ComS 573: Homework #3

Due on March 28, 2014

Professor De Brabanter at 10am

Josh Davis

Problem 1

From ISLR: Chapter 6, Problem 7.

Suppose that $Y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + e_i$ where $e_1, \ldots, e_n$ are i.i.d. distributed from a $\mathcal{N}(0, \sigma_e^2)$.

Part A

Write out the likelihood for the data and show that it is equivalent to using ordinary least squares.

Solution

First let's remind ourselves what the ordinary least squares solution is. It is the set of coefficients that minimize the residual sum of squares, or:

$$\text{RSS} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

where $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \cdots + \hat{\beta}_p x_{ip}$, our model.

To show that the maximum likelihood is equal to the least squares solution when the error is distributed from a normal, let's look at what maximum likelihood means.

Often it is useful to calculate the statistical parameters, θ, of a model. However this can be hard to do given data from the model, X. Written out, the probability of getting the data we got using the parameters given the data or $\Pr[\beta \mid \theta]$.

The idea behind the likelihood is that we want to determine the probability in the reverse and think about given the data that we got, what is most likely the values of the parameters, or $\mathcal{L}(\theta \mid X)$. Putting this together we get:

$$\mathcal{L}(\theta \mid \beta_1, \ldots, \beta_n) = P(\beta_1, \ldots, \beta_n \mid \theta)$$

Given that we know these i.i.d. which means we can that our joint probability is just equal to product of the probabilities, thus we get:

$$\begin{aligned}
\mathcal{L}(\theta \mid \beta_1, \ldots, \beta_n) &= P(\beta_1 \mid \theta) \times \cdots \times P(\beta_n \mid \theta) \\
&= \prod_{i=1}^{n} P(\beta_i \mid \theta)
\end{aligned}$$

Since we know that $e_i \sim \mathcal{N}(0, \sigma_e^2)$ and that $e_i = Y_i - \hat{Y}_i$, we can use the Normal distribution probability density function which gives us:

$$\begin{aligned}
\mathcal{L}(\theta \mid \beta_1, \ldots, \beta_n) &= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(Y_i - \hat{Y}_i)^2}{2\sigma^2} \right) \\
&= \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \prod_{i=1}^{n} \exp\left(-\frac{(Y_i - \hat{Y}_i)^2}{2\sigma^2} \right)
\end{aligned}$$

To simplify the expression, we can take the log of each side. This gives us an expression that is easier to

Problem 1 continued on next page...

work with and we get:

$$\begin{aligned}\log \mathcal{L}(\theta \mid \beta_1, \dots, \beta_n) &= \log \left[\left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \prod_{i=1}^{n} \exp\left(-\frac{(Y_i - \hat{Y}_i)^2}{2\sigma^2} \right) \right] \\ &= n \log \left(\frac{1}{\sigma\sqrt{2\pi}} \right) + \sum_{i=1}^{n} \log \left(\exp \left(-\frac{(Y_i - \hat{Y}_i)^2}{2\sigma^2} \right) \right) \\ &= n \log \left(\frac{1}{\sigma\sqrt{2\pi}} \right) + \sum_{i=1}^{n} \left(-\frac{(Y_i - \hat{Y}_i)^2}{2\sigma^2} \right) \\ &= n \log \left(\frac{1}{\sigma\sqrt{2\pi}} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2\end{aligned}$$

Now since we are looking to maximize our Likelihood, or $\mathcal{L}$, we get the following:

$$\log \mathcal{L}(\theta \mid \beta_1, \dots, \beta_n) = n \log \left(\frac{1}{\sigma\sqrt{2\pi}} \right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

We can see that the second term is negative, thus in order to maximize $\mathcal{L}$, we want to minimize the second term. If we remove the constant, we then want to minimize:

$$\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

Well, this is nice because this is our old friend, RSS, from the beginning of this problem. The RSS is what the ordinary least squares aims to minimize. Thus we can see that when our errors, e_i, are from a Normal distribution, the maximum likelihood is the same as ordinary least squares.

Problem 1 continued on next page...

Part B

Assume the following prior for β: $\beta_1, \ldots, \beta_p$, are i.i.d. according to a Laplace distribution with mean zero and common scale parameter c, i.e., $h(\beta) = \frac{1}{2c}\exp(-|\beta|/c)$. You can assume that $Y_i = \beta_0 + \sum_{j=1}^{p} x_{ij}\beta_j + e_i$.

Write out the posterior for β in this setting. Argue that the LASSO estimate is the mode for β i.e., the most likely value for β, under this posterior distribution. Determine the value for the parameter λ in the LASSO cost function.

Solution

We want to determine the posterior distribution. We know that the probability $p(\beta \mid X, Y)$ is going to be proportaional to this posterior distirbution. This gives us:

$$p(\beta \mid \mathbf{X}, \mathbf{Y}) \propto f(\mathbf{Y} \mid \mathbf{X}, \beta)p(\beta \mid \mathbf{X}) = f(\mathbf{Y} \mid \mathbf{X}, \beta)p(\beta)$$

where f is the likelihood from part (a) except keeping it in matrix form which is $(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta)$, and $p(\beta)$ is our density. Substituting these values gives us the following:

$$f(\mathbf{Y} \mid \mathbf{X}, \beta)p(\beta) = \left(\frac{1}{2c\sigma\sqrt{2\pi}}\right)\exp\left(-\frac{(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta)}{2\sigma^2} - |\beta|/c\right)$$

Like usual, let's take the log of this equation to make it easier to work with:

$$\begin{aligned}\log f(\mathbf{Y} \mid \mathbf{X}, \beta)p(\beta) &= \log\left[\left(\frac{1}{2c\sigma\sqrt{2\pi}}\right)\exp\left(-\frac{(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta)}{2\sigma^2} - |\beta|/c\right)\right] \\ &= \log\left(\frac{1}{2c\sigma\sqrt{2\pi}}\right) - \frac{(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta)}{2\sigma^2} - |\beta|/c\end{aligned}$$

Before we move on, let's remind ourselves what the LASSO is a minimization of the following in terms of β for a given λ:

$$\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j - x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$

By letting $c = 2\sigma^2/\lambda$ and substituting it back into our equation, we get:

$$\log f(\mathbf{Y} \mid \mathbf{X}, \beta)p(\beta) = \log\left(\frac{1}{2c\sigma\sqrt{2\pi}}\right) - \frac{(\mathbf{Y} - \mathbf{X}\beta)^T(\mathbf{Y} - \mathbf{X}\beta) - \lambda|\beta|}{2\sigma^2}$$

Since we want to maximize the posterior, it's obvious that we want to minimize the second term. Thus we are minimizing the numerator which is exactly the same problem of the LASSO. Therefore when the prior has a Laplace distribution, LASSO is the mode for β.

