G4.P-1

David Emanuel Craciunescu   Laura Pérez Medeiro

October 15, 2019

Introduction

The main purpose of this introductory laboratorio practice is to get students accustomed to the R development environment and to teach to these the basics of data science with such environment and language.

Work-wise, the practice is divided in the two following parts:

Development on provided dataset

In this section, the student will have to formalize and develop on a provided exercise and given dataset by the professor. This does not only enable the professor to analyze if such student has completely and correctly understood the contents of the laboratory practice, but it also serves as a starting point for any student unable to face the workload of such project without any kind of previous guidance or help.

This specific practice provides the dataset *satelites.txt*, which contains the **names** and **radii** of some of the most common moons of Uranus. Working with this dataset has been thoroughly explained by the professor in the almost three laboratory classes dedicated to this specific practice, so an explanation or any kind of further analysis into this specific dataset is considered not only redundant, but time-wasting as well.

Development on obtained dataset

This section of the laboratory practice aims to create a deeper understanding in the nature of dataset retrieval by the students and to enhance their ability in the labors of recognising good quality data sources and valid information retrieval from the various analysis performed on these datasets.

The chosen data source for this exercise was scraped off the webpage *www.a-z-animals.com*, and the data itself is formed by tuples of the names of animals and their respective life expentancies. A Python scraping script has been developed solely with the objective to obtain this data.

Also, as extra work, the authors have completed the analysis of all the mentioned datasets in Python as well.

Data analysis

Satelites dataset

This dataset has been provided in the *txt* format. In order to input it into R, the following command must be used.

```
> while (!"satelites.txt" %in% list.files(getwd()))
+ {
+ print("Data file not found. Add \"satelites.txt\" to the current directory.")
+ invisible(readline(prompt="Press [enter] to continue"))
+ }
> satelites <- read.table("satelites.txt")
> satelites
```

```
           nombre radio
1        CORDELIA    13
2          OFELIA    16
3          BIANCA    22
4         CRESIDA    33
5       LESDEMONA    39
6         JULIETA    42
7       ROSALINDA    27
8         BELINDA    34
9  LUNA-1986U1020    20
10       CALIBANO    30
11       LUNA-119    20
12     LUNA_119U2    15
```

```
> radius <- satelites $radio
> radius
```

```
 [1] 13 16 22 33 39 42 27 34 20 30 20 15
```

```
>
```

When loading external datasets into R, it is important to take into account that the working directory must be the same as the file's directory when calling read.table(). Otherwise, one the route to where the found can be found must be indicated.

Once the data has been read, the authors will proceed to analyze it in the following way:

Absolute and relative frequencies

Absolute Frequency

```
> absoluteFreq <- function(set) {table(set)}
> absoluteFreq(radius)
```

```
set
13 15 16 20 22 27 30 33 34 39 42
 1  1  1  2  1  1  1  1  1  1  1
```

Cumulative Absolute Frequency

```
> cumAbsoluteFreq <- function(set) {cumsum(absoluteFreq(set))}
> cumAbsoluteFreq(radius)
```

```
13 15 16 20 22 27 30 33 34 39 42
 1  2  3  5  6  7  8  9 10 11 12
```

Relative Frequency

```
> relativeFreq <- function(set) {table(set) / length(set)}
> relativeFreq(radius)
```

```
set
        13         15         16         20         22         27         30         33
0.08333333 0.08333333 0.08333333 0.16666667 0.08333333 0.08333333 0.08333333 0.08333333 0.
```

Cumulative Relative Frequency

```
> cumRelativeFreq <- function(set) {cumsum(relativeFreq(set))}
> cumRelativeFreq(radius)
```

```
        13         15         16         20         22         27         30         33
0.08333333 0.16666667 0.25000000 0.41666667 0.50000000 0.58333333 0.66666667 0.75000000 0.
```

Arithmetic mean

```
>
>
```

```
> arithmeticMean <- function(set, usrTrim = 0) (mean(set, trim = usrTrim))
> arithmeticMean(radius)
```

```
[1] 25.91667
```

```
>
>
>
```

Measures of dispersion

For this specific section, the following webpage has been used as a http://iridl.ldeo.columbia.edu/dochelp/Stat
- RANGE:

```
> range <- function(set) {max(set) - min(set)}
> range(radius)
```

```
[1] 29
```

- STANDARD DEVIATION

```
> stdDeviation <- function(set)
+ {
+ sd(set) * (sqrt((length(set) - 1) / length(set)))
+ }
> stdDeviation(radius)
```

```
[1] 9.277736
```

