Worksheet *Short Introduction to R*

Sven Garbade
Centre for Pediatric and Adolescent Medicine
Division for Neuropediatrics and Metabolic Medicine
University Hospital of Heidelberg
sven.garbade@med.uni-heidelberg.de

Spring 2020

Instructions

1. Solve the problems with your colleagues.
2. Use all sources of help (slides, internet, etc.) you need to solve the problems.

1 Installation

Problem 1

Install R on your computer. See `https://cran.r-project.org` on how to install R on your machine.

Problem 2

Install a good editor or an IDE to use R.

Suggestions:

- Editor: `https://notepad-plus-plus.org/`
- RStudio: `https://rstudio.com/`
- Emacs with ESS: `https://www.gnu.org/software/emacs/`, `https://ess.r-project.org/`
- There are many more, e.g. blog post

Tip: If unsure, choose RStudio. This seems to be a convenient IDE. Emacs and ESS is a more sophisticated solution for users with a strong background in Linux.

2 Get help

Problem 3

R has many ways to get help. One important way is to learn to use the internal help:

```
?ls
help(ls)
help.search("ls")
help(package="boot")
```

3 R as a calculator

Problem 4

Use R as a calculator:

```
5 + 5

## [1] 10

sqrt(100)

## [1] 10
```

Problem 5
Explore some mathematical functions and try them.

4 Working directory

Problem 6
Your working directory is the folder on your computer in which you are currently working. Explore

```
getwd()
?setwd
```

Problem 7
Use `save()`, `load()` and `save.image()`.

5 Scalars and vectors

Problem 8
Create two scalars a and b and do some calculations with these. Learn how to use `c()`. Use `<-` or `=` to assign values to variables.

Sample solution:

```
a <- 4
b <- 9
b+a

## [1] 13

c(a,b)

## [1] 4 9

sqrt(c(a,b))

## [1] 2 3
```

Problem 9
What are valid variable names in R? Check function `make.names()`.

Sample solution:

Begin with a character, not a number, avoid spaces and special signs.

```
a <- c("!a", "var1", "1var")
make.names(a)

## [1] "X.a"   "var1"  "X1var"
```

Problem 10

Build two sequences a and b with numbers 1 to 5 and 6 to 10. Use functions `seq()` and `:`

Sample solution:

```
a <- 1:5
b <- 6:10
## or
a <- seq(1,5)
b <- seq(6, 10)
```

Problem 11

Learn to use more arguments from `seq()` and build some vectors.

Sample solution:

```
seq(0, 20, by=2)

##  [1]  0  2  4  6  8 10 12 14 16 18 20

seq(0, 20, length=4)

## [1]  0.000000  6.666667 13.333333 20.000000
```

Problem 12

Use `rep()` to build vectors.

Sample solution:

```
rep(c("a", "b"), 2)

## [1] "a" "b" "a" "b"

rep(c("a", "b"), c(2, 3))

## [1] "a" "a" "b" "b" "b"
```

Problem 13

Generate the following sequence with `paste()`:

```
## [1] "Number: 1" "Number: 2" "Number: 3" "Number: 4" "Number: 5"
```

Sample solution:

```
paste("Number", 1:5, sep=": ")

## [1] "Number: 1" "Number: 2" "Number: 3" "Number: 4" "Number: 5"
```

6 Random numbers

Problem 14

R comes with a many functions to generate random numbers with different distributions. Check function `rnorm()` and its arguments.

Sample solution:

```
rnorm(10)

##  [1] -1.5044620  1.0633428 -0.6883000  0.5593195 -1.0490936  0.9524490
##  [7] -0.8408175  0.1897196 -0.5925845 -0.9495710

rnorm(10, mean=5, sd=1)

##  [1] 6.276181 4.814176 6.111247 4.462212 4.112047 3.712814 5.369493 4.097883
##  [9] 5.234840 5.856715
```

Problem 15

Try other random number functions.

Sample solution:

```
runif(10)

##  [1] 0.70069050 0.01041438 0.07617825 0.20962373 0.17465614 0.81643810
##  [7] 0.35726145 0.71963030 0.98528641 0.15405786

rchisq(10, df=3)

##  [1] 2.6302934 0.4935501 0.1163184 5.7927907 3.5493785 0.3932631 0.9105884
##  [8] 1.3150892 7.6533735 6.2831691
```

7 Matrix

Problem 16

Use `rbind()` and `cbind()` to cohere sequences a and b to a matrix.

