First Assignment

STATISTICAL MODELLING AND DESIGN OF EXPERIMENTS

Ricard Meyerhofer Parra

5/10/2018

Contents

1 FIRST QUESTION: GENERATE A RANDOM SAMPLE

The generator we are going to use is Mersenne Twister random generator with one of the parametrations provided in the SMDE website that pass the Diehard Battery of Tests of Randomness:

```
//This parametrization corresponds to the first one
unsigned long init[LENGTH]={
0x631f1690, 0x25973e32, 0xead6f57, 0x6ec9d844, 0x5c49eaee,
0x64af49b, 0x397c46bc, 0x7e448de9, 0x5a9cc3e5, 0x1afe3625,
0x3ca88ecf, 0x6ebe4208, 0xc058df5, 0xcbe5be9, 0x3102bbe2,
0x26a02c5e, 0x541c8153, 0x67906f60, 0x10db9daa, 0x697d2d2,
0x6d68ab2, 0x3a966cd0, 0x63f37e85, 0x5895f5fa, 0x38a5d054};
```

Once executed the algorithm, we are going to save the output as a ".csv" and we will load it in R so that we can start manipulating the data. We are going to analyze 3 samples that correspond to different executions (different sizes) with the same parametrization:

- Sample of 200 elements named "samples.csv".
- Sample of 3000 elements named "samples3000.csv".
- Sample of 20000 elements named "samples20000.csv".

For all of these samples we will follow the same procedure and see if the size affects our results.

```
#Loading data and setting environment:
setwd("C:/Users/Meyerhofer/Desktop/SMDELabs")
mersenne_twister <- read.csv("samples.csv", sep = "\t", header = F)
```

Once the data is loaded, we classify it in 6 segments acording to it's position and we see the frequency in which appears to each segment:

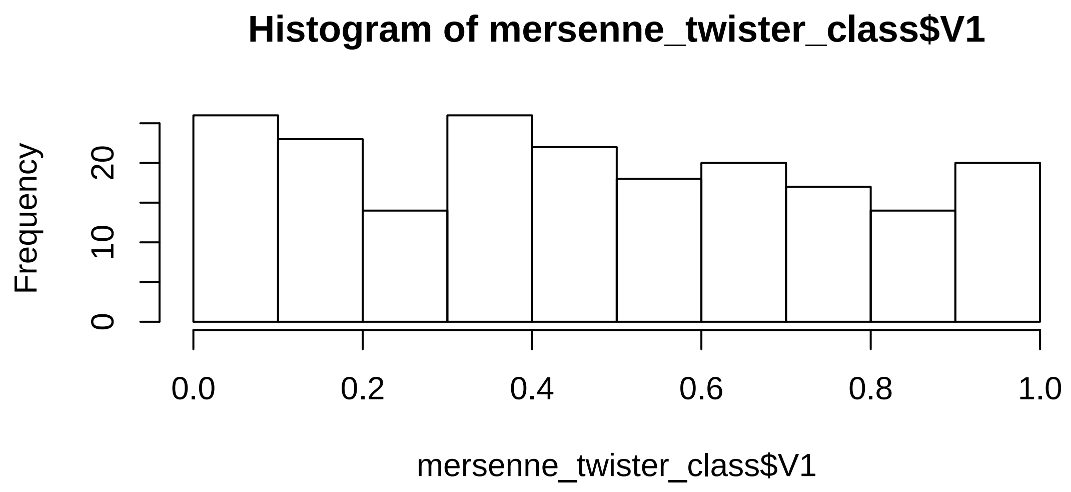
```
mersenne_twister_class = transform(mersenne_twister, cat2 = ifelse(V1 < 0,"0",
                                                                 ifelse(V1 < 0.2,"0.2",
                                                                 ifelse(V1 < 0.4,"0.4",
                                                                 ifelse(V1 < 0.6,"0.6",
                                                                 ifelse(V1 < 0.8,"0.8","1"))))))
mersenne_freqs = as.data.frame(with(mersenne_twister_class, table(cat2)))
```

Similarly, we create another data sample with a uniform distribution which as Mersenne's Algorithm has values between 0 and 1. This dataset will also be segmented according the same labels we have previously used for Mersenne. Note that to make the sample reproducible, we are using an arbitrary seed.

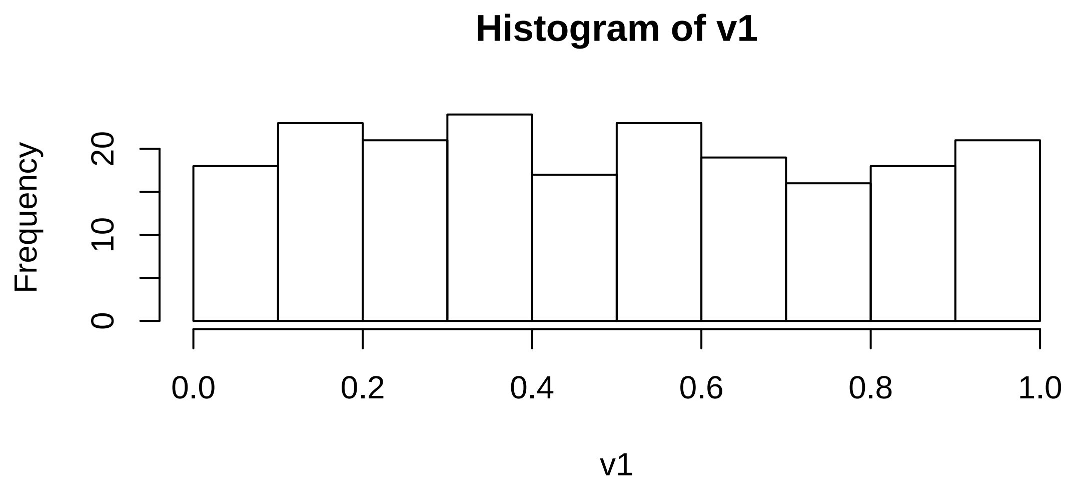
```
set.seed(1234)
v1=runif(200, min = 0, max = 1)
#Work with the data as a dataframe.
taula_v1=data.frame(x1=v1)

#Definition of the intervals, categories to be used.
taula_v1_cat=transform(taula_v1, cat1 = ifelse(x1 < 0,"0",
                                        ifelse(x1 < 0.2,"0.2",
                                        ifelse(x1 < 0.4,"0.4",
                                        ifelse(x1 < 0.6,"0.6",
                                        ifelse(x1 < 0.8,"0.8","1"))))))
```

Are these two samples similar? We can have a first idea by comparing both histograms and how the data is distributed among each execution.

```
hist(mersenne_twister_class$V1)
```



```r
hist(v1)
```



Finally we aggregate both frequencies in a same dataframe. Complementing our histogram's we do two barplots that are aggregating the data in the segments.

