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.

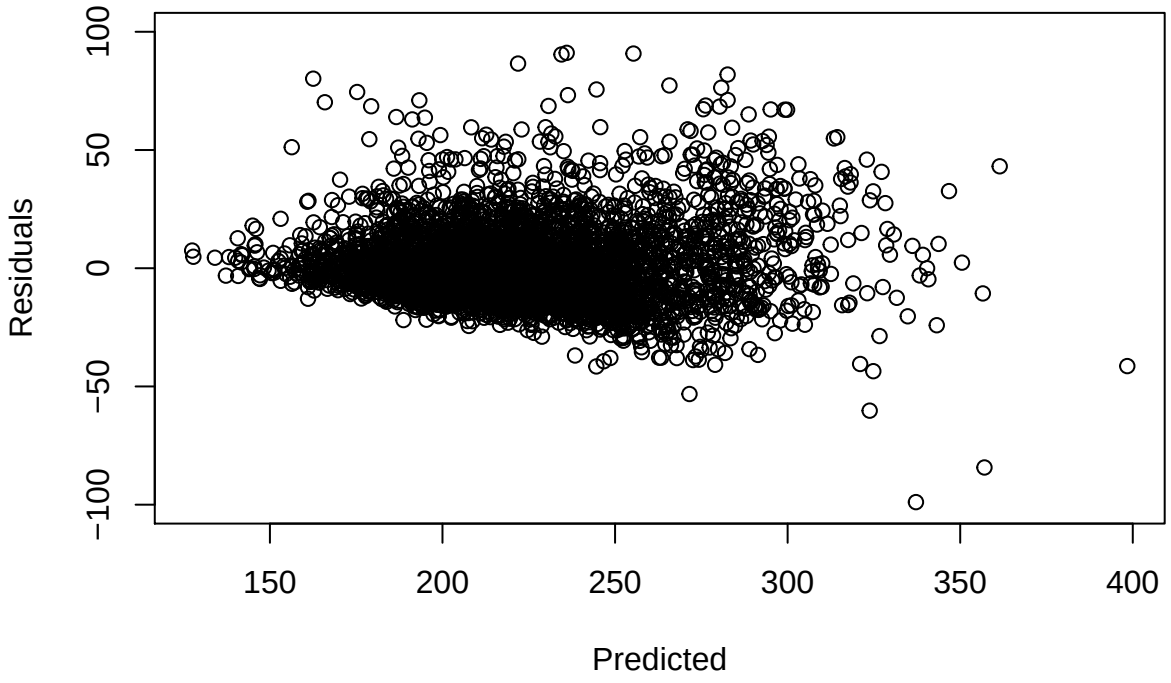
```
# First let's read in the data and calculate finish times:
dfm <- read.csv("Previous Boston Marathon study/BAA data.txt",header=T,sep=" ")
times = as.matrix(dfm[,7:15], ncol=9)
dfm$totaltime = rowSums(times)
dfm = dfm[!is.na(dfm$totaltime), ]

# Baseline regression
base.mod = lm(totaltime~Age+Gender1F2M+K0.5,data=dfm)
summary(base.mod)
```

```
##
## Call:
## lm(formula = totaltime ~ Age + Gender1F2M + K0.5, 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
```

