STATISTICS IN THE PRESENCE OF COST: COST-CONSIDERATE VARIABLE SELECTION AND MCMC CONVERGENCE DIAGNOSTICS

by

Michael David Lerch

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Statistics

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2016

ACKNOWLEDGEMENTS

My first acknowledgement deservedly belongs to my advisor Dr. Megan Higgs. I have seen few people in academia work harder than Megan, and this dedication to her work is what drove me to her as an advisor.

Of course, the rest of my committee (in no particular order) deserves proper recognition. Thank you to John Borkowski whose classes inspired a number of side projects to distract the completion of this dissertation. Thank you to Mark Greenwood who taught the first *real* statistics class I attended in GLM; I spent the next few years piecing together what I had learned. Thank you to Jim Robison-Cox for introducing me to `knitr` and `shiny` and all the ways to turn doldrum statistics into a fun challenge. And, thank you to Steve Cherry who was always paying more attention that it seemed.

Lastly, thank you to Kezia who has been a source of inspiration and comfort that extends well beyond this disseratation.

TABLE OF CONTENTS

TABLE OF CONTENTS - CONTINUED

TABLE OF CONTENTS - CONTINUED

LIST OF TABLES

LIST OF TABLES - CONTINUED

LIST OF FIGURES

LIST OF FIGURES - CONTINUED

LIST OF FIGURES - CONTINUED

ABSTRACT

The overarching objective of this research is to address and recognize the cost-benefit trade-off inherent in much of statistics. We identify two places where such a balance is present for researchers: variable selection and Markov chain Monte Carlo (MCMC) sampling.

An easily identifiable source of cost in science occurs when taking measurements. Researchers measure variables to estimate another quantity based on a model. When model building, researchers may have access to a large number of variables to include in the model and may consider using a subset of the variables so that future uses of the model need only measure this subset rather than all variables. The researchers are incentivized to proceed in this manner if some variables are prohibitively expensive to measure for future uses of the model. In this research, we present a new algorithm for cost-considerate variable selection in linear modeling when confronted with this problem. Since overfitting may be a danger when many variables at the disposal of the researcher, we build on the LARS and Lasso algorithms to perform cost-based variable selection in concert with model regularization.

In MCMC sampling for Bayesian statistics, the cost-benefit trade-off is unavoidable. Researchers sampling from a posterior distribution must run a sampler for some number of iterations before finally stopping the sampler to make inference on the finite number of samples drawn. In this situation, the cost to be reduced is time to run the sampler while realizing the longer the sampler is run, the better the convergence. Time may not be as tangible a cost as a dollar figure, but increased wait time to perform analyses incurs the cost of running a computer and any negative effects associated with a delay as the researcher waits until the sampler has finished running. In this research, we introduce new convergence assessment tools in a diagnostic and plot. Unlike commonly used convergence diagnostics, these new tools focus explicitly on posterior quantiles and probabilities which are common inferential objectives in Bayesian statistics. Additionally, we introduce equivalence testing to the convergence assessment domain by using it as the framework of the diagnostic.

CHAPTER ONE

INTRODUCTION

On February 11th, 2016, a large team of physicists officially announced the discovery of gravitational waves [Abbott, 2016]. Heretofore unobserved but predicted by Einstein's theory of relativity, confirmation of gravitational waves marked an extremely important milestone in science. The physics of gravitational waves is beyond the scope of this dissertation. However, an analogy is enough to describe the detection mechanism. Gravitational waves propagate through a compression of space itself. Consider a stretched slinky given a slight kick. The ripple of that kick can be seen travelling through the slinky. Before the kick, pick two parts of the slinky and measure the distance apart. Now, as the kick propagates through the slinky, those two parts maintain the same separation distance until the ripple is passing through either of the points. As this happens, the distance between the parts of the slinky changes slightly. By monitoring this distance, we can determine if and when a slinky wave moves through. The same principle applies to detecting gravitational waves.

Instead of a slinky, astrophysicists built the world's largest apparatus called LISA (Laser Interferometer Space Antenna). LISA is a gigantic triangle built with three spacecraft for the corners and the edges composed of five million kilometer long laser beams (for context that is nearly 400 times the diameter of the earth). These laser beams allow the astrophysicists to precisely measure the distance between the spacecraft down to incredible precision. If a gravitational wave travels through space in a direction that is along one of these edges, the distance between the spacecraft

changes ever so slightly (much like the distance between two parts of a slinky) and a gravitational wave is detected. Naturally, producing the world's largest apparatus is not a cheap undertaking. The collaboration of NASA and the European Space Agency spent 2 billion dollars to see LISA take to the heavens.

Actually, that is not quite what happened. The collaboration between NASA and the ESA dissolved under that 2 billion dollar price tag, and the dream of detecting gravitational waves came crashing to the ground, quite literally. The device was never built and instead, NASA took up the search alone and built two ground-based interferometers, one in Livingston, Louisiana and one in Hanford, Washington. With the earth's diameter about 400 times smaller than the proposed size of LISA, these ground-based detectors were slightly smaller than LISA would have been. Also, being on earth's surface, these detectors are subjected to earthquakes, weather events, and even large trucks driving near by that create enough rumbling to trigger the detectors. However, with precision calculations and multiple detectors to cancel out local noise, these ground based-based detectors were able to detect the gravitational waves created by two incredibly massive black holes circling each other and at a much lower cost than the proposed LISA.

Following this result, one may question the original, expensive plan to build LISA. Was the proposed two billion dollar bill foolish when gravitational waves could be detected for much cheaper with a ground-based apparatus? Or, is there more nuance than just whether or not a gravitational wave could be detected that should go into evaluating the cost efficiency of the program? Some even ask if we *still* spent too much money [Horgan, 2016].

It's hard to balance the cost-benefit trade-off when performing a major undertaking like this, especially when the future benefits may not yet be known. Though the stakes may be lower, carefully analyzing the cost-benefit trade-off for less

prestigious scientific research may be even more important relative to the scale of the undertaking. Economists study the push-pull of this in everyday life.

In this research, we investigate two places where a cost-benefit trade-off comes into play for the practicing statistician: variable selection and MCMC sampling. In variable selection, researchers may not recognize they are walking such a balance. Researchers may have a set of variables they want to measure and then measure those variables without a second thought. However, it is almost always clear that more variables *could* be measured and often, the researcher might be able to address their questions with fewer variables as well. In MCMC sampling, the cost-benefit trade-off is unavoidable. Researchers using a computer to sample from a posterior must run the sampler for some number of iterations and must make a decision when to finally stop the sampler to make inference on the finite number of samples drawn. In this situation, the cost we would like to reduce is the cost of time to run the sampler while realizing that the longer the sampler is run, the better the convergence. Time may not be as tangible a cost as a dollar figure, but increased wait time to perform analyses incurs the cost to run a computer and can negatively affect additional analyses that must wait until the sampler has finished running.

1.1 Variable Selection and Model Regularization

When performing statistical analyses, the phrase *variable selection* may refer to a process of deciding which variables to measure and is connected to hypothesis construction, or it may refer to an often iterative process of choosing from a collection of already measured variables to build a model performing best by some measure. The former case can be carried out before collecting data by working to align a model with the research question and study design. The analyst may wish to predict a response variable given a set of explanatory variables and must decide what explanatory

variables to measure in order to predict, or may wish to examine a relationship between the mean of the response variable and a particular explanatory variable and need to decide what additional explanatory variable the analyst wishes to account for in their model. This aspect of variable selection is rooted in knowledge of the system being studied.

In this dissertation, we are interested in variable selection as it pertains to model regularization. Model regularization is the process of constraining a model fit in order to prevent overfitting. By overfitting, we mean modeling what a appears to be structure in the training data, though is noise when predicting new observations outside of the training data to the detriment of the model's performance on data outside of the training data. Training data is the data used to fit the model. Often, modelers wishing to prevent overfitting split their dataset into training and testing sets to fit the model and then test performance. There are many approaches to perform model regularization and variable selection is one approach of model regularization. Under variable selection, only a subset of the potential explanatory variables are used in the model, under the explicit premise of assessing out-of-sample model performance. Of course, using variable selection in this way may lead to scientific insights or additional research questions, but in this dissertation we typically use the phrase variable selection to refer to the model regularization technique with the express goal of producing models with high performance outside of the training data.

Perhaps the favorite aphorism of statisticians is "All models are wrong, but some are useful." This quote is typically attributed to the famous statistician George E. Box with variations of the quote appearing in publication as early as 1976 [Box, 1976]. Today, the phrase has its own wikipedia page [Wikipedia, 2016]. And yet, it sometimes feels as though this quote is used as a cliche and not universally appreciated or understood in the statistical modeling community at large. Leo Breiman's excellent

two cultures discussion piece is a recommended read for anyone who would like to better understand the *all models are wrong* quote [Breiman, 2001].

In the case of model regularization, Box's aphorism is incredibly poignant. When performing model regularization the concept of a model being right or wrong may be irrelevant and *performance* is the only driver. By eschewing any burden of truth and constraining the model fitting process, it is possible to produce a model that predicts better outside of the training data. Variable selection in particular can provide multiple benefits to the researcher. As a method of model regularization, variable selection can result in models that predict better on out-of-sample data. Also, there are benefits to producing a model with fewer variables when cost is considered. If researchers intend to produce a model and subsequently use that model to make predictions on new data, then a model with relatively fewer variables will typically incur less cost and effort to measure those variables for the future use case. Under this motivation, it is of practical importance to evaluate the relative trade-offs between model performance and variable measurement cost. In this dissertation, we present a method to help researchers balance performance and cost for linear models.

1.2 MCMC Sampling

Markov Chain Monte Carlo (MCMC) sampling has a rich and interesting history in science and statistics [Hastings, 1970, Metropolis et al., 1953, Robert and Casella, 2011]. MCMC is especially pertinent for Bayesian statistics. The use of Bayesian statistics may imply a philosophy of statistics and probability different than that of frequentist statistics. My own philosophy is shaped by statisticians like Dennis Lindley and E.T. Jaynes and a good book on Bayesian philosophy of uncertainty is Lindley [2006]. Putting aside the philosophy, the mathematical underpinnings of Bayesian statistics are expressed in Bayes' rule

$$p(\theta|x) \propto p(x|\theta)p(\theta) \ , \tag{1.1}$$

where $p(\theta)$ is the prior, $p(x|\theta)$ is the likelihood, and the posterior is $p(\theta|x)$. The proportionality constant for Bayes rule is $p(x)$. Subscribers to Bayesian philosophy find immense power in Bayes rule. The prior is a probabilistic expression of the degree of belief for the parameter θ before observing data and the likelihood provides the probability of observing the particular data as a function of θ. Given these functions, the posterior is the coherent updated degree of belief for θ for the data observed.

For the Bayesian statistician, mathematically getting as far as the right hand side of Equation 1.1 is trivial, but finding the proportionality constant may be difficult. Since the posterior is a probability distribution, it must integrate to 1. If $p(x|\theta)p(\theta)$ can be analytically integrated over θ, the normalization constant can be found. In special cases, the form of the posterior may be recognized as a known probability distribution in which case integrating is not necessary. Sometimes, researchers will purposefully select priors and likelihood pairs that combine into a known form for the posterior, thus saving aggravation in determining the normalization constant. However, these are special cases and for general problems, we must be prepared for the case that the integral is too difficult to calculate and the form is not recognizable. In these cases, MCMC can be used to approximate the posterior by making a series of draws which, as a set, converge to a sample of draws from the posterior distribution.

Using MCMC to approximate the posterior distribution with draws, the researcher must take enough draws so that the approximation to the posterior is adequate. This requires computing power and can be very time intensive.

An exercise to gain appreciation of the scale of this problem is to first consider a posterior distribution for a single parameter that has been transformed to lie between

0 and 1. This is a one dimensional problem with two "corners" at 0 and 1 and so is easy to visualize. Consider the number of draws one would require to fill the space between these corners and produce a histogram providing a precise and accurate representation of the analytic distribution. Now, consider the number of draws that would be necessary for a problem with two parameters and dimensions, in this case there are 4 corners to the parameter space. If we extend to the case of, say, a mere 10 variables, the parameter space has $2^{10} = 1024$ corners. In addition to just the sheer number of draws that are necessary to fill the space within all of the corners, MCMC is a Markov process implying that each draw is not actually an independent draw from the posterior distribution. Instead, the set of draws eventually converge to a sample from the posterior, so we need a large number of draws so that the parameter space of interest is filled and the collection of draws converge to the distribution of interest.

From a practical perspective, the researcher must weigh the costs and benefits to decide how long to run the MCMC before stopping and using the set of draws to make inference. Sampling from an MCMC algorithm longer will almost always improve inference. However, at some point, those improvements may no longer be worth the additional wait time (or the cost of electricity to run the computer). To help practitioners choose when to stop their samplers, a number of convergence *diagnostics* have been developed, and in fact (contrary to cost-based variable selection in Chapter 2) the literature is robust. In fact, a review on convergence diagnostics was written 20 years prior to this dissertation [Cowles and Carlin, 1996]. Each of these diagnostics has its own motivation and shortcomings, and we found the existing diagnostics do not fulfill all our needs. This may have something to do with the increase in computing power over those 20 years. Over that time, the complexity of models being fit with MCMC has expanded, but many of the same diagnostics of 1996 are still

used today. In this dissertation, we present a new diagnostic for MCMC convergence assessment to help researchers navigate the cost-benefit trade-off of accurate posterior approximation at the expense of additional sampling time.

1.3 Availability of Software

Another important cost-benefit question that practitioners must balance has to do with the availability of software. The latest and greatest machine learning algorithm is not nearly as attractive to most researchers if they have to code it themselves. They may find a canned random forest algorithm more than sufficient for their needs even if the newest algorithm might provide slightly more accurate predictions.

Realistically, the value of any computational algorithm is increased many times over when code implementing the algorithm is also available. Thus, it is important to provide usable software to supplement the algorithms I present in this dissertation. Producing software has its own challenges that are different from developing statistical algorithms. I present in this dissertation design choices, challenges, and the steps involved in producing open source academic software. In particular, I detail the process of developing packages to implement my variable selection and convergence diagnostic for the open source statistical program R.

1.4 Objective

The overarching objective of this dissertation is to address and recognize the cost-benefit trade-off inherent in much of statistics. We do this through concrete examples in the development of new statistical algorithms that have a cost-benefit trade-off as motivation.

In Chapter 2, we review pertinent examples from the model regression literature as well as examples from cost-considerate modeling algorithms. We then present a new algorithm, clars, fitting the intersection of these topics. The clars algorithm builds on the popular LARS and Lasso regularization algorithms. We hope our contribution is part of a movement where the quality of data relative to its cost is considered. As the phrase "big data" has entered the common vernacular, we believe the next push should be towards high quality data rather than even "bigger" data.

In Chapter 3, we review the history of MCMC convergence diagnostics and introduce the Quantile Equivalence Diagnostic (QED) for MCMC convergence assessment. The QED uses equivalence testing as the framework of the diagnostic which is new for MCMC convergence. Additionally, the QED explicitly addresses summaries that are commonly reported in Bayesian statistics: posterior credible intervals and quantiles. We believe our diagnostic and accompanying plot can be useful for nearly all users of MCMC for Bayesian statistics.

In Chapter 4, we review best practices when producing statistical software. Research shows scientists spend more and more of their time writing software [Hannay et al., 2009, Prabhu et al., 2011]. These scientists can be more efficient by following programming paradigms and using tools available for assistance in developing software. We call out specific tools and give concrete examples to help give researchers explicit direction in making their programming endeavours more efficient. The best practices and suggestions presented in this chapter were used for the development of software packages for clars and QED which supplement this dissertation.

Finally, in Chapter 5 presents concluding remarks, final thoughts, and possible directions of future research along these topics.

CHAPTER TWO

COST-CONSIDERATE VARIABLE SELECTION AND REGULARIZATION FOR LINEAR MODELS

2.1 Variable Selection and Model Regularization

The basic tool of the statistician is the model: given a set of inputs, a prediction is calculated. Especially in the case of linear models, a distinction between *inputs* and *variables* (or predictor variables) is often made. In general, an input represents an independent variable in the model expression and a variable may be a general function of the inputs. For example, inputs x_1 and x_2 may appear in a model in forms such as x_1, ${x_1}^2$, x_2, ${x_2}^2$, and $x_1 * x_2$ while still staying within linear modeling. Here, two inputs have been used to create five model variables, and clearly many more could be considered. In this work, we take a particular focus on variable selection for linear models and when variable cost is pertinent and also investigate model regularization.

Model regularization is the process by which a general model structure is somehow constrained to prevent overfitting. A model that is overfit is one in which the model fit is finely tuned to the particularities of a given dataset (the training data) to the detriment of performance or appropriateness of the model to future data. A common strategy to prevent overfitting is shrinking coefficient estimates toward zero. Meanwhile, variable selection is the process in which variables are chosen to be included or excluded in a model. Depending on the context and field of application, there are other phrases that can be used to describe variable selection. For example, subset selection is used to indicate that from a set of potential variables, only a subset are selected. Some use the phrase *model sparsity* to indicate that coefficients

of certain variables are equal to (or set to) zero, thus suggesting variable selection. Further, *model selection* is often used in the context of choosing variables in a model.

Variable selection can be considered a type of model regularization. Especially in cases where there are a large number of potential variables, it may be possible to find correlations between variables and the response in the training data by chance even when a relationship does not exist in the whole population. In such a case, the inclusion of these predictor variables in the model results in overfitting since future data are unlikely to follow the same pattern. If such spurious correlations are relatively small, then it may be possible to identify and remove the offending predictor variable from the model while retaining variables that are more highly correlated with the response. Thus, by performing variable selection, the researchers may also be implementing model regularization; in a sense, they are shrinking variable coefficients all the way to zero. However, in this document we tend to refer to variable selection and model regularization as two *separate* pieces of the model building process.

We distinguish between variable selection and model regularization for several reasons. First, simplicity or parsimony is often considered a goal in its own right [Breiman, 1995]. In this case the researcher may perform variable selection outright without considering other model regularization techniques. Second, as an implementation of model regularization, variable selection does not tend to perform as well as other regularization methods as measured by out-of-sample prediction [Hoerl et al., 1986]. And finally, our work explores an especially practical motivation for variable selection: reduction in the cost of data collection. The objective of this work is the development of a method for performing *both* model regularization and variable selection based on variable cost.

Variable selection will always be a challenge for statisticians and researchers using statistical methods. There is no dearth in approaches to variable selection

for statistical models and so the challenge is often deciding upon the method or understanding the results of the variable selection. Appropriate techniques often depend on the goal of the research, form of the model, type of response variable, type of predictor variables, and field of study. Philosophical considerations may also play a role in the choice of variable selection routine.

The overarching theme of variable selection techniques is the balance between a simple model and performance. The Akaike Information Criterion (AIC) [Akaike, 1974, Burnham and Anderson, 2002] and other model evaluation information criteria are often composed of two parts: a performance metric related to fit and a penalty term for *model complexity*, typically a function of the number of parameters in the model. As more variables are added to the model, the *complexity* increases and in-sample performance increases as well. Variable selection methods must identify how much of an increase in performance warrants including the additional variable. After some point of added model complexity, it is possible that in-sample performance increase is merely overfitting and performance of the model for out-of-sample data will decrease.

Though reducing the number of variables can combat overfitting, there are other methods to prevent overfitting rather than simply removing variables from the model. For linear models, penalized regression is used to *shrink* model coefficients toward zero which tends to improve out-of-sample model performance. Some methods, like Lasso (Least Absolute Shrinkage and Selection Operator), shrink some model coefficients and set others explicitly to zero, thus simultaneously performing variable selection and coefficient shrinking in one procedure which can greatly improve out-of-sample model performance [Tibshirani, 1996].

Meanwhile, a potentially vital characteristic goes unconsidered: variable cost. Much of the time, acquiring variables is not free. A common example is in the medical

field [He et al., 2012, Paclík et al., 2002, Turney, 2000, Weiss and Provost, 2003, Weiss et al., 2013, for example]. It is not difficult to imagine medical tests to be performed on a patient which vary in costs. As such, the doctor may elect to perform only a subset of the available tests. Moreover, there may be multiple tests aiming to measure the same, possibly latent, quantity. Though it may be reasonable to expect more expensive tests to perform better, there may be counterexamples or cases where the degree of improvement in performance may not justify an increase in cost.

There are other examples where the measurement cost of variables is non-negligible. In many disciplines of science, researchers must physically measure variables, and instrument cost, technician time, and maintenance fees add up. Even in data science and machine learning communities, where a raw data source is often already electronically available, cost-considerate variable selection can be beneficial. Producing *features* from the raw data source for machine learning can be costly due to expensive computing time and also human time for data cleaning and feature engineering. Cost is a real-world factor, and making the decision of what variables to collect based on cost deserves critical thought.

Naturally, many researchers are already performing a type of cost-considerate variable selection when they make decisions about what measurements they will take to produce model variables. Consciously or not, scientific researchers rely on the apparatus already owned rather than continually purchasing devices that are more precise than those currently owned. Most social researchers who study people (of this and past generations at least) will spend their entire careers having never performed full genome sequencing on their subjects despite one's genes being a fundamental component of the person. Machine learning engineers do not consider every possible interaction and transformation in their process of *feature engineering*, as it is too time consuming. Suggesting that researchers *do* exhaust all possible measurements is, of

course, ludicrous. However, there may be situations where a potential set of variables that could be collected is listed and the quantities that will actually be measured are chosen from that list based on critical evaluation. Our goal is to introduce a new method for variable selection to directly incorporate variable cost.

A seemingly paradoxical element to cost-considerate variable selection is that, to appropriately evaluate the performance of a variable, it must be measured, thus incurring cost! However, there are many scenarios where this apparent conundrum is not pertinent. In many machine learning problems, the entire pipeline of feature engineering, from a raw data source to model evaluation, can be programmed to a single hierarchical algorithm. Here, it may be possible to engineer a cost efficient subset of potential features, fit the model with those features, evaluate the performance of the model, and repeat after engineering additional features if a desired performance metric is not obtained. This process assumes the feature engineering step is the cost prohibitive one rather than the model fitting and evaluation steps.

In disciplines where variable measurement and model fitting are not so connected, like sciences requiring experimentation, field work, or performing medical diagnoses on a patient, there are still common contexts that can benefit from a cost-considerate variable selection algorithm. In one context, the researcher fits a model using historic data. This historic dataset may be stored in a database or is otherwise available at negligible cost. Using the variables available in the historic data, the researcher builds a model, thus far incurring no cost aside from researcher time. If the goal of the researcher is to make inferences based on model parameters, then the researcher may have little need for cost based variable selection. However, it is possible that the objective is to predict a future observation on a new subject or study unit. Though the historic dataset was available at negligible current cost, the variables will most likely have to be collected in order to predict the new observation.

Thus, the researcher may wish to select among the variables in the historic data set in order to produce a model for which predictions of new observations can be made at a reasonable cost.

The other context we present is that of pilot studies. Pilot studies are typically used to inform study design and analysis for future larger studies. For this context, a research objective is posed and a pilot dataset is collected with some cost. Here, a smaller number of observations are taken than intended for the large study, but more variables are measured in the pilot study than intended to be measured in the larger study. Based on the analysis of the pilot study data, a subset of the pilot variables are then measured in the final study.

The pilot study context may be similar to the historic data context in that the raw data source may already exist electronically. In the era of big data and machine learning, measuring or recording data may not be the cost prohibitive step. The cost prohibitive step may be the processing or cleaning of the data. Hence, it is possible that researchers who already have access to a digital data source may wish to perform a pilot study from this data source in order to determine how much time, and therefore money, is spent in processing the dataset into potentially useful variables. Following the pilot study, an efficient subset of the variables will be generated when processing the rest of the data source.

2.2 Related Work in Cost-Considerate Modeling

Despite the vastness of variable selection methods, cost-considerate modeling has gained relatively little exposure and research. Much of the research that does exist is in the machine learning domain, itself a young field. However, that is not to say that statisticians have never considered cost in variable selection. The great Bayesian, often ahead of his time, Dennis Lindley considered the cost of variables

as early as 1968 when he detailed a Bayesian approach to variable selection in the presence of cost [Lindley, 1968]. Brown et al. [1999] built on Lindley's work, also in a Bayesian context. Perhaps notably, Brown et al. state "We omit variables not because we believe their coefficients to be zero, but because they cost too much relative to their predictive benefit." This quote perhaps demonstrates the mindset of the researcher interested in cost-considerate variable selection: practicality is favored to model exactness or correctness.

2.2.1 Different Types of Cost

Costs arise in many aspects of data collection and analysis. Turney [2000] and Weiss and Provost [2003] each provide a list of different types of costs that may be incurred in an analysis project. Turney breaks down all costs into fundamental pieces and includes easily forgotten costs including those associated with data collection, data labeling, computer expenses, and intervening based on conclusions.

Weiss et al. [2013] conclude their list by suggesting that substantial costs may require the researcher to limit the amount of training data. Limiting the amount of training data has at least two meanings which are not mutually exclusive. First, the number of sampling units can be limited. Second, the number of measurements (the columns of the dataset) can be limited. It may be possible to remove a single expensive variable from the collection procedure and reduce cost tremendously with little effect on the performance of the model. Of course, this is completely context dependent.

In Turney [2000], the cost of observations is referred to as the *cost of cases*. The variable measurement cost is referred to as the *cost of tests*, as in Turney's example: the cost of a blood test. It's clear how this language can become confusing, and though

there may be precedent for the word *test* applied to variable collection [Turney, 1995, 2000], we will refer to this cost as a cost of (measuring) a variable.

Though our interest is in reducing costs via intelligent variable selection, a quick review of classical strategies for choosing sample size is appropriate.

2.2.2 Sample Size Cost

Perhaps the first, and maybe only, place students in statistics courses incorporate cost into any sort of statistical analysis is in calculating the number of sampling units needed to obtain a prescribed power. This problem may be introduced in the context of Bernoulli trials under an assumption of normality for the sample proportion. With prior information, or assuming the worst case scenario of $p = 0.5$, students can calculate the minimum sample size necessary to obtain a specified confidence interval width for a chosen confidence level.

For more complex problems, there exist other sample size calculations and software relying on simulation to calculate power for different sample sizes given prescribed scenarios. The implication of the proliferation of proprietary sample size calculation software is that the cost of excess subjects in a study (or penalty cost of not having enough) is greater than the cost of the software. Like the preceding example, many of these calculations are based on obtaining a desired level of precision and cost efficiency is only implied. However, sampling allocation algorithms directly incorporating cost do exist, like Neyman allocation for stratified sampling. Lohr [2009] describes the objective as "to gain the most information for the least cost." Given estimates of the variance of the response in each of multiple strata, the population size, and the relative cost of measurement of a subject, Neyman allocation can be used to provide cost optimal sample sizes in each of the strata by balancing cost and variance.

