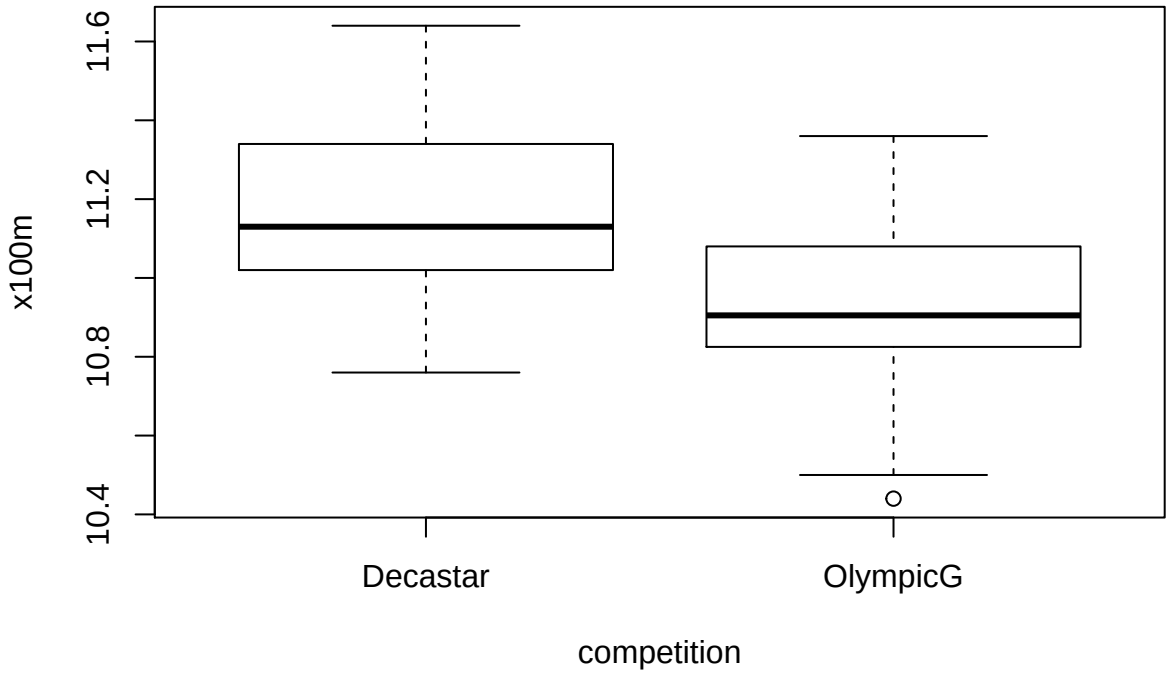
First Question: Visualization, Chi-Square and t-test

Arnau Abella

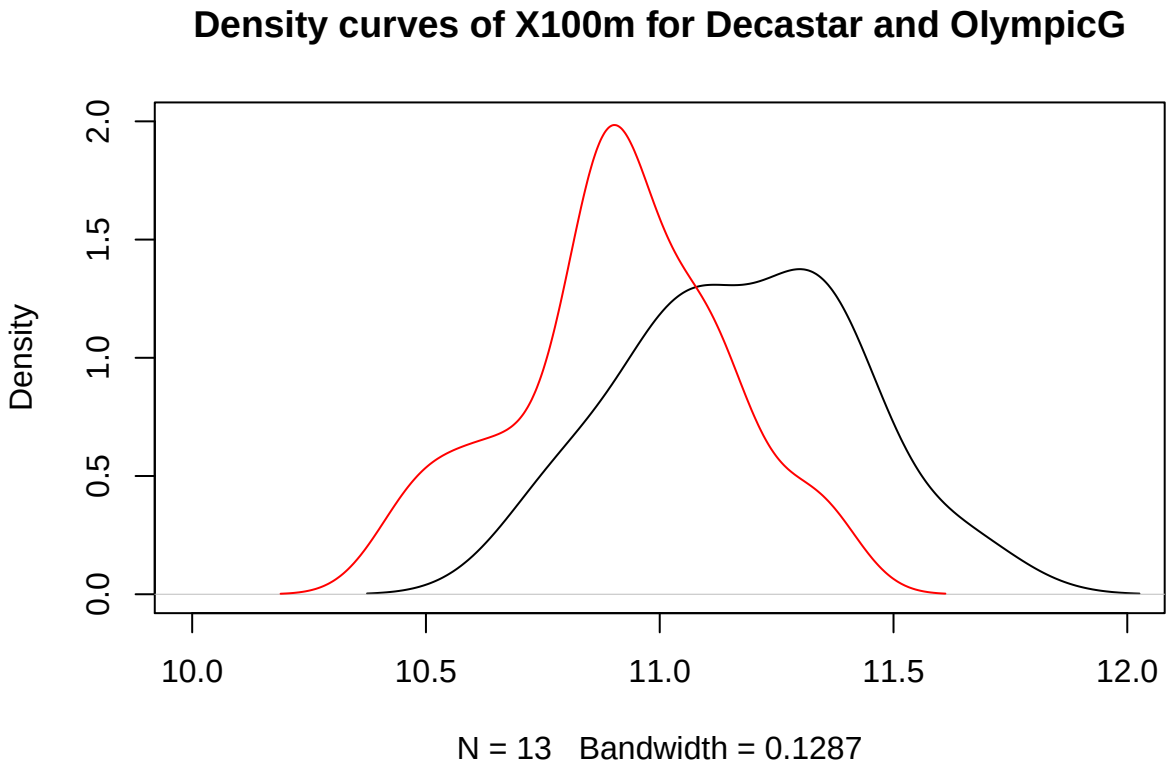
03/03/2020

a. Analyze the distribution of "X100m" according to the type of competition by using boxplot. Write your conclusion.

```
decathlon <- read.csv( "/home/arnau/MIRI/SMDE/hw1/decathlon.csv", header = TRUE, sep = ",")
dim(decathlon)
head(decathlon)
summary(decathlon)
boxplot(X100m~Competition, ylab="x100m", xlab="competition", data=decathlon)
```




```
decastar<-which(decathlon$Competition=="Decastar")
olympic <-which(decathlon$Competition=="OlympicG")
plot(density(decathlon$X100m[decastar]),main="Density curves of X100m for Decastar and OlympicG", xlim=
lines(density(decathlon$X100m[olympic]),col=2)
```



The discret random variable *X100m* follows a normal distribution by the shape of its density function.

Both distributions are very similar but the medians do not coincide.

Notice that we have less samples from Decaster (13) than from OlympicG (28) so we need to make some probabilistic analysis before refusing that the medians are equals.

From the statistics, it is easy to see that, in average, the runners from OlympicG are one second faster than the ones from Decaster.

b. Create a new categorical variable with two categories from the variable "X100m" by using 11 seconds as the cut-off point. Make a cross table from the new categorical variable and the "Competition". Are these two variables independent? Write your conclusion by checking marginal probabilities and test the independency of two variables by using Chi-Square test.

```
decathlon$X100m_11s<- cut(decathlon$X100m, c(11,0 , 12.0))
levels(decathlon$X100m_11s)<-c("< 11s","> 11s")
tab<-table(decathlon$X100m_11s, decathlon$Competition)
tab
```

```
##
##         Decastar OlympicG
##   < 11s        2       19
##   > 11s       11        9
```

```
prop.table(tab)
```

```
##
##           Decastar   OlympicG
##   < 11s 0.04878049 0.46341463
##   > 11s 0.26829268 0.21951220
```

```
chisq.test(tab) # Independent and normally
```

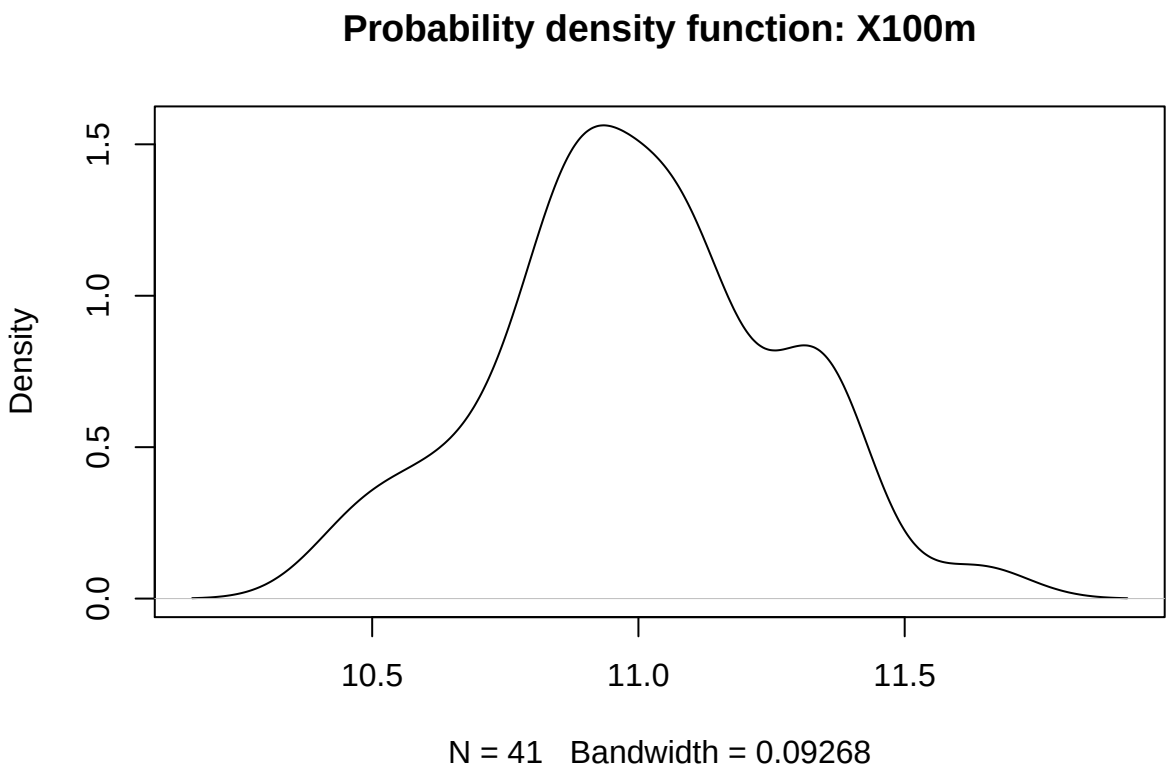
```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  tab
## X-squared = 7.7962, df = 1, p-value = 0.005236
```

As the p-value 0.0052 from χ^2 is smaller than the .05 significance level, we can reject the null hypothesis that running 100 meters in less/more than 11 seconds is independent of the kind of competition.

