Boston Marathon Finish Time Predictions

Harvard Stats E139 Fall 2015

Nathaniel Burbank
Pooja Singh
David Wihl

December 21, 2015

Abstract

This module contains the base code for the project.

```
create_folds = function (df, k){
  # Perform k-Fold Cross Validation and return average score
  # Inspired by Scikit-Learn's cross_val_score
  if (missing(k)) {
    k = 5
  }

  train1_start = train1_end = train2_start = train2_end = test_start = test_end = c()
  testsize = nrow(df) / k
#  cat("train1 start \t train1_end \t test_start \t test_end \t train2_start \t train2_end \n")
  for (i in 1:k){
    if (i < k){
      train1_start[i] = 1
      train1_end[i] = floor((k-i) * testsize)
    } else {
      train1_start[i] = 0
      train1_end[i] = 0
    }
    test_start[i] = max(ceiling((k-i) * testsize),1)
    test_end[i]   = floor((k-i+1) * testsize)
    train2_start[i] = 0
    train2_end[i] = 0
    if (i > 1){
      train2_start[i] = ceiling((k-i+1) * testsize)
      train2_end[i] = nrow(df)
    }
#    cat(train1_start, "\t\t",train1_end,
#        "\t\t", test_start, "\t\t", test_end,
#        "\t\t",train2_start, "\t\t", train2_end,"\n")

  }
  return (data.frame(train1_start, train1_end, train2_start, train2_end, test_start, test_end))
}


k = 10 # we'll do 10-fold cross validation

# First let's read in the data and calculate finish times:
dfm <- read.csv("Previous Boston Marathon study/BAA data.txt",header=T,sep=" ")
```

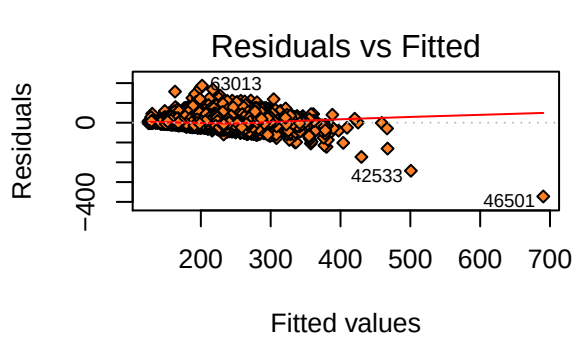
```
times = as.matrix(dfm[,7:15], ncol=9)
dfm$totaltime = rowSums(times)
dfm<- dfm[c("totaltime","Age","Gender1F2M","K0.5")] # keep only columns we need
dfm = dfm[!is.na(dfm$totaltime), ]  # eliminate rows with no finish times
#dfm = dfm[sample(nrow(dfm)),]  # in case the data is sorted, randomize the order

# Create a set of k-fold sets for cross-validation
folds = create_folds(dfm,k)
score = c()

# Baseline regression
base.mod = lm(totaltime~.,data=dfm)
summary(base.mod)
```

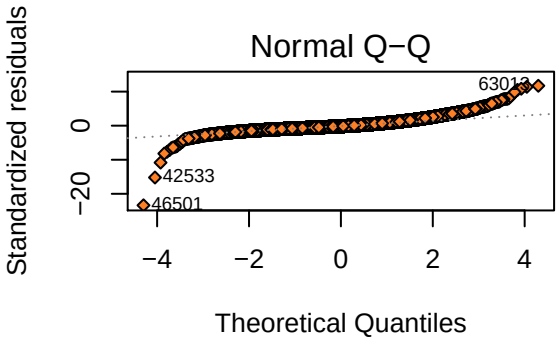
```
##
## Call:
## lm(formula = totaltime ~ ., data = dfm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -376.18   -9.26   -2.64    6.48  187.43
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.640157   0.595532  -41.38   <2e-16 ***
## Age           0.073888   0.006227   11.87   <2e-16 ***
## Gender1F2M    2.847267   0.139233   20.45   <2e-16 ***
## K0.5          9.703124   0.020687  469.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.43 on 64163 degrees of freedom
## Multiple R-squared:  0.8097, Adjusted R-squared:  0.8097
## F-statistic: 9.103e+04 on 3 and 64163 DF,  p-value: < 2.2e-16
```