2.2.3 Early Work in Cost-Based Variable Selection

Lindley [1968] investigated cost-considerate variable selection for a Bayesian multiple regression problem with known, fixed variance: $\text{var}(y|\theta, x) = \sigma^2$. Out of r measured variables, Lindley considered making a prediction of y for a new observation on the variables x_i for $i \in I$ where I is a subset of $1, 2, \ldots, r$. For a particular subset I costing c_I to measure, Lindley provided the loss function

$$(y - f(\mathbf{x}_I))^2 + c_I \tag{2.1}$$

where $\mathbf{x}_I$ is the matrix comprising x_i for $i \in I$, c_I is the cost of measuring those variables, and $f(\mathbf{x}_I)$ is the posterior expectation of y given the measured $\mathbf{x}_I$. Lindley suggested choosing the set of I variables minimizing the expected loss

$$E\left[(y - f(\mathbf{x}_I))^2 \,|\mathbf{x}_I\right] + c_I \ . \tag{2.2}$$

Building on the work of Lindley [1968], Brown et al. [1999] extended the results to the case of multivariate responses.

2.2.4 Machine Learning Community

In the data science and machine learning community, researchers have begun to develop some methods to deal with cost. The situation has improved since Turney [2000] noted that "the majority of machine learning literature ignores all types of cost (unless accuracy is interpreted as a type of cost measure)." His paper is still a valuable discussion of different types or interpretations of cost that can occur, especially in a machine learning framework as noted by Sheng et al. [2008] and Weiss et al. [2013], respectively, eight and 13 years after its publication. Turney explicitly deals with

classification problems, but his list is still pertinent, particularly his discussion of features with shared costs.

2.2.5 Grouped Features

Sometimes, measured variables share a common baseline cost [Turney, 1995, 2000]. If one variable is measured, a second variable may be available at a reduced cost compared to its acquisition without the preceding variable. Again, the medical field is used as an example: the cost of collecting blood is shared by any number of blood tests that might be performed. As such, cost-considerate variable selection algorithms ignoring this aspect may select variables from a variable-cost space that does not accurately portray true variable cost.

Paclík et al. [2002] note that though feature selection is typically used to effect the best possible performance, cost is often an important issue. In their particular application of image processing, they note that some features may be *grouped*, much like the number of tests that could be performed once blood is drawn from a patient. For example, though a set of features may be expensive, as soon as one is obtained, the rest are available for a reduced or even negligible cost. For their work, the cost they wish to reduce is the intensive computing time of processing images for features to be used in modeling. In their paper, they detail two algorithms used for variable selection in this context. They call their first algorithm Group-wise Forward Selection, where groups of variables enter the model together and a set of models is produced with each subsequent model having a higher cost. Their second method is called Group-wise Nested Forward Selection. Under this algorithm, each group of variables becomes available at a given step but not all variables in the group need to be added to the model. For example, if out-of-sample testing is used to evaluate performance,

there may be cases where better performance is obtained with only a subset of a group's variables in the model.

For an accurate portrayal of the variable-cost space, any new method for cost-considerate model selection ought to account for grouped features. We account for the possibility of grouped variables in the method we propose in this work. In addition to the cases described above, there is natural grouping when higher order variables are included in a model. Though there may be many cases where every measurement has an independent cost, any variables derived from those measurements, including interactions and squared terms, will be cost free once the constituent measurements have been obtained.

2.2.6 Dynamic Feature Selection with Cost

In the machine learning community, the feature extraction phase is often part of the same computing pipeline as model fitting. Some researchers have taken advantage of this in proposing machine learning algorithms that can perform *dynamic* feature selection while incorporating cost. For example, Weiss et al. [2013] propose the Cost-sensitive Attribute Selection algorithm using Histograms (CASH). The goal of CASH is to minimize the sum of two types of cost: misclassification costs and variable costs. Explicitly, their algorithm is devised for predicting categorical responses. He et al. [2012] start from a different point in an analysis project and assume that a model, or classifier, already exists. Their objective is to determine which features should be *purchased* in order to obtain a specified cost-accuracy balance in prediction.

Though these cost-considerate variable selection problems may be at home in many machine learning problems, they may not transfer well to more general problems. In many cases, dynamic feature selection may be impossible because it is not always feasible to measure more variables once the modeling stage has begun.

2.2.7 Researchers Are Making Decisions

If we investigate the ecological community (for one), we see that researchers are already making decisions on how to collect information for a study while considering cost. In a field that previously relied on extensive direct observation, new technology has allowed for new methods of data collection. Krause et al. [2013] reviews the state of geotracking animals fitted with GPS devices. Such miniature trackers open many new possibilities to researchers, however, this does not come without cost. Depending on the study, data may also be collected by a field technician paid to perform a focal follow. The performances and costs of these two methods of variable collection may vary, and it may not be trivial to decide which method is better for a particular objective.

Boyland et al. [2013] published an article discussing the efficacy of spatial proximity loggers. As they mention, social network analysis is becoming a popular approach in animal behavior and proximity loggers are a natural tool to consider when studying animal interactions. However, Boyland et al. [2013] explain that performance of these loggers is not consistent. As researchers are already considering cost to make decisions on how to collect data, a methodology to aid researchers in making more informed decisions may be welcomed.

2.3 Focus on Linear Models

Across many disciplines of science, performing multiple regression with linear models via ordinary least squares (OLS) has proven extremely useful. Transformations to response variables open the possibility of using linear models for many response variable types. OLS fitting can take advantage of efficient linear algebra routines resulting in computationally efficient fitting. Most researchers using

statistical models are comfortable and familiar with these tools. Further, a wealth of variable selection methods exist for linear models upon which one might build a custom routine. For these reasons, we focus on ordinary least squares linear models for the development of a new cost-considerate variable selection algorithm.

2.3.1 Common Objective Functions

The objective function for ordinary least squares is the sum of squared residuals. The set of coefficient values that minimizes the sum of squares is the OLS estimator:

$$\hat{\beta}_{ols} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 \right\} . \tag{2.3}$$

There are other options to the ordinary least squares approach. For one, least absolute deviations (LAD) [Branham Jr, 1982] minimizes the sum of the absolute residuals:

$$\hat{\beta}_{lad} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_i \left| y_i - \sum_j x_{ij}\beta_j \right| \right\} . \tag{2.4}$$

The LAD estimator might be used when outliers could be present since LAD is more robust to outliers than OLS, because the outliers are penalized less than in OLS. However, LAD is sometimes criticized for not necessarily offering a unique solution. Furthermore, the OLS estimator fits nicely with foundational statistics as it is also the maximum likelihood estimator under a linear model with Gaussian errors. Of course, any custom objective function can be used to fit a linear model, but due to the above considerations, we will approach the problem of cost-considerate variable selection from the perspective of ordinary least squares. Again, variable selection will be a trade-off between parsimony and performance, with performance measured via the sum of squares and parsimony driven by variable cost.

2.4 Related Work in Model Regularization and Selection

Even when focusing exclusively on linear models, researchers are spoiled for choices when it comes to variable selection, and this vast set of options may induce confusion. Are stepwise variable selection methods via alpha-level significance testing appropriate or should these tools only be used for testing explicit hypotheses? The answer likely depends on the statistician queried. Further, their answer to this question and actual practice may not be consistent. Another approach, the use of information criteria, provides a single numeric evaluation of a model's performance, allowing the researcher to select a model as the best performing according to that criterion. For example, AIC [Akaike, 1974, Burnham and Anderson, 2002] is used ritualistically by some while others deride such ritualistic use. The Lasso attacks the problem by shrinking linear model coefficient estimates in an attempt to reduce overfitting and out-of-sample prediction error [Tibshirani, 1996]. Shrinkage via the Lasso can reduce the coefficient estimates to exactly zero; thus regularization via the Lasso includes variable selection whereby variables are selected if their coefficients are not set to zero. LARS is another shrinkage and selection strategy which is flexible enough to include Lasso estimates as a special case [Efron et al., 2004].

In evaluating the landscape of variable selection methods, it is clear that there is no single best variable selection routine. Rather than making an attempt to cover the entire body of variable selection literature, we focus explicitly on Lasso [Tibshirani, 1996] and LARS [Efron et al., 2004], the two algorithms inspiring the cost-considerate modeling algorithm we present in this work. We present the pertinent history that has led to these algorithms and some relevant adaptations currently in use.

2.4.1 Ridge Regression

Ridge regression is an important precursor to the Lasso and a classic example of penalized regression, a popular method of model regularization [Ng, 2004]. Rather than estimating model parameters by simply minimizing the sum of squared residuals, in penalized regression the objective function contains both the sum of squares and a penalty term. In ridge regression, the model parameter estimates are penalized for being large and therefore are shrunk toward zero. The coefficient estimates for ridge regression can be found as the values minimizing the sum of squared residuals subject to a constraint on the sum of the squared coefficients:

$$\hat{\beta}_{ridge} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 \right\} \text{ subject to } \sum_j \beta_j^2 \leq t \ . \qquad (2.5)$$

This regularization of the coefficient estimates can improve out-of-sample prediction. For example, n points on a scatter plot can be perfectly predicted with a polynomial of order $n-1$. However, using that polynomial to interpolate between the n points may produce estimates that do not agree with realistic expectation. By minimizing the sum of squares subject to the constraint on the model coefficients, an $n-1$ order polynomial can be produced with improved out-of-sample prediction though losing perfect in-sample prediction.

2.4.2 The Nonnegative *Garrote*

Leo Breiman's nonnegative *garrote* [Breiman, 1995] directly inspired the Lasso. The word *garrote* is Spanish term referring to a method of execution where the condemned is strangled with an iron collar. The image is less graphic when applied to linear model coefficients. Though he applauds variable selection methods for producing simpler models, Breiman notes that variable selection routines lack the out-

of-sample prediction accuracy of shrinkage methods like ridge regression [Breiman, 1995, Hoerl et al., 1986]. Breiman's goal is to produce a method that can perform model regularization via coefficient shrinkage where the shrinkage can be strong enough to take coefficiencts to zero, thus performing variable selection as well. This method sits between other variable selection routines and ridge regression, being both accurate for out-of-sample prediction and producing parsimonious models.

The *garrote* performs both regularization and variable selection on a model of the form

$$y_i = \sum_{j=1}^{J} x_{ij}\beta_j + \epsilon_i \ , \quad \epsilon_i \sim N(0, \sigma^2) \ . \tag{2.6}$$

The first step is to obtain the OLS estimator of β, $\hat{\beta}$. Next, Breiman finds the c_j, $j = 1, \ldots, J$, constrained by the equations $c_j \geq 0$ and $\sum_{j=1}^{J} c_j \leq s$ minimizing

$$\sum_i \left(y_i - \sum_j x_{ij} c_j \hat{\beta}_j \right)^2 \ . \tag{2.7}$$

The resulting *garrote* estimates (as a function of s) are $\tilde{\beta}_j(s) = c_j(s)\hat{\beta}_j$. As s is decreased, the c_j are constricted to smaller values. Eventually, some c_j go exactly to zero thus excluding variable j for the model, and other $\tilde{\beta}_j(s)$ are shrunk toward zero, though not all the way.

This algorithm was the main inspiration for the Lasso [Tibshirani, 2011]. In Lasso however, a different objective function is posed, and the algorithm can be implemented without first fitting OLS and then performing shrinkage like the *garrote*.

2.4.3 The Lasso

With over 17,000 citations (Google Scholar September 2016), Robert Tibshirani's 1996 paper, *Regression shrinkage and selection via the Lasso* [Tibshirani, 1996] is

one of the seminal publications on model regularization. Lasso is short for "least absolute shrinkage and selection operator" which harkens to its ability to both shrink coefficients as well as perform variable selection.

The principle behind the Lasso is quite simple, and can be summed up with one sentence from the paper's abstract: "The 'lasso' minimizes the residual sum of squares subject to the sum of the absolute value of the coefficients being less than a constant." The mathematical stipulation is

$$\hat{\beta}_{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 \right\} \text{ subject to } \sum_j |\beta_j| \leq t \ , \tag{2.8}$$

which can be equivalently written as

$$\hat{\beta}_{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 + \lambda \sum_j |\beta_j| \right\} \tag{2.9}$$

where λ is related to t. There is a clear connection to ridge regression (Equation 2.5) with the difference being that, for ridge regression, the coefficients are squared in the penalty term. The difference between ridge regression and Lasso is summarized in Figure 2.1 for two dimensions [Tibshirani, 1996]. For both, we can think of contours of equal sum of squares on a space of coefficient values and a region centered at the origin representing the set of coefficients values satisfying the constraint. The model's coefficient estimate is the intersection of the constraint satisfying region with the intersecting contour of lowest sum of squares. Even in just two dimensions, we see a stronger preference toward coefficients being exactly equal to zero for Lasso than for ridge regression. This preference is further exacerbated for greater dimensions.

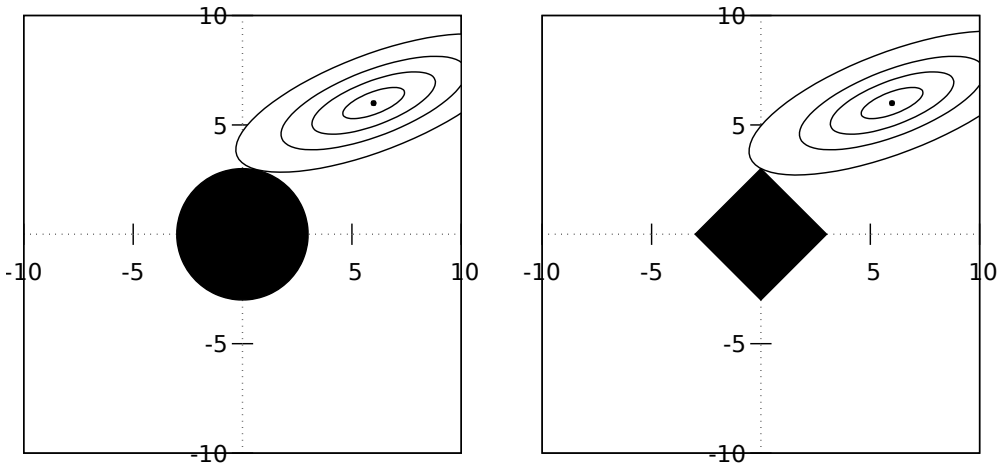

Figure 2.1: Comparison of Lasso, left, and ridge regression, right, for two variables. The two axes are for the model parameter estimates and the dark geometric shapes cover the region satisfying the two-dimensional constraints $|\beta_1| + |\beta_2| \leq t_{lasso}$ and ${\beta_1}^2 + {\beta_2}^2 \leq t_{ridge}$ for Lasso and ridge regression, respectively. The contours represent (β_1, β_2) coordinates with identical sum of squares. Since the Lasso constraint-satisfying region protrudes further along an axis than off axis, the intersection with the minimizing sum of squares contour is more likely to be on an axis, thus explicitly setting the coefficient to zero, than for ridge regression. This figure is based on Figure 2 in Tibshirani [1996].

2.4.4 Adaptive Lasso

The adaptive Lasso was proposed by Zou in 2006 [Zou, 2006]. The definition of the adaptive Lasso estimator closely resembles the Lasso estimator (Equation 2.9) with a small addition,

$$\hat{\beta}_{alasso} = \underset{\beta}{\operatorname{argmin}} \sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 + \lambda \sum_j w_j \left|\beta_j\right| \ , \tag{2.10}$$

which looks very similar to Breiman's nonnegative *garrote*. In fact, Zou shows that the Adaptive Lasso is equivalent to the nonnegative *garrote* subject to the constraint that the adaptive Lasso estimates share the same sign as the OLS estimates or are zero.

The benefit of the adaptive Lasso is that different weights, w_j, can be applied to different parameters, thus more aggressively shrinking some estimates toward zero than others. In Zou [2006], the weights are proposed to be of the form $w_j = 1/|\hat{\beta}_j|^\gamma$ for $\gamma > 0$. With such weights, the adaptive Lasso asymptotically selects the true coefficients and the adaptive Lasso estimators are asymptotically normal and centered on the true values. Sometimes, the individual Lasso parameters $\lambda_j = w_j\lambda$ are reported. An adaptive take on the Lasso could allow greater penalties for variables that are more expensive.

2.4.5 Bayesian Lasso

In the original Lasso paper, Tibshirani [1996] noted a connection between the Lasso penalty term and a class of priors under a Bayesian framework. Minimizing the Lasso objective function is equivalent to finding the posterior mode of β where β has a double exponential (or *Laplace*) prior. Though Tibshirani spent just two short paragraphs on this connection, many researchers have since published extensions

to the Lasso available under the Bayesian framework. Despite Tibshirani's clear interpretation of Lasso estimators in a Bayesian framework in 1996, it wasn't until 2008 that Trevor Park and George Casella published *The Bayesian Lasso* [Park and Casella, 2008]. However, there are some earlier publications that deserve mention.

Figueiredo [2003] published a Bayesian based method for supervised learning with a goal of producing a model to perform classification using a *sparse* subset of potential variables. Figueiredo used the Laplacian prior and noted its connection to the Lasso. In 2005, Yuan and Lin published their own hierarchical Bayesian variable selection algorithm and noted its relation to the Lasso [Yuan and Lin, 2005]. In 2004, Bae and Mallick [Bae and Mallick, 2004] published their solutions to gene selection problems (there are a large number of genes compared to number of study units). One of their strategies employed Laplacian priors on coefficients, which they noted was equivalent to the Lasso model. However, Bae and Mallick went a step further and may have actually published the original example of the Bayesian *Adaptive* Lasso (later published by [Leng et al., 2013]), by allowing the flexibility of different penalties for different coefficients.

The two main contributions from Park and Casella [2008] were to delineate an efficient MCMC sampling algorithm from the posterior and to discuss (and suggest) ways to deal with the Lasso parameter, λ. Though practitioners using the classic Lasso tend to rely on cross-validation to select the Lasso parameter, Park and Casella introduced a few methods consistent with a Bayesian perspective. One strategy they posed is to estimate λ via expectation-maximization. Another strategy is to directly incorporate it into the posterior. They suggested the gamma family of priors with the form

$$\lambda^2 \sim \frac{\delta^r}{\Gamma(r)}(\lambda^2)^{r-1}\exp\left(-\delta\lambda^2\right)$$

for $\lambda^2 > 0$ and $r, \delta > 0$.

2.4.6 Bayesian Adaptive Lasso

Building on the Bayesian Lasso and the adaptive Lasso, Leng et al. [2013] introduced the Bayesian adaptive Lasso (BaLasso). In their paper, they describe their "hybrid Bayesian-frequentist" point of view for performing variable selection and suggest the BaLasso produces improvements over both the original Lasso and the adaptive Lasso. Their prescription is to allow different Lasso penalties for each coefficient like the adaptive Lasso. Further, while the Bayesian Lasso only shrinks coefficients, Leng et al. suggest methods to explicitly set coefficients to zero depending on posterior means or medians.

2.4.7 Group Lasso

There may be cases when variables are intrinsically part of a group and a researcher would like all variables in the group to be *simultaneously* included or excluded from the model. With the Lasso, no such connection among variables is taken into account. An adaptation to the Lasso to account for grouping seems to be first suggested in Bakin's 1999 thesis [Bakin, 1999] and was further developed by Yuan and Lin [Yuan and Lin, 2006]. In cost-considerate variable selection, grouping may be relevant in the case that multiple variables are available for a single cost as discussed earlier.

2.4.8 The LARS Algorithm

The Lasso has been extremely successful, but actually fitting the Lasso may require extensive computation. The Lasso is fit by simply minimizing the objective function in Equation (2.9). However, in very high dimensional problems, for which the

Lasso is especially pertinent, it may be slow to solve if a naive optimization routine is used.

Instead, the LARS (Least Angle Regression) algorithm is capable, with small tweaks, of producing Lasso estimates on computational order with ordinary least squares [Efron et al., 2004]. The original motivation for LARS was forward stagewise regression.

In forward stagewise regression, the response and predictor variables are all typically centered and scaled to unit standard deviation. The regression equation is then built in steps. The *current* model sits at the origin and the predictor variable most correlated with the residuals is added to the model with a small coefficient, which we can think of as a small step toward the OLS model for the included predictors. The size of the step is somewhat arbitrary, but should not be so large as to reach the OLS solution. At the new *current* model, the residuals are calculated and the new (which could be the previous) predictor most correlated with the residuals is found and the model prediction is incremented along the direction of the most correlated predictor variable. This procedure is repeated producing a set of models, one for each step.

The LARS algorithm was devised to take larger steps than forward stagewise regression thus improving computation efficiency. With LARS, the variables are added not in small steps, but in different, larger step sizes. The size of the step is such that after incrementing the model, all the included variables will be equally correlated with the residuals as the most correlated variable not currently included in the model. By this process, the LARS algorithm produces a set of models where each subsequent model introduces a new variable. In the sequence of producing this set of models, the early models are heavily regularized and contain few of the variables. The final model in the set (if the number of variables is fewer than the number of

observations) is equivalent to the OLS model. Small tweaks to the algorithm can be employed so that Lasso or forward stagewise models can be produced.

2.5 Cost-Efficient LARS

In this dissertation, we attack the problem of cost-considerate variable selection with an adaptation of Lasso. Before our work, Yue [2010] also considered a Lasso adaptation for cost-considerate variable selection. Yue's strategy is to fit a linear model by minimizing the squared error subject to an additional constraint beyond Lasso:

$$\underset{\alpha,\beta}{\operatorname{argmin}} \left\{ \sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 + \lambda \sum_j |\beta_j| + n\omega \sum_j \alpha_j c_j \right\} , \qquad (2.11)$$

where c_j is the cost associated with the j^{th} variable. The coefficient α_j is either 0 or 1 such that if $\alpha_j = 0$, then β_j is defined to be 0, and variable j is not selected for the model. The quantity n is the sample size; a larger sample size implies greater cost to measure each variable on n subjects. The quantity ω is a user-set value setting the relative reluctance toward higher cost variables in deference to improved model fit. The third term where cost enters is kept separate from the Lasso parameter where the magnitude of the coefficients enter. By keeping cost and coefficient size separated, coefficients may be selected based on cost, but shrinkage does not depend on cost. For a fixed α vector, finding the minimizing β is equivalent to solving for the Lasso since the third term is a constant in β. Thus, for each of the 2^J possible α vectors, the Lasso solutions can be found and the objective function of Equation 2.11 can be evaluated. The α vector minimizing the objective function defines the variables to be selected whereby the variables selected are those j for which $\alpha_j = 1$. Of course 2^J

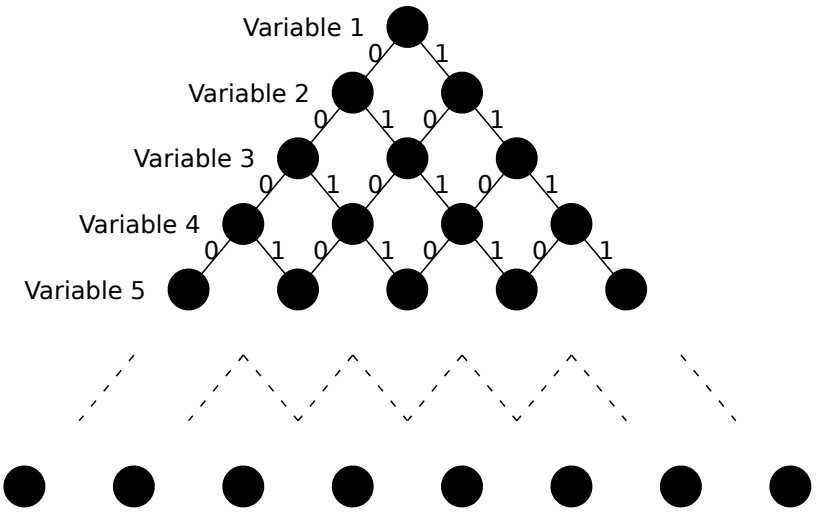

Figure 2.2: The Branching LARS algorithm considers the space of available variables in a tree structure. The root node indicates no variables available. The two children of the root node disallow and allow the first variable. This branching is followed to the leaf nodes which specify the availability for all of the variables. With branching and bounding, only a subset of the models may need to be explored. This figure is based on Figure 2.1 in Yue [2010].

increases quickly with $J = 20$ already giving over 1,000,000 potential α vectors, each requiring a Lasso fit.

Yue's solution to this exponentially increasing computation problem is to employ branching and bounding. The 2^J α vectors can be visualized as the leaves of a tree (Figure 2.2). At any non-terminal node of depth k, we can consider the case that $\alpha_l = 1$ for $l = k+1, \ldots, J$. Using $\alpha = \{\alpha_1, \ldots, \alpha_k, 1, \ldots, 1\}$, the Lasso estimates $\hat{\beta}^{lasso}$ can be found. Such an estimate represents the smallest the first two terms of the objective function in Equation 2.11 can be for any node below our node of depth k. For all nodes below the current node, the smallest that the third term can be is when $\alpha_l = 0$ for $l > k$. Thus, we can create a bound. The objective function

below node k can be no less than the Lasso solution at node k. If this bound does not beat any previously obtained objective function evaluation, then proceeding any further down that particular branch is futile and it may be pruned from the set of potential solutions. Yue calls this procedure BLARS for Branching LARS since the LARS algorithm is used repeatedly to obtain Lasso fits throughout the branching and bounding process.

Pruning in this way can reduce the problem space dramatically. In order to make this procedure efficient, the variables must be ordered in an intelligent manner. Some intuitive methods include ordering the variables based on cost, order of entry in LARS, or size of corresponding OLS estimate (all either ascending or descending). Yue suggests that the optimal ordering method will depend on the λ and ω and provides some rules of thumb for selecting a variable order. Still, this procedure requires a large number of calculations. In Yue's example with 10 variables, the number of function calls to the LARS algorithm is highly dependent on Yue's ω parameter, λ, and the ordering method. Despite there only existing $2^{10} = 1024$ potential models, in some cases the number of LARS calls is still above 900. In a problem with more variables, we are likely to see even more calls to LARS and many of those calls to LARS may be more complex, due to more variables, than in a case with fewer variables. Our goal is to develop a method that performs variable selection based on cost and model regularization on the same order as a single LARS algorithm.

2.6 Objective

Understanding the history, shortcomings, and existing methods of variable selection, we propose a new cost-considerate variable selection algorithm for linear models. The overarching goal is to provide an algorithm that performs variable selection based on variable cost for problems when pilot or historic data exist. Within

that overarching goal, the algorithm should be able to handle large datasets with many variables. This means our algorithm should be computationally efficient and will include model regularization in addition to variable selection.

