Assignment 4

Brian Schetzsle, Lindsay Brock, Gustavo Esparza

September 25, 2019

Chapter 10 Summary

Section 10.1: The Jackknife Estimate of Standard Error

Chapter 10 introduces the concepts of the Jackknife and the Bootstrap; both are computational approximations of the standard deviation of some estimated population statistic. Jackknife, originally introduced in 1957, manipulates the difference between an estimated population statistic and that same statistic minus a single data point. So if

$$x_i \sim iid \quad F \qquad \text{for } i = 1, 2, ..., n$$

$$\hat{\theta} = s(x)$$

$$x_{(i)} = (x_1, x_2, ..., x_{i-1}, x_{i+1}, ..., x_n)$$

$$\hat{\theta}_{(i)} = s(x_{(i)})$$

Then the jackknife estimate of standard error for $\hat{\theta}$ is

$$\hat{se}_{jack} = \left[\frac{n-1}{n} \sum_{i=1}^{n} (\hat{\theta}_{(i)} - \hat{\theta}_{(.)} \right]^{\frac{1}{2}}, \quad \text{with } \hat{\theta}_{(.)} = \frac{1}{n} \sum_{i=1}^{n} \hat{\theta}_{(i)}$$

So this is similar to calculating the variance within a dataset, but rather than calculate on x_i its calculated on the corresponding population estimate $\hat{\theta}_{(i)}$. The $\frac{n-1}{n}$ is there so that if the population mean is estimated, the standard error from jackknife corresponds to that from classical statistics.

Some important notes on jackknife are that it is nonparametric, so it makes no assumptions about the distribution of the data; it is "automatic" so a single algorithm can compute the jackknife standard error of any statistic; there is a hidden assumption of smoothness in the function of the statistic so discontinuous functions like median can cause problems; jackknife systematically overestimates the true standard error, especially where a value of the statistic is sensitive to small changes in the data; the jackknife is, at its core, a numerical approximation of a taylor series expansion of the function of the statistic.

Section 10.2: The Nonparametric Bootstrap

The nonparametric, one-sample bootstrap is more computationally intensive and simulates additional sampling from the underlying population from which the data originally came. Then the bootstrap can directly calculate the standard error of a population statistic which frequentist methods can only get by appealing to asymptotic theory.

Given data

$$x = (x_1, x_2, ..., x_n)$$

make n equally likely draws from this data, with replacement, to get a new dataset

$$x^* = (x_1^*, x_2^*, ..., x_n^*)$$

that is comprised of values from the original data. Calculate the desired statistic from this new data

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

repeat B many times (say $B = 500$) and then the standard error of the statistic $\hat{\theta}$ is the emperical standard deviation of the simulated $\hat{\theta}^*$ values

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

Some important notes about bootstrap are that it is, like jackknife, completely automatic, so it doesn't matter what statistic you're calculating; there are parametric and multisample versions of bootstrap discussed shortly; the bootstrap is a more dependable estimator of standard error than jackknife when a statistic's function is not smooth; 200 is generally a sufficient number of resamples; we could also use bootstrap to estimate expected absolute error $E[|\hat{\theta} - \theta|]$ rather than standard error; the bootstrap can provide robust confidence intervals that do not assume normality.

Section 10.3 and 10.4: Resampling Plans and the Parametric Bootstrap

The chapter then goes on to explore how both the jackknife and the bootstrap can be seen as functions of a sampling vector. The discussion is esoteric but suggest alternate ways to resample the data. These include the *infinitesimal jackknife*, which seems like it has a small chance at picking the same data point twice, though why this is valuable isn't explained here; the *Multisample Bootstrap*, which is used to compare samples from two or more groups; *Moving Blocks Bootstrap*, which is a way of sampling from data that are not *iid*, but instead have some dependent structure; *the Bayesian Bootstrap*, which is a bayesian interpretation of sampling from the unknown underlying distribution F; *the Parametric Bootstrap*, which is a way to enforce some parameter assumptions on the underlying distribution F when resampling.

