Schweizer Paraplegiker Forschung

Exercise: R basics

Workshop – Introduction to R

SOLUTION

1 First look at R and RStudio

a) Open RStudio and create a new R-Script. Save this script in a directory of your choice on your computer.

b) Try to run some commands in R. You can either type the commands directly into the console and run them with "Enter" or you can type them into the script and execute them in the console using the "Run" button (or keyboard shortcut "Ctrl + Enter"):

- Use R as a calculator and execute some calculations.

```
1 + 1

## [1] 2

1 - 1

## [1] 0

1 * 1

## [1] 1

1 / 1

## [1] 1
```

- Create a vector with the name `myvector` with the the four elements 1, 10, 100 and 1000.

```
myvector <- c(1, 10, 100, 1000)
myvector

## [1]    1   10  100 1000
```

- Create a vector with the name `myvector2` containing the whole numbers counting from 50 up to 60.

```
myvector2 <- 50:60
myvector2

##  [1] 50 51 52 53 54 55 56 57 58 59 60
```

- Add the value 10 to all elements of `myvector2` and store the resulting vector under the name `myvector3`.

```
myvector3 <- myvector2 + 10
myvector3

##  [1] 60 61 62 63 64 65 66 67 68 69 70
```

- Change the second element of `myvector3` to the value 8.

```
myvector3[2] <- 8
myvector3

##  [1] 60  8 62 63 64 65 66 67 68 69 70
```

- Create a character vector containing the elements `"a"`, `"bb"` and `"ccc"`, give it a name of your choice.

```
charvec <- c("a", "bb", "ccc")
charvec

## [1] "a"   "bb"  "ccc"
```

- **Extra:** Try to add the number of 10 to all elements of the created character vector and see what happens.

```
charvec + 10

## Error in charvec + 10:  non-numeric argument to binary operator
```

R returns an error because we cannot add a number to text.

- **Extra:** Create a vector which contains some numeric elements and some character elements. Look at the resulting object, what do you think happened?

```
c(1, 2, 3, 'a', 'b', 'c')

## [1] "1" "2" "3" "a" "b" "c"
```

Since numeric elements cannot be combined with character elements in the same vector the numbers are turned into text.

2 Creating a data frame

Since we have not yet discussed how to read in data to R, we will create a small data frame manually for this exercise. Copy the commands below and execute them in R to create the `dat` data frame.

```
id <-  paste0('S', 1:10)
age10 <- c(3, 4, 4, 6, 5, 5, 6, 5, 4, 8)
age1 <- c(4, 2, 1, 0, 8, 7, 2, 3, 5, 1)
status <- c('paraplegic', 'tetraplegic')
score <- c(80.3, 77.9, 89.1, 94.2, 69, 73.8, 81.3, 89, 51, 200)
dat <- data.frame(id, age10, age1, status, score)
```

3 Inspecting the data

a) There are many ways to inspect a data frame in R. The simplest way is to print it directly in the console by typing its name. Look again at the above created `dat` object by typing its name and executing it.

```
dat
```

```
##     id age10 age1      status score
## 1   S1     3    4  paraplegic  80.3
## 2   S2     4    2 tetraplegic  77.9
## 3   S3     4    1  paraplegic  89.1
## 4   S4     6    0 tetraplegic  94.2
## 5   S5     5    8  paraplegic  69.0
## 6   S6     5    7 tetraplegic  73.8
## 7   S7     6    2  paraplegic  81.3
## 8   S8     5    3 tetraplegic  89.0
## 9   S9     4    5  paraplegic  51.0
## 10 S10     8    1 tetraplegic 200.0
```

b) Additionally, there are many functions available to help with data inspection. For example, to get a quick overview of a large data frame, it often helps to only print the first couple of rows to save space. The `head` function accomplishes this task and can be applied to our object in the following fashion:

```
head(dat)
```

Run this command and look at the output.

```
head(dat)
```

```
##   id age10 age1      status score
## 1 S1     3    4  paraplegic  80.3
## 2 S2     4    2 tetraplegic  77.9
## 3 S3     4    1  paraplegic  89.1
## 4 S4     6    0 tetraplegic  94.2
## 5 S5     5    8  paraplegic  69.0
## 6 S6     5    7 tetraplegic  73.8
```

As we can see, the head function prints the first six rows of a data frame.

c) In the following list you see a couple of further useful functions which can be applied to data frames in the same fashion as the `head` function. Try them out and try to understand what their respective output is.

