MDS-SIM-Partial_21-22Q1: Template for solutions to questions

Your Name and ID Number

November 8th, 2021

Data Description

Available variables:

Nutrient analysis of pizzas (from https://data.world/sdhilip/pizza-datasets). Who likes pizza? I mean, there are so many things to like, let's take a closer look! The data set pizza. Pizza.RData contains measurements that capture the kind of things that make a pizza tasty. Can you determine which pizza brand works best for you and explain why? The variables in the data set are:

- brand: Pizza brand (class label)
- id: Sample analysed
- mois: Amount of water per 100 grams in the sample
- prot: Amount of protein per 100 grams in the sample
- fat: Amount of fat per 100 grams in the sample
- ash: Amount of ash per 100 grams in the sample
- sodium: Amount of sodium per 100 grams in the sample
- carb: Amount of carbohydrates per 100 grams in the sample
- cal: Amount of calories per 100 grams in the sample

List of Questions

Firstly, load dataset and check available variables.

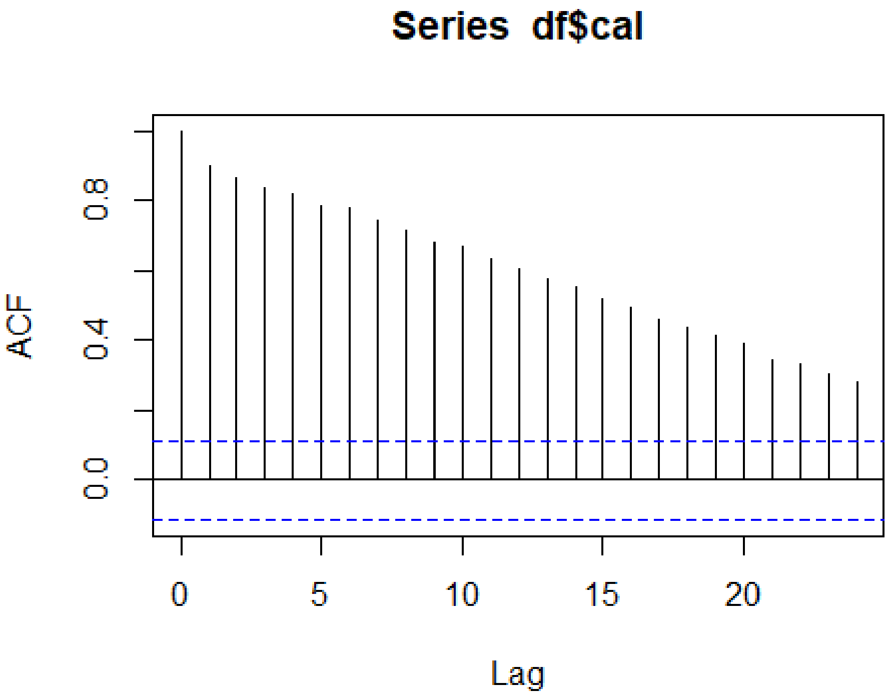
```
# Clear plots
if(!is.null(dev.list())) dev.off()

## null device
##           1

# Clean workspace
rm(list=ls())

setwd("C:/Users/lmontero/Dropbox/DOCENCIA/MUM-DATS/EXAMS/CURS21-22")
pathfile<-"C:/Users/lmontero/Dropbox/DOCENCIA/MUM-DATS/EXAMS/CURS21-22/"
# pizza <- read.table("Pizza.csv", header=T, dec=".", sep=",",stringsAsFa
ctors =T)
# df <- pizza
# row.names(pizza) <- paste0(pizza$brand,".",pizza$id)
# save(list=c("pizza","df"),file="Pizza.RData")
load(paste0(pathfile,"Pizza.RData"))

summary(df)

##      brand           id             mois            prot            fat
##  H       : 33   Min.   :14003   Min.   :25.00   Min.   : 6.98   Min.   : 4.38
##  D       : 32   1st Qu.:14094   1st Qu.:30.90   1st Qu.: 8.06   1st Qu.:14.77
##  J       : 32   Median :24021   Median :43.30   Median :10.44   Median :17.14
##  B       : 31   Mean   :20841   Mean   :40.90   Mean   :13.37   Mean   :20.23
##  F       : 30   3rd Qu.:24110   3rd Qu.:49.12   3rd Qu.:20.02   3rd Qu.:21.43
##  A       : 29   Max.   :34045   Max.   :57.22   Max.   :28.48   Max.   :47.20
##  (Other):113
##       ash            sodium            carb             cal
##  Min.   :1.170   Min.   :0.2500   Min.   : 0.510   Min.   :2.180
##  1st Qu.:1.450   1st Qu.:0.4500   1st Qu.: 3.467   1st Qu.:2.910
##  Median :2.225   Median :0.4900   Median :23.245   Median :3.215
##  Mean   :2.633   Mean   :0.6694   Mean   :22.865   Mean   :3.271
##  3rd Qu.:3.592   3rd Qu.:0.7025   3rd Qu.:41.337   3rd Qu.:3.520
##  Max.   :5.430   Max.   :1.7900   Max.   :48.640   Max.   :5.080
##

names(df)

## [1] "brand"  "id"     "mois"   "prot"   "fat"    "ash"    "sodium" "ca
rb"
## [9] "cal"

vars_res <- names(df)[c(9)]
vars_con <- names(df)[c(3:8)]
vars_dis <- names(df)[c(10:11)]
```

The amount of calories per 100 grams is selected as the numeric target.

1. Determine whether serial correlation is present on dataset or not.

You have seen acf() method to plot autocorrelation in calories. The plot clearly shows a serial correlation. Durbin-Watson test is an inferential tool that addresses whether autocorrelation is present and the null hypothesis is clearly rejected, thus again, autocorrelation is present in the cal column of the pizza dataset.

```
acf(df$cal)
```



```
library(lmtest)

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

dwtest(df$cal~1)

##
##  Durbin-Watson test
##
## data:  df$cal ~ 1
## DW = 0.1699, p-value < 2.2e-16
## alternative hypothesis: true autocorrelation is greater than 0
```

2. Define a new variable containing the total amount of ingredients (water, protein, fat, ash and carbohydrates). Check consistency.