We propose to solve this problem by basing our methods on Lasso and LARS, with additional inspiration from some of the many existing tweaks of those methods. First, we consider the objective function of the Adaptive Lasso,

$$\sum_i \left(y_i - \sum_j x_{ij}\beta_j \right)^2 + \lambda \sum w_j \left|\beta_j\right| . \tag{2.12}$$

Under the adaptive Lasso, the coefficient estimates are regularized by both the Lasso parameter, λ, and by the coefficient specific weight w_j. Rather than use the approach proposed in that original adaptive Lasso paper to set w_j [Zou, 2006], suppose we use the cost of collecting variable j to set w_j. Forgiving the lack of a function to transform cost to an appropriate Lasso weight, we can acknowledge that variable selection via this cost adaptation now depends upon the cost of the variable in addition to that variable's performance in the model. However, the effect of cost is not confined to variable selection; coefficient regularization is now also affected by the variable cost. This seems to be an unwanted effect. Our goal is to use cost for selection and not for shrinkage.

If the variable is to be included in the predictive model, the researcher is on the hook for the cost of that variable. There seems no financial or predictive benefit to shrinking the coefficient of the variable based on its cost. And thus, our goal is to provide a mechanism incorporating cost in variable selection but not regularization. Yue [2010] did this by separating the cost and coefficient magnitude to additive terms. This prevents cost from influencing the shrinkage, but for problems with many variables, this creates a very large set of models that must be explored. With

luck, the branching and bounding method can reduce the number of models needing to be evaluated. However, our goal is to produce an algorithm that can fit the cost-considerate model in comparable computation time to a single LARS routine which, in turn, is in comparable computation time to a single OLS fit.

In addition, the algorithm should be able to handle other complicating circumstances likely to occur in an analysis project. In particular, we would like an algorithm that can handle the case of variables that share cost. The cost of measuring a variable is not always independent of measuring other variables. Several variables may share the same baseline cost, for example multiple blood tests may all rely on the cost of drawing blood from a subject. Further, interactions and other transformations may be *free* variables once the constituent variables are measured.

As we alluded, we will rely on LARS and Lasso to provide the backend of our cost-considerate variable selection with regularization. A number of existing adaptations of LARS and Lasso already exist, some directly addressing cost-considerate variable selection with regularization (blars from Yue [2010]) and others initially appearing promising in this domain (adaptive Lasso from Zou [2006]). However, as we discussed, these adaptations do not meet our goals and expectations for such an algorithm, so we have developed our own relying heavily on the Lasso adjustment to the LARS algorithm. Our algorithm is called clars with the "c" indicating cost and also as a play on etymology with the sequence of adaptations of LARS to blars to clars.

2.7 LARS Algorithm Inspires clars

In this section, we overview the steps of the LARS algorithm which forms the basis of our cost-considerate variable selection with regularization algorithm. Purposefully, we eschew the mathematical details to focus on the bigger picture and

the intuition behind LARS to inspire the clars algorithm from the general principles of LARS. The details can be found in Efron et al. [2004] and Hastie et al. [2015] and may be gleaned by the details and calculations that we present for our algorithm. This allows us to build intuition about the clars algorithm before providing the rigorous mathematical calculations which, though necessary, may distract from the motivation. We roughly follow the steps as outlined in Hastie et al. [2015] in listing the steps for LARS.

The result of running the LARS algorithm is a sequence of models and model fits. These models typically run the gamut from an overall mean model ($\hat{y} = \overline{y}$) to the OLS solution. A few preparatory steps are performed and then the steps of the main fitting procedure are repeated until all p or n variables (whichever is smaller) are included in the model. Though the algorithm can accept $p > n$ variables, no more than n will be used in a single model.

The first step in LARS is to simply standardize the predictor variables to each have mean 0 and standard deviation 1. We also center the response variable so that it has mean 0 and introduce the current predictions $\hat{\mu}_0$, to be a vector of 0's equal in length to the number of observations. Essentially $\hat{\mu}_k$ is the vector of predictions for the current (iteration k) model on the observations in the data set. Often, $\hat{y}$ is used as notation for this same quantity, but we use $\hat{\mu}_k$ to be consistent with Efron et al. [2004]. For the first iteration, the coefficients are set to 0. For simplicity, we denote the centered response variable y and the p predictor variables after centering and scaling x_j where $j = 1 \dots p$. The current residuals vector is the difference between the centered response variable and current predictions, $r_k = y - \hat{\mu}_k$.

The second step is to find the predictor variable most correlated with the current residuals. This can be found simply by taking the dot product of the current residuals with each x_j, for $j = 1 \dots p$, and finding the largest absolute value. Suppose the most

correlated variable is the one indexed by l. We now define the *active set*, $\mathcal{A}$, to be $\{l\}$, the single predictor variable most correlated with the residuals. As we build a sequence of models, the active set will expand to include the predictor variables used to form the current predictions (all other variables will be set to have $\hat{\beta}_j = 0$ for $j \notin \mathcal{A}$).

Now, we proceed to steps three through six which are repeated. In step three, we find the *equiangular* or *least squares* direction, u. This direction is defined as the unit vector having equal projection with all predictor variables currently in the active set after those predictors have been multiplied by ± 1 so they all have positive correlation with the current residuals. We call this the equiangular direction because the angle between this unit vector and each of the predictor variables in the active set is the same. We might also call this the least squares direction because traveling far enough in this direction from the current predictions, $\hat{\mu}_k$, will produce the estimate of the OLS fit for the variables in the active set. For the first iteration, where we only have a single predictor variable, u is equal to that predictor variable (or its negation if it is negatively correlated with the residuals).

In step four, we consider the function $\hat{\mu}'(\gamma') = \hat{\mu}_k + \gamma' u$ where γ' represents a length to increment the model. Meanwhile, we keep track of the updated residuals $r'(\gamma') = y - \hat{\mu}'$.

In step five, we begin by calculating the correlation of these evolving residuals $r'(\gamma')$ with the predictor variables. Later, we show that the correlation between $r'(\gamma')$ and each predictor variable in the active set is the same for any value of γ'. For variables not in the active set, the correlation changes at a rate dependent upon the relationship between that variable and u. However, we notice that as γ' is increased, eventually each predictor variable $l \notin \mathcal{A}$ will become as correlated with the residuals as the variables in the active set. In Figure 2.3, the null model $\hat{\mu}_0$ is increased along

x_1 to $\hat{\mu}_1$ until x_2 and x_1 are equally correlated with the resultant residuals. This leaves us with a set of values for γ', where each is a value where a new variable is equally correlated to the residuals as variables in the active set. From these values, the smallest is selected, and we denote it γ.

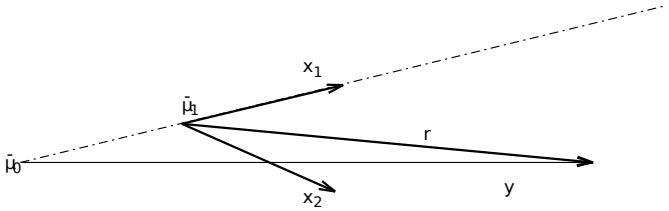

Figure 2.3: In the LARS algorithm, the current model ($\hat{\mu}_0$) is incremented in the least squares direction (the direction of x_1) until the next variable (x_2) becomes equally correlated with the residuals. This point becomes the next model ($\hat{\mu}_1$) and the model is now incremented in the new least squares direction. This figure based on Figure 2 in Efron et al. [2004]

In step six, we take the γ found in step five and update the model. This concludes the iteration as the model is updated accordingly as well as the coefficients. The new variable is added to the active set.

Steps three through six are repeated until we have exhausted the supply of predictor variables or observations (either all p variables or the first $n-1$ have entered the model). In the end, we have a sequence of models with each subsequent model adding a new variable. We may then use some method to choose among these models, thus choosing a set of variables to be used in the final model. Typically, prediction error on a hold-out set is used to choose among the models.

With this exposition of the LARS algorithm, we now discuss how some modifications may improve the performance when variable costs must be considered. First, we recognize immediately that the LARS algorithm builds a set of models where each subsequent model contains an additional variable. Thinking in cost, this means each subsequent model incurs an additional cost assuming all variables have some incurred cost. So, the LARS algorithm itself can be used to create a set of models with varying costs and one may be selected by the researcher using some method accounting for both cost and perhaps prediction error.

Now, let's consider the Lasso adjustment to LARS and its relation to cost. The LARS algorithm can be adjusted to produce Lasso fits. Between steps four and five, it is possible for a coefficient of a variable in the active set to change sign as γ is increased, and if such a crossing does not occur, the LARS steps are followed as usual. Such a transition is mathematically prohibited under Lasso, so when this happens under LARS, we must adjust the algorithm to produce Lasso fits. To do this, from step four to five we stop γ "early" at exactly the point where the coefficient is zero and before a new variable becomes equally correlated to the residuals. Choosing the shorter γ replaces step five and we proceed to step six. Under the Lasso step, no new variable is added to the active set and instead, the variable crossing zero is removed from the active set. At this point, we return to step three with one less variable in the active set. From a model cost perspective, we can see an advantage to this Lasso adjustment to LARS. Removing a variable from the active set will reduce the cost of the model by requiring one less variable to be measured. Therefore, we retain the Lasso adjustment to LARS for our clars algorithm as it is evident that this adjustment may lead to cheaper models.

Finally, consider the step of the LARS algorithm where costs are accrued: in step two the first (potentially very expensive) variable enters, and in step five γ is

chosen defining the next variable to enter the active set. By adjusting these two steps, we may be able to produce more cost efficient models.

We begin with step two which is a more intuitive step, and we generalize the strategy to step five. For extreme cost reduction, the cheapest variable may be the first introduced to the model. After all, this creates the cheapest model possible given the predictors: a model with only the cheapest variable. However, this strategy clearly suffers a lack of consideration of performance. It may be possible that the cheapest variable was measured primarily because it was cheap to measure and may have little scientific motivation; it may have almost no predictive power on the response.

Instead, we ought to consider a balance between performance and cost. In Yue's blars method, a balance is obtained by adding a cost term to the objective function for Lasso (Equation 2.11). This basic premise makes sense, but in the end is difficult to interpret. The parameter ω in Equation 2.11 sets the scaling of the penalty on cost. However, appropriate values for ω depend on the particular problem and our only intuition is that higher values for ω result in more reluctance to use expensive variables. The confusion over selecting ω only increases when we recognize that by merely changing the units in which y or cost are measured produces different model fits for the same ω. We can get around this by standardizing y and selecting a cost unit a priori, but we still lack any intuition about using ω for different datasets. In the end, the user must essentially choose a model across two somewhat arbitrary parameters, ω and λ. This two dimensional model space may be hard to appropriately cover. Though there are certainly many benefits to a highly customizable algorithm, we prefer an algorithm not introducing extra, difficult to interpret parameters that must be set by the user.

Instead, consider selecting the first variable as the variable most highly correlated with the response *per cost*. Correlation is scale independent and since we divide by

cost (as opposed to adding a cost term), the first variable that is selected is the same for any scale transformation of cost. Moreover, this strategy feels very intuitive and does not introduce any difficult to interpret parameters that must be set by the researcher.

With step two accounted for, we now consider how to deal with step five. First, we revisit the strategy used in LARS: From the current predictions, γ' is slowly increased until a new variable is equally correlated with the resultant residuals as the variables in the active set. Suppose we were manually performing these LARS steps and were slowly increasing γ' only to find that the first variable to become equally correlated with the residuals as the variables of the active set happens to be a very expensive variable despite the existence of several other cheaper variables not in the active set. Doing the steps manually, if we cared about cost, we might simply ignore this expensive variable and continue to increase γ' until a different, cheaper variable becomes equally correlated with the residuals as the variables in the active set. The fit would match the scenario that the more expensive variable had not been measured. In order to make this algorithmic instead of manual, we need to create a decision criteria for when a variable should be skipped. We already argued for a correlation-per-cost based decision for step two and now we adapt that for step five.

For each of the variables not in the active set, we can determine the value for γ necessary for that variable to enter the active set if we ignore all the other variables not in the active set (in other words, excluding any variable that would enter the active set before this particular variable). Thus, for each variable's γ, we can find the correlation between that variable (and the variables in the active set) and the residuals. Now, we can calculate a score of correlation per cost and produce a table similar to Table 2.1. However, there are two different values for cost to consider. One

is the cost of the specific variable to enter, the second is the cost of all variables in the active set as well as the specific variable together.

Table 2.1: For each variable not in the active set, the corresponding value for γ is calculated. Following, the resultant active set correlation with the resultant residuals is calculated. This correlation divided by the cost to measure the variable in the active set should that variable be selected is the clars score.

γ_j	resultant correlation	cost of model new variable	clars score
γ_1	c_1	d_1	c_1/d_1
γ_2	c_2	d_2	c_2/d_2
$\vdots$	$\vdots$	$\vdots$	$\vdots$

We recommend the second strategy where all variables' costs are considered together. One argument for this total cost strategy is that the total cost is inherently more important than the cost of any individual variable. For example, if the variables in the current active set cost 500 dollars together and two competing variables cost 1 and 2 dollars, respectively, many will prefer the more expensive variable if it improves fit slightly over the cheaper variable rather than requiring the fit to be, in some sense, twice as good. Since we are already committed to the costs of the variables in the active set, this active set cost is a reasonable baseline for comparing the relative cost of two competing variables.

Another reason for choosing the complete cost strategy over the individual variable cost strategy is that it leads directly to a non-additive approach to variable cost. In real applications, variable measurement may share some baseline costs and thus non-additive costs are necessary to correctly express these costs. We develop this idea further in Section 2.10.

In the next section, we formally define the steps of the clars algorithm. We provide the mathematical underpinnings and note special cases that may be introduced with our adjustment and how to handle such cases. As we will see, these

adjustments to LARS introduce scenarios not present in LARS for which we must account.

2.8 The clars Algorithm

In the previous section, we listed steps one through six of LARS, where steps three through six are repeated in each interation. In this section we proceed through the same steps but describe the adaptations for clars, with additional mathematical detail and more formal notation.

2.8.1 Step One: Center and Standardize

As before, we center and scale the predictor variables to have mean 0 and standard deviation 1. We use x_j to denote a vector of the n observations on the j^{th} predictor variable after standardization. We use X to denote a matrix whose columns are x_j for $j = 1 \dots p$. The response variable is centered on its mean. The centered response variable is denoted y, another vector of length n.

The phrase *current* is used to identify predictions, residuals, and coefficients in the current iteration of the algorithm. The vector $\hat{\mu}_k$ contains the current predictions of y as of the k^{th} iteration of the procedure and can be calculated with $X\hat{\beta}_k$ where $\hat{\beta}_k$ is the vector of the current coefficients. To start, we use the null model $\hat{\mu}_0 = \underline{0}$ and $\hat{\beta}_0 = \underline{0}$, 0 vectors of length n and p respectively, since y is centered at 0. The vector of the current residuals is denoted r_k which again is subscripted to denote iteration number and is calculated $r_k = y - \hat{\mu}_k$.

2.8.2 Step Two: Highest Correlation Per Unit Cost

In step two, we find the predictor variable most correlated with the current residuals per unit cost, where we have assumed that all variables have a non-zero

cost. Since the response variable y was centered, $r_0 = y$, and the predictor variables were centered, the correlation between x_j and r_0 is proportional to their dot product. The proportionality constant is n times the standard deviation of r_0 (recall that x_j has standard deviation of 1). Thus, a vector of all the correlations is proportional to the vector $X' \cdot r_0$. From this simple calculation, we can find the predictor with the largest absolute value of correlation with the current residuals per cost by dividing the correlation vector by the cost of the respective variables. Suppose this is predictor l; we now initiate the active set $\mathcal{A}$ to be $\{l\}$.

2.8.3 Step Three: Find the Least Squares Direction

Steps three through six are repeated in each iteration of the algorithm. In step three, we find u, the least squares direction. When the active set contains a single predictor variable, the least squares direction is simply $\pm x_l$ if l is the variable contained in the active set. When the active set contains multiple predictors, we must calculate u.

In Hastie et al. [2015], u is called the least-squares direction to provide intuition to its relevance to fitting the model. Fitting with clars or LARS, the model is advanced *toward* the least squares solution, but is stopped early to introduce an additional variable. However, calculating u is more easily understood when noting that u is the equiangular direction, the unit vector that has equal projection with predictor variables in the active set, as it is described in Efron et al. [2004]. We now introduce $X_{\mathcal{A}}$ to be a matrix whose columns are $sign_l \cdot x_l$ for $l \in \mathcal{A}$ where $sign_l$ is the sign of the correlation between x_l and the residuals. Again, for the first iteration, $X_{\mathcal{A}}$ is a single column.

We use a few tricks to find the solution to u. Since u has equal projection with all predictor variables in the active set, we note

$$X'_{\mathcal{A}}u = a\underline{1} \tag{2.13}$$

where $\underline{1}$ is a vector of 1's with length equal to the number of variables in the active set and a is a constant. We would like to solve for u and can assume that a is unknown. Now, we left multiply both sides of Equation 2.13 by $(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}$ to give

$$(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}X'_{\mathcal{A}}u = (X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}a\underline{1}\ . \tag{2.14}$$

Next, we left multiply Equation 2.14 by the transpose of Equation 2.13 to obtain

$$u'X_{\mathcal{A}}(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}X'_{\mathcal{A}}u = a\underline{1}'(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}a\underline{1}\ . \tag{2.15}$$

Now, we recognize that $X_{\mathcal{A}}(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}X'_{\mathcal{A}} = H$ is the hat matrix for the variables in the active set, and we define u to be in the space spanned by the columns of $X_{\mathcal{A}}$ so $Hu = u$. Finally, using the fact that u is a unit vector, we are able to solve for a:

$$u'Hu = a\underline{1}'(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}a\underline{1} \tag{2.16}$$

$$u'u = a\underline{1}'(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}a\underline{1} \tag{2.17}$$

$$1 = a^2\underline{1}'(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}\underline{1} \tag{2.18}$$

$$a^2 = (\underline{1}'(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}\underline{1})^{-1} \tag{2.19}$$

$$a = (\underline{1}'(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}\underline{1})^{-1/2}\ . \tag{2.20}$$

where the a is taken as the positive solution.

Having solved for a, we can solve for u starting with $u = Hu$:

$$u = Hu \tag{2.21}$$

$$u = X_{\mathcal{A}}(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}X'_{\mathcal{A}}u \tag{2.22}$$

$$u = X_{\mathcal{A}}(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}a\underline{1} \tag{2.23}$$

$$u = aX_{\mathcal{A}}(X'_{\mathcal{A}}X_{\mathcal{A}})^{-1}\underline{1}\,. \tag{2.24}$$

2.8.4 Step Four: List Potential Model Updates

In step four, we increment the model along the direction of u,

$$\hat{\mu}'(\gamma') = \hat{\mu}_k + \gamma' * u\,. \tag{2.25}$$

In the step five, we choose a specific value for γ and we will increment $\hat{\mu}_k$ to $\hat{\mu}_{k+1}$, but momentarily, we consider $\hat{\mu}'$ to be a function of γ'. Our goal for step four will be to produce a set of values of γ' from which we will select γ in step five. As a function of γ', we can calculate the projection of a predictor variable x_j on the resultant residuals to obtain

$$x_j \cdot r(\gamma') = x_j \cdot (y - \hat{\mu}'(\gamma')) \tag{2.26}$$

$$= x_j \cdot (y - \hat{\mu}_k - \gamma' * u) \tag{2.27}$$

$$= x_j \cdot (y - \hat{\mu}_k) - \gamma' * x_j \cdot u \tag{2.28}$$

$$= x_j \cdot r_k - \gamma' * x_j \cdot u\,. \tag{2.29}$$

In Equation 2.29, the first term is the projection on the current residuals. If the variables in the active set shared the same correlation with the residuals at the

beginning of the iteration (which we will show is the case), then we see the first term $x_j \cdot r_k$ is a constant across $j \in \mathcal{A}$, and we call it C_k. Further, since u has equal projection with all variables in the active set, the second term is also a constant across $j \in \mathcal{A}$ and we call it $\gamma' * a$, which is the same a as in step three. Thus, for $j \in \mathcal{A}$, Equation 2.29 becomes

$$x_j \cdot r(\gamma') = C - \gamma' a \,. \tag{2.30}$$

Therefore, incrementing our model in the equiangular direction u results in a model such that all variables in the active set have the same correlation with the residuals regardless of how far we increment the model in the least squares direction.

For variables not in the active set, we can find the value of γ' resulting in residuals where the projection of the predictor variable is equally correlated with the variables in the active set. Using C_j to denote $x_j \cdot (y - \hat{\mu}')$ and a_j for $x_j \cdot u$ for $j \notin \mathcal{A}$, we can solve for variable j's particular γ' value(s), where variable j becomes equally correlated with the residuals as the variables already in the active set. For variable $j \notin \mathcal{A}$, this length is

$$\gamma'_j = \frac{C \pm C_j}{a \pm a_j} \,. \tag{2.31}$$

There are two solutions for γ'_j corresponding to both a positive and negative correlation, with magnitude equal to that of the correlation of variables in the active set.

A negative value for γ'_j indicates movement of the model in the direction opposite to the direction to the OLS solutions of variables in the active set. Thus, a negative γ'_j results in a model that necessarily explains less variance than the preceding model (on the training data at least). Further, Equation 2.30 shows that a value of γ'_j of

C/a results in a model where the active set has zero correlation with the resultant residuals. In other words, using C/a gives the OLS solution for the variables in the active set and going beyond C/a will begin to reduce the performance of the model, on the active set, at least. The relationship between correlation and the size of γ' is shown graphically in Figure 2.4 and the geometry resulting from $\gamma' > C/a$ is shown in Figure 2.5. Such a scenario can reasonably be described as overfitting.

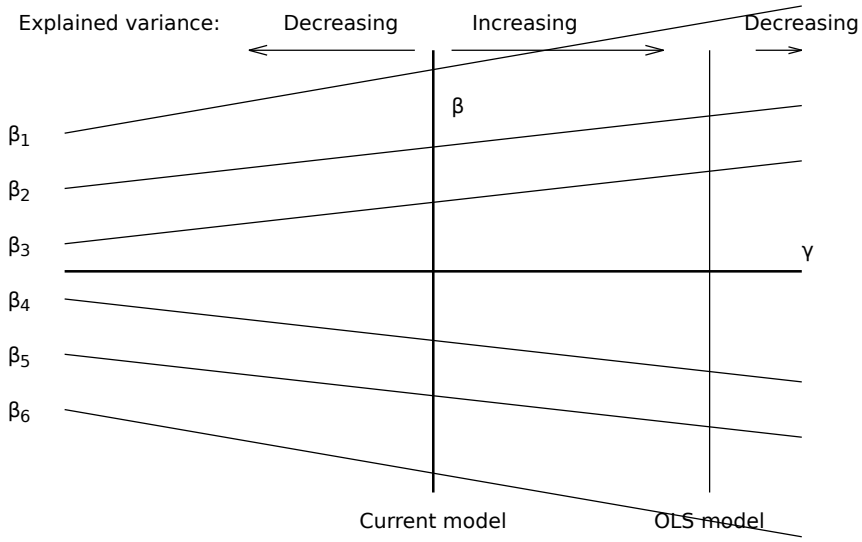

Figure 2.4: As γ' increases, the coefficients move towards the least squares solution. Approaching the least squares solution, the variance explained increases until the least squares solution is obtained at $\gamma' = C/a$. Beyond C/a, we move away from the least squares solution and the variance explained drops.

Thus, for each j not in the active set, we examine the two γ'_j from Equation 2.31 and select at most one value to be a potential γ. For each j, we take the smallest positive value for γ'_j and even then, only if it is less than C/a.

In the LARS algorithm, we always have a γ'_j available, but this is not necessarily the case under clars. In clars, we *skip* expensive variables that would have been selected under LARS. Thus, clars may introduce variables more correlated with the residuals than variables in the active set, which is not possible under LARS. For large

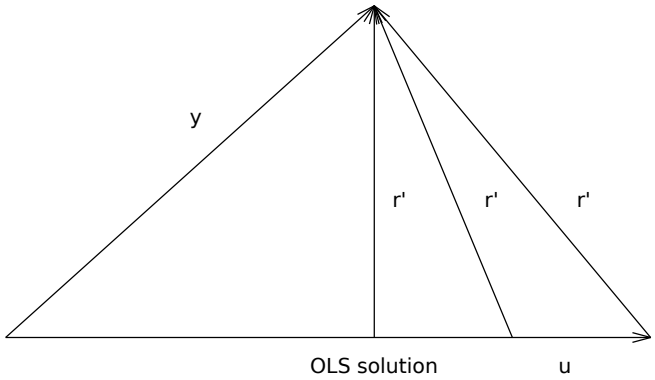

Figure 2.5: The ordinary least squares solution is obtained when the residual vector is perpendicular to the least squares direction u. This is obtained when $\gamma' = C/a$. (Shown as a 2D projection)

problems with many variables, the kind of problem for which we have designed clars, this may not prove to be an issue until the supply of variables not in the active set nears exhaustion. However, for small problems, we could be left without a valid γ'_j proposal for γ and therefore must find a method to resolve the issue of no valid γ'_j. We detail this in step five.

If all variables are already included in the active set, we create a single proposal for γ of C/a as we attempt to fit the OLS solution.

2.8.5 Step Five: Choose The Next Variable

Now in step five, we must select one of the γ'_j for $j \notin \mathcal{A}$ calculated in step four. As we described in Section 2.7, we choose the γ'_j resulting in the greatest correlation per total cost of the active set. We introduce $cost_{\mathcal{A} \bigcup \{j\}}$ to indicate the total cost of measuring the variables in $\mathcal{A}$ as well as variable j. For simple problems where costs are additive, total cost will be the sum of the individual costs: $cost_{\mathcal{A} \bigcup \{j\}} =$

$cost_j + \sum_{k \in \mathcal{A}} cost_k$. If cost is not additive, as discussed in Section 2.10, the cost must be specified by the researchers as a function of which variables are in the model.

For each $j \notin \mathcal{A}$, we find the correlation between variable j and the resultant residuals after incrementing the model predictions along u by γ_j. We calculate the score S_j as that correlation divided by the cost of the active set and variable j together. An easy mistake in this calculation would be to merely calculate

$$S_j = \frac{C - \gamma_j' a}{cost_{\mathcal{A} \bigcup \{j\}}} \, . \tag{2.32}$$

However, this is wrong because $C - \gamma_j' a$ is the projection of a variable in the active set and the resultant residuals, but not the actual correlation. Though using the projection in step two instead of correlation was acceptable, it is not in this case. In step two, the calculation performed for each variable not in the active set used the same residual vector. Here, the scale of the residuals depends on the value for γ'. For a fixed value of γ', this calculation will result in a value proportional to correlation, but as we vary γ', the variance of the residuals also varies. Thus, we must scale Equation 2.32 by the variance of resultant residuals for that particular γ_j'. An easy way to do this would be to calculate the resultant residuals for each γ_j' and find the variance. However, this calculation must be performed for each variable not in the active set, which could be a large number of variables, thus reducing the efficiency of the algorithm. Luckily, a more efficient calculation is available.