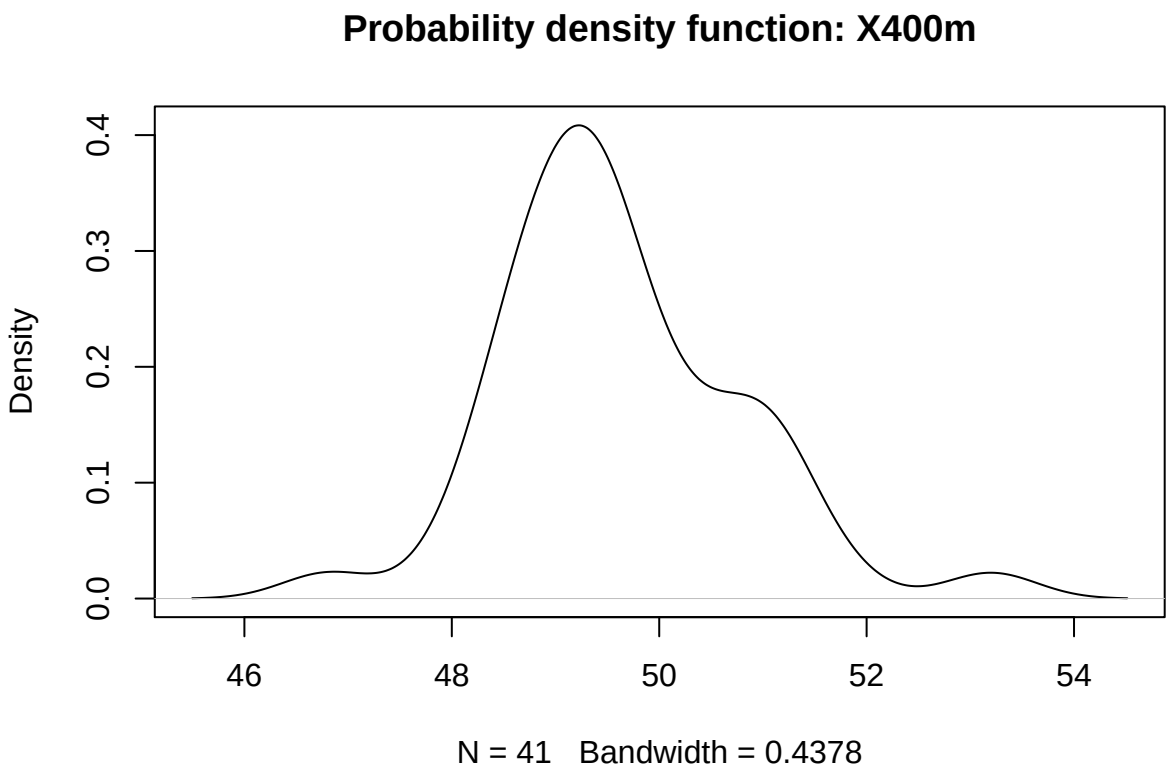
c. Visualize the distribution of quantitative variables by using proper graph. Which of these variables follows a Normal distribution?

All quantitative variables follow a Normal distribution except for **Rank** and **Points**. Here are some sample plots:

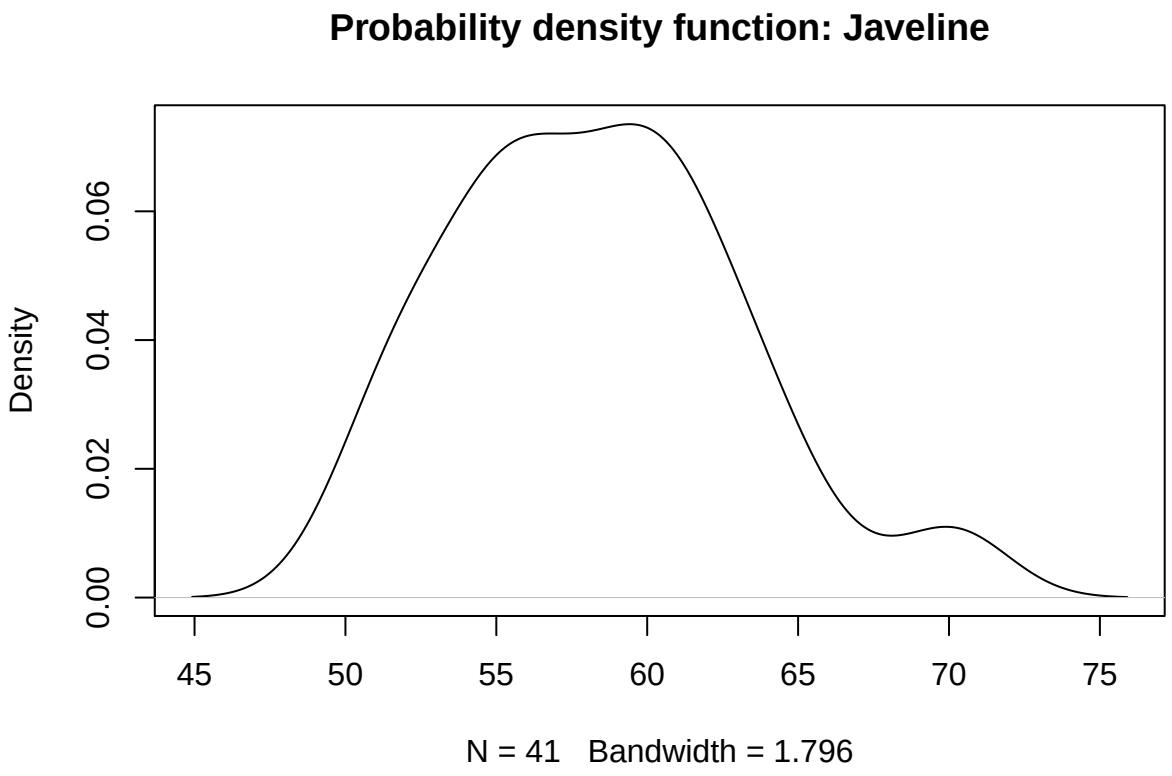
```
plot(density(decathlon$X100m), main="Probability density function: X100m")
```




```
plot(density(decathlon$X400m), main="Probability density function: X400m")
```

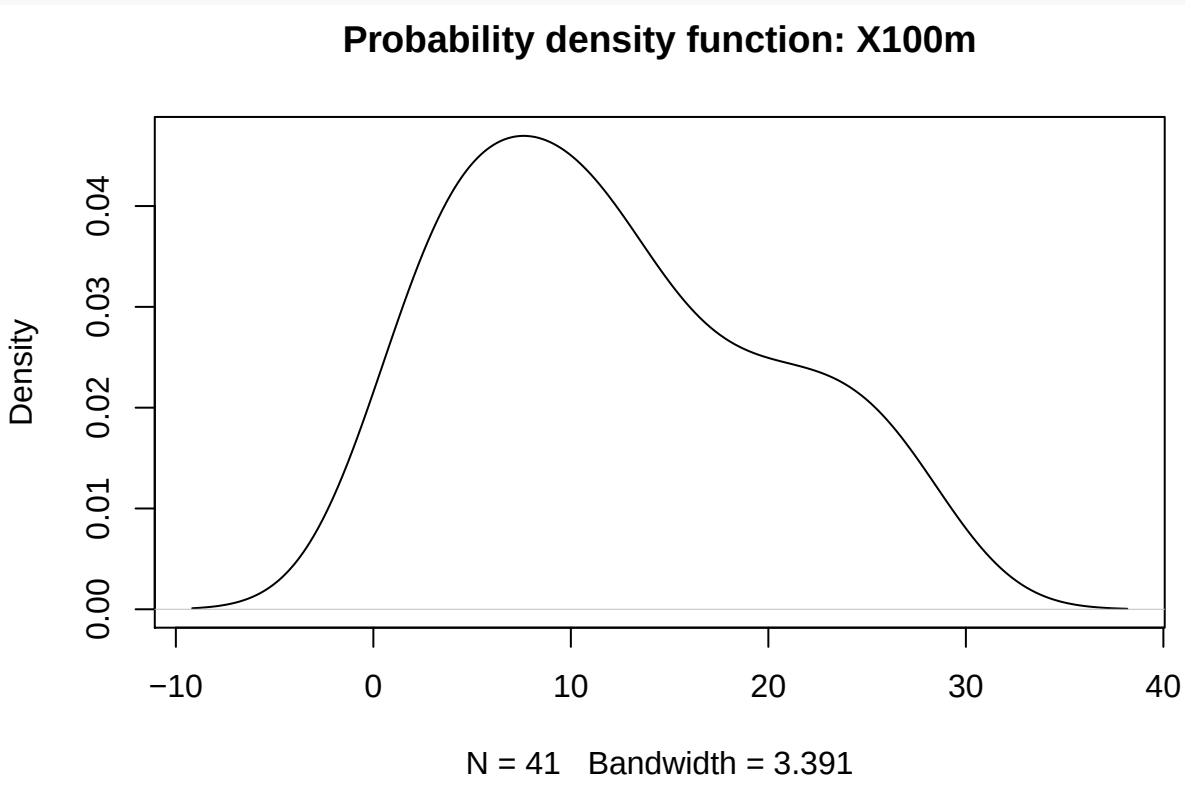



```
plot(density(decathlon$Javeline), main="Probability density function: Javeline")
```