Section 10.5: Influence Functions and Robust Estimation

There is a section on *Influence Functions* and *Robust Estimation* that deals with an effort begun in the 1960s to improve the statistical properties of the mean. The influence function evaluated at a point x is defined to be

$$IF(x) = \lim_{\epsilon \to 0} \frac{T((1-\epsilon)F + \epsilon\delta_x) - T(F)}{\epsilon}$$

where F is the cumulative distribution of the data and δ_x is the "one-point probability distribution" putting probability 1 on x. The influence function measures how sensitive F is to local changes around x. Distributions with heavy tails have large influence functions when x is far from the mean, which is not desirable. A modification to the traditional calculation for the mean removes the bottom α and top $(1-\alpha)$ percentiles and then calculates the mean on the remaining data, so the mean is less susceptible to outliers. Bootstrap provides evidence supporting this method of robust estimation.

Chapter 11 Summary

Section 11.1: Bootstrap Confidence Intervals

This chapter will focus on the bootstrap automation of confidence intervals. We are accustomed to the standard interval

$$\hat{\theta} \pm 1.96\hat{se}$$

for approximate 95% coverage, but they can prove to be inaccurate in many cases. For example, the Poisson 95% confidence interval is poorly explained by the computation above. This is due to the fact that our standard interval notation is symmetric around $\hat{\theta}$, while the Poisson (and many other distributions) do not hold this symmetric quality. Capturing the distribution of our data for confidence intervals is the main goal of bootstrap confidence interval theory.

Section 11.1: Neyman's Construction for One Parameter Problems

We begin by discussing a set of data comprised of $n = 22$ pairs,

$$x_i = (m_i, v_i), i = 1, \ldots, 22$$

Now, consider the sample correlation coefficient $\hat{\theta}$ between m_i and v_i. We should ask ourselves what we can infer about the true correlation. If we assume that the pairs originate from a bivariate normal distribution then we can numerically compute two density functions and solve for their MLEs. These two values, $\hat{\theta}(lo)$ and $\hat{\theta}(up)$ correspond to the smallest and largest value of θ putting probability at least .025 above $\hat{\theta}$. Thus, we have

$$\theta \in [\hat{\theta}(lo), \hat{\theta}(up)]$$

representing the 95% confidence interval for the true correlation θ. This is defined as Neyman's construction of confidence intervals for one parameter problems $f_\theta(\hat{\theta})$

We can now discuss the property of transformation invariance for confidence intervals. Consider our primary interest being $log(\theta)$ rather than θ itself. Generally, for any monotonic increasing function of θ, $m(\theta) = \hat{\phi}$, the Confidence interval for the transformation can be defined as

$$C^\phi(\hat{\phi}) = \{\phi = m(\theta) \text{ for } \theta \in C(\hat{\theta})\}$$

This simply states that the event $\theta \in C(\hat{\theta})$ is the same as $\phi \in c^\phi(\hat{\phi})$. When working with transformation invariance, Fisher was able to make a dramatic suggestion for ϕ. The suggestion was the proposal for the following transformation:

$$\phi = m(\theta) = \frac{1}{2} log(\frac{1+\theta}{1-\theta})$$

This transformation greatly approximates the following normal distribution:

$$\hat{\phi} \sim N(\phi, \frac{1}{n-3})$$

Thus, we are once again able to construct confidence intervals with the useful assumptions of normality, where

$$C^\phi(\hat{\phi}) = \hat{\phi} \pm 1.96\frac{1}{\sqrt{n-3}}$$

This interval is then a close approximation for Neyman's original construction. Bayesian confidence intervals are generally reliant on transformation invariance, and therefore Neyman's construction is much more palatable to Bayesian statisticians as opposed to standard intervals

Section 11.2: The Percentile Method

In bootstrapping confidence intervals, we desire to automatically produce an appropriate confidence interval for the unseen parameter θ. As our sample size increases, we generally utilize the assumptions of normality to create this automatic interval, but we will focus on methods when this assumption is not validated.

