SMDE SECOND ASSIGMENT

Balbina Virgili Rocosa

27th November 2017

FIRST QUESTION: DEFINE YOUR RNG (OPTIONAL)

The main objective of this first exercise is to choose a RNG, implement it and, finally, test the correctness of the implemented RNG.

A random number generator is a computational device designed to generate a sequence of numbers or symbols that cannot be reasonably predicted better than by a random chance. There are several types of RNGs, the one that has been chosen for this exercise is a Congruential Random Number Generator.

Congruential generators are based on one of the oldest and best-known algorithms, which is based on congruence and linear function. The one chosen here is a type of Congruential generator which is known as the Park–Miller random number generator. The main characteristic of it is that it always set the incremental parameter to 0.

To implement it, the Java code found on the wiki has been adapted to R:

```
library(randtests)
```

```
#Function implemented in R
executeCongruentialRNG <- function(number=10, multiplier=16807, increment=0,seed=3793){
  #Variables initialization
  x2 <- 0
  x1 <- seed
  #Create empty outputFrame by default
  output <- c()

  #Iterate as many times as the value of number is defined
  for(i in number:1){
    #Algorithm implementation for each iteration
    x2 <- (multiplier*x1 + increment)%%((2^31)-1)
    calculated <- x2 / ((2^31)-1)
    output <- c(output,calculated)
    x1 <- x2
  }
  #return all value calculated
  return(output)

}
```

As it can be seen on the implemented code, that some parameters are needed to achieve a correct performance. These parameters are:

- Number: define the amount of generated values
- Multiplier: multiplier factor
- Increment: it is set to 0 to improve the performance of the test
- Seed: set the initial value

The default values for each of the parameters have been determined using the first parametrization of the wiki, as it is known that are desired values because it had already passed several random number tests.

To be able to check the correct performance of the implemented RNG, first, 100 values are desired to be created.

```
#Execute the RNG function created
example = executeCongruentialRNG(100)
```

Finally, several tests of correctness must be passed to prove the validity of the implemented RNG, both of them measure the randomness of the generated values.

1-. Randomness test

It is possible to test the randomness of the response with Mann-Kendall Rank test. If none alternative attribute is defined, the alternative hypothesis by default is 'two.sided'.

```
#Mann-Kendall Rank test
rank.test(example)
```

```
##
##  Mann-Kendall Rank Test
##
## data:  example
## statistic = 0.11912, n = 100, p-value = 0.9052
## alternative hypothesis: trend
```

The calculated p-value is higher than the significance level 0.05, so the alternative hypothesis can be accepted So, there is no indication that randomness is violated.

2-. Randomness test

It is possible to test the randomness of the response with Bartels rank test of randomness. If none alternative attribute is defined, the alternative hypothesis by default is 'two.sided'.

```
#Bartels rank test
bartels.rank.test(example)
```

```
##
##  Bartels Ratio Test
##
## data:  example
## statistic = 2.6029, n = 100, p-value = 0.009245
## alternative hypothesis: nonrandomness
```

The calculated p-value is lower than the significance level 0.05, so the alternative hypothesis cannot be accepted. In other words, there is indication that nonrandomness is violated.

SECOND QUESTION: SIMULATE YOUR DATA

In this exercise, a dataset with the structure specified is generated. To do it, first of all, a distribution has been defined for each of the first five factors.

The different factors defined are the following ones:

- **Factor 1:** Normal distribution with mean 0 and sd 1
- **Factor 2:** Uniform distribution with min 0 and max 12
- **Factor 3:** Exponential distribution with rate 1
- **Factor 4:** Normal distribution with mean 3 and sd 3
- **Factor 5:** Uniform distribution with min 6 and max 25

Then, a different combination of the already defined factors has been determined for each of the other five factors.

- **Factor 6:** 2Factor1 + Factor3
- **Factor 7:** Factor4 + Factor1
- **Factor 8:** Factor3 + 3Factor4 + 2Factor5
- **Factor 9:** 2Factor2 + Factor5
- **Factor 10:** 3Factor2 + Factor3 + Factor1

As it is shown below, all factors defined are created using R.

```
#Generation of the different factors as defined
factor1 <- rnorm(100, mean=0, sd=1)
factor2 <- runif(100, min=0, max=12)
factor3 <- rexp(100, rate=1)
factor4 <- rnorm(100, mean=3, sd=3)
factor5 <- runif(100, min=6, max=25)

#Generation of the factors as a combination of the previous ones
factor6 <- (2*factor1) + factor3
factor7 <- factor4 + factor1
factor8 <- factor3 + (3*factor4) + (2*factor5)
factor9 <- (2*factor2) + factor5
factor10 <- (3*factor2) + factor3 + factor1
```

Now, it is time to define the answer variable which must be composed as a subset of the previous factors plus a known normal distribution which add random noise.

- **Answer:** Factor9 + Factor8 - 2Factor3 + 2Factor5 + Normal distribution with mean 3 and sd 1

NOTE: Taking into account the factor combinations previously defined, this answer variable could be also written as: *Answer= F5 + 2F2 + 3F4 - F3 + Normal distribution with mean 3 and sd 1*

It is also created using R.

```
#Generation of the answer variable
answer <- factor9 + factor8 - (2*factor3) - (2*factor5) + rnorm(100, mean=1, sd=1)
```

Afterwards, a table with all factors and the answer variable defined has been created using R in order to obtain the table specified.

```
#Generation of the dataset
data <- data.frame(factor1, factor2, factor3, factor4, factor5, factor6,
                   factor7, factor8, factor9, factor10, answer)
```