Problem 17

Generate a matrix m of dimension 4 rows and 5 columns with random numbers. Use function `matrix()`.

Sample solution:

```r
m <- matrix(rnorm(20), nrow=4, ncol=5)
m

##              [,1]       [,2]      [,3]        [,4]       [,5]
## [1,] -1.555501865  0.1053054 -1.068104  0.58124134  0.2171429
## [2,] -0.669192742  0.8760066  1.341684 -0.33636965 -1.4043993
## [3,] -0.009041392  0.2168320  0.337415 -0.02566362  1.5110124
## [4,] -0.680032339 -0.8099667 -1.021301  0.70919410 -0.4958138
```

Problem 18

Round the values to 1 digit.

Sample solution:

```r
round(m, digits=1)

##      [,1] [,2] [,3] [,4] [,5]
## [1,] -1.6  0.1 -1.1  0.6  0.2
## [2,] -0.7  0.9  1.3 -0.3 -1.4
## [3,]  0.0  0.2  0.3  0.0  1.5
## [4,] -0.7 -0.8 -1.0  0.7 -0.5
```

8 Lists

Problem 19

What are lists in R? Hint: use function `list()` and read help pages.

Problem 20

Create a list l with a vector v and matrix m. Name the elements of the list „v“ and „m“

Sample solution:

```
l <- list("v"=1:10,
          "m"=m)
l

## $v
##  [1]  1  2  3  4  5  6  7  8  9 10
##
## $m
##              [,1]       [,2]      [,3]        [,4]       [,5]
## [1,] -1.555501865  0.1053054 -1.068104  0.58124134  0.2171429
## [2,] -0.669192742  0.8760066  1.341684 -0.33636965 -1.4043993
## [3,] -0.009041392  0.2168320  0.337415 -0.02566362  1.5110124
## [4,] -0.680032339 -0.8099667 -1.021301  0.70919410 -0.4958138
```

Problem 21

Make row and column names in m. Explore `dimnames()`, `rownames()` and `colnames()`.

Sample solution:

```r
dimnames(m) <- list(paste("row", 1:nrow(m), sep=":"),
                    paste("col", 1:ncol(m), sep=":"))
m

##              col:1      col:2     col:3       col:4      col:5
## row:1 -1.555501865  0.1053054 -1.068104  0.58124134  0.2171429
## row:2 -0.669192742  0.8760066  1.341684 -0.33636965 -1.4043993
## row:3 -0.009041392  0.2168320  0.337415 -0.02566362  1.5110124
## row:4 -0.680032339 -0.8099667 -1.021301  0.70919410 -0.4958138

dimnames(m)

## [[1]]
## [1] "row:1" "row:2" "row:3" "row:4"
##
## [[2]]
## [1] "col:1" "col:2" "col:3" "col:4" "col:5"

rownames(m)

## [1] "row:1" "row:2" "row:3" "row:4"

colnames(m)

## [1] "col:1" "col:2" "col:3" "col:4" "col:5"
```

9 Indexing

Problem 22

Create a vector x with values from 1 to 10. Check:

```r
x <- 1:10
i <- x <= 5
i

##  [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE

x[i]

## [1] 1 2 3 4 5

x[!i]

## [1]  6  7  8  9 10
```

```r
sum(i)

## [1] 5
```

Explain the results.

Problem 23

Index a) the 3. row in matrix m, 2) the 4 row in matrix m and 3) value in 1 row, 1 column.

Sample solution:

```r
m[,3]

##     row:1     row:2     row:3     row:4
## -1.068104  1.341684  0.337415 -1.021301

m[4,]

##      col:1      col:2      col:3      col:4      col:5
## -0.6800323 -0.8099667 -1.0213005  0.7091941 -0.4958138

m[1,1]

## [1] -1.555502
```

Problem 24

How to you get the values of the second row in your matrix m from list l?

Sample solution:

```r
l[["m"]][,2]

## [1]  0.1053054  0.8760066  0.2168320 -0.8099667
```

Problem 25

Make familiar with *data frames*. What are *factors* in R?

Problem 26

Use `expand.grid()` to create a data frame d with 2 factors: factor a with levels a,b,c and factor b with levels a,b,c.