- VARIANCE:

```
> variance <- function(set) {var(set) * (length(set) - 1 / length(set))}
> variance(radius)
```

```
[1] 1118.993
```

```
>
>
>
```

- ROOT MEAN SQUARE:

```
> rootMeanSqr <- function(set) {sqrt(mean(set ^ 2))}
> rootMeanSqr(radius)
```

```
[1] 27.52726
```

```
>
>
>
```

- ROOT MEAN SQUARE ANOMALY:

```
> rootMeanSqrAn <- function(set) {sqrt(sum(set - mean(set)) ^ 2) / length(set)}
> rootMeanSqrAn(radius)
```

```
[1] 1.184238e-15
```

```
>
>
>
```

- INTERQUARTILE RANGE:

```
> interQuartRange <- function(set) {IQR(set)}
> interQuartRange(radius)
```

```
[1] 14.25
```

```
>
>
>
```

- MEDIAN ABSOLUTE DEVIATION

```
> medAbsDeviation <- function(set) {mad(set)}
> medAbsDeviation(radius)
```

```
[1] 12.6021
```

```
>
>
>
```

d) Finally, measures of order:
-MEDIAN:

```
> getMedian <- function(set) {median(set)}
> getMedian(radius)
```

```
[1] 24.5
```

```
>
>
>
```

-MODE:

```
> getMode <- function(set)
+
+
+
+ {
+
+
+
+ uniqueVal <- unique(set)
+
+
+
+ uniqueVal[which.max(tabulate(match(set, uniqueVal)))]
+
+
+
+ }
> getMode(radius)
```

```
[1] 20
```

```
>
>
>
```

-QUARTILES:

```
> getQuartiles <- function(set) {quantile(set)}
> getQuartiles(radius)
```

```
   0%   25%   50%   75%  100%
13.00 19.00 24.50 33.25 42.00
```

```
>
>
>
```

-54th QUANTILE:

```
> getQuantiles <- function(set, range = 0) {quantile(set, probs = range)}
> getQuantiles(radius)
```

```
0%
13
```

```
>
>
>
```

Cardata dataset

The same analysis the authors have performed on the previous dataset will be performed on the Cardata dataset. This time, the variable to use will be called *mpg* and the 54th quantile and the frequencies are not needed.

In order to analyze *.sav* format, R needs to import the *foreign* library.

```
> library(foreign)
>
```

Once the file is read, only the data related to *mpg* is going to matter. Also, there may be empty rows or NAs in these records, one must filter these in order to perform a correct statistical analysis.

```
> dataset = read.spss("cardata.sav", to.data.frame=TRUE)
> mpg = dataset$mpg
> mpg = mpg[!is.na(mpg)]
```

Once the data is prepared, the exact same functions as the previous section can be used.

Arithmetic mean

```
> arithmeticMean(mpg)
```

```
[1] 28.79351
```

Measures of dispersion

Range

```
> range(mpg)
```

```
[1] 31.1
```

```
>
>
>
```

Standard Deviation

```
> stdDeviation(mpg)
```

[1] 7.353219

```
>
>
>
```

Variance

```
> variance(mpg)
```

[1] 8380.823

Root mean square

```
> rootMeanSqr(mpg)
```

[1] 29.7176

Root mean square anomaly

```
> rootMeanSqrAn(mpg)
```

[1] 1.522592e-15

Interquartile range

```
> interQuartRange(mpg)
```

[1] 11.725

```
>
>
>
```

Median absolute deviation

```
> medAbsDeviation(mpg)
```

[1] 8.37669

```
>
>
>
```

Measures of order

Median

```
> getMedian(mpg)
```

[1] 28.9

Mode

```
> getMode(mpg)
```

[1] 36

Cardata dataset

The same analysis the authors have performed on the previous dataset will be performed on the Cardata dataset. This time, the variable to use will be called *mpg* and the 54th quantile and the frequencies are not needed.

animals <- read.csv(head= T, sep=",", "animals2.csv")

animals

lifespan <- animals $lifespanlifespan$

a) Calculate absolute and relative satellite animals frequencies:

ABSOLUTE FRECUENCY:

```
> absoluteFreq(lifespan)

set
  0.5     1   1.5     2   2.5     3   3.5     4   4.5     5   5.5     6   6.5     7   7.5
    1    20     3    17     2    25     3    11     5    17     3    10     5     8     1
   13  13.5    14  14.5    15  15.5    16    17  17.5    18  18.5    19    20    21  21.5
   11     4    34     1    83     1    14     1     8    12     3     4    26     1     2
   30  32.5    35  37.5    40  42.5  47.5    50  52.5    55  57.5    60    65    70    75
   14     2     7     3     9     3     2     9     1     8     1     5     1     2     1
```