THIRTH QUESTION: OBTAIN AN EXPRESSION TO GENERATE NEW DATA

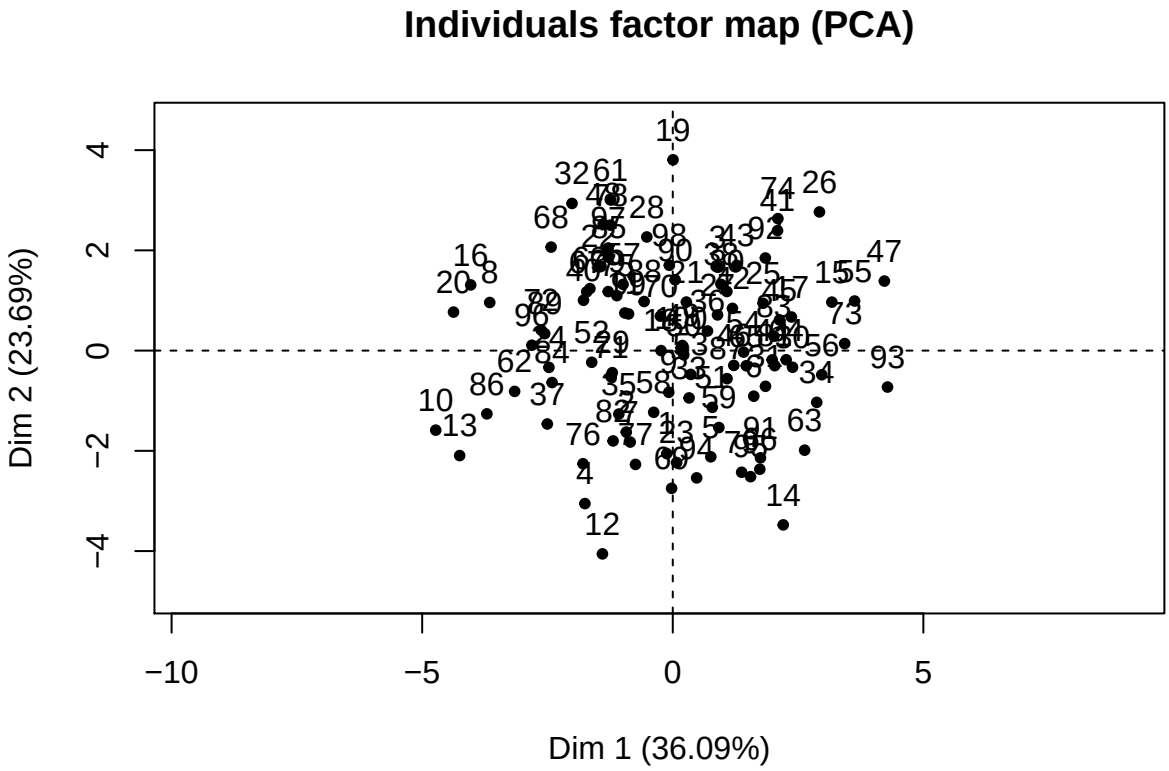
In this exercise, the dataset generated needs to be explored in order to, finally, be able to define an expression to obtain new data.

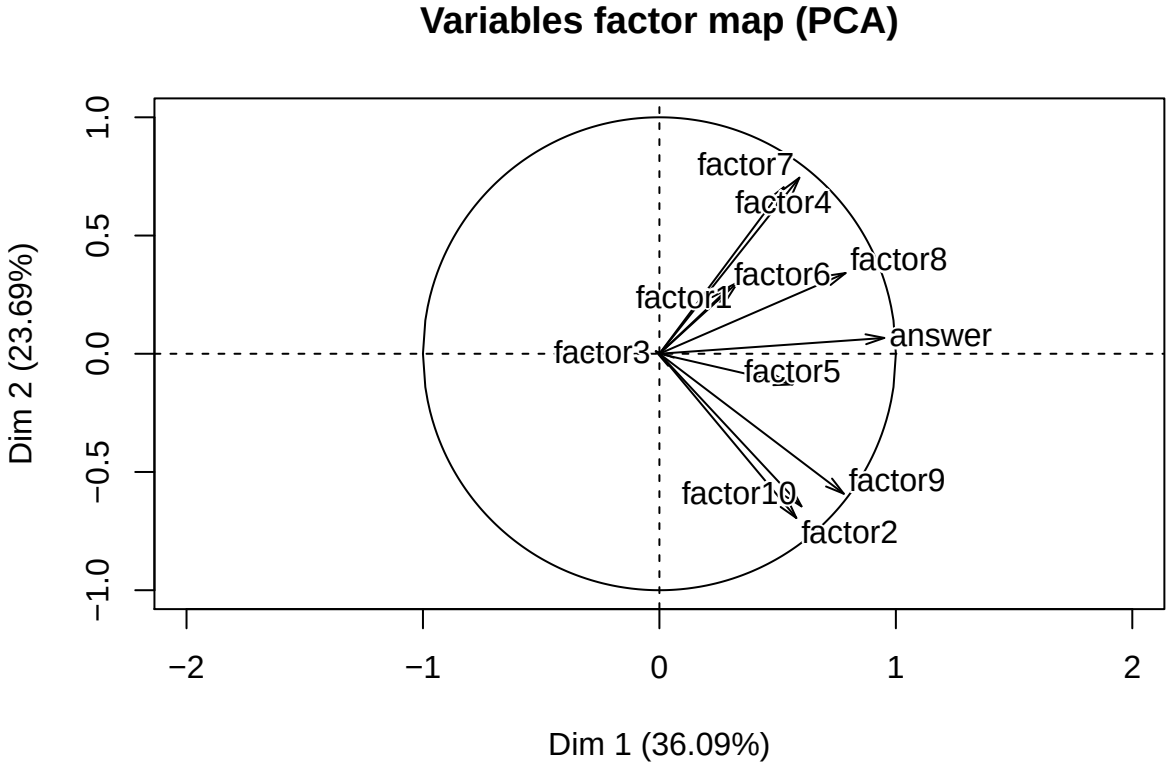
So, first of all, the different relations and interactions between each factors need to be explored. To do it, the PCA analysis will be used to describe the main variables that exists on the dataset created. As we already saw in the last assignment, the PCA analysis is useful to see if there are linear relationships between variables. So, for the analysis, all variables will be used as active variables because, at this point, it is supposed to know nothing about the dataset.