Total sum of indicated ingredients accounts for 100 g in 292 observations and a less than 1 g deviation (above/below) can be seen in the rest of observations.

```
names(df)
```

```
## [1] "brand"   "id"      "mois"    "prot"    "fat"     "ash"     "sodium" "ca
rb"
## [9] "cal"
```

```
df$ingredients <- rowSums(df[,c(3:6,8)])
summary( df$ingredients )
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   99.37  100.00  100.00  100.00  100.00  100.83
```

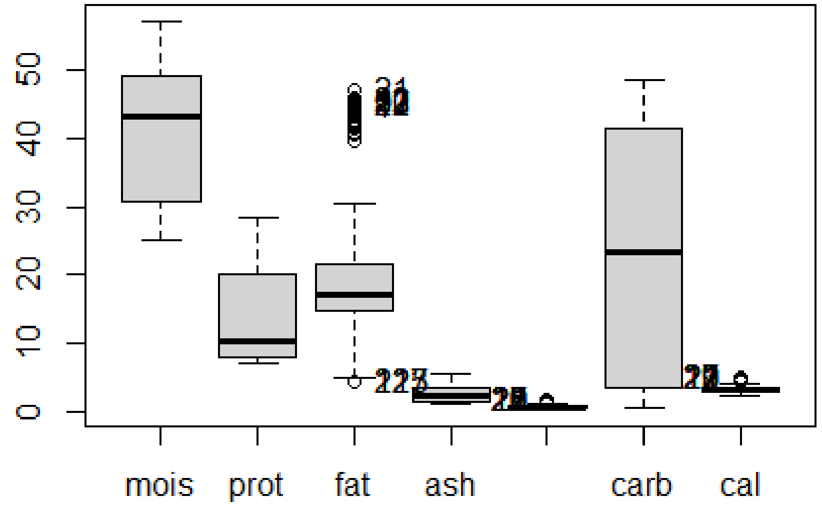
```
table( df$ingredients )
```

```
## 
##  99.37  99.99    100 100.01 100.05  100.1 100.27 100.63 100.83 
##      1      1    292      1      1      1      1      1      1
```

3. Univariant severe outliers are also present in some variables. Determine them.

Univariate severe outlier checking is addressed for all variables. There 271 observation without any outlier, 4 observations having 1 outlier (in sodium) and 25 observations having 2 outliers (on fat and sodium).

```
Boxplot( df[,c(3:9)])
```



```
##  [1] "123" "125" "217" "21"  "3"   "11"  "22"  "10"  "27"  "13"  "9"  
"12" 
## [13] "1"   "2"   "1"   "16"  "8"   "9"   "20"  "22"  "25"  "17"  "29"
```

```
"21"
## [25] "11"  "12"  "22"  "3"   "10"  "1"   "9"   "13"  "27"
```

```
unidesfat <- summary( df$fat ); unidesfat
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    4.38   14.77   17.14   20.23   21.43   47.20
```

```
thrusout <- unidesfat[5]+3*( unidesfat[5]-unidesfat[2]);thrusout
```

```
## 3rd Qu. 
##   41.43
```

```
llusout <- which( df$fat > thrusout ); llusout
```

```
##  [1]  1  2  3  4  5  6  8  9 10 11 12 13 14 15 16 20 21 22 23 24 25 26
27 28 29
```

```
df$sevout <- 0
df$sevout[ llusout ] <- 1

unidesash <- summary( df$ash ); unidesash
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   1.170   1.450   2.225   2.633   3.592   5.430
```

```
thrusout <- unidesash[5]+3*( unidesash[5]-unidesash[2]);thrusout
```

```
## 3rd Qu. 
##   10.02
```

```
llusout <- which( df$ash > thrusout ); llusout
```

```
## integer(0)
```

```
if (length(llusout) > 0 ) df$sevout[ llusout ] <- df$sevout[ llusout ] +
1

unidessod <- summary( df$sodium ); unidessod
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##  0.2500  0.4500  0.4900  0.6694  0.7025  1.7900
```

```
thrusout <- unidessod[5]+3*( unidessod[5]-unidessod[2]);thrusout
```

```
## 3rd Qu. 
##    1.46
```

```
llusout <- which( df$sodium > thrusout ); llusout
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
23 24 25
## [26] 26 27 28 29
```

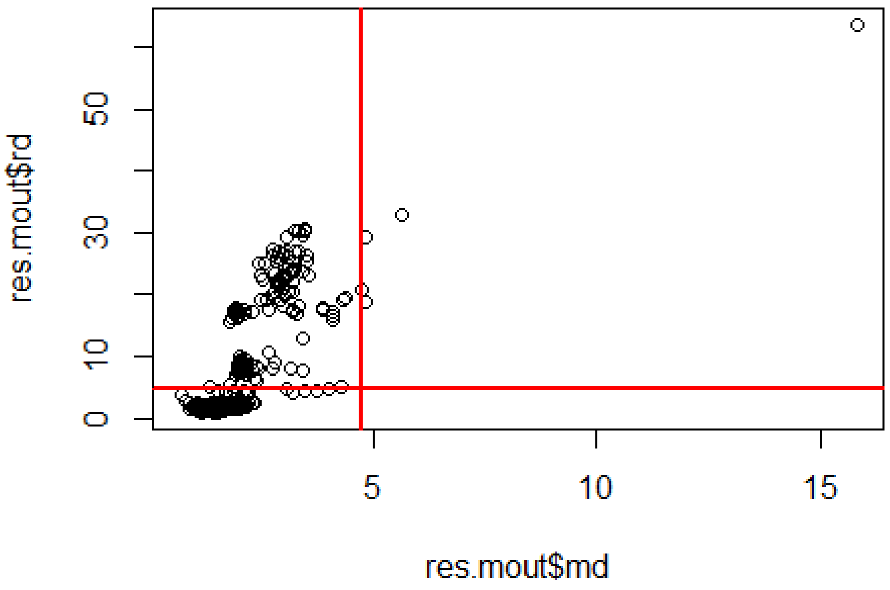
```
if (length(llusout) > 0 ) df$sevout[ llusout ] <- df$sevout[ llusout ] +
1

unidescal <- summary( df$cal ); unidescal

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.180   2.910   3.215   3.271   3.520   5.080

thrusout <- unidescal[5]+3*( unidescal[5]-unidescal[2]);thrusout

## 3rd Qu.
##    5.35

llusout <- which( df$cal > thrusout ); llusout

## integer(0)

if (length(llusout) > 0 ) df$sevout[ llusout ] <- df$sevout[ llusout ] +
1

summary( df$sevout )

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00    0.18    0.00    2.00

table( df$sevout )

##
##   0   1   2
## 271   4  25
```

4. Are there multivariant outliers? Find them. Try to explain their singularity. Multivariant outliers,if present, are not going to be treated in this exercise: keep them as suplementary observations in the rest of the exercise. Which is the cut-off at 99.9% CI?