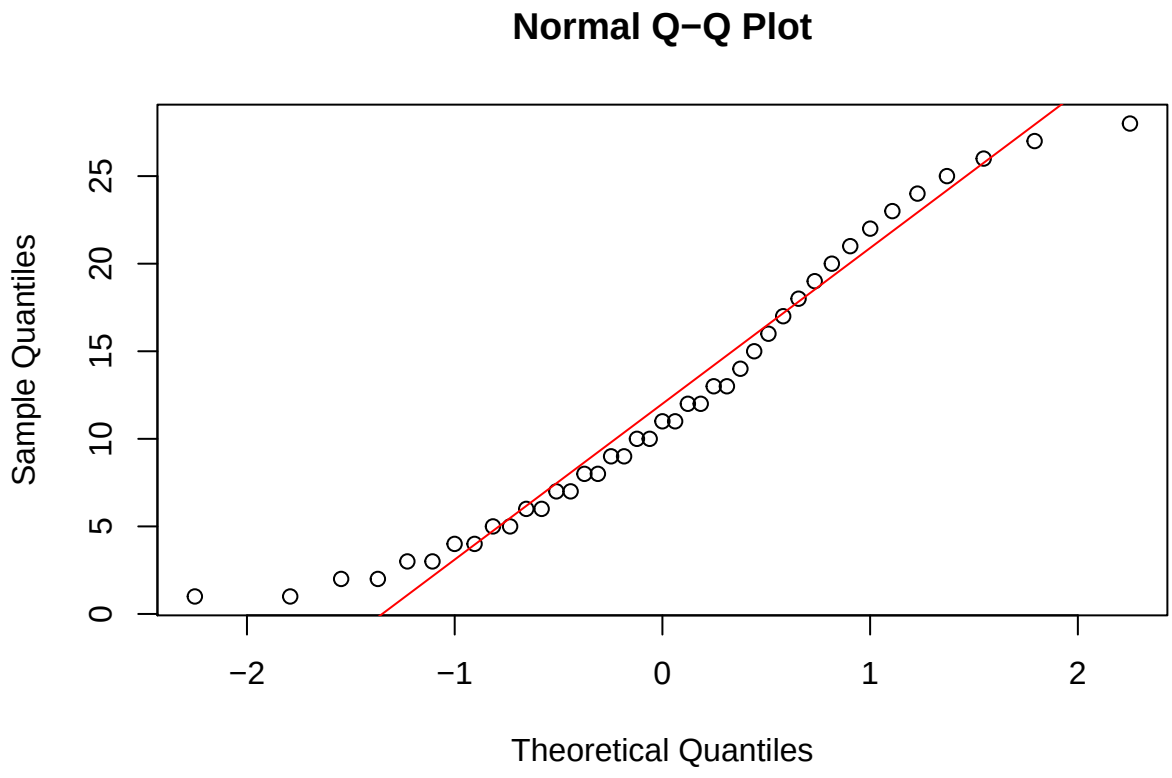


To check that Rank and Position do not follow a Normal distribution we used qqnorm and Shapiro-Wilk normality test.

```r
plot(density(decathlon$Rank), main="Probability density function: X100m")
```




```r
qqnorm(decathlon$Rank); qqline(decathlon$Rank, col = 2)
```

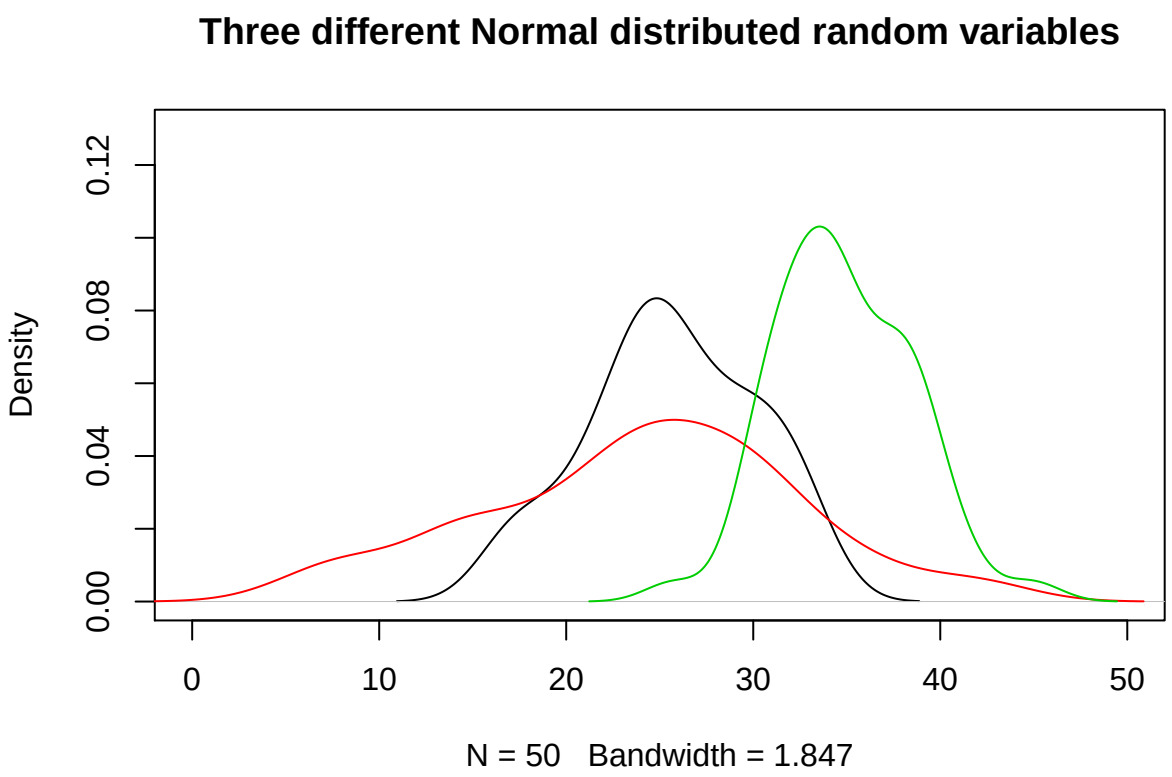


```r
shapiro.test(decathlon$Rank)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  decathlon$Rank
## W = 0.94188, p-value = 0.03649
```

d. Generate three Normally distributed random variables of length 50. Two of them should have the same mean, different standard deviations while the third one has a different mean but the same standard deviation with the first distribution. Use t test to compare mean differences between three variables.

```
s1<-rnorm(50, mean=25,sd=4)
s2<-rnorm(50, mean=25,sd=8)
s3<-rnorm(50, mean=35,sd=4)

plot(density(s1),xlim=c(0,50),ylim=c(0,0.13), main="Three different Normal distributed random variables"
lines(density(s2),col=2)
lines(density(s3),col=3)
```



Let's compare the mean difference using t-test:

```
t.test(s1, s2, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  s1 and s2
## t = 1.1862, df = 98, p-value = 0.2384
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.070083  4.250051
## sample estimates:
## mean of x mean of y
##  25.62440  24.03441
```

We can't refuse that the mean of **s1** and **s2** are different because the p-value 0.56 is greater than α. However, we can refuse this for **s1** and **s3** as expected because the p-value is way slower than α.

```r
t.test(s1, s3, var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  s1 and s3
## t = -11.264, df = 98, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -10.83724  -7.59059
## sample estimates:
## mean of x mean of y
##  25.62440  34.83832
```

Additionaly, we can test the variance of the distributions. The distributions **s1** and **s2** have differente variance as expected.

```r
var.test(s1, s2, var.equal=TRUE)
```

```
##
##  F test to compare two variances
##
## data:  s1 and s2
## F = 0.28903, num df = 49, denom df = 49, p-value = 2.73e-05
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##  0.1640200 0.5093327
## sample estimates:
## ratio of variances
##          0.2890342
```

e. Test if there is a difference between two type of competitions according to the variables "X100m" and "X400m" by using t-test.

After performing the Student's t-test on the *X100m*, we **can** refuse the null hypothesis H_0 that there is no significant difference in the means because the p-value 0.00407 is smaller than α. So we can state with 95% of confidence that there is a statistical difference in the times for the 100 meters race dependening on the competition type.

```
t.test(X100m ~ Competition, data = decathlon)
```

```
##
##  Welch Two Sample t-test
##
## data:  X100m by Competition
## t = 3.2037, df = 22.168, p-value = 0.00407
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.09164794 0.42769272
## sample estimates:
## mean in group Decastar mean in group OlympicG
##               11.17538               10.91571
```

After performing the Student's t-test on the *X400m*, we **can't** refuse the null hypothesis H_0 that there is no significant difference in the means because the p-value 0.9543 is greater than α. So there is no significative evidence that the mean time on the 400 meters race is different depending on the competition type.

```
t.test(X400m ~ Competition, data = decathlon)
```

```
##
##  Welch Two Sample t-test
##
## data:  X400m by Competition
## t = 0.05771, df = 32.106, p-value = 0.9543
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.6858299  0.7258299
## sample estimates:
## mean in group Decastar mean in group OlympicG
##                  49.63                  49.61
```

Second Question: ANOVA

Arnau Abella

10/03/2020

Anova Test

a) Generate three population using your own algorithm.

In order to generate the three normal populations I used the following Haskell script

```haskell
stdDev :: Double
stdDev = 1.0

main = do
  IO.withFile "normal.csv" IO.WriteMode $
    \handle -> do
      vss <- traverse (normalV 10000) [0.0, 0.0, 10.0]
      forM_ vss $ \vs ->
        let bs = Csv.encode [GV.toList vs]
        in LBS.hPut handle bs
  where
    normalV n mean =
      withSystemRandom $
        \(gen::GenST s) -> normalVector mean stdDev gen n :: ST s (UV.Vector Double)

normalVector :: (PrimMonad m, Vector v Double)
             => Double            -- ^ Mean
             -> Double            -- ^ Standard deviation
             -> Gen (PrimState m)
             -> Int               -- ^ vector length
             -> m (v Double)
normalVector mean std gen n =
  GV.replicateM n (MWCD.normal mean std gen)

standardVector :: (PrimMonad m, Vector v Double)
             => Gen (PrimState m)
             -> Int               -- ^ vector length
             -> m (v Double)
standardVector = normalVector 0.0 1.0
```

b) Analyze using an ANOVA if these three populations are different (or not) depending on the parameter selected.

