

PRE-PRESENTATION

- Welcome to this R course!
- What you can already do:
 - Launch R/R Studio and create a new script
 - Set your `working` directory to a folder on your drive P
 - Install and load the packages `rvest`, `RSelenium` and `tidyverse`
 - Download the files `uebung1.html`, `uebung2.html` and `WebScrapingInR.pdf` from the *Lernraum*
- Run the following R Code

```
library(RSelenium)
rD <- rsDriver(browser=c("firefox")) # customize browser
```

and if it does not work,

- search the web for solutions
- install a Java version that is in accordance with your R version (32 or 64 bit), restart RStudio and try again
- use another browser / reinstall your browser
- install a more recent R and RStudio version

Web Scraping in R

26.03.2021

Peter Pütz

peter.puetz@uni-bielefeld.de

Agenda

- 1 Introduction: Web Scraping?
- 2 Crash course: Structure of a website
- 3 Scraping websites with R – the package `rvest`
 - Overview of important commands
 - Example: Scraping `www.immowelt.de`
- 4 Navigating through the web with R – the R package `RSelenium`
- 5 Miscellaneous

About this course

- This is an advanced R course, thus...
- ...skills gained during the R courses 'First steps in R', 'Tidy Data in R' and 'Graphics in R' may be helpful
- Today's foci
 - Making data from websites accessible in R
 - Reading out entire websites into R automatically
 - Navigating to websites via R
- Overall aims
 - Getting an idea why web scraping might be useful
 - Getting the basic skills and material to look up for scraping the web

Why web scraping?

- An empirical study often starts with the collection of data
- There are many datasets freely available and easily accessible in the web (e.g. offered by the OECD, The World Bank, Statistical Offices)
- However, many websites include information not offered in a `.csv`, `.xlsx` or similar file format
- A (time-consuming and error-prone) option: Copying the data from the website of interest
- Another option: Using the R package `rvest`
 - Advantages: Time saving, reproducible
 - Necessary: Very basic knowledge of the structure of a website

Notation

- R code / commands are displayed in this font:
`cat("We love R")`
- While presenting functions / commands ↑ indicates advanced options which are not further discussed and should be left to default

Boxes include examples

Crash course: Structure of a website

HTML (Hypertext Markup Language)

- HTML is a text-based, machine-readable language
- HTML structures electronic documents such as texts with hyperlinks, pictures and further contents
- HTML documents are the basis of the World Wide Web
- Aside from HTML other languages are used to format / create a website, e.g. CSS for the visual appearance or JavaScript for the functionality
- Yet, HTML is the "end product" sent to the browser

EXERCISE

1. Open a text editor, e.g. *Editor*, *Notepad* or the like
2. Copy the text written on the next slide into your document and save it with the file ending *.html*.
3. Close the document and open it again, e.g. by a double click.

HTML Basics

```
<html>
<head>
<title> A very cool website </title>
</head>

<body>
<h1> Hello World! </h1>
<p> Hooray! </p>
</body>
</html>
```

Structure of a website

- HTML tags assign a structure to texts and pictures of a website and provide the browser with this structure
- Tags begin with `<...>` and end with `</...>`
- Every document begins with `<html>` and ends with `</html>`
- In the `<head>...</head>` environment you find general information on the document, e.g. on...
- ...the title shown in the browser (`<title>...</title>`)
- The `<body>...</body>` environment includes text, links etc.

