ECON 187: Project 1

Austin Pham (905318112), Shannan Liu (305172952), Christina Zhang(605325840), Zachary Wrubel (20

For this project you will need to use two different datasets of your choice. One will be used for classification and the other for regularization. For the classification dataset make sure you have more than two classes. For your regularization dataset, since the methods focus on variable selection, please make sure to have as many predictors as possible (e.g., 10s or 100s).

Classification

```
body <- read_csv("bodyPerformance.csv")
body <- body %>% mutate(gender = factor(ifelse(gender == "M", 1, 0)))
sum(is.na(body))
```

```
## [1] 0
```

```
head(body)
```

```
## # A tibble: 6 x 12
##     age gender height_cm weight_kg `body fat_%` diastolic systolic gripForce
##   <dbl> <fct>      <dbl>     <dbl>        <dbl>     <dbl>    <dbl>     <dbl>
## 1    27 1           172.      75.2         21.3        80      130      54.9
## 2    25 1           165       55.8         15.7        77      126      36.4
## 3    31 1           180.      78           20.1        92      152      44.8
## 4    32 1           174.      71.1         18.4        76      147      41.4
## 5    28 1           174.      67.7         17.1        70      127      43.5
## 6    36 0           165.      55.4         22          64      119      23.8
## # ... with 4 more variables: `sit and bend forward_cm` <dbl>,
## #   `sit-ups counts` <dbl>, `broad jump_cm` <dbl>, class <chr>
```

Below are the graphs that shows the possible correlation between all the predictors with Class A,B,C,and D. From the grpahs, we can see that all the correlations are significant, so we will be using all the predictors for classification analysis.

```
plot1 <- body %>% ggpairs(columns = c(1:4,12), ggplot2::aes(col = class, fill = class))
plot2 <- body %>% ggpairs(columns = c(5:8,12), ggplot2::aes(col = class, fill = class))
plot3 <- body %>% ggpairs(columns = c(9:12), ggplot2::aes(col = class, fill = class))
plot1
plot2
plot2
```

Logistic Regression

```r
set.seed(42)
fit.control.cv <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
fit.control.boot <-  trainControl(method = "boot")
mn.fit.cv <- train(class ~ ., data = body, method = "multinom",
                   trControl = fit.control.cv, trace = FALSE,
                   preProcess = c("center", "scale"))
mn.fit.boot <- train(class ~ ., data = body, method = "multinom",
                     trControl = fit.control.boot, trace = FALSE,
                     preProcess = c("center", "scale"))
```

```r
mn.fit.cv$finalModel
```

```
## Call:
## nnet::multinom(formula = .outcome ~ ., data = dat, decay = param$decay,
##     trace = FALSE)
##
## Coefficients:
##   (Intercept)       age  gender1   height_cm weight_kg `\\`body fat_%\\``
## B   1.6436941 -1.031072 1.228708 -0.07014251 0.8745149        0.029745697
## C   1.8396621 -1.891377 1.977937  0.11469195 1.2557692        0.005482127
## D   0.4281742 -2.824718 2.598717 -0.30909023 2.4963687        0.459483497
##    diastolic    systolic gripForce `\\`sit and bend forward_cm\\``
## B 0.05764353 -0.01869305 -0.920936                       -1.331405
## C 0.13202220 -0.07684313 -1.519526                       -2.285391
## D 0.25219705 -0.14232365 -2.101412                       -3.597352
##   `\\`sit-ups counts\\`` `\\`broad jump_cm\\``
## B              -1.423940            -0.6803174
## C              -2.603735            -1.1445756
## D              -4.161081            -1.2372507
##
## Residual Deviance: 23128.6
## AIC: 23200.6
```

```r
mn.fit.boot$finalModel
```

```
## Call:
## nnet::multinom(formula = .outcome ~ ., data = dat, decay = param$decay,
##     trace = FALSE)
##
## Coefficients:
##   (Intercept)       age  gender1   height_cm weight_kg `\\`body fat_%\\``
## B   1.6436941 -1.031072 1.228708 -0.07014251 0.8745149        0.029745697
## C   1.8396621 -1.891377 1.977937  0.11469195 1.2557692        0.005482127
## D   0.4281742 -2.824718 2.598717 -0.30909023 2.4963687        0.459483497
##    diastolic    systolic gripForce `\\`sit and bend forward_cm\\``
## B 0.05764353 -0.01869305 -0.920936                       -1.331405
## C 0.13202220 -0.07684313 -1.519526                       -2.285391
## D 0.25219705 -0.14232365 -2.101412                       -3.597352
##   `\\`sit-ups counts\\`` `\\`broad jump_cm\\``
## B              -1.423940            -0.6803174
```

```
## C               -2.603735            -1.1445756
## D               -4.161081            -1.2372507
##
## Residual Deviance: 23128.6
## AIC: 23200.6
```

```r
confusionMatrix(mn.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.4  5.9  2.0  0.3
##          B  6.1 11.1  5.4  1.3
##          C  0.5  7.1 12.9  4.1
##          D  0.0  0.8  4.7 19.2
##
##  Accuracy (average) : 0.6173
```

```r
confusionMatrix(mn.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.4  6.0  2.0  0.3
##          B  6.1 11.1  5.4  1.3
##          C  0.5  7.0 13.0  4.1
##          D  0.0  0.8  4.6 19.2
##
##  Accuracy (average) : 0.6177
```

```r
mn.fit.cv$results
```

```
##   decay  Accuracy     Kappa  AccuracySD    KappaSD
## 1 0e+00 0.6172122 0.4896154 0.009252803 0.01233862
## 2 1e-04 0.6172122 0.4896154 0.009252803 0.01233862
## 3 1e-01 0.6173367 0.4897813 0.009319778 0.01242802
```

```r
mn.fit.boot$results
```

```
##   decay  Accuracy     Kappa  AccuracySD     KappaSD
## 1 0e+00 0.6177344 0.4903212 0.005948138 0.007938624
## 2 1e-04 0.6177344 0.4903212 0.005948138 0.007938624
## 3 1e-01 0.6177418 0.4903313 0.005897619 0.007870582
```

The logistic regression method shows an overall accuracy of 0.6177.

LDA

```r
lda.fit.cv <- train(class ~ ., data = body, method = "lda",
                    trControl = fit.control.cv, trace = FALSE,
                    preProcess = c("center", "scale"))
lda.fit.boot <- train(class ~ ., data = body, method = "lda",
                      trControl = fit.control.boot, trace = FALSE,
                      preProcess = c("center", "scale"))
```

```r
lda.fit.cv$finalModel
```

```
## Call:
## lda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##            age    gender1    height_cm   weight_kg `body fat_%`   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce `sit and bend forward_cm` `sit-ups counts`
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   `broad jump_cm`
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
##
## Coefficients of linear discriminants:
##                                   LD1         LD2         LD3
## age                       -0.64678307  0.22309394  0.47011181
## gender1                    0.69584600 -1.28409193  1.15159573
## height_cm                 -0.05207736 -0.93671269 -0.35916540
## weight_kg                  0.53802507  1.12001910  0.15508149
## `body fat_%`               0.16993470  0.16486574  0.50244600
## diastolic                  0.06595464  0.07418465 -0.27129251
## systolic                  -0.04539913 -0.07760683  0.30887408
## gripForce                 -0.48748656  0.34068971 -0.19140457
## `sit and bend forward_cm` -0.74403894 -0.28310233  0.43563764
## `sit-ups counts`          -1.04324820  0.33643160  0.13732175
## `broad jump_cm`           -0.26279633  0.68788193  0.04374994
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.9785 0.0195 0.0019
```

```r
lda.fit.boot$finalModel
```

```
## Call:
## lda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##            age     gender1    height_cm   weight_kg `body fat_%`   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce `sit and bend forward_cm` `sit-ups counts`
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   `broad jump_cm`
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
##
## Coefficients of linear discriminants:
##                                   LD1         LD2         LD3
## age                       -0.64678307  0.22309394  0.47011181
## gender1                    0.69584600 -1.28409193  1.15159573
## height_cm                 -0.05207736 -0.93671269 -0.35916540
## weight_kg                  0.53802507  1.12001910  0.15508149
## `body fat_%`               0.16993470  0.16486574  0.50244600
## diastolic                  0.06595464  0.07418465 -0.27129251
## systolic                  -0.04539913 -0.07760683  0.30887408
## gripForce                 -0.48748656  0.34068971 -0.19140457
## `sit and bend forward_cm` -0.74403894 -0.28310233  0.43563764
## `sit-ups counts`          -1.04324820  0.33643160  0.13732175
## `broad jump_cm`           -0.26279633  0.68788193  0.04374994
##
## Proportion of trace:
##    LD1    LD2    LD3
## 0.9785 0.0195 0.0019
```

```r
confusionMatrix(lda.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.2  6.1  2.1  0.3
```

```
##          B  6.2 11.0  5.4  1.4
##          C  0.6  7.4 14.1  5.0
##          D  0.0  0.5  3.4 18.3
##
##  Accuracy (average) : 0.6156
```

```
confusionMatrix(lda.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.2  6.2  2.1  0.3
##          B  6.4 10.9  5.5  1.6
##          C  0.6  7.3 13.9  4.9
##          D  0.0  0.5  3.4 18.3
##
##  Accuracy (average) : 0.6126
```

```
lda.fit.cv$results
```

```
##   parameter Accuracy     Kappa AccuracySD   KappaSD
## 1      none 0.615571 0.4874285 0.01318156 0.0175775
```

```
lda.fit.boot$results
```

```
##   parameter  Accuracy     Kappa  AccuracySD     KappaSD
## 1      none 0.6125766 0.4834807 0.004671325 0.006265792
```

LDA shows an overall accuracy of 0.6126.

QDA

```
qda.fit.cv <- train(class ~ ., data = body, method = "qda",
                    trControl = fit.control.cv, trace = FALSE,
                    preProcess = c("center", "scale"))
qda.fit.boot <- train(class ~ ., data = body, method = "qda",
                      trControl = fit.control.boot, trace = FALSE,
                      preProcess = c("center", "scale"))
```

```
qda.fit.cv$finalModel
```

```
## Call:
## qda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
```

```
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##            age     gender1    height_cm   weight_kg `body fat_%`   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce `sit and bend forward_cm` `sit-ups counts`
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   `broad jump_cm`
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
```

```r
qda.fit.boot$finalModel
```

```
## Call:
## qda(x, grouping = y, trace = FALSE)
##
## Prior probabilities of groups:
##         A         B         C         D
## 0.2499813 0.2499067 0.2500560 0.2500560
##
## Group means:
##            age     gender1    height_cm   weight_kg `body fat_%`   diastolic
## A -0.110591989 -0.15645294 -0.081738935 -0.25349141   -0.3722011 -0.08317758
## B  0.021943176  0.02852539  0.002433177 -0.06986770   -0.1653865 -0.01306765
## C -0.005540626  0.07416582  0.071483362 -0.05749133   -0.0820744 -0.02291650
## D  0.094169521  0.05373205  0.007799443  0.38073302    0.6194521  0.11912909
##      systolic   gripForce `sit and bend forward_cm` `sit-ups counts`
## A -0.06421067  0.15546979                0.73108017       0.56575012
## B  0.02749309  0.08908903                0.26704551       0.20088507
## C -0.02103213 -0.03587942               -0.09680201      -0.07361708
## D  0.05774696 -0.20857978               -0.90094590      -0.69272922
##   `broad jump_cm`
## A      0.31640410
## B      0.13050414
## C     -0.03766627
## D     -0.40906956
```

```r
confusionMatrix(qda.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
```

```
##          A 19.0  6.0  1.9  0.3
##          B  5.5 12.2  5.7  1.6
##          C  0.4  6.2 15.8  4.4
##          D  0.1  0.6  1.6 18.6
##
##  Accuracy (average) : 0.6572
```

```r
confusionMatrix(qda.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.8  6.1  1.9  0.4
##          B  5.7 11.9  5.5  1.7
##          C  0.5  6.3 15.7  4.4
##          D  0.1  0.6  1.7 18.6
##
##  Accuracy (average) : 0.6506
```

```r
qda.fit.cv$results
```

```
##   parameter  Accuracy     Kappa AccuracySD    KappaSD
## 1      none 0.6571846 0.5429133 0.01127552 0.01503324
```

```r
qda.fit.boot$results
```

```
##   parameter  Accuracy    Kappa AccuracySD    KappaSD
## 1      none 0.6505498 0.534103 0.01300762 0.01732938
```

QDA shows an overall accuracy of 0.6549.

kNN

```r
knn.fit.cv <- train(class ~ ., data = body, method = "knn",
                    trControl = fit.control.cv, preProcess = c("center", "scale"))
knn.fit.boot <- train(class ~ ., data = body, method = "knn",
                      trControl = fit.control.boot, preProcess = c("center", "scale"))
```

```r
knn.fit.cv$finalModel
```

```
## 9-nearest neighbor model
## Training set outcome distribution:
##
##    A    B    C    D
## 3348 3347 3349 3349
```

```r
knn.fit.boot$finalModel
```

```
## 9-nearest neighbor model
## Training set outcome distribution:
##
##    A    B    C    D
## 3348 3347 3349 3349
```

```r
confusionMatrix(knn.fit.cv)
```

```
## Cross-Validated (10 fold, repeated 3 times) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 19.9  8.2  2.8  0.6
##          B  4.5 11.7  7.4  2.0
##          C  0.6  4.4 13.6  5.7
##          D  0.1  0.6  1.2 16.6
##
##  Accuracy (average) : 0.6179
```

```r
confusionMatrix(knn.fit.boot)
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D
##          A 18.5  8.2  2.9  0.7
##          B  5.1 11.0  7.6  2.2
##          C  1.1  5.0 12.5  5.9
##          D  0.2  0.9  1.9 16.4
##
##  Accuracy (average) : 0.584
```

```r
knn.fit.cv$results
```

```
##   k  Accuracy     Kappa AccuracySD    KappaSD
## 1 5 0.6045941 0.4727952 0.01216431 0.01622369
## 2 7 0.6142261 0.4856382 0.01132453 0.01510491
## 3 9 0.6178839 0.4905149 0.01147146 0.01529822
```

```r
knn.fit.boot$results
```

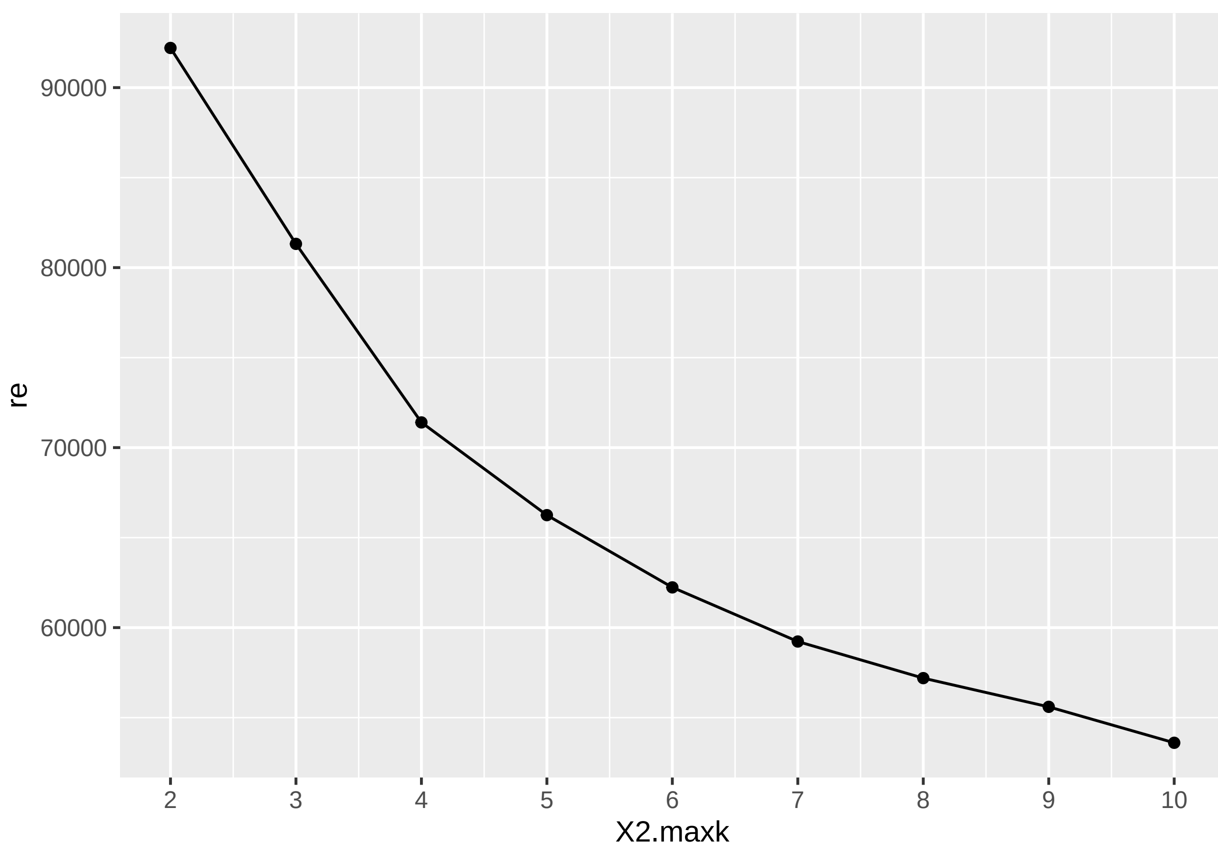
```
##   k  Accuracy     Kappa  AccuracySD     KappaSD
## 1 5 0.5599129 0.4132660 0.006783944 0.009123717
## 2 7 0.5754512 0.4339888 0.006548934 0.008788444
## 3 9 0.5840070 0.4454149 0.006918906 0.009270053
```

KNN yields an accuracy of 0.5849.

k-Means

```r
#k means

bp_ <- body[,c(-2,-12)]
bp_ <- na.omit(bp_)
bp_ <-scale(bp_)
```

```r
#create the function that runs the k-mean algorithm and get the clusters sum of squares

kmean_withinss <- function(k) {
    cluster <- kmeans(bp_, k)
    return (cluster$tot.withinss)
}

maxk <-10
re <- sapply(2:maxk, kmean_withinss)

re. <-data.frame(2:maxk, re) #create a data frame to store all values

ggplot(re., aes(x = X2.maxk, y = re)) +
    geom_point() +
    geom_line() +
    scale_x_continuous(breaks = seq(1, 10, by = 1))
```



From the graph, we can see the optimal k is 6, where the curve starts to diminish the return.

```r
set.seed(240) # Setting seed
kmeans.re <- kmeans(bp_, centers=6, nstart = 20)

# Confusion Matrix
table(body$class, kmeans.re$cluster)
```

```
##
##         1    2    3    4    5    6
##   A   404    2 1169  751  225  797
##   B   580   36  794  816  361  760
##   C   552  127  661  844  480  685
##   D   394 1144  370  361  828  252
```

```r
body_kmeans <- cbind(body, cluster = kmeans.re$cluster)
head(body_kmeans)
```

```
##   age gender height_cm weight_kg body fat_% diastolic systolic gripForce
## 1  27      1     172.3     75.24       21.3        80      130      54.9
## 2  25      1     165.0     55.80       15.7        77      126      36.4
## 3  31      1     179.6     78.00       20.1        92      152      44.8
## 4  32      1     174.5     71.10       18.4        76      147      41.4
## 5  28      1     173.8     67.70       17.1        70      127      43.5
## 6  36      0     165.4     55.40       22.0        64      119      23.8
##   sit and bend forward_cm sit-ups counts broad jump_cm class cluster
## 1                    18.4             60           217     C       6
## 2                    16.3             53           229     A       4
## 3                    12.0             49           181     C       6
## 4                    15.2             53           219     B       6
## 5                    27.1             45           217     B       4
## 6                    21.0             27           153     B       3
```

```r
kmeans.re$centers #matrix of cluster centers
```