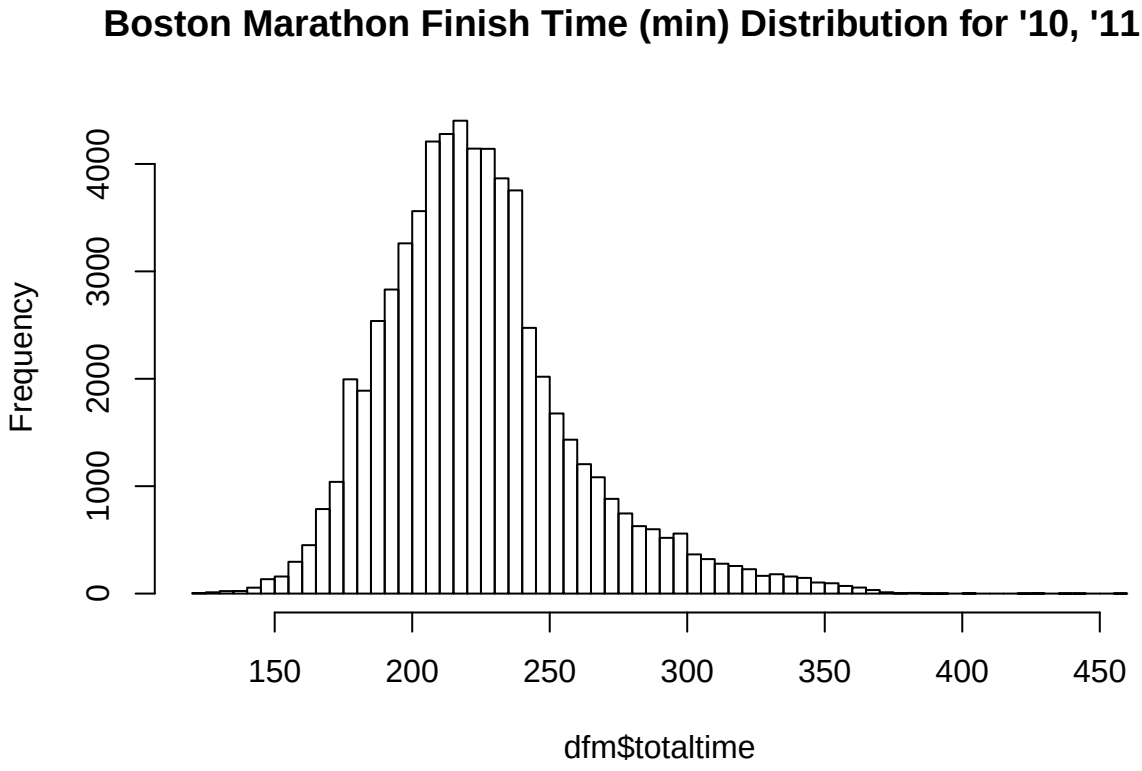
```
cat ("Baseline mean error is ",mean(base.mod$residuals^2),
    " or approximately 15 minutes.\n")
```

```
## Baseline mean error is  238.002  or approximately 15 minutes.
```

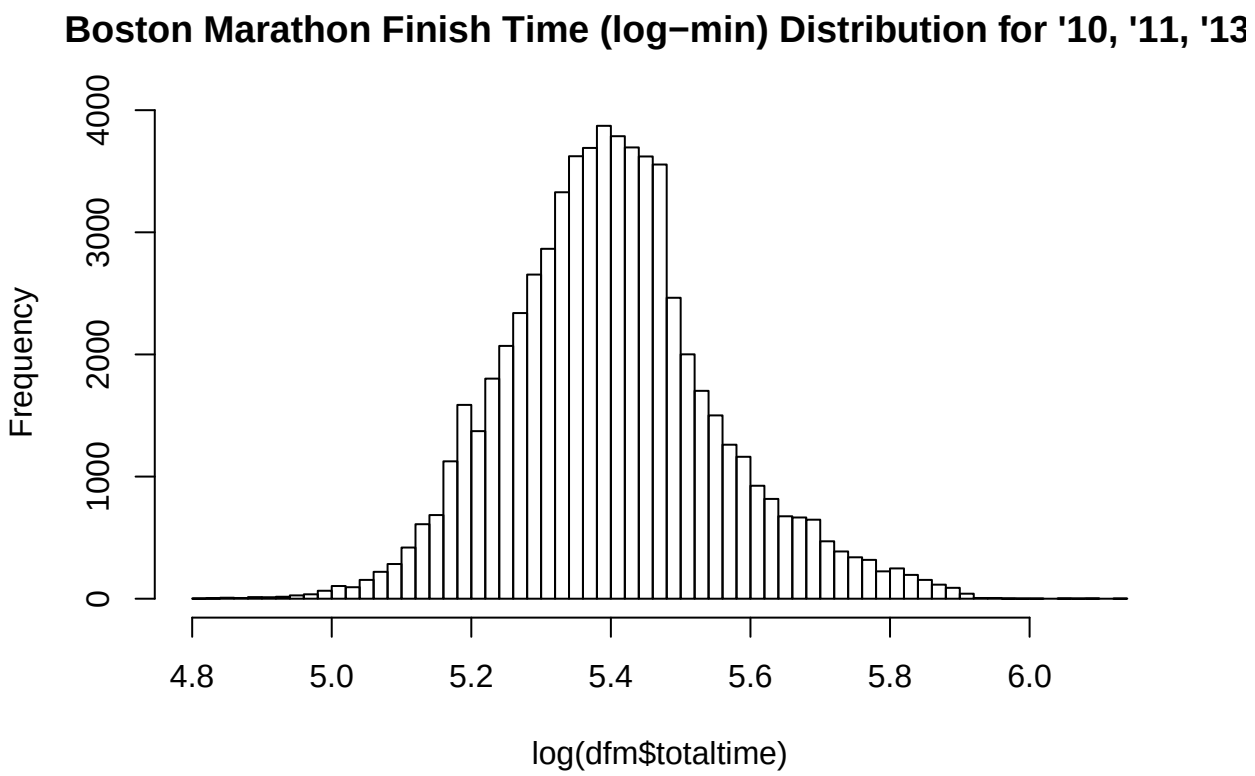
```
y.hat = predict(base.mod)
xydata = data.frame(x=y.hat, y=resid(base.mod))
xydata = xydata[sample(1:nrow(xydata), 5000, replace=FALSE),]
plot(xydata$x,xydata$y,ylim=c(-100,100), xlab="Predicted", ylab="Residuals")
```




