Exploratory Data Analysis of the USDA data

Student number: 10333429 Student name: Galina Lopez Module: Statistics for Data Analysis

# Q1.

Explain the term Exploratory Data Analysis and list some of its goals.

In statistics, Exploratory Data Analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task. Exploratory data analysis was promoted by John Tukey to encourage statisticians to explore the data, and possibly formulate hypotheses that could lead to new data collection and experiments. EDA is different from initial data analysis (IDA), which focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

EDA is a critical first step in analyzing the data from an experiment. Main reasons we use EDA: 1. Detection of mistakes 2. Checking of assumptions 3. Preliminary selection of appropriate models 4. Determining relationships among the explanatory variables 5. Assessing the direction and rough size of relationships between explanatory and outcome variables. Loosely speaking, any method of looking at data that does not include formal statistical modeling and inference falls under the term exploratory data analysis.

The primary goal of EDA is to maximize the analyst's insight into a data set and into the underlying structure of a data set, while providing all of the specific items that an analyst would want to extract from a data set, such as: 1. A good-fitting, parsimonious model 2. A list of outliers 3. A sense of robustness of conclusions 4. Estimates for parameters 5. Uncertainties for those estimates 6. A ranked list of important factors 7. Conclusions as to whether individual factors are statistically significant 8. Optimal settings.

# Q2. Open R Studio

1. How do you check what Directory you are in?

Once you start R Studio you can type on the console window (also called command window) getwd() which stands for get working directory, hit return, and then R gives you the path to the folder you are in.

> getwd()

[1] "C:/Users/End-User/Documents/R"

II. How do you set your working Directory?

There are two ways to set your working directory.

At the command window you can type setwd("directoryname")

> setwd("C:/Users/End-User/Desktop/R")

Within R Studio you can also go to Session / Set working directory / Choose Directory

III. How do you clear the console?

To clear the console you simply go to Edit and select Clear Console.

IV. What happens when you use the Up Arrow on the keyboard?

If we use the Up Arrow on our keyboard we retrieve the last command we typed in.

V. How do you remove the non-available entries before computing statistical measures?

To remove the non-available entries before computing statistical measures we have to explicitly tell R to remove these non-available entries by typing na.rm=TRUE.

# Q3.

Read the USDA.csv in to R Studio and save the output to a data frame.

> USDA = read.csv("USDA.csv")

I. Use the structure function or str in R

> str(USDA)

|  |
| --- |
| 'data.frame': 7058 obs. of 16 variables:  $ ID : int 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...  $ Description : Factor w/ 7054 levels "ABALONE,MIXED SPECIES,RAW",..: 1303 1302 1298 2303 2304 2305 2306 2307 2308 2309 ...  $ Calories : int 717 717 876 353 371 334 300 376 403 387 ...  $ Protein : num 0.85 0.85 0.28 21.4 23.24 ...  $ TotalFat : num 81.1 81.1 99.5 28.7 29.7 ...  $ Carbohydrate: num 0.06 0.06 0 2.34 2.79 0.45 0.46 3.06 1.28 4.78 ...  $ Sodium : int 714 827 2 1395 560 629 842 690 621 700 ...  $ SaturatedFat: num 51.4 50.5 61.9 18.7 18.8 ...  $ Cholesterol : int 215 219 256 75 94 100 72 93 105 103 ...  $ Sugar : num 0.06 0.06 0 0.5 0.51 0.45 0.46 NA 0.52 NA ...  $ Calcium : int 24 24 4 528 674 184 388 673 721 643 ...  $ Iron : num 0.02 0.16 0 0.31 0.43 0.5 0.33 0.64 0.68 0.21 ...  $ Potassium : int 24 26 5 256 136 152 187 93 98 95 ...  $ VitaminC : num 0 0 0 0 0 0 0 0 0 0 ...  $ VitaminE : num 2.32 2.32 2.8 0.25 0.26 0.24 0.21 NA 0.29 NA ...  $ VitaminD : num 1.5 1.5 1.8 0.5 0.5 0.5 0.4 NA 0.6 NA ... |
|  |
| |  | | --- | |  | |

II. Use the summary function

> summary(USDA)