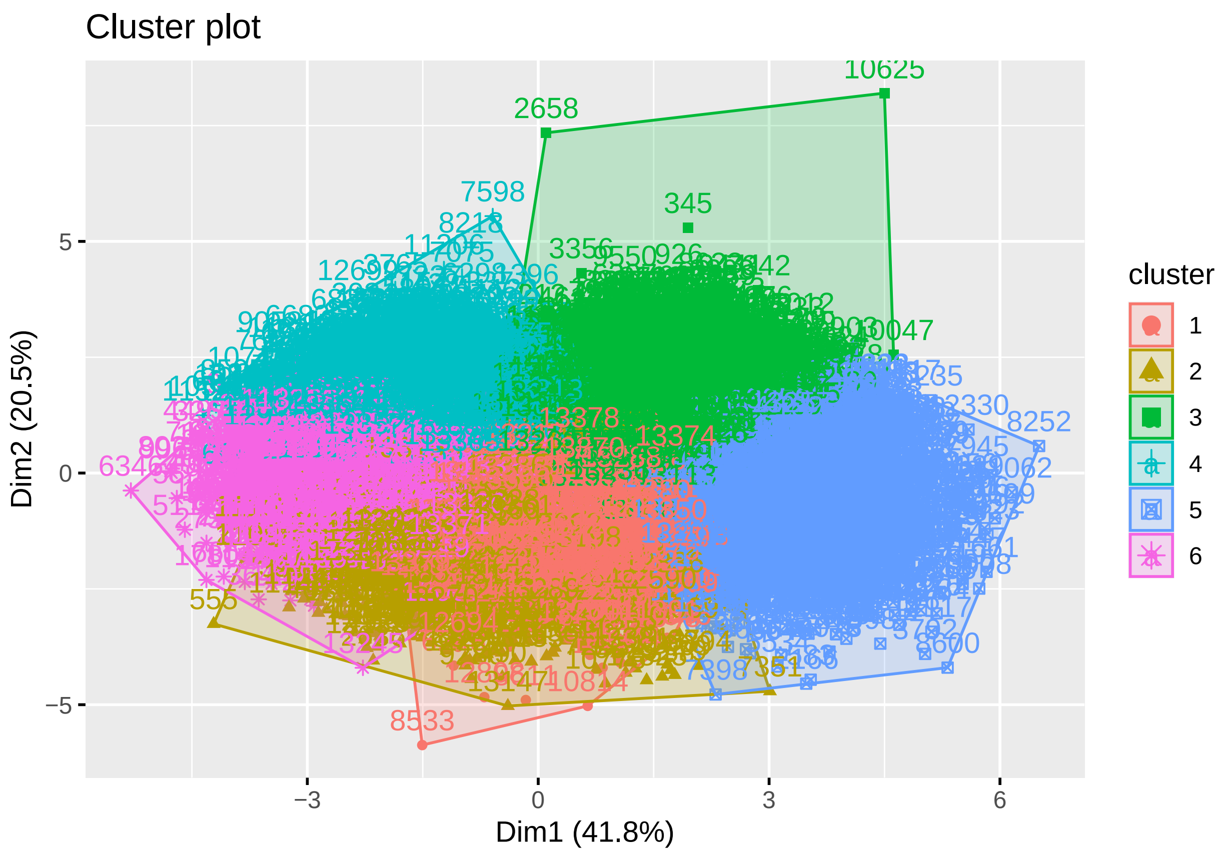
```
##          age   height_cm  weight_kg body fat_%   diastolic    systolic
## 1  1.3093285  0.05239229  0.1066542 -0.1490933  0.54992415  0.64165957
## 2 -0.2044529  0.75095325  1.2803784  0.5784224  0.48411057  0.37676075
## 3 -0.5843503 -0.75152887 -0.9759586  0.3929658 -0.52772620 -0.76164170
## 4 -0.5628393  0.61639755  0.2828347 -0.8674661 -0.67966223 -0.50721974
## 5  1.1668925 -1.33496525 -0.7806493  1.2249225 -0.05727179 -0.01854236
## 6 -0.4650096  0.79620397  0.6955469 -0.6260350  0.75278844  0.79787684
##    gripForce sit and bend forward_cm sit-ups counts broad jump_cm
## 1  0.2046815             -0.22392263    -0.37074296   -0.11269288
## 2  0.4189320             -1.44220187    -0.28534222    0.07211187
## 3 -0.9398905              0.60664814    -0.09882091   -0.56221858
## 4  0.6526715             -0.05376574     0.77779309    0.84769071
## 5 -1.2337413              0.07878567    -1.50290672   -1.51799134
## 6  0.9615539              0.20189534     0.83214898    0.93490866
```

```r
kmeans.re$size #number of points in each clusters
```

```
## [1] 1930 1309 2994 2772 1894 2494
```

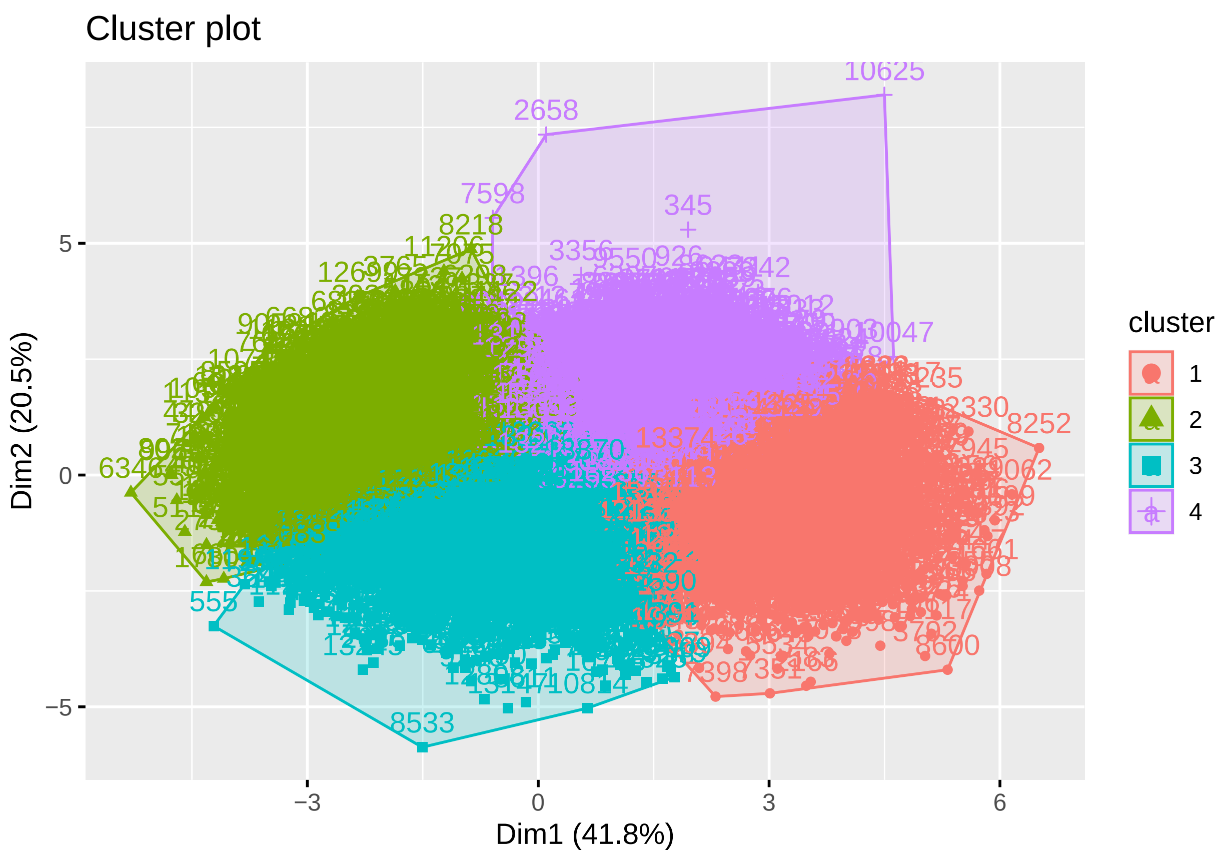
The cluster centers shows that cluster 5 have the highest body fat % and age average among all clusters, and cluster 3 has the highest average weight among all.

```
fviz_cluster(kmeans.re, data = bp_)
```



The plot displays that each group have similar dimensions but there are still some overlaps.

```
fviz_cluster(kmeans(bp_, centers=4, nstart = 20), data = bp_)
```



Clusters of 4 might work better in this case. Now we can see that all the clusters are more separated form each other.

Based on my fits, a non-linear is more appropriate since the accuracy generated from QDA has the highest accuracy, though the accuracy are pretty close to each other in all classification methods.

Regularization

For this portion of the project, we're going to use a data set containing information on the Australian housing market and apply regularization techniques to make predictions on house prices. The data set will have 81 columns of data.

Overall procedure Numeric pipeline: - We will impute missing values with medians - After that, we will standardize each vector of data to ensure each feature is given equal weighting by our models

Categorical pipeline: - In each vector of data, we will fill missing values with the most commonly observed sample - After that, we will transform each vector to numeric representation using label encoding or by making them dummy variables, where appropriate

Afterwards, we will fit the Principal Component, LASSO, Ridge, and Elastic Net models to our data and see which produces the best fit.

For the LASSO, Ridge, and Elastic Net regressions, we have first remove multicollinearity before applying them. To address this issue, we will examine each variable's variance inflation factor. As a rule of thumb, a VIF > 5 implies that the variable is a potential cause of collinearity, so we will remove it from our dataset.

```r
rm(list = ls())
# importing data
df = read.csv('house_data.csv')
cat("Number of variables:",length(df))
```

```
## Number of variables: 81
```

```r
head(df) # check
```

```
##   Id MSSubClass MSZoning LotFrontage LotArea Street Alley LotShape LandContour
## 1  1         60       RL          65    8450   Pave  <NA>      Reg         Lvl
## 2  2         20       RL          80    9600   Pave  <NA>      Reg         Lvl
## 3  3         60       RL          68   11250   Pave  <NA>      IR1         Lvl
## 4  4         70       RL          60    9550   Pave  <NA>      IR1         Lvl
## 5  5         60       RL          84   14260   Pave  <NA>      IR1         Lvl
## 6  6         50       RL          85   14115   Pave  <NA>      IR1         Lvl
##   Utilities LotConfig LandSlope Neighborhood Condition1 Condition2 BldgType
## 1    AllPub    Inside       Gtl      CollgCr       Norm       Norm     1Fam
## 2    AllPub       FR2       Gtl      Veenker      Feedr       Norm     1Fam
## 3    AllPub    Inside       Gtl      CollgCr       Norm       Norm     1Fam
## 4    AllPub    Corner       Gtl      Crawfor       Norm       Norm     1Fam
## 5    AllPub       FR2       Gtl      NoRidge       Norm       Norm     1Fam
## 6    AllPub    Inside       Gtl      Mitchel       Norm       Norm     1Fam
##   HouseStyle OverallQual OverallCond YearBuilt YearRemodAdd RoofStyle RoofMatl
## 1     2Story           7           5      2003         2003     Gable  CompShg
## 2     1Story           6           8      1976         1976     Gable  CompShg
## 3     2Story           7           5      2001         2002     Gable  CompShg
## 4     2Story           7           5      1915         1970     Gable  CompShg
## 5     2Story           8           5      2000         2000     Gable  CompShg
## 6     1.5Fin           5           5      1993         1995     Gable  CompShg
##   Exterior1st Exterior2nd MasVnrType MasVnrArea ExterQual ExterCond Foundation
## 1     VinylSd     VinylSd    BrkFace        196        Gd        TA      PConc
## 2     MetalSd     MetalSd       None          0        TA        TA     CBlock
## 3     VinylSd     VinylSd    BrkFace        162        Gd        TA      PConc
## 4     Wd Sdng     Wd Shng       None          0        TA        TA     BrkTil
## 5     VinylSd     VinylSd    BrkFace        350        Gd        TA      PConc
## 6     VinylSd     VinylSd       None          0        TA        TA       Wood
##   BsmtQual BsmtCond BsmtExposure BsmtFinType1 BsmtFinSF1 BsmtFinType2
## 1       Gd       TA           No          GLQ        706          Unf
## 2       Gd       TA           Gd          ALQ        978          Unf
## 3       Gd       TA           Mn          GLQ        486          Unf
## 4       TA       Gd           No          ALQ        216          Unf
## 5       Gd       TA           Av          GLQ        655          Unf
## 6       Gd       TA           No          GLQ        732          Unf
##   BsmtFinSF2 BsmtUnfSF TotalBsmtSF Heating HeatingQC CentralAir Electrical
## 1          0       150         856    GasA        Ex          Y      SBrkr
## 2          0       284        1262    GasA        Ex          Y      SBrkr
## 3          0       434         920    GasA        Ex          Y      SBrkr
## 4          0       540         756    GasA        Gd          Y      SBrkr
## 5          0       490        1145    GasA        Ex          Y      SBrkr
## 6          0        64         796    GasA        Ex          Y      SBrkr
##   X1stFlrSF X2ndFlrSF LowQualFinSF GrLivArea BsmtFullBath BsmtHalfBath FullBath
## 1       856       854            0      1710            1            0        2
```

```
## 2      1262       0            0      1262            0            1        2
## 3       920     866            0      1786            1            0        2
## 4       961     756            0      1717            1            0        1
## 5      1145    1053            0      2198            1            0        2
## 6       796     566            0      1362            1            0        1
##   HalfBath BedroomAbvGr KitchenAbvGr KitchenQual TotRmsAbvGrd Functional
## 1        1            3            1          Gd            8        Typ
## 2        0            3            1          TA            6        Typ
## 3        1            3            1          Gd            6        Typ
## 4        0            3            1          Gd            7        Typ
## 5        1            4            1          Gd            9        Typ
## 6        1            1            1          TA            5        Typ
##   Fireplaces FireplaceQu GarageType GarageYrBlt GarageFinish GarageCars
## 1          0        <NA>     Attchd        2003          RFn          2
## 2          1          TA     Attchd        1976          RFn          2
## 3          1          TA     Attchd        2001          RFn          2
## 4          1          Gd     Detchd        1998          Unf          3
## 5          1          TA     Attchd        2000          RFn          3
## 6          0        <NA>     Attchd        1993          Unf          2
##   GarageArea GarageQual GarageCond PavedDrive WoodDeckSF OpenPorchSF
## 1        548         TA         TA          Y          0          61
## 2        460         TA         TA          Y        298           0
## 3        608         TA         TA          Y          0          42
## 4        642         TA         TA          Y          0          35
## 5        836         TA         TA          Y        192          84
## 6        480         TA         TA          Y         40          30
##   EnclosedPorch X3SsnPorch ScreenPorch PoolArea PoolQC Fence MiscFeature
## 1             0          0           0        0   <NA>  <NA>        <NA>
## 2             0          0           0        0   <NA>  <NA>        <NA>
## 3             0          0           0        0   <NA>  <NA>        <NA>
## 4           272          0           0        0   <NA>  <NA>        <NA>
## 5             0          0           0        0   <NA>  <NA>        <NA>
## 6             0        320           0        0   <NA> MnPrv        Shed
##   MiscVal MoSold YrSold SaleType SaleCondition SalePrice
## 1       0      2   2008       WD        Normal    208500
## 2       0      5   2007       WD        Normal    181500
## 3       0      9   2008       WD        Normal    223500
## 4       0      2   2006       WD       Abnorml    140000
## 5       0     12   2008       WD        Normal    250000
## 6     700     10   2009       WD        Normal    143000
```

```r
# convert ID column into row names
# and drop it
rownames(df) <- df$Id
drops <- c('Id')
df <- df[ , !(names(df) %in% drops)]

# check length of our data
cat("Length of data:",length(df$MSSubClass))
```

```
## Length of data: 1460
```

```r
# check if our target variable has any missing values
cat("Missing values in our target var:",sum(is.na(df$SalePrice)))
```

```
## Missing values in our target var: 0
```

```r
# great, now we can train-test-split
set.seed(42)
train_index = createDataPartition(df$SalePrice, p = .7, list = FALSE)
train <- df[train_index,]
test  <- df[-train_index,]

# drop columns with high number of NA values
# define "high" as >20% null values
drops <- c()

# find the columns with a high number of NAs
for (col in names(train)){
  if (sum(is.na(df[col])) > 0.2*nrow(df[col])){
    drops <- c(drops,col)
  }
}
# drop those columns from our data
train <- train[ , !(names(train) %in% drops)]
test <- test[ , !(names(test) %in% drops)]

# find if there are any variables that don't
# give any information i.e. 0 variance
# this would be the case if a column only
# has 1 unique value
cat("In the training set these variables have 0 variance:",names(sapply(lapply(train, unique), length)[
```

```
## In the training set these variables have 0 variance:
```

```r
cat("In the test set these variables have 0 variance:",names(sapply(lapply(test, unique), length)[sapply
    '\n')
```

```
## In the test set these variables have 0 variance: Utilities
```

```r
# we will drop the utilities variable
drops <- c('Utilities')
train<- train[ , !(names(train) %in% drops)]
test <- test[ , !(names(test) %in% drops)]

# split into train-test sets
drops <- c('SalePrice')
X_train <-train[ , !(names(train) %in% drops)]
y_train <- train$SalePrice
X_test <- test[ , !(names(test) %in% drops)]
y_test <- test$SalePrice
```

Pipeline

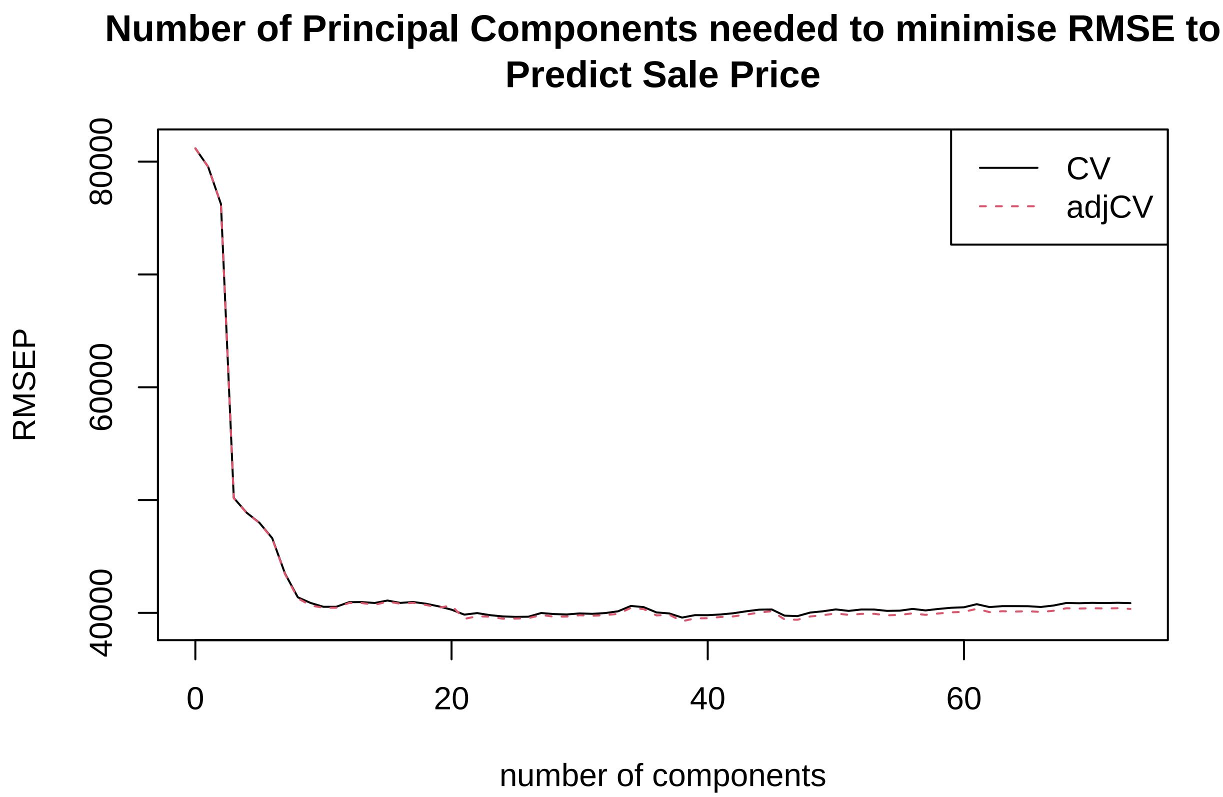
```r
# handling numeric data
# (1) impute > median
# (2) scale
X_train_scaled <- X_train %>% mutate_if(is.numeric,function(x) ifelse(is.na(x),median(x,na.rm=T),x)) %>%

X_test_scaled <- X_test %>% mutate_if(is.numeric,function(x) ifelse(is.na(x),median(x,na.rm=T),x)) %>% m

# handling categorical data
# (1) impute with mode
X_train_scaled <- X_train_scaled %>% mutate_if(is.character,function(x) ifelse(is.na(x),mode(x),x))
X_test_scaled <- X_test_scaled %>% mutate_if(is.character,function(x) ifelse(is.na(x),mode(x),x))

# (2) encode data
X_train_scaled <- X_train_scaled %>% mutate_if(is.character,function(x) as.integer(factor(x)))
X_test_scaled <- X_test_scaled %>% mutate_if(is.character,function(x) as.integer(factor(x)))
```

PCR

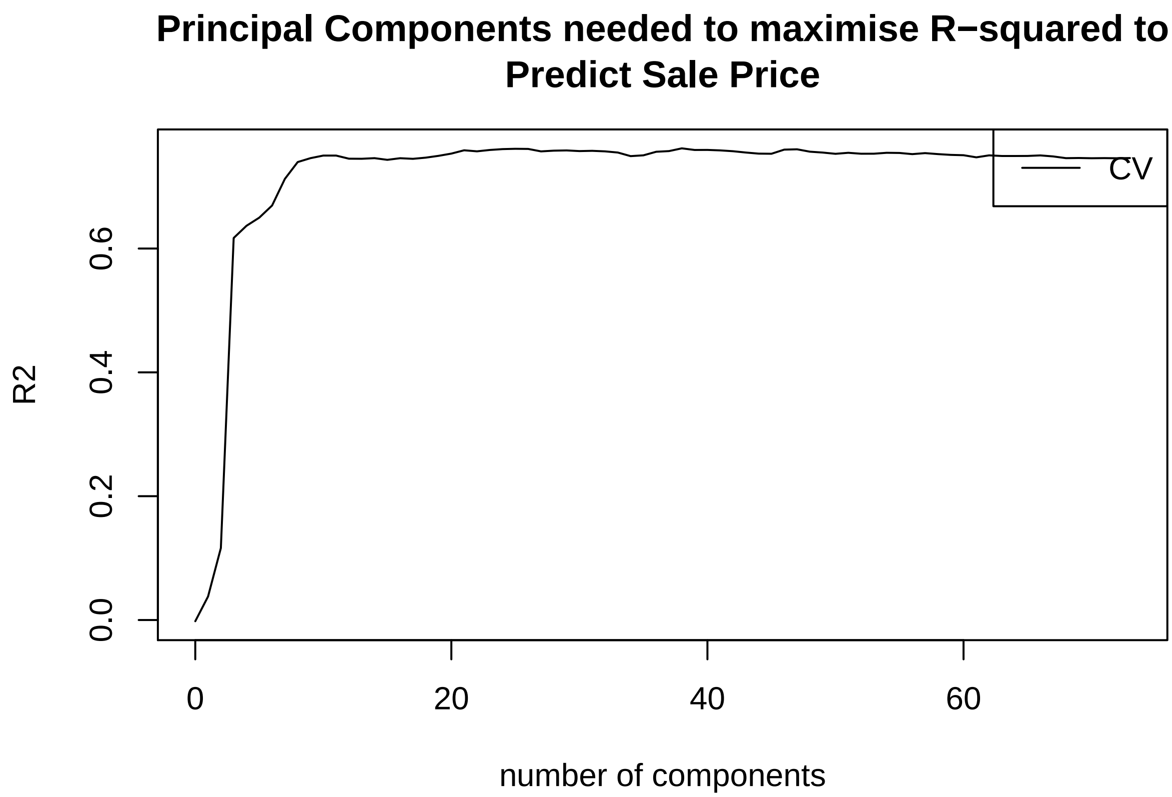
```r
library(pls)
```

```
##
## Attaching package: 'pls'

## The following object is masked from 'package:caret':
##
##     R2

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
set.seed(42)

# combine y_train and X_train_scaled
train_scaled <- X_train_scaled
train_scaled['SalePrice'] <- y_train

# combine y_test and X_test_scaled
test_scaled <- X_test_scaled
test_scaled['SalePrice'] <- y_test

# fit principal component analysis regression
pcr.fit <- pcr(SalePrice ~ .,data = train_scaled,validation = "CV")

# plot RMSE vs number of components
validationplot(pcr.fit, val.type = "RMSEP",
               legendpos='topright',
               main = 'Number of Principal Components needed to minimise RMSE to
Predict Sale Price')
```




```r
# plot of Rsquared vs number of components
validationplot(pcr.fit, val.type = "R2",
               legendpos='topright',
               main = 'Principal Components needed to maximise R-squared to
Predict Sale Price')
```



From the validation plot, we observe that the number of components with the lowest cross-validation error is in the range of 21 to 40 components. Therefore, we will test ncomps in [21,40] to see which principal component regression model performs best on our test set.

```
ncomps = seq(1:20) + 20
ncomp_score <- c()
for (n in ncomps){
  pcr.pred <- predict(pcr.fit, X_test_scaled, ncomp = n)
  ncomp_score <- c(ncomp_score,sqrt(mean((pcr.pred-y_test)^2)))
}

# table of ncomps and respective test scores
data.frame(ncomps,ncomp_score)
```

```
##    ncomps ncomp_score
## 1      21    28476.00
## 2      22    28580.16
## 3      23    28582.13
## 4      24    28344.71
## 5      25    28349.25
## 6      26    28379.93
## 7      27    28271.16
## 8      28    28265.73
## 9      29    28308.92
## 10     30    28243.94
## 11     31    28141.38
```

```
## 12      32    28215.65
## 13      33    28715.08
## 14      34    28714.30
## 15      35    28804.00
## 16      36    28346.10
## 17      37    28302.48
## 18      38    28882.26
## 19      39    29028.45
## 20      40    29050.29
```

This table shows that ncomps = 31 performs the best with an RMSE of $28,141.38$. This performance will later be compared against the rest of the regressions.