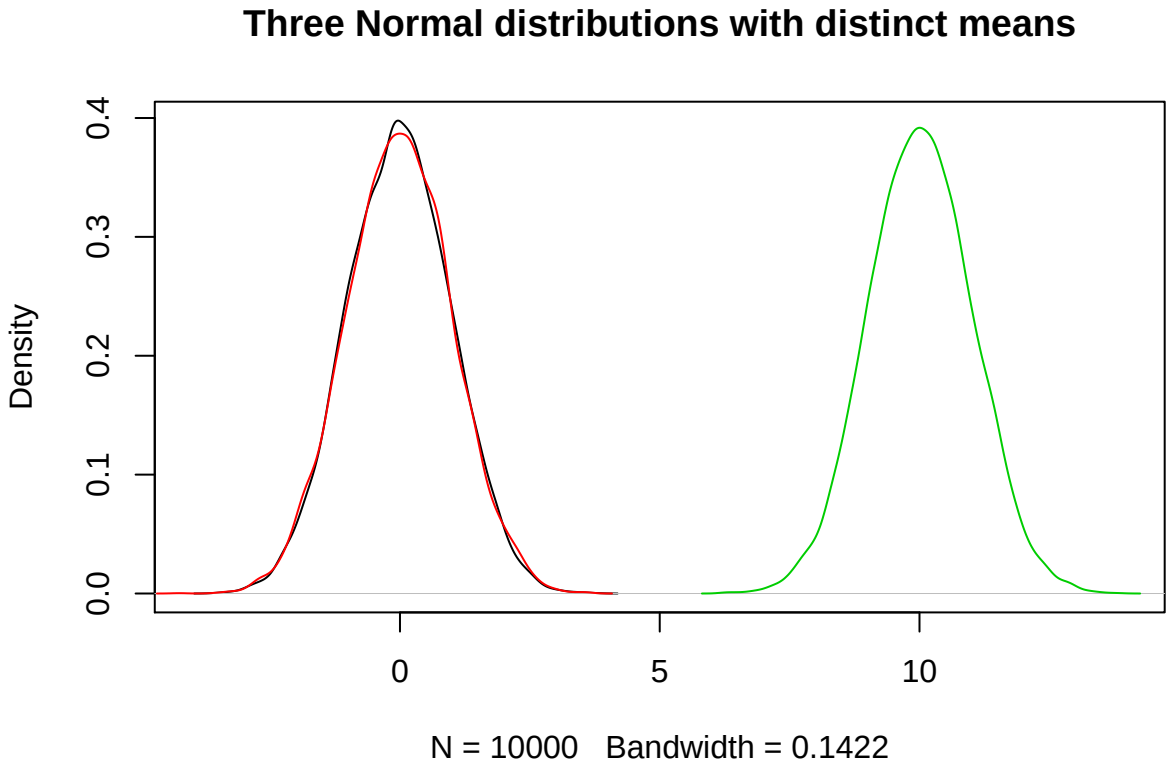
```r
# 30,000 values in total.
v <- sapply(read.csv( paste(root, 'normal.csv', sep='/'), header = FALSE, sep = ","), as.numeric)
v1 <- v[1, ] # 10,000 values
v2 <- v[2, ] # 10,000 values
v3 <- v[3, ] # 10,000 values

plot(density(v1),xlim=c(-4,14),main="Three Normal distributions with distinct means")
lines(density(v2),col=2)
lines(density(v3),col=3)
```

```r
v1n=data.frame(x1=v1, x2="v1")
v2n=data.frame(x1=v2, x2="v2")
v3n=data.frame(x1=v3, x2="v3")

library(RcmdrMisc)
```

```
## Loading required package: car
```

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

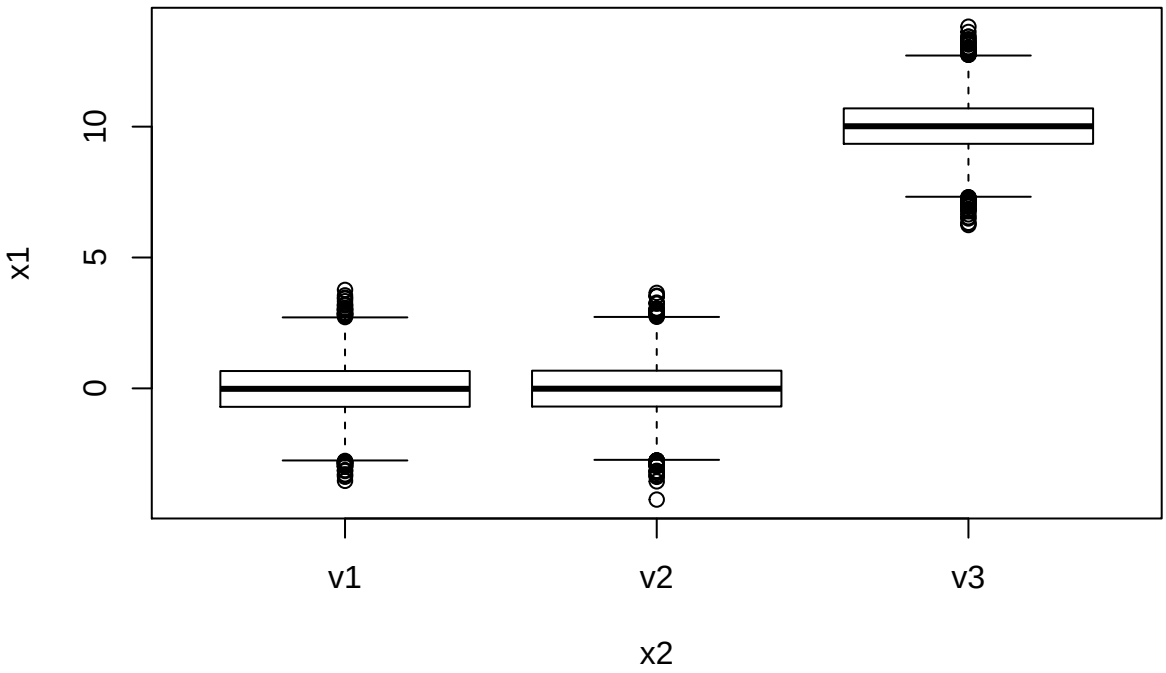
```
## Loading required package: sandwich
```




```r
# We create a single data frame
data=mergeRows(v1n, v2n, common.only=FALSE)
data=mergeRows(as.data.frame(data), v3n, common.only=FALSE)

AnovaModel.1 <- aov(x1 ~ x2, data=data)
summary(AnovaModel.1) # Pr(>F) = p-value
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## x2              2 671726  335863  334459 <2e-16 ***
## Residuals   29997  30123       1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
Boxplot(x1~x2,data=data,id=FALSE)
```



From the output of the ANOVA test, we can see that the PR(>F) is smaller than the p-value so we **can** refuse the null hypothesis that there is no significant difference between means of the different groups.

Red and White Wine Quality

We want to analyze if in both (type or quality) affects some properties of the wine. After combining the two datasets (one for red wines and one for white wines), you should create two variables: "type" that identifies if the wine is red or white, and wine quality categorized in three groups: <5 (low), 5-6(medium) and >6 (high). Once you complete preprocessing steps, please answers the following questions applying appropriate statistical techniques:

```
red  <-read.csv2(paste(root, 'winequality-red.csv', sep='/'), dec=".") # 1599 x 12
white<-read.csv2(paste(root, 'winequality-white.csv', sep='/'), dec=".") # 4898 x 12

# Combine the rows
winequal<-rbind(red,white)

# Categorical Variable: type
winequal$type<-as.factor(rep(c(1,2),c(nrow(red),nrow(white))))
levels(winequal$type)<-c("red","white")
summary(winequal$type)
```

```
##   red white
##  1599  4898
```

```
# Categorical Variable: category (low, medium, high)
winequal$category<- cut(winequal$quality,c(1,5,6,10))
summary(winequal$category)
```

```
##  (1,5]  (5,6] (6,10]
##   2384   2836   1277
```

Before answer the questions, we are going to check if the assumptions of ANOVA are fulfilled for each numerical variable.

```
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
anova1 <- aov(alcohol ~ quality, data=winequal)
```

Independent obs.

```
# Durbin Watson, Ho = autocorrelation of the disturbances is 0.
dwtest(anova1, alternative ="two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  anova1
## DW = 1.488, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is not 0
```

Normality

```
#Shapiro test (Normality)
# shapiro.test(residuals(anova1))
```

Homogeneity

```
#Breusch Pagan test (Variance equality)
bptest(anova1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  anova1
## BP = 101.75, df = 1, p-value < 2.2e-16
```

```
# leveneTest(alcohol~quality, data=winequal)
```

a) Which of the chemical properties influence the quality of the wines?

```
for (i in 1:11){
  print(colnames(winequal)[i])
  print(summary(aov(winequal[,i]~category,data=winequal)))
}
```

```
## [1] "fixed.acidity"
##               Df Sum Sq Mean Sq F value   Pr(>F)
## category       2     57  28.455   17.01 4.27e-08 ***
## Residuals   6494  10861   1.672
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "volatile.acidity"
##               Df Sum Sq Mean Sq F value Pr(>F)
## category       2  13.09   6.547   260.9 <2e-16 ***
## Residuals   6494 162.98   0.025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "citric.acid"
##               Df Sum Sq Mean Sq F value   Pr(>F)
## category       2   0.89  0.4472   21.31 5.98e-10 ***
## Residuals   6494 136.28  0.0210
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "residual.sugar"
##               Df Sum Sq Mean Sq F value   Pr(>F)
## category       2    614  307.11   13.62 1.25e-06 ***
## Residuals   6494 146434   22.55
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "chlorides"
##               Df Sum Sq Mean Sq F value Pr(>F)
## category       2  0.345 0.17233   146.7 <2e-16 ***
## Residuals   6494  7.628 0.00117
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "free.sulfur.dioxide"
##               Df  Sum Sq Mean Sq F value  Pr(>F)
## category       2    4122  2060.8   6.553 0.00144 **
## Residuals   6494 2042386   314.5
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "total.sulfur.dioxide"
##               Df   Sum Sq Mean Sq F value   Pr(>F)
## category       2    73818   36909   11.59 9.44e-06 ***
## Residuals   6494 20679084    3184
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "density"
##               Df  Sum Sq  Mean Sq F value Pr(>F)
## category       2 0.00629 0.003144   391.7 <2e-16 ***
## Residuals   6494 0.05212 0.000008
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "pH"
##               Df Sum Sq Mean Sq F value Pr(>F)
## category       2   0.15 0.07318   2.832  0.059 .
## Residuals   6494 167.79 0.02584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "sulphates"
##               Df Sum Sq Mean Sq F value  Pr(>F)
## category       2   0.25 0.12739   5.761 0.00316 **
## Residuals   6494 143.59 0.02211
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "alcohol"
##               Df Sum Sq Mean Sq F value Pr(>F)
## category       2   2069  1034.7   936.9 <2e-16 ***
## Residuals   6494   7172     1.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

All properties except for the pH affect the quality of the wine.

b) Which of the chemical properties are related with type of the wines ?