ID Description

Min. : 1001 BEEF,CHUCK,UNDER BLADE CNTR STEAK,BNLESS,DENVER CUT,LN,0" FA: 2

1st Qu.: 8387 CAMPBELL,CAMPBELL'S SEL MICROWAVEABLE BOWLS,HEA : 2

Median :13294 OIL,INDUSTRIAL,PALM KERNEL (HYDROGENATED),CONFECTION FAT : 2

Mean :14260 POPCORN,OIL-POPPED,LOFAT : 2

3rd Qu.:18337 ABALONE,MIXED SPECIES,RAW : 1

Max. :93600 ABALONE,MXD SP,CKD,FRIED : 1

(Other) :7048

Calories Protein TotalFat Carbohydrate Sodium

Min. : 0.0 Min. : 0.00 Min. : 0.00 Min. : 0.00 Min. : 0.0

1st Qu.: 85.0 1st Qu.: 2.29 1st Qu.: 0.72 1st Qu.: 0.00 1st Qu.: 37.0

Median :181.0 Median : 8.20 Median : 4.37 Median : 7.13 Median : 79.0

Mean :219.7 Mean :11.71 Mean : 10.32 Mean : 20.70 Mean : 322.1

3rd Qu.:331.0 3rd Qu.:20.43 3rd Qu.: 12.70 3rd Qu.: 28.17 3rd Qu.: 386.0

Max. :902.0 Max. :88.32 Max. :100.00 Max. :100.00 Max. :38758.0

NA's :1 NA's :1 NA's :1 NA's :1 NA's :84

SaturatedFat Cholesterol Sugar Calcium

Min. : 0.000 Min. : 0.00 Min. : 0.000 Min. : 0.00

1st Qu.: 0.172 1st Qu.: 0.00 1st Qu.: 0.000 1st Qu.: 9.00

Median : 1.256 Median : 3.00 Median : 1.395 Median : 19.00

Mean : 3.452 Mean : 41.55 Mean : 8.257 Mean : 73.53

3rd Qu.: 4.028 3rd Qu.: 69.00 3rd Qu.: 7.875 3rd Qu.: 56.00

Max. :95.600 Max. :3100.00 Max. :99.800 Max. :7364.00

NA's :301 NA's :288 NA's :1910 NA's :136

Iron Potassium VitaminC VitaminE

Min. : 0.000 Min. : 0.0 Min. : 0.000 Min. : 0.000

1st Qu.: 0.520 1st Qu.: 135.0 1st Qu.: 0.000 1st Qu.: 0.120

Median : 1.330 Median : 250.0 Median : 0.000 Median : 0.270

Mean : 2.828 Mean : 301.4 Mean : 9.436 Mean : 1.488

3rd Qu.: 2.620 3rd Qu.: 348.0 3rd Qu.: 3.100 3rd Qu.: 0.710

Max. :123.600 Max. :16500.0 Max. :2400.000 Max. :149.400

NA's :123 NA's :409 NA's :332 NA's :2720

VitaminD

Min. : 0.0000

1st Qu.: 0.0000

Median : 0.0000

Mean : 0.5769

3rd Qu.: 0.1000

Max. :250.0000

NA's :2834

What information those this give us about the USDA file?

To learn about our data we use the structure function or str in R and give it the input USDA. This gives us the following information. We have 7,058 observations or foods in our dataset along with 16 different variables. The first variable gives a unique identification number for each of the foods, starting with the number 1,001. The second variable gives a text description of each of the foods. The third variable is the amount of calories in 100 grams of these foods and it is given to us in kilocalories. We also have information about the protein, total fat, carbohydrate, saturated fat and sugar levels in grams, as well as the sodium, cholesterol, calcium, iron, potassium and vitamin C levels in milligrams. Finally, the amount of vitamin E and vitamin D in what is known as international units, as this is the standard measurement for drugs and vitamins.

To obtain high-level statistical information about our dataset we use the summary function and give it as input the USDA data frame. The summary function gives us information such as the minimum, the maximum and the mean values across all 7,058 foods for each of the 16 different variables. For instance, the maximum amount of saturated fat is 95,600 milligrams, whereas the mean is only 3,452 milligrams. We also have information about the number of non-available entries. For instance, we have 2,720 foods that are missing entries for their vitamin E levels. Scrolling through this information, a startling observation is the maximum level of sodium, which is 38,758 milligrams. This number is huge, given that the daily recommended maximum is only 2,300 milligrams. We can investigate this variable further with some basic data analysis.

# Q4.

We are told too much Salt. Check the values of sodium levels in the foods within the data set. Find the min and max foods.

We realized when we scroll through the summary of the USDA data frame, that the maximum level of Sodium was 38,758 milligrams, which is very high. To check the values of sodium levels in the foods within the data set, we can type USDA$Sodium.

> USDA$Sodium

This gives us a series of numbers associated with the amount of sodium in all the foods in our data set. This is a vector associated with the variable Sodium. For instance, the sodium level of the first food in our data set is 714 milligrams and the sodium level of the last food in our data set is 68 milligrams.