- The first one, is a barplot of each of the functions (so that we see how a function is distributing from [0-1]).
- The second is each of the segments compared one to each other in the same plot (so we see both functions behave the same).

```r
#Counting the amount of elements in each category "table" function.
taula_freq_v1=as.data.frame(with(taula_v1_cat, table(cat1)))

taula_freq = rbind(taula_freq_v1$Freq, mersenne_freqs$Freq)

aux = cbind(taula_freq_v1$Freq, mersenne_freqs$Freq)
```

We also provide of the frequencies to see which is the data that is being treated in the barplots:

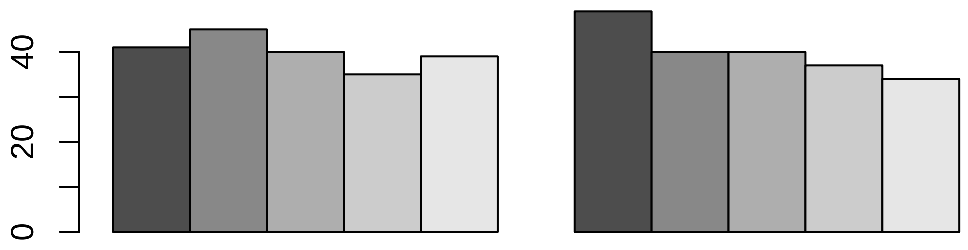
```r
taula_freq_v1
```

```
##   cat1 Freq
## 1  0.2   41
## 2  0.4   45
## 3  0.6   40
## 4  0.8   35
## 5    1   39
```

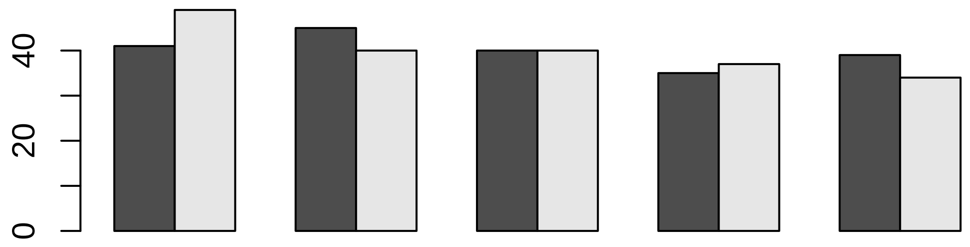
```r
mersenne_freqs
```

```
##   cat2 Freq
## 1  0.2   49
## 2  0.4   40
## 3  0.6   40
## 4  0.8   37
## 5    1   34
```

```r
barplot(aux, beside=T)
```



```r
barplot(taula_freq,beside=T)
```



As we can see, the distribution among the different segments is not what one should expect since we would expect a complete uniformity (all segments with same or almost same number of elements which is not happening with 200 elements). We will see if in the next samples if this shape keeps appearing or if it dissapears and tends to uniformity.

Finally we are going to see if both samples are similar by applying a Chi-Square test:

```
Test=chisq.test(taula_freq, correct = FALSE)
Test
```

```
##
##  Pearson's Chi-squared test
##
## data:  taula_freq
## X-squared = 1.4033, df = 4, p-value = 0.8436
```

Chi-square Test. The Chi-Square test of Independence is used to determine if there is a significant relationship between two nominal (categorical) variables. When we apply a Chi-Square test we are formulating the following:

- Ho: The data is consistent with a specified distribution.
- Ha: The data is not consistent with a specified distribution.

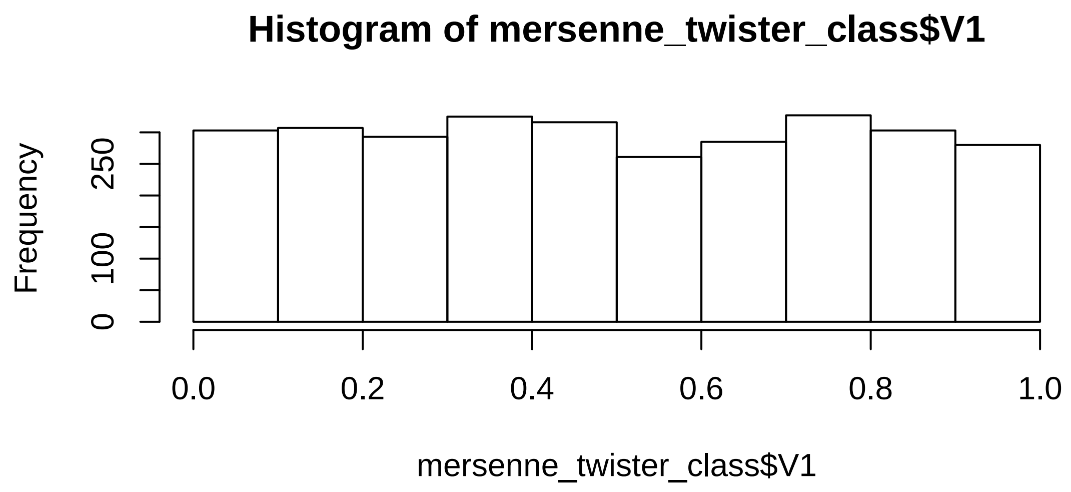
As the p-value is greater than the 0.05 significance level, we do not reject the null hypothesis therefore, there is relationship between mersenne and uniform distribution (which is expectable at first sight).