The percentile method uses the shape of the bootstrap distribution to improve upon the standard intervals. When producing B bootstrap replications for θ, we use the obvious percentiles of their distribution to define the percentile confidence limits. More precisely, we can define our interval in terms of the bootstrap CDF, $\hat{G}(t)$, the proportion of bootstrap samples less than t:

$$\hat{G}(t) = \#\{\hat{\theta^{*b}} \leq t\}/B$$

Then, the αth percentile point $\hat{\theta}^{*\alpha}$ of the bootstrap distribution is given by the inverse function of $\hat{G}$,

$$\hat{\theta}^{*\alpha} = \hat{G}^{-1}(\alpha)$$

This is simply the value putting proportion α of the bootstrap to its left. In this context, the percentile interval can be defined as follows:

$$\left(\hat{\theta}_{\%ile}[.025], \hat{\theta}_{\%ile}[.975]\right)$$

As expected, the percentile intervals are indeed transformation invariant. Similar to the previous section, the bootstrap transformation percentiles are defined as

$$\hat{\phi}^{*(\alpha)} = m(\hat{\theta}^{*(\alpha)}) \rightarrow \hat{\phi}_{\%ile}[\alpha] = m(\hat{\theta}_{\%ile}[\alpha])$$

Once again, we should consider how transformation invariance can assist in constructing normally distributed intervals. Thus considering a transformation ϕ such that

$$\hat{\phi} \sim N(\phi, \sigma^2)$$

If this were the case for our monotonic increasing transformation, then it would follow that the bootstrapped transformations would also be normally distributed. Then, our percentiles are able to be defined as

$$\hat{\phi}_{\%ile}[\alpha] = \hat{\phi}^{*(\alpha)} = \hat{\phi} + z^{\alpha}\sigma$$

Where z corresponds to the standard normal percentiles. Considering a 95% confidence interval would thus provide a simplistic definition for the bootstrapped confidence interval for ϕ:

$$\hat{\phi} \pm 1.96\sigma$$

It is clear that transformation invariance carries great weight when attempting to create easy to interpret confidence intervals when considering bootstrapping.

What follows are several notes regarding the percentile method that are very useful for understanding the underlying theory:

∗ The method does not require knowledge regarding the transformation to normality, only that it exists.

∗ The assumption for standard normality becomes more accurate as n increases, but convergence can vary depending on distributions.

∗ The broader assumption of general normality speeds up convergence, which is explored in a future section regarding asymptotic normality.

∗The method requires bootstrap samples on the order of B = 2000.

Section 11.3: Bias-Corrected Confidence Intervals

When considering the percentile method for confidence intervals, the ideal form is that of the normal distribution with a constant variance:

$$\hat{\phi}^* \sim N(\hat{\phi}, \sigma^2)$$

While this is certainly ideal, we should also consider instances where bias or non-constant variance occur. We will begin by examining the occurrence of bias. If our transformed values are truly normally distributed, then we have the following result for our bootstrapped interval:

$$Pr_*\{\hat{\phi}^* \leq \hat{\phi}\} = 0.50$$

Here, Pr_* indicates the bootstrapped probability. This is our theoretical value, but bootstrapped samples may be biased upwards or downwards, depending on the results from our bootstrap. For this reason, we should consider the *Bias Corrected Percentile Method* or BC for short. This method is a data-based algorithm that proceeds as follows:

Having simulated B bootstrap replications $\hat{\theta}^{*1}, \hat{\theta}^{*2}, \ldots, \hat{\theta}^{*B}$, let p_0 be the proportion of replications less that $\hat{\theta}$:

$$p_0 = \#\{\hat{\theta}^{*b} \leq \hat{\theta}\}/B$$

Now, define the bias-correction value

$$z_0 = \Phi^{-1}(p_0), \text{ where } \Phi \text{ is the inverse standard normal CDF}$$

Then, the BC level α confidence interval endpoint is defined to be

$$\hat{\theta}_{BC}[\alpha] = \hat{G^{-1}}[\Phi(2z_0 + z^{(\alpha)})]$$

Where $\hat{G}$ is the bootstrap CDF and $z^{(\alpha)} = \Phi^{-1}(\alpha)$ We can note the case where $P_0 = .5$, then $Z_0 = 0$ and

$$\hat{\theta}_{BC}[\alpha] = \hat{G^{-1}}[\Phi(z^{(\alpha)})] = \hat{G^{-1}}(\alpha) = \hat{\theta_{\%ile}}[\alpha]$$

In this instance, we simply have the percentile limit. Any other instance where $P_0 \neq .5$, will have a bias correction being made. This outline is a useful guide for constructing a percentile confidence interval when a distribution does not have a median unbiased situation.

