Boston Marathon Finish Time Predictions

Harvard Stats E139 Fall 2015

Nathaniel Burbank
Pooja Singh
David Wihl

December 21, 2015

Abstract

We want to build a reasonably accurate model for predicting Boston Marathon finish times based on gender, age on 5k split times. We will build a baseline using untransformed linear regression. Then, we will analyze the data and perform appropriate transformations. After transforming the data, we will cluster the data into different subgroups using unsupervised learning algorithms. Finally, we will run different regression algorithms on the transformed and subsetted data to see our improvements over the baseline.

Objectives and Motivation

Boston Marathon Runners are very concerned about their finish times, moreso than many other races. Boston has a limited course size, constrainted by the 30' width of road in Hopkington, MA (source). Due to this limitation, the B.A.A. has to be very selective in the choice of runners and has established restrictive qualifying times (source). In order to run the Boston Marathon, not only does an applicant need to run a previous qualifying marathon, but must also finish much faster than the typical marathoner. While other races like New York and Chicago have increased their participation size significantly in recent years, Boston is currently and the foreseeable future restricted to approximately 27,000 runners.

Mean Finish Time(min)	Men	Women
Worldwide(*)	253	282
Boston(**)	216	235

(*) The Guardian

(**) See Data Sources, below

Boston is also known as a hilly course so a candidate who may have run a fast race on a flat course would be challenged by New England's rolling hills, especally between miles 16-21, which are among the most difficult of the course. The peak is affectionately known as Heartbreak Hill.

For all these reasons plus the long history of the race, the Boston Marathon is a very prestigious event in the running community. Forecasting an accurate finish time in the race is a key motivator for runners to improve their time during the hundreds of miles of training a marathon requires.

Data Sources

We obtained our data from two sources. First, we downloaded a previously assembled dataset from a group of researchers University of North Carolina-Chapel Hill who analyzed historic runner performance with the goal of predicting the anticpated finish times for the approximately 5000 runners who were unable to complete the 2013 Boston Marathon due the bombings at the finish line. This data set contained that split and finish

times for 64106 runners in the 2010, 2011, and 2013 Boston Marathons. (The 2012 Boston marathon was intentionally excluded from their dataset because it was unusually hot that year.) We also obtained finish and split times for 78042 runners who completed the Chicago Marathon in 2014 and 2015. We obtained these results directly from the Chicago Marathon's results website using a custom Python scraper.

Methodology

We are attempting to build the most accurate prediction model possible on the smallest amount of interesting data. We are not attempting to infer or analyze the data extensively. Our method consists of the following steps:

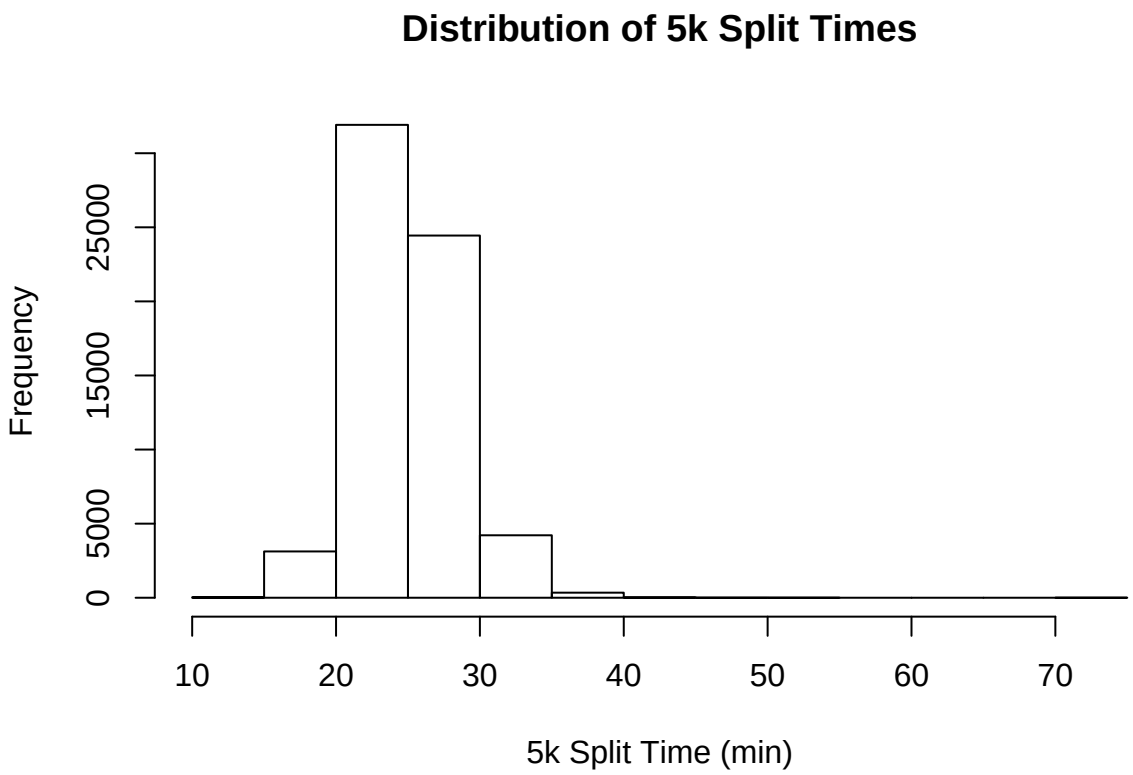
1. Data Cleansing
2. Create a baseline regression using OLS
3. Analyze the data and perform appropriate transformations
4. Cluster the data into subgroups using unsupervised learning, such as k-means
5. Verify the improved accuracy of our predictions.