```
for (i in 1:11){
  print(colnames(winequal)[i])
  print(summary(aov(winequal[,i]~type,data=winequal)))
}
```

```
## [1] "fixed.acidity"
##               Df Sum Sq Mean Sq F value Pr(>F)
## type           1   2587  2586.7    2017 <2e-16 ***
## Residuals   6495   8331     1.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "volatile.acidity"
##               Df Sum Sq Mean Sq F value Pr(>F)
## type           1  75.09   75.09    4829 <2e-16 ***
## Residuals   6495 100.99    0.02
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "citric.acid"
##               Df Sum Sq Mean Sq F value Pr(>F)
## type           1   4.82   4.817   236.4 <2e-16 ***
```

```
## Residuals   6495 132.36   0.020
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "residual.sugar"
##              Df Sum Sq Mean Sq F value Pr(>F)
## type          1  17892   17892   899.8 <2e-16 ***
## Residuals   6495 129156      20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "chlorides"
##              Df Sum Sq Mean Sq F value Pr(>F)
## type          1  2.096  2.0956    2316 <2e-16 ***
## Residuals   6495  5.877  0.0009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "free.sulfur.dioxide"
##              Df  Sum Sq Mean Sq F value Pr(>F)
## type          1  455241  455241    1858 <2e-16 ***
## Residuals   6495 1591267     245
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "total.sulfur.dioxide"
##              Df   Sum Sq  Mean Sq F value Pr(>F)
## type          1 10179301 10179301    6253 <2e-16 ***
## Residuals   6495 10573600     1628
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "density"
##              Df  Sum Sq  Mean Sq F value Pr(>F)
## type          1 0.00891 0.008914    1170 <2e-16 ***
## Residuals   6495 0.04950 0.000008
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "pH"
##              Df Sum Sq Mean Sq F value Pr(>F)
## type          1  18.19  18.192     789 <2e-16 ***
## Residuals   6495 149.75   0.023
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "sulphates"
##              Df Sum Sq Mean Sq F value Pr(>F)
## type          1  34.15   34.15    2022 <2e-16 ***
## Residuals   6495 109.70    0.02
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## [1] "alcohol"
##              Df Sum Sq Mean Sq F value  Pr(>F)
## type          1     10  10.045   7.068 0.00787 **
## Residuals   6495   9231   1.421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the ANOVA test, all p-values of each property are below the acceptance area, hence all properties are directly correlated with the type of wine.

c) How does type and quality of wines affect (separately and together) the percentage of alcohol present in the wine ?

```
print(summary(aov(winequal$alcohol~category,data=winequal)))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## category       2   2069  1034.7   936.9 <2e-16 ***
## Residuals   6494   7172     1.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
print(summary(aov(winequal$alcohol~type,data=winequal)))
```

```
##               Df Sum Sq Mean Sq F value  Pr(>F)
## type           1     10  10.045   7.068 0.00787 **
## Residuals   6495   9231   1.421
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

For the ANOVA test, the p-value for the category $< 2e-16$ is smaller than the p-value for the type 0.00787 so the **category** has a bigger impact on the quantity of alcohol of the wine.

```
print(summary(aov(winequal$alcohol~category+type,data=winequal)))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## category       2   2069  1034.7 937.567 <2e-16 ***
## type           1      6     6.0   5.458 0.0195 *
## Residuals   6493   7166     1.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The two-way ANOVA shows that the alcohol mean of the wine is still affected by both the category and the type, altough the category has a greater impact on the amount of alcohol of the wine.

d) Detail the results of a two-way ANOVA considering as dependent variable "fixed acidity", and independent variable "type" and "quality".

```
print(summary(aov(winequal$fixed.acidity~category+type,data=winequal)))
```

```
##               Df Sum Sq Mean Sq F value   Pr(>F)
## category       2     57    28.5   22.18 2.51e-10 ***
## type           1   2531  2531.1 1972.91  < 2e-16 ***
## Residuals   6493   8330     1.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the two-way ANOVA table we can conclude that both category and type are statistically significant. The category is the most significant factor variable. These results would lead us to believe that changing the category or the quality of the wine, will impact significantly the mean of the acidity level.

Not the above fitted model is called *additive model*. It makes an assumption that the two factor variables are independent.

Third Question: Define a linear model for an athlete in the 1500m

Arnau Abella

24/03/2020

Picking the best model

What is the linear expression that better predicts the behaviour of an athlete of 1500m ?

```
# Load the dataset and preprocess it.
library(FactoMineR)
data(decathlon)
head(decathlon)
colnames(decathlon)[c(1,5,6,10)]<-c("x100m","x400m","x110m.hurdle","x1500m")
colnames(decathlon)
```

Let's construct some simple linear regression models and check which better predicts the behaviour:

```
reg1<-lm(x1500m~x100m,data=decathlon)
summary(reg1)
```

```
##
## Call:
## lm(formula = x1500m ~ x100m, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -16.005  -9.105  -1.706   5.624  37.604
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  308.578     78.038   3.954 0.000314 ***
## x100m         -2.687      7.094  -0.379 0.706885
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.8 on 39 degrees of freedom
## Multiple R-squared:  0.003666,   Adjusted R-squared:  -0.02188
## F-statistic: 0.1435 on 1 and 39 DF,  p-value: 0.7069
```

```
reg2<-lm(x1500m~x110m.hurdle,data=decathlon)
summary(reg2)
```

```
##
## Call:
## lm(formula = x1500m ~ x110m.hurdle, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.226  -7.804  -0.702   5.653  37.646
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  265.4584    57.8566   4.588 4.55e-05 ***
## x110m.hurdle   0.9288     3.9592   0.235    0.816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.81 on 39 degrees of freedom
## Multiple R-squared:  0.001409,   Adjusted R-squared:  -0.0242
## F-statistic: 0.05504 on 1 and 39 DF,  p-value: 0.8157
```

```
reg3<-lm(x1500m~x400m,data=decathlon)
summary(reg3)
```

```
##
## Call:
## lm(formula = x1500m ~ x400m, data = decathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.0877  -6.9098  -0.7062   4.7360  31.5996
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   74.102     73.424   1.009  0.31909
## x400m          4.130      1.479   2.792  0.00808 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.79 on 39 degrees of freedom
## Multiple R-squared:  0.1666, Adjusted R-squared:  0.1452
## F-statistic: 7.793 on 1 and 39 DF,  p-value: 0.008078
```

We are going to use the third model **x1500~x400m** because it has smaller residual standard error, larger R^2 (better fit) and better F-statistic.

Correlation Tests

Only in the third model the coefficient of correlation 0.408 is significant ($p < 0.05$)

```
# In all cases the coefficient of correlation is 0.816 and significant (p=0.002).
cor.test(decathlon$x100m       , decathlon$x1500m)
```

```
##
##  Pearson's product-moment correlation
##
## data:  decathlon$x100m and decathlon$x1500m
## t = -0.37881, df = 39, p-value = 0.7069
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3614639  0.2517942
## sample estimates:
##         cor
## -0.06054645
```

```
cor.test(decathlon$x110m.hurdle, decathlon$x1500m)
```

```
##
```

```
##  Pearson's product-moment correlation
##
## data:  decathlon$x110m.hurdle and decathlon$x1500m
## t = 0.2346, df = 39, p-value = 0.8157
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.2732662  0.3412495
## sample estimates:
##        cor
## 0.03754024
```

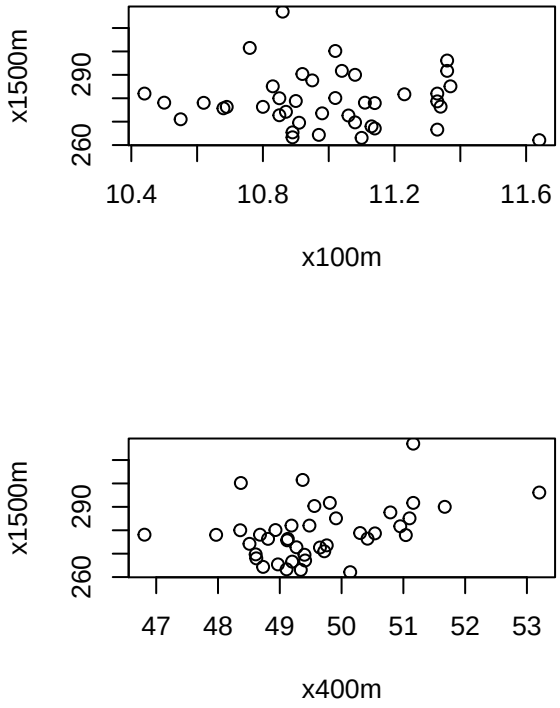
```
cor.test(decathlon$x400m       , decathlon$x1500m)
```

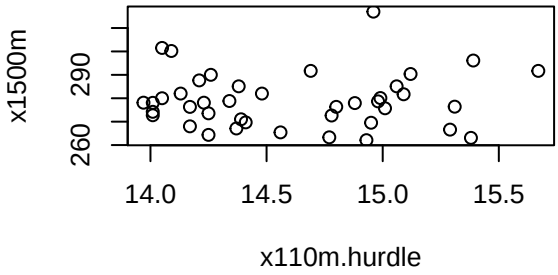
```
##
##  Pearson's product-moment correlation
##
## data:  decathlon$x400m and decathlon$x1500m
## t = 2.7917, df = 39, p-value = 0.008078
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1148796 0.6359151
## sample estimates:
##       cor
## 0.4081064
```

Scatterplots

The first and the second model are totally scattered but on the third model we can appreciate a positive correlation.