As was stated before, if we set $c = 2\sigma^2/\lambda$, then the following holds. Thus the tuning parameter that we want to use is then $\lambda = 2\sigma^2/c$.

Problem 2

Suppose we estimate some statistics (e.g. median) based on a sample X.

Part A

Carefully describe how you might estimate the standard deviation of the median of the statistic.

Solution

We want to determine the standard deviation of the median. We know that this median is of the population yet we only have a sample, X. This looks like a perfect situation where we can use the Bootstrap.

The Bootstrap works by taking some number of elements from our sample *with replacement*. The replacement part is important because if it was not with replacement, we'd be introducing more variance. We do this re-sampling a large number of times and let this number equal B. This value of B works best when it is in the thousands.

Each time we pull out a new re-sampling of our original sample, X, we calculate the median of this sample. We store this and then continue on until we are done with all B re-sampling. At this point, we can then calculate our standard deviation. And we hope that it will be close to the true value.

A figure of this process is below:

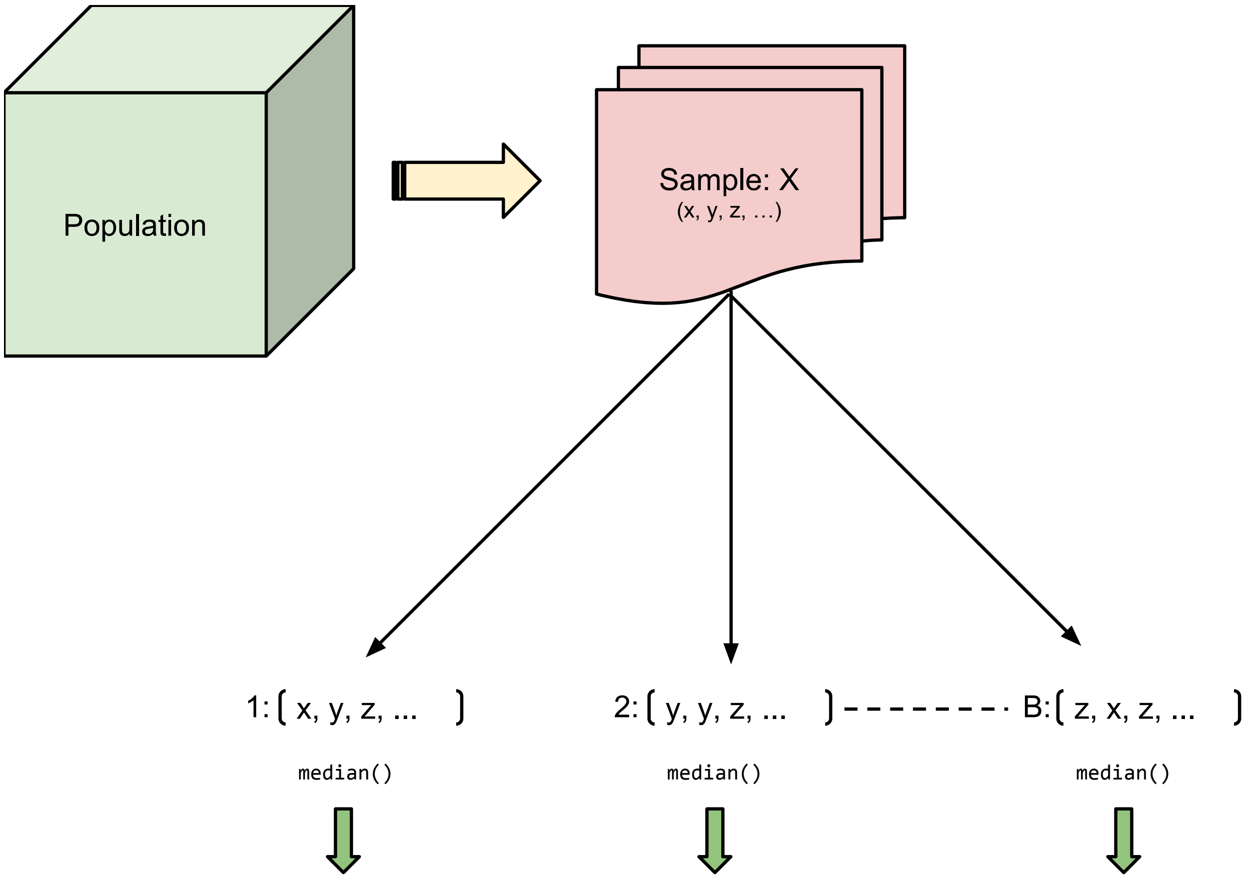

Figure 1: Illustration of the Bootstrap.

Problem 2 continued on next page. . .

Part B

Write R code that calculates the standard deviation of the median given a sample X.