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



```r
#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
```




```r
# That seems better.

# Let's try a multiple regression on the log transformed data:

tx.mod = lm(log(totaltime)~Age+Gender1F2M+K0.5, data=dfm)
summary(tx.mod)
```

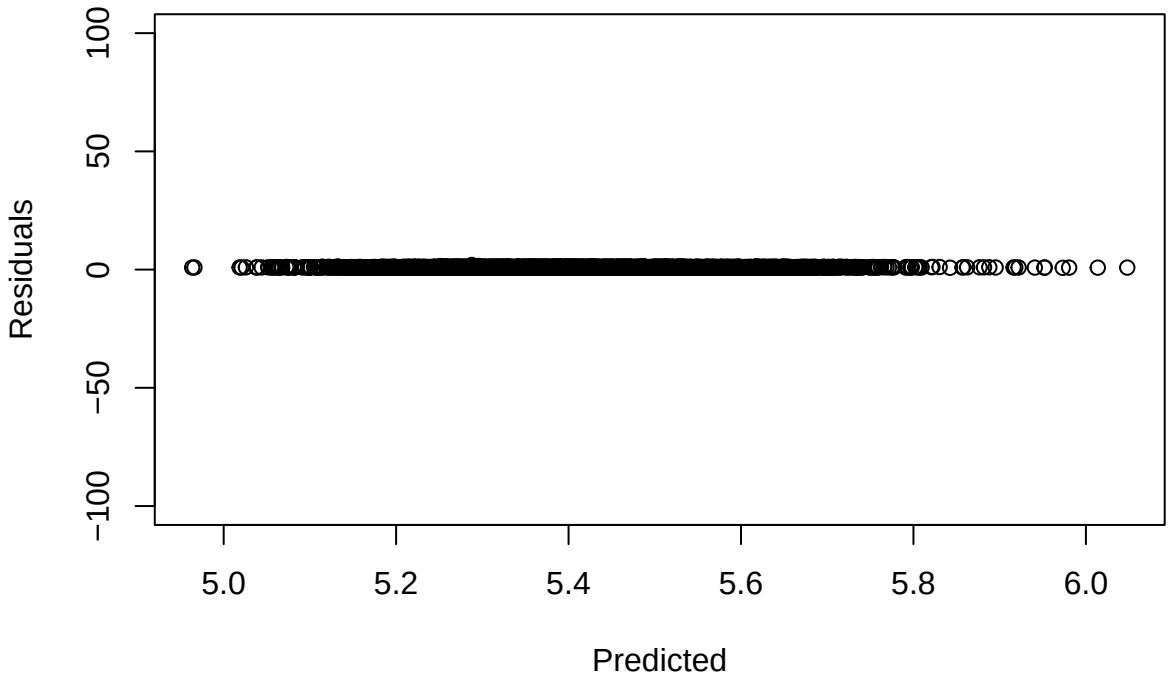
```
##
## Call:
## lm(formula = log(totaltime) ~ Age + Gender1F2M + K0.5, data = dfm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.64168 -0.04097 -0.01051  0.03117  0.66161
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 4.341e+00  2.469e-03 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
```

```
cat("Our transformed mean error is ",sqrt(mean(exp(tx.mod$residuals)^2)),"\n")
```

```
## Our transformed mean error is  1.004277
```