Ridge

Removing linearly dependent variables

Before moving onto any of the other regressions, we're going to eliminate the linearly dependent variables from our data. We didn't confront this problem earlier because principal component analysis naturally eliminates multicollinearity.

```r
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:gtools':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
## The following object is masked from 'package:purrr':
##
##     some
```

```r
# find linearly dependent variables
fit <- lm(SalePrice~.,data = train_scaled)
ld_vars <- attributes(alias(fit)$Complete)$dimnames[[1]]
cat('Linearly dependent variables:',ld_vars)
```

```
## Linearly dependent variables: TotalBsmtSF GrLivArea
```

```r
# eliminate linearly dep variables
train_scaled_reg <- train_scaled[,-which(names(train_scaled) %in% ld_vars)]
test_scaled_reg <- test_scaled[,-which(names(test_scaled) %in% ld_vars)]

# find variables with VIF > 5
fit <- lm(SalePrice~. ,data = train_scaled_reg)
cat("Variables with VIF > 5:",names(vif(fit)[vif(fit) > 5]))
```

```
## Variables with VIF > 5: MSSubClass YearBuilt BsmtFinSF1 BsmtUnfSF X1stFlrSF X2ndFlrSF TotRmsAbvGrd Ga
```

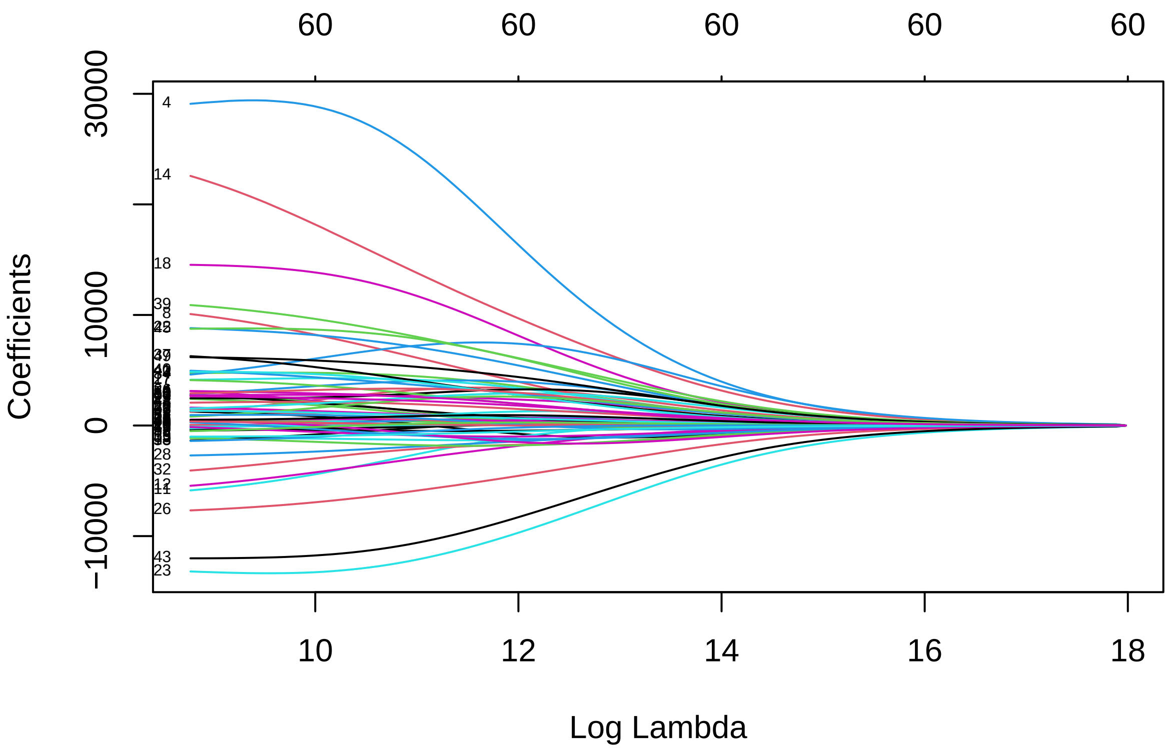
```r
# eliminate variables with VIF > 5
train_scaled_reg <- train_scaled_reg[,-which(names(train_scaled_reg) %in% names(vif(fit)[vif(fit) > 5]))

test_scaled_reg <-test_scaled_reg[,-which(names(test_scaled_reg) %in% names(vif(fit)[vif(fit) > 5]))]
```

The ridge model is a useful model to consider in modeling Sale Price because our feature space has 63 variables, and the the L2 regularization penalty, $\lambda \sum_{i=1}^{63} \beta_i^2$, can help reduce the coefficients of less important variables in our data set to near-zero values.

```r
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-1

##
## Attaching package: 'glmnet'

## The following object is masked from 'package:gtools':
##
##     na.replace
```

```r
library(Matrix)
model_ridge = cv.glmnet(x = as.matrix(train_scaled_reg[,-which(names(train_scaled_reg) %in% c("SalePric
plot(model_ridge$glmnet.fit, "lambda", label=TRUE)
```



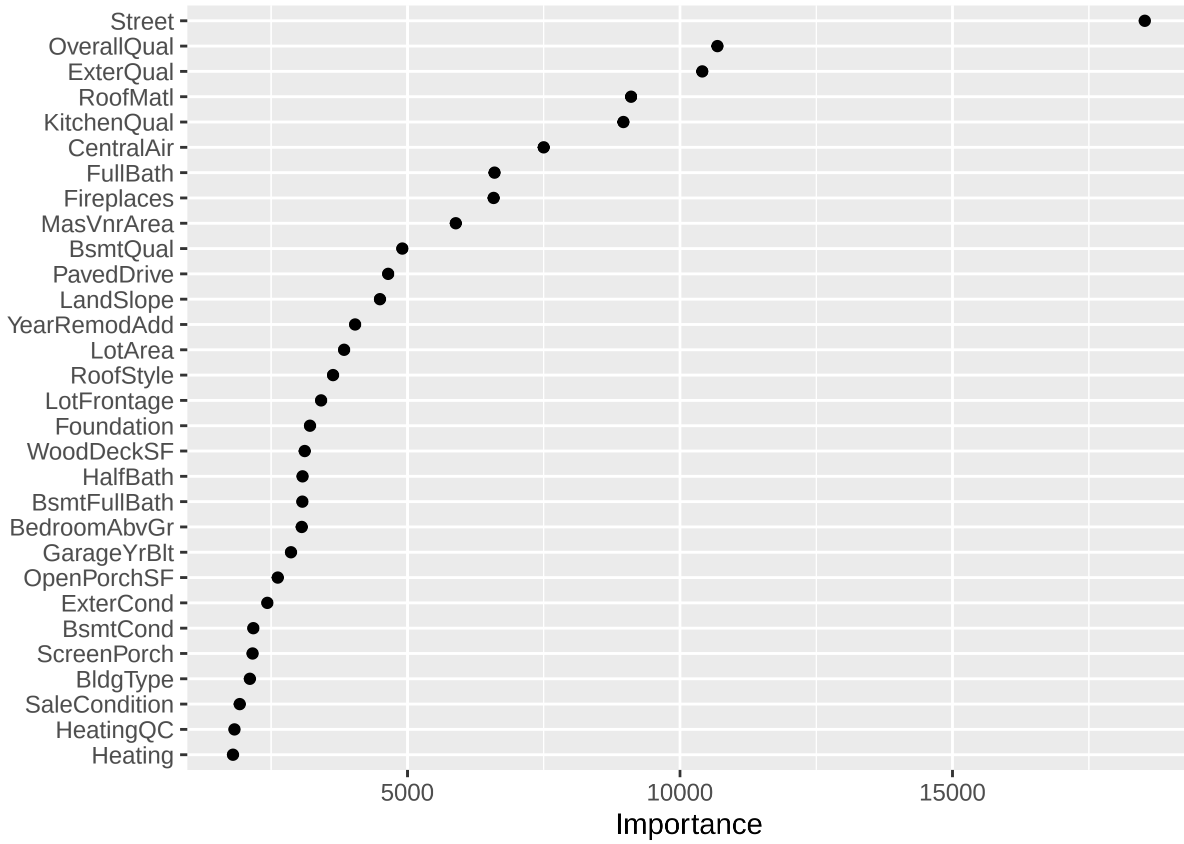
This plot shows how increasing the size of lambda affects the coefficients of different variables. We observe that our 4th, 14th, 45th, 23rd, 36th, 37th, and 26th variables are highly important, while the other variables are less so.

Now let's take a closer look at our important variables by creating a variable importance plot.

```r
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```r
vip(model_ridge, num_features = 30, geom = "point")
```



This shows us that our street variable (which is the type of street the property is on) is the most important variable by far, then exterior quality, overall quality, and kitchen quality are the next most important variables. On a qualitative level, this makes sense because most people evaluate real-estate from the outside first, then the inside. Kitchen quality is unsurprising because many people spend a lot of time in their kitchens, either eating food or preparing food. Finally, on the less important side of the 30 variables, we see heating, basement condition, and so forth, which is also expected – at least in Australia, where it's generally quite warm, and because the basement aesthetics are less noticeable.

```r
# we can utilize cross validation to train our ridge model
# and find the best lambda
train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 1,
                              search = "random",
                              verboseIter = FALSE)

ridge_model     <- train(SalePrice ~ .,
                         data = train_scaled_reg,
                         metrics = 'RMSE',
                         method = "ridge",
                         tuneLength = 25,
                         trControl = train_control)

# Predict using the testing data
ridge_preds = predict(ridge_model, newdata = X_test_scaled)
```

```r
# Evaluate performance
postResample(pred = ridge_preds, obs = y_test)
```

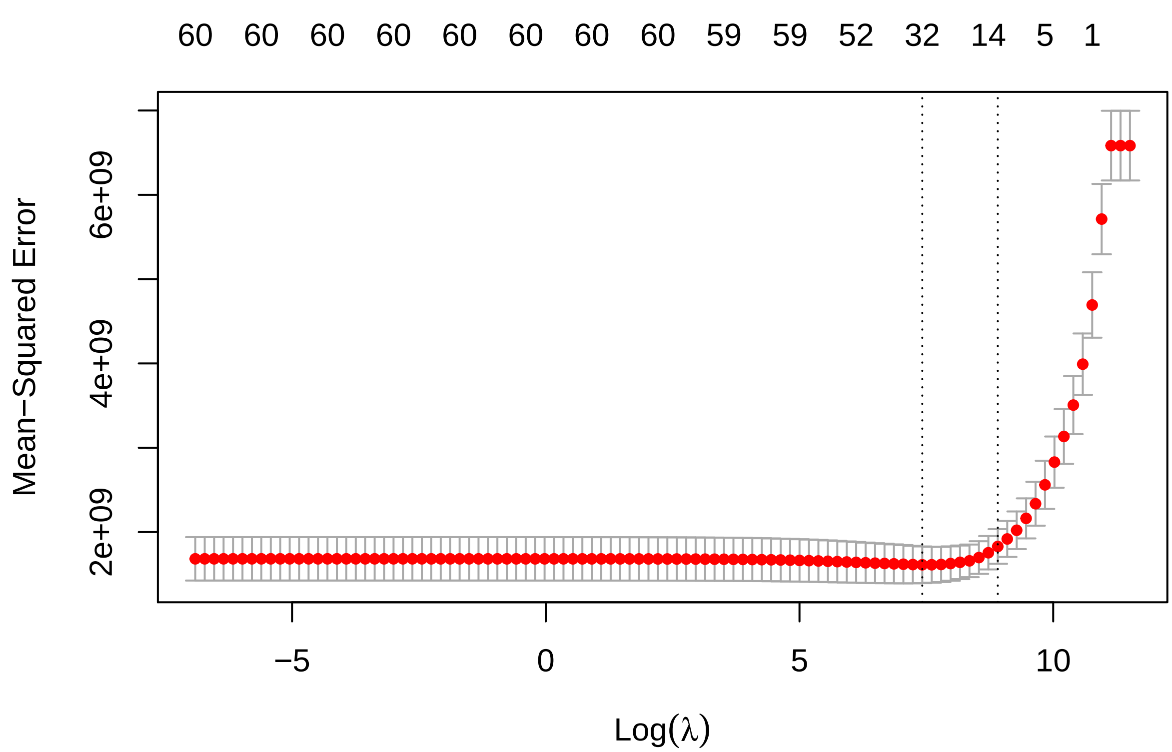
```
##         RMSE     Rsquared          MAE
## 3.314731e+04 8.070257e-01 2.371836e+04
```

This shows us that our ridge model's best cross-validated performance is 33,211.91, which is worse than our PCR model (by roughly 4,000). The Rsquared is 0.8071006.

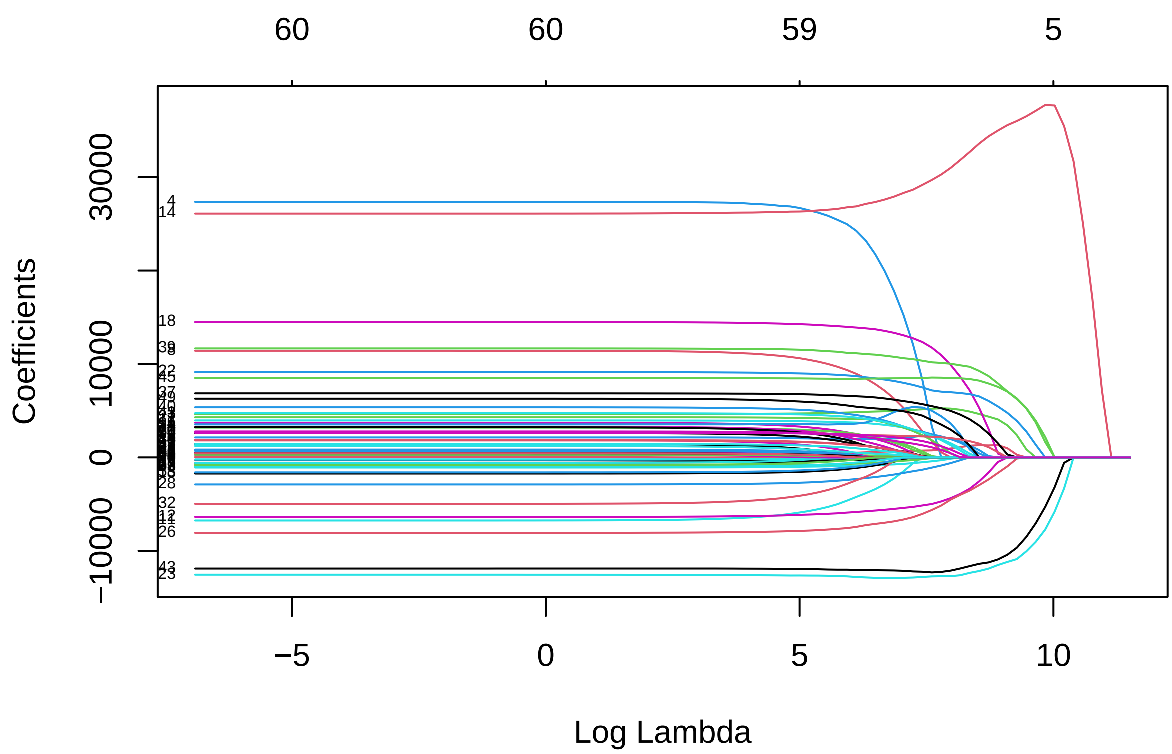
LASSO

We will now try a LASSO model to regularize our data. The LASSO model works similarly to Ridge, except by using a different penalty term that takes the absolute value of β, which results inthe coefficients of less important variables can be reduced to exactly 0 (not just near-zero as in our Ridge Model). Thus, by using the penalty term $\lambda \sum_{i=1}^{p} |\beta_i|$, our LASSO model will have high predictive power and be simple to interpret.

```r
# Perform 10-fold cross-validation to select lambda
lambdas_to_try <- 10^seq(-3, 5, length.out = 100)

# Setting alpha = 1 implements lasso regression
lasso_cv <- cv.glmnet(x = as.matrix(train_scaled_reg[,-which(names(train_scaled_reg) %in% c("SalePrice")
                      standardize = TRUE, nfolds = 10)
# Plot cross-validation results
plot(lasso_cv)
```



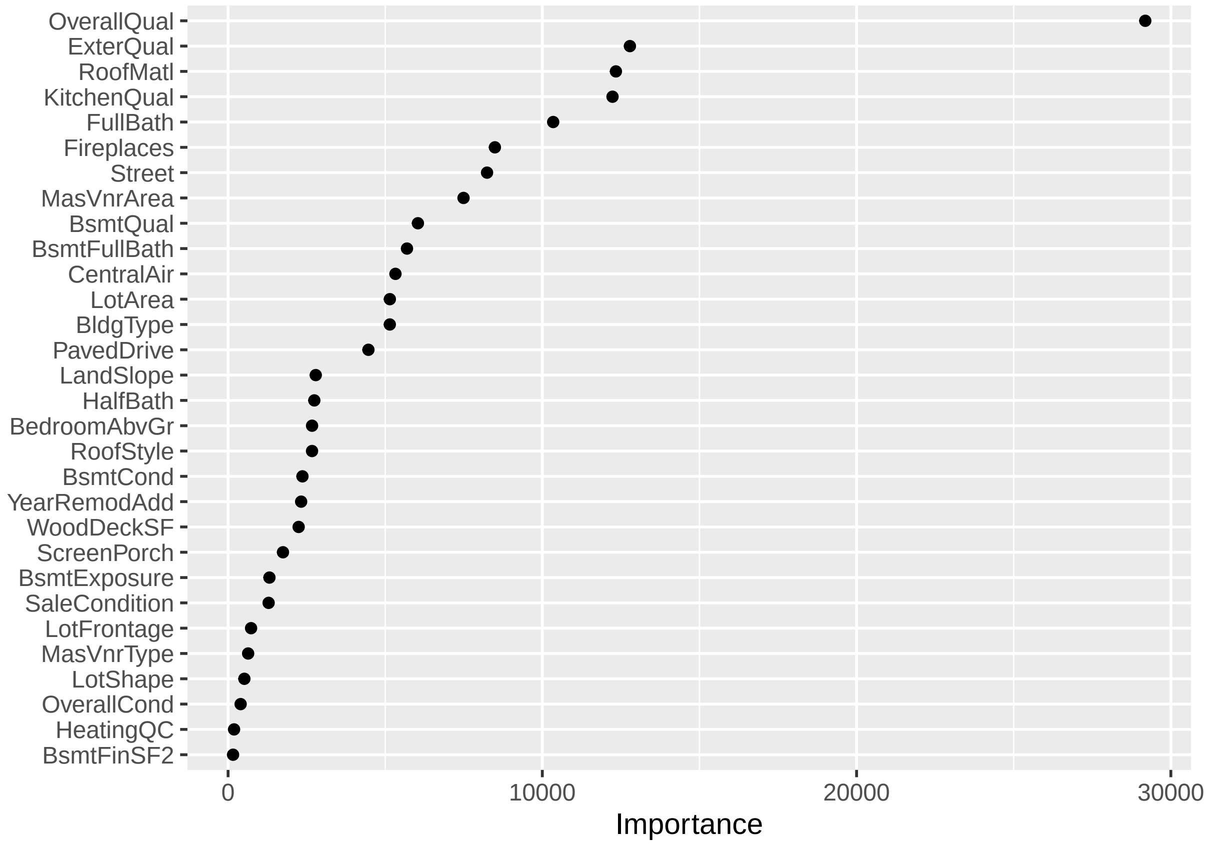
```r
# Best cross-validated lambda
lambda_cv <- lasso_cv$lambda.min

# Fit final model
model_lasso <- glmnet(x = as.matrix(train_scaled_reg[,-which(names(train_scaled_reg) %in% c("SalePrice"

#Compare variables across lambdas
plot(lasso_cv$glmnet.fit, "lambda", label=TRUE)
```



As with the Ridge Model, this plot shows how the coefficients of different variables react to increasing lambda. From this plot we find the 14th, 23rd, 43rd, 30th, 45th, and 22nd variables are highly important, while other variables are less so.

