models from the high weight condition ($|b|$ = .25) were more likely to include calories (RR_C = 2.62, 95% HPDI = 2.53, 2.67) than final models in the low weight condition. As is shown in Table 7, in the high weight condition all final models, regardless of modeling method, included sodium. Mixed model results indicated that high weight condition models were 1.5 times as likely to include sodium (RR_C = 1.50, 95% HPDI = 1.50, 1.50) than final models in the low weight condition ($|b|$ = .01).

Model 4 added the interaction between method and weight condition to determine whether the effect of weight condition varied according to the method used. A comparison of model three (WAIC = 3306.3, *SE* = 62.7) and model four (WAIC = 3150.30, *SE* = 61.80) indicated that including the interaction between method and weight condition improved the model's predictiveness of predictor inclusion so interactions were maintained in the final model. Results from Model 4 (included in the Tables below)

indicated that there was an interactive effect of method and weight condition on predictor inclusion for some predictors but not for others. For calorie inclusion, the odds estimate for the interaction between weight condition and difference between the all susbests and random forest methods the Bayesian 95% HPDI included one. This indicates that that there was no interactive effect of weight condition on the comparative likelihood that the all-subsets and random forest method would include calories. For sodium inclusion, the 95% HPDI for all estimates of the interaction between weight condition and the difference between methods also included one, indicating there was no differential impact of weight condition on the odds that all subsets, stepwise, or random forest models would include sodium.

There was an interaction between coefficient strength and comparative impact of the all subsets and stepwise inclusion of calories. In the low weight condition, the all-subsets method was more likely to include calories (RR_C = 1.76, 95% HPDI = 1.73, 1.77), but among models in the high weight condition, there was no difference between the two methods inclusion of calories (RR_C = 0.13, 95% HPDI = 0.00, 1.19). There was an impact of weight condition on the difference in likelihood of calorie inclusion for the stepwise method compared to the random forest method. Specifically, the likelihood that a final model would include calories was no different for the random forest method than the stepwise method when the calorie coefficient was low (when $|b|$ = .01: RR_C = 0.81, 95% HPDI = 0.48, 1.24), but the likelihood of inclusion was higher for the stepwise method than the random forest method when the calorie coefficient was high ($|b|$ = .25: RR_C = 2.51, 95% HPDI = 2.24, 2.52).

A pattern similar to that of calories was found for the methods' inclusion of sugar. There was no difference between the all-subsets method and the random forest method's inclusion of sugar in the low weight condition (RR_C = 0.81, 95% HPDI = 0.58, 1.07), but in the high weight condition the all-subsets method was more likely to include sugar than the random forest method (RR_C = 1.77, 95% HPDI = 1.26, 2.01). The stepwise method was less likely to include sugar in final models than the random forest method in the low weight condition (RR_C = 0.11, 95% HPDI = 0.06, 0.20) but in the high weight condition the difference between the two methods was reversed, although the HPDI indicated there was no difference between the two methods (RR_C = 1.15, 95% HPDI = 0.45, 1.79). Similarly, the difference between the likelihood of the all subsets and stepwise methods inclusion of sugar decreased as a function of weight condition. In both weight conditions the all-subsets method had a greater likelihood of sugar inclusion, but that increased likelihood was larger in the low weight condition (RR_C = 3.19, 95% HPDI = 2.70, 3.55) than in the high weight condition (RR_C = 2.35, 95% HPDI = 1.01, 3.46).

Weight condition also impacted the comparative performance of the stepwise method and the two other methods in their inclusion of dietary fiber. The all-subsets method was more likely to include dietary fiber in a final model than the stepwise method in both weight conditions but the difference in likelihood was greater in the low weight condition (RR_C = 3.19, 95% HPDI = 2.87, 3.39) than in the high weight condition (RR_C = 2.39, 95% HPDI = 1.14, 3.26). Similarly, the random forest method was more likely than the stepwise method to include dietary fiber in both weight conditions but the difference in likelihood was greater in the low weight condition (RR_C = 3.46, 95% HPDI = 3.27, 3.56) than in the high weight condition (RR_C = 2.95, 95% HPDI = 1.64, 3.48). Weight

condition had no interactive effect on the comparative performance of the all-subsets method and random forest methods in including dietary fiber (i.e., the random forest models were more likely than the all subsets models to include dietary fiber in both weight conditions).

Table 9

Summary of Calories and Sodium Multilevel Analysis: Fixed Effect Estimates and Highest Posterior Density Intervals (HPDI) for Predicting Calorie & Sodium Inclusion

Parameter	Fixed Effects							
	Model 1		Model 2		Model 3		Model 4	
Calories Inclusion	Posterior Prob Mean	Probability 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI
Intercept (Probability)	0.65	[0.60, 0.69]						
Level 1 (Model)								
Stepwise vs. Random Forest			1.70	[1.43, 1.93]	1.77	[1.50, 2.00]	0.81	[0.48, 1.24]
All Subsets vs. Random Forest			2.44	[2.37, 2.48]	2.45	[2.39, 2.48]	2.45	[2.36, 2.49]
All Subsets vs. Stepwise			1.69	[1.63, 1.73]	1.69	[1.63, 1.73]	1.76	[1.73, 1.77]
Level 2 (Data Set)								
High Weight vs. Low Weight					2.62	[2.53, 2.67]	2.48	[2.25, 2.62]
Level 1 & 2 Interaction (Weight = .25)[a]								
Stepwise vs. Random Forest							2.51	[2.24, 2.52]
All Subsets vs. Random Forest							2.31[b]	[1.60, 2.47]
All Subsets vs. Stepwise							0.13	[0.00, 1.29]
Sodium Inclusion	Posterior Prob Mean	Probability 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI
Intercept (Probability)	0.92	[0.88, 0.95]						
Level 1 (Model)								
Stepwise vs. Random Forest			0.30	[0.13, 0.55]	0.39	[0.19, 0.63]	0.40	[0.19, 0.63]
All Subsets vs. Random Forest			0.69	[0.45, 0.86]	0.76	[0.57, 0.90]	0.77	[0.57, 0.90]
All Subsets vs. Stepwise			1.31	[1.22, 1.36]	1.31	[1.22, 1.36]	1.31	[1.22, 1.36]
Level 2 (Data Set)								
High Weight vs. Low Weight					1.50	[1.50, 1.50]	1.50	[1.49, 1.49]
Level 1 & 2 Interaction (Weight = .25)[a]								
Stepwise vs. Random Forest							1.04[b]	[0.00, 1.04]
All Subsets vs. Random Forest							1.04[b]	[0.00, 1.04]
All Subsets vs. Stepwise							1.43[b]	[0.00, 1.43]

[a] In a logistic model including an interaction term, the adjusted risk ratios for the reference group (i.e., weight = .01) are the Level 1 (Model) values.
[b]Estimate of interactive effect of method and weight condition HPDI includes 0 (i.e., interactive effect not statistically significant).

Table 10

Summary of Sugar and Dietary Fiber Multilevel Analysis: Fixed Effect Estimates and Highest Posterior Density Intervals (HPDI) for Predicting Sugar & Dietary Fiber Inclusion

Parameter	Fixed Effects							
	Model 1		Model 2		Model 3		Model 4	
Sugar Inclusion	Posterior Prob Mean	Probability 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI
Intercept (Probability)	0.44	[0.40, 0.47]						
Level 1 (Model)								
Stepwise vs. Random Forest			0.49	[0.37, 0.65]	0.48	[0.36, 0.64]	0.11	[0.06, 0.20]
All Subsets vs. Random Forest			1.31	[1.13, 1.47]	1.32	[1.15, 1.49]	0.81	[0.58, 1.07]
All Subsets vs. Stepwise			2.58	[2.20, 2.92]	2.63	[2.25, 2.96]	3.19	[2.70, 3.55]
Level 2 (Data Set)								
High Weight vs. Low Weight					0.99	[0.81, 1.18]	0.31	[0.18, 0.50]
Level 1 & 2 Interaction (Weight = .25)[a]								
Stepwise vs. Random Forest							1.15	[0.45, 1.79]
All Subsets vs. Random Forest							1.77	[1.26, 2.01]
All Subsets vs. Stepwise							2.35	[1.01, 3.46]
Dietary Fiber Inclusion	Posterior Prob Mean	Probability 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI	Posterior RR_C Mean	RR_C 95% HPDI
Intercept (Probability)	0.60	[0.57, 0.64]						
Level 1 (Model)								
Stepwise vs. Random Forest			0.23	[0.15, 0.34]	0.22	[0.15, 0.33]	0.11	[0.05, 0.21]
All Subsets vs. Random Forest			0.81	[0.69, 0.92]	0.81	[0.69, 0.91]	0.78	[0.60, 0.93]
All Subsets vs. Stepwise			2.79	[2.48, 3.05]	2.82	[2.50, 3.07]	3.19	[2.87, 3.39]
Level 2 (Data Set)								
High Weight vs. Low Weight					1.14	[0.97, 1.29]	0.89	[0.60, 1.18]
Level 1 & 2 Interaction (Weight = .25)[a]								
Stepwise vs. Random Forest							0.37	[0.09, 0.83]
All Subsets vs. Random Forest							0.83[b]	[0.41, 1.09]
All Subsets vs. Stepwise							2.39	[1.14, 3.26]

[a] In a logistic model including an interaction term, the adjusted risk ratios for the reference group (i.e., weight = .01) are the Level 1 (Model) values.
[b]Estimate of interactive effect of method and weight condition HPDI includes 0 (i.e., interactive effect not statistically significant).

Discussion

Results of Study Two confirm that the three modeling methods differ in their ability to accurately extract the correct model given a task environment with high redundancy. The first important result to note is that none of the three methods extracted a model that retained *only* the four nutrients included in generating the criterion ratings. This result is important as it indicates that all of the methods are susceptible to the impact of multicollinearity and that judgment analysts should be wary in their interpretations of variable importance measures when the task environment has high collinearity. A second result of Study Two is the comparative performance of the models in extracting models that at least included all four relevant nutrients. The stepwise method extracted the fewest such models, 25 out of a total of 299 extracted models. Out of 300 total models, the random forest method extracted 58 models that included the four nutrients used in the generating process. The all-subsets method extracted 120 such models.

Across three methods, the likelihood of extracting a model with all four relevant nutrients was impacted by the weight condition of the original generation process. In the low weight condition, the stepwise method only extracted a single model with all four relevant nutrients. Fewer than half (n = 16) of the random forest models with all four nutrients were extracted in the low weight condition. The impact of condition was less stark on the all-subsets method, but there was still a notable difference. That is, 50 of the all subsets that included the four relevant nutrients were from the low weight condition, 70 were from the high weight condition.

The proportion of final models that included the relevant predictors varied according to the modeling method, but the impact of method was not consistent across

the predictors. The all-subsets method performed best in overall inclusion of calories and sugar; whereas, the random forest method outperformed the other methods in overall inclusion of sodium and dietary fiber. Compared to the other methods, the performance of the random forest method in including calories was particularly low under the high weight condition. Under the low weight condition, the random forest's calorie inclusion rate (16%) was comparable to that of stepwise regression (13%) but under the high weight condition the random forest method included calories in far fewer final models (63%) than both the stepwise (99%) and all subsets (95%) methods. The random forest methods' inclusion of sugar follows a similar pattern. In the low weight condition, the random forest inclusion of sugar in final models (63%) is higher than that of the stepwise (13%), and all-subsets methods (56%). In contrast, in the high weight condition, the random forest's sugar inclusion rate (31%) dropped lower than the inclusion rate for the stepwise (36%) and all subsets (65%) methods.

The random forest method's performance in capturing the importance of sodium and dietary fiber was better across weight condition, but its performance in the low weight condition was what distinguished it from the other methods. Random forest's inclusion of sodium (93%) and dietary fiber (85%) in the low weight condition was much higher than that of the stepwise method rates of 40% and 17% respectively. The all-subsets method inclusion of sodium (67%) and dietary fiber (71%) in the low weight condition was lower than that of the random forest method, but not as low at that of the stepwise method. In the high weight condition, inclusion rates for dietary fiber slightly decreased for the random forest (81%) and the all-subsets method (69%) though increasing for the stepwise method (37%). Sodium inclusion in the high weight condition

was maximized for all three methods, with 100% of all methods' final models including sodium.

The impact of coefficient weight on nutrient inclusion was most consistently seen for the stepwise regression method. The stepwise method's inclusion rates were consistently better in the high weight condition, indicating that the strength of relationship between the predictors and criterion is a key factor for the performance of stepwise regression. In contrast, the impact of weight on the inclusion of relevant predictors in the all subsets' final models was not as great. For random forest, the relationship between weight and predictor inclusion was at times in the opposite of the expected direction.

The results of Study Two highlight two potentially important factors in the comparison of these three models. The first is the possible differential impact of multicollinearity on the three methods. In the task environment, calorie count was significantly related to two other nutrients used in the generation process, sugars ($r = .52$, $p < .05$) and dietary fiber ($r = .53$, $p < .05$). Sugars was also related to calories and sodium ($r = .27$, $p < .05$) in the generative process. In contrast, the two nutrients for which the random forest method had better performance are each related to only one other nutrient in the generative process; dietary fiber was not significantly related to sodium ($r = -.23$, $p > .05$) or sugar ($r = .15$, $p > .05$) and sodium was not related to calories ($r = .03$, $p > .05$). In generating the judgment ratings, calories and dietary fiber were positively related to judgment rating and sodium and sugars were negatively related to judgment rating. These findings suggest the possibility that the random forest method

might perform best under conditions when multicollinearity is low but falter when multicollinearity is high.

An additional possibility is that the all-subsets method's higher performance in extracting models that include calories and sugars and indeed, high (greater than 50%) inclusion of the four relevant nutrients in general is not due to resilience against multicollinearity. Rather, it is possible that (as was indicated in Study One) the all-subsets method has higher inclusion of the relevant predictors because it has higher inclusion of nutrients in general. Indeed, the results of the current analysis are consistent with this hypothesis. As can be seen in Table 2, all twelve of the nutrients that were not including in the generating process were included in 50% or more of the all subsets' final models. In contrast, the random forest method included only five of the twelve irrelevant nutrients in 50% or more of the final models. The stepwise method's inclusion rates for the irrelevant nutrients were all under 50%. Although the generalizability of the methods was not specifically examined in this study, the models containing many more predictors than the four relevant ones would likely have reduced performance in generalizing to new data generated by the same simulation process.

Lastly, it is worth recalling that the generative process of the judgment observations in this case was a linear model. With this is mind, the performance of the random forest method compared to the stepwise and all-subsets method becomes more comprehendible and, perhaps, compelling. That is, even though the generative process for the observed data was linear, the random forest method still performed better than the two linear models in including two out of the four variables.

CHAPTER 5: STUDY THREE

Study Two indicated that there are differences in the extent to which the three methods perform in including the correct variables in their final models. Additionally, the second study indicated that the extent of these differences in method were dependent on the strength of the relationship between the predictors and the criterion variable. Differences in the performance of the three methods suggested that the methods may be differentially impacted by inter-correlations between the relevant predictors and other predictors in the environment, but that this may also be due to high predictor inclusion by some methods compared to others. To provide a more systematic examination of the effect of multicollinearity on the importance measures provided by the stepwise regression, all subsets, and random forest modeling methods, a subsequent simulation study was conducted.

This study used the approach outlined by Azen et al. (2001), including the correlational structures used in that study. In addition to using these correlational structures to examine the all-subsets method, as was done in Azen et al., this study compares the all-subsets method's results to those of stepwise regression and the random forest method. The main goal of Study Three is to replicate the findings of Azen et al. regarding the all-subsets method and to add an examination of the two additional modeling methods in order to clarify the performance of these methods. Specifically, the following study should clarify the findings of Study Two and provide a more direct indication of which of the three methods should be preferred under conditions of multicollinearity.

Method

As outlined in Azen et al. (2001), several correlational patterns between predictors and the criterion were specified. Subsequently, a multivariate normal dataset was generated for each of the specified correlation patterns. Then, each method of interest, stepwise regression, all subsets, and random forest, were used to model each generated dataset. Consistent with the Azen et al. (2001) simulations, for the all-subsets methods, each randomly generated dataset was bootstrapped 500 times. Lastly, as in the Azen et al. (2001) simulation approach, the generation of multivariate normal datasets and modeling of each dataset was repeated 40 times to eliminate possible bias and provide a distribution of variable importance measures provided by each modeling method. The code for this process, as well as for the three analytic methods described below, are included in Appendix B.

Data

To examine the effect of multicollinearity on the variable importance measures of the three methods, two groups of simulations were conducted. Following Azen et al. (2001), the first set of simulated correlational structures (simulation A) kept the criterion-predictor relationship constant across structures and varied the predictor correlations—the multicollinearity. The A set of simulations included four correlational structures, each with four predictor variables. In all four sets, the correlations between predictors and the criterion variable were 0.1, 0.3, 0.5, and 0.7. Simulated multicollinearity across sets began at 0.75 (in A1) and was decreased in each set by .25 so in the final set (A1) all inter-predictor correlations were equal to 0. Table 11 shows each of the A group correlational structures as well as correlation measures of each structure's population (50

observations across 40 replications). Correlation measures include the squared correlation of a linear regression of the full model, standardized regression coefficients for each predictor, as well as the squared partial and semi-partial correlations for each predictor.

Again, in line with Azen et al. (2001)'s simulations, a second group of simulations were conducted to examine how systematic variation in the intercorrelation of predictors affects the importance ranking of the various measures. As in Azen et al. (2001), the second set of simulated correlational structures (simulation B) kept the criterion-predictor relationship constant across structures, but inter-predictor correlations were varied across predictors. The B set of simulations included five correlational structures, each with three predictor variables. In all five sets, only a single predictor (X1) was correlated with the criterion variable ($\rho = 0.6$), the other two predictors (X2 and X3) had no correlation with the criterion (ρ's = 0.0). The correlation between the criterion-related predictor and the two unrelated predictors was varied in each set. In the first set (B1), the two predictors (X2 and X3) were unrelated to X1 (ρ's = 0.0). In the second set (B2), X2 remained uncorrelated to X1 ($\rho = 0.0$), but X3 was positively correlated with X1 ($\rho = \sqrt{0.2}$). The third set (B3) correlated X1 with both X2 ($\rho = \sqrt{0.1}$) and X3 ($\rho = \sqrt{0.1}$). The fourth (B4) and fifth (B5) set mirrored the second and third sets respectively: B4 maintained a zero correlation between X1 and X2, but increased the correlation between X1 and X3 ($\rho = \sqrt{0.4}$); in the fifth set both X2 and X3 had a $\sqrt{0.2}$ correlation with X1. Table 12 shows each of the group B correlational structures as well as correlation measures of each structure's population (50 observations across 40 replications). Correlation measures include the squared correlation of a linear regression of the full

model, standardized regression coefficients for each predictor, as well as the squared partial and semi-partial correlations for each predictor.

Table 11

A1 – A4: Importance Measures in the Population

		Population Correlation Matrix						Importance Measures (Population)			
Simulation		Y	X_1	X_2	X_3	X_4	R^2 of full model	$\rho^2(Y, X_i)$	B_i	Partial ρ^2	Semi-partial ρ^2
A1	Y	1					0.9970				
	X1	0.1	1					0.01	-1.0909	0.9919	0.3714
	X2	0.3	0.75	1				0.09	-0.2766	0.8877	0.0241
	X3	0.5	0.75	0.75	1			0.23	0.5240	0.9661	0.0868
	X4	0.7	0.75	0.75	0.75	1		0.50	1.3411	0.9949	0.5937
A2	Y	1					0.6664				
	X1	0.1	1					0.01	-0.4653	0.2927	0.1380
	X2	0.3	0.5	1				0.07	-0.0602	0.0072	0.0024
	X3	0.5	0.5	0.5	1			0.23	0.3681	0.2083	0.0878
	X4	0.7	0.5	0.5	0.5	1		0.48	0.7779	0.5370	0.3869
A3	Y	1					0.6548				
	X1	0.1	1					0.00	-0.1760	0.0752	0.0281
	X2	0.3	0.25	1				0.10	0.0911	0.0205	0.0072
	X3	0.5	0.25	0.25	1			0.23	0.3738	0.2615	0.1222
	X4	0.7	0.25	0.25	0.25	1		0.50	0.6261	0.5015	0.3473
A4	Y	1					0.8376				
	X1	0.1	1					0.01	0.1113	0.0708	0.0124
	X2	0.3	0.0	1				0.09	0.2954	0.3491	0.0871
	X3	0.5	0.0	0.0	1			0.28	0.5044	0.6096	0.2537
	X4	0.7	0.0	0.0	0.0	1		0.50	0.6823	0.7411	0.4651

Table 12

B1 – B5: Importance Measures in the Population

Simulation		Population Correlation Matrix Y	X_1	X_2	X_3	R^2 of full model	Importance Measures (Population) $\rho^2(Y, X_i)$	B_i	Partial ρ^2	Semi-partial ρ^2
B1	Y	1				0.3589				
	X1	0.6	1				0.36	0.5960	0.3559	0.3543
	X2	0.0	0.0	1			0.00	-0.0001	0.0000	0.0000
	X3	0.0	0.0	0.0	1		0.00	-0.0377	0.0022	0.0014
B2	Y	1				0.4391				
	X1	0.6	1				0.35	0.7474	0.4390	0.4390
	X2	0.0	0.0	1			0.00	-0.0073	0.0000	0.0000
	X3	0.0	$\sqrt{0.2}$	0.0	1		0.00	-0.3433	0.1414	0.0924
B3	Y	1				0.4311				
	X1	0.6	1				0.34	0.7401	0.4310	0.4309
	X2	0.0	$\sqrt{0.1}$	1			0.00	-0.2443	0.0833	0.0517
	X3	0.0	$\sqrt{0.1}$	0.0	1		0.00	-0.2307	0.0769	0.0474
B4	Y	1				0.5902				
	X1	0.6	1				0.35	1.0038	0.5899	0.5895
	X2	0.0	0.0	1			0.00	-0.0110	0.0003	0.0001
	X3	0.0	$\sqrt{0.4}$	0.0	1		0.00	-0.6369	0.3667	0.2373
B5	Y	1				0.6135				
	X1	0.6	1				0.34	1.0284	0.6132	0.6129
	X2	0.0	$\sqrt{0.2}$	1			0.00	-0.4768	0.3061	0.1705
	X3	0.0	$\sqrt{0.2}$	0.0	1		0.00	-0.4798	0.3020	0.1673

Analysis and Results

In the first group of simulations, the question of interest was how different levels of multicollinearity impacted each variable selection method's capability to determine the importance of each predictor. Accordingly, stepwise regression, all subsets, and random forest were used to model the relationship between the predictors and criterion for each of the 40 replicate datasets. Modeling strategies for each method were like those taken in the previous studies.

In the first analysis, stepwise linear regression was applied to extract reduced models for each of the policies. For each of the 40 replications in group A's four sets, stepwise regression was used to develop a linear model relating the 50 observations to the predictor variables using the 4 predictors as the original predictor pool. The analytic strategy for the 40 replications in each of group B's five sets was the same, except that the original predictor pool consisted of only three predictors.

To conduct the all-subsets method, the strategy employed by Azen et al. (2001) in their simulations was followed. Namely, for simulation sets in both group A and B, 500 bootstrapped datasets were drawn for each of the 40 datasets of 50 observations, and all possible models were subsequently fit to each of the bootstrapped samples. Once all possible models were fit, adjusted R^2 values were used to determine the best fit model for each of the 500 datasets with the model with the highest probability of being chosen as the best-fit model selected as the final model. The predictor variables' criticality was determined according to the probability that it was included in a best-fitting model.

For the random forest method, the VSURF approach was again used for variable selection (Genuer et al., 2015). Following the VSURF default, for each forest in the

variable selection process of the analysis, each of the 40 simulated observation sets of 50 observations was resampled with replacement to generated 2000 bootstrapped datasets of 50 scores each. Default numbers were used to determine the number of predictors randomly selected for consideration at each split ($p = 1$), forests for determining the minimum threshold (*threshold forests* = 50) and forests for the interpretative step (*interpret forests* = 25). This method was applied to each of the 40 replications for each set of simulated correlational structures in both group A and group B.

As in Azen et al. (2001), mean variable importance measures and standard errors across 40 replications are reported (see Tables 13 and 15). Tables 14 and 16 show the proportion of replications in each correlation structure that included each predictor variable. In Azen et al. (2001), the authors note how the A group of simulations highlight how criticality responds to high multicollinearity by including all predictors. Azen et al. (2001) explain that removing any of the predictors would result in a misspecified final model when inter-predictor correlations are high.

Azen et al. (2001) compared three different methods for choosing best-fitting-models. One was adjusted r-squared as was used in this study, but Azen et al. also used Akaike's (1973) information criterion (AIC) and Mallow's C_p (Mallows, 1973). Results from Azen et al.'s A group of simulations indicate some differences between the three methods. When multicollinearity is at its highest ($\rho = 0.75$ in A1), all methods give all predictors a maximum predictor criticality equal to one. In the other collinearity conditions, there are small differences (in the second and third decimal place) between the predictor criticalities given by using the adjusted r-squared and Mallow's C_p but the AIC gives predictor criticalities that are smaller than either (i.e., differences at times in

the first decimal place's value). This is particularly true for predictors that are given lower values.

There are differences between final models found to be the best-fitting in Azen et al. (2001) and in the current study. In Azen et al. the overall best-fitting model for all groups but A2 was one that included all four predictors. In A2 the overall best-fitting model was one that included predictors X1, X3, and X4. In the current study, the overall best-fitting model for A1 and A2 included all four predictors. The overall best-fitting model for the A3 group included predictors X1, X3, and X4 and the best-fititng model for the A4 group included predictors X2, X3, and X4.

Differences in the overall best-fitting model are related to differences in the adjusted r-squared predictor criticalities in Azen et al. (2001) and those of the current study. Namely, in the current study predictor criticality values for X1-X3 are lower in the Azen et al. A2 simluation set ($C = 0.978$, $C = 0.450$, and $C = 0.967$ respectively) compared to the current study's A2 simulation set (X1: $C = 0.997$, X2: $C = 0.860$, and X3: $C = 0.996$). Azen et al.'s adjusted r-squared criticality values for X1-X3 in the A3 simulation set are $C = 0.729$, $C = 0.644$, and $C = 0.981$ respectively. In the current study criticality values for X1-X3 in A3 are $C = 0.927$, $C = 0.397$, and $C = 0.988$ respectively. In the A4 set, both this study and that of Azen et al. resulted in a criticality value of 1 for both X3 and X4. However, in the current study X1 had a value of $C = 0.400$ and X2 had a value of 1 and in Azen et al.'s study the adjusted r-squared predictor criticality for X1 was $C = 0.730$ and for X2 it was $C = 0.994$.

Desipte differences in values, rankings for the predictor criticalities in this study were consistent with those of Azen et al. (2001) and both results lead to the same

conclusions regarding the impact of collinearity. That is, despite some predictors being much more highly correlated with the criterion than others, when the intercorrelation between predictors is high (ρ = 0.75 as in A1), the all subsets predictor criticality measure indicates that all predictors are maximally critical (i.e., equal to one; see Table 13). Across all correlational structures, the all-subsets method, as in Azen et al. (2001), highly ranked the two predictors most highly correlated with Y (X1 and X2). However, the criticalities of the two predictors that were not as highly correlated (X3 and X4) had greater variation across correlational structures. Notably, the order of the criticalities of these two variables were at times reversed. For instance, in A2, X1 (ρ = 0.1) had a mean predictor criticality value of 0.927, but X2 (ρ = 0.3) had a mean predictor criticality of 0.397, essentially being given a lower importance ranking than X1 despite having a stronger relationship with the criterion. This order reversal was also seen, though less starkly, in A2, where even the X3 (ρ = 0.5) ranking was misaligned with the simulated relationships.

The stepwise and random forest results differed from the all subsets results in the highest inter-predictor correlation group. In the two simulation groups with highest multicollinearity (A1 and A2), stepwise regression's beta values for all but the most highly related predictor (X4) did not match the ranking they were simulated to have. The most highly related predictor (X4, ρ = 0.7) had the largest mean beta value ($\bar{\beta}$ = 1.51) but the next largest mean beta value was that of X1 ($\bar{\beta}$ = -1.17), which had the lowest correlation with Y (ρ = 0.1). Similarly, in all but the group with no multicollinearity (A4), stepwise regression rankings, particularly between X1 and X2 were reversed.

In the random forest method, variable importance rankings for X1 and X2 were reversed in all but the group with no multicollinearity (A4). However, the impact of multicollinearity on the random forest rankings across correlational structures was substantively different from its impact on stepwise regression and the all-subsets method. When multicollinearity is high (in the ρ = .75 and the ρ = 0.5 groups), stepwise regression and all subsets include all variables in the final model. As can be seen in Table 14, this is the case over all 40 replications. In contrast, in the highest multicollinearity group (ρ = .75) the random forest method excludes the variable that it ranks as lowest (X2). In the A2 (ρ = .5) and A3 (ρ = .25) groups, the random forest method includes only the two predictors that are most related to the criterion. For the stepwise regression and all-subsets methods, it is only in the A3 group that the pattern of inclusion mirrors that of the random forest in the A1 group; that is, X2 is excluded from the final model but X1 is maintained. In the A4 group, where multicollinearity was simulated at 0, all three of the methods yield final models in which the lowest relating predictor, X1, is the only excluded predictor.

Table 13

Simulations A1 – A4 Importance Measure Results: Importance Measures and their Standard Errors Over 40 Replications

			Mean Importance Measure (Standard Error) $n = 50$, $R = 40$		
Simulation	$\rho(X_i, X_j)$	Predictor $\rho(Y, X_i)$	Stepwise $\bar{\beta}_i$	All Subsets $\bar{C}_i$	Random Forest $\overline{\Delta MSE_{OOB}}_i$
A1	0.75	X1 (0.1)	-1.1661 (0.000)	1.0000 (0.000)	0.0922 (0.000)
		X2 (0.3)	-0.3196 (0.000)	1.0000 (0.000)	0.0894 (0.000)
		X3 (0.5)	0.5754 (0.000)	1.0000 (0.000)	0.2998 (0.000)
		X4 (0.7)	1.5130 (0.000)	1.0000 (0.000)	0.5831 (0.000)
A2	0.5	X1 (0.1)	-0.4830 (0.000)	0.9967 (0.000)	0.0303 (0.000)
		X2 (0.3)	-0.2222 (0.000)	0.8598 (0.002)	0.0167 (0.000)
		X3 (0.5)	0.4307 (0.000)	0.9959 (0.000)	0.2593 (0.000)
		X4 (0.7)	0.9057 (0.000)	1.0000 (0.000)	0.3279 (0.000)
A3	0.25	X1 (0.1)	-0.2026 (0.000)	0.9266 (0.002)	0.0045 (0.000)
		X2 (0.3)	--	0.3971 (0.004)	-0.0055 (0.000)
		X3 (0.5)	0.2987 (0.000)	0.9882 (0.001)	0.0481 (0.000)
		X4 (0.7)	0.8065 (0.000)	1.0000 (0.000)	0.5296 (0.000)
A4	0.0	X1 (0.1)	--	0.4002 (0.004)	-0.0313 (0.000)
		X2 (0.3)	0.3751 (0.000)	1.0000 (0.000)	0.0880 (0.000)
		X3 (0.5)	0.5019 (0.000)	1.0000 (0.000)	0.1211 (0.000)
		X4 (0.7)	0.6604 (0.000)	1.0000 (0.000)	0.2809 (0.000)

Table 14

A1 – A4 Inclusion Results: Proportion of Final Models that Included Predictors

Simulation $\rho(X_i, X_j)$	Predictor $\rho(Y, X_i)$	Stepwise Regression ($R = 40$)	All Subsets ($R = 40$)	Random Forest ($R = 40$)
A1 (0.75)	X1 (0.1)	40 (100%)	40 (100%)	40 (100%)
	X2 (0.3)	40 (100%)	40 (100%)	0 (0%)
	X3 (0.5)	40 (100%)	40 (100%)	40 (100%)
	X4 (0.7)	40 (100%)	40 (100%)	40 (100%)
A2 (0.5)	X1 (0.1)	40 (100%)	40 (100%)	0 (0%)
	X2 (0.3)	40 (100%)	40 (100%)	0 (0%)
	X3 (0.5)	40 (100%)	40 (100%)	40 (100%)
	X4 (0.7)	40 (100%)	40 (100%)	40 (100%)
A3 (0.25)	X1 (0.1)	40 (100%)	40 (100%)	0 (0%)
	X2 (0.3)	0 (0%)	0 (0%)	0 (0%)
	X3 (0.5)	40 (100%)	40 (100%)	40 (100%)
	X4 (0.7)	40 (100%)	40 (100%)	40 (100%)
A4 (0.0)	X1 (0.1)	0 (0%)	0 (0%)	0 (0%)
	X2 (0.3)	40 (100%)	40 (100%)	40 (100%)
	X3 (0.5)	40 (100%)	40 (100%)	40 (100%)
	X4 (0.7)	40 (100%)	40 (100%)	40 (100%)

The goal of the second set of simulations was to clarify how multicollinearity between predictors affects the importance ranking of variables that are not related to the criterion. That is, whether the ranking of a predictor that is not related to the criterion is impacted by that predictor being correlated with a predictor that is related to the criterion. The five correlational structures in set B have only one predictor (X1) that is correlated

with Y (ρ = .6). The correlation of X1 with two predictors (X2 and X3) that are not correlated with Y (ρ = 0) is varied across the B sets in order to investigate how this inter-predictor correlation impacts the three variable selection methods.

Results from Azen et al. (2001)'s comparisons of the three different methods for choosing best-fitting-models in simulation set B were largely similar to those of simulation set A. Overall there are small differences between the criticalities given by each approach with AIC yielding lower criticality values compared to adjusted r-squared and Mallow's C_p. In both Azen et al. results using adjusted r-squared and in the current study, final best-fitting models for all simulations sets included all three predictors expect for B4, which included only X1 and X3.

As in simulation set A, the predictor criticality values given by adjusted r-squared in the Azen et al. study varied somewhat from those in the current study. In both Azen et al. and the current study, X1 was given a maximal criticality of 1 in all simulation sets but for the other predictors values in this study were different from those in Azen et al. In the B1 simulation set, the Azen et al. adjusted r-squared predictor criticalities for X2 and X3 (C =0.512 and C = 0.496 respecitively) were lower than those of the current study (X2: C = 0.8841 and X3: C = 0.8164). Following a similar pattern, the Azen et al. X2 and X3 values in B2 (C = 0.498 and C = 0.909 respectively) were smaller than those in the current study (C = 0.558 and C = 0.971 respectively). In B3, Azen et al. criticality values for X2 and X3 were 0.749 and 0.824 respectively while those in the current study were 0.966 and 0.660 respectively. Among the Azen et al. X2 and X3 values in B4 (C = 0.461 and C = 1 respectively) and B5 (C = 0.982 and C = 0.985), some differed from those in the current study. The current study's predictor criticality values for X2 in B4 (C = 0.382)

and B5 (C = 0.889) were smaller than in Azen et. al. The current study's predictor criticality for X3 (C = 1) was the same as that of Azen et al. in B4, but larger than that of Azen et al. in B5 (C = 1).

In Azen et al. (2001), simulation results indicated that increases in the correlation between the predictors and X1 resulted in higher criticality measures. As can be seen in Table 15, despite the predictor criticality differences outline above, results in this study reflect those in Azen et al. The differences in predictor criticality are likely due to random sampling. The overall pattern found for the all-subsets method follows that reported by Azen et al. (2001); in general, the higher the relationship between the predictor and X1, the higher the predictors importance ranking.

Stepwise regression rankings, measured again by beta-values, reflect those of the all-subsets method. The X1 predictor consistently has a high beta value, but the beta values of X1 and X2 increase in the simulations where their relationship with X1 is higher. As in the all subsets' results, there is a discrepancy between the ranking of X3 in B1 and B3, with X3 in B1 having a higher ranking than it does in B3. Similarly, X2 is ranked slightly higher in B1 (where both X2 and X3 are unrelated to X1) than in B5, where both X2 and X3 are related to X1 (both ρ's = $\sqrt{0.2}$).

Random forest variable importance rankings showed greater resiliency to multicollinearity than the two other methods. As with the all subsets and stepwise regression methods, X1 had the highest variable importance across all simulation sets. However, X2 was ranked very low ($\overline{\Delta MSE_{OOB_i}} < 0$) in four of the five sets, including in B3 and B5 where it was correlated with X1. Similar to the other two methods, random forest gave X2 a higher ranking in the B1 set, where no multicollinearity was simulated,

than any other set. The random forest method's rankings of X3 also seemed to improve upon those of stepwise regression and the all-subsets method. Again, the $\overline{\Delta MSE_{OOB_i}}$ measure for X3 was higher in the B1 correlation structure, where no correlation with X1 existed than in B2 and B3, where X3 was correlated with X1. However, the X3 predictor was given a very low ranking ($\overline{\Delta MSE_{OOB_i}} < 0$) in three of the five sets, including B2 and B3. Rankings for X3 were highest in the B5 condition, where both X2 and X3 were related to X1 (both ρ's = $\sqrt{0.2}$) and in B4, where X3 alone was related to X1 ($\rho = \sqrt{0.4}$).

Overall, results from the B set of simulations indicated that all methods are more likely to exclude unrelated predictors when they are the only predictors that are unrelated to relevant predictors in the model. That is, when one of the irrelevant predictors is related to the relevant predictor, it has a suppressor effect on irrelevant variance in the relevant predictor. This results in an improvement of the model fit and a higher partial correlation between that irrelevant predictor and the criterion. That irrelevant predictor is then more likely to be retained in the final model while the other unrelated predictor is excluded. When this was the case in the simulation (in sets B2 and B4), stepwise regression method excluded the unrelated predictor (X2) entirely; in these same sets the all-subsets method ranked X2 its lowest. The random forest method is the exception to this pattern, in general its relative rankings of the unrelated predictors were lower than either of the other two methods.

As with the first set of simulations, the advantage of the random methods approach is highlighted by examining the inclusion of predictors in the models for simulation set B (shown in Table 16). Consistent with simulation set A, there was no variation across the 40 replications; predictors were either included or excluded across all

40 replications. In the stepwise regression method, final models in the B1, B3, and B5 simulations included all three predictors. Final models in the B2 and B4 simulations, excluded X2 but kept X3 (which was correlated with X1 in those simulations). The all-subsets method excluded an unrelated predictor in only one of the simulation sets, in B4 (where X3 had the highest relationship with X1 and X2 was unrelated to X1), the X2 predictor was not included in the final models. This finding is consistent with Azen et al.'s (2001) finding that, when using adjusted R^2 to determine predictor criticality, the full model with all three predictors was found to be the best-fitting model in all simulation sets except for B4, when a model with only X1 and X3 was found to be the best fitting. The random forest method's final models exclude X2 in all simulation sets except for B1, in which they exclude X3. Random forest final models exclude both X2 and X3 in the B2 and B3 conditions.