To find which food has the highest level of sodium we use the function which.max, which takes as an input the Sodium vector and it gives us the index of the food with the highest sodium level.

> which.max(USDA$Sodium)

[1] 265

In this case, the 265th food in our data set has the maximum sodium content. To know which food that is, we need to take a look at the vector corresponding to the text description of the foods. We can use the function names, which takes as an input the USDA data frame to give us the exact names of all the variables as stored in the USDA data frame.

> names(USDA)

[1] "ID" "Description" "Calories" "Protein" "TotalFat"

[6] "Carbohydrate" "Sodium" "SaturatedFat" "Cholesterol" "Sugar"

[11] "Calcium" "Iron" "Potassium" "VitaminC" "VitaminE"

[16] "VitaminD"

And now we know that the name of the variable we are looking at is Description. To get the name of the 256th food, we need to ask R to pick the 256th element from the vector Description. So, using our dollar notation to call the Description vector and then the square brackets around the index 265 and we get table salt.

> USDA$Description[265]

[1] SALT,TABLE

7054 Levels: ABALONE,MIXED SPECIES,RAW ABALONE,MXD SP,CKD,FRIED ... ZWIEBACK

We also realized when we scroll through the summary of the USDA data frame, that the minimum level of Sodium was 0 milligrams.

To find which food this belongs to, i.e. has the lowest level of sodium we use the function which.min, which takes as an input the Sodium vector and it gives us the index of the food with the lowest sodium level.

> which.min(USDA$Sodium)

[1] 289

In this case, the 289th food in our data set has the minimum sodium content. To know which food that is, we need to get the name of the 289th food, we need to ask R to pick the 289th element from the vector Description. So, using our dollar notation to call the Description vector and then the square brackets around the index 289 and we get beef tallow fat.

> USDA$Description[289]

[1] FAT,BEEF TALLOW

7054 Levels: ABALONE,MIXED SPECIES,RAW ABALONE,MXD SP,CKD,FRIED ... ZWIEBACK

# Q5.

I. Create a new data frame of foods with more than 10,000 milligrams of sodium as a subset of our original data frame. Show your code and the results of your analysis.

From our previous analysis, 38,758 milligrams of sodium in 100 grams of table salt makes sense but none of us would eat 100 grams of salt in one sitting. So, it is more interesting to find out which foods contain more than 10,000 milligrams of sodium. To do so, we create a new data frame and call it HighSodium. And this is going to be a subset of our original data frame, USDA, with only the foods that have sodium content that exceeds 10,000.

> HighSodium = subset(USDA, Sodium>10000)

And now we created this new data frame, to see how many foods there exist in this new data frame, we need to see how many observations is data frame has. This can be done by using the function nrow, which computes the number of rows in the data frame HighSodium.

> nrow(HighSodium)

