Imperial College London

Bootstrapping and its Applications to Credit Scoring

Author
Luke Kanczes

Supervisor
Dr. Tony Bellotti

This is entirely my own work unless otherwise stated.

June 12, 2012

Abstract

The bootstrap method provides a way to measure the accuracy of a sample statistic. For example, it allows us to estimate the variance of a particular statistic such as the mean. The specific aim of this paper is to demonstrate some ways in which bootstrapping can be applied to credit scoring. We present the bootstrap method as a way to compute confidence intervals on the difference between score cards, citing this as an alternative to the Mann-Whitney U Statistic. We also demonstrate how it might be useful in the case of a limited data set, before considering default rates within a homogeneous population. We compare block bootstrapping methods with bootstrapping residuals within the context of a first-order autoregressive model and show how we might bootstrap the Kaplan Meier estimate and compare the standard error result with that obtained by using Greenwood's formula.

Acknowledgments

I would like to thank Dr Tony Bellotti for his help over the course of the year. This project would not have been possible without his fantastic help and guidance. I would also like to thank Abbas Asaria and Neil Watt for their help with proof-reading. Finally, I would like to thank the credit card company which provided the data set.

Contents

Chapter 1

Introduction

1.1 Background Explanation

"Statistics is the science of learning from experience, especially experience that arrives a little bit at a time" [1, p.1]. The term bootstrap derives from the phrase "to pull oneself up by one's bootstrap" from the Adventures of Baron Munchausen by Rudolf Erich Raspe, "The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps" [1, p.5]. When you pull yourself up by your own bootstraps, you succeed on your own, despite limited resources [2, p.709].

Bootstrapping is a computer-based method for assigning an accuracy measure to statistical estimates [1, p.10]. They allow us to estimate sampling distributions and their characteristics, in particular, it enables us to estimate the variance of a statistic and therefore allows us to construct confidence intervals and hypothesis tests. We often apply them when the distribution of the population is unknown or when we have limited data sampled from said population.

Credit scoring provides a systematic means by which banks and other financial institutions gather information about borrowers or applicants for loans regarding their creditworthiness. In credit scoring we normally deal with large data sets, with a relatively small number of defaults, taken from an unknown distribution.

Much work has been done on building consumer credit score cards. However, little work has been done on how to apply the bootstrap algorithm in the context of credit scoring with the particular aim of comparing score cards. Further, what has not been researched in any particular scope, is whether it is possible to apply time dependent block bootstrapping techniques to model consumer credit default rates.

We shall begin by introducing the methods used to build credit score cards and how we might determine the performance of a particular score card. We demonstrate how bootstrapping can be used when a data set is of a limited size and then describe how we might apply the bootstrap methodology to place confidence intervals on the relative difference between score cards. Finally we consider time indexed data and demonstrate how macroeconomic pressures are likely to have an effect on the relative credit worthiness of an individual, before critically analysing time dependent, block bootstrapping techniques and elements of survival analysis.

1.2 Building a score card

Logistic regression is the most commonly used method for building scorecards [7, p.79]. Throughout this paper we shall suppose we have an overall data set of N points. We suppose our data set is of the form $(\mathbf{q}^i, r^i)$ for $i = 1, 2, ..., N$, where $\mathbf{q}^i = (q_1^i, q_2^i, ..., q_m^i)$ are the characteristics of each borrower (e.g. income, age etc.) and

$$r^i = \begin{cases} 1 & \text{if good} \\ 0 & \text{if bad} \end{cases}$$

Here, good corresponds to a borrower who does not default and bad corresponds to a borrower who defaults. This is divided into two independent data sets, which we refer to as *training* and *test*. For

the purposes of our introductory explanation we suppose the method used to divide these data sets is of limited importance. We build our model on the training data set and assess its effectiveness using the test data set. If we suppose our training data set is of size l and our test data set is of size n, then N is such that $N = l + n$. Without loss of generality, the data sets are relabelled such that our training set is of the form $(\mathbf{q}^j, r^j)$ for $j = 1, 2, ..., l$ and our test data set is of the form $(\mathbf{q}^k, r^k)$ for $k = l+1, l+2, ..., l+n$.

Dealing just with our training data set, we define $\Pr\{\text{good}|\text{data } \mathbf{q}\} = p(G|\mathbf{q}) = p(\mathbf{q})$ and $\Pr\{\text{bad}|\text{data } \mathbf{q}\} = p(B|\mathbf{q})$, such that $p(G|\mathbf{q}) + p(B|\mathbf{q}) = 1$. The log odds score, $score(\mathbf{q})$, is then defined such that [7, p.79]

$$score(\mathbf{q}) = \log\left(\frac{p(G|\mathbf{q})}{p(B|\mathbf{q})}\right) = \log\left(\frac{p(\mathbf{q})}{1-p(\mathbf{q})}\right) = \mathbf{c}\cdot\mathbf{q}$$

$$\Rightarrow p(\mathbf{q}) = \frac{e^{score(\mathbf{q})}}{1+e^{score(\mathbf{q})}} = \frac{e^{c\cdot\mathbf{q}}}{1+e^{c\cdot\mathbf{q}}}$$

Maximum likelihood estimation is then used to find estimates $\hat{\mathbf{c}}$ of the parameters $\mathbf{c}$, where we define

$$L(\mathbf{c}) = \prod_{j=1}^{l}\left(\frac{e^{\mathbf{c}\cdot\mathbf{q}^j}}{1+e^{\mathbf{c}\cdot\mathbf{q}^j}}\right)^{r^j}\left(\frac{1}{1+e^{\mathbf{c}\cdot\mathbf{q}^j}}\right)^{(1-r^j)}$$

Which is equivalent to maximising

$$\log(L(\mathbf{c})) = \sum_{j=1}^{l} r^j \log\left(\frac{e^{\mathbf{c}\cdot\mathbf{q}^j}}{1+e^{\mathbf{c}\cdot\mathbf{q}^j}}\right) + \sum_{j=1}^{l}(1-r^j)\left(\frac{1}{1+e^{\mathbf{c}\cdot\mathbf{q}^j}}\right) \tag{1.1}$$

Differentiating (1.1) and setting its derivatives equal to 0, implies that maximisation occurs when

$$\sum_{j=1}^{l}\left(r^j - \left(\frac{e^{\mathbf{c}\cdot\mathbf{q}^j}}{1+e^{\mathbf{c}\cdot\mathbf{q}^j}}\right)\right) = 0$$

and for $i = 1, ..., m$

$$\sum_{j=1}^{l} q_i^j\left(r^j - \left(\frac{e^{\mathbf{c}\cdot\mathbf{q}^j}}{1+e^{\mathbf{c}\cdot\mathbf{q}^j}}\right)\right) = 0$$

These are solved numerically, allowing us to generate a general linear model. For subsequent chapters we shall denote our test data set as $\mathbf{x} = (x_1, ..., x_n)$ where x_i corresponds to $(\mathbf{q}^{l+i}, r^{l+i})$.

1.2.1 Assessing the quality of a scorecard

Having built a scorecard we might now wish to assess its quality. The most widely used discrimination measures are generated from the Receiver Operating Characteristic (ROC) curve [7, p.101]. The ROC curve originated from estimating the errors in transmitting and receiving messages, however can be applied in the context of credit scoring. A cut-off value c, provides a simple decision rule to classify people as potential defaulters and potential non-defaulters. Any individual with a credit score lower than c are treated as defaulters (bad) while all individuals with a score higher than c are treated as non-defaulters (good) [8, p.146]. Under this decision rule, four scenarios can occur, which are summarised in Table 1.1. We refer to this as a *Confusion Matrix*.

	Actual non-default	Actual default
Predicted non-default	Specificity	Type II error
Predicted default	Type I error	Sensitivity

Table 1.1: Confusion Matrix

The ROC curve shows the trade-off between the true positive rate and the true negative rate. More, precisely it is a plot of the sensitivity, defined as $F(s|G) = \Pr\{\text{score} \leq s|G\}$, versus 1-specificity, defined as $F(s|B) = \Pr\{\text{score} \leq s|B\}$ over all possible values of s [6, p745].

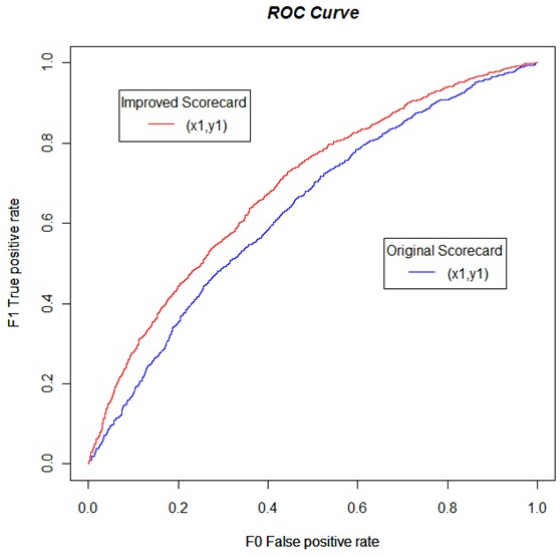

Figure 1.1: ROC Curves for two different score cards

A score card with perfect discrimination would be such that all the bad individuals have scores less than c and all the good individuals have scores above c. In this case $F(c|G) = 0$ and $F(c|B) = 1$ and our curve would consist of a vertical line from $(0, 0)$ to $(0, 1)$ and a horizontal line from $(0, 1)$ to $(1, 1)$. By comparison a score card with no discriminatory power would be such that $F(c|G) = F(c|B)$ and would consist of a diagonal line from $(0, 0)$ to $(1, 1)$.

Intuitively therefore, the closer the curve is to the point $(0, 1)$, the better the discriminatory power of the scorecard. We formalise this intuition by considering the area under the ROC curve, (referred to as AUROC or AUC), where the larger this value the "better" the discrimination is.

Mathematically, we define AUC as [7, p.117]

$$\text{AUC} = \int F(s|B)dF(s|G) = \int F(s|B)f(s|G)ds$$

Since AUC takes values between $1/2$ and 1, we could instead use the Gini coefficient as a measure of the discrimination, where we define $\text{Gini} = 2 \cdot \text{AUC} - 1$. The Gini coefficient therefore takes values between 0 and 1, where 0 corresponds to a model which is no better than random and 1 corresponds to a model which has perfect discriminatory power [7, p.117].

However, we should note that lenders are normally anxious to accept a large number of people for credit and therefore cut-off scores are normally taken from the lower end of the graph [7, p.125]. As a result there is a case to be made that the AUC places too much emphasis on larger scores than might be considered optimal. In spite of this, AUC remains the most widely used measure of the discrimination of a score card, and thus will form the basis of our study.

1.3 Comparing score cards

In Figure 1.1 we can see two different ROC curves. The left uppermost curve has a higher true positive rate for all cut-off values c. Therefore, we might conclude that this score card is better than the original score card. However, to fully justify rejecting one score card in favour of another we may wish to place confidence intervals on the AUCs generated for each score card. Conventional techniques to place confidence internals on a statistic require us to calculate the standard error. However, there exists no closed explicit method to estimate the standard error for the performance of a general linear model. It is with the aim of comparing the performance of score cards that we study the bootstrap methodology.

1.4 Time dependent data

Having explored how bootstrap methods can be used to directly compare score cards, we then wish to add an extra dimension to our analysis by considering time dependent data. It might be considered intuitively obvious that consumer credit data has a time dependent element. During a recession, macroeconomic pressures mean that it is likely that default rates are higher than during a high growth period.

A consumer credit card data set, consisting of 4635 individuals, will form the basis of our research. This data set is time indexed. If we consider the month the account was opened as a time stamp, where month one corresponds to January 2008, we can see in Figure 1.2 that there were low default rates for accounts opened in the earlier months, with spikes in the default rate for accounts that were opened around month fifteen, corresponding to April 2009.

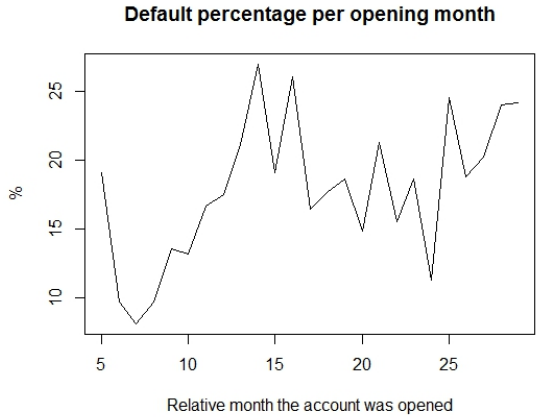

Figure 1.2: Relative number of defaults per month

We shall consider whether it is possible to fit a first order autoregressive model to default rates before critically analysing some of the methods we can use to apply bootstrapping to assess the accuracy of such a method. Underpinning such theory is the assumption that individuals within the population are homogeneous. However, we know this is not the case. Different groups of people are likely to have different associated default rates, which forms the basis of the credit scoring methodology. In spite of this we will consider the assumption of a homogeneous population. We are also faced with the problem of accounts being closed prematurely. This is known as right-censoring. Subsequently elements of survival analysis will be considered to deal with the specific issue of right-censored data.

Chapter 2

Bootstrapping

2.1 Introduction

When discussing the concept of *bootstrapping*, it is important to distinguish between the parametric and non-parametric bootstrap. For the parametric bootstrap, we make some assumptions about the distributional form. By comparison, for the non-parametric bootstrap estimate we make no distributional assumptions and instead employ the empirical distribution. This has the advantage of allowing us to obtain standard errors, however complicated the estimator. The non-parametric bootstrap methodology will therefore form the basis of our research.

2.2 Non-parametric Bootstrap

We begin by considering a simple problem. Suppose we have a random sample $\mathbf{x} = (x_1, x_2, \dots x_n)$ from some unknown probability distribution F and we wish to estimate a parameter of interest θ on the basis of $\mathbf{x}$. For this purpose we calculate an estimate $\hat{\theta} = s(\mathbf{x})$ from $\mathbf{x}$, where $s(\cdot)$ denotes our statistic of interest. For example, if we suppose θ is the mean then we take $s(\mathbf{x})$ as the sample mean $\bar{x}$. We now ask the question, how accurate is our estimate $\hat{\theta}$? The non-parametric bootstrap provides a method to estimate the standard error of $\hat{\theta}$ without making any prior theoretical assumptions. We denote the standard error of $\hat{\theta}$ as $se_F(\hat{\theta})$ [1, p.40].

If we denote $\hat{F}$ as the empirical distribution defined as [4, p.1]

$$\hat{F}(t) = \frac{\text{number of elements in the sample } \leq t}{n} = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq t)$$

A bootstrap sample is then defined to be a random sample of size n drawn from $\hat{F}$. This is denoted as $\mathbf{x}^* = (x_1^*, ..., x_n^*)$. Each bootstrap sample is equivalent to drawing with replacement a random sample of size n from the population of n data points $(x_1, ..., x_n)$.

Correspondingly we find the *bootstrap replication* of $\hat{\theta}$ as

$$\hat{\theta}^* = s(\mathbf{x}^*)$$

where the quantity $s(\mathbf{x}^*)$ is the result of applying the same function $s(\cdot)$ to $\mathbf{x}^*$ as was applied to $\mathbf{x}$. The bootstrap estimate of $se_F(\hat{\theta})$, is a plug-in estimate that uses the empirical distribution function $\hat{F}$ in place of the unknown distribution F. We therefore estimate $se_F(\hat{\theta})$ using $se_{\hat{F}}(\hat{\theta}^*)$, which we refer to as the *ideal bootstrap estimate*. This process is repeated a large number of times to provide a good numerical approximation of the value of $se_{\hat{F}}(\hat{\theta}^*)$. A diagrammatic demonstration of the method is given in Figure 2.1.

2.2.1 Non-parametric bootstrap algorithm

The above is an outline of what is referred to as the *One-Sample Problem*. Our random sample $\mathbf{x}$ is drawn from one unknown probability distribution F. We will later consider the Two-Sample problem

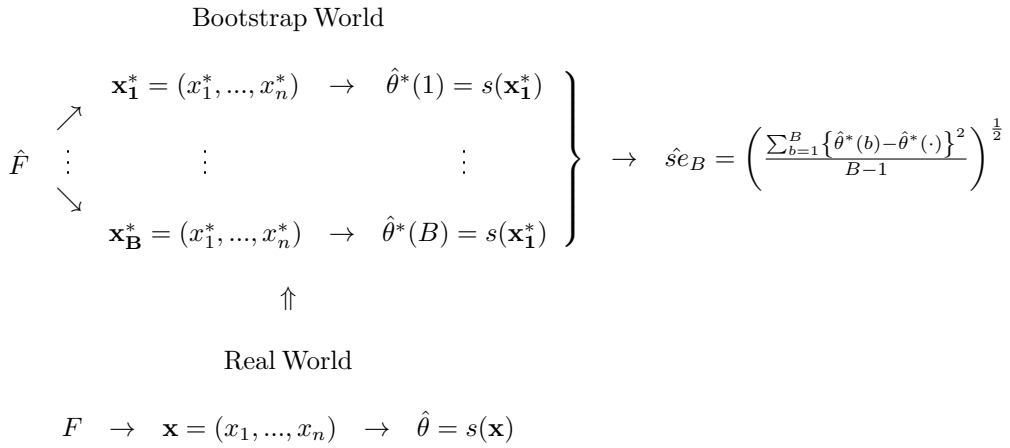

Figure 2.1: Visualisation of the bootstrap process for estimating the standard error of a statistic $s(\mathbf{x})$.

which deals with two mutually independent random samples drawn from two probability distributions. We summarise the above with the following algorithm:

1. Select B independent bootstrap samples $\mathbf{x_1^*}, \mathbf{x_2^*}, ..., \mathbf{x_B^*}$, each consisting of n data points drawn with replacement from the observed data set $\mathbf{x} = (x_1, ..., x_n)$.

2. Evaluate the bootstrap replication corresponding to each bootstrap sample

$$\hat{\theta}^*(b) = s(\mathbf{x_b^*}) \text{ for } b = 1, 2, ..., B$$

3. Estimate the standard error $se_F(\hat{\theta})$ by the sample standard deviation of the B replications

$$\hat{se}_B = \left(\frac{\sum_{b=1}^{B} \left\{ \hat{\theta}^*(b) - \hat{\theta}^*(\cdot) \right\}^2}{B-1} \right)^{\frac{1}{2}}$$

where $\hat{\theta}^*(\cdot) = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^*(b)$

We can see that $\lim_{B \to \infty} \hat{se}_B = se_{\hat{F}} = se_{\hat{F}}(\hat{\theta}^*)$. This is equivalent to noting that as $B \to \infty$ the empirical standard deviation approaches the population standard deviation.

2.2.2 Example

Having described the algorithm that allows us to implement the non-parametric bootstrap, we shall now demonstrate the power of this technique. In Table 2.1 we have credit scores for 25 individuals. If we suppose that the population distribution is unknown, we can estimate the standard error of the mean of these scores using the bootstrap algorithm and compare this to theoretical result.

0.75311889	0.90697848	0.79900132	-0.81674408	0.70691348
-0.66770845	-0.04024196	0.85663310	0.70299466	1.38542186
0.72114495	-0.60780066	0.29926675	-1.20001968	0.51914446
-0.46625055	1.31689987	-0.99389013	0.20462409	-1.57084974
-1.22438156	1.62587981	-0.40231527	1.15071131	-0.35977559

Table 2.1: Credit scores for the 25 individuals

The sample mean of this data is given as 0.1439502. For 200 bootsamples the standard error is estimated as 0.1753753. [Appendix 7.1]

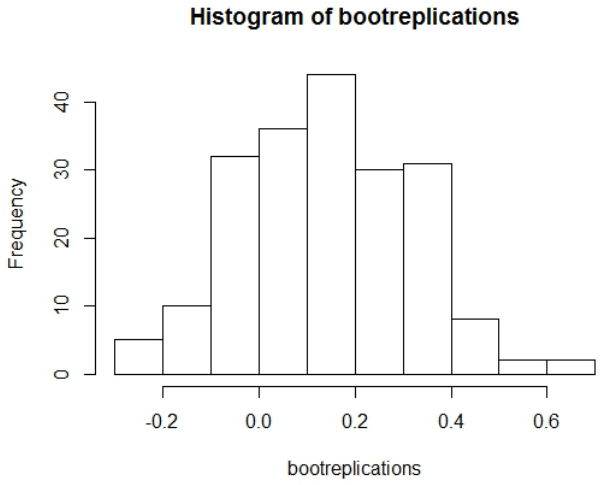

Figure 2.2: Histogram of the 200 bootstrap replications of the mean credit score

We can also estimate the standard error by applying the central limit theorem. If we have data points $X_1, ..., X_n$, independently identically distributed, then

$$\frac{\frac{1}{n}\sum_{i=1}^{n} X_i - \mu}{\sigma/\sqrt{n}} = \frac{\bar{X} - \mu}{\sigma/\sqrt{n}} \to N(0,1) \quad \text{as } n \to \infty$$

Hence $\bar{X} \to N(\mu, \frac{\sigma^2}{n})$ and therefore, we can estimate the standard error of the sample mean as $\frac{\sigma}{\sqrt{n}}$.