Table 15

Simulations B1 – B5 Importance Measure Results: Mean Importance Measures and their Standard Errors Over 40 Replications

				Mean Importance Measure (Standard Error) $n = 50$, $R = 40$		
Simulation	$\rho(X_1, X_2)$	$\rho(X_1, X_3)$	Predictor $\rho(Y, X_i)$	Stepwise $\bar{\beta}_i$	All Subsets $\bar{C}_i$	Random Forest $\overline{\Delta MSE_{OOB_i}}$
B1	0	0	X1 (0.6)	0.5942 (0.000)	1.0000 (0.000)	0.3142 (0.000)
			X2 (0.0)	-0.2419 (0.000)	0.8841 (0.003)	0.1810 (0.000)
			X3 (0.0)	-0.1877 (0.000)	0.8164 (0.003)	-0.0230 (0.000)
B2	0	$\sqrt{0.2}$	X1 (0.6)	0.8049 (0.000)	1.0000 (0.000)	0.4324 (0.000)
			X2 (0.0)	--	0.5575 (0.004)	-0.0485 (0.000)
			X3 (0.0)	-0.3371 (0.000)	0.9706 (0.001)	-0.0431 (0.000)
B3	$\sqrt{0.1}$	$\sqrt{0.1}$	X1 (0.6)	0.8067 (0.000)	1.0000 (0.000)	0.4058 (0.000)
			X2 (0.0)	-0.3820 (0.000)	0.9655 (0.001)	-0.0004 (0.000)
			X3 (0.0)	-0.1640 (0.000)	0.6599 (0.003)	-0.0405 (0.000)
B4	0	$\sqrt{0.4}$	X1 (0.6)	1.0848 (0.000)	1.0000 (0.000)	0.6486 (0.000)
			X2 (0.0)	--	0.3819 (0.004)	-0.0489 (0.000)
			X3 (0.0)	-0.6322 (0.000)	0.9999 (0.000)	0.0808 (0.000)
B5	$\sqrt{0.2}$	$\sqrt{0.2}$	X1 (0.6)	0.8588 (0.000)	1.0000 (0.000)	0.4154 (0.000)
			X2 (0.0)	-0.2283 (0.000)	0.8889 (0.002)	-0.0587 (0.000)
			X3 (0.0)	-0.6220 (0.000)	0.9999 (0.000)	0.1393 (0.000)

Table 16

B1 – B5 Inclusion Results: Proportion of Final Models that Included Predictors

Simulation $\rho(X_1, X_i)$	Predictors $\rho(Y, X_i)$	Stepwise Regression ($R = 40$)	All Subsets ($R = 40$)	Random Forest ($R = 40$)
B1	X1 (0.6)	40 (100%)	40 (100%)	40 (100%)
$\rho(X_1, X_2) = 0$	X2 (0.0)	40 (100%)	40 (100%)	40 (100%)
$\rho(X_1, X_3) = 0$	X3 (0.0)	40 (100%)	40 (100%)	0 (0%)
B2	X1 (0.6)	40 (100%)	40 (100%)	40 (100%)
$\rho(X_1, X_2) = 0$	X2 (0.0)	0 (0%)	40 (100%)	0 (0%)
$\rho(X_1, X_3) = \sqrt{0.2}$	X3 (0.0)	40 (100%)	40 (100%)	0 (0%)
B3	X1 (0.6)	40 (100%)	40 (100%)	40 (100%)
$\rho(X_1, X_2) = \sqrt{0.1}$	X2 (0.0)	40 (100%)	40 (100%)	0 (0%)
$\rho(X_1, X_3) = \sqrt{0.1}$	X3 (0.0)	40 (100%)	40 (100%)	0 (0%)
B4	X1 (0.6)	40 (100%)	40 (100%)	40 (100%)
$\rho(X_1, X_2) = 0$	X2 (0.0)	0 (0%)	0 (0%)	0 (0%)
$\rho(X_1, X_3) = \sqrt{0.4}$	X3 (0.0)	40 (100%)	40 (100%)	40 (100%)
B5	X1 (0.6)	40 (100%)	40 (100%)	40 (100%)
$\rho(X_1, X_2) = \sqrt{0.2}$	X2 (0.0)	40 (100%)	40 (100%)	0 (0%)
$\rho(X_1, X_3) = \sqrt{0.2}$	X3 (0.0)	40 (100%)	40 (100%)	40 (100%)

Discussion

Results from Study Three further clarify the results of Study Two and offer some further insight on the performance of the three modeling methods. The first set of analyses in Study Three indicated that for all three methods, high multicollinearity leads to misidentification of the relative importance of predictors, particularly those with smaller relationships with the criterion variable. In the case of high multicollinearity (ρ = .75), the all-subsets method ranked all predictors as equally and maximally critical. In contrast, the stepwise method provided different rankings for each predictor, but their order did not match their simulated rankings; the lowest related predictor had, on average, the second largest beta-value. The random forest method too, differentially ranked each predictor but confused the ranking of the two predictors with the smallest relationship to the criterion. In the next multicollinearity condition (ρ = .5), disorder amongst the predictor rankings was evident for all methods. The stepwise and all-subsets methods provided similarly sized rankings for the smallest related predictor and the second largest predictor, with a lower ranking for the second to smallest predictor. The random forest method mistook the order of the two predictors that were least related to the criterion, but correctly identified the two predictors most related to the criterion in their correct order. The pattern for the next collinearity condition (ρ = .25) was largely similar, with the second least related predictor being given the lowest ranking.

When no multicollinearity was included in the model, differences in the performance of the three methods in providing relative rankings are again apparent. The stepwise and random forest methods here provide rankings that correctly order the predictors according to their impact on the criterion variables. In contrast, the all-subsets

method gave the lowest ranking to the predictor least related to the criterion but ranked the other three predictors as equally critical.

The results of this first set of analyses indicates that random forest is the preferred variable selection method for reducing predictor pools and interpreting predictor impact. In cases of extreme multicollinearity, the all-subsets method resorts to rating all predictors equally and maximally critical, giving the researcher the impression that all predictors are equally important when in fact their relationship with the criterion is not equal. In such a setting the final model that the judgment researcher would be left with would include all the predictors; however, the varying relationship between each predictor with the criterion would not be interpretable. Under conditions of median to high multicollinearity, the stepwise method would result in a final model that included all predictors but misidentified the rankings of predictors with a lower relationship to the criterion, leading the researcher to incorrectly estimate the relative impact of some predictors. Such misidentification of predictor rankings also occurs with the all-subsets method in cases of less extreme multicollinearity.

The random forest method rankings for the two least related predictors were unordered, but in all except the most extreme condition of multicollinearity, the final model extracted by this method was consistent with the actual predictor-criterion relationship order. In the most extreme case, the predictor with the second smallest relationship with the criterion was excluded from the model. In the two other cases of multicollinearity, however, the two final models resulted in the two predictors with the smallest relationships to the criterion being excluded, and correctly ranked the two predictors that were maintained in the model. In contrast, the stepwise and all-subsets

method in the $\rho = .25$ collinearity condition resulted in a final model similar to the model produced by the random forest when multicollinearity was at its highest ($\rho = .75$).

The goal of the researcher and the correlational structure of the task environment may also determine which variable selection method is preferred. If the researcher would prefer that model retain all predictors in cases when multicollinearity is high, then the stepwise or all-subsets method, which retain all predictors in that case, would be preferable to the random forest method, which retains a predictor that is less relevant than one that is excluded. However, the stepwise and all-subsets methods behave in the same way under conditions when multicollinearity is much lower, while under those conditions the random forest method results in smaller models with more accurate interpretations of the relative impact of the predictors.

The second set of analyses provides an even clearer endorsement of the comparative advantage of the random forest method. Here, under all five collinearity conditions, the random forest excluded at least one of the irrelevant variables. In one condition it correctly excluded both. Stepwise regression excluded one irrelevant variable under two collinearity conditions and the all-subsets method also excluded one irrelevant variable, but only under one collinearity condition.

The random forest method's variable importance measure also had interpretive advantages over the all subsets' predictor criticality and stepwise's beta-weights. Although all variable importance methods incorrectly ranked the two irrelevant predictors as having different levels of importance in predicting the criterion variable, the random forest metric's rankings for these irrelevant predictors were much lower, when compared to the relevant predictors ranking, than those supplied by the stepwise and all-subsets

method. Indeed, in this interpretative aspect, as well as in the final model composition, the all-subsets method performed worse than the stepwise method as well as the random forest method. This was due to the all subsets predictor criticality rankings for the irrelevant predictors being relatively high in several of the collinearity conditions. All three methods correctly ranked the single relevant predictor as the most important in every collinearity condition. However, in several conditions the difference in ranking of the relevant and irrelevant predictors is practically insignificant. For instance, in all five of the collinearity conditions the relevant predictor had the maximum criticality ranking of 1.00, but in four of the collinearity conditions, an irrelevant predictor was given a value above 0.95, and, in two cases, that value is 0.9999 which would likely lead a researcher to interpret the irrelevant predictor as just as important as the relevant one.

The results of Study Three further indicate that the random forest method may provide a valuable alternative strategy to modeling human judgment. The random forest method's results more accurately reflected the relative importance of predictors compared to the stepwise and all-subsets methods, and it resulted in models that were more likely to exclude irrelevant predictors. The results of this study also help to clarify the results of Study Two, indicating that a key difference between the all subsets and random forest methods is that the former is more likely to retain predictors in a final model and the latter is more likely to exclude predictors. This feature of these two methods should be attended to by judgment researchers in their application of these models to ensure that modeling choice is consistent with researchers' task environment, task data, and overall research goals.

CHAPTER 6: GENERAL DISCUSSION

Judgment research has benefitted from the development and application of judgment analysis. The usefulness of judgment analysis to the field of psychology has been demonstrated by hundreds of studies across the last six decades. The emphases of judgment analysis—its focus on the importance of the task environment, representative design, vicarious functioning, and the individual as the unit of analysis—are the foundation of the judgment analysis approach but also complicate its application.

This study's introduction outlines the ways in which each of the foundational principles contribute to a more representative, constructive approach to human judgment, but may also impede its widespread implementation. When theoretical guidance on reducing a potential predictor pool is lacking, the use of automatic variable selection methods can facilitate a more widespread application of judgment analysis. Rapid, automated, reduction of full models, when appropriate, could promote researchers' use of more task environments that better represent the judgment task. Furthermore, such methods could enable researchers to apply judgment analysis to judgment research even when large numbers of individual judgments are not viable or possible.

The current research has implications for variable selection methods employed by judgment researchers. In the first study, a comparison of goodness-of-fit and cross-validation measures across methods found that these measures are not necessarily congruent. In that study, final judgment models generated by the all-subsets method had, on average, higher goodness-of-fit than final judgment models from both the stepwise regression and random forest methods. However, final judgment models from the all-subsets method did not cross-validate as well as the two other methods; both stepwise

regression and random forest final models had higher cross-validity, on average, than the all-subsets method. The random forest method was found to result in the highest degree of cross-validity, on average, despite being second to the all-subsets method in average goodness-of-fit. This finding, along with differences in the average sizes of the three methods final models, indicated the possibility that the all-subsets method was overfitting the data; maximizing goodness-of-fit at the cost of generalizability.

The incongruence in goodness-of-fit and generalizability maximization found by Study One was further illuminated by Study Two. In Study Two, models' inclusion of preset relevant variables was impacted both by the modeling method used and the weight of the predictor in simulated judgment rankings. Here again, as in Study One, differences in average model size indicated over-fitting by the all-subsets method in comparison to the other two methods. In the question of whether a methods' final model included the correct variables, the all-subsets method outperformed stepwise regression in including calories, sodium, sugars, and dietary fiber, and the random forest method in including calories and sugars. The random forest method had greater likelihood than the all-subsets method of including two of the four relevant variables, and greater likelihood than stepwise regression in including three of the four relevant variables.

The effect of weight in Study Two was not consistent across predictors. In two of the four cases, the higher simulated coefficient values were associated with greater likelihood that predictors would be included in the final models after controlling for method, but this was not the case for sugar and dietary fiber. In examining inclusion proportions, sugar frequency in model inclusion is in the expected direction for stepwise regression and all subsets models, but in the reverse for random forest models (i.e., higher

frequency for the low coefficient weight condition). Frequency of dietary fiber inclusion is greater in the high weight condition for stepwise regression models, but reversed for the all subsets and random forest method models (see Table 7).

The third and final study added insight into possible causes behind findings in the first two studies and provided some generalizable take-aways for future researchers. This study found that all three methods were susceptible to rank order reversals when multicollinearity between predictors was high. Nonetheless, the random forest method was more likely to exclude predictors, even under conditions of high multicollinearity. Notably, this study did not include a comparison of the impact of random forest's hyperparameters on variable selection importance. The values for *m*, trees, and forest hyperparameters used in this study were based on previous research finding these values to improve variable selection. Smaller values for *m* than *p*/3 could lead to lower probability of the correct predictors being included in the final model and smaller tree numbers could lead to higher variance in variable importance estimates (Genuer et al., 2010).

The current studies provide a comparison of three variable selection methods, implemented from a judgment analysis perspective. This study has sought to provide practical guidance for judgment researchers in applying automatic variable selection methods to allow for the wider application of the judgment analysis approach. Based on the results of these studies, the random forest methodology is recommended for further implementation in judgment analysis research, particularly in high-predictor environments when predictor reduction is desirable or necessary due to a high predictor to observation ratio. The current studies indicate that the random forest method yields

higher average cross-validity than the stepwise or all-subsets methods. Furthermore, its implementation given current technological tools is easy to accomplish and its computational complexity unlikely to be daunting to judgment researchers accustomed to the more traditional regression approach particularly when compared to the all-subsets method. The all-subsets analyses conducted in this study, when modeling and cross-validating 298 datasets of approximately 40 observations (in Study One) and modeling 300 datasets of size 73 (in Study Two) took approximately 5 days each. In contrast, the random forest analyses of the same datasets took approximately 20 minutes.

Even disregarding the added convenience of the computational simplicity of the random forest, Study Three indicated that the random forest method has generalizable advantages over the two alternatives examined. Namely, the random forest is less susceptible to multicollinearity than the all subsets and stepwise regression approaches, and it was more likely to disregard irrelevant predictors even when those predictors are related to other, relevant predictors. Yet, Study Three also indicated that the random forest is not impervious to variable selection errors due to multicollinearity. Results from the A group of correlational structures indicated that ranking confusion (particularly among variables that are weakly related to the criterion) is likely among all methods when multicollinearity is high. This is an indication that researchers—whichever modelling method they use and regardless of whether or not they employ a variable selection method—should be cautious in drawing strong conclusions from relative importance measures when multicollinearity is high.

The current body of work is not without limitations. As with any methodological study, the findings of this study will not necessarily apply to data that systematically

varies from the archival and simulated datasets used here. Despite this, the fact that all three studies had similar implications increases the researcher's confidence in the generalizability of these findings beyond the current work. Additionally, the use of an archival dataset from an actual judgment task in which predictors include high levels of multicollinearity increases this study's applicability to judgment analysis. By demonstrating the comparative performance of these three methods with actual judgment analysis data, this study provides insight into the performance judgment analysts might expect from these methods for tasks whose characteristics on the TCI are similar to those described here.

Finally, it should be noted that alternative variable selection methods, unexamined by the current work, may offer advantages beyond those documented among the methods tested here. A least absolute shrinkage and selection operator (LASSO) approach to variable selection tends to aggressively exclude predictors that are correlated with already selected predictors (Lu & Petkova, 2014). This is advantageous if many predictors in a highly-correlated task environment are expected to be irrelevant to judgment but whether or not this is the case may not be known. Other methods, such elastic net may be preferred to LASSO if there are many correlated predictors that are expected to be utilized by individuals in a judgment task (Lu & Petkova, 2014). Alternately, a Bayesian approach to variable selection can capture and integrate the uncertainty across fitted models. Unlike the variable selection method approaches used in the current studies, a Bayesian averaging approach integrates the uncertainty of a predictor's inclusion in a final model, allowing this uncertainty to weight importance of that predictor. Such a strategy allows for a more representative approach in prediction as it incorporates a

greater proportion of information from the available data (i.e., the uncertainty in predictor inclusion) rather than reducing that information to a dichotomous determination of a single accepted model that disregards all other possible models.

The current study did not examine a Bayesian approach to variable selection; however, given its integration of model uncertainty it seems particularly suited to the Brunswikian focus on probabilistic modeling. Although judgment analysis researchers have been far more consistent in their incorporation of probabilistic models than researchers in other programs of judgment research, the application of Bayesian analysis to model selection in judgment analysis has not been investigated. Future research should include the implementation of this approach to judgment analysis and provide a comparison to the more traditional, final-model approaches described here as well as other approaches (e.g., LASSO and elastic net).

Despite this study's limitations, it demonstrates that a random forest modeling approach to judgment analysis may provide variable selection advantages over the traditional linear regression approach. Not only can the inclusion of the random forest's variable selection methods that expand the applicability of judgment analysis, the random forest also provides a conception of the judgment process as a decision tree rather than a linear model. As was discussed in the introduction, judgment analysis is not restricted to a single modeling approach. Ineed, although the standard lens model equation statistics are conceptualized from a linear perspective these can be derived from a random forest as well. For instance, when incorporating variable selection using random forest, an adjusted r-squared value as referenced in Equation 6 can be used in place of the traditional r-squared value as a measure of the judge's consistency. The predicted values given by a

criterion random forest model and individual random forest model can be correlated in order to determine G—the agreement between the criterion and judgment models. In a random forest context, the error between the predicted values and actual values for both the criterion and individual random forest models can be correlated in order to provide a measure for C. In a random forest context C would still be a measure of the systematic variation that is shared by both the judgment and criterion that is not captured by the model. However, because the random forest model is non-linear, a C value calculated from two random forest models can no longer be interpreted as nonlinear variation shared by the criterion and judgment. Nonetheless, C would still be an indicator of shared systematic variation that was not accounted for by the random forest model, either due to relevant cues that are unaccounted for or failures in the modeling approach.

In summary, the current research indicates that applying random forest modeling to judgment analysis would be a fruitful union. Random forest's variable selection allows researchers to explore more representative judgment task environments even in cases where there is insufficient theory to reduce task predictors a priori. Future research comparing random forest's approach to variable selection to that of other methods, as well as explorations of the environments under linear models or decision-tree models better capture the judgment process would further aid judgment researchers' decision-making.

REFERENCES

Akaike, H. (1973). Informatino theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csáki (Eds.), *Proceeedings of the 2nd International Symposium on Information Theory*, 267-281. Budapest: Akadémiai Kiado.

Altmann, A., Tolosi, L., Sander, & O., Lengauer, T. (2010). Permutation importance: A corrected feature importance measure. *Bioinformatics, 26* (10), 1340–1347.

Azen, R., Budescu, D. V., & Reiser, B. (2001). Criticality of predictors in multiple regression. *British Journal of Mathematical and Statistical Psychology, 54*, 129-225.

Babyak, M. A. (2004). What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models. *Psychosomatic Medicine*, *66*(3), 411-421.

Barnes, G., & Hyatt, J. M. (2012). Classifying adult probationers by forecasting future offending. Washington, DC: National Institute of Justice.

Baron, J. (2004). Normative models of judgment and decision making. In D. J., Koehler & N. Harvey (Eds.), *Blackwell Handbook of Judgment and Decision Making,* (19-36), Malden: Blackwell Publishing.

Behnamian, A., Millard, K., Banks, S. N., White, L., Richardson, M., & Pasher, J. (2017). A systematic approach for variable selection with Random Forests: Achieving stable variable importance values. *IEEE Geoscience and Remote Sensing Letters*, *14*(11), 1988-1992.

Ben Ishak, A. (2016). Variable selection using support vector regression and random forests: A comparative study. *Intelligent Data Analysis*, *20*(1), 83-104.

Bonat, W. H., Ribeiro Jr, P. J., & Shimakura, S. E. (2015). Bayesian analysis for a class of beta mixed models. *Chilean Journal of Statistics (ChJS)*, *6*(1), 3-13.

Brehmer, B. (1988). The development of social judgment theory. In B. Brehmer, & C. R. B. Joyce (Eds.), *Human judgment: The SJT view* (13–40). Amsterdam: Elsevier.

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Breiman, L., & Cutler, A. (2008). Random forests. Available at: http://www.stat.berkeley.edu/users/breiman/RandomForests/cc_home.htm (accessed: 08.15.18)

Brighton, H., & Gigerenzer, G. (2015). The bias bias. *Journal of Business Research*, *68*(8), 1772-1784.

Brooks, S. P., & Gelman, A. (1998). General methods for monitoring convergence of iterative simulations. *Journal of Computational and Graphical Statistics*, *7*(4), 434-455.

Brunswik, E. (1952). The conceptual framework of psychology. In E. Brunswik (Ed.), Vol. 1. *International Encyclopedia of Unified Science* (p. 678). Chicago: University of Chicago Press, 1952.

Bürkner, P. C. (2018). Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, 1-15.

Carter, K. A., & González-Vallejo, C. (2018). Nutrient-specific system versus full fact panel: Testing the benefits of nutrient-specific front-of-package labels in a student sample. *Appetite*, *125*, 512-526.

Cooksey, R. W. (1996). *Judgment analysis: Theory, methods and applications.* San Diego: Academic Press.

Cornish, S. L., & Moraes, C. (2015). The impact of consumer confusion on nutrition literacy and subsequent dietary behavior. *Psychology & Marketing*, *32*(5), 558-574.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis, 1*(1-4), 131-156.

Dawes, R. M. (1979). The robust beauty of improper linear models in decision making. *American Psychologist*, *34*(7), 571.

Dawes, R. M., & Corrigan, B. (1974). Linear models in decision making. *Psychological Bulletin*, *81*(2), 95.

Derksen, S., & Keselman, H. J. (1992). Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables. *British Journal of Mathematical and Statistical Psychology*, *45*(2), 265-282.

Dhami, M. K., & Harries, C. (2001). Fast and frugal versus regression models of human judgement. *Thinking & Reasoning*, *7*(1), 5-27.

Díaz-Uriarte, R., & De Andres, S. A. (2006). Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, *7*(1), 3.

Dougherty, T.W., Ebert, R. J., & Callender, J. C. (1986). Policy capturing in the employment interview. *Journal of Applied Psychology*, *71*, 9-15.

Fischhoff, B. (2010). Judgment and decision making. *Wiley Interdisciplinary Reviews: Cognitive Science*, *1*(5), 724-735.

Gelman, A., Hwang, J., & Vehtari, A. (2014). Understanding predictive information criteria for Bayesian models. *Statistics and Computing*, *24*(6), 997-1016.

Gelman, A., Jakulin, A., Pittau, M. G., & Su, Y. S. (2008). A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*, *2*(4), 1360-1383.

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*(14), 2225-2236.

Genuer, R., Poggi, J. M., & Tuleau-Malot, C. (2015). VSURF: An R package for variable selection using random forests. *The R Journal*, *7*(2), 19-33.

Genuer, R., Poggi, J. M., Tuleau-Malot, C., & Genuer, M. R. (2018). Package 'VSURF'. R package version 1.04. https://cran.r-project.org/web/packages/VSURF/index.html

Gibson, C. J., & Hobson, F. W. (1983). Policy capturing as an approach to understanding and improving performance appraisal: a review of the literature. *Academy of Management Review*, *8*(4), 640-649.

Gigerenzer, G. (1996). The psychology of good judgment: Frequency formats and simple algorithms. *Medical Decision Making*, *16*(3), 273-280.

Girden, E. R. (1992). *ANOVA: Repeated measures* (No. 84). London: Sage.

Goldstein, W. M. (2004). Social judgment theory: Applying and extending Brunswik's probabilistic functionalism. In D. J., Koehler & N. Harvey (Eds.), *Blackwell Handbook of Judgment and Decision Making,* (37-61), Malden: Blackwell Publishing.

González-Vallejo, C., & Lavins, B. D. (2016). Evaluation of breakfast cereals with the current nutrition facts panel (NFP) and the Food and Drug Administration's NFP proposal. *Public Health Nutrition*, *19*(6), 1047-1058.

González-Vallejo, C., Lavins, B. D., & Carter, K. A. (2016). Analysis of nutrition judgments using the Nutrition Facts Panel. *Appetite*, *105*, 71-84.

Graves, L. M., & Karren, R. J. (1992). Interviewer decision processes and effectiveness: An experimental policy-capturing investigation. *Personnel Psychology*, *45*(2), 313-340.

Greenland, S. (1989). Modeling and variable selection in epidemiologic analysis. *American Journal of Public Health*, *79*(3), 340-349.

Grömping, U. (2009). Variable importance assessment in regression: linear regression versus random forest. *The American Statistician*, *63*(4), 308-319.

Guthrie, J. F., Derby, B. M., & Levy, A. S. (1999). What people know and do not know about nutrition. *America's Eating habits: Changes and Consequences*, 243-290.

Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research, 3*(March), 1157-1182.

Hammond, K. R. (1954). Representative vs. systematic design in clinical psychology. *Psychological Bulletin*, *51*(2), 150.

Hammond, K. R., & Adelman, L. (1976). Science, values, and human judgment. *Science*, *194*(4263), 389-396.

Hammond, K. R., Hamm, R. M., Grassia, J., & Pearson, T. (1987). Direct comparison of the efficacy of intuitive and analytical cognition in expert judgment. *IEEE Transactions on Systems, Man, and Cybernetics*, *17*(5), 753-770.

Hammond, K. R., Hursch, C. J., & Todd, F. J. (1964). Analyzing the components of clinical inference. *Psychological Review*, *71*(6), 438.

Hammond, K. R., McClelland, G. H., & Mumpower, J. (1980). *Human Judgment and Decision Making: Theories, Methods, and Procedures.* New York: Praeger Publishers.

Hammond, K., Stewart, T., Brehmer, B., & Steinmann, D. (1975). Social judgment theory. In M. Kaplan & S. Schwartz (Eds.), *Human Judgment and Decision Processes.* New York: Academic Press.

Hapfelmeier, A., & Ulm, K. (2013). A new variable selection approach using random forests. *Computational Statistics & Data Analysis*, *60*, 50-69.

Hartwig, M., & Bond Jr, C. F. (2011). Why do lie-catchers fail? A lens model meta-analysis of human lie judgments. *Psychological Bulletin*, *137*(4), 643.

Hawkins, D. M. (2004). The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1-12.

Hocking, R. R. (1976). A Biometrics invited paper. The analysis and selection of variables in linear regression. *Biometrics*, *32*(1), 1-49.

Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*(1), 1593-1623.

International Food Information Council (IFIC) (2017). A healthy perspective: Understanding American food values. *Food and Health Survey.* Available at: https://www.foodinsight.org/2017-food-and-health-survey

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112). New York: Springer.

Johnson, J. W., & LeBreton, J. M. (2004). History and use of relative importance indices in organizational research. *Organizational Research Methods*, *7*(3), 238-257.

Karelaia N. & Hogarth R. M. (2008). Determinants of linear judgment: A meta-analysis of lens model studies. *Psychological Review, 134* (3), 404-426.

Karren, R. J., & Barringer, M. W. (2002). A review and analysis of the policy-capturing methodology in organizational research: Guidelines for research and practice. *Organizational Research Methods*, *5*(4), 337-361.

Kaufmann, E., Reips, U., & Wittmann, W. W. (2013). A critical meta-analysis of lens model studies in human judgment and decision-making. *PLOS One, 8*(12).

Kim, C. N., Yang, K. H., & Kim, J. (2008). Human decision-making behavior and modeling effects. *Decision Support Systems*, *45*(3), 517-527.

Kiker, G. A., Bridges, T. S., Varghese, A., Seager, T. P., & Linkov, I. (2005). Application of multicriteria decision analysis in environmental decision making. *Integrated Environmental Assessment and Management*, *1*(2), 95-108.

Klein, S. B. (2014). What can recent replication failures tell us about the theoretical commitments of psychology? *Theory & Psychology*, *24*(3), 326-338.

Koehler, D. J., & Harvey, N. (Eds.). (2008). *Blackwell handbook of judgment and decision making*. Malden: Blackwell Publishing.

Kruschke, J. (2015). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. (2nd ed.). London: Academic Press.

Kutner, M. H., Nachtsheim, C. J., Neter, J., & Li, W. (2005). *Applied Linear Statistical Models.* Boston: McGraw-Hill Irwin.

Lafond, D., Vallières, B. R., Vachon, F., St-Louis, M. È., & Tremblay, S. (2015, September). Capturing non-linear judgment policies using decision tree models of classification behavior. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* (Vol. 59, No. 1, pp. 831-835). Los Angeles: SAGE Publications.

Lamiell, J. T. (1998). Nomothetic'andIdiographic' Contrasting Windelband's Understanding with Contemporary Usage. *Theory & Psychology*, *8*(1), 23-38.

Lin, Y., & Jeon, Y. (2006). Random forests and adaptive nearest neighbors. *Journal of the American Statistical Association*, *101*(474), 578-590.

Li, B., Lingsma, H. F., Steyerberg, E. W., & Lesaffre, E. (2011). Logistic random effects regression models: a comparison of statistical packages for binary and ordinal outcomes. *BMC Medical Research Methodology*, *11*(1), 77.

Li, J., Tran, M., & Siwabessy, J. (2016). Selecting optimal random forest predictive models: A case study on predicting the spatial distribution of seabed hardness. *PLOS One*, *11*(2), e0149089.

Liaw, A., & Wiener, M. (2002). Classification and regression by random forest. *R News*, *2*(3), 18-22.

Liu, H., & Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering, 17*(4), 491-502.

Lu, F., & Petkova, E. (2014). A comparative study of variable selection methods in the context of developing psychiatric screening instruments. *Statistics in Medicine, 33*(3), 401-421.

Mallows, C. L. (1973). Some comments on C_p. *Technometrics, 15,* 661-675.

Meehl, P. E. (1967). Theory-testing in psychology and physics: A methodological paradox. *Philosophy of Science*, *34*(2), 103-115.

Morozova, O., Levina, O., Uusküla, A., & Heimer, R. (2015). Comparison of subset selection methods in linear regression in the context of health-related quality of life and substance abuse in Russia. *BMC Medical Research Methodology*, *15*(1), 71.

Nisbett, R. E., & Wilson, T. D. (1977). Telling more than we can know: Verbal reports on mental processes. *Psychological Review*, *84*(3), 231.

Oshiro, T. M., Perez, P. S., & Baranauskas, J. A. (2012, July). How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (pp. 154-168). Heidelberg: Springer.

Pargament, K. I., Sullivan, M. S., Balzer, W. K., Van Haitsma, K. S., & Raymark, P. H. (1995). The many meanings of religiousness: A policy-capturing approach. *Journal of Personality*, *63*(4), 953-983.

Pitt, M. A., & Myung, I. J. (2002). When a good fit can be bad. *Trends in Cognitive Sciences*, *6*(10), 421-425.

Pitt, M. A., Kim, W., & Myung, I. J. (2003). Flexibility versus generalizability in model selection. *Psychonomic Bulletin & Review*, *10*(1), 29-44.

Plonsky, O., Erev, I., Hazan, T., & Tennenholtz, M. (2017). Psychological forest: Predicting human behavior. *In Thirty-First AAAI Conference on Artificial Intelligence* (pp. 656-662).

Probst, P., & Boulesteix, A. L. (2018). To Tune or Not to Tune the Number of Trees in Random Forest. *Journal of Machine Learning Research*, *18*(181), 1-18.

Ritter, N. (2013). Predicting recidivism risk: New tool in Philadelphia shows great promise. *National Institute of Justice Journal*, *271*, 4-13.

Sandri M. & Zuccolotto P. (2006). Variable selection using random forests. In: Zani S., Cerioli A., Riani M., Vichi M. (eds) *Data Analysis, Classification and the Forward Search. Studies in Classification, Data Analysis, and Knowledge Organization.* Springer, Berlin, Heidelberg

Shmueli, G. (2010). To explain or to predict? *Statistical Science, 25* (3), 289-310.

Slovic, P., & Lichtenstein, S. (1971). Comparison of Bayesian and regression approaches to the study of information processing in judgment. *Organizational Behavior and Human Performance*, *6* (6), 649-744.

Sorum, P. C., Stewart, T. R., Mullet, E., González-Vallejo, C., Shim, J., Chasseigne, G., Sastre, M. T. M., & Grenier, B. (2002). Does choosing a treatment depend on making a diagnosis? US and French physicians' decision making about acute otitis media. *Medical Decision Making*, *22*(5), 394-402.

Stewart, T. R. (1976). Components of correlation and extensions of the lens model equation. *Psychometrika, 41* (1), 101-120.

Stewart, T. R. (1988). Judgment analysis: Procedures. In B. Brehmer & C. R. B. Joyce (Eds.), *Human Judgment: The SJT View* (41-74). Amsterdam: Elsevier.

Stewart, T. R. (2001). The lens model equation. In K. R. Hammond & T. R. Stewart (Eds.), *The Essential Brunswik: Beginnings, Explications, Applications* (357-362). Oxford: Oxford University Press.

Stewart, T. R., Moninger, W. R., Grassia, J., Brady, R. H., & Merrem, F. H. (1989). Analysis of expert judgment in a hail forecasting experiment. *Weather and forecasting*, *4*(1), 24-34.

Stewart, T. R., Roebber, P. J., & Bosart, L. F. (1997). The importance of the task in analyzing expert judgment. *Organizational Behavior and Human Decision Processes*, *69*(3), 205-219.

Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: Rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological Methods*, *14* (4), 323-348.

Strobl, C., & Zeileis, A. (2008). Danger: High power!–exploring the statistical properties of a test for random forest variable importance. In P. Brito (Ed.) COMPSTAT 2008 – Proceedings in Computational Statistics, Volume II (59-66). Heidelberg: Physica Verlag.

Thompson, B. (1995). Stepwise regression and stepwise discriminant analysis need not apply here: A guidelines editorial. *Educational and Psychological Measurement, 55* (4), 525-534.

Tomassetti, A. J., Dalal, R. S., & Kaplan, S. A. (2016). Is policy capturing really more resistant than traditional self-report techniques to socially desirable responding? *Organizational Research Methods*, *19*(2), 255-285.

Watanabe, S. (2010). Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, *11*, 3571-3594.

Whittingham, M. J., Stephens, P. A., Bradbury, R. B., & Freckleton, R. P. (2006). Why do we still use stepwise modelling in ecology and behaviour? *Journal of Animal Ecology*, *75*(5), 1182-1189.

Windelband, W. (1998). History and natural science. *Theory and Psychology, 8* (1), 5.

Yarkoni, T., & Westfall, J. (2017). Choosing prediction over explanation in psychology: Lessons from machine learning. *Perspectives on Psychological Science*, *12*(6), 1100-1122.

Zeleny, M. (1976). On the inadequacy of the regression paradigm used in the study of human judgment. *Theory and Decision*, *7*(1-2), 57-65.

Zhang, J., & Kai, F. Y. (1998). What's the relative risk? A method of correcting the odds ratio in cohort studies of common outcomes. *Journal of the American Medical Association*, *280*(19), 1690-1691.

APPENDIX A

Food Evaluation Instructions and Example:

In this part of the study we want you to evaluate some common packaged foods. For each food item you will see a screen with a picture that shows its front packaging information. You will also have its Nutrition Facts Panel and the list of its ingredients. This information should be helpful as you evaluate the foods with regards to their nutrition. Please view each item and then answer the questions that follow as best as you can so that they represent your opinions.

In addition to evaluating each food item, we will ask you to hypothetically purchase the item by placing it in a shopping cart if you think it is desirable (a yes/no answer). For this part, assume cost is not an obstacle to possessing the item. However, also consider that you wouldn't want to add something to your cart that you wouldn't consume.

Condition One

Ingredients: Rice, sugar, maltodextrin, partially hydrogenated soybean oil, contains 2% or less of salt, natural and artificial flavor (contains milk), gelatin, malt flavoring, color added, BHT for freshness.

Vitamins and Minerals: Vitamin C (sodium ascorbate and ascorbic acid), niacinamide, reduced iron, vitamin B_6 (pyridoxine hydrochloride), vitamin B_2 (riboflavin), vitamin B_1 (thiamin hydrochloride), vitamin A palmitate, folic acid, vitamin D, vitamin B_{12}.

CONTAINS MILK INGREDIENTS.

Nutrition Facts

Serving Size ¾ Cup (30g)
Servings Per Container About 11

Amount Per Serving	Cereal	with ½ cup skim milk
Calories	120	160
Calories from Fat	10	10
		% Daily Value**
Total Fat 1g*	**2%**	**2%**
Saturated Fat 0g	**0%**	**0%**
Trans Fat 0g		
Polyunsaturated Fat 0g		
Monounsaturated Fat 0g		
Cholesterol 0mg	**0%**	**0%**
Sodium 170mg	**7%**	**10%**
Potassium 15mg	**0%**	**6%**
Total Carbohydrate 26g	**9%**	**11%**
Dietary Fiber 0g	**0%**	**0%**
Sugars 9g		
Protein 1g		
Vitamin A	10%	15%
Vitamin C	10%	10%
Calcium	0%	15%
Iron	10%	10%
Vitamin D	10%	25%
Thiamin	25%	30%
Riboflavin	25%	35%
Niacin	25%	25%
Vitamin B_6	25%	25%
Folic Acid	25%	25%
Vitamin B_{12}	25%	35%

* Amount in cereal. One half cup of skim milk contributes an additional 40 calories, 65mg sodium, 6g total carbohydrates (6g sugars), and 4g protein.

** Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs:

	Calories	2,000	2,500
Total Fat	Less than	65g	80g
Sat. Fat	Less than	20g	25g
Cholesterol	Less than	300mg	300mg
Sodium	Less than	2,400mg	2,400mg
Potassium		3,500mg	3,500mg
Total Carbohydrate		300g	375g
Dietary Fiber		25g	30g

Condition Two

Ingredients: Rice, sugar, maltodextrin, partially hydrogenated soybean oil, contains 2% or less of salt, natural and artificial flavor (contains milk), gelatin, malt flavoring, color added, BHT for freshness.

Vitamins and Minerals: Vitamin C (sodium ascorbate and ascorbic acid), niacinamide, reduced iron, vitamin B_6 (pyridoxine hydrochloride), vitamin B_2 (riboflavin), vitamin B_1 (thiamin hydrochloride), vitamin A palmitate, folic acid, vitamin D, vitamin B_{12}.

CONTAINS MILK INGREDIENTS.

Nutrition Facts

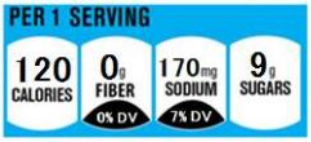
Serving Size 3/4 Cup (30g)
Servings Per Container About 11

Amount Per Serving	Cereal	with 1/2 cup skim milk
Calories	120	160
Calories from Fat	10	10
		% Daily Value**
Total Fat 1g*	**2%**	**2%**
Saturated Fat 0g	**0%**	**0%**
Trans Fat 0g		
Polyunsaturated Fat 0g		
Monounsaturated Fat 0g		
Cholesterol 0mg	**0%**	**0%**
Sodium 170mg	**7%**	**10%**
Potassium 15mg	**0%**	**6%**
Total Carbohydrate 26g	**9%**	**11%**
Dietary Fiber 0g	**0%**	**0%**
Sugars 9g		
Protein 1g		
Vitamin A	10%	15%
Vitamin C	10%	10%
Calcium	0%	15%
Iron	10%	10%
Vitamin D	10%	25%
Thiamin	25%	30%
Riboflavin	25%	35%
Niacin	25%	25%
Vitamin B_6	25%	25%
Folic Acid	25%	25%
Vitamin B_{12}	25%	35%

* Amount in cereal. One half cup of skim milk contributes an additional 40 calories, 65mg sodium, 6g total carbohydrates (6g sugars), and 4g protein.

** Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs:

	Calories	2,000	2,500
Total Fat	Less than	65g	80g
Sat. Fat	Less than	20g	25g
Cholesterol	Less than	300mg	300mg
Sodium	Less than	2,400mg	2,400mg
Potassium		3,500mg	3,500mg
Total Carbohydrate		300g	375g
Dietary Fiber		25g	30g



Condition Three

Ingredients: Rice, sugar, maltodextrin, partially hydrogenated soybean oil, contains 2% or less of salt, natural and artificial flavor (contains milk), gelatin, malt flavoring, color added, BHT for freshness.

Vitamins and Minerals: Vitamin C (sodium ascorbate and ascorbic acid), niacinamide, reduced iron, vitamin B_6 (pyridoxine hydrochloride), vitamin B_2 (riboflavin), vitamin B_1 (thiamin hydrochloride), vitamin A palmitate, folic acid, vitamin D, vitamin B_{12}.

CONTAINS MILK INGREDIENTS.

Nutrition Facts

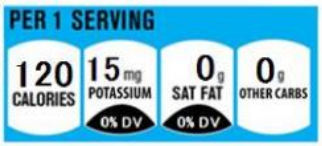
Serving Size 3/4 Cup (30g)
Servings Per Container About 11

Amount Per Serving	Cereal	with 1/2 cup skim milk
Calories	120	160
Calories from Fat	10	10
		% Daily Value**
Total Fat 1g*	**2%**	**2%**
Saturated Fat 0g	**0%**	**0%**
Trans Fat 0g		
Polyunsaturated Fat 0g		
Monounsaturated Fat 0g		
Cholesterol 0mg	**0%**	**0%**
Sodium 170mg	**7%**	**10%**
Potassium 15mg	**0%**	**6%**
Total Carbohydrate 26g	**9%**	**11%**
Dietary Fiber 0g	**0%**	**0%**
Sugars 9g		
Protein 1g		
Vitamin A	10%	15%
Vitamin C	10%	10%
Calcium	0%	15%
Iron	10%	10%
Vitamin D	10%	25%
Thiamin	25%	30%
Riboflavin	25%	35%
Niacin	25%	25%
Vitamin B_6	25%	25%
Folic Acid	25%	25%
Vitamin B_{12}	25%	35%

* Amount in cereal. One half cup of skim milk contributes an additional 40 calories, 65mg sodium, 6g total carbohydrates (6g sugars), and 4g protein.

** Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs:

	Calories	2,000	2,500
Total Fat	Less than	65g	80g
Sat. Fat	Less than	20g	25g
Cholesterol	Less than	300mg	300mg
Sodium	Less than	2,400mg	2,400mg
Potassium		3,500mg	3,500mg
Total Carbohydrate		300g	375g
Dietary Fiber		25g	30g



APPENDIX B

Environment Analysis R Code

Stepwise Regression Analysis

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("purr")
#install.packages("leaps")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)

CerealLevel <- read_excel("~/Ohio University/Carter.FOP Study.Cereal Level Dataset
Abbreviated.xlsx")

NuValdfVar <- c("V1", "Cereal", "NuValScore", "RecordID", "NumberID", "Condition",
"Setting", "UPC", "CerealNumber", "Calories", "Total_Fatg", "Saturated_Fatg",
"Poly_Fatg", "Mono_Fatg", "Sodiummg", "Potassiummg", "TotalCarbsg",
"DietaryFiberg", "Sugars_g", "Proteing", "VitsMinerals_NuVal")

NuValComplete <- SingleList[complete.cases(SingleList[, NuValdfVar]), NuValdfVar]

rm(list = ls()[!ls()%in%c("NuValComplete")])

####################STEPWISE ANALYSIS###################################
set.seed(711)
  datalist <- as.data.frame(NuValComplete)
  full_model <- lm(NuValScore ~ Calories + Total_Fatg + Saturated_Fatg +
                     Poly_Fatg + Mono_Fatg + Sodiummg + Potassiummg +
                     DietaryFiberg + Sugars_g +
                     TotalCarbsg + Proteing + VitsMinerals_NuVal,
                     data = datalist)

  null_model <- lm(NuValScore ~ 1, data = datalist)
```

```
stepwise <- step(null_model, scope = list(upper = full_model))

scale_full_model <- lm(scale(NuValScore) ~ scale(Calories) + scale(Total_Fatg) +
                         scale(Saturated_Fatg) +  scale(Poly_Fatg) + scale(Mono_Fatg) +
                         scale(Sodiummg) + scale(Potassiummg) + scale(DietaryFiberg) +
                         scale(Sugars_g) + scale(TotalCarbsg) + scale(Proteing) +
                         scale(VitsMinerals_NuVal),
                       data = datalist)

scale_null_model <- lm(scale(NuValScore) ~ 1, data = datalist)

scale_stepwise <- step(scale_null_model, scope = list(upper = scale_full_model))

summaries <- summary(stepwise)

require(broom)
df_coefficients <- tidy(stepwise)
df_betas <- tidy(scale_stepwise)
df_fstat <- bind_rows(summaries[["fstatistic"]])

#################################################################
### Model Statistics ###

#Adjusted R^2#
NuValmodel_stats <- as.data.frame(rbind(summaries[9]))
NuValmodel <- map_df(NuValmodel_stats, ~as.data.frame(.x), .id="NumberID")
NuValmodel$NumberID <- 'NuVal'

#R^2 included in person judgments in order to maintain individuals for whom
#stepwise models could not be created in dataset, not needed here.

#F-Statistics#
NuValfstats <- bind_rows(df_fstat)
NuValfstats$fstat <- NuValfstats$value
NuValfstats$model_var <- NuValfstats$numdf
NuValfstatVar <- c("fstat", "numdf", "dendf", "model_var")
fstatistics <- NuValfstats[complete.cases(NuValfstats[, NuValfstatVar]), NuValfstatVar]

#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_model <- merge(NuValmodel, fstatistics)
stepwise_model <- as.data.frame(stepwise_model)

#Computing p-values, assigning binary significance values#
stepwise_model$pvalue<- pf(stepwise_model$fstat, stepwise_model$numdf,
stepwise_model$dendf, lower.tail = FALSE)
```

```
stepwise_model$modelsig <- ifelse(stepwise_model$pvalue < .05, 1, 0)
#####################################################################

#Creating dataframe for unstandardized coefficient estimates#
coefficients <- bind_rows(df_coefficients)

#Creating mergeable dataframe for standardized coefficient estimates#
betas_mid <- bind_rows(df_betas)
betas_mid$term2 <- gsub("scale\\(", "", betas_mid$term)
betas_mid$term3 <- gsub("g[)]", "g", betas_mid$term2)
betas_mid$term <- gsub("s[)]", "s", betas_mid$term3)

betas_mid$beta <- betas_mid$estimate
betaVar <- c("term", "beta")
betas <- betas_mid[complete.cases(betas_mid[, betaVar]), betaVar]
betas$beta_value <- ifelse(betas$term == '(Intercept)', NA, betas$beta)

#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_coeff <- merge(coefficients, betas, by = c('term'))

########Assigning Meta Values:########################################

#Significance values#
stepwise_coeff$coeffsig <- ifelse(stepwise_coeff$p.value < .05, 1, 0)

#Positive/negative betas#
stepwise_coeff$coeffdirection <- ifelse(stepwise_coeff$beta > 0, 'positive', 'negative')

#Ranking values#
  coeff_data <- as.data.frame(stepwise_coeff)
  coeff_data$rank <- rank(-abs(coeff_data$beta_value), na.last = FALSE, ties.method =
"average")
  coeff_data$adjrank <- coeff_data$rank - 1
  final_stepwise_coeff <- bind_rows(coeff_data)

#####################################################################
##SAVING ON PERSONAL PC: PC1###.
write.csv(final_stepwise_coeff, file =
"C:/Users/Kristina/Documents/NuVal_Onlyfinal_stepwise_coeff.csv")
write.csv(stepwise_model, file =
"C:/Users/Kristina/Documents/NuVal_Onlyfinal_stepwise_model.csv")
```

All Subsets Analysis

```
#install.packages("tidyverse")
#install.packages("broom")
```

```
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("olsrr")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
#library(purrr)
#library(dplyr)
library(leaps)

CerealLevel <- read_excel("~/Ohio University/Carter.FOP Study.Cereal Level Dataset
Abbreviated.xlsx")

SingleList <- (subset(CerealLevel, NumberID == 1))

NuValdfVar <- c("V1", "Cereal", "NuValScore", "RecordID", "NumberID", "Condition",
"Setting", "UPC", "CerealNumber", "Calories", "Total_Fatg",
"Saturated_Fatg", "Poly_Fatg", "Mono_Fatg", "Sodiummg", "Potassiummg",
"TotalCarbsg", "DietaryFiberg", "Sugars_g", "Proteing", "VitsMinerals_NuVal")

NuValComplete <- SingleList[complete.cases(SingleList[, NuValdfVar]), NuValdfVar]

rm(list = ls()[!ls()%in%c("NuValComplete")])

datalist <- list()
resamples <- list()
judgment_set <- list()
leaps <- list()
summary <- list()
maxadjr2 <- list()
variables1 <- list()
variables2 <- list()
variables3 <- list()
coefficients1 <- list()
coefficients2 <- list()
bestmodel <- list()
judge_bestmodels <- list()
count <- list()

bestfit <- list()
```

```
only_bestmodels <- list()

modelsonly1 <- list()
modelsonly2 <- list()
modelsonly3 <- list()
modelsonly4 <- list()
freqmodel <- list()
variable_importance <- list()
num <- list()

#####ENVIRONMENT ALL SUBSETS ANALYSIS############################
start <- 1    #number judge to begin at
judges <- 1   #number of judges to include
bootset <- 100000  #number of datasets to bootstrap
################################################################################

#RUN BELOW AS SINGLE BLOCK:
t1 <- Sys.time()
set.seed(711)
for(k in start:judges){
  num[[k]] <- k
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(NuValComplete, NumberID == k))
  resamples[[k]] <- lapply(1:bootset, function(i) sample_n(datalist[[k]], 74, replace = T))
  for(j in 1:bootset){
    t3 <- Sys.time()
    judgment_set[[j]] <- resamples[[k]][[j]]
    leaps[[j]] <-
      regsubsets(scale(NuValScore) ~ scale(Calories) + scale(Total_Fatg) +
scale(Saturated_Fatg) +  scale(Poly_Fatg) + scale(Mono_Fatg) + scale(Sodiummg) +
scale(Potassiummg) +  scale(DietaryFiberg) + scale(Sugars_g) +
                  scale(TotalCarbsg) + scale(Proteing) + scale(VitsMinerals_NuVal),
                data = judgment_set[[j]],
                nbest = 1,       # 1 best model for each number of predictors
                nvmax = NULL,    # NULL for no limit on number of variables
                force.in = NULL, force.out = NULL,
                method = "exhaustive")
    summary[[j]] <- summary(leaps[[j]])
    maxadjr2[[j]] <- which.max(summary[[j]]$adjr2)
    variables1[[j]] <- map_df(summary[[j]]$which[maxadjr2[[j]],], ~as.data.frame(.x),
.id="term")
    variables1[[j]]$included <- variables1[[j]]$.x
    variables2[[j]] <- variables1[[j]][-c(2)]
    variables3[[j]] <- as.data.frame(subset(variables2[[j]]))
    variables3[[j]]$adj.r.squared <- summary[[j]]$adjr2[maxadjr2[[j]]]
    variables3[[j]]$inclusion <- ifelse(variables3[[j]]$included == TRUE, 1, 0)
```

```
    variables3[[j]]$modelcode <- paste0(variables3[[j]]$inclusion, collapse = "")
    coefficients1[[j]] <- map_df(coef(leaps[[j]],maxadjr2[[j]],vcov=FALSE),
~as.data.frame(.x), .id="term")
    coefficients1[[j]]$estimate <- coefficients1[[j]]$`coef(leaps[[j]], maxadjr2[[j]], vcov =
FALSE)`
    coefficients1[[j]]$estimate <- coefficients1[[j]]$.x
    coefficients2[[j]] <- coefficients1[[j]][-c(2)]

    bestmodel[[j]] <- merge(variables3[[j]], coefficients2[[j]], by = "term")
    modelsonly1[[j]] <- bestmodel[[j]][!duplicated(bestmodel[[j]]$modelcode),]

    print(bootstrapset <- Sys.time()-t3)

  }
  modelsonly2[[k]] <- map_df(modelsonly1, ~as.data.frame(.x), .id="bootset1")
  modelsonly3[[k]] <- count(modelsonly2[[k]], modelcode)
  freqmodel[[k]] <- which.max(modelsonly3[[k]]$n)
  modelsonly3[[k]]$bestfit <- modelsonly3[[k]]$modelcode[freqmodel[[k]]]
  modelsonly4 <- map_df(modelsonly3, ~as.data.frame(.x), .id="NumberID")


  judge_bestmodels[[k]] <- map_df(bestmodel, ~as.data.frame(.x), .id="bootset")
  bestfit[[k]] <- merge(judge_bestmodels[[k]], modelsonly3[[k]], by = "modelcode")
  bestfit[[k]]$bestmodel <- ifelse(bestfit[[k]]$modelcode == bestfit[[k]]$bestfit, TRUE,
FALSE)
  only_bestmodels[[k]] <- subset(bestfit[[k]], bestmodel == TRUE)
  variable_importance[[k]] <- count(judge_bestmodels[[k]], term)
  variable_importance[[k]]$frequency <- variable_importance[[k]]$n
  variable_importance[[k]]$probability <- (variable_importance[[k]]$frequency)/(bootset)
  print(judgeanalysis <- Sys.time()-t2)

  complete_bootsets <- map_df(bestfit, ~as.data.frame(.x), .id="NumberID")
  NuValcomplete_bootsets <- complete_bootsets[
    with(complete_bootsets, order(NumberID, bootset, term)),
    ]

  final_bestmodels <- map_df(only_bestmodels, ~as.data.frame(.x), .id="NumberID")
  NuValfinal_bestmodels <- as.data.frame(final_bestmodels[
    with(final_bestmodels, order(NumberID, bootset, term)),
    ])


  NuValpredictor_importance <- map_df(variable_importance, ~as.data.frame(.x),
.id="NumberID")

}
```

```
print(totalanalysis <- (Sys.time()-t1))

#####################################################################
##SAVING RESULTS Personal PC: PC1##
write.csv(NuValpredictor_importance, file = "C:/Users/Kristina/Documents/NuVal_Only
WITH BETA NuValpredictor_importance.csv")
write.csv(NuValfinal_bestmodels, file = "C:/Users/Kristina/Documents/NuVal_Only
WITH BETA NuValfinal_bestmodels.csv")
write.csv(NuValcomplete_bootsets, file = "C:/Users/Kristina/Documents/NuVal_Only
WITH BETA NuValcomplete_bootsets.csv")

best_models <- list()
bootset_model <- list()
All_Subsets_Model <- list()
term_model <- list()
weights <- list()
for(k in start:judges){
  t2 <- Sys.time()
  best_models[[k]] <- as.data.frame(subset(NuValfinal_bestmodels, NumberID == k))
  bootset_model[[k]] <- as.data.frame(subset(best_models[[k]], bootset == min(bootset)))

  All_Subsets_Model[[k]] <-
bootset_model[[k]][!duplicated(bootset_model[[k]]$NumberID),]

  All_Subsets_Model[[k]]$model_var <- sum(bootset_model[[k]]$inclusion) - 1

  All_Subsets_Model[[k]]$mean_adjR2 <- mean(best_models[[k]]$adj.r.squared)

  term_model[[k]] <- split(best_models[[k]], best_models[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  All_Subsets_Model[[k]]$intercept_mean <- sum(ifelse(weights[[k]]$Predictor ==
'(Intercept)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$cal_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Calories)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$tfat_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Total_Fatg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$sfat_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Saturated_Fatg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$pfat_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Poly_Fatg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$mfat_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Mono_Fatg)', weights[[k]]$Mean, 0))
```

```
  All_Subsets_Model[[k]]$sod_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Sodiummg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$potas_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Potassiummg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$dfiber_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(DietaryFiberg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$sugar_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Sugars_g)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$tcarb_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(TotalCarbsg)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$protein_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(Proteing)', weights[[k]]$Mean, 0))
  All_Subsets_Model[[k]]$vitmin_nuval_mean <- sum(ifelse(weights[[k]]$Predictor ==
'scale(VitsMinerals_NuVal)', weights[[k]]$Mean, 0))

}

All_Subsets_Predictors_Included <- map_df(All_Subsets_Model, ~as.data.frame(.x),
.id="NumberID")

predictors <- c("NumberID", "model_var", "mean_adjR2", "intercept_mean",
"cal_mean", "tfat_mean", "sfat_mean", "pfat_mean", "mfat_mean", "sod_mean",
"potas_mean", "dfiber_mean", "sugar_mean", "tcarb_mean", "protein_mean",
"vitmin_nuval_mean")

NuVal_All_Subsets_Model <-
All_Subsets_Predictors_Included[complete.cases(All_Subsets_Predictors_Included[,
predictors]), predictors]

write.csv(NuVal_All_Subsets_Model, file = "C:/Users/Kristina/Documents/NuVal_Only
WITH BETA NuValAll_Subsets_Model.csv")
```

Random Forest Analysis

```
#install.packages("VSURF")
#install.packages("randomForest")
#install.packages("party")
#install.packages("readxl")
#install.packages("purrr")
#install.packages("data.table")
#install.packages("dplyr")

rm(list = ls())

library(VSURF)
library(randomForest)
library(party)
```

```
library(readxl)
library(purrr)
library(data.table)
library(dplyr)

CerealLevel <- read_excel("~/Ohio University/Carter.FOP Study.Cereal Level Dataset
Abbreviated.xlsx")

SingleList <- (subset(CerealLevel, NumberID == 1))
NuValdfVar <- c("V1", "Cereal", "NuValScore", "RecordID", "NumberID", "Condition",
"Setting", "UPC", "CerealNumber", "Calories", "Total_Fatg", "Saturated_Fatg",
"Poly_Fatg", "Mono_Fatg", "Sodiummg", "Potassiummg", "TotalCarbsg",
"DietaryFiberg", "Sugars_g", "Proteing", "VitsMinerals_NuVal")

NuValComplete <- SingleList[complete.cases(SingleList[, NuValdfVar]), NuValdfVar]

rm(list = ls()[!ls()%in%c("NuValComplete")])

datalist <- list()
sample <- list()
model_sample <- list()
validate_sample <- list()
VSURF.output <- list()
predictor_num <- list()
n <- list()
actual <- list()
predicted <- list()
R2 <- list()
adjR2 <- list()
predictors <- list()
thres_predictors <- list()
var_imp <- list()

correlate <- list()
R2_predict <- list()
MSE_predict <- list()
PRESS_predict <- list()


############RANDOM FOREST ENVIRONMENT ANALYSIS#################
start <- 1
judges <- 1
###############################################

############RUN BELOW AS SINGLE BLOCK#########################
set.seed(711)
```

```
t1 <- Sys.time()
for(k in start:judges){
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(NuValComplete, NumberID == k))

  VSURF.output[[k]] <- VSURF(datalist[[k]][,10:21], datalist[[k]]$NuValScore, ntree =
2000,
                             mtry = max(floor(ncol(datalist[[k]][,10:21])/3), 1),
                             nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1)


  n[[k]] <- nrow(datalist[[k]]) #will need to change this to reflect actual number of
judgments among participants
  predictor_num[[k]] <- nrow(as.data.frame(VSURF.output[[k]]$varselect.interp))

  datalist[[k]]$actual <- datalist[[k]]$NuValScore
  datalist[[k]]$predicted <- predict(VSURF.output[[k]], datalist[[k]], step = c("interp"))
  R2[[k]] <- 1 - (sum((datalist[[k]]$actual-
datalist[[k]]$predicted)^2)/sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))
  adjR2[[k]]  <- 1-((sum((datalist[[k]]$actual-datalist[[k]]$predicted)^2))/(n[[k]]-
predictor_num[[k]]-1))/((sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))/n[[k]]-
1)

  predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.interp)

  predictors[[k]]$predictor_num <- predictors[[k]]$`VSURF.output[[k]]$varselect.interp`

  predictors[[k]]$term <- colnames(datalist[[k]])[9+predictors[[k]]$predictor_num]
  predictors[[k]] <- subset(predictors[[k]], select = -1)
  predictors[[k]]$rank <- row.names(predictors[[k]])

  included_predictors <- map_df(predictors, ~as.data.frame(.x), .id="NumberID")

  thres_predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.thres)
  thres_predictors[[k]]$predictor_num <-
thres_predictors[[k]]$`VSURF.output[[k]]$varselect.thres`
  thres_predictors[[k]]$term <-
colnames(datalist[[k]])[9+thres_predictors[[k]]$predictor_num]
  thres_predictors[[k]] <- select(thres_predictors[[k]], -1)
  thres_predictors[[k]]$var_importance <- VSURF.output[[k]]$imp.varselect.thres
  thres_predictors[[k]]$rank <- row.names(thres_predictors[[k]])
  threshold_step <- map_df(thres_predictors, ~as.data.frame(.x), .id="NumberID")

  var_imp[[k]] <- as.data.frame(VSURF.output[[k]]$imp.mean.dec.ind)
  var_imp[[k]]$predictor_num <- var_imp[[k]]$`VSURF.output[[k]]$imp.mean.dec.ind`
  var_imp[[k]]$term <- colnames(datalist[[k]])[9+var_imp[[k]]$predictor_num]
```

```
  var_imp[[k]] <- select(var_imp[[k]], -1)
  var_imp[[k]]$var_importance <- VSURF.output[[k]]$imp.mean.dec
  var_imp[[k]]$rank <- row.names(var_imp[[k]])

  var_importance <- map_df(var_imp, ~as.data.frame(.x), .id="NumberID")

  print(judgeanalysis <- Sys.time()-t2)
}
print(totalanalysis <- (Sys.time()-t1))

##SAVING ON PERSONAL PC: PC1###.
write.csv(included_predictors, file = "C:/Users/Kristina/Documents/NuVal_Only
Random Forest Final Predictors.csv")
write.csv(threshold_step, file = "C:/Users/Kristina/Documents/NuVal_Only Random
Forest Threshold Predictors.csv")
write.csv(var_importance, file = "C:/Users/Kristina/Documents/NuVal_Only Analysis
Random Forest Variable Importance.csv")

################################################
#Model Statistics#
R2_model <- map_df(R2, ~as.data.frame(.x), .id = "NumberID")
R2_model$R2_model <- R2_model$.x
R2_model <- select(R2_model, -2)

adjR2_model <- map_df(adjR2, ~as.data.frame(.x), .id = "NumberID")
adjR2_model$adjrR2 <- adjR2_model$.x
adjR2_model <- select(adjR2_model, -2)

### Merging Model Stats ###
R_model_stats <- merge(R2_model, adjR2_model, by = c('NumberID'), all.x = TRUE)

##SAVING ON PERSONAL PC: PC1###.
write.csv(R_model_stats, file = "C:/Users/Kristina/Documents/NuVal_Only Random
Forest Model Stats.csv")
```

Study One Analyses R Code

Splitting Observed Data into Modeling and Validating Sub-Samples

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("olsrr")
```

```
rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(data.table)

#Data from Personal PC: PC1
CerealLevel <- read_excel("~/Ohio University/Carter.FOP Study.Cereal Level Dataset
Abbreviated.xlsx")

dfVar <- c("V1", "Cereal", "Judgment", "RecordID", "NumberID", "Condition",
"Setting", "UPC", "CerealNumber", "Calories", "Calories_Fat", "Total_Fatg",
"Saturated_Fatg", "Trans_Fatg", "Poly_Fatg", "Mono_Fatg", "Cholesterolmg",
"Sodiummg", "Potassiummg", "TotalCarbsg", "DietaryFiberg", "Soluble_Fiberg",
"Insoluble_Fiberg", "Sugars_g", "Other_Carbsg", "Proteing", "VitsMinerals")

CLComplete <- CerealLevel[complete.cases(CerealLevel[, dfVar]), dfVar]
CLComplete <- as.data.frame(CLComplete[with(CLComplete, order(NumberID,
Judgment)),])

rm(list = ls()[!ls()%in%c("CLComplete")])

datalist <- list()
sample <- list()
model_sample <- list()
validate_sample <- list()

t1 <- Sys.time()
set.seed(71118)
for(k in 1:298){
  #set.seed(711)
  datalist[[k]] <- as.data.frame(subset(CLComplete, NumberID == k))
  datalist[[k]]
  sample[[k]] <- sample.int(n = nrow(datalist[[k]]), size = floor(40), replace = F)
  model_sample[[k]] <- datalist[[k]][sample[[k]], ]
  validate_sample[[k]] <- datalist[[k]][-sample[[k]], ]

  model_samples <- map_df(model_sample, ~as.data.frame(.x), .id="NumberID")
  model_samples <- as.data.frame(model_samples[
    with(model_samples, order(NumberID, Cereal)),
```

```
    ])

  validate_samples <- map_df(validate_sample, ~as.data.frame(.x), .id="NumberID")
  validate_samples <- as.data.frame(validate_samples[
    with(validate_samples, order(NumberID, Cereal)),
    ])
}
print(totalanalysis <- (Sys.time()-t1))

write.csv(model_samples, file = "C:/Users/Kristina/Documents/CV Model Samples.csv")
write.csv(validate_samples, file = "C:/Users/Kristina/Documents/CV Validate
Samples.csv")
```

Stepwise Regression Analysis

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("lm.beta")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(lm.beta)


ModelSample<- read.csv(file="C:/Users/Kristina/Documents/CV Model Samples.csv",
header=TRUE, sep=",")
ModelSample <- as.data.frame(ModelSample[with(ModelSample, order(NumberID,
Judgment)),])

ValidateSample<- read.csv(file="C:/Users/Kristina/Documents/CV Validate
Samples.csv", header=TRUE, sep=",")
ValidateSample <- as.data.frame(ValidateSample[with(ValidateSample,
order(NumberID, Judgment)),])

full_model <- list()
```

```
null_model <- list()
stepwise <- list()
summaries <- list()
list_names <- list()
df_coefficients <- list()
df_betas <- list()
df_fstat <- list()
scale_full_model <- list()
scale_null_model <- list()
scale_stepwise <- list()
sample <- list()
model_sample <- list()
validate_sample <- list()
predict <- list()
correlate <- list()
R2_predict <- list()
MSE_predict <- list()
PRESS_predict <- list()


#########CROSS VALIDATED STEPWISE ANALYSIS#################
start <- 1
judges <- 298  #number of judges to include

##### RUN BELOW TILL BREAK AS SINGLE BLOCK#######################
set.seed(101518)
t1 <- Sys.time()
for(i in start:judges){
 t2 <- Sys.time()
  index <- start - 1

  model_sample[[i]] <- as.data.frame(subset(ModelSample, NumberID == i))
  validate_sample[[i]] <- as.data.frame(subset(ValidateSample, NumberID == i))

  full_model[[i]] <- lm(Judgment ~ Calories + Calories_Fat + Total_Fatg +
Saturated_Fatg + Poly_Fatg + Mono_Fatg + Sodiummg + Potassiummg +
DietaryFiberg + Soluble_Fiberg + Insoluble_Fiberg + Sugars_g +
TotalCarbsg + Other_Carbsg + Proteing + VitsMinerals,
                        data = model_sample[[i]])

  null_model[[i]] <- lm(Judgment ~ 1, data = model_sample[[i]])

  stepwise[[i]] <- step(null_model[[i]], scope = list(upper = full_model[[i]]))

scale_full_model[[i]] <- lm(scale(Judgment) ~ scale(Calories) + scale(Calories_Fat) +
scale(Total_Fatg) + scale(Saturated_Fatg) +  scale(Poly_Fatg) + scale(Mono_Fatg) +
```

```
scale(Sodiummg) + scale(Potassiummg) + scale(DietaryFiberg) + scale(Soluble_Fiberg)
+ scale(Insoluble_Fiberg) + scale(Sugars_g) + scale(TotalCarbsg) + scale(Other_Carbsg)
+ scale(Proteing) + scale(VitsMinerals),
                        data = model_sample[[i]])

scale_null_model[[i]] <- lm(scale(Judgment) ~ 1, data = model_sample[[i]])

scale_stepwise[[i]] <- step(scale_null_model[[i]], scope = list(upper =
scale_full_model[[i]]))

  summaries[[i]] <- summary(stepwise[[i]])
  list_names[[i]] <- paste0("ID_", unique(model_sample[[i]]$NumberID))

  require(broom)
  df_coefficients[[i]] <- tidy(stepwise[[i]])
  df_betas[[i]] <- tidy(scale_stepwise[[i]])
  df_fstat[[i]] <- as.data.frame((summaries[[i]][["r.squared"]]))
  colnames(df_fstat[[i]]) <- "r.squared"
  df_fstat[[i]]$value <- ifelse(is_empty(summaries[[i]]$fstatistic[[1]]) == TRUE, 0,
summaries[[i]]$fstatistic[[1]])
  df_fstat[[i]]$numdf <- ifelse(is_empty(summaries[[i]]$fstatistic[[2]]) == TRUE, 0,
summaries[[i]]$fstatistic[[2]])
  df_fstat[[i]]$dendf <- ifelse(is_empty(summaries[[i]]$fstatistic[[3]]) == TRUE, 0,
summaries[[i]]$fstatistic[[3]])


  validate_sample[[i]]$predict <- predict(stepwise[[i]], validate_sample[[i]], se.fit =
FALSE,
                      scale = NULL, df = Inf,
                      interval = c("none", "confidence", "prediction"),
                      level = 0.95, type = c("response", "terms"),
                      terms = NULL, na.action = na.pass,
                      pred.var = res.var/weights, weights = 1)

  correlate[[i]] <- cor(x = validate_sample[[i]]$Judgment,
                    y = validate_sample[[i]]$predict, method="pearson")

  R2_predict[[i]] <- (correlate[[i]]^2)

  MSE_predict[[i]] <- ((sum((validate_sample[[i]]$Judgment -
validate_sample[[i]]$predict)^2))/
                    (nrow(validate_sample[[i]])))

  PRESS_predict[[i]]<- (sum((validate_sample[[i]]$Judgment -
validate_sample[[i]]$predict)^2))
```

```
  test_sample <- map_df(validate_sample, ~as.data.frame(.x), .id="NumberID")
  test_sample$NumberID <- as.numeric(test_sample$NumberID) + index

 print(judgeanalysis <- Sys.time()-t2)
}
print(totalanalysis <- (Sys.time()-t1))

##SAVING ON PERSONAL PC: PC1###.
write.csv(test_sample, file = paste0("C:/Users/Kristina/Documents/IDs", start, "-",
judges, " CV Final Stepwise Validate Sample.csv", sep = ""))

#############################################################################

### Cross-Validated R ####
Rpredict <- map_df(correlate, ~as.data.frame(.x), .id="NumberID")
Rpredict$R_test<- Rpredict$.x
Rpredict <- select(Rpredict, -2)

### Cross-Validated R2 ###
R2predict <- map_df(R2_predict, ~as.data.frame(.x), .id="NumberID")
R2predict$R2_test<- R2predict$.x
R2predict <- select(R2predict, -2)

### Cross-Validated MSE ###
MSEpredict <- map_df(MSE_predict, ~as.data.frame(.x), .id="NumberID")
MSEpredict$MSE_test<- MSEpredict$.x
MSEpredict <- select(MSEpredict, -2)

### Cross-Validated PRESS ###
PRESSpredict <- map_df(PRESS_predict, ~as.data.frame(.x), .id="NumberID")
PRESSpredict$PRESS_test<- PRESSpredict$.x
PRESSpredict <- select(PRESSpredict, -2)

### Merging Cross-Validated Stats ###
R_test_stats <- merge(Rpredict, R2predict, by = c('NumberID'), all.x = TRUE)
Error_test_stats <- merge(MSEpredict, PRESSpredict, by = c('NumberID'), all.x =
TRUE)
cv_stats <- merge(R_test_stats, Error_test_stats, by = c('NumberID'), all.x = TRUE)

### Adjusted R^2 ###
model_stats <- rbind(lapply(summaries, `[`, 9))
model <- map_df(model_stats, ~as.data.frame(.x), .id="NumberID")

### F-Statistics ###
fstats <- bind_rows(df_fstat, .id = 'NumberID')
fstats$fstat <- fstats$value
```

```
fstatVar <- c("NumberID", "fstat", "numdf", "dendf")
fstatistics <- fstats[complete.cases(fstats[, fstatVar]), fstatVar]

### Merging extracted model stats together ###
extracted_stats <- merge(model, fstatistics, by = c('NumberID'), all.x = TRUE)

### Merging extracted and test model stats ###
stepwise_model <- merge(cv_stats, extracted_stats, by = c('NumberID'), all.x = TRUE)

### Computing p-values, assigning binary significance values ###
stepwise_model$pvalue<- pf(stepwise_model$fstat, stepwise_model$numdf,
stepwise_model$dendf, lower.tail = FALSE)
stepwise_model$modelsig <- ifelse(stepwise_model$pvalue < .05, 1, 0)


stepwise_model$NumberID <- as.numeric(stepwise_model$NumberID) + index

#####################################################################
### Creating dataframe for unstandardized coefficient estimates ###
coefficients <- bind_rows(df_coefficients, .id = 'NumberID')

### Creating mergeable dataframe for standardized coefficient estimates ###
betas_mid <- bind_rows(df_betas, .id = 'NumberID')
betas_mid$term2 <- gsub("scale\\(", "", betas_mid$term)
betas_mid$term3 <- gsub("g[)]", "g", betas_mid$term2)
betas_mid$term4 <- gsub("s[)]", "s", betas_mid$term3)
betas_mid$term5 <- gsub("t[)]", "t", betas_mid$term4)
betas_mid$term <- gsub("[(]Intercept", "(Intercept)", betas_mid$term5)

betas_mid$beta <- betas_mid$estimate
betaVar <- c("NumberID", "term", "beta")
betas <- betas_mid[complete.cases(betas_mid[, betaVar]), betaVar]
betas$beta_value <- ifelse(betas$term == '(Intercept)', NA, betas$beta)

### Merging standardized coefficient estimates with larger coefficient dataframe ###
stepwise_coeff <- merge(coefficients, betas, by = c('NumberID', 'term'))

### Assigning meta values ###
#Significance values#
stepwise_coeff$coeffsig <- ifelse(stepwise_coeff$p.value < .05, 1, 0)
#Positive/negative betas
stepwise_coeff$coeffdirection <- ifelse(stepwise_coeff$beta > 0, 'positive', 'negative')
stepwise_coeff$NumberID <- as.numeric(stepwise_coeff$NumberID) + index

#Ranking values#
coeff_data <- list()
```

```
for(i in start:judges){
  coeff_data[[i]] <- as.data.frame(subset(stepwise_coeff, NumberID == i))
  coeff_data[[i]]$rank <- rank(-abs(coeff_data[[i]]$beta_value), na.last = FALSE,
ties.method = "average")
  coeff_data[[i]]$adjrank <- coeff_data[[i]]$rank - 1
  final_stepwise_coeff <- bind_rows(coeff_data)
}

final_model <- as.data.frame(subset(final_stepwise_coeff))
predictors <- c("NumberID", "term")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]

predictorlist <- list()
for(i in start:judges){
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, NumberID == i))
  predictorlist[[i]]$Count <- (nrow(predictorlist[[i]]))-1
}

final_model_predictors <- map_df(predictorlist, ~as.data.frame(.x), .id="NumberID")

#Adding Predictor Count to File#
final_model <- as.data.frame(subset(final_stepwise_coeff))
predictors <- c("NumberID", "term")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]

stepmodel <- list()
predictorlist <- list()
for(i in 1:judges){
  stepmodel[[i]] <- as.data.frame(subset(stepwise_model, NumberID == i))
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, NumberID == i))
  predictorlist[[i]]$Count <- (nrow(predictorlist[[i]]))-1
  stepmodel[[i]]$model_var <- (nrow(predictorlist[[i]]))-1
}

stepwise_model <- map_df(stepmodel, ~as.data.frame(.x), .id="NumberID")
final_model_predictors <- map_df(predictorlist, ~as.data.frame(.x), .id="NumberID")

#########################################################################
##SAVING ON PERSONAL PC: PC1###.
write.csv(final_stepwise_coeff, file = paste0("C:/Users/Kristina/Documents/IDs", start, "-
", judges, " CV Final Stepwise Coefficients.csv", sep = ""))
```

