Project Zencefil

2355170 - Abdullah Burkan Bereketoglu
2361244 - Eren Durali
2429363 - Fatma Ulasti
2468825 - Ozmen Cilesiz
2429330 - Yusuf Turan

12/12/2021

Abstract

In this project, we as a group analyzed a dataset called Abalone, which is not previously look at in the STAT291 lectures and recitations. Feature types, head, tail analysis and statistical measurements are done in the analysis. Enjoy!!

Contents

Introduction

Group Zencefil Welcomes you!!!

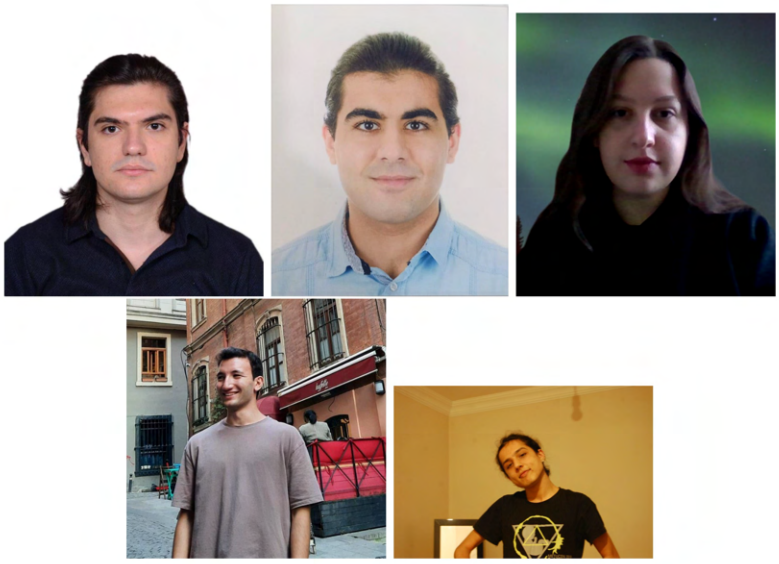

Figure 1: Images of Group Members- Burkan, Yusuf, Fatma, Özmen, Eren(Left-Up to Right-Down)

In this project our, the Group Zencefil's, aim was to locate a data-set, measure its statistical properties, analyze it's features(variables), in a basic manner, in each question there are steps completed for to accomplish this goal.

Question 1

In the first question, we will locate a package that is not used in the class. We used AppliedPredictiveModeling package and it's Abalone dataset. To check whether it has at least two numeric and at least one categoric we will look at the class of features, by str function. Data is from CRAN (Comprehensive R Archive Network)

```
#install.packages("AppliedPredictiveModeling")
library("AppliedPredictiveModeling")
data("abalone")

str(abalone)
```

```
## 'data.frame':    4177 obs. of  9 variables:
##  $ Type         : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
##  $ LongestShell : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 .
##  $ Diameter     : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44
##  $ Height       : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15
##  $ WholeWeight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ ShuckedWeight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
##  $ VisceraWeight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ ShellWeight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ..
##  $ Rings        : int  15 7 9 10 7 8 20 16 9 19 ...
```

```
head(abalone)
```

```
##   Type LongestShell Diameter Height WholeWeight ShuckedWeight VisceraWeight
```

```
## 1    M       0.455    0.365  0.095       0.5140         0.2245         0.1010
## 2    M       0.350    0.265  0.090       0.2255         0.0995         0.0485
## 3    F       0.530    0.420  0.135       0.6770         0.2565         0.1415
## 4    M       0.440    0.365  0.125       0.5160         0.2155         0.1140
## 5    I       0.330    0.255  0.080       0.2050         0.0895         0.0395
## 6    I       0.425    0.300  0.095       0.3515         0.1410         0.0775
##   ShellWeight Rings
## 1       0.150    15
## 2       0.070     7
## 3       0.210     9
## 4       0.155    10
## 5       0.055     7
## 6       0.120     8
```

Questions 1.1 (Brief Description of the data)

The data-set consists of data from 4177 Abalones. The dataset includes various types of measurements, such as; type of abalones (Male, Female, infant), height, longest Shell, diameter, rings, and various weight measures. The Abalone data-set has only 1 categorical variable (M,F,I), still passes the minimum threshold of 1 categorical, 2 numeric features(variables) requirement. Data-set mainly describes different Abalones with their different selected measures.