Structure of a website

Elements that can be inserted into `<body>...</body>`:

- `<h1>`, `<h2>` etc. are headlines
- `<p>` indicate a paragraph
- `
` causes a linebreak
- Also relevant: Pictures, links and tables

Structure of a website

Pictures

- Pictures are added via `` (Standalone Tag without closing `< /... >`)
- `src` indicates the source/url of the picture
- `alt` denotes an alternative text if the picture is not found
- `src` and `alt` are attributes, they provide additional information about the tag / configure the tag

```
<html>
<head>
<title> A fancy website including a picture </title>
</head>
<body>
<h1> R course </h1>
<p> Here comes a picture</p>

</body>
</html>
```

Structure of a website

Links

- By `<a> ... ` links are included
- `href` denotes the path
- `title` defines the text shown when scrolling over the link

```
<html>
<head>
<title> A fancy website including a picture </title>
</head>
<body>
<h1> R course </h1>
<p>Here comes a picture</p>

<a href="http://www.startpage.com" title="Search engine">
Here you can move to a search engine to learn more about R</a>
</body>
</html>
```

Structure of a website

Tables

- By `<table> ... </table>` tables are created
- `border` creates a frame around the cells of the table
- `<tr>` leads to a jump into a new line
- `<td>` creates single cells

```
...  
<table border="1">  
  <tr>  
    <td>cell 1</td>  
    <td>cell 2</td>  
  </tr>  
  <tr>  
    <td>cell 3</td>  
    <td>cell 4</td>  
  </tr>  
</table>  
...
```

Structure of a website

Design elements

- Coloured paragraphs are obtained by, e.g.,

```
<p><font color='red'> This is a coloured sentence. </p></font>
```

- There are many more design possibilities: `` leads to bold printing, `<u>` underlines the text, ...
- What are the downsides of such a way of designing websites?

Structure of a website

■ Example: News platform

- Politics: grey text
- Economy: orange text
- Sports: darkblue text
- Other news: black text

```
...  
<h1> News</h1>  
<p><font color="orange"> Economic headline 1 </p></font>  
<p><font color="orange"> Economic headline 2 </p></font>  
<p><font color="grey"> Politics headline 1 </p></font>  
<p><font color="darkblue"> Sports headline 1 </p></font>  
<p><font color="grey"> Politics headline 2 </p></font>  
<p><font color="black"> Miscellaneous 1 </p></font>  
<p><font color="darkblue"> Sports headline 2 </p></font>  
<p><font color="black"> Miscellaneous 2 </p></font>  
<p>...</p>  
...
```

Structure of a website

Stylesheets

- Looking at the previous example: Changing colours means lots of work
- Easier by using 'Stylesheets'
- CSS: Cascading Style Sheets

Structure of a website

Stylesheets

- CSS can be included (e.g.) in `<head>...</head>`:

```
<head>
<style>
.economics {
color:orange;
}
</style>
</head>
...
```

- The part before the curly brackets is called selector
- class selectors start with a "." in the CSS-definition

Structure of a website

Stylesheets

- CSS can be included (e.g.) in `<head>...</head>`:

```
<head>
<style>
#economics {
color:orange;
}
</style>
</head>
...
```

- id selectors start with a "#" in the CSS-definition
- id selectors are used for elements appearing only once, whereas by class recurring design elements are defined

Structure of a website

Stylesheets

- CSS elements can be referred to by `id` and `class` selectors

```
<html>
<head>
<style>
.economics {color:orange;}
</style>
</head>
<body>
<h1> News</h1>
<p class="economics"> Should economic activities serve
the common good? </p>
</body>
</html>
```

Structure of a website

Excursion: CSS selectors

- CSS selectors are used to select the content you want to style
- CSS selectors select HTML elements according to its id, class, type, attribute etc.
- As we will see later, CSS selectors can be used to access basically everything from an html document

Virtues of CSS

- Separation of content and style
- Time-saving
- Alterations are facilitated

EXERCISE

1. Visit a website of your choice.
2. Right-click and click something like “view source code” (depends on the browser, see <https://kb.iu.edu/d/agao>).
3. Do you recognize anything?