[1] 10

And we obtain 10 foods with sodium levels above 10,000 milligrams. Since there are not many we can output the names of these foods by looking at their Description vector. But this time, the Description vector is not associated with the USDA data frame but with the HighSodium data frame.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| > HighSodium$Description  [1] SALT,TABLE  [2] SOUP,BF BROTH OR BOUILLON,PDR,DRY  [3] SOUP,BEEF BROTH,CUBED,DRY  [4] SOUP,CHICK BROTH OR BOUILLON,DRY  [5] SOUP,CHICK BROTH CUBES,DRY  [6] GRAVY,AU JUS,DRY  [7] ADOBO FRESCO  [8] LEAVENING AGENTS,BAKING PDR,DOUBLE-ACTING,NA AL SULFATE  [9] LEAVENING AGENTS,BAKING SODA  [10] DESSERTS,RENNIN,TABLETS,UNSWTND  7054 Levels: ABALONE,MIXED SPECIES,RAW ABALONE,MXD SP,CKD,FRIED ... ZWIEBACK  We obtain the names of these 10 food. So definitely table salt is one of them. We also have dry soup, gravy,  fresco adobo, some leavening agents and some unsweetened tablets rennin desserts.  The values we worked with so far with respect to high levels of sodium are 10,000 milligrams and 38,000  milligrams, maybe the best way to figure out how big a value is, is by comparing it to the mean and the  standard deviation of the sodium levels across the data set. To find the mean, we use the summary  function, and this time, give it as input the Sodium vector instead of the whole USDA data frame.  > summary(USDA$Sodium)  Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  0.0 37.0 79.0 322.1 386.0 38760.0 84  We can see that the mean sodium value is 322.1 milligrams. However, the summary function does not give  us the standard deviation information, but we can do this using the function sd, which stands for standard  deviation. Give it as input the sodium vector and explicitly tell R to remove the non-available entries by  typing na.rm=TRUE.  > sd(USDA$Sodium, na.rm = TRUE)  [1] 1045.417  And the standard deviation is 1,045.415 milligrams.  Now that we have done a basic analysis of our data set we can look at the plotting functionality in R.  III. Explain Visualization.  Visualization is a crucial step for initial data exploration. It helps us discern relationships, patterns and outliers.  There are three ways in which we can visualize our data, using scatterplots, histograms and box plots. Q6. Create the following plots from the USDA file using appropriate labels on the x-axis and the y-axis.  Add titles to all plots. Show all plots you draw and include some comments on the plots. Explain how  you exported your plots from R Studio and what formats your plots could be saved to.  I. Scatterplot  To create a scatterplot with Protein on the x-axis and fat on the y-axis, we can use the plot function in R and give it  as a first input the Protein vector on the x-axis, and as a second input the TotalFat vector on the y-axis. For the  aesthetics of the graph, since R gives default names for the x-axis and the y-axis using the vector dollar  notation, we modify these labels by adding more arguments to the plot function. Add the argument , xlab = "Protein"  and that gives us the label of the x-axis. Then ylab=”Fat” gives us the label to the y-axis. To add a title "Protein vs. Fat"  to the plot use the argument main. Can also change the colour to red.  > plot(USDA$Protein, USDA$TotalFat, xlab="Protein", ylab = "Fat", main = "Protein vs Fat", col = "red")  Pressing Enter and going back to the graphics window, we see R made all the modifications we requested.    The plot has a very interesting triangular shape. It looks like foods that are higher in protein are typically lower  in fat and vice versa.  II. Histogram  Another way we can visualize our data is by plotting histograms. We can do this using the histogram function in R,  Which takes only one variable as an input, because the y-axis should have the frequencies. To create a histogram  for instance of VitaminC, we are going to use the hist function or histogram. The argument it takes is the VitaminC  vector. We label the x-axis as “Vitamin C”, this is given to us in milligrams. And give it a title, “Histogram of Vitamin  C Levels”.  Even though the maximum vitamin C content is 2000 milligrams, most of our foods,, to be more precise, more  than 6,000 of them--have less than 200 milligrams of vitamin C. And the histogram lumps them all together in  one cell. To zoom into this section and get a finer understanding of the data, we need to limit the x-axis to go  from zero to, say, 100 milligrams. So we are going to add the argument xlim to limit the x-axis. And using the  combine function or the c function, we set the first input to be zero, which is the lowest value that we want to  see on the x-axis, and the second argument as being 100, which is the highest value that we want to see on the  x-axis. Pressing Enter, we see R gives us 0 to 100 on the x-axis. But we only see this one big cell. It seems that R  only zoomed into the area, but it didn't break that huge cell, and this doesn't give us any additional information.  So we need to break up the cell into smaller pieces. Say we want 100 cells, and since the interval goes from 0 to  100, then we would expect R to create divisions that are one milligram in length. So we add the argument  breaks = 100 and this sets the number of cells we want to see to 100.  We see five cells, and each cell is 20 milligrams long. The histogram originally went far beyond 100 milligrams.  