- `dim` function

```
dim(dat)
```

```
## [1] 10  5
```

The dim function shows the dimensions of a data frame, i.e. its number of rows and columns (represented in a vector).

- `View` function

```
View(dat)
```

The View function opens a viewer in RStudio in which the data frame is shown.

- `summary` function

```
summary(dat)
```

```
##       id                age10           age1         status
##  Length:10          Min.   :3.00   Min.   :0.00   Length:10
##  Class :character   1st Qu.:4.00   1st Qu.:1.25   Class :character
##  Mode  :character   Median :5.00   Median :2.50   Mode  :character
##                     Mean   :5.00   Mean   :3.30
##                     3rd Qu.:5.75   3rd Qu.:4.75
##                     Max.   :8.00   Max.   :8.00
##      score
##  Min.   : 51.00
##  1st Qu.: 74.83
##  Median : 80.80
##  Mean   : 90.56
##  3rd Qu.: 89.08
##  Max.   :200.00
```

The summary function returns summary statistics for each column.

- `str` function

```
str(dat)
```

```
## 'data.frame': 10 obs. of  5 variables:
##  $ id    : chr  "S1" "S2" "S3" "S4" ...
##  $ age10 : num  3 4 4 6 5 5 6 5 4 8
##  $ age1  : num  4 2 1 0 8 7 2 3 5 1
##  $ status: chr  "paraplegic" "tetraplegic" "paraplegic" "tetraplegic" ...
##  $ score : num  80.3 77.9 89.1 94.2 69 73.8 81.3 89 51 200
```

The str function shows the structure of an R object. For data frames it lists all colunns and gives some information about the type of each column.

4 Preprocessing the data

a) We continue working with the `dat` object. When we look at the data frame we can see that there are two age columns, one called `age10` and one `age1`. For some reason the data has been collected in a format where the age of people has been divided into two columns. A person with the value `age10` $= 4$ and the value `age1` $= 5$ actually has the age 45. Try to add a new `age` column to the data frame which contains the calculated age for every person.

```
dat$age <- 10*dat$age10 + dat$age1
dat
```

```
##    id age10 age1      status score age
## 1  S1     3    4  paraplegic  80.3  34
## 2  S2     4    2 tetraplegic  77.9  42
## 3  S3     4    1  paraplegic  89.1  41
## 4  S4     6    0 tetraplegic  94.2  60
```

```
## 5   S5     5    8  paraplegic  69.0  58
## 6   S6     5    7 tetraplegic  73.8  57
## 7   S7     6    2  paraplegic  81.3  62
## 8   S8     5    3 tetraplegic  89.0  53
## 9   S9     4    5  paraplegic  51.0  45
## 10 S10     8    1 tetraplegic 200.0  81
```

b) Since we do not need the `age10` and `age1` columns anymore, remove them from the data frame.

```
dat$age10 <- NULL
dat$age1 <- NULL
dat
```

```
##     id      status score age
## 1   S1  paraplegic  80.3  34
## 2   S2 tetraplegic  77.9  42
## 3   S3  paraplegic  89.1  41
## 4   S4 tetraplegic  94.2  60
## 5   S5  paraplegic  69.0  58
## 6   S6 tetraplegic  73.8  57
## 7   S7  paraplegic  81.3  62
## 8   S8 tetraplegic  89.0  53
## 9   S9  paraplegic  51.0  45
## 10 S10 tetraplegic 200.0  81
```