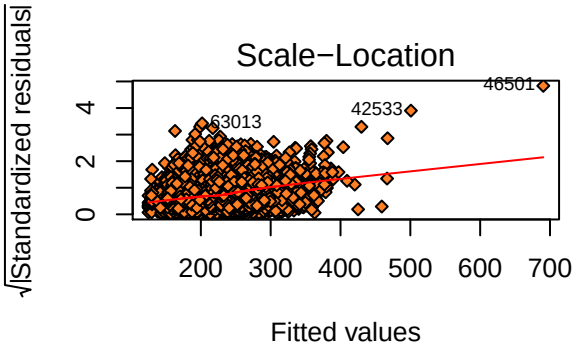
```
# get baseline score
for (i in 1:k) {
  train = dfm[c(folds[i,"train1_start"]:folds[i,"train1_end"],folds[i,"train2_start"]:folds[i,"train2_e
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  model = lm(totaltime~.,data=train)
  score[i] = sum((test$totaltime - predict(model,new=test))^2) / as.numeric(nrow(test))
}
cat ("Baseline mean score is: ", mean(score),"\n")
```

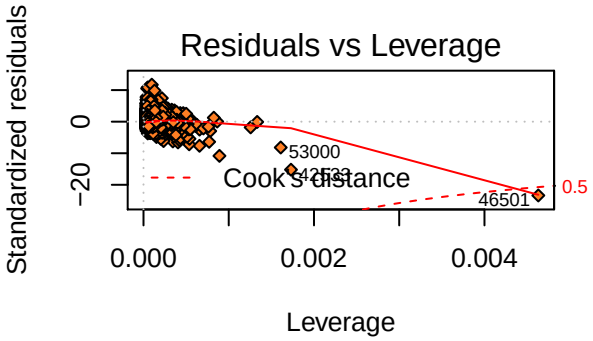
```
## Baseline mean score is:  260.604
```

```
par(mfrow=c(2,2))
plot(model, pch=23 ,bg="chocolate1",cex=.8)
```