The maximum was 2000 milligrams. And now if we were to divide the original interval from 0 to 2000 into 100  cells, then 2000 divided by 100, each cell would be 20 milligrams long. It actually divided all of the spectrum of  values from 0 to 2000 into 100 cells, and not only the spectrum from 0 to 100.  We still want to divide the interval 0 to 100 into 100 cells, each of length 1 milligram. We need to set the breaks  to 2000. And now we obtain our refined histogram.  > hist(USDA$VitaminC, xlab = "Vitamin C (mg)", main = "Histogram of Vitamin C")  > hist(USDA$VitaminC, xlab = "Vitamin C (mg)", main = "Histogram of Vitamin C", xlim = c(0,100))  > hist(USDA$VitaminC, xlab = "Vitamin C (mg)", main = "Histogram of Vitamin C", xlim = c(0,100), breaks=100)  > hist(USDA$VitaminC, xlab = "Vitamin C (mg)", main = "Histogram of Vitamin C", xlim = c(0,100), breaks=2000)    We see new information here come up. Our initial conclusion was that more than 6,000 foods have less than  200 milligrams of vitamin C. But now that we refined our graph, we obtained additional level of information.  Actually, more than 5,000 of them have less than one milligram of vitamin C.  III. Box plot  Third way we can visualize the data is using box plots. To create a box plot for sugar, the function we use is boxplot,  which takes as an input a single vector and in this case it would be the Sugar vector. We add a title that says  “Boxplot of Sugar Levels” and label the y-axis as the sugar level by setting it to “Sugar (g)". And now we have our  box plot with the right labels.  > boxplot(USDA$Sugar, ylab = "Sugar (g)", main = "Boxplot of Sugar")    It looks a bit strange. The average of sugar across the data set seems to be pretty low, somewhere around  five grams. But we have outliers with extremely high values of sugar. There exist some foods that have almost  100 grams of sugar in 100 grams. Candies are definitely among these foods.  So we just reviewed three ways in which we visualized our data.  We exported the plots from R Studio by going into the Graphics window and clicking Export then selecting  Save as Image and finally selecting jpeg as Image format. Other formats our plots could be saved to are:  png, tiff, bmp, metafile, svg, eps. Once in Microsoft Word we just inserted them as pictures into the project.  By clicking the Insert tab, selecting Pictures and finding the image on our working directory. Q7. How can we add a new variable to the USDA data frame that takes the value 1 if the food has higher  sodium than average and 0 if the food has lower sodium than average?  To check if the first food in the dataset has a higher amount of sodium compared to the average, we can simply  ask R to dig up the first value in the Sodium vector, using the square brackets and the index 1. And then compare  it using the greater-than sign to the mean of the Sodium vector, and then do not forget to remove the non-available  entries. And we obtain TRUE. How about the 50th food? Well, let's go back using the Up arrow, and simply change  the index I to 50, and now we get FALSE. This means that the first food has higher sodium content than average,  and the 50th food has lower sodium content than average. Now, we can write the same command, but on all the  vector Sodium. Let's use the Up arrow, and delete the square brackets with the index 50. But we know we have  7,000 foods, and we really don't want to output 7,000 values right now. So how about instead, we just save the  output to a vector, and we're going to call it HighSodium. And now let's look at the structure of the HighSodium  vector. And then we see that the HighSodium vector indeed has all these values-- TRUE and FALSE -- which are  called logicals. So basically the type of the HighSodium vector is logical. But remember, we said we wanted values  1's and 0's. So instead of TRUE, we want 1. And instead of FALSE, we want a value of 0. Well, to do this, we need to  change the data type of HighSodium to numeric, and we can do this using the as.numeric function. So let's use the  Up arrow twice, and then enclose this logical expression by the as.numeric function. So as.numeric, and now look  up the structure of HighSodium, and now we see that we turned it into a numerical vector with values 0's and I's.  Now, this vector, HighSodium, is not associated with the USDA data frame. How can we add a variable, HighSodium,  to our dataframe? Well, simply we need to use the dollar notation. So let's go back twice to the command where  we created the HighSodium vector, and then simply right now, instead ofjust calling it HighSodium, we associate it  with the USDA data frame using the dollar notation. Now, pressing Enter, and going and checking the structure of  the USDA data frame, we see that we just added the HighSodium variable that was not present before, and it's a  numerical variable with values 1's and 0's.  > HighSodium = as.numeric(USDA$Sodium > mean(USDA$Sodium, na.rm=TRUE))  > str(HighSodium)  num [1:7058] 1 1 0 1 1 1 1 1 1 1 ...  > USDA$HighSodium = as.numeric(USDA$Sodium > mean(USDA$Sodium, na.rm=TRUE))  I. Add new variables for HighProtein, HighCarbs and HighFat to the USDA data frame.  Now we can do the same, and add the variables HighProtein, HighCarbs, HighFat, similarly to our data frame.  