We will now create a variable importance plot to closer examine these important variables.

```r
library(vip)
vip(model_lasso, num_features = 30, geom = "point")
```



This shows us that Overall Quality is the most important variable and Street is the second most important and both of these variables are far more important than the rest. While our Ridge Model had Street as the most important variable and Overall Quality at second, both our Ridge Model and Lasso Model agree on the top two most important variables. Our Lasso Model then shows that Roof Material, External Quality, Kitchen Quality, and Full Bathrooms are the next most important variables. Our Ridge Model resulted in the same aside from Central Air replacing Full Bathrooms. On the other end, both our Lasso and our Ridge Models agree that heating is not that important of a variable.

We will now train our Lasso Model and predict our test set to measure RMSE:

```
# we can utilize cross validation to train our lasso model
# and find the best lambda
train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 5,
                              search = "random",
                              verboseIter = FALSE)

lasso_model     <- train(SalePrice ~ .,
                         data = train_scaled_reg,
                         metrics = 'RMSE',
                         method = "glmnet",
                         tuneGrid = expand.grid(alpha = 1,
                                                lambda = 1),
                         tuneLength = 25,
                         trControl = train_control)
```

```r
# Predict using the testing data
lasso_preds = predict(lasso_model, newdata = X_test_scaled)

# Evaluate performance
postResample(pred = lasso_preds, obs = y_test)
```

```
##         RMSE     Rsquared          MAE
## 33128.049508     0.806975 23602.019008
```

This shows us that our LASSO model's best cross-validated performance is $33,128.05$, which is worse than our PCR model (by roughly $4,000$). The Rsquared is 0.806975.