In actual fact the 25 data points were sampled from a $N(0,1)$ distribution and therefore we can calculate the standard error as $\frac{1}{\sqrt{25}} = 0.2$. We can see that our bootstrap estimate of the standard error is close to the theoretical value, yet was calculated without having to make any theoretical assumptions. This is particularly useful in the context of credit scoring, where we cannot guarantee that individuals within a given data set are independently identically distributed.

2.2.3 How large to make B

In the above algorithm the number of bootsamples is defined as B. The ideal bootstrap estimate $\hat{se}_\infty$ takes $B = \infty$, in which case $\hat{se}_\infty$ equals the plug-in estimate $se_{\hat{F}}(\hat{\theta}^*)$. However, for most practical purposes, excluding the calculation of confidence intervals, a value of $B = 200$ will suffice. For the specific case of calculating confidence intervals Efron [1, p.52] recommends a value of $B = 1000$ or more. This is important to ensure that the bootstrap replications approximately take the expected distributional form, which is especially important when calculating confidence intervals using the percentile confidence interval method. We shall explore this in more detail in the following Section.

2.3 Bootstrapping Confidence Intervals

We now turn our attention to the problem of bootstrapping confidence intervals. This shall form the basis of our study, in particular when comparing credit score cards. We start by considering how we might place confidence intervals on one isolated statistic.

2.3.1 Introduction

Suppose we are in the one-sample situation where the data $\mathbf{x}$ is obtained by random sampling from some unknown distribution F. Let $\hat{\theta}$ be our estimate of some statistic of interest θ and let $\hat{se}$ be our estimate of the standard error for $\hat{\theta}$. Under the central limit theorem $\hat{\theta} \sim N(\theta, \hat{se}^2)$ or equivalently

$$Z = \frac{\hat{\theta} - \theta}{\hat{se}} \sim N(0,1) \tag{2.1}$$

Therefore, as $n \to \infty$ if we let $z^{(\alpha)}$ indicate the $100 \cdot \alpha$th percentile point of a $N(0,1)$ distribution, confidence intervals for θ are obtained as [1, p.158]

$$\left[\hat{\theta} - z^{(1-\alpha)}\hat{se}, \hat{\theta} - z^{(\alpha)}\hat{se}\right]$$

Alternatively we can write our confidence intervals as $\hat{\theta} \pm z^{(1-\alpha)} \cdot \hat{se}$.

The above methodology, however, is only valid as $n \to \infty$. More generally, for finite samples, we use the Student's t distribution [1, p.158].

2.3.2 Student's t interval

Here we say that for a sample of size n

$$Z = \frac{\hat{\theta} - \theta}{\hat{se}} \sim t_{n-1}$$

where t_{n-1} denotes the student's t distribution with $n-1$ degrees of freedom. If we let $t_{n-1}^{(\alpha)}$ indicate the $100 \cdot \alpha$th percentile point of t_{n-1} then using this approximation confidence intervals are obtained as

$$\left[\hat{\theta} - t_{n-1}^{(1-\alpha)} \cdot \hat{se}, \hat{\theta} + t_{n-1}^{(\alpha)} \cdot \hat{se}\right]$$

We can demonstrate how we might bootstrap confidence intervals using the student's t method as follows. If we take the 25 data points generated from a $N(0,1)$, as in our one-sample problem in Section 2.2.2, our mean is given as $\hat{\theta} = 0.1439502$ with standard error $\hat{se} = 0.1753753$. Using the Student's t distribution with $n - 1 = 24$ degree's of freedom, 95% confidence intervals are given as $0.1439502 \pm 2.059539 \cdot 0.1753753 = [-0.2172421, 0.5051425]$. As might have been expected, our 95% confidence interval includes the value 0.

We now aim to remove the normal theory assumptions as made in (2.1). The following method estimates the distribution of Z directly from the data, which is then used to construct confidence intervals.

2.3.3 The bootstrap-t interval

The method proceeds as follows [1, p.160]. Generate B bootstrap samples $\mathbf{x_1^*}, ..., \mathbf{x_B^*}$ and for each compute

$$Z^*(b) = \frac{\hat{\theta}^*(b) - \hat{\theta}}{\hat{se}^*(b)}$$

where $\hat{\theta}^*(b) = s(\mathbf{x_b^*})$ is the value of $\hat{\theta}$ for the bootstrap sample $\mathbf{x_b^*}$ and $\hat{se}^*(b)$ is the estimated standard error of $\hat{\theta}^*$ for the bootstrap sample $\mathbf{x_b^*}$. For the specific case of $\hat{\theta}$ equal to the mean, use the plug-in estimate

$$\hat{se}^*(b) = \left\{ \frac{\sum_{i=1}^{n} \left(x_i^{*b} - \bar{x}^{*b}\right)^2}{n^2} \right\}^{\frac{1}{2}}$$

The αth percentile of $Z^*(b)$ is estimated by the value $\hat{t}^{(\alpha)}$ such that

$$\frac{\#\left\{Z^*(b) \leq \hat{t}^{(\alpha)}\right\}}{B} = \alpha$$

For example, when $B = 5000$, an estimate of the 5% point is the 250th largest value of $Z^*(b)$'s and an estimate of the 95% point is the 4750th largest value of the $Z^*(b)$'s. If $B \cdot \alpha$ is not an integer then calculate $k = \text{ceiling}(\alpha \cdot (B+1))$ and take the k^{th} largest and the $(B+1-k)^{\text{th}}$ largest values of $Z^*(b)$. The bootstrap confidence intervals are then given as

$$\left[\hat{\theta} - \hat{t}^{(1-\alpha)} \cdot \hat{se}, \hat{\theta} + \hat{t}^{(\alpha)} \cdot \hat{se}\right]$$

Efron and Tibshirani [1, p.161] demonstrate that in large samples the convergence of the bootstrap-t interval tends to be closer to the desired level than the convergence using either the student's t or standard

normal. However, this gain in accuracy comes at the price of generality. Standard normal tables can be applied to all sample sizes, the student's t distribution applies to all samples of fixed size n, however the bootstrap-t methodology applies only to the given sample. Hence a large value of B is especially important.

There are further problems with bootstrap-t confidence intervals. In the above example we have an explicit formula for our standard error. However, if we do not have an explicit formula for our standard error then Efron [1, p.162] explains that we are required to create a bootstrap estimate for each bootstrap sample. This implies we require two levels of bootstrap sampling, which would require an even larger number of iterations. This is particularly important in our case, where there exists no standard method to estimate the standard error for a particular score cards AUC.

2.3.4 Percentile bootstrap confidence intervals

If we consider the histogram of our bootstrap samples as in Figure 2.2 we can see it is roughly normal in shape. Therefore, it is possible for us to use percentiles from the bootstrap histogram to define our confidence intervals.

Suppose that we generate our bootstrap data set $\mathbf{x}^*$ and the bootstrap replications $\hat{\theta}^* = s(\mathbf{x}^*)$ are computed. Now let $\hat{G}$ be the cumulative distribution function of $\hat{\theta}^*$. The $1-2\alpha$ percentile interval as $B \to \infty$ is then defined as

$$\left[\hat{G}^{-1}(\alpha), \hat{G}^{-1}(1-\alpha)\right]$$

For a finite number of bootsamples we generate B independent bootstrap data sets $\mathbf{x_1^*}, \mathbf{x_2^*}, ..., \mathbf{x_B^*}$ and compute the bootstrap replications $\hat{\theta}^*(b) = s(\mathbf{x_b^*})$ for $b = 1, ..., B$. Ordering these bootstrap replications, we then define $\hat{\theta}_B^{*(\alpha)}$ to be the $100 \cdot \alpha$th empirical percentile of the $\hat{\theta}^*(b)$ values. Similarly, define $\hat{\theta}_B^{*(1-\alpha)}$ to be the $100 \cdot (1-\alpha)$th empirical percentile. The $1-2\alpha$ percentile interval is then defined as

$$\left[\hat{\theta}_B^{*(\alpha)}, \hat{\theta}_B^{*(1-\alpha)}\right]$$

If the bootstrap distribution is roughly normal then the percentile confidence intervals and the standard normal confidence intervals will nearly agree. From the central limit theorem, as $n \to \infty$ the bootstrap histogram will become normal shaped, however for smaller n this might not always be the case. In the context of credit scoring data, we normally deal with large data sets and using modern day computers can use a large value for B without significant time constraint and therefore we might be led to believe we are no better favouring either technique over the other. In actual fact, the percentile method is our preferred choice. We shall demonstrate why as follows.

If we were to consider the case of a statistic θ such that it has undergone a transformation $\hat{\phi} = m(\hat{\theta})$ where $\hat{\phi} \sim N(\phi, c^2)$, for some value c, then confidence intervals on $\hat{\theta}$ equals

$$[m^{-1}(\hat{\phi} - z^{(1-\alpha)}c), m^{-1}(\hat{\phi} - z^{(\alpha)}c)]$$

Percentile confidence intervals adjust for said transformation, however, if we were to naively apply standard confidence interval techniques they do not automatically adjust for the transformation and therefore we would yield unreliable confidence intervals [1, p.175]. The above invariance to transformations makes the percentile bootstrap method arguably the most attractive.

2.4 Non-bootstrap methods

A different method used to compute confidence intervals is using the properties of the Mann-Whitney statistic. It has been shown that the area under an empirical ROC curve is equal to the Mann-Whitney two-sample statistic and therefore can be interpreted as the probability that a randomly drawn positive case has a lower score than a randomly drawn negative case [10, p.838]. We give a brief overview of our problem in Chapter 4.

Chapter 3

Bootstrapping a Credit Scorecard

3.1 Building a basic Credit Scorecard

We start by building a basic credit scorecard, which aims to predict whether a person does not default. Our data set consists of 4635 credit card accounts. This has been indexed in time, which means we know when a particular account was opened and if applicable, when an individual defaulted. This potentially adds an interesting extra dimension to our analysis. However, for simplicity, we'll start by assuming the data points are independent of time.

3.1.1 Preparing the data

The data has several variables which could potentially be used to build our score card. These are given in Table 4.1. It is important to note that our definition of default is three missed payments within 12 months of opening the account.

Variable	Description	Values
ID	Applicant/account id to be used for data matching	Integer
Age	Age of applicant at time of application	Integer
Employment Status	Employment status at time of application	Categories
Tenure	Home ownership status at time of application	Categories
Months at Address	Total months at current address at time of application	Integer
Application Channel	Channel which the application was made	Categories
Open Year	Year of account opening	1008 to 2011
Open Month	Month of account opening	1 to 12
Open Relative Month	Months from January 2008 to date of account opening	5 to 41
Default	Did the account default (3 months down) within 12 months of opening?	True/False
Statement Number	Statement number at time of default	4 to 37
Statement Year	Year of default occurring	2008 to 2011
Statement Month	Month of default occurring	1 to 12
Statement Relative Month	Months from January 2008 to date of default	8 to 41

Table 3.1: Data Variables. Here our categories for Employment Status are: EM - Employed, HO - Homemaker, RE - Retired, SE - Self-Employed, ST - Student; for Tenure: CT - Council Tenant, HO - Homeowner, LP - Living with Parents, PT - Private Tenant and for Application Channel: Cold Call, Internet, Mail, Other.

We start by considering some basic transformation of variables. For the categories, we create indicator variables for each of the possible values. Further, we shall divide Age and Months at Address into quintiles. It is also important to consider outliers. An outlier is an extreme value which is therefore considered highly unlikely; however we must proceed with caution since determining an outlier depends on the underlying distribution. We shall consider as potential outliers any value which is greater than $1.5 \cdot$ Interquartile Range above or below the upper and lower quartile. An extreme outlier is taken as

more than $3 \cdot$ Interquartile Range above or below the upper and lower quartile. For Months at Address we find there are 8 records which could be described as extreme outliers. With the aim of building a robust model, we shall discard these records. We also note there are 147 records with Tenure blank. We shall add an indicator variable for these records.

3.1.2 Building the Logistic Regression Model

After doing this we build a scorecard using logistic regression, modeling whether people are "good". The scorecard is used to compute the predicted probabilities of being a good payer on the test data set. Finally the Receiver Operating Characteristic (ROC) curve is plotted. We assume the data points are independent of time and thus assume the method used to divide the data into test and training data sets is statistically insignificant. Therefore, for simplicity random sampling, without replacement, shall be used to divide our data sets. We shall divide test:training into the ratio 1:3.

Our general linear model is built using the following variables: Age divided into quintiles, Months at Address divided into quintiles, Tenure as category variables, Employment Status as category variables and Application Channel as category variables. A summary for our model is given in Table 4.2.

	Estimate	Standard Error	z value	p value	Significance
(Intercept)	2.61344	1.07269	2.436	0.01484	*
age1	-0.40073	0.16478	-2.432	0.01502	*
age2	-0.09380	0.16245	-0.577	0.56367	
age3	-0.10854	0.24212	-0.448	0.65395	
age4	-0.01420	0.17325	-0.082	0.93470	
MONTHS_AT_ADDRESS1	-0.08437	0.15959	-0.529	0.59704	
MONTHS_AT_ADDRESS2	-0.45323	0.15213	-2.979	0.00289	**
MONTHS_AT_ADDRESS3	-0.46458	0.17036	-2.727	0.00639	**
MONTHS_AT_ADDRESS4	-0.42934	0.19073	-2.251	0.02439	*
TENURE_CT	-0.37857	0.17336	-2.184	0.02898	*
TENURE_HO	0.97949	0.15327	6.391	1.65e-10	***
TENURE_PT	-0.09681	0.14744	-0.657	0.51146	
EMPLOY_STATUS_EM	0.42133	0.17871	2.358	0.01839	*
EMPLOY_STATUS_HO	0.14514	0.29777	0.487	0.62596	
EMPLOY_STATUS_RE	0.45275	0.38633	1.172	0.24123	
EMPLOY_STATUS_SE	0.48262	0.23951	2.015	0.04390	*
APPLICATION_CHANNEL_C	-1.00227	1.09143	-0.918	0.35846	
APPLICATION_CHANNEL_I	-1.18398	1.04569	-1.132	0.25753	
APPLICATION_CHANNEL_M	-1.48169	1.08007	-1.372	0.17011	

Table 3.2: Summary for our General Linear Model, with significance codes for each variable given as: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

We also plot our ROC curve and calculate the models AUC, given as 0.6550542 [Appendix 7.2]. The bootstrap algorithm is employed with the aim of placing confidence intervals on this estimate.

3.2 Bootstrapping AUC

In Section 2.3 we discussed four different methods to place confidence intervals on a particular models AUC. We shall first explain how we bootstrap our AUC before explaining how we interpret the algorithms presented and then compare each method. In all four algorithms, our statistic of interest, θ, is the model's AUC and our estimate, $\hat{\theta}$, is given as 0.6550542 as in the previous Section. We shall denote $\theta = AUC$ and $\hat{\theta} = \hat{AUC} = 0.6550542$.

There are two different methods to bootstrap AUC as shown in Figure 3.1. The first method divides our original data set in two and then generates one model on the training set and bootstraps the test data set B times to produce B AUCs formed using the same model. The second method produces B training data sets and B models based on these data sets. These are then tested on the corresponding test data set to produce B AUCs.

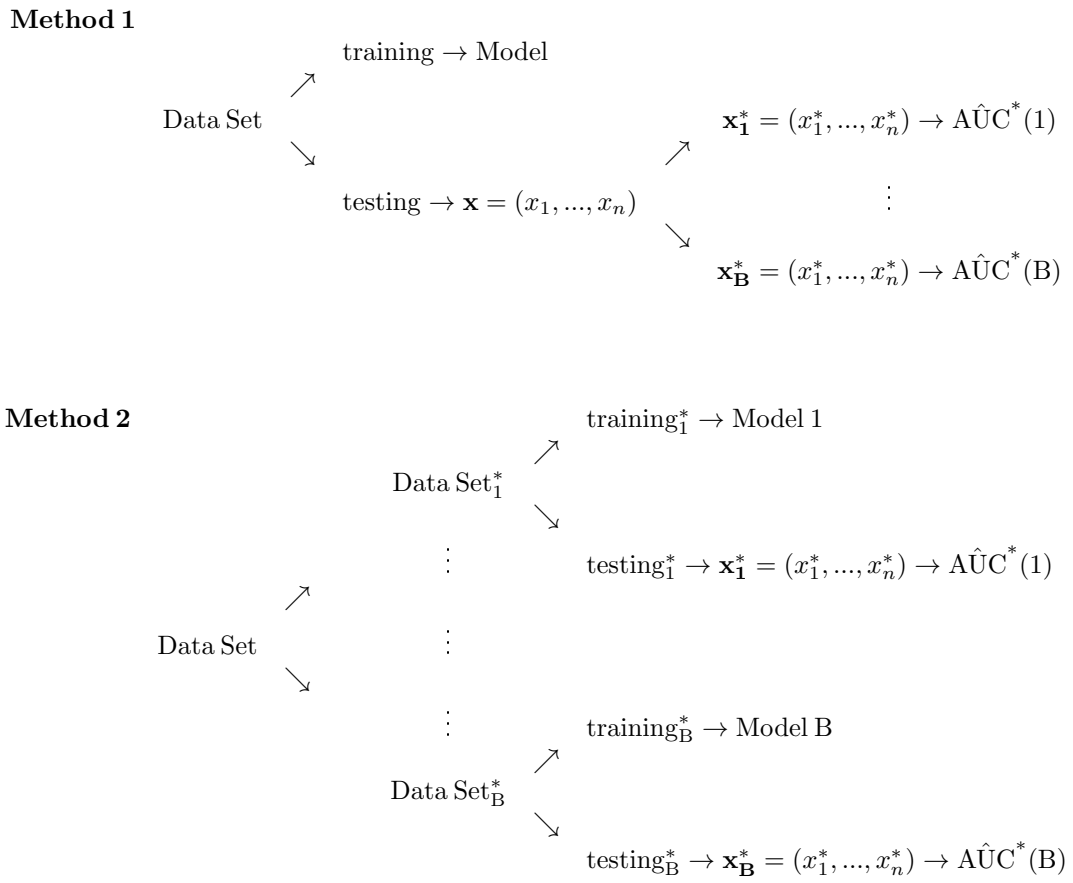

Figure 3.1: Two different methods to bootstrap AUC

There is a subtle difference between each method. It's important to emphasise that the first method builds just one model based on the borrower characteristics. However, the second builds B different models based on the same characteristics, which are optimised with respect to each of the B training data sets.

The first method is widely referred to in the literature as standard. From a business perspective, credit rating agencies often build just one model, colloquially referred to as the *Champion*, and wish to measure this models performance relative to a *Challenger*. Furthermore, it is clear that the second method, where we generate B models and a corresponding AUC for each, is computationally more expensive. The first method will therefore form the basis of our study. We shall now consider how we might bootstrap confidence intervals for our model [Appendix 7.3].

3.2.1 Gaussian Normal interval

The first method that was described used the central limit theorem. Here we say that as $n \rightarrow \infty$

$$Z = \frac{\hat{AUC} - AUC}{\hat{se}} \sim N(0,1) \tag{3.1}$$

and therefore, if we let $z^{(\alpha)}$ indicate the $100 \cdot \alpha$th percentile point of a $N(0,1)$ distribution, our confidence intervals are given as $\hat{AUC} \pm z^{(1-\alpha)} \cdot \hat{se}$. However, this is only valid as $n \rightarrow \infty$, yet our data set contains information about 1158 individuals. Using $B = 1000$ our confidence intervals are given as

- 95% upper confidence interval = 0.694923
- 95% lower confidence interval = 0.6151854

3.2.2 Student's t interval

For a sample of size n

$$Z = \frac{A\hat{U}C - AUC}{\hat{se}} \sim t_{n-1}$$

Using this approximation confidence intervals are given as

$$\left[A\hat{U}C - t_{n-1}^{(1-\alpha)} \cdot \hat{se}, A\hat{U}C + t_{n-1}^{(\alpha)} \cdot \hat{se}\right]$$

For $n = 1158$ we therefore obtain our confidence intervals using $B = 1000$ as

- 95% upper confidence interval = 0.6949647
- 95% lower confidence interval = 0.6151437

In both cases our standard error, $\hat{se}$, is estimated using the bootstrap method.