The derivation of the more efficient algorithm depends upon the residuals being centered at 0, so we begin by showing this is true. First, assume the residuals are centered at zero at iteration k. Now, the resultant residuals are

$$r' = y - \hat{\mu}' \tag{2.33}$$

$$= y - (\hat{\mu}_k + \gamma' u) \tag{2.34}$$

$$= r_k - \gamma' u\,, \tag{2.35}$$

and the center of the resultant residuals is

$$\text{mean}(r') = \text{mean}(r_k) - \gamma' \text{mean}(u) = 0 - \gamma' \text{mean}(u) \tag{2.36}$$

since we assumed that r_k are centered at 0. Recall, u is the equiangular vector in the space spanned by the predictor variables in the active set, and therefore can be written as

$$u = \sum_{j \in \mathcal{A}} a_j x_j\ . \tag{2.37}$$

More to the point, u is a sum of no more than p vectors, themselves centered at 0 (via step one). Thus, u is centered at 0 and r' is as well, if r_k is. Again, in step one, we centered the response variable y which is the residual vector for the first iteration, r_0. Hence, r' for any iteration is centered at 0.

Having shown that the residuals are centered at 0, we can continue to the efficient calculation of the correlation between r' and x_j. Since r' and x_j are both centered at 0 and the standard deviation of x_j is 1, the correlation between them is equal to $r' \cdot x_j/(n * sd(r'))$. We also know that $r' \cdot x_j = C - \gamma' a$ for variables in the active set. Therefore, if we can find $sd(r')$ efficiently, we can find $(C - \gamma' a)/(n * sd(r'))$ and thus efficiently calculate the resultant correlation.

Since r' is centered at 0, the variance of r' is $r' \cdot r'/n = |r'|^2/n$. We can avoid calculating the computationally expensive dot product if the length of r' is known. In fact, the length can be calculated efficiently assuming we know the length of r_k. From Equation 2.35, we see the relationship from r_k to r' and since u is a unit vector, the second term has magnitude of γ', and graphically is demonstrated in Figure 2.6.

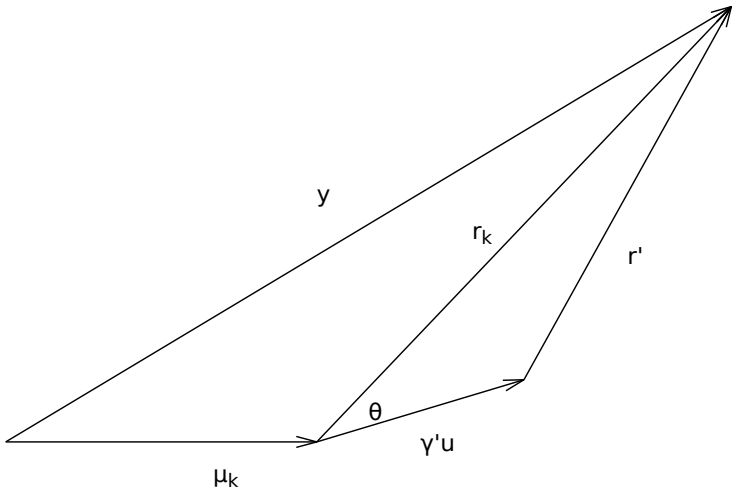

Figure 2.6: A triangle is formed with the vectors r' (the updated residuals), r_k (the current residuals), and $\gamma' u$ (a length along the least squares direction). The angle formed between r_k and $\gamma' u$ is independent of γ' since it is a scalar. Calculating this angle, we can efficiently determine the length of r' for any value of γ'.

Thus, we know the length of the r_k side of the residual triangle shown in Figure 2.6 as well as the length of the $\gamma' u$ side. Given one angle in that triangle we can calculate the length of the third side, the resultant residuals r'. Two of the angles depend on the value of γ' and, as pointed out previously, we may have as many potential γ''s as variables not in the active set. The third angle, θ between r_k and u, only depends on r_k and u, and can be found a single time and used for each of γ' values. We can find θ by taking the dot product of r_k and u and solving $\theta = acos(r_k \cdot u/|r_k|)$

where $|r_k|$ is the length of r_k which we have previously assumed to be known. With θ known, we can use the law of cosines to find the length of r':

$$var(r') \propto |r'|^2 = |r_k|^2 + \gamma'^2 - 2\gamma'|r_k|cos(\theta) \; . \tag{2.38}$$

Thus, we can efficiently calculate the variance of the resultant residuals as long as we know the length of the preceding residuals. For the first iteration, we have $r_0 = y$ and it is simple to calculate the variance of y from the start and subsequently calculate $|r_k|$ using Equation 2.38 for each iteration.

At last, we calculate the appropriate clars score:

$$S_j = \frac{C - \gamma'_j a}{cost_{\mathcal{A} \bigcup \{j\}} |r'(j)|/n} \tag{2.39}$$

where $r'(j) = y - \hat{\mu}'(\gamma'_j)$.

Having finally found the score for each variable j not in the active set, we turn to selecting the appropriate value for γ. Before we finalize our selection of γ, we must check if a *Lasso step* is necessary. A Lasso step follows the same Lasso adaptation of LARS described earlier: if a coefficient will cross zero as we increment γ', we stop at that point and drop the variable whose coefficient crosses zero from the active set. For all variables, j in the active set, we calculate the value for γ' (if any), at which point the coefficient will cross zero, which we will call $\tilde{\gamma}'_j$ and calculate with

$$\tilde{\gamma}'_j = -\hat{\beta}_{k,j} / (a(X'_{\mathcal{A}} X_{\mathcal{A}})^{-1} \underline{1}) \tag{2.40}$$

where $\hat{\beta}_{k,j}$ is the coefficient for variable j after the k^{th} iteration. The value in the denominator will become clear in step six. We now take the smallest $\tilde{\gamma}'_j$ and compare to our presumptive γ selection. If $\tilde{\gamma}'_j$ is shorter than γ, then $\tilde{\gamma}'_j$ becomes our γ selection

and we note that we will be performing a Lasso step in step six. Otherwise, we continue to step six with the presumptive selection of γ confirmed.

If the selected value of γ does not match the selection that would have been selected by the Lasso adaptation of the LARS algorithm, we record the iteration as a deviation from the Lasso algorithm. As we described in step four, once clars leaves the LARS solution path, we may be left in an iteration with no acceptable values of γ_j for step five. If this is the case, we return to the last clars iteration that was along the LARS solution path and remove the variable selected by clars last time from the options for this point in the solution path.

2.8.6 Step Six: Update the Model

Finally, we update the model; the new model predictions vector is

$$\hat{\mu}_{k+1} = \hat{\mu}_k + \gamma u \ , \tag{2.41}$$

and the residuals are

$$r_{k+1} = r_k - \gamma u \ . \tag{2.42}$$

Finally, we need to update the model coefficients. Since $\hat{\mu}_k = X\hat{\beta}_k$, after the update we must have

$$\hat{\mu}_{k+1} = \hat{\mu}_k + \gamma u = X(\hat{\beta}_k + w) \ , \tag{2.43}$$

where w is a p-vector which is the increment from the preceding $\hat{\beta}$. Since Xw must be equal to γu, we can use Equation 2.24 and see that the elements of w corresponding to members of the active set must be

$$w_{\mathcal{A}} = a(X'_{\mathcal{A}} X_{\mathcal{A}})^{-1} \underline{1} \tag{2.44}$$

and 0 if not in the active set, while being careful with predictors having negative correlation with the residuals. Finally, we add the selected variable to the active set unless we are performing a Lasso step, in which case we remove the selected variable from the active set.

2.9 Additive Costs on the Diabetes Dataset

Like similar algorithms before, we demonstrate the use of clars on the diabetes dataset. This dataset was used in the introduction of LARS [Efron et al., 2004], the LARS R package [Hastie and Efron, 2013], and Yue's thesis introducing blars [Yue, 2010]. The dataset contains 442 observations on 10 predictor variables: age, sex, BMI, average blood pressure, and six blood serum measurements. The response variable is a quantitative measure of diabetes progression one year after a baseline measurement. By today's standards, this is a relatively small dataset. Using a small dataset makes it easier to investigate the performance of clars and compare to other procedures like LARS and Lasso.

We excluded blars from the examples because of the difficulty in setting the blars parameter ω in tandem with the lasso parameter λ. Using a fixed ω and adjusting λ, we could create similar plots to those shown in the examples. However, these plots would only be specific to that ω, and different values for ω would result in very different model fits. We felt that including blars fits in the plots with a single ω would be misleading if we did not discuss the selection of ω, or distracting if we presented an explanation for choosing a selected ω. Further, we could have produced a number of

blars sequences each with a different value for ω, but again, we felt this would result in an overly inked plot which would also distract from our greater message.

For our comparison, we remove 88 randomly selected observations (about 20%) from the diabetes data set, fit the set of models for LARS, Lasso, and clars on the remaining 354 observations, and evaluate the performance of the model fits on the withheld observations. We used the mean of the squared prediction errors as our evaluation criterion. Each algorithm produces a sequence of model fits, and each fit will be associated with a specific model cost (the sum of the costs to measure each variable included in the fit). We plot the performance of the model (mean of squared prediction error) against the model cost for both the training data and hold-out data sets. The diabetes dataset does not include costs per variable. We simulate variable costs under several scenarios to demonstrate the performance of each of these models for a variety of variable cost conditions. The costs of the variables for each scenario are provided in Table 2.2 and we discuss the scenarios under which we simulate them in the sections that follow.

Table 2.2: The cost of variables is recorded for each of three scenarios of additive costs. In the first column, each variable has the same cost. In the second column, all variables have a cost on the same order of magnitude. In the third column, the costs span orders of magnitude.

variable	equal	sameOrder	differentOrder
age	1.00	2.37	78.58
sex	1.00	6.18	48.01
bmi	1.00	1.48	26.26
map	1.00	1.63	88.55
tc	1.00	5.40	2645.31
ldl	1.00	2.37	4.06
hdl	1.00	4.37	4531.24
tch	1.00	3.70	4.46
ltg	1.00	9.73	3627.13
glu	1.00	3.73	40.31

Our use of clars for this problem falls under the scenario of a historic data set. We assume the listed variables have been recorded for previous subjects, but for future subjects, we wish to predict using only a subset of the variables. In these examples, we use financial cost as the driver for the cost-based variable selection but other considerations could also be appropriate. The emotional cost to subject a patient to a litany of medical tests may be be worth evaluating.

2.9.1 Equal Costs

Our first scenario is that of equal costs. For simplicity and illustration, we've set each variable to cost 1 unit (equal column of Table 2.2). Thus, the total cost of a particular model is equal to the number of variables included in that model. This may not be an interesting example as far as cost is concerned but it is important to investigate simple scenarios both as a baseline to compare to more interesting costs and to evaluate the performance of the algorithm.

Before investigating the results, we can predict the relative performance of clars, LARS, and Lasso. Of course, LARS and Lasso do not consider costs at all, and their performances should be similar since the LARS and Lasso are such similar algorithms. In fact, depending on the data, LARS and Lasso may align exactly. For a cost-considerate version, we recognize that in the equal cost case, no variable has any preference to another as far as cost is concerned. By inspecting steps two and five (where cost plays a role) of the clars algorithm, we can see that clars reduces to Lasso in the equal cost case. In step two, we calculate correlation per cost, but since all the costs of all variables are the same, it is equivalent to selecting the variable with the largest correlation with the residuals, identical to LARS/Lasso. With equal costs, step five reduces to selecting the γ_j' which maximizes the correlation between x in the active set and r'. Since that correlation monotonically decreases to 0 as

γ' increases, we select the smallest γ'_j, equivalent to the LARS procedure. Thus, the clars algorithm for additive equal cost variables, should result in identical fits to Lasso.

In fact, the clars and Lasso fit are identical and the LARS fit is nearly the same. By inspecting a plot of the mean square prediction error (MSPE) on the training data as a function of cost, we may select a single model from our set (left plot of Figure 2.7). If we are interested in out-of-sample prediction performance, a similar plot on the hold-out set may be more valuable (right plot Figure 2.7). Again, clars and Lasso perform identically, and LARS nearly identically. Based on the out-of-sample plot, we might recognize that the model costing 7 units offers adequate performance at a comparatively low price. Or, depending on the cost the researchers are comfortable paying, the model at cost 2 units may be good enough.

2.9.2 Costs on Same Order of Magnitude

For our next scenario, we randomly assign costs to the 10 predictor variables. We selected the costs from a uniform distribution from 1 to 10 units so all are on the same order of magnitude (Table 2.2 column sameOrder). Such a scenario may arise when the process to acquire a measurement is similar for each variable. In this example, we should no longer expect clars to match exactly with Lasso.

First, we discuss what should happen to LARS and Lasso in this scenario. Since LARS and Lasso do not incorporate cost in their model fit at all, we produce the same LARS and Lasso fits as in the preceding example with equal costs. However, the costs of those same fits will now be different values. We see the LARS and Lasso points in Figure 2.7 maintain the same y-value (mean square prediction error), but the spacing along the x-axis (cost) is irregular. If the clars points dip below the LARS and Lasso points in the same region of the x-axis, then models produced by clars in

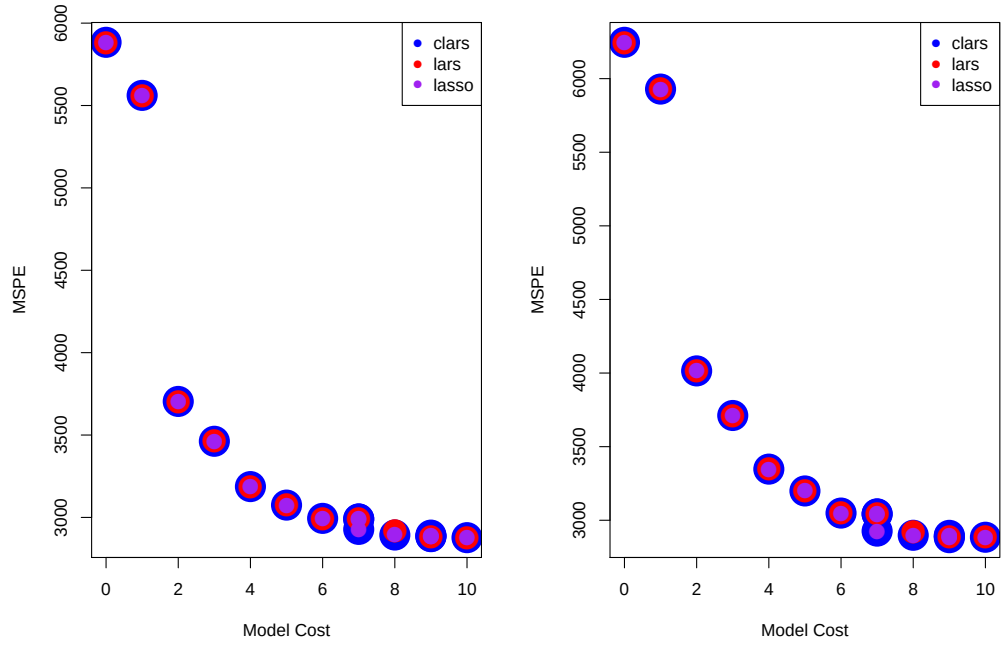

Figure 2.7: The mean square prediction error (MSPE) is plotted against model cost for diabetes dataset when each variable has a one unit cost. The left panel predicts on the training set and the right panel predicts on a hold-out set.

that cost area outperform the other methods in terms of MSPE. Likewise, it may be possible for LARS or Lasso to actually outperform clars in certain cost areas.

In this case, clars tends to outperform LARS and Lasso in MSPE for nearly all model costs. In particular, we notice the introduction in the clars algorithm of several cheap models performing well at a cost under 10 units. The LARS and Lasso model fit associated with a model cost of 2 units in the previous, equal cost example now costs about 10 units. This means the second variable costs about 9 units and is relatively expensive for this example. Under clars, that variable was evidently too expensive to enter the model at that point. Instead, we see high performing cheaper models under 10 units in cost, and even at 10 units of cost, clars outperforms the expensive two variable model of LARS/Lasso. Beyond 10 units of cost, we see a small range of model cost where LARS/Lasso may actually be preferred to clars, but subsequently, researchers may find little distinction between performance of the different algorithms while incorporating cost.

Interestingly, we see some separation between LARS and Lasso near the more expensive models not previously observed. This is due to the fact that the Lasso fit may remove a variable from the active set thus changing from a model with k variables to a model with $k-1$ variables. In the equal cost example, any model with the same number of variables has the same cost, but in this example, models with the same number of variables can have different costs if the variables are different. We also notice that some models with the same costs (again, a result of Lasso steps) in the left panel of Figure 2.8 have a greater separation in mean square prediction error than in the right panel, which uses the out-of-sample data. The degree of regularization performed by clars, Lasso, and LARS may vary even for models of the same cost. In this case, the same general pattern among the algorithms is observed in both the training and test data sets (left and right panel of Figure 2.8, respectively).

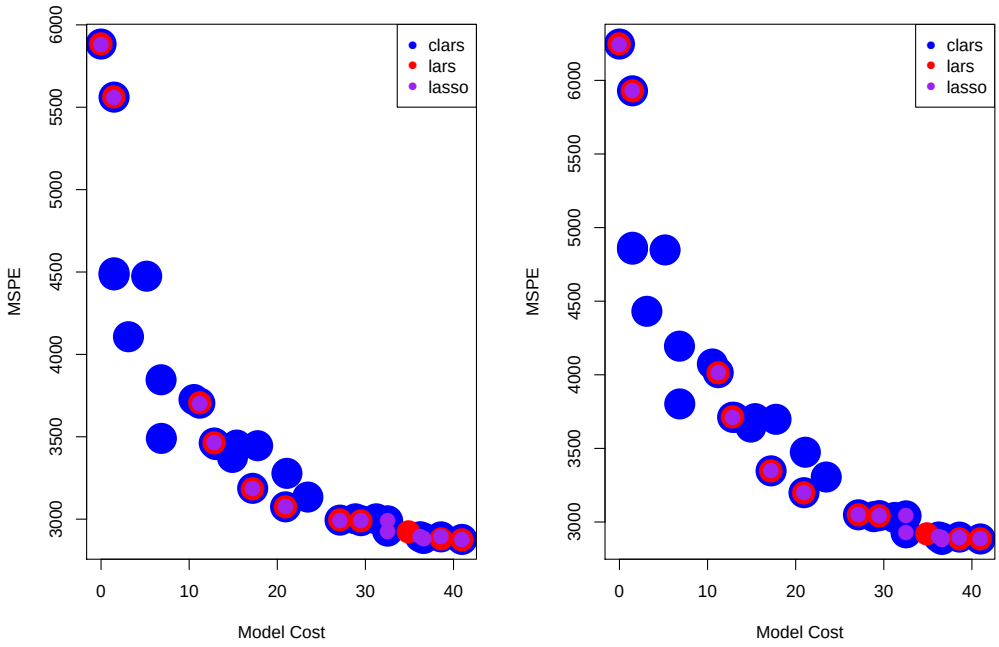

Figure 2.8: The mean square prediction error (MSPE) is plotted against model cost for diabetes dataset for random costs of variables when costs are on the same order of magnitude. The left panel predicts on the training set and the right panel predicts on a hold-out set.

2.9.3 Costs on Different Order of Magnitude

Finally, we come to an example of variables with costs of different orders of magnitude. In this example, we randomly draw exponents from a uniform distribution from 1 to 10 for base e to obtain costs on different orders of magnitude. This scenario represents a case where the cost of taking a measurement depends greatly on which variable is being measured. Again, we do not expect the clars algorithm to match LARS or Lasso.

Like before, we obtain the same LARS and Lasso fits, but their positions on the x-axis are staggered due to the great differences in variable costs. With some variables very expensive, we should expect to see that in the clars results, more expensive variables tend to be saved for later in the algorithm.

In fact, a model at almost negligible cost can perform nearly as well as the most expensive models from LARS and Lasso (Figure 2.9). Only once the upper echelon of model cost is reached do LARS and Lasso tend to reach the performance of clars. Depending on the desires of the researchers, the most expensive models may be disregarded in favor of a cheaper model proposed by clars.

This scenario shows greater discrepancy between clars and the other algorithms than seen in the scenario with same order of magnitude costs, with clars typically outperforming LARS and Lasso at many cost levels. We reason that with variables on different orders of magnitude, clars may find models using a number of variables at a lower cost than a LARS or Lasso model naively introducing an expensive variable early in the sequence. This scenario may be representative of many real-world problems and may indicate the benefit to using clars when cost-considerate variable selection is needed.

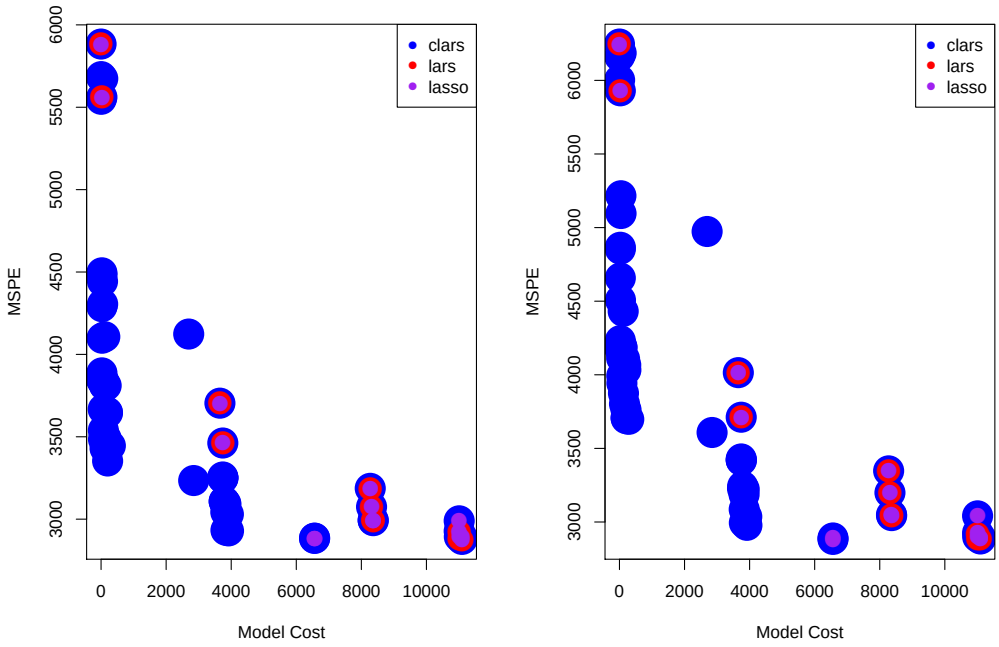

Figure 2.9: The mean square prediction error (MSPE) is plotted against model cost for diabetes dataset for random costs of variables when costs are on a different order of magnitude. The left panel predicts on the training set and the right panel predicts on a hold-out set.

2.10 Non-Additive Costs on Diabetes Dataset

Additive costs are intuitive and easy to explain, but not necessarily how variable measurement works. For example, in the diabetes data set, we note there are six blood measurements. For these measurements, the bulk of the cost may be in extracting blood from the patient and sending it to a lab. Thus, after paying this base cost for a single blood measurement, subsequent measurements may be relatively cheap. This idea of shared costs is discussed in [Turney, 1995] and [Turney, 2000]. Certainly, this shared costs idea is not specific to blood measurements or even the medical field. Shared costs may even be present in image analysis problems where different measures share a common base processing algorithm and the cost is computer time.

Non-additive costs can arise not only due to the study design itself but due to the form of the model. For example, when building a model with interactions (say x_1x_2), once x_1 and x_2 are measured, their interaction x_1x_2 may be essentially free. Similarly, having measured x_1, any transformations of x_1 ($\log(x_1)$, ${x_1}^p$, etc) require only (trivial, typically) computation time.

In general, non-additive costs are likely to be an accurate description of real world problems. Making sure our algorithm can handle such model cost functions is crucial. Our algorithm simply and efficiently handles such a scenario. As we previously mentioned, handling non-additive costs in clars merely requires appropriately calculating the denominator in Equation 2.39. We consider a model cost function dependent upon the set of variables identified as necessary for the model. For example, in the additive cost framework, if the cost of measuring variable j is denoted $cost_j$, then

$$cost_{\mathcal{A}} = \sum_{j \in \mathcal{A}} cost_j \ . \tag{2.45}$$

For non-additive models, the exact cost function is defined by a researcher with intimate understanding of the cost structure of potential study designs. For demonstration purposes, we provide a few simple examples.

First, consider the case that the variables x_1 and x_2 are potential inputs and potential predictors in the model are x_1, x_2, and x_1x_2 where the interaction has 0 cost after measuring both x_1 and x_2. Listing the predictors included in the model explicitly in place of the symbol $\mathcal{A}$, the model cost function is defined as follows

$$cost_{\{x_1\}} = cost_1 \tag{2.46}$$

$$cost_{\{x_2\}} = cost_2 \tag{2.47}$$

$$cost_{\{x_1,x_2\}} = cost_1 + cost_2 \tag{2.48}$$

$$cost_{\{x_1,x_2,x_1x_2\}} = cost_1 + cost_2 \tag{2.49}$$

$$cost_{\{x_1,x_1x_2\}} = cost_1 + cost_2 \tag{2.50}$$

$$cost_{\{x_2,x_1x_2\}} = cost_1 + cost_2 \ . \tag{2.51}$$

Here, six distinct models correspond to only three different costs.

In another scenario, it may be the case that dollar quantities do not accurately reflect the effective cost to the researchers. Spending anything below a certain threshold may be represented by the dollar quantity, but each dollar above that quantity may accrue a larger and larger hardship upon the researchers, non-linearly. This additional hardship may represent the effort that will be required to obtain more funding or a perceived penalty for transferring funds intended to be spent elsewhere. Suppose, after careful consideration, researchers have identified that a reasonable model of their effective cost is an additive function of dollar costs below a cutoff, a,

and an exponentially increasing function above that cutoff. For such a cost model, the researchers might use the following cost function

$$cost(\mathcal{A}) = \begin{cases} \sum_{j \in \mathcal{A}} cost_j & \text{for } \sum_{j \in \mathcal{A}} cost_j < a \\ a \exp\left(\left(\sum_{j \in \mathcal{A}} cost_j - a \right) / a \right) & \text{for } \sum_{j \in \mathcal{A}} cost_j > a \end{cases} . \tag{2.52}$$

This cost function is represented in Figure 2.10 with an example cutoff of $a = 100$ by plotting effective cost against financial cost.