Elastic Net

We will now try an Elastic Net model to regularize our data. Elastic Net improves upon Ridge and Lasso by combining the penalties from each model resulting in a shrinkage model less dependent on the data. The new penalty term is $\frac{1-\alpha}{2}\sum_{i=1}^{p}\beta_i^2 + \alpha\sum_{i=1}^{p}|\beta_i|$ where $\alpha = 0$ (Ridge) % $\alpha = 1$ (Lasso).

```r
#use cross validation to train our elastic net model
train_control <- trainControl(method = "repeatedcv",
                              number = 5,
                              repeats = 5,
                              search = "random",
                              verboseIter = TRUE)

# Train the model
elastic_net_model <- train(SalePrice ~ .,
                           data = train_scaled_reg,
                           metrics = 'RMSE',
                           method = "glmnet",
                           preProcess = c("center", "scale"),
                           tuneLength = 25,
                           trControl = train_control)
```

```
## + Fold1.Rep1: alpha=0.63608, lambda=0.0017708
## - Fold1.Rep1: alpha=0.63608, lambda=0.0017708
## + Fold1.Rep1: alpha=0.97864, lambda=0.1779302
## - Fold1.Rep1: alpha=0.97864, lambda=0.1779302
## + Fold1.Rep1: alpha=0.42631, lambda=0.0633234
## - Fold1.Rep1: alpha=0.42631, lambda=0.0633234
## + Fold1.Rep1: alpha=0.37574, lambda=0.0015142
## - Fold1.Rep1: alpha=0.37574, lambda=0.0015142
## + Fold1.Rep1: alpha=0.80531, lambda=0.0023148
## - Fold1.Rep1: alpha=0.80531, lambda=0.0023148
## + Fold1.Rep1: alpha=0.06683, lambda=0.1639706
## - Fold1.Rep1: alpha=0.06683, lambda=0.1639706
## + Fold1.Rep1: alpha=0.01230, lambda=0.3197005
## - Fold1.Rep1: alpha=0.01230, lambda=0.3197005
## + Fold1.Rep1: alpha=0.75340, lambda=0.1553943
## - Fold1.Rep1: alpha=0.75340, lambda=0.1553943
## + Fold1.Rep1: alpha=0.65786, lambda=0.0962132
## - Fold1.Rep1: alpha=0.65786, lambda=0.0962132
```

```
## + Fold1.Rep1: alpha=0.42268, lambda=4.0716342
## - Fold1.Rep1: alpha=0.42268, lambda=4.0716342
## + Fold1.Rep1: alpha=0.87538, lambda=0.0525596
## - Fold1.Rep1: alpha=0.87538, lambda=0.0525596
## + Fold1.Rep1: alpha=0.49128, lambda=6.5121646
## - Fold1.Rep1: alpha=0.49128, lambda=6.5121646
## + Fold1.Rep1: alpha=0.59242, lambda=0.0997138
## - Fold1.Rep1: alpha=0.59242, lambda=0.0997138
## + Fold1.Rep1: alpha=0.29222, lambda=0.0009879
## - Fold1.Rep1: alpha=0.29222, lambda=0.0009879
## + Fold1.Rep1: alpha=0.29635, lambda=1.0847852
## - Fold1.Rep1: alpha=0.29635, lambda=1.0847852
## + Fold1.Rep1: alpha=0.89740, lambda=0.0104120
## - Fold1.Rep1: alpha=0.89740, lambda=0.0104120
## + Fold1.Rep1: alpha=0.38484, lambda=0.0798488
## - Fold1.Rep1: alpha=0.38484, lambda=0.0798488
## + Fold1.Rep1: alpha=0.50231, lambda=0.0405177
## - Fold1.Rep1: alpha=0.50231, lambda=0.0405177
## + Fold1.Rep1: alpha=0.87686, lambda=2.7659135
## - Fold1.Rep1: alpha=0.87686, lambda=2.7659135
## + Fold1.Rep1: alpha=0.78346, lambda=0.3752296
## - Fold1.Rep1: alpha=0.78346, lambda=0.3752296
## + Fold1.Rep1: alpha=0.81532, lambda=0.0358768
## - Fold1.Rep1: alpha=0.81532, lambda=0.0358768
## + Fold1.Rep1: alpha=0.93018, lambda=0.0013239
## - Fold1.Rep1: alpha=0.93018, lambda=0.0013239
## + Fold1.Rep1: alpha=0.91108, lambda=4.6314855
## - Fold1.Rep1: alpha=0.91108, lambda=4.6314855
## + Fold1.Rep1: alpha=0.74424, lambda=0.0070056
## - Fold1.Rep1: alpha=0.74424, lambda=0.0070056
## + Fold1.Rep1: alpha=0.31997, lambda=0.0066009
## - Fold1.Rep1: alpha=0.31997, lambda=0.0066009
## + Fold2.Rep1: alpha=0.63608, lambda=0.0017708
## - Fold2.Rep1: alpha=0.63608, lambda=0.0017708
## + Fold2.Rep1: alpha=0.97864, lambda=0.1779302
## - Fold2.Rep1: alpha=0.97864, lambda=0.1779302
## + Fold2.Rep1: alpha=0.42631, lambda=0.0633234
## - Fold2.Rep1: alpha=0.42631, lambda=0.0633234
## + Fold2.Rep1: alpha=0.37574, lambda=0.0015142
## - Fold2.Rep1: alpha=0.37574, lambda=0.0015142
## + Fold2.Rep1: alpha=0.80531, lambda=0.0023148
## - Fold2.Rep1: alpha=0.80531, lambda=0.0023148
## + Fold2.Rep1: alpha=0.06683, lambda=0.1639706
## - Fold2.Rep1: alpha=0.06683, lambda=0.1639706
## + Fold2.Rep1: alpha=0.01230, lambda=0.3197005
## - Fold2.Rep1: alpha=0.01230, lambda=0.3197005
## + Fold2.Rep1: alpha=0.75340, lambda=0.1553943
## - Fold2.Rep1: alpha=0.75340, lambda=0.1553943
## + Fold2.Rep1: alpha=0.65786, lambda=0.0962132
## - Fold2.Rep1: alpha=0.65786, lambda=0.0962132
## + Fold2.Rep1: alpha=0.42268, lambda=4.0716342
## - Fold2.Rep1: alpha=0.42268, lambda=4.0716342
## + Fold2.Rep1: alpha=0.87538, lambda=0.0525596
## - Fold2.Rep1: alpha=0.87538, lambda=0.0525596
```

```
## + Fold2.Rep1: alpha=0.49128, lambda=6.5121646
## - Fold2.Rep1: alpha=0.49128, lambda=6.5121646
## + Fold2.Rep1: alpha=0.59242, lambda=0.0997138
## - Fold2.Rep1: alpha=0.59242, lambda=0.0997138
## + Fold2.Rep1: alpha=0.29222, lambda=0.0009879
## - Fold2.Rep1: alpha=0.29222, lambda=0.0009879
## + Fold2.Rep1: alpha=0.29635, lambda=1.0847852
## - Fold2.Rep1: alpha=0.29635, lambda=1.0847852
## + Fold2.Rep1: alpha=0.89740, lambda=0.0104120
## - Fold2.Rep1: alpha=0.89740, lambda=0.0104120
## + Fold2.Rep1: alpha=0.38484, lambda=0.0798488
## - Fold2.Rep1: alpha=0.38484, lambda=0.0798488
## + Fold2.Rep1: alpha=0.50231, lambda=0.0405177
## - Fold2.Rep1: alpha=0.50231, lambda=0.0405177
## + Fold2.Rep1: alpha=0.87686, lambda=2.7659135
## - Fold2.Rep1: alpha=0.87686, lambda=2.7659135
## + Fold2.Rep1: alpha=0.78346, lambda=0.3752296
## - Fold2.Rep1: alpha=0.78346, lambda=0.3752296
## + Fold2.Rep1: alpha=0.81532, lambda=0.0358768
## - Fold2.Rep1: alpha=0.81532, lambda=0.0358768
## + Fold2.Rep1: alpha=0.93018, lambda=0.0013239
## - Fold2.Rep1: alpha=0.93018, lambda=0.0013239
## + Fold2.Rep1: alpha=0.91108, lambda=4.6314855
## - Fold2.Rep1: alpha=0.91108, lambda=4.6314855
## + Fold2.Rep1: alpha=0.74424, lambda=0.0070056
## - Fold2.Rep1: alpha=0.74424, lambda=0.0070056
## + Fold2.Rep1: alpha=0.31997, lambda=0.0066009
## - Fold2.Rep1: alpha=0.31997, lambda=0.0066009
## + Fold3.Rep1: alpha=0.63608, lambda=0.0017708
## - Fold3.Rep1: alpha=0.63608, lambda=0.0017708
## + Fold3.Rep1: alpha=0.97864, lambda=0.1779302
## - Fold3.Rep1: alpha=0.97864, lambda=0.1779302
## + Fold3.Rep1: alpha=0.42631, lambda=0.0633234
## - Fold3.Rep1: alpha=0.42631, lambda=0.0633234
## + Fold3.Rep1: alpha=0.37574, lambda=0.0015142
## - Fold3.Rep1: alpha=0.37574, lambda=0.0015142
## + Fold3.Rep1: alpha=0.80531, lambda=0.0023148
## - Fold3.Rep1: alpha=0.80531, lambda=0.0023148
## + Fold3.Rep1: alpha=0.06683, lambda=0.1639706
## - Fold3.Rep1: alpha=0.06683, lambda=0.1639706
## + Fold3.Rep1: alpha=0.01230, lambda=0.3197005
## - Fold3.Rep1: alpha=0.01230, lambda=0.3197005
## + Fold3.Rep1: alpha=0.75340, lambda=0.1553943
## - Fold3.Rep1: alpha=0.75340, lambda=0.1553943
## + Fold3.Rep1: alpha=0.65786, lambda=0.0962132
## - Fold3.Rep1: alpha=0.65786, lambda=0.0962132
## + Fold3.Rep1: alpha=0.42268, lambda=4.0716342
## - Fold3.Rep1: alpha=0.42268, lambda=4.0716342
## + Fold3.Rep1: alpha=0.87538, lambda=0.0525596
## - Fold3.Rep1: alpha=0.87538, lambda=0.0525596
## + Fold3.Rep1: alpha=0.49128, lambda=6.5121646
## - Fold3.Rep1: alpha=0.49128, lambda=6.5121646
## + Fold3.Rep1: alpha=0.59242, lambda=0.0997138
## - Fold3.Rep1: alpha=0.59242, lambda=0.0997138
```

```
## + Fold3.Rep1: alpha=0.29222, lambda=0.0009879
## - Fold3.Rep1: alpha=0.29222, lambda=0.0009879
## + Fold3.Rep1: alpha=0.29635, lambda=1.0847852
## - Fold3.Rep1: alpha=0.29635, lambda=1.0847852
## + Fold3.Rep1: alpha=0.89740, lambda=0.0104120
## - Fold3.Rep1: alpha=0.89740, lambda=0.0104120
## + Fold3.Rep1: alpha=0.38484, lambda=0.0798488
## - Fold3.Rep1: alpha=0.38484, lambda=0.0798488
## + Fold3.Rep1: alpha=0.50231, lambda=0.0405177
## - Fold3.Rep1: alpha=0.50231, lambda=0.0405177
## + Fold3.Rep1: alpha=0.87686, lambda=2.7659135
## - Fold3.Rep1: alpha=0.87686, lambda=2.7659135
## + Fold3.Rep1: alpha=0.78346, lambda=0.3752296
## - Fold3.Rep1: alpha=0.78346, lambda=0.3752296
## + Fold3.Rep1: alpha=0.81532, lambda=0.0358768
## - Fold3.Rep1: alpha=0.81532, lambda=0.0358768
## + Fold3.Rep1: alpha=0.93018, lambda=0.0013239
## - Fold3.Rep1: alpha=0.93018, lambda=0.0013239
## + Fold3.Rep1: alpha=0.91108, lambda=4.6314855
## - Fold3.Rep1: alpha=0.91108, lambda=4.6314855
## + Fold3.Rep1: alpha=0.74424, lambda=0.0070056
## - Fold3.Rep1: alpha=0.74424, lambda=0.0070056
## + Fold3.Rep1: alpha=0.31997, lambda=0.0066009
## - Fold3.Rep1: alpha=0.31997, lambda=0.0066009
## + Fold4.Rep1: alpha=0.63608, lambda=0.0017708
## - Fold4.Rep1: alpha=0.63608, lambda=0.0017708
## + Fold4.Rep1: alpha=0.97864, lambda=0.1779302
## - Fold4.Rep1: alpha=0.97864, lambda=0.1779302
## + Fold4.Rep1: alpha=0.42631, lambda=0.0633234
## - Fold4.Rep1: alpha=0.42631, lambda=0.0633234
## + Fold4.Rep1: alpha=0.37574, lambda=0.0015142
## - Fold4.Rep1: alpha=0.37574, lambda=0.0015142
## + Fold4.Rep1: alpha=0.80531, lambda=0.0023148
## - Fold4.Rep1: alpha=0.80531, lambda=0.0023148
## + Fold4.Rep1: alpha=0.06683, lambda=0.1639706
## - Fold4.Rep1: alpha=0.06683, lambda=0.1639706
## + Fold4.Rep1: alpha=0.01230, lambda=0.3197005
## - Fold4.Rep1: alpha=0.01230, lambda=0.3197005
## + Fold4.Rep1: alpha=0.75340, lambda=0.1553943
## - Fold4.Rep1: alpha=0.75340, lambda=0.1553943
## + Fold4.Rep1: alpha=0.65786, lambda=0.0962132
## - Fold4.Rep1: alpha=0.65786, lambda=0.0962132
## + Fold4.Rep1: alpha=0.42268, lambda=4.0716342
## - Fold4.Rep1: alpha=0.42268, lambda=4.0716342
## + Fold4.Rep1: alpha=0.87538, lambda=0.0525596
## - Fold4.Rep1: alpha=0.87538, lambda=0.0525596
## + Fold4.Rep1: alpha=0.49128, lambda=6.5121646
## - Fold4.Rep1: alpha=0.49128, lambda=6.5121646
## + Fold4.Rep1: alpha=0.59242, lambda=0.0997138
## - Fold4.Rep1: alpha=0.59242, lambda=0.0997138
## + Fold4.Rep1: alpha=0.29222, lambda=0.0009879
## - Fold4.Rep1: alpha=0.29222, lambda=0.0009879
## + Fold4.Rep1: alpha=0.29635, lambda=1.0847852
## - Fold4.Rep1: alpha=0.29635, lambda=1.0847852
```

```
## + Fold4.Rep1: alpha=0.89740, lambda=0.0104120
## - Fold4.Rep1: alpha=0.89740, lambda=0.0104120
## + Fold4.Rep1: alpha=0.38484, lambda=0.0798488
## - Fold4.Rep1: alpha=0.38484, lambda=0.0798488
## + Fold4.Rep1: alpha=0.50231, lambda=0.0405177
## - Fold4.Rep1: alpha=0.50231, lambda=0.0405177
## + Fold4.Rep1: alpha=0.87686, lambda=2.7659135
## - Fold4.Rep1: alpha=0.87686, lambda=2.7659135
## + Fold4.Rep1: alpha=0.78346, lambda=0.3752296
## - Fold4.Rep1: alpha=0.78346, lambda=0.3752296
## + Fold4.Rep1: alpha=0.81532, lambda=0.0358768
## - Fold4.Rep1: alpha=0.81532, lambda=0.0358768
## + Fold4.Rep1: alpha=0.93018, lambda=0.0013239
## - Fold4.Rep1: alpha=0.93018, lambda=0.0013239
## + Fold4.Rep1: alpha=0.91108, lambda=4.6314855
## - Fold4.Rep1: alpha=0.91108, lambda=4.6314855
## + Fold4.Rep1: alpha=0.74424, lambda=0.0070056
## - Fold4.Rep1: alpha=0.74424, lambda=0.0070056
## + Fold4.Rep1: alpha=0.31997, lambda=0.0066009
## - Fold4.Rep1: alpha=0.31997, lambda=0.0066009
## + Fold5.Rep1: alpha=0.63608, lambda=0.0017708
## - Fold5.Rep1: alpha=0.63608, lambda=0.0017708
## + Fold5.Rep1: alpha=0.97864, lambda=0.1779302
## - Fold5.Rep1: alpha=0.97864, lambda=0.1779302
## + Fold5.Rep1: alpha=0.42631, lambda=0.0633234
## - Fold5.Rep1: alpha=0.42631, lambda=0.0633234
## + Fold5.Rep1: alpha=0.37574, lambda=0.0015142
## - Fold5.Rep1: alpha=0.37574, lambda=0.0015142
## + Fold5.Rep1: alpha=0.80531, lambda=0.0023148
## - Fold5.Rep1: alpha=0.80531, lambda=0.0023148
## + Fold5.Rep1: alpha=0.06683, lambda=0.1639706
## - Fold5.Rep1: alpha=0.06683, lambda=0.1639706
## + Fold5.Rep1: alpha=0.01230, lambda=0.3197005
## - Fold5.Rep1: alpha=0.01230, lambda=0.3197005
## + Fold5.Rep1: alpha=0.75340, lambda=0.1553943
## - Fold5.Rep1: alpha=0.75340, lambda=0.1553943
## + Fold5.Rep1: alpha=0.65786, lambda=0.0962132
## - Fold5.Rep1: alpha=0.65786, lambda=0.0962132
## + Fold5.Rep1: alpha=0.42268, lambda=4.0716342
## - Fold5.Rep1: alpha=0.42268, lambda=4.0716342
## + Fold5.Rep1: alpha=0.87538, lambda=0.0525596
## - Fold5.Rep1: alpha=0.87538, lambda=0.0525596
## + Fold5.Rep1: alpha=0.49128, lambda=6.5121646
## - Fold5.Rep1: alpha=0.49128, lambda=6.5121646
## + Fold5.Rep1: alpha=0.59242, lambda=0.0997138
## - Fold5.Rep1: alpha=0.59242, lambda=0.0997138
## + Fold5.Rep1: alpha=0.29222, lambda=0.0009879
## - Fold5.Rep1: alpha=0.29222, lambda=0.0009879
## + Fold5.Rep1: alpha=0.29635, lambda=1.0847852
## - Fold5.Rep1: alpha=0.29635, lambda=1.0847852
## + Fold5.Rep1: alpha=0.89740, lambda=0.0104120
## - Fold5.Rep1: alpha=0.89740, lambda=0.0104120
## + Fold5.Rep1: alpha=0.38484, lambda=0.0798488
## - Fold5.Rep1: alpha=0.38484, lambda=0.0798488
```

```
## + Fold5.Rep1: alpha=0.50231, lambda=0.0405177
## - Fold5.Rep1: alpha=0.50231, lambda=0.0405177
## + Fold5.Rep1: alpha=0.87686, lambda=2.7659135
## - Fold5.Rep1: alpha=0.87686, lambda=2.7659135
## + Fold5.Rep1: alpha=0.78346, lambda=0.3752296
## - Fold5.Rep1: alpha=0.78346, lambda=0.3752296
## + Fold5.Rep1: alpha=0.81532, lambda=0.0358768
## - Fold5.Rep1: alpha=0.81532, lambda=0.0358768
## + Fold5.Rep1: alpha=0.93018, lambda=0.0013239
## - Fold5.Rep1: alpha=0.93018, lambda=0.0013239
## + Fold5.Rep1: alpha=0.91108, lambda=4.6314855
## - Fold5.Rep1: alpha=0.91108, lambda=4.6314855
## + Fold5.Rep1: alpha=0.74424, lambda=0.0070056
## - Fold5.Rep1: alpha=0.74424, lambda=0.0070056
## + Fold5.Rep1: alpha=0.31997, lambda=0.0066009
## - Fold5.Rep1: alpha=0.31997, lambda=0.0066009
## + Fold1.Rep2: alpha=0.63608, lambda=0.0017708
## - Fold1.Rep2: alpha=0.63608, lambda=0.0017708
## + Fold1.Rep2: alpha=0.97864, lambda=0.1779302
## - Fold1.Rep2: alpha=0.97864, lambda=0.1779302
## + Fold1.Rep2: alpha=0.42631, lambda=0.0633234
## - Fold1.Rep2: alpha=0.42631, lambda=0.0633234
## + Fold1.Rep2: alpha=0.37574, lambda=0.0015142
## - Fold1.Rep2: alpha=0.37574, lambda=0.0015142
## + Fold1.Rep2: alpha=0.80531, lambda=0.0023148
## - Fold1.Rep2: alpha=0.80531, lambda=0.0023148
## + Fold1.Rep2: alpha=0.06683, lambda=0.1639706
## - Fold1.Rep2: alpha=0.06683, lambda=0.1639706
## + Fold1.Rep2: alpha=0.01230, lambda=0.3197005
## - Fold1.Rep2: alpha=0.01230, lambda=0.3197005
## + Fold1.Rep2: alpha=0.75340, lambda=0.1553943
## - Fold1.Rep2: alpha=0.75340, lambda=0.1553943
## + Fold1.Rep2: alpha=0.65786, lambda=0.0962132
## - Fold1.Rep2: alpha=0.65786, lambda=0.0962132
## + Fold1.Rep2: alpha=0.42268, lambda=4.0716342
## - Fold1.Rep2: alpha=0.42268, lambda=4.0716342
## + Fold1.Rep2: alpha=0.87538, lambda=0.0525596
## - Fold1.Rep2: alpha=0.87538, lambda=0.0525596
## + Fold1.Rep2: alpha=0.49128, lambda=6.5121646
## - Fold1.Rep2: alpha=0.49128, lambda=6.5121646
## + Fold1.Rep2: alpha=0.59242, lambda=0.0997138
## - Fold1.Rep2: alpha=0.59242, lambda=0.0997138
## + Fold1.Rep2: alpha=0.29222, lambda=0.0009879
## - Fold1.Rep2: alpha=0.29222, lambda=0.0009879
## + Fold1.Rep2: alpha=0.29635, lambda=1.0847852
## - Fold1.Rep2: alpha=0.29635, lambda=1.0847852
## + Fold1.Rep2: alpha=0.89740, lambda=0.0104120
## - Fold1.Rep2: alpha=0.89740, lambda=0.0104120
## + Fold1.Rep2: alpha=0.38484, lambda=0.0798488
## - Fold1.Rep2: alpha=0.38484, lambda=0.0798488
## + Fold1.Rep2: alpha=0.50231, lambda=0.0405177
## - Fold1.Rep2: alpha=0.50231, lambda=0.0405177
## + Fold1.Rep2: alpha=0.87686, lambda=2.7659135
## - Fold1.Rep2: alpha=0.87686, lambda=2.7659135
```

```
## + Fold1.Rep2: alpha=0.78346, lambda=0.3752296
## - Fold1.Rep2: alpha=0.78346, lambda=0.3752296
## + Fold1.Rep2: alpha=0.81532, lambda=0.0358768
## - Fold1.Rep2: alpha=0.81532, lambda=0.0358768
## + Fold1.Rep2: alpha=0.93018, lambda=0.0013239
## - Fold1.Rep2: alpha=0.93018, lambda=0.0013239
## + Fold1.Rep2: alpha=0.91108, lambda=4.6314855
## - Fold1.Rep2: alpha=0.91108, lambda=4.6314855
## + Fold1.Rep2: alpha=0.74424, lambda=0.0070056
## - Fold1.Rep2: alpha=0.74424, lambda=0.0070056
## + Fold1.Rep2: alpha=0.31997, lambda=0.0066009
## - Fold1.Rep2: alpha=0.31997, lambda=0.0066009
## + Fold2.Rep2: alpha=0.63608, lambda=0.0017708
## - Fold2.Rep2: alpha=0.63608, lambda=0.0017708
## + Fold2.Rep2: alpha=0.97864, lambda=0.1779302
## - Fold2.Rep2: alpha=0.97864, lambda=0.1779302
## + Fold2.Rep2: alpha=0.42631, lambda=0.0633234
## - Fold2.Rep2: alpha=0.42631, lambda=0.0633234
## + Fold2.Rep2: alpha=0.37574, lambda=0.0015142
## - Fold2.Rep2: alpha=0.37574, lambda=0.0015142
## + Fold2.Rep2: alpha=0.80531, lambda=0.0023148
## - Fold2.Rep2: alpha=0.80531, lambda=0.0023148
## + Fold2.Rep2: alpha=0.06683, lambda=0.1639706
## - Fold2.Rep2: alpha=0.06683, lambda=0.1639706
## + Fold2.Rep2: alpha=0.01230, lambda=0.3197005
## - Fold2.Rep2: alpha=0.01230, lambda=0.3197005
## + Fold2.Rep2: alpha=0.75340, lambda=0.1553943
## - Fold2.Rep2: alpha=0.75340, lambda=0.1553943
## + Fold2.Rep2: alpha=0.65786, lambda=0.0962132
## - Fold2.Rep2: alpha=0.65786, lambda=0.0962132
## + Fold2.Rep2: alpha=0.42268, lambda=4.0716342
## - Fold2.Rep2: alpha=0.42268, lambda=4.0716342
## + Fold2.Rep2: alpha=0.87538, lambda=0.0525596
## - Fold2.Rep2: alpha=0.87538, lambda=0.0525596
## + Fold2.Rep2: alpha=0.49128, lambda=6.5121646
## - Fold2.Rep2: alpha=0.49128, lambda=6.5121646
## + Fold2.Rep2: alpha=0.59242, lambda=0.0997138
## - Fold2.Rep2: alpha=0.59242, lambda=0.0997138
## + Fold2.Rep2: alpha=0.29222, lambda=0.0009879
## - Fold2.Rep2: alpha=0.29222, lambda=0.0009879
## + Fold2.Rep2: alpha=0.29635, lambda=1.0847852
## - Fold2.Rep2: alpha=0.29635, lambda=1.0847852
## + Fold2.Rep2: alpha=0.89740, lambda=0.0104120
## - Fold2.Rep2: alpha=0.89740, lambda=0.0104120
## + Fold2.Rep2: alpha=0.38484, lambda=0.0798488
## - Fold2.Rep2: alpha=0.38484, lambda=0.0798488
## + Fold2.Rep2: alpha=0.50231, lambda=0.0405177
## - Fold2.Rep2: alpha=0.50231, lambda=0.0405177
## + Fold2.Rep2: alpha=0.87686, lambda=2.7659135
## - Fold2.Rep2: alpha=0.87686, lambda=2.7659135
## + Fold2.Rep2: alpha=0.78346, lambda=0.3752296
## - Fold2.Rep2: alpha=0.78346, lambda=0.3752296
## + Fold2.Rep2: alpha=0.81532, lambda=0.0358768
## - Fold2.Rep2: alpha=0.81532, lambda=0.0358768
```

```
## + Fold2.Rep2: alpha=0.93018, lambda=0.0013239
## - Fold2.Rep2: alpha=0.93018, lambda=0.0013239
## + Fold2.Rep2: alpha=0.91108, lambda=4.6314855
## - Fold2.Rep2: alpha=0.91108, lambda=4.6314855
## + Fold2.Rep2: alpha=0.74424, lambda=0.0070056
## - Fold2.Rep2: alpha=0.74424, lambda=0.0070056
## + Fold2.Rep2: alpha=0.31997, lambda=0.0066009
## - Fold2.Rep2: alpha=0.31997, lambda=0.0066009
## + Fold3.Rep2: alpha=0.63608, lambda=0.0017708
## - Fold3.Rep2: alpha=0.63608, lambda=0.0017708
## + Fold3.Rep2: alpha=0.97864, lambda=0.1779302
## - Fold3.Rep2: alpha=0.97864, lambda=0.1779302
## + Fold3.Rep2: alpha=0.42631, lambda=0.0633234
## - Fold3.Rep2: alpha=0.42631, lambda=0.0633234
## + Fold3.Rep2: alpha=0.37574, lambda=0.0015142
## - Fold3.Rep2: alpha=0.37574, lambda=0.0015142
## + Fold3.Rep2: alpha=0.80531, lambda=0.0023148
## - Fold3.Rep2: alpha=0.80531, lambda=0.0023148
## + Fold3.Rep2: alpha=0.06683, lambda=0.1639706
## - Fold3.Rep2: alpha=0.06683, lambda=0.1639706
## + Fold3.Rep2: alpha=0.01230, lambda=0.3197005
## - Fold3.Rep2: alpha=0.01230, lambda=0.3197005
## + Fold3.Rep2: alpha=0.75340, lambda=0.1553943
## - Fold3.Rep2: alpha=0.75340, lambda=0.1553943
## + Fold3.Rep2: alpha=0.65786, lambda=0.0962132
## - Fold3.Rep2: alpha=0.65786, lambda=0.0962132
## + Fold3.Rep2: alpha=0.42268, lambda=4.0716342
## - Fold3.Rep2: alpha=0.42268, lambda=4.0716342
## + Fold3.Rep2: alpha=0.87538, lambda=0.0525596
## - Fold3.Rep2: alpha=0.87538, lambda=0.0525596
## + Fold3.Rep2: alpha=0.49128, lambda=6.5121646
## - Fold3.Rep2: alpha=0.49128, lambda=6.5121646
## + Fold3.Rep2: alpha=0.59242, lambda=0.0997138
## - Fold3.Rep2: alpha=0.59242, lambda=0.0997138
## + Fold3.Rep2: alpha=0.29222, lambda=0.0009879
## - Fold3.Rep2: alpha=0.29222, lambda=0.0009879
## + Fold3.Rep2: alpha=0.29635, lambda=1.0847852
## - Fold3.Rep2: alpha=0.29635, lambda=1.0847852
## + Fold3.Rep2: alpha=0.89740, lambda=0.0104120
## - Fold3.Rep2: alpha=0.89740, lambda=0.0104120
## + Fold3.Rep2: alpha=0.38484, lambda=0.0798488
## - Fold3.Rep2: alpha=0.38484, lambda=0.0798488
## + Fold3.Rep2: alpha=0.50231, lambda=0.0405177
## - Fold3.Rep2: alpha=0.50231, lambda=0.0405177
## + Fold3.Rep2: alpha=0.87686, lambda=2.7659135
## - Fold3.Rep2: alpha=0.87686, lambda=2.7659135
## + Fold3.Rep2: alpha=0.78346, lambda=0.3752296
## - Fold3.Rep2: alpha=0.78346, lambda=0.3752296
## + Fold3.Rep2: alpha=0.81532, lambda=0.0358768
## - Fold3.Rep2: alpha=0.81532, lambda=0.0358768
## + Fold3.Rep2: alpha=0.93018, lambda=0.0013239
## - Fold3.Rep2: alpha=0.93018, lambda=0.0013239
## + Fold3.Rep2: alpha=0.91108, lambda=4.6314855
## - Fold3.Rep2: alpha=0.91108, lambda=4.6314855
```

```