Well, let's do this quickly using the Up arrow, and then let's go and replace Sodium now by Protein. So again,  here Sodium is replaced by Protein. And then we're going to call this new variable HighProtein. And do the same  with TotalFat. So instead of Protein, we're going to have TotalFat, and then replace it here again, and the variable  name is going to be HighFat. And finally, Carbohydrates-- so here is the vector of Carbohydrates, and this is, too,  getting the Carbohydrates vector. And finally, this last variable that we want to add is called HighCarbs. And now  looking at the structure of the USDA data frame, we see that we successfully added these three new variables,  which are high HighProtein, HighFat, and HighCarbs, in addition to the HighSodium variable that we added  previously.  II. Show the code you used and the new structure of the USDA data frame.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | > USDA$HighCarbs = as.numeric(USDA$Carbohydrate > mean(USDA$Carbohydrate, na.rm=TRUE))  > USDA$HighProtein = as.numeric(USDA$Protein > mean(USDA$Protein, na.rm=TRUE))  > USDA$HighFat = as.numeric(USDA$TotalFat > mean(USDA$TotalFat, na.rm=TRUE))   |  | | --- | | > str(USDA)  'data.frame': 7058 obs. of 20 variables:  $ ID : int 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...  $ Description : Factor w/ 7054 levels "ABALONE,MIXED SPECIES,RAW",..: 1303 1302 1298 2303 2304 2305 2306 2307 2308 2309 ...  $ Calories : int 717 717 876 353 371 334 300 376 403 387 ...  $ Protein : num 0.85 0.85 0.28 21.4 23.24 ...  $ TotalFat : num 81.1 81.1 99.5 28.7 29.7 ...  $ Carbohydrate: num 0.06 0.06 0 2.34 2.79 0.45 0.46 3.06 1.28 4.78 ...  $ Sodium : int 714 827 2 1395 560 629 842 690 621 700 ...  $ SaturatedFat: num 51.4 50.5 61.9 18.7 18.8 ...  $ Cholesterol : int 215 219 256 75 94 100 72 93 105 103 ...  $ Sugar : num 0.06 0.06 0 0.5 0.51 0.45 0.46 NA 0.52 NA ...  $ Calcium : int 24 24 4 528 674 184 388 673 721 643 ...  $ Iron : num 0.02 0.16 0 0.31 0.43 0.5 0.33 0.64 0.68 0.21 ...  $ Potassium : int 24 26 5 256 136 152 187 93 98 95 ...  $ VitaminC : num 0 0 0 0 0 0 0 0 0 0 ...  $ VitaminE : num 2.32 2.32 2.8 0.25 0.26 0.24 0.21 NA 0.29 NA ...  $ VitaminD : num 1.5 1.5 1.8 0.5 0.5 0.5 0.4 NA 0.6 NA ...  $ HighSodium : num 1 1 0 1 1 1 1 1 1 1 ...  $ HighCarbs : num 0 0 0 0 0 0 0 0 0 0 ...  $ HighProtein : num 0 0 0 1 1 1 1 1 1 1 ...  $ HighFat : num 1 1 1 1 1 1 1 1 1 1 ... | |  | | |  | | --- | | Q8. We created four new variables, HighSodium, HighFat, HighCarbs, and HighProtein, let’s try to understand our data and the relationships between our variables better, using the table and tapply functions. Show the code you used and the results of this analysis and briefly discuss your findings.  We will try to understand our data and the relationships between our variables better, using the table and tapply functions. To figure out how many foods have higher sodium level than average, we want to look at the HighSodium variable and count the foods that have values 1. We can do this using the table function, and give  It as an input the HighSodium vector.  > table(USDA$HighSodium)  0 1  4884 2090  We obtain the following information. Most of the foods in our data set, and precisely 4,800 of them, have lower sodium than average, and we have 2090 foods that have higher sodium content than average.  Now let's see how many foods have both high sodium and high fat.  To do this we can also use the table function, but instead of giving it one input, now we can give it two inputs,  the first input being the HighSodium vector and the second input being the HighFat vector. And we obtain the following table.  > table(USDA$HighSodium, USDA$HighFat)    0 1  0 3529 1355  1 1378 712  The rows belong to the first input, which is HighSodium, and the columns correspond to the second input,  which is HighFat. So from the table we see that we have 3,529 foods with low sodium and low fat, 1,355 foods  with low sodium and high fat, 1,378 foods with high sodium but low fat, and finally 712 foods with both  high sodium and high fat.  If we want to compute the average amount of iron sorted by high and low protein, we use the tapply  function. The tapply function takes three arguments, and groups the first argument according to the  second argument, and then applies argument three. For instance, we wanted to compute the average  amount of iron sorted by high and low protein. In this case, the first argument is whatever we are trying  to analyse, which is the Iron vector, and we are sorting it according to the vector HighProtein, which is  our second argument. And finally we apply the mean function in R on the sorted Iron values.  And we should not forget to remove the non-available entries.  > tapply(USDA$Iron, USDA$HighProtein, mean, na.rm=TRUE)  0 1  2.558945 3.197294  Foods with low protein content have on average 2.55 milligrams of iron and foods with high protein content  have on average 3.2 milligrams of iron.  Now how about the maximum level of vitamin C in foods with high and low carbs?  Again, we're going to use the tapply function, so use the up arrow to go back to the previous command,  but now we're trying to analyse the VitaminC vector. So this is our first argument and we are sorting it  according to high and low carbs, so the second argument is the vector HighCarbs.  And instead of the mean, we're applying here the max statistic, and we obtain the following.  > tapply(USDA$VitaminC, USDA$HighCarbs, max, na.rm=TRUE)  0 1  1677.6 2400.0  The maximum vitamin C level, which is 2,400 milligrams is actually present in a food that is high in carbs.  