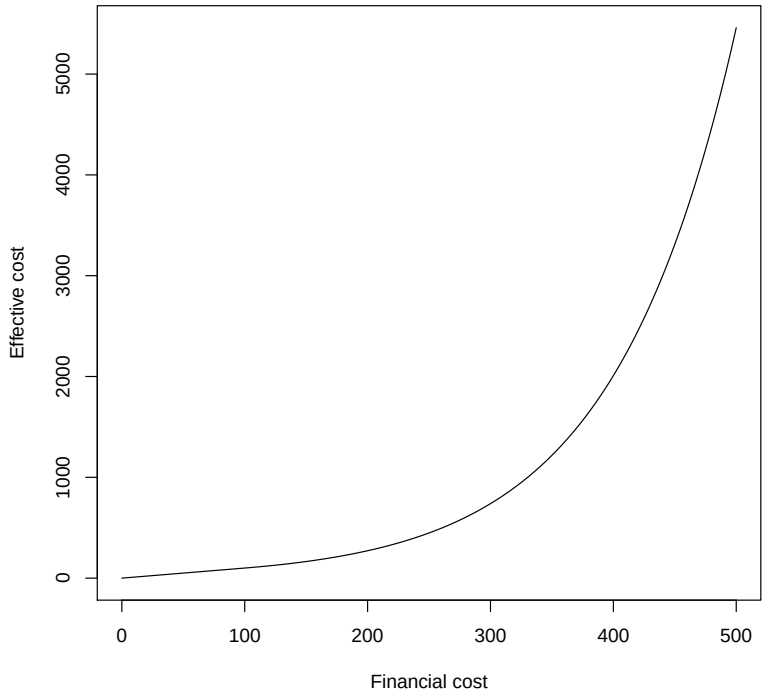

Figure 2.10: The effective or perceived cost for researchers to measure a set of variables need not equal the financial cost. An example non-linear cost function (Equation 2.52) for effective cost based on a financial cost is plotted.

2.10.1 Blood Tests

In the diabetes dataset, we note six variables are based on analyses of blood. Thus, a common base cost might be shared among these variables and after the first blood test, subsequent tests may be cheaper. For this scenario, we consider all variables to have a random cost, but all on the same order of magnitude. We use the same costs as in the `sameOrder` column of Table 2.2, except we set the blood test variables to share a common cost. The blood test variables are `tc`, `ldl`, `hdl`, `tch`, `ltc` and `glu`, the cheapest of which is `ldl` at 2.37. In our example, we assume a cost of 2.00 is shared among the blood test variables. Thus, the first blood test variable costs as much as reported in the table, and the second and beyond cost 2.00 less than reported (Table 2.3).

Table 2.3: In the scenario of shared cost among blood tests, the cost to measure a variable depends upon which variables have already been measured. The first column indicates the cost of variables when no blood tests have yet been performed. The second column indicates the cost of variables after at least one blood test has been performed.

variable	tc-glu not in model	tc-glu in model
age	2.37	2.37
sex	6.18	6.18
bmi	1.48	1.48
map	1.63	1.63
tc	5.40	3.40
ldl	2.37	1.37
hdl	4.37	2.37
tch	3.70	1.70
ltg	9.73	7.73
glu	3.73	1.73

Under this scenario, we find only small difference from the scenario of additive costs where costs were on the same order of magnitude (compare Figure 2.11 and

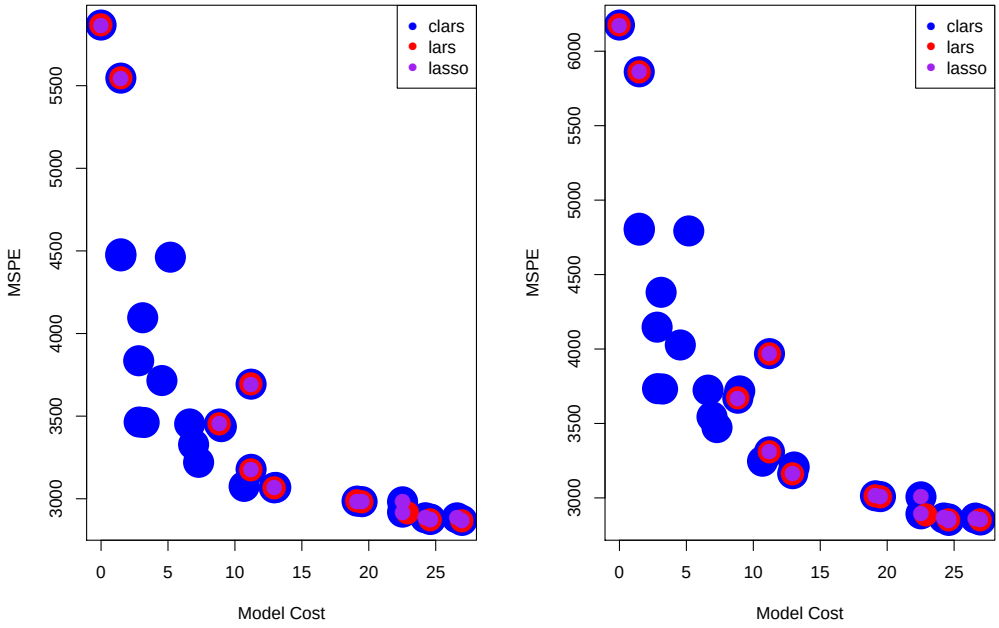

Figure 2.11: The mean square prediction error (MSPE) is plotted against model cost for diabetes dataset for random costs of variable when costs are shared among variables. The left panel predicts on the training set and the right panel predicts on a hold-out set.

Figure 2.8). The main take-away for this scenario is the ease at which a non-additive cost function can be implemented with clars.

2.11 Overlapping Predictor Variables

The examples above include 10 predictor variables. Forgetting about different levels of regularization for the moment and only considering what variables are included in the model, we see each variable can be included or not. Thus, there are $2^{10} = 1024$ potential variable combinations. Certainly, this is a large enough number of models we would not wish to explore them by hand. However, a computer would likely make quick work of fitting merely 1000 different models, assuming the sample size n is relatively small (like this case where $n = 442$).

We believe it is instructive to observe the performance of clars on a problem of this size because it is small enough one might be able to wrap their mind around the different models produced. Further, this data set has been fit previously with LARS, blars, and Lasso and thus we think using clars adds to the suite of existing fits. However, we described the motivation of clars to be computationally efficient with high dimensionality data. Therefore, we created a larger artificial data set to examine with clars. We took the existing diabetes data, duplicated the variables, and added noise to the duplicated variables. For each variable from the original data set, we standardized the variable (set mean to 0 and sd to 1) and added random noise (mean 0 and sd 0.5 normal distribution). For the `sex` variable, we selected 20% of the observations and switched categories rather than adding random continuous noise to the dichotomous variable.

Presumably, these duplicated, or overlapping, variables are "worse" than their corresponding original, because of the added random noise. Thus, we can investigate the performance of clars when two similar variables are available at different costs.

Further, we have greatly increased the size of our problem. While 10 variables resulted in 1024 different variable combinations, 20 results in $2^{20} = 1048576$ different variable combinations.

This example allows us to demonstrate the flexibility afforded by non-additive cost functions, in addition to the use of clars on a large model space. In the case of two or more variables ostensibly measuring the same fundamental quantity, researchers may wish only to select one for their model. It is easy to simulate such a scenario with non-additive costs. Once one of the variables measuring a common quantity is included in the active set, we can define the cost to include another variable measuring this quantity to be arbitrarily high.

For this problem, we use variable costs in the differentOrder column of Table 2.2 for the original variables and half of those costs for the noise-added variants. We define the model cost to be equal to the sum of the variables in the model when no overlapping variables are included and to be 1,000,000 units when overlapping variables are present. Under such a contrived cost function, we must investigate the results slightly differently. Here, we only plot the models produced by the various algorithms which do not have both of any overlapping variable pairs (Figure 2.12) and discard the models artificially costing 1,000,000 units or more.

In this case, there are 10 pairs of variables and for each pair we can include one or the other, but not both variables. This results in $3^{10} = 59049$ allowable models, but LARS and Lasso lack the ability to preferentially consider these models. Thus, there are few satisfactory options using LARS and Lasso (Figure 2.12). Meanwhile, the clars algorithm produces many satisfactory models that outperform LARS and Lasso at a given cost.

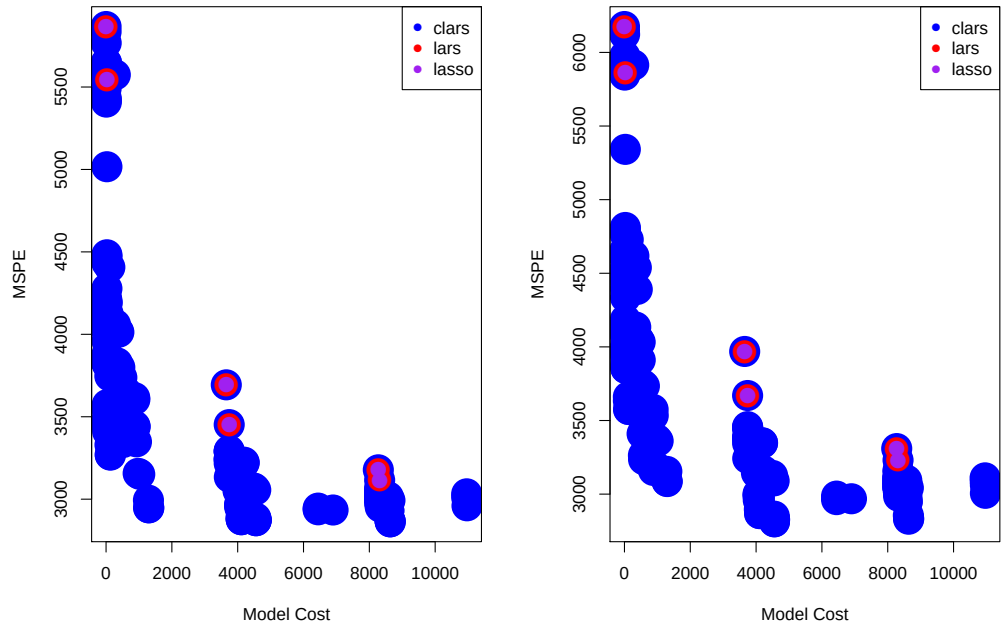

Figure 2.12: The mean square prediction error (MSPE) is plotted against model cost for diabetes dataset for random costs of variables when costs are on a different order of magnitude and each variable has a cheaper, noisy version available. Each algorithm produces models freely, but we subset the resultant models to only those which include at most one variable for each of the paired variables. The left panel predicts on the training set and the right panel predicts on a hold-out set.

2.12 Discussion

In this chapter, we develop a new algorithm based on the Lasso adaptation of LARS to perform simultaneous cost-considerate variable selection and model regularization. We demonstrate the use of this clars algorithm in analyzing the classic diabetes dataset, and investigate relative performance of clars to the LARS and Lasso algorithms for several cost scenarios. In this section, we further discuss a few of the design choices of the clars algorithm and discuss several important points.

First, we revisit the introduction of cost to the algorithm via a cost function. As we previously discussed, two clear choices arise for the cost function in the denominator in steps two and five: the cost to measure all variables in the active set or the cost to measure the next variable to enter the active set. We find several reasons to use the cost of all variables. First, using the cost of the active set easily allows for non-additive cost functions. Also, the penalty on relatively expensive variables becomes less steep as the model cost grows. Using the cost of the next variable only, we may find that the most expensive variables are always saved until the end of the fitting algorithm. Lastly, the total model cost strategy handles interactions and other transformation variables more efficiently. Presumably, interactions and transformations have no cost after the constituent variables are already in the model. Using the individual variable cost strategy, zero-cost variables will be the first choice to add. However, by considering the full model cost, such zero-variables are not guaranteed to be selected, thus potentially improving out-of-sample prediction. This discussion leads to an important point. Researchers should avoid setting a variable to zero cost unless it requires other non-zero cost variables to already be selected for the model. Otherwise, all of these variable will be pressured to be the first variables to enter the model, assuming we adjust the algorithm slightly to account for the

possibility of a denominator of zero (we could simply select the largest correlation of the cases where the denominator is zero).

Another point worth discussing is that the algorithm performs regularization in two ways. It produces models where only a subset of the variables are active and even these models are regularized relative to the OLS model for that subset of variables. The same is true of LARS and Lasso. We suspect that in many cases, especially for large problems, this will result in the best performing models on out-of-sample data. However, in cases when researchers want a very cheap model, they may find clars has produced an overly regularized model. In these cases, the researchers might consider fitting a sequence of OLS models adding variables one at a time in the same order they enter via the clars algorithm. A similar idea is proposed in Efron et al. [2004].

Finally, we consider the greater vision of cost-considerate variable selection and the place of clars within it. Cost-considerate variable selection is a relatively unexplored problem in statistics. In recent years, there has been a tremendous push toward big data from science to private companies. The infrastructure necessary to handle data at the largest scales is extensive. Maintaining the associated databases and employing efficient analyses on that data take skill, time, and money. If the costs associated with maintaining this infrastructure can be reduced while preserving the functionality and performance of data analytics, then from a practical perspective, these steps should be taken. Cost-considerate variable selection, as the field grows, may prove a valuable consideration for business and science alike.

It remains to be seen where the clars algorithm will fit in the greater cost-considerate variable selection domain. It is unlikely for a single algorithm to solve all cost-considerate variable selection applications since the space where it may be useful is very broad. The types of statistical models and the particular domains of application are likely to dictate specific demands that influence the development of

novel algorithms. The clars algorithm represents an approach for linear modeling. Linear models have proven very effective in a large number of domains. We believe our contribution to cost-considerate variable selection is valuable, but the field is young, and it will be interesting to see how it evolves.

CHAPTER THREE

EQUIVALENCE TESTING FOR MCMC CONVERGENCE ASSESSMENT

3.1 Markov Chain Monte Carlo and Convergence Assessment

A typical objective of Bayesian data analysis is to obtain summaries of a posterior distribution $p(\theta|y)$, such as credible intervals. Analysts can use Markov chain Monte Carlo (MCMC) methods to produce a set of samples $\Theta_n = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(n)}\}$ from the target posterior distribution. Under appropriate conditions, the set of samples converges to a set of draws from the posterior distribution as the number of samples, or length of the chain, increases [Meyn and Tweedie, 1993]. From these draws, the researcher can calculate summaries such as medians, means, intervals, and probabilities. MCMC is widely used because it is typically easier to produce a set of samples, Θ_n, than to analytically calculate summary values of $p(\theta|y)$ [Flegal et al., 2008, Gelman et al., 2003, Liu, 2001, Robert and Casella, 1999]. In fact, today there are many software solutions automating much of the process to obtain Θ_n [Lunn et al., 2000, Plummer, 2003, Stan Development Team, 2014b, etc.].

In order to make a claim of convergence, researchers must make a decision about how long to sample until the summaries are close enough to intractable, analytical results, keeping in mind that different MCMC summaries may approach the analytic summary at different rates. Chains that have not sampled for long enough to appropriately explore the posterior space are liable to result in MCMC summaries that are not good approximations to the posterior value [Brooks and Roberts, 1998, Cowles and Carlin, 1996].

Tail quantiles, as endpoints of credible intervals, are often summaries of interest and may be more sensitive to MCMC chains that have not appropriately explored the parameter space than the posterior mean is, for example. Therefore, the researcher should explicitly assess whether the calculated MCMC summaries are adequate estimates of the posterior summaries of interest. If so, the researcher may say that the chains have "converged", meaning that they are comfortable terminating the sampler and providing summaries based on the MCMC draws.

We have found the philosophy of MCMC convergence checking to be similar to that of equivalence testing. In equivalence testing, a null hypothesis of a difference between two quantities is posed and the strength of evidence for similarity is assessed. Early work in, and advocation for, equivalence testing was performed by Altman and Bland [Altman and Bland, 1983, 1995] mainly in the biomedical field where it has been referred to as non-inferiority testing and bioequivalence due to the nature of its applications [Berger and Hsu, 1996]. When a new drug or therapy must be shown to be equivalent to an existing therapy (rather than strictly better), equivalence testing may be used [Windeler and Trampisch, 1996]. Now, equivalence testing is used in a variety of applications and disciplines including industrial quality assurance [Richter and Richter, 2002] and statistical model validation [Robinson and Froese, 2004]. In convergence assessment, the default stance should be that chains have not converged and therefore, summaries from chains are different from the true posterior value as well as from other chains. Only after substantial evidence for similarity across chains can we assert that chains have reached approximate convergence. Despite this obvious connection, equivalence testing is a novel framework within the convergence testing literature.

There are at least two aspects to consider when assessing the adequacy of MCMC summaries: (1) whether the set of draws mimic draws from the posterior distribution

and (2) whether the number of draws is enough to approximate posterior quantities to a desired precision. In the first case, the MCMC sampler may not have been run long enough to produce a set of draws where those draws mimic a sample from the posterior. It is possible the empirical chain distribution is still influenced by the chain's starting value in which case the empirical chain distribution is unlikely to be a good approximation of the posterior [Gelman and Rubin, 1992]. To remove the influence of the starting values, many MCMC software packages remove initial parts of chains, called 'warm-up', by default. Some early work in calculating appropriate warm-up lengths was performed by Schruben and others [Schruben et al., 1983, Schruben, 1983, 1982]. Today, generating samples is computationally cheaper and MCMC software solutions like RStan [Stan Development Team, 2014a] default to large warm-up lengths (half of the chain).

A second potential source of error in summaries constructed from MCMC is the fact that these summaries are based on a finite number of samples. For example, consider 100 independent draws taken from a target distribution. Although these draws are unaffected by a starting value and are not autocorrelated, having only 100 draws may still be insufficient to report the 0.975 quantile to a desired precision. Thus, even in exact or perfect sampling, summaries can be produced that do not meet the precision required [Casella et al., 2001, Propp and Wilson, 1996]. Methods to account for the precision of Monte Carlo nature of MCMC estimates have been previously discussed [Flegal et al., 2008, Geyer, 1992]. Our MCMC convergence diagnostic takes into account both whether the chains adequately approximate the target distribution and the precision of the summary because both of these qualities are fundamental in assessing convergence, though other diagnostics may only explicitly address the former.

The fundamental idea behind many existing diagnostics is to take multiple chains and evaluate, by some metric, how similar they are where greater evidence for convergence is provided by greater similarity. Chains are initiated based on draws from an overly dispersed starting distribution and as sampling proceeds, the chains come together. Thus, our default stance typically is that the chains are different and we need evidence in order to conclude the chains are similar. Evidence of similarity is atypical for much of statistical practice and, especially, null hypothesis significance testing (NHST). In NHST, a null hypothesis of equivalence is posed as the default and the strength of evidence *against* equivalence is assessed. For convergence checking, compared to NHST, we need to flip the burden of proof; we must require evidence to conclude the chains are similar. Such a statistical framework exists in the form of equivalence testing.

In this paper, we propose the Quantile Equivalence Diagnostic (QED), along with a graphical display, the Quantile Equivalence plot (QEplot). Our diagnostic and plot supplement a number of existing convergence assessment diagnostics. Existing diagnostics supply the user a difficult to interpret measure of convergence and their use is often independent of the inferential objectives of the researcher. We approach the problem by specifically addressing quantities of interest with more interpretable measures of convergence. Our diagnostic relies on equivalence testing as the tool to assess whether multiple chains have summary values that are similar enough to terminate sampling and report the summary.

3.2 Related Work in MCMC Convergence Diagnostics

There are many existing diagnostics for assessing MCMC convergence that were developed as the technique has become more widely used. Here, we review the Geweke diagnostic [Geweke, 1992], the Potential Scale Reduction Factor (PSRF) [Gelman and

Rubin, 1992], Boone's Hellinger distance [Boone et al., 2014], and the Raftery-Lewis [Raftery and Lewis, 1992] diagnostics. We have selected these diagnostics due to their popularity and relevance for our Quantile Equivalence Diagnostic. Cowles and Carlin [1996] can be consulted for a more in-depth review that, although published in 1996, covers the most commonly used diagnostics.

3.2.1 Geweke Diagnostic

Since MCMC is a Markov process, the draws of a chain are autocorrelated and draws early in the chain may be influenced by the starting value. Geweke's diagnostic aims to assess whether there is a shift between of the beginning and ending parts of a chain. The diagnostic compares the means of these parts of a chain by calculating a two sample z-score where one sample is the first n_1 draws of the chain and the other sample is the last n_2 draws of the chain. The `geweke.diag()` function in the CODA R package [Plummer et al., 2006] defaults to n_1 and n_2 being the first 10% and the last 50% of the chain, respectively.

A large z-score indicates discrepancy in the mean for the two parts of the chain which implies that at least one part does not represent a set of draws from the posterior whereas a small z-score may indicate both parts represent a set of draws from the posterior. Geweke's diagnostic could incorrectly suggest a lack of convergence if both the end part and beginning part are individually poor approximations to the posterior even if the whole chain provides a good approximation to the posterior. Diagnostics like the Potential Scale Reduction Factor (PSRF) use whole chains as the comparison unit rather than parts of chains, which can prevent high autocorrelation from falsely indicating a lack of convergence.

3.2.2 The Potential Scale Reduction Factor

The PSRF, commonly denoted as $\hat{R}$, has been published multiple times with slight changes [Brooks and Gelman, 1998, Gelman and Rubin, 1992, Gelman et al., 2003]. With multiple chains, the posterior variance of a scalar parameter can be estimated by the within-chain variances and the variance of the means of the chains (multiplied by the number of samples within a chain). By starting the chains from overly dispersed locations in the posterior space, it is expected that before convergence, the within-chain variance estimate will be less than the between-chain estimate. As sampling continues, the two variance estimates approach the same value. Seeing this in practice suggests that each of the multiple chains represents draws from distributions with the same mean and variance, thus suggesting convergence. The guiding motivation of the PSRF, that we expect the values to vary among the chains and only with evidence that the values have approached the same value do we consider convergence, is reminiscent of equivalence testing.

The recommendation of Gelman et al. [2003] is to compute $\hat{R}$ and verify that it is near 1.00 for all scalar parameters of interest. The authors suggest that values below 1.10 are acceptable for most cases and that values closer to 1.00 may be necessary for more critical problems. However, such cutoffs are arbitrary and often there is little intuition about what an acceptable value may be for specific posterior quantities of interest. Further, the authors recommend the parameter be transformed so that the target distribution is approximately normal before computing the PSRF. However, this may not be realistic for some distributions like bimodal distributions or distributions for variance parameters, and many users of the PSRF may forgo this additional step.

Brooks and Gelman [1998] note that the PSRF may be inadequate for quantities other than the posterior mean and Gelman et al. [2003] state that additional

examination of the extreme quantiles should be performed if such quantities will be reported. In our experience, it is common for the inferential objectives to include obtaining extreme quantiles to construct credible intervals. While the calculation of the PSRF is reliant on only the mean and variance of each chain, a diagnostic proposed by Boone et al. [2014] aims to use each chain's empirical distribution to more completely assess convergence.

3.2.3 Boone's Hellinger Distance

Boone et al. [2014] suggest a test for convergence using the Hellinger distance to measure discrepancies between the empirical distributions obtained from different MCMC chains, with smaller discrepancies indicating better convergence. The Hellinger distance is a symmetric distance between two probability distributions f and g, defined as

$$H(f,g) = \sqrt{\frac{1}{2}\int \left(\sqrt{f(x)} - \sqrt{g(x)}\right)^2 dx}\,.$$

The authors approximate Hellinger distances between pairs of chains via two approximations: (1) the chain densities are approximated along a grid in the posterior space via a Gaussian kernel density, and (2) the Hellinger distance integral is approximated with a summation over those points. In examples, they show the distance is not overly sensitive to these approximations. They propose several cutoff values, above which the researcher should conclude the chains have not reached convergence and continue sampling. Their proposed cutoff values are motivated by simulation studies where samples are drawn from known distributions in order to mimic MCMC chains. In practice, however, it may be difficult to choose a cutoff value for this diagnostic that has some intuitive meaning for specific inferences such as posterior quantiles.

While the comparison of entire distributions is certainly an admirable and ideal situation, it may also prove excessive depending on the inferences that are of interest to the researcher. Further, the Hellinger distance does not provide an indication of *where* two chains may deviate and therefore is not explicitly connected to the ultimate use of the target distribution for inference. If reporting posterior quantiles or probabilities is an objective, it makes sense to use a diagnostic explicitly assessing convergence of these quantities, possibly in addition to Boone's or other diagnostics.

3.2.4 Raftery-Lewis Diagnostic

One diagnostic explicitly addressing posterior quantiles and probabilities is the Raftery-Lewis diagnostic [Raftery and Lewis, 1992]. This diagnostic differs from the preceding diagnostics in a major way; it does not diagnose convergence for a completed set of chains, but rather predicts the total number of MCMC samples needed in a single chain to meet a desired precision based on a pilot run of the sampler. Specifically, the proposed number of samples is the number to ensure with probability s that $p(\theta < C|\mathbf{y})$ is found to within $\pm r$, where θ is the parameter of interest, C is a chosen value of θ and $\mathbf{y}$ is the data. This calculation is reliant on the pilot run providing a good indication of the final run. However, as Cowles and Carlin [1996] note, diagnostics based on calculating a necessary number of samples may suggest an impractical sampling length, especially for complicated models.

One could use this diagnostic to obtain a preliminary number of samples to draw and supplement with a convergence check on the final samples, perhaps even using the Raftery-Lewis diagnostic again to check if the goals were met or using a different diagnostic. We consider the specificity of the Raftery-Lewis diagnostic to quantiles and the interpretability of the diagnostic to be an advantage over the previously discussed diagnostics. The diagnostic we propose also focuses on quantiles

while incorporating multiple chains like the Hellinger distance and PSRF convergence diagnostics.

3.3 Assessment of Quantile Equivalence

3.3.1 Motivation for Summary Measures

In Bayesian data analysis, posterior distributions are often summarized with simple measures such as means, probabilities, and credible intervals. In many cases, the summary value can be expressed as the posterior expectation of some function, g, of a parameter θ, $E(g) = \int g(\theta)p(\theta|y)d\theta$ where $p(\theta|y)$ is the posterior distribution of θ. For example, the posterior mean is calculated with $g(\theta) = \theta$. When we approximate a posterior distribution with samples via MCMC, we approximate the posterior expectation of $g(\theta)$ by calculating the average value of the function over the n MCMC samples,

$$\hat{g}_n = \frac{1}{n}\sum_{i=1}^{n} g(\theta_i) \ , \tag{3.1}$$

where the notation $\hat{g}_n$ indicates it is an estimate based on n samples. Under the appropriate conditions (see for example Meyn and Tweedie [1993]), $\hat{g}_n$ converges to the true posterior value $E(g)$ as the number of MCMC samples approaches infinity.