## + Fold3.Rep2: alpha=0.74424, lambda=0.0070056
## - Fold3.Rep2: alpha=0.74424, lambda=0.0070056
## + Fold3.Rep2: alpha=0.31997, lambda=0.0066009
## - Fold3.Rep2: alpha=0.31997, lambda=0.0066009
## + Fold4.Rep2: alpha=0.63608, lambda=0.0017708
## - Fold4.Rep2: alpha=0.63608, lambda=0.0017708
## + Fold4.Rep2: alpha=0.97864, lambda=0.1779302
## - Fold4.Rep2: alpha=0.97864, lambda=0.1779302
## + Fold4.Rep2: alpha=0.42631, lambda=0.0633234
## - Fold4.Rep2: alpha=0.42631, lambda=0.0633234
## + Fold4.Rep2: alpha=0.37574, lambda=0.0015142
## - Fold4.Rep2: alpha=0.37574, lambda=0.0015142
## + Fold4.Rep2: alpha=0.80531, lambda=0.0023148
## - Fold4.Rep2: alpha=0.80531, lambda=0.0023148
## + Fold4.Rep2: alpha=0.06683, lambda=0.1639706
## - Fold4.Rep2: alpha=0.06683, lambda=0.1639706
## + Fold4.Rep2: alpha=0.01230, lambda=0.3197005
## - Fold4.Rep2: alpha=0.01230, lambda=0.3197005
## + Fold4.Rep2: alpha=0.75340, lambda=0.1553943
## - Fold4.Rep2: alpha=0.75340, lambda=0.1553943
## + Fold4.Rep2: alpha=0.65786, lambda=0.0962132
## - Fold4.Rep2: alpha=0.65786, lambda=0.0962132
## + Fold4.Rep2: alpha=0.42268, lambda=4.0716342
## - Fold4.Rep2: alpha=0.42268, lambda=4.0716342
## + Fold4.Rep2: alpha=0.87538, lambda=0.0525596
## - Fold4.Rep2: alpha=0.87538, lambda=0.0525596
## + Fold4.Rep2: alpha=0.49128, lambda=6.5121646
## - Fold4.Rep2: alpha=0.49128, lambda=6.5121646
## + Fold4.Rep2: alpha=0.59242, lambda=0.0997138
## - Fold4.Rep2: alpha=0.59242, lambda=0.0997138
## + Fold4.Rep2: alpha=0.29222, lambda=0.0009879
## - Fold4.Rep2: alpha=0.29222, lambda=0.0009879
## + Fold4.Rep2: alpha=0.29635, lambda=1.0847852
## - Fold4.Rep2: alpha=0.29635, lambda=1.0847852
## + Fold4.Rep2: alpha=0.89740, lambda=0.0104120
## - Fold4.Rep2: alpha=0.89740, lambda=0.0104120
## + Fold4.Rep2: alpha=0.38484, lambda=0.0798488
## - Fold4.Rep2: alpha=0.38484, lambda=0.0798488
## + Fold4.Rep2: alpha=0.50231, lambda=0.0405177
## - Fold4.Rep2: alpha=0.50231, lambda=0.0405177
## + Fold4.Rep2: alpha=0.87686, lambda=2.7659135
## - Fold4.Rep2: alpha=0.87686, lambda=2.7659135
## + Fold4.Rep2: alpha=0.78346, lambda=0.3752296
## - Fold4.Rep2: alpha=0.78346, lambda=0.3752296
## + Fold4.Rep2: alpha=0.81532, lambda=0.0358768
## - Fold4.Rep2: alpha=0.81532, lambda=0.0358768
## + Fold4.Rep2: alpha=0.93018, lambda=0.0013239
## - Fold4.Rep2: alpha=0.93018, lambda=0.0013239
## + Fold4.Rep2: alpha=0.91108, lambda=4.6314855
## - Fold4.Rep2: alpha=0.91108, lambda=4.6314855
## + Fold4.Rep2: alpha=0.74424, lambda=0.0070056
## - Fold4.Rep2: alpha=0.74424, lambda=0.0070056
## + Fold4.Rep2: alpha=0.31997, lambda=0.0066009
## - Fold4.Rep2: alpha=0.31997, lambda=0.0066009
```

```
## + Fold5.Rep2: alpha=0.63608, lambda=0.0017708
## - Fold5.Rep2: alpha=0.63608, lambda=0.0017708
## + Fold5.Rep2: alpha=0.97864, lambda=0.1779302
## - Fold5.Rep2: alpha=0.97864, lambda=0.1779302
## + Fold5.Rep2: alpha=0.42631, lambda=0.0633234
## - Fold5.Rep2: alpha=0.42631, lambda=0.0633234
## + Fold5.Rep2: alpha=0.37574, lambda=0.0015142
## - Fold5.Rep2: alpha=0.37574, lambda=0.0015142
## + Fold5.Rep2: alpha=0.80531, lambda=0.0023148
## - Fold5.Rep2: alpha=0.80531, lambda=0.0023148
## + Fold5.Rep2: alpha=0.06683, lambda=0.1639706
## - Fold5.Rep2: alpha=0.06683, lambda=0.1639706
## + Fold5.Rep2: alpha=0.01230, lambda=0.3197005
## - Fold5.Rep2: alpha=0.01230, lambda=0.3197005
## + Fold5.Rep2: alpha=0.75340, lambda=0.1553943
## - Fold5.Rep2: alpha=0.75340, lambda=0.1553943
## + Fold5.Rep2: alpha=0.65786, lambda=0.0962132
## - Fold5.Rep2: alpha=0.65786, lambda=0.0962132
## + Fold5.Rep2: alpha=0.42268, lambda=4.0716342
## - Fold5.Rep2: alpha=0.42268, lambda=4.0716342
## + Fold5.Rep2: alpha=0.87538, lambda=0.0525596
## - Fold5.Rep2: alpha=0.87538, lambda=0.0525596
## + Fold5.Rep2: alpha=0.49128, lambda=6.5121646
## - Fold5.Rep2: alpha=0.49128, lambda=6.5121646
## + Fold5.Rep2: alpha=0.59242, lambda=0.0997138
## - Fold5.Rep2: alpha=0.59242, lambda=0.0997138
## + Fold5.Rep2: alpha=0.29222, lambda=0.0009879
## - Fold5.Rep2: alpha=0.29222, lambda=0.0009879
## + Fold5.Rep2: alpha=0.29635, lambda=1.0847852
## - Fold5.Rep2: alpha=0.29635, lambda=1.0847852
## + Fold5.Rep2: alpha=0.89740, lambda=0.0104120
## - Fold5.Rep2: alpha=0.89740, lambda=0.0104120
## + Fold5.Rep2: alpha=0.38484, lambda=0.0798488
## - Fold5.Rep2: alpha=0.38484, lambda=0.0798488
## + Fold5.Rep2: alpha=0.50231, lambda=0.0405177
## - Fold5.Rep2: alpha=0.50231, lambda=0.0405177
## + Fold5.Rep2: alpha=0.87686, lambda=2.7659135
## - Fold5.Rep2: alpha=0.87686, lambda=2.7659135
## + Fold5.Rep2: alpha=0.78346, lambda=0.3752296
## - Fold5.Rep2: alpha=0.78346, lambda=0.3752296
## + Fold5.Rep2: alpha=0.81532, lambda=0.0358768
## - Fold5.Rep2: alpha=0.81532, lambda=0.0358768
## + Fold5.Rep2: alpha=0.93018, lambda=0.0013239
## - Fold5.Rep2: alpha=0.93018, lambda=0.0013239
## + Fold5.Rep2: alpha=0.91108, lambda=4.6314855
## - Fold5.Rep2: alpha=0.91108, lambda=4.6314855
## + Fold5.Rep2: alpha=0.74424, lambda=0.0070056
## - Fold5.Rep2: alpha=0.74424, lambda=0.0070056
## + Fold5.Rep2: alpha=0.31997, lambda=0.0066009
## - Fold5.Rep2: alpha=0.31997, lambda=0.0066009
## + Fold1.Rep3: alpha=0.63608, lambda=0.0017708
## - Fold1.Rep3: alpha=0.63608, lambda=0.0017708
## + Fold1.Rep3: alpha=0.97864, lambda=0.1779302
## - Fold1.Rep3: alpha=0.97864, lambda=0.1779302
```

```
## + Fold1.Rep3: alpha=0.42631, lambda=0.0633234
## - Fold1.Rep3: alpha=0.42631, lambda=0.0633234
## + Fold1.Rep3: alpha=0.37574, lambda=0.0015142
## - Fold1.Rep3: alpha=0.37574, lambda=0.0015142
## + Fold1.Rep3: alpha=0.80531, lambda=0.0023148
## - Fold1.Rep3: alpha=0.80531, lambda=0.0023148
## + Fold1.Rep3: alpha=0.06683, lambda=0.1639706
## - Fold1.Rep3: alpha=0.06683, lambda=0.1639706
## + Fold1.Rep3: alpha=0.01230, lambda=0.3197005
## - Fold1.Rep3: alpha=0.01230, lambda=0.3197005
## + Fold1.Rep3: alpha=0.75340, lambda=0.1553943
## - Fold1.Rep3: alpha=0.75340, lambda=0.1553943
## + Fold1.Rep3: alpha=0.65786, lambda=0.0962132
## - Fold1.Rep3: alpha=0.65786, lambda=0.0962132
## + Fold1.Rep3: alpha=0.42268, lambda=4.0716342
## - Fold1.Rep3: alpha=0.42268, lambda=4.0716342
## + Fold1.Rep3: alpha=0.87538, lambda=0.0525596
## - Fold1.Rep3: alpha=0.87538, lambda=0.0525596
## + Fold1.Rep3: alpha=0.49128, lambda=6.5121646
## - Fold1.Rep3: alpha=0.49128, lambda=6.5121646
## + Fold1.Rep3: alpha=0.59242, lambda=0.0997138
## - Fold1.Rep3: alpha=0.59242, lambda=0.0997138
## + Fold1.Rep3: alpha=0.29222, lambda=0.0009879
## - Fold1.Rep3: alpha=0.29222, lambda=0.0009879
## + Fold1.Rep3: alpha=0.29635, lambda=1.0847852
## - Fold1.Rep3: alpha=0.29635, lambda=1.0847852
## + Fold1.Rep3: alpha=0.89740, lambda=0.0104120
## - Fold1.Rep3: alpha=0.89740, lambda=0.0104120
## + Fold1.Rep3: alpha=0.38484, lambda=0.0798488
## - Fold1.Rep3: alpha=0.38484, lambda=0.0798488
## + Fold1.Rep3: alpha=0.50231, lambda=0.0405177
## - Fold1.Rep3: alpha=0.50231, lambda=0.0405177
## + Fold1.Rep3: alpha=0.87686, lambda=2.7659135
## - Fold1.Rep3: alpha=0.87686, lambda=2.7659135
## + Fold1.Rep3: alpha=0.78346, lambda=0.3752296
## - Fold1.Rep3: alpha=0.78346, lambda=0.3752296
## + Fold1.Rep3: alpha=0.81532, lambda=0.0358768
## - Fold1.Rep3: alpha=0.81532, lambda=0.0358768
## + Fold1.Rep3: alpha=0.93018, lambda=0.0013239
## - Fold1.Rep3: alpha=0.93018, lambda=0.0013239
## + Fold1.Rep3: alpha=0.91108, lambda=4.6314855
## - Fold1.Rep3: alpha=0.91108, lambda=4.6314855
## + Fold1.Rep3: alpha=0.74424, lambda=0.0070056
## - Fold1.Rep3: alpha=0.74424, lambda=0.0070056
## + Fold1.Rep3: alpha=0.31997, lambda=0.0066009
## - Fold1.Rep3: alpha=0.31997, lambda=0.0066009
## + Fold2.Rep3: alpha=0.63608, lambda=0.0017708
## - Fold2.Rep3: alpha=0.63608, lambda=0.0017708
## + Fold2.Rep3: alpha=0.97864, lambda=0.1779302
## - Fold2.Rep3: alpha=0.97864, lambda=0.1779302
## + Fold2.Rep3: alpha=0.42631, lambda=0.0633234
## - Fold2.Rep3: alpha=0.42631, lambda=0.0633234
## + Fold2.Rep3: alpha=0.37574, lambda=0.0015142
## - Fold2.Rep3: alpha=0.37574, lambda=0.0015142
```

```
## + Fold2.Rep3: alpha=0.80531, lambda=0.0023148
## - Fold2.Rep3: alpha=0.80531, lambda=0.0023148
## + Fold2.Rep3: alpha=0.06683, lambda=0.1639706
## - Fold2.Rep3: alpha=0.06683, lambda=0.1639706
## + Fold2.Rep3: alpha=0.01230, lambda=0.3197005
## - Fold2.Rep3: alpha=0.01230, lambda=0.3197005
## + Fold2.Rep3: alpha=0.75340, lambda=0.1553943
## - Fold2.Rep3: alpha=0.75340, lambda=0.1553943
## + Fold2.Rep3: alpha=0.65786, lambda=0.0962132
## - Fold2.Rep3: alpha=0.65786, lambda=0.0962132
## + Fold2.Rep3: alpha=0.42268, lambda=4.0716342
## - Fold2.Rep3: alpha=0.42268, lambda=4.0716342
## + Fold2.Rep3: alpha=0.87538, lambda=0.0525596
## - Fold2.Rep3: alpha=0.87538, lambda=0.0525596
## + Fold2.Rep3: alpha=0.49128, lambda=6.5121646
## - Fold2.Rep3: alpha=0.49128, lambda=6.5121646
## + Fold2.Rep3: alpha=0.59242, lambda=0.0997138
## - Fold2.Rep3: alpha=0.59242, lambda=0.0997138
## + Fold2.Rep3: alpha=0.29222, lambda=0.0009879
## - Fold2.Rep3: alpha=0.29222, lambda=0.0009879
## + Fold2.Rep3: alpha=0.29635, lambda=1.0847852
## - Fold2.Rep3: alpha=0.29635, lambda=1.0847852
## + Fold2.Rep3: alpha=0.89740, lambda=0.0104120
## - Fold2.Rep3: alpha=0.89740, lambda=0.0104120
## + Fold2.Rep3: alpha=0.38484, lambda=0.0798488
## - Fold2.Rep3: alpha=0.38484, lambda=0.0798488
## + Fold2.Rep3: alpha=0.50231, lambda=0.0405177
## - Fold2.Rep3: alpha=0.50231, lambda=0.0405177
## + Fold2.Rep3: alpha=0.87686, lambda=2.7659135
## - Fold2.Rep3: alpha=0.87686, lambda=2.7659135
## + Fold2.Rep3: alpha=0.78346, lambda=0.3752296
## - Fold2.Rep3: alpha=0.78346, lambda=0.3752296
## + Fold2.Rep3: alpha=0.81532, lambda=0.0358768
## - Fold2.Rep3: alpha=0.81532, lambda=0.0358768
## + Fold2.Rep3: alpha=0.93018, lambda=0.0013239
## - Fold2.Rep3: alpha=0.93018, lambda=0.0013239
## + Fold2.Rep3: alpha=0.91108, lambda=4.6314855
## - Fold2.Rep3: alpha=0.91108, lambda=4.6314855
## + Fold2.Rep3: alpha=0.74424, lambda=0.0070056
## - Fold2.Rep3: alpha=0.74424, lambda=0.0070056
## + Fold2.Rep3: alpha=0.31997, lambda=0.0066009
## - Fold2.Rep3: alpha=0.31997, lambda=0.0066009
## + Fold3.Rep3: alpha=0.63608, lambda=0.0017708
## - Fold3.Rep3: alpha=0.63608, lambda=0.0017708
## + Fold3.Rep3: alpha=0.97864, lambda=0.1779302
## - Fold3.Rep3: alpha=0.97864, lambda=0.1779302
## + Fold3.Rep3: alpha=0.42631, lambda=0.0633234
## - Fold3.Rep3: alpha=0.42631, lambda=0.0633234
## + Fold3.Rep3: alpha=0.37574, lambda=0.0015142
## - Fold3.Rep3: alpha=0.37574, lambda=0.0015142
## + Fold3.Rep3: alpha=0.80531, lambda=0.0023148
## - Fold3.Rep3: alpha=0.80531, lambda=0.0023148
## + Fold3.Rep3: alpha=0.06683, lambda=0.1639706
## - Fold3.Rep3: alpha=0.06683, lambda=0.1639706
```

```
## + Fold3.Rep3: alpha=0.01230, lambda=0.3197005
## - Fold3.Rep3: alpha=0.01230, lambda=0.3197005
## + Fold3.Rep3: alpha=0.75340, lambda=0.1553943
## - Fold3.Rep3: alpha=0.75340, lambda=0.1553943
## + Fold3.Rep3: alpha=0.65786, lambda=0.0962132
## - Fold3.Rep3: alpha=0.65786, lambda=0.0962132
## + Fold3.Rep3: alpha=0.42268, lambda=4.0716342
## - Fold3.Rep3: alpha=0.42268, lambda=4.0716342
## + Fold3.Rep3: alpha=0.87538, lambda=0.0525596
## - Fold3.Rep3: alpha=0.87538, lambda=0.0525596
## + Fold3.Rep3: alpha=0.49128, lambda=6.5121646
## - Fold3.Rep3: alpha=0.49128, lambda=6.5121646
## + Fold3.Rep3: alpha=0.59242, lambda=0.0997138
## - Fold3.Rep3: alpha=0.59242, lambda=0.0997138
## + Fold3.Rep3: alpha=0.29222, lambda=0.0009879
## - Fold3.Rep3: alpha=0.29222, lambda=0.0009879
## + Fold3.Rep3: alpha=0.29635, lambda=1.0847852
## - Fold3.Rep3: alpha=0.29635, lambda=1.0847852
## + Fold3.Rep3: alpha=0.89740, lambda=0.0104120
## - Fold3.Rep3: alpha=0.89740, lambda=0.0104120
## + Fold3.Rep3: alpha=0.38484, lambda=0.0798488
## - Fold3.Rep3: alpha=0.38484, lambda=0.0798488
## + Fold3.Rep3: alpha=0.50231, lambda=0.0405177
## - Fold3.Rep3: alpha=0.50231, lambda=0.0405177
## + Fold3.Rep3: alpha=0.87686, lambda=2.7659135
## - Fold3.Rep3: alpha=0.87686, lambda=2.7659135
## + Fold3.Rep3: alpha=0.78346, lambda=0.3752296
## - Fold3.Rep3: alpha=0.78346, lambda=0.3752296
## + Fold3.Rep3: alpha=0.81532, lambda=0.0358768
## - Fold3.Rep3: alpha=0.81532, lambda=0.0358768
## + Fold3.Rep3: alpha=0.93018, lambda=0.0013239
## - Fold3.Rep3: alpha=0.93018, lambda=0.0013239
## + Fold3.Rep3: alpha=0.91108, lambda=4.6314855
## - Fold3.Rep3: alpha=0.91108, lambda=4.6314855
## + Fold3.Rep3: alpha=0.74424, lambda=0.0070056
## - Fold3.Rep3: alpha=0.74424, lambda=0.0070056
## + Fold3.Rep3: alpha=0.31997, lambda=0.0066009
## - Fold3.Rep3: alpha=0.31997, lambda=0.0066009
## + Fold4.Rep3: alpha=0.63608, lambda=0.0017708
## - Fold4.Rep3: alpha=0.63608, lambda=0.0017708
## + Fold4.Rep3: alpha=0.97864, lambda=0.1779302
## - Fold4.Rep3: alpha=0.97864, lambda=0.1779302
## + Fold4.Rep3: alpha=0.42631, lambda=0.0633234
## - Fold4.Rep3: alpha=0.42631, lambda=0.0633234
## + Fold4.Rep3: alpha=0.37574, lambda=0.0015142
## - Fold4.Rep3: alpha=0.37574, lambda=0.0015142
## + Fold4.Rep3: alpha=0.80531, lambda=0.0023148
## - Fold4.Rep3: alpha=0.80531, lambda=0.0023148
## + Fold4.Rep3: alpha=0.06683, lambda=0.1639706
## - Fold4.Rep3: alpha=0.06683, lambda=0.1639706
## + Fold4.Rep3: alpha=0.01230, lambda=0.3197005
## - Fold4.Rep3: alpha=0.01230, lambda=0.3197005
## + Fold4.Rep3: alpha=0.75340, lambda=0.1553943
## - Fold4.Rep3: alpha=0.75340, lambda=0.1553943
```

```
## + Fold4.Rep3: alpha=0.65786, lambda=0.0962132
## - Fold4.Rep3: alpha=0.65786, lambda=0.0962132
## + Fold4.Rep3: alpha=0.42268, lambda=4.0716342
## - Fold4.Rep3: alpha=0.42268, lambda=4.0716342
## + Fold4.Rep3: alpha=0.87538, lambda=0.0525596
## - Fold4.Rep3: alpha=0.87538, lambda=0.0525596
## + Fold4.Rep3: alpha=0.49128, lambda=6.5121646
## - Fold4.Rep3: alpha=0.49128, lambda=6.5121646
## + Fold4.Rep3: alpha=0.59242, lambda=0.0997138
## - Fold4.Rep3: alpha=0.59242, lambda=0.0997138
## + Fold4.Rep3: alpha=0.29222, lambda=0.0009879
## - Fold4.Rep3: alpha=0.29222, lambda=0.0009879
## + Fold4.Rep3: alpha=0.29635, lambda=1.0847852
## - Fold4.Rep3: alpha=0.29635, lambda=1.0847852
## + Fold4.Rep3: alpha=0.89740, lambda=0.0104120
## - Fold4.Rep3: alpha=0.89740, lambda=0.0104120
## + Fold4.Rep3: alpha=0.38484, lambda=0.0798488
## - Fold4.Rep3: alpha=0.38484, lambda=0.0798488
## + Fold4.Rep3: alpha=0.50231, lambda=0.0405177
## - Fold4.Rep3: alpha=0.50231, lambda=0.0405177
## + Fold4.Rep3: alpha=0.87686, lambda=2.7659135
## - Fold4.Rep3: alpha=0.87686, lambda=2.7659135
## + Fold4.Rep3: alpha=0.78346, lambda=0.3752296
## - Fold4.Rep3: alpha=0.78346, lambda=0.3752296
## + Fold4.Rep3: alpha=0.81532, lambda=0.0358768
## - Fold4.Rep3: alpha=0.81532, lambda=0.0358768
## + Fold4.Rep3: alpha=0.93018, lambda=0.0013239
## - Fold4.Rep3: alpha=0.93018, lambda=0.0013239
## + Fold4.Rep3: alpha=0.91108, lambda=4.6314855
## - Fold4.Rep3: alpha=0.91108, lambda=4.6314855
## + Fold4.Rep3: alpha=0.74424, lambda=0.0070056
## - Fold4.Rep3: alpha=0.74424, lambda=0.0070056
## + Fold4.Rep3: alpha=0.31997, lambda=0.0066009
## - Fold4.Rep3: alpha=0.31997, lambda=0.0066009
## + Fold5.Rep3: alpha=0.63608, lambda=0.0017708
## - Fold5.Rep3: alpha=0.63608, lambda=0.0017708
## + Fold5.Rep3: alpha=0.97864, lambda=0.1779302
## - Fold5.Rep3: alpha=0.97864, lambda=0.1779302
## + Fold5.Rep3: alpha=0.42631, lambda=0.0633234
## - Fold5.Rep3: alpha=0.42631, lambda=0.0633234
## + Fold5.Rep3: alpha=0.37574, lambda=0.0015142
## - Fold5.Rep3: alpha=0.37574, lambda=0.0015142
## + Fold5.Rep3: alpha=0.80531, lambda=0.0023148
## - Fold5.Rep3: alpha=0.80531, lambda=0.0023148
## + Fold5.Rep3: alpha=0.06683, lambda=0.1639706
## - Fold5.Rep3: alpha=0.06683, lambda=0.1639706
## + Fold5.Rep3: alpha=0.01230, lambda=0.3197005
## - Fold5.Rep3: alpha=0.01230, lambda=0.3197005
## + Fold5.Rep3: alpha=0.75340, lambda=0.1553943
## - Fold5.Rep3: alpha=0.75340, lambda=0.1553943
## + Fold5.Rep3: alpha=0.65786, lambda=0.0962132
## - Fold5.Rep3: alpha=0.65786, lambda=0.0962132
## + Fold5.Rep3: alpha=0.42268, lambda=4.0716342
## - Fold5.Rep3: alpha=0.42268, lambda=4.0716342
```

```
## + Fold5.Rep3: alpha=0.87538, lambda=0.0525596
## - Fold5.Rep3: alpha=0.87538, lambda=0.0525596
## + Fold5.Rep3: alpha=0.49128, lambda=6.5121646
## - Fold5.Rep3: alpha=0.49128, lambda=6.5121646
## + Fold5.Rep3: alpha=0.59242, lambda=0.0997138
## - Fold5.Rep3: alpha=0.59242, lambda=0.0997138
## + Fold5.Rep3: alpha=0.29222, lambda=0.0009879
## - Fold5.Rep3: alpha=0.29222, lambda=0.0009879
## + Fold5.Rep3: alpha=0.29635, lambda=1.0847852
## - Fold5.Rep3: alpha=0.29635, lambda=1.0847852
## + Fold5.Rep3: alpha=0.89740, lambda=0.0104120
## - Fold5.Rep3: alpha=0.89740, lambda=0.0104120
## + Fold5.Rep3: alpha=0.38484, lambda=0.0798488
## - Fold5.Rep3: alpha=0.38484, lambda=0.0798488
## + Fold5.Rep3: alpha=0.50231, lambda=0.0405177
## - Fold5.Rep3: alpha=0.50231, lambda=0.0405177
## + Fold5.Rep3: alpha=0.87686, lambda=2.7659135
## - Fold5.Rep3: alpha=0.87686, lambda=2.7659135
## + Fold5.Rep3: alpha=0.78346, lambda=0.3752296
## - Fold5.Rep3: alpha=0.78346, lambda=0.3752296
## + Fold5.Rep3: alpha=0.81532, lambda=0.0358768
## - Fold5.Rep3: alpha=0.81532, lambda=0.0358768
## + Fold5.Rep3: alpha=0.93018, lambda=0.0013239
## - Fold5.Rep3: alpha=0.93018, lambda=0.0013239
## + Fold5.Rep3: alpha=0.91108, lambda=4.6314855
## - Fold5.Rep3: alpha=0.91108, lambda=4.6314855
## + Fold5.Rep3: alpha=0.74424, lambda=0.0070056
## - Fold5.Rep3: alpha=0.74424, lambda=0.0070056
## + Fold5.Rep3: alpha=0.31997, lambda=0.0066009
## - Fold5.Rep3: alpha=0.31997, lambda=0.0066009
## + Fold1.Rep4: alpha=0.63608, lambda=0.0017708
## - Fold1.Rep4: alpha=0.63608, lambda=0.0017708
## + Fold1.Rep4: alpha=0.97864, lambda=0.1779302
## - Fold1.Rep4: alpha=0.97864, lambda=0.1779302
## + Fold1.Rep4: alpha=0.42631, lambda=0.0633234
## - Fold1.Rep4: alpha=0.42631, lambda=0.0633234
## + Fold1.Rep4: alpha=0.37574, lambda=0.0015142
## - Fold1.Rep4: alpha=0.37574, lambda=0.0015142
## + Fold1.Rep4: alpha=0.80531, lambda=0.0023148
## - Fold1.Rep4: alpha=0.80531, lambda=0.0023148
## + Fold1.Rep4: alpha=0.06683, lambda=0.1639706
## - Fold1.Rep4: alpha=0.06683, lambda=0.1639706
## + Fold1.Rep4: alpha=0.01230, lambda=0.3197005
## - Fold1.Rep4: alpha=0.01230, lambda=0.3197005
## + Fold1.Rep4: alpha=0.75340, lambda=0.1553943
## - Fold1.Rep4: alpha=0.75340, lambda=0.1553943
## + Fold1.Rep4: alpha=0.65786, lambda=0.0962132
## - Fold1.Rep4: alpha=0.65786, lambda=0.0962132
## + Fold1.Rep4: alpha=0.42268, lambda=4.0716342
## - Fold1.Rep4: alpha=0.42268, lambda=4.0716342
## + Fold1.Rep4: alpha=0.87538, lambda=0.0525596
## - Fold1.Rep4: alpha=0.87538, lambda=0.0525596
## + Fold1.Rep4: alpha=0.49128, lambda=6.5121646
## - Fold1.Rep4: alpha=0.49128, lambda=6.5121646
```

```
## + Fold1.Rep4: alpha=0.59242, lambda=0.0997138
## - Fold1.Rep4: alpha=0.59242, lambda=0.0997138
## + Fold1.Rep4: alpha=0.29222, lambda=0.0009879
## - Fold1.Rep4: alpha=0.29222, lambda=0.0009879
## + Fold1.Rep4: alpha=0.29635, lambda=1.0847852
## - Fold1.Rep4: alpha=0.29635, lambda=1.0847852
## + Fold1.Rep4: alpha=0.89740, lambda=0.0104120
## - Fold1.Rep4: alpha=0.89740, lambda=0.0104120
## + Fold1.Rep4: alpha=0.38484, lambda=0.0798488
## - Fold1.Rep4: alpha=0.38484, lambda=0.0798488
## + Fold1.Rep4: alpha=0.50231, lambda=0.0405177
## - Fold1.Rep4: alpha=0.50231, lambda=0.0405177
## + Fold1.Rep4: alpha=0.87686, lambda=2.7659135
## - Fold1.Rep4: alpha=0.87686, lambda=2.7659135
## + Fold1.Rep4: alpha=0.78346, lambda=0.3752296
## - Fold1.Rep4: alpha=0.78346, lambda=0.3752296
## + Fold1.Rep4: alpha=0.81532, lambda=0.0358768
## - Fold1.Rep4: alpha=0.81532, lambda=0.0358768
## + Fold1.Rep4: alpha=0.93018, lambda=0.0013239
## - Fold1.Rep4: alpha=0.93018, lambda=0.0013239
## + Fold1.Rep4: alpha=0.91108, lambda=4.6314855
## - Fold1.Rep4: alpha=0.91108, lambda=4.6314855
## + Fold1.Rep4: alpha=0.74424, lambda=0.0070056
## - Fold1.Rep4: alpha=0.74424, lambda=0.0070056
## + Fold1.Rep4: alpha=0.31997, lambda=0.0066009
## - Fold1.Rep4: alpha=0.31997, lambda=0.0066009
## + Fold2.Rep4: alpha=0.63608, lambda=0.0017708
## - Fold2.Rep4: alpha=0.63608, lambda=0.0017708
## + Fold2.Rep4: alpha=0.97864, lambda=0.1779302
## - Fold2.Rep4: alpha=0.97864, lambda=0.1779302
## + Fold2.Rep4: alpha=0.42631, lambda=0.0633234
## - Fold2.Rep4: alpha=0.42631, lambda=0.0633234
## + Fold2.Rep4: alpha=0.37574, lambda=0.0015142
## - Fold2.Rep4: alpha=0.37574, lambda=0.0015142
## + Fold2.Rep4: alpha=0.80531, lambda=0.0023148
## - Fold2.Rep4: alpha=0.80531, lambda=0.0023148
## + Fold2.Rep4: alpha=0.06683, lambda=0.1639706
## - Fold2.Rep4: alpha=0.06683, lambda=0.1639706
## + Fold2.Rep4: alpha=0.01230, lambda=0.3197005
## - Fold2.Rep4: alpha=0.01230, lambda=0.3197005
## + Fold2.Rep4: alpha=0.75340, lambda=0.1553943
## - Fold2.Rep4: alpha=0.75340, lambda=0.1553943
## + Fold2.Rep4: alpha=0.65786, lambda=0.0962132
## - Fold2.Rep4: alpha=0.65786, lambda=0.0962132
## + Fold2.Rep4: alpha=0.42268, lambda=4.0716342
## - Fold2.Rep4: alpha=0.42268, lambda=4.0716342
## + Fold2.Rep4: alpha=0.87538, lambda=0.0525596
## - Fold2.Rep4: alpha=0.87538, lambda=0.0525596
## + Fold2.Rep4: alpha=0.49128, lambda=6.5121646
## - Fold2.Rep4: alpha=0.49128, lambda=6.5121646
## + Fold2.Rep4: alpha=0.59242, lambda=0.0997138
## - Fold2.Rep4: alpha=0.59242, lambda=0.0997138
## + Fold2.Rep4: alpha=0.29222, lambda=0.0009879
## - Fold2.Rep4: alpha=0.29222, lambda=0.0009879
```

