Workshop: Data Wrangling of Web Data in R

Simon Ress | Ruhr-Universität Bochum

October 18, 2021

Content

1. Setup
2. Functions
3. Iteration: Loops & Apply-family
4. Dplyr - Gramma of Data Manipulation
5. Purr
6. Helpful Sources
7. Helpful sources

Setup

Target

Meta information

- Finanzausschuss
- Ausschüsse der 19. Wahlperiode (2017-2021)
- Öffentliche Anhörungen

URL: https://www.bundestag.de/webarchiv/Ausschuesse/ausschuesse19/a07/Anhoerungen

Unit information

- Committees

URL: Needs to be scraped from main page

Configurate & Start Selenium/Browser

```r
library(RSelenium)
library(rvest) #for read_html(), html_elements()...
#Free all ports
  system("taskkill /im java.exe /f", intern=FALSE,
   ↪  ignore.stdout=FALSE)
#Start a selenium & Assign client to an R-object
  rD <- rsDriver(port = 4561L, browser = "firefox")
  remDr <- rD[["client"]]
  #remDr$quit
```

Functions

Overview

- Functions are **blocks of codes** which can be executed repeatedly by calling them
- **Parameters** (data) can be passed into them, which are used by the code inside
- **Data can be returned** from a function

Syntax:

```
function_name <- function(arg_1, arg_2, ...) {
     Function body
}
```

Function Components

The four parts of a function are:

- **Function Name:** This is the actual name of the function. It is stored in R environment as an object with this name.
- **Arguments (*optional*):** An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments *can* have default values.
- **Function Body:** The function body contains a collection of statements that defines what the function does.
- **Return Value:** The return value of a function is the last expression in the function body to be evaluated.

Examplary Function

```r
square <- function(value = 1, factor = 1) {
    return(value^factor)
}
square() #use defaut args
```

```
## [1] 1
```

```r
square(2,3) #use args by position
```

```
## [1] 8
```

```r
square(factor=2, value=5) #use args by name
```

```
## [1] 25
```

Define savepage()

```r
#Load url & return content as r-object
  savepage <- function(url){
    #Navigate to starting page
      remDr$navigate(url)
    #Wait until page is loaded
      Sys.sleep(abs(rnorm(1, 2, 1)))
    #Save content to an R-object
      remDr$getPageSource(header = TRUE)[[1]] %>%
        read_html() %>%
        return()
  }
```

Note: [[1]] behinde getPageSource() unlist the output -> makes it searchable

Usage of savepage()

```r
#navigate to url & save content as r-object
page <- savepage("https://www.bundestag.de/
↪   webarchiv/Ausschuesse/ausschuesse19/a07/
↪   Anhoerungen")
page
```

```
## {html_document}
## <html xml:lang="de" dir="ltr" class="detection-firefox"
## [1] <head>\n<meta http-equiv="Content-Type" content="te
## [2] <body class="bt-archived-page">\n  <div class="bt-a
```

Iteration: Loops & Apply-family

Overview

for-loop

- A for loop is used for iterating over a sequence:
- With the break statement, we can stop the loop before it has looped through all the items:
- With the next statement, we can skip an iteration without terminating the loop:

```
for (x in 1:10) {
  if (x == 4)  break
  print(x)
}
```

```
## [1] 1
## [1] 2
## [1] 3
```

for-loop: break

Breaking the loop at certain conditions

```r
for (x in cars$dist) {
  if (x > 20)  break
  print(x)
}
```

```
## [1] 2
## [1] 10
## [1] 4
```

for-loop: next

Skip the code below and start over at certain conditions

```r
fruits <- list("apple", "banana", "cherry")

for (x in fruits) {
  if (x == "banana") next
  print(x)
}
```

```
## [1] "apple"
## [1] "cherry"
```

while-loop

- Execute a set of statements as long as a condition is TRUE
- *break* statement stops the loop even if the while condition is TRUE:
- *next* statement skips an iteration without terminating the loop:

```r
i <- 0
while (i < 20) {
  i <- i + 1
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

while-loop: break

Breaking the loop at certain conditions

```r
i <- 0
while (i < 20) {
  i <- i + 1
  if (i == 5) break
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

while-loop: next

Skip the code below and start over at certain conditions

```r
i <- 0
while (i < 10) {
  i <- i + 1
  if (i %% 2) next
  print(i)
}
```

```
## [1] 2
## [1] 4
## [1] 6
## [1] 8
## [1] 10
```

apply-family

- The apply in R function can be feed with many functions to perform redundant application on a collection of object (data frame, list, vector, etc.).
- The purpose of apply() is primarily to avoid explicit uses of loop constructs.
- Any function can be passed into

Main apply functions

Function	Arguments	~~Objective~~	Input	Output
apply	apply(x, MARGIN, FUN)	Apply a function to the rows or columns or both	Data frame or matrix	vector, list, array
lapply (list)	lapply(X, FUN)	Apply a function to all the elements of the input	List, vector or data frame	list
sapply (simple)	sapply(X, FUN)	Apply a function to all the elements of the input	List, vector or data frame	vector or matrix
tapply (tagged)	tapply(X, grouping, FUN)	Apply a function for each factor variable in an vector	Vector	matrix or array

apply()-usage

gh

lapply()-usage

gg

sapply()-usage

dd

tapply()-usage

aa

Dplyr - Gramma of Data Manipulation

Overview

Purr

Overview

“purrr enhances R’s functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors.”

```r
if(!require("purrr")) install.packages("purrr")
  library(purrr) # for fill()
mtcars %>%
  split(.$cyl) %>% # from base R
  map(~ lm(mpg ~ wt, data = .)) %>%
  map(summary) %>%
  map_dbl("r.squared")
```

```
##         4         6         8
## 0.5086326 0.4645102 0.4229655
```

Helpful Sources

Helpful Sources

- purr: Overview
- purr: References
- purr: Cheatsheet

Helpful sources

Helpful sources

- Stringr: Overview
- Stringr: Introduction
- Stringr: Cheatsheet
- Stringr: Reference manual
- Base R String-functions vs Stringr
- Working with strings in R
- Regular expressions
- Primary R functions for dealing with regular expressions

References

All graphics are taken from String manipulaton with stringr Cheatsheet