Throughout we will be using 10-fold cross validation testing with sum of squares errors as the key quality metric.

Data Cleansing

Even though we downloaded the data from another study that had performed some scrubbing, we needed to perform additional data cleansing. We had to eliminate any rows that were missing any split times as imputation might have tainted the predictive model. Since we were using k-Fold cross-validation, we randomized the order of the data to ensure that each fold had a reasonable sample. Finally, gender had a value of $\{1, 2\}$ which we had to convert into a factor in R.

Removing Outliers



As the histogram reveals, there are several outliers in 5k split times. These could be due to timing errors, an injury or accident at the start of the race. Due to OLS sensitivity, we elected to remove outliers that were more than three standard deviations greater than the mean. In total there were 416 rows like this out of 64106 (0.6%).

Age was in the range [18,81]. (The minimum age to run the Boston Marathon is 18). No outliers were removed. Males represented 58.7% of total runners, which seemed reasonable.

Baseline Regression

We ran a simple OLS regression using all the Boston data. The predictor variables were age, gender (as factor), and 5k split time. The response variable is the finish time in minutes.

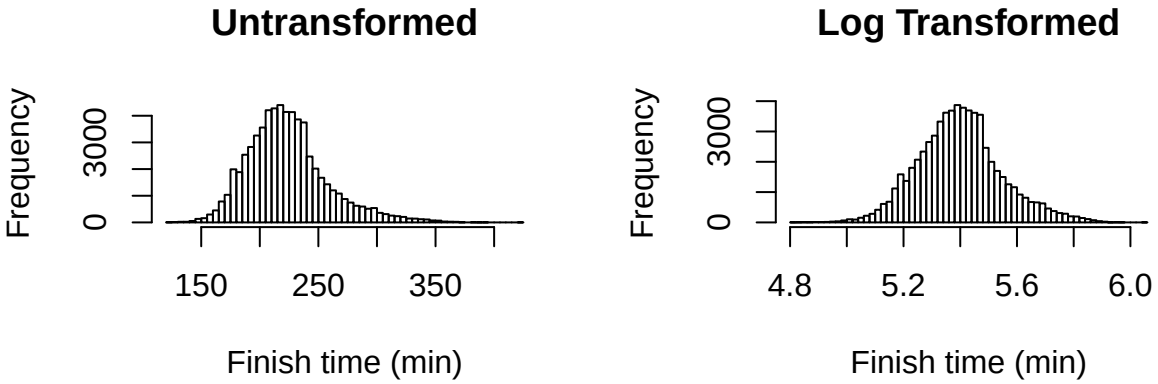
The baseline model was quite good, but we felt we could do better.

```
##
## Call:
## lm(formula = totaltime ~ ., data = dfm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -79.265  -9.239  -2.611   6.443 187.552
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -24.32412    0.54076 -44.981   <2e-16 ***
## Age           0.06106    0.00619   9.864   <2e-16 ***
```

```
## Gender1F2M2   3.16388    0.13832  22.874   <2e-16 ***
## K0.5          9.82057    0.02154 455.856   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 15.17 on 63686 degrees of freedom
## Multiple R-squared:  0.8036, Adjusted R-squared:  0.8036
## F-statistic: 8.688e+04 on 3 and 63686 DF,  p-value: < 2.2e-16
```