```
## + Fold2.Rep4: alpha=0.29635, lambda=1.0847852
## - Fold2.Rep4: alpha=0.29635, lambda=1.0847852
## + Fold2.Rep4: alpha=0.89740, lambda=0.0104120
## - Fold2.Rep4: alpha=0.89740, lambda=0.0104120
## + Fold2.Rep4: alpha=0.38484, lambda=0.0798488
## - Fold2.Rep4: alpha=0.38484, lambda=0.0798488
## + Fold2.Rep4: alpha=0.50231, lambda=0.0405177
## - Fold2.Rep4: alpha=0.50231, lambda=0.0405177
## + Fold2.Rep4: alpha=0.87686, lambda=2.7659135
## - Fold2.Rep4: alpha=0.87686, lambda=2.7659135
## + Fold2.Rep4: alpha=0.78346, lambda=0.3752296
## - Fold2.Rep4: alpha=0.78346, lambda=0.3752296
## + Fold2.Rep4: alpha=0.81532, lambda=0.0358768
## - Fold2.Rep4: alpha=0.81532, lambda=0.0358768
## + Fold2.Rep4: alpha=0.93018, lambda=0.0013239
## - Fold2.Rep4: alpha=0.93018, lambda=0.0013239
## + Fold2.Rep4: alpha=0.91108, lambda=4.6314855
## - Fold2.Rep4: alpha=0.91108, lambda=4.6314855
## + Fold2.Rep4: alpha=0.74424, lambda=0.0070056
## - Fold2.Rep4: alpha=0.74424, lambda=0.0070056
## + Fold2.Rep4: alpha=0.31997, lambda=0.0066009
## - Fold2.Rep4: alpha=0.31997, lambda=0.0066009
## + Fold3.Rep4: alpha=0.63608, lambda=0.0017708
## - Fold3.Rep4: alpha=0.63608, lambda=0.0017708
## + Fold3.Rep4: alpha=0.97864, lambda=0.1779302
## - Fold3.Rep4: alpha=0.97864, lambda=0.1779302
## + Fold3.Rep4: alpha=0.42631, lambda=0.0633234
## - Fold3.Rep4: alpha=0.42631, lambda=0.0633234
## + Fold3.Rep4: alpha=0.37574, lambda=0.0015142
## - Fold3.Rep4: alpha=0.37574, lambda=0.0015142
## + Fold3.Rep4: alpha=0.80531, lambda=0.0023148
## - Fold3.Rep4: alpha=0.80531, lambda=0.0023148
## + Fold3.Rep4: alpha=0.06683, lambda=0.1639706
## - Fold3.Rep4: alpha=0.06683, lambda=0.1639706
## + Fold3.Rep4: alpha=0.01230, lambda=0.3197005
## - Fold3.Rep4: alpha=0.01230, lambda=0.3197005
## + Fold3.Rep4: alpha=0.75340, lambda=0.1553943
## - Fold3.Rep4: alpha=0.75340, lambda=0.1553943
## + Fold3.Rep4: alpha=0.65786, lambda=0.0962132
## - Fold3.Rep4: alpha=0.65786, lambda=0.0962132
## + Fold3.Rep4: alpha=0.42268, lambda=4.0716342
## - Fold3.Rep4: alpha=0.42268, lambda=4.0716342
## + Fold3.Rep4: alpha=0.87538, lambda=0.0525596
## - Fold3.Rep4: alpha=0.87538, lambda=0.0525596
## + Fold3.Rep4: alpha=0.49128, lambda=6.5121646
## - Fold3.Rep4: alpha=0.49128, lambda=6.5121646
## + Fold3.Rep4: alpha=0.59242, lambda=0.0997138
## - Fold3.Rep4: alpha=0.59242, lambda=0.0997138
## + Fold3.Rep4: alpha=0.29222, lambda=0.0009879
## - Fold3.Rep4: alpha=0.29222, lambda=0.0009879
## + Fold3.Rep4: alpha=0.29635, lambda=1.0847852
## - Fold3.Rep4: alpha=0.29635, lambda=1.0847852
## + Fold3.Rep4: alpha=0.89740, lambda=0.0104120
## - Fold3.Rep4: alpha=0.89740, lambda=0.0104120
```

```
## + Fold3.Rep4: alpha=0.38484, lambda=0.0798488
## - Fold3.Rep4: alpha=0.38484, lambda=0.0798488
## + Fold3.Rep4: alpha=0.50231, lambda=0.0405177
## - Fold3.Rep4: alpha=0.50231, lambda=0.0405177
## + Fold3.Rep4: alpha=0.87686, lambda=2.7659135
## - Fold3.Rep4: alpha=0.87686, lambda=2.7659135
## + Fold3.Rep4: alpha=0.78346, lambda=0.3752296
## - Fold3.Rep4: alpha=0.78346, lambda=0.3752296
## + Fold3.Rep4: alpha=0.81532, lambda=0.0358768
## - Fold3.Rep4: alpha=0.81532, lambda=0.0358768
## + Fold3.Rep4: alpha=0.93018, lambda=0.0013239
## - Fold3.Rep4: alpha=0.93018, lambda=0.0013239
## + Fold3.Rep4: alpha=0.91108, lambda=4.6314855
## - Fold3.Rep4: alpha=0.91108, lambda=4.6314855
## + Fold3.Rep4: alpha=0.74424, lambda=0.0070056
## - Fold3.Rep4: alpha=0.74424, lambda=0.0070056
## + Fold3.Rep4: alpha=0.31997, lambda=0.0066009
## - Fold3.Rep4: alpha=0.31997, lambda=0.0066009
## + Fold4.Rep4: alpha=0.63608, lambda=0.0017708
## - Fold4.Rep4: alpha=0.63608, lambda=0.0017708
## + Fold4.Rep4: alpha=0.97864, lambda=0.1779302
## - Fold4.Rep4: alpha=0.97864, lambda=0.1779302
## + Fold4.Rep4: alpha=0.42631, lambda=0.0633234
## - Fold4.Rep4: alpha=0.42631, lambda=0.0633234
## + Fold4.Rep4: alpha=0.37574, lambda=0.0015142
## - Fold4.Rep4: alpha=0.37574, lambda=0.0015142
## + Fold4.Rep4: alpha=0.80531, lambda=0.0023148
## - Fold4.Rep4: alpha=0.80531, lambda=0.0023148
## + Fold4.Rep4: alpha=0.06683, lambda=0.1639706
## - Fold4.Rep4: alpha=0.06683, lambda=0.1639706
## + Fold4.Rep4: alpha=0.01230, lambda=0.3197005
## - Fold4.Rep4: alpha=0.01230, lambda=0.3197005
## + Fold4.Rep4: alpha=0.75340, lambda=0.1553943
## - Fold4.Rep4: alpha=0.75340, lambda=0.1553943
## + Fold4.Rep4: alpha=0.65786, lambda=0.0962132
## - Fold4.Rep4: alpha=0.65786, lambda=0.0962132
## + Fold4.Rep4: alpha=0.42268, lambda=4.0716342
## - Fold4.Rep4: alpha=0.42268, lambda=4.0716342
## + Fold4.Rep4: alpha=0.87538, lambda=0.0525596
## - Fold4.Rep4: alpha=0.87538, lambda=0.0525596
## + Fold4.Rep4: alpha=0.49128, lambda=6.5121646
## - Fold4.Rep4: alpha=0.49128, lambda=6.5121646
## + Fold4.Rep4: alpha=0.59242, lambda=0.0997138
## - Fold4.Rep4: alpha=0.59242, lambda=0.0997138
## + Fold4.Rep4: alpha=0.29222, lambda=0.0009879
## - Fold4.Rep4: alpha=0.29222, lambda=0.0009879
## + Fold4.Rep4: alpha=0.29635, lambda=1.0847852
## - Fold4.Rep4: alpha=0.29635, lambda=1.0847852
## + Fold4.Rep4: alpha=0.89740, lambda=0.0104120
## - Fold4.Rep4: alpha=0.89740, lambda=0.0104120
## + Fold4.Rep4: alpha=0.38484, lambda=0.0798488
## - Fold4.Rep4: alpha=0.38484, lambda=0.0798488
## + Fold4.Rep4: alpha=0.50231, lambda=0.0405177
## - Fold4.Rep4: alpha=0.50231, lambda=0.0405177
```

```
## + Fold4.Rep4: alpha=0.87686, lambda=2.7659135
## - Fold4.Rep4: alpha=0.87686, lambda=2.7659135
## + Fold4.Rep4: alpha=0.78346, lambda=0.3752296
## - Fold4.Rep4: alpha=0.78346, lambda=0.3752296
## + Fold4.Rep4: alpha=0.81532, lambda=0.0358768
## - Fold4.Rep4: alpha=0.81532, lambda=0.0358768
## + Fold4.Rep4: alpha=0.93018, lambda=0.0013239
## - Fold4.Rep4: alpha=0.93018, lambda=0.0013239
## + Fold4.Rep4: alpha=0.91108, lambda=4.6314855
## - Fold4.Rep4: alpha=0.91108, lambda=4.6314855
## + Fold4.Rep4: alpha=0.74424, lambda=0.0070056
## - Fold4.Rep4: alpha=0.74424, lambda=0.0070056
## + Fold4.Rep4: alpha=0.31997, lambda=0.0066009
## - Fold4.Rep4: alpha=0.31997, lambda=0.0066009
## + Fold5.Rep4: alpha=0.63608, lambda=0.0017708
## - Fold5.Rep4: alpha=0.63608, lambda=0.0017708
## + Fold5.Rep4: alpha=0.97864, lambda=0.1779302
## - Fold5.Rep4: alpha=0.97864, lambda=0.1779302
## + Fold5.Rep4: alpha=0.42631, lambda=0.0633234
## - Fold5.Rep4: alpha=0.42631, lambda=0.0633234
## + Fold5.Rep4: alpha=0.37574, lambda=0.0015142
## - Fold5.Rep4: alpha=0.37574, lambda=0.0015142
## + Fold5.Rep4: alpha=0.80531, lambda=0.0023148
## - Fold5.Rep4: alpha=0.80531, lambda=0.0023148
## + Fold5.Rep4: alpha=0.06683, lambda=0.1639706
## - Fold5.Rep4: alpha=0.06683, lambda=0.1639706
## + Fold5.Rep4: alpha=0.01230, lambda=0.3197005
## - Fold5.Rep4: alpha=0.01230, lambda=0.3197005
## + Fold5.Rep4: alpha=0.75340, lambda=0.1553943
## - Fold5.Rep4: alpha=0.75340, lambda=0.1553943
## + Fold5.Rep4: alpha=0.65786, lambda=0.0962132
## - Fold5.Rep4: alpha=0.65786, lambda=0.0962132
## + Fold5.Rep4: alpha=0.42268, lambda=4.0716342
## - Fold5.Rep4: alpha=0.42268, lambda=4.0716342
## + Fold5.Rep4: alpha=0.87538, lambda=0.0525596
## - Fold5.Rep4: alpha=0.87538, lambda=0.0525596
## + Fold5.Rep4: alpha=0.49128, lambda=6.5121646
## - Fold5.Rep4: alpha=0.49128, lambda=6.5121646
## + Fold5.Rep4: alpha=0.59242, lambda=0.0997138
## - Fold5.Rep4: alpha=0.59242, lambda=0.0997138
## + Fold5.Rep4: alpha=0.29222, lambda=0.0009879
## - Fold5.Rep4: alpha=0.29222, lambda=0.0009879
## + Fold5.Rep4: alpha=0.29635, lambda=1.0847852
## - Fold5.Rep4: alpha=0.29635, lambda=1.0847852
## + Fold5.Rep4: alpha=0.89740, lambda=0.0104120
## - Fold5.Rep4: alpha=0.89740, lambda=0.0104120
## + Fold5.Rep4: alpha=0.38484, lambda=0.0798488
## - Fold5.Rep4: alpha=0.38484, lambda=0.0798488
## + Fold5.Rep4: alpha=0.50231, lambda=0.0405177
## - Fold5.Rep4: alpha=0.50231, lambda=0.0405177
## + Fold5.Rep4: alpha=0.87686, lambda=2.7659135
## - Fold5.Rep4: alpha=0.87686, lambda=2.7659135
## + Fold5.Rep4: alpha=0.78346, lambda=0.3752296
## - Fold5.Rep4: alpha=0.78346, lambda=0.3752296
```

```
## + Fold5.Rep4: alpha=0.81532, lambda=0.0358768
## - Fold5.Rep4: alpha=0.81532, lambda=0.0358768
## + Fold5.Rep4: alpha=0.93018, lambda=0.0013239
## - Fold5.Rep4: alpha=0.93018, lambda=0.0013239
## + Fold5.Rep4: alpha=0.91108, lambda=4.6314855
## - Fold5.Rep4: alpha=0.91108, lambda=4.6314855
## + Fold5.Rep4: alpha=0.74424, lambda=0.0070056
## - Fold5.Rep4: alpha=0.74424, lambda=0.0070056
## + Fold5.Rep4: alpha=0.31997, lambda=0.0066009
## - Fold5.Rep4: alpha=0.31997, lambda=0.0066009
## + Fold1.Rep5: alpha=0.63608, lambda=0.0017708
## - Fold1.Rep5: alpha=0.63608, lambda=0.0017708
## + Fold1.Rep5: alpha=0.97864, lambda=0.1779302
## - Fold1.Rep5: alpha=0.97864, lambda=0.1779302
## + Fold1.Rep5: alpha=0.42631, lambda=0.0633234
## - Fold1.Rep5: alpha=0.42631, lambda=0.0633234
## + Fold1.Rep5: alpha=0.37574, lambda=0.0015142
## - Fold1.Rep5: alpha=0.37574, lambda=0.0015142
## + Fold1.Rep5: alpha=0.80531, lambda=0.0023148
## - Fold1.Rep5: alpha=0.80531, lambda=0.0023148
## + Fold1.Rep5: alpha=0.06683, lambda=0.1639706
## - Fold1.Rep5: alpha=0.06683, lambda=0.1639706
## + Fold1.Rep5: alpha=0.01230, lambda=0.3197005
## - Fold1.Rep5: alpha=0.01230, lambda=0.3197005
## + Fold1.Rep5: alpha=0.75340, lambda=0.1553943
## - Fold1.Rep5: alpha=0.75340, lambda=0.1553943
## + Fold1.Rep5: alpha=0.65786, lambda=0.0962132
## - Fold1.Rep5: alpha=0.65786, lambda=0.0962132
## + Fold1.Rep5: alpha=0.42268, lambda=4.0716342
## - Fold1.Rep5: alpha=0.42268, lambda=4.0716342
## + Fold1.Rep5: alpha=0.87538, lambda=0.0525596
## - Fold1.Rep5: alpha=0.87538, lambda=0.0525596
## + Fold1.Rep5: alpha=0.49128, lambda=6.5121646
## - Fold1.Rep5: alpha=0.49128, lambda=6.5121646
## + Fold1.Rep5: alpha=0.59242, lambda=0.0997138
## - Fold1.Rep5: alpha=0.59242, lambda=0.0997138
## + Fold1.Rep5: alpha=0.29222, lambda=0.0009879
## - Fold1.Rep5: alpha=0.29222, lambda=0.0009879
## + Fold1.Rep5: alpha=0.29635, lambda=1.0847852
## - Fold1.Rep5: alpha=0.29635, lambda=1.0847852
## + Fold1.Rep5: alpha=0.89740, lambda=0.0104120
## - Fold1.Rep5: alpha=0.89740, lambda=0.0104120
## + Fold1.Rep5: alpha=0.38484, lambda=0.0798488
## - Fold1.Rep5: alpha=0.38484, lambda=0.0798488
## + Fold1.Rep5: alpha=0.50231, lambda=0.0405177
## - Fold1.Rep5: alpha=0.50231, lambda=0.0405177
## + Fold1.Rep5: alpha=0.87686, lambda=2.7659135
## - Fold1.Rep5: alpha=0.87686, lambda=2.7659135
## + Fold1.Rep5: alpha=0.78346, lambda=0.3752296
## - Fold1.Rep5: alpha=0.78346, lambda=0.3752296
## + Fold1.Rep5: alpha=0.81532, lambda=0.0358768
## - Fold1.Rep5: alpha=0.81532, lambda=0.0358768
## + Fold1.Rep5: alpha=0.93018, lambda=0.0013239
## - Fold1.Rep5: alpha=0.93018, lambda=0.0013239
```

```
## + Fold1.Rep5: alpha=0.91108, lambda=4.6314855
## - Fold1.Rep5: alpha=0.91108, lambda=4.6314855
## + Fold1.Rep5: alpha=0.74424, lambda=0.0070056
## - Fold1.Rep5: alpha=0.74424, lambda=0.0070056
## + Fold1.Rep5: alpha=0.31997, lambda=0.0066009
## - Fold1.Rep5: alpha=0.31997, lambda=0.0066009
## + Fold2.Rep5: alpha=0.63608, lambda=0.0017708
## - Fold2.Rep5: alpha=0.63608, lambda=0.0017708
## + Fold2.Rep5: alpha=0.97864, lambda=0.1779302
## - Fold2.Rep5: alpha=0.97864, lambda=0.1779302
## + Fold2.Rep5: alpha=0.42631, lambda=0.0633234
## - Fold2.Rep5: alpha=0.42631, lambda=0.0633234
## + Fold2.Rep5: alpha=0.37574, lambda=0.0015142
## - Fold2.Rep5: alpha=0.37574, lambda=0.0015142
## + Fold2.Rep5: alpha=0.80531, lambda=0.0023148
## - Fold2.Rep5: alpha=0.80531, lambda=0.0023148
## + Fold2.Rep5: alpha=0.06683, lambda=0.1639706
## - Fold2.Rep5: alpha=0.06683, lambda=0.1639706
## + Fold2.Rep5: alpha=0.01230, lambda=0.3197005
## - Fold2.Rep5: alpha=0.01230, lambda=0.3197005
## + Fold2.Rep5: alpha=0.75340, lambda=0.1553943
## - Fold2.Rep5: alpha=0.75340, lambda=0.1553943
## + Fold2.Rep5: alpha=0.65786, lambda=0.0962132
## - Fold2.Rep5: alpha=0.65786, lambda=0.0962132
## + Fold2.Rep5: alpha=0.42268, lambda=4.0716342
## - Fold2.Rep5: alpha=0.42268, lambda=4.0716342
## + Fold2.Rep5: alpha=0.87538, lambda=0.0525596
## - Fold2.Rep5: alpha=0.87538, lambda=0.0525596
## + Fold2.Rep5: alpha=0.49128, lambda=6.5121646
## - Fold2.Rep5: alpha=0.49128, lambda=6.5121646
## + Fold2.Rep5: alpha=0.59242, lambda=0.0997138
## - Fold2.Rep5: alpha=0.59242, lambda=0.0997138
## + Fold2.Rep5: alpha=0.29222, lambda=0.0009879
## - Fold2.Rep5: alpha=0.29222, lambda=0.0009879
## + Fold2.Rep5: alpha=0.29635, lambda=1.0847852
## - Fold2.Rep5: alpha=0.29635, lambda=1.0847852
## + Fold2.Rep5: alpha=0.89740, lambda=0.0104120
## - Fold2.Rep5: alpha=0.89740, lambda=0.0104120
## + Fold2.Rep5: alpha=0.38484, lambda=0.0798488
## - Fold2.Rep5: alpha=0.38484, lambda=0.0798488
## + Fold2.Rep5: alpha=0.50231, lambda=0.0405177
## - Fold2.Rep5: alpha=0.50231, lambda=0.0405177
## + Fold2.Rep5: alpha=0.87686, lambda=2.7659135
## - Fold2.Rep5: alpha=0.87686, lambda=2.7659135
## + Fold2.Rep5: alpha=0.78346, lambda=0.3752296
## - Fold2.Rep5: alpha=0.78346, lambda=0.3752296
## + Fold2.Rep5: alpha=0.81532, lambda=0.0358768
## - Fold2.Rep5: alpha=0.81532, lambda=0.0358768
## + Fold2.Rep5: alpha=0.93018, lambda=0.0013239
## - Fold2.Rep5: alpha=0.93018, lambda=0.0013239
## + Fold2.Rep5: alpha=0.91108, lambda=4.6314855
## - Fold2.Rep5: alpha=0.91108, lambda=4.6314855
## + Fold2.Rep5: alpha=0.74424, lambda=0.0070056
## - Fold2.Rep5: alpha=0.74424, lambda=0.0070056
```

```
## + Fold2.Rep5: alpha=0.31997, lambda=0.0066009
## - Fold2.Rep5: alpha=0.31997, lambda=0.0066009
## + Fold3.Rep5: alpha=0.63608, lambda=0.0017708
## - Fold3.Rep5: alpha=0.63608, lambda=0.0017708
## + Fold3.Rep5: alpha=0.97864, lambda=0.1779302
## - Fold3.Rep5: alpha=0.97864, lambda=0.1779302
## + Fold3.Rep5: alpha=0.42631, lambda=0.0633234
## - Fold3.Rep5: alpha=0.42631, lambda=0.0633234
## + Fold3.Rep5: alpha=0.37574, lambda=0.0015142
## - Fold3.Rep5: alpha=0.37574, lambda=0.0015142
## + Fold3.Rep5: alpha=0.80531, lambda=0.0023148
## - Fold3.Rep5: alpha=0.80531, lambda=0.0023148
## + Fold3.Rep5: alpha=0.06683, lambda=0.1639706
## - Fold3.Rep5: alpha=0.06683, lambda=0.1639706
## + Fold3.Rep5: alpha=0.01230, lambda=0.3197005
## - Fold3.Rep5: alpha=0.01230, lambda=0.3197005
## + Fold3.Rep5: alpha=0.75340, lambda=0.1553943
## - Fold3.Rep5: alpha=0.75340, lambda=0.1553943
## + Fold3.Rep5: alpha=0.65786, lambda=0.0962132
## - Fold3.Rep5: alpha=0.65786, lambda=0.0962132
## + Fold3.Rep5: alpha=0.42268, lambda=4.0716342
## - Fold3.Rep5: alpha=0.42268, lambda=4.0716342
## + Fold3.Rep5: alpha=0.87538, lambda=0.0525596
## - Fold3.Rep5: alpha=0.87538, lambda=0.0525596
## + Fold3.Rep5: alpha=0.49128, lambda=6.5121646
## - Fold3.Rep5: alpha=0.49128, lambda=6.5121646
## + Fold3.Rep5: alpha=0.59242, lambda=0.0997138
## - Fold3.Rep5: alpha=0.59242, lambda=0.0997138
## + Fold3.Rep5: alpha=0.29222, lambda=0.0009879
## - Fold3.Rep5: alpha=0.29222, lambda=0.0009879
## + Fold3.Rep5: alpha=0.29635, lambda=1.0847852
## - Fold3.Rep5: alpha=0.29635, lambda=1.0847852
## + Fold3.Rep5: alpha=0.89740, lambda=0.0104120
## - Fold3.Rep5: alpha=0.89740, lambda=0.0104120
## + Fold3.Rep5: alpha=0.38484, lambda=0.0798488
## - Fold3.Rep5: alpha=0.38484, lambda=0.0798488
## + Fold3.Rep5: alpha=0.50231, lambda=0.0405177
## - Fold3.Rep5: alpha=0.50231, lambda=0.0405177
## + Fold3.Rep5: alpha=0.87686, lambda=2.7659135
## - Fold3.Rep5: alpha=0.87686, lambda=2.7659135
## + Fold3.Rep5: alpha=0.78346, lambda=0.3752296
## - Fold3.Rep5: alpha=0.78346, lambda=0.3752296
## + Fold3.Rep5: alpha=0.81532, lambda=0.0358768
## - Fold3.Rep5: alpha=0.81532, lambda=0.0358768
## + Fold3.Rep5: alpha=0.93018, lambda=0.0013239
## - Fold3.Rep5: alpha=0.93018, lambda=0.0013239
## + Fold3.Rep5: alpha=0.91108, lambda=4.6314855
## - Fold3.Rep5: alpha=0.91108, lambda=4.6314855
## + Fold3.Rep5: alpha=0.74424, lambda=0.0070056
## - Fold3.Rep5: alpha=0.74424, lambda=0.0070056
## + Fold3.Rep5: alpha=0.31997, lambda=0.0066009
## - Fold3.Rep5: alpha=0.31997, lambda=0.0066009
## + Fold4.Rep5: alpha=0.63608, lambda=0.0017708
## - Fold4.Rep5: alpha=0.63608, lambda=0.0017708
```

```
## + Fold4.Rep5: alpha=0.97864, lambda=0.1779302
## - Fold4.Rep5: alpha=0.97864, lambda=0.1779302
## + Fold4.Rep5: alpha=0.42631, lambda=0.0633234
## - Fold4.Rep5: alpha=0.42631, lambda=0.0633234
## + Fold4.Rep5: alpha=0.37574, lambda=0.0015142
## - Fold4.Rep5: alpha=0.37574, lambda=0.0015142
## + Fold4.Rep5: alpha=0.80531, lambda=0.0023148
## - Fold4.Rep5: alpha=0.80531, lambda=0.0023148
## + Fold4.Rep5: alpha=0.06683, lambda=0.1639706
## - Fold4.Rep5: alpha=0.06683, lambda=0.1639706
## + Fold4.Rep5: alpha=0.01230, lambda=0.3197005
## - Fold4.Rep5: alpha=0.01230, lambda=0.3197005
## + Fold4.Rep5: alpha=0.75340, lambda=0.1553943
## - Fold4.Rep5: alpha=0.75340, lambda=0.1553943
## + Fold4.Rep5: alpha=0.65786, lambda=0.0962132
## - Fold4.Rep5: alpha=0.65786, lambda=0.0962132
## + Fold4.Rep5: alpha=0.42268, lambda=4.0716342
## - Fold4.Rep5: alpha=0.42268, lambda=4.0716342
## + Fold4.Rep5: alpha=0.87538, lambda=0.0525596
## - Fold4.Rep5: alpha=0.87538, lambda=0.0525596
## + Fold4.Rep5: alpha=0.49128, lambda=6.5121646
## - Fold4.Rep5: alpha=0.49128, lambda=6.5121646
## + Fold4.Rep5: alpha=0.59242, lambda=0.0997138
## - Fold4.Rep5: alpha=0.59242, lambda=0.0997138
## + Fold4.Rep5: alpha=0.29222, lambda=0.0009879
## - Fold4.Rep5: alpha=0.29222, lambda=0.0009879
## + Fold4.Rep5: alpha=0.29635, lambda=1.0847852
## - Fold4.Rep5: alpha=0.29635, lambda=1.0847852
## + Fold4.Rep5: alpha=0.89740, lambda=0.0104120
## - Fold4.Rep5: alpha=0.89740, lambda=0.0104120
## + Fold4.Rep5: alpha=0.38484, lambda=0.0798488
## - Fold4.Rep5: alpha=0.38484, lambda=0.0798488
## + Fold4.Rep5: alpha=0.50231, lambda=0.0405177
## - Fold4.Rep5: alpha=0.50231, lambda=0.0405177
## + Fold4.Rep5: alpha=0.87686, lambda=2.7659135
## - Fold4.Rep5: alpha=0.87686, lambda=2.7659135
## + Fold4.Rep5: alpha=0.78346, lambda=0.3752296
## - Fold4.Rep5: alpha=0.78346, lambda=0.3752296
## + Fold4.Rep5: alpha=0.81532, lambda=0.0358768
## - Fold4.Rep5: alpha=0.81532, lambda=0.0358768
## + Fold4.Rep5: alpha=0.93018, lambda=0.0013239
## - Fold4.Rep5: alpha=0.93018, lambda=0.0013239
## + Fold4.Rep5: alpha=0.91108, lambda=4.6314855
## - Fold4.Rep5: alpha=0.91108, lambda=4.6314855
## + Fold4.Rep5: alpha=0.74424, lambda=0.0070056
## - Fold4.Rep5: alpha=0.74424, lambda=0.0070056
## + Fold4.Rep5: alpha=0.31997, lambda=0.0066009
## - Fold4.Rep5: alpha=0.31997, lambda=0.0066009
## + Fold5.Rep5: alpha=0.63608, lambda=0.0017708
## - Fold5.Rep5: alpha=0.63608, lambda=0.0017708
## + Fold5.Rep5: alpha=0.97864, lambda=0.1779302
## - Fold5.Rep5: alpha=0.97864, lambda=0.1779302
## + Fold5.Rep5: alpha=0.42631, lambda=0.0633234
## - Fold5.Rep5: alpha=0.42631, lambda=0.0633234
```

```
## + Fold5.Rep5: alpha=0.37574, lambda=0.0015142
## - Fold5.Rep5: alpha=0.37574, lambda=0.0015142
## + Fold5.Rep5: alpha=0.80531, lambda=0.0023148
## - Fold5.Rep5: alpha=0.80531, lambda=0.0023148
## + Fold5.Rep5: alpha=0.06683, lambda=0.1639706
## - Fold5.Rep5: alpha=0.06683, lambda=0.1639706
## + Fold5.Rep5: alpha=0.01230, lambda=0.3197005
## - Fold5.Rep5: alpha=0.01230, lambda=0.3197005
## + Fold5.Rep5: alpha=0.75340, lambda=0.1553943
## - Fold5.Rep5: alpha=0.75340, lambda=0.1553943
## + Fold5.Rep5: alpha=0.65786, lambda=0.0962132
## - Fold5.Rep5: alpha=0.65786, lambda=0.0962132
## + Fold5.Rep5: alpha=0.42268, lambda=4.0716342
## - Fold5.Rep5: alpha=0.42268, lambda=4.0716342
## + Fold5.Rep5: alpha=0.87538, lambda=0.0525596
## - Fold5.Rep5: alpha=0.87538, lambda=0.0525596
## + Fold5.Rep5: alpha=0.49128, lambda=6.5121646
## - Fold5.Rep5: alpha=0.49128, lambda=6.5121646
## + Fold5.Rep5: alpha=0.59242, lambda=0.0997138
## - Fold5.Rep5: alpha=0.59242, lambda=0.0997138
## + Fold5.Rep5: alpha=0.29222, lambda=0.0009879
## - Fold5.Rep5: alpha=0.29222, lambda=0.0009879
## + Fold5.Rep5: alpha=0.29635, lambda=1.0847852
## - Fold5.Rep5: alpha=0.29635, lambda=1.0847852
## + Fold5.Rep5: alpha=0.89740, lambda=0.0104120
## - Fold5.Rep5: alpha=0.89740, lambda=0.0104120
## + Fold5.Rep5: alpha=0.38484, lambda=0.0798488
## - Fold5.Rep5: alpha=0.38484, lambda=0.0798488
## + Fold5.Rep5: alpha=0.50231, lambda=0.0405177
## - Fold5.Rep5: alpha=0.50231, lambda=0.0405177
## + Fold5.Rep5: alpha=0.87686, lambda=2.7659135
## - Fold5.Rep5: alpha=0.87686, lambda=2.7659135
## + Fold5.Rep5: alpha=0.78346, lambda=0.3752296
## - Fold5.Rep5: alpha=0.78346, lambda=0.3752296
## + Fold5.Rep5: alpha=0.81532, lambda=0.0358768
## - Fold5.Rep5: alpha=0.81532, lambda=0.0358768
## + Fold5.Rep5: alpha=0.93018, lambda=0.0013239
## - Fold5.Rep5: alpha=0.93018, lambda=0.0013239
## + Fold5.Rep5: alpha=0.91108, lambda=4.6314855
## - Fold5.Rep5: alpha=0.91108, lambda=4.6314855
## + Fold5.Rep5: alpha=0.74424, lambda=0.0070056
## - Fold5.Rep5: alpha=0.74424, lambda=0.0070056
## + Fold5.Rep5: alpha=0.31997, lambda=0.0066009
## - Fold5.Rep5: alpha=0.31997, lambda=0.0066009
## Aggregating results
## Selecting tuning parameters
## Fitting alpha = 0.0668, lambda = 0.164 on full training set
```