Before performing it, it can be easily realized that PCA assumptions are fulfilled, since all the data is numeric and the number of rows of the dataset is bigger than the number of columns.

```r
#FactoMineR needs to be loaded to be able to perform PCA
library("FactoMineR")
```

```r
#PCA analysis
pcaAnalysis = PCA(data, scale.unit=TRUE,graph=T)
```





With the results obtained, it can be determined all the relationships between the different factors of the dataset created during the previous exercise. And as we can see with the plot printed, we can also assume that the first two dimensions resume almost the 63% of the total variance of the dataset. Analyzing the data obtained, it can be assumed that there two different groups of positive correlated variables. The first one is F4, F7, F5, F8 and F1. The second one is F5, F9, F10 and F2. Also, answer seems to be positive correlated by all the factors (specially F1, F5 and F8), except F3, with which has a negative correlation.

So, at this moment, it is not easy to specify the real relation between all factors and the answer variable because the dataset has too many correlated dimensions to determine the behavior just analyzing the retrieved PCA graphic.

To continue exploring the possible relations between all variables of the dataset, several linear models will be performed.

The first linear model test performed analyze the relation of answer against all the other categories of the dataset.

```
#Generate the linear model of answer distribution
lRegression1 <- lm( answer  ~ ., data=data)
```

```
##
## Call:
## lm(formula = answer ~ ., data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.12620 -0.69170 -0.03608  0.85535  2.85856
##
## Coefficients: (5 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.62879    0.48219   1.304    0.195
## factor1     -0.13281    0.11419  -1.163    0.248
## factor2      2.02538    0.03817  53.061  < 2e-16 ***
## factor3     -0.98192    0.13654  -7.192 1.52e-10 ***
## factor4      3.00774    0.04257  70.661  < 2e-16 ***
## factor5      0.99971    0.02262  44.191  < 2e-16 ***
## factor6           NA         NA      NA       NA
## factor7           NA         NA      NA       NA
## factor8           NA         NA      NA       NA
## factor9           NA         NA      NA       NA
## factor10          NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.225 on 94 degrees of freedom
## Multiple R-squared:  0.9904, Adjusted R-squared:  0.9898
## F-statistic:  1930 on 5 and 94 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that from factors 6 to 10 a NA value is defined. This value NA indicates that each of these factors are linear combinations of the other factors, in other words, their behavior can be determined from different combinations of the other factors used. So, more tests will need to be performed to determine their relation. Moreover, the factors 2,3,4 and 5 have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we can determine that factor2, factor3, factor4 and factor5 have a relation with the answer distribution. Otherwise, the results show that factor1 is not much related with answer distribution so it can be discarded from the linear model as well as the factors which are linear combinations.

Before performing the new linear model with the discarded variables, the relation between each factor wants to be determined. To achieve it, a linear model for each of the factors which retrieved a NA value is performed.

```
#Generate the linear model of factor6 distribution
lRegression2 <- lm(formula =  factor6  ~  factor1 + factor2  +  factor3  + factor4 + factor5,
                   data = data)
```

```
##
## Call:
## lm(formula = factor6 ~ factor1 + factor2 + factor3 + factor4 +
##     factor5, data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.380e-14 -2.675e-15 -1.639e-16  2.318e-15  1.298e-14
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  1.599e-15  2.048e-15  7.810e-01    0.437
## factor1      2.000e+00  4.851e-16  4.123e+15   <2e-16 ***
## factor2     -2.093e-16  1.621e-16 -1.291e+00    0.200
## factor3      1.000e+00  5.800e-16  1.724e+15   <2e-16 ***
## factor4     -1.503e-16  1.808e-16 -8.310e-01    0.408
## factor5      3.393e-17  9.610e-17  3.530e-01    0.725
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.204e-15 on 94 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.225e+30 on 5 and 94 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that factor1 and factor3 have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we could determine that this two categories have a relation with the factor6 distribution. In this case, the R-squared value is 1 so the probability to predict the factor6 distribution in this calculated linear model can be considered as very good one. So it can be determined that:

Factor6: 2.0Factor1 + 1.0Factor3

```
#Generate the linear model of factor7 distribution
lRegression3 <- lm( formula =  factor7  ~  factor1 + factor2  +  factor3  + factor4 + factor5,
                    data = data)
```

```
##
## Call:
## lm(formula = factor7 ~ factor1 + factor2 + factor3 + factor4 +
##     factor5, data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.823e-14 -2.855e-15  5.757e-16  2.265e-15  1.354e-14
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -4.441e-17  2.071e-15 -2.100e-02    0.983
## factor1      1.000e+00  4.905e-16  2.039e+15   <2e-16 ***
```

```
## factor2     -2.196e-16  1.640e-16 -1.339e+00    0.184
## factor3     -7.583e-19  5.865e-16 -1.000e-03    0.999
## factor4      1.000e+00  1.828e-16  5.470e+15   <2e-16 ***
## factor5     -5.701e-19  9.717e-17 -6.000e-03    0.995
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.262e-15 on 94 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.305e+30 on 5 and 94 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that factor1 and factor4 have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we could determine that this two categories have a relation with the factor6 distribution. In this case, the R-squared value is 1 so the probability to predict the factor6 distribution in this calculated linear model can be considered as very good one. So it can be determined that:

Factor7: 1.0Factor1 + 1.0Factor4