3.2.3 The bootstrap-t interval

The procedure is as follows [1, p.160]. We generate B_1 bootstrap samples $\mathbf{x_1^*}, ..., \mathbf{x_{B_1}^*}$ and for each compute

$$Z^*(b) = \frac{A\hat{U}C^*(b) - A\hat{U}C}{\hat{se}^*(b)}$$

where $\hat{se}^*(b)$ is the estimated standard error of $A\hat{U}C^*$ for the bootstrap sample $\mathbf{x_b^*}$. Since there exists no closed formula to calculate $\hat{se}^*(b)$, we estimate the standard error using a second bootstrap level. Here for each bootstrap sample $\mathbf{x_b^*}$, $b = 1, ..., B_1$, we generate B_2 bootstrap samples $\mathbf{x_{b,1}^*}, ..., \mathbf{x_{b,B_2}^*}$. The AUC is then calculated for each $A\hat{U}C^*(b,1), ..., A\hat{U}C^*(b,B_2)$ and used to calculate the standard error

$$\hat{se}^*(b) = \left(\frac{\sum_{i=1}^{B_2} \left\{ A\hat{U}C^*(b,i) - A\hat{U}C^*(b,\cdot) \right\}^2}{B_2 - 1} \right)^{\frac{1}{2}}$$

where $A\hat{U}C^*(b,\cdot) = \sum_{i=1}^{B_2} \frac{A\hat{U}C^*(b,i)}{B_2}$. The αth percentile of $Z^*(b)$ is estimated by the value $\hat{t}^{(\alpha)}$ such that

$$\frac{\#\left\{Z^*(b) \leq \hat{t}^{(\alpha)}\right\}}{B_1} = \alpha$$

The bootstrap confidence interval is given as

$$\left[A\hat{U}C - \hat{t}^{(1-\alpha)} \cdot \hat{se}, A\hat{U}C + \hat{t}^{(\alpha)} \cdot \hat{se}\right]$$

Where we estimate $\hat{se}$ using

$$\hat{se}_{B_1} = \left(\frac{\sum_{b=1}^{B_1} \left\{ A\hat{U}C^*(b) - A\hat{U}C^*(\cdot) \right\}^2}{B_1 - 1} \right)^{\frac{1}{2}}$$

A visualisation of this method is shown in Figure 4.3. Using $B_1 = 1000$ and $B_2 = 200$, our confidence intervals are calculated as

- 95% upper confidence interval = 0.6917405
- 95% lower confidence interval = 0.6115309

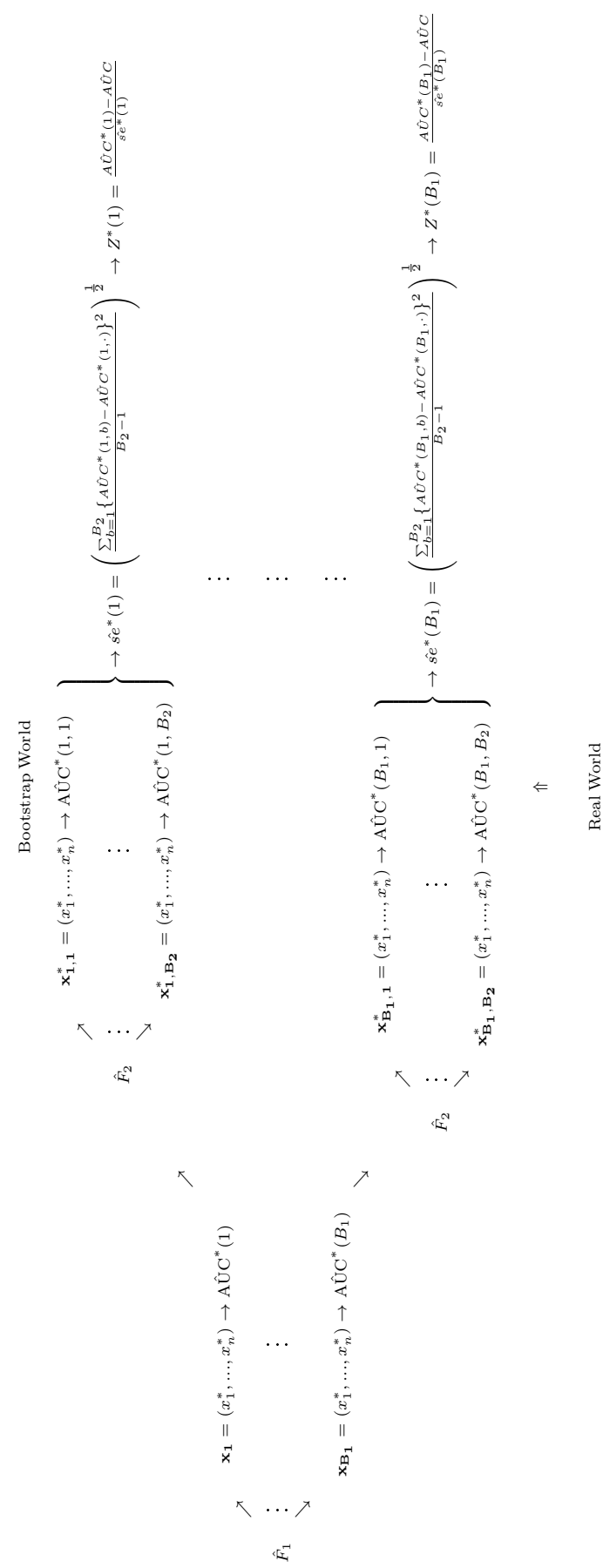

Figure 3.2: Visualisation of the bootstrap process for implementing bootstrap-t confidence intervals on AUC using two layers of bootstrapping.

3.2.4 Percentile bootstrap confidence intervals

Here we generate B independent bootstrap data sets $\mathbf{x_1^*}, \mathbf{x_2^*}, ..., \mathbf{x_B^*}$ and for $b = 1, ..., B$ we compute $AUC^*(b)$. We order these bootstrap replications and define $AUC_B^{*(\alpha)}$ to be the $100 \cdot \alpha$th empirical percentile of the $AUC^*(b)$ values. Similarly, we define $AUC_B^{*(1-\alpha)}$ to be the $100 \cdot (1-\alpha)$th empirical percentile. The $1-2\alpha$ percentile interval is then defined as

$$\left[AUC_B^{*(\alpha)}, AUC_B^{*(1-\alpha)}\right]$$

Our confidence intervals using $B = 1000$ are obtained as

- 95% upper confidence interval = 0.6967083
- 95% lower confidence interval = 0.6180078

3.2.5 Comparison

We summarise our results in Table 3.3. As we might have expected there is only a small discrepancy between Student's t and Gaussian Normal confidence intervals, due to the large data set. Interestingly our bootstrap-t confidence intervals have a smaller upper and lower value and the percentile method produces a larger lower and upper confidence interval.

	Lower Confidence Interval	Upper Confidence Interval
Gaussian Normal	0.6151854	0.694923
Student's t	0.6151437	0.6949647
bootstrap-t	0.6115309	0.6917405
Percentiles	0.6180078	0.6967083

Table 3.3: Summary of our Results for one General Linear Model

In the literature, we find that percentiles is the preferred method due to its robustness, as indicated in Skalska (2006) [9]. Furthermore, bootstrap-t confidence intervals are computationally expensive to compute. With both of these things in mind, we shall use this as our favoured method for future applications.

An explanation for the difference in results between the percentile and both the normal and Student's t intervals could be due to the nature of a stochastic process, which the bootstrap methodology falls under. We might expect that for larger values of B we obtain values which are closer. We shall explore the effect of B on our standard error in the following Section.

3.3 Number of bootsamples

In the above algorithm we took our value of B as 1000. If we consider Figure 3.5, we can see that the value for the mean of our bootstrap converges as the number of bootstrap samples increases. Furthermore, a value of $B = 200$ provides a relatively accurate estimation of our AUC, given limited computer processing speed [Appendix 7.4].

3.4 Limited data set

We shall now look at a specific application of the bootstrap methodology. When building a score card we have emphasised that it is important for us to measure its performance on an independent data set, which we refer to as the testing data set. If we measure the performance of a model on the same data set as it is built on, it is likely we will over estimate its AUC due to over-fitting. However, if we consider a situation where we are dealing with a small data set it might not be practical to obtain a separate hold-out sample. If this is the case, one can use cross-validation or bootstrapping to obtain unbiased estimates as outlined by Lyn Thomas [7, p.102]. We shall first look at cross-validation as a potential solution before analysing how we might implement the bootstrap algorithm.

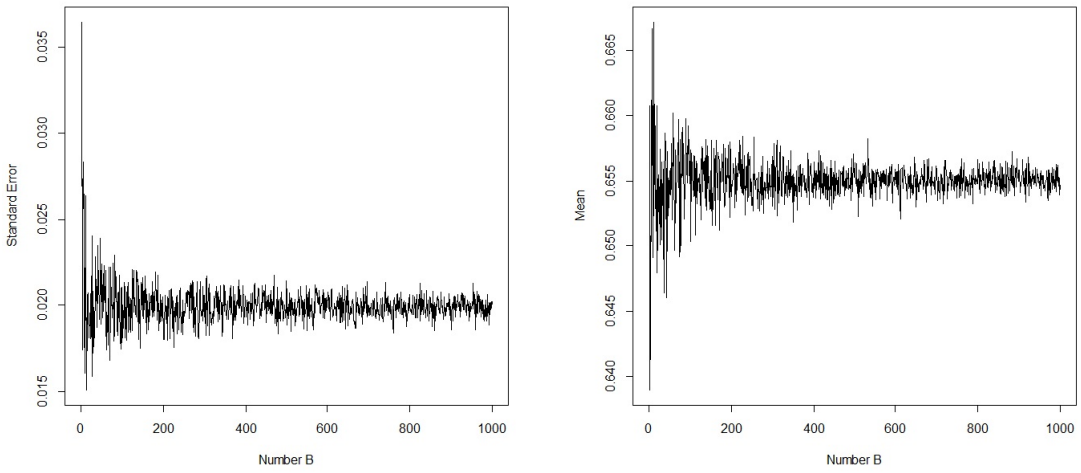

Figure 3.3: Effect on the standard error and mean as the number of bootsamples is increased.

3.4.1 Cross-validation

Here we divide our data set into K subsets, building a model on each and then testing on the complement. This generates K models and K unbiased estimates of each models performance. A visualisation of this is given in Figure 3.6.

$$\text{Data Set} \begin{cases} \nearrow & \text{Data Set}_1 \rightarrow \text{Model 1} \\ & \text{Data Set}_1^c \rightarrow \text{AUC}_1 \\ & \vdots \\ \searrow & \text{Data Set}_K \rightarrow \text{Model K} \\ & \text{Data Set}_K^c \rightarrow \text{AUC}_\text{K} \end{cases}$$

Figure 3.4: Cross-validation method. We divide the data set into K subsets.

There are two methods to implement this, the rotation method and the leave-one-out method. In the rotation method the data set is divided into K non-intersecting subsets. A score card is then built on $K-1$ data sets and tested on the other data set. The overall score card is made up of the average of the estimates for each characteristic and a measure of its performance is the average of the AUCs.

In the leave-one-out approach each data point is left out in turn and a score card is developed on the remaining data. The overall score card is then an average of those obtained. However, using this approach, one would need to implement a different performance measure than AUC and additionally one would need to generate a large number of score cards. Therefore, for many practical purposes the rotation method might be preferred.

3.4.2 Bootstrapping

Our objective is to obtain an unbiased estimate for the performance of a model. In an ideal scenario this model is built on one data set and the corresponding AUC is calculated on an independent testing data set. However, for a situation where we do not have a large enough data set to do this, we are restricted to building and testing a model on the same data. In order to estimate the performance of this model, had it been tested on an independent data set, we could use the bootstrapping methodology.

For a data set of size N, we re-sample this with replacement B times to obtain B training and test data sets. We then develop B models on each and calculate the corresponding AUC on both the training

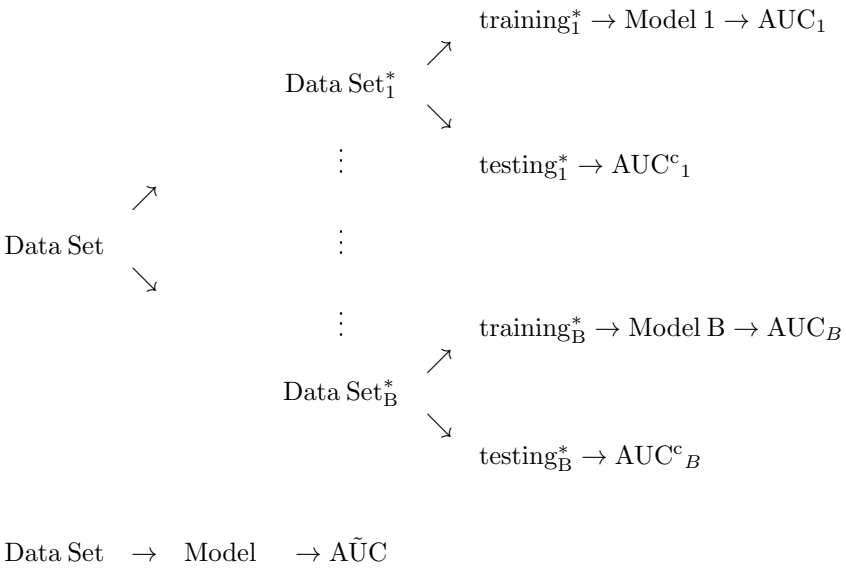

Figure 3.5: Visualisation of how we might bootstrap a limited data set.

data set and the test data set. As discussed previously we expect the AUC obtained for the training data set to be higher than that obtained using the independent test data set. Here we shall denote AUC_b as the AUC obtained using the training data set and $\text{AUC}^\text{c}_\text{b}$ as the AUC obtained using the test data set, for $b = 1, ..., B$. A visualisation of the methodology is given in Figure 3.7.

An important assumption is that the mean of the errors $\text{AUC}_\text{b} - \text{AUC}^\text{c}{}_\text{b}$ is a good estimate for the difference between the AUC obtained when the model is tested on the same data set, denoted $\tilde{\text{AUC}}$, minus the AUC obtained if its performance were measured on an independent data set, denoted $\hat{\text{AUC}}$. Therefore, we have

$$\hat{\text{AUC}} \approx \tilde{\text{AUC}} - \left(\frac{\sum_{b=1}^{B} \text{AUC}_\text{b} - \text{AUC}^\text{c}{}_\text{b}}{B} \right)$$

Since we re-sample the original data set with replacement, the chance that Data Set^*_b does not contain an element of the original data set is $\left(1 - \frac{1}{N}\right)^N$. As $N \to \infty$ this converges to $1/e$. Therefore, there is a $1/e$ chance the data point is not in Data Set^*_b and a $1 - 1/e$ chance it is in Data Set^*_b. As a result, a more stable estimate can be obtained as [7, p.104]

$$\hat{\text{AUC}} \approx \sum_{b=1}^{B} \left(1 - \frac{1}{e}\right) \text{AUC}^\text{c}_\text{b} + \left(\frac{1}{e}\right) \text{AUC}_\text{b}$$

We now seek to demonstrate this technique.

3.4.2.1 Application of the bootstrap

Consider the same model as before, however with a smaller data set of only 1000 individuals. Using $B = 1000$, the AUC for the model when its performance is measured on the same data set, denoted $\tilde{\text{AUC}}$, is given as 0.690485. Using the bootstrap technique, we calculate an estimate for $\hat{\text{AUC}}$ as 0.6872121. As expected this estimate is lower than $\tilde{\text{AUC}}$. [Appendix 7.5]

We shall now consider how we might apply the bootstrap methodology with the aim of comparing score cards.

Chapter 4

Comparing Models

4.1 Hypothesis testing

A hypothesis test allows us to compare models in an effective manner. If we have an established model A and a new model B such that B has a higher AUC than A, then we might wish to test whether this difference is statistically significant. In credit scoring literature, the new model is often referred to as the *Challenger*, whilst the current established model is referred to as our *Champion*. We consider the null hypothesis that the AUCs produced by each model are equivalent.

We start by considering the Mann-Whitney test as a potential method to find confidence intervals on the difference between models. We demonstrate some of the limitations of the methodology and show how we might employ the bootstrap algorithm as a way of constructing a hypothesis test.

4.2 Mann-Whitney Test

If we consider our data set $\mathbf{x}$ of n individuals to be made up of p bads and q goods such that $\mathbf{x} = (r_1, ..., r_p, s_1, ..., s_q) = (\mathbf{r}, \mathbf{s})$, $n = p+q$, then the area under the ROC curve is equal to the Mann-Whitney two-sample statistic applied to the two samples $\mathbf{r}$ and $\mathbf{s}$. We can therefore apply the general theory of U-statistics. The Mann-Whitney statistic estimates the probability, θ, that a randomly drawn good individual has a lower score than a randomly drawn bad individual [10, p.838]. Our estimate is computed as

$$\hat{\theta} = \frac{1}{pq} \sum_{j=1}^{q} \sum_{i=1}^{p} \psi(r_i, s_j) \tag{4.1}$$

where

$$\psi(R, S) = \begin{cases} 1 & S < R \\ \frac{1}{2} & S = R \\ 0 & S > R \end{cases}$$

Therefore, $E(\hat{\theta}) = \theta = \Pr(S < R) + \frac{1}{2}\Pr(S = R)$. This is equal to the area under our ROC curve [8, p.276]. Confidence intervals can then be found using the theory as outlined in DeLong [10, p.838-841].

However, further complications arise when comparing the difference in AUCs between two models based on the same data. In this case we must also take into account the correlated nature of the data. Therefore, we not only have to compute the variance of each AUC, we must also compute the covariance between them. If we define the Null Hypothesis to be such that both AUCs are equal, we find that test statistic T is defined as

$$T = \frac{(U_1 - U_2)^2}{\sigma_{U_1}^2 + \sigma_{U_2}^2 - 2\sigma_{U_1}\sigma_{U_2}}$$

Where U_1 and U_2 are found using the formula as given in (4.1). This T statistic is asymptotically χ^2 distributed with one degree of freedom, which allows us to compute our critical value for a given confidence interval α [8, p.281].

As we can see, this methodology is not only complex in its execution, it also requires considerable theoretical consideration. Another potential method is to apply the Wilcoxon signed-rank test, however, this also suffers similar constraints. As a consequence bootstrapping may be considered preferable due to its simplicity.

4.3 Bootstrapping the difference in AUCs

We made reference to a fundamental problem in the previous section, when comparing AUCs produced by models based on the same test data set, there exists correlation between these values. Bearing this in mind we shall consider the Two-Sample bootstrap problem, with the aim of comparing two models.

4.3.1 Two Sample Bootstrap

So far we have just considered the basic case of a one-sample bootstrap problem. This is where our data $\mathbf{x}$ is sampled from a single probability distribution F. In this case we can refer to F as the *probability model*. We'll now consider the more complex two-sample problem [1, p.202].

Suppose we have a generalised probability model P which is comprised from two probability distributions F and G. We denote this as $P = (F, G)$. If we denote $\mathbf{z} = (z_1, ..., z_m)$ as a random sample observed from G and $\mathbf{y} = (y_1, ..., y_r)$ as a random sample observed from F, then the observed data $\mathbf{x}$ comprises of $\mathbf{z}$ and $\mathbf{y}$. We denote this as $\mathbf{x} = (\mathbf{z}, \mathbf{y})$ where $\mathbf{x}$ has length $n = m + r$. We therefore have $\mathbf{y}$ and $\mathbf{z}$ mutually independent random samples taken from F and G respectively. This set-up is known as a two sample problem.

If we consider the case of model building with only one test data set, we can see that we do not have two mutually independent random samples and therefore it would be naive to directly compare confidence intervals on AUC. A potential solution is to therefore split our original data set into three mutually exclusive sets: one training set and two test data sets, such that the test data sets are of the same size. We can then independently compute each model's AUC. However, this potentially generates further problems. How do we generate these data sets? How can we assume that if a particular model outperforms another this is not as a result of it being tested on a different data set?

Our specific question is whether the distribution F is the same as the distribution G. Recalling that the data set is time indexed, this might not always be the case, dependent on the method used to divide the data sets up. To proceed we can employ the two sample problem set-up to formulate a hypothesis test. If we find a lack of evidence that the distributional forms of the two data sets differ, then we might assume we could effectively compare score cards built on different data sets.

