Workshop: Data Wrangling of Web Data in R

Simon Ress | Ruhr-Universität Bochum

October 18, 2021

Content

1. Setup
2. Functions
    - Define savepage()
    - Usage of savepage()
3. Loops & apply-family
    - Overview
    - for-loop
    - while-loop
    - apply functions
4. Dplyr - Gramma of Data Manipulation
    - Overview
5. Purr
    - Overview

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

```
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 different 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:** An argument is a placeholder. When a function is invoked, you pass a value to the argument. Arguments are *optional*; that is, a function may contain no arguments. Also 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

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

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

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

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

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

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

Define savepage()

```
#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()

```
#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
```

Loops & apply-family

while-loop

- With the while loop we can execute a set of statements as long as a condition is TRUE
- With the break statement, we can stop the loop even if the while condition is TRUE:
- With the next statement, we can skip an iteration without terminating the loop:

```
i <- 1
while (i < 6) {
  print(i)
  i <- i + 1
}
```

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

Dplyr - Gramma of Data Manipulation

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."

```
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