Question 2

```
library("AppliedPredictiveModeling")
data("abalone")
class(abalone)
```

```
## [1] "data.frame"
```

```
# or

is.data.frame(abalone)
```

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

Hence, here we can see that our data-set Abalone is indeed in the form of a data.frame .

Question 3

Here we need to ask two indexing questions, and answer them with the proper code, also one must answer it in dataframe.

Question 3.1

Get the 465th element from Diameter feature(variable)

```
library("AppliedPredictiveModeling")
data("abalone")

abalone$Diameter[465]
```

```
## [1] 0.195
```

```
# or
abalone[465, 3]
```

```
## [1] 0.195
```

```
# or
abalone[465,which(colnames(abalone) == "Diameter")] # If you don't know which col
```

```
## [1] 0.195
```

Question 3.2

Get the 2987th element of data-set Type variable(feature)

```
library("AppliedPredictiveModeling")
data("abalone")

abalone$Type[2987]
```

```
## [1] M
## Levels: F I M
```

```
# or
abalone[2987, 1]
```

```
## [1] M
## Levels: F I M
```

```
# or
abalone[2987,which(colnames(abalone) == "Type")] # If you don't know which column
```

```
## [1] M
## Levels: F I M
```

Question 4

In this question we will first, show if there is NA value in the data-set and count them for each column. Later we will continue with showing names of variables in our data-set. If the data-set doesn't have names for the features given, we need to give them names, hence we have names we will only show them.

In the third part of the question we create a new data frame called short_data with 10 rows from the head and 10 rows from the tail of the data. Continuing with showing mean, sd, median, and mode of the numeric features of the data-set, and contingency tables for some of our categorical variables/features in the data-set.

Question 4.1

Here is the part that, we show each column and their count of NA.

```
library("AppliedPredictiveModeling")
data("abalone")


i <- 1
while(i <= length(colnames(abalone))){
  print(sum(is.na(abalone[,i])))
  i = i + 1
}
```

```
## [1] 0
## [1] 0
```

```
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
## [1] 0
```

```
i <- 1

# or

anyNA(abalone) #our data frame has no NA value but this is not looking for each
```

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

From the results printed we can see that there is no NA value in any of our columns/variables. The method with while by just printing is not the most sufficient way, but for data-sets with features less than 20 it can be used fine since there won't be much uneasiness for the eyes of the viewer when looking at thousands of zeros, but just 20 or less.

Question 4.2

Here is the part that we continue with variable names, this has an easy function that works.

```
library("AppliedPredictiveModeling")
data("abalone")

colnames(abalone)
```

```
## [1] "Type"          "LongestShell"  "Diameter"      "Height"
## [5] "WholeWeight"   "ShuckedWeight" "VisceraWeight" "ShellWeight"
## [9] "Rings"
```

Question 4.3

Here we created the new data frame called short_data from head and tail part of the dataset by binding with row bind, with 10 from each taken (head and tail).

```
library("AppliedPredictiveModeling")
data("abalone")


short_data <- rbind(head(abalone, n = 10),tail(abalone, n = 10))
short_data
```

```
##      Type LongestShell Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1       M        0.455    0.365  0.095      0.5140        0.2245        0.1010
## 2       M        0.350    0.265  0.090      0.2255        0.0995        0.0485
## 3       F        0.530    0.420  0.135      0.6770        0.2565        0.1415
## 4       M        0.440    0.365  0.125      0.5160        0.2155        0.1140
## 5       I        0.330    0.255  0.080      0.2050        0.0895        0.0395
## 6       I        0.425    0.300  0.095      0.3515        0.1410        0.0775
## 7       F        0.530    0.415  0.150      0.7775        0.2370        0.1415
## 8       F        0.545    0.425  0.125      0.7680        0.2940        0.1495
## 9       M        0.475    0.370  0.125      0.5095        0.2165        0.1125
## 10      F        0.550    0.440  0.150      0.8945        0.3145        0.1510
## 4168    M        0.500    0.380  0.125      0.5770        0.2690        0.1265
## 4169    F        0.515    0.400  0.125      0.6150        0.2865        0.1230
## 4170    M        0.520    0.385  0.165      0.7910        0.3750        0.1800
## 4171    M        0.550    0.430  0.130      0.8395        0.3155        0.1955
## 4172    M        0.560    0.430  0.155      0.8675        0.4000        0.1720
## 4173    F        0.565    0.450  0.165      0.8870        0.3700        0.2390
## 4174    M        0.590    0.440  0.135      0.9660        0.4390        0.2145
## 4175    M        0.600    0.475  0.205      1.1760        0.5255        0.2875
## 4176    F        0.625    0.485  0.150      1.0945        0.5310        0.2610
## 4177    M        0.710    0.555  0.195      1.9485        0.9455        0.3765
##      ShellWeight Rings
## 1         0.1500    15
## 2         0.0700     7
## 3         0.2100     9
```

```
## 4         0.1550    10
## 5         0.0550     7
## 6         0.1200     8
## 7         0.3300    20
## 8         0.2600    16
## 9         0.1650     9
## 10        0.3200    19
## 4168      0.1535     9
## 4169      0.1765     8
## 4170      0.1815    10
## 4171      0.2405    10
## 4172      0.2290     8
## 4173      0.2490    11
## 4174      0.2605    10
## 4175      0.3080     9
## 4176      0.2960    10
## 4177      0.4950    12
```