There are 4 multivariant outliers according to the indicated criterion: observations 66-C 82-C 166-F and 296-J. Observation 166 without any doubt a multivariate outlier, the one with the largest Mahalanobis distance.

```
library(chemometrics)
res.mout <- Moutlier( df[ , c(4:9)], quantile = 0.999, plot=F )
```

```
par(mfrow=c(1,1))
plot( res.mout$md, res.mout$rd )
abline( h=res.mout$cutoff, lwd=2, col="red")
abline( v=res.mout$cutoff, lwd=2, col="red")
```

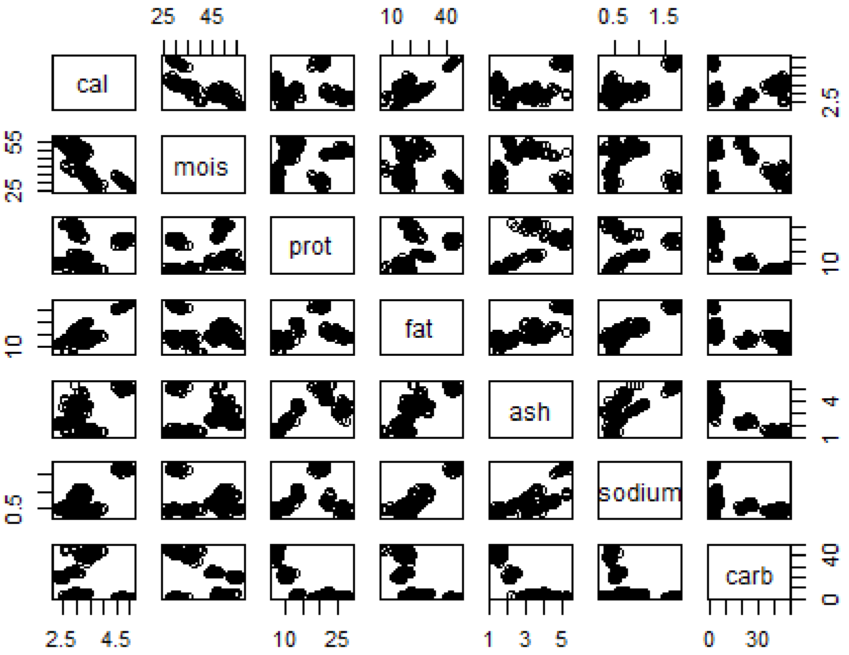


```
llmout <- which( ( res.mout$md > res.mout$cutoff ) & (res.mout$rd > res.m
out$cutoff) );llmout

## [1]  66  82 166 296

df[llmout,]

##     brand    id  mois  prot   fat  ash sodium  carb  cal ingredients sevout
## 66      C 14029 49.73 25.65 19.98 2.51   0.52  2.13 2.91      100.00      0
## 82      C 24124 49.57 26.91 18.00 2.21   0.41  3.31 2.83      100.00      0
## 166     F 24055 27.93  7.88 17.49 1.44   0.47 45.26 3.96      100.00      0
## 296     J 34044 44.91 11.07 17.00 2.49   0.66 25.36 2.91      100.83      0

res.mout$md[llmout]

## [1]  4.843280  5.651025 15.825626  4.826506

df$mout <- 0
df$mout[ llmout ] <- 1
df$mout <- factor( df$mout, labels = c("MvOut.No","MvOut.Yes"))
```

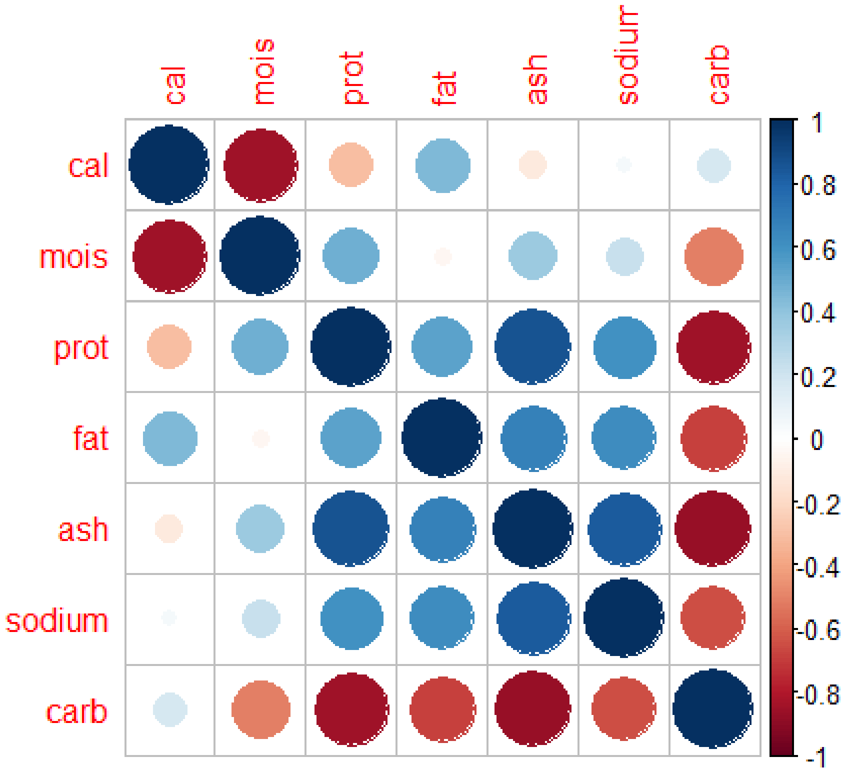
5. Indicate by using exploratory data analysis tools which are apparently the most associated variables with the numeric response variable (only the contributing variables to the ingredients are to be taken as active variables). Use also FactoMineR profiling tools at 99% significance level.

