Applied Statistics with R

Topic 3: Descriptive Statistical Analysis. Data Visualization in R

Kaloyan Ganev

2022/2023

Lecture contents

Some more stuff on data frames

Dimensions, elements...

- Let's use some data on Bulgarian municipalities population
- Download Pop_6.1.1_Pop_DR_EN.xls from the course page (or from the NSI webpage)
- Read the data into R:

```
pop <- read_xls("Pop_6.1.1_Pop_DR_EN.xls",
  skip = 4,
  na = "-",
  sheet = "2021")
```

- The pop object is a data frame
- Remove columns containing data on urban and rural populations, leaving just totals

```
pop <- pop[,1:4]
```

- . . . and country totals

```
pop <- pop[-1,]
```

Dimensions, elements...(2)

- Rename columns

```
colnames(pop)[2:4] <- c("Total", "Male", "Female")
```

- Check data frame dimensions with:

```
nrow(pop)
ncol(pop)
```

- Access a specific element by row and column

```
pop[2,3]
```

- Access variables by names:

```
pop$Total, pop$Male, pop$Female
```

Descriptive statistics

Basic descriptive statistics

- Note: We work with samples, so we need sample formulas!
- R uses them by default
- Sample minimum and maximum of a variable:

```
max(pop$Total)
min(pop$Total)
```

- To get the minimum and the maximum at the same time, i. e. the range of data:

```
range(pop$Male)
```

- How about finding the observation to to which the maximum and the minimum belong?
- There are special functions for this:

```
which.max(pop$Total)
which.min(pop$Total)
```

Basic descriptive statistics (2)

- Now, let’s write a complete program (very simple) which will tell us which city has the maximum population figure when we run it
- The code of the program is as follows:

```r
# This program will display a message stating the city with the
    maximum population
maxm <- which.max(pop$Total)
city <- pop[maxm, 1, drop = TRUE]
cat("The Bulgarian municipality with the largest population (
    totalling", max(pop$Total), "inhabitants) is", city, ".")
```

- Note: The `drop = TRUE` argument is necessary because a *tibble* is subset (not a regular data frame)

Basic descriptive statistics (3)

- Median: the value that lies in the middle of all observations[1]

```
median(pop$Total, na.rm = T) # or
quantile(pop$Total, 0.5, na.rm = T)
```

- Note: `na.rm = T` will first remove all missing observations
- The median is also known as the 50th percentile
- Quantiles: cut points in theoretical and empirical distributions

```
quantile(pop$Total, probs=c(0, 0.25, 0.5, 0.75, 1.0),
  na.rm = T)
```

- Interquartile range:

```
IQR(pop$Total, na.rm = T)
```

- Check out also

```
table(pop$Total)
```

[1]What if they are an even number?

Mode of data

- The mode shows the value which has the highest frequency
- There is no built-in function in base R
- But we can write one ourselves

```
Mode <- function(x) {
  temp <- table(as.vector(x))
  as.numeric(names(temp)[temp == max(temp)])
}
```

- Then we can use it on our variables, e. g.

```
  Mode(pop$Male)
```

- Nice try but not really a solution. Why?
- Check out the **modeest** package

Mean, variance, standard deviation

- Sample mean:

$$\overline{x} = \frac{1}{n}\sum_{i}^{n} x_i$$

- This is obviously the simple arithmetic average of all data points of a variable
- Example: the mean population

```
mean(pop$Total, na.rm = T)
```

Mean, variance, standard deviation (2)

- Sample variance:

$$s_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

- Example:

```
var(pop$Total)
```

- Standard deviation: square root of the variance

$$s_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2}$$

```
sd(pop$Total)
```

- Check out also:

```
summary(pop) # Summary stats
str(pop) # Structure of frame
```

Measures of association

Covariance and correlation

- Covariance:

$$\text{Cov}(x,y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})$$

```
cov(pop$Male, pop$Female)
```

- Simple (Pearson) correlation:

$$\text{Corr}(x,y) = \frac{\text{Cov}(x,y)}{s_x s_y}$$

```
cor(pop$Male, pop$Female)
```

Two more correlation coefficients

- The default correlation is the Pearson one, and it works well only with normally distributed variables
- The Spearman rank correlation coefficient (Spearman's ρ) is a non-parametric (therefore distribution-insensitive) measure of statistical dependence between **two** variables
- The values of the two variables are ranked and then the differences of ranks are computed
- The formula for ρ is:

$$\rho = 1 - 6\frac{\sum_i d_i^2}{n(n^2-1)}$$

- where d_i are the differences in ranks, or, equivalently:

$$\rho = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum t_i)^2}}$$

Two more correlation coefficients (2)

- Example in R:

```
pearson <- read.xlsx("pearson.xlsx", sheet = "Sheet1")
attach(pearson)
cor(var1, var2, method = "pearson")
cor(var1, var2, method = "spearman")
```

Two more correlation coefficients (3)

- Kendall's τ: a non-parametric measure of dependence, too
- It is a rank correlation coefficient, like Spearman's
- Pairs of observations x_i, y_i and x_j, y_j are compared
- If it is true that either both $x_i > x_j$ and $y_i > y_j$, or both $x_i < x_j$ and $y_i < y_j$, then the observations are concordant; otherwise they are discordant
- The formula is:

$$\tau = \frac{2(n_c - n_d)}{n(n-1)}$$

- In R:

```
    cor(var1, var2, method = "kendall")
```

Data and function plotting

Histograms

- Before that we load some more data:

```
infl <- read.csv("cpi_infl_avg.csv", header = TRUE,
    stringsAsFactors = FALSE)
```