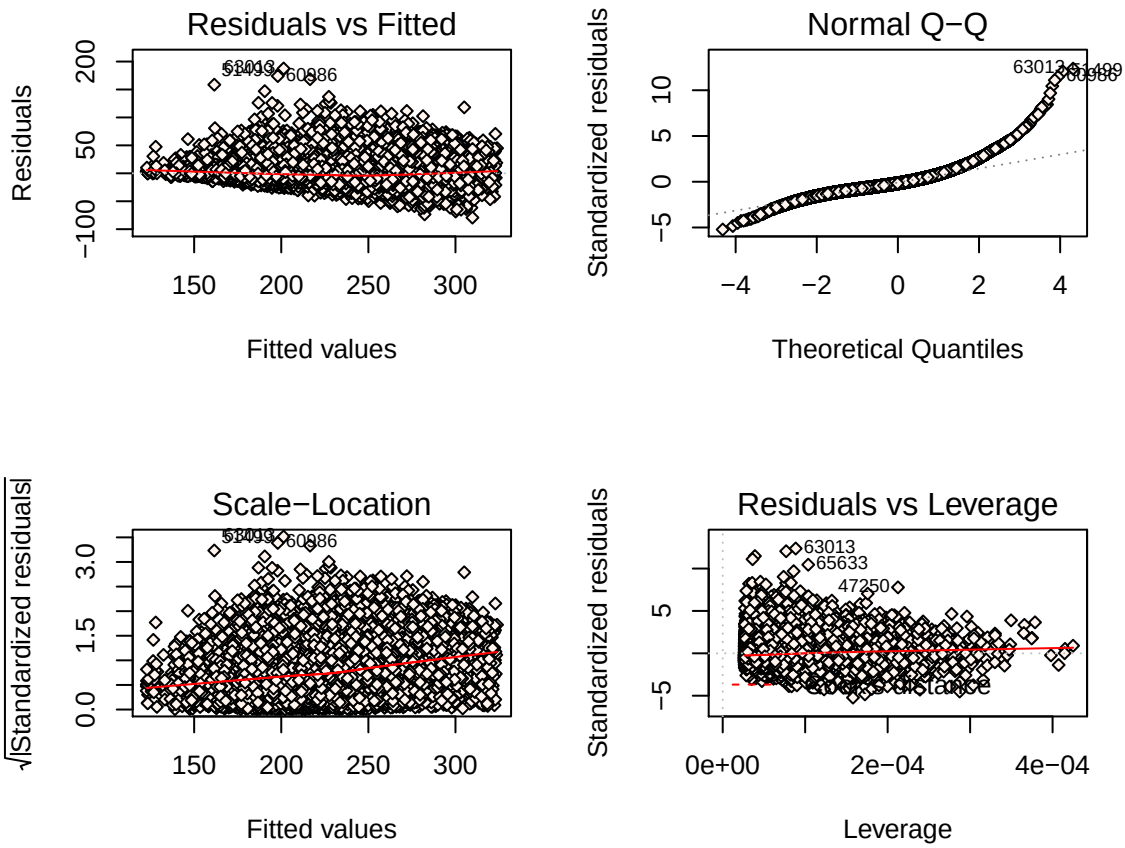
Using k-Fold cross validation, and taking the sum of squares error for each fold resulted in a baseline mean score of 230 or approximately 15 minutes. While $\pm$ 15 minutes over the course of a four hour race may not seem like a large error, it makes a significant difference in terms of ability to qualify for Boston.

Examination of the Data and Transformations



The untransformed data appears slightly right skewed. A log transformation of the response variable would result in a more normal distribution.

We then examined the base model regression diagnostics such as residuals vs. predicted, Cook's Distance, heteroskedasticity.



Given the right tail distribution of the QQplot, the fan out of the residuals vs. predicted, and the slight right skewness of the finish time histogram, we elected to perform a log transformation of the response variable.

Using k-Fold cross validation, and taking the sum of squares error for each fold resulted in a log-transformed mean score of 227 or approximately 15 minutes, a neglible improvement.