Sample solution:

```
d <- expand.grid(a=letters[1:2], b=letters[1:3])
d

##   a b
## 1 a a
## 2 b a
## 3 a b
## 4 b b
## 5 a c
## 6 b c
```

Problem 27

Use `$` to access variables in d.

Sample solution:

```
d$a

## [1] a b a b a b
## Levels: a b
```

Problem 28

Add a response variable y to d filled with random numbers. Check function `nrow()`! Check result with `str()`. Interpretation?

Sample solution:

```
d$y <- rnorm(nrow(d))
str(d)

## 'data.frame': 6 obs. of  3 variables:
##  $ a: Factor w/ 2 levels "a","b": 1 2 1 2 1 2
##  $ b: Factor w/ 3 levels "a","b","c": 1 1 2 2 3 3
##  $ y: num  0.233 -2.25 0.339 0.807 1.807 ...
##  - attr(*, "out.attrs")=List of 2
##   ..$ dim     : Named int  2 3
##   .. ..- attr(*, "names")= chr  "a" "b"
##   ..$ dimnames:List of 2
##   .. ..$ a: chr  "a=a" "a=b"
##   .. ..$ b: chr  "b=a" "b=b" "b=c"
```

Problem 29

Use data frame d: Index all `b's` in variable a where $y \geq 0$. Use operator `>=` for *greater or equal than*. How many `b's` do you find?

Sample solution:

```
i <- d$a == "b" & d$y >= 0
d$y[i]

## [1] 0.8069537 0.2209517

sum(i)

## [1] 2
```

Problem 30

Try function `summary(d)`. Explain result.

10 Read an write data

Problem 31

Use function `write.table()`, `read.csv()`, `read.csv2()`. What are the differences? Try the functions with data frame d.

11 Formulas in R

Problem 32

R has a compact symbolic form to describe association between variables. See `formula` and help pages of `lm()` (linear model) and `aov` (analysis of variance). Try them!

$y \sim m$	y is expressed by m
`+`	add operator/variable
`-`	remove variable
`.`	all variables
`:`	interaction
`*`	main effect + interaction
`a*b`	same as `a+b+a:b`
`|`	group by