Is it true that foods that are high in carbs have generally high vitamin C content? To see if this is the case,  we're going to go back to our tapply function, and instead of the max statistic we're going to use the summary function.  > tapply(USDA$VitaminC, USDA$HighCarbs, summary, na.rm=TRUE)  $`0`  Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  0.000 0.000 0.000 6.364 2.800 1678.000 248  $`1`  Min. 1st Qu. Median Mean 3rd Qu. Max. NA's  0.00 0.00 0.20 16.31 4.50 2400.00 83  We obtain the above two sets of information. The first set corresponds to the foods with low carb content,  and the second set of information corresponds to foods with higher carb content than average. Now the  statistical information that the summary function gives us pertains to the vitamin C levels. This means that  we have on average 6.36 milligrams of vitamin c in foods with low carb content, and on average 16.31 milligrams  of vitamin C in foods with high carb content. So, it does seem like a general trend. Foods with high carb content  are on average richer in vitamin C compared to foods with low carb content. Q9. What did you learn from the Exploratory Data Analysis of the USDA data?  Through this Exploratory Data Analysis of the USDA data we got more familiar with R and the R Studio  environment and we learn to perform basic functions to read, visualize and analyse the data set. We  looked at the nutritional data of over 7000 foods distributed by the US Department of Agricultural to  understand various nutritional facts about food, while getting more practice using R.  Good nutrition is an important part of leading a healthy lifestyle. Obesity has been rising at an alarming  rate in the US for instance, while all states in 1990 had less than 14% obesity, figures started increasing.  By 2000, half of the country has more than 20% of its population obese. The trend continues. In 2010, all  states had at least more than 20% of their population obese. Many states across the country reached an  alarming situation. More than 35% of American adults are obese.  The trends Worldwide are no different. Obesity has nearly doubled across the globe. Obesity is one of  today’s visible public health problems and increases people’s risk of heart disease, stroke and diabetes.  So good nutrition is essential for an overall healthy lifestyle and promoting it now is more important than  ever. We have access to hundreds of nutrition and weight loss applications and around 15% of adults with  mobile devices use health applications on them. These apps are mostly powered by the United States  Department of Agricultural (USDA) food database.  Example of these apps can be found in the apple and google stores.  Apps like Calorie Counter – MyFitnessPal in the google app store. A health and fitness app that uses the  largest food database by far (over 5,000,000 foods). This amazingly fast and easy to use calorie counter  helps its users to lose weight.  The USDA distributes nutritional information of over 7000 food items including amount of calories, carbs,  protein, fat, sodium, among other nutrients. It is exactly this data that we analysed.  While performing basic analysis of this data set, we learn about our data. The structure function or str in  R showed us the basic structure of our data set. We had 7058 observations or foods in our dataset along  with 16 different variables . The first variable gave a unique identification number for each of the foods, starting with the number 1,001. The second variable gave a text description of each of the foods. The third variable was the amount of calories in 100 grams of these foods and it was given to us in kilocalories. We also have information about the protein, total fat, carbohydrate, saturated fat and sugar levels in grams, as well as the sodium, cholesterol, calcium, iron, potassium and vitamin C levels in milligrams. Finally, the amount of vitamin E and vitamin D in what is known as international units, as this is the standard measurement for drugs and vitamins.  To obtain high-level statistical information about our dataset we used the summary function and gave it as input the USDA data frame. The summary function gave us information such as the minimum, the maximum and the mean values across all 7,058 foods for each of the 16 different variables. For instance, the maximum amount of saturated fat was 95,600 milligrams, whereas the mean was only 3,452 milligrams. We also had information about the number of non-available entries. For instance, we have 2,720 foods that are missing entries for their vitamin E levels.  Scrolling through this information, a startling observation was the maximum level of sodium, which was 38,758 milligrams. This number is huge, given that the daily recommended maximum is only 2,300 milligrams. We investigated this Sodium variable further with some basic data analysis. We found out which food this corresponds to. The result was table salt. We also realized when we scroll through the summary of the USDA data frame, that the minimum level of Sodium was 0 milligrams. We found which food this belongs to and we got beef tallow fat.  From our previous analysis, 38,758 milligrams of sodium in 100 grams of table salt makes sense but none of us would eat 100 grams of salt in one sitting. So, it was more interesting to find out which foods contain more than 10,000 milligrams of sodium. To do so, we created a new data frame and called it HighSodium. And this was going to be a subset of our original data frame, USDA, with only the foods that have sodium content that exceeds 10,000.  