Solution

```
# Load the data
lawstat <- read.table("lawstat.dat")

# Consistency...
set.seed(1)

# Sample obuseration indices
s <- sample(1:nrow(lawstat), 15, replace = TRUE)
s

##  [1] 22 31 47 75 17 74 78 55 52  6 17 15 57 32 64

# Given data of 15 observations
X <- lawstat[s, ]
L <- length(X$GPA)

# Manual Bootstrap

# Bootstrap 4k times
B <- 4000

results <- rep(0, B)

# I choose you, Bootstrap!
for (i in 1:B) {
    obs <- round(L * runif(L, 0, 1))
    sample <- X[obs, ]
    results[i] <- median(sample$GPA)
}

sd(results)

## [1] 0.06355

# Using Boot Library

boot.fn <- function(data, indices) {
    sample <- X[indices, ]
    median(sample$GPA)
}

results <- boot(data = X, statistic = boot.fn, R = B)
results

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
```

Problem 2 continued on next page. . .

```
##
##
## Call:
## boot(data = X, statistic = boot.fn, R = B)
##
##
## Bootstrap Statistics :
##     original   bias    std. error
## t1*     3.19 0.009232     0.06118
```

```
plot(results)
```

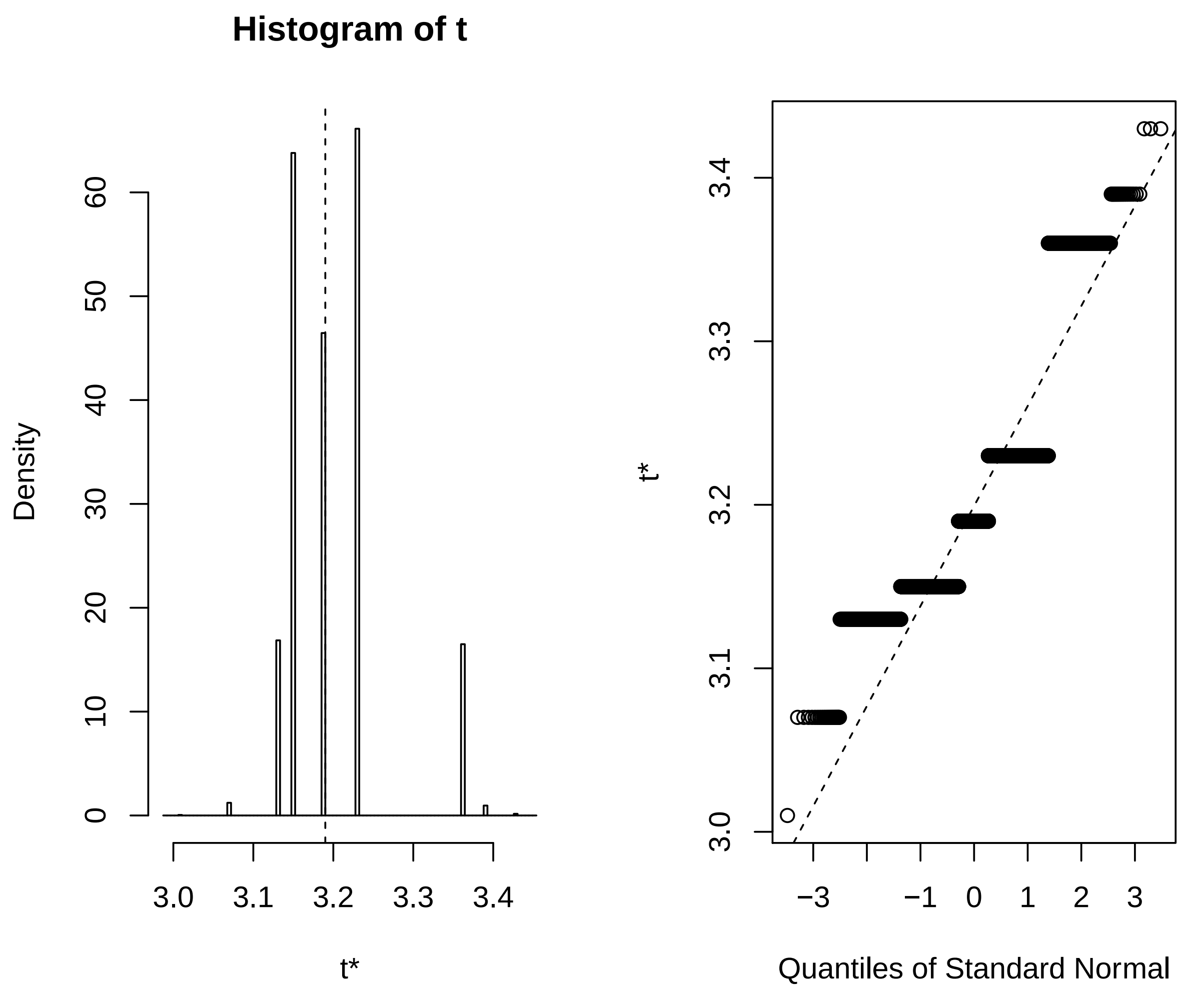

Figure 2: Confidence interval plots

Problem 2 continued on next page. . .

```
# Calculate the median from the results
sd(results$t[, 1])

## [1] 0.06118
```

Part C

Suppose you were interested in a $100(1-\alpha)$ (pointwise) confidence interval for the correlation coefficient of a sample of X and Y (the joint distribution of X and Y is not bivariate normal). Clearly explain and derive how you would do this? Write R code that calculates the 95% confidence interval for the correlation coefficient in case of the lawstat.dat data.

Solution

The idea is identical to above. However, instead of the median, we want the confidence interval for the correlation coefficient. Once again we take B replacement re-samples and then calculate the correlation of them. We repeat this over and over and then can come up with the confidence intervals.

```
# Bootstrap 4k times
B <- 4000

boot.fn <- function(data, indices) {
    sample <- X[indices, ]
    cor(sample$LSAT, sample$GPA)
}

results <- boot(data = X, statistic = boot.fn, R = B)

results

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = X, statistic = boot.fn, R = B)
##
##
## Bootstrap Statistics :
##     original    bias    std. error
## t1*   0.7787 -0.006404      0.1303
```

```
plot(results)
```

Problem 2 continued on next page...