Scraping websites with R – the package `rvest`

Web scraping with R

rvest

- The command `read_html()` reads out the source code of a website:

```
read_html (x, encoding = "", ...,  
options = c("RECOVER", "NOERROR",  
"NOBLANKS"))
```

- `x` has to be a string specifying the link/url
- ↑ `encoding`, `options`
- The command creates an object of the class `xml_document` including the source code

Web scraping with R

rvest

- The command `html_nodes()` extracts parts from the source code, e.g. single HTML tags or CSS selectors:
`html_nodes (x, css, xpath)`
- `x`: Object including the source code
- `css`: HTML tags or CSS selectors
- `↑ xpath`
- The command creates an object of the class `xml_nodeset` that includes source code

Web scraping with R

Finding CSS selectors

To find CSS selectors in the source code, common browsers provide a simple opportunity:

- Select text ~> right-click ~> inspect(Chrome) / inspect element (Firefox), then...
 - either: ~> right-click the selected source code ~> Copy ~> CSS selector
 - or: Write down respective selector directly, e.g. for a class: *.selectorname*

Web scraping with R

rvest

- The command `html_text()` deletes all HTML tags from the source code:

```
html_text (x, trim = FALSE)
```

- `x`: Object including the source code
- `trim`: Delete leading and trailing whitespaces?
- The command creates an object of the class `character`

EXERCISE

1. Move to the website <https://uni-bielefeld.de/>.
2. Scrape the current month by using the three commands learned on the last slides.

Web scraping with R

rvest

- The command `html_table()` extracts table(s) from the source code:

```
html_table (x, header = NA, trim = TRUE, fill = FALSE, dec = ".")
```
- `x`: Object including the source code
- `header`: Use first row as header?
- `trim`: Delete leading and trailing whitespaces?
- `fill`: Fill missing values in rows with number of columns smaller than maximum number of columns with NA?
- `dec`: Character used as decimal mark
- The command creates an object of the class `list` with each entry of the list being a `data.frame`

EXERCISE

1. Move to the website <https://en.wikipedia.org/wiki/Bielefeld>.
2. Scrape a table you are interested in. For this, you may either
 - select one entry of the list of data.frames that you obtain by applying `html_table()` to the source code of the whole website
 - apply `html_table()` to the source code of the table selected by `html_nodes()`

Note: Sometimes it is tough to find the right selector, but it might be the right one if it includes the word *table*

EXERCISE

1. Download the file `uebung1.html` from the *Lernraum* and save it in a folder on your P drive. Open it in your browser to take a look at it.
2. Scrape the two sports headline, i.e. the result is supposed to be character vector in R with two entries.
3. Also scrape the politics headline.

Web scraping with R

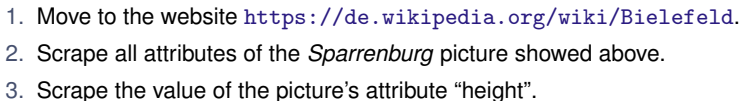
Scraping attributes

- The command `html_attrs()` extracts attributes from the source code:
`html_attrs (x)`
- `x`: Object including the source code
- The command creates an object of the class `list`

Web scraping with R

Scraping attributes

- The command `html_attr()` extracts the content from a specified attribute:
`html_attr (x, name, default = NA_character_)`
- `x`: Object including the source code
- `name` Name of the attribute
- `↑ default`
- The command creates an object of the class `character`



EXERCISE

Download the file `uebung2.html` from the *Lernraum* and scrape...

1. the alternative text of the picture
2. the file name (source) of the picture
3. the table on the page. What is the dimension of the resulting `data.frame`?

Web scraping with R – an exemplary application

EXERCISE

1. Move to the website <https://www.immowelt.de/liste/bielefeld/wohnungen/mieten?sort=relevanz> and left-click on a flat advertisement .
If available, scrape the following information:
 - Basic rent (Kaltmiete)
 - Living area (Wohnfläche)
 - Number of rooms (Zimmeranzahl)
 - Furnishing (Ausstattung)
2. Do the same for another flat ad. You may write a function or a loop to do this (I guess you have not learned this so far!?), but this is definitely not necessary. Store the results in a `data.frame` of dimension 2 x 4.

More efficiency needed for real fun...