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



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.

Note to Pooja and Nathaniel: I'm using the untransformed data set because the transformation part is not complete. When we've completed the transformations, we'll change this to use the transformed data.

```
# Code inspired from https://datayo.wordpress.com/2015/05/06/using-k-means-to-cluster-wine-dataset/
columnstokeep <- c("totaltime","Age","Gender1F2M","K0.5")
dfm2<- dfm[columnstokeep]

# 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(dfm2, num_clusters)


  cluster.resid = c()

  cat ("Number of clusters:",num_clusters,"\n")
  for (i in 1:num_clusters){
    df = dfm2[ fit.km$cluster == i, ]
    mod = lm(totaltime~Age+Gender1F2M+K0.5,data=df)
    cluster.resid[i] = sqrt(mean(mod$residuals^2))
    cat("For cluster ",i," the mean error is ", cluster.resid[i], "\n")
  }
  cat ("Mean error rate overall:", mean(cluster.resid),"\n\n")
}
```

```
## Number of clusters: 4
## For cluster  1  the mean error is  6.638874
## For cluster  2  the mean error is  7.413076
## For cluster  3  the mean error is  19.91491
## For cluster  4  the mean error is  10.80708
## Mean error rate overall: 11.19349

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

```
## Number of clusters: 8
## For cluster  1  the mean error is  7.944135
## For cluster  2  the mean error is  16.68903
## For cluster  3  the mean error is  6.049377
## For cluster  4  the mean error is  10.52434
## For cluster  5  the mean error is  5.642851
## For cluster  6  the mean error is  4.9905
## For cluster  7  the mean error is  6.449311
## For cluster  8  the mean error is  5.482703
## Mean error rate overall: 7.971531
##
## Number of clusters: 10
## For cluster  1  the mean error is  15.63314
## For cluster  2  the mean error is  5.98107
## For cluster  3  the mean error is  5.454204
## For cluster  4  the mean error is  4.813172
## For cluster  5  the mean error is  5.817653
## For cluster  6  the mean error is  5.511561
## For cluster  7  the mean error is  9.600518
## For cluster  8  the mean error is  8.028015
## For cluster  9  the mean error is  5.451246
## For cluster  10  the mean error is  8.002856
## Mean error rate overall: 7.429344
##
## Number of clusters: 20
## For cluster  1  the mean error is  7.660649
## For cluster  2  the mean error is  6.564126
## For cluster  3  the mean error is  3.765913
## For cluster  4  the mean error is  4.481247
## For cluster  5  the mean error is  4.179111
## For cluster  6  the mean error is  8.174163
## For cluster  7  the mean error is  5.595151
## For cluster  8  the mean error is  4.236852
## For cluster  9  the mean error is  4.806284
## For cluster  10  the mean error is  8.678914
## For cluster  11  the mean error is  4.676358
## For cluster  12  the mean error is  4.250104
## For cluster  13  the mean error is  3.703455
## For cluster  14  the mean error is  5.689772
## For cluster  15  the mean error is  3.701783
## For cluster  16  the mean error is  6.232966
## For cluster  17  the mean error is  13.10157
## For cluster  18  the mean error is  4.002039
## For cluster  19  the mean error is  4.468027
## For cluster  20  the mean error is  3.761239
## Mean error rate overall: 5.586486
```

```
# 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...