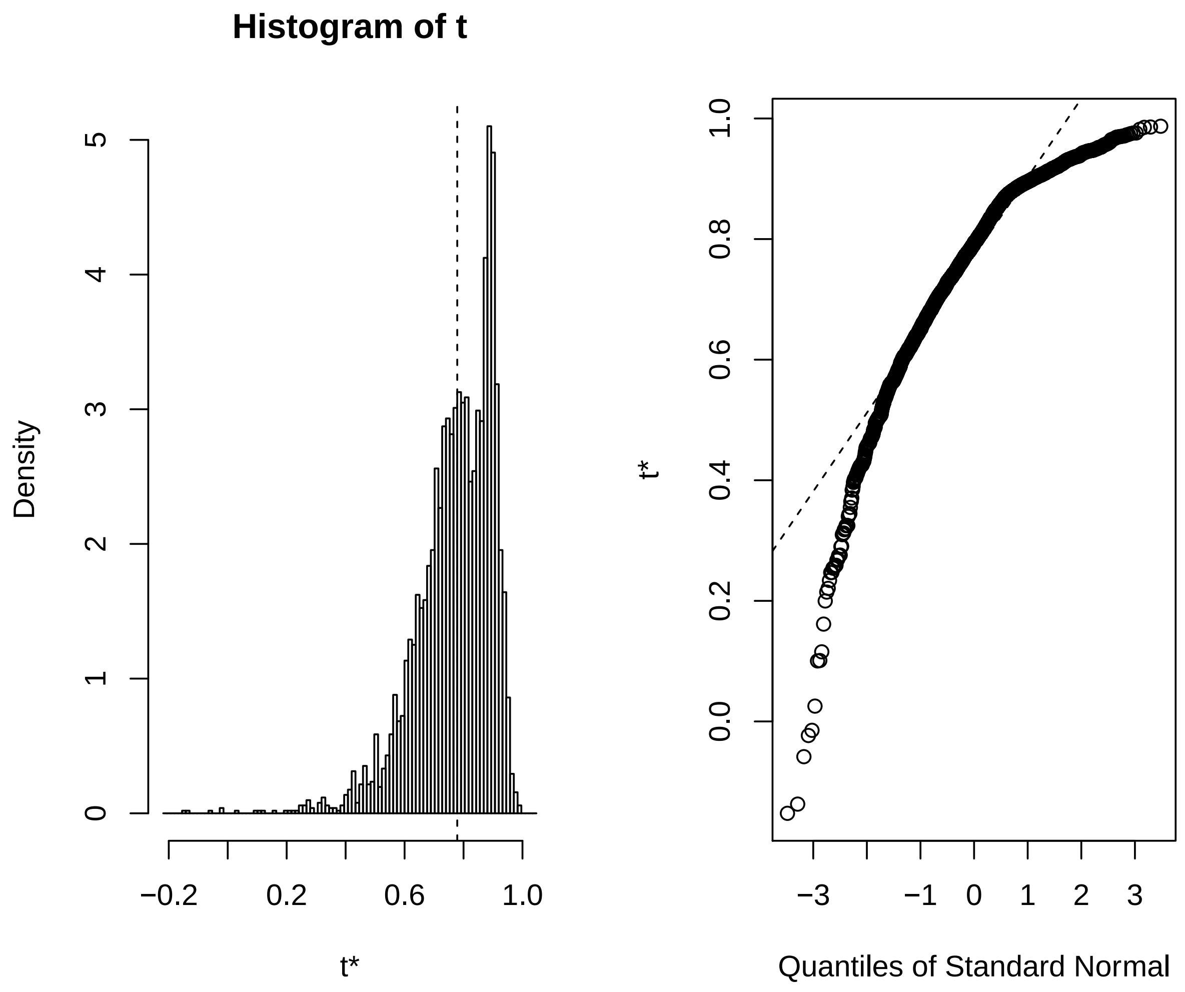

Figure 3: Confidence interval plots

```
boot.ci(results, type = "bca")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 4000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = results, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.2543,  0.9178 )
## Calculations and Intervals on Original Scale
```

Problem 3

From ISLR: Chapter 6, Problem 11.

Using the Boston Housing data set (in the MASS library) complete the following.

Part A

Try out some of the regression methods explore in Chapter 6 of the textbook. These include best subset selection, the lasso, ridge regression, and PCR. Present and discuss results for the approaches that you consider.

Solution

First let's setup a few variables that we'll use through out.

```r
set.seed(1)

# Inital setup

# Grab X and Ys
x <- model.matrix(medv ~ ., Boston)[, -1]
y <- Boston$medv

# Test/Train
train <- sample(1:nrow(x), nrow(x)/2)
test <- (-train)

x.train <- x[train, ]
y.train <- y[train]

x.test <- x[test, ]
y.test <- y[test]
```

Problem 3 continued on next page. . .

Best Subset Selection

Next let's try the best subset selection on our data:

```
library("leaps")
```

```
set.seed(1)

p <- ncol(Boston) - 1

predict.regsubsets = function(object, newdata, id, ...) {
    form = as.formula(object$call[[2]])
    mat = model.matrix(form, newdata)
    coefi = coef(object, id = id)
    mat[, names(coefi)] %*% coefi
}

best.train <- sample(c(TRUE, FALSE), nrow(Boston), rep = TRUE)
best.test <- (!best.train)

fit.best <- regsubsets(medv ~ ., data = Boston[best.train, ], nvmax = p)

cv.errors = rep(0, p)
for (i in 1:p) {
    pred = predict.regsubsets(fit.best, Boston[best.test, ], id = i)

    cv.errors[i] = mean((Boston$medv[best.test] - pred)^2)
}

mean.best <- mean(cv.errors)
mean.best

## [1] 34.88
```

```
# Plot using the regsubsets() plots
par(mfrow = c(2, 2))

# RSS
plot(fit.best, scale = "r2")
plot(fit.best, scale = "adjr2")
plot(fit.best, scale = "Cp")
plot(fit.best, scale = "bic")
```

Problem 3 continued on next page. . .