The globally most associated numeric variables are fat (direct relation) and mois (inverse relation), following sodium and ash (direct relation). Brand is globally *associated to cal, thus the number of calories per 100g of pizza depends on the brand. In particular, A and F brands are those having average caloric contribution greater than the mean (being A brand very remarkable) and J and I brands lie under the mean caloric contribution (mainly I brand).*

```
res.con <- condes( df[,c(1,3:9)], num.var=8, proba = 0.01 )
res.con$quanti

##        correlation      p.value
## fat      0.7645671 8.656128e-59
## sodium   0.6719575 9.322465e-41
## ash      0.3264685 7.021205e-09
## mois    -0.7644405 9.279582e-59

res.con$quali

##              R2       p.value
## brand 0.9293717 3.264747e-161

res.con$category

##           Estimate      p.value
## brand=A  1.4988622 1.983234e-66
## brand=F  0.3210691 2.348708e-03
## brand=G  0.3202415 2.911370e-03
## brand=D -0.2714934 9.575679e-03
## brand=C -0.4260420 1.821239e-04
## brand=J -0.3964934 1.300208e-04
## brand=I -0.8907930 8.680093e-18

plot(df[,c(9,3:8)])
```



```
cor(df[,c(9,3:8)], method="spearman")

##               cal        mois       prot         fat        ash     sodium
## cal     1.0000000 -0.84373723 -0.3038314  0.44513377 -0.1158363  0.0416364
## mois   -0.8437372  1.00000000  0.4804072 -0.04594197  0.3613436  0.2216707
## prot   -0.3038314  0.48040722  1.0000000  0.53426888  0.8659212  0.6002271
## fat     0.4451338 -0.04594197  0.5342689  1.00000000  0.6713663  0.6281431
## ash    -0.1158363  0.36134362  0.8659212  0.67136632  1.0000000  0.8321020
## sodium  0.0416364  0.22167073  0.6002271  0.62814312  0.8321020  1.0000000
## carb    0.1770379 -0.50109708 -0.8448299 -0.68186774 -0.8653723 -0.6440725
##              carb
## cal     0.1770379
## mois   -0.5010971
## prot   -0.8448299
## fat    -0.6818677
## ash    -0.8653723
## sodium -0.6440725
## carb    1.0000000

corrplot(cor(df[,c(9,3:8)], method="spearman"), is.corr=T)
```



6. Use brand target factor and determine the most relevant global associations at 99% CI. Profile A and I brands.

Global association to brand factor is depending on mean carbohidrates, protein and water composition. Nevertheless, all available ingredients are shown to have average values depending on brand.

- *Brand A has remarkable sodium, fat, cal, protein and ash over the global mean for those ingredients, while water and carbohidrates lie under their mean for A brand.*
- *Brand B has a mean content of water, sodium, fat and ashes over the mean. Carbohidrate contents is under the mean.*
- *Brand C shows a mean content of proteins, water and ashes over the mean and mean sodium, calories and carbohidrates under the mean.*

The same interpretation pattern has to be followed for the other brands.

```
res.cat <- catdes( df[,c(1,3:9)], num.var=1, proba = 0.01 )
res.cat$quanti.var

##             Eta2       P-value
## carb   0.9911957 3.087183e-292
## prot   0.9861491 1.036517e-263
## mois   0.9774313 5.578472e-233
## sodium 0.9709591 4.116312e-217
## ash    0.9683272 1.184316e-211
## cal    0.9293717 3.264747e-161
## fat    0.9273634 1.889056e-159
```

