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 course material from the *Lernraum*
- Run the following R Code

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

A new (empty) browser window should emerge. If not,

- install a `Java` version that is in accordance with you R version (32 or 64 bit), restart RStudio and try again
- use another browser (in the code snippet above), e.g. ”chrome”
- reinstall your browser and try again
- install a more recent R and RStudio version and try again
- search the web for solutions

UNIVERSITÄT BIELEFELD
Fakultät für Wirtschaftswissenschaften

Web Scraping in R

18.03.2022

Peter Pütz
peter.puetz@uni-bielefeld.de

Agenda

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

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*, *Word*, *RStudio* 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. If the text on the next slide appears again (it should not), try another editor.

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>
- The <head>...</head> tag contains general information on the document, e.g. on the title shown in the browser (<title> ... </title>)
- The <body> ... </body> contains text, links etc.

UNIVERSITÄT BIELEFELD
Fakultät für Wirtschaftswissenschaften

Structure of a website

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

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

Structure of a website
Pictures

- Pictures are added via <img> (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>
<img src="R_Logo.png" alt="R icon">
</body>
</html>
```

Structure of a website

Links

- By <a> ... </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>
<img src="R_Logo.png" alt="R icon">
<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> ... </tr> leads to a jump into a new line
- <td> ... </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: <b> 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 solution: 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 elements can, e.g., be referred to by `class` selectors

```
<html>
<head>
<style>
.economics {color:orange;}
</style>
</head>
<body>
<h1> News</h1>
<p class="economics"> Economic headline </p>
</body>
</html>
```

Structure of a website
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`

Notation

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

Boxes include examples

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 of the website
- $\Uparrow$ `encoding, options`
- Output: 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 of the website
- `css`: HTML tags or CSS selectors
- ⇑ `xpath`
- Output: The command creates an object of the class `xml_nodeset` that includes the extracted 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 $\rightsquigarrow$ right-click $\rightsquigarrow$ inspect(Chrome) / inspect element (Firefox), then...
  - either: $\rightsquigarrow$ right-click the selected source code $\rightsquigarrow$ Copy $\rightsquigarrow$ 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 all the three commands learned on the last slides.

Hints: Make sure the packages you need are loaded (see Slide 1). Ask your colleagues, the course instructor, the slides, the R help or the web if anything does not work.

Web scraping with R

`rvest`

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

  `html_table (x, header = NA, trim = TRUE, dec = ".")`
- `x`: Object including the source code
- `header`: Use first row as header?
- `trim`: Delete leading and trailing whitespaces?
- `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
   - apply `html_table()` to the source code of the whole website and select the table of interest from the resulting list of data.frames
   - Select the source code of the table of interest first by using `html_nodes()` (see last slides) and then apply `html_table()`

Hint: 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 headlines, 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`

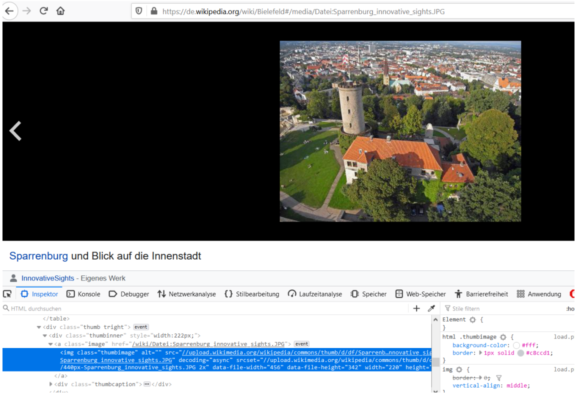
UNIVERSITÄT BIELEFELD
Fakultät für Wirtschaftswissenschaften

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
- $\Uparrow$ `default`
- The command creates an object of the class `character`

EXERCISE



Sparrenburg und Blick auf die Innenstadt

InnovativeSights - Eigenes Werk

```
        </table>
      <div class="thumb tright"> event
        <div class="thumbinner" style="width:222px;">
          <a class="image" href="/wiki/Datei:Sparrenburg_innovative_sights.JPG"> event
            <img class="thumbimage" alt="" src="//upload.wikimedia.org/wikipedia/commons/thumb/d/df/Sparrenb…nnovative_sig
            Sparrenburg_innovative_sights.JPG" decoding="async" srcset="//upload.wikimedia.org/wikipedia/commons/thumb/d/c
            /440px-Sparrenburg_innovative_sights.JPG 2x" data-file-width="456" data-file-height="342" width="220" height="
          </a>
          <div class="thumbcaption"> ⋯ </div>
```

```
Element {
}
html .thumbimage {                          load.p
    background-color: #fff;
    border: 1px solid #c8ccd1;
}
img {                                       load.p
    ~~border: 0;~~
    vertical-align: middle;
```

1. Move to the website `https://de.wikipedia.org/wiki/Bielefeld`.
2. Scrape all attributes of the *Sparrenburg* picture showed above.
3. Scrape the value of the picture's attribute "height".

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 (=number of rows and columns) 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)
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 3.
3. Calculate the mean rent of both flats. This might require the transformation of the scraped data (you might search the web for help).

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

UNIVERSITÄT BIELEFELD
Fakultät für Wirtschaftswissenschaften

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?” 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 the 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$sendKeysToElement(list(key = "end"))
url <- remDr$getCurrentUrl() %>%
    unlist()
links <- url %>%
read_html() %>%
    html_nodes("a") %>%
    html_attr("href") %>%
    str_subset(pattern = "/expose/") %>%
    unique()
```

UNIVERSITÄT
BIELEFELD
Fakultät für
Wirtschaftswissenschaften

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 data, e.g. for processing (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

UNIVERSITÄT BIELEFELD
Fakultät für Wirtschaftswissenschaften

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`
  ($\rightsquigarrow$ R package `pdftools`)
- Running a script automatically at a certain time
  ($\rightsquigarrow$ 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://steviep42.github.io/webscraping/book/bagofwords.html`
  - `https://smac-group.github.io/ds/web-scraping-in-r.html`

More specialized web scraping tools

- Get data from OpenStreetMap, e.g. `https://www.r-bloggers.com/2018/11/accessing-openstreetmap-data-with-r/`
  ($\rightsquigarrow$ R package `osmdata`)
- Get twitter data , e.g. `https://towardsdatascience.com/a-guide-to-mining-and-analysing-tweets-with-r-2f56818fdd16`
  ($\rightsquigarrow$ R package `rtweet`)

Exercise

Feel free to work on previous exercises or to scrape data from a website of your interest.

Thank you very much for your attention!