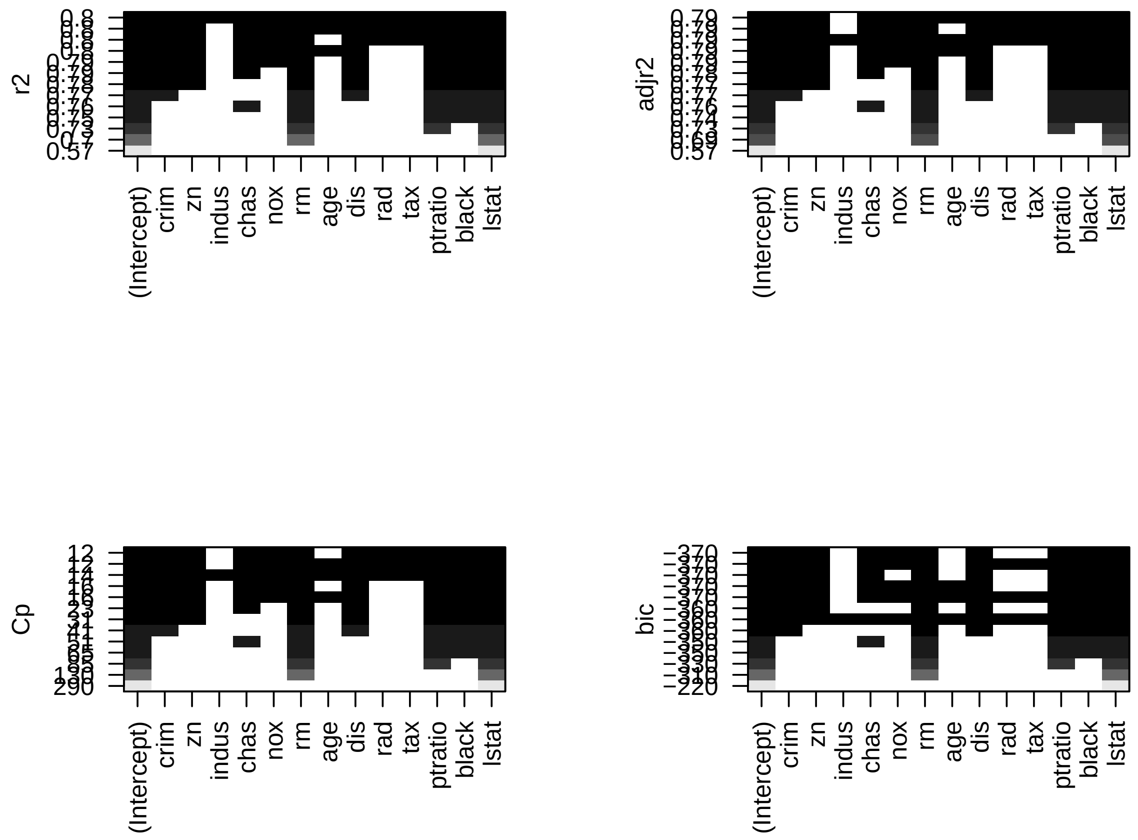

Figure 4: Best Subset Plots

Problem 3 continued on next page. . .

Ridge Regression

```r
library("glmnet")
```

```r
set.seed(1)
fit.ridge.full <- glmnet(x, y, alpha = 0)
fit.ridge <- cv.glmnet(x.train, y.train, alpha = 0)
```

```r
# Visualize how the coefficients shrink with increasing lambda
plot(fit.ridge)
```

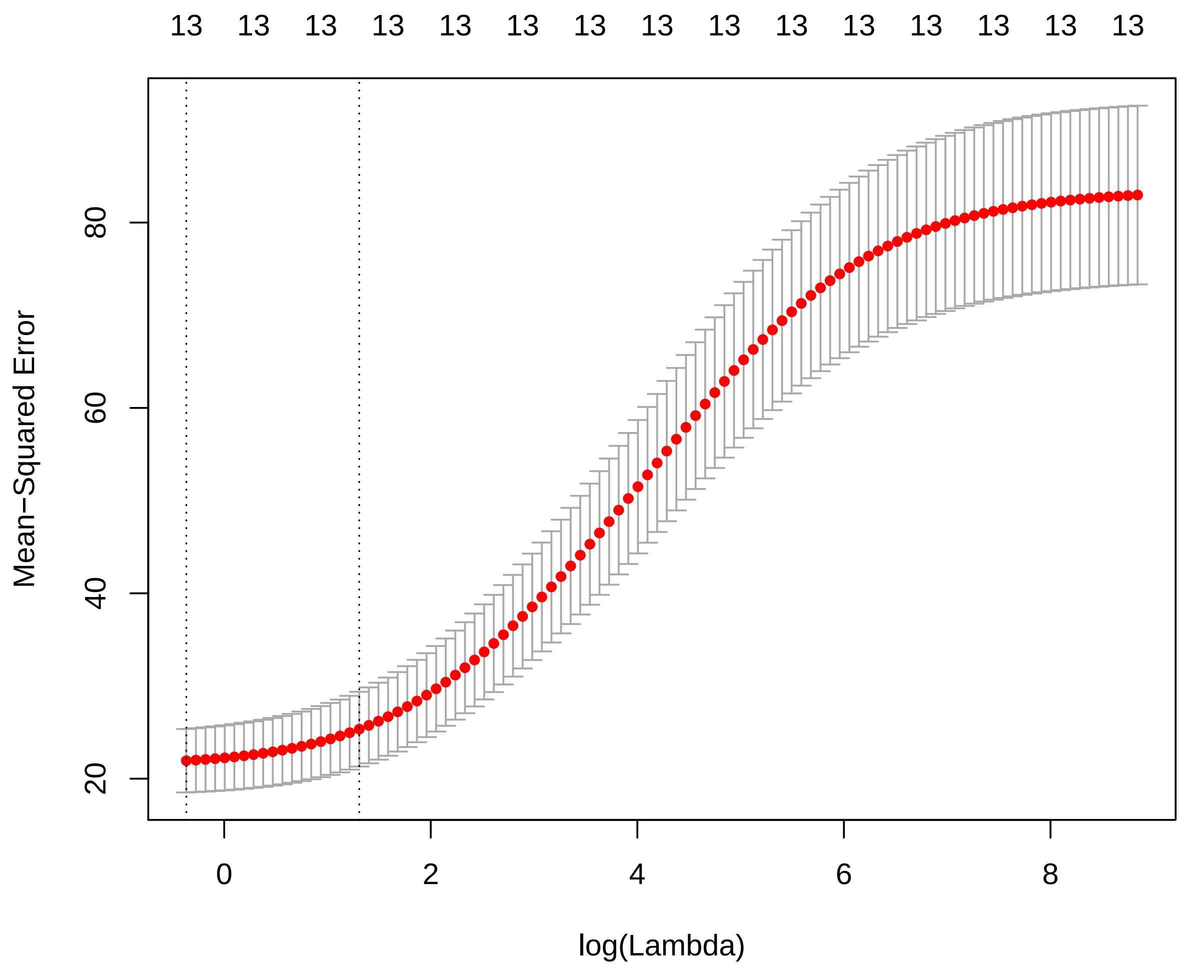

Figure 5: Ridge Regression Cross Validated

Problem 3 continued on next page. . .

```r
lambda.tune <- fit.ridge$lambda.min

# Tuned lambda
lambda.tune

## [1] 0.6928
```