```
write.csv(final_model_predictors, file = paste0("C:/Users/Kristina/Documents/IDs", start,
"-", judges, " CV Final Stepwise Model Predictors.csv", sep = ""))
write.csv(stepwise_model, file = paste0("C:/Users/Kristina/Documents/IDs", start, "-",
judges, " CV Final Stepwise Model.csv", sep = ""))
```

All Subsets Analysis

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("olsrr")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(data.table)


#Data from Personal PC: PC1
CerealLevel <- read_excel("~/Ohio University/Carter.FOP Study.Cereal Level Dataset
Abbreviated.xlsx")

ModelSample<- read.csv(file="C:/Users/Kristina/Documents/Model Samples2.csv",
header=TRUE, sep=",")
ModelSample <- as.data.frame(ModelSample[with(ModelSample, order(NumberID,
Judgment)),])

ValidateSample<- read.csv(file="C:/Users/Kristina/Documents/Validate Samples.csv",
header=TRUE, sep=",")
ValidateSample <- as.data.frame(ValidateSample[with(ValidateSample,
order(NumberID, Judgment)),])

#################ALL SUBSETS ANALYSIS FUNCTION#######################
all_subsets <- function(last, group, end, bootset){

  rm(list = ls()[!ls()%in%c("CLComplete", "all_subsets", "tfunc", "last", "group", "end",
"bootset")])
```

```
  t1 <- Sys.time()

  sample <- list()
  model_sample <- list()
  validate_sample <- list()

  resamples <- list()
  judgment_set <- list()
  leaps <- list()
  summary <- list()
  maxadjr2 <- list()
  variables1 <- list()
  variables2 <- list()
  variables3 <- list()
  coefficients1 <- list()
  coefficients2 <- list()
  bestmodel <- list()
  judge_bestmodels <- list()
  count <- list()

  bestfit <- list()
  only_bestmodels <- list()

  modelsonly1 <- list()
  modelsonly2 <- list()
  modelsonly3 <- list()
  modelsonly4 <- list()
  freqmodel <- list()
  variable_importance <- list()

start <- ifelse((1 + last <= end), (1 + last), end)
judges <- ifelse((last + group <= end), (last + group), end)

#set.seed(711)
for(k in start:judges){
  #set.seed(711)
  index <- start - 1
  t2 <- Sys.time()
  #datalist[[k]] <- as.data.frame(subset(CLComplete, NumberID == k))

  model_sample[[k]] <- as.data.frame(subset(ModelSample, NumberID == k))
  validate_sample[[k]] <- as.data.frame(subset(ValidateSample, NumberID == k))

  resamples[[k]] <- lapply(1:bootset, function(i) sample_n(model_sample[[k]], 40, replace
= T))
```

```
  for(j in 1:bootset){
    #set.seed(711)
    t3 <- Sys.time()
    judgment_set[[j]] <- resamples[[k]][[j]]
    leaps[[j]] <-
      regsubsets(Judgment ~ Calories + Calories_Fat + Total_Fatg + Saturated_Fatg +
                   Poly_Fatg + Mono_Fatg + Sodiummg + Potassiummg +
                   DietaryFiberg + Soluble_Fiberg + Insoluble_Fiberg + Sugars_g +
                   TotalCarbsg + Other_Carbsg + Proteing + VitsMinerals,
                 data = judgment_set[[j]],
                 nbest = 1,       # 1 best model for each number of predictors
                 nvmax = NULL,    # NULL for no limit on number of variables
                 force.in = NULL, force.out = NULL,
                 method = "exhaustive")
    summary[[j]] <- summary(leaps[[j]])
    maxadjr2[[j]] <- which.max(summary[[j]]$adjr2)
    variables1[[j]] <- map_df(summary[[j]]$which[maxadjr2[[j]],], ~as.data.frame(.x),
.id="term")
    variables1[[j]]$included <- variables1[[j]]$.x
    variables2[[j]] <- variables1[[j]][-c(2)]
    variables3[[j]] <- as.data.frame(subset(variables2[[j]]))
    variables3[[j]]$adj.r.squared <- summary[[j]]$adjr2[maxadjr2[[j]]]
    variables3[[j]]$inclusion <- ifelse(variables3[[j]]$included == TRUE, 1, 0)
    variables3[[j]]$modelcode <- paste0(variables3[[j]]$inclusion, collapse = "")
    coefficients1[[j]] <- map_df(coef(leaps[[j]],maxadjr2[[j]],vcov=FALSE),
~as.data.frame(.x), .id="term")
    coefficients1[[j]]$estimate <- coefficients1[[j]]$`coef(leaps[[j]], maxadjr2[[j]], vcov =
FALSE)`
    coefficients1[[j]]$estimate <- coefficients1[[j]]$.x
    coefficients2[[j]] <- coefficients1[[j]][-c(2)]

    bestmodel[[j]] <- merge(variables3[[j]], coefficients2[[j]], by = "term")
    modelsonly1[[j]] <- bestmodel[[j]][!duplicated(bestmodel[[j]]$modelcode),]

    print(bootstrapset <- Sys.time()-t3)

  }

  modelsonly2[[k]] <- map_df(modelsonly1, ~as.data.frame(.x), .id="bootset1")
  modelsonly3[[k]] <- count(modelsonly2[[k]], modelcode)
  freqmodel[[k]] <- which.max(modelsonly3[[k]]$n)
  modelsonly3[[k]]$bestfit <- modelsonly3[[k]]$modelcode[freqmodel[[k]]]
  modelsonly4 <- map_df(modelsonly3, ~as.data.frame(.x), .id="NumberID")
```

```
  judge_bestmodels[[k]] <- map_df(bestmodel, ~as.data.frame(.x), .id="bootset")
  bestfit[[k]] <- merge(judge_bestmodels[[k]], modelsonly3[[k]], by = "modelcode")
  bestfit[[k]]$bestmodel <- ifelse(bestfit[[k]]$modelcode == bestfit[[k]]$bestfit, TRUE,
FALSE)
  only_bestmodels[[k]] <- subset(bestfit[[k]], bestmodel == TRUE)
  variable_importance[[k]] <- count(judge_bestmodels[[k]], term)
  variable_importance[[k]]$frequency <- variable_importance[[k]]$n
  variable_importance[[k]]$probability <- (variable_importance[[k]]$frequency)/(bootset)
  print(judgeanalysis <- Sys.time()-t2)

  #Commented code here can enable complete boosets files to be saved
  #complete_bootsets <- map_df(bestfit, ~as.data.frame(.x), .id="NumberID")
  #complete_bootsets <- complete_bootsets[
   # with(complete_bootsets, order(NumberID, bootset, term)),
    #]
  #complete_bootsets$NumberID <- as.numeric(complete_bootsets$NumberID) + index

  final_bestmodels <- map_df(only_bestmodels, ~as.data.frame(.x), .id="NumberID")
  final_bestmodels <- as.data.frame(final_bestmodels[
    with(final_bestmodels, order(NumberID, bootset, term)),
    ])
  final_bestmodels$NumberID <- as.numeric(final_bestmodels$NumberID) + index

    predictor_importance <- map_df(variable_importance, ~as.data.frame(.x),
.id="NumberID")
  predictor_importance$NumberID <- as.numeric(predictor_importance$NumberID) +
index
}
print(totalanalysis <- (Sys.time()-t1))


###SAVE RELEVANT FILES
##SAVING ON PERSONAL PC: PC1###.
write.csv(final_bestmodels, file = paste0("C:/Users/Kristina/Documents/IDs", start, "-",
judges, " All Subsets final_bootsets.csv", sep = ""))
write.csv(predictor_importance, file = paste0("C:/Users/Kristina/Documents/IDs", start,
"-", judges, " All Subsets predictor_importance.csv", sep = ""))

modelVar <- c("NumberID", "term", "estimate")

model_estimates <- final_bestmodels[complete.cases(final_bestmodels[, modelVar]),
modelVar]

person_model <- list()
term_model <- list()
weights <- list()
```

```
intercept <- list()
cal_b <- list()
calf_b <- list()
tfat_b <- list()
sfat_b <- list()
pfat_b <- list()
mfat_b <- list()
sod_b <- list()
potas_b <- list()
dfiber_b <- list()
sfiber_b <- list()
ifiber_b <- list()
sugar_b <- list()
tcarb_b <- list()
ocarb_b <- list()
protein_b <- list()
vitmin_b <- list()

correlate <- list()
R2_predict <- list()
MSE_predict <- list()
PRESS_predict <- list()
cross_validation <- list()


t1a <- Sys.time()
for(k in start:judges){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, NumberID == k))
  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  intercept[[k]] <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)', weights[[k]]$Mean,
0))
  cal_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Calories', weights[[k]]$Mean, 0))
  calf_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Calories_Fat', weights[[k]]$Mean,
0))
  tfat_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Total_Fatg', weights[[k]]$Mean, 0))
  sfat_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Saturated_Fatg',
weights[[k]]$Mean, 0))
  pfat_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Poly_Fatg', weights[[k]]$Mean, 0))
  mfat_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Mono_Fatg', weights[[k]]$Mean,
0))
```

```
  sod_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Sodiummg', weights[[k]]$Mean, 0))
  potas_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Potassiummg',
weights[[k]]$Mean, 0))
  dfiber_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'DietaryFiberg',
weights[[k]]$Mean, 0))
  sfiber_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Soluble_Fiberg',
weights[[k]]$Mean, 0))
  ifiber_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Insoluble_Fiberg',
weights[[k]]$Mean, 0))
  sugar_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Sugars_g', weights[[k]]$Mean,
0))
  tcarb_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'TotalCarbsg', weights[[k]]$Mean,
0))
  ocarb_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Other_Carbsg',
weights[[k]]$Mean, 0))
  protein_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'Proteing', weights[[k]]$Mean,
0))
  vitmin_b[[k]] <- sum(ifelse(weights[[k]]$Predictor == 'VitsMinerals',
weights[[k]]$Mean, 0))

   validate_sample[[k]]$predict <- intercept[[k]] +
cal_b[[k]]*validate_sample[[k]]$Calories +
   calf_b[[k]]*validate_sample[[k]]$Calories_Fat +
tfat_b[[k]]*validate_sample[[k]]$Total_Fatg +
   sfat_b[[k]]*validate_sample[[k]]$Saturated_Fatg +
pfat_b[[k]]*validate_sample[[k]]$Poly_Fatg +
   mfat_b[[k]]*validate_sample[[k]]$Mono_Fatg +
sod_b[[k]]*validate_sample[[k]]$Sodiummg +
   potas_b[[k]]*validate_sample[[k]]$Potassiummg +
dfiber_b[[k]]*validate_sample[[k]]$DietaryFiberg +
   sfiber_b[[k]]*validate_sample[[k]]$Soluble_Fiberg +
ifiber_b[[k]]*validate_sample[[k]]$Insoluble_Fiberg +
   sugar_b[[k]]*validate_sample[[k]]$Sugars_g +
tcarb_b[[k]]*validate_sample[[k]]$TotalCarbsg +
   ocarb_b[[k]]*validate_sample[[k]]$Other_Carbsg +
protein_b[[k]]*validate_sample[[k]]$Proteing +
   vitmin_b[[k]]*validate_sample[[k]]$VitsMinerals

  correlate[[k]] <- cor(x = validate_sample[[k]]$Judgment,
                        y = validate_sample[[k]]$predict, method = "pearson")

  R2_predict[[k]] <- (correlate[[k]]^2)

  MSE_predict[[k]] <- ((sum((validate_sample[[k]]$Judgment -
validate_sample[[k]]$predict)^2))/
                         (nrow(validate_sample[[k]])))
```

```
  PRESS_predict[[k]] <- (sum(validate_sample[[k]]$Judgment -
validate_sample[[k]]$predict)^2)

  cross_validation <- map_df(validate_sample, ~as.data.frame(.x), .id="NumberID")
  cross_validation$NumberID <- as.numeric(cross_validation$NumberID) + index
 }

#############################################################################

### Cross-Validated R ###
Rpredict <- map_df(correlate, ~as.data.frame(.x), .id = "NumberID")
Rpredict$R_test <- Rpredict$.x
Rpredict <- select(Rpredict, -2)

### Cross-Validated R2 ###
R2predict <- map_df(R2_predict, ~as.data.frame(.x), .id = "NumberID")
R2predict$R2_test <- R2predict$.x
R2predict <- select(R2predict, -2)

### Cross-Validated MSE ###
MSEpredict <- map_df(MSE_predict, ~as.data.frame(.x), .id = "NumberID")
MSEpredict$MSE_test <- MSEpredict$.x
MSEpredict <- select(MSEpredict, -2)

### Cross-Validated PRESS ###
PRESSpredict <- map_df(PRESS_predict, ~as.data.frame(.x), .id="NumberID")
PRESSpredict$PRESS_test<- PRESSpredict$.x
PRESSpredict <- select(PRESSpredict, -2)

### Merging Cross-Validated Stats ###
R_test_stats <- merge(Rpredict, R2predict, by = c('NumberID'), all.x = TRUE)
Error_test_stats <- merge(MSEpredict, PRESSpredict, by = c('NumberID'), all.x =
TRUE)
all_subsets_model <- merge(R_test_stats, Error_test_stats, by = c('NumberID'), all.x =
TRUE)
all_subsets_model$NumberID <- as.numeric(all_subsets_model$NumberID) + index



##SAVING ON PERSONAL PC: PC1###.
write.csv(cross_validation, file = paste0("C:/Users/Kristina/Documents/IDs", start, "-",
judges, " All Subsets Cross Validation.csv", sep = ""))
write.csv(all_subsets_model, file = paste0("C:/Users/Kristina/Documents/IDs", start, "-",
judges, " All Subsets Model Outcomes.csv", sep = ""))
```

```
print(cross_validated <- (Sys.time()-t1a))

}

############RUN ALL SUBSETS ANALYSIS FUNCTION#######################
set.seed(1015)
last <- seq.int(0, 300, by = 10)#by should be same size as group,indicates how many
participants to increase by at next cycle

for(a in last){
tfunc <- Sys.time()
all_subsets(last=a, group = 10, end = 298, bootset = 100000)
print(totalanalysis <- (Sys.time()-tfunc))
}
#############################################################################
```

Random Forest Analysis

```
#install.packages("VSURF")
#install.packages("randomForest")
#install.packages("party")
#install.packages("readxl")
#install.packages("purrr")
#install.packages("data.table")
#install.packages("dplyr")

rm(list = ls())

library(VSURF)
library(randomForest)
library(party)
library(readxl)
library(purrr)
library(data.table)
library(dplyr)

ModelSample<- read.csv(file="C:/Users/Kristina/Documents/CV Model Samples.csv",
header=TRUE, sep=",")
ModelSample <- as.data.frame(ModelSample[with(ModelSample, order(NumberID,
Judgment)),])

ValidateSample<- read.csv(file="C:/Users/Kristina/Documents/CV Validate
Samples.csv", header=TRUE, sep=",")
ValidateSample <- as.data.frame(ValidateSample[with(ValidateSample,
order(NumberID, Judgment)),])
```

```
ModelSample <- subset(ModelSample, select=-c(Trans_Fatg,Cholesterolmg))
########################################
sample <- list()
model_sample <- list()
validate_sample <- list()
VSURF.output <- list()
predictor_num <- list()
n <- list()
actual <- list()
predicted <- list()
R2 <- list()
adjR2 <- list()
predictors <- list()
correlate <- list()
R2_predict <- list()
MSE_predict <- list()
PRESS_predict <- list()

#############CROSS VALIDATED RANDOM FOREST ANALYSIS############
start <- 1
judges <- 298
################################################


############RUN BELOW AS SINGLE BLOCK############################
set.seed(101518)
t1 <- Sys.time()
for(k in start:judges){
  t2 <- Sys.time()

  model_sample[[k]] <- as.data.frame(subset(ModelSample, NumberID == k))
  validate_sample[[k]] <- as.data.frame(subset(ValidateSample, NumberID == k))

  VSURF.output[[k]] <- VSURF(model_sample[[k]][,12:27],
model_sample[[k]]$Judgment, ntree = 2000,
                    mtry = max(floor(ncol(model_sample[[k]][,12:27])/3), 1),
                    nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1)

  validate_sample[[k]]$predict <- predict(VSURF.output[[k]], validate_sample[[k]], step
= c("interp"))

  n[[k]] <- nrow(model_sample[[k]])
  predictor_num[[k]] <- nrow(as.data.frame(VSURF.output[[k]]$varselect.interp))

  model_sample[[k]]$actual <- model_sample[[k]]$Judgment
```

```
  model_sample[[k]]$predicted <- predict(VSURF.output[[k]], model_sample[[k]], step =
c("interp"))
  R2[[k]] <- 1 - (sum((model_sample[[k]]$actual-
model_sample[[k]]$predicted)^2)/sum((model_sample[[k]]$actual-
mean(model_sample[[k]]$actual))^2))
  adjR2[[k]]  <- 1-((sum((model_sample[[k]]$actual-
model_sample[[k]]$predicted)^2))/(n[[k]]-predictor_num[[k]]-
1))/((sum((model_sample[[k]]$actual-mean(model_sample[[k]]$actual))^2))/n[[k]]-1)

  predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.interp)

  predictors[[k]]$predictor_num <- predictors[[k]]$`VSURF.output[[k]]$varselect.interp`

  predictors[[k]]$term <-
colnames(model_sample[[k]])[11+predictors[[k]]$predictor_num]
  predictors[[k]] <- subset(predictors[[k]], select = -1)

  predictors[[k]]$rank <- row.names(predictors[[k]])
  predictors[[k]]$count <- nrow(predictors[[k]])

  included_predictors <- map_df(predictors, ~as.data.frame(.x), .id="NumberID")

  correlate[[k]] <- cor(x = validate_sample[[k]]$Judgment,
                        y = validate_sample[[k]]$predict, method = "pearson")

  R2_predict[[k]] <- (correlate[[k]]^2)

  MSE_predict[[k]] <- ((sum((validate_sample[[k]]$Judgment -
validate_sample[[k]]$predict)^2))/
                         (nrow(validate_sample[[k]])))

  PRESS_predict[[k]] <- (sum(validate_sample[[k]]$Judgment -
validate_sample[[k]]$predict)^2)

  cross_validation <- map_df(validate_sample, ~as.data.frame(.x), .id="NumberID")

  print(judgeanalysis <- Sys.time()-t2)
}
print(totalanalysis <- (Sys.time()-t1))

##SAVING ON PERSONAL PC: PC1###.
write.csv(included_predictors, file = "C:/Users/Kristina/Documents/Random Forest Final
Predictors.csv")
write.csv(cross_validation, file = "C:/Users/Kristina/Documents/Random Forest Validate
Sample.csv")
```

```
################################################
### Model Statistics ###
R2_model <- map_df(R2, ~as.data.frame(.x), .id = "NumberID")
R2_model$R2_model <- R2_model$.x
R2_model <- select(R2_model, -2)

adjR2_model <- map_df(adjR2, ~as.data.frame(.x), .id = "NumberID")
adjR2_model$adjrR2 <- adjR2_model$.x
adjR2_model <- select(adjR2_model, -2)

################################################
### Cross-Validated R ###
Rpredict <- map_df(correlate, ~as.data.frame(.x), .id = "NumberID")
Rpredict$R_test <- Rpredict$.x
Rpredict <- select(Rpredict, -2)

### Cross-Validated R2 ###
R2predict <- map_df(R2_predict, ~as.data.frame(.x), .id = "NumberID")
R2predict$R2_test <- R2predict$.x
R2predict <- select(R2predict, -2)

### Cross-Validated MSE ###
MSEpredict <- map_df(MSE_predict, ~as.data.frame(.x), .id = "NumberID")
MSEpredict$MSE_test <- MSEpredict$.x
MSEpredict <- select(MSEpredict, -2)

### Cross-Validated PRESS ###
PRESSpredict <- map_df(PRESS_predict, ~as.data.frame(.x), .id="NumberID")
PRESSpredict$PRESS_test<- PRESSpredict$.x
PRESSpredict <- select(PRESSpredict, -2)

### Merging Model Stats & Merging Cross-Validated Stats ###
forest_model <- list(R2_model, adjR2_model, Rpredict, R2predict, MSEpredict,
PRESSpredict) %>%
  reduce(left_join, by = "NumberID")

#Adding Variable Count to File#
final_model <- as.data.frame(subset(included_predictors))
predictors <- c("NumberID", "term", "rank")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]

predictorlist <- list()
statslist <- list()
for(i in start:judges){
```

```
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, NumberID == i))
  statslist[[i]] <- as.data.frame(subset(forest_model, NumberID == i))

  statslist[[i]]$model_var <- nrow(predictorlist[[i]])#CHANGES MADE HERE
}

forest_model <- map_df(statslist, ~as.data.frame(.x), .id="NumberID")

##SAVING ON PERSONAL PC: PC1###.
write.csv(forest_model, file = "C:/Users/Kristina/Documents/Random Forest Model
Stats.csv")
```

Multilevel Model Bayesian Analysis

```
############################################################
#PLEASE NOTE: TO RUN brms YOU WILL NEED A C++ COMPILER; THE Rtools
PROGRAM COMES WITH A C++ COMPILER
#AND IS AVAILABLE AT https://cran.r-project.org/bin/windows/Rtools/
#Rtools NEEDS TO BE INSTALLED ACCORDING TO SPECIFICATIONS
OUTLINED HERE: https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-
Windows
#OTHERWISE THE FOLLOWING CODE WILL NOT WORK

#AFTER INSTALLING Rtools, CODE FOR FINALIZING STAN INSTALLATION
install.packages("pkgbuild")
pkgbuild::has_build_tools(debug = TRUE)
fx <- inline::cxxfunction( signature(x = "integer", y = "numeric" ) , '
                    return ScalarReal( INTEGER(x)[0] * REAL(y)[0] ) ;
                    ' )
fx( 2L, 5 ) # should be 10
#If this returns anything other than 10, then go back to the previous section and install
Rtools correctly.

dotR <- file.path(Sys.getenv("HOME"), ".R")
if (!file.exists(dotR))
  dir.create(dotR)
M <- file.path(dotR, "Makevars")
if (!file.exists(M))
  file.create(M)
cat("\nCXX14FLAGS=-O3 -Wno-unused-variable -Wno-unused-function",
    "CXX14 = $(BINPREF)g++ -m$(WIN) -std=c++1y",
    "CXX11FLAGS=-O3 -Wno-unused-variable -Wno-unused-function",
    file = M, sep = "\n", append = TRUE)

remove.packages("rstan")
if (file.exists(".RData")) file.remove(".RData")
```

```
install.packages("rstan", repos = "https://cloud.r-project.org/", dependencies = TRUE)

############################################################
#install.packages("brms")

###############PREPPING DATA FILE, IF YOU HAVE UNPREPPED DATA FILES, USE THIS####################
#Loading Condition Data PC1
VS_RF <- read.csv(file="C:/Users/Kristina/Documents/Random Forest Validate Sample.csv", header=TRUE, sep=",")
dfVar <- c("NumberID", "Condition", "Setting")
Conditions <- (VS_RF[, dfVar])
Conditions = subset(Conditions, !duplicated(NumberID))

#Loading Data Files PC 1
Stepwise<- read.csv(file="C:/Users/Kristina/Documents/IDs1-298 CV Final Stepwise Model.csv", header=TRUE, sep=",")
All_Subsets<- read.csv(file="C:/Users/Kristina/Documents/All IDs All Subsets Model Outcomes.csv", header=TRUE, sep=",")
Random_Forest<- read.csv(file="C:/Users/Kristina/Documents/Results Data Updated 11.30.2018/Random Forest Model Stats.csv", header=TRUE, sep=",")

Stepwise$Step_Method <- 1
Stepwise$AS_Method <- 0
Stepwise$RF_Method <- 0
Stepwise$Method1 <- "3Stepwise"
Stepwise$Method2 <- "1Stepwise"

All_Subsets$Step_Method <- 0
All_Subsets$AS_Method <- 1
All_Subsets$RF_Method <- 0
All_Subsets$Method1 <- "2All_Subsets"
All_Subsets$Method2 <- "2All_Subsets"

Random_Forest$Step_Method <- 0
Random_Forest$AS_Method <- 0
Random_Forest$RF_Method <- 1
Random_Forest$Method1 <- "1Random_Forest"
Random_Forest$Method2 <- "3Random_Forest"

Stepwise$adjR2_model <- Stepwise$adj.r.squared
Random_Forest$adjR2_model <- Random_Forest$adjrR2
All_Subsets$adjR2_model <- All_Subsets$mean_adjR2
```

```
dfVar <- c("NumberID", "Method1", "Method2",
"Step_Method","AS_Method","RF_Method", "adjR2_model", "R_test", "R2_test",
"MSE_test", "PRESS_test", "model_var")

Stepwise1 <- (Stepwise[, dfVar])
All_Subsets1 <- (All_Subsets[, dfVar])
Random_Forest1 <- (Random_Forest[, dfVar])

All_Models <- rbind(Stepwise1, All_Subsets1, Random_Forest1)
All_Models <- merge(All_Models, Conditions, by = "NumberID")

All_Models$NFP_Only <- ifelse(All_Models$Condition == 1, 1, 0)
All_Models$FOP1 <- ifelse(All_Models$Condition == 2, 1, 0)
All_Models$FOP2 <- ifelse(All_Models$Condition == 3, 1, 0)

All_Models$Condition1a <- ifelse(All_Models$Condition == 1, "2NFP_Only",
                          ifelse(All_Models$Condition == 2, "1FOP1",
                               "3FOP2"))

All_Models$Condition1b <- ifelse(All_Models$Condition == 1, "2NFP_Only",
                          ifelse(All_Models$Condition == 2, "3FOP1",
                               "1FOP2"))

rm(list=(ls()[ls()!="All_Models"]))

write.csv(All_Models, file = "C:/Users/Kristina/Documents/Merged Methods Model
Outcomes.csv")

All_Models1 <- All_Models[complete.cases(All_Models),]

#################################################################
######IF YOU HAVE ALL MODELS FILE, LOAD FILE HERE#############
#All_Models <- read.csv(file="C:/Users/Kristina/Documents/Merged Methods Model
Outcomes.csv", header=TRUE, sep=",")

#All_Models1 <- All_Models[complete.cases(All_Models),]
#################################################################
library(rstan)
library(brms)
#########################################################
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())


set.seed(1121)
### Intercept Only ADJR2 Model: Model0a ###
```

```
t1 <- Sys.time()
fit1a <- brm(adjR2_model ~ (1|NumberID),
          data = All_Models,
          family = Beta(),
          iter = 30000,
          warmup = 2000,
          chains = 4,
          cores = 4,
          save_all_pars = TRUE,
          control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1 <- (Sys.time()-t1))

saveRDS(fit1a, file = "C:/Users/Kristina/Documents/fit1a.rds")


### Intercept Only CVR2 Model: Model0b ###
t1 <- Sys.time()
fit1b <- brm(R2_test ~ (1|NumberID),
           data = All_Models,
           family = Beta(),
           iter = 30000,
           warmup = 2000,
           chains = 4,
           cores = 4,
           save_all_pars = TRUE,
           control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1 <- (Sys.time()-t1))

saveRDS(fit1b, file = "C:/Users/Kristi/Documents/fit1b.rds")


### Intercept Only Model: Model1 ###
t1 <- Sys.time()
fit1 <- brm(cbind(adjR2_model, R2_test) ~ (1|p|NumberID),
          data = All_Models,
          family = Beta(),
          iter = 30000,
          warmup = 2000,
          chains = 4,
          cores = 4,
          save_all_pars = TRUE,
          control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1 <- (Sys.time()-t1))

saveRDS(fit1, file = "C:/Users/Kristina/Documents/fit1.rds")
```

```
rm("fit1")

### Model 2 Fixed Effects: Level-1 Predictors ###
### Random Forest Reference Group:
t2a <- Sys.time()
fit2a <- brm(cbind(adjR2_model, R2_test) ~ Method1 + (1|p|NumberID),
             data = All_Models,
             family = Beta(),
             iter = 30000,
             warmup = 2000,
             chains = 4,
             cores = 4,
             save_all_pars = TRUE,
             control = list(adapt_delta = .99, max_treedepth = 15))
(tfit2a <- (Sys.time()-t2a))

saveRDS(fit2a, file = "C:/Users/Kristina/Documents/fit2a.rds")

rm("fit2a")

### Stepwise Reference Group:
t2b <- Sys.time()
fit2b <- brm(cbind(adjR2_model, R2_test) ~ Method2 + (1|p|NumberID),
             data = All_Models,
             family = Beta(),
             iter = 30000,
             warmup = 2000,
             chains = 4,
             cores = 4,
             save_all_pars = TRUE,
             control = list(adapt_delta = .99, max_treedepth = 15))
(tfit2b <- (Sys.time()-t2b))

saveRDS(fit2b, file = "C:/Users/Kristina/Documents/fit2b.rds")

rm("fit2b")

### Model 3 Fixed Effects: Level-2 Predictors ###
### Fixed Effects: Level-2 Predictors (FOP1 Reference) ###
#Random Forest Reference
t3a <- Sys.time()
fit3a <- brm(cbind(adjR2_model, R2_test) ~ Method1 + Condition1a + (1|p|NumberID),
             data = All_Models,
             family = Beta(),
             iter = 30000,
             warmup = 2000,
```

```
        chains = 4,
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit3a <- (Sys.time()-t3a))


saveRDS(fit3a, file = "C:/Users/Kristina/Documents/fit3a.rds")

rm("fit3a")

### Fixed Effects: Level-2 Predictors (FOP2 Reference) ###
#Random Forest Reference
t3b <- Sys.time()
fit3b <- brm(cbind(adjR2_model, R2_test) ~ Method1 + Condition1b + (1|p|NumberID),
        data = All_Models,
        family = Beta(),
        iter = 30000,
        warmup = 2000,
        chains = 4,
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit3b <- (Sys.time()-t3b))

saveRDS(fit3b, file = "C:/Users/Kristina/Documents/fit3b.rds")

rm("fit3b")

### Fixed Effects: Level-2 Predictors (FOP1 Reference) ###
#Stepwise Reference
t3c <- Sys.time()
fit3c <- brm(cbind(adjR2_model, R2_test) ~ Method2 + Condition1a + (1|p|NumberID),
        data = All_Models,
        family = Beta(),
        iter = 30000,
        warmup = 2000,
        chains = 4,
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit3c <- (Sys.time()-t3c))

saveRDS(fit3c, file = "C:/Users/Kristina/Documents/fit3c.rds")

rm("fit3c")
```

```
### Fixed Effects: Level-2 Predictors (FOP2 Reference) ###
#Stepwise Reference
t3d <- Sys.time()
fit3d <- brm(cbind(adjR2_model, R2_test) ~ Method2 + Condition1b + (1|p|NumberID),
             data = All_Models,
             family = Beta(),
             iter = 30000,
             warmup = 2000,
             chains = 4,
             cores = 4,
             save_all_pars = TRUE,
             control = list(adapt_delta = .99, max_treedepth = 15))
(tfit3d <- (Sys.time()-t3d))

saveRDS(fit3d, file = "C:/Users/Kristina/Documents/fit3d.rds")

rm("fit3d")


### Model 4 Random Effects: ###
### Random Forest Reference Group:
t4a <- Sys.time()
fit4a <- brm(cbind(adjR2_model, R2_test) ~ Method1 + (1 + Method1|p|NumberID),
             data = All_Models,
             family = Beta(),
             iter = 30000,
             warmup = 2000,
             chains = 4,
             cores = 4,
             save_all_pars = TRUE,
             control = list(adapt_delta = .99, max_treedepth = 15))
(tfit4a <- (Sys.time()-t4a))

saveRDS(fit4a, file = "C:/Users/Kristina/Documents/ fit6a.rds")

rm("fit4a")

### Stepwise Reference Group:
t4b <- Sys.time()
fit4b <- brm(cbind(adjR2_model, R2_test) ~ Method2 + (1 + Method2|p|NumberID),
             data = All_Models,
             family = Beta(),
             iter = 30000,
             warmup = 2000,
             chains = 4,
```

```
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit4b <- (Sys.time()-t6b))

saveRDS(fit4b, file = "C:/Users/Kristina/Documents/fit6b.rds")

rm("fit4b")
```

Study Two Analyses R Code

Simulating Judgment Data

```
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)

SIMData <- read_excel("~/Ohio University/Dissertation 9.19.17/R Code/Dissertation
Simulated Data/Carter.FOP Study.Cereal Level Simulation.xlsx")


######JUDGMENT POLICY 1 VALUES##############################
datalist1 <- as.data.frame(subset(SIMData, NumberID == 1))

x1a <- datalist1$Calories
x2a <- datalist1$Sodiummg
x3a <- datalist1$Sugars_g
x4a <- datalist1$DietaryFiberg

b0a <- 50
b1a <- .01
b2a <- -.01
b3a <- -.01
b4a <- .01
sigma <- 7
```

```
set.seed(628)
epsa <- rnorm(73, 0, sigma)

datalist1$Judgment <- b0a + b1a*x1a  + b2a*x2a  + b3a*x3a + b4a*x4a + epsa

min(datalist1$Judgment)
max(datalist1$Judgment)
datalist1$Judgment

datalist1$Judgment_trunc <- ifelse(datalist1$Judgment > 100, 100,
ifelse(datalist1$Judgment < 1, 1, datalist1$Judgment))

min(datalist1$Judgment_trunc)
max(datalist1$Judgment_trunc)
mean(datalist1$Judgment_trunc)
median(datalist1$Judgment_trunc)

##################JUDGMENT POLICY 2 VALUES############################
datalist2 <- as.data.frame(subset(SIMData, NumberID == 2))

x1b <- datalist2$Calories
x2b <- datalist2$Sodiummg
x3b <- datalist2$Sugars_g
x4b <- datalist2$DietaryFiberg

b0b <- 50
b1b <- .25
b2b <- -.25
b3b <- -.25
b4b <- .25
sigma <- 7

set.seed(628)
epsb <- rnorm(73, 0, sigma)

datalist2$Judgment <- b0b + b1b*x1b  + b2b*x2b  + b3b*x3b + b4b*x4b + epsb

min(datalist2$Judgment)
max(datalist2$Judgment)

datalist2$Judgment
datalist2$Judgment_trunc <- ifelse(datalist2$Judgment > 100, 100,
ifelse(datalist2$Judgment < 1, 1, datalist2$Judgment))

datalist2$Judgment_trunc
```

```
min(datalist2$Judgment_trunc)
max(datalist2$Judgment_trunc)
mean(datalist2$Judgment_trunc)
median(datalist2$Judgment_trunc)

NEWSIMData <- bind_rows(datalist1, datalist2)

write.csv(NEWSIMData, file = "C:/Users/Kristina/Documents/NEWSIMData.csv")

#NEWSIMData$Judgment_trun
model1s <- lm(scale(Judgment_trunc) ~ scale(Calories) + scale(Sodiummg) +
scale(Sugars_g)
          + scale(DietaryFiberg), data = datalist1)


model2s <- lm(scale(Judgment_trunc) ~ scale(Calories) + scale(Sodiummg) +
scale(Sugars_g)
           + scale(DietaryFiberg), data = datalist2)

model1 <- lm(Judgment_trunc ~ Calories + Sodiummg + Sugars_g + DietaryFiberg, data
= datalist1)
model1
model2 <- lm(Judgment_trunc ~ Calories + Sodiummg + Sugars_g + DietaryFiberg, data
= datalist2)
model2

summary(model1)
summary(model2)
x1b <- datalist2$Calories
x2b <- datalist2$Sodiummg
x3b <- datalist2$Sugars_g
x4b <- datalist2$DietaryFiberg

################RESAMPLING SIMULATED DATA#############################
NewSIMData <- read.csv(file="C:/Users/Kristina/Documents/NewSIMData.csv",
header=TRUE, sep=",")

NewSIMData$Beta.Condition <- NewSIMData$NumberID
NewSIMData$bWeight <- ifelse(NewSIMData$Beta.Condition == 1, .01, .25)

datalist <- list()
resamples <- list()
set <- list()
generated <- list()
start <- 1
```

```
total <- 2
bootset <- 150

set.seed(1117)
t1 <- Sys.time()
for(k in start:total){
  datalist[[k]] <- as.data.frame(subset(NewSIMData, Beta.Condition == k))

  resamples[[k]] <- lapply(1:bootset, function(i) sample_n(datalist[[k]], 73, replace = T))

  for(j in 1:bootset){
    set[[j]] <- resamples[[k]][[j]]

  }
  generated[[k]] <- map_df(set, ~as.data.frame(.x), .id="bootset")
}
(tresample <- (Sys.time()-t1))


SIMresampled <- map_df(generated, ~as.data.frame(.x), .id="test")
SIMresampled$UniqueID <- ifelse(SIMresampled$Beta.Condition == 1,
as.numeric(SIMresampled$bootset), as.numeric(SIMresampled$bootset) + bootset)

write.csv(SIMresampled, file = "C:/Users/Kristina/Documents/SIM Sampling
Distribution Data.csv")
```

Stepwise Regression Analysis

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("lm.beta")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(lm.beta)
library(data.table)
```

```
SampDist <- read.csv(file="C:/Users/Kristina/Documents/SIM Sampling Distribution
Data.csv", header=TRUE, sep=",")

dfVar <- c("bootset", "bWeight", "Judgment_trunc", "RecordID", "UniqueID",
"Beta.Condition", "UPC", "CerealNumber", "Calories", "Calories_Fat", "Total_Fatg",
"Saturated_Fatg", "Trans_Fatg", "Poly_Fatg", "Mono_Fatg", "Cholesterolmg",
"Sodiummg", "Potassiummg", "TotalCarbsg", "DietaryFiberg", "Soluble_Fiberg",
"Insoluble_Fiberg", "Sugars_g", "Other_Carbsg", "Proteing", "VitsMinerals")

SDComplete <- SampDist[complete.cases(SampDist[, dfVar]), dfVar]
SDComplete <- as.data.frame(SDComplete[with(SDComplete, order(UniqueID,
Judgment_trunc)),])

rm(list = ls()[!ls()%in%c("SDComplete")])

count<-length(unique(SDComplete$UniqueID))
datalist <- list()
full_model <- list()
null_model <- list()
stepwise <- list()
summaries <- list()
list_names <- list()
df_coefficients <- list()
df_betas <- list()
df_fstat <- list()
scale_full_model <- list()
scale_null_model <- list()
scale_stepwise <- list()


####SIMULATED DATA STEPWISE ANALYSIS###############
set.seed(115)
t1 <- Sys.time()
for(i in 1:count){
  datalist[[i]] <- as.data.frame(subset(SDComplete, UniqueID == i))

  full_model[[i]] <- lm(Judgment_trunc ~ Calories + Calories_Fat + Total_Fatg +
Saturated_Fatg +
                        Poly_Fatg + Mono_Fatg + Sodiummg + Potassiummg +
                        DietaryFiberg + Soluble_Fiberg + Insoluble_Fiberg + Sugars_g +
                        TotalCarbsg + Other_Carbsg + Proteing + VitsMinerals,
                      data = datalist[[i]])

  null_model[[i]] <- lm(Judgment_trunc ~ 1, data = datalist[[i]])
```

```
  stepwise[[i]] <- step(null_model[[i]], scope = list(upper = full_model[[i]]))

  scale_full_model[[i]] <- lm(scale(Judgment_trunc) ~ scale(Calories) +
scale(Calories_Fat) + scale(Total_Fatg) +
                              scale(Saturated_Fatg) +  scale(Poly_Fatg) + scale(Mono_Fatg) +
                              scale(Sodiummg) + scale(Potassiummg) + scale(DietaryFiberg) +
                              scale(Soluble_Fiberg) + scale(Insoluble_Fiberg) + scale(Sugars_g) +
                              scale(TotalCarbsg) + scale(Other_Carbsg) + scale(Proteing) +
                              scale(VitsMinerals),
                            data = datalist[[i]])

  scale_null_model[[i]] <- lm(scale(Judgment_trunc) ~ 1, data = datalist[[i]])

  scale_stepwise[[i]] <- step(scale_null_model[[i]], scope = list(upper =
scale_full_model[[i]]))

  summaries[[i]] <- summary(stepwise[[i]])
  list_names[[i]] <- paste0("ID_", unique(datalist[[i]]$NumberID))

  require(broom)
  df_coefficients[[i]] <- tidy(stepwise[[i]])
  df_betas[[i]] <- tidy(scale_stepwise[[i]])
  df_fstat[[i]] <- as.data.frame((summaries[[i]][["r.squared"]]))
  colnames(df_fstat[[i]]) <- "r.squared"
  df_fstat[[i]]$value <- ifelse(is_empty(summaries[[i]]$fstatistic[[1]]) == TRUE, 0,
summaries[[i]]$fstatistic[[1]])
  df_fstat[[i]]$numdf <- ifelse(is_empty(summaries[[i]]$fstatistic[[2]]) == TRUE, 0,
summaries[[i]]$fstatistic[[2]])
  df_fstat[[i]]$dendf <- ifelse(is_empty(summaries[[i]]$fstatistic[[3]]) == TRUE, 0,
summaries[[i]]$fstatistic[[3]])

}
print(totalanalysis <- (Sys.time()-t1))

##############Model Statistics############################
#Adjusted R^2#
model_stats <- rbind(lapply(summaries, `[`, 9))
model <- map_df(model_stats, ~as.data.frame(.x), .id="UniqueID")

#F-Statistics#
fstats <- bind_rows(df_fstat, .id = 'UniqueID')
fstats$fstat <- fstats$value
fstatVar <- c("UniqueID", "fstat", "numdf", "dendf")
fstatistics <- fstats[complete.cases(fstats[, fstatVar]), fstatVar]
```

```
#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_model <- merge(model, fstatistics, by = c('UniqueID'), all.x = TRUE)