c) **Extra:** The data frame contains the `score` variable which expresses the performance in a task. These scores can only range from 0 to 100. However, in the data there is an erroneous entry with a value that is impossibly large. Since we do not know what the correct value for this person was, we want to set this element to "missing". In R, the value `NA` (no quotes, just NA) is especially reserved for missing values. Try to assign the value `NA` to the erroneous entry in the `score` column.

```
dat[dat$score > 100, "score"] <- NA
dat
```

```
##     id      status score age
## 1   S1  paraplegic  80.3  34
## 2   S2 tetraplegic  77.9  42
## 3   S3  paraplegic  89.1  41
## 4   S4 tetraplegic  94.2  60
## 5   S5  paraplegic  69.0  58
## 6   S6 tetraplegic  73.8  57
## 7   S7  paraplegic  81.3  62
## 8   S8 tetraplegic  89.0  53
## 9   S9  paraplegic  51.0  45
## 10 S10 tetraplegic    NA  81
```

5 Selection of elements

Try to perform the following selections in the `dat` data frame.

a) Only show the rows (and all columns) of people with an age higher than 60.

```
dat[dat$age > 60, ]
```

```
##     id      status score age
## 7   S7  paraplegic  81.3  62
## 10 S10 tetraplegic    NA  81
```

b) Only show the rows (and all columns) of tetraplegic patients.

```
dat[dat$status == 'tetraplegic', ]
```

```
##     id      status score age
## 2   S2 tetraplegic  77.9  42
## 4   S4 tetraplegic  94.2  60
## 6   S6 tetraplegic  73.8  57
## 8   S8 tetraplegic  89.0  53
## 10 S10 tetraplegic    NA  81
```

c) **Extra:** Show all rows but only the columns `id`, `status` and `age`

```
dat[, c("id", "status", "age")]   # Selection with names in vector is also possible
```

```
##     id      status age
## 1   S1  paraplegic  34
## 2   S2 tetraplegic  42
## 3   S3  paraplegic  41
## 4   S4 tetraplegic  60
## 5   S5  paraplegic  58
## 6   S6 tetraplegic  57
## 7   S7  paraplegic  62
## 8   S8 tetraplegic  53
## 9   S9  paraplegic  45
## 10 S10 tetraplegic  81
```

6 The AND and OR operators

Extra: We can create more complex selections by combining multiple logical vectors. For example, the AND operator (written in R as `&`) only returns the value `TRUE` if all the corresponding logical elements are `TRUE`. In the following code we combine two logical vectors using the AND operator:

```
c(TRUE, FALSE, TRUE) & c(FALSE, FALSE, TRUE)
```

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

As we can see, in the resulting logical vector only the third element is `TRUE` because only the third element was `TRUE` in both vectors.

The OR operator (written in R as `|`) on the other hand returns the value `TRUE` if at least one logical value is `TRUE`. Let's look at the result if we apply it to the same vectors as above:

```
c(TRUE, FALSE, TRUE) | c(FALSE, FALSE, TRUE)
```

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

In the resulting vector only the second element is `FALSE` because at the second position neither the first nor the second vector contained a `TRUE` value.

a) We can use such combinations of logical vectors to make more complex selections in our data. We will again work with the `dat` object from the previous exercise. Try to use the AND operator to select only the rows (and all columns) of people who are paraplegic **and** have an age above 55.

```
dat[dat$status == 'paraplegic'  &  dat$age > 55, ]

##   id     status score age
## 5 S5 paraplegic  69.0  58
## 7 S7 paraplegic  81.3  62
```

b) In the same fashion, try to use the OR operator to select only the rows of people who are either paraplegic **or** have an age above 55.

```
dat[dat$status == 'paraplegic'  |  dat$age > 55, ]

##     id      status score age
## 1   S1  paraplegic  80.3  34
## 3   S3  paraplegic  89.1  41
## 4   S4 tetraplegic  94.2  60
## 5   S5  paraplegic  69.0  58
## 6   S6 tetraplegic  73.8  57
## 7   S7  paraplegic  81.3  62
## 9   S9  paraplegic  51.0  45
## 10 S10 tetraplegic    NA  81
```

7 R cheatsheets

We have added to the github page (`https://github.com/Swiss-Paraplegic-Research/Workshop/tree/main/RCheatSheets`) a couple of different R cheatsheets which contain a lot of information about the use of R. Take a look at the file "baseR_cheatsheet.pdf".