Section 11.4: Second-Order Accuracy

Coverage errors of standard confidence intervals decrease at an order of $O(1/\sqrt{n})$ and the actual coverage probability is below, where the value of c_1 depends on the situation. This is also called "first-order accuracy" and can provide slow convergence to the nominal coverage level α, requiring 4 times the samples size to cut the error rate in half.

$$Pr_\theta\{\theta \leq \hat{\theta}_{stan}[\alpha]\} \doteq \alpha + c_1/\sqrt{n}$$

Decreasing at an order of $O(1/n)$, the equation for "second-order accurate" method is below. It often provides almost the entire claimed coverage probabilities even with small sample sizes. Neither the percentile method or BC method is second-order accurate.

$$Pr_\theta\{\theta \leq \hat{\theta}_{2nd}[\alpha]\} \doteq \alpha + c_2/n$$

The bias-corrected and accelerated method, or BC_a, takes its level-α confidence limit to be the equation below and it makes three corrections to standard intervals. One, for non-normality of $\hat{\theta}$ by using bootstrap percentiles instead of only bootstrap standard errors. Two, for bias by using the bias correction z_0. Three, for non-constant standard error through the a term. All three of these corrections have a large impact on the methods and are important to achieve second-order accuracy. This increased accuracy can also be extended to small sample sizes.

$$\hat{\theta}_{BC_a}[\alpha] = \hat{G}^{-1}\left[\Phi\left(z_0 + \frac{z_0 + z^{(\alpha)}}{1 - a(z_0 + z^{(\alpha)})}\right)\right]$$

The percentile and BC methods mentioned above can be used whenever a large number of bootstrap replications are available, greater than 2,000, but not for BC_a. A drawback of the latter method is that the acceleration a is not a function of the bootstrap distribution and must be computed separately. This can be done in three ways depending on the situation: by setting a equal to z_0 for one-parameter exponential families (i.e. Poisson), estimating a using $\hat{a} = \frac{1}{6}\frac{\sum_{i=1}^n (\hat{\theta}_{(i)} - \hat{\theta}_{(.)})^3}{\left[\sum_{i=1}^n (\hat{\theta}_i - \hat{\theta}_{(.)})^2\right]^{1.5}}$ for one-sample nonparametric problems, and using the *abc method* for multiparameter exponential families (along with **accel** in R).

On the topic of multiple parameters, high bias can be introduced when the parameter vector μ is high dimensional. As an example, suppose we have $x_i \sim N(\mu_i, 1)$ for $i = 1, \cdots, n$, which are independent, and we want to find a confidence interval for $\theta = \sum_1^n \mu_i^2$. The MLE $\hat{\theta} = \sum_1^n x_i^2$ will be sharply biased upward if n is large. If $n = 10$ and $\hat{\theta} = 20$, $z_0 = \Phi^{-1}(0.156) = -1.01$ which makes $\hat{\theta}_{BC}[0.025]$ equal an extremely tiny bootstrap percentile, $\hat{G}^{-1}(0.000034)$. This is a warning against the BC or BC_a intervals as they work best with small $|z_0|$ and $|a|$ less than 0.2. Using the maximum likelihood estimates in higher dimensions can also become an unseen issue which can be countered by computing z_0, even if not used for BC or BC_a, as it may red flag unknown biases.

Section 11.5: Bootstrap-t Intervals

To understand Bootstrap-t intervals, it is first important to understand what a student's t distribution is. Let's begin with an example. Suppose we have a set of independent observations from two possibly different Normal distributions, $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ where $x_i \sim N(\mu_x, \sigma^2)$ and $y_i \sim N(\mu_y, \sigma^2)$. We want to form a 0.95 central confidence interval for $\theta = \mu_y - \mu_x$ where the estimate is $\hat{\theta} = \bar{y} - \bar{x}$ but its distribution depends on the nuisance parameter σ^2. Using the student's t statistic we can base inference about θ on the pivotal quantity below, where $\hat{se}^2$ is an unbiased estimate of σ^2. The degrees of freedom are $n_x + n_y - 2$ if $\mu_x = \mu_y$, no matter what σ^2 may actually be.

$$t = \frac{\hat{\theta} - \theta}{\hat{se}} \qquad where \qquad \hat{se}^2 = \left(\frac{1}{n_x} + \frac{1}{n_y}\right) \frac{\sum_1^{n_x}(x_i - \bar{x})^2 + \sum_1^{n_y}(y_i - \bar{y})^2}{n_x + n_y - 2}$$