#Computing p-values, assigning binary significance values
stepwise_model$pvalue<- pf(stepwise_model$fstat, stepwise_model$numdf,
stepwise_model$dendf, lower.tail = FALSE)
stepwise_model$modelsig <- ifelse(stepwise_model$pvalue < .05, 1, 0)

######################################################################
######################################################################
#Creating dataframe for unstandardized coefficient estimates#
coefficients <- bind_rows(df_coefficients, .id = 'UniqueID')

#Creating mergeable dataframe for standardized coefficient estimates#
betas_mid <- bind_rows(df_betas, .id = 'UniqueID')
betas_mid$term2 <- gsub("scale\\(", "", betas_mid$term)
betas_mid$term3 <- gsub("g[)]", "g", betas_mid$term2)
betas_mid$term4 <- gsub("s[)]", "s", betas_mid$term3)
betas_mid$term5 <- gsub("t[)]", "t", betas_mid$term4)
betas_mid$term <- gsub("[(]Intercept", "(Intercept)", betas_mid$term5)
betas_mid$beta <- betas_mid$estimate
betaVar <- c("UniqueID", "term", "beta")
betas <- betas_mid[complete.cases(betas_mid[, betaVar]), betaVar]
betas$beta_value <- ifelse(betas$term == '(Intercept)', NA, betas$beta)


#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_coeff <- merge(coefficients, betas, by = c('UniqueID', 'term'))

######Assigning Meta Values################
#Significance values#
stepwise_coeff$coeffsig <- ifelse(stepwise_coeff$p.value < .05, 1, 0)
#Positive/negative betas#
stepwise_coeff$coeffdirection <- ifelse(stepwise_coeff$beta > 0, 'positive', 'negative')

#Ranking values#
coeff_data <- list()
for(i in 1:count){
  coeff_data[[i]] <- as.data.frame(subset(stepwise_coeff, UniqueID == i))
  coeff_data[[i]]$rank <- rank(-abs(coeff_data[[i]]$beta_value), na.last = FALSE,
ties.method = "average")
  coeff_data[[i]]$adjrank <- coeff_data[[i]]$rank - 1
  final_stepwise_coeff <- bind_rows(coeff_data)
}
```

```
#Adding Predictor Count to File#
final_model <- as.data.frame(subset(final_stepwise_coeff))
predictors <- c("UniqueID", "term")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]


stepmodel <- list()
predictorlist <- list()
for(i in 1:count){
  stepmodel[[i]] <- as.data.frame(subset(stepwise_model, UniqueID == i))
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, UniqueID == i))
  predictorlist[[i]]$Count <- (nrow(predictorlist[[i]]))-1
  stepmodel[[i]]$model_var <- (nrow(predictorlist[[i]]))-1
}

stepwise_model <- map_df(stepmodel, ~as.data.frame(.x), .id="UniqueID")
final_model_predictors <- map_df(predictorlist, ~as.data.frame(.x), .id="UniqueID")

#########################################################################

modelVar <- c("UniqueID", "term", "estimate")

model_estimates <- final_stepwise_coeff[complete.cases(final_stepwise_coeff[,
modelVar]), modelVar]

person_model <- list()
term_model <- list()
step_model <- list()
weights <- list()

for(k in 1:count){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, UniqueID == k))
  step_model[[k]] <- as.data.frame(subset(stepwise_model, UniqueID == k))
  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  step_model[[k]]$intercept <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)', 1, 0))
  step_model[[k]]$cal <- sum(ifelse(weights[[k]]$Predictor == 'Calories', 1, 0))
  step_model[[k]]$calf <- sum(ifelse(weights[[k]]$Predictor == 'Calories_Fat', 1, 0))
```

```
  step_model[[k]]$tfat <- sum(ifelse(weights[[k]]$Predictor == 'Total_Fatg', 1, 0))
  step_model[[k]]$sfat <- sum(ifelse(weights[[k]]$Predictor == 'Saturated_Fatg', 1, 0))
  step_model[[k]]$pfat <- sum(ifelse(weights[[k]]$Predictor == 'Poly_Fatg', 1, 0))
  step_model[[k]]$mfat <- sum(ifelse(weights[[k]]$Predictor == 'Mono_Fatg', 1, 0))
  step_model[[k]]$sod <- sum(ifelse(weights[[k]]$Predictor == 'Sodiummg', 1, 0))
  step_model[[k]]$potas <- sum(ifelse(weights[[k]]$Predictor == 'Potassiummg', 1, 0))
  step_model[[k]]$dfiber <- sum(ifelse(weights[[k]]$Predictor == 'DietaryFiberg', 1, 0))
  step_model[[k]]$sfiber <- sum(ifelse(weights[[k]]$Predictor == 'Soluble_Fiberg', 1, 0))
  step_model[[k]]$ifiber <- sum(ifelse(weights[[k]]$Predictor == 'Insoluble_Fiberg', 1,
0))
  step_model[[k]]$sugar <- sum(ifelse(weights[[k]]$Predictor == 'Sugars_g', 1, 0))
  step_model[[k]]$tcarb <- sum(ifelse(weights[[k]]$Predictor == 'TotalCarbsg', 1, 0))
  step_model[[k]]$ocarb <- sum(ifelse(weights[[k]]$Predictor == 'Other_Carbsg', 1, 0))
  step_model[[k]]$protein <- sum(ifelse(weights[[k]]$Predictor == 'Proteing', 1, 0))
  step_model[[k]]$vitmin <- sum(ifelse(weights[[k]]$Predictor == 'VitsMinerals', 1, 0))


  step_model[[k]]$intercept_b <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  step_model[[k]]$cal_b <- sum(ifelse(weights[[k]]$Predictor == 'Calories',
weights[[k]]$Mean, 0))
  step_model[[k]]$calf_b <- sum(ifelse(weights[[k]]$Predictor == 'Calories_Fat',
weights[[k]]$Mean, 0))
  step_model[[k]]$tfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Total_Fatg',
weights[[k]]$Mean, 0))
  step_model[[k]]$sfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Saturated_Fatg',
weights[[k]]$Mean, 0))
  step_model[[k]]$pfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Poly_Fatg',
weights[[k]]$Mean, 0))
  step_model[[k]]$mfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Mono_Fatg',
weights[[k]]$Mean, 0))
  step_model[[k]]$sod_b <- sum(ifelse(weights[[k]]$Predictor == 'Sodiummg',
weights[[k]]$Mean, 0))
  step_model[[k]]$potas_b <- sum(ifelse(weights[[k]]$Predictor == 'Potassiummg',
weights[[k]]$Mean, 0))
  step_model[[k]]$dfiber_b <- sum(ifelse(weights[[k]]$Predictor == 'DietaryFiberg',
weights[[k]]$Mean, 0))
  step_model[[k]]$sfiber_b <- sum(ifelse(weights[[k]]$Predictor == 'Soluble_Fiberg',
weights[[k]]$Mean, 0))
  step_model[[k]]$ifiber_b <- sum(ifelse(weights[[k]]$Predictor == 'Insoluble_Fiberg',
weights[[k]]$Mean, 0))
  step_model[[k]]$sugar_b <- sum(ifelse(weights[[k]]$Predictor == 'Sugars_g',
weights[[k]]$Mean, 0))
  step_model[[k]]$tcarb_b <- sum(ifelse(weights[[k]]$Predictor == 'TotalCarbsg',
weights[[k]]$Mean, 0))
```

```
  step_model[[k]]$ocarb_b <- sum(ifelse(weights[[k]]$Predictor == 'Other_Carbsg',
weights[[k]]$Mean, 0))
  step_model[[k]]$protein_b <- sum(ifelse(weights[[k]]$Predictor == 'Proteing',
weights[[k]]$Mean, 0))
  step_model[[k]]$vitmin_b <- sum(ifelse(weights[[k]]$Predictor == 'VitsMinerals',
weights[[k]]$Mean, 0))

}

stepwise_model <- map_df(step_model, ~as.data.frame(.x), .id="UniqueID")

Conditions <- subset(SDComplete, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "Beta.Condition", "bWeight")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

final_stepwise_coeff <- merge(condition, final_stepwise_coeff, by = c('UniqueID'), all.x
= TRUE)

##SAVING ON PERSONAL PC: PC1###
write.csv(final_stepwise_coeff, file =
"C:/Users/Kristina/Documents/SIMfinal_stepwise_coeff.csv")
write.csv(stepwise_model, file =
"C:/Users/Kristina/Documents/SIMfinal_stepwise_model.csv")
write.csv(final_model_predictors, file =
"C:/Users/Kristina/Documents/SIMfinal_model_predictors.csv")
```

All Subsets Analysis

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("olsrr")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(data.table)
```

```
#Data from Personal PC: PC1
SampDist <- read.csv(file="C:/Users/Kristina/Documents/SIM Sampling Distribution
Data.csv", header=TRUE, sep=",")

dfVar <- c("bootset", "bWeight", "Judgment_trunc", "RecordID", "UniqueID",
"Beta.Condition", "Product", "UPC", "CerealNumber", "Calories", "Calories_Fat",
"Total_Fatg", "Saturated_Fatg", "Trans_Fatg", "Poly_Fatg", "Mono_Fatg",
"Cholesterolmg", "Sodiummg", "Potassiummg", "TotalCarbsg", "DietaryFiberg",
"Soluble_Fiberg", "Insoluble_Fiberg", "Sugars_g", "Other_Carbsg", "Proteing",
"VitsMinerals")

SDComplete <- SampDist[complete.cases(SampDist[, dfVar]), dfVar]
SDComplete <- as.data.frame(SDComplete[with(SDComplete, order(UniqueID,
Judgment_trunc)),])

rm(list = ls()[!ls()%in%c("SDComplete")])


######################ALL SUBSETS ANALYSIS FUNCTION#################
all_subsets <- function(last, group, end, bootset){

  rm(list = ls()[!ls()%in%c("SDComplete", "all_subsets", "tfunc", "last", "group", "end",
"bootset")])

  t1 <- Sys.time()

  datalist <- list()
   sample <- list()
  resamples <- list()
  judgment_set <- list()
  leaps <- list()
  summary <- list()
  maxadjr2 <- list()
  variables1 <- list()
  variables2 <- list()
  variables3 <- list()
  coefficients1 <- list()
  coefficients2 <- list()
  bestmodel <- list()
  judge_bestmodels <- list()
  count <- list()
  bestfit <- list()
  only_bestmodels <- list()
```

```
  modelsonly1 <- list()
  modelsonly2 <- list()
  modelsonly3 <- list()
  modelsonly4 <- list()
  freqmodel <- list()
  variable_importance <- list()

start <- ifelse((1 + last <= end), (1 + last), end)
judges <- ifelse((last + group <= end), (last + group), end)

for(k in start:judges){
  index <- start - 1
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(SDComplete, UniqueID == k))

  resamples[[k]] <- lapply(1:bootset, function(i) sample_n(datalist[[k]], 73, replace = T))

  for(j in 1:bootset){
    t3 <- Sys.time()
    judgment_set[[j]] <- resamples[[k]][[j]]
    leaps[[j]] <-
      regsubsets(Judgment_trunc ~ Calories + Calories_Fat + Total_Fatg + Saturated_Fatg
+ Poly_Fatg + Mono_Fatg + Sodiummg + Potassiummg + DietaryFiberg +
Soluble_Fiberg + Insoluble_Fiberg + Sugars_g + TotalCarbsg + Other_Carbsg + Proteing
+ VitsMinerals, data = judgment_set[[j]],
                 nbest = 1,       # 1 best model for each number of predictors
                 nvmax = NULL,    # NULL for no limit on number of variables
                 force.in = NULL, force.out = NULL,
                 method = "exhaustive")
    summary[[j]] <- summary(leaps[[j]])
    maxadjr2[[j]] <- which.max(summary[[j]]$adjr2)
    variables1[[j]] <- map_df(summary[[j]]$which[maxadjr2[[j]],], ~as.data.frame(.x),
.id="term")
    variables1[[j]]$included <- variables1[[j]]$.x
    variables2[[j]] <- variables1[[j]][-c(2)]
    variables3[[j]] <- as.data.frame(subset(variables2[[j]]))
    variables3[[j]]$adj.r.squared <- summary[[j]]$adjr2[maxadjr2[[j]]]
    variables3[[j]]$inclusion <- ifelse(variables3[[j]]$included == TRUE, 1, 0)
    variables3[[j]]$modelcode <- paste0(variables3[[j]]$inclusion, collapse = "")
    coefficients1[[j]] <- map_df(coef(leaps[[j]],maxadjr2[[j]],vcov=FALSE),
~as.data.frame(.x), .id="term")
    coefficients1[[j]]$estimate <- coefficients1[[j]]$`coef(leaps[[j]], maxadjr2[[j]], vcov =
FALSE)`
    coefficients1[[j]]$estimate <- coefficients1[[j]]$.x
    coefficients2[[j]] <- coefficients1[[j]][-c(2)]
```

```
  bestmodel[[j]] <- merge(variables3[[j]], coefficients2[[j]], by = "term")
  modelsonly1[[j]] <- bestmodel[[j]][!duplicated(bestmodel[[j]]$modelcode),]

  print(bootstrapset <- Sys.time()-t3)

 }

 modelsonly2[[k]] <- map_df(modelsonly1, ~as.data.frame(.x), .id="bootset1")
 modelsonly3[[k]] <- count(modelsonly2[[k]], modelcode)
 freqmodel[[k]] <- which.max(modelsonly3[[k]]$n)
 modelsonly3[[k]]$bestfit <- modelsonly3[[k]]$modelcode[freqmodel[[k]]]
 modelsonly4 <- map_df(modelsonly3, ~as.data.frame(.x), .id="UniqueID")

 judge_bestmodels[[k]] <- map_df(bestmodel, ~as.data.frame(.x), .id="bootset")
 bestfit[[k]] <- merge(judge_bestmodels[[k]], modelsonly3[[k]], by = "modelcode")
 bestfit[[k]]$bestmodel <- ifelse(bestfit[[k]]$modelcode == bestfit[[k]]$bestfit, TRUE,
FALSE)
 only_bestmodels[[k]] <- subset(bestfit[[k]], bestmodel == TRUE)
 variable_importance[[k]] <- count(judge_bestmodels[[k]], term)
 variable_importance[[k]]$frequency <- variable_importance[[k]]$n
 variable_importance[[k]]$probability <- (variable_importance[[k]]$frequency)/(bootset)
 print(judgeanalysis <- Sys.time()-t2)

 #complete_bootsets <- map_df(bestfit, ~as.data.frame(.x), .id="UniqueID")
 #complete_bootsets <- complete_bootsets[
  # with(complete_bootsets, order(UniqueID, bootset, term)),
   #]
 #complete_bootsets$UniqueID <- as.numeric(complete_bootsets$UniqueID) + index

 SIMfinal_bestmodels <- map_df(only_bestmodels, ~as.data.frame(.x), .id="UniqueID")
 SIMfinal_bestmodels <- as.data.frame(SIMfinal_bestmodels[
  with(SIMfinal_bestmodels, order(UniqueID, bootset, term)),
  ])
 SIMfinal_bestmodels$UniqueID <- as.numeric(SIMfinal_bestmodels$UniqueID) +
index

 SIMpredictor_importance <- map_df(variable_importance, ~as.data.frame(.x),
.id="UniqueID")
 SIMpredictor_importance$UniqueID <-
as.numeric(SIMpredictor_importance$UniqueID) + index
}
print(totalanalysis <- (Sys.time()-t1))

###SAVE RELEVANT FILES
#write.csv(SIMfinal_bestmodels, file = paste0("C:/Users/Kristina/Documents/Study One
Analysis/IDs", start, "-", judges, " All Subsets SIMfinal_bootsets.csv", sep = ""))
```

```
#write.csv(SIMpredictor_importance, file = paste0("C:/Users/Kristina/Documents/Study One Analysis/IDs", start, "-", judges, " All Subsets SIMpredictor_importance.csv", sep = ""))

best_models <- list()
bootset_model <- list()
AS_model <- list()
criticality <- list()
VI_model <- list()
VI <- list()
term_model <- list()
weights <- list()

t1a <- Sys.time()
for(k in start:judges){
  t2 <- Sys.time()
  best_models[[k]] <- as.data.frame(subset(SIMfinal_bestmodels, UniqueID == k))
  bootset_model[[k]] <- as.data.frame(subset(best_models[[k]], bootset == min(bootset)))

  criticality[[k]] <- as.data.frame(subset(SIMpredictor_importance, UniqueID == k))

  AS_model[[k]] <- bootset_model[[k]][!duplicated(bootset_model[[k]]$UniqueID),]

  AS_model[[k]]$model_var <- sum(bootset_model[[k]]$inclusion) - 1

  AS_model[[k]]$mean_adjR2 <- mean(best_models[[k]]$adj.r.squared)

  term_model[[k]] <-  split(best_models[[k]], best_models[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))

  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")


  VI_model[[k]] <- split(criticality[[k]], criticality[[k]]$term)
  VI[[k]] <- as.data.frame(sapply(VI_model[[k]], function(x) mean(x$probability)))

  setDT(VI[[k]], keep.rownames = TRUE)[]
  setnames(VI[[k]], 1, "term")
  setnames(VI[[k]], 2, "Mean")

  AS_model[[k]]$intercept <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)', 1, 0))
  AS_model[[k]]$cal <- sum(ifelse(weights[[k]]$Predictor == 'Calories', 1, 0))
  AS_model[[k]]$calf <- sum(ifelse(weights[[k]]$Predictor == 'Calories_Fat', 1, 0))
  AS_model[[k]]$tfat <- sum(ifelse(weights[[k]]$Predictor == 'Total_Fatg', 1, 0))
```

```
  AS_model[[k]]$sfat <- sum(ifelse(weights[[k]]$Predictor == 'Saturated_Fatg', 1, 0))
  AS_model[[k]]$pfat <- sum(ifelse(weights[[k]]$Predictor == 'Poly_Fatg', 1, 0))
  AS_model[[k]]$mfat <- sum(ifelse(weights[[k]]$Predictor == 'Mono_Fatg', 1, 0))
  AS_model[[k]]$sod <- sum(ifelse(weights[[k]]$Predictor == 'Sodiummg', 1, 0))
  AS_model[[k]]$potas <- sum(ifelse(weights[[k]]$Predictor == 'Potassiummg', 1, 0))
  AS_model[[k]]$dfiber <- sum(ifelse(weights[[k]]$Predictor == 'DietaryFiberg', 1, 0))
  AS_model[[k]]$sfiber <- sum(ifelse(weights[[k]]$Predictor == 'Soluble_Fiberg', 1, 0))
  AS_model[[k]]$ifiber <- sum(ifelse(weights[[k]]$Predictor == 'Insoluble_Fiberg', 1, 0))
  AS_model[[k]]$sugar <- sum(ifelse(weights[[k]]$Predictor == 'Sugars_g', 1, 0))
  AS_model[[k]]$tcarb <- sum(ifelse(weights[[k]]$Predictor == 'TotalCarbsg', 1, 0))
  AS_model[[k]]$ocarb <- sum(ifelse(weights[[k]]$Predictor == 'Other_Carbsg', 1, 0))
  AS_model[[k]]$protein <- sum(ifelse(weights[[k]]$Predictor == 'Proteing', 1, 0))
  AS_model[[k]]$vitmin <- sum(ifelse(weights[[k]]$Predictor == 'VitsMinerals', 1, 0))


  AS_model[[k]]$intercept_b <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  AS_model[[k]]$cal_b <- sum(ifelse(weights[[k]]$Predictor == 'Calories',
weights[[k]]$Mean, 0))
  AS_model[[k]]$calf_b <- sum(ifelse(weights[[k]]$Predictor == 'Calories_Fat',
weights[[k]]$Mean, 0))
  AS_model[[k]]$tfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Total_Fatg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$sfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Saturated_Fatg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$pfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Poly_Fatg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$mfat_b <- sum(ifelse(weights[[k]]$Predictor == 'Mono_Fatg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$sod_b <- sum(ifelse(weights[[k]]$Predictor == 'Sodiummg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$potas_b <- sum(ifelse(weights[[k]]$Predictor == 'Potassiummg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$dfiber_b <- sum(ifelse(weights[[k]]$Predictor == 'DietaryFiberg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$sfiber_b <- sum(ifelse(weights[[k]]$Predictor == 'Soluble_Fiberg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$ifiber_b <- sum(ifelse(weights[[k]]$Predictor == 'Insoluble_Fiberg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$sugar_b <- sum(ifelse(weights[[k]]$Predictor == 'Sugars_g',
weights[[k]]$Mean, 0))
  AS_model[[k]]$tcarb_b <- sum(ifelse(weights[[k]]$Predictor == 'TotalCarbsg',
weights[[k]]$Mean, 0))
  AS_model[[k]]$ocarb_b <- sum(ifelse(weights[[k]]$Predictor == 'Other_Carbsg',
weights[[k]]$Mean, 0))
```

```
  AS_model[[k]]$protein_b <- sum(ifelse(weights[[k]]$Predictor == 'Proteing',
weights[[k]]$Mean, 0))
  AS_model[[k]]$vitmin_b <- sum(ifelse(weights[[k]]$Predictor == 'VitsMinerals',
weights[[k]]$Mean, 0))

  AS_model[[k]]$intercept_PC <- sum(ifelse(VI[[k]]$term == '(Intercept)',
VI[[k]]$Mean, 0))
  AS_model[[k]]$cal_PC <- sum(ifelse(VI[[k]]$term == 'Calories', VI[[k]]$Mean, 0))
  AS_model[[k]]$calf_PC <- sum(ifelse(VI[[k]]$term == 'Calories_Fat', VI[[k]]$Mean,
0))
  AS_model[[k]]$tfat_PC <- sum(ifelse(VI[[k]]$term == 'Total_Fatg', VI[[k]]$Mean, 0))
  AS_model[[k]]$sfat_PC <- sum(ifelse(VI[[k]]$term == 'Saturated_Fatg', VI[[k]]$Mean,
0))
  AS_model[[k]]$pfat_PC <- sum(ifelse(VI[[k]]$term == 'Poly_Fatg', VI[[k]]$Mean, 0))
  AS_model[[k]]$mfat_PC <- sum(ifelse(VI[[k]]$term == 'Mono_Fatg', VI[[k]]$Mean,
0))
  AS_model[[k]]$sod_PC <- sum(ifelse(VI[[k]]$term == 'Sodiummg', VI[[k]]$Mean, 0))
  AS_model[[k]]$potas_PC <- sum(ifelse(VI[[k]]$term == 'Potassiummg', VI[[k]]$Mean,
0))
  AS_model[[k]]$dfiber_PC <- sum(ifelse(VI[[k]]$term == 'DietaryFiberg',
VI[[k]]$Mean, 0))
  AS_model[[k]]$sfiber_PC <- sum(ifelse(VI[[k]]$term == 'Soluble_Fiberg',
VI[[k]]$Mean, 0))
  AS_model[[k]]$ifiber_PC <- sum(ifelse(VI[[k]]$term == 'Insoluble_Fiberg',
VI[[k]]$Mean, 0))
  AS_model[[k]]$sugar_PC <- sum(ifelse(VI[[k]]$term == 'Sugars_g', VI[[k]]$Mean, 0))
  AS_model[[k]]$tcarb_PC <- sum(ifelse(VI[[k]]$term == 'TotalCarbsg', VI[[k]]$Mean,
0))
  AS_model[[k]]$ocarb_PC <- sum(ifelse(VI[[k]]$term == 'Other_Carbsg',
VI[[k]]$Mean, 0))
  AS_model[[k]]$protein_PC <- sum(ifelse(VI[[k]]$term == 'Proteing', VI[[k]]$Mean, 0))
  AS_model[[k]]$vitmin_PC <- sum(ifelse(VI[[k]]$term == 'VitsMinerals',
VI[[k]]$Mean, 0))
}

all_subsets_model <- map_df(AS_model, ~as.data.frame(.x), .id="UniqueID")

##SAVING ON PERSONAL PC: ###.
#write.csv(all_subsets_model, file = paste0("C:/Users/Kristina/Documents/Study One
Analysis/IDs", start, "-", judges, " All Subsets SIM Model Outcomes.csv", sep = ""))

print(cross_validated <- (Sys.time()-t1))

}
```

```
################################################################################
#############RUN ALL SUBSETS ANALYSIS FUNCTION#####################
set.seed(1015)
set <- seq.int(0, 300, by = 5)#by should be same size as group,indicates how many
participants to increase by at next cycle
for(a in set){
tfunc <- Sys.time()
all_subsets(last=a, group = 5, end = 300, bootset = 100000)
print(totalanalysis <- (Sys.time()-tfunc))
}
################################################################################
```

Random Forest Analysis

```
#install.packages("VSURF")
#install.packages("randomForest")
#install.packages("party")
#install.packages("readxl")
#install.packages("purrr")
#install.packages("data.table")
#install.packages("dplyr")

rm(list = ls())

library(VSURF)
library(randomForest)
library(party)
library(readxl)
library(purrr)
library(data.table)
library(dplyr)

#Data from Personal PC:
SampDist <- read.csv(file="C:/Users/Kristina/Documents/Dissertation Simulated
Data/SIM Sampling Distribution Data.csv", header=TRUE, sep=",")

dfVar <- c("bootset", "bWeight", "Judgment_trunc", "RecordID", "UniqueID",
"Beta.Condition", "Product", "UPC", "CerealNumber", "Calories", "Calories_Fat",
"Total_Fatg", "Saturated_Fatg", "Trans_Fatg", "Poly_Fatg", "Mono_Fatg",
"Cholesterolmg", "Sodiummg", "Potassiummg", "TotalCarbsg", "DietaryFiberg",
"Soluble_Fiberg", "Insoluble_Fiberg", "Sugars_g", "Other_Carbsg", "Proteing",
"VitsMinerals")

SDComplete <- SampDist[complete.cases(SampDist[, dfVar]), dfVar]
```

```
SDComplete <- as.data.frame(SDComplete[with(SDComplete, order(UniqueID,
Judgment_trunc)),])

rm(list = ls()[!ls()%in%c("SDComplete")])

SDComplete <- subset(SDComplete, select=-c(Trans_Fatg,Cholesterolmg))

########################################
datalist <- list()
sample <- list()
model_sample <- list()
validate_sample <- list()
VSURF.output <- list()
predictor_num <- list()
n <- list()
actual <- list()
predicted <- list()
R2 <- list()
adjR2 <- list()
predictors <- list()
thres_predictors <- list()
var_imp <- list()
correlate <- list()
R2_predict <- list()
MSE_predict <- list()
PRESS_predict <- list()


################SIMULATED DATA RANDOM FOREST ANALYSIS##########
start <- 1
judges <- 300
#################################################################################


#############RUN BELOW AS SINGLE BLOCK#############################
set.seed(115)
t1 <- Sys.time()
for(k in start:judges){
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(SDComplete, UniqueID == k))

  VSURF.output[[k]] <- VSURF(datalist[[k]][,10:25], datalist[[k]]$Judgment_trunc, ntree
= 2000,
                     mtry = max(floor(ncol(datalist[[k]][,10:25])/3), 1),
                     nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1)
```

```
  n[[k]] <- nrow(datalist[[k]])
  predictor_num[[k]] <- nrow(as.data.frame(VSURF.output[[k]]$varselect.interp))

  datalist[[k]]$actual <- datalist[[k]]$Judgment_trunc
  datalist[[k]]$predicted <- predict(VSURF.output[[k]], datalist[[k]], step = c("interp"))
  R2[[k]] <- 1 - (sum((datalist[[k]]$actual-
datalist[[k]]$predicted)^2)/sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))
  adjR2[[k]]  <- 1-((sum((datalist[[k]]$actual-datalist[[k]]$predicted)^2))/(n[[k]]-
predictor_num[[k]]-1))/((sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))/n[[k]]-
1)

  predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.interp)
  predictors[[k]]$predictor_num <- predictors[[k]]$`VSURF.output[[k]]$varselect.interp`
  predictors[[k]]$term <- colnames(datalist[[k]])[9+predictors[[k]]$predictor_num]
  predictors[[k]] <- subset(predictors[[k]], select = -1)
  predictors[[k]]$rank <- row.names(predictors[[k]])
  included_predictors <- map_df(predictors, ~as.data.frame(.x), .id="UniqueID")

  thres_predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.thres)
  thres_predictors[[k]]$predictor_num <-
thres_predictors[[k]]$`VSURF.output[[k]]$varselect.thres`
  thres_predictors[[k]]$term <-
colnames(datalist[[k]])[9+thres_predictors[[k]]$predictor_num]
  thres_predictors[[k]] <- select(thres_predictors[[k]], -1)
  thres_predictors[[k]]$var_importance <- VSURF.output[[k]]$imp.varselect.thres
  thres_predictors[[k]]$rank <- row.names(thres_predictors[[k]])

  threshold_step <- map_df(thres_predictors, ~as.data.frame(.x), .id="UniqueID")

  var_imp[[k]] <- as.data.frame(VSURF.output[[k]]$imp.mean.dec.ind)
  var_imp[[k]]$predictor_num <- var_imp[[k]]$`VSURF.output[[k]]$imp.mean.dec.ind`
  var_imp[[k]]$term <- colnames(datalist[[k]])[9+var_imp[[k]]$predictor_num]
  var_imp[[k]] <- select(var_imp[[k]], -1)
  var_imp[[k]]$var_importance <- VSURF.output[[k]]$imp.mean.dec
  var_imp[[k]]$rank <- row.names(var_imp[[k]])

  var_importance <- map_df(var_imp, ~as.data.frame(.x), .id="UniqueID")

  print(judgeanalysis <- Sys.time()-t2)
}
print(totalanalysis <- (Sys.time()-t1))
```

```
##SAVING RELEVANT DATA FILES:###
write.csv(included_predictors, file = "C:/Users/Kristina/Documents/SIMULATED
DATA Random Forest Final Predictors.csv")
write.csv(threshold_step, file = "C:/Users/Kristina/Documents/SIMULATED DATA
Random Forest Threshold Predictors.csv")
write.csv(var_importance, file = "C:/Users/Kristina/Documents/SIMULATED DATA
Random Forest Variable Importance.csv")

##########################################
### Model Statistics ###

R2_model <- map_df(R2, ~as.data.frame(.x), .id = "UniqueID")
R2_model$R2_model <- R2_model$.x
R2_model <- select(R2_model, -2)

adjR2_model <- map_df(adjR2, ~as.data.frame(.x), .id = "UniqueID")
adjR2_model$adjrR2 <- adjR2_model$.x
adjR2_model <- select(adjR2_model, -2)

### Merging Model Stats ###
R_model_stats <- merge(R2_model, adjR2_model, by = c('UniqueID'), all.x = TRUE)

#Adding Variable Count to File#
final_model <- as.data.frame(subset(included_predictors))
predictors <- c("UniqueID", "term", "rank")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]

predictorlist <- list()
statslist <- list()
for(i in start:judges){
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, UniqueID == i))
  statslist[[i]] <- as.data.frame(subset(R_model_stats, UniqueID == i))

  statslist[[i]]$model_var <- nrow(predictorlist[[i]])
}

R_model_stats <- map_df(statslist, ~as.data.frame(.x), .id="UniqueID")

#########################################################################

modelVar <- c("UniqueID", "term", "var_importance", "rank")

model_estimates <- var_importance[complete.cases(var_importance[, modelVar]),
modelVar]
```

```
person_model <- list()
term_model <- list()
forest_model <- list()
final_model <- list()
weights <- list()

for(k in 1:judges){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, UniqueID == k))

  forest_model[[k]] <- as.data.frame(subset(R_model_stats, UniqueID == k))

  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x)
mean(x$var_importance)))
  final_model[[k]] <- as.data.frame(subset(final_model_predictors, UniqueID == k))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  forest_model[[k]]$intercept <- sum(ifelse(final_model[[k]]$term == '(Intercept)', 1, 0))
  forest_model[[k]]$cal <- sum(ifelse(final_model[[k]]$term == 'Calories', 1, 0))
  forest_model[[k]]$calf <- sum(ifelse(final_model[[k]]$term == 'Calories_Fat', 1, 0))
  forest_model[[k]]$tfat <- sum(ifelse(final_model[[k]]$term == 'Total_Fatg', 1, 0))
  forest_model[[k]]$sfat <- sum(ifelse(final_model[[k]]$term == 'Saturated_Fatg', 1, 0))
  forest_model[[k]]$pfat <- sum(ifelse(final_model[[k]]$term == 'Poly_Fatg', 1, 0))
  forest_model[[k]]$mfat <- sum(ifelse(final_model[[k]]$term == 'Mono_Fatg', 1, 0))
  forest_model[[k]]$sod <- sum(ifelse(final_model[[k]]$term == 'Sodiummg', 1, 0))
  forest_model[[k]]$potas <- sum(ifelse(final_model[[k]]$term == 'Potassiummg', 1, 0))
  forest_model[[k]]$dfiber <- sum(ifelse(final_model[[k]]$term == 'DietaryFiberg', 1, 0))
  forest_model[[k]]$sfiber <- sum(ifelse(final_model[[k]]$term == 'Soluble_Fiberg', 1,
0))
  forest_model[[k]]$ifiber <- sum(ifelse(final_model[[k]]$term == 'Insoluble_Fiberg', 1,
0))
  forest_model[[k]]$sugar <- sum(ifelse(final_model[[k]]$term == 'Sugars_g', 1, 0))
  forest_model[[k]]$tcarb <- sum(ifelse(final_model[[k]]$term == 'TotalCarbsg', 1, 0))
  forest_model[[k]]$ocarb <- sum(ifelse(final_model[[k]]$term == 'Other_Carbsg', 1, 0))
  forest_model[[k]]$protein <- sum(ifelse(final_model[[k]]$term == 'Proteing', 1, 0))
  forest_model[[k]]$vitmin <- sum(ifelse(final_model[[k]]$term == 'VitsMinerals', 1, 0))

  forest_model[[k]]$intercept_VI <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  forest_model[[k]]$cal_VI <- sum(ifelse(weights[[k]]$Predictor == 'Calories',
weights[[k]]$Mean, 0))
```

```
  forest_model[[k]]$calf_VI <- sum(ifelse(weights[[k]]$Predictor == 'Calories_Fat',
weights[[k]]$Mean, 0))
  forest_model[[k]]$tfat_VI <- sum(ifelse(weights[[k]]$Predictor == 'Total_Fatg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$sfat_VI <- sum(ifelse(weights[[k]]$Predictor == 'Saturated_Fatg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$pfat_VI <- sum(ifelse(weights[[k]]$Predictor == 'Poly_Fatg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$mfat_VI <- sum(ifelse(weights[[k]]$Predictor == 'Mono_Fatg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$sod_VI <- sum(ifelse(weights[[k]]$Predictor == 'Sodiummg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$potas_VI <- sum(ifelse(weights[[k]]$Predictor == 'Potassiummg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$dfiber_VI <- sum(ifelse(weights[[k]]$Predictor == 'DietaryFiberg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$sfiber_VI <- sum(ifelse(weights[[k]]$Predictor == 'Soluble_Fiberg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$ifiber_VI <- sum(ifelse(weights[[k]]$Predictor == 'Insoluble_Fiberg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$sugar_VI <- sum(ifelse(weights[[k]]$Predictor == 'Sugars_g',
weights[[k]]$Mean, 0))
  forest_model[[k]]$tcarb_VI <- sum(ifelse(weights[[k]]$Predictor == 'TotalCarbsg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$ocarb_VI <- sum(ifelse(weights[[k]]$Predictor == 'Other_Carbsg',
weights[[k]]$Mean, 0))
  forest_model[[k]]$protein_VI <- sum(ifelse(weights[[k]]$Predictor == 'Proteing',
weights[[k]]$Mean, 0))
  forest_model[[k]]$vitmin_VI <- sum(ifelse(weights[[k]]$Predictor == 'VitsMinerals',
weights[[k]]$Mean, 0))

}

R_model_stats <- map_df(forest_model, ~as.data.frame(.x), .id="UniqueID")

Conditions <- subset(SDComplete, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "Beta.Condition", "bWeight")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

R_model_stats <- subset(R_model_stats, select=-c(UniqueID, UniqueID.1))
R_model_stats <- merge(condition, R_model_stats, by = c('UniqueID'), all.x = TRUE)

R_model_stats

##SAVING RELEVANT DATA FILES:###
```

```
write.csv(R_model_stats, file = "C:/Users/Kristina/Documents/SIMULATED DATA Random Forest Model Stats.csv")
```

Multilevel Model Bayesian Analysis

```
############################################################
#PLEASE NOTE: TO RUN brms YOU WILL NEED A C++ COMPILER; THE Rtools PROGRAM COMES WITH A C++ COMPILER
#AND IS AVAILABLE AT https://cran.r-project.org/bin/windows/Rtools/
#Rtools NEEDS TO BE INSTALLED ACCORDING TO SPECIFICATIONS OUTLINED HERE: https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-Windows
#OTHERWISE THE FOLLOWING CODE WILL NOT WORK

#AFTER INSTALLING Rtools, CODE FOR FINALIZING STAN INSTALLATION
install.packages("pkgbuild")
pkgbuild::has_build_tools(debug = TRUE)
fx <- inline::cxxfunction( signature(x = "integer", y = "numeric" ) , '
                    return ScalarReal( INTEGER(x)[0] * REAL(y)[0] ) ;
                    ' )
fx( 2L, 5 ) # should be 10
#If this returns anything other than 10, then go back to the previous section and install Rtools correctly.

dotR <- file.path(Sys.getenv("HOME"), ".R")
if (!file.exists(dotR))
  dir.create(dotR)
M <- file.path(dotR, "Makevars")
if (!file.exists(M))
  file.create(M)
cat("\nCXX14FLAGS=-O3 -Wno-unused-variable -Wno-unused-function",
    "CXX14 = $(BINPREF)g++ -m$(WIN) -std=c++1y",
    "CXX11FLAGS=-O3 -Wno-unused-variable -Wno-unused-function",
    file = M, sep = "\n", append = TRUE)

remove.packages("rstan")
if (file.exists(".RData")) file.remove(".RData")
install.packages("rstan", repos = "https://cloud.r-project.org/", dependencies = TRUE)

############################################################
###############PREPPING DATA FILE, IF YOU HAVE UNPREPPED DATA FILES, USE THIS################
#Loading Condition Data
Sim <- read.csv(file="C:/Users/Kristina/Documents/SIM Sampling Distribution Data.csv", header=TRUE, sep=",")
```

```
dfVar <- c("UniqueID", "RecordID", "Beta.Condition")
Conditions <- (Sim[, dfVar])
Conditions = subset(Conditions, !duplicated(UniqueID))

#Loading Data Files
SIM_Stepwise<-
read.csv(file="C:/Users/Kristina/Documents/SIMfinal_stepwise_model.csv",
header=TRUE, sep=",")
SIM_All_Subsets<- read.csv(file="C:/Users/Kristina/Documents/All IDs All Subsets
SIM Model Outcomes.csv", header=TRUE, sep=",")
SIM_Random_Forest<- read.csv(file="C:/Users/Kristina/Documents/SIMULATED
DATA Random Forest Model Stats.csv", header=TRUE, sep=",")

SIM_Stepwise$Step_Method <- 1
SIM_Stepwise$AS_Method <- 0
SIM_Stepwise$RF_Method <- 0
SIM_Stepwise$Method1 <- "3Stepwise"
SIM_Stepwise$Method2 <- "1Stepwise"

SIM_All_Subsets$Step_Method <- 0
SIM_All_Subsets$AS_Method <- 1
SIM_All_Subsets$RF_Method <- 0
SIM_All_Subsets$Method1 <- "2All_Subsets"
SIM_All_Subsets$Method2 <- "2All_Subsets"

SIM_Random_Forest$Step_Method <- 0
SIM_Random_Forest$AS_Method <- 0
SIM_Random_Forest$RF_Method <- 1
SIM_Random_Forest$Method1 <- "1Random_Forest"
SIM_Random_Forest$Method2 <- "3Random_Forest"

SIM_Stepwise$adjR2_model <- SIM_Stepwise$adj.r.squared
SIM_Random_Forest$adjR2_model <- SIM_Random_Forest$adjrR2
SIM_All_Subsets$adjR2_model <- SIM_All_Subsets$mean_adjR2

dfVar <- c("UniqueID", "Method1", "Method2",
"Step_Method","AS_Method","RF_Method", "model_var", "cal", "calf", "tfat", "sfat",
"pfat", "mfat", "sod", "potas", "dfiber", "sfiber", "ifiber", "sugar", "tcarb", "ocarb",
"protein", "vitmin")

SIM_Stepwise1 <- (SIM_Stepwise[, dfVar])
SIM_All_Subsets1 <- (SIM_All_Subsets[, dfVar])
SIM_Random_Forest1 <- (SIM_Random_Forest[, dfVar])

SIM_All_Models <- rbind(SIM_Stepwise1, SIM_All_Subsets1, SIM_Random_Forest1)
SIM_All_Models <- merge(SIM_All_Models, Conditions, by = "UniqueID")
```

```
SIM_All_Models$Beta.Condition1a <- ifelse(SIM_All_Models$Beta.Condition == 1,
"1SIM1","2SIM2")
SIM_All_Models$Beta.Condition1a
SIM_All_Models$Method1

rm(list=(ls()[ls()!="SIM_All_Models"]))

write.csv(SIM_All_Models, file = "C:/Users/Kristina/Documents/SIM Merged Methods
Model Outcomes.csv")

SIM_All_Models1 <- SIM_All_Models[complete.cases(SIM_All_Models),]

#####################################################################
######IF YOU HAVE ALL MODELS FILE, LOAD FILE HERE############
SIM_All_Models <- read.csv(file="C:/Users/Kristina/Documents/SIM Merged Methods
Model Outcomes.csv", header=TRUE, sep=",")

#SIM_All_Models1 <- SIM_All_Models[complete.cases(SIM_All_Models),]
#####################################################################
rm(list=(ls()[ls()!="SIM_All_Models"]))

#####################################################################
library(rstan)
library(brms)
library(dplyr)

rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
#############################################################
set.seed(1206)

cor.test(SIM_All_Models$cal, SIM_All_Models$sod)
cor.test(SIM_All_Models$cal, SIM_All_Models$sugar)
cor.test(SIM_All_Models$cal, SIM_All_Models$dfiber)
cor.test(SIM_All_Models$sod, SIM_All_Models$sugar)
cor.test(SIM_All_Models$sod, SIM_All_Models$dfiber)
cor.test(SIM_All_Models$sugar, SIM_All_Models$dfiber)


### Intercept Only Model: Model1 ###
t1a <- Sys.time()
SIMfit1a <- brm(cal ~ (1|UniqueID),
            data = SIM_All_Models,
            family = bernoulli(),
            iter = 30000,
```

```
            warmup = 2000,
            chains = 4,
            cores = 4,
            save_all_pars = TRUE,
            control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1a <- (Sys.time()-t1a))

saveRDS(SIMfit1a, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIM fit1a.rds")

rm("SIMfit1a")

t1b <- Sys.time()
SIMfit1b <- brm(sod ~ (1|UniqueID),
            data = SIM_All_Models,
            family = bernoulli(),
            iter = 30000,
            warmup = 2000,
            chains = 4,
            cores = 4,
            save_all_pars = TRUE,
            control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1b <- (Sys.time()-t1b))

saveRDS(SIMfit1b, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIM fit1b.rds")

rm("SIMfit1b")

t1c <- Sys.time()
SIMfit1c <- brm(sugar ~ (1|UniqueID),
            data = SIM_All_Models,
            family = bernoulli(),
            iter = 30000,
            warmup = 2000,
            chains = 4,
            cores = 4,
            save_all_pars = TRUE,
            control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1c <- (Sys.time()-t1c))

saveRDS(SIMfit1c, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIM fit1c.rds")

rm("SIMfit1c")
```

```
t1d <- Sys.time()
SIMfit1d <- brm(dfiber ~ (1|UniqueID),
                data = SIM_All_Models,
                family = bernoulli(),
                iter = 30000,
                warmup = 2000,
                chains = 4,
                cores = 4,
                save_all_pars = TRUE,
                control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1d <- (Sys.time()-t1d))

saveRDS(SIMfit1d, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIM fit1d.rds")

rm("SIMfit1d")


### Intercept Only Model: Model1 ###
t1 <- Sys.time()
SIMfit1 <- brm(cbind(cal, sod, sugar, dfiber) ~ (1|p|UniqueID),
               data = SIM_All_Models,
               family = bernoulli(),
               iter = 30000,
               warmup = 2000,
               chains = 4,
               cores = 4,
               save_all_pars = TRUE,
               control = list(adapt_delta = .99, max_treedepth = 15))
(tfit1 <- (Sys.time()-t1))

saveRDS(SIMfit1, file = "C:/Users/Kristina/Documents/SIM Overall Analysis/SIM Stan
Multivariate Version 1/SIM fit1.rds")

rm("SIMfit1")


### Model 2 Fixed Effects: Level-1 Predictors ###
### Random Forest Reference Group:
t2a <- Sys.time()
SIMfit2a <- brm(cbind(cal, sod, sugar, dfiber) ~ Method1 + (1|p|UniqueID),
              data = SIM_All_Models,
              family = bernoulli(),
              iter = 30000,
              warmup = 2000,
```

```
        chains = 4,
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit2a <- (Sys.time()-t2a))

saveRDS(SIMfit2a, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIM fit2a.rds")

rm("SIMfit2a")

### Stepwise Reference Group:
t2b <- Sys.time()
SIMfit2b <- brm(cbind(cal, sod, sugar, dfiber) ~ Method2 + (1|p|UniqueID),
        data = SIM_All_Models,
        family = bernoulli(),
        iter = 30000,
        warmup = 2000,
        chains = 4,
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit2b <- (Sys.time()-t2b))

saveRDS(SIMfit2b, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIM fit2b.rds")

rm("SIMfit2b")


### Model 3 Fixed Effects: Level-2 Predictors ###
### Fixed Effects: Level-2 Predictors (b = .01 Reference) ###
#Random Forest Reference
t3a <- Sys.time()
SIMfit3a <- brm(cbind(cal, sod, sugar, dfiber) ~ Method1 + Beta.Condition1a +
(1|p|UniqueID),
        data = SIM_All_Models,
        family = bernoulli(),
        iter = 30000,
        warmup = 2000,
        chains = 4,
        cores = 4,
        save_all_pars = TRUE,
        control = list(adapt_delta = .99, max_treedepth = 15))
(tfit3a <- (Sys.time()-t3a))
```

```
saveRDS(SIMfit3a, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIMfit3a.rds")

rm("SIMfit3a")


### Fixed Effects: Level-2 Predictors (b = .01 Reference) ###
#Stepwise Reference
t3b<- Sys.time()
SIMfit3b <- brm(cbind(cal, sod, sugar, dfiber) ~ Method2 + Beta.Condition1a +
(1|p|UniqueID),
            data = SIM_All_Models,
            family = bernoulli(),
            iter = 30000,
            warmup = 2000,
            chains = 4,
            cores = 4,
            save_all_pars = TRUE,
            control = list(adapt_delta = .99, max_treedepth = 15))
(tfit3b <- (Sys.time()-t3b))

saveRDS(SIMfit3b, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIMfit3b.rds")

rm("SIMfit3b")

set.seed(252019)

### Model 4 Fixed-Effect Interaction: (b = .01 Reference) ###
#Random Forest Reference
t4a <- Sys.time()
SIMfit4a <- brm(cbind(cal, sod, sugar, dfiber) ~ Method1 + Beta.Condition1a +
Method1*Beta.Condition1a + (1|p|UniqueID),
            data = SIM_All_Models,
            family = bernoulli(),
            iter = 30000,
            warmup = 2000,
            chains = 4,
            cores = 4,
            save_all_pars = TRUE,
            control = list(adapt_delta = .99, max_treedepth = 16))
(tfit4a <- (Sys.time()-t4a))

#PC1
```

```
saveRDS(SIMfit5a, file = "C:/Users/Kristina/Documents/SIM Stan Multivariate Version
1/SIMfit5a.rds")

rm("SIMfit4a")


set.seed(272019)

### Model 4 Fixed-Effect Interaction: (b = .01 Reference) ###
#Stepwise Reference
t4b<- Sys.time()
SIMfit4b <- brm(cbind(cal, sod, sugar, dfiber) ~ Method2 + Beta.Condition1a +
Method2*Beta.Condition1a + (1|p|UniqueID),
            data = SIM_All_Models,
            family = bernoulli(),
            iter = 30000,
            warmup = 2000,
            chains = 4,
            cores = 4,
            save_all_pars = TRUE,
            control = list(adapt_delta = .99, max_treedepth = 16))
(tfit4b <- (Sys.time()-t4b))

saveRDS(SIMfit5b, file = "C:/Users/Kristina/Documents/Ohio University/Dissertation
9.19.17/R Code/Study One Analysis/SIM Overall Analysis/SIM Stan Multivariate
Version 1/SIMfit5b.rds")

rm("SIMfit4b")
```

Study Three Analyses R Code

Simulating Data for Correlation Structure A

```
rm(list = ls())
library(lattice)
library(car)

nobs = 50

#####SET SEED ONLY ONCE INITIALLY###########
set.seed(2911)
###################################

#######CREATE CORRELATION MATRICES#############
#Simulation Matrix A
```

```
A1 = matrix(c(1, 0.1, 0.3, 0.5, 0.7,
              0.1, 1, 0.75, 0.75, 0.75,
              0.3, 0.75, 1, 0.75, 0.75,
              0.5, 0.75, 0.75, 1, 0.75,
              0.7, 0.75, 0.75, 0.75, 1), nrow=5, ncol=5)

A2 = matrix(c(1, 0.1, 0.3, 0.5, 0.7,
              0.1, 1, 0.5, 0.5, 0.5,
              0.3, 0.5, 1, 0.5, 0.5,
              0.5, 0.5, 0.5, 1, 0.5,
              0.7, 0.5, 0.5, 0.5, 1), nrow=5, ncol=5)

A3 = matrix(c(1, 0.1, 0.3, 0.5, 0.7,
              0.1, 1, 0.25, 0.25, 0.25,
              0.3, 0.25, 1, 0.25, 0.25,
              0.5, 0.25, 0.25, 1, 0.25,
              0.7, 0.25, 0.25, 0.25, 1), nrow=5, ncol=5)

A4 = matrix(c(1, 0.1, 0.3, 0.5, 0.7,
              0.1, 1, 0.0, 0.0, 0.0,
              0.3, 0.0, 1, 0.0, 0.0,
              0.5, 0.0, 0.0, 1, 0.0,
              0.7, 0.0, 0.0, 0.0, 1), nrow=5, ncol=5)

############################################################
#####Rerun Below Until Break for 40 Replications, Renaming and Saving Between Each
Run#################

# Cholesky decomposition

#Correlational Structure A1
LA1 = chol(A1)
nvars = dim(LA1)[1]
t(LA1)
t(LA1) %*% LA1

rA1 = t(LA1) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rA1 = t(rA1)

A1data = as.data.frame(rA1)
names(A1data) = c('Y', 'X1', 'X2', 'X3', 'X4')
A1data$rStructure <- "A1data"
A1data$UniqueID <- 1

#Correlational Structure A2
LA2 = chol(A2)
```

```
nvars = dim(LA2)[1]
t(LA2)
t(LA2) %*% LA2

rA2 = t(LA2) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rA2 = t(rA2)

A2data = as.data.frame(rA2)
names(A2data) = c('Y', 'X1', 'X2', 'X3', 'X4')
A2data$rStructure <- "A2data"
A2data$UniqueID <- 2

#Correlational Structure A3
LA3 = chol(A3)
nvars = dim(LA3)[1]
t(LA3)
t(LA3) %*% LA3

rA3 = t(LA3) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rA3 = t(rA3)

A3data = as.data.frame(rA3)
names(A3data) = c('Y', 'X1', 'X2', 'X3', 'X4')
A3data$rStructure <- "A3data"
A3data$UniqueID <- 3

#Correlational Structure A4
LA4 = chol(A4)
nvars = dim(LA4)[1]
t(LA4)
t(LA4) %*% LA4

rA4 = t(LA4) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rA4 = t(rA4)

A4data = as.data.frame(rA4)
names(A4data) = c('Y', 'X1', 'X2', 'X3', 'X4')
A4data$rStructure <- "A4data"
A4data$UniqueID <- 4

##############################################
cor(A1data[1:5])
cor(A2data[1:5])
cor(A3data[1:5])
cor(A4data[1:5])
```

```
#########################################################################
####UPDATE NAMES FOR EACH REPLICATION HERE AND SAVE BEFORE
RERUNNING###############

ACorrSim1 <- rbind(A1data, A2data, A3data, A4data)

write.csv(ACorrSim1, file = "C:/Users/Kristina/Documents/ACorrSim40.csv")
```

Simulating Data for Correlation Structure B

```
rm(list = ls())
library(lattice)
library(car)

#####SET SEED ONLY ONCE INITIALLY###########
set.seed(121)
#####################################

#######CREATE CORRELATION MATRICES#############
#####Rerun Below Until Break for 40 Replications, Renaming and Saving Between Each
Run#################
#Simulation Matrix B
B1 = matrix(c(1, 0.6, 0.0, 0.0,
              0.6, 1, 0.0, 0.0,
              0.0, 0.0, 1, 0.0,
              0.0, 0.0, 0.0, 1), nrow=4, ncol=4)

#Correlational Structure B1
nobs <- 50
LB1 = chol(B1)
nvars = dim(LB1)[1]
t(LB1)
t(LB1) %*% LB1

rB1 = t(LB1) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rB1 = t(rB1)

B1data = as.data.frame(rB1)
names(B1data) = c('Y', 'X1', 'X2', 'X3')
B1data$rStructure <- "B1data"
B1data$UniqueID <- 1


B2 = matrix(c(1, 0.6, 0.0, 0.0,
              0.6, 1, 0.0, ((0.2)^(1/2)),
              0.0, 0.0, 1, 0.0,
```

```
        0.0, ((0.2)^(1/2)), 0.0, 1), nrow=4, ncol=4)

#Correlational Structure B2

LB2 = chol(B2)
nvars = dim(LB2)[1]
nobs = 50
t(LB2)
t(LB2) %*% LB2

rB2 = t(LB2) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rB2 = t(rB2)

B2data = as.data.frame(rB2)
names(B2data) = c('Y', 'X1', 'X2', 'X3')

B2data$rStructure <- "B2data"
B2data$UniqueID <- 2


B3 = matrix(c(1, 0.6, 0.0, 0.0,
        0.6, 1, ((0.1)^(1/2)), ((0.1)^(1/2)),
        0.0, ((0.1)^(1/2)), 1, 0.0,
        0.0, ((0.1)^(1/2)), 0.0, 1), nrow=4, ncol=4)

#Correlational Structure B3
nobs <- 50
LB3 = chol(B3)
nvars = dim(LB3)[1]
t(LB3)
t(LB3) %*% LB3

rB3 = t(LB3) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rB3 = t(rB3)

B3data = as.data.frame(rB3)
names(B3data) = c('Y', 'X1', 'X2', 'X3')

B3data$rStructure <- "B3data"
B3data$UniqueID <- 3

B4 = matrix(c(1, 0.6, 0.0, 0.0,
        0.6, 1, 0.0, ((0.4)^(1/2)),
        0.0, 0.0, 1, 0.0,
        0.0, ((0.4)^(1/2)), 0.0, 1), nrow=4, ncol=4)
```

```
#Correlational Structure B4
nobs <- 50
LB4 = chol(B4)
nvars = dim(LB4)[1]
t(LB4)
t(LB4) %*% LB4

rB4 = t(LB4) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rB4 = t(rB4)

B4data = as.data.frame(rB4)
names(B4data) = c('Y', 'X1', 'X2', 'X3')

B4data$rStructure <- "B4data"
B4data$UniqueID <- 4

B5 = matrix(c(1, 0.6, 0.0, 0.0,
              0.6, 1, ((0.2)^(1/2)), ((0.2)^(1/2)),
              0.0, ((0.2)^(1/2)), 1, 0.0,
              0.0, ((0.2)^(1/2)), 0.0, 1), nrow=4, ncol=4)

#Correlational Structure B5
nobs <- 50
LB5 = chol(B5)
nvars = dim(LB5)[1]
t(LB5)
t(LB5) %*% LB5

rB5 = t(LB5) %*% matrix(rnorm(nvars*nobs), nrow=nvars, ncol=nobs)
rB5 = t(rB5)

B5data = as.data.frame(rB5)
names(B5data) = c('Y', 'X1', 'X2', 'X3')

B5data$rStructure <- "B5data"
B5data$UniqueID <- 5


############################################################
cor(B1data[1:4])
cor(B2data[1:4])
cor(B3data[1:4])
cor(B4data[1:4])
cor(B5data[1:4])
#############################################################################
```

```
####UPDATE NAMES FOR EACH REPLICATION HERE AND SAVE BEFORE
RERUNNING ABOVE################
BCorrSim1 <- rbind(B1data, B2data, B3data, B4data, B5data)
BCorrSim1

write.csv(BCorrSim1, file = "C:/Users/Kristina/Documents/BCorrSim40.csv")
```

Stepwise Regression Analysis Data Structure A

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("lm.beta")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(lm.beta)
library(data.table)


#Data from Personal PC: PC1
ACorrSim1 <-
read.csv(file="C:/Users/Kristina/Documents/CorrSimData/ACorrSim1.csv",
header=TRUE, sep=",")
ACorrSim2 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim2.csv",
header=TRUE, sep=",")
ACorrSim3 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim3.csv",
header=TRUE, sep=",")
ACorrSim4 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim4.csv",
header=TRUE, sep=",")
ACorrSim5 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim5.csv",
header=TRUE, sep=",")
ACorrSim6 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim6.csv",
header=TRUE, sep=",")
```

```
ACorrSim7 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim7.csv",
header=TRUE, sep=",")
ACorrSim8 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim8.csv",
header=TRUE, sep=",")
ACorrSim9 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim9.csv",
header=TRUE, sep=",")
ACorrSim10 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim10.csv",
header=TRUE, sep=",")

ACorrSim11 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim11.csv",
header=TRUE, sep=",")
ACorrSim12 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim12.csv",
header=TRUE, sep=",")
ACorrSim13 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim13.csv",
header=TRUE, sep=",")
ACorrSim14 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim14.csv",
header=TRUE, sep=",")
ACorrSim15 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim15.csv",
header=TRUE, sep=",")
ACorrSim16 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim16.csv",
header=TRUE, sep=",")
ACorrSim17 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim17.csv",
header=TRUE, sep=",")
ACorrSim18 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim18.csv",
header=TRUE, sep=",")
ACorrSim19 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim19.csv",
header=TRUE, sep=",")
ACorrSim20 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim20.csv",
header=TRUE, sep=",")

ACorrSim21 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim21.csv",
header=TRUE, sep=",")
ACorrSim22 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim22.csv",
header=TRUE, sep=",")
ACorrSim23 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim23.csv",
header=TRUE, sep=",")
ACorrSim24 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim24.csv",
header=TRUE, sep=",")
ACorrSim25 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim25.csv",
header=TRUE, sep=",")
ACorrSim26 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim26.csv",
header=TRUE, sep=",")
ACorrSim27 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim27.csv",
header=TRUE, sep=",")
ACorrSim28 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim28.csv",
header=TRUE, sep=",")
```

```
ACorrSim29 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim29.csv",
header=TRUE, sep=",")
ACorrSim30 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim30.csv",
header=TRUE, sep=",")

ACorrSim31 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim31.csv",
header=TRUE, sep=",")
ACorrSim32 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim32.csv",
header=TRUE, sep=",")
ACorrSim33 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim33.csv",
header=TRUE, sep=",")
ACorrSim34 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim34.csv",
header=TRUE, sep=",")
ACorrSim35 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim35.csv",
header=TRUE, sep=",")
ACorrSim36 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim36.csv",
header=TRUE, sep=",")
ACorrSim37 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim37.csv",
header=TRUE, sep=",")
ACorrSim38 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim38.csv",
header=TRUE, sep=",")
ACorrSim39 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim39.csv",
header=TRUE, sep=",")
ACorrSim40 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim40.csv",
header=TRUE, sep=",")

####SIMULATED DATA STEPWISE ANALYSIS###############
step_analysis <- function(data, CorrSIMGroup){

   datagroup <- CorrSIMGroup
   count<-length(unique(data$UniqueID))

  datalist <- list()
  full_model <- list()
  null_model <- list()
  stepwise <- list()
  summaries <- list()
  list_names <- list()
  df_coefficients <- list()
  df_betas <- list()
  df_fstat <- list()
  scale_full_model <- list()
  scale_null_model <- list()
  scale_stepwise <- list()

t1 <- Sys.time()
```

```
for(i in 1:count){
  datalist[[i]] <- as.data.frame(subset(data, UniqueID == i))

  full_model[[i]] <- lm(Y ~ X1 + X2 + X3 + X4,
                        data = datalist[[i]])

  null_model[[i]] <- lm(Y ~ 1, data = datalist[[i]])

  stepwise[[i]] <- step(null_model[[i]], scope = list(upper = full_model[[i]]))

  scale_full_model[[i]] <- lm(scale(Y) ~ scale(X1) + scale(X2) + scale(X3) + scale(X4),
                              data = datalist[[i]])

  scale_null_model[[i]] <- lm(scale(Y) ~ 1, data = datalist[[i]])

  scale_stepwise[[i]] <- step(scale_null_model[[i]], scope = list(upper =
scale_full_model[[i]]))

  summaries[[i]] <- summary(stepwise[[i]])

  require(broom)
  df_coefficients[[i]] <- tidy(stepwise[[i]])
  df_betas[[i]] <- tidy(scale_stepwise[[i]])
  df_fstat[[i]] <- as.data.frame((summaries[[i]][["r.squared"]]))
  colnames(df_fstat[[i]]) <- "r.squared"
  df_fstat[[i]]$value <- ifelse(is_empty(summaries[[i]]$fstatistic[[1]]) == TRUE, 0,
summaries[[i]]$fstatistic[[1]])
  df_fstat[[i]]$numdf <- ifelse(is_empty(summaries[[i]]$fstatistic[[2]]) == TRUE, 0,
summaries[[i]]$fstatistic[[2]])
  df_fstat[[i]]$dendf <- ifelse(is_empty(summaries[[i]]$fstatistic[[3]]) == TRUE, 0,
summaries[[i]]$fstatistic[[3]])

}
print(totalanalysis <- (Sys.time()-t1))

df_coefficients[[1]]
##############Model Statistics#############################
#Adjusted R^2#
model_stats <- rbind(lapply(summaries, `[`, 9))
model <- map_df(model_stats, ~as.data.frame(.x), .id="UniqueID")

#F-Statistics#
fstats <- bind_rows(df_fstat, .id = 'UniqueID')
fstats$fstat <- fstats$value
fstatVar <- c("UniqueID", "fstat", "numdf", "dendf")
fstatistics <- fstats[complete.cases(fstats[, fstatVar]), fstatVar]
```

```
#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_model <- merge(model, fstatistics, by = c('UniqueID'), all.x = TRUE)


#Computing p-values, assigning binary significance values
stepwise_model$pvalue<- pf(stepwise_model$fstat, stepwise_model$numdf,
stepwise_model$dendf, lower.tail = FALSE)
stepwise_model$modelsig <- ifelse(stepwise_model$pvalue < .05, 1, 0)

#####################################################################
#Creating dataframe for unstandardized coefficient estimates#
coefficients <- bind_rows(df_coefficients, .id = 'UniqueID')

#Creating mergeable dataframe for standardized coefficient estimates#
betas_mid <- bind_rows(df_betas, .id = 'UniqueID')
betas_mid
betas_mid$term2 <- gsub("scale\\(", "", betas_mid$term)
betas_mid$term3 <- gsub("[)]", "", betas_mid$term2)
betas_mid$term <- gsub("[(]Intercept", "(Intercept)", betas_mid$term3)

betas_mid$beta <- betas_mid$estimate

betaVar <- c("UniqueID", "term", "beta")
betas <- betas_mid[complete.cases(betas_mid[, betaVar]), betaVar]
betas$beta_value <- ifelse(betas$term == '(Intercept)', NA, betas$beta)

#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_coeff <- merge(coefficients, betas, by = c('UniqueID', 'term'))

######Assigning Meta Values################
#Significance values#
stepwise_coeff$coeffsig <- ifelse(stepwise_coeff$p.value < .05, 1, 0)
#Positive/negative betas#
stepwise_coeff$coeffdirection <- ifelse(stepwise_coeff$beta > 0, 'positive', 'negative')

#Ranking values#
coeff_data <- list()
for(i in 1:count){
  coeff_data[[i]] <- as.data.frame(subset(stepwise_coeff, UniqueID == i))
  coeff_data[[i]]$rank <- rank(-abs(coeff_data[[i]]$beta_value), na.last = FALSE,
ties.method = "average")
  coeff_data[[i]]$adjrank <- coeff_data[[i]]$rank - 1
  final_stepwise_coeff <- bind_rows(coeff_data)
}
```

```
#Adding Predictor Count to File#
final_model <- as.data.frame(subset(final_stepwise_coeff))
predictors <- c("UniqueID", "term")

CORSIMfinal_model_predictors <- final_model[complete.cases(final_model[,
predictors]), predictors]

stepmodel <- list()
predictorlist <- list()
for(i in 1:count){
  stepmodel[[i]] <- as.data.frame(subset(stepwise_model, UniqueID == i))
  predictorlist[[i]] <- as.data.frame(subset(CORSIMfinal_model_predictors, UniqueID ==
i))
  predictorlist[[i]]$Count <- (nrow(predictorlist[[i]]))-1
  stepmodel[[i]]$model_var <- (nrow(predictorlist[[i]]))-1
}

stepwise_model <- map_df(stepmodel, ~as.data.frame(.x), .id="UniqueID")
final_model_predictors <- map_df(predictorlist, ~as.data.frame(.x), .id="UniqueID")

#########################################################################

modelVar <- c("UniqueID", "term", "estimate", "beta")

model_estimates <- final_stepwise_coeff[complete.cases(final_stepwise_coeff[,
modelVar]), modelVar]

person_model <- list()
term_model <- list()
step_model <- list()
weights <- list()
stdweights <- list()

for(k in 1:count){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, UniqueID == k))
  step_model[[k]] <- as.data.frame(subset(stepwise_model, UniqueID == k))
  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  stdweights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$beta)))
  setDT(stdweights[[k]], keep.rownames = TRUE)[]
```

```
  setnames(stdweights[[k]], 1, "Predictor")
  setnames(stdweights[[k]], 2, "Mean")

  step_model[[k]]$intercept <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)', 1, 0))
  step_model[[k]]$X1 <- sum(ifelse(weights[[k]]$Predictor == 'X1', 1, 0))
  step_model[[k]]$X2 <- sum(ifelse(weights[[k]]$Predictor == 'X2', 1, 0))
  step_model[[k]]$X3 <- sum(ifelse(weights[[k]]$Predictor == 'X3', 1, 0))
  step_model[[k]]$X4 <- sum(ifelse(weights[[k]]$Predictor == 'X4', 1, 0))

  step_model[[k]]$intercept_b <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  step_model[[k]]$X1_b <- sum(ifelse(weights[[k]]$Predictor == 'X1',
weights[[k]]$Mean, 0))
  step_model[[k]]$X2_b <- sum(ifelse(weights[[k]]$Predictor == 'X2',
weights[[k]]$Mean, 0))
  step_model[[k]]$X3_b <- sum(ifelse(weights[[k]]$Predictor == 'X3',
weights[[k]]$Mean, 0))
  step_model[[k]]$X4_b <- sum(ifelse(weights[[k]]$Predictor == 'X4',
weights[[k]]$Mean, 0))

  step_model[[k]]$intercept_beta <- sum(ifelse(stdweights[[k]]$Predictor == '(Intercept)',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X1_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X1',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X2_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X2',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X3_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X3',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X4_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X4',
stdweights[[k]]$Mean, 0))

 }

stepwise_model <- map_df(step_model, ~as.data.frame(.x), .id="NumberID")

Conditions <- subset(data, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "rStructure")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

CORSIMfinal_stepwise_coeff <- merge(condition, final_stepwise_coeff, by =
c('UniqueID'), all.x = TRUE)
CORSIMstepwise_model <- merge(condition, stepwise_model, by = c('UniqueID'), all.x
= TRUE)

##SAVING ON PERSONAL PC: PC1###
```

```
write.csv(CORSIMfinal_stepwise_coeff, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, "
CORSIMfinal_stepwise_coeff.csv", sep = ""))
write.csv(CORSIMstepwise_model, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, "
CORSIMstepwise_model.csv", sep = ""))
write.csv(CORSIMfinal_model_predictors, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, "
CORSIMfinal_model_predictors.csv", sep = ""))

}

##############################################################################
##################RUN STEPWISE ANALYSIS FUNCTION##################
#Set seed run function with different datasets

sets <- list(ACorrSim1, ACorrSim2, ACorrSim3, ACorrSim4, ACorrSim5, ACorrSim6,
ACorrSim7, ACorrSim8, ACorrSim9, ACorrSim10, ACorrSim11, ACorrSim12,
ACorrSim13, ACorrSim14, ACorrSim15, ACorrSim16, ACorrSim17, ACorrSim18,
ACorrSim19, ACorrSim20, ACorrSim21, ACorrSim22, ACorrSim23, ACorrSim24,
ACorrSim25, ACorrSim26, ACorrSim27, ACorrSim28, ACorrSim29, ACorrSim30,
ACorrSim31, ACorrSim32, ACorrSim33, ACorrSim34, ACorrSim35, ACorrSim36,
ACorrSim37, ACorrSim38, ACorrSim39, ACorrSim40)

sets_num <- list(1, 2, 3, 4, 5,  6,  7,  8,  9, 10,
          11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
          21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
          31, 32, 33, 34, 35, 36, 37, 38, 39, 40)