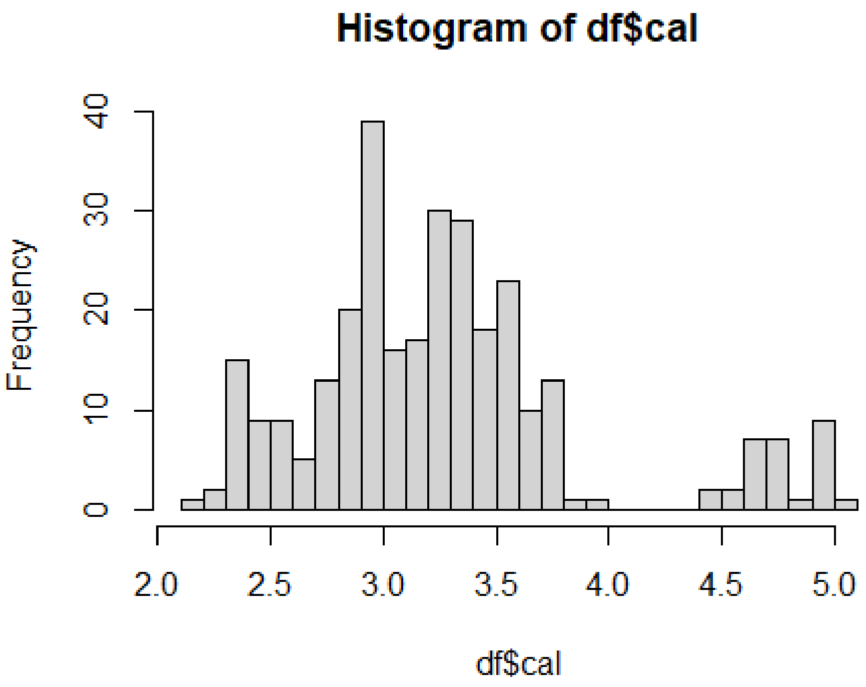
```
res.cat$quanti
```

```
## $A
##            v.test Mean in category Overall mean sd in category Overall sd
## sodium 15.096829         1.656207     0.669400     0.06354004   0.369740
## fat    14.656206        43.446897    20.229533     1.83069820   8.960686
## cal    13.732781         4.773793     3.271000     0.16078286   0.619000
## ash    10.626022         5.014483     2.633233     0.20088745   1.267606
## prot    5.929520        20.107241    13.373567     1.06066634   6.423659
## mois   -6.486769        29.966207    40.903067     1.92140289   9.537052
## carb   -6.718160         1.486897    22.864767     0.60894541  17.999648
##             p.value
## sodium 1.699107e-51
## fat    1.229536e-48
## cal    6.460533e-43
## ash    2.255290e-26
## prot   3.038206e-09
## mois   8.769645e-11
## carb   1.840330e-11
##
## $B
##            v.test Mean in category Overall mean sd in category Overall sd
## mois    6.404050       51.3077419    40.903067     1.70058569   9.537052
## sodium  5.007937        0.9848387     0.669400     0.09641546   0.369740
## fat     4.841610       27.6203226    20.229533     2.14682283   8.960686
## ash     3.845024        3.4635484     2.633233     0.23394993   1.267606
## carb   -6.162050        3.9696774    22.864767     0.60446449  17.999648
##             p.value
## mois   1.513088e-10
## sodium 5.501641e-07
## fat    1.287911e-06
## ash    1.205406e-04
## carb   7.180902e-10
##
## $C
##            v.test Mean in category Overall mean sd in category Overall sd
## prot   10.710567       26.0255556    13.373567     1.18318883   6.423659
## mois    4.889031       49.4774074    40.903067     1.48152722   9.537052
## ash     2.788895        3.2833333     2.633233     0.40139571   1.267606
## sodium -3.008943        0.4648148     0.669400     0.08941664   0.369740
## cal    -3.708278        2.8488889     3.271000     0.17299825   0.619000
## carb   -6.289570        2.0462963    22.864767     0.80335015  17.999648
##             p.value
## prot   9.080429e-27
## mois   1.013338e-06
## ash    5.288819e-03
## sodium 2.621584e-03
## cal    2.086735e-04
## carb   3.183464e-10
##
## $D
##          v.test Mean in category Overall mean sd in category Overall sd
## prot  8.239120        22.231250    13.373567      1.3674012   6.423659
## ash   7.931710         4.315937     2.633233      0.4426896   1.267606
## mois  4.240345        47.671250    40.903067      1.1203480   9.537052
## cal  -2.582723         3.003437     3.271000      0.1504287   0.619000
## carb -6.217024         4.136250    22.864767      1.7560729  17.999648
##           p.value
## prot 1.734818e-16
## ash  2.161490e-15
## mois 2.231768e-05
## cal  9.802407e-03
## carb 5.066716e-10
## 
## $E
##           v.test Mean in category Overall mean sd in category Overall sd
## carb    5.155775       39.5921429    22.864767     3.12111971  17.999648
## mois   -2.803818       36.0832143    40.903067     1.72366315   9.537052
## fat    -3.166170       15.1157143    20.229533     4.03342059   8.960686
## sodium -3.302798        0.4492857     0.669400     0.03358662   0.369740
## prot   -4.871710        7.7328571    13.373567     0.38725091   6.423659
## ash    -5.064542        1.4760714     2.633233     0.09611523   1.267606
##              p.value
## carb   2.525838e-07
## mois   5.050147e-03
## fat    1.544605e-03
## sodium 9.572536e-04
## prot   1.106367e-06
## ash    4.093841e-07
## 
## $F
##           v.test Mean in category Overall mean sd in category Overall sd
## carb    7.020075        44.787333    22.864767     1.72395849  17.999648
## cal     3.026266         3.596000     3.271000     0.15116437   0.619000
## sodium -3.233156         0.462000     0.669400     0.03177001   0.369740
## prot   -4.913160         7.898000    13.373567     0.21575295   6.423659
## ash    -5.272613         1.473667     2.633233     0.08557583   1.267606
## mois   -6.949446        29.404333    40.903067     0.90676231   9.537052
##              p.value
## carb   2.217497e-12
## cal    2.475947e-03
## sodium 1.224306e-03
## prot   8.962010e-07
## ash    1.344947e-07
## mois   3.667226e-12
## 
## $G
##           v.test Mean in category Overall mean sd in category Overall sd
## carb    7.406098       46.4317241    22.864767     1.64332520  17.999648
## cal     2.962343        3.5951724     3.271000     0.13197755   0.619000
## fat    -2.894797       15.6437931    20.229533     1.85630968   8.960686
## sodium -3.451484        0.4437931     0.669400     0.02265389   0.369740
## prot   -4.523538        8.2365517    13.373567     0.18475101   6.423659
## ash    -5.293876        1.4468966     2.633233     0.10399282   1.267606
## mois   -7.509988       28.2410345    40.903067     1.47993789   9.537052
##              p.value
## carb   1.300697e-13
## cal    3.053074e-03
## fat    3.794038e-03
## sodium 5.575118e-04
## prot   6.081431e-06
## ash    1.197503e-07
## mois   5.913294e-14
##
## $H
##           v.test Mean in category Overall mean sd in category Overall sd
## carb    5.984341       40.5839394    22.864767     2.56523251  17.999648
## mois   -3.236743       35.8251515    40.903067     1.36110499   9.537052
## fat    -4.028443       14.2915152    20.229533     3.53491770   8.960686
## sodium -4.165271        0.4160606     0.669400     0.02214818   0.369740
## prot   -5.185103        7.8945455    13.373567     0.41615186   6.423659
## ash    -5.885163        1.4060606     2.633233     0.12782420   1.267606
##             p.value
## carb   2.172671e-09
## mois   1.209022e-03
## fat    5.614757e-05
## sodium 3.109829e-05
## prot   2.158952e-07
## ash    3.976621e-09
##
## $I
##           v.test Mean in category Overall mean sd in category Overall sd
## mois    8.119504       54.5927586     40.90307     0.96211296   9.537052
## prot   -2.633334       10.3831034     13.37357     0.53367369   6.423659
## sodium -2.786784        0.4872414      0.66940     0.02317796   0.369740
## fat    -4.525408       13.0606897     20.22953     0.90310167   8.960686
## cal    -8.104298        2.3841379      3.27100     0.07151162   0.619000
##             p.value
## mois   4.680938e-16
## prot   8.455122e-03
## sodium 5.323398e-03
## fat    6.027898e-06
## cal    5.305112e-16
##
## $J
##         v.test Mean in category Overall mean sd in category Overall sd
## mois  3.215215        46.035000     40.90307     1.37306227   9.537052
## fat  -2.604206        16.324063     20.22953     1.11237319   8.960686
## prot -2.611272        10.566250     13.37357     0.63352067   6.423659
## cal  -3.789321         2.878437      3.27100     0.09718248   0.619000
##           p.value
## mois 0.0013034674
## fat  0.0092087434
## prot 0.0090206160
## cal  0.0001510599
```

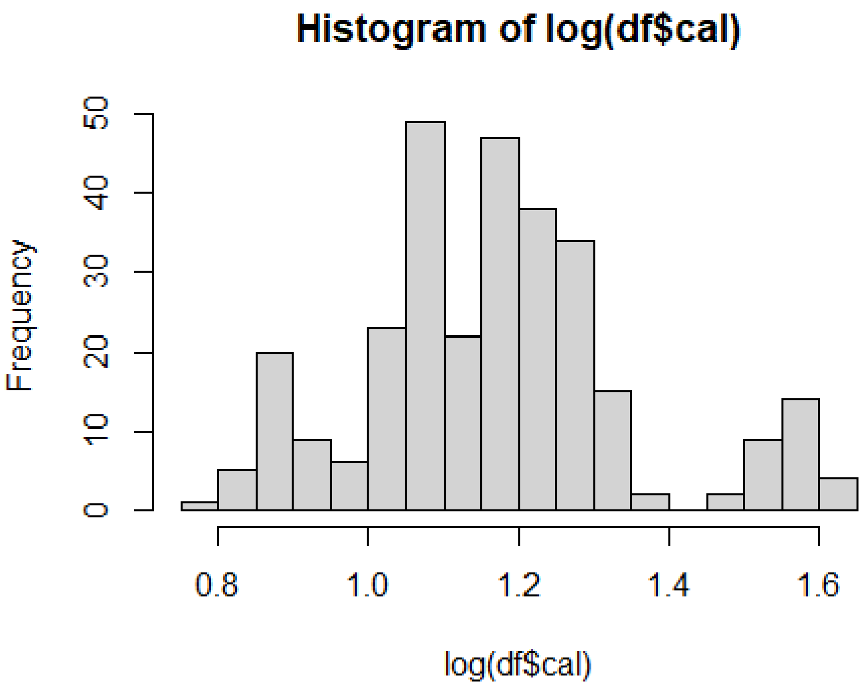
7. Say a few words about the hypothetical distribution that was assumed in the past. Use graphical and inferential arguments.