Question 4.4

Here we solved the question of mean, median, standard deviation, contingency table, and mode of the data-set.

```
library("AppliedPredictiveModeling")
data("abalone")

mean = c()
median = c()
sd = c()
columnNames = c()


cout <- 1

while(cout <= length(colnames(abalone))){
  if (is.numeric(abalone[,cout])){
```

```
    columnNames = c(columnNames, names(abalone[cout]))
  }
  cout = cout + 1
}

cout <- 1

while(cout <= length(colnames(abalone))){
  if (is.numeric(abalone[,cout])){
    mean = c(mean, mean(abalone[,cout]))
    median = c(median, median(abalone[,cout]))
    sd = c(sd, sd(abalone[,cout]))
  }
  cout = cout + 1
}

cout <- 1
```

Question 4.4.1

Here we see, mean for each column with its name and their value.

```
mean = rbind(columnNames, mean)
mean
```

```
##             [,1]                [,2]                [,3]
## columnNames "LongestShell"      "Diameter"          "Height"
## mean        "0.523992099593009" "0.407881254488868" "0.139516399329662"
##             [,4]                [,5]                [,6]
## columnNames "WholeWeight"       "ShuckedWeight"     "VisceraWeight"
## mean        "0.828742159444577" "0.359367488628202" "0.180593607852526"
##             [,7]                [,8]
## columnNames "ShellWeight"       "Rings"
## mean        "0.238830859468518" "9.93368446253292"
```

Question 4.4.2

Here we see, median for each column with its name and their value.

```
median = rbind(columnNames,median)
median
```

```
##             [,1]           [,2]       [,3]     [,4]          [,5]
## columnNames "LongestShell" "Diameter" "Height" "WholeWeight" "ShuckedWeight"
## median      "0.545"        "0.425"    "0.14"   "0.7995"      "0.336"
##             [,6]            [,7]          [,8]
## columnNames "VisceraWeight" "ShellWeight" "Rings"
## median      "0.171"         "0.234"       "9"
```

Question 4.4.3

Here we see, standard deviation for each column with its name and their value.

```
sd = rbind(columnNames,sd)
sd
```

```
##             [,1]              [,2]                  [,3]
## columnNames "LongestShell"    "Diameter"            "Height"
## sd          "0.1200929125648" "0.0992398661336595" "0.0418270566072573"
##             [,4]                [,5]                [,6]
## columnNames "WholeWeight"       "ShuckedWeight"     "VisceraWeight"
## sd          "0.490389018230998" "0.22196294903322" "0.109614250259684"
##             [,7]                [,8]
## columnNames "ShellWeight"       "Rings"
## sd          "0.139202669522386" "3.22416903206813"
```

Question 4.4.4

Here we see, mode for each column with its name and their value.

```
library("AppliedPredictiveModeling")
data("abalone")

mode = c()

getMode <- function(x) {
  unique <- unique(x)
  modes <- tabulate(match(x, unique))
  unique[modes == max(modes)]
}

cout <- 1
while(cout <= length(columnNames)){
  if (is.numeric(abalone[,cout])){
    mode = c(mode, getMode(abalone[,cout]))
  }
  cout = cout + 1
}
cout <- 1

mode = rbind(columnNames, mode)

mode
```

```
##             [,1]            [,2]       [,3]     [,4]          [,5]
## columnNames "LongestShell" "Diameter" "Height" "WholeWeight" "ShuckedWeight"
## mode        "0.55"         "0.625"    "0.45"   "0.15"        "0.2225"
##             [,6]             [,7]          [,8]
## columnNames "VisceraWeight" "ShellWeight" "Rings"
## mode        "0.175"          "0.1715"      "0.275"
```