Before we have a set of draws to calculate $\hat{g}_n$ we must choose a sampling algorithm and prepare the initial values for the m chains, though these might be done automatically depending on the software package. After the sampling process we can calculate a summary value for each chain indexed by $j = 1 \ldots m$. To differentiate among the chains, we denote the summary as $\hat{g}_n^j$ and recognize that different random seeds could result in different summary values despite identical initial values. Thus, given the sampling algorithm, chain length, and starting value, there is a distribution

of possible $\hat{g}_n^j$'s. From this distribution, we can consider the expected value of $\hat{g}_n^j$ which we denote g_n^j. This expected value converges to the true posterior value $E(g)$ as the chain length increases. High variance among the $\hat{g}_n^k$ indicates the chains have not yet come to a consensus and we may not be comfortable reporting any summaries. However, if the variability is small, we are more confident that the chains have converged and provide a reasonable approximation to the posterior, at least for our particular summary value of interest. The goal of the QED is to assess whether the g_n^j's are within some tolerance of $E(g)$ for the particular case that quantiles and tail probabilities are the summaries of interest. After introducing the specific notation for our problems of interest below, we provide the assumptions, approximations, and test to assess convergence with the QED.

3.3.2 Diagnostic

For posterior probabilities and quantiles, the statement of interest can be expressed as $p = Pr(\theta < C|y)$. For such problems, $g(\theta)$ is defined as

$$g(\theta) = \begin{cases} 1 & \text{if } \theta < C \\ 0 & \text{if } \theta \geq C \end{cases} \tag{3.2}$$

and thus, $p = E(g)$. For *probability* problems, C is given and we estimate p with Equation (3.1), which is simply the proportion of the MCMC samples less than C. For *quantile* problems, p is given and we estimate C by finding the empirical p^{th} quantile of the MCMC samples. To use the same framework as *probability* problems, we use a similar strategy to Raftery and Lewis [1992] and treat the problem as though the found C were given and p is to be estimated.

To be more explicit for quantile and probability problems, we introduce more appropriate notation, with the general notation provided earlier shown in brackets.

The true posterior value is p $[E(g)]$; the summary value from the j^{th} chain is $\hat{p}_n^j$ $[\hat{g}_n^j]$; and the expected summary value from the j^{th} chain is p_n^j $[g_n^j]$. We also introduce $\hat{p}$, the summary value from the amalgamation of all chains, where we have removed the dependence on the length of the chains from the notation.

In order to calculate variance, we need a distribution for the possible $\hat{p}_n^j$'s. As an exercise to empirically derive this sampling distribution, one may prepare a set of chains all with the same starting value, sample these chains with different random seeds, and observe the distribution of the resulting $\hat{p}_n^j$'s. However, running enough chains to approximate this distribution may not be practical. Even using resampling techniques to investigate bootstrap distributions of $\hat{p}_n^j$ for chain j may be prohibitively slow. Instead, consider a transformation of the parameter of interest θ. Let θ_i^j be the i^{th} draw from the j^{th} chain, and introduce $z_i^j = g(\theta_i^j)$ where $g(\cdot)$ is defined in Equation 3.2 so that z_i^j is a binary indicator of whether θ_i^j is less than (or greater than or equal to) C.

Such a two-state Markov chain is described by Cox and Miller who calculate the variance of the sum of n draws of a two-state Markov chain to be $n\frac{\alpha\beta(2-\alpha-\beta)}{(\alpha+\beta)^3}$ where α is the transition probability from state 0 to state 1 and β is the transition property from state 1 to state 0 [Cox and Miller, 1977]. The values for α and β can be estimated from the chain. Since the quantity we are interested in, $\hat{p}_n^k$, is the sum of n draws of a two-state Markov chain divided by n, we can use the variance above divided by n^2 as an approximation to the variance of the sampling distribution of $\hat{p}_n^k$:

$$\frac{\alpha\beta(2-\alpha-\beta)}{n(\alpha+\beta)^3} \; . \tag{3.3}$$

It is worth noting that for independent draws, which may be the case for exceptionally large thinning, α is simply the probability of a "success", β is simply

$1-\alpha$, and the variance reduces to the familiar variance of a binomial distribution. Given this variance approximation, we follow the lead of Raftery and Lewis [1992] and argue that after a large number of draws (as typical with MCMC), $\frac{1}{n}\sum_{i=1}^{n} z_i^j$ is approximately normal.

Thus, we use a normal distribution centered at p_n^j to approximate the sampling distribution of $\hat{p}_n^j$:

$$\hat{p}_n^j \overset{approx}{\sim} N\left(p_n^j, \frac{\alpha^j\beta^j(2-\alpha^j-\beta^j)}{n(\alpha^j+\beta^j)^3}\right) . \tag{3.4}$$

with α^j and β^j estimated from each chain j. If a chain is completely above or below the quantile of interest (so $z_i^j = 1$ or 0 for all i), α^j and β^j cannot be accurately estimated and we immediately conclude more sampling is necessary.

With the variance in Equation (3.4), we have a precision of $\hat{p}_n^j$ to provide an indication of how well we know p_n^j based on the observed $\hat{p}_n^j$. By initializing our chains from overly dispersed starting values, the p_n^j's may not be close to the true posterior value p. This naturally leads to assessment of evidence of equivalence, rather than the typical evidence of a difference. As the sampling continues, all p_n^j's will approach p and we can assess the evidence that the p_n^j's are close to the posterior value. By performing a hypothesis test, we can prescribe specific tolerance objectives and assess the evidence for the defined objective rather than relying on arbitrary cutoffs. We test the null hypothesis

$$H_0 : |p_n^j - p| \geq \epsilon \text{ vs } H_A : |p_n^j - p| < \epsilon \tag{3.5}$$

simultaneously for all m chains. The tolerance ϵ is discussed in Section 3.3.3. Wellek specifies the uniformly most powerful test for a one parameter equivalence test with a normal sampling distribution with known variance [Wellek, 2010, Chapter 4]. To

use Wellek's result, we make two simplifications: (1) in Equation (3.4), we use $\hat{p}$ in the variance rather than the unknown p_n^j (assuming constant variance across chains), and (2) in Equation (3.5), we replace p with $\hat{p}$ which is the quantity we intend to report as p.

With these assumptions and approximations, we use Wellek's uniformly most powerful test for the equivalence test on each chain. The α level UMP equivalence test rejects the null hypothesis when $|\hat{p}_n^j - \hat{p}|\sqrt{n(\alpha^j + \beta^j)^3/(\alpha^j\beta^j(2 - \alpha^j - \beta^j))}$ is less than the square root of the α^{th} quantile from a chi-squared distribution with a single degree of freedom and a non-centrality parameter of $\frac{n(\alpha^j+\beta^j)^3}{\alpha^j\beta^j(2-\alpha^j-\beta^j)}\epsilon^2$.

To complete the diagnostic, we combine the result of the hypothesis for each chain with the intersection-union principle [Berger and Hsu, 1996]. The intersection-union principle rejects the set of tests if *all* tests individually reject the null hypothesis. Otherwise, we fail to reject the set as a whole. We choose this strategy because we believe that we should be wary of all chains if any one chain demonstrates a lack of convergence. So, if for every chain, the null hypothesis is rejected, the QED suggests the desired precision of ϵ has been obtained and no further sampling is needed. However, if, for *any* chain, the null hypothesis is not rejected, the QED suggests the researcher continue sampling in order to report results at the specified precision.

3.3.3 Choosing the Desired Precision

Convergence as diagnosed by the QED implies that each of the p_n^j is within ϵ of $\hat{p}$. The QED only accommodates specification of tolerance on the probability scale because we are able to use a sensible approximation for the variance on that scale. On the quantile scale, the variance is likely to change for each application in unknown ways. However, there is no reason why a similar diagnostic could not be created

provided the variance can be reasonably approximated. Without an approximation and if the researcher's desired precision can only be expressed on the quantile scale, we recommend careful evaluation of the accompanying plot we discuss below.

To motivate a method of choosing a value for ϵ, we work backward assuming the QED has suggested convergence for a given ϵ. This suggests the p_n^j's are all within ϵ of p. As a worst case scenario, we can imagine half of the p_n^j's are ϵ below and half are ϵ above, giving a standard deviation of $\sigma = \sqrt{m\epsilon^2/(m-1)}$ Borrowing a multiplier of 2 from the commonly used 0.975 quantile of the normal distribution, we posit that the average of the m p_n^j's is within $b = 2\sigma/\sqrt{m} = 2\epsilon/\sqrt{m-1}$ of p. Though the p_n^j's are not known, we might use $\hat{p}$ as an approximation of their average. Thus, we suggest users specify a b within which they wish to determine p leading to an ϵ of $\epsilon = (b\sqrt{m-1})/2$. For example, if we are willing to allow $\hat{p}$ to deviate from p by as much as $b = 0.02$, we find ϵ to be $(0.02\sqrt{m-1})/2$; for $m = 4$, we have $\epsilon = (0.02\sqrt{3})/2 \approx 0.017$.

3.3.4 Graphical Display

We recommend using diagnostics to assess convergence in conjunction with graphical displays. The venerable trace plot should always be produced to investigate any characteristics that may indicate a lack of convergence of the sampler both within and across chains. Another strategy to graphically assess convergence uses linear discriminant analysis to produce two dimensional visualizations for the draws from chains sampling multiple parameters [Peltonen et al., 2009]. Mixing of the chains is observed as clustering of draws from various chains. If draws from a particular chain do not cluster with those of other chains, the researcher may question the convergence of their sampler.

If the researchers are running multiple chains and interested in probabilities or quantiles, the Quantile Equivalence plot (QEplot) may also be employed. While other MCMC diagnostic plots take an overarching approach to visualization, the QEplot provides a graphical summary specific to quantiles and probabilities and thus supplements, rather than replaces, these other diagnostic plots.

The objective of the QEplot is to display the variability of the computed quantiles and probabilities among the chains (examples in Section 3.4). To display the variability in the m $\hat{p}_n^j$'s, we plot a point for each chain (x-value of $\hat{p}_n^j$ and a y-value of C) creating a set of m points $(\hat{p}_n^j,\ C)$. These points form a horizontal band since they have the same y-value and contain the information going into the QED calculation. To gain intuition about the variability among the chains on the quantile scale, we also plot a vertical band of points where $\hat{p}$ is the x-value for all chains and the y-values are the empirical $\hat{p}^{\text{th}}$ quantile of the j^{th} chain, $\hat{C}^j$; the m points $(\hat{p},\ \hat{C}^j)$ are plotted as points. The two points, $(\hat{p}_n^j,\ C)$ and $(\hat{p},\ \hat{C}^j)$ are connected with a line to indicate these points come from the same chain. Finally, the point $(\hat{p},\ C)$, which is the intersection of the horizontal and vertical bands, is plotted for reference.

One of the objectives of the QED is to provide more intuition behind the decision criteria compared to some other common diagnostics. The QEplot adds to this by clearly displaying the variability of the summaries across the chains. The plot can be used on its own in lieu of the statistical tests at a particular α level, or simply to understand the conclusion from the QED. If the points on the QEplot are variable enough that the researcher is uncomfortable reporting the summaries, then the researcher should continue sampling the chains regardless of the conclusion of any convergence diagnostic.

3.4 Examples

Examples from Boone et al. [2014] include a number using artificial MCMC chains constructed via independent draws from known distributions to demonstrate weaknesses of the PSRF compared to their diagnostic and demonstrate the performance of their diagnostic in the context of a full data analysis on real data. We provide a different example using artificial chains and compare results of the QED to the PSRF and Boone's Hellinger distance diagnostic as well as provide an example for a familiar hierarchical Bayesian analysis used in examples for Gelman et al. [2014] and Gelman and Hill [2007].

3.4.1 Clipped Tails Example

Our first example simulates a scenario where the tails of a posterior distribution have not been adequately explored by one or more chains. We use a normal distribution as our target distribution and simulate three chains by simply drawing three sets of independent draws from the normal distribution and then manipulating them. Each chain contains 10500 draws, but we *clip* the smallest 500 from the first chain, the largest 500 from the second chain, and the smallest and largest 250 each from the third chain for a total of 10000 draws remaining in each chain. Our objective for this example is to create a scenario where the upper tail of one chain has not been explored, the lower tail for another, and both tails for a third (Figure 3.1 and Table 3.1).

A glance at the summary statistics in Table 3.1 shows that if a 95% credible interval is of interest, the intervals reported by each chain differ. For example, the 0.025 quantile varies from about -1.5 to -2.0, which is also clearly seen in the QEplot (Figure 3.2). Depending on the aims of the researcher, this difference may

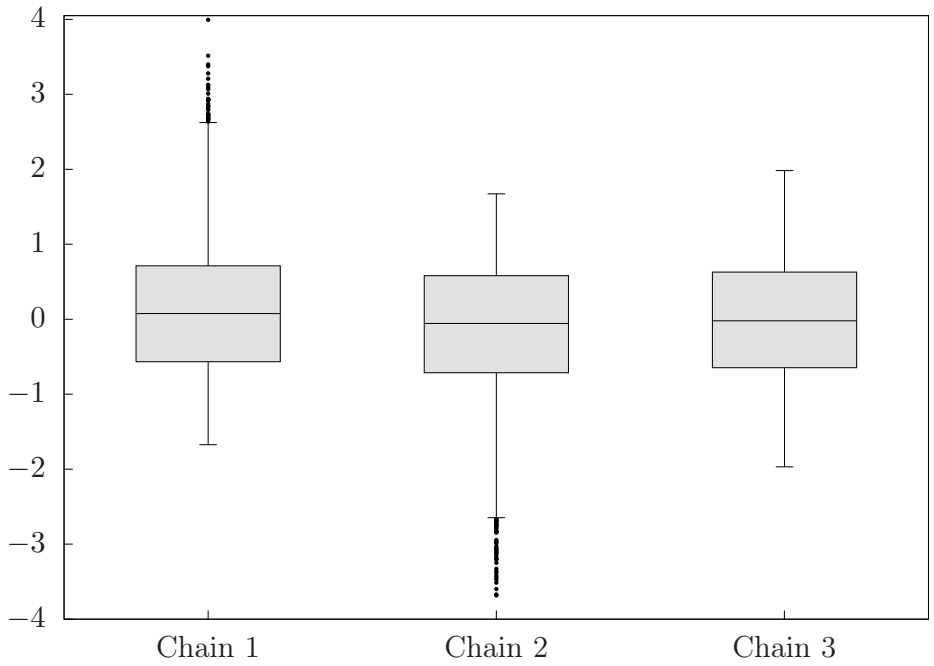

Figure 3.1: Boxplots show the distribution of samples from three hypothetical chains after removing low, high, and both low and high samples from chains 1, 2, and 3, respectively (Section 3.4.1).

Table 3.1: Summary statistics are provided for each of the three clipped normal distribution chains and also the amalgamation of the chains (Described in 3.4.1). Convergence assessment is performed for the Potential Scale Reduction Factor (PSRF), the Hellinger distance diagnostic (BMK after the authors last names), and the QED. For the QED, a 0 indicates further sampling is required; a 1 indicates that convergence to the specified tolerance has been obtained at α-level 0.05.

	2.5%	97.5%	PSRF	BMK	QED 2.5% $\epsilon = 0.01$	QED 97.5% $\epsilon = 0.01$	QED 2.5% $\epsilon = 0.05$	QED 97.5% $\epsilon = 0.05$
Chain 1	-1.48	2.00	—	—	—	—	—	—
Chain 2	-1.98	1.49	—	—	—	—	—	—
Chain 3	-1.67	1.67	—	—	—	—	—	—
All Chains	-1.68	1.68	1.04	0.18	0	0	0	0

exceed the tolerance desired and immediately suggest continued sampling. Though the third chain (with both ends clipped) is consistent with the summaries from the amalgamation of the chains, the other two chains are not and the QED will fail to suggest convergence for $\epsilon = 0.01$ and $\alpha = 0.05$ (Table 3.1). Depending on the inferences of interest and the desired precision, the summary statistics in Table 3.1, coupled with the information in Figures 3.1 and 3.2, may convince the researcher that further sampling is required.

We assess convergence for this example with several diagnostics and use the QED to assess convergence for both the 0.025 and 0.975 quantiles, testing at $\epsilon = 0.01$ (corresponds to $b \approx 0.014$) and $\epsilon = 0.05$ (corresponds to $b \approx 0.07$) both with $\alpha = 0.05$. At $\epsilon = 0.01$, the QED indicates further sampling is needed for both quantiles. Even after adjusting the precision to $b = 0.07$, the QED suggests further sampling is required. In Figure 3.2, we can see that for chain 1, there are no draws below the overall quantile, thus there are no transitions in the two-state representation which renders α and β unable to be estimated for that chain. Meanwhile, the PSRF is smaller than the 1.1 often used as a cutoff point in evaluating convergence. Boone's

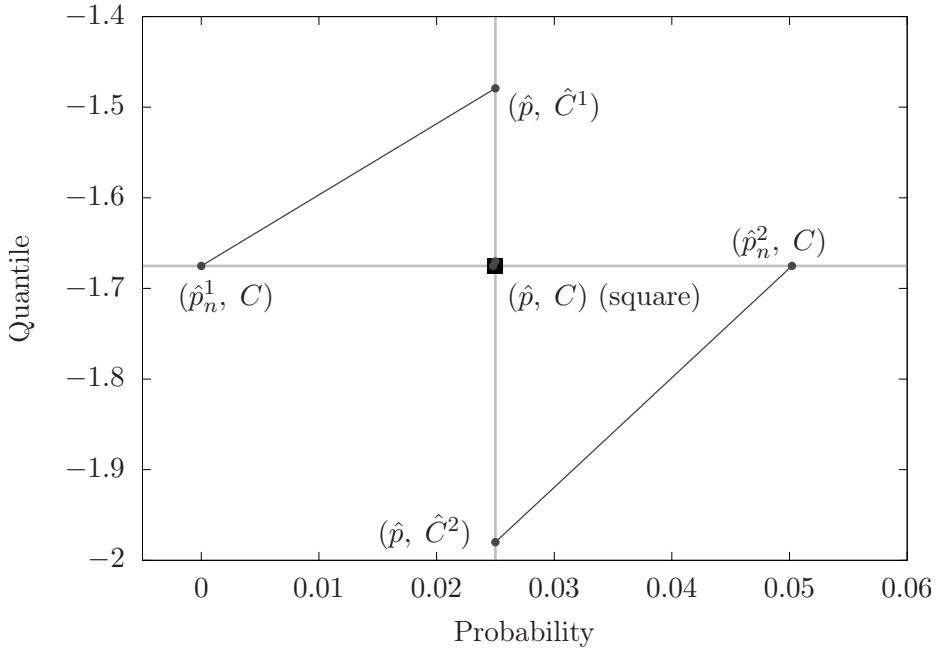

Figure 3.2: The QEplot shows empirical quantiles and probabilities associated with the 0.025 quantile of three simulated clipped chains (described in Section 3.4.1). Points for chain 3 overlap with the $(\hat{p}, C)$ and are not labeled.

Hellinger diagnostic was calculated using the BMK R package [Krachey and Boone, 2012]. The smallest Hellinger distance discrepancy among the pairs of chains is 0.10 and the largest, the value we have reported in the table, is 0.18. Boone provided a set of potential cutoff values above which indicate a lack of convergence; the largest proposed value was 0.1. The reported BMK diagnostic of 0.18 is well above this cutoff as expected with the chain distributions differing considerably as seen in Figure 3.1. This is an example where we would not like to conclude convergence for the quantiles in this scenario and the PSRF has failed to adequately assess convergence for this contrived scenario whereas the QED and BMK diagnostics perform adequately.

3.4.2 Eight Schools Example

The reader may be familiar with the *eight schools* data set used as an example in Gelman and Hill [2007] and Gelman et al. [2014]. In eight different schools, SAT

coaching was employed as a means to help students improve scores on the SAT-verbal test. In each of the schools, the estimated coaching effect on exam scores and its associated standard error are reported in the original publications, Alderman and Powers [1980] and Rubin [1981]. This dataset is also used as an example for the Stan [Stan Development Team, 2014b] computer package for Bayesian inference. The following hierarchical model was proposed in Gelman et al. [2014]. For the schools indexed $j = 1, \ldots, 8$, y_j is the estimated coaching effect at the j^{th} school, and σ_j is its standard error. The model is

$$\begin{aligned} y_j &\sim N(\theta_j, \sigma_j)\,, \\ \theta_j &= \mu + \tau\eta_j\,, \\ \eta_j &\sim N(0,1)\,, \\ p(\tau) &\propto 1I_{0,\infty}(\tau)\,, \text{ and} \\ p(\mu) &\propto 1\,, \end{aligned}$$

where θ_j is the true coaching effect at school j which is composed of an overall effect of coaching μ and a school effect decomposed into τ and η_j to improve convergence. For our example, we narrow our focus to the parameter μ which captures the mean change in SAT scores over all eight schools and consider two potential quantities of interest: a 95% credible interval for μ and the posterior probability that $\mu > 0$ which is the posterior probability that coaching is effective.

We used RStan [Stan Development Team, 2014a] to sample eight chains of length 20000 taking snapshots of the chains after 500, 1000, 2000, 4000, 10000 and 20000 samples. For each snapshot, we discard the first half of each chain as warm-up, the default in RStan, and then calculate the quantities of interest (Table 3.2) along with

the PSRF, BMK, and QED convergence diagnostics (Table 3.3). For the PSRF and BMK, a single value is reported corresponding to the μ parameter, and for the QED, a value is reported for each of the posterior summaries of interest (each endpoint of the interval and the posterior probability that $\mu > 0$).

Table 3.2: Summary values of interest are provided for each of six snapshots (Section 3.4.2). The chain length is the number of post burn-in samples.

Chain length	0.025	0.975	$Pr(\mu > 0\|y)$
250	-1.40	17.09	0.951
500	-0.87	17.39	0.957
1000	-1.48	17.58	0.953
2000	-1.80	17.58	0.947
5000	-1.83	17.71	0.947
10000	-2.22	18.17	0.943

Table 3.3: Convergence diagnostics are computed at six snapshots of the chains (Section 3.4.2). The chain length is the number of post burn-in samples and the QED results are calculated using $\alpha = 0.05$ and $\epsilon = 0.015$, where a 0 indicates additional sampling is required, and a 1 indicates convergence to the specified tolerance has been obtained at $\alpha = 0.05$.

Chain length	PSRF	BMK	QED 0.025	QED 0.975	QED >0
250	1.02	0.10	0	0	0
500	1.00	0.08	0	0	0
1000	1.00	0.06	0	0	0
2000	1.00	0.07	0	0	0
5000	1.01	0.08	0	0	0
10000	1.00	0.04	1	0	1

After just 250 post warm-up samples, the PSRF suggests researchers may stop sampling according to the cutoff of 1.1, the BMK value is at the cusp of the 0.10 cutoff, and the QED suggests continued sampling. Again, we are given pause in the suggestion of convergence from the PSRF. Even the BMK may suggest convergence prematurely when considering inference for quantiles especially when we see the range

of 0.025 quantiles that would be reported at the various chain lengths, all of which could be argued to have converged (Table 3.2).

By inspecting a QEplot, we can judge whether we are comfortable reporting the single summary quantity based on the variability in the quantity of interest across our chains. The evolution of the QEplot as the chain length increases provides insight into convergence of the chains. Focusing on the empirical quantile from each chain, Figure 3.4 gives a better indication of how the empirical chain quantiles change in location and spread as chain length increases. If all chains are in close enough agreement with the single summary value, we may have no hesitation in reporting the value, however if the summaries from different chains vary greatly, we make the decision to continue sampling before reporting results. In this example, the discrepancy of the interval endpoints in Table 3.2 could be explained by long tails where a small amount of posterior probability is spread across a wide range of parameter values. This can be seen in a QEplot where chains have similar probability values $\hat{p}_n^j$ but variable quantile values $\hat{C}^j$. How much the $\hat{C}^j$'s may vary and remain acceptable depends on the context of the research problem.

If the researcher does not find the QEplot sufficient for diagnosing convergence, the QED may be calculated for a more automatic decision similar to how the BMK and PSRF are often used in practice. Here, we have used $\epsilon = 0.015$ ($b \approx 0.01$) and $\alpha = 0.05$ for each QED in Table 3.3. We implore the researcher to consider their own requirements in choosing b or ϵ which need not be the same for all quantities of interest.