```
#Generate the linear model of factor8 distribution
lRegression4 <- lm( formula =  factor8  ~  factor1 + factor2  +  factor3  + factor4 + factor5,
                    data = data)
```

```
##
## Call:
## lm(formula = factor8 ~ factor1 + factor2 + factor3 + factor4 +
##     factor5, data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.486e-13 -4.274e-14  1.424e-15  3.541e-14  1.804e-13
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -4.031e-15  2.503e-14 -1.610e-01    0.872
## factor1      4.231e-15  5.928e-15  7.140e-01    0.477
## factor2      2.426e-15  1.982e-15  1.224e+00    0.224
## factor3      1.000e+00  7.088e-15  1.411e+14   <2e-16 ***
## factor4      3.000e+00  2.210e-15  1.358e+15   <2e-16 ***
## factor5      2.000e+00  1.174e-15  1.703e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.36e-14 on 94 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 9.32e+29 on 5 and 94 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that factor3, factor4 and factor5 have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we could determine that this two categories have a relation with the factor6 distribution. In this case, the R-squared value is 1 so the probability to predict the factor6 distribution in this calculated linear model can be considered as very good one. So it can be determined that:

Factor8: 1.0Factor3 + 3.0Factor4 + 2.0Factor5

```r
#Generate the linear model of factor9 distribution
lRegression5 <- lm( formula =  factor9  ~  factor1 + factor2  +  factor3  + factor4 + factor5,
                    data = data)
```

```
##
## Call:
## lm(formula = factor9 ~ factor1 + factor2 + factor3 + factor4 +
##     factor5, data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.050e-13 -3.282e-14  1.500e-16  2.480e-14  1.243e-13
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -2.094e-14  1.758e-14 -1.192e+00    0.236
## factor1      1.599e-15  4.162e-15  3.840e-01    0.702
## factor2      2.000e+00  1.391e-15  1.437e+15   <2e-16 ***
## factor3     -3.687e-15  4.977e-15 -7.410e-01    0.461
## factor4     -1.541e-16  1.552e-15 -9.900e-02    0.921
## factor5      1.000e+00  8.246e-16  1.213e+15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.466e-14 on 94 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 7.802e+29 on 5 and 94 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that factor2 and factor5 have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we could determine that this two categories have a relation with the factor6 distribution. In this case, the R-squared value is 1 so the probability to predict the factor6 distribution in this calculated linear model can be considered as very good one. So it can be determined that:

Factor9: 2.0Factor2 + 1.0Factor5

```r
#Generate the linear model of factor10 distribution
lRegression6 <- lm( formula =  factor10  ~  factor1 + factor2  +  factor3  + factor4 + factor5,
                    data = data)
```

```
##
## Call:
## lm(formula = factor10 ~ factor1 + factor2 + factor3 + factor4 +
##     factor5, data = data)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -1.465e-13 -2.179e-14 -2.476e-15  2.616e-14  1.428e-13
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  3.615e-14  1.633e-14  2.213e+00   0.0293 *
## factor1      1.000e+00  3.868e-15  2.585e+14   <2e-16 ***
## factor2      3.000e+00  1.293e-15  2.320e+15   <2e-16 ***
## factor3      1.000e+00  4.625e-15  2.162e+14   <2e-16 ***
```

```
## factor4     -4.250e-17  1.442e-15 -2.900e-02   0.9765
## factor5     -2.613e-16  7.663e-16 -3.410e-01   0.7339
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.15e-14 on 94 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.146e+30 on 5 and 94 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that factor1, factor2 and factor3 have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we could determine that this two categories have a relation with the factor6 distribution. In this case, the R-squared value is 1 so the probability to predict the factor6 distribution in this calculated linear model can be considered as very good one. So it can be determined that:

Factor10: 1.0Factor1 + 3.0Factor2 + 1.0Factor3

Comparing the linear combinations calculated of all the factors from 6 to 10, we can determine that the results obtained are exactly the same than the ones that were calculated.

After being able to determine all the relations between the factors that have linear combinations, it is time to perform the linear model of the answer without the discarded attributes, after taking into account the conclusions of the first linear model performed.

```
#Generate the linear model without discarted distributions
lRegression7 <- lm(formula =  answer  ~  factor2  +  factor3  + factor4 + factor5, data = data)
```

```
##
## Call:
## lm(formula = answer ~ factor2 + factor3 + factor4 + factor5,
##     data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3.14561 -0.75798 -0.00027  0.73321  3.06249
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.72747    0.47554   1.530    0.129
## factor2      2.02354    0.03821  52.960  < 2e-16 ***
## factor3     -0.99130    0.13655  -7.259 1.06e-10 ***
## factor4      3.00319    0.04246  70.723  < 2e-16 ***
## factor5      0.99744    0.02258  44.174  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.227 on 95 degrees of freedom
## Multiple R-squared:  0.9902, Adjusted R-squared:  0.9898
## F-statistic:  2403 on 4 and 95 DF,  p-value: < 2.2e-16
```

With the results retrieved, we can see that all values have a p-value lower than 0.05 and the t-value much higher or much lower than 0. So we could determine that all categories have a relation with the answer distribution. In this case, the R-squared value is 0.9898 so the probability to predict the answer distribution in this calculated linear model can be considered as very good one (the value has not change from the previous linear model when any factor was discarded).

Finally, Linear model test assumptions must be calculated to prove the test validity.

1-. Normality test

It is necessary to test that the response is normally distributed with Shapiro-Wilk normality test.

```
#Shapiro-Wilk normality test
shapiro.test(residuals(lRegression7))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(lRegression7)
## W = 0.99464, p-value = 0.9648
```

The calculated p-value is higher than the significance level 0.05, so there is no indication that normality distribution is violated.