4.3.2 Proposed Methodology

We aim to use the bootstrap method to test whether the difference between two AUCs is statistically significant. We therefore impose the following hypothesis test:

- H_1: The difference between the two AUCs is significant.
- H_0: We have no evidence that the difference between the two AUCs is significant.

In order to tackle this hypothesis test we divide our data set into three mutually exclusive sets, one training set and two test data sets, such that the distributional form of the two test data sets are F and G, respectively. We then consider the hypothesis test:

- H_1: $F \neq G$.
- H_0: $F = G$.

If it is the case that we have no evidence that $F \neq G$, then we might assume the distributional forms are identical. We then proceed by considering whether the mean bootstrapped AUC generated from each data set are equivalent.

- H_1: The mean of bootstrapped AUCs generated from $\mathbf{z}$ differs from the mean of the bootstrapped AUCs generated from $\mathbf{y}$.

- H_0: Mean of the bootstrapped AUCs generated from $\mathbf{z}$ equals the mean of the bootstrapped AUCs generated from $\mathbf{y}$.

We shall outline how we might tackle these two hypothesis tests before outlining several potential problems in Section 4.3.6.

4.3.3 Algorithm for testing distributional forms

We consider the null hypothesis H_0: $F = G$ vs. the alternate hypothesis H_1: $F \neq G$ and interpret the algorithm given in Efron [1, p.221] as follows

1. Draw B bootstrap samples of size $m + r$ with replacement from $\mathbf{x}$. Call the first m observations $\mathbf{z}^*$ and the remaining r observations $\mathbf{y}^*$.
2. Evaluate $s(\cdot)$ defined by
$$s(\mathbf{x_b}^*) = AUC_2(b) - AUC_1(b), \quad \text{for} \quad b = 1, 2, ..., B$$
where $AUC_1(b)$ corresponds to the AUC generated under the first model from data set $\mathbf{z}^*$ for bootstrap sample b.
3. Approximate the achieved significant level (ASL_{boot}) as
$$\hat{\text{ASL}}_{\text{boot}} = \frac{\#\{s(\mathbf{x_b^*}) \geq s_{obs}\}}{B}$$
where $s_{obs} = s(\mathbf{x})$ is the observed value of the statistic.

A visualisation of this procedure is given in Figure 4.1.

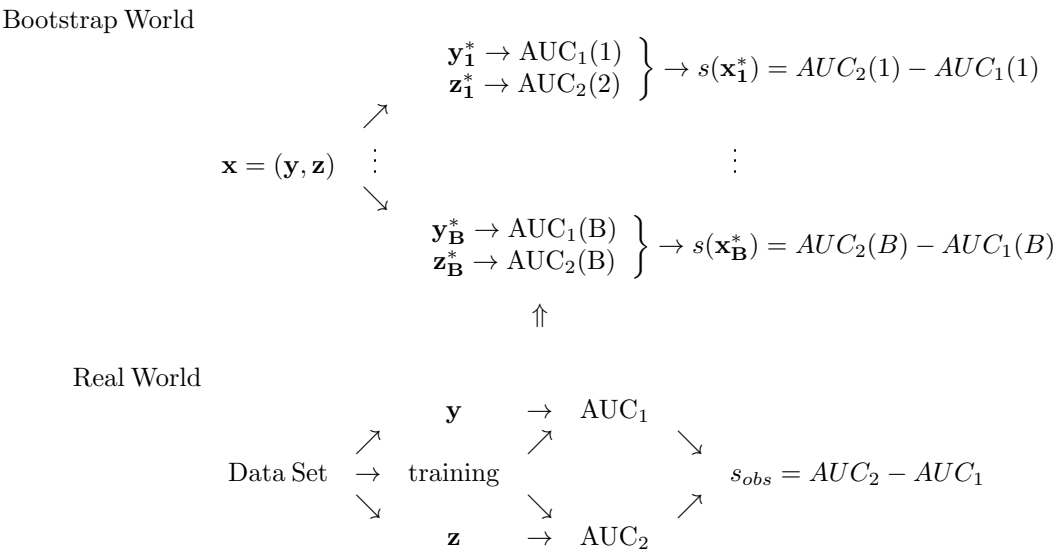

Figure 4.1: The two different models are built on the same training data set, but are tested on different test data sets.

4.3.4 Algorithm for testing equality of means

If the two data sets have the same probability distribution, then we could consider whether the mean of the two statistics are equal. We consider the null hypothesis H_0: The mean AUC generated from the model tested on $\mathbf{z}$ is equal to the mean AUC generated from the model tested $\mathbf{y}$ vs. the alternate hypothesis H_1: The mean AUC generated from $\mathbf{z}$ differs from the mean generated from $\mathbf{y}$. This could potentially allow us to compare whether one model outperforms another by comparing AUCs. The algorithm as presented in Efron [1, p.224] can be interpreted as follows

1. Form B_1 bootstrap data sets $(\mathbf{z}^*, \mathbf{y}^*)$ where $\mathbf{z}^*$ is sampled with replacement from $z_1, z_2, ..., z_m$ and $\mathbf{y}^*$ is sampled with replacement from $y_1, y_2, ..., y_r$.

2. Evaluate $\hat{\text{AUC}}_1^*(1), ..., \hat{\text{AUC}}_1^*(B_1)$ and $\hat{\text{AUC}}_2^*(1), ..., \hat{\text{AUC}}_2^*(B_1)$ where $\hat{\text{AUC}}_1^*(b)$ is generated from $\mathbf{y}^*$ and $\hat{\text{AUC}}_2^*(b)$ is generated from $\mathbf{z}^*$. For ease of notation we'll refer to these sets as $\text{AUC}_1(1), ..., \text{AUC}_1(B_1)$ and $\text{AUC}_2(1), ..., \text{AUC}_2(B_1)$.

3. Evaluate $\tilde{\text{AUC}}_1(i) = \text{AUC}_1(i) - \text{AUC}_1(\cdot) + \overline{\text{AUC}}$ and $\tilde{\text{AUC}}_2(i) = \text{AUC}_2(i) - \text{AUC}_2(\cdot) + \overline{\text{AUC}}$, $i = 1, 2, ..., B_1$, where

$$\text{AUC}_j(\cdot) = \sum_{b=1}^{B_1} \frac{\text{AUC}_j(b)}{B_1} \quad \text{for} \quad j = 1, 2$$

   and $\overline{\text{AUC}}$ is the mean of $\text{AUC}_1(\cdot)$ and $\text{AUC}_2(\cdot)$.

4. Form B_2 bootstrap data sets $(\mathbf{AUC_1}^*, \mathbf{AUC_2}^*)$ where $\mathbf{AUC_1}^*$ is sampled with replacement from $\tilde{\text{AUC}}_1(1), ..., \tilde{\text{AUC}}_1(B_1)$ and $\mathbf{AUC_2}^*$ is sampled with replacement from $\tilde{\text{AUC}}_2(1), ..., \tilde{\text{AUC}}_2(B_1)$. Denote $\tilde{\text{AUC}}_1^*(b) = (\tilde{\text{AUC}}_1^*(b,1), ..., \tilde{\text{AUC}}_1^*(b, B_1))$ and $\tilde{\text{AUC}}_2^*(b) = (\tilde{\text{AUC}}_2^*(b,1), ..., \tilde{\text{AUC}}_2^*(b, B_1))$ for $b = 1, ..., B_2$.

5. Calculate $\overline{\text{AUC}}_1^*(b)$ and $\overline{\text{AUC}}_2^*(b)$ for $b = 1, ..., B_2$, where

$$\overline{\text{AUC}}_j^*(b) = \sum_{i=1}^{B_1} \frac{\tilde{\text{AUC}}_j(b,i)}{B_1} \quad \text{for} \quad j = 1, 2$$

6. Evaluate $s(\cdot)$ defined by

$$s(\mathbf{AUC_b}^*) = \frac{\overline{\text{AUC}}_2^*(b) - \overline{\text{AUC}}_1^*(b)}{\sqrt{\frac{\bar{\sigma}_1^{2*}}{B_1} + \frac{\bar{\sigma}_2^{2*}}{B_1}}} \quad \text{for} \quad b = 1, 2, ..., B_2 \tag{4.2}$$

   where $\bar{\sigma}_1^{2*} = \frac{1}{B_1 - 1} \sum_{i=1}^{B_1} \left(\tilde{\text{AUC}}_1(b,i) - \overline{\text{AUC}}_1^*(b) \right)$ and $\bar{\sigma}_2^{2*} = \frac{1}{B_1 - 1} \sum_{i=1}^{B_1} \left(\tilde{\text{AUC}}_2(b,i) - \overline{\text{AUC}}_2^*(b) \right)$.

7. Approximate the achieved significant level (ASL_{boot}) by

$$\hat{\text{ASL}}_{\text{boot}} = \frac{\#\left\{ s(\mathbf{AUC_b}^*) \geq s_{obs} \right\}}{B_2}$$

   where $s_{obs} = s(\mathbf{x})$ is the observed value of the statistic.

For normal populations equation (4.2) no longer has a Student's t distribution, which in the literature is referred to as the Behrens–Fisher problem [1, p.223]. A visualisation of the procedure is given in Figure 4.2.

4.3.5 Rational

In both hypothesis tests we made reference to an Achieved Significance Level (ASL), this is a measure of the rate of evidence against H_0. We shall use the guidelines given in Table 4.1 as a basis for any decisions made.

$\text{ASL} < 0.10$	Evidence against H_0
$\text{ASL} < 0.05$	Reasonably strong evidence against H_0
$\text{ASL} < 0.025$	Strong evidence against H_0
$\text{ASL} < 0.01$	Very strong evidence against H_0

Table 4.1: Rate of evidence against H_0 as given in Efron [1, p.204]

We also note some fundamental subtleties in the above methodology. In our second hypothesis test, H_0 is that the means are equivalent. We therefore require a distribution that estimates the population

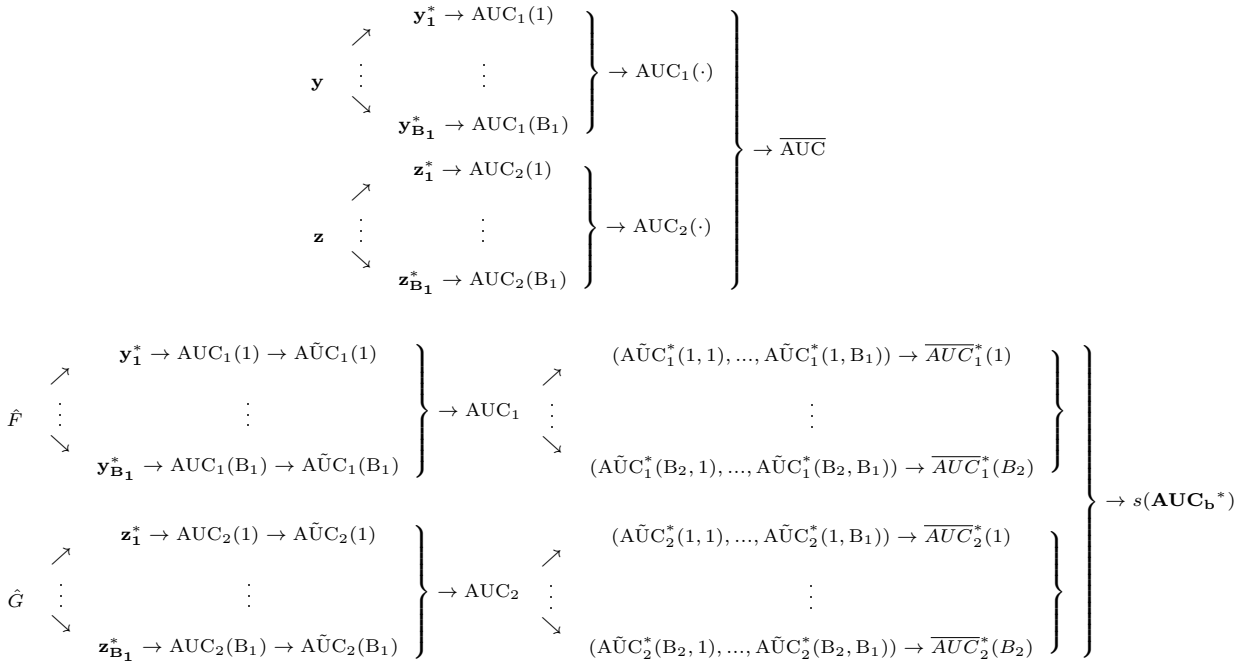

Figure 4.2: Method used to obtain $\mathrm{AUC}_1(\cdot)$, $\mathrm{AUC}_2(\cdot)$, $\overline{\mathrm{AUC}}$ used to evaluate $\tilde{\mathrm{AUC}}_1(i)$ and $\tilde{\mathrm{AUC}}_2(i)$ for $i = 1, 2, ..., B_1$ and how we then evaluate our statistic $s(\mathbf{AUC_b}^*)$.

under H_0. Neither the distribution $\hat{F}$ nor $\hat{G}$ satisfy this condition. Using the following translation $\tilde{\mathrm{AUC}}_1(i) = \mathrm{AUC}_1(i) - \mathrm{AUC}_1(\cdot) + \overline{\mathrm{AUC}}$, however, we are able to construct a null distribution for the data under H_0 [1, p223].

Also, we should note that in both of the above algorithms, it is assumed that $s_{obs} \geq 0$. For the case when $s_{obs} < 0$ we adjust $\hat{\mathrm{ASL}}_{\mathrm{boot}}$ to

$$\hat{\mathrm{ASL}}_{\mathrm{boot}} = \frac{\#\left\{s(\mathbf{AUC_b^*}) \leq s_{obs}\right\}}{B_2}$$

Using the above algorithms we therefore could attempt to use the methodology outlined in Section 4.3.2 to compare score cards. However, there are problems with this approach as we shall demonstrate with the following example.

4.3.6 Example

We consider the probabilities of default for two separate data sets each comprising of 100 individuals. The first data set, $\mathbf{y}$, is sampled from a $N(0.5, 0.1)$ distribution and the second data set, $\mathbf{z}$, is sampled from a $U(0.4, 0.6)$ distribution. If we compare histograms of the observations as in Figure 4.3, it might be considered obvious that the two distributions are not the same. However, when implementing our algorithm to compare distributional forms, our hypothesis test provides a ASL of 0.199 and therefore finds no significant evidence that the distributions are different [Appendix 7.6]. We obtain similar problems when using our actual data set.

Let's consider fixing our two testing sets, such that the first testing set is taken from the first 25% of the data and the second data set is taken from the last 25%. We build just one score card using the same variables as in the previous chapter on the centre 50% of the data and measure its performance on both test data sets. Using just one model we find that the AUC when tested on the first 25% of the data is 0.7533052, however when it is tested on the last 25% of the data is only 0.5664593. This suggests that the model seems to perform far better on past data than future data. We implement our distributional form algorithm and find an Achieved Significance Level of 0.528, indicating that the data is from the same distribution, however the Achieved Significance Level when testing the equality of means is 0, which provides incredibly strong evidence that the means are different. The key point here is that we only built

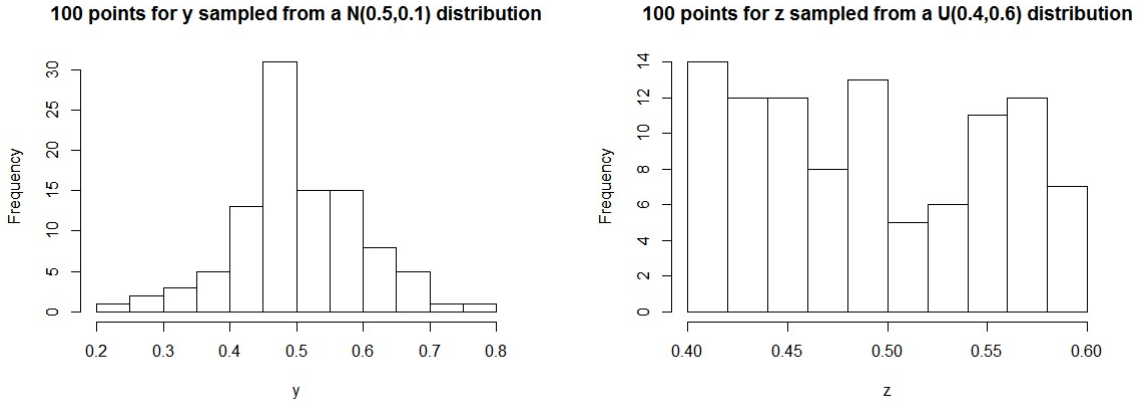

Figure 4.3: Histogram of 100 observations taken from a $N(0.5, 0.1)$ distribution and 100 points sampled from a $U(0.4, 0.6)$ distribution.

one model, so any differences in the AUCs must be due to differences in the distributional form of the test sets. As we can see there is a definite flaw in our method as we have neglected one key fact, that the data is time dependent.

This finding is particularly problematic. If we cannot fairly or accurately test whether the distribution of the two data sets are identical any further results are unreliable. We shall therefore consider a different method to bootstrap performance measures.

4.4 Bootstrap the difference in AUC

When discussing the Mann-Whitney method, we noted that there exists correlation between the values when comparing AUCs produced by models based on the same test data set. As we have demonstrated, it is naive to simply take two test data sets to try and avoid this problem. However, we can employ the bootstrap algorithm to provide a potential solution. We note that when we find percentile bootstrap confidence intervals we make no assumptions about the form of the standard error. The intervals are inferred directly from the distribution of the bootstrap replications. It is therefore possible to obtain estimates for our confidence intervals on the difference in AUC. We state this formally as follows [1, p.107]

1. Generate B independent bootstrap testing data sets $\mathbf{x_1^*}, \mathbf{x_2^*}, ..., \mathbf{x_B^*}$ and for $b = 1, ..., B$ compute $AUC_1^*(b)$ and $AUC_2^*(b)$
2. Calculate $d_b = AUC_1^*(b) - AUC_2^*(b)$
3. Order these differences and define $d_B^{(\alpha)}$ to be the $100 \cdot \alpha$th empirical percentile of the d_b values. Similarly, we define $d_B^{(1-\alpha)}$ to be the $100 \cdot (1-\alpha)$th empirical percentile. The $1 - 2\alpha$ percentile interval is then defined as
$$\left[d_B^{(\alpha)}, d_B^{(1-\alpha)}\right]$$

We then consider the hypothesis H_0: The difference is zero vs. H_0: the difference is not zero. If the value 0 is not within our confidence intervals then we can reject the null hypothesis at significance level α.

4.4.1 Difference between models is potentially insignificant

We consider two identical models, which differ only by one term, corresponding to a random variable, r_i, where $r_i \backsim N(0, 0.1)$. We might expect that this extra random variable will do little to change the model in any significant manner. The other variables are: the first age quintile, the second and third months at address quintile, and whether the person is a home owner or a council tenant. We obtain the following results [Appendix 7.8]:

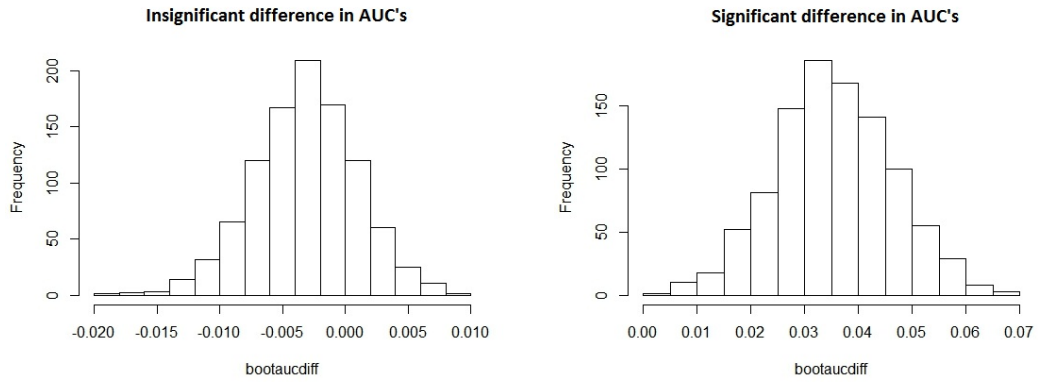

Figure 4.4: Histogram for an statistically insignificant and significant differences between AUCs.

- AUC from model 1 = 0.6613746
- AUC from model 2 = 0.6649872
- Difference in AUCs= -0.003612672
- 95% upper percentile Confidence Interval = 0.004792017
- 95% lower percentile Confidence Interval = -0.01156245

We can see that the confidence intervals contain zero and therefore we do not accept H_1.

4.4.1.1 Difference between models is significant

Here we consider the first model as above, however the second model has just one variable, whether the person is a home owner. We obtain the following results [Appendix 7.7]:

- AUC from model 1 = 0.6734853
- AUC from model 2 = 0.638587
- Difference in AUCs= 0.03489835
- 95% upper percentile Confidence Interval = 0.05669581
- 95% lower percentile Confidence Interval = 0.0142205

Since zero is not included in our confidence intervals, the difference between these two models AUC is statistically significant. We therefore reject H_0 with significance level α. A histogram of our bootstrap replications for both examples is given in Figure 4.4.

As we made clear when comparing AUCs generated from a model tested on the first and last quarter of the data, the time dependence is incredibly important. We shall now consider how we might employ the bootstrap methodology for time dependent data with a specific focus on homogeneous populations.

Chapter 5

Homogeneous Time Dependent Data

5.1 Introduction

As we mentioned in the previous chapter, we have thus far neglected the key time dependent element of our data. We demonstrated that if we build a model on the middle half of the data and then test its performance on the first and last quarter we obtain very different performance measures. We shall start by investigating the nature of the default rates.

It is important to emphasise that this study assumes a homogeneous, or equivalent, population. There is also the problem of right-censored data, due to a considerable number of accounts being closed prematurely, or failing to default before our study ends at relative month 41. We will later consider survival analysis which deals with this specific issue.

We consider the month the account was opened as a time stamp and we can see in Figure 5.1 that if we divide the data up into blocks of 5%, accounts which were opened earlier in the data set, were on average less likely to result in a default.

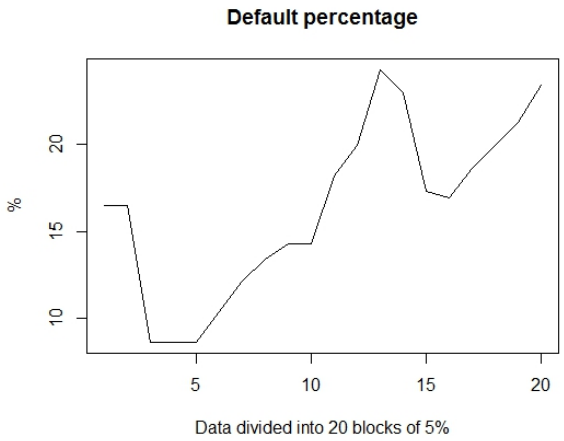

Figure 5.1: Relative number of defaults dividing the data set up by month or in blocks of 5%

We might take the view that the default rates are not simply just a random sample from a distribution, arguing that there is too much structure. If we assume that on average the individuals that make up each month are roughly equivalent, then the changes in the default rate are, at least in part, as a result of changes in the macro-economic environment.

We consider whether it is possible to model this default percentage as a time series. Using the *ar* command in R it can be shown that it fits a first order autoregressive model to the data, we'll therefore use this as the basis of our study.

5.2 Autoregressive model theory

A first order autoregressive model, denoted AR(1), satisfies the following difference equation

$$Y_t = c + \phi Y_{t-1} + \varepsilon_t$$

where $\{\varepsilon_t\}$ is a white noise process, such that its mean is zero, its variance is σ^2 and the ε's are uncorrelated across time [14, p.47].

We say that $\{Y_t\}$ is stationary if both the mean and variance are independent of t and $E\{Y_{t_1}, Y_{t_2}\}$ is a function of the absolute distance $|t_2 - t_1|$ only. Similarly for the covariance between Y_{t_1} and Y_{t_2}.

For a discrete time stationary process define the autocovariance sequence s_τ as $s_\tau = \operatorname{cov}\{X_t, X_{t+\tau}\}$, where τ is called the lag. We also define the autocorrelation sequence by

$$\rho_\tau = \frac{s_\tau}{s_0} = \frac{\operatorname{cov}\{X_t, X_{t+\tau}\}}{\sigma^2}$$

here $s_0 = \operatorname{cov}\{X_t, X_t\} = \operatorname{var}\{X_t\}$.

We are able to write our difference equation as

$$\begin{aligned} Y_t &= c + \phi Y_{t-1} + \varepsilon_t \\ &= c + \phi\left(c + \phi Y_{t-2} + \varepsilon_{t-1}\right) + \varepsilon_t \\ &= \phi^2 Y_{t-2} + \phi\left(c + \varepsilon_{t-1}\right) + \left(\varepsilon_t + c\right) \\ &= \vdots \\ &= \textstyle\sum_{k=0}^{\infty} \phi^k \left(c + \varepsilon_{t-k}\right) \\ &= \frac{c}{1-\phi} + \textstyle\sum_{k=0}^{\infty} \phi^k \varepsilon_{t-k} \end{aligned}$$

Note $E(Y_t) = \mu = \frac{c}{1-\phi}$ and $\operatorname{var}(Y_t) = E(Y_t - \mu)^2 = E\left(\sum_{k=0}^{\infty} \phi^k \varepsilon_{t-k}\right)^2 = \sigma^2 \sum_{k=0}^{\infty} \phi^{2k} = \frac{\sigma^2}{1-\phi^2}$. For $\operatorname{var}(Y_t) < \infty$ we therefore require $|\phi| < 1$.

It can also be shown that $\rho_\tau = \phi^{|\tau|}$ for $\tau = 0, \pm 1, \pm 2$ [14, p.54] and thus obtain an exponentially declining autocovariance sequence. We shall now consider how we might fit an autoregressive model to a time series.

5.3 First order autoregressive model simulation

Let's consider the following data in Table 5.1 which we might wish to fit a AR(1) model to.

Period	Score	Period	Score	Period	Score	Period	Score	Period	Score
1	0.000	11	0.091	21	-0.157	31	0.006	41	0.197
2	-0.074	12	0.122	22	-0.071	32	0.027	42	0.203
3	-0.300	13	0.026	23	0.075	33	0.156	43	0.094
4	-0.340	14	-0.003	24	0.068	34	0.083	44	0.094
5	-0.175	15	0.008	25	0.097	35	0.196	45	0.164
6	-0.007	16	0.001	26	0.236	36	0.190	46	0.350
7	-0.075	17	-0.125	27	0.079	37	0.058	47	0.331
8	-0.021	18	-0.179	28	0.009	38	0.119	48	0.201
9	0.020	19	-0.094	29	0.202	39	0.130	49	0.275
10	0.082	20	-0.093	30	0.066	40	0.245	50	0.117

Table 5.1: Average credit scores over 50 periods

We shall demonstrate how we might estimate our parameter using least squares estimation as described in Efron [1, p.94].

Defining y_t as the realisation of Y_t we begin by estimating $E(Y_t) = \mu$ using $\bar{y}$ (0.05412222 for our data) and then setting $x_t = y_t - \bar{y}$, such that x_t is the realisation of the first-order autoregressive process X_t. If we let φ be our guess for the true value of ϕ. We then define the residual squared error (RSE) as

$$\operatorname{RSE}(\varphi) = \sum_{i=2}^{N} \left(x_t - \varphi x_{t-1}\right)^2$$

The residual squared error then achieves its minimum when φ is close to the true value of ϕ, so we estimate our value of ϕ as

$$\text{RSE}(\hat{\phi}) = \min_{\varphi} \text{RSE}(\varphi)$$

For an AR(1) process this is computed as

$$\hat{\phi} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{x}$$

where $\mathbf{x} = (x_2, x_3, ..., x_N)$ and $\mathbf{X} = (x_1, x_2, ..., x_{N-1})$.

We shall demonstrate how bootstrapping residuals or block bootstrapping can be used to determine how accurate our estimate $\hat{\phi}$ is.

5.3.1 Bootstrapping residuals

Let's assume our probability distribution F and ϕ are unknown, however our mean μ is known and equal to $\bar{y}$. We begin by computing what Efron [1, p.95] defines as *approximate disturbances*

$$\hat{\varepsilon}_t = x_t - \hat{\phi}x_{t-1}$$

If we let $N =$ Total number of observations, we have $N = 50$ for our example. We therefore estimate F using the empirical distribution $\hat{F}$ where this puts a probability $\frac{1}{N-1}$ on each of the points $\varepsilon_2, ..., \varepsilon_N$.

In order to implement our bootstrap algorithm we begin with an initial value $x_1 = y_1 - \bar{y}$ which is treated as a fixed constant. We then calculate the bootstrap time series as

$$\begin{aligned} x_2^* &= \hat{\phi}x_1 + \varepsilon_2^* \\ x_3^* &= \hat{\phi}x_2^* + \varepsilon_3^* \\ &\vdots \quad\quad \vdots \\ x_N^* &= \hat{\phi}x_{N-1}^* + \varepsilon_N^* \end{aligned}$$

where our ε_t^* values are randomly sampled with replacement from $(\varepsilon_2^*, ..., \varepsilon_N^*)$. This algorithm is repeated for some large number B to provide B estimates $\hat{\phi}^*$.

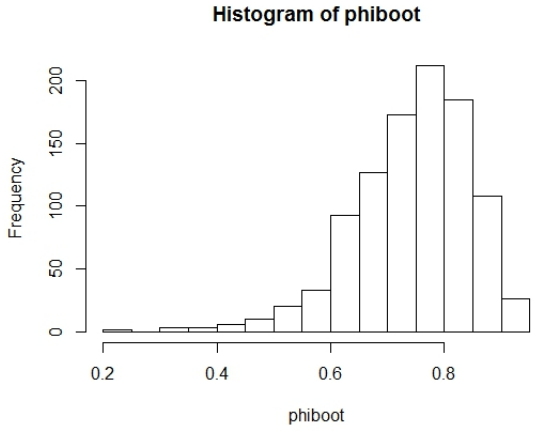

Figure 5.2: Histogram of the $B = 1000$ bootstrap replications of $\hat{\phi}$. Interestingly this appears to have a long lower tail and does not appear to be perfectly normal.

Using the original data as given in Table 5.2 we estimate $\hat{\phi}$ as 0.770174. On implementation of our algorithm with $B = 1000$ we estimate our percentile confidence intervals as [Appendix 7.10]

- 95% upper confidence interval = 0.9006665
- 95% lower confidence interval = 0.5088065

We also estimate the standard error for our bootstrapped estimates of $\hat{\phi}^*$ as 0.1007956. In Figure 5.2 we can see that the histogram does not appear to be normal. Lower values for ϕ would correspond to an autoregressive process where the random, white noise process has a greater influence. Taking the mean of our bootstrap sample for $\hat{\phi}$ gives a value of 0.7416875, below that estimated using the original data. However, this might not be considered surprising, given that such a small sample was provided for our autoregressive process.

We shall now consider block bootstrapping as an alternate method to estimate the accuracy of $\hat{\phi}$.

5.3.2 Block bootstrapping

Block bootstrapping is a method which samples the time series in blocks. Chernick explains that for stationary time series, successive observations are correlated but observations separated by a large time gap are uncorrelated [13, p100]. This can be seen by the exponentially declining autocorrelation function for a stationary AR(1) model. The autocorrelation function, shown in Figure 5.3, follows this approximate structure.

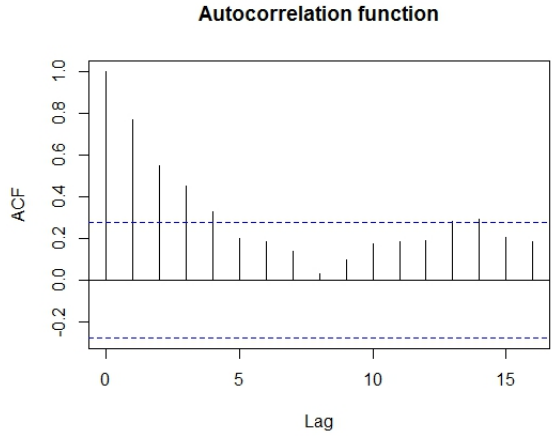

Figure 5.3: Autocorrelation function for our data

For this method, rather than fitting a model and then sampling from the residuals, we sample from the time series itself and then fit the autoregressive model [1, p.101]. However, unlike with the normal bootstrap methodology, instead of re-sampling individual data points we sample blocks of data points. This allows us to retain the time dependent nature of the data. We illustrate the general principle in Figure 5.4.

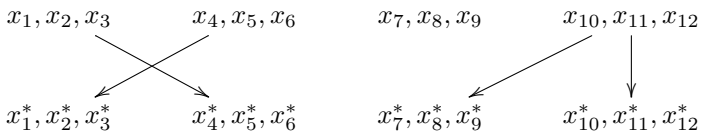

Figure 5.4: Block bootstrapping methodology. Specifically this is the non-overlapping block bootstrap method with block length 3.

There are 4 main block bootstrapping methods we shall describe. These are [13, p.104]

- Moving block bootstrap
- Non-overlapping
- Circular
- Stationary

The idea is to re-sample the data points in blocks such that we re-sample just enough blocks to obtain a series of roughly the same size as the original time series [1, p.101]. The block length l is chosen such that observations more than l units apart are independent. We therefore retain the correlation within the blocks. We shall provide an overview of each method.

5.3.2.1 Moving block

For the moving block bootstrap we choose k blocks of length l, such that $n \approx k \cdot l$, where it is possible for the blocks to overlap. Specifically for an original data set of size n, define $k = \text{floor}(\frac{n}{l})$ and $N = k \cdot l$. We then generate k blocks of size l to form a bootstrapped data set of size N. When estimating the standard error we must remember to multiply by $\sqrt{N/n}$ to adjust for the different lengths in the series [1, p.101]. We illustrate the method with the following example. Suppose we have data: $\mathbf{x} = (x_1, \ldots, x_{12})$ and wish to construct blocks of size 3, we begin by constructing 10 blocks of length 3: $\mathbf{y}_1 = (x_1, x_2, x_3)$, $\mathbf{y}_2 = (x_2, x_3, x_4)$, $\ldots$, $\mathbf{y}_{10} = (x_{10}, x_{11}, x_{12})$. We can then sample 4 of the blocks $\mathbf{y}_i$ randomly and put these together to form $\mathbf{x}^*$. This keeps some part of the dependence, however some is lost when we connect blocks. Furthermore, we can see that points at the beginning and end of the data series are less likely to be sampled.

5.3.2.2 Non-overlapping

This is similar to the moving block bootstrap, except that the blocks do not overlap. For example if we have data: $\mathbf{x} = (x_1, \ldots, x_{12})$ we can construct 4 blocks of length 3, $\mathbf{y}_1 = (x_1, x_2, x_3)$, $\mathbf{y}_2 = (x_4, x_5, x_6)$, $\mathbf{y}_3 = (x_7, x_8, x_9)$, $\mathbf{y}_4 = (x_{10}, x_{11}, x_{12})$ and then re-sample the blocks $\mathbf{y}_i$.

5.3.2.3 Circular

The circular block bootstrapping method is again similar to the moving block bootstrap however it periodically extends the series of points to form the blocks. For example if we have data: $\mathbf{x} = (x_1, \ldots, x_{12})$, in the normal moving block bootstrap we can only construct 10 blocks. In the circular block bootstrap we can construct 12 blocks of length 3 by periodically extending the data e.g. $\mathbf{y}_1 = (x_1, x_2, x_3)$, $\mathbf{y}_2 = (x_2, x_3, x_4)$, $\ldots$, $\mathbf{y}_{10} = (x_{10}, x_{11}, x_{12}), \mathbf{y}_{11} = (x_{11}, x_{12}, x_1), \mathbf{y}_{12} = (x_{12}, x_1, x_2)$. We can then sample the blocks $\mathbf{y}_i$. Unlike the moving block bootstrap and the non-overlapping bootstrap, the circular bootstrap method assigns equal probabilities to each of the original observations.

5.3.2.4 Stationary

Here we again periodically extend the data structure in a similar manner as in the circular block bootstrap. However, instead of using a fixed block length, the block length is given using the random length L, where

$$\Pr(L = j) = (1-p)^{j-1}p \quad \text{for } j = 1, 2, 3, ...$$

This follows the geometric distribution with parameter p. The mean block length for L is p^{-1}. We therefore choose p in the same manner as we choose a fixed block length [13, p.105] and select the start point of each block by sampling with replacement from $\{1, ..., n\}$. The blocks are then placed together to form our bootstrap replication [3, p.1304].

5.3.2.5 Implementing the block bootstrap algorithm

We now aim to implement the block bootstrapping algorithms, but first must select a block length l. From our autocorrelation sequence in Figure 5.3 we can see that for lag 5 or more the observations are approximately uncorrelated and therefore $l = 5$ might be a sensible choice for our block length. Furthermore for $l = 5$ in the moving block, non-overlapping and circular algorithms, we ensure that the block bootstrap replications are of the same size as the original data set. For the stationary block bootstrap we set $p = 0.2$ such that the mean block length is 5.

In Table 5.2 we find summary statistics for our block bootstrap replications. Our parameter, $\hat{\phi}$, using the original data was estimated as 0.770174. However, we see that in all four block bootstrapping cases we estimate our parameter $\hat{\phi}$ closer to 0.6. Furthermore, the Stationary and Moving Block algorithms

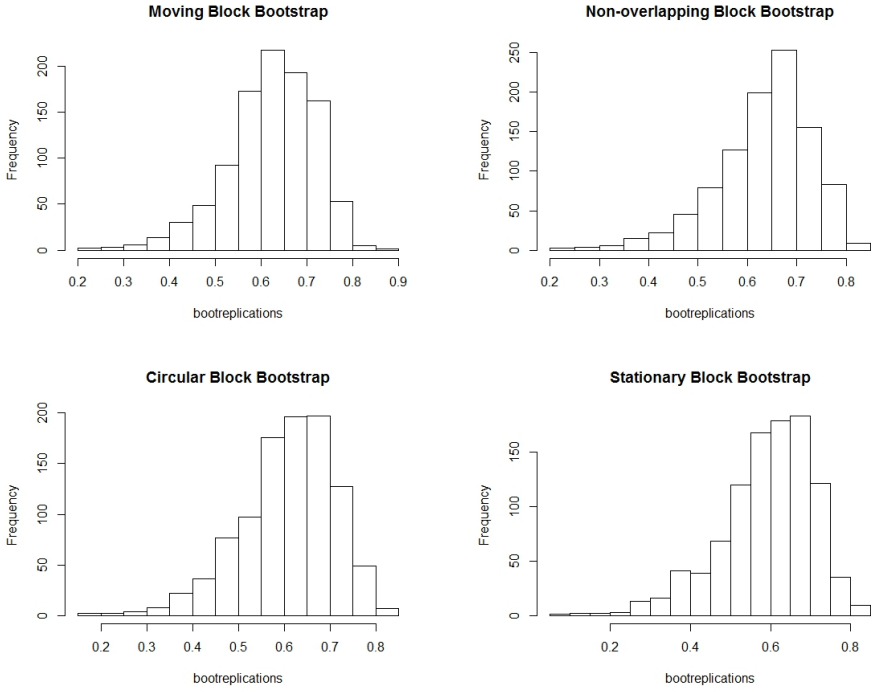

Figure 5.5: Histograms from our block bootstrap replications

	Standard Error	Upper CI	Lower CI	Mean
Moving Block	0.0952694	0.7672566	0.4011084	0.6217096
Non-overlapping	0.09758468	0.7788323	0.3936644	0.6338416
Circular	0.1029845	0.7730011	0.3683701	0.6080378
Stationary	0.1845657	0.7648305	0.3127508	0.5918667

Table 5.2: Summary statistics for our block bootstrap replications [Appendix 7.11 - 7.14].

do not include 0.770174 within the 95% confidence interval. This is likely to be as a result of the long tailed, non-normal shaped histograms for our bootstrap replications, as shown in Figure 5.5.

In actual fact the data in Table 5.1 was generated using $Y_t = 0.8Y_{t-1} + \varepsilon_t$ where $\varepsilon_t \sim N(0, 0.1)$ and initial condition $Y_1 = 0$ [Appendix 7.9]. Using the inbuilt $ar(\cdot)$ function in R we obtained the following estimates

- Coefficient estimate: 0.7673
- Standard error: 0.09447

Chernick describes some of the problems of block based bootstrapping methods, in particular that the re-sampled blocks do not quite mimic the behavior of the time series and that they have a tendency to weaken the dependency in the series. A potential solution is to re-sample blocks of blocks [13, p.105]. This is described in detail in Reference 5.

An overall problem with either of the above methods is that we need to know which model to fit to the data beforehand. Without considering more, higher order autoregressive models we cannot accurately say that this model is appropriate.

We shall now consider the case of fitting an autoregressive model to our default rates as shown in Figure 5.1.

5.4 Default Rates autoregressive model

We shall use the default rates when the data is divided into 5% blocks. A breakdown for these values is given in Table 5.3.

Percentile	Default Rate (%)	Percentile	Default Rate (%)
5%	16.88	55%	17.32
10%	16.45	60%	20.35
15%	8.66	65%	24.24
20%	8.66	70%	23.81
25%	8.23	75%	16.45
30%	10.39	80%	17.32
35%	12.55	85%	18.61
40%	12.99	90%	19.48
45%	12.99	95%	21.21
50%	15.58	100%	23.38

Table 5.3: Realised default rates for our data divided into blocks of 5%

We estimate $\hat{\phi}$ as 0.8608599 and start by bootstrapping the residuals. With $B = 1000$ we estimate our percentile confidence intervals as

- 95% upper confidence interval = 1.033468
- 95% lower confidence interval = 0.3993687

We plot our histogram for bootstrap replications of $\hat{\phi}$ in Figure 5.6 and similarly to what was obtained previously, the distribution appears relatively non-normal, with a long lower tail. We estimate our standard error as 0.1661599 and the mean of our bootstrap sample for $\hat{\phi}$ gives a value of 0.8034312, which is again below that estimated using the original data. Interestingly the 95% confidence interval has an upper bound greater than 1. For a stationary autocorrelation function we require $|\phi| < 1$ and therefore we cannot assume that if an autoregressive model is appropriate for our default rates, that this process is stationary.

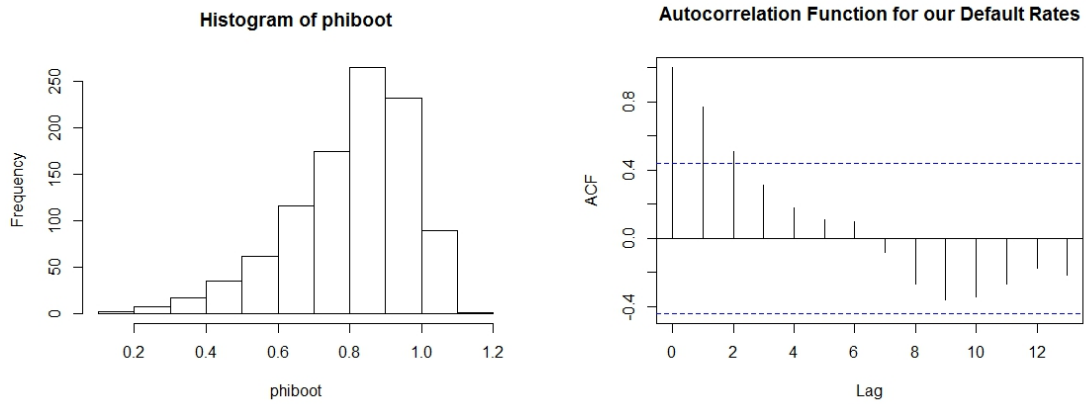

Figure 5.6: Histogram of our bootstrap replications of $\hat{\phi}$ for $B = 1000$ when bootstrapping residuals for our Default Rates and the corresponding autocorrelation function for our data.

We shall now consider block bootstrapping as an alternate method to estimate the accuracy of $\hat{\phi}$. We consider the moving block, non-overlapping, circular and stationary block bootstrap techniques. From our autocorrelation function as shown in Figure 5.6 we can see that after approximately lag 5 the observations appear to be uncorrelated and therefore we shall use a block length of 5. For the stationary block bootstrap we set $p = 0.2$ such that the mean block length is 5. We might also like to note that

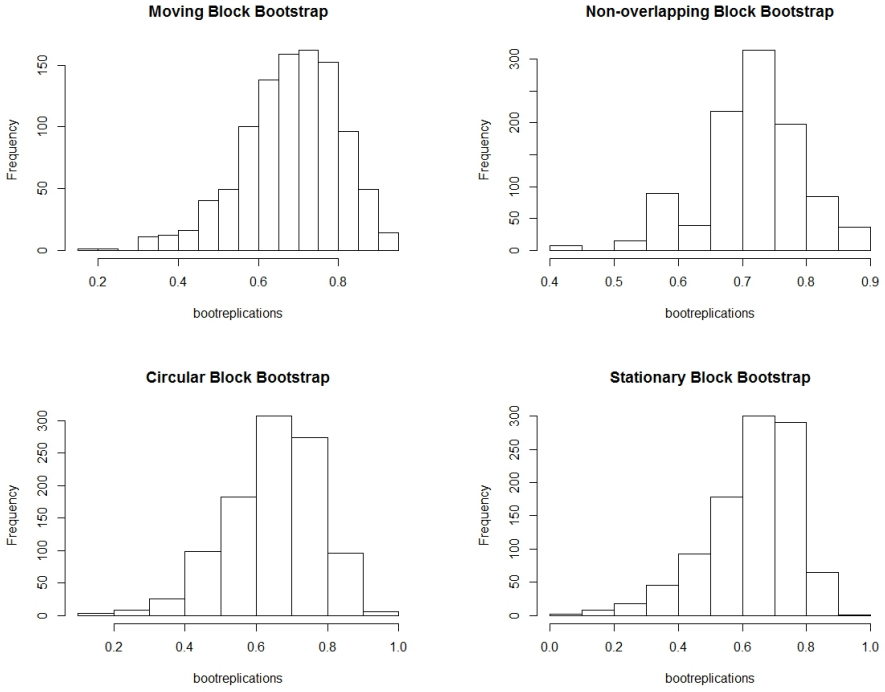

Figure 5.7: Histograms from our block bootstrap replications

compared to our previous example the confidence intervals for our autocorrelation function are wider, as shown by the dotted line in Figure 5.6.

	Standard Error	Upper CI	Lower CI	Mean
Moving Block	0.1213944	0.88054	0.4055437	0.6808143
Non-overlapping	0.08035733	0.8601131	0.5517897	0.7171497
Circular	0.1295804	0.8601131	0.3514478	0.6485303
Stationary	0.1400218	0.8355515	0.2824104	0.630226

Table 5.4: Summary statistics for our block bootstrap replications.

In Table 5.4 we find summary statistics for our block bootstrap replications. Our parameter, $\hat{\phi}$, using the original data was estimated as 0.8608599. However, we see that in all four block bootstrapping cases we estimate our parameter $\hat{\phi}$ closer to 0.65. In addition the Stationary, Non-overlapping and Circular algorithms do not include 0.8608599 within the 95% confidence interval. This is likely to be as a result as the long tailed, non-normal shaped histograms for our bootstrap replications.

If we fit an autoregressive model to the process using the inbuilt function in R we obtain the following estimates

- Coefficient estimate: 0.7707
- Standard error: 0.032955

The above methodology demonstrates how we might model default rates as an autoregressive process. However, we must note that this interpretation is only meaningful if the characteristics of lenders are roughly equivalent. In other words, the above requires that the individuals which make up our population to be homogeneous. As we know this is not necessarily the case.

Furthermore, there exists more problems when calculating the default rates in this manner. Our method to determine the relative time when the accounts defaulted was to time stamp them using the opening month. A potential problem is that some of our accounts were closed early. This is referred to as right censoring. Lyn Thomas [7, p.251] provides an indication about the importance of survival analysis

within the context of right censored credit scoring data. It is with this in mind, that we shall consider the Kaplan Meier Estimate of the survivor function, $S(t)$.

5.5 Survival Analysis

Continuing on the path of assuming the individuals within the population are roughly homogeneous, we can introduce non-parametric methods to study the survival function for our data [7, p.253]. We shall start by introducing the *survivor function*, $S(t)$, which is defined such that

$$S(t) = P(T > t)$$

In the context of credit scoring, we note that the probability of a loan surviving until time t is equivalent to the person being classed as *good*. Therefore our survivor function is such that $S(t) = P_G(t)$, where for ease of notation, we denote $\Pr(\text{good}|\text{data } \mathbf{q} \text{ at time t}) = P_G(t)$. We also define our *cumulative distribution function*, $F(t)$, such that

$$F(t) = P(T \leq t)$$

Therefore $F(t) = 1 - S(t) = 1 - P_G(t) = P_B(t)$, with the usual definition of the *probability density function* as $f(t) = \frac{d}{dt}F(t)$. We also introduce a *hazard function*, $\mu(t)$, such that

$$\mu(t) = \lim_{h \to 0} \frac{P(T \leq t + h | T > t)}{h}$$

We therefore interpret this as the instantaneous default rate for an individual who has not defaulted up to time t. We also have that $\mu(t) = -\frac{d}{dt}\log(S(t)) = -\frac{d}{dt}\log(P_G(t))$. Defining the *cumulative hazard rate* as

$$M(t) = \int_0^t \mu(s)ds$$

it therefore follows that $S(t) = P_G(t) = \exp(-M(t))$. Finally the log-odds score which was previously defined as $score(\mathbf{q}) = \log\left(\frac{P_G(t)}{P_B(t)}\right)$, is such that

$$score(\mathbf{q}) = -\log\left(\exp\left(M(t)\right) - 1\right)$$

For our data set we have a number of records where the accounts are closed prior to observing a default. We aim to estimate our survivor function, based on our data in the presence of right-censoring using the Kaplan-Meier estimate.

5.5.1 Kaplan-Meier Estimate

Suppose we have n independently and identically distributed individuals who take out a credit product, but due to the presence of right-censoring we only observe r defaults. If we let $t_1 < t_2 < ... < t_k$ be the ordered default times, with $k \leq r$ so that it's possible for more than one account to default at the same time, then we define d_j to be the number of defaults that occur at t_j, such that $\sum_{j=1}^{k} d_j = r$. We define c_j to be the censoring time within the interval $[t_j, t_{j+1})$, such that the numbered of censored observations is $n - r$. We define

$$n_j = n - \sum_{i \leq j-1} c_i - \sum_{i \leq j-1} d_i$$

to be the number of loans still outstanding at time t_j. The maximum likelihood estimate for our survivor function is defined as

$$\hat{P}_G(t) = \hat{S}(t) = \prod_{j:t_j \leq t} \left(1 - \frac{d_j}{n_j}\right) \tag{5.1}$$

We also define the maximum likelihood estimate of the hazard function μ_j to be $\hat{\mu}_j = \frac{d_j}{n_j}$. An estimate of the standard error of the Kaplan-Meier estimate is given as

$$se\left\{\hat{P}_G(t)\right\} = se\left\{\hat{S}(t)\right\} = \hat{S}(t)\sqrt{\sum_{t_j \leq t} \frac{d_j}{n_j\left(n_j - d_j\right)}} \tag{5.2}$$

which is known as *Greenwood's formula* [18]. Using this we are able to apply the central limit theorem as $n \to \infty$ to construct confidence intervals. We plot our Kaplan-Meier estimate for our data in Figure 5.8.

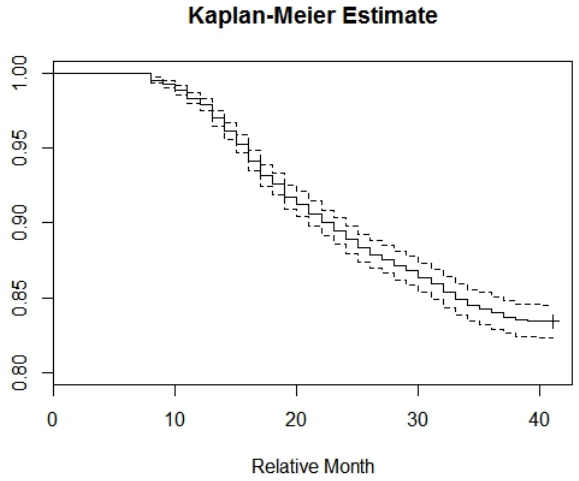

Figure 5.8: Kaplan-Meier estimate. Confidence intervals for the empirical survivor function are shown as dotted lines.

Akrita [15] outlines two different methods to bootstrap a Kaplan-Meier estimate, citing the method proposed by Efron [18] as providing asymptotically correct confidence intervals. We shall bootstrap the standard error and compare that with the result obtained using Greenwood's formula.

5.5.2 Bootstrapping the Kaplan-Meier estimate

We construct a Kaplan-Meier estimate for our data and compare its standard error with that obtained by Greenwood's formula. Suppose our data set $\mathbf{x}$ is such that it is drawn from an unknown probability distribution F and $\mathbf{x} = \{(x_1, D_1), ..., (x_n, D_n)\}$ where x_i corresponds to the month an individual either defaults or is censored and where we define $D_i = 1$ if individual i defaults and $D_i = 0$ if they are right-censored.

Efron [12, p.64] begins by making the following - arguably unrealistic - assumptions about the data. We suppose that the real lifetime X_i^0 of each credit product is selected randomly according to the survival curve

$$S(t) = \Pr\{X_i^0 > t\}$$

and a censoring time W_i is independently selected according to another survival curve

$$R(t) = \Pr\{W_i > t\}$$

We then observe $X_i = \min\{X_i^0, W_i\}$ and

$$D_i = \begin{cases} 1 & X_i = X_i^0 \\ 0 & X_i = W_i \end{cases}$$

The true survivor function $S(t)$ is then given as the product of the survivor curves for the censored and uncensored observations, denoted $R(t) = \Pr(W > t)$ and $S^0(t) = \Pr(X^0 > t)$, respectively.

A method to bootstrap the randomly censored data is to independently sample X_i^{0*} from $\hat{S}^0$ and W_i^* from $\hat{R}$, and define $X_i^* = \min(X_i^{0*}, W_i^*)$. A more intuitive method, which avoids the above assumptions, is to directly sample pairs of data points (x_i, D_i) from $\mathbf{x} = \{(x_1, D_1), ..., (x_n, D_n)\}$. In actual fact, Efron [18, p.314] demonstrates that both methods yield the same result. Our algorithm then proceeds as follows

1. Select B independent bootstrap samples $\mathbf{x_1^*}, \mathbf{x_2^*}, ..., \mathbf{x_B^*}$, each consisting of n pairs drawn with replacement from the observed data set $\mathbf{x} = \{(x_1, D_1), ..., (x_n, D_n)\}$.

2. Evaluate the bootstrap replication corresponding to each bootstrap sample

$$\hat{S}_b^*(t) = \prod_{j:t_j \leq t} \left(1 - \frac{d_j^*}{n_j^*}\right) \text{ for } b = 1, 2, ..., B$$

3. Estimate the standard error by the sample standard deviation of the B replications

$$\hat{se}_B(t) = \left(\frac{\sum_{b=1}^{B} \left\{ \hat{S}_b^*(t) - \hat{S}^*(t) \right\}^2}{B-1} \right)^{\frac{1}{2}}$$

where $\hat{S}^*(t) = \frac{1}{B} \sum_{b=1}^{B} \hat{S}_b^*(t)$ and

Here $t = 8, ..., 40$ corresponds to the relative months that the accounts either defaulted or were right-censored. We compare our bootstrapped standard error using $B = 100$ with that obtained using Greenwood's formula in Table 5.5 [Appendix 7.15-7.16].

$t =$	10	15	20	25	30	35	40
$\hat{S}(t)$	0.989	0.953	0.913	0.883	0.863	0.843	0.834
$\hat{se}_{Boot}$	0.001470	0.003241	0.004337	0.004708	0.005348	0.005629	0.005489
$\hat{se}_{Green}$	0.001547	0.003110	0.004145	0.004725	0.005071	0.005395	0.005527

Table 5.5: The Standard Error of the Kaplan-Meier curve for our account data using $B = 100$.

As might be expected there is a small discrepancy in the values for the standard error produced using both estimates. Here we used a relatively small value for B. This was due to considerable computing constraints due to the large data set which we are dealing with. Nevertheless, the data in Table 5.5 does provide a demonstration of the power of the bootstrap algorithm.

One of the problems with the above methodology is that the data is not continuous and is grouped by month. A potential solution is to use what is referred to as the *actuarial assumption*. This assumes that the losses to censorship occur uniformly over the month. We therefore adjust our value n_j to $n_j^{'}$ using the following transition

$$n_j^{'} = n - \sum_{i \leq j-1} c_i - \sum_{i \leq j-1} d_i - \frac{c_j}{2}$$

Our maximum likelihood estimate for our survivor function is then defined as

$$\hat{S}^{'}(t) = \prod_{j:t_j \leq t} \left(1 - \frac{d_j}{n_j^{'}}\right)$$

So far we have assumed that the individuals within our study are independently identically distributions. However, we know this is not true. We could consider the case of an inhomogeneous population, defined by a series of explanatory variables. For our data set these would be the characteristics of each borrower, for example their age or employment status. Lyn Thomas [7, p.256] outlines how we might apply the Cox's Proportional Hazards model within the context of credit scoring, comparing this with logistic regression models [16]. This falls beyond the scope of this project yet remains is an interesting area of further research.

Chapter 6

Conclusion

6.1 Overview

We have demonstrated the power of the bootstrap methodology and some of its many applications within the context of credit scoring. Bootstrapping was used to generate confidence intervals to compare score cards and we have provided some indication of the advantages of such methods over the more conventional Mann-Whitney U Statistic. In particular, the simplicity of the bootstrap algorithm was emphasised. Two different methods to bootstrap AUC were considered, before concluding that the first method, where we build just one model and test its performance on B bootstrapped test data sets, is more appropriate for our purposes.

When comparing score cards we proposed a solution based on testing two different models on independent data sets, as shown in Figure 6.1. We divided this into two hypothesis tests, before demonstrating some of the ways in which this methodology breaks down.

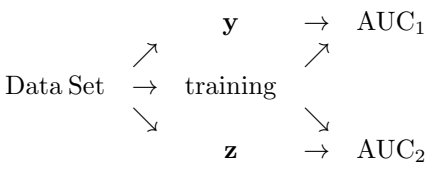

Figure 6.1: Proposed methodology to compare score cards

We also explored four different methods to bootstrap AUC, before concluding that the percentile bootstrap method is superior due to its invariance to transformations and its ability to adjust for correlation between score cards built on the same data set. Another interesting application of the bootstrap methodology is when estimating AUC in the case a limited data set.

The effect of the number of bootsamples on the estimate for the mean and standard error was also explored. Furthermore, the time dependent element of the credit scoring data set was highlighted by comparing the AUCs found by testing a model on the first and last quarter of the data set. This is known as backtesting and forecasting, respectively. The corresponding ROC curves are shown in Figure 6.2.

We moved on to study how we might fit a first order auto regressive model to default rates, before analysing two different bootstrapping methods. The first bootstrapping residuals and the second using the block bootstrapping methods. We concluded by researching elements of survival analysis and how we might bootstrap the Kaplan Meier curve within the context of credit scoring. A comparison between the bootstrap estimate for the standard error and that obtained using Greenwood's formula were found to be relatively similar.

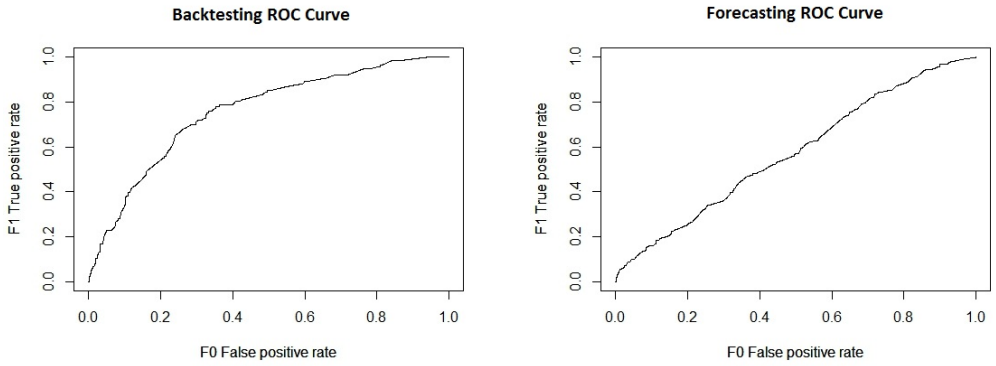

Figure 6.2: AUC when backtesting is 0.7533052 but only 0.5664593 when forecasting.

6.2 Discussion

The presented report highlights just some of the different ways in which the algorithms were tested and implemented. It was originally believed by the author that testing two different models on different data sets could have provided an innovative solution to the problem of comparing two different models AUC. It was only after the algorithms were implemented that the method broke down. This led to the consideration of default rates.

