R lab 1

## Data downloads

Data used in this Rlab is the textfile **popdata.txt**

## Getting started

Install R, install R Studio. [link](https://www.rstudio.com) [link](https://www.r-project.org)

Create a folder named “RSust” on your computer where you save everything within this R course. Create a subfolder named “Data” where you may save all the textfiles with downloaded data. Open R studio and create a new R project. Save it in your RSust-folder. Create a new R script for this Rlab and save it in your RSust-folder with an appropriate name, eg. “Rlab1”

Follow the instructions in the Rlab documents. All grey area is code that you should copy from here and paste into your own R script.

All lines that start with ## means that it is my printed output within the grey area. You should get the same output in your console.

# To execute a command:

press ctrl+r on the line of interest,

or mark the commands with the cursor and then press ctrl+r

To use an argument on a df: argument(df)

To read about an argument use: ?argument

# Bookmarks in R studio ——–

To insert bookmark: ctrl+shift+r

Go to a bookmark: shift+alt+j

# Find and set your working directory

Check your working directory

getwd()

Your R working space could be redirected either from the Session icon or by using the argument setwd()

# Import data

During this course you will only import data from textfiles, so the following code should be useful at all times. However, there is numerous ways to import data from other data sources e.g. other types of textfiles or from excel sheets. If you use another method, make sure that your import is correct!

If you have downloaded the textfiles to a folder within your working directory the short version will work. Otherwise you need to redirect the reading of the table e.g. “C:/Users/emmaholm/Documents/Folderexample/popdatatxt”,…

The argument read.table()

pop<-read.table("Data/popdata.txt",  
 header=T, sep="\t", na.strings="NA",dec=".", strip.white=T)

will read the textfile and create a dataframe, in addition it will

read the first line as headers of the columns (header=T)

the separation of the data is done by tabs “(sep= )

if there is any missing data it will get NA in the dataframe (na.settings=T)

if there is a . it will be read as a decimal(dec=“.”)

if there is unnecessary white spece in columns it will be removed(strip.white=T)

# Export data

Use this if you need to export a dataframe to a text file. Might be handy later on.

#write.table(pop,   
# file = "dataexport.txt",  
# append = FALSE, quote = TRUE, sep = "\t",  
# eol = "\n", na = "NA", dec = ".", row.names = F,  
# col.names = TRUE, qmethod = c("escape", "double"))

# Explore data

If the import worked as it should, you will now see a dataframe called pop listed in the frame in the right upper corner. It says automatically that pop have 189 observations (rows) and 6 variables(columns). Open the data frame by click on it with your mouse.

Or within your R script

pop

However, when working with large data, opening the full datasets works poorly for getting an overview. You might use some other arguments to get a quick overview.

head() or tail() give you the 6 first and last rows

head(pop)  
tail(pop)

Use the following arguments to explore the data, what information do you get?

names(pop)  
str(pop)  
summary(pop)

Other ways to look at specific parts of the data, or use it in functions is to adress only specific rows or columns in the dataframe. The structure is always df[row,column]

Read the first row and first column

pop[1,1]

Read the first 4 rows and column 3 and 4 See my output on the rows with ##, you should get the same

pop[1:4,c(3,4)]

The same but use column names instead of numbers

pop[1:4,c("height","dia")]

To use only one of the columns use df$column

pop$dia

In that way you could use arguments both on the full dataframe or only on one of the variables

summary(pop)  
summary(pop$dia)

Check the factorial levels in the factor clone According to str(pop) you have a factor called clones. Check how many unique clones there is in the data. You also have a factor called fert wich is short for fertilization, check how many levels.

unique(pop$clone)  
unique(pop$fert)

And if you want to know how many seedlings you have in each block which is fertilized or not, you may see them in a table with both factors, where the first factor is displayd in the row and the second as in columns.

table(pop$block,pop$fert)

# Add columns and ifelse statement

As you see when using str(pop$fert), the fertilizing column is numerical and not a factor. In order to change this we could by adding a new column create a new factor where all fertilized seedlings (fert=1) will be named and assigned the value “fert” and the unfertilized will be named “control”.

Ifelse is a logical argument so we give the arguments for testing, what it should be called if true and what it should be named if false.

ifelse(test,True,False) *To read more about ifelse argument type ?ifelse.*

pop$ffert<-ifelse(pop$fert==1,"fert","control")  
str(pop)

As you see in the structure,the new variable is just characters, we prefer to define it as a factor, so we will add a new argument that the created column must be a factor. Thereafter, take a look at the two variables fert and ffert, which should be the same only that one is numerical and the other one factorial.

pop$ffert<-as.factor(ifelse(pop$fert==1,"fert","control"))  
str(pop)  
table(pop$ffert,pop$fert)

## Q1 Rassignment 1

Question 1-3 should be answered in a word file where you add the answers to Q1, Q2 and add the figure in Q3. Save the wordfile as Rassignment1\_firstname\_lastname

Upload on the webportal.