As we previosly said, the data was not as balanced as one would expect. Because of this we are going to generate two more executions with the same parametrization:

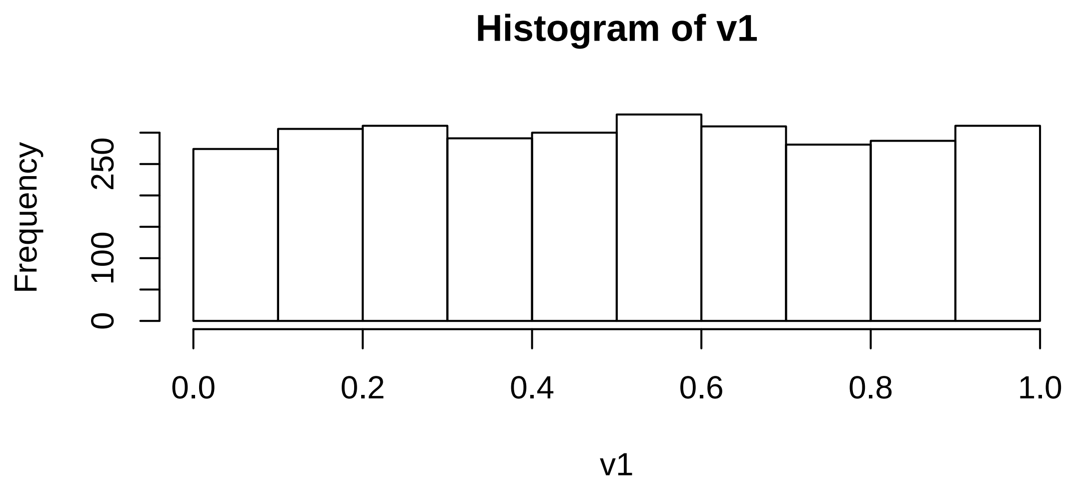
- One with 3000 elements
- Another with 20000.

Here we can see that in comparasion with the samples of 200 elements, the segments tend to be equal and the more elements that have, the more equal are.

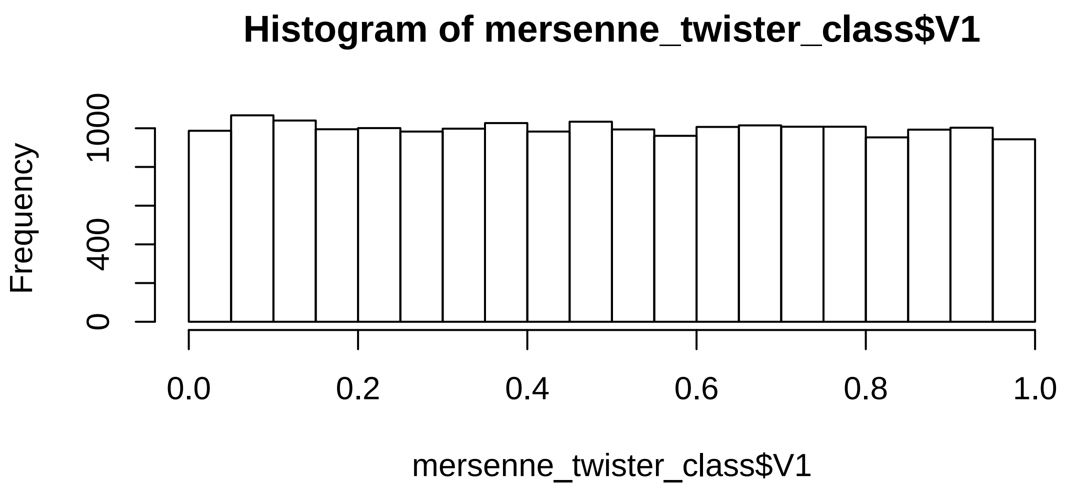
```
hist(mersenne_twister_class$V1)
```



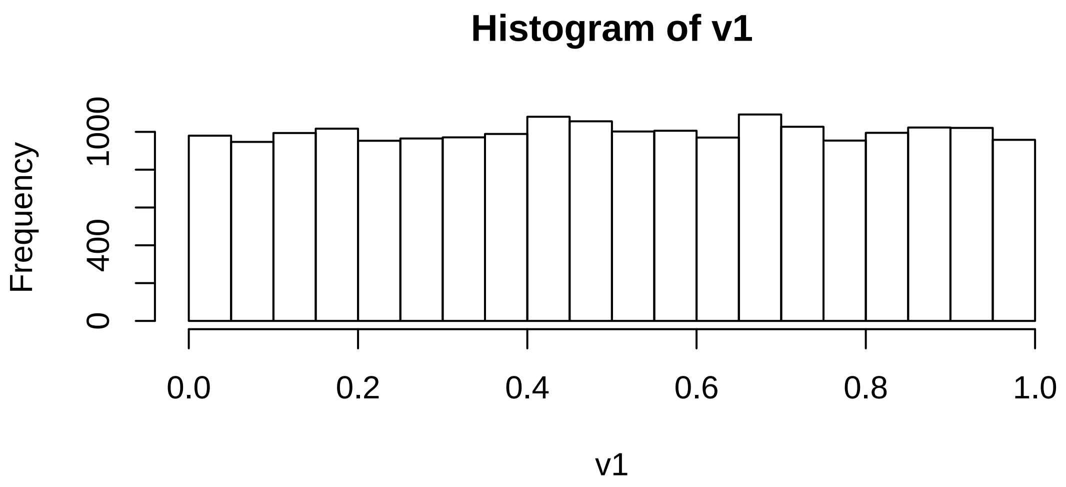
```r
hist(v1)
```



```r
hist(mersenne_twister_class$V1)
```



```r
hist(v1)
```



As a last comment I would like to make note that by doing a Chi-Square test we are not comparing the quality of the RNG or the RVA but what we are doing is to given a RNG that passes the DieHard tests, see if the runif distribution is similar to this RNG so somehow we are comparing if is as good as the Mersenne Twister but not necessarily has to be even that both look alike.

2 SECOND QUESTION: ANOVA

First we are generating three populations and we are going to change one of the parameters and see with the ANOVA if these three populations are different (or not) depending on the parameter selected. In this case, we are going to have three populations following a normal distribution two with media=0 and one with media=10.

We create the populations and we merge them in a single data frame in order to use ANOVA.

Now we can see with the ANOVA if these 3 populations are different:

We can conclude from the ANOVA test that since the p-value is < than 0.05, we can reject the null hypothesis and conclude that these distributions have different means.