set.seed(1015)
tfunc <- Sys.time()
mapply(step_analysis, sets, sets_num)
print(totalanalysis <- (Sys.time()-tfunc))
```

Stepwise Regression Analysis Data Structure B

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("lm.beta")

rm(list=ls())
```

```
library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(lm.beta)
library(data.table)

#Data from Personal PC: PC1
BCorrSim1 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim1.csv",
header=TRUE, sep=",")
BCorrSim2 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim2.csv",
header=TRUE, sep=",")
BCorrSim3 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim3.csv",
header=TRUE, sep=",")
BCorrSim4 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim4.csv",
header=TRUE, sep=",")
BCorrSim5 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim5.csv",
header=TRUE, sep=",")
BCorrSim6 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim6.csv",
header=TRUE, sep=",")
BCorrSim7 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim7.csv",
header=TRUE, sep=",")
BCorrSim8 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim8.csv",
header=TRUE, sep=",")
BCorrSim9 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim9.csv",
header=TRUE, sep=",")
BCorrSim10 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim10.csv",
header=TRUE, sep=",")

BCorrSim11 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim11.csv",
header=TRUE, sep=",")
BCorrSim12 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim12.csv",
header=TRUE, sep=",")
BCorrSim13 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim13.csv",
header=TRUE, sep=",")
BCorrSim14 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim14.csv",
header=TRUE, sep=",")
BCorrSim15 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim15.csv",
header=TRUE, sep=",")
BCorrSim16 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim16.csv",
header=TRUE, sep=",")
BCorrSim17 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim17.csv",
header=TRUE, sep=",")
```

```
BCorrSim18 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim18.csv",
header=TRUE, sep=",")
BCorrSim19 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim19.csv",
header=TRUE, sep=",")
BCorrSim20 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim20.csv",
header=TRUE, sep=",")

BCorrSim21 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim21.csv",
header=TRUE, sep=",")
BCorrSim22 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim22.csv",
header=TRUE, sep=",")
BCorrSim23 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim23.csv",
header=TRUE, sep=",")
BCorrSim24 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim24.csv",
header=TRUE, sep=",")
BCorrSim25 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim25.csv",
header=TRUE, sep=",")
BCorrSim26 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim26.csv",
header=TRUE, sep=",")
BCorrSim27 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim27.csv",
header=TRUE, sep=",")
BCorrSim28 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim28.csv",
header=TRUE, sep=",")
BCorrSim29 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim29.csv",
header=TRUE, sep=",")
BCorrSim30 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim30.csv",
header=TRUE, sep=",")

BCorrSim31 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim31.csv",
header=TRUE, sep=",")
BCorrSim32 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim32.csv",
header=TRUE, sep=",")
BCorrSim33 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim33.csv",
header=TRUE, sep=",")
BCorrSim34 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim34.csv",
header=TRUE, sep=",")
BCorrSim35 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim35.csv",
header=TRUE, sep=",")
BCorrSim36 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim36.csv",
header=TRUE, sep=",")
BCorrSim37 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim37.csv",
header=TRUE, sep=",")
BCorrSim38 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim38.csv",
header=TRUE, sep=",")
BCorrSim39 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim39.csv",
header=TRUE, sep=",")
```

```
BCorrSim40 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim40.csv",
header=TRUE, sep=",")

BCorrSim1 <- as.data.frame(BCorrSim1[with(BCorrSim1, order(UniqueID, Y)),])
BCorrSim2 <- as.data.frame(BCorrSim1[with(BCorrSim2, order(UniqueID, Y)),])
BCorrSim3 <- as.data.frame(BCorrSim1[with(BCorrSim3, order(UniqueID, Y)),])
BCorrSim4 <- as.data.frame(BCorrSim1[with(BCorrSim4, order(UniqueID, Y)),])
BCorrSim5 <- as.data.frame(BCorrSim1[with(BCorrSim5, order(UniqueID, Y)),])
BCorrSim6 <- as.data.frame(BCorrSim1[with(BCorrSim6, order(UniqueID, Y)),])
BCorrSim7 <- as.data.frame(BCorrSim1[with(BCorrSim7, order(UniqueID, Y)),])
BCorrSim8 <- as.data.frame(BCorrSim1[with(BCorrSim8, order(UniqueID, Y)),])
BCorrSim9 <- as.data.frame(BCorrSim1[with(BCorrSim9, order(UniqueID, Y)),])
BCorrSim10 <- as.data.frame(BCorrSim1[with(BCorrSim10, order(UniqueID, Y)),])

BCorrSim11 <- as.data.frame(BCorrSim1[with(BCorrSim11, order(UniqueID, Y)),])
BCorrSim12 <- as.data.frame(BCorrSim1[with(BCorrSim12, order(UniqueID, Y)),])
BCorrSim13 <- as.data.frame(BCorrSim1[with(BCorrSim13, order(UniqueID, Y)),])
BCorrSim14 <- as.data.frame(BCorrSim1[with(BCorrSim14, order(UniqueID, Y)),])
BCorrSim15 <- as.data.frame(BCorrSim1[with(BCorrSim15, order(UniqueID, Y)),])
BCorrSim16 <- as.data.frame(BCorrSim1[with(BCorrSim16, order(UniqueID, Y)),])
BCorrSim17 <- as.data.frame(BCorrSim1[with(BCorrSim17, order(UniqueID, Y)),])
BCorrSim18 <- as.data.frame(BCorrSim1[with(BCorrSim18, order(UniqueID, Y)),])
BCorrSim19 <- as.data.frame(BCorrSim1[with(BCorrSim19, order(UniqueID, Y)),])
BCorrSim20 <- as.data.frame(BCorrSim1[with(BCorrSim20, order(UniqueID, Y)),])

BCorrSim21 <- as.data.frame(BCorrSim1[with(BCorrSim21, order(UniqueID, Y)),])
BCorrSim22 <- as.data.frame(BCorrSim1[with(BCorrSim22, order(UniqueID, Y)),])
BCorrSim23 <- as.data.frame(BCorrSim1[with(BCorrSim23, order(UniqueID, Y)),])
BCorrSim24 <- as.data.frame(BCorrSim1[with(BCorrSim24, order(UniqueID, Y)),])
BCorrSim25 <- as.data.frame(BCorrSim1[with(BCorrSim25, order(UniqueID, Y)),])
BCorrSim26 <- as.data.frame(BCorrSim1[with(BCorrSim26, order(UniqueID, Y)),])
BCorrSim27 <- as.data.frame(BCorrSim1[with(BCorrSim27, order(UniqueID, Y)),])
BCorrSim28 <- as.data.frame(BCorrSim1[with(BCorrSim28, order(UniqueID, Y)),])
BCorrSim29 <- as.data.frame(BCorrSim1[with(BCorrSim29, order(UniqueID, Y)),])
BCorrSim30 <- as.data.frame(BCorrSim1[with(BCorrSim30, order(UniqueID, Y)),])

BCorrSim31 <- as.data.frame(BCorrSim1[with(BCorrSim31, order(UniqueID, Y)),])
BCorrSim32 <- as.data.frame(BCorrSim1[with(BCorrSim32, order(UniqueID, Y)),])
BCorrSim33 <- as.data.frame(BCorrSim1[with(BCorrSim33, order(UniqueID, Y)),])
BCorrSim34 <- as.data.frame(BCorrSim1[with(BCorrSim34, order(UniqueID, Y)),])
BCorrSim35 <- as.data.frame(BCorrSim1[with(BCorrSim35, order(UniqueID, Y)),])
BCorrSim36 <- as.data.frame(BCorrSim1[with(BCorrSim36, order(UniqueID, Y)),])
BCorrSim37 <- as.data.frame(BCorrSim1[with(BCorrSim37, order(UniqueID, Y)),])
BCorrSim38 <- as.data.frame(BCorrSim1[with(BCorrSim38, order(UniqueID, Y)),])
BCorrSim39 <- as.data.frame(BCorrSim1[with(BCorrSim39, order(UniqueID, Y)),])
BCorrSim40 <- as.data.frame(BCorrSim1[with(BCorrSim40, order(UniqueID, Y)),])
```

```
###################SIMULATED DATA STEPWISE ANALYSIS###############
step_analysis <- function(data, CorrSIMGroup){

   datagroup <- CorrSIMGroup
   count<-length(unique(data$UniqueID))

  datalist <- list()
  full_model <- list()
  null_model <- list()
  stepwise <- list()
  summaries <- list()
  list_names <- list()
  df_coefficients <- list()
  df_betas <- list()
  df_fstat <- list()
  scale_full_model <- list()
  scale_null_model <- list()
  scale_stepwise <- list()

t1 <- Sys.time()
for(i in 1:count){
  datalist[[i]] <- as.data.frame(subset(data, UniqueID == i))

  full_model[[i]] <- lm(Y ~ X1 + X2 + X3,
                        data = datalist[[i]])

  null_model[[i]] <- lm(Y ~ 1, data = datalist[[i]])

  stepwise[[i]] <- step(null_model[[i]], scope = list(upper = full_model[[i]]))

  scale_full_model[[i]] <- lm(scale(Y) ~ scale(X1) + scale(X2) + scale(X3),
                              data = datalist[[i]])

  scale_null_model[[i]] <- lm(scale(Y) ~ 1, data = datalist[[i]])

  scale_stepwise[[i]] <- step(scale_null_model[[i]], scope = list(upper =
scale_full_model[[i]]))

  summaries[[i]] <- summary(stepwise[[i]])

  require(broom)
  df_coefficients[[i]] <- tidy(stepwise[[i]])
  df_betas[[i]] <- tidy(scale_stepwise[[i]])
  df_fstat[[i]] <- as.data.frame((summaries[[i]][["r.squared"]]))
  colnames(df_fstat[[i]]) <- "r.squared"
```

```
  df_fstat[[i]]$value <- ifelse(is_empty(summaries[[i]]$fstatistic[[1]]) == TRUE, 0,
summaries[[i]]$fstatistic[[1]])
  df_fstat[[i]]$numdf <- ifelse(is_empty(summaries[[i]]$fstatistic[[2]]) == TRUE, 0,
summaries[[i]]$fstatistic[[2]])
  df_fstat[[i]]$dendf <- ifelse(is_empty(summaries[[i]]$fstatistic[[3]]) == TRUE, 0,
summaries[[i]]$fstatistic[[3]])

}
print(totalanalysis <- (Sys.time()-t1))

df_coefficients[[1]]

#############Model Statistics############################
#Adjusted R^2#
model_stats <- rbind(lapply(summaries, `[`, 9))
model <- map_df(model_stats, ~as.data.frame(.x), .id="UniqueID")

#F-Statistics#
fstats <- bind_rows(df_fstat, .id = 'UniqueID')
fstats$fstat <- fstats$value
fstatVar <- c("UniqueID", "fstat", "numdf", "dendf")
fstatistics <- fstats[complete.cases(fstats[, fstatVar]), fstatVar]

#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_model <- merge(model, fstatistics, by = c('UniqueID'), all.x = TRUE)

#Computing p-values, assigning binary significance values
stepwise_model$pvalue<- pf(stepwise_model$fstat, stepwise_model$numdf,
stepwise_model$dendf, lower.tail = FALSE)
stepwise_model$modelsig <- ifelse(stepwise_model$pvalue < .05, 1, 0)
#####################################################################
#Creating dataframe for unstandardized coefficient estimates#
coefficients <- bind_rows(df_coefficients, .id = 'UniqueID')

#Creating mergeable dataframe for standardized coefficient estimates#
betas_mid <- bind_rows(df_betas, .id = 'UniqueID')
betas_mid
betas_mid$term2 <- gsub("scale\\(", "", betas_mid$term)
betas_mid$term3 <- gsub("[)]", "", betas_mid$term2)
betas_mid$term <- gsub("[(]Intercept", "(Intercept)", betas_mid$term3)
betas_mid$beta <- betas_mid$estimate

betaVar <- c("UniqueID", "term", "beta")
betas <- betas_mid[complete.cases(betas_mid[, betaVar]), betaVar]
betas$beta_value <- ifelse(betas$term == '(Intercept)', NA, betas$beta)
```

```
#Merging standardized coefficient estimates with larger coefficient dataframe#
stepwise_coeff <- merge(coefficients, betas, by = c('UniqueID', 'term'))

######Assigning Meta Values################
#Significance values#
stepwise_coeff$coeffsig <- ifelse(stepwise_coeff$p.value < .05, 1, 0)
#Positive/negative betas#
stepwise_coeff$coeffdirection <- ifelse(stepwise_coeff$beta > 0, 'positive', 'negative')

#Ranking values#
coeff_data <- list()
for(i in 1:count){
  coeff_data[[i]] <- as.data.frame(subset(stepwise_coeff, UniqueID == i))
  coeff_data[[i]]$rank <- rank(-abs(coeff_data[[i]]$beta_value), na.last = FALSE,
ties.method = "average")
  coeff_data[[i]]$adjrank <- coeff_data[[i]]$rank - 1
  final_stepwise_coeff <- bind_rows(coeff_data)
}

#Adding Predictor Count to File#
final_model <- as.data.frame(subset(final_stepwise_coeff))
predictors <- c("UniqueID", "term")

CORSIMfinal_model_predictors <- final_model[complete.cases(final_model[,
predictors]), predictors]

stepmodel <- list()
predictorlist <- list()
for(i in 1:count){
  stepmodel[[i]] <- as.data.frame(subset(stepwise_model, UniqueID == i))
  predictorlist[[i]] <- as.data.frame(subset(CORSIMfinal_model_predictors, UniqueID ==
i))
  predictorlist[[i]]$Count <- (nrow(predictorlist[[i]]))-1
  stepmodel[[i]]$model_var <- (nrow(predictorlist[[i]]))-1
}

stepwise_model <- map_df(stepmodel, ~as.data.frame(.x), .id="UniqueID")
final_model_predictors <- map_df(predictorlist, ~as.data.frame(.x), .id="UniqueID")
##########################################################################

modelVar <- c("UniqueID", "term", "estimate", "beta")

model_estimates <- final_stepwise_coeff[complete.cases(final_stepwise_coeff[,
modelVar]), modelVar]

person_model <- list()
```

```
term_model <- list()
step_model <- list()
weights <- list()
stdweights <- list()

for(k in 1:count){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, UniqueID == k))
  step_model[[k]] <- as.data.frame(subset(stepwise_model, UniqueID == k))
  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  stdweights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$beta)))
  setDT(stdweights[[k]], keep.rownames = TRUE)[]
  setnames(stdweights[[k]], 1, "Predictor")
  setnames(stdweights[[k]], 2, "Mean")

  step_model[[k]]$intercept <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)', 1, 0))
  step_model[[k]]$X1 <- sum(ifelse(weights[[k]]$Predictor == 'X1', 1, 0))
  step_model[[k]]$X2 <- sum(ifelse(weights[[k]]$Predictor == 'X2', 1, 0))
  step_model[[k]]$X3 <- sum(ifelse(weights[[k]]$Predictor == 'X3', 1, 0))

  step_model[[k]]$intercept_b <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  step_model[[k]]$X1_b <- sum(ifelse(weights[[k]]$Predictor == 'X1',
weights[[k]]$Mean, 0))
  step_model[[k]]$X2_b <- sum(ifelse(weights[[k]]$Predictor == 'X2',
weights[[k]]$Mean, 0))
  step_model[[k]]$X3_b <- sum(ifelse(weights[[k]]$Predictor == 'X3',
weights[[k]]$Mean, 0))

  step_model[[k]]$intercept_beta <- sum(ifelse(stdweights[[k]]$Predictor == '(Intercept)',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X1_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X1',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X2_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X2',
stdweights[[k]]$Mean, 0))
  step_model[[k]]$X3_beta <- sum(ifelse(stdweights[[k]]$Predictor == 'X3',
stdweights[[k]]$Mean, 0))

}

stepwise_model <- map_df(step_model, ~as.data.frame(.x), .id="NumberID")
```

```
Conditions <- subset(data, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "rStructure")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

CORSIMfinal_stepwise_coeff <- merge(condition, final_stepwise_coeff, by =
c('UniqueID'), all.x = TRUE)
CORSIMstepwise_model <- merge(condition, stepwise_model, by = c('UniqueID'), all.x
= TRUE)

##SAVING ON PERSONAL PC: PC1###
write.csv(CORSIMfinal_stepwise_coeff, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, "
CORSIMfinal_stepwise_coeff.csv", sep = ""))
write.csv(CORSIMstepwise_model, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, "
CORSIMstepwise_model.csv", sep = ""))
write.csv(CORSIMfinal_model_predictors, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, "
CORSIMfinal_model_predictors.csv", sep = ""))

}

##################RUN STEPWISE ANALYSIS FUNCTION####################
#Set seed run function with different datasets
sets <- list(BCorrSim1, BCorrSim2, BCorrSim3, BCorrSim4, BCorrSim5, BCorrSim6,
BCorrSim7, BCorrSim8, BCorrSim9, BCorrSim10, BCorrSim11, BCorrSim12,
BCorrSim13, BCorrSim14, BCorrSim15, BCorrSim16, BCorrSim17, BCorrSim18,
BCorrSim19, BCorrSim20, BCorrSim21, BCorrSim22, BCorrSim23, BCorrSim24,
BCorrSim25, BCorrSim26, BCorrSim27, BCorrSim28, BCorrSim29, BCorrSim30,
BCorrSim31, BCorrSim32, BCorrSim33, BCorrSim34, BCorrSim35, BCorrSim36,
BCorrSim37, BCorrSim38, BCorrSim39, BCorrSim40)

sets_num <- list(1, 2, 3, 4, 5,  6,  7,  8,  9, 10,
          11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
          21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
          31, 32, 33, 34, 35, 36, 37, 38, 39, 40)

set.seed(122)
tfunc <- Sys.time()
mapply(step_analysis, sets, sets_num)
print(totalanalysis <- (Sys.time()-tfunc))
```

All Subsets Analysis Data Structure A

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("olsrr")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(data.table)

#Data from Personal PC: PC1
ACorrSim1 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim1.csv",
header=TRUE, sep=",")
ACorrSim2 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim2.csv",
header=TRUE, sep=",")
ACorrSim3 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim3.csv",
header=TRUE, sep=",")
ACorrSim4 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim4.csv",
header=TRUE, sep=",")
ACorrSim5 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim5.csv",
header=TRUE, sep=",")
ACorrSim6 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim6.csv",
header=TRUE, sep=",")
ACorrSim7 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim7.csv",
header=TRUE, sep=",")
ACorrSim8 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim8.csv",
header=TRUE, sep=",")
ACorrSim9 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim9.csv",
header=TRUE, sep=",")
ACorrSim10 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim10.csv",
header=TRUE, sep=",")


ACorrSim11 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim11.csv",
header=TRUE, sep=",")
```

```
ACorrSim12 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim12.csv",
header=TRUE, sep=",")
ACorrSim13 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim13.csv",
header=TRUE, sep=",")
ACorrSim14 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim14.csv",
header=TRUE, sep=",")
ACorrSim15 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim15.csv",
header=TRUE, sep=",")
ACorrSim16 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim16.csv",
header=TRUE, sep=",")
ACorrSim17 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim17.csv",
header=TRUE, sep=",")
ACorrSim18 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim18.csv",
header=TRUE, sep=",")
ACorrSim19 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim19.csv",
header=TRUE, sep=",")
ACorrSim20 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim20.csv",
header=TRUE, sep=",")

ACorrSim21 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim21.csv",
header=TRUE, sep=",")
ACorrSim22 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim22.csv",
header=TRUE, sep=",")
ACorrSim23 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim23.csv",
header=TRUE, sep=",")
ACorrSim24 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim24.csv",
header=TRUE, sep=",")
ACorrSim25 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim25.csv",
header=TRUE, sep=",")
ACorrSim26 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim26.csv",
header=TRUE, sep=",")
ACorrSim27 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim27.csv",
header=TRUE, sep=",")
ACorrSim28 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim28.csv",
header=TRUE, sep=",")
ACorrSim29 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim29.csv",
header=TRUE, sep=",")
ACorrSim30 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim30.csv",
header=TRUE, sep=",")

ACorrSim31 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim31.csv",
header=TRUE, sep=",")
ACorrSim32 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim32.csv",
header=TRUE, sep=",")
ACorrSim33 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim33.csv",
header=TRUE, sep=",")
```

```
ACorrSim34 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim34.csv",
header=TRUE, sep=",")
ACorrSim35 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim35.csv",
header=TRUE, sep=",")
ACorrSim36 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim36.csv",
header=TRUE, sep=",")
ACorrSim37 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim37.csv",
header=TRUE, sep=",")
ACorrSim38 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim38.csv",
header=TRUE, sep=",")
ACorrSim39 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim39.csv",
header=TRUE, sep=",")
ACorrSim40 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim40.csv",
header=TRUE, sep=",")

ACorrSim1 <- as.data.frame(ACorrSim1[with(ACorrSim1, order(UniqueID, Y)),])
ACorrSim2 <- as.data.frame(ACorrSim1[with(ACorrSim2, order(UniqueID, Y)),])
ACorrSim3 <- as.data.frame(ACorrSim1[with(ACorrSim3, order(UniqueID, Y)),])
ACorrSim4 <- as.data.frame(ACorrSim1[with(ACorrSim4, order(UniqueID, Y)),])
ACorrSim5 <- as.data.frame(ACorrSim1[with(ACorrSim5, order(UniqueID, Y)),])
ACorrSim6 <- as.data.frame(ACorrSim1[with(ACorrSim6, order(UniqueID, Y)),])
ACorrSim7 <- as.data.frame(ACorrSim1[with(ACorrSim7, order(UniqueID, Y)),])
ACorrSim8 <- as.data.frame(ACorrSim1[with(ACorrSim8, order(UniqueID, Y)),])
ACorrSim9 <- as.data.frame(ACorrSim1[with(ACorrSim9, order(UniqueID, Y)),])
ACorrSim10 <- as.data.frame(ACorrSim1[with(ACorrSim10, order(UniqueID, Y)),])

ACorrSim11 <- as.data.frame(ACorrSim1[with(ACorrSim11, order(UniqueID, Y)),])
ACorrSim12 <- as.data.frame(ACorrSim1[with(ACorrSim12, order(UniqueID, Y)),])
ACorrSim13 <- as.data.frame(ACorrSim1[with(ACorrSim13, order(UniqueID, Y)),])
ACorrSim14 <- as.data.frame(ACorrSim1[with(ACorrSim14, order(UniqueID, Y)),])
ACorrSim15 <- as.data.frame(ACorrSim1[with(ACorrSim15, order(UniqueID, Y)),])
ACorrSim16 <- as.data.frame(ACorrSim1[with(ACorrSim16, order(UniqueID, Y)),])
ACorrSim17 <- as.data.frame(ACorrSim1[with(ACorrSim17, order(UniqueID, Y)),])
ACorrSim18 <- as.data.frame(ACorrSim1[with(ACorrSim18, order(UniqueID, Y)),])
ACorrSim19 <- as.data.frame(ACorrSim1[with(ACorrSim19, order(UniqueID, Y)),])
ACorrSim20 <- as.data.frame(ACorrSim1[with(ACorrSim20, order(UniqueID, Y)),])

ACorrSim21 <- as.data.frame(ACorrSim1[with(ACorrSim21, order(UniqueID, Y)),])
ACorrSim22 <- as.data.frame(ACorrSim1[with(ACorrSim22, order(UniqueID, Y)),])
ACorrSim23 <- as.data.frame(ACorrSim1[with(ACorrSim23, order(UniqueID, Y)),])
ACorrSim24 <- as.data.frame(ACorrSim1[with(ACorrSim24, order(UniqueID, Y)),])
ACorrSim25 <- as.data.frame(ACorrSim1[with(ACorrSim25, order(UniqueID, Y)),])
ACorrSim26 <- as.data.frame(ACorrSim1[with(ACorrSim26, order(UniqueID, Y)),])
ACorrSim27 <- as.data.frame(ACorrSim1[with(ACorrSim27, order(UniqueID, Y)),])
ACorrSim28 <- as.data.frame(ACorrSim1[with(ACorrSim28, order(UniqueID, Y)),])
ACorrSim29 <- as.data.frame(ACorrSim1[with(ACorrSim29, order(UniqueID, Y)),])
```

```
ACorrSim30 <- as.data.frame(ACorrSim1[with(ACorrSim30, order(UniqueID, Y)),])

ACorrSim31 <- as.data.frame(ACorrSim1[with(ACorrSim31, order(UniqueID, Y)),])
ACorrSim32 <- as.data.frame(ACorrSim1[with(ACorrSim32, order(UniqueID, Y)),])
ACorrSim33 <- as.data.frame(ACorrSim1[with(ACorrSim33, order(UniqueID, Y)),])
ACorrSim34 <- as.data.frame(ACorrSim1[with(ACorrSim34, order(UniqueID, Y)),])
ACorrSim35 <- as.data.frame(ACorrSim1[with(ACorrSim35, order(UniqueID, Y)),])
ACorrSim36 <- as.data.frame(ACorrSim1[with(ACorrSim36, order(UniqueID, Y)),])
ACorrSim37 <- as.data.frame(ACorrSim1[with(ACorrSim37, order(UniqueID, Y)),])
ACorrSim38 <- as.data.frame(ACorrSim1[with(ACorrSim38, order(UniqueID, Y)),])
ACorrSim39 <- as.data.frame(ACorrSim1[with(ACorrSim39, order(UniqueID, Y)),])
ACorrSim40 <- as.data.frame(ACorrSim1[with(ACorrSim40, order(UniqueID, Y)),])


##################ALL SUBSETS ANALYSIS FUNCTION#####################
all_subsets <- function(data, bootset, CorrSIMGroup){

 t1 <- Sys.time()

 data <- data
 datagroup <- CorrSIMGroup
 start <- 1
 end <- 4

 datalist <- list()
 resamples <- list()
 boot_set <- list()
 leaps <- list()
 summary <- list()
 maxadjr2 <- list()
 variables1 <- list()
 variables2 <- list()
 variables3 <- list()
 coefficients1 <- list()
 coefficients2 <- list()
 bestmodel <- list()
 corsim_bestmodels <- list()

 bestfit <- list()
 only_bestmodels <- list()

 modelsonly1 <- list()
 modelsonly2 <- list()
 modelsonly3 <- list()
 freqmodel <- list()
 variable_importance <- list()
```

```
for(k in start:end){
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(data, UniqueID == k))

  resamples[[k]] <- lapply(1:bootset, function(i) sample_n(datalist[[k]], 50, replace = T))

  for(j in 1:bootset){
    t3 <- Sys.time()
    boot_set[[j]] <- resamples[[k]][[j]]
    leaps[[j]] <-
      regsubsets(Y ~ X1 + X2 + X3 + X4,
                 data = boot_set[[j]],
                 nbest = 1,       # 1 best model for each number of predictors
                 nvmax = NULL,    # NULL for no limit on number of variables
                 force.in = NULL, force.out = NULL,
                 method = "exhaustive")
    summary[[j]] <- summary(leaps[[j]])
    maxadjr2[[j]] <- which.max(summary[[j]]$adjr2)
    variables1[[j]] <- map_df(summary[[j]]$which[maxadjr2[[j]],], ~as.data.frame(.x),
.id="term")
    variables1[[j]]$included <- variables1[[j]]$.x
    variables2[[j]] <- variables1[[j]][-c(2)]
    variables3[[j]] <- as.data.frame(subset(variables2[[j]]))
    variables3[[j]]$adj.r.squared <- summary[[j]]$adjr2[maxadjr2[[j]]]
    variables3[[j]]$inclusion <- ifelse(variables3[[j]]$included == TRUE, 1, 0)
    variables3[[j]]$modelcode <- paste0(variables3[[j]]$inclusion, collapse = "")
    coefficients1[[j]] <- map_df(coef(leaps[[j]],maxadjr2[[j]],vcov=FALSE),
~as.data.frame(.x), .id="term")
    coefficients1[[j]]$estimate <- coefficients1[[j]]$`coef(leaps[[j]], maxadjr2[[j]], vcov =
FALSE)`
    coefficients1[[j]]$estimate <- coefficients1[[j]]$.x
    coefficients2[[j]] <- coefficients1[[j]][-c(2)]

    bestmodel[[j]] <- merge(variables3[[j]], coefficients2[[j]], by = "term")
    modelsonly1[[j]] <- bestmodel[[j]][!duplicated(bestmodel[[j]]$modelcode),]

    print(bootstrapset <- Sys.time()-t3)

  }

  modelsonly2[[k]] <- map_df(modelsonly1, ~as.data.frame(.x), .id="bootset1")
  modelsonly3[[k]] <- count(modelsonly2[[k]], modelcode)
  freqmodel[[k]] <- which.max(modelsonly3[[k]]$n)
  modelsonly3[[k]]$bestfit <- modelsonly3[[k]]$modelcode[freqmodel[[k]]]
  modelsonly4 <- map_df(modelsonly3, ~as.data.frame(.x), .id="UniqueID")
```

```
  corsim_bestmodels[[k]] <- map_df(bestmodel, ~as.data.frame(.x), .id="bootset")
  bestfit[[k]] <- merge(corsim_bestmodels[[k]], modelsonly3[[k]], by = "modelcode")
  bestfit[[k]]$bestmodel <- ifelse(bestfit[[k]]$modelcode == bestfit[[k]]$bestfit, TRUE,
FALSE)
  only_bestmodels[[k]] <- subset(bestfit[[k]], bestmodel == TRUE)
  variable_importance[[k]] <- count(corsim_bestmodels[[k]], term)
  variable_importance[[k]]$frequency <- variable_importance[[k]]$n
  variable_importance[[k]]$probability <- (variable_importance[[k]]$frequency)/(bootset)
  print(judgeanalysis <- Sys.time()-t2)

 #Hashtagged text included below for saving all bootstrapped models if desired
 #complete_bootsets <- map_df(bestfit, ~as.data.frame(.x), .id="UniqueID")
  #complete_bootsets <- complete_bootsets[
   # with(complete_bootsets, order(UniqueID, bootset, term)),
    #]
  #complete_bootsets$UniqueID <- as.numeric(complete_bootsets$UniqueID) + index

  CORSIMfinal_bestmodels <- map_df(only_bestmodels, ~as.data.frame(.x),
.id="UniqueID")
  CORSIMfinal_bestmodels <- as.data.frame(CORSIMfinal_bestmodels[
    with(CORSIMfinal_bestmodels, order(UniqueID, bootset, term)),
    ])

  CORSIMpredictor_importance <- map_df(variable_importance, ~as.data.frame(.x),
.id="UniqueID")
}
print(totalanalysis <- (Sys.time()-t1))

CORSIMpredictor_importance
CORSIMfinal_bestmodels

###SAVE RELEVANT FILES
##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMfinal_bestmodels, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, " All Subsets
CORSIMfinal_bootsets.csv", sep = ""))
write.csv(CORSIMpredictor_importance, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, " All Subsets
CORSIMpredictor_importance.csv", sep = ""))
#write.csv(SIMcomplete_bootsets, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, " All Subsets
SIMcomplete_bootsets.csv", sep=""))

best_models <- list()
bootset_model <- list()
```

```
AS_model <- list()
criticality <- list()
VI_model <- list()
VI <- list()
term_model <- list()
final_model <- list()
weights <- list()

t1a <- Sys.time()
for(k in start:end){
  t2 <- Sys.time()
  best_models[[k]] <- as.data.frame(subset(CORSIMfinal_bestmodels, UniqueID == k))
  bootset_model[[k]] <- as.data.frame(subset(best_models[[k]], bootset == min(bootset)))

  criticality[[k]] <- as.data.frame(subset(CORSIMpredictor_importance, UniqueID == k))

  AS_model[[k]] <- bootset_model[[k]][!duplicated(bootset_model[[k]]$UniqueID),]

  AS_model[[k]]$model_var <- sum(bootset_model[[k]]$inclusion) - 1

  AS_model[[k]]$mean_adjR2 <- mean(best_models[[k]]$adj.r.squared)

  term_model[[k]] <-  split(best_models[[k]], best_models[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))

  VI_model[[k]] <- split(criticality[[k]], criticality[[k]]$term)
  VI[[k]] <- as.data.frame(sapply(VI_model[[k]], function(x) mean(x$probability)))

  final_model[[k]] <- as.data.frame(subset(CORSIMfinal_bestmodels, UniqueID == k))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  setDT(VI[[k]], keep.rownames = TRUE)[]
  setnames(VI[[k]], 1, "term")
  setnames(VI[[k]], 2, "Mean")

  AS_model[[k]]$intercept <- sum(ifelse(final_model[[k]]$term == '(Intercept)', 1, 0))
  AS_model[[k]]$X1 <- sum(ifelse(final_model[[k]]$term == 'X1', 1, 0))
  AS_model[[k]]$X2 <- sum(ifelse(final_model[[k]]$term == 'X2', 1, 0))
  AS_model[[k]]$X3 <- sum(ifelse(final_model[[k]]$term == 'X3', 1, 0))
  AS_model[[k]]$X4 <- sum(ifelse(final_model[[k]]$term == 'X4', 1, 0))

  AS_model[[k]]$intercept_b <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
```

```
  AS_model[[k]]$X1_b <- sum(ifelse(weights[[k]]$Predictor == 'X1',
weights[[k]]$Mean, 0))
  AS_model[[k]]$X2_b <- sum(ifelse(weights[[k]]$Predictor == 'X2',
weights[[k]]$Mean, 0))
  AS_model[[k]]$X3_b <- sum(ifelse(weights[[k]]$Predictor == 'X3',
weights[[k]]$Mean, 0))
  AS_model[[k]]$X4_b <- sum(ifelse(weights[[k]]$Predictor == 'X4',
weights[[k]]$Mean, 0))

  AS_model[[k]]$intercept_PC <- sum(ifelse(VI[[k]]$term == '(Intercept)',
VI[[k]]$Mean, 0))
  AS_model[[k]]$X1_PC <- sum(ifelse(VI[[k]]$term == 'X1', VI[[k]]$Mean, 0))
  AS_model[[k]]$X2_PC <- sum(ifelse(VI[[k]]$term == 'X2', VI[[k]]$Mean, 0))
  AS_model[[k]]$X3_PC <- sum(ifelse(VI[[k]]$term == 'X3', VI[[k]]$Mean, 0))
  AS_model[[k]]$X4_PC <- sum(ifelse(VI[[k]]$term == 'X4', VI[[k]]$Mean, 0))
}

CORSIMall_subsets_model <- map_df(AS_model, ~as.data.frame(.x), .id="UniqueID")

##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMall_subsets_model, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, " All Subsets CORSIM
Model Outcomes.csv", sep = ""))

print(cross_validated <- (Sys.time()-t1))

}

##############################################################################
#############RUN ALL SUBSETS ANALYSIS FUNCTION#####################
#Set seed once and run function with different datasets

sets <- list(ACorrSim1, ACorrSim2, ACorrSim3, ACorrSim4, ACorrSim5, ACorrSim6,
ACorrSim7, ACorrSim8, ACorrSim9, ACorrSim10, ACorrSim11, ACorrSim12,
ACorrSim13, ACorrSim14, ACorrSim15, ACorrSim16, ACorrSim17, ACorrSim18,
ACorrSim19, ACorrSim20, ACorrSim21, ACorrSim22, ACorrSim23, ACorrSim24,
ACorrSim25, ACorrSim26, ACorrSim27, ACorrSim28, ACorrSim29, ACorrSim30,
ACorrSim31, ACorrSim32, ACorrSim33, ACorrSim34, ACorrSim35, ACorrSim36,
ACorrSim37, ACorrSim38, ACorrSim39, ACorrSim40)

sets_num <- list(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
          11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
          21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
          31, 32, 33, 34, 35, 36, 37, 38, 39, 40)

set.seed(1015)
```

```
tfunc <- Sys.time()
mapply(all_subsets, sets, 500, sets_num)
print(totalanalysis <- (Sys.time()-tfunc))
```

All Subsets Analysis Data Structure B

```
#install.packages("tidyverse")
#install.packages("broom")
#install.packages("dplyr")
#install.packages("readxl")
#install.packages("lmf")
#install.packages("leaps")
#install.packages("olsrr")

rm(list=ls())

library(tidyverse)
library(broom)
library(readxl)
library(lmf)
library(purrr)
library(dplyr)
library(leaps)
library(data.table)

#Data from Personal PC: PC1
BCorrSim1 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim1.csv",
header=TRUE, sep=",")
BCorrSim2 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim2.csv",
header=TRUE, sep=",")
BCorrSim3 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim3.csv",
header=TRUE, sep=",")
BCorrSim4 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim4.csv",
header=TRUE, sep=",")
BCorrSim5 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim5.csv",
header=TRUE, sep=",")
BCorrSim6 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim6.csv",
header=TRUE, sep=",")
BCorrSim7 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim7.csv",
header=TRUE, sep=",")
BCorrSim8 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim8.csv",
header=TRUE, sep=",")
BCorrSim9 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim9.csv",
header=TRUE, sep=",")
```

```
BCorrSim10 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim10.csv",
header=TRUE, sep=",")

BCorrSim11 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim11.csv",
header=TRUE, sep=",")
BCorrSim12 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim12.csv",
header=TRUE, sep=",")
BCorrSim13 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim13.csv",
header=TRUE, sep=",")
BCorrSim14 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim14.csv",
header=TRUE, sep=",")
BCorrSim15 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim15.csv",
header=TRUE, sep=",")
BCorrSim16 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim16.csv",
header=TRUE, sep=",")
BCorrSim17 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim17.csv",
header=TRUE, sep=",")
BCorrSim18 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim18.csv",
header=TRUE, sep=",")
BCorrSim19 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim19.csv",
header=TRUE, sep=",")
BCorrSim20 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim20.csv",
header=TRUE, sep=",")

BCorrSim21 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim21.csv",
header=TRUE, sep=",")
BCorrSim22 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim22.csv",
header=TRUE, sep=",")
BCorrSim23 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim23.csv",
header=TRUE, sep=",")
BCorrSim24 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim24.csv",
header=TRUE, sep=",")
BCorrSim25 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim25.csv",
header=TRUE, sep=",")
BCorrSim26 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim26.csv",
header=TRUE, sep=",")
BCorrSim27 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim27.csv",
header=TRUE, sep=",")
BCorrSim28 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim28.csv",
header=TRUE, sep=",")
BCorrSim29 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim29.csv",
header=TRUE, sep=",")
BCorrSim30 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim30.csv",
header=TRUE, sep=",")
```

```
BCorrSim31 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim31.csv",
header=TRUE, sep=",")
BCorrSim32 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim32.csv",
header=TRUE, sep=",")
BCorrSim33 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim33.csv",
header=TRUE, sep=",")
BCorrSim34 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim34.csv",
header=TRUE, sep=",")
BCorrSim35 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim35.csv",
header=TRUE, sep=",")
BCorrSim36 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim36.csv",
header=TRUE, sep=",")
BCorrSim37 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim37.csv",
header=TRUE, sep=",")
BCorrSim38 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim38.csv",
header=TRUE, sep=",")
BCorrSim39 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim39.csv",
header=TRUE, sep=",")
BCorrSim40 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim40.csv",
header=TRUE, sep=",")

BCorrSim1 <- as.data.frame(BCorrSim1[with(BCorrSim1, order(UniqueID, Y)),])
BCorrSim2 <- as.data.frame(BCorrSim1[with(BCorrSim2, order(UniqueID, Y)),])
BCorrSim3 <- as.data.frame(BCorrSim1[with(BCorrSim3, order(UniqueID, Y)),])
BCorrSim4 <- as.data.frame(BCorrSim1[with(BCorrSim4, order(UniqueID, Y)),])
BCorrSim5 <- as.data.frame(BCorrSim1[with(BCorrSim5, order(UniqueID, Y)),])
BCorrSim6 <- as.data.frame(BCorrSim1[with(BCorrSim6, order(UniqueID, Y)),])
BCorrSim7 <- as.data.frame(BCorrSim1[with(BCorrSim7, order(UniqueID, Y)),])
BCorrSim8 <- as.data.frame(BCorrSim1[with(BCorrSim8, order(UniqueID, Y)),])
BCorrSim9 <- as.data.frame(BCorrSim1[with(BCorrSim9, order(UniqueID, Y)),])
BCorrSim10 <- as.data.frame(BCorrSim1[with(BCorrSim10, order(UniqueID, Y)),])

BCorrSim11 <- as.data.frame(BCorrSim1[with(BCorrSim11, order(UniqueID, Y)),])
BCorrSim12 <- as.data.frame(BCorrSim1[with(BCorrSim12, order(UniqueID, Y)),])
BCorrSim13 <- as.data.frame(BCorrSim1[with(BCorrSim13, order(UniqueID, Y)),])
BCorrSim14 <- as.data.frame(BCorrSim1[with(BCorrSim14, order(UniqueID, Y)),])
BCorrSim15 <- as.data.frame(BCorrSim1[with(BCorrSim15, order(UniqueID, Y)),])
BCorrSim16 <- as.data.frame(BCorrSim1[with(BCorrSim16, order(UniqueID, Y)),])
BCorrSim17 <- as.data.frame(BCorrSim1[with(BCorrSim17, order(UniqueID, Y)),])
BCorrSim18 <- as.data.frame(BCorrSim1[with(BCorrSim18, order(UniqueID, Y)),])
BCorrSim19 <- as.data.frame(BCorrSim1[with(BCorrSim19, order(UniqueID, Y)),])
BCorrSim20 <- as.data.frame(BCorrSim1[with(BCorrSim20, order(UniqueID, Y)),])

BCorrSim21 <- as.data.frame(BCorrSim1[with(BCorrSim21, order(UniqueID, Y)),])
BCorrSim22 <- as.data.frame(BCorrSim1[with(BCorrSim22, order(UniqueID, Y)),])
BCorrSim23 <- as.data.frame(BCorrSim1[with(BCorrSim23, order(UniqueID, Y)),])
```

```
BCorrSim24 <- as.data.frame(BCorrSim1[with(BCorrSim24, order(UniqueID, Y)),])
BCorrSim25 <- as.data.frame(BCorrSim1[with(BCorrSim25, order(UniqueID, Y)),])
BCorrSim26 <- as.data.frame(BCorrSim1[with(BCorrSim26, order(UniqueID, Y)),])
BCorrSim27 <- as.data.frame(BCorrSim1[with(BCorrSim27, order(UniqueID, Y)),])
BCorrSim28 <- as.data.frame(BCorrSim1[with(BCorrSim28, order(UniqueID, Y)),])
BCorrSim29 <- as.data.frame(BCorrSim1[with(BCorrSim29, order(UniqueID, Y)),])
BCorrSim30 <- as.data.frame(BCorrSim1[with(BCorrSim30, order(UniqueID, Y)),])

BCorrSim31 <- as.data.frame(BCorrSim1[with(BCorrSim31, order(UniqueID, Y)),])
BCorrSim32 <- as.data.frame(BCorrSim1[with(BCorrSim32, order(UniqueID, Y)),])
BCorrSim33 <- as.data.frame(BCorrSim1[with(BCorrSim33, order(UniqueID, Y)),])
BCorrSim34 <- as.data.frame(BCorrSim1[with(BCorrSim34, order(UniqueID, Y)),])
BCorrSim35 <- as.data.frame(BCorrSim1[with(BCorrSim35, order(UniqueID, Y)),])
BCorrSim36 <- as.data.frame(BCorrSim1[with(BCorrSim36, order(UniqueID, Y)),])
BCorrSim37 <- as.data.frame(BCorrSim1[with(BCorrSim37, order(UniqueID, Y)),])
BCorrSim38 <- as.data.frame(BCorrSim1[with(BCorrSim38, order(UniqueID, Y)),])
BCorrSim39 <- as.data.frame(BCorrSim1[with(BCorrSim39, order(UniqueID, Y)),])
BCorrSim40 <- as.data.frame(BCorrSim1[with(BCorrSim40, order(UniqueID, Y)),])

######################ALL SUBSETS ANALYSIS FUNCTION#################
all_subsets <- function(data, bootset, CorrSIMGroup){

  t1 <- Sys.time()

  data <- data
  datagroup <- CorrSIMGroup
  start <- 1
  end <- 5

  datalist <- list()
  resamples <- list()
  boot_set <- list()
  leaps <- list()
  summary <- list()
  maxadjr2 <- list()
  variables1 <- list()
  variables2 <- list()
  variables3 <- list()
  coefficients1 <- list()
  coefficients2 <- list()
  bestmodel <- list()
  corsim_bestmodels <- list()

  bestfit <- list()
  only_bestmodels <- list()
```

```
  modelsonly1 <- list()
  modelsonly2 <- list()
  modelsonly3 <- list()
  freqmodel <- list()
  variable_importance <- list()

for(k in start:end){
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(data, UniqueID == k))

  resamples[[k]] <- lapply(1:bootset, function(i) sample_n(datalist[[k]], 50, replace = T))

  for(j in 1:bootset){
    t3 <- Sys.time()
    boot_set[[j]] <- resamples[[k]][[j]]
    leaps[[j]] <-
      regsubsets(Y ~ X1 + X2 + X3,
                 data = boot_set[[j]],
                 nbest = 1,       # 1 best model for each number of predictors
                 nvmax = NULL,    # NULL for no limit on number of variables
                 force.in = NULL, force.out = NULL,
                 method = "exhaustive")
    summary[[j]] <- summary(leaps[[j]])
    maxadjr2[[j]] <- which.max(summary[[j]]$adjr2)
    variables1[[j]] <- map_df(summary[[j]]$which[maxadjr2[[j]],], ~as.data.frame(.x),
.id="term")
    variables1[[j]]$included <- variables1[[j]]$.x
    variables2[[j]] <- variables1[[j]][-c(2)]
    variables3[[j]] <- as.data.frame(subset(variables2[[j]]))
    variables3[[j]]$adj.r.squared <- summary[[j]]$adjr2[maxadjr2[[j]]]
    variables3[[j]]$inclusion <- ifelse(variables3[[j]]$included == TRUE, 1, 0)
    variables3[[j]]$modelcode <- paste0(variables3[[j]]$inclusion, collapse = "")
    coefficients1[[j]] <- map_df(coef(leaps[[j]],maxadjr2[[j]],vcov=FALSE),
~as.data.frame(.x), .id="term")
    coefficients1[[j]]$estimate <- coefficients1[[j]]$`coef(leaps[[j]], maxadjr2[[j]], vcov =
FALSE)`
    coefficients1[[j]]$estimate <- coefficients1[[j]]$.x
    coefficients2[[j]] <- coefficients1[[j]][-c(2)]

    bestmodel[[j]] <- merge(variables3[[j]], coefficients2[[j]], by = "term")
    modelsonly1[[j]] <- bestmodel[[j]][!duplicated(bestmodel[[j]]$modelcode),]

    print(bootstrapset <- Sys.time()-t3)

  }
```

```
  modelsonly2[[k]] <- map_df(modelsonly1, ~as.data.frame(.x), .id="bootset1")
  modelsonly3[[k]] <- count(modelsonly2[[k]], modelcode)
  freqmodel[[k]] <- which.max(modelsonly3[[k]]$n)
  modelsonly3[[k]]$bestfit <- modelsonly3[[k]]$modelcode[freqmodel[[k]]]
  modelsonly4 <- map_df(modelsonly3, ~as.data.frame(.x), .id="UniqueID")

  corsim_bestmodels[[k]] <- map_df(bestmodel, ~as.data.frame(.x), .id="bootset")
  bestfit[[k]] <- merge(corsim_bestmodels[[k]], modelsonly3[[k]], by = "modelcode")
  bestfit[[k]]$bestmodel <- ifelse(bestfit[[k]]$modelcode == bestfit[[k]]$bestfit, TRUE,
FALSE)
  only_bestmodels[[k]] <- subset(bestfit[[k]], bestmodel == TRUE)
  variable_importance[[k]] <- count(corsim_bestmodels[[k]], term)
  variable_importance[[k]]$frequency <- variable_importance[[k]]$n
  variable_importance[[k]]$probability <- (variable_importance[[k]]$frequency)/(bootset)
  print(judgeanalysis <- Sys.time()-t2)

  complete_bootsets <- map_df(bestfit, ~as.data.frame(.x), .id="UniqueID")
  CORSIMcomplete_bootsets <- complete_bootsets[
   with(complete_bootsets, order(UniqueID, bootset, term)),
    ]
  complete_bootsets$UniqueID <- as.numeric(complete_bootsets$UniqueID)