1a. How many of fertilized and unfertilized (control) seedlings respectively, in the data?

1b. How many different blocks and clones?

# Mean and range of the data, with() statement

Calling for a single variable could be used for overview but also for calculations

mean(pop$height)  
median(pop$height)  
min(pop$height)  
max(pop$height)

Display range

max(pop$height)

min(pop$height)  
#should give you the same as:  
range(pop$height)

Calculating the mean value

sum(pop$height)/nrow(pop)  
#should give you the same as:  
mean(pop$height)

One of the most useful arguments is with(). Instead of defining the dataframe everytime, we could simplify the code by starting to defining the dataframe that will be used for the whole argument. E.g. when calculating a HD (height-diameter ratio) :

pop$hd1<-pop$height/pop$dia  
mean(pop$hd1)  
#will be shorter written like:  
pop$hd2<-with(pop,height/dia)  
mean(pop$hd2)

## Q2

2a. What is the minimum diameter?

2b. What is the maximum height?

2c. What is the mean hd-ratio?

# Subset data

Sometimes you want to use only parts of the data based on a value or based on a category. To create a new dataframe you could use subset(df,argument). Make one dataframe with only block 1. You should see the new dataframe in the global environment on your upper right.

pop.block1<-subset(pop,block==1)

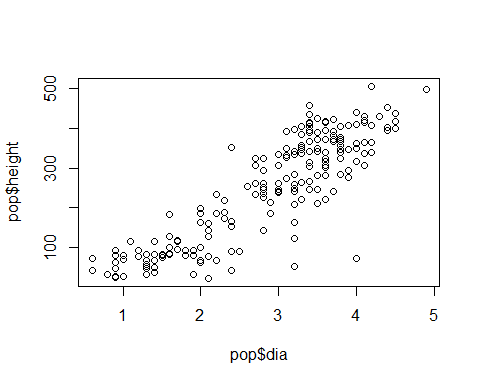
Another way is to use the [ ] brackets and identify what you want to keep. Make a subset of block 2.

pop.block2<-pop[pop$block==2,]

## Plot data in nice figures

There is many ways to plot figures in R. Today we will use the basic plot function plot(x,y). The figure should be plotted in the window lower right. There you could choose to zoom in or to export/copy to a jpeg or pdf if you want to.

plot(pop$dia,pop$height)



To simplify the code use the with() argument. with(df,plot(x,y)).

Now we also want to add some extra details in the plot, e.g. labels of the axis, or define the range of the axis. Also, change number for the color or size of the point as examplified below.

with(pop,plot(dia,height))

#define a title  
with(pop,plot(dia,height, main="Simple plot"))

#define labels  
with(pop,plot(dia,height, xlab="Diameter (mm)", ylab="Height(mm)"))

#define limits on y and x axis  
with(pop,plot(dia,height,xlim=c(0,7),ylim=c(0,550)))

#change apperance of color, point type and size   
with(pop,plot(dia,height,col="red"))   
with(pop,plot(dia,height,col=2))   
with(pop,plot(dia,height,pch=8))   
with(pop,plot(dia,height,cex=2))   
#and combine several of them  
with(pop,plot(dia,height,col="gold",pch=16,cex=2))

# Plot on plot

We could use the numerical variables for symbolizing different colors in the points.

with(pop,plot(dia,height,col=pop$ffert))

But often could it be more convenient to subset the data and then add layers of plots for each subset of the data. plot(x,y) The first argument creates the plot with the first data points(x,y) The second argument just adds more points.

range(pop$dia)  
range(pop$height)

#First layer of the figure displays fertilized seedlings using plot()  
with(pop[pop$ffert=="fert",],plot(dia,height,col="green",xlim=c(0,5),ylim=c(0,600)))

#Second layer added to the figure shows control, using points()  
with(pop[pop$ffert=="control",],points(dia,height,col="purple"))

# Multipanel plots

To visualize the data in several panels instead of on top each other: Change “par” argument of the plot window and change number of rows and columns with figures. Does the figure look strange in the preview window? Press on the “zoom” icon in the plot window.

par (mfrow=c(3,1)) #3 row, 1 column  
with(pop[pop$clone=="A",],plot(dia,height)) #Only clone A  
with(pop[pop$clone=="B",],plot(dia,height,col="red"))  
with(pop[pop$clone=="C",],plot(dia,height,col="purple"))  
par (mfrow=c(1,1)) #1 row, 1 column, change back to normal

Q3. Make a figure of height vs diameter for the pop-data.

Make sure you have appropriate labels for x an y axis

Choose color and point types and symbolize attributes after your own choice

Choose appropriate limits for the x and y axis

Type your name in the title inside the figure (main= )

Click on the Export icon in the Plots window and save the figure as a jpeg or paste it directly in your word file. Upload the R assignment on the portal in the assignment.