Now we are going to proceed with testing the ANOVA assumptions which are the following:

- **Durbin Watson:** The observations within each sample must be independent.
- **Shapiro test:** The populations from which the samples are selected must be normal.
- **Breusch Pagan test:** The populations from which the samples are selected must have equal variances (homogeneity of variance)

2.1 Independence

Even that we already know that this data is independent from each other since we created them, we have to test it with Durbin Watson test.

```
library("lmtest")
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.9929, p-value = 0.866
## alternative hypothesis: true autocorrelation is not 0
```

We can see that the p-value < DL. Therefore,there is no correlation, which implies independece. (Passed the test)

2.2 Normality

We apply the Saphiro test:

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.99772, p-value = 0.5982
```

The p-value > 0.05, so we can say that the sample follows a normal distribution (Passed the test).

2.3 Homogeneity

We are going to apply the Breusch Pagan test:

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.075827, df = 2, p-value = 0.9628
```

Two or more normal distributions are homoscedastic if they share a common covariance (or are correlated). A p-value > 0.05 indicates there is homogenety (Passed the test).

2.4 Decathlon

In this decathlon dataset, we want to analyze if both competitions achieve different results (we want to see if they are different) in the different disciplines analyzed.

```
# Loading decathlon dataset.
data(decathlon, package="FactoMineR")
names(decathlon)
```

```
##  [1] "100m"        "Long.jump"   "Shot.put"    "High.jump"   "400m"
##  [6] "110m.hurdle" "Discus"      "Pole.vault"  "Javeline"    "1500m"
## [11] "Rank"        "Points"      "Competition"
```

In order to see if they achieve different results, what we are going to do is to apply the assumptions and anova per each of the variables that contains the dataset. In order not to make it very harsh to follow I'm going to make it as synthetic as possible.

2.4.1 100M

```
data = data.frame(x1=decathlon$`100m`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## x2           1 0.5986  0.5986   10.77 0.00218 **
## Residuals   39 2.1686  0.0556
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.2595, p-value = 0.008235
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.97525, p-value = 0.5023
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.040711, df = 1, p-value = 0.8401
```

The Durbin-Watson test has a p-value < 0.5 so we can't assume independency. The p-value that Anova gives us is < 0.05 which means that we reject null hypothesis (the distributions are different). We cannot guarantee that the Anova test gives us a right result even that Durbin's assumption is not correct.

2.4.2 Long Jump

```r
data = data.frame(x1=decathlon$Long.jump, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1  0.003 0.00288   0.028  0.868
## Residuals   39  4.002 0.10260
```

```r
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.3741, p-value = 0.02544
## alternative hypothesis: true autocorrelation is not 0
```

```r
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.98831, p-value = 0.9437
```

```r
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 1.2574, df = 1, p-value = 0.2621
```

The Durbin-Watson test has a p-value < 0.5 so we can't assume independency. The p-value that Anova gives us is > 0.05, it means that we accept null hypothesis (the distributions are equal). We cannot guarantee that the Anova test gives us a right result even that Durbin's assumption is not correct.

2.4.3 Shot Put

```r
data = data.frame(x1=decathlon$Shot.put, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1  1.932  1.9324   2.984  0.092 .
## Residuals   39 25.255  0.6476
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.4425, p-value = 0.04577
## alternative hypothesis: true autocorrelation is not 0
```

```r
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.9726, p-value = 0.4175
```

```r
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 1.2645, df = 1, p-value = 0.2608
```

All assumptions have a p-value < 0.05. The p-value that Anova gives is > 0.05, it means that we accept null hypothesis(distributions are equal). We can not say that Anova result is confident since all assumptions are incorrect.

2.4.4 High Jump

```r
data = data.frame(x1=decathlon$High.jump, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq  Mean Sq F value Pr(>F)
## x2           1 0.0000 0.000014   0.002  0.967
## Residuals   39 0.3165 0.008115
```

```r
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 2.1012, p-value = 0.8665
## alternative hypothesis: true autocorrelation is not 0
```

```r
shapiro.test(residuals(AnovaModel.1))
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.93672, p-value = 0.02428
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.0097937, df = 1, p-value = 0.9212
```

Normality test has p-value < 0.05 so we cannot guarantee normality. The p-value that Anova gives us is >0.05 so we accept null hypothesis (equal distributions). We can not assume that the Anova test is not correct even Saphiro test is wrong.

2.4.5 400M

```
data = data.frame(x1=decathlon$`400m`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1   0.00  0.0036   0.003   0.96
## Residuals   39  53.21  1.3645
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.3938, p-value = 0.03032
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.95672, p-value = 0.1206
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 1.0517, df = 1, p-value = 0.3051
```

The Durbin-Watson test has a p-value < 0.5 so we can't assume independency. The p-value that Anova gives us is > 0.05, it means that we accept null hypothesis (the distributions are equal). We cannot guarantee that the Anova test gives us a right result even that Durbin's assumption is not correct.

2.4.6 110M Hurdle

```
data = data.frame(x1=decathlon$`110m.hurdle`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1  0.241  0.2414   1.087  0.304
## Residuals   39  8.662  0.2221
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.5083, p-value = 0.07622
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.93428, p-value = 0.02009
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 1.1326, df = 1, p-value = 0.2872
```

Normality test has p-value < 0.05 so we cannot guarantee normality. The p-value that Anova gives us is >0.05 so we accept null hypothesis (equal distributions). We can not assume that the Anova test is not correct even Saphiro test is wrong.

2.4.7 Discus

```
data = data.frame(x1=decathlon$`Discus`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1    0.2   0.222   0.019  0.891
## Residuals   39  456.2  11.697
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.342, p-value = 0.01889
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.97003, p-value = 0.3455
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.21492, df = 1, p-value = 0.6429
```

Durbin-Watson has a p-value < 0.05 we can not assume independency. The p value that Anova gives us is $>$ 0.05 so these distributions are equal since we accept the null hypothesis but this does not guarantee that Anova gives us the right result.

2.4.8 Pole Vault

```
data = data.frame(x1=decathlon$`Pole.vault`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1 0.0811 0.08105    1.05  0.312
## Residuals   39 3.0103 0.07719
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 2.0017, p-value = 0.8782
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.97879, p-value = 0.6299
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.50795, df = 1, p-value = 0.476
```

Assumptions are correct therefore, we can say that the distributions are equal.

