Brief „R/ R-Studio“ introduction

09.09.2019 – Katharina Kurz

1. General information:

- R distinguishes between use of capital and small initial letters

- every “;” – “,” “_” “()” etc. is important – this are the most prominent sources of errors

- R doesn’t care about spaces

you can write comments in a skript with “#” – with this you tell R not to process this line/command but it can be very helpful for you

> shows you a command/line you can run in “R”. If you copy the command from the skript don’t copy it or you will get an error message

2. First steps

- open a new R-Skript

A Skript is basically the recipe or the input you give “R” that it knows what it should do – like a cooking recipe

- save it as “*R-Introduction*”

- R consists out of one basic program and thousands of *packages.* A package needs to be installed once.

3. How to install packages

```
> install.packages()
    # “library()” shows you all already installed packages
```

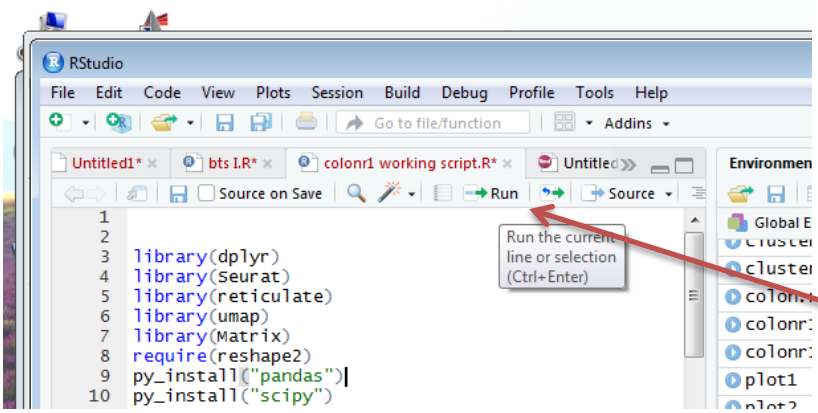
4. Load packages

```
    # Packages has to be load after every re-start of the
program

> Library(nameofpackage)

    # name of package you want to load comes in parenthesis
```

5. How to run a command in a line or a section



Click „run“ to only run the selected line or select all and “run”

Shortcut: Ctrl + Enter

6. What to see in the console

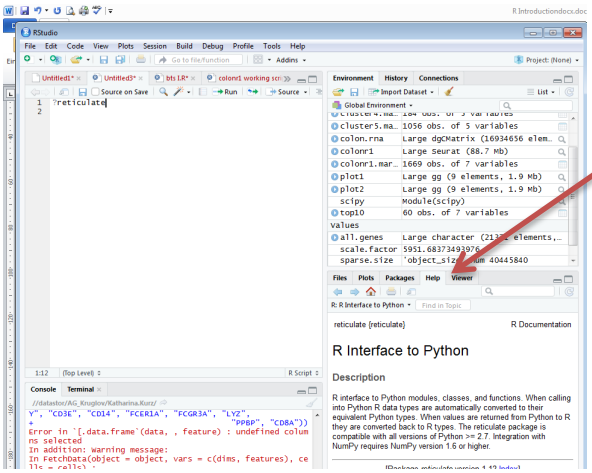
#Basically: Warnings and your output/results

```
> library(Seurat)
Warning message:                                    #warning
Paket ‘Seurat’ wurde unter R Version 3.5.3 erstellt


#warning message is very helpful because most of the time it
directs you to the error in your skript

> colonr1                                           #blue: command
An object of class Seurat                           #output/results
21371 features across 664 samples within 1 assay
Active assay: RNA (21371 features)
```

7. Help



?packagename + run

- Output in help window

8. Let's get started

```
| In its simplest form, R can be used as an interactive
| calculator. Type 5 + 7 and press Enter.

> 5+7
[1] 12


| To assign the result of 5 + 7 to a new variable
| called x, you type x <- 5 + 7. This can be read as 'x
| gets 5 plus 7'. Give it a try now.

> x <- 5 + 7

#try now typing x as your variable

> x

[1] 12

# now let's create a vector:

| The easiest way to create a vector is with the c()
| function, which stands for 'concatenate' or
| 'combine'. To create a vector containing the numbers
| 1.1, 9, and 3.14, type c(1.1, 9, 3.14). Try it now
| and store the result in a variable called z.

> z <- c(1.1, 9, 3.14)

| You can combine vectors to make a new vector. Create
| a new vector that contains z, 555, then z again in
| that order. Don't assign this vector to a new
| variable, so that we can just see the result
| immediately.

> c(z, 555, z)
[1]   1.10   9.00   3.14 555.00   1.10   9.00   3.14
```

9. Workspace & Files

```
| Determine which directory your R session is using as
| its current working directory using getwd().

> getwd()
[1] "\\\\datastor/AG_Kruglov/Katharina.Kurz"


Use dir.create() to create a directory in the current
| working directory called "testdir".

> dir.create("testdir")


Set your working directory to "testdir" with the
| setwd() command.

> setwd("testdir")

| Create a file in your working directory called "mytest.R"
| using the file.create() function.

> file.create("mytest.R")
[1] TRUE

| This should be the only file in this newly created
| directory. Let's check this by listing all the files in
| the current directory.

> dir()
 [1] "mytest.R"

| Change the name of the file "mytest.R" to "mytest2.R" by
| using file.rename().

> file.rename("mytest.R", "mytest2.R")
[1] TRUE
```