Now let's take a look at the errors:

```r
ridge.pred <- predict(fit.ridge.full, s = lambda.tune, newx = x.test)

mean.ridge <- mean((ridge.pred - y.test)^2)
mean.ridge

## [1] 24.34

# Our final model
predict(fit.ridge.full, type = "coefficients", s = lambda.tune)

## 14 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  27.893933
## crim         -0.087329
## zn            0.032513
## indus        -0.038530
## chas          2.901233
## nox         -11.832766
## rm            4.012296
## age          -0.003782
## dis          -1.113115
## rad           0.152118
## tax          -0.005694
## ptratio      -0.853473
## black         0.009067
## lstat        -0.471493
```

Problem 3 continued on next page...

LASSO

```
library("glmnet")
```

```
set.seed(1)

lambda <- seq(0, 10000)
fit.lasso.full <- glmnet(x, y, alpha = 1, lambda = lambda)
```

```
plot(fit.lasso.full)
```

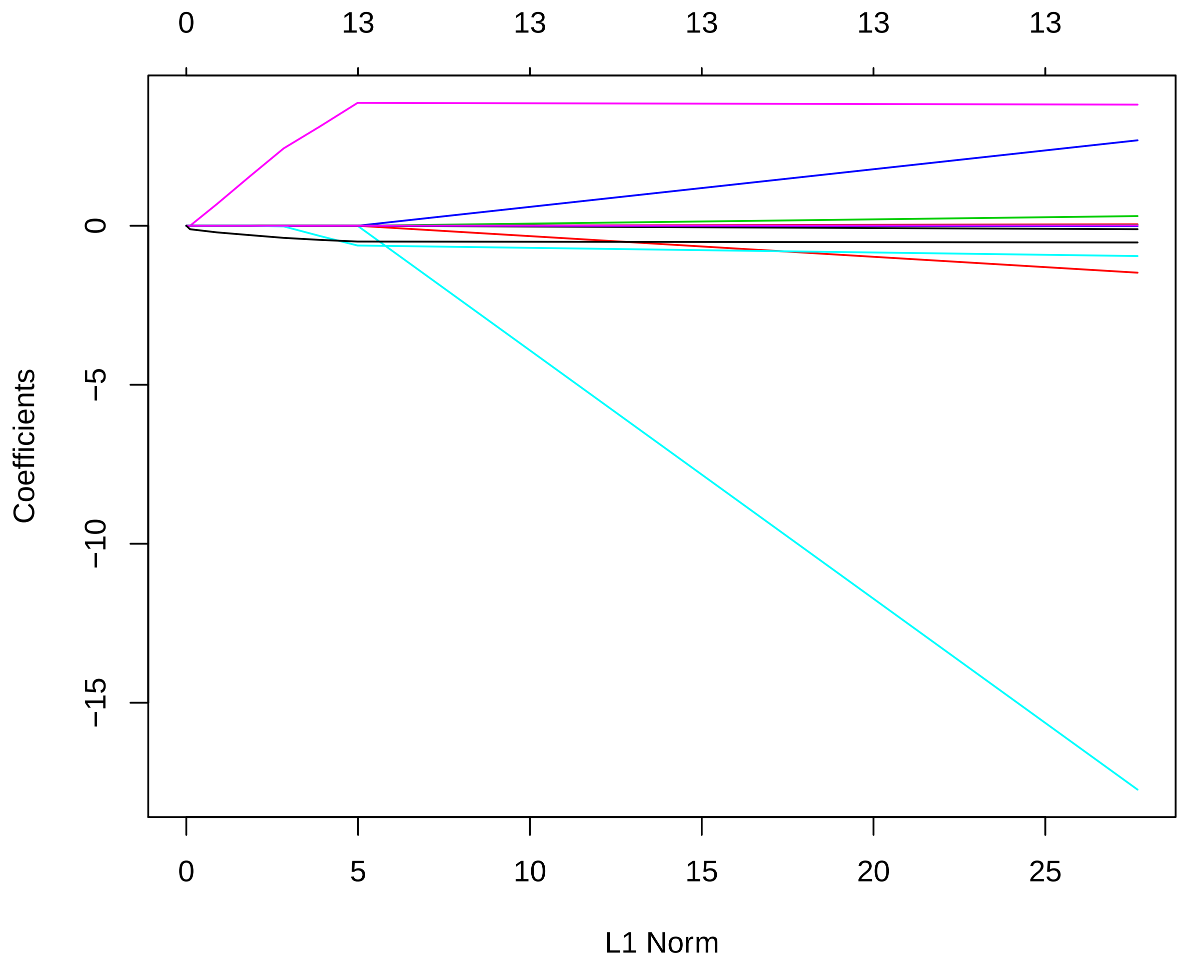

Figure 6: Lasso

```r
fit.lasso <- cv.glmnet(x.train, y.train, alpha = 1)
```

```r
plot(fit.lasso)
```