Either considering histogram chart, or by assessing normal distribution on the logarithm transformation. Using input data analysis and distribution fitting tools, the distribution is shown to lie close to lognormal in the beta area.

```
hist(df$cal,30)
```

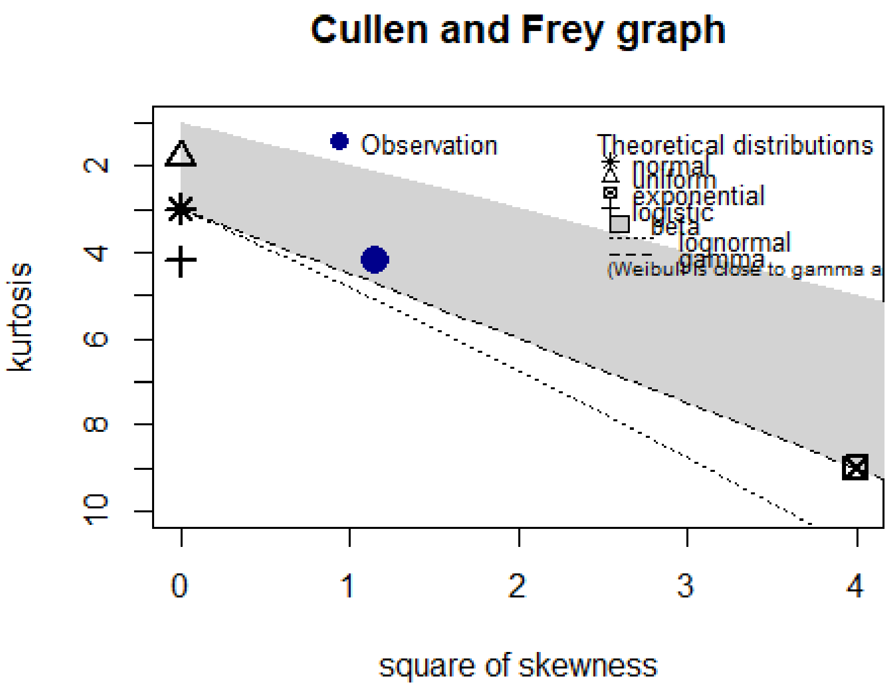


```
hist(log(df$cal),30)
```



```
shapiro.test( log(df$cal) )
```

```
##
##  Shapiro-Wilk normality test
##
## data:  log(df$cal)
## W = 0.9536, p-value = 3.768e-08

library(fitdistrplus)

descdist( df$cal )
```

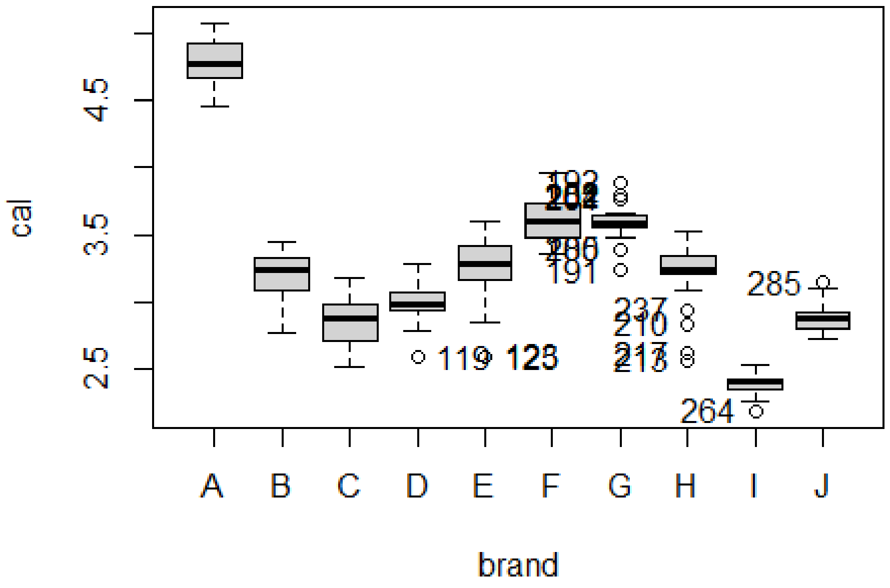


```
## summary statistics
## ------
## min:  2.18   max:  5.08
## median:  3.215
## mean:  3.271
## estimated sd:  0.6200343
## estimated skewness:  1.074278
## estimated kurtosis:  4.188494
```

8. Let us focus on cal variate dispersion behavior according to the brand. Use numeric, graphics and inferential tools to address the topic.

Numeric summary across brand standard deviation for calories, allow to classify brands according to low standard deviation for I and J, medium for A, B, C, D, F and G and high for E and H. Inferential testing using Fligner-Killeen test a null hypothesis of homogeinity of variance allows to reject the null hypothesis because pvalue is 0.0003 lower than any common significative level.

```
tapply( df$cal, df$brand, sd ) # E and H overdispersed I and J underdispe
rsed

##          A          B          C          D          E          F          G
## 0.16362880 0.17197587 0.17629375 0.15283576 0.25444696 0.15374856 0.13431362
##          H          I          J
## 0.22080946 0.07277741 0.09873749

Boxplot( cal~brand, data = df )
```

