Introduction to Data Science With

OLGUN AYDIN

Lead Data Analyst- zingat

olgun.aydin@zingat.com
info@olgunaydin.com

Outline

- The R
- ***Fundementals***
  - Vectors & Matrix
  - Import & Export Data
  - Probablities ,Distributions, Random Numbers
  - Descriptive Statistics & Graphs
  - Scripts
- *Applications*
  - Correlation & Regression Analysis
  - Clustering
  - Time Series Analysis
  - Parametric / Non-Parametric Analysis

The R

- R is a statistical computer program made available through the Internet under the General Public License (GPL).
- That is, it is supplied with a license that allows you to use it freely, distribute it, or even sell it, as long asthe receiver has the same rights and the source code is freely available.
- R provides an environment in which you can perform statistical analysis and produce graphics.

The R

- https://www.r-project.org/
- https://cran.r-project.org/
- https://www.rstudio.com/
- https://journal.r-project.org/
- http://www.inside-r.org/
- http://www.r-bloggers.com/

Vectors & Matrix

- One of the simplest possible tasks in R is to enter an arithmetic expression and receive a result.

```
> 2 + 2
[1] 4
> exp(-2)
[1] 0.1353353
```

```
> x <- 2
> x
[1] 2
> x + x
[1] 4
```

Vectors & Matrix

- A data vector is simply an array of numbers, and a vector variable can be constructed like this:

```
> weight <- c(60, 72, 57, 90, 95, 72)
> weight
[1] 60 72 57 90 95 72
```

- The body mass index (BMI) is defined for each person as the weight in kilograms divided by the square of the height in meters. This could be calculated as follows:

```
height <- c(1.75, 1.80, 1.65, 1.90, 1.74, 1.91)
> bmi <- weight/height^2
> bmi
[1] 19.59184 22.22222 20.93664 24.93075 31.37799 19.73630
```

Vectors & Matrix

- It's possible to concatenate vectors of more than one element as in

```
> x <- c(1, 2, 3)
> y <- c(10, 20)
> c(x, y, 5)
[1] 1 2 3 10 20 5
>length(c(x, y, 5))
[1] 6
```

- It is also possible to assign names to the elements. This modifies the way the vector is printed and is often used for display purposes.

```
> x <- c(name="Olgun", surname="Aydin", comp="REIDIN")
> x
name surname comp
"Olgun" "Aydin" "REIDIN"
```

Vectors & Matrix

- The second function, seq ("sequence"), is used for equidistant series of numbers. Writing
- `> seq(4,9)`
- `[1] 4 5 6 7 8 9`

- The integers from 4 to 9. If you want a sequence in jumps of 2, write
- `> seq(4,10,2)`
- `[1] 4 6 8 10`

- The rep function is often used for things such as group codes: If it is known that the first 10 observations are men and the last 15 are women,you can use

- `> rep(1:2,c(10,15))`
- `[1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2`

Vectors & Matrix

- A convenient way to create matrices is to use the matrix function:

```
a<-matrix(1:9,3,3)
> a
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- Product a with its transpose

```
a%*%t(a)
     [,1] [,2] [,3]
[1,]   66   78   90
[2,]   78   93  108
[3,]   90  108  126
```

Vectors & Matrix

- A convenient way to create matrices is to use the matrix function:

```
a<-matrix(1:9,3,3)
> a
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

- Product a with its transpose

```
a%*%t(a)
     [,1] [,2] [,3]
[1,]   66   78   90
[2,]   78   93  108
[3,]   90  108  126
```

Vectors & Matrix

- Inverse of a calculated as follows:
- `solve(a)`

- To get dimension of matrix
- `> dim(a)`
- `[1] 3 3`

- Can "glue" vectors together, columnwise or rowwise, using the cbind and rbind functions.

```
> cbind(A=1:4,B=5:8,C=9:12)
     A B  C
[1,] 1 5  9
[2,] 2 6 10
[3,] 3 7 11
[4,] 4 8 12
```

Vectors & Matrix

- `> rbind(A=1:4,B=5:8,C=9:12)`

```
   [,1] [,2] [,3] [,4]
A    1    2    3    4
B    5    6    7    8
C    9   10   11   12
```

Import & Export Data

- **Get working directory**

- `getwd()`
- `[1] "C:/Users/ol/Desktop“`

- **Set working directory**

- `>setwd("C:/Users/ol/Documents")`
- `getwd()`
- `[1] "C:/Users/ol/Documents”`

Import & Export Data

- **From A Comma Delimited Text File**
- `mydata <- read.table("c:/mydata.csv", header=TRUE, sep=",", row.names="id")`

- **From Excel**
- `library(xlsx)`
- `mydata <- read.xlsx("c:/myexcel.xlsx", sheetName=mysheet")`

- **From SPSS**
- `library(Hmisc)`
- `mydata <- spss.get("c:/mydata.por",use.value.labels=TRUE)`

Import & Export Data

- **List files in working directory**

```
>list.files()
```

- [1] "adaptive7_h200.csv"
- [2] "adaptive8_h100.csv"
- [3] "adaptive8_h250.csv"
- [4] "adaptive9_h200.csv"
- [5] "adaptive9_h200_2.csv"
- [6] "adaptive9_h300.csv"