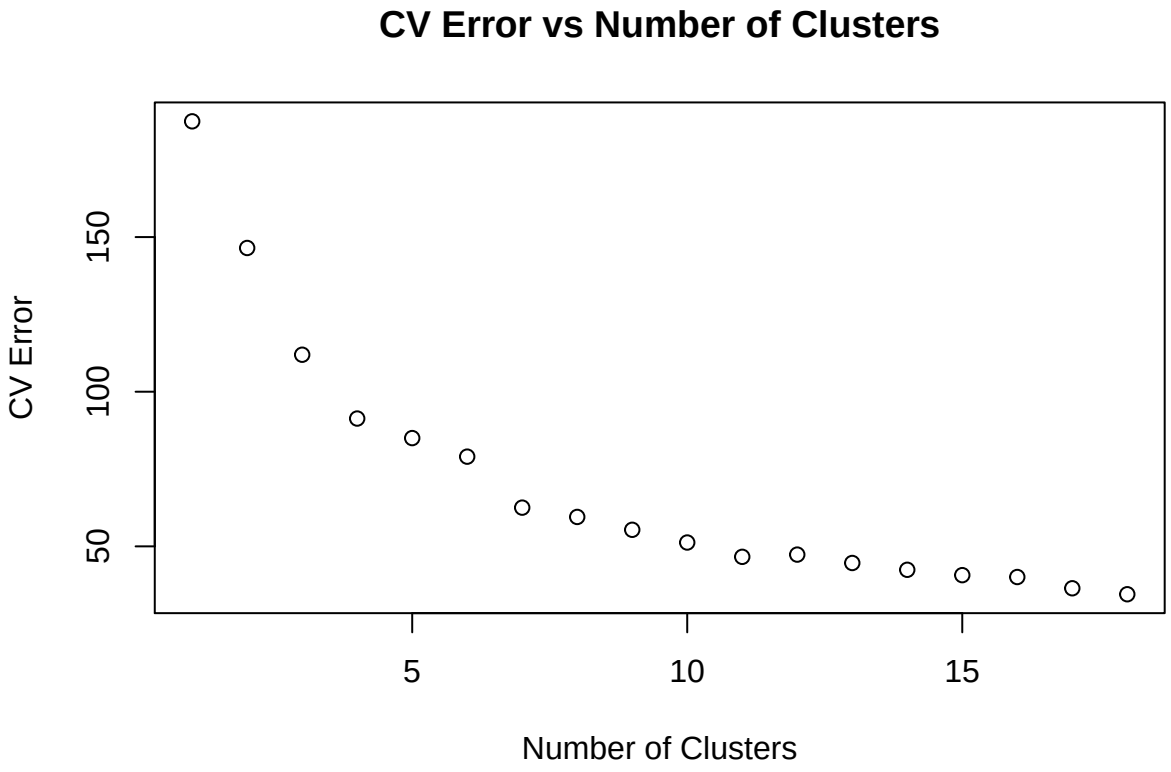
R^2 went from 0.8036325 to 0.8224044, a slight improvement. Adjusted R^2 improved by an equally small amount.

We also attempted a range of polynominal transformations of the 5K time using an exponent in the range [1.0,2.5] but this did not improve model accuracy significantly. For report brevity, these results have been omitted but are available upon request (and can be found in the project GitHub site - See Appendix B).

Clustering the Data into Subgroups

Since the transformation yielded only a small improvement, we elected to try clustering the data into subgroups using k-Means. We attempted $k = [3, 20]$.

By clustering, the resulting error rates decreased significantly. Using $k = 3$, the mean error rate was 187 or $\pm$ 13 min. At $k = 20$, the mean error rate was 34 or $\pm$ 5 min, a significant improvement.



Using the elbow method, we determined that the optimal number of clusters is $k = 8$, which results in a cross validation error of 79 or $\pm$ 8 min, which is approximately half the error as compared to unclustered linear regression.

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

Conclusion and Next Steps

By subsetting the data, we were able to bring down the cross validation error significantly and improve the accuracy of the model.

Potential next steps of this project include:

- Attempt other clustering models besides k-Means (example).
- Attempt other algorithms of regression, such as Random Forests.
- Making the prediction model available on a public website using R/Shiny. We fill that our prediction model is far more accurate than what is currently available to runners.