- So far, we manually headed for single websites to scrape data
- This is already cool, but highly inefficient if we want to obtain data from many websites / URLs

Navigating through the web with R – the package `RSelenium`

Navigating through the web with R

RSelenium

- Using `RSelenium` allows to navigate through the web via R
- The package is also beneficial if websites use *JavaScript*
 - Example: Scrolling down websites to show further content

Navigating through the web with R

RSelenium

- The command `rsDriver` starts a selenium server and browser
`rsDriver (browser, ...)`
- `browser`: the browser that should be opened and used for navigation via R,
e.g. *firefox*
- ↑...
- The command creates a `(list)` including a server and a web client

```
rD <- rsDriver(browser=c("firefox"))  
remDr <- rD$client
```

Navigating through the web with R

RSelenium

- The client in the browser enables us to move to websites (via R!)

```
url <- "https://www.immowelt.de/"  
remDr$navigate(url)
```

EXERCISE

1. Start a browser that can be steered via R commands and use R commands to move to the website <https://www.immowelt.de/> (see the last slides).
2. Try to run and understand the R commands on the following slide, they are supposed to do the following:
 - Inserting “Bielefeld” in the “Wo suchen Sie?” field and confirming the entry
 - Scrolling down the entire page
 - Scraping all URLs for flat ads; these are all URLs starting with <https://www.immowelt.de/expose/...>)
3. For advanced scrapers (later or homework): Scrape all URLs of all available flat ads for Bielefeld.

Exercise R Code

```

ortsfeld <- remDr$findElement(using = "css",
"#tbLocationInput")
ortsfeld$sendKeysToElement(list("Bielefeld"))
ortsfeld$sendKeysToElement(list("", key = "enter"))
webElem <- remDr$findElement("css", "body")
webElem$sendKeysToElement(list(key = "end"))
table.body <- remDr$findElements(using = "class",
"modern_browser")
html <- unlist(sapply(table.body, function(x) {
x$getElementAttribute("innerHTML"))}))
links <- html %>% read_html() %>%
  html_nodes("a") %>%
  html_attr("href") %>%
  str_subset(pattern = "/expose/") %>% unique()
links <- paste("https://www.immowelt.de", links, sep="")

```

As amazing scraping might be...

- Usually, scraping is only a first step to collect data
- Data collection and processing need many further steps
- Hint: In the web, you may find numerous tips and tutorials to process (character) strings

Legal issues

- Robots exclusion protocols (*website/robots.txt*) contain information on the permissions to scrape, e.g. <https://www.immowelt.de/robots.txt>, see also https://en.wikipedia.org/wiki/Robots_exclusion_standard
- Whether you may face legal issue depends on the the extent and type of data you scrape and use
- Be cautious when encountering defensive measures (e.g. captchas)
- Especially for research purposes it might be advisable to ask the maintainer of the website for permission

Maybe useful complementary tools

- Scraping PDFs from the web, e.g. https://www.bielefeld.de/sites/default/files/datei/2021/Bevoelkerung_31.12.2020n.pdf
(↪ R package pdfutils)
- Running a script automatically at a certain time
(↪ R package taskscheduleR)
- ...
- You will find further tools as well as more literature on what we learned today in the web, e.g. on
 - <https://stevie42.github.io/webscraping/book/bagofwords.html>
 - <https://smac-group.github.io/ds/web-scraping-in-r.html>

Final exercise

1. Move to the website <https://www.wegweiser-kommune.de/statistik/bielefeld+nachhaltigkeit-sdgs+2016-2018+tabelle>.
2. On the right side, add a couple of "Vergleichskommunen". For practice, you may use RSelenium for that, but this is not necessary.
3. Extract the resulting table from the website.
4. Manipulate the data until it fits the needs of R to do some calculations. You may pay attention to missing values, decimal signs,
5. For the sake of comparison, you may download the same table from the website directly (via the field "Speichern").

Note: This is only an exemplary exercise (for a German website - sorry!), but feel free to work on anything you are (more) interested in!

Thank you very much for your attention!