```
op<-par(mfrow=c(2,2))
plot(x1500m~x100m       , data=decathlon)
plot(x1500m~x110m.hurdle, data=decathlon)
plot(x1500m~x400m       , data=decathlon)
par(op)
```



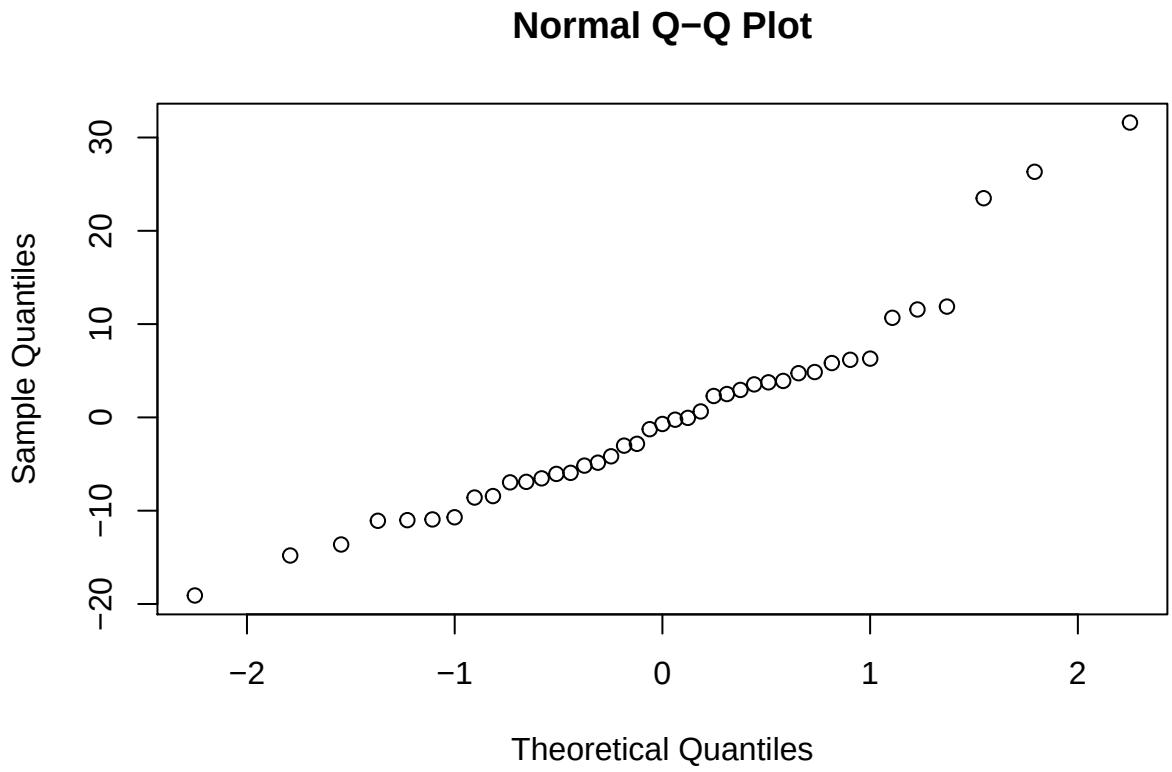


Testing assumptions of the linear model

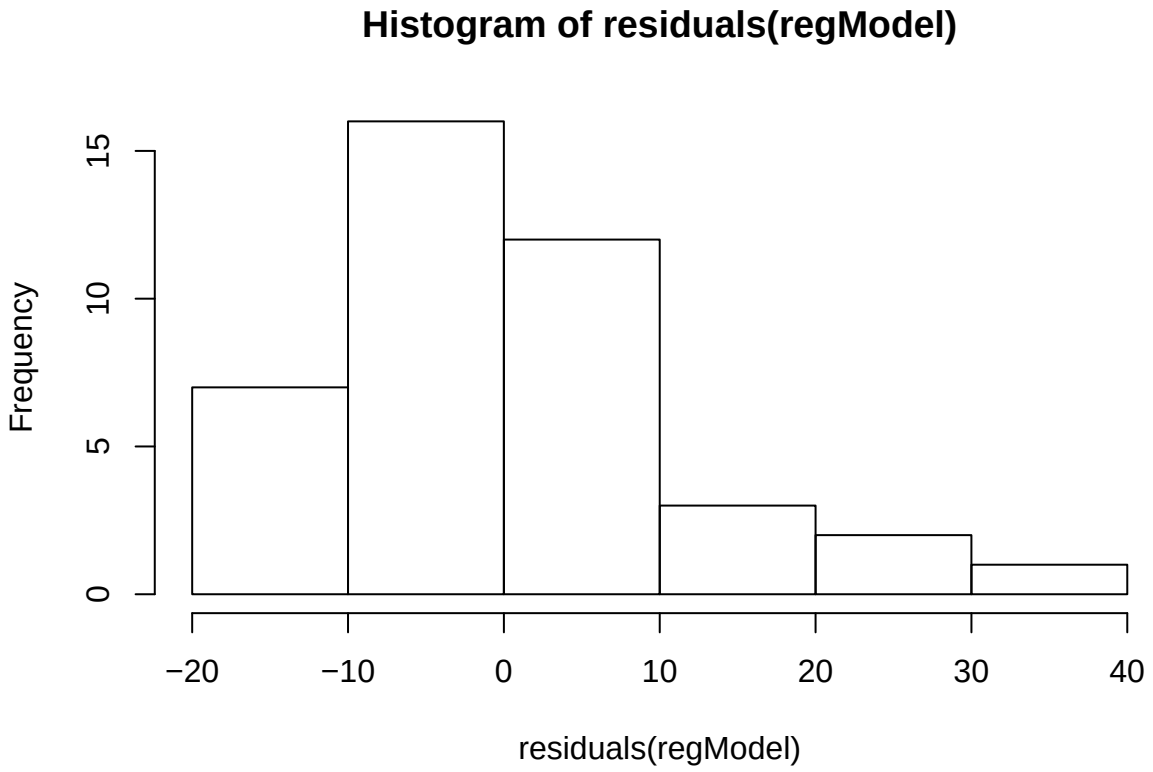
```
regModel <-lm(x1500m~x400m, data=decathlon)
summary(regModel)
```

Normality of the Error Term

```
# QQPlot
qqnorm(residuals(regModel))
```




```
# Since the values are taking part close to the diagonal,
#   the distribution is approximately normal.

# Histogram
hist(residuals(regModel))
```



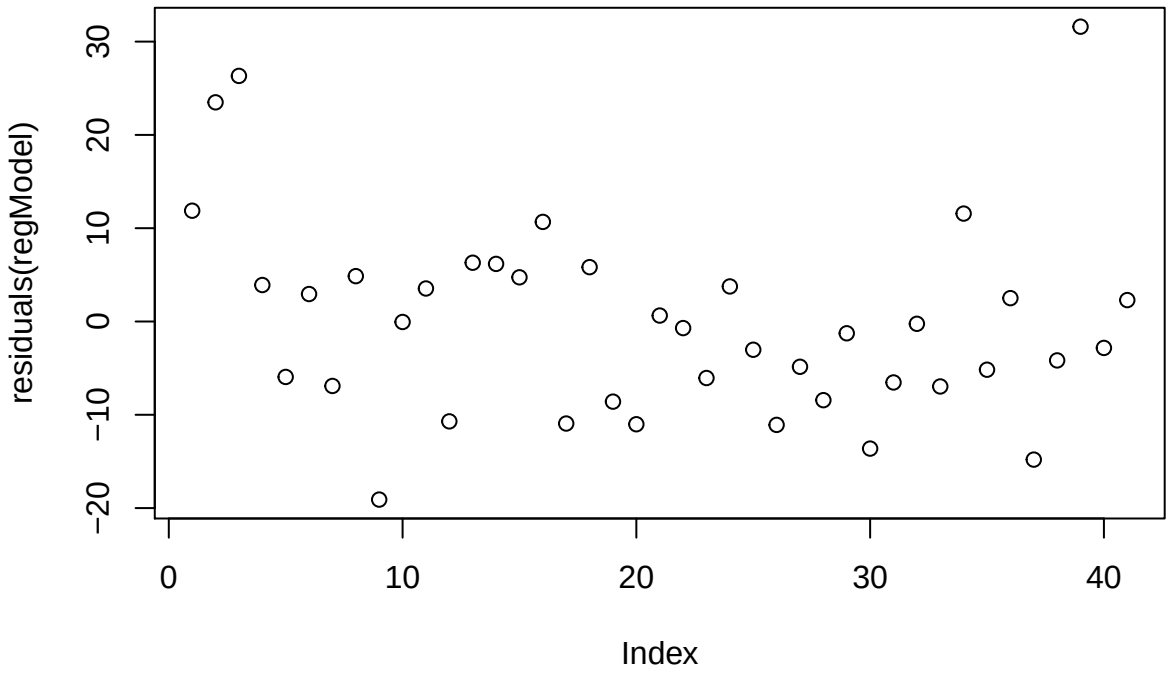

```
# It is approximately normal (skew to the left).

# Shapiro Wilks Test
shapiro.test(residuals(regModel))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(regModel)
## W = 0.93244, p-value = 0.01742
```

```
# The error term doesn't follows a Normal distribution. (p<0.05)
# This should be taken into consideration.
```

Homogenity of Variance

```
# Residual Analysis #
plot(residuals(regModel))
```




```
# Residuals have a rectangular pattern around the zero mean.
# There is no violation of this assumption.

##Breusch Pagan Test
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
bptest(regModel)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  regModel
## BP = 0.0010727, df = 1, p-value = 0.9739
```

```
# H0 is accepted (p>0.05). Hence, the homogenity of variances is provided.
```

The independence of errors

```
# Durbin-Watson Test
dwtest(regModel, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  regModel
## DW = 1.7274, p-value = 0.3458
```

```
## alternative hypothesis: true autocorrelation is not 0
```

```
# There is not an autocorrelaiton in the data set (p>0.05).
# The errors/observations are independent.
```

Predicting new values

Is the model accurate ? What do you expect ?

The F test shows that the model is significant ($p<0.05$).