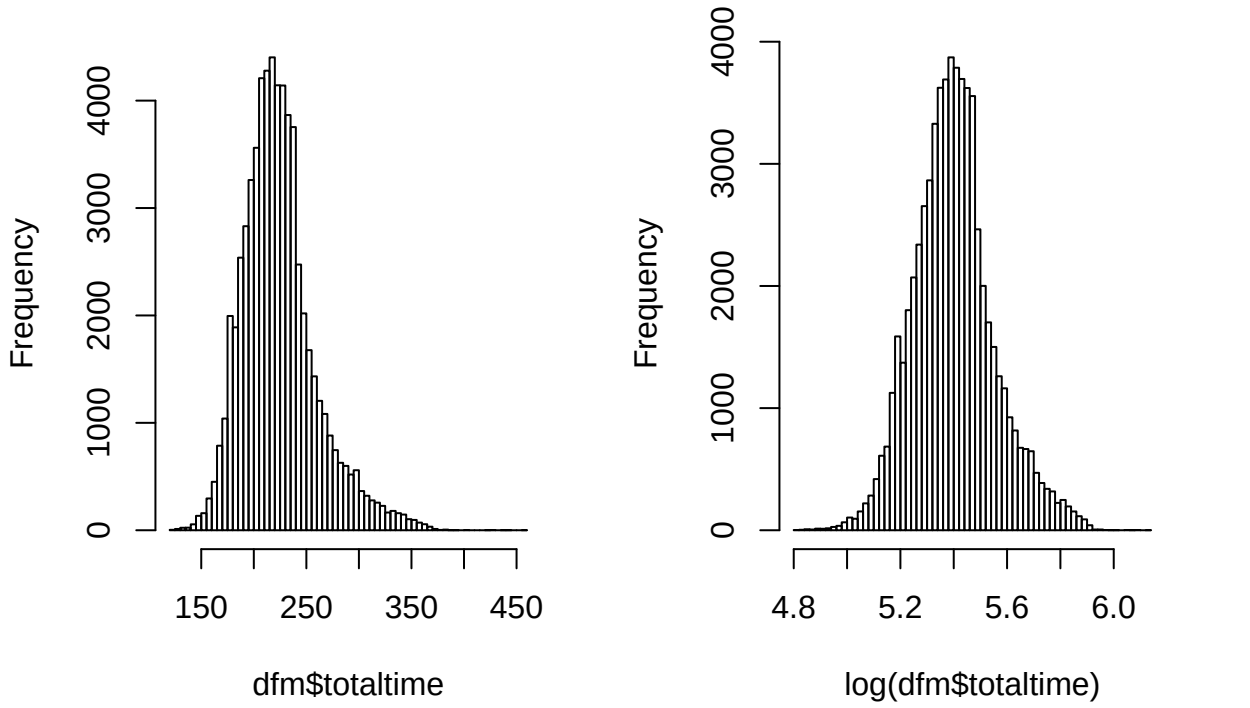









```
#y.hat = predict(base.mod)
#xydata = data.frame(x=y.hat, y=resid(base.mod))
#xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
#plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="Predicted", ylab="Residuals")

## Step 2: Examination of the data and transformations.
par(mfrow=c(1,2))
hist(dfm$totaltime,breaks=50, main="Boston Marathon Finish Time (min) Distribution for '10, '11, '13")

#The data appears somewhat right skewed. Let's try a log transform of the predicted variable:
hist(log(dfm$totaltime),breaks=50, main="Boston Marathon Finish Time (log-min) Distribution for '10, '1
```