2.4.9 Javeline

```
data = data.frame(x1=decathlon$`Javeline`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1   35.3   35.31   1.536  0.223
## Residuals   39  896.6   22.99
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 2.1099, p-value = 0.8442
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.97474, p-value = 0.4852
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.41184, df = 1, p-value = 0.521
```

Assumptions are correct therefore, we can say that the distributions are equal.

2.4.10 1500M

```
data = data.frame(x1=decathlon$`1500m`, x2=decathlon$Competition)
AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## x2           1    192   191.9   1.423   0.24
## Residuals   39   5259   134.8
```

```
#Assumptions
dwtest(AnovaModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  AnovaModel.1
## DW = 1.7303, p-value = 0.3011
## alternative hypothesis: true autocorrelation is not 0
```

```
shapiro.test(residuals(AnovaModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(AnovaModel.1)
## W = 0.93826, p-value = 0.02741
```

```
lmtest::bptest(AnovaModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  AnovaModel.1
## BP = 0.030018, df = 1, p-value = 0.8624
```

Normality test has p-value < 0.05 so we cannot guarantee normality. The p-value that Anova gives us is >0.05 so we accept null hypothesis (equal distributions). We can not assume that the Anova test is not correct even Saphiro test is wrong.

2.4.11 Rank & Points & Competition

Not variables to measure.

3 THIRD QUESTION: DEFINE A LINEAR MODEL FOR AN ATHLETE IN THE 1500M

First we are going to load our data:

```r
library(Rcmdr)
data(decathlon, package="FactoMineR")

train <- subset(decathlon, Competition == "Decastar")
test <- subset(decathlon, Competition == "OlympicG")
```

One very naive way to approach this problem, would be to generate a model with all the variables and to “backtrack” throught it by removing variables until the model is very good and seeing with ANOVA if two models are same or not (and ironically, without having any idea why is good). But as a reference, can be beneficial to see which is the predict ratio with all the variables.

```r
train <- subset(decathlon, Competition == "Decastar")
test <- subset(decathlon, Competition == "OlympicG")
train <- na.omit(train)
RegModel.1 <- lm(`1500m` ~ `100m` + `Long.jump` + `Shot.put` + `High.jump` + `400m` +
                   `110m.hurdle` + `Discus` + `Pole.vault` + `Javeline` +`Rank`+ `Points`,
                 data=train)

summary(RegModel.1)
```

```
##
## Call:
## lm(formula = `1500m` ~ `100m` + Long.jump + Shot.put + High.jump +
##     `400m` + `110m.hurdle` + Discus + Pole.vault + Javeline +
##     Rank + Points, data = train)
##
## Residuals:
##      SEBRLE        CLAY      KARPOV     BERNARD      YURKOV     WARNERS
##   0.0028267  -0.0177720   0.0062708   0.0134629  -0.0070351  -0.0134365
##   ZSIVOCZKY    McMULLEN   MARTINEAU       HERNU      BARRAS        NOOL
##   0.0237488  -0.0002955  -0.0067691   0.0112844  -0.0017888  -0.0161802
## BOURGUIGNON
##   0.0056838
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.399e+03  3.377e+00  414.43  0.00154 **
## `100m`         -3.640e+01  1.350e-01 -269.65  0.00236 **
## Long.jump       3.786e+01  1.349e-01  280.64  0.00227 **
## Shot.put        8.986e+00  5.906e-02  152.14  0.00418 **
## High.jump       1.435e+02  6.382e-01  224.89  0.00283 **
## `400m`         -7.147e+00  4.424e-02 -161.54  0.00394 **
## `110m.hurdle`  -1.879e+01  1.050e-01 -178.86  0.00356 **
## Discus          3.333e+00  1.575e-02  211.58  0.00301 **
## Pole.vault      4.759e+01  2.699e-01  176.30  0.00361 **
## Javeline        2.278e+00  1.079e-02  211.06  0.00302 **
## Rank           -2.701e-01  2.347e-02  -11.51  0.05518 .
## Points         -1.606e-01  3.286e-04 -488.68  0.00130 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.04253 on 1 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.034e+04 on 11 and 1 DF,  p-value: 0.002595
```

```r
# Linearity
summary(RegModel.1)
```

```
##
## Call:
## lm(formula = `1500m` ~ `100m` + Long.jump + Shot.put + High.jump +
##     `400m` + `110m.hurdle` + Discus + Pole.vault + Javeline +
##     Rank + Points, data = train)
##
## Residuals:
##      SEBRLE        CLAY      KARPOV     BERNARD      YURKOV     WARNERS
##   0.0028267  -0.0177720   0.0062708   0.0134629  -0.0070351  -0.0134365
##   ZSIVOCZKY    McMULLEN   MARTINEAU       HERNU      BARRAS        NOOL
##   0.0237488  -0.0002955  -0.0067691   0.0112844  -0.0017888  -0.0161802
## BOURGUIGNON
##   0.0056838
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.399e+03  3.377e+00  414.43  0.00154 **
## `100m`        -3.640e+01  1.350e-01 -269.65  0.00236 **
## Long.jump      3.786e+01  1.349e-01  280.64  0.00227 **
## Shot.put       8.986e+00  5.906e-02  152.14  0.00418 **
## High.jump      1.435e+02  6.382e-01  224.89  0.00283 **
## `400m`        -7.147e+00  4.424e-02 -161.54  0.00394 **
## `110m.hurdle` -1.879e+01  1.050e-01 -178.86  0.00356 **
## Discus         3.333e+00  1.575e-02  211.58  0.00301 **
## Pole.vault     4.759e+01  2.699e-01  176.30  0.00361 **
## Javeline       2.278e+00  1.079e-02  211.06  0.00302 **
## Rank          -2.701e-01  2.347e-02  -11.51  0.05518 .
## Points        -1.606e-01  3.286e-04 -488.68  0.00130 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04253 on 1 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.034e+04 on 11 and 1 DF,  p-value: 0.002595
```

```r
# Independence of residuals
dwtest(RegModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  RegModel.1
## DW = 2.5981, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is not 0
```

```r
# Normality
shapiro.test(residuals(RegModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(RegModel.1)
## W = 0.9731, p-value = 0.9283
```

```
# Homoscedasticity
lmtest::bptest(RegModel.1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  RegModel.1
## BP = 12.491, df = 11, p-value = 0.3279
```

```
#We see that pass the assumptinos...
```

Once the model is created and verified what we want to see is the results of this model in the test set.