2-. Homogeneity of variance

It is necessary to test that the variance is similar within different groups with Breusch Pagan test.

```
#Breusch Pagan test
lmtest::bptest(lRegression7)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lRegression7
## BP = 6.0222, df = 4, p-value = 0.1975
```

The calculated p-value is higher than the significance level 0.05, so there is no indication of heteroscedasticity on the calculated model. So we can conclude that there is homogeneity of variance. Otherwise, some transformations to eliminate heteroscedasticity would have been performed.

3-. Independence of variables

It is necessary to test that the data values are independent with Durbin Watson test.

```
#Durbin Watson test
lmtest::dwtest(lRegression7)
```

```
##
##  Durbin-Watson test
##
## data:  lRegression7
## DW = 1.9561, p-value = 0.4066
## alternative hypothesis: true autocorrelation is greater than 0
```

The calculated p-value is higher than the significance level 0, so the alternative hypothesis can be accepted. It means that the variables are not independent and it exits a correlation between them.

After checking the validity of the test performed, it can be determined that a possible expression to generate new data is the following one:

answer = 2.02538 * factor2 - 0.98192 * factor3 + 3.00774 * factor4 + 0.99971 * factor5

Comparing the linear combination calculated from the one defined during the previous exercise, we can determine that the results obtained are approximately the same than the ones that were calculated (see NOTE coomment from the previous exercise). It can also be determined that the decimal variation of the value of each multiplied factor it's due to the error added on the initial definition of the answer.

To finally check if the linear model just calculated is correct for our dataset, a prediction of the next value of answer will be performed. To do it, another dataset with the same characteristics of the initial dataset will

be created. But this time, the answer parameter will be calculated using the new linear model defined.

```
#Generation of the different factors as defined
factor1 <- rnorm(30, mean=0, sd=1)
factor2 <- runif(30, min=0, max=12)
factor3 <- rexp(30, rate=1)
factor4 <- rnorm(30, mean=3, sd=3)
factor5 <- runif(30, min=6, max=25)

#Generation of the factors as a combination of the previous ones
factor6 <- (2*factor1) + factor3
factor7 <- factor4 + factor1
factor8 <- factor3 + (3*factor4) + (2*factor5)
factor9 <- (2*factor2) + factor5
factor10 <- (3*factor2) + factor3 + factor1

#Generation of the answer variable
answer <- 2.01467 * factor2 - 1.10249 * factor3 + 2.99932 * factor4 + 0.99738 * factor5
data2 <- data.frame(factor1, factor2, factor3, factor4, factor5, factor6,
                    factor7, factor8, factor9, factor10, answer)
```

```
#Predict the behavior of the athletes with prediction interval
predictedData <- predict(lRegression7, newdata=data2, interval="prediction")

#Determine if the data of OlympicsG competition lies between the rangs calculated
#TRUE if fits into the rang, FALSE OTHERWISE
compData <- c()
predictedDataFrame <- data.frame(x1=predictedData)
answerDataFrame <- data.frame(x1=answer)
for(x in 1:nrow(predictedDataFrame)) {
  if( answerDataFrame$x1[x] >= predictedDataFrame$x1.lwr[x] &&
      answerDataFrame$x1[x] <= predictedDataFrame$x1.upr[x]){
    compData[x] <-  TRUE
  }else{
    compData[x] <-  FALSE
  }
}
predictedDataFrame["prediction"] <- compData
predictedDataFrame
```

```
##      x1.fit    x1.lwr   x1.upr prediction
## 1  30.93187 28.442685 33.42106       TRUE
## 2  22.90076 20.429417 25.37210       TRUE
## 3  48.01578 45.533952 50.49760       TRUE
## 4  20.44470 17.917455 22.97194       TRUE
## 5  62.05865 59.518956 64.59835       TRUE
## 6  22.61424 20.045983 25.18249       TRUE
## 7  56.13237 53.624066 58.64068       TRUE
## 8  11.07783  8.562190 13.59347       TRUE
## 9  40.26569 37.775424 42.75596       TRUE
## 10 42.96917 40.510921 45.42743       TRUE
## 11 47.75909 45.269355 50.24882       TRUE
## 12 33.04420 30.587687 35.50071       TRUE
## 13 30.52411 28.048640 32.99958       TRUE
## 14 37.20157 34.721515 39.68163       TRUE
```

```
## 15 39.99186 37.457577 42.52614       TRUE
## 16 13.75780 11.220858 16.29473       TRUE
## 17 41.09871 38.599571 43.59785       TRUE
## 18 33.84947 31.315756 36.38318       TRUE
## 19 50.31974 47.836528 52.80296       TRUE
## 20 12.07337  9.539398 14.60735       TRUE
## 21 12.55209 10.038506 15.06566       TRUE
## 22 41.63192 39.113894 44.14994       TRUE
## 23 50.47772 47.982697 52.97274       TRUE
## 24 46.16128 43.668865 48.65370       TRUE
## 25 42.14501 39.655368 44.63465       TRUE
## 26 35.92995 33.429058 38.43085       TRUE
## 27 27.96005 25.455942 30.46416       TRUE
## 28 40.51405 38.053741 42.97436       TRUE
## 29 33.99165 31.520491 36.46280       TRUE
## 30 27.88735 25.420382 30.35433       TRUE
```

As it was expected, we have obtained 30 well-predicted values and zero wrong-predicted. Then, the prediction interval obtained is 100% and can be confirmed that the values obtained lie within the prediction interval of 95% of the samples so it can be determined that the model designed is an accurate one.

FORTH QUESTION: DOE

The main objective of this last exercise is to perform a design of experiments with the model obtained from the previous exercise, in order to explore what parametrization of the 10 factors the answer obtains the best value.

To perform a design of experiments, the following points must be developed.

- **Set the objectives.**

Determine the parametrization of the 10 factors with what the answer obtains the maximum value.