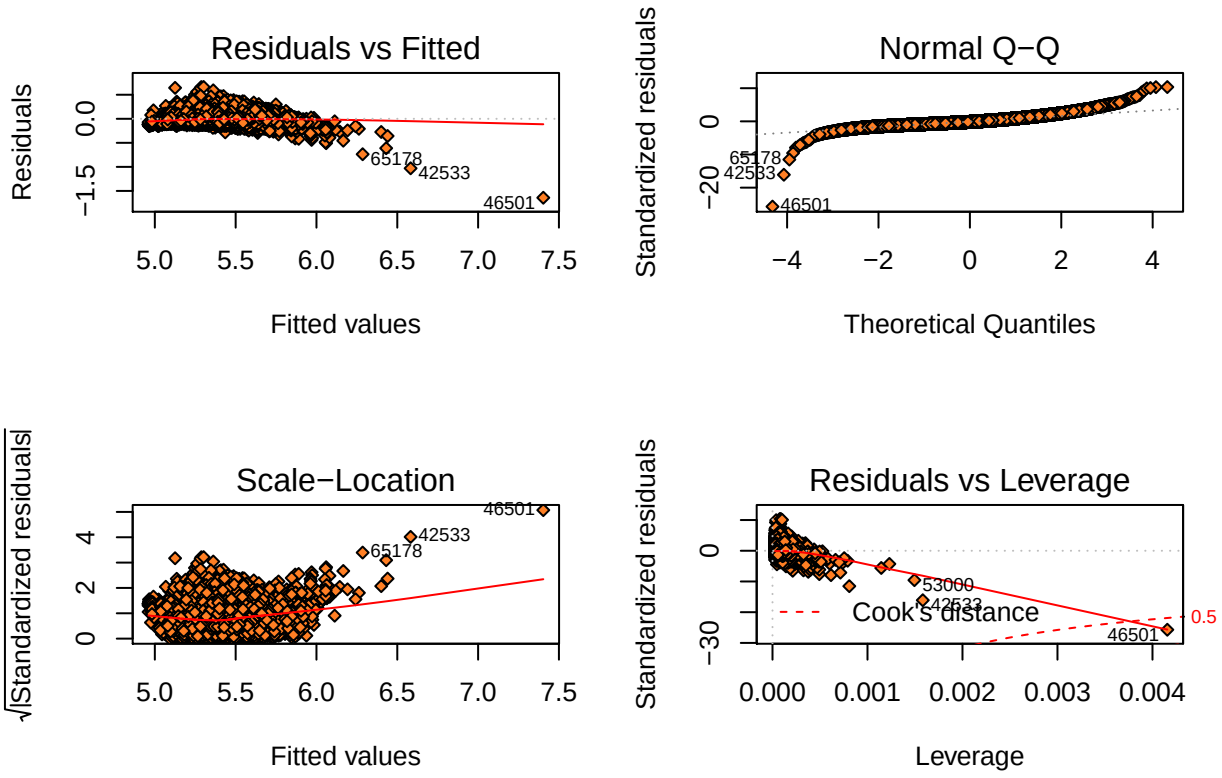

```
# That seems better.

# Let's try a multiple regression on the log transformed data:
tx.mod = lm(log(totaltime)~., data=dfm)
summary(tx.mod)
```

```
##
## Call:
## lm(formula = log(totaltime) ~ ., data = dfm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64168 -0.04097 -0.01051  0.03117  0.66161
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 4.341e+00  2.469e-03 1757.820  < 2e-16 ***
## Age         7.015e-04  2.582e-05   27.169  < 2e-16 ***
## Gender1F2M  2.568e-03  5.773e-04    4.449 8.66e-06 ***
## K0.5        4.131e-02  8.578e-05  481.571  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06397 on 64163 degrees of freedom
## Multiple R-squared:  0.8232, Adjusted R-squared:  0.8231
## F-statistic: 9.955e+04 on 3 and 64163 DF,  p-value: < 2.2e-16
```

```
score = c()
for (i in 1:k) {
  train = dfm[c(folds[i,"train1_start"]:folds[i,"train1_end"],folds[i,"train2_start"]:folds[i,"train2_en
  test = dfm[folds[i,"test_start"]:folds[i,"test_end"],]

  model = lm(log(totaltime)~.,data=train)
  score[i] = sum((test$totaltime - exp(predict(model,new=test)))^2) / as.numeric(nrow(test))
}
cat ("Log transformed mean score is: ", mean(score),"\n")
```

```
## Log transformed mean score is:  278.0591
```

```
#plot(dfms$totaltime,model.resid,ylim=c(-100,100))
#y.hat = predict(tx.mod)
#xydata = data.frame(x=y.hat, y=resid(tx.mod))
#xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
#plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="Predicted", ylab="Residuals")
par(mfrow=c(2,2))
plot(tx.mod, pch=23 ,bg="chocolate1",cex=.8)
```



This appears to be spreading out and trending up.

Let's try a polynomial transformation on the 5k time to see if that improves things:

```
#bestresidsum = 9e9
#bestpoly = 0
#for (i in seq(from=1.0, to= 2.5, by=0.1)){
#  dfms$k5xform = dfms$K0.5 ^ i
```

```
#  model = lm(totaltime~Age+Gender1F2M+k5xform, data=dfms)
#  model.residsum = sum(resid(model)^2)
# if (model.residsum < bestresidsum) {
#    bestresid = resid(model)
#    bestresidsum = model.residsum
#    bestpoly = i
#  }
#}

#cat("total error went from ",sum(model.resid^2), " (untransformed) to ", bestresidsum," (transformed)\
#plot(dfms$totaltime,bestresid,ylim=c(-100,100))
#xydata = data.frame(x=dfms$totaltime, y=bestresid)
#xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
#plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="total time(min)", ylab="Residuals")
```