  CORSIMfinal_bestmodels <- map_df(only_bestmodels, ~as.data.frame(.x),
.id="UniqueID")
  CORSIMfinal_bestmodels <- as.data.frame(CORSIMfinal_bestmodels[
    with(CORSIMfinal_bestmodels, order(UniqueID, bootset, term)),
    ])

  CORSIMpredictor_importance <- map_df(variable_importance, ~as.data.frame(.x),
.id="UniqueID")
}
print(totalanalysis <- (Sys.time()-t1))

###SAVE RELEVANT FILES
##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMfinal_bestmodels, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, " All Subsets
CORSIMfinal_bootsets.csv", sep = ""))
write.csv(CORSIMpredictor_importance, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, " All Subsets
CORSIMpredictor_importance.csv", sep = ""))
write.csv(CORSIMcomplete_bootsets, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, " All Subsets
CORSIMcomplete_bootsets.csv", sep = ""))
```

```
#write.csv(SIMcomplete_bootsets, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, " All Subsets
SIMcomplete_bootsets.csv", sep=""))

best_models <- list()
bootset_model <- list()
AS_model <- list()
criticality <- list()
VI_model <- list()
VI <- list()
term_model <- list()
final_model <- list()
weights <- list()

t1a <- Sys.time()
for(k in start:end){
  t2 <- Sys.time()
  best_models[[k]] <- as.data.frame(subset(CORSIMfinal_bestmodels, UniqueID == k))
  bootset_model[[k]] <- as.data.frame(subset(best_models[[k]], bootset == min(bootset)))

  criticality[[k]] <- as.data.frame(subset(CORSIMpredictor_importance, UniqueID == k))

  AS_model[[k]] <- bootset_model[[k]][!duplicated(bootset_model[[k]]$UniqueID),]

  AS_model[[k]]$model_var <- sum(bootset_model[[k]]$inclusion) - 1

  AS_model[[k]]$mean_adjR2 <- mean(best_models[[k]]$adj.r.squared)

  term_model[[k]] <-  split(best_models[[k]], best_models[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x) mean(x$estimate)))

  VI_model[[k]] <- split(criticality[[k]], criticality[[k]]$term)
  VI[[k]] <- as.data.frame(sapply(VI_model[[k]], function(x) mean(x$probability)))

  final_model[[k]] <- as.data.frame(subset(CORSIMfinal_bestmodels, UniqueID == k))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  setDT(VI[[k]], keep.rownames = TRUE)[]
  setnames(VI[[k]], 1, "term")
  setnames(VI[[k]], 2, "Mean")

  AS_model[[k]]$intercept <- sum(ifelse(final_model[[k]]$term == '(Intercept)', 1, 0))
  AS_model[[k]]$X1 <- sum(ifelse(final_model[[k]]$term == 'X1', 1, 0))
  AS_model[[k]]$X2 <- sum(ifelse(final_model[[k]]$term == 'X2', 1, 0))
```

```
  AS_model[[k]]$X3 <- sum(ifelse(final_model[[k]]$term == 'X3', 1, 0))

  AS_model[[k]]$intercept_b <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  AS_model[[k]]$X1_b <- sum(ifelse(weights[[k]]$Predictor == 'X1',
weights[[k]]$Mean, 0))
  AS_model[[k]]$X2_b <- sum(ifelse(weights[[k]]$Predictor == 'X2',
weights[[k]]$Mean, 0))
  AS_model[[k]]$X3_b <- sum(ifelse(weights[[k]]$Predictor == 'X3',
weights[[k]]$Mean, 0))

  AS_model[[k]]$intercept_PC <- sum(ifelse(VI[[k]]$term == '(Intercept)',
VI[[k]]$Mean, 0))
  AS_model[[k]]$X1_PC <- sum(ifelse(VI[[k]]$term == 'X1', VI[[k]]$Mean, 0))
  AS_model[[k]]$X2_PC <- sum(ifelse(VI[[k]]$term == 'X2', VI[[k]]$Mean, 0))
  AS_model[[k]]$X3_PC <- sum(ifelse(VI[[k]]$term == 'X3', VI[[k]]$Mean, 0))
}

CORSIMall_subsets_model <- map_df(AS_model, ~as.data.frame(.x), .id="NumberID")

Conditions <- subset(data, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "rStructure")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

CORSIMall_subsets_model <- merge(condition, CORSIMall_subsets_model, by =
c('UniqueID'), all.x = TRUE)

##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMall_subsets_model, file =
paste0("C:/Users/Kristina/Documents/BCorrSim", datagroup, " All Subsets CORSIM
Model Outcomes.csv", sep = ""))

print(cross_validated <- (Sys.time()-t1))

}

##############################################################################
#############RUN ALL SUBSETS ANALYSIS FUNCTION#####################
#Set seed once and run function with different datasets
sets <- list(BCorrSim1, BCorrSim2, BCorrSim3, BCorrSim4, BCorrSim5, BCorrSim6,
BCorrSim7, BCorrSim8, BCorrSim9, BCorrSim10, BCorrSim11, BCorrSim12,
BCorrSim13, BCorrSim14, BCorrSim15, BCorrSim16, BCorrSim17, BCorrSim18,
BCorrSim19, BCorrSim20, BCorrSim21, BCorrSim22, BCorrSim23, BCorrSim24,
BCorrSim25, BCorrSim26, BCorrSim27, BCorrSim28, BCorrSim29, BCorrSim30,
BCorrSim31, BCorrSim32, BCorrSim33, BCorrSim34, BCorrSim35, BCorrSim36,
BCorrSim37, BCorrSim38, BCorrSim39, BCorrSim40)
```

```
sets_num <- list(1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
          11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
          21, 22, 23, 24, 25, 26, 27, 28, 29, 30,
          31, 32, 33, 34, 35, 36, 37, 38, 39, 40)

set.seed(128)
tfunc <- Sys.time()
mapply(all_subsets, sets, 500, sets_num)
print(totalanalysis <- (Sys.time()-tfunc))
```

Random Forest Analysis Data Structure A

```
#install.packages("VSURF")
#install.packages("randomForest")
#install.packages("party")
#install.packages("readxl")
#install.packages("purrr")
#install.packages("data.table")
#install.packages("dplyr")

rm(list = ls())

library(VSURF)
library(randomForest)
library(party)
library(readxl)
library(purrr)
library(data.table)
library(dplyr)

#Data from Personal PC: PC1
ACorrSim1 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim1.csv",
header=TRUE, sep=",")
ACorrSim2 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim2.csv",
header=TRUE, sep=",")
ACorrSim3 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim3.csv",
header=TRUE, sep=",")
ACorrSim4 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim4.csv",
header=TRUE, sep=",")
ACorrSim5 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim5.csv",
header=TRUE, sep=",")
```

```
ACorrSim6 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim6.csv",
header=TRUE, sep=",")
ACorrSim7 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim7.csv",
header=TRUE, sep=",")
ACorrSim8 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim8.csv",
header=TRUE, sep=",")
ACorrSim9 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim9.csv",
header=TRUE, sep=",")
ACorrSim10 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim10.csv",
header=TRUE, sep=",")


ACorrSim11 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim11.csv",
header=TRUE, sep=",")
ACorrSim12 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim12.csv",
header=TRUE, sep=",")
ACorrSim13 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim13.csv",
header=TRUE, sep=",")
ACorrSim14 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim14.csv",
header=TRUE, sep=",")
ACorrSim15 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim15.csv",
header=TRUE, sep=",")
ACorrSim16 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim16.csv",
header=TRUE, sep=",")
ACorrSim17 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim17.csv",
header=TRUE, sep=",")
ACorrSim18 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim18.csv",
header=TRUE, sep=",")
ACorrSim19 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim19.csv",
header=TRUE, sep=",")
ACorrSim20 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim20.csv",
header=TRUE, sep=",")

ACorrSim21 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim21.csv",
header=TRUE, sep=",")
ACorrSim22 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim22.csv",
header=TRUE, sep=",")
ACorrSim23 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim23.csv",
header=TRUE, sep=",")
ACorrSim24 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim24.csv",
header=TRUE, sep=",")
ACorrSim25 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim25.csv",
header=TRUE, sep=",")
ACorrSim26 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim26.csv",
header=TRUE, sep=",")
```

```
ACorrSim27 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim27.csv",
header=TRUE, sep=",")
ACorrSim28 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim28.csv",
header=TRUE, sep=",")
ACorrSim29 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim29.csv",
header=TRUE, sep=",")
ACorrSim30 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim30.csv",
header=TRUE, sep=",")

ACorrSim31 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim31.csv",
header=TRUE, sep=",")
ACorrSim32 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim32.csv",
header=TRUE, sep=",")
ACorrSim33 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim33.csv",
header=TRUE, sep=",")
ACorrSim34 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim34.csv",
header=TRUE, sep=",")
ACorrSim35 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim35.csv",
header=TRUE, sep=",")
ACorrSim36 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim36.csv",
header=TRUE, sep=",")
ACorrSim37 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim37.csv",
header=TRUE, sep=",")
ACorrSim38 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim38.csv",
header=TRUE, sep=",")
ACorrSim39 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim39.csv",
header=TRUE, sep=",")
ACorrSim40 <- read.csv(file="C:/Users/Kristina/Documents/ACorrSim40.csv",
header=TRUE, sep=",")

ACorrSim1 <- as.data.frame(ACorrSim1[with(ACorrSim1, order(UniqueID, Y)),])
ACorrSim2 <- as.data.frame(ACorrSim1[with(ACorrSim2, order(UniqueID, Y)),])
ACorrSim3 <- as.data.frame(ACorrSim1[with(ACorrSim3, order(UniqueID, Y)),])
ACorrSim4 <- as.data.frame(ACorrSim1[with(ACorrSim4, order(UniqueID, Y)),])
ACorrSim5 <- as.data.frame(ACorrSim1[with(ACorrSim5, order(UniqueID, Y)),])
ACorrSim6 <- as.data.frame(ACorrSim1[with(ACorrSim6, order(UniqueID, Y)),])
ACorrSim7 <- as.data.frame(ACorrSim1[with(ACorrSim7, order(UniqueID, Y)),])
ACorrSim8 <- as.data.frame(ACorrSim1[with(ACorrSim8, order(UniqueID, Y)),])
ACorrSim9 <- as.data.frame(ACorrSim1[with(ACorrSim9, order(UniqueID, Y)),])
ACorrSim10 <- as.data.frame(ACorrSim1[with(ACorrSim10, order(UniqueID, Y)),])

ACorrSim11 <- as.data.frame(ACorrSim1[with(ACorrSim11, order(UniqueID, Y)),])
ACorrSim12 <- as.data.frame(ACorrSim1[with(ACorrSim12, order(UniqueID, Y)),])
ACorrSim13 <- as.data.frame(ACorrSim1[with(ACorrSim13, order(UniqueID, Y)),])
ACorrSim14 <- as.data.frame(ACorrSim1[with(ACorrSim14, order(UniqueID, Y)),])
ACorrSim15 <- as.data.frame(ACorrSim1[with(ACorrSim15, order(UniqueID, Y)),])
```

```
ACorrSim16 <- as.data.frame(ACorrSim1[with(ACorrSim16, order(UniqueID, Y)),])
ACorrSim17 <- as.data.frame(ACorrSim1[with(ACorrSim17, order(UniqueID, Y)),])
ACorrSim18 <- as.data.frame(ACorrSim1[with(ACorrSim18, order(UniqueID, Y)),])
ACorrSim19 <- as.data.frame(ACorrSim1[with(ACorrSim19, order(UniqueID, Y)),])
ACorrSim20 <- as.data.frame(ACorrSim1[with(ACorrSim20, order(UniqueID, Y)),])

ACorrSim21 <- as.data.frame(ACorrSim1[with(ACorrSim21, order(UniqueID, Y)),])
ACorrSim22 <- as.data.frame(ACorrSim1[with(ACorrSim22, order(UniqueID, Y)),])
ACorrSim23 <- as.data.frame(ACorrSim1[with(ACorrSim23, order(UniqueID, Y)),])
ACorrSim24 <- as.data.frame(ACorrSim1[with(ACorrSim24, order(UniqueID, Y)),])
ACorrSim25 <- as.data.frame(ACorrSim1[with(ACorrSim25, order(UniqueID, Y)),])
ACorrSim26 <- as.data.frame(ACorrSim1[with(ACorrSim26, order(UniqueID, Y)),])
ACorrSim27 <- as.data.frame(ACorrSim1[with(ACorrSim27, order(UniqueID, Y)),])
ACorrSim28 <- as.data.frame(ACorrSim1[with(ACorrSim28, order(UniqueID, Y)),])
ACorrSim29 <- as.data.frame(ACorrSim1[with(ACorrSim29, order(UniqueID, Y)),])
ACorrSim30 <- as.data.frame(ACorrSim1[with(ACorrSim30, order(UniqueID, Y)),])

ACorrSim31 <- as.data.frame(ACorrSim1[with(ACorrSim31, order(UniqueID, Y)),])
ACorrSim32 <- as.data.frame(ACorrSim1[with(ACorrSim32, order(UniqueID, Y)),])
ACorrSim33 <- as.data.frame(ACorrSim1[with(ACorrSim33, order(UniqueID, Y)),])
ACorrSim34 <- as.data.frame(ACorrSim1[with(ACorrSim34, order(UniqueID, Y)),])
ACorrSim35 <- as.data.frame(ACorrSim1[with(ACorrSim35, order(UniqueID, Y)),])
ACorrSim36 <- as.data.frame(ACorrSim1[with(ACorrSim36, order(UniqueID, Y)),])
ACorrSim37 <- as.data.frame(ACorrSim1[with(ACorrSim37, order(UniqueID, Y)),])
ACorrSim38 <- as.data.frame(ACorrSim1[with(ACorrSim38, order(UniqueID, Y)),])
ACorrSim39 <- as.data.frame(ACorrSim1[with(ACorrSim39, order(UniqueID, Y)),])
ACorrSim40 <- as.data.frame(ACorrSim1[with(ACorrSim40, order(UniqueID, Y)),])


##################SIMULATED DATA RANDOM FOREST ANALYSIS##########
#Set seed once only at the beginning#
set.seed(1015)

##UPDATE DATA AND GROUP NUMBER HERE FOR EACH SET:###
data <- ACorrSim1
CorrSIMGroup <- 1

#####################################################################

##########RUN ALL CODE BELOW AS SINGLE BLOCK#################
  datagroup <- CorrSIMGroup
  count<-length(unique(data$UniqueID))
  test <- list()

  datalist <- list()
  sample <- list()
```

```
  model_sample <- list()
  validate_sample <- list()
  VSURF.output <- list()
  predictor_num <- list()
  n <- list()
  actual <- list()
  predicted <- list()
  R2 <- list()
  adjR2 <- list()
  predictors <- list()
  thres_predictors <- list()
  var_imp <- list()
  correlate <- list()
  R2_predict <- list()
  MSE_predict <- list()
  PRESS_predict <- list()

  t1 <- Sys.time()
  for(k in 1:count){
  t2 <- Sys.time()
  datalist[[k]] <- as.data.frame(subset(data, UniqueID == k))

  VSURF.output[[k]] <- VSURF(datalist[[k]][,3:6], datalist[[k]]$Y, ntree = 2000,
                    mtry = max(floor(ncol(datalist[[k]][,3:6])/3), 1),
                    nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1)

  n[[k]] <- nrow(datalist[[k]])
  predictor_num[[k]] <- nrow(as.data.frame(VSURF.output[[k]]$varselect.interp))

  datalist[[k]]$actual <- datalist[[k]]$Y
  datalist[[k]]$predicted <- predict(VSURF.output[[k]], datalist[[k]], step = c("interp"))

  R2[[k]] <- 1 - (sum((datalist[[k]]$actual-
datalist[[k]]$predicted)^2)/sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))
  adjR2[[k]]  <- 1-((sum((datalist[[k]]$actual-datalist[[k]]$predicted)^2))/(n[[k]]-
predictor_num[[k]]-1))/((sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))/n[[k]]-
1)

  predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.interp)
  predictors[[k]]$predictor_num <- predictors[[k]]$`VSURF.output[[k]]$varselect.interp`
  predictors[[k]]$term <- colnames(datalist[[k]])[2+predictors[[k]]$predictor_num]
  predictors[[k]] <- subset(predictors[[k]], select = -1)
  predictors[[k]]$rank <- row.names(predictors[[k]])
  CORSIMincluded_predictors <- map_df(predictors, ~as.data.frame(.x),
.id="UniqueID")
```

```
  thres_predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.thres)
  thres_predictors[[k]]$predictor_num <-
thres_predictors[[k]]$`VSURF.output[[k]]$varselect.thres`
  thres_predictors[[k]]$term <-
colnames(datalist[[k]])[2+thres_predictors[[k]]$predictor_num]
  thres_predictors[[k]] <- select(thres_predictors[[k]], -1)
  thres_predictors[[k]]$var_importance <- VSURF.output[[k]]$imp.varselect.thres
  thres_predictors[[k]]$rank <- row.names(thres_predictors[[k]])

  CORSIMthreshold_step <- map_df(thres_predictors, ~as.data.frame(.x),
.id="UniqueID")

  var_imp[[k]] <- as.data.frame(VSURF.output[[k]]$imp.mean.dec.ind)
  var_imp[[k]]$predictor_num <- var_imp[[k]]$`VSURF.output[[k]]$imp.mean.dec.ind`
  var_imp[[k]]$term <- colnames(datalist[[k]])[2+var_imp[[k]]$predictor_num]
  var_imp[[k]] <- select(var_imp[[k]], -1)
  var_imp[[k]]$var_importance <- VSURF.output[[k]]$imp.mean.dec
  var_imp[[k]]$rank <- row.names(var_imp[[k]])

  CORSIMvar_importance <- map_df(var_imp, ~as.data.frame(.x), .id="UniqueID")

  print(judgeanalysis <- Sys.time()-t2)
}
print(totalanalysis <- (Sys.time()-t1))


##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMincluded_predictors, file =
paste0("C:/Users/Kristina/Documents/Ohio University/Dissertation 9.19.17/R
Code/Study Three Analysis/ACorrSim", datagroup, " CORSIM Random Forest Final
Predictors.csv", sep = ""))
write.csv(CORSIMthreshold_step, file = paste0("C:/Users/Kristina/Documents/Ohio
University/Dissertation 9.19.17/R Code/Study Three Analysis/ACorrSim", datagroup, "
CORSIM Random Forest Threshold Predictors.csv", sep = ""))
write.csv(CORSIMvar_importance, file = paste0("C:/Users/Kristina/Documents/Ohio
University/Dissertation 9.19.17/R Code/Study Three Analysis/ACorrSim", datagroup, "
CORSIM Random Forest Variable Importance.csv", sep = ""))

##############################################
### Model Statistics ###

R2_model <- map_df(R2, ~as.data.frame(.x), .id = "UniqueID")
R2_model$R2_model <- R2_model$.x
R2_model <- select(R2_model, -2)
```

```
adjR2_model <- map_df(adjR2, ~as.data.frame(.x), .id = "UniqueID")
adjR2_model$adjrR2 <- adjR2_model$.x
adjR2_model <- select(adjR2_model, -2)

### Merging Model Stats ###
R_model_stats <- merge(R2_model, adjR2_model, by = c('UniqueID'), all.x = TRUE)

#Adding Variable Count to File#
final_model <- as.data.frame(subset(CORSIMincluded_predictors))
predictors <- c("UniqueID", "term", "rank")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]

predictorlist <- list()
statslist <- list()
for(i in 1:count){
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, UniqueID == i))
  statslist[[i]] <- as.data.frame(subset(R_model_stats, UniqueID == i))

  statslist[[i]]$model_var <- nrow(predictorlist[[i]])
}

R_model_stats <- map_df(statslist, ~as.data.frame(.x), .id="UniqueID")

#########################################################################

modelVar <- c("UniqueID", "term", "var_importance", "rank")

model_estimates <- CORSIMvar_importance[complete.cases(CORSIMvar_importance[,
modelVar]), modelVar]

person_model <- list()
term_model <- list()
forest_model <- list()
final_model <- list()
weights <- list()

for(k in 1:count){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, UniqueID == k))

  forest_model[[k]] <- as.data.frame(subset(R_model_stats, UniqueID == k))

  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
```

```
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x)
mean(x$var_importance)))
  final_model[[k]] <- as.data.frame(subset(final_model_predictors, UniqueID == k))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  forest_model[[k]]$intercept <- sum(ifelse(final_model[[k]]$term == '(Intercept)', 1, 0))
  forest_model[[k]]$X1 <- sum(ifelse(final_model[[k]]$term == 'X1', 1, 0))
  forest_model[[k]]$X2 <- sum(ifelse(final_model[[k]]$term == 'X2', 1, 0))
  forest_model[[k]]$X3 <- sum(ifelse(final_model[[k]]$term == 'X3', 1, 0))
  forest_model[[k]]$X4 <- sum(ifelse(final_model[[k]]$term == 'X4', 1, 0))

  forest_model[[k]]$intercept_VI <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X1_VI <- sum(ifelse(weights[[k]]$Predictor == 'X1',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X2_VI <- sum(ifelse(weights[[k]]$Predictor == 'X2',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X3_VI <- sum(ifelse(weights[[k]]$Predictor == 'X3',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X4_VI <- sum(ifelse(weights[[k]]$Predictor == 'X4',
weights[[k]]$Mean, 0))

}

R_model_stats <- map_df(forest_model, ~as.data.frame(.x), .id="UniqueID")

Conditions <- subset(data, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "rStructure")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

R_model_stats <- subset(R_model_stats, select=-c(UniqueID, UniqueID.1))
CORSIMR_model_stats <- merge(condition, R_model_stats, by = c('UniqueID'), all.x =
TRUE)

##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMR_model_stats, file =
paste0("C:/Users/Kristina/Documents/ACorrSim", datagroup, " CORSIM Random Forest
Model Stats.csv", sep = ""))
```

Random Forest Analysis Data Structure B

```
#install.packages("VSURF")
#install.packages("randomForest")
```

```
#install.packages("party")
#install.packages("readxl")
#install.packages("purrr")
#install.packages("data.table")
#install.packages("dplyr")

rm(list = ls())

library(VSURF)
library(randomForest)
library(party)
library(readxl)
library(purrr)
library(data.table)
library(dplyr)

#Data from Personal PC: PC1
BCorrSim1 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim1.csv",
header=TRUE, sep=",")
BCorrSim2 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim2.csv",
header=TRUE, sep=",")
BCorrSim3 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim3.csv",
header=TRUE, sep=",")
BCorrSim4 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim4.csv",
header=TRUE, sep=",")
BCorrSim5 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim5.csv",
header=TRUE, sep=",")
BCorrSim6 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim6.csv",
header=TRUE, sep=",")
BCorrSim7 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim7.csv",
header=TRUE, sep=",")
BCorrSim8 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim8.csv",
header=TRUE, sep=",")
BCorrSim9 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim9.csv",
header=TRUE, sep=",")
BCorrSim10 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim10.csv",
header=TRUE, sep=",")


BCorrSim11 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim11.csv",
header=TRUE, sep=",")
BCorrSim12 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim12.csv",
header=TRUE, sep=",")
BCorrSim13 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim13.csv",
header=TRUE, sep=",")
```

```
BCorrSim14 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim14.csv",
header=TRUE, sep=",")
BCorrSim15 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim15.csv",
header=TRUE, sep=",")
BCorrSim16 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim16.csv",
header=TRUE, sep=",")
BCorrSim17 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim17.csv",
header=TRUE, sep=",")
BCorrSim18 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim18.csv",
header=TRUE, sep=",")
BCorrSim19 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim19.csv",
header=TRUE, sep=",")
BCorrSim20 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim20.csv",
header=TRUE, sep=",")

BCorrSim21 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim21.csv",
header=TRUE, sep=",")
BCorrSim22 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim22.csv",
header=TRUE, sep=",")
BCorrSim23 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim23.csv",
header=TRUE, sep=",")
BCorrSim24 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim24.csv",
header=TRUE, sep=",")
BCorrSim25 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim25.csv",
header=TRUE, sep=",")
BCorrSim26 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim26.csv",
header=TRUE, sep=",")
BCorrSim27 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim27.csv",
header=TRUE, sep=",")
BCorrSim28 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim28.csv",
header=TRUE, sep=",")
BCorrSim29 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim29.csv",
header=TRUE, sep=",")
BCorrSim30 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim30.csv",
header=TRUE, sep=",")

BCorrSim31 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim31.csv",
header=TRUE, sep=",")
BCorrSim32 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim32.csv",
header=TRUE, sep=",")
BCorrSim33 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim33.csv",
header=TRUE, sep=",")
BCorrSim34 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim34.csv",
header=TRUE, sep=",")
BCorrSim35 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim35.csv",
header=TRUE, sep=",")
```

```
BCorrSim36 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim36.csv",
header=TRUE, sep=",")
BCorrSim37 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim37.csv",
header=TRUE, sep=",")
BCorrSim38 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim38.csv",
header=TRUE, sep=",")
BCorrSim39 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim39.csv",
header=TRUE, sep=",")
BCorrSim40 <- read.csv(file="C:/Users/Kristina/Documents/BCorrSim40.csv",
header=TRUE, sep=",")

BCorrSim1 <- as.data.frame(BCorrSim1[with(BCorrSim1, order(UniqueID, Y)),])
BCorrSim2 <- as.data.frame(BCorrSim1[with(BCorrSim2, order(UniqueID, Y)),])
BCorrSim3 <- as.data.frame(BCorrSim1[with(BCorrSim3, order(UniqueID, Y)),])
BCorrSim4 <- as.data.frame(BCorrSim1[with(BCorrSim4, order(UniqueID, Y)),])
BCorrSim5 <- as.data.frame(BCorrSim1[with(BCorrSim5, order(UniqueID, Y)),])
BCorrSim6 <- as.data.frame(BCorrSim1[with(BCorrSim6, order(UniqueID, Y)),])
BCorrSim7 <- as.data.frame(BCorrSim1[with(BCorrSim7, order(UniqueID, Y)),])
BCorrSim8 <- as.data.frame(BCorrSim1[with(BCorrSim8, order(UniqueID, Y)),])
BCorrSim9 <- as.data.frame(BCorrSim1[with(BCorrSim9, order(UniqueID, Y)),])
BCorrSim10 <- as.data.frame(BCorrSim1[with(BCorrSim10, order(UniqueID, Y)),])

BCorrSim11 <- as.data.frame(BCorrSim1[with(BCorrSim11, order(UniqueID, Y)),])
BCorrSim12 <- as.data.frame(BCorrSim1[with(BCorrSim12, order(UniqueID, Y)),])
BCorrSim13 <- as.data.frame(BCorrSim1[with(BCorrSim13, order(UniqueID, Y)),])
BCorrSim14 <- as.data.frame(BCorrSim1[with(BCorrSim14, order(UniqueID, Y)),])
BCorrSim15 <- as.data.frame(BCorrSim1[with(BCorrSim15, order(UniqueID, Y)),])
BCorrSim16 <- as.data.frame(BCorrSim1[with(BCorrSim16, order(UniqueID, Y)),])
BCorrSim17 <- as.data.frame(BCorrSim1[with(BCorrSim17, order(UniqueID, Y)),])
BCorrSim18 <- as.data.frame(BCorrSim1[with(BCorrSim18, order(UniqueID, Y)),])
BCorrSim19 <- as.data.frame(BCorrSim1[with(BCorrSim19, order(UniqueID, Y)),])
BCorrSim20 <- as.data.frame(BCorrSim1[with(BCorrSim20, order(UniqueID, Y)),])

BCorrSim21 <- as.data.frame(BCorrSim1[with(BCorrSim21, order(UniqueID, Y)),])
BCorrSim22 <- as.data.frame(BCorrSim1[with(BCorrSim22, order(UniqueID, Y)),])
BCorrSim23 <- as.data.frame(BCorrSim1[with(BCorrSim23, order(UniqueID, Y)),])
BCorrSim24 <- as.data.frame(BCorrSim1[with(BCorrSim24, order(UniqueID, Y)),])
BCorrSim25 <- as.data.frame(BCorrSim1[with(BCorrSim25, order(UniqueID, Y)),])
BCorrSim26 <- as.data.frame(BCorrSim1[with(BCorrSim26, order(UniqueID, Y)),])
BCorrSim27 <- as.data.frame(BCorrSim1[with(BCorrSim27, order(UniqueID, Y)),])
BCorrSim28 <- as.data.frame(BCorrSim1[with(BCorrSim28, order(UniqueID, Y)),])
BCorrSim29 <- as.data.frame(BCorrSim1[with(BCorrSim29, order(UniqueID, Y)),])
BCorrSim30 <- as.data.frame(BCorrSim1[with(BCorrSim30, order(UniqueID, Y)),])

BCorrSim31 <- as.data.frame(BCorrSim1[with(BCorrSim31, order(UniqueID, Y)),])
BCorrSim32 <- as.data.frame(BCorrSim1[with(BCorrSim32, order(UniqueID, Y)),])
```

```
BCorrSim33 <- as.data.frame(BCorrSim1[with(BCorrSim33, order(UniqueID, Y)),])
BCorrSim34 <- as.data.frame(BCorrSim1[with(BCorrSim34, order(UniqueID, Y)),])
BCorrSim35 <- as.data.frame(BCorrSim1[with(BCorrSim35, order(UniqueID, Y)),])
BCorrSim36 <- as.data.frame(BCorrSim1[with(BCorrSim36, order(UniqueID, Y)),])
BCorrSim37 <- as.data.frame(BCorrSim1[with(BCorrSim37, order(UniqueID, Y)),])
BCorrSim38 <- as.data.frame(BCorrSim1[with(BCorrSim38, order(UniqueID, Y)),])
BCorrSim39 <- as.data.frame(BCorrSim1[with(BCorrSim39, order(UniqueID, Y)),])
BCorrSim40 <- as.data.frame(BCorrSim1[with(BCorrSim40, order(UniqueID, Y)),])


################SIMULATED DATA RANDOM FOREST ANALYSIS##########
#Set seed once only at the beginning#
set.seed(1015)

##UPDATE DATA AND GROUP NUMBER HERE FOR EACH SET:###
data <- BCorrSim1
CorrSIMGroup <- 1
#####################################################################

##########RUN ALL CODE BELOW AS SINGLE BLOCK#################
 datagroup <- CorrSIMGroup
 count<-length(unique(data$UniqueID))
 test <- list()

 datalist <- list()
 sample <- list()
 model_sample <- list()
 validate_sample <- list()
 VSURF.output <- list()
 predictor_num <- list()
 n <- list()
 actual <- list()
 predicted <- list()
 R2 <- list()
 adjR2 <- list()
 predictors <- list()
 thres_predictors <- list()
 var_imp <- list()
 correlate <- list()
 R2_predict <- list()
 MSE_predict <- list()
 PRESS_predict <- list()

 t1 <- Sys.time()
 for(k in 1:count){
 t2 <- Sys.time()
```

```
  datalist[[k]] <- as.data.frame(subset(data, UniqueID == k))

  VSURF.output[[k]] <- VSURF(datalist[[k]][,3:5], datalist[[k]]$Y, ntree = 2000,
                    mtry = max(floor(ncol(datalist[[k]][,3:5])/3), 1),
                    nfor.thres = 50, nmin = 1, nfor.interp = 25, nsd = 1)

  n[[k]] <- nrow(datalist[[k]])
  predictor_num[[k]] <- nrow(as.data.frame(VSURF.output[[k]]$varselect.interp))

  datalist[[k]]$actual <- datalist[[k]]$Y
  datalist[[k]]$predicted <- predict(VSURF.output[[k]], datalist[[k]], step = c("interp"))

  R2[[k]] <- 1 - (sum((datalist[[k]]$actual-
datalist[[k]]$predicted)^2)/sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))
  adjR2[[k]]  <- 1-((sum((datalist[[k]]$actual-datalist[[k]]$predicted)^2))/(n[[k]]-
predictor_num[[k]]-1))/((sum((datalist[[k]]$actual-mean(datalist[[k]]$actual))^2))/n[[k]]-
1)

  predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.interp)
  predictors[[k]]$predictor_num <- predictors[[k]]$`VSURF.output[[k]]$varselect.interp`
  predictors[[k]]$term <- colnames(datalist[[k]])[2+predictors[[k]]$predictor_num]
  predictors[[k]] <- subset(predictors[[k]], select = -1)
  predictors[[k]]$rank <- row.names(predictors[[k]])
  CORSIMincluded_predictors <- map_df(predictors, ~as.data.frame(.x),
.id="UniqueID")

  thres_predictors[[k]] <- as.data.frame(VSURF.output[[k]]$varselect.thres)
  thres_predictors[[k]]$predictor_num <-
thres_predictors[[k]]$`VSURF.output[[k]]$varselect.thres`
  thres_predictors[[k]]$term <-
colnames(datalist[[k]])[2+thres_predictors[[k]]$predictor_num]
  thres_predictors[[k]] <- select(thres_predictors[[k]], -1)
  thres_predictors[[k]]$var_importance <- VSURF.output[[k]]$imp.varselect.thres
  thres_predictors[[k]]$rank <- row.names(thres_predictors[[k]])

  CORSIMthreshold_step <- map_df(thres_predictors, ~as.data.frame(.x),
.id="UniqueID")

  var_imp[[k]] <- as.data.frame(VSURF.output[[k]]$imp.mean.dec.ind)
  var_imp[[k]]$predictor_num <- var_imp[[k]]$`VSURF.output[[k]]$imp.mean.dec.ind`
  var_imp[[k]]$term <- colnames(datalist[[k]])[2+var_imp[[k]]$predictor_num]
  var_imp[[k]] <- select(var_imp[[k]], -1)
  var_imp[[k]]$var_importance <- VSURF.output[[k]]$imp.mean.dec
  var_imp[[k]]$rank <- row.names(var_imp[[k]])

  CORSIMvar_importance <- map_df(var_imp, ~as.data.frame(.x), .id="UniqueID")
```

```
  print(judgeanalysis <- Sys.time()-t2)
}
print(totalanalysis <- (Sys.time()-t1))


##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMincluded_predictors, file =
paste0("C:/Users/Kristina/Documents/Ohio University/Dissertation 9.19.17/R
Code/Study Three Analysis/BCorrSIM", datagroup, " CORSIM Random Forest Final
Predictors.csv", sep = ""))
write.csv(CORSIMthreshold_step, file = paste0("C:/Users/Kristina/Documents/Ohio
University/Dissertation 9.19.17/R Code/Study Three Analysis/BCorrSIM", datagroup, "
CORSIM Random Forest Threshold Predictors.csv", sep = ""))
write.csv(CORSIMvar_importance, file = paste0("C:/Users/Kristina/Documents/Ohio
University/Dissertation 9.19.17/R Code/Study Three Analysis/BCorrSIM", datagroup, "
CORSIM Random Forest Variable Importance.csv", sep = ""))

##############################################
### Model Statistics ###

R2_model <- map_df(R2, ~as.data.frame(.x), .id = "UniqueID")
R2_model$R2_model <- R2_model$.x
R2_model <- select(R2_model, -2)


adjR2_model <- map_df(adjR2, ~as.data.frame(.x), .id = "UniqueID")
adjR2_model$adjrR2 <- adjR2_model$.x
adjR2_model <- select(adjR2_model, -2)

### Merging Model Stats ###
R_model_stats <- merge(R2_model, adjR2_model, by = c('UniqueID'), all.x = TRUE)

#Adding Variable Count to File#
final_model <- as.data.frame(subset(CORSIMincluded_predictors))
predictors <- c("UniqueID", "term", "rank")

final_model_predictors <- final_model[complete.cases(final_model[, predictors]),
predictors]

predictorlist <- list()
statslist <- list()
for(i in 1:count){
  predictorlist[[i]] <- as.data.frame(subset(final_model_predictors, UniqueID == i))
  statslist[[i]] <- as.data.frame(subset(R_model_stats, UniqueID == i))
```

```
  statslist[[i]]$model_var <- nrow(predictorlist[[i]])
}

R_model_stats <- map_df(statslist, ~as.data.frame(.x), .id="UniqueID")

########################################################################

modelVar <- c("UniqueID", "term", "var_importance", "rank")

model_estimates <- CORSIMvar_importance[complete.cases(CORSIMvar_importance[,
modelVar]), modelVar]

person_model <- list()
term_model <- list()
forest_model <- list()
final_model <- list()
weights <- list()

for(k in 1:count){
  t2 <- Sys.time()
  person_model[[k]] <- as.data.frame(subset(model_estimates, UniqueID == k))

  forest_model[[k]] <- as.data.frame(subset(R_model_stats, UniqueID == k))

  term_model[[k]] <- split(person_model[[k]], person_model[[k]]$term)
  weights[[k]] <- as.data.frame(sapply(term_model[[k]], function(x)
mean(x$var_importance)))
  final_model[[k]] <- as.data.frame(subset(final_model_predictors, UniqueID == k))
  setDT(weights[[k]], keep.rownames = TRUE)[]
  setnames(weights[[k]], 1, "Predictor")
  setnames(weights[[k]], 2, "Mean")

  forest_model[[k]]$intercept <- sum(ifelse(final_model[[k]]$term == '(Intercept)', 1, 0))
  forest_model[[k]]$X1 <- sum(ifelse(final_model[[k]]$term == 'X1', 1, 0))
  forest_model[[k]]$X2 <- sum(ifelse(final_model[[k]]$term == 'X2', 1, 0))
  forest_model[[k]]$X3 <- sum(ifelse(final_model[[k]]$term == 'X3', 1, 0))

  forest_model[[k]]$intercept_VI <- sum(ifelse(weights[[k]]$Predictor == '(Intercept)',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X1_VI <- sum(ifelse(weights[[k]]$Predictor == 'X1',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X2_VI <- sum(ifelse(weights[[k]]$Predictor == 'X2',
weights[[k]]$Mean, 0))
  forest_model[[k]]$X3_VI <- sum(ifelse(weights[[k]]$Predictor == 'X3',
weights[[k]]$Mean, 0))
```

```
}

forest_model[[5]]

R_model_stats <- map_df(forest_model, ~as.data.frame(.x), .id="UniqueID")

Conditions <- subset(data, !duplicated(UniqueID))
ConditionVar <- c("UniqueID", "rStructure")
condition <- Conditions[complete.cases(Conditions[, ConditionVar]), ConditionVar]

R_model_stats <- subset(R_model_stats, select=-c(UniqueID, UniqueID.1))
CORSIMR_model_stats <- merge(condition, R_model_stats, by = c('UniqueID'), all.x =
TRUE)

##SAVING ON PERSONAL PC: PC1###.
write.csv(CORSIMR_model_stats, file =
paste0("C:/Users/Kristina/Documents/BCorrSIM", datagroup, " CORSIM Random Forest
Model Stats.csv", sep = ""))
```

Thesis and Dissertation Services