Appendix A - Code

The following is the code used to prepare this report:

```
# Boston Marathon Finish Time Predictions
# Harvard Stats E139 Fall 2015
# Nathaniel Burbank, Pooja Singh, David Wihl
#
# December 21, 2015
```

```
# Common Functions


percent <- function(x, digits = 1, format = "f", ...) {
  paste0(formatC(100 * x, format = format, digits = digits, ...), "%")
}

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 = 10
  }

  train1_start = train1_end = train2_start = train2_end = test_start = test_end = c()
  testsize = nrow(df) / k
  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)
    }
  }
  return (data.frame(train1_start, train1_end, train2_start, train2_end, test_start, test_end))
}

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

# Read in the data for multiple years
dfm <- read.csv("Previous Boston Marathon study/BAA data.txt",header=T,sep=" ")
dfm$Age2014 = NULL # remove unneeded column which is mostly NA
ok = complete.cases(dfm)
dfm = dfm[ok,] # remove rows that have any NA values
times = as.matrix(dfm[,7:15], ncol=9)
dfm$totaltime = rowSums(times)
dfm<- dfm[c("totaltime","Age","Gender1F2M","K0.5")] # keep only columns we need
dfm$Gender1F2M = as.factor(dfm$Gender1F2M) # make gender into a factor
dfm = dfm[!is.na(dfm$totaltime), ]  # eliminate rows with no finish times
dfm = dfm[sample(nrow(dfm)),]  # in case the data is sorted, randomize the order
# Find mean finish time by gender
agg = aggregate(dfm$totaltime, by=list(dfm$Gender1F2M), FUN=mean)[2]
```

```
men = as.integer(agg$x[2])
women = as.integer(agg$x[1])
# Remove outliers (5k > 3 sigma from mean)
hist (dfm$K0.5, breaks=15, main="Distribution of 5k Split Times", xlab="5k Split Time (min)")
mean5k = mean(dfm$K0.5)
sd5k = sd(dfm$K0.5)
outliers5k = dfm$K0.5>(mean5k + 3*sd5k)
outliers5ksum = sum(outliers5k)
dfm.rows = nrow(dfm)
dfm = dfm[!outliers5k,]
malepct = table(dfm$Gender1F2M)[2]  / (table(dfm$Gender1F2M)[2] + table(dfm$Gender1F2M)[1])
# 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)

# 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"],]

  cvmodel = lm(totaltime~.,data=train)
  score[i] = sum((test$totaltime - predict(cvmodel,new=test))^2) / as.numeric(nrow(test))
}
score.mean = mean(score)
summary(base.mod)
# Display histograms of response variable untransformed and log transformed
par(mfrow=c(1,2))
hist(dfm$totaltime,breaks=50, main="Untransformed", xlab="Finish time (min)")
hist(log(dfm$totaltime),breaks=50, main="Log Transformed", xlab="Finish time (min)")
par(mfrow=c(2,2))
plot(base.mod, pch=23 ,bg="seashell",cex=.8)
# Try regression on log(response)
dfm$logtotaltime = log(dfm$totaltime)
folds = create_folds(dfm,k)
score = c()

tx.mod = lm(logtotaltime~.-totaltime,data=dfm)

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"],]

  cv.tx.mod = lm(logtotaltime~.-totaltime,data=train)
  score[i] = sum((test$totaltime - exp(predict(cv.tx.mod,new=test)))^2) / as.numeric(nrow(test))
}
score.mean = mean(score)

# Clustering the Data into Subgroups
# Let's try subsetting the data set in subgroups using an unsupervised learning algorithm.
```

```r
minclusters = 3
maxclusters = 20
overallclustererror = c()
for (num_clusters in minclusters:maxclusters) {
  fit.km <- kmeans(dfm, num_clusters)
  meanclustererror = c()
  # Find cv score for each cluster
  for (i in 1:num_clusters){
    df = dfm[ fit.km$cluster == i, ]
    folds = create_folds(df,k)
    foldscore = 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~.-logtotaltime,data=train)
      foldscore[j] = sum((test$totaltime - predict(model,new=test))^2) / as.numeric(nrow(test))
    }
    meanclustererror[i] = mean(foldscore)
  }
  # store mean error for a given k clusters
  overallclustererror[num_clusters-2] = mean(meanclustererror)
}
# TODO fix x axis of plot because it is off by -2
plot(overallclustererror, xlab="Number of Clusters", ylab="CV Error", main="CV Error vs Number of Cluste
# Show the resulting clusters
#fit.km <- kmeans(dfm, 8)
# install.packages("fpc")
#library(fpc)
#plotcluster(dfm, fit.km$cluster)
##
```

Appendix B - References

This entire project code including source data can be found on GitHub (https://github.com/wihl/statse139-project)

K-Means Clustering in R Sample Code

R Packages Used

fpc used to plot clusters

Solutions Found on StackOverflow

Place R Code in Appendix

Takes mean by subgroup in R

Formatting R Values as Percent