R Programming Course

Bernhard Koschicek

Lesson 1

Basic Building Blocks

Learned Code:

```
variable <- stuff
1:6
c(1, 2, 3, 4)
```

1. Asign a variable
2. Represents all numbers between 1 to
3. Create a vector

Intereseting Stuff:

Well I learned basic numeric operators and the strange variable definition with <-, which is an abnmination. The cat function is cool and easy to use and I love the auto completion for variables. Additionally we learnd how to calculate vectors with each other, which is quite easy. The info about the auto completion at the end told me, that I need only two chars, but I need three instead, but with tab everything works fine.

Questions:

Lesson 2

Workspace and Files

So R uses also basic cmd line commands. It is quite interesting, that RStudio saves a history of the commands and also the variables in .RData and .Rhistory.

> This is the R analog to "Take only pictures, leave only footprints."

Learned Code:

```
getwd() - setwd()
ls()
args(function)
dir.create()
file.rename() remove() create() path() list()
```

1. Get or set working dir

2. Lists objects in WD
3. Shows all arguments of function
4. creates dir
5. file manipulation

Intereseting Stuff:

What is interesting is that during the file creation I got an permission denied error, but it created nonetheless the file. ~ In file.create("mytest.R") : cannot create file 'mytest.R', reason 'Permission denied' ~

Ok file manipulation is funny, I created an folder and not the R file they wanted, but I could change the name with

```
file.rename()
```

So a folder is for R also a file, ok, but the I used the

```
file.copy()
```

function and it made a R.file out of the folder. Something is fishy here.

Questions:

At the path creation I wonder if it also takes the *mode="0777"* flag on Windows, because it usually has another user control system then unix systems.

Lesson 3

Sequences of Numbers

How to create sequences of numbers in R.

Learned Code:

```
length(my_seq)
seq(along.with = my_seq)
seq_along(my_seq)
rep(0, times = 40); rep(c(0, 1, 2), times = 10)
rep(c(0, 1, 2), each = 10)
rep()
```

1. Length of object
2. Length of object
3. Length of object
4. Mupltiply n times x
5. Multiply x,y,z n times
6. Kind of a loop

Intereseting Stuff:

Questions:

Lesson 4

Vectors

Vectors come in two different flavors: atomic vectors and lists. An atomic vector contains exactly one data type, whereas a list may contain multiple data types ### Learned Code:

```
paste(my_char, collapse = " ")
```

1. Is not the primarly print function. It is only for vectors and >> Concatenate vectors after converting to character. >> -[see https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/paste]

Intereseting Stuff:

- As in the first lesson, conditions for vectors are used for every variable. T/F condition is also used on all. We learn the basiv logical Operators.
- We can Join to vectors easily with paste, but all integers (are there floats or something else?) get chars.

Lesson 5

Missing Values

Missing values play an important role in statistics and data analysis. Often, missing values must not be ignored, but rather they should be carefully studied to see if there's an underlying pattern or cause for their missingness. In R, NA is used to represent any value that is 'not available' or 'missing' (in the statistical sense). Any operation involving NA generally yields NA as the result.

Learned Code:

```
rnorm(1000)
rep(NA, 1000)
sample(c(y, z), 100)
is.na()
sum()
```

1. Normal distribution n times (Statistic)
2. just repeat X n times
3. take n random examples
4. whether each element of a vector is NA
5. prints sum of values in variable;

Intereseting Stuff:

- *NA* is just a placeholder not a value!

Lesson 6

Subsetting vectors

In this lesson, we'll see how to extract elements from a vector based on some conditions that we specify.