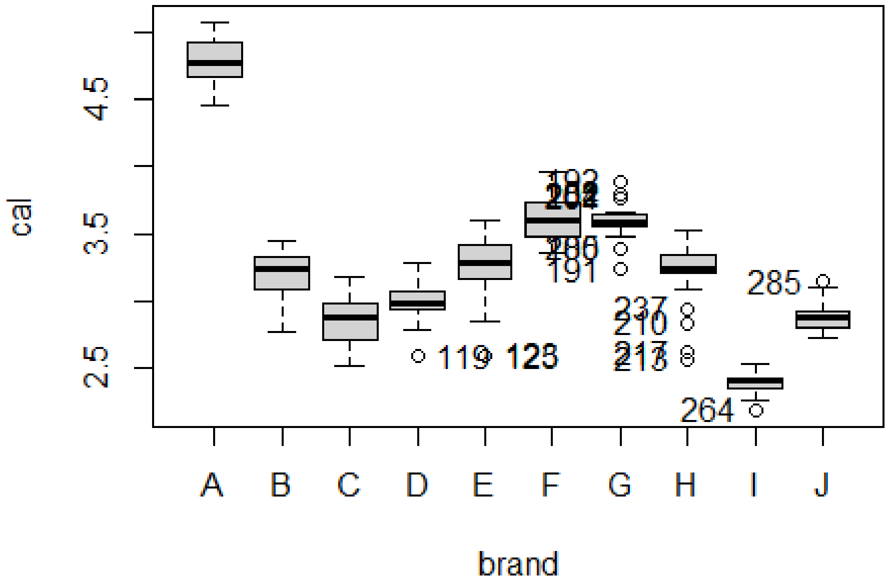


```
##  [1] "119" "123" "125" "180" "191" "205" "182" "192" "201" "202" "204"
"210"
## [13] "213" "217" "237" "264" "285"

fligner.test( cal~brand, data = df )

##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  cal by brand
## Fligner-Killeen:med chi-squared = 30.956, df = 9, p-value = 0.0003012
```

9. Let us focus on cal variate mean behavior according to the brand. Use numeric, graphics and inferential tools to address whether the mean of the target depends on the brand or not.

The mean calories for C, I and J brands seems to be less than those for other brands taking into account mean summary across brand levels. A brand has the highest value *for mean calories and it is remarkable higher than the rest. Using the non-parametric Kruskal-Wallis homogeneity test for means, the null hypothesis is absolutely rejected (pvalue 2.2e-16).*

```
tapply( df$cal, df$brand, mean ) # A overmean I undermean

##        A        B        C        D        E        F        G        H 
## 4.773793 3.190968 2.848889 3.003437 3.253929 3.596000 3.595172 3.224545 
##        I        J 
## 2.384138 2.878437

Boxplot( cal~brand, data = df )
```



```
##  [1] "119" "123" "125" "180" "191" "205" "182" "192" "201" "202" "204"
"210"
## [13] "213" "217" "237" "264" "285"

kruskal.test( cal~brand, data = df )

## 
##  Kruskal-Wallis rank sum test
## 
## data:  cal by brand
## Kruskal-Wallis chi-squared = 255.14, df = 9, p-value < 2.2e-16
```

10. Continuing with the former question, on the positive case which brands show a remarkable difference in mean behavior among them. Use one-sided tests.

Pairwise non-parametric one-sided tests can be applied. A brand mean calories are significantively greater than other brand means. D brand mean is greater than C brand mean at 5% significance level. E brand mean calories is greater than C and D brand means. F and G brand mean calories are greater that B, C, D, and E brand means. J brand mean calories is greater than I brand mean.

```
pairwise.wilcox.test( df$cal, df$brand, alternative="less" )

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity cor
rection
##
## data:  df$cal and df$brand
##
##   A       B       C       D       E       F       G       H       I
## B 5.8e-10 -       -       -       -       -       -       -       -
## C 2.1e-09 1.2e-06 -       -       -       -       -       -       -
## D 4.5e-10 0.00051 1.00000 -       -       -       -       -       -
## E 1.5e-09 1.00000 1.00000 1.00000 -       -       -       -       -
## F 7.9e-10 1.00000 1.00000 1.00000 1.00000 -       -       -       -
## G 1.1e-09 1.00000 1.00000 1.00000 1.00000 1.00000 -       -       -
## H 3.3e-10 1.00000 1.00000 1.00000 1.00000 8.3e-09 5.7e-09 -       -
## I 1.1e-09 5.8e-10 2.5e-09 4.5e-10 1.5e-09 7.9e-10 1.1e-09 3.3e-10 -
## J 4.5e-10 7.7e-08 1.00000 0.00039 1.3e-06 3.2e-10 4.5e-10 1.6e-07 1.00000
##
## P value adjustment method: holm

pairwise.wilcox.test( df$cal, df$brand, alternative="greater" )

##
##  Pairwise comparisons using Wilcoxon rank sum test with continuity cor
rection
##
## data:  df$cal and df$brand
##
##   A       B       C       D       E       F       G       H       I
## B 1.00000 -       -       -       -       -       -       -       -
## C 1.00000 1.00000 -       -       -       -       -       -       -
## D 1.00000 1.00000 0.03808 -       -       -       -       -       -
## E 1.00000 1.00000 4.3e-06 0.00014 -       -       -       -       -
## F 1.00000 1.2e-09 2.1e-09 3.2e-10 3.1e-06 -       -       -       -
## G 1.00000 3.3e-09 2.8e-09 6.5e-10 1.0e-06 1.00000 -       -       -
## H 1.00000 1.00000 1.1e-06 0.00013 1.00000 1.00000 1.00000 -       -
## I 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 -
## J 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 4.7e-10
##
## P value adjustment method: holm
```

11. The standard deviation of the number of calories for brand A should not exceed 0.15cal. For the simple random sample in your dataset calculate the

deviation of calories for 100g assuming a normal distribution for 100 g calories. Stating any assumptions, you need (write them), test at the 1% level the null hypothesis that the population standard deviation is not larger than 0.15cal against the alternative that it is.

Hipothesis testing for population variances on normal data is addressed using chi-squared distribution and sample variance. Null hypothesis is stated one-sided

$$H_0: \sigma^2 = 0.15^2$$

And

$$H_1: \sigma^2 > 0.15^2$$

Chi-squared statistic takes a 33.32 value with 29-1=28 degrees of freedom. P value is 0.224 > 0.01 significance level, thus H0 fails to be rejected taking into account the subsample defined for A brand. Using varTest() is possible and the same result is obtained once arguments are properly set.

```
tapply(df$cal,df$brand,sd)

##          A          B          C          D          E          F          G 
## 0.16362880 0.17197587 0.17629375 0.15283576 0.25444696 0.15374856 0.13431362 
##          H          I          J 
## 0.22080946 0.07277741 0.09873749

table(df$brand)

## 
##  A  B  C  D  E  F  G  H  I  J 
## 29 31 27 32 28 30 29 33 29 32

ss <- 0.16362880
# H0: sigma^2= 0.15^2  H1: sigma > 0.15^2 Normal population  (n-1)ss^2/si
gma^2 ~ X2(n-1)
# (n-1)ss^2/sigma^2

chi<-(29-1)*(ss^2)/(0.15^2);chi

## [1] 33.31923

1-pchisq(chi,28) # pvalue > 0.01 H0 can not be rejected

## [1] 0.2241883

# b - 99% CI
library(EnvStats)

x <- df$cal[df$brand=="A"];x

##  [1] 4.93 4.84 4.95 4.74 4.67 4.67 4.63 4.72 4.93 4.95 4.98 4.97 4.91 4.72 4.67
## [16] 4.77 4.47 4.46 4.53 4.66 5.08 4.97 4.68 4.70 4.56 4.80 4.91 4.79 4.78

varTest(x, sigma.squared=0.15^2, alternative="greater",conf.level=0.99)
```

```
##
##  Chi-Squared Test on Variance
##
## data:  x
## Chi-Squared = 33.319, df = 28, p-value = 0.2242
## alternative hypothesis: true variance is greater than 0.0225
## 99 percent confidence interval:
##  0.01552838        Inf
## sample estimates:
##   variance
## 0.02677438
```

12. Figure out the 99% upper threshold for the number of calories for brand A population variance. Normal distribution for calories is assumed to hold.