Question 4.4.5

Here we find the contingency table, but there is one important thing to say about this. Since we only had one categorical variable in our data-set, we just for this question added one more feature which is a categorical variable to make a 3x3 contingency table.

```r
separate <- 1

colors_variable <- c("red", "green", "blue")
ColorList = c()

while(separate <= length(abalone[,1])){
  ColorList = c(ColorList, rep(sample(colors_variable, 1)))
  separate = separate + 1
}

q_4_4_5 <- cbind(abalone,ColorList)

contingency_table <- table(q_4_4_5$ColorList, q_4_4_5$Type)
percent_contingency_table <- prop.table(table(q_4_4_5$ColorList, q_4_4_5$Type)) #

contingency_table
```

```
##
##            F   I   M
##   blue   426 451 497
##   green  444 435 509
##   red    437 456 522
```

```r
percent_contingency_table  # gives ratios in 0 and 1. Hence percents of continge
```

```
##
##                  F         I         M
##   blue   0.1019871 0.1079722 0.1189849
##   green  0.1062964 0.1041417 0.1218578
##   red    0.1046205 0.1091693 0.1249701
```

```r
rowSums(contingency_table)
```

```
##  blue green   red
##  1374  1388  1415
```

```
rowSums(percent_contingency_table)
```

```
##      blue     green       red
## 0.3289442 0.3322959 0.3387599
```

```
colSums(contingency_table)
```

```
##    F    I    M
## 1307 1342 1528
```

```
colSums(percent_contingency_table)
```

```
##         F         I         M
## 0.3129040 0.3212832 0.3658128
```

One can see from the percent contingency table that the percentage in the observations, that a Red Infant Abalone, or Blue Female Abalone.

Question 4.5

Here we order, hence sort our data-set with one of the variables/features in our data frame. We picked height feature and showed decreasing and increasing sorting for this variable.

```
library("AppliedPredictiveModeling")
data("abalone")

str(abalone) # I pick Height from features.
```

```
## 'data.frame':    4177 obs. of  9 variables:
##  $ Type         : Factor w/ 3 levels "F","I","M": 3 3 1 3 2 2 1 1 3 1 ...
##  $ LongestShell : num  0.455 0.35 0.53 0.44 0.33 0.425 0.53 0.545 0.475 0.55 .
##  $ Diameter     : num  0.365 0.265 0.42 0.365 0.255 0.3 0.415 0.425 0.37 0.44
```

```
##  $ Height       : num  0.095 0.09 0.135 0.125 0.08 0.095 0.15 0.125 0.125 0.15
##  $ WholeWeight  : num  0.514 0.226 0.677 0.516 0.205 ...
##  $ ShuckedWeight: num  0.2245 0.0995 0.2565 0.2155 0.0895 ...
##  $ VisceraWeight: num  0.101 0.0485 0.1415 0.114 0.0395 ...
##  $ ShellWeight  : num  0.15 0.07 0.21 0.155 0.055 0.12 0.33 0.26 0.165 0.32 ..
##  $ Rings        : int  15 7 9 10 7 8 20 16 9 19 ...
```

```
head(abalone[order(abalone$Height,decreasing = TRUE),],n = 5) # descending height
```

```
##      Type LongestShell Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 2052    F        0.455    0.355  1.130      0.5940        0.3320        0.1160
## 1418    M        0.705    0.565  0.515      2.2100        1.1075        0.4865
## 1429    F        0.815    0.650  0.250      2.2550        0.8905        0.4200
## 1764    M        0.775    0.630  0.250      2.7795        1.3485        0.7600
## 2180    F        0.595    0.470  0.250      1.2830        0.4620        0.2475
##      ShellWeight Rings
## 2052      0.1335     8
## 1418      0.5120    10
## 1429      0.7975    14
## 1764      0.5780    12
## 2180      0.4450    14
```

```
head(abalone[order(abalone$Height),],n = 5) # Ascending height order sorting for
```

```
##      Type LongestShell Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1258    I        0.430    0.340  0.000      0.4280        0.2065        0.0860
## 3997    I        0.315    0.230  0.000      0.1340        0.0575        0.0285
## 237     I        0.075    0.055  0.010      0.0020        0.0010        0.0005
## 1175    F        0.635    0.495  0.015      1.1565        0.5115        0.3080
## 2170    I        0.165    0.115  0.015      0.0145        0.0055        0.0030
##      ShellWeight Rings
## 1258      0.1150     8
## 3997      0.3505     6
## 237       0.0015     1
## 1175      0.2885     9
## 2170      0.0050     4
```

Question 5

Here we firstly make a linear combination of two or more of our numeric variables with our group name, which we picked as Zencefil. Later we will append it to our dataset for the next part, in the next part we will use loops and multiply one of our features/variables with another one element by element.