Letting t_{df}^{α} represent the $100\alpha th$ percentile of a t_{df} distribution gives us the equation below as the upper level-α interval of a Student's t confidence limit.

$$\hat{\theta}_t[\alpha] = \hat{\theta} - \hat{se} * t_{df}^{(1-\alpha)}$$

This theory depends on the normality assumptions given at the beginning of the example. The bootstrap-t approach is to accept that the t statistic is pivotal but to estimate its distribution via bootstrap resampling. Nonparametric bootstrap samples are drawn separately from x and y and denoted by x^* and y^*. Then $\hat{\theta}^*$ and $\hat{se}^*$ can be calculated and plugged in to find the statistic t^* using the original equation above. Repeating this process multiple times provides us with $\{t^{*b}, b = 1, \ldots, B\}$ which will in turn give us estimated percentiles $t^{*(\alpha)}$ and corresponding confidence limits $\hat{\theta}_t^*[\alpha] = \hat{\theta} - \hat{se} * t^{*(1-\alpha)}$.

Bootstrap-t intervals can perform well or poorly depending on the scale of the application (not transformation invariant). Using them on Fisher's scale they agree with exact intervals for the correlation coefficient. The exact formula for $\hat{se}$ can also become an issue.

Section 11.6: Objective Bayes Intervals and the Confidence Distribution

Suppose we have a one-parameter family of densities $f_\theta(\hat{\theta})$ with prior density $g(\theta)$. The posterior density of θ is below, where $f(\hat{\theta})$ is the marginal density $\int f(\hat{\theta})g(\theta)d\theta$,

$$g(\theta|\hat{\theta}) = g(\theta)f_\theta(\hat{\theta})/f(\hat{\theta})$$

The Bayes 0.95 *credible interval* $C(\theta|\hat{\theta})$ spans the central 0.95 region of $g(\theta|\hat{\theta})$, where $C(\theta|\hat{\theta}) = (a(\hat{\theta}), b(\hat{\theta}))$ with $\int_{a(\hat{\theta})}^{b(\hat{\theta})} g(\theta|\hat{\theta})d\theta = 0.95$ and with posterior probability 0.025 in each tail region.

These Bayesian intervals are credible based on uninformative priors. A *matching prior* for one-parameter problems is the Jeffrey's prior:

$$g(\theta) = I_\theta^{1/2} \qquad where \qquad I_\theta = \int \left[\frac{\partial}{\partial\theta} log f_\theta(\hat{\theta})\right]^2 f_\theta(\hat{\theta})d\hat{\theta}$$

With multiparameter families, difficulties arise due to "nuisance parameters", or the effects of the other parameters we are not interested in. Bayesians get rid of these nuisances by integrating them out of the posterior density $g(\mu|\boldsymbol{x}) = g(\mu)f_\mu(\boldsymbol{x})/f(\boldsymbol{x})$, with $\boldsymbol{x}$ representing all of the data. A credible interval for θ, $C(\theta, \boldsymbol{x})$ is then constructed much like before where $h(\theta|\boldsymbol{x})$ now plays the role of $g(\theta|\hat{\theta})$. This leaves us with finding an uninformative multidimensional prior $g(\mu)$, discussed later.

First, we must discuss Fiducial constructions. It rearranges the Student t pivotal $t = (\hat{\theta} - \theta)/\hat{se}$ as $\theta = \hat{\theta} - \hat{se} * t$. With the observed data this Fiducial theory assigns θ the distribution implied in the previous sentence, as if $\hat{\theta}$ and $\hat{se}$ were fixed at their calculated values while t was distributed as t_{df}. Then $\hat{\theta}_t[\alpha]$, the Student t level-α confidence limit, is the $100\alpha th$ percentile of θ's fiducial distribution. In the end, this method was not considered viable.