Using varTest() once arguments are properly set, an upper value of 0.05527 is obtained for the variance and thus 0.2351 for the standard deviation of A brand calories.

```
varTest(x, sigma.squared=0.15^2, alternative="less",conf.level=0.99)
```

```
##
##  Chi-Squared Test on Variance
##
## data:  x
## Chi-Squared = 33.319, df = 28, p-value = 0.7758
## alternative hypothesis: true variance is less than 0.0225
## 99 percent confidence interval:
##  0.00000000 0.05526714
## sample estimates:
##   variance
## 0.02677438
```

```
sqrt(0.05526714)
```

```
## [1] 0.2350896
```

13. Build a 99% confidence interval for the difference in the mean of 100 g calories between brands A and C. Assume that equal variances in the population calories per brand does not hold.

Equal variances for A and C brand calories can be tested using Fligner-Killeen test. Homogeneity of variances can not be rejected, thus a pooled variance can be calculated and a two-sided t.test assuming equal variances for the null hypothesis of mean calories being equal in both brands is clearly rejected. Mean cal difference (A - C) 99% CI is defined to be between 1.80 and 2.05.

```
tapply(df$cal,df$brand,mean)
```

```
##        A        B        C        D        E        F        G        H 
## 4.773793 3.190968 2.848889 3.003437 3.253929 3.596000 3.595172 3.224545
##        I        J
## 2.384138 2.878437
```

```
tapply(df$cal,df$brand,sd)
```

```
##          A          B          C          D          E          F          G
## 0.16362880 0.17197587 0.17629375 0.15283576 0.25444696 0.15374856 0.13431362
##          H          I          J
## 0.22080946 0.07277741 0.09873749
```

```
table(df$brand)
```

```
##
##  A  B  C  D  E  F  G  H  I  J
## 29 31 27 32 28 30 29 33 29 32
```

```
ll <- which( df$brand %in% c("A","C"))
dff <- df[ll,]
dff$brand <- factor(dff$brand)

muA <- 4.773793
muC <- 2.848889
ssA <-0.16362880
ssC <- 0.16362880

ssta <-((ssA^2)/29)
sstc <- ((ssC^2)/27)
deffree <- ((ssta+sstc)^2); deffree
```

```
## [1] 3.666836e-06
```

```
denom<- ((ssta^2)/28)+((sstc^2)/26);denom
```

```
## [1] 6.826427e-08
```

```
deffree <- deffree / denom; deffree
```

```
## [1] 53.71531
```

```
loth <- (muA - muC) - qt(0.995, deffree)*sqrt(((ssA^2)/29)+((ssC^2)/27));
loth
```

```
## [1] 1.808044
```

```
upth <- (muA - muC) + qt(0.995, deffree)*sqrt(((ssA^2)/29)+((ssC^2)/27));
upth
```

```
## [1] 2.041764
```

```
loth;upth
```

```
## [1] 1.808044
```

```
## [1] 2.041764
```

```
t.test(dff$cal~dff$brand, conf.level=0.99)
```

```
##
##  Welch Two Sample t-test
##
## data:  dff$cal by dff$brand
## t = 42.264, df = 52.858, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  1.803204 2.046605
## sample estimates:
## mean in group A mean in group C
##        4.773793        2.848889
```

```
fligner.test(dff$cal,dff$brand, conf.level=0.99)
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  dff$cal and dff$brand
## Fligner-Killeen:med chi-squared = 0.078778, df = 1, p-value = 0.779
```

```
t.test(dff$cal~dff$brand, conf.level=0.99, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  dff$cal by dff$brand
## t = 42.378, df = 54, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  1.803629 2.046180
## sample estimates:
## mean in group A mean in group C
##        4.773793        2.848889
```

14. Out of 100 people, 60 prefer A to C. Determine a 99% confidence interval for the population proportion that favors A in front of C. Test the null hypothesis that selecting A and C has equal probability.

An approximate proportion test can be addressed using prop.test() method and setting properly the arguments. Null hypothesis assuming equal A and C brand choice can not be rejected at the 1% significance level. 99% CI for A brand choice percentage lies between 47.14% and 71.61%.

```
prop.test(60, n=100, p=0.5, conf.level=0.99, correct=F) # H0 can not be r
ejected at 1%
```

```
##
##  1-sample proportions test without continuity correction
##
```

```
## data:  60 out of 100, null probability 0.5
## X-squared = 4, df = 1, p-value = 0.0455
## alternative hypothesis: true p is not equal to 0.5
## 99 percent confidence interval:
##  0.4714191 0.7161367
## sample estimates:
##   p
## 0.6
```

15. A second survey considered 200 people, 110 prefer A to C. Determine a 99% confidence interval for the difference in the population proportion that favors A in front of C accounting the two surveys. Test the null hypothesis that selecting A brand has a lower probability in the second of the surveys.

Difference in proportions stands for 1st - 2on survey comparison. One-sided test is stated as $H0: \pi_1 <= \pi_2$ *and the alternative hypothesis is* $H1: \pi_1 > \pi_2$*. H0 fails to be rejected and 99% CI for proportion difference has a lower threshold of -0.090 (0 is included).*

```
prop.test(c(60,110), n=c(100,200), conf.level=0.99, correct=F, alternativ
e="greater") # H0 can not be rejected at 1%
```

```
##
##  2-sample test for equality of proportions without continuity
##  correction
##
## data:  c(60, 110) out of c(100, 200)
## X-squared = 0.67873, df = 1, p-value = 0.205
## alternative hypothesis: greater
## 99 percent confidence interval:
##  -0.09030597  1.00000000
## sample estimates:
## prop 1 prop 2
##   0.60   0.55
```

Do not forget to Knit to .pdf before posting your answers in ATENEA.