Question 5.1

Here we make the linear combination, and save this as a new column in our dataset. We assumed a cylindrical (without taking square) Abalone, and this variable is named as Zencefil.

```
library("AppliedPredictiveModeling")
data("abalone")

abalone_for_q_5 <- abalone

Zencefil = c()
values <- 1

while(values <= length(abalone$Height)){
  Zencefil = c(Zencefil, abalone$Height[values]*(abalone$Diameter[values]/2)*pi)
  values = values +1
}

abalone_for_q_5 <- cbind(abalone_for_q_5,Zencefil)
head(abalone_for_q_5, n = 1)
```

```
##   Type LongestShell Diameter Height WholeWeight ShuckedWeight VisceraWeight
## 1    M        0.455    0.365  0.095       0.514        0.2245         0.101
##   ShellWeight Rings   Zencefil
## 1        0.15    15 0.05446736
```

Question 5.2

```
library("AppliedPredictiveModeling")
data("abalone")

multiplication = c()

for(i in 1:length(abalone$ShuckedWeight)){
 multiplication = c(multiplication, abalone$ShuckedWeight[i] * abalone$VisceraWei
}

head(multiplication)
```

```
## [1] 0.02267450 0.00482575 0.03629475 0.02456700 0.00353525 0.01092750
```

```
length(multiplication)
```

```
## [1] 4177
```

Question 6

Here we create a new vector that only includes values from one of our numeric variables in abalone data-set, and these values are only the ones that are greater than the median of that feature. We again used Height variable for to create this new vector. The median of the height variable is 0.14 .

```
library("AppliedPredictiveModeling")
data("abalone")

abalone <- abalone_for_q_5

mynewvector <- abalone$Height[which(abalone$Height > median(abalone$Height))]
head(mynewvector, n = 10)
```

```
##  [1] 0.150 0.150 0.145 0.155 0.165 0.185 0.180 0.165 0.165 0.165
```

```
min(mynewvector) > median(abalone$Height)
```

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

Here from the minimum and median comparison, we return TRUE, hence we can say that we did the code right.

Question 7

Here we appended a new categorical variable by comparing two of the numeric variables in our data set. We compared whether they are equal, left of the comparison larger (Big), or smaller (Small). We compared VisceraWeight with ShellWeight since comparison in same dimensions make logical comparison table.

```
library("AppliedPredictiveModeling")
data("abalone")

abalone <- abalone_for_q_5

mynewcat = c()
for (i in 1:length(abalone$LongestShell)){
  if(abalone$VisceraWeight[i]>abalone$ShellWeight[i]){
    mynewcat = c(mynewcat,"Big")
  }else if(abalone$VisceraWeight[i]<abalone$ShellWeight[i]){
    mynewcat = c(mynewcat,"Small")
  }else{
    mynewcat = c(mynewcat,"Equal")
  }
}
length(mynewcat[which(mynewcat == "Equal")])
```

```
## [1] 12
```

```
length(mynewcat[which(mynewcat == "Small")])
```

```
## [1] 3893
```

```
length(mynewcat[which(mynewcat == "Big")])
```

```
## [1] 272
```

Here from the analysis one can see that most of the time Viscera Weight of Abalones are smaller than Shell Weights of the Abalones.

Question 8

Here in this question, we convert our data frame into a list, for that we use as.list rather than list since list accepts many items and as.list accepts only x variable, hence ignores any y and z as they are seen as not used arguments and coerces x argument to a list. e.g as.list(df) is df == list of df. df[1] = df-feature_one, df[2] == df -feature_two

On the other hand, list() accepts many arguments and builds a list on them, therefore will only make a list onto the data frame i.e list(df) = a list onto data frame e.g list_of_df[[1]] == df.

```
library("AppliedPredictiveModeling")
data("abalone")

abalone <- abalone_for_q_5

abalone_ls <- as.list(abalone)


### Question Answer for this part

sapply(as.list(abalone),class)
```

```
##          Type  LongestShell      Diameter        Height   WholeWeight
##      "factor"     "numeric"     "numeric"     "numeric"     "numeric"
## ShuckedWeight VisceraWeight   ShellWeight         Rings      Zencefil
##     "numeric"     "numeric"     "numeric"     "integer"     "numeric"
```

```
# One line code for showing the class of each member of the list
```

the sapply function shows the one line code for the class of each member of our list from our data set.