```
summary(regModel)
```

```
##
## Call:
## lm(formula = x1500m ~ x400m, data = decathlon)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.0877  -6.9098  -0.7062   4.7360  31.5996
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   74.102     73.424   1.009  0.31909
## x400m          4.130      1.479   2.792  0.00808 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.79 on 39 degrees of freedom
## Multiple R-squared:  0.1666, Adjusted R-squared:  0.1452
## F-statistic: 7.793 on 1 and 39 DF,  p-value: 0.008078
```

```
confint(regModel)
```

```
##                   2.5 %     97.5 %
## (Intercept) -74.412562 222.616246
## x400m         1.137685   7.122619
```

```
# The null hypothesis is H0:B1=0.
# If the confidence interval includes 0 => we accept the null hypothesis.
#
# (1.137685, 7.122619) the confidence intervals of the parameters does not include 0.
#      => The null hypothesis B0=0 and B1=0 are rejected.
#
# Therefore the coefficients are significant.
```

Let's predict the behaviour of an athlete in the 1500m that runned the 400m in 55.5 seconds.

```
new=data.frame(x400m=55.5)
predict.lm(regModel, newdata=new, interval="prediction")
```

```
##        fit      lwr      upr
## 1 303.3253 275.0733 331.5773
```

The model predicted that the athlete would run the 1500m in between (303.32, 331.57) seconds with a high probability.

Fourth Question: Working With Real Data

Arnau Abella

03/04/2020

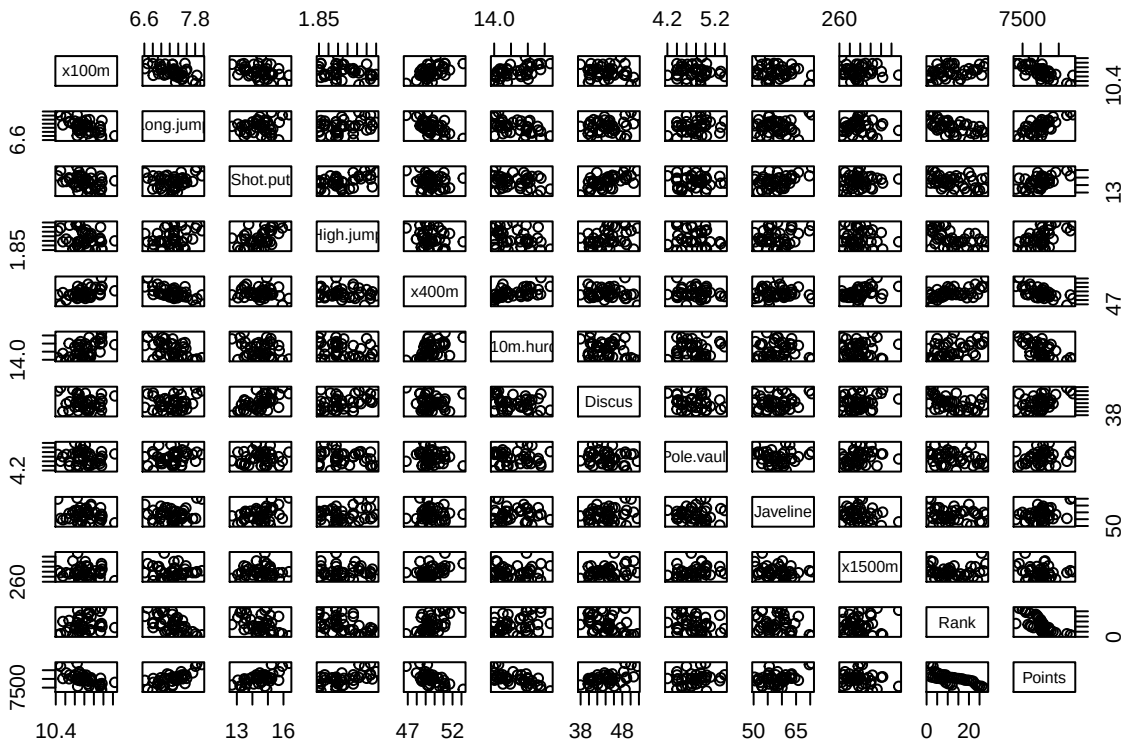
Decathlon Dataset

It is easy to see from the Variables factor map of the PCA that the points is inversly proportional to the rank i.e. the more points you get the lower rank you achieve (lower rank is better).__

From the chart we also see that either the variable has an effect to the rank or to the points, which, at the end of the day, is equivalent. Notice that some results do not have the same impact on the puntuation/rank such as 1500m.

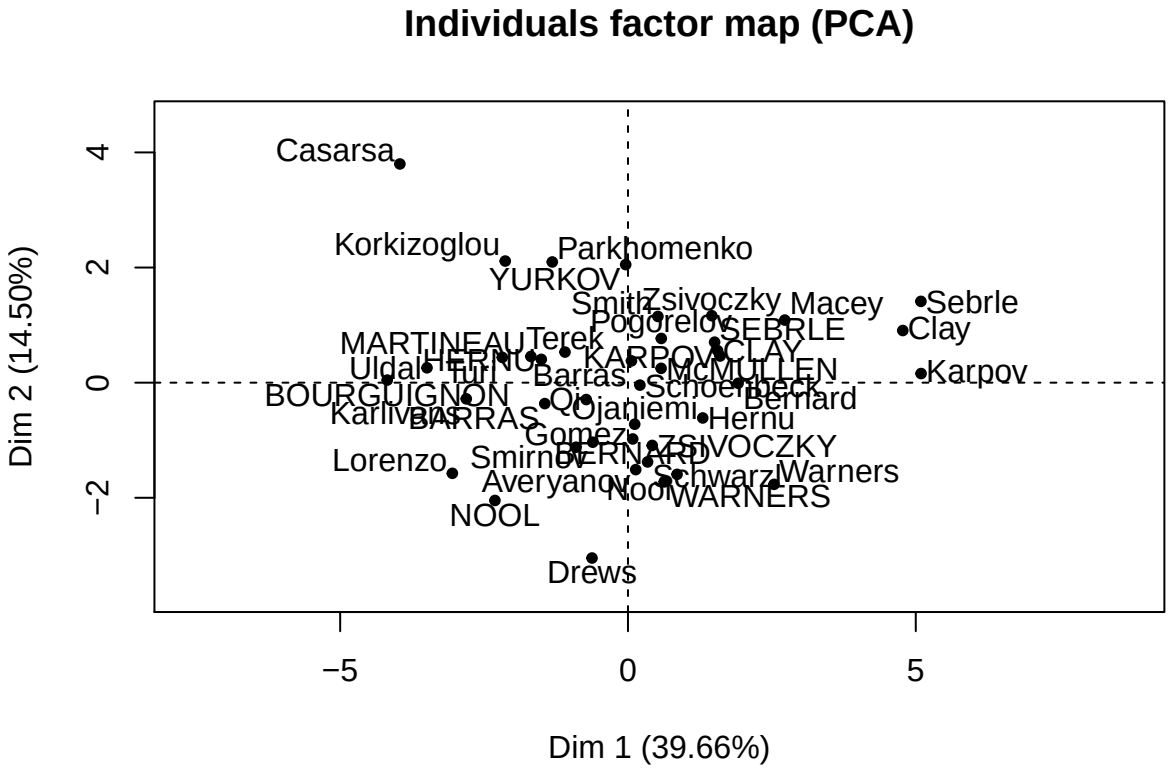
```
# Load the dataset and preprocess it.
data(decathlon)
colnames(decathlon)[c(1,5,6,10)]<-c("x100m","x400m","x110m.hurdle","x1500m")
colnames(decathlon)
```

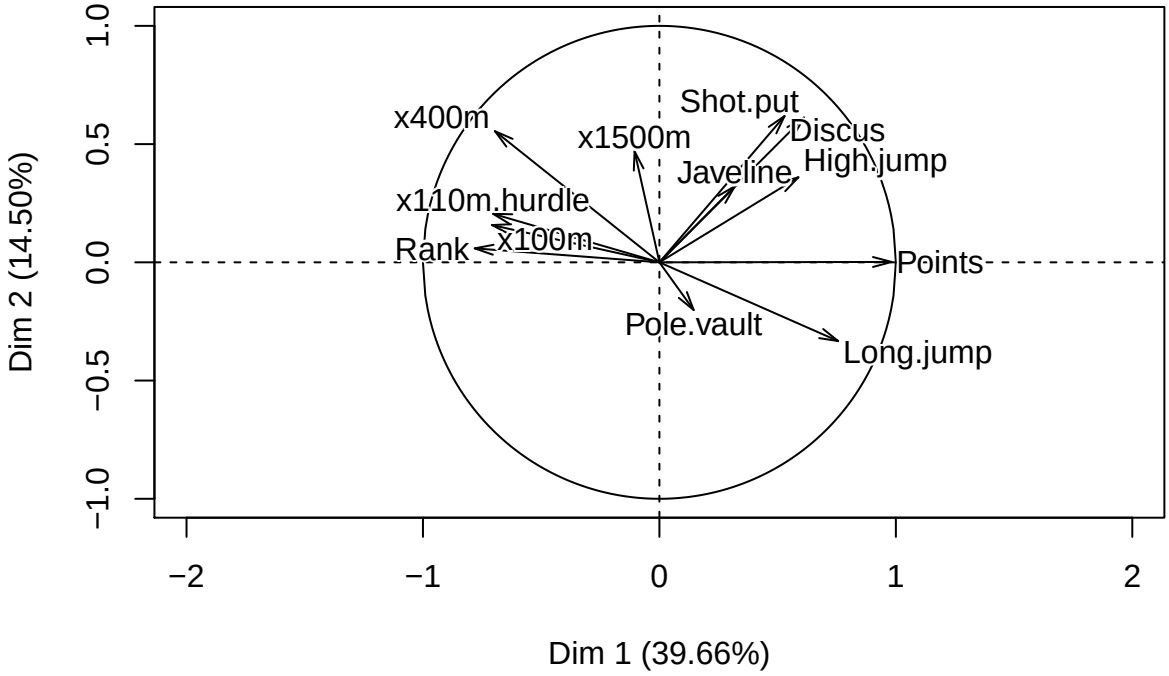
```
competition <- which(colnames(decathlon) == "Competition")
plot(decathlon[,-c(competition)])
```