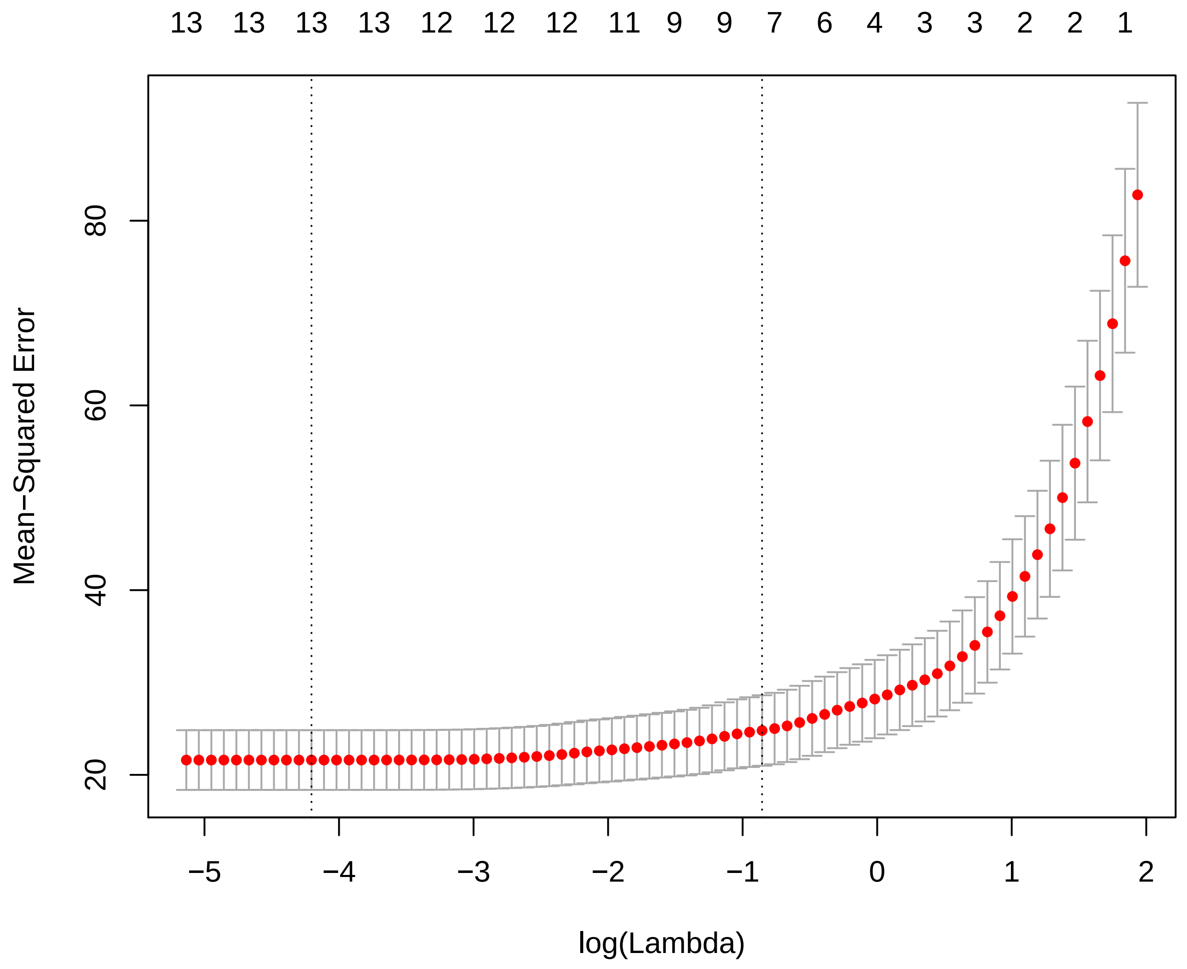

Figure 7: Cross Validated Lasso

```r
lambda.tune <- fit.lasso$lambda.min

# Tuned lambda
lambda.tune

## [1] 0.01493
```

Now let's take a look at the errors:

Problem 3 continued on next page. . .

```
lasso.pred <- predict(fit.lasso.full, s = lambda.tune, newx = x.test)

mean.lasso <- mean((lasso.pred - y.test)^2)
mean.lasso

## [1] 24.23

# Our final model
predict(fit.lasso.full, type = "coefficients", s = lambda.tune)

## 14 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept)  3.611e+01
## crim        -1.062e-01
## zn           4.572e-02
## indus        1.979e-02
## chas         2.648e+00
## nox         -1.747e+01
## rm           3.811e+00
## age          6.994e-04
## dis         -1.453e+00
## rad          3.004e-01
## tax         -1.211e-02
## ptratio     -9.470e-01
## black        9.198e-03
## lstat       -5.245e-01
```

Problem 3 continued on next page. . .

PCR

```
library("pls")
```

```
set.seed(1)

fit.pcr.full <- pcr(medv ~ ., data = Boston, scale = TRUE, validation = "CV")

fit.pcr <- pcr(medv ~ ., data = Boston, scale = TRUE, subset = train, validation = "CV")

summary(fit.pcr)

## Data:  X dimension: 253 13
##  Y dimension: 253 1
## Fit method: svdpc
## Number of components considered: 13
##
## VALIDATION: RMSEP
## Cross-validated using 10 random segments.
##        (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps
## CV           9.124    7.108    6.975    5.616    5.539    5.129    5.126
## adjCV        9.124    7.105    6.974    5.602    5.525    5.103    5.113
##        7 comps  8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## CV       5.090    5.027    5.023     5.112     4.970     4.805     4.712
## adjCV    5.081    5.014    5.019     5.090     4.947     4.785     4.692
##
## TRAINING: % variance explained
##       1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X       48.28    59.06    67.38    74.12    80.51    86.04    90.29
## medv    39.49    43.87    64.19    65.83    69.78    69.94    70.93
##       8 comps  9 comps  10 comps  11 comps  12 comps  13 comps
## X       93.34    95.22     96.92     98.39     99.55     100.0
## medv    72.59    72.87     73.94     75.14     76.06      77.1
```

```
# Plot validation
validationplot(fit.pcr, val.type = "MSEP", main = "Cross Validation Scores for PCR")
```

Problem 3 continued on next page. . .