Question 9

Here we selected our list elements by their names, we made a list from the data frame that the previous part.

```
library("AppliedPredictiveModeling")
data("abalone")

names(abalone_ls)
```

```
##  [1] "Type"          "LongestShell"  "Diameter"      "Height"
##  [5] "WholeWeight"   "ShuckedWeight" "VisceraWeight" "ShellWeight"
##  [9] "Rings"         "Zencefil"
```

```
#Here the examples.
abalone_ls[["LongestShell"]][7]
```

```
## [1] 0.53
```

```
abalone_ls[["Diameter"]][9]
```

```
## [1] 0.37
```

```
abalone_ls[["Height"]][16]
```

```
## [1] 0.13
```

```
abalone_ls[["WholeWeight"]][65]
```

```
## [1] 0.58
```

```
abalone_ls[["ShuckedWeight"]][103]
```

```
## [1] 0.316
```

```
abalone_ls[["VisceraWeight"]][245]
```

```
## [1] 0.039
```

```
abalone_ls[["ShellWeight"]][306]
```

```
## [1] 0.011
```

```
abalone_ls[["Rings"]][486]
```

```
## [1] 13
```

Question 10

Here we will show the greatest values in the numeric variables in our list, and their place in our list as with their number and with their variable name.

```
library("AppliedPredictiveModeling")
data("abalone")

counter <- 1

while(counter <= length(names(abalone))){
  if(is.numeric(abalone_ls[[names(abalone_ls)[counter]]])){
    cat("My ", counter, " variable name is ",
        names(abalone_ls)[counter],
        " and the greatest value for my vector is ",
        max(abalone_ls[[names(abalone_ls)[counter]]]),
        "\n")
  }
  counter = counter + 1
}
```

```
## My  2  variable name is  LongestShell  and the greatest value for my vector is
## My  3  variable name is  Diameter  and the greatest value for my vector is  0.6
## My  4  variable name is  Height  and the greatest value for my vector is  1.13
## My  5  variable name is  WholeWeight  and the greatest value for my vector is
## My  6  variable name is  ShuckedWeight  and the greatest value for my vector is
## My  7  variable name is  VisceraWeight  and the greatest value for my vector is
## My  8  variable name is  ShellWeight  and the greatest value for my vector is
## My  9  variable name is  Rings  and the greatest value for my vector is  29
```

```
counter <- 1
```

Question 11

Here we will show the most frequent item in the categorical variables in our list, and their place in our list as with their number and with their variable name.

```
library("AppliedPredictiveModeling")
data("abalone")

counter <- 1

while(counter <= length(names(abalone))){
  if(!is.numeric(abalone_ls[[names(abalone_ls)[counter]]])){
    cat("My ", counter, " variable name is ",
        names(abalone_ls)[counter],
        " and the most frequent item for my vector is ",
        names(which.max(table(abalone_ls[[names(abalone_ls)[counter]]]))),
        "\n")
  }
  counter = counter + 1
}
```

```
## My  1  variable name is  Type  and the most frequent item for my vector is  M
```

```
counter <- 1
```

Question 12

Here we will randomly, make one of the list elements Null valued hence all the values will be removed from our list.

```
library("AppliedPredictiveModeling")
data("abalone")

abalone_ls[[sample(names(abalone_ls),1)]] <- NULL

length(colnames(abalone))
```

```
## [1] 9
```

```
length(names(abalone_ls))
```

```
## [1] 9
```

Normally after adding the group named variable we had 10 features in our system. But now we have 9, therefore we can deduce that one of them is now Null.

Question 13

At last, here we remove everything from our environment, then collect garbage and empty it with gc() for prevent overload. Lastly, we will show our directory of work.

```
rm(list=ls())
gc()
```

```
##           used (Mb) gc trigger (Mb) max used (Mb)
## Ncells  626964 33.5    1326544 70.9  1326544 70.9
## Vcells 1137361  8.7    8388608 64.0  8388585 64.0
```

```
getwd()
```

```
## [1] "C:/Users/Wilkins Inc/OneDrive/Documents"
```

Here we now end our great project Zencefil. Zencefil Group, Thanks for such an opportunity to work together and appreciate each other's talent and give support.

The End.

Sincerely Yours, Zencefil Group