- **List objects in current workspace**
- ls()
- [1] "a" "boundary" "data" "smry"

Import & Export Data

- **Show first 2 rows of the matrix**

- `> head(data,2)`

```
  CityID  CityName CountyID CountyName ActivityTypeID ActivityType PropertySubTypeID PropertySubTypeName AmountSize to_char
1    231 İstanbul      248  Kadıköy             11        Sales               119              Office   6141.429    2015
2    231 İstanbul      248  Kadıköy             11        Sales               119              Office   5646.226    2015
```

- **Show coloumn names of matrix**

- `>colnames(data)`

```
 [1] "CityID"             "CityName"           "CountyID"           "CountyName"          "ActivityTypeID"
 [6] "ActivityType"       "PropertySubTypeID"  "PropertySubTypeName" "AmountSize"          "to_char"
```

- **Show first 2 rows of "AmountSize" coloumn**

- `head(data$AmountSize,2)`
- `[1] 6141.429 5646.226`

Probablities &Distributions&Random Numbers

- In R, you can simulate these situations with the sample function. If you want to pick five numbers at random from the set 1:40, then you can write

- `> sample(1:40,5)`
- `[1] 4 30 28 40 13`

- There are five possibilities for the first number, and for each of these there are four possibilities for the second, and so forth; that is, the number is 5 × 4 × 3 × 2 × 1. This number is also written as 5! (5 factorial).

- `>prod(5:1)`
- `[1] 120`

Probablities &Distributions&Random Numbers

- So all we need to do is to calculate the number of ways to choose 5 numbers out of 40.
- `> 1/choose(40,5)`
- `[1] 1.519738e-06`

- way of creating the plot is to use curve as follows:
- `> curve(dnorm(x), from=-4, to=4)`

- For discrete distributions, where variables can take on only distinct values,it is preferable to draw a pin diagram, here for the binomial distribution with n = 50 and p = 0.33
- `> x <- 0:50`
- `> plot(x,dbinom(x,size=50,prob=.33),type="h")`

Probablities &Distributions&Random Numbers

- Say that it is known that some biochemical measure in healthy individuals is well described by a normal distribution with a mean of 132 and a standard deviation of 13. Then, if a patient has a value of less than 160, there is

- `> 1-pnorm(160,mean=132,sd=13)`
- `[1] 0.01562612`

- `> pnorm(1.96)`
- `[1] 0.9750021`

- `> 1-pnorm(1.96)`
- `[1] 0.0249979`

- `> qnorm(0.975)`
- `[1] 1.959964`

Probablities &Distributions&Random Numbers

- The use of the functions that generate random numbers is straightforward. The first argument specifies the number of random numbers to compute, and the bsequent arguments are similar to those for other functions related to the same distributions. For instance,

```
> rnorm(10)
[1] -0.2996466 -0.1718510 -0.1955634 1.2280843 -2.6074190
[6] -0.2999453 -0.4655102 -1.5680666 1.2545876 -1.8028839
```

```
>rnorm(10,mean=7,sd=5)
[1] 8.934983 8.611855 4.675578 3.670129 4.223117 5.484290
[7] 12.141946 8.057541 -2.893164 13.590586
```

```
> rbinom(10,size=20,prob=.5)
[1] 12 11 10 8 11 8 11 8 8 13
```

Descriptive Statistics & Graphs

- It is easy to calculate simple summary statistics with R. Here is how to calculate the mean, standard deviation, variance, and median.

```
> x <- rnorm(50)
> mean(x)
[1] 0.03301363
> sd(x)
[1] 1.069454
> var(x)
[1] 1.143731
> median(x)
[1] -0.08682795
```

Descriptive Statistics & Graphs

- **Empirical quantiles may be obtained with the function quantile like this:**

```
>quantile(data$AmountSize)
        0%        25%        50%        75%       100%
    0.5685   360.5000   588.4818   929.3559 33636.3636
```

- **Summary statistics**

```
>summary(data$AmountSize)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
    0.57   360.50   588.50   792.80   929.40 33640.00
```

Descriptive Statistics & Graphs

- **Empirical quantiles may be obtained with the function quantile like this:**
- `>quantile(data$AmountSize)`
- `        0%        25%        50%        75%       100%`
- `    0.5685   360.5000   588.4818   929.3559 33636.3636`

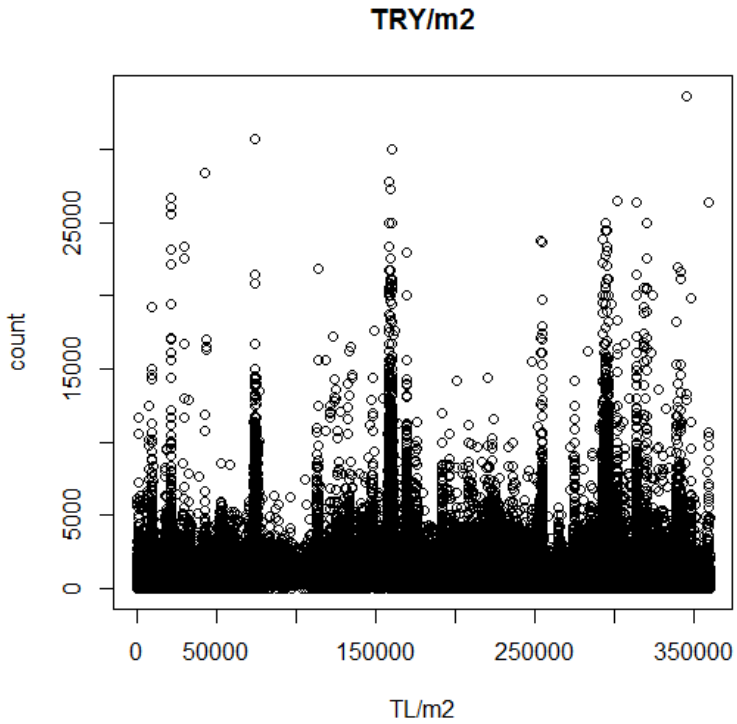
- **Summary statistics**
- `>summary(data$AmountSize)`
- `    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.`
- `    0.57   360.50   588.50   792.80   929.40 33640.00`

- **Learn how many rows of AmountSize less than 1000 USD**
- `> length(data$AmountSize[data$AmountSize<1000])`
- `[1] 281621`

Descriptive Statistics & Graphs

- **Draw plot of AmountSize**

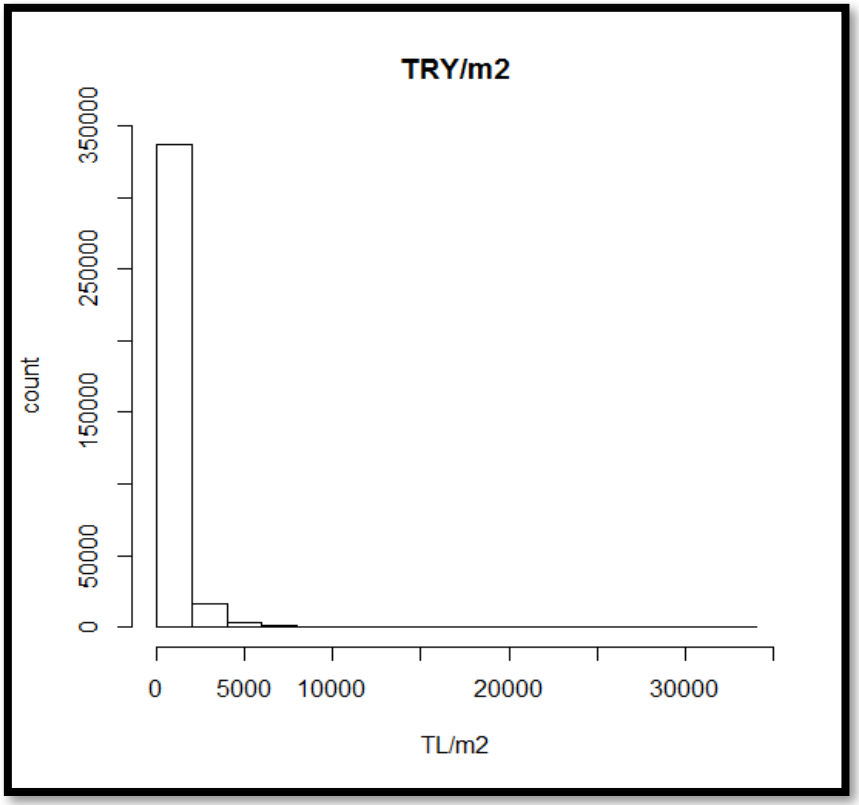
```
>plot(data$AmountSize,main="TRY/m2",xlab="TL/m2",ylab="count")
```



Descriptive Statistics & Graphs

- **Draw histogram of AmountSize**

```
>hist(data$AmountSize,main="TRY/m2",xlab="TL/m2",ylab="count")
```



Summarize

- The doBy package provides much of the functionality of SAS PROC SUMMARY. It defines the desired table using a model formula and a function. Here is a simple example.

```
>library(doBy)
>summaryBy(mpg + wt ~ cyl + vs, data = mtcars, FUN = function(x) { c(m =
  mean(x), s = sd(x)) } )
```

Summarize

```
boundary<- function(x)
{
Pricetran<-log(x)
meantdata<-mean(Pricetran)
stdtdata<-sd(Pricetran)

skw<-skewness(x)
kurt<-kurtosis(x)

UpB_2_51<-exp(meantdata+2.51*stdtdata)
LoB_2_51<-exp(meantdata-2.51*stdtdata)

return(c(skw,kurt,LoB_2_51,UpB_2_51)
}

smry <-
   summaryBy(AmountSize~CityID+CityName+CountyID+CountyName+ActivityTypeID+Activ
   ityType+PropertySubTypeName, FUN=boundary, data=data)
```

Applications

- ***Correlation & Regression Analysis***
- ***Clustering***
- ***Time Series Analysis***
- ***Parametric / Non-Parametric Analysis***