- **Select the process variables.**

The process variables needed are Factor 1-10. But, taking into account the results obtained during the previous exercise, it was determined that answer is just correlated with factor2, factor3, factor4 and factor5. So, a simplification can be made with just using this four factors mentioned. In the following point, the implications of this simplification will be discused.

- **Define an experimental design.**

To be able to determine the parametrization of the 10 factors with what the answer obtains the maximum value, a full factorial design must be performed in order to assure that all the combinations with economy are being analyzed during the experimentation.

Since our factor values have more than 2 levels, n*vˆk experiments should be performed, where n are the replications of the experiment, v the levels for each factor and k the number of factors being investigated. With 10 factors and more than two labels, the total amount of experiments needed is too high.

So, the easiest factorial design to achieve the objective is the 2ˆk experimental design, where k is the number of factors being investigated in the experiment. This factorial experiments, also known as full factorial two level experiments, is a factorial experiments in which each factor is investigated at only two levels, which are considered as the most important ones, usually the maximum and the minimum values are chosen. With this design, all combinations of the levels of the factors are run.

In the current experimental design, 10 factors need to be investigated. Then, with full factorial two level experiment, 2ˆ10=1024 experiments need to be performed for a single replicate. Hopefully, during the previous exercise, it has been determined that the answer distribution is just correlated with factor2, factor3,

factor4 and factor5. In this way, the number of factors can be simplified and the number of runs needed is now set to 2^4=16.

Additional, replications are needed to be able to have information regarding the errors in the measurements. Unfortunately, due to the limited budget, just two replications can be performed. Otherwise, the needed number of replications to obtain a result within a desired confidence level should be calculated. So, finally, the number of runs for this experiment is calculated by n*2^k, which is 2*2^4 = 32 runs.

To conclude, a 2^k experimental design needs to be performed where:

- K = 4, regarding factor2, factor3, factor4 and factor5
- N = 2
- The two levels defined for each factor will be the minimum and the maximum value of each of them
- 32 runs will be performed
- Yates algorithm will be used to simplify the interaction calculus of the experiment

- **Execute the design.**

Now it is time to execute the designed experiment.

To be able to execute it, the dataset of the two replications need to be performed. Thanks to the new expression found during the previous exercise, the generation of the datasets needed can be simplified with the factors 2, 3, 4 and 5.

```
#First replication
#Generation of the different factors needed as defined
factor2 <- runif(20, min=0, max=12)
factor3 <- rexp(20, rate=1)
factor4 <- rnorm(20, mean=3, sd=3)
factor5 <- runif(20, min=6, max=25)

#Generation of the answer variable using the linear model defined
answer <- 2.01467 * factor2 - 1.10249 * factor3 + 2.99932 * factor4 + 0.99738 * factor5

#Generation of the dataset
data3 <- data.frame(factor2, factor3, factor4, factor5, answer)
```

```
#Second replication
#Generation of the different factors needed as defined
factor2 <- runif(20, min=0, max=12)
factor3 <- rexp(20, rate=1)
factor4 <- rnorm(20, mean=3, sd=3)
factor5 <- runif(20, min=6, max=25)

#Generation of the answer variable using the linear model defined
answer <- 2.01467 * factor2 - 1.10249 * factor3 + 2.99932 * factor4 + 0.99738 * factor5

#Generation of the dataset
data4 <- data.frame(factor2, factor3, factor4, factor5, answer)
```

Once the needed datasets are already generated, the two levels for each factor needs to be determined. As it has been defined during the previous points, the maximum and the minimum value will be used for each of this levels. So, this values need to be calculated from the datasets created.

```
summary(data3)
```

```
##     factor2           factor3            factor4          factor5
##  Min.   : 0.2993   Min.   :0.001798   Min.   :-2.439   Min.   : 6.306
##  1st Qu.: 4.4087   1st Qu.:0.206745   1st Qu.: 2.361   1st Qu.:10.036
```

```
##  Median : 6.4284   Median :0.830833   Median : 3.360   Median :12.325
##  Mean   : 6.3874   Mean   :1.146096   Mean   : 3.092   Mean   :14.504
##  3rd Qu.: 9.5021   3rd Qu.:1.874165   3rd Qu.: 4.198   3rd Qu.:20.469
##  Max.   :11.1728   Max.   :3.109435   Max.   : 6.952   Max.   :24.259
##      answer
##  Min.   :20.54
##  1st Qu.:28.14
##  Median :33.87
##  Mean   :35.35
##  3rd Qu.:39.77
##  Max.   :64.80
```

```
summary(data4)
```

```
##     factor2           factor3          factor4           factor5
##  Min.   : 0.4419   Min.   :0.0832   Min.   :-3.7635   Min.   : 6.417
##  1st Qu.: 4.0937   1st Qu.:0.2018   1st Qu.: 0.2565   1st Qu.:10.189
##  Median : 6.3155   Median :0.3970   Median : 1.8027   Median :13.263
##  Mean   : 6.3740   Mean   :0.9572   Mean   : 1.9190   Mean   :15.296
##  3rd Qu.: 8.7735   3rd Qu.:1.1367   3rd Qu.: 3.4419   3rd Qu.:20.994
##  Max.   :11.7357   Max.   :5.0795   Max.   : 6.5442   Max.   :24.711
##      answer
##  Min.   :14.83
##  1st Qu.:23.23
##  Median :33.06
##  Mean   :32.80
##  3rd Qu.:40.82
##  Max.   :56.45
```

With the maximum and minimum value of each of the factors, we create the Yates table for the two replications using Excel. Then, the response is calculated for both replications using the extreme defined values. Finally, the mean of the two obtained values is calculated and this calculated value is the one that will be used as a response for the Yates algorithm calculus.

	A	B	C	D	E	F	G	H
1	Factor2	Factor3	Factor4	Factor5	Values_data3	Values_data4	MeanValues	
2								
3	-	-	-	-	3,036	4,168	3,602	
4	+	-	-	-	13,910	15,042	14,47575	
5	-	+	-	-	8,032	7,276	7,65415	
6	+	+	-	-	18,906	18,149	18,5274	
7	-	-	+	-	13,344	13,559	13,45135	
8	+	-	+	-	24,217	24,433	24,3251	
9	-	+	+	-	18,340	16,667	17,5035	
10	+	+	+	-	29,214	27,54	28,37675	
11	-	-	-	+	21,330	22,121	21,7255	
12	+	-	-	+	32,204	32,995	32,59925	
13	-	+	-	+	26,326	25,229	25,77765	
14	+	+	-	+	37,200	36,102	36,6509	
15	-	-	+	+	31,638	31,512	31,57485	
16	+	-	+	+	42,511	42,386	42,4486	
17	-	+	+	+	36,634	34,62	35,627	
18	+	+	+	+	47,508	45,493	46,50025	
19								
20								
21	data3							
22	factor2		factor3		factor4		factor5	
23	min	0,4419	min	0,1204	min	-3	min	6
24	max	12	max	3	max	6	max	25
25								
26								
27	data4							
28	factor2		factor3		factor4		factor5	
29	min	0,2993	min	0,0832	min	-4	min	6
30	max	11	max	5	max	7	max	25
31								

The final table calculated is copied to a new datafile named yates_data5.xls and it is imported to RStudio.

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
```

```
##     startsWith
```

```
yatesTable <- read.xls("/Users/balbinavirgili/yates_data5.xls")
```

```
##    Factor2 Factor3 Factor4 Factor5   Values
## 1        -       -       -       -  3.60200
## 2        +       -       -       - 14.47575
## 3        -       +       -       -  7.65415
## 4        +       +       -       - 18.52740
## 5        -       -       +       - 13.45135
## 6        +       -       +       - 24.32510
## 7        -       +       +       - 17.50350
## 8        +       +       +       - 28.37675
## 9        -       -       -       + 21.72550
## 10       +       -       -       + 32.59925
## 11       -       +       -       + 25.77765
## 12       +       +       -       + 36.65090
## 13       -       -       +       + 31.57485
## 14       +       -       +       + 42.44860
## 15       -       +       +       + 35.62700
## 16       +       +       +       + 46.50025
```

Now, Yates algorithm iterations will be performed using the library Dae.

```
library("dae")
```

```
## Loading required package: ggplot2
```

```
#Yates algorithm
anovaModel <- aov(Values~ Factor2*Factor3*Factor4*Factor5, data=yatesTable)
yatesModel <- yates.effects(anovaModel, data = yatesTable)
```

```
##                         Factor2                         Factor3
##                    1.087350e+01                    4.051900e+00
##                         Factor4                         Factor5
##                    9.849350e+00                    1.812350e+01
##                 Factor2:Factor3                 Factor2:Factor4
##                   -2.500000e-04                    6.687346e-17
##                 Factor3:Factor4                 Factor2:Factor5
##                    3.628545e-16                   -2.196917e-16
##                 Factor3:Factor5                 Factor4:Factor5
##                    2.296853e-15                   -2.869706e-17
##         Factor2:Factor3:Factor4         Factor2:Factor3:Factor5
##                    4.010783e-16                   -6.446241e-16
##         Factor2:Factor4:Factor5         Factor3:Factor4:Factor5
##                   -5.109720e-16                   -8.069530e-16
## Factor2:Factor3:Factor4:Factor5
##                   -8.451582e-16
```

Finally, the mean of the different factors are also calculated to be able to determine the main effects of the calculus.

```
meanData3 <- (sum(yatesTable$Values)/16)
```

```
## [1] 25.05125
```

- **Check that the data are consistent with the experimental assumptions.**

To prove the data consistency, we perform and ANOVA test and the related assumptions to the data generated

for this experiment.

```
#ANOVA test for the data genereted for the first replication
anovaModel <- aov(answer~ factor2*factor3*factor4*factor5 , data=data3)
```

```
##                                 Df Sum Sq Mean Sq   F value Pr(>F)
## factor2                          1  628.9   628.9 2.465e+29 <2e-16 ***
## factor3                          1  151.1   151.1 5.923e+28 <2e-16 ***
## factor4                          1 1068.0  1068.0 4.186e+29 <2e-16 ***
## factor5                          1  450.9   450.9 1.767e+29 <2e-16 ***
## factor2:factor3                  1    0.0     0.0 1.874e+00  0.243
## factor2:factor4                  1    0.0     0.0 2.158e+00  0.216
## factor3:factor4                  1    0.0     0.0 7.300e-01  0.441
## factor2:factor5                  1    0.0     0.0 1.526e+00  0.284
## factor3:factor5                  1    0.0     0.0 3.820e-01  0.570
## factor4:factor5                  1    0.0     0.0 2.350e-01  0.653
## factor2:factor3:factor4          1    0.0     0.0 7.270e-01  0.442
## factor2:factor3:factor5          1    0.0     0.0 1.417e+00  0.300
## factor2:factor4:factor5          1    0.0     0.0 4.070e-01  0.558
## factor3:factor4:factor5          1    0.0     0.0 4.140e-01  0.555
## factor2:factor3:factor4:factor5  1    0.0     0.0 8.000e-01  0.422
## Residuals                        4    0.0     0.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1-. Normality test

It is necessary to test that the response is normally distributed with Shapiro-Wilk normality test.

```
#Shapiro-Wilk normality test
shapiro.test(residuals(anovaModel))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(anovaModel)
## W = 0.93068, p-value = 0.1592
```

The calculated p-value is higher than the significance level 0.05, so there is no indication that normality distribution is violated.