12 Plots

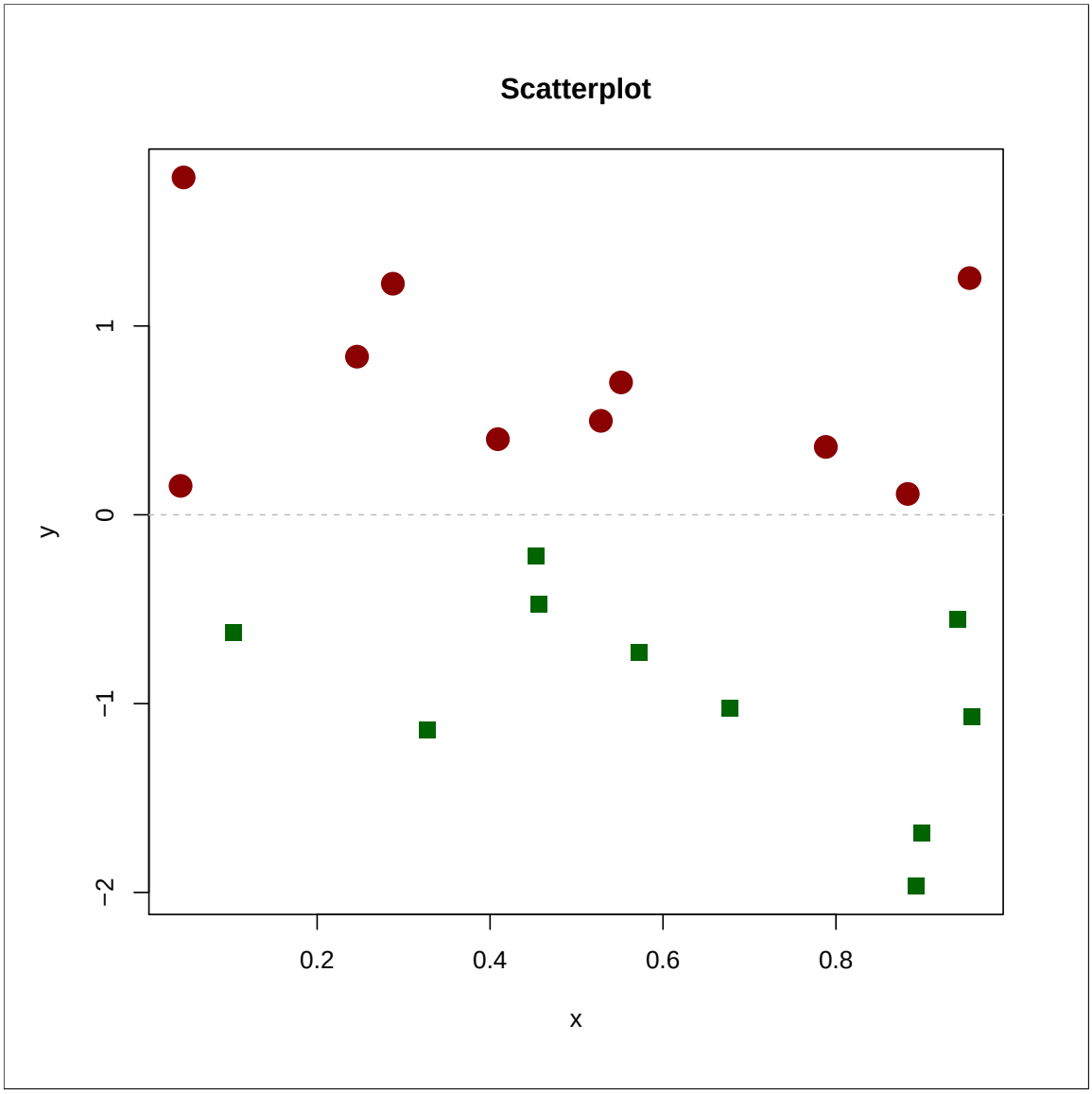
Problem 33

Create two random number vectors x,y each with 20 values. Make a scatter-plot of x and y, and use different colors for y below and y above zero. Use the following R functions:

- plot()
- ifelse()
- At least two color names.
- Function `abline()` to draw a horizontal line at 0.

Sample solution:

```
set.seed(123)
x <- runif(20)
y <- rnorm(20) # mean = 0, sd = 1
i <- y>0
plot(x, y, main="Scatterplot",
     cex=ifelse(i, 3, 1.5),
     col=ifelse(i, "darkred", "darkgreen"),
     pch=ifelse(i, 20,15))
abline(h=0, col="gray", lty=2)
```