10. Vectors & tables

```
| Let's create a vector containing the numbers 1 through 20 using the `:` o
perator. Store
| the result in a variable called my_vector.

> my_vector <- 1:20; my_vector

Creating a table:

| Bring up the help file for the matrix() function now using the `?` functi
on.

> ?matrix

| Now, look at the documentation for the matrix function and see if you can
figure out how to create a matrix containing the same numbers (1-20) and di
mensions (4 rows, 5 columns)
| by calling the matrix() function. Store the result in a variable called m
y_matrix2.
```

```
> my_matrix2 <- matrix(data = 1:20, nrow=4, ncol=5)
```

NOW ITS GETTING HOT:

```
| Now, imagine that the numbers in our table represent some measurements fr
om a clinical experiment, where each row represents one patient and each co
lumn represents one variable for which measurements were taken.

| We may want to label the rows, so that we know which numbers belong to ea
ch patient in the experiment. One way to do this is to add a column to the
matrix, which contains the names of all four people.

| Let's start by creating a character vector containing the names of our pa
tients -- Bill, Gina, Kelly, and Sean. Remember that double quotes tell R t
hat something is a character string. Store the result in a variable called
patients.

> patients <- c("Bill", "Gina", "Kelly", "Sean")


| Now we'll use the cbind() function to 'combine columns'. Don't worry abou
t storing
| the result in a new variable. Just call cbind() with two arguments -- the
patients vector and my_matrix.

> cbind(patients, my_matrix)
     patients
[1,] "Bill"   "1" "5" "9"  "13" "17"
[2,] "Gina"   "2" "6" "10" "14" "18"
[3,] "Kelly"  "3" "7" "11" "15" "19"
[4,] "Sean"   "4" "8" "12" "16" "20"

| Something is fishy about our result! It appears that combining the charac
ter vector with our matrix of numbers caused everything to be enclosed in
double quotes. This means we're left with a matrix of character strings, wh
ich is no good.

So, we're still left with the question of how to include the names of our p
atients in the table without destroying the integrity of our numeric data.
Try the following -- my_data <- data.frame(patients, my_matrix)

my_data <- data.frame(patients, my_matrix)

#now view table
>my_data

| Since we have six columns (including patient names), we'll need to first
create a vector containing one element for each column. Create a character
vector called cnames that contains the following values (in order) -- "pati
ent", "age", "weight", "bp", "rating", "test".

> cnames <- c("patient", "age", "weight", "bp", "rating", "test")
#add the colum names to the data frame:
> colnames(my_data) <- cnames
> My_data
```

Other way to design a table:

```
# Create objects:
age = c(20,22,24)
name = c("A", "B", "c")
# create table
```

```
table = data.frame(age, name)
#view table
View(table)
```

11.a HELP with arguments

```
> args(list.files)
function (path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE
,
    recursive = FALSE, ignore.case = FALSE, include.dirs = FALSE,
    no.. = FALSE)
NULL
```

12.Basic Graphics

```
| Load the included data frame cars with data(cars).

> data(cars)
| Run head() on the cars data.

#To plot the data:

> plot(x = cars$speed, y = cars$dist)

Use plot() command to show dist on the x-axis and speed on the y-axis from
the cars data frame.
> plot(x = cars$dist, y = cars$speed)



Recreate the plot with the label of the x-axis set to "Speed".
> plot(x = cars$speed, y = cars$dist, xlab = "Speed")

| Recreate the plot with the label of the y-axis set to "Stopping Distance"
.

> plot(x = cars$speed, y = cars$dist, xlab = "Speed", ylab="Stopping Distan
ce")



# add title:

Plot cars with a main title of "My Plot". Note that the argument for the
main title is "main" not "title"

> plot(cars, main = "My Plot")


| Plot cars with a sub title of "My Plot Subtitle".

> plot(cars, sub = "My Plot Subtitle")

| Plot cars so that the plotted points are colored red. (Use col = 2 to ach
ieve this effect.)

> plot(cars, col=2)
| Plot cars while limiting the x-axis to 10 through 15.  (Use xlim = c(10,1
5) to achieve this effect.)
Use xlim = c(10, 15))
```

```
> plot(cars, xlim = c(10, 15))

#from here work with new data set “mtcars”

 #load &viewn “mtcars”

data(mtcars)
mtcars
#see helpfunction of boxplot

?boxplot

#generate boxplot of mtcars 1st and second column mpg and cycl with “foruml
a = mpg ~ cyl

> boxplot(formula = mpg ~ cyl, data = mtcars)

#Histogram of “mpg” of mtcars for data exploration
> hist(mtcars$mpg)

#generate heatmap:
#Load “gplots”

library(gplots)

#generate matrix out of mtcars

mymatrix5 <- (data.matrix(mtcars))

#generate heatmap with a colour code and without a trace

heatmap.2(mymatrix5, key=TRUE, trace = "none")
```

<3 YAY YOU MADE IT YOU AWESOME NERD