The underlying logic to this approach is that $\hat{\theta}$ and $\hat{se}$ exhaust the information about θ available from the data which leaves an irreducible component of randomness described by t. An upper confidence limit $\hat{\theta}_x[\alpha]$ satisfies $Pr\{\theta \leq \hat{\theta}_x[\alpha]\} = \alpha$ which leads to $Pr\{\hat{\theta}_x[\alpha] \leq \theta \leq \hat{\theta}_x[\alpha + \epsilon]\} = \epsilon$. $\hat{\theta}_x[\alpha]$ is a one-to-one function between α in (0,1) and θ a point in its parameter space Θ. Letting ϵ go to zero determines the confidence density of θ where $\tilde{g}_x(\theta) = d\alpha/d\theta$, the local derivative of probability at location θ for the unknown parameter.

Integrating $\tilde{g}_x(\theta)$ recovers α as a function of θ and let $\theta_1 = \hat{\theta}_x[\alpha_1]$ and $\theta_2 = \hat{\theta}_x[\alpha_2]$ for any two values $\alpha_1 < \alpha_2$ in (0,1). Then,

$$\int_{\theta_1}^{\theta_2} \tilde{g}_x(\theta)d\theta = \int_{\theta_1}^{\theta_2} \frac{d\alpha}{d\theta}d\theta = \alpha_1 - \alpha_2 = Pr\{\theta_1 \leq \theta \leq \theta_2\}$$

The above is correct as long as we remember the random quantity of θ in $Pr\{\theta_1 \leq \theta \leq \theta_2\}$ which varies as a function of $\boldsymbol{x}$. Whether or not the above equation is accepted as true, there is definitely a connection with *matching priors*.

Making our way back to bootstrap confidence intervals, they end up providing easily computable confidence densities. Let $\hat{G}(\theta)$ be the bootstrap CDF and $\hat{g}(\theta)$ its density function. The percentile confidence limits $\hat{\theta}[\alpha] = \hat{G}^{-1}(\alpha)$ have $\alpha = \hat{G}(\theta)$ giving $\tilde{g}_x(\theta) = \hat{g}(\theta)$. The bootstrap density ends up actually being the confidence density for the percentile method.

In BC_a intervals, the confidence density is found by reweighting $\hat{g}(\theta)$ using $\tilde{g}_x(\theta) = cw(\theta)\hat{g}(\theta)$ where,

$$w(\theta) = \frac{\varphi[z_\theta/(1+az_\theta) - z_0]}{(1+az_\theta)^2\varphi(z_\theta + z_0)} \qquad with \qquad z_\theta = \Phi^{-1}\hat{G}(\theta) - z_0$$

φ is the standard normal density, Φ is the CDF, and c is the constant that makes $\tilde{g}_x(\theta)$ integrate to 1.

$$W_b = w(\hat{\theta}^{*b}) / \sum_{i=1}^{B} w(\hat{\theta}^{*i})$$

With the above definition of W_b, the BC_a confidence density is the discrete density putting weight W_b on $\hat{\theta}^{*b}$.

In the end, Bayesian analysis is useful in that after examining the data, it allows us to express the remaining uncertainty in the language of probability. This unique ability is not available with many other methods. For confidence intervals, fiducial and confidence densities allow the frequentist to move away from the interpretive limitations of Neyman's intervals.

GLM versus our own logistic regression code

First, we will import our needed data:

```
file <- "http://mathfaculty.fullerton.edu/sbehseta/Ed.txt"
data <- read.table(file, header=TRUE)
attach(data)

X = cbind(model.matrix( ~ School, data=data), data$Math.Scaled.Scores.2013)
Y = ifelse(data$Gender=="F",0,1)
```

Comparison

We will be comparing the output of a GLM between our IRLS code and the built in R function.

Here is the code for the IRLS GLM model:

```
JacfW_Bern = function(beta,X){
  xb = X%*%beta
  exb =  exp(xb)
  p = exb/(1+exb)
  q=1-p
  f=p
  frac = exb/((1+exb)^2)
  J= (diag(as.vector(frac))) %*% X
  W = diag(1/as.vector(p*q))
  list(f=f, J=J, W=W)}

GN = function(y,X, beta0, Jac, Wt = 1, maxit,IRLS = TRUE){
  for (it in 1: maxit){
    a = do.call(Jac, list(beta0, X))
    J = a$J
    f = a$f
    if (IRLS==TRUE){
      Wt = a$W }
    JW = t(J) %*% Wt
    dir = solve(JW %*% J) %*% JW %*% (y-f)
    beta1 = beta0 + dir
    beta0=beta1}
  MLE_beta<<-beta0}
```