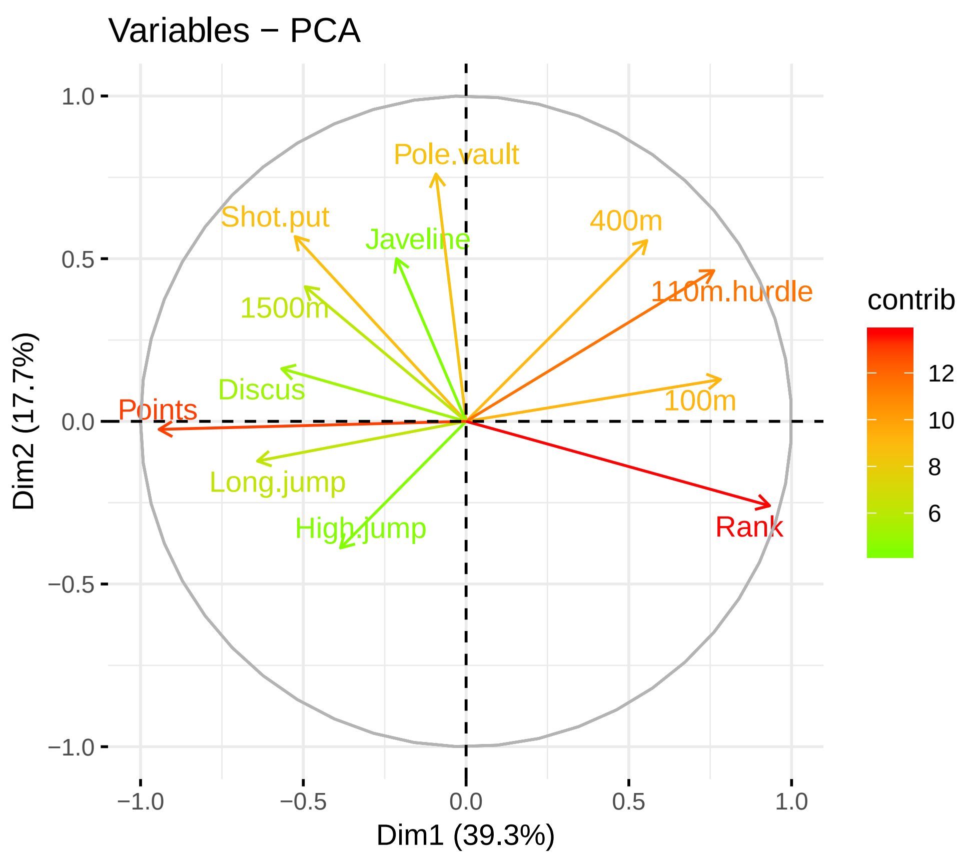
```
prediction <- as.data.frame(predict(RegModel.1, newdata=test, interval="prediction"))
prediction$difference <- sapply((test$`1500m` <= prediction$upr) & (test$`1500m` >= prediction$lwr),
                                ifelse, 'yes', 'no')

hitRate <- sum(prediction$difference == "yes")/
  (sum(prediction$difference == "no")+sum(prediction$difference == "yes"))
```

We got arround a 50% by including everything in the model (better flip a coin). Let's now analyze which variables are really relevant:

```
library(factoextra)
train$Competition <-NULL
pca <- prcomp(train, scale = TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5    PC6     PC7
## Standard deviation     2.1709 1.4573 1.2052 1.1781 0.98630 0.7558 0.71957
## Proportion of Variance 0.3927 0.1770 0.1210 0.1157 0.08107 0.0476 0.04315
## Cumulative Proportion  0.3927 0.5697 0.6907 0.8064 0.88745 0.9351 0.97820
##                            PC8     PC9    PC10    PC11      PC12
## Standard deviation     0.35299 0.31463 0.16837 0.09807 0.0002368
## Proportion of Variance 0.01038 0.00825 0.00236 0.00080 0.0000000
## Cumulative Proportion  0.98859 0.99684 0.99920 1.00000 1.0000000
```

```
fviz_pca_var(pca,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("chartreuse", "darkgoldenrod1", "RED"),
             repel = TRUE     # Avoid text overlapping
             )
```



If we pay attention to the PCA, we can clearly see that the variables that come from similar disciplines (e.g 100m,110m hurdle,400m) need to be present in the model.Because there will be athletes that are more Speed alike, Stenght alike… and missing this in the model would be a huge mistake. Variables like Rank or Points don't add any complexity to the model since are relative to other data that we already have.

Once that now we have a better idea of what model we need to represent our data I propose the following model (which could probably be improved). The main idea is to get from each quartile the most representative even that I would even say taht just with left side ones, would be enough since who is good in those, is good in 1500m.

```
RegModel.1 <- lm(`1500m` ~ `Shot.put` + `High.jump`
                 + `110m.hurdle` + `Discus` + `Pole.vault` , data=train)
summary(RegModel.1)
```

```
##
## Call:
## lm(formula = `1500m` ~ Shot.put + High.jump + `110m.hurdle` +
##     Discus + Pole.vault, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.727  -5.239   3.359   6.904  13.436
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   225.317    244.822   0.920    0.388
## Shot.put       -2.433     10.806  -0.225    0.828
## High.jump       5.377     63.394   0.085    0.935
## `110m.hurdle`   -5.694      8.931  -0.638    0.544
## Discus           1.118      1.773   0.631    0.548
## Pole.vault      23.834     25.753   0.925    0.386
##
## Residual standard error: 13.33 on 7 degrees of freedom
## Multiple R-squared:  0.3084, Adjusted R-squared:  -0.1856
## F-statistic: 0.6243 on 5 and 7 DF,  p-value: 0.6878
```

```
# Linearity
summary(RegModel.1)
```

```
##
## Call:
## lm(formula = `1500m` ~ Shot.put + High.jump + `110m.hurdle` +
##     Discus + Pole.vault, data = train)
##
## Residuals:
##     Min       1Q  Median      3Q     Max
## -23.727  -5.239   3.359   6.904  13.436
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    225.317    244.822   0.920    0.388
## Shot.put        -2.433     10.806  -0.225    0.828
## High.jump        5.377     63.394   0.085    0.935
## `110m.hurdle`   -5.694      8.931  -0.638    0.544
## Discus           1.118      1.773   0.631    0.548
## Pole.vault      23.834     25.753   0.925    0.386
##
## Residual standard error: 13.33 on 7 degrees of freedom
## Multiple R-squared:  0.3084, Adjusted R-squared:  -0.1856
## F-statistic: 0.6243 on 5 and 7 DF,  p-value: 0.6878
```

```
# Independence of residuals
dwtest(RegModel.1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  RegModel.1
## DW = 2.4357, p-value = 0.6634
## alternative hypothesis: true autocorrelation is not 0
```

```
# Normality
shapiro.test(residuals(RegModel.1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(RegModel.1)
## W = 0.92333, p-value = 0.2781
```

```
# Homoscedasticity
lmtest::bptest(RegModel.1)
```

```
##
##  studentized Breusch-Pagan test
```

```
##
## data:  RegModel.1
## BP = 3.0903, df = 5, p-value = 0.6861
```

```
#We see that pass the assumptinos...
prediction <- as.data.frame(predict(RegModel.1, newdata=test, interval="prediction"))
prediction$difference <- sapply((test$`1500m` <= prediction$upr) & (test$`1500m` >= prediction$lwr), if

hitRate <- sum(prediction$difference == "yes")/
  (sum(prediction$difference == "no")+sum(prediction$difference == "yes"))
hitRate
```