Step 3: Clustering the Data into Subgroups

Let's try subsetting the data set in subgroups using an unsupervised learning algorithm.

```
# Code inspired from https://datayo.wordpress.com/2015/05/06/using-k-means-to-cluster-wine-dataset/

# Warning: using NbClust never finished even when I tried 2-4 instead of 2-15 clusters.
# install.packages("NbClust")
#library(NbClust)
#nc <- NbClust(data=dfm2,
#              min.nc=2, max.nc=4,
#              method="kmeans")
#barplot(table(nc$Best.n[1,]),
#        xlab="Numer of Clusters",
#        ylab="Number of Criteria",
#        main="Number of Clusters Chosen by 26 Criteria")


for (num_clusters in c(4,8,10,20)) {
  fit.km <- kmeans(dfm, num_clusters)
  cat ("Number of clusters:",num_clusters,"\n")
  overall = c()
  for (i in 1:num_clusters){
    df = dfm[ fit.km$cluster == i, ]
    folds = create_folds(df,k)
    score = c()
    for (j in 1:k) {
      train = df[c(folds[j,"train1_start"]:folds[j,"train1_end"],folds[j,"train2_start"]:folds[j,"train
      test = df[folds[j,"test_start"]:folds[j,"test_end"],]

      model = lm(totaltime~.,data=train)
      score[j] = sum((test$totaltime - predict(model,new=test))^2) / as.numeric(nrow(test))
    }
    cat("For cluster ",j," the mean error is ",mean(score), "\n")
    overall[i] = mean(score)
  }
  cat ("Mean error rate overall:", mean(overall),"\n\n")
```

```
}
```

```
## Number of clusters: 4
## For cluster  10  the mean error is  433.4699
## For cluster  10  the mean error is  129.0135
## For cluster  10  the mean error is  59.1684
## For cluster  10  the mean error is  47.37417
## Mean error rate overall: 167.2565
##
## Number of clusters: 8
## For cluster  10  the mean error is  37.78126
## For cluster  10  the mean error is  29.674
## For cluster  10  the mean error is  32.61125
## For cluster  10  the mean error is  30.87166
## For cluster  10  the mean error is  59.52192
## For cluster  10  the mean error is  278.3505
## For cluster  10  the mean error is  106.9132
## For cluster  10  the mean error is  62.54474
## Mean error rate overall: 79.78357
##
## Number of clusters: 10
## For cluster  10  the mean error is  38.90997
## For cluster  10  the mean error is  33.20614
## For cluster  10  the mean error is  267.8664
## For cluster  10  the mean error is  71.18586
## For cluster  10  the mean error is  25.59165
## For cluster  10  the mean error is  72.20127
## For cluster  10  the mean error is  36.55928
## For cluster  10  the mean error is  31.98628
## For cluster  10  the mean error is  102.6544
## For cluster  10  the mean error is  32.55892
## Mean error rate overall: 71.27201

## Warning: Quick-TRANSfer stage steps exceeded maximum (= 3208350)

## Number of clusters: 20
## For cluster  10  the mean error is  18.34942
## For cluster  10  the mean error is  22.18061
## For cluster  10  the mean error is  26.38441
## For cluster  10  the mean error is  13.32067
## For cluster  10  the mean error is  24.47372
## For cluster  10  the mean error is  66.6071
## For cluster  10  the mean error is  13.92566
## For cluster  10  the mean error is  80.32565
## For cluster  10  the mean error is  72.33379
## For cluster  10  the mean error is  14.53117
## For cluster  10  the mean error is  16.33678
## For cluster  10  the mean error is  27.55778
## For cluster  10  the mean error is  13.96222
## For cluster  10  the mean error is  14.36023
## For cluster  10  the mean error is  37.82077
## For cluster  10  the mean error is  185.9403
## For cluster  10  the mean error is  19.72543
```

```
## For cluster  10  the mean error is  28.60399
## For cluster  10  the mean error is  20.47039
## Mean error rate overall: 38.31808
```

```
# Warning: this takes awhile to run because it includes all data points
# install.packages("fpc")
#library(fpc)
#plotcluster(dfm2, fit.km$cluster)
```

Note: as you can see, the unsupervised learning found groupings that would have difficult if not impossible to us to find on our own

Conclusion

By transforming and subsetting the data we were able to bring down the size of the residuals significantly and improve the accuracy of the model...