ACUMULATIVE ABSOLUTE FRECUENCY:

```
> cumAbsoluteFreq(lifespan)

  0.5     1   1.5     2   2.5     3   3.5     4   4.5     5   5.5     6   6.5     7   7.5
    1    21    24    41    43    68    71    82    87   104   107   117   122   130   131
   13  13.5    14  14.5    15  15.5    16    17  17.5    18  18.5    19    20    21  21.5
  298   302   336   337   420   421   435   436   444   456   459   463   489   490   492
   30  32.5    35  37.5    40  42.5  47.5    50  52.5    55  57.5    60    65    70    75
  527   529   536   539   548   551   553   562   563   571   572   577   578   580   581

>
```

RELATIVE FRECUENCY

```
> relativeFreq(lifespan)

set
        0.5           1         1.5           2         2.5           3         3.5
0.001709402 0.034188034 0.005128205 0.029059829 0.003418803 0.042735043 0.005128205 0.0188
          6         6.5           7         7.5           8         8.5           9
0.017094017 0.008547009 0.013675214 0.001709402 0.039316239 0.006837607 0.006837607 0.0957
       12.5          13        13.5          14        14.5          15        15.5
0.008547009 0.018803419 0.006837607 0.058119658 0.001709402 0.141880342 0.001709402 0.0239
       18.5          19          20          21        21.5          22        22.5
0.005128205 0.006837607 0.044444444 0.001709402 0.003418803 0.003418803 0.010256410 0.0034
       27.5          28          30        32.5          35        37.5          40
0.001709402 0.001709402 0.023931624 0.003418803 0.011965812 0.005128205 0.015384615 0.0051
         55        57.5          60          65          70          75          80
0.013675214 0.001709402 0.008547009 0.001709402 0.003418803 0.001709402 0.001709402 0.0017
```

```
>
>
>
```

ACUMULATIVE RELATIVE FRECUENCY

```
> cumRelativeFreq(lifespan)

        0.5           1         1.5           2         2.5           3         3.5
0.001709402 0.035897436 0.041025641 0.070085470 0.073504274 0.116239316 0.121367521 0.1401
          6         6.5           7         7.5           8         8.5           9
0.200000000 0.208547009 0.222222222 0.223931624 0.263247863 0.270085470 0.276923077 0.3726
       12.5          13        13.5          14        14.5          15        15.5
0.490598291 0.509401709 0.516239316 0.574358974 0.576068376 0.717948718 0.719658120 0.7435
       18.5          19          20          21        21.5          22        22.5
0.784615385 0.791452991 0.835897436 0.837606838 0.841025641 0.844444444 0.854700855 0.8581
       27.5          28          30        32.5          35        37.5          40
0.875213675 0.876923077 0.900854701 0.904273504 0.916239316 0.921367521 0.936752137 0.9418
         55        57.5          60          65          70          75          80
0.976068376 0.977777778 0.986324786 0.988034188 0.991452991 0.993162393 0.994871795 0.9965

>
>
>
```

b) Arithmetic mean

```
> arithmeticMean(lifespan)

[1] 15.86581

>
>
>
```

c) Measures of dispersion, where the following page was used as a reference for this section:
http://iridl.ldeo.columbia.edu/dochelp/StatTutorial/Dispersion/index.htmlIntro
- RANGE:

```
> range(lifespan)

[1] 124.5

>
```

- STANDARD DEVIATION

```
> stdDeviation(lifespan)

[1] 14.4033
```

```
>
>
>
```

- VARIANCE:

```
> variance(lifespan)

[1] 121568.7

>
>
>
```

- ROOT MEAN SQUARE:

```
> rootMeanSqr(lifespan)

[1] 21.42846

>
>
>
```

- ROOT MEAN SQUARE ANOMALY:

```
> rootMeanSqrAn(lifespan)

[1] 3.491984e-16

>
>
>
```

- INTERQUARTILE RANGE:

```
> interQuartRange(lifespan)

[1] 9.5

>
>
>
```

- MEDIAN ABSOLUTE DEVIATION

```
> medAbsDeviation(lifespan)

[1] 7.413

>
>
>
```

d) Finally, measures of order:
-MEDIAN:

```
> getMedian(lifespan)

[1] 13

>
>
>
```

-MODE:

```
> getMode(lifespan)

[1] 15

>
>
>
```

-QUARTILES:

```
> getQuartiles(lifespan)

   0%   25%   50%   75%  100% 
  0.5   8.0  13.0  17.5 125.0 

>
>
>
```

-54th QUANTILE:

```
> getQuantiles(lifespan)

 0% 
0.5 

>
>
>
```