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

Note that the prediction interval is much larger than the confidence interval. The interval works with the performance of an athlete predicted the performance that comes after and just one after. Meanwhile the confidence interval give us information about the performances that comes later (more than one).

4 FOURTH QUESTION: WORKING WITH REAL DATA

First we are going to set up the environment and we are going to read the downloaded data.

```
setwd("C:/Users/Meyerhofer/Desktop/SMDELabs/Lab3/")
dataset <- read.csv("dataset/marathon_results_2017.csv", sep=",",
                    stringsAsFactors = default.stringsAsFactors(), header = T)
```

Our target group will be men of 20-30 years. It is chosen arbitrarly (with a fast overlook at the data it is cleaner that women one). Since we have data that does not fit this criteria, we are going to remove it since it's not relevant to our question:

```
#Removing females & Removing ages not in [20-30]years
condition <- (dataset$Age >= 20 & dataset$Age <= 30 & dataset$M.F == "M")
data_used = dataset[condition,]
```

Once selected our data to start with the process, we will first identify the labels that appear in the dataset and we will explore them to have a first insight if can be useful or not to our model (pending to justify why each one is or is not useful). Remove columns that are what we want to predict so we can't use them to generate a model

```
data_used[, c("X","X.1","Citizen", "X15K", "X20K", "X25K", "X30K",
              "X35K", "X40K", "Half", "Pace", "Gender", "Division", "Overall", "M.F")] <- list(NULL)

names(data_used)
```

```
##  [1] "Bib"           "Name"          "Age"           "City"
##  [5] "State"         "Country"       "X5K"           "X10K"
##  [9] "Proj.Time"     "Official.Time"
```

Now that we have the data that we want, we will pass all relevant time variables to seconds in order to make it easier to work with

```
library(lubridate)
data_used$X5K <- as.numeric(hms(data_used$X5K))
data_used$X10K <- as.numeric(hms(data_used$X10K))
data_used$Official.Time <- as.numeric(hms(data_used$Official.Time))
data_used$Bib <- as.numeric(data_used$Bib)
```

Now that we have what we need and we preprocessed some of the data, we will split between train and test. This will be done in a proportion of 70-30 with a random distribution.

```
set.seed(1234)
index <- sample(nrow(data_used),nrow(data_used)*0.70)
train <- as.data.frame(data_used[index,])
test <- as.data.frame(data_used[-index,])
```

Now let's proceed to model our training test. Note that we have in training the Official time, but we are not going to use it. I only have it as reference.

The parameters that I am going to use are the following (this is a simple model, we are going to try more complex models later)

- X10K
- Bib
- X5K

```
model <- lm(`Official.Time` ~ `X5K` + `X10K` +`Bib`, data=train)
```

```r
library("lmtest")
# Independence of residuals
dwtest(model, alternative ="two.sided", data=train)
```

```
##
##  Durbin-Watson test
##
## data:  model
## DW = 2.0043, p-value = 0.9372
## alternative hypothesis: true autocorrelation is not 0
```

```r
# Normalitat
shapiro.test(residuals(model))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(model)
## W = 0.88912, p-value < 2.2e-16
```

```r
# Homoscedasticity
lmtest::bptest(model)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model
## BP = 23.338, df = 3, p-value = 3.433e-05
```

Once tested our model, we are going to predict our test dataset and see which is the error that our model has:

```r
prediction <- as.data.frame(predict(model, newdata=test, interval="prediction"))
```

We calculate the mean standart error:

```r
#Average of time error.
mse <- mean(prediction$fit - test$Official.Time)^2

as.hours <- function (seconds) {
  paste(floor(seconds/3600L), floor (seconds/60L) %% 60, floor (seconds) %% 60L, sep=":")
}
as.hours(sqrt(mse))
```

```
## [1] "0:1:54"
```

We can see that our average error is of arround 2-5 minutes (depending on the seed and also on the training set).

In order to select the most appropiate parameters it's usually a good idea to see with PCA which of these parameters, are the most relevant ones:

```r
pca.train <- as.data.frame(train)
names(pca.train)
```

```
##  [1] "Bib"           "Name"          "Age"           "City"
##  [5] "State"         "Country"       "X5K"           "X10K"
##  [9] "Proj.Time"     "Official.Time"
```