We obtained 10 foods and got the names of these 10 food. So definitely table salt was one of them. We also had dry soup, gravy, fresco adobo, some leavening agents and some unsweetened tablets rennin desserts.  The values we worked with so far with respect to high levels of sodium were 10,000 milligrams and 38,000  milligrams, maybe the best way to figure out how big a value was, was by comparing it to the mean and the  standard deviation of the sodium levels across the data set. To find the mean, we used the summary function,  and this time, gave it as input the Sodium vector instead of the whole USDA data frame. We could see that the mean sodium value was 322.1 milligrams. And using the sd function we saw that the standard deviation was 1,045.415 milligrams. By summing the mean and the standard deviation we obtained around 1400 milligrams, which is a more meaningful value to compare against. For example, we thought caviar would be high in sodium but it was not in our top ten high sodium foods. So we explored it further.  USDA$Sodium[match("CAVIAR", USDA$Description)]  Got its sodium level, 1500 milligrams and when we compared it to the average 1400 we concluded that caviar was pretty rich in sodium compared to most foods in our data set.  After completing this basic analysis of our data set we looked at the plotting functionality in R.  Visualization was a crucial step for initial data exploration. It helped us discern relationships, patterns and outliers.  There were three ways in which we visualized our data, using scatterplots, histograms and box plots.  We created a scatterplot of Protein vs Fat using the plot function. The plot had a very interesting triangular shape. It looked like foods that are higher in protein are typically lower in fat and vice versa. We learned to adjust the aesthetics of the graph in R by adding labels to the x and y axis, adding a title and changing the colour.  Another way we visualised our data was by plotting histograms. We created a histogram of VitaminC using the histogram function. We also labelled the x-axis and gave it a title. We then performed a series of refinements to our original histogram until we got the desired one. From this refined histogram, new information came up. Our initial conclusion was that more than 6,000 foods had less than 200 milligrams of vitamin C. But now that we refined our graph, we obtained additional level of information. Actually, more than 5,000 of them had less than one milligram of vitamin C.  Third way we visualized the data was using box plots. We created a box plot for sugar, the function we used was boxplot. It looked a bit strange. The average of sugar across the data set seemed to be pretty low, somewhere around five grams. But we had outliers with extremely high values of sugar. There exist some foods that had almost 100 grams of sugar in 100 grams. Candies were definitely among these foods.  So we reviewed three ways in which we visualized our data.  We exported the plots from R Studio by going into the Graphics window and clicking Export then selecting  Save as Image and finally selecting jpeg as Image format. Other formats our plots could be saved to are:  png, tiff, bmp, metafile, svg, eps. Once in Microsoft Word we just inserted them as pictures into the project.  By clicking the Insert tab, selecting Pictures and finding the image on our working directory.  We created four new variables, HighSodium, HighFat, HighCarbs, and HighProtein and used the table and tapply functions to understand our data and the relationships between our four new variables better. The table and tapply functions helped us find relationships between these variables and also the original variables in the USDA.  With the table function we found:  How many foods have higher sodium level than average?  Most of the foods in our data set, and precisely 4,800 of them, have lower sodium than average, and we have 2090 foods that have higher sodium content than average.  How many foods have both high sodium and high fat?  We have 3,529 foods with low sodium and low fat, 1,355 foods with low sodium and high fat, 1,378 foods with high sodium but low fat, and finally 712 foods with both high sodium and high fat.  With the tapply function we found:  Average amount of iron sorted by high and low protein.  Foods with low protein content had on average 2.55 milligrams of iron and foods with high protein content  had on average 3.2 milligrams of iron.  Maximum level of Vitamin C in foods with high and low carbs.  The maximum vitamin C level, which was 2,400 milligrams was actually present in a food that was high in carbs.  Finally we used the summary function with tapply to check if is true that foods that are high in carbs have generally high vitamin C content? We had on average 6.36 milligrams of vitamin C in foods with low carb content and on average 16.31 milligrams of vitamin C in foods with high carb content. So, it did seem like a general trend. Foods with high carb content were on average richer in vitamin C compared to foods with low carb content.  This EDA of the USDA data was a great way to familiarise ourselves better with R, and learn some interesting facts about nutrition. | | | | References. Class notes, Exploratory Data Analysis.  Google app store.  NIST, Exploratory Data Analysis.  Wikipedia, Exploratory Data Analysis. | |  | |
|  |
| |  | | --- | |  | |