Simulated data also was used prior to implementing a number of the algorithms. This has the advantage that we can control the parameters and therefore compare the expected result with the actual result. We started by considering an environment where

$$y_{i,t} = \alpha + \beta x_i + \gamma e_i + k_t + \varepsilon_i$$

Here α, β and γ are constants and $y_{i,t}$ is the time dependent probability of default. We define $x_i \sim N(0,1)$, $e_i \sim N(0, \sigma_e^2)$, $\varepsilon_i \sim N(0, \sigma_\varepsilon^2)$, where σ_e and σ_ε are some unknown standard deviations. The last term, ε_i, is the unknown error term and k_t represents changes in the macro-economic environment.

We can find the expected value of $y_{i,t}$ as follows:

$$E(y_{i,t}) = E(\alpha + \beta x_i + \gamma e_i + k_t + \varepsilon_i) = \alpha + \beta E(x_i) + \gamma E(e_i) + k_t$$

Different functions for k_t were considered, for example, a step function. This aimed to model a situation where there is a sudden change in the macro-economic environment, such as the failure of a major bank. This work provided the basis on which modeling default rates using a first-order autoregressive process was based and prompted the study of block bootstrapping methods.

6.3 Further Work

Much more work is still to be done exploring the bootstrap methodology. With more time we could fully develop some of the time dependent models. In particular, we could develop a Cox's Proportional Hazards model, for the case of an inhomogeneous population. It would also be interesting to implement the Mann-Whitney U Statistic and compare the results obtained with that obtained using bootstrapping.

We might also like to build on some of the work relating to survival analysis. Whilst it is clear we can model default rates using the Kaplan Meier estimate, it would be interesting to demonstrate some of the reasons why this might be useful.

Chapter 7

Appendix

A selection of some of the more important R code is included below. This is by no means an exhaustive list of all computing code used for this report.

7.1 One Sample Bootstrap algorithm

```
bootstrap <- function(x,B) {
        bootreplications<-NULL
        n=length(x)
        stat = mean(x)
        #Calculate the statistic of interest
        cat("Mean␣of␣original␣sample=",stat,"\n")
        for (i in 1:B) {
                bootsample <- sample(x,replace="TRUE")
                bootreplications[i] <- mean(bootsample)
        }
        meanbootstrap = mean(bootreplications)
        cat("Estimated␣bootstrap␣mean␣=",meanbootstrap,"\n")
        #Display the estimated mean for the statistic
        varbootstrap = var(bootreplications)
        #calculate the variance of the bootstrap replications
        sebootstrap = sqrt(varbootstrap)
        #calculate the standard error of the bootstrap replications
        cat("Estimated␣bootstrap␣standard␣error␣=",sebootstrap,"\n")
        #Display the estimated standard error for the statistic
        hist(bootreplications)
        hist(x)
}
x <- rnorm(25,0,1)
bootstrap(x,200)
```

7.2 Building a basic credit score card

```
x <- NULL
y <- NULL
yp1 <- NULL
yp2 <- NULL
yp3 <- NULL
setwd("C:/Users/Luke/Dropbox/Dissertation_new")
ccdata<-read.delim("ordered_data_cc_app.txt")
attach(ccdata)
ccdata$EMPLOY_STATUS_EM <- as.numeric((EMPLOY_STATUS=="EM"))
ccdata$EMPLOY_STATUS_HO <- as.numeric((EMPLOY_STATUS=="HO"))
ccdata$EMPLOY_STATUS_RE <- as.numeric((EMPLOY_STATUS=="RE"))
ccdata$EMPLOY_STATUS_SE <- as.numeric((EMPLOY_STATUS=="SE"))
ccdata$EMPLOY_STATUS_ST <- as.numeric((EMPLOY_STATUS=="ST"))
ccdata$TENURE_CT <- as.numeric((TENURE=="CT"))
ccdata$TENURE_HO <- as.numeric((TENURE=="HO"))
ccdata$TENURE_LP <- as.numeric((TENURE=="LP"))
ccdata$TENURE_PT <- as.numeric((TENURE=="PT"))
ccdata$TENURE_B <- as.numeric((TENURE==""))
ccdata$APPLICATION_CHANNEL_C <- as.numeric((APPLICATION_CHANNEL=="Cold␣Calls"))
ccdata$APPLICATION_CHANNEL_I <- as.numeric((APPLICATION_CHANNEL=="Internet"))
ccdata$APPLICATION_CHANNEL_M <- as.numeric((APPLICATION_CHANNEL=="Mail"))
ccdata$APPLICATION_CHANNEL_O <- as.numeric((APPLICATION_CHANNEL=="Other"))
ccdata$ageq1 <- as.numeric((age<=26))
ccdata$ageq2 <- as.numeric((26<age & age<=33))
```

```
ccdata$ageq3 <- as.numeric((33<age & age<=35))
ccdata$ageq4 <- as.numeric((35<age & age<=42))
ccdata$ageq5 <- as.numeric((42<age & age<=81))
ccdata$MONTHS_AT_ADDRESS1 <- as.numeric((MONTHS_AT_ADDRESS<=24))
ccdata$MONTHS_AT_ADDRESS2 <- as.numeric((24<MONTHS_AT_ADDRESS & MONTHS_AT_ADDRESS<=54))
ccdata$MONTHS_AT_ADDRESS3 <- as.numeric((54<MONTHS_AT_ADDRESS & MONTHS_AT_ADDRESS<=91))
ccdata$MONTHS_AT_ADDRESS4 <- as.numeric((91<MONTHS_AT_ADDRESS & MONTHS_AT_ADDRESS<=125))
ccdata$MONTHS_AT_ADDRESS5 <- as.numeric((125<MONTHS_AT_ADDRESS & MONTHS_AT_ADDRESS<=780))
woe <- function(x,y) {
        nn<-y
        n1<-y
        woe<-y
        for (i in 1:length(x)) {
                nn[i]<-sum(x==x[i])
                n1[i]<-sum(y*(x==x[i]))
                if (n1[i]==0) {
                        nn[i]<-nn[i]+1
                        n1[i]<-1
                }
                if (n1[i]==nn[i]) {
                        n1[i]<-n1[i]-1
                }
        }
        woe<-log((nn-n1)/n1)
        woe
}
ccdata <- ccdata[MONTHS_AT_ADDRESS<428,]
### Remove Outliers
leng<-length(t(ccdata))/length(ccdata)
len<-length(ccdata)
roc <- function(y, s) {
        yav <- rep(tapply(y, s, mean), table(s))
        rocx <- cumsum(yav)
        rocy <- cumsum(1 - yav)
        area <- sum(yav * (rocy - 0.5 * (1 - yav)))
        x1 <- c(0, rocx)/sum(y)
        y1 <- c(0, rocy)/sum(1 - y)
        auc <- area/(sum(y) * sum(1 - y))
        print(auc)
}
roc_plot <- function(y, s) {
        yav <- rep(tapply(y, s, mean), table(s))
        rocx <- cumsum(yav)
        rocy <- cumsum(1 - yav)
        area <- sum(yav * (rocy - 0.5 * (1 - yav)))
        x1 <- c(0, rocx)/sum(y)
        y1 <- c(0, rocy)/sum(1 - y)
        auc <- area/(sum(y) * sum(1 - y))
        plot(x1,y1,"l", xlab ="FO␣False␣positive␣rate", ylab = "F1␣True␣positive␣rate")
        title(main="ROC␣Curve", xlab ="FO␣False␣positive␣rate", ylab = "F1␣True␣positive␣rate", font.main= 4)
}
detach(ccdata)
## Building our general linear model
attach(ccdata)
nu<-sample(leng, floor(leng*.25), replace=FALSE)
cctest <- ccdata[nu,]
cctrain <- ccdata[-nu,]
cctestback <- ccdata[1:floor(leng*.25),]
cctrainback <- ccdata[(floor(leng*.25)+1):leng,]
cctestfor <- ccdata[(leng-floor(leng*.25)+1):leng,]
cctrainfor <- ccdata[1:(leng-floor(leng*.25)),]
detach(ccdata)
attach(cctrain)
glm1.out <- glm(good ~ ageq1 + ageq2 + ageq3 + ageq4 + MONTHS_AT_ADDRESS1 + MONTHS_AT_ADDRESS2 +
MONTHS_AT_ADDRESS3 + MONTHS_AT_ADDRESS4 + TENURE_CT + TENURE_HO + TENURE_PT + EMPLOY_STATUS_EM +
EMPLOY_STATUS_HO + EMPLOY_STATUS_RE + EMPLOY_STATUS_SE + APPLICATION_CHANNEL_C + APPLICATION_CHANNEL_I +
APPLICATION_CHANNEL_M, family = binomial("logit"))
detach(cctrain)
attach(cctest)
yp1 <- predict(glm1.out, cctest, type="response")
OriginalAUC <- roc(good,yp1)
roc_plot(good,yp1)
detach(cctest)
summary(glm1.out)
```

7.3 Bootstrapping Confidence Intervals

```
B = 1000
B2 = 200
## Using the model in Appendix 7.2.2 we generate four different confidence intervals
bootauc1 <- NULL
bootauc2 <- NULL
```

```
Z<-vector("numeric")
se<-vector("numeric")
for (i in 1:B) {
        attach(cctest)
        nx<-sample(floor(leng*.25), floor(leng*.25), replace=TRUE)
        cctestboot <- cctest[nx,]
        detach(cctest)
        attach(cctestboot)
        yp1 <- predict(glm1.out, cctestboot, type="response")
        bootauc1[i] <- roc(good,yp1)
        detach(cctestboot)
        ## For bootstrap-t confidence intervals - create the second bootstrap layer
        for(j in 1:B2) {
                attach(cctestboot)
                nx1<-sample(floor(leng*.25), floor(leng*.25), replace=TRUE)
                cctestboot2 <- cctestboot[nx1,]
                detach(cctestboot)
                attach(cctestboot2)
                yp2 <- predict(glm1.out, cctestboot2, type="response")
                bootauc2[j] <- roc(good,yp2)
                detach(cctestboot2)
        }
        se[i]=sqrt(var(bootauc2))
        Z[i]<-(bootauc1[i]-OriginalAUC)/se[i]
}
meanbootstrap1 = mean(bootauc1)
#calculate the mean of the bootstrap replications
varbootstrap1 = var(bootauc1)
#calculate the variance of the bootstrap replications
sebootstrap1 = sqrt(varbootstrap1)
#calculate the standard error of the bootstrap replications
cat("Estimated␣bootstrap␣standard␣error␣for␣the␣model␣=",sebootstrap1,"\n")
#Display the estimated standard error for the statistic
cat("Mean␣of␣the␣bootstrap␣of␣the␣model␣=", meanbootstrap1,"\n")
#Display the mean of the bootstrap replications
hist(bootauc1)
#plot a histogram
stu <- qt(0.975,df=floor(leng*0.25)-1)
gaus<- qnorm(0.975)
#95% CI on the bootstrap
lower1 = OriginalAUC - stu * (sebootstrap1)
upper1 = OriginalAUC + stu * (sebootstrap1)
cat("95%␣upper␣student-t␣Confidence␣Interval␣for␣the␣model␣=",upper1,"\n")
cat("95%␣lower␣student-t␣Confidence␣Interval␣for␣the␣model␣=",lower1,"\n")
#95% CI on the bootstrap
lower2 = OriginalAUC - gaus * (sebootstrap1)
upper2 = OriginalAUC + gaus * (sebootstrap1)
cat("95%␣upper␣Gaussian␣Normal␣Confidence␣Interval␣for␣the␣model␣=",upper2,"\n")
cat("95%␣lower␣Gaussian␣Normal␣Confidence␣Interval␣for␣the␣model␣=",lower2,"\n")
## Confidence intervals using bootstrap-t
Z<-sort(Z)
k<-ceiling(0.025*(B+1))
t1<-Z[k]
t2<-Z[B+1-k]
upper3 = OriginalAUC - t1 * (sebootstrap1)
lower3 = OriginalAUC - t2 * (sebootstrap1)
cat("Lower␣bootstrap-t␣Confidence␣Interval␣for␣the␣model␣=",lower3,"\n")
cat("Upper␣bootstrap-t␣Confidence␣Interval␣for␣the␣model␣=",upper3,"\n")
## Confidence intervals using percentiles
bootaucsort<-sort(bootauc1)
k<-ceiling(0.025*(B+1))
p1<-bootaucsort[k]
p2<-bootaucsort[B+1-k]
upper4 = p2
lower4 = p1
cat("Lower␣percentile␣Confidence␣Interval␣for␣the␣model␣=",lower4,"\n")
cat("Upper␣percentile␣Confidence␣Interval␣for␣the␣model␣=",upper4,"\n")
```

7.4 Number of Bootsamples

```
K=1000
bootauc2 <- matrix(data=0,nrow=K,ncol=K)
bootmean<-NULL
bootse<-NULL
for (B in 1:K) {
        bootauc1 <- NULL
        for (i in 1:B) {
                attach(cctest)
                nx<-sample(floor(leng*.25), floor(leng*.25), replace=TRUE)
                cctestboot <- cctest[nx,]
                detach(cctest)
                attach(cctestboot)
```

```
                yp1 <- predict(glm1.out, cctestboot, type="response")
                bootauc1[i] <- roc(good,yp1) detach(cctestboot)
        }
        bootauc2[1:B,B] <- bootauc1
        bootmean[B] <- mean(bootauc1)
        bootse[B] <- sqrt(var(bootauc1))
}
plot(bootmean,y,"l")
plot(bootse,y,"l")
```

7.5 Limited data set

```
sam <- sample(leng, 1000, replace=FALSE)
ccdata <- ccdata[sam,]
## Reducing the data set size
leng<-length(t(ccdata))/length(ccdata)
len<-length(ccdata)
## Model
attach(ccdata)
glm1.out <- glm(good ~ ageq1 + ageq2 + ageq3 + ageq4 + MONTHS_AT_ADDRESS1 + MONTHS_AT_ADDRESS2 +
MONTHS_AT_ADDRESS3 + MONTHS_AT_ADDRESS4 + TENURE_CT + TENURE_HO + TENURE_PT + EMPLOY_STATUS_EM +
EMPLOY_STATUS_HO + EMPLOY_STATUS_RE + EMPLOY_STATUS_SE + APPLICATION_CHANNEL_C + APPLICATION_CHANNEL_I +
APPLICATION_CHANNEL_M, family = binomial("logit"))
yp1 <- predict(glm1.out, ccdata, type="response")
OriginalAUC <- roc(good,yp1)
roc_plot(good,yp1)
detach(ccdata)
B = 1000
bootauc1 <- NULL
bootauc2 <- NULL
for (i in 1:B) {
        attach(ccdata)
        nx<-sample(leng, leng, replace=TRUE)
        ccdataboot <- ccdata[nx,]
        cctestboot <- ccdataboot[1:floor(leng*.25),]
        cctrainboot <- ccdataboot[(floor(leng*.25)+1):leng,]
        detach(ccdata)
        attach(cctrainboot)
        glm1B.out <- glm(good ~ ageq1 + ageq2 + ageq3 + ageq4 + MONTHS_AT_ADDRESS1 + MONTHS_AT_ADDRESS2 +
        MONTHS_AT_ADDRESS3 + MONTHS_AT_ADDRESS4 + TENURE_CT + TENURE_HO + TENURE_PT + EMPLOY_STATUS_EM +
        EMPLOY_STATUS_HO + EMPLOY_STATUS_RE + EMPLOY_STATUS_SE + APPLICATION_CHANNEL_C + APPLICATION_CHANNEL_I +
        APPLICATION_CHANNEL_M, family = binomial("logit"))
        yp1B <- predict(glm1B.out, cctrainboot, type="response")
        bootauc1[i] <- roc(good,yp1B)
        detach(cctrainboot)
        attach(cctestboot)
        yp2B <- predict(glm1B.out, cctestboot, type="response")
        bootauc2[i] <- roc(good,yp2B)
        detach(cctestboot)
        }
sum=0
for(i in 1:B){
        sum = sum + (1/B)*((1-1/exp(1))*bootauc2[i]+ (1/exp(1))*bootauc1[i])
}
cat("AUC␣for␣the␣model␣when␣tested␣on␣itself␣=",OriginalAUC,"\n")
cat("Estimated␣bootstrapped␣AUC␣for␣the␣model␣=",sum,"\n")
```

7.6 Hypothesis test - distributional form

```
bootstrap.disform <- function(z,y,B){
        meanz <- mean(z)
        cat("Mean␣of␣original␣sample␣from␣dataset␣z␣=",meanz,"\n")
        meany <- mean(y)
        cat("Mean␣of␣original␣sample␣from␣dataset␣y␣=",meany,"\n")
        x<-c(z,y)
        stat <- meanz - meany
        cat("Statistic␣of␣interest␣for␣the␣paired␣dataset␣=",stat,"\n")
        m=length(z)
        r=length(y)
        bootreplications <- NULL
        zstar <- NULL
        ystar <- NULL
        for(i in 1:B){
                bootsample <- sample(x,replace="TRUE")
                for(j in 1:m){
                        zstar[j]<-bootsample[j]
                }
                for(j in (m+1):(m+r)){
                        ystar[j-m]<-bootsample[j]
                }
```

```
                bootreplications[i] <- mean(zstar)-mean(ystar)
        }
        if(stat<0) stat<-stat*-1
        ASL <- NULL
        for(i in 1:B){
                if(stat>0) if(bootreplications[i]>=stat) ASL[i]<-1
                if(stat>0) if(bootreplications[i]<stat) ASL[i]<-0
                if(stat<=0) if(bootreplications[i]<=stat) ASL[i]<-1
                if(stat<=0) if(bootreplications[i]>stat) ASL[i]<-0
        }
        ASLboot = sum(ASL)/B
        cat("Achieved␣Significance␣Level␣=",ASLboot,"\n")
        hist(bootreplications)
}
z <- rnorm(25,0.65,0.1)
y <- runif(25,0,1.4)
bootstrap.disform(z,y,1000)
```

7.7 Hypothesis test - significant difference

```
attach(ccdata)
nu<-sample(leng, floor(leng*.25), replace=FALSE)
cctest <- ccdata[nu,]
cctrain <- ccdata[-nu,]
detach(ccdata)
attach(cctrain)
glm1.out <- glm(good ~ ageq1 + MONTHS_AT_ADDRESS2 + MONTHS_AT_ADDRESS3 + TENURE_CT + TENURE_HO,
family = binomial("logit"))
glm2.out <- glm(good ~ TENURE_HO, family = binomial("logit"))
detach(cctrain)
attach(cctest)
yp1 <- predict(glm1.out, cctest, type="response")
yp2 <- predict(glm2.out, cctest, type="response")
OriginalAUC1 <- roc(good,yp1)
OriginalAUC2 <- roc(good,yp2)
AUCdiff<-OriginalAUC1 - OriginalAUC2
roc_plot(good,yp1)
roc_plot(good,yp2)
detach(cctest)
B = 1000
bootauc1 <- NULL
bootauc2 <- NULL
bootaucdiff <- NULL
for (i in 1:B) {
        attach(cctest)
        nx<-sample(floor(leng*.25), floor(leng*.25), replace=TRUE)
        cctestboot <- cctest[nx,]
        detach(cctest)
        attach(cctestboot)
        yp1 <- predict(glm1.out, cctestboot, type="response")
        yp2 <- predict(glm2.out, cctestboot, type="response")
        bootauc1[i] <- roc(good,yp1)
        bootauc2[i] <- roc(good,yp2)
        bootaucdiff[i] <- bootauc1[i]-bootauc2[i]
        detach(cctestboot)
}
meanbootstrap = mean(bootaucdiff)
cat("Mean␣of␣the␣bootstrap␣of␣differences␣=", meanbootstrap,"\n")
cat("AUC␣for␣model␣1␣=", OriginalAUC1,"\n")
cat("AUC␣for␣model␣2␣=", OriginalAUC2,"\n")
cat("AUC␣difference␣=", AUCdiff,"\n")
hist(bootaucdiff,main="Histogram␣of␣the␣difference␣in␣AUCs")
#plot a histogram
## Confidence intervals using percentiles
bootaucsort<-sort(bootaucdiff)
k<-ceiling(0.025*(B+1))
p1<-bootaucsort[k]
p2<-bootaucsort[B+1-k]
upper4 = p2
lower4 = p1
cat("Lower␣percentile␣Confidence␣Interval␣for␣the␣model␣=",lower4,"\n")
cat("Upper␣percentile␣Confidence␣Interval␣for␣the␣model␣=",upper4,"\n")
```