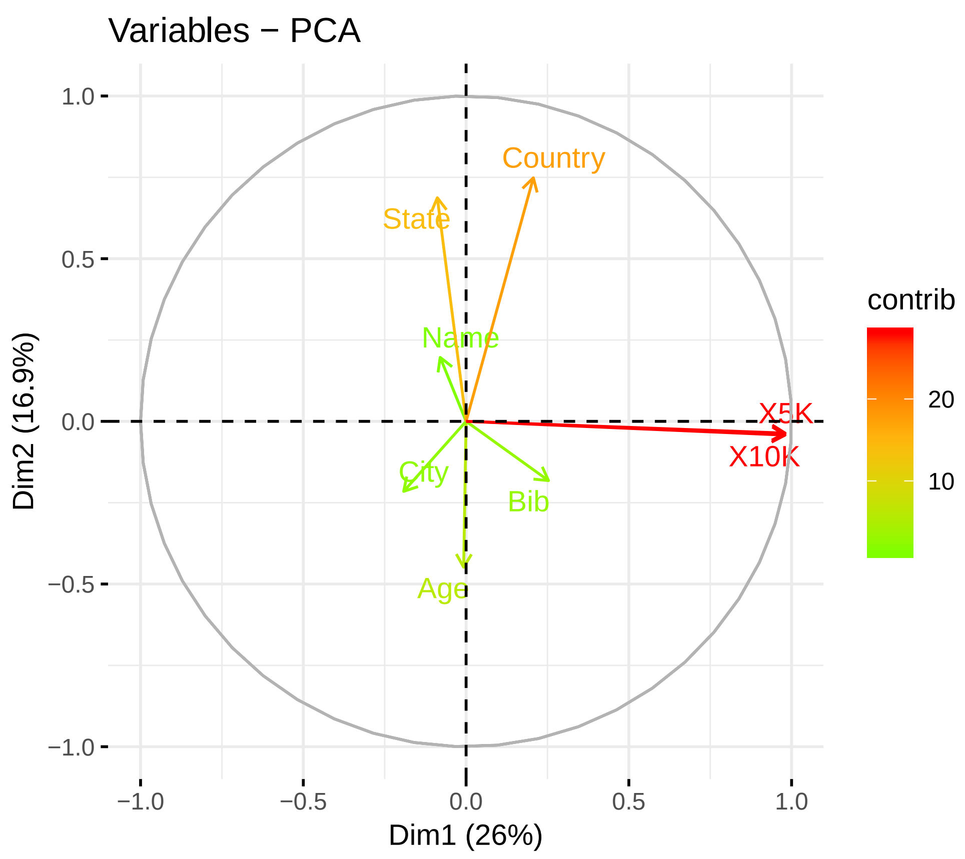
```
pca.train$Name <- as.numeric(as.factor(pca.train$Name))
pca.train$City <- as.numeric(as.factor(pca.train$City))
pca.train$Country <- as.numeric(as.factor(pca.train$Country))
pca.train$State <- as.numeric(as.factor(pca.train$State))
train$State <- as.numeric(as.factor(train$Country))
testState <- as.numeric(as.factor(test$Country))
pca.train$Official.Time <- NULL
pca.train$Proj.Time <- NULL
pca.train$Age <- as.numeric(pca.train$Age)
names(pca.train)
```

```
## [1] "Bib"     "Name"    "Age"     "City"    "State"   "Country" "X5K"
## [8] "X10K"
```

```
pca <- prcomp(~ ., data=pca.train, na.action=na.omit, scale=TRUE)
summary(pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5    PC6    PC7
## Standard deviation     1.4418 1.1641 1.0153 0.9936 0.9777 0.9625 0.8114
## Proportion of Variance 0.2599 0.1694 0.1289 0.1234 0.1195 0.1158 0.0823
## Cumulative Proportion  0.2599 0.4292 0.5581 0.6815 0.8010 0.9168 0.9991
##                            PC8
## Standard deviation     0.08570
## Proportion of Variance 0.00092
## Cumulative Proportion  1.00000
```

```
#install.packages("factoextra")
library(factoextra)
fviz_pca_var(pca, col.var = "contrib",
             gradient.cols = c("chartreuse", "darkgoldenrod1", "RED"),
             repel = TRUE)
```



```r
eig.val <- get_eigenvalue(pca)
res.var <- get_pca_var(pca)
res.var$contrib        # Contributions to the PCs
```

```
##                Dim.1       Dim.2      Dim.3       Dim.4      Dim.5
## Bib      3.085090584  2.44754501 18.1944448  6.93798147 15.0050473
## Name     0.304429443  2.85053207 48.5879071  0.04190264 44.9924302
## Age      0.002646312 14.84329892  7.4993970 19.71181131 16.0987332
## City     1.768458937  3.41789278  8.9483198 69.21777635 10.4668141
## State    0.375234082 34.86453234 10.6941975  0.06880867 12.6872420
## Country  2.054524336 41.34809847  5.2913546  0.93310998  0.5177764
## X5K     46.221570120  0.09637817  0.3826983  1.55043573  0.1003492
## X10K    46.188046186  0.13172223  0.4016809  1.53817385  0.1316077
##               Dim.6      Dim.7        Dim.8
## Bib     54.30011535  0.0295887 1.867873e-04
## Name     0.92351293  2.2986501 6.355853e-04
## Age     36.39270188  5.4505330 8.784181e-04
## City     0.22749763  5.9530181 2.223018e-04
## State    5.81315574 35.4963517 4.779589e-04
## Country  0.01214283 49.8426387 3.547437e-04
## X5K      1.16431276  0.4779096 5.000635e+01
## X10K     1.16656087  0.4513102 4.999090e+01
```

By the PCA we can see that X10K and X5K are the same and we can see that the factors that contribute the most aside from the times, are the country and the state that most likely we will discard State because is very specific of USA. City is not relevant at all and Age is not relevant neither.

Another model I would consider is:

```r
model <- lm(`Official.Time` ~ `X10K` +`Bib`+ `Age` + `Country`, data=train)
```

We see how good the model is (we should also apply the assumptions):

```r
#We do this bad replacement because it would imply a work arround to make appear or labels in train tha
#So I am doing it like this even is not the best.
test$Country[test$Country == "NED"] <- "USA"
test$Country[test$Country == "SIN"] <- "USA"
prediction <- as.data.frame(predict(model, newdata=test, interval="prediction"))
```

We calculate the mean standart error:

```r
#Average of time error.
mse <- mean(prediction$fit - test$Official.Time)^2

as.hours <- function (seconds) {
  paste(floor(seconds/3600L), floor (seconds/60L) %% 60, floor (seconds) %% 60L, sep=":")
}
as.hours(sqrt(mse))
```

```
## [1] "0:1:52"
```

We can see that the final result has been improved a bit but I do not think personally that by adding/substracting variables of the PCA we will significantly get better. By doing this we are not creating new knowledge or highlighting something like for instance the pace that could be interesting to test (even that by having the X5K and X10K you already have it).