3.5 Discussion

In this paper, we motivate and develop new MCMC convergence diagnostic tools, the QED and the accompanying QEplot, built on the framework of equivalence

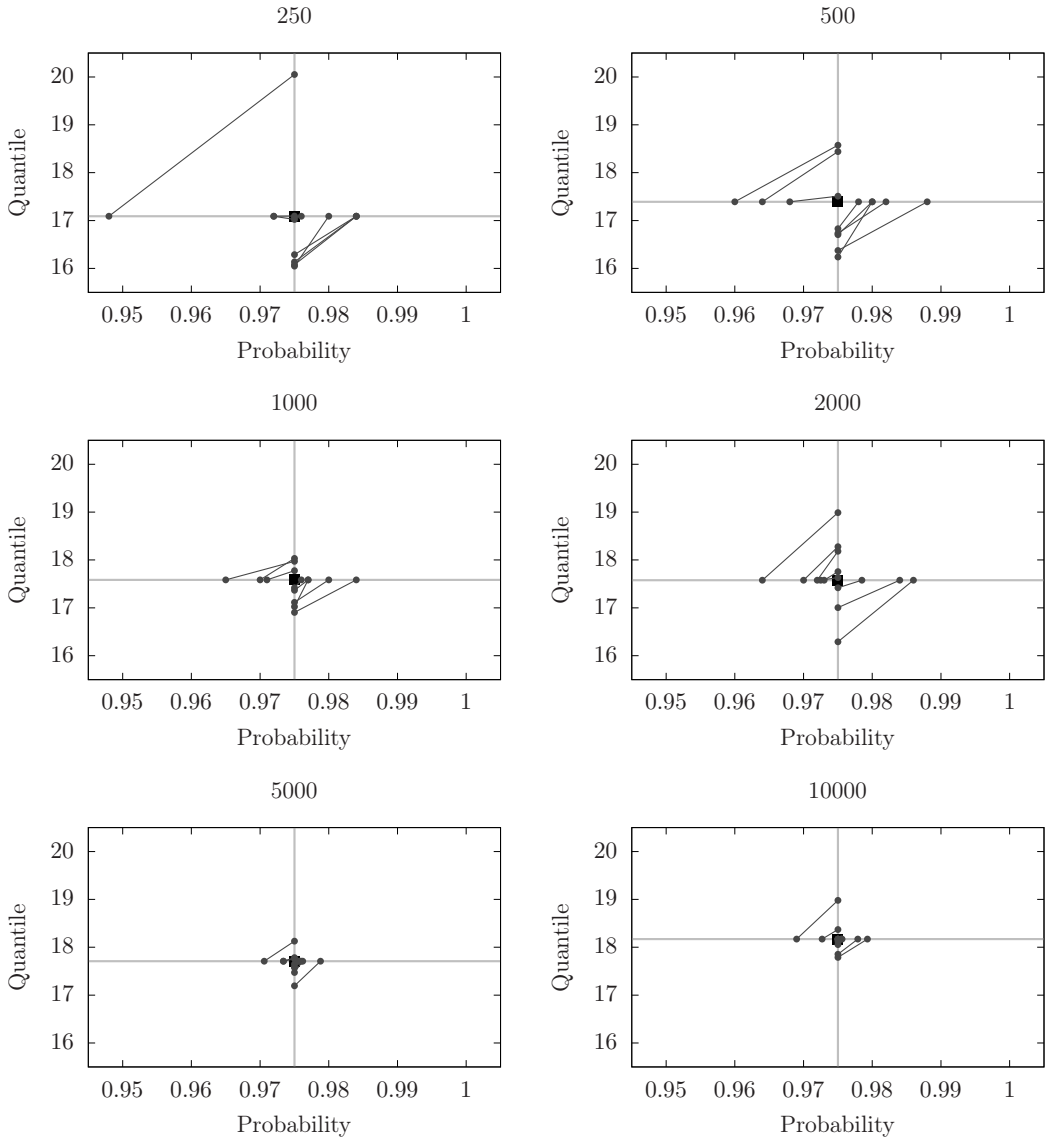

Figure 3.3: QEplots are shown for the 0.975 quantile for each of the six snapshots (described in Section 3.4.2). Panel titles indicate the number of post warm-up samples, and x- and y-ranges are held fixed across the sequence. As sampling continues, the spread of quantile and probability across chains decreases while the overall quantile estimate moves from near 17 to beyond 18. Regardless the result of a convergence diagnostic, a user may be hesitant to report a quantile for early snapshots based on the spread of values from the individual chains.

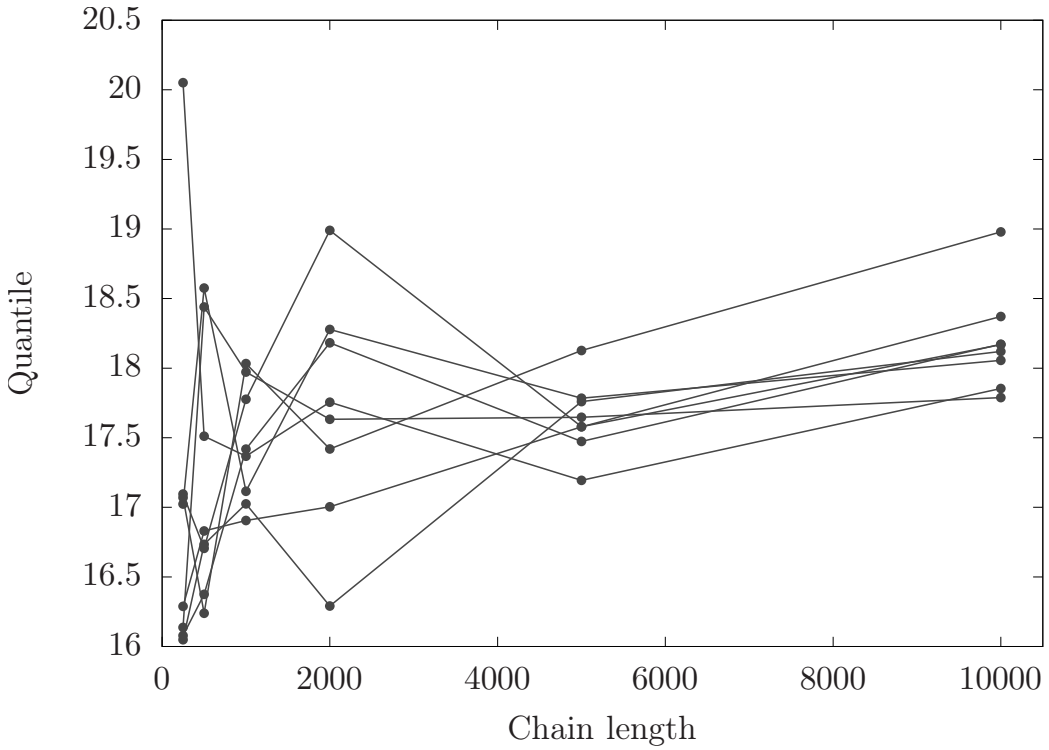

Figure 3.4: For each chain, the 0.975 quantile is plotted on the y-axis against the number of post warm-up samples for each chain. Quantiles from a single chain are connected with lines.

testing. Our introduction of equivalence testing to convergence assessment in MCMC is a new contribution to convergence checking, and we believe this is a natural connection. Reviewing existing diagnostics, we see that many are motivated by running multiple chains and checking the consistency among the chains. However, these same diagnostics tend to quantify the degree of difference between the chains. Equivalence testing allows us to explicitly assess statements of similarity compared to methods that assess differences and often with difficult to interpret measures. Even without a formal hypothesis test, the equivalence approach guides the QEplot. Our diagnostic allows researchers to assess convergence based on specified tolerances on tail probabilities, quantities that are often of interest to researchers.

In addition to introducing equivalence testing to the MCMC convergence literature, we argue for diagnostics that explicitly consider the quantities of interest. In this paper, those quantities of interest are quantiles and probabilities. The QED is not the first diagnostic to specifically address quantiles and probabilities, but this is in contrast to some of the most common diagnostics, like the PSRF. While the diagnostic we present explicitly targets probabilities and quantiles, a host of diagnostics can be produced in a very similar fashion to address any number of summaries. As we alluded to, the crux to generalizing the QED amounts to identifying an estimate of variance of the summary measure. However, lacking such an estimate of variance, a plot similar to the QEplot could still be produced. The QEplot may also be valuable for those wary of a formal hypothesis test of convergence.

A reasonable concern that may be raised against the QED is that when more chains are run, passing the intersection-union hypothesis test underlying the QED becomes more difficult. The diagnostic consists of a test for each chain and fails as soon as one chain does not pass. Hence, increasing the number of chains increases the number of opportunities the QED has to *fail* the test. Though this issue is easy to

see for the QED, other diagnostics are also dependent on the number of chains. For the reader concerned with this behavior, we argue that the researcher's objective is to create accurate and precise inferences rather than to pass a convergence diagnostic test. Also, in Section 3.3.3 and the examples, we provided a method of selecting ϵ given the overall tolerance b that scales with the number of chains. Further, we believe the reader should be reassured of this response to more chains rather than concerned. Some practitioners favor a single chain approach to MCMC arguing that a single chain of length nm is more likely to have reached convergence than m chains of length n. Though this is true, we suggest multiple chains for improved testing of convergence. Running more chains *should* lead to a more sensitive test (for a given chain length). Further, with parallel processing available on most modern personal computers, chains can often be sampled in parallel and researchers may be able to run many long chains rather than choosing between a single long chain and multiple, comparatively, short chains.

Finally, when reporting quantiles or probabilities based on MCMC techniques, the values reported are approximations of the true posterior quantiles. Though this may be an obvious observation, it may be under-appreciated. With this understanding, we built our Quantile Equivalence Diagnostic to check for consistency of quantiles or probabilities across chains which we argue indicates the values are near the true value. Between computing the diagnostic and generating the QEplots, we hope researchers will carefully consider the variability of quantile estimates and the magnitude of errors that are acceptable for a problem when reporting values obtained with MCMC techniques.

We provide examples of the use and interpretation of the QED, contrast the conclusions of the QED to the PSRF and BMK, and identify scenarios where the QED is an improvement over existing diagnostics. Catch-all convergence tests like

the PSRF and BMK may fail to diagnose issues that are of primary importance to the researcher or, similarly, may diagnose issues that are not important to the researcher. The QED specifically addresses issues related to quantiles and probabilities. If obtaining quantiles or probabilities is part of the inferential objective, we advocate the use of the QED and QEplot as important additions to the suite of tools used by researchers to decide when to stop sampling when using MCMC.

CHAPTER FOUR

DISTRIBUTING STATISTICAL ALGORITHMS AS SOFTWARE

4.1 Introduction to Scientific Software

Computing competence has become a fundamental component of scientific research. In some research labs, computing is as fundamentally important as more traditional scientific practices and experimental apparatuses. Today, scientists are not just consumers of scientific software, but they are also the writers of software. Surveys have found that scientists on average tend to devote at least 30% of their work time in developing software, and these same surveys suggest that the majority of scientists are self-taught in software development [Hannay et al., 2009, Prabhu et al., 2011].

Publishing of journal articles is the traditional means of disseminating scientific research. Now, the release of scientific software supplements traditional published articles. While I do not mean to undervalue the published article, software availability may have an even greater effect on subsequent research. The foundation of science is to build from the research of others, and using software written by another researcher may be one of the strongest examples. Likewise, as software may be directly used by others, computing errors can propagate across many studies if not discovered quickly [Merali, 2010, Wilson et al., 2014]. Therefore, it is important for researchers to employ best practices in software writing to minimize errors as well as facilitate distribution of software.

Software is now fundamental to scientific progress and dissemination, and to underestimate its role in science would be foolish. Observing this trend, a number of

researchers have published on the influence of, state of, and best practices in scientific computation [Hannay et al., 2009, Hook and Kelly, 2009, Ince et al., 2012, Merali, 2010, Prabhu et al., 2011, Wilson et al., 2014, for example]. Moreover, workshops aiming to improve software development by scientific researchers are now available [Wilson, 2006].

In one type of scientific software, the software produced is a means to an end. Software may be written to perform simulations or analyze data. Here, the end goal is typically to address a scientific question and computation is merely a tool in the process. Much like an experimentalist must carefully detail the steps in an experiment in order for another scientist to comfortably trust the results and certainly to replicate the study, the scientist using the computer to perform analysis or simulation must make the steps used on the computer sufficiently clear.

A second case of producing scientific software is where producing the software itself is the goal. It may be the case that a new algorithm or data analysis technique has been proposed and software is produced in concert to either demonstrate the use of the technique or, moreover, allow other researchers to easily use the technique. In this chapter, we pay particular focus to the latter scenario, but the tools and practices we present are pertinent in many examples of the former scenario as well.

4.2 Core Best Practices for Scientific Software Writing

To supplement this dissertation, we have produced two packages for the R statistical computing language to implement the algorithms described in the preceding chapters. Though the production of each package presents its own specific challenges, there are often best practices like those laid out by Wilson et al. [2014] or Prlić and Procter [2012] that are pertinent for nearly any scientific computing R package (we reproduce their suggestions in Tables 4.1 and 4.2). The R package to perform the

clars algorithm is available at `https://github.com/mdlerch/ccs` and the R package implementing the QED diagnostic is available at `https://github.com/mdlerch/QuantileEquivalenceMCMC`. In this section, we review several best practices common to the development of each of the two packages associated with this dissertation while specifically targeting writers of packages for R and using concrete examples whenever possible.

Wilson et al. [2014] provide a list of eight high level topics for producing scientific software. We find the list an excellent reference, but due to the large target audience of scientists writing software, the lists suffers from being very general and abstract. The list provided by Prlić and Procter [2012] is even more general and those interested in scientific software will need more specific examples to enact the principles listed. While there are certainly benefits to speaking in general terms rather than directly promoting particular software solutions and providers, most new users will need more concrete examples in order to get started with these tools and will have difficulty in deciding which tool to use.

A common refrain in writing software is to *keep it simple* [Prlić and Procter, 2012]. This principle is occasionally referred to as KISS for keep it simple, *stupid*. The principle can be applied to code writing philosophy where the authors produce code that is as basic as possible and only add features and complexity after simple, working code exists. Additionally, the KISS philosophy applies to the tools used by the software writer. We will see a common refrain below in the balance of choosing tools to facilitate code writing. In many cases, tools and packages designed to make code development easier are merely abstractions on already existing functionality. Though these tools may seem to provide an easier interface to perform a possibly complicated task, using these tools is not *simple* in the KISS sense since they are an additional layer – more abstraction between the code writer and the action to be

performed. When considering such helper tools, it is important to assess the value that these tools add. We will see concrete examples below.

Table 4.1: Best scientific coding practices from Wilson et al. [2014].

Best practices
Write programs for people not computers
Let the computer do the work
Make incremental changes
Don't repeat yourself or others
Plan for mistakes
Optimize software after it works correctly
Document design and purpose, not mechanics
Collaborate

Table 4.2: Best scientific coding practices from Prlić and Procter [2012].

Best practices
Don't reinvent the wheel
Code well
Be your own user
Be transparent
Be simple
Don't be a perfectionist
Nurture and grow your community
Promote your project

4.2.1 Code Testing

As code is being written, authors often periodically run the code to make sure things seem to be working. Obviously, this is an important part of developing code that works, but many users may not realize that this is another part of the development cycle that can be automated. Some self-taught programmers may have themselves seen the value in automating testing but may still be unaware of the resources that exist to help in the process. This code testing part of the development

cycle is called *unit testing*. The term is used to indicate that the small *units* (functions, modules, pieces of data, etc) are all performing as expected [Huizinga and Kolawa, 2007].

Writers of R packages have many options for unit testing, however, our own experience is that trying to learn about unit testing of R packages via web searching may not provide a representative overview. Web searchers may get the sense that the package RUnit [Burger et al., 2015] is antiquated and may never come across the simple but effective TestIt package [Xie, 2016]. Instead, the web is likely to point the searcher in the direction of Hadley Wickham's TestThat package [Wickham, 2011]. However, R provides package testing out of the box which many new test writers may overlook. Our own use finds the built-in testing sufficient in all respects, except perhaps in promotion.

As previously discussed, software writers are encouraged to *keep it simple* and this is especially true for writers of scientific software [Prlić and Procter, 2012]. Following the principle to keep things simple, the software author should question whether adding layers of abstraction (e.g., testing packages) on to an existing framework (R's built-in testing) provides enough additional value to justify the additional complexity. In our experience, the value added by testing packages is not enough to justify their use. In order to use R's built-in testing, we merely create a folder called `tests/` in the top level of the package and write R code in as many scripts as desired in that folder. Those scripts are run during testing to verify the scripts execute without error.

There are two main methods to perform testing in R's built-in framework. First, a test is not passed if code does not successfully execute through the duration of the test script. Of course, this does not necessarily indicate the function has run *correctly*, only that it has run to completion. Testing that the function also returns the correct

values is obviously important and we may select representative cases where a known value should result and test that the expected result is provided. This is simple to do in an R testing script and may look something like Figure 4.1.

```
# testing whether myFunction doubles the input
library(myPackage)
if(myFunction(1) != 2) { stop("myFunction(1) doesn't equal 2") }
if(myFunction(2) != 4) { stop("myFunction(2) doesn't equal 4") }
if(myFunction(3) != 6) { stop("myFunction(3) doesn't equal 6") }
if(myFunction(4) != 8) { stop("myFunction(4) doesn't equal 8") }
```

Figure 4.1: Testing a function in a package can be as simple as verifying that the output equals a prescribed value and if not, raising an error.

For many cases, testing whether a function runs without error and returns the correct value for example cases is likely sufficient. The author of the code may need to check individual components when functions return lists or other objects as output and exert creativity in creating a suite of tests, but this is not a fundamentally different kind of test.

The second style of testing for R's built-in unit testing is to write a script, run that script to collect the output as displayed in the R console, and verify that the output of that script produces the same output in future executions. This type of testing may be useful if the code author knows the code currently performs as desired and intends to do some editing and wants to make sure the code continues to run correctly. Generally, it is better to make assertions over specific values like in the preceding example. In this second strategy, it is more difficult to assess where in the code problems may have appeared. However, this form of testing may be useful if the code writer wants to verify the output is identical after updates to the code.

In order to perform this style of testing, the test script of interest is still placed in the `tests/` folder of the package. Next, the output script is created by running

R in command mode with `R CMD BATCH test_script.R`. This generates an output file called `test_script.Rout`. The output file should be placed in the `tests/` folder and named `test_script.Rout.save`. The output file should be reviewed manually to make sure the output looks as expected. Now, testing will verify that running the `test_script.R` script will produce output matching the saved file.

Now that we have reviewed how to perform testing for R packages, we should consider exactly *what* should be tested. Though the existence of tests should comfort users of the package that the code authors are testing their package, testing is explicitly used to make the development cycle easier. As such, tests should be written so that if any test fails, the code writer can quickly identify where the problem exists. For example, if a test merely says that a particular function breaks for a particular input and that function is thousands of lines of code, the author may have a difficult time placing exactly where the problem exists. In this case, more specific tests may be helpful or breaking the large function into smaller, testable functions may make the development cycle easier on the author. Generally, a test can be written before or after the code it will be testing is. Authors may write tests after code is working to prevent any regressions after future edits, or authors may write tests for planned additions to code and continue working on those additions until the tests are passed. This is referred to as test driven development. As the code is written, the source code and the test code may need to be edited in concert and after satisfying the test, the test script is kept as part of the suite of tests to prevent regressions after future edits.

4.2.2 Version Control

Especially if the authors intend to openly release or collaborate with others on the software they are producing, the authors of scientific software should use a version control system (VCS). Before discussing the details of using a VCS, consider the

venerable lab notebook. In many sciences, the lab notebook contains every action and observation performed by a researcher. In some labs, the notebook has legal ramifications. Suffice to say, the lab notebook is extremely important. It contains the history of a study, including ideas, exact protocols and procedures, observations, tweaks, and more. As we have argued that software is another apparatus in the lab bench of the scientist, there ought to exist an analogy to the scientist's lab notebook for software writing. Presumably, an actual lab notebook could be employed to record the history of a software project, but we may be able to leverage the digital nature of software to better capture the history of its development. Such functionality can be obtained with a VCS.

In its most basic use case, a VCS is used to take snapshots as digital files evolve, record metadata like notebook entries for those snapshots, and allow users to revert to or merely examine the snapshots from earlier in the development history. By taking these snapshots during the development of software and recording sufficient messages with the snapshots, the author is able to use a VCS as a notebook for the software. In slightly more advanced usage, a VCS can also improve collaboration and help the directed development of software in defined objectives. In the eight topics of Wilson et al. [2014], using a VCS can facilitate *letting the computer do the work*, *make incremental changes*, and *collaborate*. Coupled with online hosting, a VCS helps code writers *be transparent* [Prlić and Procter, 2012].

The phrase *version control system* is a general term and new users will find multiple software options from which to choose. We recommend the use of `git` as a VCS (`https://git-scm.com/`). Most of the benefits scientific software writers gain from `git` can be found with other modern VCS programs, but `git` has the benefit of being one of the most popular version control systems [Finley, 2011, Pearson, 2013].

The full power of `git` is unlocked when paired with an online hosting provider like github (`https://github.com/`). Using github simplifies collaboration, is a means of dissemination of the software, and also acts as a cloud-based back-up system. Users of R may also recognize that many popular R packages are hosted on github. For a more detailed review of `git` for scientific software, Ram [2013] can be consulted.

4.2.3 Build System

In the process of developing software, the authors will find that many processes will be repeated over and over. Processes that are repeated are prime candidates to be automated on the computer. When writing packages for R, some common procedures include building the package, installing the package, and, as we discussed earlier, testing the package. A *build system* can automate these procedures from a series of steps to a single command or perhaps even a single key press. For R users, an integrated development environment (IDE), like Rstudio (`https://www.rstudio.com/`), provides built-in tools to facilitate the process of building, installing, and testing a package. However, learning a full fledged build system like `make` (`https://www.gnu.org/software/make/`) will allow much greater flexibility and extensibility than Rstudio's clickable menu items.

We recommend the use of the software program `make` as a means for automating all sorts of tasks associated with scientific software development. However, the `make` program is not specific to scientific software. It has been used by software developers for decades. To use `make`, a file to be generated (called a *target*) is identified and other files that are required to make the target are listed as *dependencies*. The *recipe*, or series of commands, that takes the dependencies and creates the *target* is also provided. It is possible for *targets* of one recipe to be dependencies of another. These targets, recipes, and dependencies are written into a Makefile which the `make` program

then reads and interprets. For example, suppose documentation for a scientific software program is being written in LaTeX and a pdf plot must be generated from an R script that will be included as a figure in the documentation. A simple Makefile for this scenario is shown in Figure 4.2.

```
documentation.pdf: documentation.tex figure.pdf
    pdflatex documentation.tex

figure.pdf: Rscript.R
    Rscript -e "source('Rscript.R')"
```

Figure 4.2: A makefile can be used to build software or, like in this example, build documentation.

In the example Makefile, there are two targets: documentation.pdf and figure.pdf. The target documentation.pdf has two dependencies: documentation.tex and figure.pdf (itself the other target). The figure.pdf target has a single dependency of Rscript.R. Each target is created from its dependencies with a single line recipe. The power of `make` is to follow the dependency graph specified in a Makefile and build up-to-date targets. Suppose the user asks `make` to build the documentation (`make documentation.pdf`). Chasing back the dependency graph, `make` will recognize that figure.pdf depends on Rscript.R. If Rscript.R is newer than the existing version of figure.pdf (or if figure.pdf does not exist), `make` recognizes that before building documentation.pdf, it must build figure.pdf from its recipe. After making an up-to-date version of figure.pdf, `make` runs the recipe to build documentation.pdf. In this example, we free ourselves not just from the necessity of manually running multiple commands (one for each target), but more importantly from needing to remember to run both commands.

The example in Figure 4.2 is obviously very simple, but it is easy to extend the example to a scenario with many figures that quickly becomes a pain to deal with

manually. Further, figures may depend not only on a script but also on input data. Specifying the input data as a dependency will save the author the hassle of manually verifying plots are up-to-date with the latest data.

A single Makefile can contain independent targets whose dependency graphs do not overlap at all. When writing scientific software, the authors will likely also write documentation of some sort, so our example is likely to be pertinent, but we can add to the Makefile in Figure 4.2 targets and recipes to build, test, and install an R package as well. Any process that can be decomposed into targets, dependencies, and recipes is a candidate to be included in a Makefile. The more times the process will be performed in the cycle of writing the code or text, the more value is derived from using `make`.

4.3 Software for the Algorithms in This Dissertation

The preceding chapters detailed two new statistical algorithms: clars for cost-based variable selection with regularization in linear models and QED for MCMC convergence assessment. In addition to the introduction of these new algorithms, this dissertation is supplemented with software implementing those algorithms. The importance of proper scientific software is argued earlier in this chapter. In the following sections, we detail particular aspects of producing and disseminating these software packages, beginning with a review of best practices for producing scientific software.

4.4 clars

An important aspect of clars, like LARS before it, is computational efficiency [Tibshirani, 1996]. For small enough data sets, clars will run essentially instanta-

neously. However, as the number of variables and samples size increases, clars (and LARS and blars) begin to slow. Our particular challenge for clars is to improve the computational efficiency of a pure R-based solution, our simple starting point. However, as Wilson et al. [2014] describes, it is important to optimize software only after it works correctly.

Our target audience for clars are likely to be R users and as such we provide an R package. In the quest for performance, the first step is to implement the algorithm correctly in R. Predominantly R users ourselves, we are most comfortable writing the algorithm in the high-level language R and so begin in that language even though we expect the algorithm will benefit from a speedup from a lower level language like C, C++ or Fortran in the future [Wilson et al., 2014]. After obtaining working code, we can begin to diagnose what aspects of the algorithm will benefit from a more efficient language.

Generally, R users will find that for-loops can be slow and may be made quicker with a compiled language. However, just because a for-loop is part of an algorithm does not mean that a switch to C or C++ is necessary. It may be the case that for expected data input, the for-loop in R code is fast enough. Or, it may be possible to improve the R code itself to speed things up. In one class of for-loops, each successive iteration depends on the calculations of the preceding iteration. For such for-loops, a move to a compiled language may improve speed. However, R users often write for-loops where the calculation of each iteration does not depend on the preceding iteration, for example, a for-loop written to perform a calculation on each element of a vector. In these cases, it may be possible to *vectorize* the calculation which will typically be quicker and may be easier to read as well.

When improving the existing R code is not enough, R package writers may turn to compiled languages like C, C++, and Fortran. R has built-in support to

accommodate packages written in both R and in these languages. Beyond the native support, Rcpp is an R package to facilitate writing C++ code in an R package [Eddelbuettel, Eddelbuettel et al., 2011]. Since Rcpp is an additional abstraction layer on top of R's built-in support for C++ functions, it is important to assess the value added to see if adding the complexity of an abstraction layer is worthwhile. In this case, we find Rcpp to be beneficial. There are three main ways Rcpp improves the process of writing C++ for R packages. First, it cleanly handles the passing of stored data between R and C++ in both directions which can be difficult without the package. Second, packages using Rcpp compile locally, meaning a user writing a package on one operating system can safely send the package to users on different operating systems and not worry about compatibility of binaries. Lastly, Rcpp includes addons. In particular, RcppArmadillo facilitates the use of linear algebra packages in C++ [Francois et al., 2012].

4.5 QED

Implementing the algorithm for the quantile equivalence diagnostic is relatively simple. In fact, in its most basic form, it may take less than a dozen lines of R code to perform the diagnostic, and the algorithm is not particularly computationally expensive, unless a large number of posterior parameters will be tested. Despite this, there is a challenge involved in producing a package to perform the QED that is not present for the clars algorithm. That challenge is accepting data from the user on which to perform the QED. For clars, it is reasonable to ask the user to provide a vector for the response variable, a matrix for the predictor variables, and a vector for the costs of those predictor variables. However, we cannot expect such basic input for the QED.

The QED is performed on draws from MCMC, and therefore, we must consider where users will obtain those draws. It's possible the user will write their own sampler in R and produce draws stored in a matrix. However, it's also possible, and more likely, the user will use a previously developed MCMC sampling package such as RStan [Stan Development Team, 2014a] or jags [Plummer, 2003] to obtain their draws. If so, we should not expect the user to take the time, or even have the know-how, to process the output of the package to provide a simple input for the QED. Instead, we must accommodate the user and accept varied inputs that are processed in our code. As we have discussed, we believe the QED to be an improvement in at least some respects to the many existing diagnostics. This statement implies there are existing diagnostics! If implementing the QED is difficult relative to other diagnostics, we may not gain users no matter what advantage or offering the QED has over other diagnostics. Therefore, our specific challenge for the QED is to identify the MCMC samplers that our potential users are likely using and accommodate the output of those samplers as the input for the QED. This addresses the principle of Wilson et al. [2014] to design software for users and the principle of Prlić and Procter [2012] to help the community of users grow.