Problem 34

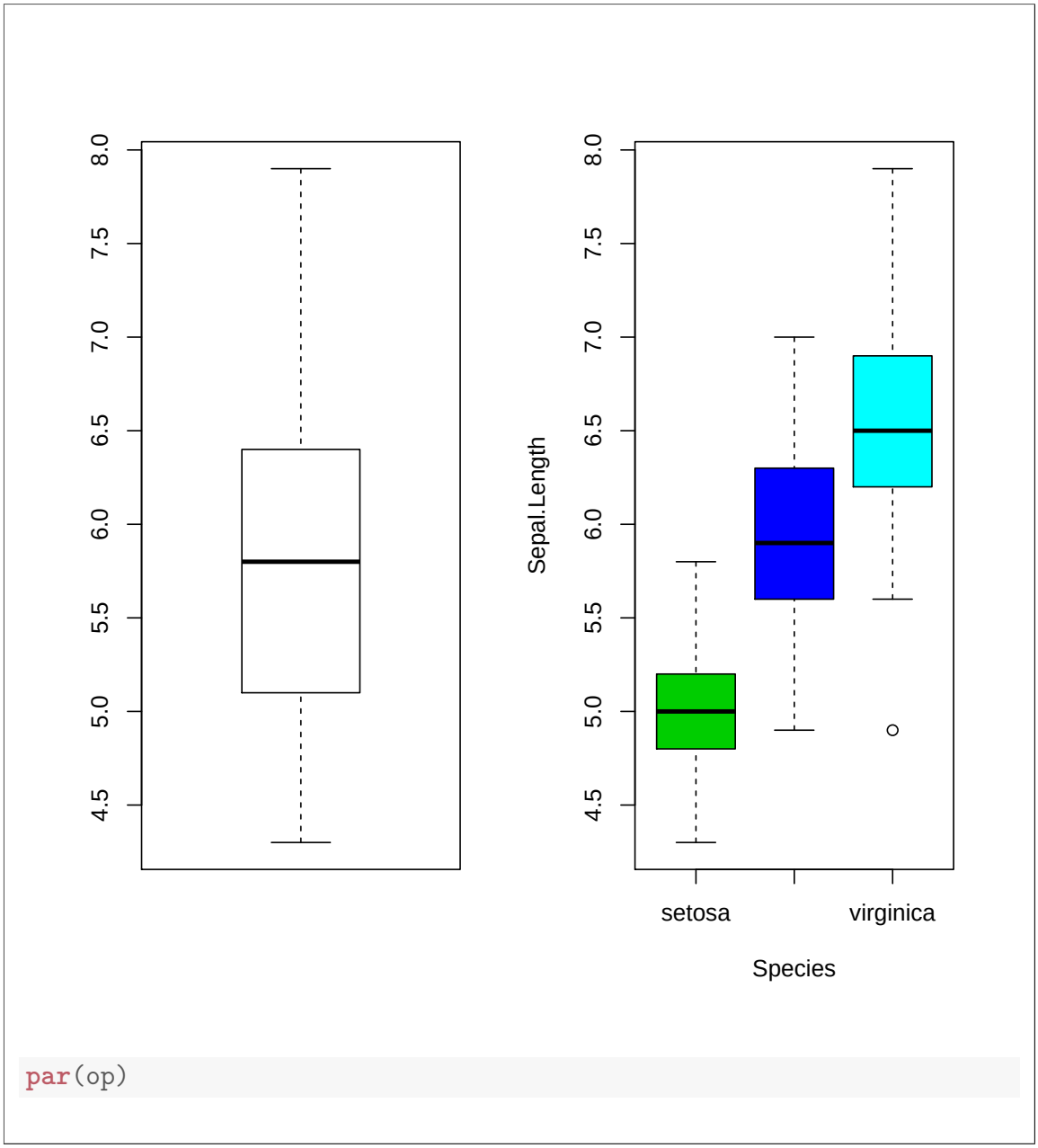
Add a legend to the plot: `legend()`

Problem 35

What are boxplots? Use data set *iris* to explore boxplots.

Sample solution:

```
## layout: 1 row, 2 columns
op <- par(mfrow=c(1,2))
boxplot(iris$Sepal.Length)
boxplot(Sepal.Length ~ Species, data=iris, col=3:5)
```




```
par(op)
```

Problem 36
Use package `lattice` to create more sophisticated plots.

Problem 37
Additional: Install package `ggplot2`, use `install.packages()` or the menu. Create some graphics.

13 Aggregation and new functions

Problem 38
Use function `aggregate()` to compute the mean of y in data frame d for all levels of a and b.

Sample solution:

```
aggregate(y ~ a+b, data=d, FUN=mean)

##   a b          y
## 1 a a  0.2329805
## 2 b a -2.2502269
## 3 a b  0.3394514
## 4 b b  0.8069537
## 5 a c  1.8069123
## 6 b c  0.2209517
```

Problem 39

Do the same with function `tapply()`. Differences?

Sample solution:

```
tapply(d$y, list(d$a, d$b), mean)

##            a         b         c
## a  0.2329805 0.3394514 1.8069123
## b -2.2502269 0.8069537 0.2209517

##
with(d, { tapply(y, list(a, b), mean) })

##            a         b         c
## a  0.2329805 0.3394514 1.8069123
## b -2.2502269 0.8069537 0.2209517
```

Problem 40

Try programming new functions in R: `function()`. Program a function which computes the mean and standard deviation of a vector.

Sample solution:

```r
msstat <- function(x) {
    return(c("mean"=mean(x), "sd"=sd(x)))
}
msstat(1:20)

##     mean       sd
## 10.50000  5.91608
```

14 Tabulation

Problem 41

Use `table()`, `xtabs()` and `ftable()` to tabulate a and b in d. Differences?

Sample solution:

```r
table(d$a)

##
## a b
## 3 3

table(d$a, d$b)

##
##     a b c
##   a 1 1 1
##   b 1 1 1

xtabs( ~ a+b, data=d)

##    b
## a   a b c
##   a 1 1 1
##   b 1 1 1

ftable(a ~b , data=d)

##   a a b
## b
## a   1 1
## b   1 1
## c   1 1
```