Our cross validation results in the tuning parameters $\alpha = 0.001489799$ and $\lambda = 0.007510219$ for our Elastic Net Model. We will now predict the testing data in order to measure the RMSE of our model.

We will now look at the variables our E-Net model finds the most important:

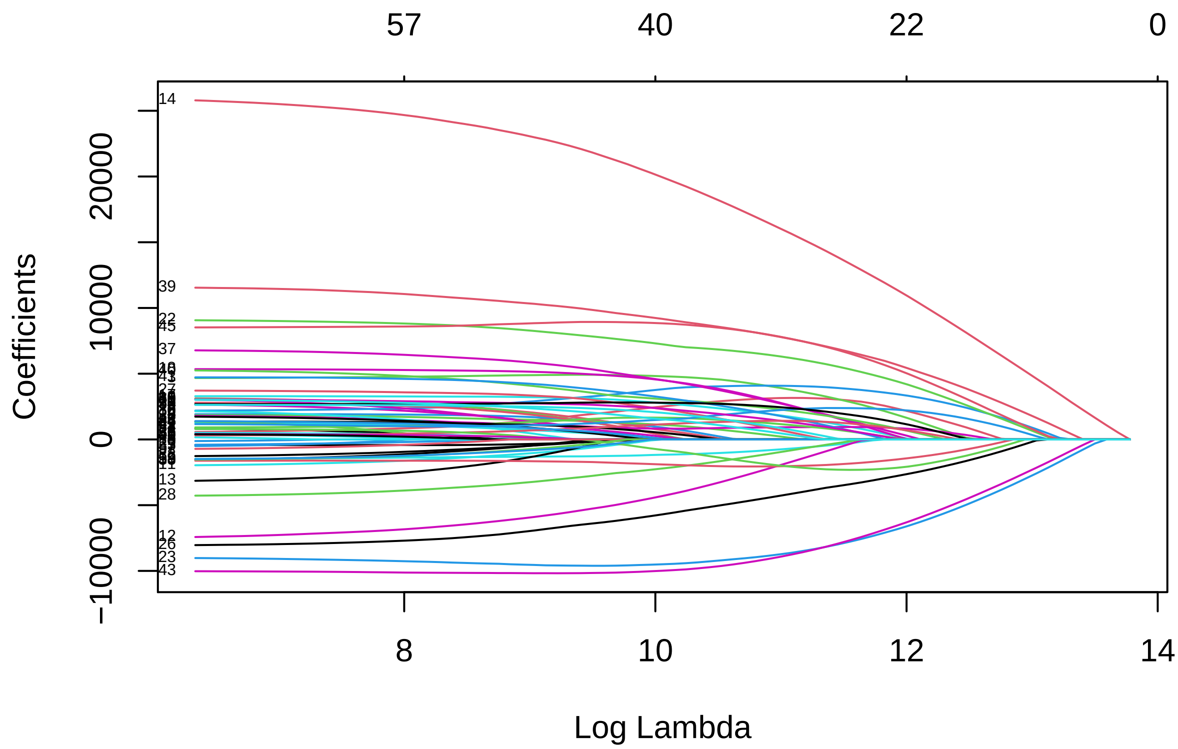
```r
print(elastic_net_model)
```

```
## glmnet
##
## 1024 samples
##   60 predictor
##
## Pre-processing: centered (60), scaled (60)
## Resampling: Cross-Validated (5 fold, repeated 5 times)
## Summary of sample sizes: 820, 818, 820, 819, 819, 819, ...
## Resampling results across tuning parameters:
##
##   alpha       lambda        RMSE      Rsquared   MAE
##   0.01229978  0.3197005277  40745.39  0.7545629  25312.08
##   0.06683004  0.1639706105  40737.38  0.7547006  25306.71
##   0.29222196  0.0009879343  40753.04  0.7546061  25324.78
##   0.29634765  1.0847852177  40752.62  0.7546095  25324.72
##   0.31997283  0.0066008589  40753.54  0.7546021  25325.25
##   0.37573718  0.0015141704  40756.00  0.7545802  25327.66
##   0.38483874  0.0798487895  40756.26  0.7545791  25327.66
##   0.42268111  4.0716341923  40757.09  0.7545720  25328.96
##   0.42630757  0.0633233870  40757.22  0.7545708  25329.08
##   0.49128262  6.5121645952  40758.09  0.7545660  25329.57
##   0.50231095  0.0405176509  40758.35  0.7545634  25329.82
##   0.59242344  0.0997138392  40759.41  0.7545555  25331.33
##   0.63608417  0.0017707770  40760.68  0.7545379  25332.49
##   0.65786011  0.0962131861  40760.70  0.7545378  25332.62
##   0.74423815  0.0070055641  40761.24  0.7545346  25333.33
##   0.75339509  0.1553943124  40760.86  0.7545406  25333.29
##   0.78346385  0.3752296113  40760.74  0.7545452  25333.03
##   0.80531494  0.0023148114  40760.99  0.7545428  25333.27
##   0.81532431  0.0358768471  40760.21  0.7545502  25332.06
##   0.87537882  0.0525596350  40761.54  0.7545354  25333.37
##   0.87685746  2.7659135237  40761.55  0.7545353  25333.38
##   0.89739502  0.0104119842  40761.19  0.7545399  25333.04
##   0.91107522  4.6314855213  40761.31  0.7545388  25333.15
##   0.93017705  0.0013239303  40761.42  0.7545377  25333.26
##   0.97864070  0.1779302308  40761.77  0.7545346  25333.58
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.06683004 and lambda
##  = 0.1639706.
```

```r
elastic_net_model$bestTune
```

```
##        alpha    lambda
## 2 0.06683004 0.1639706
```

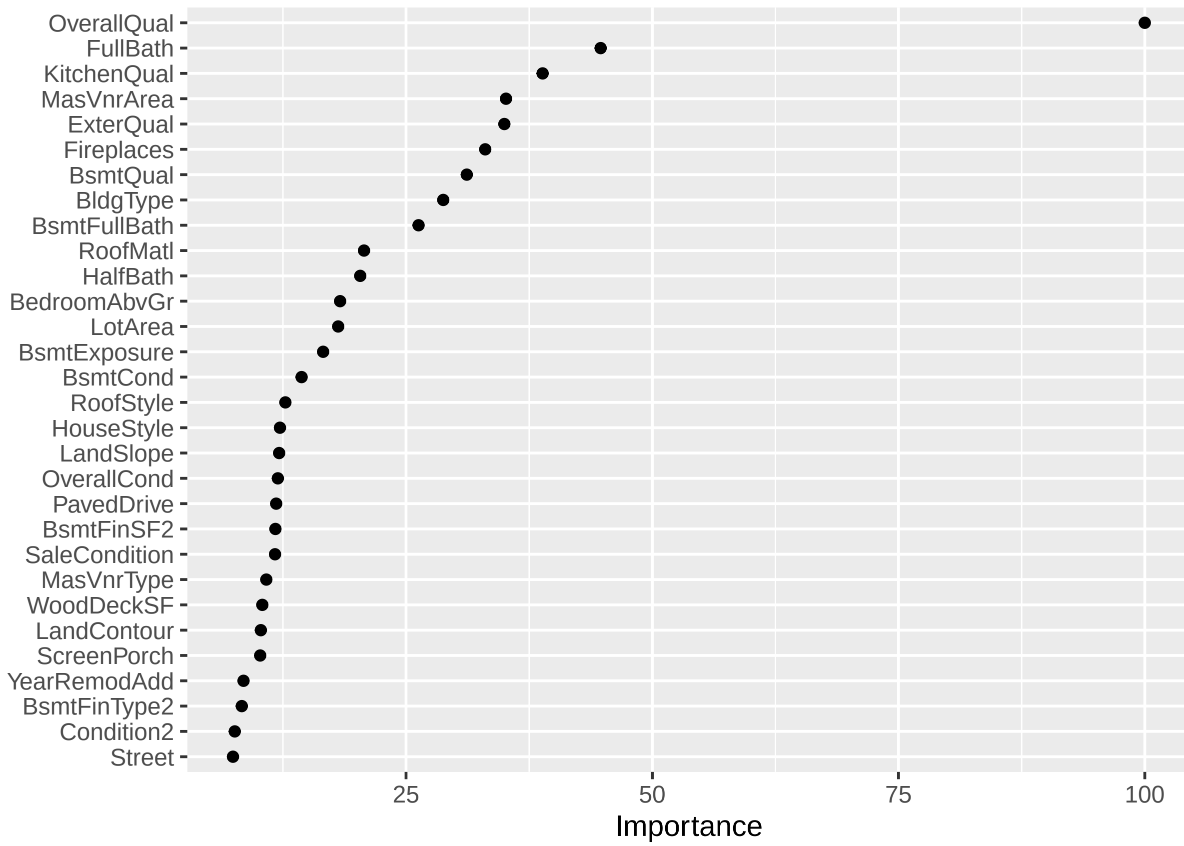
```r
plot(elastic_net_model$finalModel, "lambda", label=TRUE)
```



This plot again shows how the coefficients of different variables react to increasing lambda. This plot demonstrates that the 14th, 39th, 43rd, 23rd, 22nd, 46th, and 26th are highly important, while other variables are less so.

We will now create a variable importance plot to closer examine these important variables.

```r
vip(elastic_net_model, num_features = 30, geom = "point")
```



Our Elastic Net Model shows that Overall Quality is the most important variable by far, then followed by Full Bathrooms, Kitchen Quality, and External Quality - which is in agreement with our Ridge and Lasso models aside from the Street variable.

We will now use our elastic net model fit on the full training set to predict the test set in order to measure RMSE and R-squared.

```
# Predict using the testing data
enet_preds = predict(elastic_net_model, newdata = X_test_scaled)

# Evaluate performance
postResample(enet_preds,y_test)
```

```
##         RMSE     Rsquared          MAE
## 3.311439e+04 8.070347e-01 2.354398e+04
```

This shows us that our Elastic Net Model's best cross-validated performance is $33,115.49$ which is almost $4,000$ higher than our Ridge model's performance.

Model Comparison

Comparing the RMSEs from predicting the test set Sale Price using our four regularization models, we find our principal component regression model performed best with the lowest RMSE of $28,141.38$.

RMSE Comparison: PCR: $28,141.38$ Ridge: $33,211.91$ Lasso: $33,128.05$ E-Net: $33,115.49$

This PCR model found that the number of components with the lowest cross-validation error was ncomps = 31. Thus our final regularization model is a PCR model with ncomps = 31:

```
#Final PCR fit
pcr.fit.final <- pcr(SalePrice ~., data = train_scaled, ncomp = 31)
summary(pcr.fit.final)
```

```
## Data:    X dimension: 1024 73
##  Y dimension: 1024 1
## Fit method: svdpc
## Number of components considered: 31
## TRAINING: % variance explained
##            1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps
## X           26.031    41.39    50.09    54.59    58.06    61.12    63.75
## SalePrice    4.143    12.34    62.22    64.24    65.48    67.62    71.83
##            8 comps  9 comps  10 comps  11 comps  12 comps  13 comps  14 comps
## X            65.99    68.02     70.02     71.83     73.35     74.84     76.07
## SalePrice    74.57    75.20     75.74     75.77     75.78     75.82     76.24
##            15 comps  16 comps  17 comps  18 comps  19 comps  20 comps  21 comps
## X             77.27     78.39     79.50     80.51     81.51     82.41     83.29
## SalePrice     76.29     76.49     76.58     76.85     77.29     77.29     79.01
##            22 comps  23 comps  24 comps  25 comps  26 comps  27 comps  28 comps
## X             84.12     84.93     85.72     86.49     87.22     87.94     88.60
## SalePrice     79.10     79.10     79.29     79.30     79.31     79.43     79.82
##            29 comps  30 comps  31 comps
## X             89.24     89.87     90.49
## SalePrice     79.84     79.85     79.94
```

```
#Predicting Sale Price
pcr.pred.final <- predict(pcr.fit.final, X_test_scaled, ncomp = 31)
```

This PCR model reduced dimensionality from 73 variables down to 31 components and outperformed Ridge, Lasso, and Elastic Net regularization methods making it our model of choice.

Sources

https://remiller1450.github.io/s230f19/caret3.html https://dataaspirant.com/knn-implementation-r-using-caret-package/ https://www.rdocumentation.org/packages/caret/versions/4.47/topics/train