Now, we will compute the coefficients of our GLM using our code:

```
beta0 = matrix(rep(0,25),ncol=1)
maxit=10
coef1 = GN(Y,X, beta0, 'JacfW_Bern', Wt = 1, maxit, IRLS = TRUE)
```

Next we compute the coefficients from the built-in GLM function:

```
model= glm(Gender~ School +  Math.Scaled.Scores.2013,data=data, binomial)
```

Here is a side-by-side comparison of the estimated coefficients:

```
coef2 = model$coefficients
coef = cbind(coef2,coef1)
colnames(coef) = c("GLM Function","Our Code")
coef %>%
kable() %>%
kable_styling()
```

	GLM Function	Our Code
(Intercept)	-0.0146178	-0.0146178
SchoolB	-0.0207269	-0.0207269
SchoolC	-0.0626283	-0.0626283
SchoolD	-0.0538871	-0.0538871
SchoolE	-0.0098035	-0.0098035
SchoolF	-0.3565478	-0.3565478
SchoolG	-0.1831949	-0.1831949
SchoolH	-0.2419928	-0.2419928
SchoolI	0.0094157	0.0094157
SchoolJ	0.1124463	0.1124463
SchoolK	-0.0425272	-0.0425272
SchoolL	-0.0280981	-0.0280981
SchoolML	0.0808450	0.0808450
SchoolN	0.0429780	0.0429780
SchoolO	-0.0397597	-0.0397597
SchoolP	0.0079215	0.0079215
SchoolQ	-0.2555387	-0.2555387
SchoolR	-0.4126581	-0.4126581
SchoolS	0.0190138	0.0190138
SchoolT	-0.2694967	-0.2694967
SchoolU	-0.0446347	-0.0446347
SchoolV	-0.4146873	-0.4146873
SchoolW	-0.0164779	-0.0164779
SchoolX	-0.1172260	-0.1172260
Math.Scaled.Scores.2013	0.0004205	0.0004205

Both methods produced nearly the same values for our Logistic regression coefficients.

Comparison of Error Rates

We will now compare the error rates for the two models

First, we must make training and testing data.

```
n =dim(data)[1]
train.index = sample(seq(1:n), trunc(0.9*n), replace=FALSE)
train.data = data[train.index,]
test.data = data[-train.index,]
```

Using our GLM function, we will build a Logistic Regression Model for the training data.

```
model.glm = glm(Gender ~ School + Math.Scaled.Scores.2013, data=train.data, family="binomial")
```

Now, we can make predictions for the test data using our trained model:

```
pred.glm = ifelse(predict(model.glm, newdata = test.data, type="response")>0.5,"M","F")
```

We are able to compute the error rate for the GLM by dividing the number of correct predictions by the total number of predictions:

```
error.glm = sum(test.data$Gender!=pred.glm)/length(pred.glm)
error.glm
```

```
## [1] 0.4457275
```

For our IRLS model, we will once again build a model using our training data:

```
beta0 = matrix(rep(0,25),ncol=1)
maxit=10

train.X = cbind(model.matrix( ~ School, data=train.data), train.data$Math.Scaled.Scores.2013)
train.Y = ifelse(train.data$Gender=="F",0,1)
coef= GN(train.Y,train.X, beta0, 'JacfW_Bern', Wt = 1, maxit, IRLS = TRUE)
```

Having computed our coefficients, we can directly compute the response values in order to make predictions:

```
#Model that can be used for predictions
test.X = cbind(model.matrix( ~ School, data=test.data), test.data$Math.Scaled.Scores.2013)
expb = exp(test.X %*% coef)
pi = expb/(1+expb)

pred.irls = ifelse( pi>0.5,"M","F")
```

We are able to compute the error rate for the IRLS by dividing the number of correct predictions by the total number of predictions:

```
error.irls = sum(test.data$Gender!=pred.irls)/length(pred.irls)
error.irls
```

```
## [1] 0.4457275
```

We can see that both error rates are the same, which is not surprising because the two methods produce the same model.

Model Selection

We will compute AIC for all three possible models and choose the model with minimized AIC:

```
model_Full= glm(Gender~ School +  Math.Scaled.Scores.2013, binomial)
model_Math = glm(Gender ~ Math.Scaled.Scores.2013, binomial)
model_School = glm(Gender ~ School, binomial)
```

```
#stepAIC(model_Full)
AIC(model_Full)
```

```
## [1] 6023.845
```

```
AIC(model_Math)
```

```
## [1] 6000.133
```

```
AIC(model_School)
```

```
## [1] 6023.232
```

It appears that the model only including Math Test Scores as a predictor produces the lowest AIC. Therefore, via our model selection process, this single predictor model is the optimal choice.

Multinomial

Changing the response to School, we will create a Multinomial Logistic Model:

```r
X = cbind(model.matrix( ~ School, data=data), data$Math.Scaled.Scores.2013)
Gender = ifelse(data$Gender=="F",0,1)

multi =  multinom(School ~ Gender +  Math.Scaled.Scores.2013)
```

```
## # weights:  96 (69 variable)
## initial  value 13754.616978
## iter  10 value 13667.920340
## iter  20 value 13620.959377
## iter  30 value 13620.583859
## iter  30 value 13620.583768
## iter  40 value 13620.531762
## iter  50 value 13598.297621
## iter  60 value 13597.683754
## iter  70 value 13586.433899
## iter  70 value 13586.433881
## final  value 13586.318091
## converged
```

```r
summary(multi)
```

```
## Call:
## multinom(formula = School ~ Gender + Math.Scaled.Scores.2013)
##
## Coefficients:
##     (Intercept)       Gender Math.Scaled.Scores.2013
## B   0.578533192  0.115497595           -8.010186e-04
## C  -0.717585837  0.001009719            2.592140e-03
## D  -0.758721514 -0.022868264            1.633437e-03
## E   0.425057193  0.113532653           -8.185213e-04
## F   0.245667011 -0.217972532           -7.209188e-04
## G   0.054527150 -0.064725224           -4.335136e-04
## H  -1.106449715 -0.212964022            2.934651e-03
## I   1.075027019  0.168724544           -2.745318e-03
## J  -0.440860403  0.162375101            3.685477e-04
## K  -0.463469024  0.012263606            6.377713e-04
## L   0.581423391  0.158307595           -3.786686e-04
## ML -0.007079831  0.182923271           -4.612203e-04
## N  -0.129474222  0.134975701           -1.041681e-04
## O  -0.289896440  0.044346548            7.397813e-04
## P   0.424560136  0.146742504           -1.395684e-03
## Q   0.515503930 -0.151714518           -1.594207e-04
## R  -0.066264511 -0.285853129            8.694777e-04
## S  -0.052738946  0.114811624            5.598022e-04
## T  -0.038952243 -0.158257406           -9.125314e-05
## U   0.763857979  0.099519494           -1.276597e-03
## V  -0.446068778 -0.311861500            1.612568e-03
## W   0.527494774  0.120219326           -1.374391e-03
## X   0.433201458  0.017076977           -9.186899e-04
##
## Std. Errors:
##    (Intercept)    Gender Math.Scaled.Scores.2013
## B    0.2934479 0.1998077            0.0008113891
## C    0.3153628 0.2020292            0.0008231269
```

```
## D    0.3820327 0.2246507           0.0009910163
## E    0.3163385 0.2071782           0.0008667093
## F    0.3652184 0.2242508           0.0009914463
## G    0.3697023 0.2244214           0.0009950258
## H    0.3729705 0.2207374           0.0009528150
## I    0.3055731 0.2085633           0.0008626948
## J    0.3898325 0.2297353           0.0010258422
## K    0.3902561 0.2291390           0.0010260913
## L    0.2717746 0.1925846           0.0007546384
## ML   0.3609234 0.2215990           0.0009682867
## N    0.3654646 0.2222091           0.0009751444
## O    0.3505740 0.2154856           0.0009289601
## P    0.3429463 0.2176061           0.0009385843
## Q    0.2900475 0.1979531           0.0008002235
## R    0.3311007 0.2101524           0.0008865876
## S    0.3176546 0.2051924           0.0008522344
## T    0.3732962 0.2252704           0.0010003214
## U    0.2904438 0.1997314           0.0008099305
## V    0.3509581 0.2158682           0.0009232636
## W    0.3276875 0.2122356           0.0009013960
## X    0.3273128 0.2109436           0.0008966338
##
## Residual Deviance: 27172.64
## AIC: 27310.64
```