Learned Code:

```
x[n:m]
x[c(-n, -m)] | x[-c(n, m)]
vect <- c(foo = 11, bar = 2, norf = NA)
names()
identical()
vect["name", "name2"]
```

1. shows items of vector x
2. Shows not item n and m, strangely it is done with - and not !
3. Asign values to names
4. gives names of vector
5. compare names of vectors (T/F)
6. gives the value of the elementname

Intereseting Stuff:

- Again ***NA* is just a placeholder not a value!** so >0 also includes *NA*'s.
- R uses 'one-based indexing'!!!!!!!!!!!!
- If we want to know a item of a vector and the itemnumber is out of range, R doesn't give any useful error, just an NA output.

Questions:

Are vecotrs really orderd?

Lesson 7

In this lesson, we'll cover matrices and data frames. Both represent 'rectangular' data types, meaning that they are used to store tabular data, with rows and columns. The main difference, as you'll see, is that matrices can only contain a single class of data, while data frames can consist of many different classes of data.

Matrices and Data Frames

Learned Code:

```
dim(var)
dim(my_vector) <- c(4, 5)
attributes(var)
class(var)
matrix() [matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE, dimnames = NULL)]
cbind(patients, my_matrix)
data.frame(patients, my_matrix)
colnames(my_data) <- cnames
```

1. dimension of an object (not working on vector, instead use lenght())
2. assigneds a dimension for a vector -> it is now a matrix!
3. same as dimension (a least the tutorial said it so)
4. shows class of the variable
5. create Matrix
6. "concat" vector as column, and changes the type to char if the vector contains chars, because a matrix can only contain one datatype
7. combines vector and matrix, and change the values to an object
8. Set colname vor my_data which is a dataframe

Intereseting Stuff:

> to illustrate the point that a matrix is simply an atomic vector with a dimension attribute

Questions:

Lesson 8

Logic

There are two logical values in R, also called boolean values. They are TRUE and FALSE. In R you can construct logical expressions which will evaluate to either TRUE or FALSE.

Learned Code:

```
&&
||
isTRUE()
xor()
which()
```

1. Only first value of the vector is compares
2. also only first member
3. just strate if something is true
4. classical xor function
5. find indecies of argument

Intereseting Stuff:

Basic Logic operators, nothing special

Questions:

Lesson 9

Functions

Learned Code:

```
mean()
function_name <- function(arg1, arg2){}
args(remainder)
```

1. Mean of vector
2. Basic function
3. call arg input

Intereseting Stuff:

1. Everything that exists is an object. 2. Everything that happens is a function call

- No return, just the last expression is the output
- Partial mapping of arguments, so we can either go with the sequence or call directly the argument.
- `...` which is referred to as an ellipsis or simply dot-dot-dot. The ellipsis allows an indefinite number of arguments to be passed into a function
- all arguments after an ellipses must have default values

Questions:

Lesson 10

lapply and sapply

These powerful functions, along with their close relatives (vapply() and tapply(), among others) offer a concise and convenient means of implementing the Split-Apply-Combine strategy for data analysis.

Learned Code:

```
as.character(cls_list)
sapply()
```

1. displays a list as a vector?!
2. Simple list

Intereseting Stuff:

> In general, if the result is a list where every element is of length one, then sapply() returns a vector. If the result is a list where every element is a vector of the same length (> 1), sapply() returns a matrix. If sapply() can't figure things out, then it just returns a list, no different from what lapply() would give you.

- to simplifying it, apply is just a loop through a matrix/dataform

Questions:

lapply are forms and sapply chars?

Lesson 11

vapply and tapply

In this lesson, you'll learn how to use vapply() and tapply(), each of which serves a very specific purpose within the Split-Apply-Combine methodology.

Learned Code:

```
vapply(flags, class, character(1))
tapply()
table(x$y) -> tapply(flags$animate, flags$landmass, mean)
```

1. The 'character(1)' argument tells R that we expect the class function to return a character vector of length 1 when applied to EACH column of the flags dataset
2. splits dataset into groups
3. uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. So e.g. how many flags have animate images.

Intereseting Stuff:

- vapply() as a safer alternative to sapply()
- vapply() is fastern than sapply() for large datasets. sapply() better at the prompt.