- To plot the histogram using base R:

```
hist(infl$value, main="Distribution of Inflation Rates", xlab="
    Values", breaks = 120)
```

- There are some outliers which skew the distribution enormously
- We can exclude those value by limiting the value range

```
hist(infl$value, main="Distribution of Inflation Rates", xlab="
    Values", breaks = 1200, xlim = c(-50,50))
```

Scatter plots

- Scatter plots are diagrams to display the values of two variables in the plane
- Let's import some GDP data:

```
gdp <- read.csv2("gdp.csv")
```

- To create a scatter plot of GDP and consumption:

```
plot(gdp$GDP, gdp$Cons)
```

Line graphs

- To plot e. g. exports against time:

```
plot(gdp$Year, gdp$Exports, type = "l")
```

- We may add for example points to this chart:

```
points(gdp$Year, gdp$Exports)
```

- To add to the same chart e. g. Imports using a different color:

```
lines(gdp$Year, gdp$Imports, col = "red")
```

- ...and add points to Imports:

```
points(gdp$Year, gdp$Imports, col = "red")
```

Bar graphs and box plots

- Bar graphs:

```
barplot(gdp$GDP,names.arg=gdp$Year, col = "blue")
```

- Box plots – we will not deal with them for now, just try them:

```
boxplot(gdp$Cons, gdp$GDP)
```

Plotting function curves

- One option is to do the following:

```
curve(x^2 + log(x))
```

- Another option is to first define the function and then plot it:

```
ownf <- function(xvar) {
    1/(1 + exp(-5*xvar + 6))
}
curve(ownf,from=-10,to=10)
```

Better quality of graphs: ggplot2

- Install the package and load it:

```
library(ggplot2)
```

- Check out the package website: `https://ggplot2.tidyverse.org/`
- To repeat the scatter plot, use either of the two:

```
qplot(gdp$GDP,gdp$Cons)
qplot(GDP,Cons, data=gdp)
```

- To save the plot to an external file (e. g. `jpeg, png, ps, pdf`), etc.:

```
dev.copy(png, "scatter.png")
dev.off()
```

- Pay attention to the last line: if you do not run it, your plot will be (partially) unavailable!

Better quality of graphs: ggplot2 (2)

- Let’s first see what geoms and aesthetics mean: https://beanumber.github.io/sds192/lab-ggplot2.html
- To repeat line graphs:

```
qplot(Year, Cons, data=gdp, geom = c("line","point"))
```

This is equivalent to (using the `aesthetics` option of `ggplot2`):

```
ggplot(gdp, aes(x=Year,y=GDP)) + geom_line() + geom_point()
```

- To beautify:

```
ggplot(gdp, aes(x=Year,y=GDP)) + geom_line(colour = "red",size
    =1.5) + geom_point(colour="red",size=4)
```

Better quality of graphs: ggplot2 (3)

- To repeat bar graphs:

```
ggplot(gdp, aes(x=Year,y=GDP)) + geom_bar(stat="identity")
```

- If you don’t like the black colour, you can change it:

```
ggplot(gdp, aes(x=Year,y=GDP)) + geom_bar(stat="identity", fill="
    orange", colour = "blue")
```

- Multiple lines:

```
ggplot(gdp, aes(Year)) + geom_line(aes(y=GDP),colour = "red",size
    =1.5) + geom_line(aes(y=Cons),colour = "green",size=1.5)
```

- Adding a legend is a bit trickier – you need to reshape the data and identify it by a factor (category):

```
library(reshape2)
meltgdp <- melt(gdp, id="Year")
ggplot(meltgdp, aes(x=Year, y= value, color = variable, group=
    variable)) + geom_line(size=1.5)
```

Better quality of graphs: ggplot2 (4)

- To create better-looking function plots:

```
func1 <- function(x){
    sin(x) + cos(x)
  }
plot1 <- ggplot(data.frame(x=c(0, 20)), aes(x=x)) + stat_function
    (fun=func1, geom="line", size=1, color = "red" )
```

- Define a second function:

```
func2 <- function(x) {
    2*sin(x)
  }
```

- Add it to the plot:

```
plot2 <- plot1 + stat_function(fun=func2,size=1,color="blue")
```

3D Graphs

- **ggplot2** does not directly support 3d surfaces yet
- There are, however, other ways (basically two) to get 3d graphs
- One of the ways is to use the 'core' plotting capabilities of R
- The other is to use other packages such as the **lattice** package or the **scatterplot3d** package
- Note: there are some narrowly specialized packages for plotting very specific data: **Rfacebook**, **xts**, **zoo**, **maps**, **animations**, etc.

3D Graphs (2)

- To use the core functionality – see the following example

```
dome <- function(x,y){
  -(x^2 + y^2)
  }
x <- seq(from=-3, to = 3, by=0.01)
y <- seq(from=-3, to = 3, by=0.01)
z <- outer(x,y,dome)
persp(x,y,z,col="blue",theta=70,phi=-10)
```

- New keywords:
  `outer`: applies a function to two arrays (technically this is an outer product – see e.g. Wikipedia on this)
  `persp()`: this is the plotting function
- Check out the **plot3D** package which builds upon `persp()`

3D Graphs (3)

- The **lattice** package is good but rarely used by economists so we look only at demos:

```
library(lattice)
demo(lattice)
```

- 3d scatterplots using `scatterplot3d`:

```
require(scatterplot3d)
attach(gdp)
scatterplot3d(Year,Cons,GDP)
```

- etc.