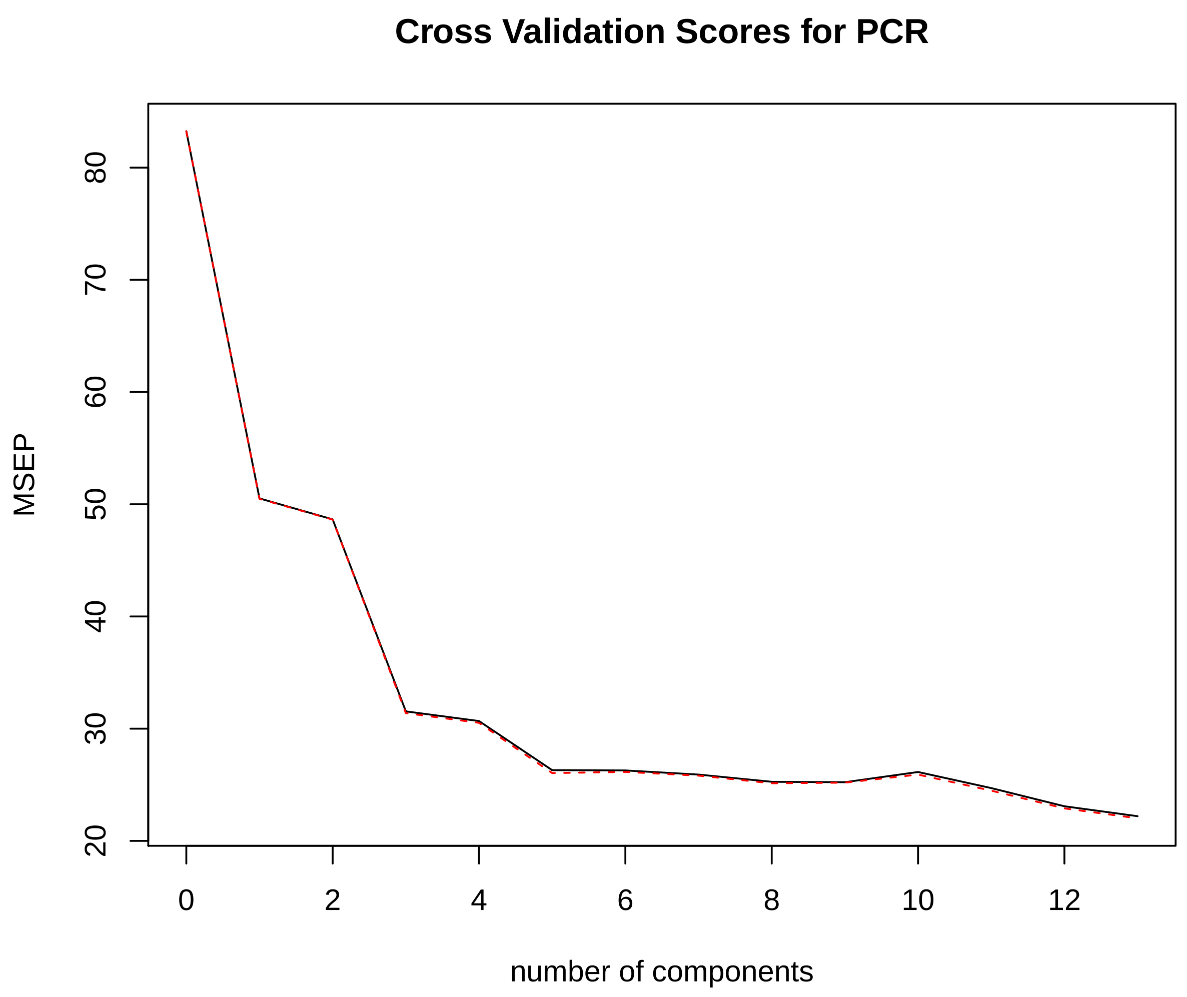

Figure 8: PCR Fit.

```
res <- RMSEP(fit.pcr)
pcr.best <- which.min(res$val[1, , ]) - 1
pcr.best

## 13 comps
##       13

pcr.pred <- predict(fit.pcr, x.test, ncomp = pcr.best)
mean.pcr <- mean((pcr.pred - y.test)^2)
mean.pcr

## [1] 26.29
```

According to the summary and the graph, the smallest RMSEP for the CV and adjCV, is the model with 13 components.

Problem 3 continued on next page. . .

This is the maximum number of components that can be in our model. Thus we might want to trade off for a simpler model with less components. If we look at the summary, it levels off around 5 components for awhile until 11 when it drops down until it reaches 13. Picking 5 might be a good option if we want a simpler model.

Part B

Propose a model (or a set of models) that seem to perform well on this data set, and justify your answer. Clearly explain what you will do.

Solution

Looking at our models, we get the following values:

1. Best Subset: 34.885
2. Ridge: 24.3366
3. Lasso: 24.2335
4. PCR: 26.2868

As we can see, the Lasso and Ridge models are very close when you consider the cross validated mean squared errors. With PCR just a bit higher than Lasso/Ridge. Our Best subset model is off by a large amount in comparison.

I'd propose that we use Ridge or Lasso on this data. Given by the mean square errors, they perform better than the other two methods.

Problem 4

Describe how you can efficiently solve the least squares linear system $(\mathbf{X}^T\mathbf{X})\beta = \mathbf{X}^T\mathbf{Y}$ (i.e., by not calculating an interverse) where $\mathbf{X} \in \mathbb{R}^{n\times p}$ has p linearly independent columns, $\beta \in \mathbb{R}^p$ and $\mathbf{Y} \in \mathbb{R}^{n\times 1}$?

Hint: Think in terms of matrix decompositions (it's not SVD!). Use Wikipedia.

Solution

To improve upon the previous ways of solving the least squares linear system, we can use a more advanced decomposition to handle this. There are primarily two ways that are used. The first is called the Cholesky decomposition that works the fastest out of two. The second is using QR decomposition which is slower than Cholseky but is more numerically stable than Cholesky.

We'll show the Cholesky below which goes like this:

1. Calculate and let $A = \mathbf{X}^T\mathbf{X}$.
2. Determine the Cholesky factorization for A, let this equal $\mathbf{L}\mathbf{R}^T$.
3. Now on the right side, calculate $b = \mathbf{X}^T\mathbf{Y}$.
4. Then solve $\mathbf{L}z = b$ by forward substitution
5. Then solve $\mathbf{R}^T\beta = z$ by back substitution.

Forward/back substitution is an interative process for lower/upper triangular matrices. While the link to the details can be read about below, I won't solve it here. Hopefully that isn't required as the question says to "Describe" and not to show/solve.

Reference

1. Explanation of the forward/back substitution techniques for matrices: http://en.wikipedia.org/wiki/Triangular_matrix#Forward_and_back_substitution