7.8 Hypothesis test - insignificant difference

```
attach(ccdata)
ccdata$ranm <- rnorm(leng,0,0.1)
## Add's an extra random term to the data sampled from the Normal(0,0.1) distribution
nu<-sample(leng, floor(leng*.25), replace=FALSE)
cctest <- ccdata[nu,]
```

```
cctrain <- ccdata[-nu,]
detach(ccdata)
attach(cctrain)
glm1.out <- glm(good ~ ageq1 + MONTHS_AT_ADDRESS2 + MONTHS_AT_ADDRESS3 + TENURE_CT + TENURE_HO,
family = binomial("logit"))
glm2.out <- glm(good ~ ageq1 + MONTHS_AT_ADDRESS2 + MONTHS_AT_ADDRESS3 + TENURE_CT + TENURE_HO
+ ranm, family = binomial("logit"))
## The second model is identical apart from the extra random term
detach(cctrain)
attach(cctest)
yp1 <- predict(glm1.out, cctest, type="response")
yp2 <- predict(glm2.out, cctest, type="response")
OriginalAUC1 <- roc(good,yp1)
OriginalAUC2 <- roc(good,yp2)
AUCdiff<-OriginalAUC1 - OriginalAUC2
roc_plot(good,yp1)
roc_plot(good,yp2)
detach(cctest)
B = 1000
## Build just one model with different style of confidence intervals...
bootauc1 <- NULL
bootauc2 <- NULL
bootaucdiff <- NULL
for (i in 1:B) {
        attach(cctest)
        nx<-sample(floor(leng*.25), floor(leng*.25), replace=TRUE)
        cctestboot <- cctest[nx,]
        detach(cctest)
        attach(cctestboot)
        yp1 <- predict(glm1.out, cctestboot, type="response")
        yp2 <- predict(glm2.out, cctestboot, type="response")
        bootauc1[i] <- roc(good,yp1)
        bootauc2[i] <- roc(good,yp2)
        bootaucdiff[i] <- bootauc1[i]-bootauc2[i]
        detach(cctestboot)
}
meanbootstrap = mean(bootaucdiff)
cat("Mean␣of␣the␣bootstrap␣of␣differences␣=", meanbootstrap,"\n")
cat("AUC␣for␣model␣1␣=", OriginalAUC1,"\n")
cat("AUC␣for␣model␣2␣=", OriginalAUC2,"\n")
cat("AUC␣difference␣=", AUCdiff,"\n")
hist(bootaucdiff,main="Histogram␣of␣the␣difference␣in␣AUCs")
bootaucsort<-sort(bootaucdiff)
k<-ceiling(0.025*(B+1))
p1<-bootaucsort[k]
p2<-bootaucsort[B+1-k]
upper4 = p2
lower4 = p1
cat("Lower␣percentile␣Confidence␣Interval␣for␣the␣model␣=",lower4,"\n")
cat("Upper␣percentile␣Confidence␣Interval␣for␣the␣model␣=",upper4,"\n")
```

7.9 Generating an AR(1) model

```
Y <- NULL
Y[1] = 0
for(i in 2:50){
        Y[i] <- 0.8*Y[i-1]+rnorm(1,0,0.1)
}
ar(Y,order.max=1)
acf(Y,main="Autocorrelation␣function")
```

7.10 Bootstrapping residuals

```
residualboot <- function(Y) {
        n=length(Y)
        ybar <- mean(Y)
        X <- NULL
        for(i in 1:n){
                X[i] <- Y[i]-ybar
        }
        Z<-NULL
        z<-NULL
        for(i in 2:n){
                z[i-1] <- X[i]
        }
        for(i in 1:(n-1)){
                Z[i] <- X[i]
        }
        phi_hat <- solve(t(Z)%*%Z)*t(Z)%*%z
        ep <- NULL
```

```
        for(i in 2:n){
                ep[i-1] <- X[i] - phi_hat*X[i-1]
        }
        B=1000
        phiboot <- NULL leng<-length(ep) epboot <- NULL
        for(i in 1:B){
                X[1] <- Y[1] - ybar
                nu <- sample(leng,leng,replace=TRUE)
                epboot <- ep[nu]
                for(j in 2:n){
                        X[j] <- phi_hat*X[j-1] + epboot[j-1]
                }
                Z<-NULL
                z<-NULL
                for(j in 2:n){
                        z[j-1] <- X[j]
                }
                for(j in 1:(n-1)){
                        Z[j] <- X[j]
                }
                phiboot[i] <- solve(t(Z)%*%Z)*t(Z)%*%z
        }
        hist(phiboot)
        meanbootstrap = mean(phiboot)
        var(phiboot)
        sebootstrap = sqrt(var(phiboot))
        phi_hat
        cat("Estimated bootstrap standard error =",sebootstrap,"\n")
        cat("Mean of the bootstrap for phihat =",meanbootstrap,"\n")
        ## Confidence intervals using percentiles
        phisort<-sort(phiboot)
        k<-ceiling(0.025*(B+1))
        p1<-phisort[k]
        p2<-phisort[B+1-k]
        upper = p2
        lower = p1
        cat("Lower percentile Confidence Interval for the model =",lower,"\n")
        cat("Upper percentile Confidence Interval for the model =",upper,"\n")
}
```

7.11 Moving block bootstrap

```
bootstrap.movingblock <- function(Y,B,l){
        bootreplications = NULL
        bootsample = NULL
        blockbootsample = NULL
        ybar <- mean(Y)
        n = length(Y)
        X <- NULL
        for(i in 1:n){
                X[i] <- Y[i]-ybar
        }
        ## Find my residuals
        Z<-NULL
        z<-NULL
        for(i in 2:n){
                z[i-1] <- X[i]
        }
        for(i in 1:(n-1)){
                Z[i] <- X[i]
        }
        phi_hat <- solve(t(Z)%*%Z)*t(Z)%*%z
        cat("Estimate for phi from the original sample=",phi_hat,"\n")
        k = floor(n/l)
        for (i in 1:B) {
                startpt=sample(1:(n-l+1),k,replace="TRUE")
                for (j in 1:k) {
                        blockbootsample[((j-1)*l+1):(j*l)] = Y[(startpt[j]):(startpt[j]+l-1)]
                }
                X <- NULL
                for(p in 1:(k*l)){
                        X[p] <- blockbootsample[p]-ybar
                }
                Z<-NULL
                z<-NULL
                for(q in 2:(k*l)){
                        z[q-1] <- X[q]
                }
                for(q in 1:(k*l-1)){
                        Z[q] <- X[q]
                }
                bootreplications[i] <- solve(t(Z)%*%Z)*t(Z)%*%z
```

```
        }
        meanbootstrap = mean(bootreplications)
        varbootstrap = var(bootreplications)
        sebootstrap = sqrt(varbootstrap)*sqrt(k*l/n)
        #calculate the standard error of the bootstrap replications
        cat("Estimated bootstrap standard error =",sebootstrap,"\n")
        #Display the estimated standard error for the statistic
        cat("Mean of the bootstrap for phihat =",meanbootstrap,"\n")
        #Display the mean of the bootstrap difference of means
        hist(bootreplications, main="Moving Block Bootstrap")
        ## Confidence intervals using percentiles
        phisort<-sort(bootreplications)
        ku<-ceiling(0.025*(B+1))
        p1<-phisort[ku]
        p2<-phisort[B+1-ku]
        upper = p2
        lower = p1
        cat("Lower percentile Confidence Interval for the model =",lower,"\n")
        cat("Upper percentile Confidence Interval for the model =",upper,"\n")
}
bootstrap.movingblock(Y,1000,5)
```

7.12 Non-overlapping block bootstrap

```
bootstrap.nonoverlapping <- function(Y,B,l) {
        bootreplications = NULL
        bootsample = NULL
        blockbootsample = NULL
        ybar <- mean(Y)
        n = length(Y)
        X <- NULL
        for(i in 1:n){
                X[i] <- Y[i]-ybar
        }
        Z<-NULL
        z<-NULL
        for(i in 2:n){
                z[i-1] <- X[i]
        }
        for(i in 1:(n-1)){
                Z[i] <- X[i]
        }
        phi_hat <- solve(t(Z)%*%Z)*t(Z)%*%z
        cat("Estimate for phi from the original sample=",phi_hat,"\n")
        k = floor(n/l)
        startpt <- NULL
        for (i in 1:B) {
                Q <- c(1:(k*l))
                for(g in 1:k){
                        startpt[g] = Q[((g-1)*l+1)]
                }
                startptrand=sample(startpt,k,replace="TRUE")
                for (j in 1:k) {
                        blockbootsample[((j-1)*l+1):(j*l)] = Y[(startptrand[j]):(startptrand[j]+l-1)]
                }
                X <- NULL
                for(p in 1:(k*l)){
                        X[p] <- blockbootsample[p]-ybar
                }
                Z<-NULL
                z<-NULL
                for(q in 2:(k*l)){
                        z[q-1] <- X[q]
                }
                for(q in 1:(k*l-1)){
                        Z[q] <- X[q]
                }
                bootreplications[i] <- solve(t(Z)%*%Z)*t(Z)%*%z
        }
        meanbootstrap = mean(bootreplications)
        varbootstrap = var(bootreplications)
        sebootstrap = sqrt(varbootstrap)*sqrt(k*l/n)
        cat("Estimated bootstrap standard error =",sebootstrap,"\n")
        cat("Mean of the bootstrap for phihat =",meanbootstrap,"\n")
        hist(bootreplications, main="Non-overlapping Block Bootstrap")
        phisort<-sort(bootreplications)
        ku<-ceiling(0.025*(B+1))
        p1<-phisort[ku]
        p2<-phisort[B+1-ku]
        upper = p2
        lower = p1
        cat("Lower percentile Confidence Interval for the model =",lower,"\n")
```

```
        cat("Upper␣percentile␣Confidence␣Interval␣for␣the␣model␣=",upper,"\n")
}
bootstrap.nonoverlapping(Y,1000,5)
```

7.13 Circular block bootstrap

```
bootstrap.circular <- function(Y,B,l) {
        bootreplications = NULL
        bootsample = NULL
        blockbootsample = NULL
        ybar <- mean(Y)
        n = length(Y)
        X <- NULL
        for(i in 1:n){
                X[i] <- Y[i]-ybar
        }
        Z<-NULL
        z<-NULL
        for(i in 2:n){
                z[i-1] <- X[i]
        }
        for(i in 1:(n-1)){
                Z[i] <- X[i]
        }
        phi_hat <- solve(t(Z)%*%Z)*t(Z)%*%z
        cat("Estimate␣for␣phi␣from␣the␣original␣sample=",phi_hat,"\n")
        k = floor(n/l)
        Y <- c(Y,Y)
        for (i in 1:B) {
                startpt=sample(1:n,k,replace="TRUE")
                for (j in 1:k) {
                        blockbootsample[((j-1)*l+1):(j*l)] = Y[(startpt[j]):(startpt[j]+l-1)]
                }
                X <- NULL
                for(p in 1:(k*l)){
                        X[p] <- blockbootsample[p]-ybar
                }
                Z<-NULL
                z<-NULL
                for(q in 2:(k*l)){
                        z[q-1] <- X[q]
                }
                for(q in 1:(k*l-1)){
                        Z[q] <- X[q]
                }
                bootreplications[i] <- solve(t(Z)%*%Z)*t(Z)%*%z
        }
        meanbootstrap = mean(bootreplications)
        varbootstrap = var(bootreplications)
        sebootstrap = sqrt(varbootstrap)*sqrt(k*l/n)
        cat("Estimated␣bootstrap␣standard␣error␣=",sebootstrap,"\n")
        cat("Mean␣of␣the␣bootstrap␣for␣phihat␣=",meanbootstrap,"\n")
        hist(bootreplications, main="Circular␣Block␣Bootstrap")
        phisort<-sort(bootreplications)
        ku<-ceiling(0.025*(B+1))
        p1<-phisort[ku]
        p2<-phisort[B+1-ku]
        upper = p2
        lower = p1
        cat("Lower␣percentile␣Confidence␣Interval␣for␣the␣model␣=",lower,"\n")
        cat("Upper␣percentile␣Confidence␣Interval␣for␣the␣model␣=",upper,"\n")
}
bootstrap.circular(Y,1000,5)
```

7.14 Stationary block bootstrap

```
bootstrap.stationary <- function(Y,B,p){
        bootreplications = NULL
        bootsample = NULL
        blockbootsample = NULL
        ybar <- mean(Y)
        n = length(Y)
        X <- NULL
        for(i in 1:n){
                X[i] <- Y[i]-ybar
        }
        Z<-NULL
        z<-NULL
        for(i in 2:n){
                z[i-1] <- X[i]
```

```
        }
        for(i in 1:(n-1)){
                Z[i] <- X[i]
        }
        phi_hat <- solve(t(Z)%*%Z)*t(Z)%*%z
        cat("Estimate for phi from the original sample=",phi_hat,"\n")
        Y <- c(Y,Y)
        for (i in 1:B) {
                blockbootsample <- NULL
                len = 0
                while(len<n){
                        Q <- c(1:n)
                        L <- rgeom(1,p)
                        if(L == 0) L <- rgeom(1,p)
                        if(L == 0) L <- rgeom(1,p)
                        if(L == 0) L <- rgeom(1,p)
                        if(L == 0) L <- rgeom(1,p)
                        if(L == 0) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        if(L > n) L <- rgeom(1,p)
                        ## to ensure L is between 0 and n
                        startpt = sample(Q,1)
                        extra <- NULL
                        extra <- Y[(startpt):(startpt+L-1)]
                        blockbootsample <- c(blockbootsample,extra)
                        len = length(blockbootsample)
                }
                len = length(blockbootsample)
                X <- NULL
                for(q in 1:len){
                        X[q] <- blockbootsample[q]-ybar
                }
                Z<-NULL
                z<-NULL
                for(q in 2:len){
                        z[q-1] <- X[q]
                }
                for(q in 1:(len-1)){
                        Z[q] <- X[q]
                }
                bootreplications[i] <- solve(t(Z)%*%Z)*t(Z)%*%z
                }
        meanbootstrap = mean(bootreplications)
        varbootstrap = var(bootreplications)
        sebootstrap = sqrt(varbootstrap)
        cat("Estimated bootstrap standard error =",sebootstrap,"\n")
        cat("Mean of the bootstrap for phihat =",meanbootstrap,"\n")
        hist(bootreplications, main="Stationary Block Bootstrap")
        phisort<-sort(bootreplications)
        ku<-ceiling(0.025*(B+1))
        p1<-phisort[ku]
        p2<-phisort[B+1-ku]
        upper = p2
        lower = p1
        cat("Lower percentile Confidence Interval for the model =",lower,"\n")
        cat("Upper percentile Confidence Interval for the model =",upper,"\n")
}
bootstrap.stationary(Y,1000,0.2)
```

7.15 Greenwood's formula

```
KapMeier <- function(t){
        setwd("C:/Users/Luke/Dropbox/Dissertation_new")
        ccdata<-read.delim("ordered_data_cc_app.txt")
        library(survival)
        attach(ccdata)
        vec <- cbind(stmt_rel_mth,1-good)
        N <- NULL
        D <- NULL
        n <- length(stmt_rel_mth)
        for(j in 1:40){
                value = 0
                sum = 0
                for(i in 1:n){
                        if(stmt_rel_mth[i]<j) sum=1 else sum=0
                        value = sum + value
                }
```

```
                N[j] <- n - value
        }
        for(j in 1:41){
                value = 0
                sum = 0
                for(i in 1:n){
                        if(stmt_rel_mth[i]<j) sum=1 else sum=0
                        if(good[i]<1) sum=sum*1 else sum=sum*0
                        value = sum + value
                }
                D[j] <- value
        }
        for(i in 1:41){
                D[i] <- D[i+1]
        }
        D[41] <- 0
        D
        sum = 0
        for(j in 2:39){
                sum = sum + D[j]
                D[j+1] <- D[j+1] - sum
        }
        product=1
        for(i in 1:t){
                product<-product*(1-(D[i]/N[i]))
        }
        S<-product

        ##Greenwoods forumla
        sum = 0
        for(i in 1:t){
                sum = sum + (D[i]/(N[i]*(N[i]-D[i])))
        }
        Green <- S*sqrt(sum)
        cat("S-hat ",S,"\n")
        cat("Greenwood's Standard Error ",Green,"\n")
        detach(ccdata)
}
```

7.16 Bootstrapping the Kaplan Meier Curve

```
BootKapMeier <- function(t,B){
        setwd("C:/Users/Luke/Dropbox/Dissertation_new")
        ccdata<-read.delim("ordered_data_cc_app.txt")
        attach(ccdata)
        vec <- data.frame(stmt_rel_mth,1-good,good)
        n <- length(stmt_rel_mth)
        Sboot<-NULL
        detach(ccdata)
        for(b in 1:B){
                nu<-sample(n, n, replace=TRUE)
                vecstar <- vec[nu,]
                attach(vecstar)
                N <- NULL
                D <- NULL
                for(j in 1:40){
                        value = 0
                        sum = 0
                        for(i in 1:n){
                                if(stmt_rel_mth[i]<j) sum=1 else sum=0
                                value = sum + value
                        }
                        N[j] <- n - value
                }
                for(j in 1:41){
                        value = 0
                        sum = 0
                        for(i in 1:n){
                                if(stmt_rel_mth[i]<j) sum=1 else sum=0
                                if(good[i]<1) sum=sum*1 else sum=sum*0
                                value = sum + value
                        }
                        D[j] <- value
                }
                for(i in 1:41){
                        D[i] <- D[i+1]
                }
                D[41] <- 0
                D
                sum = 0
                for(j in 2:39){
                        sum = sum + D[j]
```

```
                        D[j+1] <- D[j+1] - sum
                }
                product=1
                for(i in 1:t){
                        product<-product*(1-(D[i]/N[i]))
                }
                Sboot[b]<-product
                detach(vecstar)
        }
        SE<-sqrt(var(Sboot))
        cat("Bootstrap Standard Error =",SE,"\n")
}
```

Chapter 8

References

1. Bradley Efron and Robert J. Tibshirani. An Introduction to the Bootstrap. London. Chapham & Hall; 1993.
2. Humberto Barreto, Frank M. Howland. Introductory Econometrics: Using Monte Carlo Simulation with Microsoft Excel. Cambridge University Press.
3. The Stationary Bootstrap Dimitris N. Politis and Joseph P. Romano Journal of the American Statistical Association , Vol. 89, No. 428 (Dec., 1994), pp. 1303-1313
4. Shorack, G.R.; Wellner, J.A. (1986). Empirical processes with applications to statistics. New York: Wiley.
5. Stephen M. S. Lee and P. Y. Lai. Double block bootstrap confidence intervals for dependent data. Biometrika (2009) 96 (2): 427-443. doi: 10.1093/biomet/asp018
6. Molodianovitch, K., Faraggi, D. and Reiser, B. (2006), Comparing the Areas Under Two Correlated ROC Curves: Parametric and Non-Parametric Approaches. Biom. J., 48: 745–757. doi: 10.1002/bimj.200610223
7. Lyn C. Thomas. Consumer Credit Models. Oxford University Press - Special; 2009.
8. Bernd Engelmann. XII Measures of a Rating's Discriminative Power - Applications and Liminations. The Basel II Risk Parameters, Pages 263-287. Springer; 2006.
9. Hana Skalska and Vaclav Freylich. Web-Bootstrap Estimate of Area Under ROC Curve. AUSTRIAN JOURNAL OF STATISTICS Volume 35 (2006), Number 2&3, 325–330.
10. Elizabeth R. DeLong, David M. DeLong, Daniel L. Clarke-Pearson. Comparing the Areas under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. Biometrics, Vol. 44, No. 3 (Sep., 1988), pp. 837-845
11. Jun Shao and Dongsheng Tu. The Jackknife and Bootstrap. Springer Series in Statistics. 1995.
12. B. Efron and R. Tibshirani. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. Statistical Science. 1986. Vol. 1, 54-77.
13. Michael R. Chernick. Bootstrap Methods A Guide for Practitioners and Researchers. Second Edition. Wiley.
14. James D. Hamilton. Times Series Analysis. Princeton University Press. 1994.
15. Michael G. Akritas. Bootstrapping the Kaplan-Meier Estimator. Journal of the American Statistical Association , Vol. 81, No. 396 (Dec., 1986), pp. 1032-1038
16. D. R. Cox. Regression Models and Life-Tables. Journal of the Royal Statistical Society. Series B (Methodological) , Vol. 34, No. 2 (1972), pp. 187-220

17. J. D. Kalbfleisch, Ross L. Prentice. The statistical analysis of failure time data. Wiley, New York. 1980

18. Bradley Efron. Censored Data and the Bootstrap. Journal of the American Statistical Association , Vol. 76, No. 374 (Jun., 1981), pp. 312-319