```
# cor(decathlon[,-c(competition)])
```

```r
# Remove the dependent variable score.
pca<-PCA(decathlon[,-c(competition)])
```

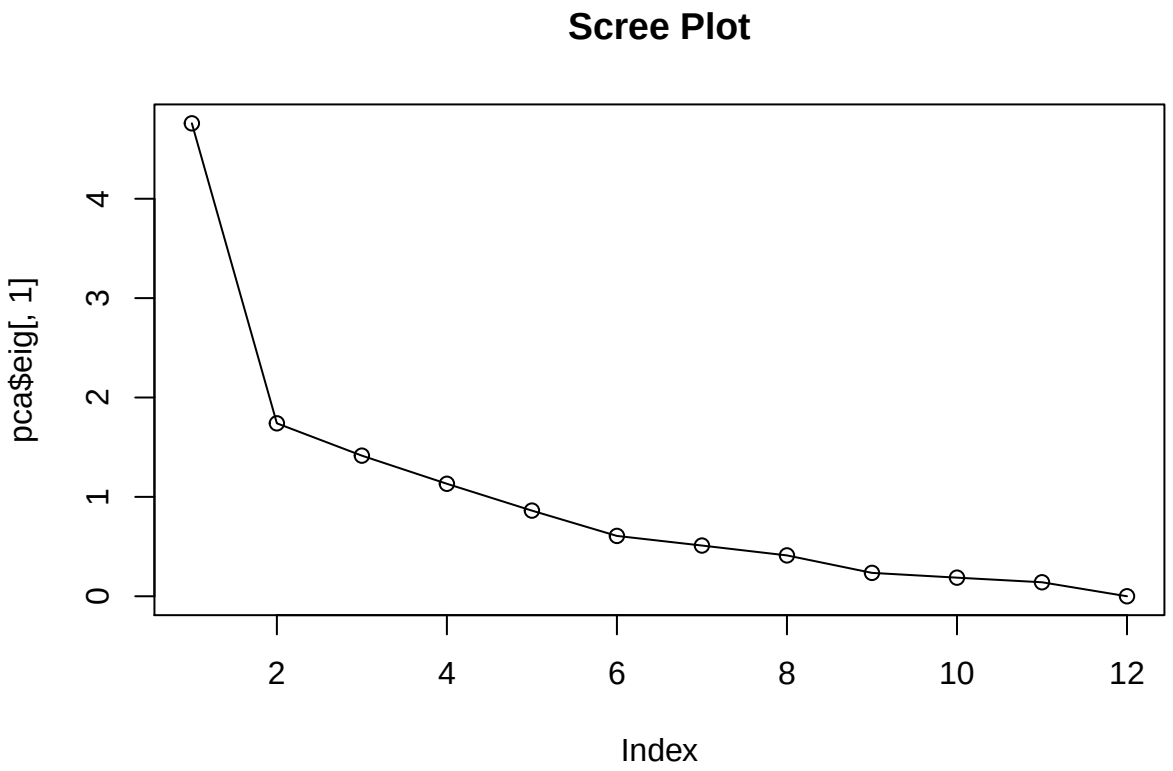




Principal Component Regression

From the cummulative percentage of variance, we need at least 4 principal components to have an accumulative variance $\geq \frac{2}{3}$.

```
library(FactoMineR)
competition <- which(colnames(decathlon) == "Competition")

pca$eig
plot(pca$eig[,1], type="o", main="Scree Plot")
```




```
summary(pca)

decathlon$PC1<-pca$ind$coord[,1]
decathlon$PC2<-pca$ind$coord[,2]
decathlon$PC3<-pca$ind$coord[,3]
decathlon$PC4<-pca$ind$coord[,4]
reg_pc<-lm(Points~PC1 + PC2 + PC3 + PC4, data=decathlon)
```

```
summary(reg_pc)
```

```
##
## Call:
## lm(formula = Points ~ PC1 + PC2 + PC3 + PC4, data = decathlon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -77.522 -35.990   5.294  33.767  89.454
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
```

```
## (Intercept) 8005.3659     6.9336 1154.574  < 2e-16 ***
## PC1          152.1889     3.1784   47.882  < 2e-16 ***
## PC2            0.3998     5.2561    0.076  0.93979
## PC3          -17.8926     5.8290   -3.070  0.00406 **
## PC4           41.6555     6.5175    6.391 2.09e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 44.4 on 36 degrees of freedom
## Multiple R-squared:  0.9849, Adjusted R-squared:  0.9832
## F-statistic: 585.7 on 4 and 36 DF,  p-value: < 2.2e-16
```

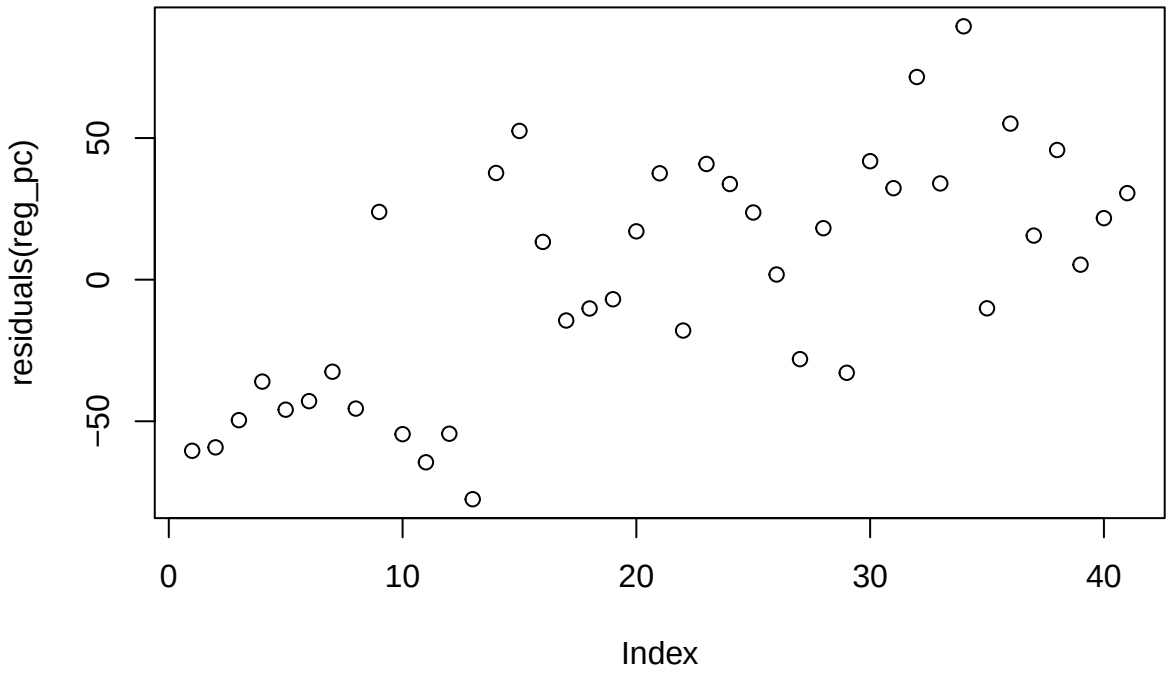
Testing the assumptions of the regression model

- Normality: the error term does follow a normal distribution which is desired.
- Homogenity: the variance is homogeneous.
- Independence of errors: the errors do have correlation which may affect the results.

```
# Normality
shapiro.test(residuals(reg_pc))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(reg_pc)
## W = 0.96366, p-value = 0.2108
```

```
# Homogenity
plot(residuals(reg_pc))
```




```
bptest(reg_pc)
```

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

```
## data:  reg_pc
## BP = 13.223, df = 4, p-value = 0.01024
```

```
# Independence of errors
dwtest(reg_pc, alternative = "two.sided")
```

```
##
##  Durbin-Watson test
##
## data:  reg_pc
## DW = 1.0195, p-value = 0.0004396
## alternative hypothesis: true autocorrelation is not 0
```

Predicting the points of an athelete using the regression model

Nota bene, the RMSE is small compared to the points scale so we can conclude that the model is accurate "enough".

```
n <- nrow(decathlon)
train.sample <- sample(1:n, round(0.67*n))

train.set <- decathlon[train.sample, ]
test.set <- decathlon[-train.sample, ]

train.model <- lm(Points ~ PC1+PC2+PC3+PC4 , data = train.set)
summary(train.model)
```

```
##
## Call:
## lm(formula = Points ~ PC1 + PC2 + PC3 + PC4, data = train.set)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -60.83 -33.70 -13.57  34.43  65.69
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7998.1407     8.5430 936.223  < 2e-16 ***
## PC1          152.0412     4.4297  34.323  < 2e-16 ***
## PC2            0.5286     8.0539   0.066  0.94826
## PC3          -25.6946     7.0097  -3.666  0.00136 **
## PC4           42.7321     7.4789   5.714 9.54e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 43.97 on 22 degrees of freedom
## Multiple R-squared:  0.9826, Adjusted R-squared:  0.9795
## F-statistic:   311 on 4 and 22 DF,  p-value: < 2.2e-16
```

```
yhat<-predict(train.model, test.set, interval="prediction")
yhat
```

```
##                fit      lwr      upr
## MARTINEAU 7776.188 7680.956 7871.420
## NOOL      7708.842 7604.981 7812.704
## Sebrle    8851.531 8741.777 8961.286
```

```
## Karpov    8702.490 8594.922 8810.058
## Warners   8338.320 8236.283 8440.358
## Zsivoczky 8298.769 8199.168 8398.369
## Schwarzl  8047.820 7950.861 8144.778
## Smith     8055.547 7954.041 8157.053
## Ojaniemi  8035.316 7941.154 8129.478
## Qi        7902.937 7807.772 7998.101
## Drews     7838.609 7729.806 7947.412
## Terek     7779.868 7682.123 7877.613
## Lorenzo   7582.013 7478.281 7685.745
## Casarsa   7366.519 7248.579 7484.459
```

```
y<-test.set$score

error<-cbind(yhat[,1,drop=FALSE],y,(yhat[,1]-y)^2)
sqr_err<-error[,1]
sse<-sum(sqr_err)

# Root Mean Square Error = sqrt(SSE/N)
RMSE<-sqrt(sse/(nrow(test.set)))
RMSE
```

```
## [1] 89.55635
```