As discussed, the actual computation of the diagnostic is quite straightforward when the draws from the chains are easy to access. Our strategy is to convert the output from the MCMC samplers to matrices. For users writing their own samplers, providing the draws as a matrices is a reasonable request. For users of previously developed MCMC samplers, we will convert to a matrix from custom MCMC sampler types. The MCMC packages we accommodate are CODA [Plummer et al., 2006], jags [Plummer, 2003], and RStan [Stan Development Team, 2014a].

4.6 Discussion

As more scientists are spending more time writing software, computational competence needs to be expected of scientists. Publications like Prlić and Procter [2012] and Wilson et al. [2014] are important contributions to the scientific community. However, scientists just entering the computational realm may not gain much from such publications due to the general nature of the discussion. These scientists may learn they should use a version control system, and then turn to the internet to find a plethora of choices, each with its own benefits. In cases like these, it may be important to provide a concrete direction, lest the new scientific software writer become lost among the options.

Table 4.3: Review of pertinent software tools to implement best practices in writing scientific software

Purpose	Recommended tools	Other options
Version control	git	hg, svn
Build system	make	IDE built-in buttons
R package testing	R built-in testing	TestThat, TestIt, RUnit
R package computational speed	Rcpp, RcppArmadillo	C, C++, fortran

In this chapter, we have reviewed several tools for software writing that we find particularly valuable in implementing good practices like those proposed by Prlić and Procter [2012] and Wilson et al. [2014]. A review of our recommended tools to facilitate best practices is given in Table 4.3. Some of the tools and practices are pertinent for almost every scientific software project and some may not be relevant for certain projects. We have discussed computing practices with concrete examples and in the case of the latter, motivating examples based on the software we have written to supplement this dissertation.

CHAPTER FIVE

CONCLUSION AND FUTURE WORK

An easily identifiable source of cost in science is taking measurements. Researchers may measure variables in order to estimate another quantity based on a model with the intention of using that model in the future to predict an unmeasured variable. If the researchers had, at model-building time, access to a large number of measured variables to build the model, they may consider using only a subset of the variables so future uses of the model require only measurements on this subset rather than all variables. The researchers are incentivized to proceed in this manner if some variables are prohibitively expensive to measure for future uses of the model.

Another source of cost in science is computing time. Famously, Folding at Home [Pande, 2013] is an example of computational science that would be impossible without sharing the cost of computing across volunteers worldwide. The scale of Folding at Home is atypical, but many scientists who rely on computation will find cost, either literal costs to power the computer or the time it takes to wait for the completion of computation, as a burden. Many statisticians will point to the use of Markov Chain Monte Carlo (MCMC) for Bayesian inference as a common and computationally intensive task in statistics. In one sense, the user of MCMC never finishes the computation. Using MCMC for inferences relies on assessing convergence and requires users of MCMC to diagnose at what point the sampling is adequate to report summaries. To assist in this decision, users may rely on convergence diagnostics designed for MCMC convergence assessment.

In this body of research, I provide new algorithms for MCMC convergence diagnosis and for cost-considerate variable selection. In cost-considerate variable

selection, I consider the scenario where a number of variables are available for modeling but, to reduce the cost to measure future observations, only a subset might be selected for a model. In this scenario, it may be impractical to fully explore the space of potential variable combinations. Additionally, overfitting may be a danger in such a scenario with many variables at the disposal of the researcher. I build on the foundation provided in LARS and Lasso to perform cost-based variable selection in conjunction with model regularization. For MCMC convergence, I focus directly on evidence for convergence of quantities typically of interest in Bayesian statistics: posterior quantiles and probabilities. In contemplating how to assess such evidence, I choose equivalence testing as the framework for the new diagnostic to test for evidence of similarities in multiple chains, rather than making a decision based on a lack of evidence for a difference. Equivalence testing is a new addition the MCMC convergence literature.

It remains to be seen where these algorithms will stand in their respective fields, but we can offer insight into our expectations for these fields and additional work that may be undertaken relating to these algorithms. Cost-considerate variable selection is a young field, only beginning to be explored. We expect to see a variety of work in this field, approached from different perspectives with slightly different goals. Future algorithms proposed may differ greatly from clars. However, there are directions for future work building directly on clars.

The clars algorithm produces a sequence of models that, in general, range from an intercept only model to the OLS solution with all variables. By slowly introducing new variables along the sequence, model costs can be kept lower by selecting one of the models produced towards the beginning of the sequence. It is by design that models early in the sequence are cheaper and also regularized in much the same way that LARS and Lasso models early in their sequences are. Some researchers may find

the regularization overly aggressive for the early cheaper models. They may decide using a subset of the variables available provides enough regularization relative to OLS and additional coefficient shrinkage is not necessary. For these researchers, clars may be used as a means to provide an ordered sequence of models (specifically, the variables to be included in the model) to be fit with OLS. A simultaneous evaluation of model performance (perhaps measured on the prediction of an out-of-sample dataset) and model cost can then be used to choose among the OLS candidates. The fewer the number of variables to be included in the final model, the better the relative performance of this strategy will be compared to using the heavily regularized clars models.

Additionally, there may be scenarios where it is desired or necessary to include specific variables in the model. Similarly, there may be variables the researchers intend to collect that are truly deemed free of cost. In Chapter 2, we suggested that clars could not accommodate zero-cost variables. However, a slight adaptation could allow zero-cost and necessary variables. These variables can be fit to the response variable using OLS before beginning clars and the residuals can be used as the response in clars.

Our last consideration on the clars algorithm is about uncertainty estimates for predictions or coefficients. In the original Lasso paper, it was noted that obtaining uncertainty estimates of Lasso coefficients is a difficult problem, but two potential strategies were discussed [Tibshirani, 1996]. First, the bootstrap can be used either fixing the Lasso parameter or using a method to select it for each bootstrap sample. Second, a calculation to estimate the covariance matrix of the coefficients was suggested based on an approximation to ridge regression. This calculation results in an estimated variance of 0 for variables excluded from the model. Meanwhile, the Bayesian nature of the Bayesian Lasso naturally provides uncertainty for all variables,

included those with posterior mode at 0 [Park and Casella, 2008]. Recent publications like Lockhart et al. [2014] and Efron [2013] may provide additional insight into the uncertainty in Lasso regression. For clars, we recommend a bootstrap approach for uncertainty estimation, but are interested in future work to explore other methods of uncertainty estimation. A Bayesian take on cost-considerate variable selection with regularization may provide provide uncertainty estimation but a Bayesian approach may require a new approach to the problem. The researcher may also decide that OLS estimates on a subset of the variables provide enough regularization and choose to forgo any coefficient shrinkage. In this case, the researcher may consider several models produced by clars and make OLS fits using the constituent variables in those models. Under OLS, standard methods can be used to produce uncertainty estimates, but it may be pertinent to consider inflation of standard errors due to selecting among multiple models [Efron, 2013].

Unlike cost-considerate variable selection, MCMC convergence diagnostics already have an established place in statistics. Since we believe cost-considerate variable selection is a domain that will soon experience a growth in research, we hesitate to presume or necessarily argue for the wide-spread adoption use of clars. However, we find the MCMC convergence diagnostic domain to be stable enough to strongly recommend the use of our diagnostic, the QED. We understand some practitioners will hesitate to use a diagnostic based on a hypothesis test, even if based on equivalence testing, but even for these users, we strongly encourage inspection of the QEplot. The most popular diagnostics and diagnostic plots for MCMC do not focus explicitly on summaries of interest. We believe supplementing the existing diagnostics with a specific look at quantiles and probabilities will improve the diagnosis of convergence.

While we strongly recommend the use of our diagnostic, we do appreciate that its strength as a diagnostic focusing on a specific summary measure (probabilities

or quantiles) means that it may not be relevant for other summary measures, such as means and modes. The hurdle to accommodate other summaries is the estimate of variance of the summary measure, but future work may explore other methods to estimate this variance. Along with an approximate normality assumption, if a variance estimate can be provided, convergence can be assessed under the same framework as the QED. Even lacking a variance estimate, however, a plot like the QEplot can be generated for many summaries of interest. Even a one-dimensional summary measure plot is a valuable tool in assessing convergence by examining the consistency among chains.

Researchers using the tools and methods of statistical analysis carry a heavy responsibility. Results of statistical analyses form scientific theories and motivate crucial business decisions. We believe it is pertinent for the researcher to recognize that cost-benefit trade-offs are nearly unavoidable in the course of data-based decision making. Appreciating this balance may provide researchers a perspective that helps in understanding nuances of their analyses. In some circumstances, the cost-benefit trade-off is strong enough that it should be considered outright. We hope to play a role in a movement toward a greater appreciation for and library of cost-considerate approaches in statistics.

Bibliography

B P Abbott. Observation of Gravitational Waves from a Binary Black Hole Merger. 061102(February):1–16, 2016. doi: 10.1103/PhysRevLett.116.061102.

H Akaike. A new look at the statistical model identification. *IEEE Transactionson Automatic Control*, 19(6):716–723, 1974. ISSN 0018-9286. doi: 10.1109/TAC.1974.1100705.

D. L. Alderman and D. E. Powers. The Effects of Special Preparation on SAT-Verbal Scores. *American Educational Research Journal*, 17(2):239–251, 1980. ISSN 0002-8312. doi: 10.3102/00028312017002239.

Douglas G Altman and J Martin Bland. Measurement in medicine: the analysis of method comparison studies. *The statistician*, 32(July 1981):307–317, 1983.

Douglas G Altman and J Martin Bland. Statistics notes: Absence of evidence is not evidence of absence. *BMJ*, 311(7003):485, 1995.

Kyounghwa Bae and Bani K. Mallick. Gene selection using a two-level hierarchical Bayesian model. *Bioinformatics*, 20(18):3423–3430, 2004. ISSN 13674803. doi: 10.1093/bioinformatics/bth419.

Serge Bakin. Adaptive regression and model selection in data mining problems. (2), 1999.

Roger L Berger and Jason C Hsu. Bioequivalence trials, intersection-union tests and equivalence confidence sets. *Statistical Science*, 11(4):283–302, 1996.

Edward L Boone, Jason R W Merrick, and Matthew J Krachey. A Hellinger distance approach to MCMC diagnostics. *Journal of Statistical Computation and Simulation*, 84(4):833–849, apr 2014. ISSN 0094-9655. doi: 10.1080/00949655.2012.729588.

George E. P. Box. Science and Statistics. *Journal of the American Statistical Association*, 71(356):791–799, 1976. ISSN 13514180. doi: 10.1007/s13398-014-0173-7.2.

N K Boyland, R James, D T Mlynski, J R Madden, and D P Croft. Spatial proximity loggers for recording animal social networks : consequences of inter-logger variation in performance. 67:1877–1890, 2013. doi: 10.1007/s00265-013-1622-6.

RL Branham Jr. Alternatives to least squares. *The Astronomical Journal*, 87:928–937, 1982.

Leo Breiman. Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384, 1995. ISSN 00401706. doi: 10.1080/00401706.1995.10484371. URL http://amstat.tandfonline.com/doi/abs/10.1080/00401706.1995.10484371.

Leo Breiman. Statistical Modeling: The Two Cultures. *Statistical Science*, 16(3): 199–215, 2001. ISSN 08834237. doi: 10.2307/2676681. URL http://www.jstor.org/stable/2676681.

Stephen P Brooks and Andrew Gelman. General methods for monitoring convergence of iterative simulations. *Journal of computational and graphical statistics*, 7(4), 1998.

Stephen P Brooks and Gareth O Roberts. Convergence assessment techniques for Markov chain Monte Carlo. *Statistics and Computing*, 8(4):319–335, 1998. ISSN 0960-3174, 1573-1375. doi: 10.1023/A:1008820505350.

PJ Brown, T Fearn, and M Vannucci. The choice of variables in multivariate regression: A non-conjugate Bayesian decision theory approach. pages 635–648, 1999. ISSN 0006-3444. doi: 10.1093/biomet/86.3.635. URL http://discovery.ucl.ac.uk/151970/.

Matthias Burger, Klaus Juenemann, and Thomas Koenig. *RUnit: R Unit Test Framework*, 2015. URL https://cran.r-project.org/package=RUnit.

Kenneth P Burnham and David R Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer, 2002.

George Casella, Michael Lavine, and Christian P Robert. Explaining the Perfect Sampler. *The American Statistician*, 55(4):299–305, nov 2001. ISSN 0003-1305. doi: 10.1198/000313001753272240.

Mary K Cowles and Bradley P Carlin. Markov chain Monte Carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434):883–904, 1996.

David Roxbee Cox and Hilton David Miller. *The theory of stochastic processes*, volume 134. CRC Press, 1977.

Dirk Eddelbuettel. *Seamless R and C++ intgration with rcpp*. ISBN 9781461468677.

Dirk Eddelbuettel, Romain François, J Allaire, John Chambers, Douglas Bates, and Kevin Ushey. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011.

Bradley Efron. Estimation and accuracy after model selection. *Journal of the American Statistical Association*, pages 1–26, 2013. URL http://www.tandfonline.com/doi/abs/10.1080/01621459.2013.823775.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004. ISSN 1095-9572. doi: 10.1214/009053604000000067. URL http://projecteuclid.org/euclid.aos/1083178935http://projecteuclid.org/DPubS?service=UI{&}version=1.0{&}verb=Display{&}handle=euclid.aos/1083178935$\delimiter"026E30F$nhttp://projecteuclid.org/DPubS?verb=Display{&}version=1.0{&}service=UI{&}handle=euclid.aos/1083178935{&}page=record$\delimiter"026E30F$nht.

Mário a T Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1150–1159, 2003. ISSN 01628828. doi: 10.1109/TPAMI.2003.1227989.

Klint Finley. *Github has surpassed SourceForge and Google Code in Popularity*, 2011. URL http://readwrite.com/2011/06/02/github-has-passed-sourceforge/.

James M. Flegal, Murali Haran, and Galin L. Jones. Markov Chain Monte Carlo: Can We Trust the Third Significant Figure? *Statistical Science*, 23(2):250–260, may 2008. ISSN 0883-4237. doi: 10.1214/08-STS257.

Romain Francois, Dirk Eddelbuettel, and Doug Bates. RcppArmadillo: Rcpp integration for Armadillo templated linear algebra library. *R package version 0.3*, 6, 2012.

Andrew Gelman and Jennifer Hill. *Data analysis using regression and multilevel/hierarchical models*, volume 1. Cambridge University Press New York, 2007.

Andrew Gelman and Donald B Rubin. Inference from iterative simulation using multiple sequences. *Statistical science*, 7(4):457–472, 1992.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. CRC press, 2003.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*, volume 3. Taylor & Francis, 2014.

John Geweke. Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In *Bayesian Statistics 4*, pages 169–193. 1992. URL http://www.mpls.frb.org/research/SR/SR148.pdf.

Cj Geyer. Practical Markov Chain Monte Markov. *Statistical Science*, 7(4):473–483, 1992.

Jo Erskine Hannay, Carolyn Macleod, and Janice Singer. How Do Scientists Develop and Use Scienti c Software? *American Scientist*, pages 1–8, 2009.

Trevor Hastie and Brad Efron. *lars: Least Angle Regression, Lasso and Forward Stagewise*, 2013. URL `https://cran.r-project.org/package=lars`.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC Press, 2015.

W K Hastings. {Monte Carlo} sampling methods using {Markov} chain and their applications, 1970. ISSN 0006-3444.

H He, H Daumé Iii, and J Eisner. Cost-sensitive Dynamic Feature Selection. *ICML Workshop on Inferning*, (Icml), 2012. URL `http://www.umiacs.umd.edu/{~}hhe/dynafea{_}slides.pdf`.

Roger W Hoerl, John H Schuenemeyer, and Arthur E Hoerl. A Simulation of Biased Estimation and Subset Selection Regression Techniques. *Technometrics*, 28(4): 369–380, 1986. ISSN 00401706. doi: 10.2307/1268986.

Daniel Hook and Diane Kelly. Testing for (Code) Trustworthiness in Scientific Software. pages 1–31, 2009.

John Horgan. *Is the gravitational-wave claim true? And was it worth the cost?*, 2016. URL `http://blogs.scientificamerican.com/cross-check/is-the-gravitational-wave-claim-true-and-was-it-worth-the-cost/`.

Dorota Huizinga and Adam Kolawa. *Automated defect prevention: best practices in software management*. John Wiley & Sons, 2007.

Darrel C Ince, Leslie Hatton, and John Graham-Cumming. The case for open computer programs. *Nature*, 482(7386):485–8, 2012. ISSN 1476-4687. doi: 10.1038/nature10836. URL `http://www.ncbi.nlm.nih.gov/pubmed/22358837`.

Matthew Krachey and Edward L Boone. bmk: MCMC diagnostics package, 2012.

Jens Krause, Stefan Krause, Robert Arlinghaus, Ioannis Psorakis, Stephen Roberts, and Christian Rutz. Reality mining of animal social systems. *Trends in Ecology & Evolution*, 28(9):541–551, 2013. ISSN 0169-5347. doi: 10.1016/j.tree.2013.06.002. URL `http://dx.doi.org/10.1016/j.tree.2013.06.002`.

Chenlei Leng, Minh-Ngoc Tran, and David Nott. Bayesian adaptive Lasso. *Annals of the Institute of Statistical Mathematics*, (July 2012), sep 2013. ISSN 0020-3157. doi: 10.1007/s10463-013-0429-6. URL `http://link.springer.com/10.1007/s10463-013-0429-6`.

D V Lindley. The choice of variables in multiple regression. *Journal of the Royal Statistical Society. Series B (Methodological)*, 30(1):31–66, 1968. ISSN 00359246. URL http://www.jstor.org/stable/2984458.

Dennis V Lindley. *Understanding uncertainty*. John Wiley & Sons, 2006.

Jun S Liu. *Monte Carlo Strategies in Scientific Computing*. Springer, 2001. ISBN 0387952306.

Richard Lockhart, Jonathan Taylor, Ryan J Tibshirani, and Robert Tibshirani. A significance test for the lasso. *Annals of statistics*, 42(2):413, 2014.

Sharon Lohr. *Sampling: design and analysis*. Cengage Learning, 2009.

David J Lunn, Andrew Thomas, Nicky Best, and David Spiegelhalter. WinBUGS A Bayesian modelling framework: Concepts, structure, and extensibility. *Statistics and Computing*, 10:325–337, 2000. ISSN 09603174. doi: 10.1023/A:1008929526011.

Zeeya Merali. Error: Why scientific programming does not compute. *Nature*, 467 (7317):775–777, 2010. ISSN 0028-0836. doi: 10.1038/467775a. URL http://dx.doi.org/10.1038/467775a.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *Journal Chemical Physics*, 21(6):1087–1092, 1953. ISSN 00219606. doi: http://dx.doi.org/10.1063/1.1699114. URL http://jcp.aip.org/resource/1/jcpsa6/v21/i6/p1087{_}s1?bypassSSO=1.

SP Meyn and RL Tweedie. *Markov chains and stochastic stability*. Springer-Verlag, London, 1993.

Andrew Y Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

Pavel Paclík, Robert Duin, Geert van Kempen, and Reinhard Kohlus. On feature selection with measurement cost and grouped features. *Structural, Syntactic, and ...*, pages 461–469, 2002. URL http://link.springer.com/chapter/10.1007/3-540-70659-3{_}48.

Vijay Pande. *Folding@home*, 2013. URL http://folding.stanford.edu/.

Trevor Park and George Casella. The Bayesian Lasso. *Journal of the American Statistical Association*, 103(482):681–686, jun 2008. ISSN 0162-1459. doi: 10.1198/016214508000000337. URL http://www.tandfonline.com/doi/abs/10.1198/016214508000000337.

Dan Pearson. *Github sees 3 millionth member account*, 2013. URL http://www.gamesindustry.biz/articles/2013-01-17-Github-sees-3-millionth-member-account.

Jaakko Peltonen, Jarkko Venna, and Samuel Kaski. Visualizations for assessing convergence and mixing of Markov chain Monte Carlo simulations. *Computational Statistics and Data Analysis*, 53(12):4453–4470, 2009. ISSN 01679473. doi: 10.1016/j.csda.2009.07.001. URL http://dx.doi.org/10.1016/j.csda.2009.07.001.

Martyn Plummer. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, 2003.

Martyn Plummer, Nicky Best, Mary K Cowles, and Karen Vines. CODA: Convergence Diagnosis and Output Analysis for MCMC. *R News*, 6(1):7–11, 2006.

Prakash Prabhu, Thomas Jablin, Arun Raman, Yun Zhang, Jialu Huang, Hanjun Kim, Nick Johnson, Feng Liu, Soumyadeep Ghosh, Stephen Beard, Taewook Oh, Matthew Zoufaly, David Walker, and David August. A survey of the practice of computational science. *State of the Practice Reports (SC '11)*, pages 19:1—-19:12, 2011. doi: 10.1145/2063348.2063374. URL http://dl.acm.org/citation.cfm?id=2063374.

Andreas Prlić and James B. Procter. Ten Simple Rules for the Open Development of Scientific Software. *PLoS Computational Biology*, 8(12):8–10, 2012. ISSN 1553734X. doi: 10.1371/journal.pcbi.1002802.

J G Propp and D B Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures & Algorithms*, 9:223–252, 1996. ISSN 1042-9832. doi: 10.1002/(SICI)1098-2418(199608/09)9:1/2⟨223::AID-RSA14⟩3.0.CO;2-O.

Adrian E Raftery and Steven M Lewis. How many iterations in the Gibbs sampler. *Bayesian statistics*, 4(2):763–773, 1992.

Karthik Ram. Git can facilitate greater reproducibility and increased transparency in science. *Source code for biology and medicine*, 8(1):7, 2013. ISSN 1751-0473. doi: 10.1186/1751-0473-8-7. URL http://www.scfbm.org/content/8/1/7.

Scott J Richter and Carri Richter. A Method for Determining Equivalence in Industrial Applications. *Quality Engineering*, 14(3):375–380, mar 2002. ISSN 0898-2112. doi: 10.1081/QEN-120001876.

Christian P Robert and George Casella. *Monte Carlo statistical methods*. Springer, 1999.

Christian P Robert and George Casella. A Short History of Markov Chain Monte Carlo: Subjective Recollections from Incomplete Data. *Statistical Science*, 26(1): 102–115, feb 2011. ISSN 0883-4237. doi: 10.1214/10-STS351. URL `http://projecteuclid.org/euclid.ss/1307626568`.

Andrew P. Robinson and Robert E. Froese. Model validation using equivalence tests. *Ecological Modelling*, 176(3-4):349–358, 2004. ISSN 03043800. doi: 10.1016/j.ecolmodel.2004.01.013. URL `http://linkinghub.elsevier.com/retrieve/pii/S0304380004000985`.

Donald B Rubin. Estimation in Parallel Randomized Experiments. *Journal of Educational and Behavioral Statistics*, 6(4):377–401, 1981. ISSN 1076-9986. doi: 10.3102/10769986006004377.

L. Schruben, H. Singh, and L. Tierney. Optimal Tests for Initialization Bias in Simulation Output. *Operations Research*, 31(6):1167–1178, 1983. ISSN 0030-364X. doi: 10.1287/opre.31.6.1167.

Lee Schruben. Confidence Interval Estimation Using Standardized Time Series. *Operations Research*, 31(6):1090–1108, 1983. ISSN 0030-364X. doi: 10.1287/opre.31.6.1090.

Lee W Schruben. Detecting Initialization Bias in Simulation Output. *Operations Research*, 30(3):569–590, 1982. ISSN 0030-364X. doi: 10.1287/opre.30.3.569.

Victor S. Sheng, Foster Provost, and Panagiotis G. Ipeirotis. Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, page 614, 2008. doi: 10.1145/1401890.1401965. URL `http://dl.acm.org/citation.cfm?id=1401965$\delimiter"026E30F$nhttp://dl.acm.org/citation.cfm?doid=1401890.1401965`.

Stan Development Team. RStan: the R interface to Stan, Version 2.5.0, 2014a.

Stan Development Team. Stan: A C++ Library for Probability and Sampling, Version 2.5.0, 2014b.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996. URL `http://www.jstor.org/stable/10.2307/2346178http://www.jstor.org/stable/2346178`.

Robert Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3): 273–282, 2011. ISSN 0035-9246.

P D Turney. Cost-Sensitive Classification: Empirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm. *Journal of Artificial Intelligence Research*, 2: 369–409, 1995. doi: 10.1613/jair.120. URL http://cogprints.org/1805/.

Peter D. Turney. Types of Cost in Inductive Concept Learning. page 7, 2000. URL http://arxiv.org/abs/cs/0212034.

Gary M. Weiss and Foster Provost. Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, 19:315–354, 2003. ISSN 10769757. doi: 10.1613/jair.1199.

Yael Weiss, Yuval Elovici, and Lior Rokach. The CASH algorithm-cost-sensitive attribute selection using histograms. *Information Sciences*, 222:247–268, 2013. ISSN 00200255. doi: 10.1016/j.ins.2011.01.035. URL http://dx.doi.org/10.1016/j.ins.2011.01.035.

Stefan Wellek. *Testing statistical hypotheses of equivalence and noninferiority*. CRC Press, 2010.

Hadley Wickham. testthat: Get Started with Testing. *The R Journal*, 3:5–10, 2011. URL http://journal.r-project.org/archive/2011-1/RJournal{_}2011-1{_}Wickham.pdf.

Wikipedia. All models are wrong, 2016. URL https://en.wikipedia.org/wiki/All{_}models{_}are{_}wrong.

Greg Wilson. Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive. *Computing in Science & Engineering*, 2006.

Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H D Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best Practices for Scientific Computing. *PLoS Biology*, 12(1), 2014. ISSN 15449173. doi: 10.1371/journal.pbio.1001745.

J Windeler and H J Trampisch. Recommendations concerning studies on therapeutic equivalence. *Drug Information Journal*, 30:195–200, 1996.

Yihui Xie. *testit: A Simple Package for Testing R Packages*, 2016. URL https://cran.r-project.org/package=testit.

Ming Yuan and Yi Lin. Efficient Empirical Bayes Variable Selection and Estimation in Linear Models. *Journal of the American Statistical Association*, 100(472):1215–1225, dec 2005. ISSN 0162-1459. doi: 10.1198/016214505000000367. URL http://www.tandfonline.com/doi/abs/10.1198/016214505000000367.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Li Hua Yue. *Cost-efficient Variable Selection Using Branching LARS*. PhD thesis, University of Western Ontario, 2010. URL `http://ir.lib.uwo.ca/etd/43/`.

Hui Zou. The Adaptive Lasso and Its Oracle Properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006. ISSN 0162-1459. doi: 10.1198/016214506000000735.