2-. Homogeneity of variance

It is necessary to test that the variance is similar within different groups with Breusch Pagan test.

```
#Breusch Pagan test
lmtest::bptest(anovaModel)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  anovaModel
## BP = 10.436, df = 15, p-value = 0.7914
```

The calculated p-value is higher than the significance level 0.05, so there is no indication of heteroscedasticity on the calculated model. So we can conclude that there is homogeneity of variance. Otherwise, some transformations to eliminate heteroscedasticity would have been performed.

3-. Independence of variables

It is necessary to test that the data values are independent with Durbin Watson test.

```
#Durbin Watson test
lmtest::dwtest(anovaModel)
```

```
##
##  Durbin-Watson test
##
## data:  anovaModel
## DW = 2.4348, p-value = 0.8865
## alternative hypothesis: true autocorrelation is greater than 0
```

The calculated p-value is higher than the significance level 0, so the alternative hypothesis can be accepted. It means that the variables are not independent and it exits a correlation between them.

```
#ANOVA test for the data genereted for the second replication
anovaModel <- aov(answer~ factor2*factor3*factor4*factor5 , data=data4)
```

```
##                                Df Sum Sq Mean Sq   F value Pr(>F)
## factor2                         1  315.4   315.4 1.529e+29 <2e-16 ***
## factor3                         1  260.6   260.6 1.263e+29 <2e-16 ***
## factor4                         1 1485.6  1485.6 7.201e+29 <2e-16 ***
## factor5                         1  777.1   777.1 3.767e+29 <2e-16 ***
## factor2:factor3                 1    0.0     0.0 3.433e+00 0.1375
## factor2:factor4                 1    0.0     0.0 4.057e+00 0.1142
## factor3:factor4                 1    0.0     0.0 8.280e-01 0.4144
## factor2:factor5                 1    0.0     0.0 4.350e-01 0.5456
## factor3:factor5                 1    0.0     0.0 6.499e+00 0.0634 .
## factor4:factor5                 1    0.0     0.0 8.000e-03 0.9339
## factor2:factor3:factor4         1    0.0     0.0 1.000e-03 0.9735
## factor2:factor3:factor5         1    0.0     0.0 3.192e+00 0.1485
## factor2:factor4:factor5         1    0.0     0.0 3.200e-02 0.8671
## factor3:factor4:factor5         1    0.0     0.0 6.760e-01 0.4571
## factor2:factor3:factor4:factor5 1    0.0     0.0 1.340e-01 0.7333
## Residuals                       4    0.0     0.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

1-. Normality test

It is necessary to test that the response is normally distributed with Shapiro-Wilk normality test.

```
#Shapiro-Wilk normality test
shapiro.test(residuals(anovaModel))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals(anovaModel)
## W = 0.95713, p-value = 0.4883
```

The calculated p-value is higher than the significance level 0.05, so there is no indication that normality distribution is violated.

2-. Homogeneity of variance

It is necessary to test that the variance is similar within different groups with Breusch Pagan test.

```
#Breusch Pagan test
lmtest::bptest(anovaModel)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  anovaModel
## BP = 14.018, df = 15, p-value = 0.5241
```

The calculated p-value is higher than the significance level 0.05, so there is no indication of heteroscedasticity on the calculated model. So we can conclude that there is homogeneity of variance. Otherwise, some transformations to eliminate heteroscedasticity would have been performed.

3-. Independence of variables

It is necessary to test that the data values are independent with Durbin Watson test.

```
#Durbin Watson test
lmtest::dwtest(anovaModel)
```

```
##
##  Durbin-Watson test
##
## data:  anovaModel
## DW = 1.4834, p-value = 0.1197
## alternative hypothesis: true autocorrelation is greater than 0
```

The calculated p-value is higher than the significance level 0, so the alternative hypothesis can be accepted. It means that the variables are not independent and it exits a correlation between them.

All assumptions are fullfiled for both datasets generated.

- **Analyze and interpret the results, detect effects of main factors and interactions.**

The results obtained are showed again for a better understanding of the conclusions.

```
yatesModel
```

```
##                         Factor2                         Factor3
##                    1.087350e+01                    4.051900e+00
##                         Factor4                         Factor5
##                    9.849350e+00                    1.812350e+01
##                 Factor2:Factor3                 Factor2:Factor4
##                   -2.500000e-04                    6.687346e-17
##                 Factor3:Factor4                 Factor2:Factor5
##                    3.628545e-16                   -2.196917e-16
##                 Factor3:Factor5                 Factor4:Factor5
##                    2.296853e-15                   -2.869706e-17
##         Factor2:Factor3:Factor4         Factor2:Factor3:Factor5
##                    4.010783e-16                   -6.446241e-16
##         Factor2:Factor4:Factor5         Factor3:Factor4:Factor5
##                   -5.109720e-16                   -8.069530e-16
## Factor2:Factor3:Factor4:Factor5
##                   -8.451582e-16
```

As it was stablished at the begining of the DOE, the main objective of this experiment is determine the parametrization of the 10 factors with what the answer obtains the maximum value. So, the factors that we are looking for are the ones that effect the most to the answer value.

With the results retrieved, all the factors that have a negative effect will be discarted. In this case, none of the factors have a negative impact to the answer because their relation was previously checked. Otherwise, there are some of interactions between the factors that have a negative impact (i.e Factor2:Factor3), all of this combinations can be discated.

For the factors and interactions that have a positive effect, the value retrived needs to be evaluated. Taking into consideration that the calculated mean is 25,05, the factors that are considered that effect the most are the following one (ordered by importance):

- Factor5 (18.12350)
- Factor2 (10.87350)
- Factor4 (9.849350)
- Factor3 (4.051900)

As we can see, there is no interaction with a significant effect.