Questions:

Lesson 12

Looking at Data

Whenever you're working with a new dataset, the first thing you should do is look at it! What is the format of the data? What are the dimensions? What are the variable names? How are the variables stored? Are there missing data? Are there any flaws in the data? This lesson will teach you how to answer these questions and more using R's built-in functions.

Learned Code:

```
nrow(table)
ncol(table)
object.size(object)
names(table)
head(table, x)
tail(table, x)
summary(table)
str(object)
```

1. Only number of rows from the table
2. Only number of columns from the table
3. How much space an object occupies.
4. Give the variable column.
5. Preview the top of the dataset (first six or x = how many rows)
6. Same as head, obviously
7. shows a summary of each column, very handy.
8. is *NOT* a string! It is the structure from the object. Combines features from the other functions.

Intereseting Stuff:

Questions:

- Is there a better human readable version of object.size()

Lesson 13

Simulation

This lesson assumes familiarity with a few common probability distributions, but these topics will only be discussed with respect to random number generation.

Learned Code:

```
sample(range, size, replace T/F)
LETTERS
sample(c(0,1), 100, replace = TRUE, prob = c(0.3, 0.7))
rbinom()
rnorm()
rpois()
replicate(n, function)
colMeans()
hist(object)
```

1. sample takes a sample of the specified size from the elements of x using either with or without replacement. Replacement -> TRUE, same numbers can appear; FLASE only unique numbers

2. English alphabet
3. prob asigns the probability
4. Each probability distribution in R has an r*** function (for “random”), a d*** function (for “density”), a p*** (for “probability”), and q*** (for “quantile”). A binomial random variable represents the number of ‘successes’ (heads) in a given number of independent ‘trials’ (coin flips). Therefore, we can generate a single random variable that represents the number of heads in 100 flips of our unfair coin using rbinom(1, size = 100, prob = 0.7). 5.The standard normal distribution has mean 0 and standard deviation 1. As you can see under the ‘Usage’ section in the documentation, the default values for the ‘mean’ and ‘sd’ arguments to rnorm() are 0 and 1, respectively. Thus, rnorm(10) will generate 10 random numbers from a standard normal distribution
5. Poisson distribuition, whatever this is.
6. Loop n times
7. Mean of the columns?!
8. histogram of object

Intereseting Stuff:

Questions:

Lesson 14

Dates and Times

R has a special way of representing dates and times, which can be helpful if you’re working with data that show how something changes over time (i.e. time-series data) or if your data contain some other temporal information, like dates of birth.

Learned Code:

```
Sys.Date()
unclass()
weekdays(), months(), quarters()
strptime()
difftime(Sys.time(), t1, units = 'days')
```

1. obviously sys date
2. how object looks like internally
3. extrat wanted things
4. strptime() converts character vectors to POSIXlt
5. Time differents in days

Intereseting Stuff:

- basic operators work on date

Questions:

Lesson 15

Base Graphics

One of the greatest strengths of R, relative to other programming languages, is the ease with which we can create publication-quality graphics. In this lesson, you'll learn about base graphics in R.

Learned Code:

```
plot(object) (plot(x = cars$speed, y = cars$dist))
... xlab ="Speed", ylab="Stopping Distance"
... main = "My Plot"
... sub = "My Plot Subtitle"
... col = 2
... xlim = c(10, 15)
... pch = 2
boxplot(formula, data = NULL)
hist(object$columnname)
```

1. Easy scatterplot diagram
2. Argument to name the axis
3. Argument to name the scatterplot
4. Argument to name subtitle
5. Argument to colour of points in red
6. Argument to limit x axis between 10 and 15
7. ARgument to change point shape
8. Create a boxplot diagram, note formular is befor data.
9. Histogram

Intereseting Stuff:

- http://varianceexplained.org/r/teach_ggplot2_to_beginners/
- Things we learned hear could be applied for many other diagramm varients.

Questions: