1 Web Scraping

Problem 1: CSS Diner

Answers to the CSS Diner exercise:

1. plate
2. bento
3. fancy
4. plate apple
5. fancy pickle
6. apple.small/.small
7. orange.small
8. bento orange.small
9. plate, bento
10. *
11. plate *
12. plate + apple
13. bento + pickle
14. plate > apple
15. orange:first-child
16. plate apple:only-child, plate pickle:only-child
17. apple:last-child, pickle:last-child
18. :nth-child(3)
19. bento:nth-last-child(3)
20. apple:first-of-type
21. :nth-of-type(even)
22. plate:nth-of-type(3n+2)
23. plate apple:only-child
24. apple:last-of-type, orange:last-of-type
25. bento:empty
26. apple:not(.small)

27. [for]
28. plate[for]
29. [for = ”Vitaly”]
30. [for^ =”Sa”]
31. [for$=”ato”]
32. [for*=”obb”].

Problem 2

First follow instructions here. SelectorGadget
Modified code:

```
    library("robotstxt")

# We check that robots.txt allow to scrape from the following
   link

paths_allowed(
  path = "/en/real-estate/rent/city-basel",
  domain = "https://www.immoscout24.ch/"
)

# We can check robots.txt either by this function or visit the
   site directly.
# In this case the function get_robotstxt does not provide the
   correct file.

get_robotstxt(domain = "https://www.immoscout24.ch/")

# Download xml2 package to be able to use read_html function.

library("xml2")

# Extract html file

real_estate <- read_html(
  "https://www.immoscout24.ch/en/real-estate/rent/city-basel"
)

# To scrape data we need to use another two packages:

library("rvest")
# For using pipe function
library("magrittr")
# Scraping the data
flats <- real_estate %>%
  html_nodes(".HgListingCard_info_RKrwz") %>%
  html_text()
```

```
# Printing the obtained text
flats
# Clean data from irrelevante information
flats_df <- data.frame(
  rooms = gsub(pattern = " room.*", "", flats) %>%
    as.numeric(),
# Be careful with "-": do not mix up dash and hyphen! In this
   listing dash should be!
  price = gsub(".*, CHF |.-.*", "", flats) %>%
    gsub(pattern = ",", replacement = "") %>%
    as.numeric()
)
# Print the resulted dataset
flats_df
```

Alternative Workflow using CSS Selectors

Instead of using the *SelectorGadget* tool, you can directly use CSS selectors found in your browser's developer tools. Here's an alternative solution in R using `rvest`:

```
    # 2. Alternative solution using CSS selectors

# To scrape data we use rvest package
library(rvest)
# We read the html file
real_estate <- read_html("https://www.immoscout24.ch/en/real-
   estate/rent/city-basel")
# We extract prices from the nodes "span"
prices <- real_estate %>% html_nodes("span + span") %>% html_text
   ()
# and print the result
prices
# remove the first and two last elements
prices <- prices[-c(1,length(prices)-1,length(prices))]
prices
# Now we extract data regarding the number of rooms and the space
tmp_m2_rooms <- real_estate %>% html_nodes("strong") %>% html_
   text()
# and print the result
tmp_m2_rooms
# m2 contains information regarding the square of an appartment
m2 <- tmp_m2_rooms[seq(2, length(m2_rooms), 2)]
m2
# rooms corresponds to the number of rooms
rooms <- tmp_m2_rooms[seq(1, length(m2_rooms), 2)]
rooms
# making a nice dataset
flats_df <- data.frame(
  rooms = gsub(pattern = "\\s*rooms", "", rooms) %>%
    as.numeric(),
  meter_square = gsub("m  ","",m2) %>%
```

```
    as.numeric(),
  price = gsub("\\s*CHF\\s*", "", prices) %>%
    gsub(pattern = "\\W", replacement = "") %>%
    as.numeric()
)
#printing the resuling dataset
flats_df
```

Problem 3: Web Scraping with `RSelenium` or `chromote`

Using `chromote`:

```
    # 3. Repeat exercise 2. using `RSelenium` or `chromote`.
library(chromote)
b <- ChromoteSession$new()
b$Page$navigate("https://www.immoscout24.ch/en/real-estate/rent/
   city-basel")
tmp_m2_rooms <- b$Runtime$evaluate("document.querySelector('html
   ').outerHTML")$result$value %>%
  read_html() %>%
  html_nodes("strong") %>%
  html_text()
prices <- b$Runtime$evaluate("document.querySelector('html').
   outerHTML")$result$value %>%
  read_html() %>%
  html_nodes("span + span") %>%
  html_text()
# then as in 2.
b$close()
prices
# Note you can use b$screenshot("browser.png") to take a
   screenshot of the browser window.
# Result might differ from the one obtained with rvest because
   the website might have changed or display differently in
   chromote.
```

Problem 4: Extracting World Bank Data using Regular Expressions

```
    # 1. Extract the dataset from the table in https://data.
       worldbank.org/indicator/SP.ADO.TFRT


# The package "Chromote" is used to deal with dynamically
   changing sites
library(chromote)
# To work with html files we should use tidyverse library (or
   rvest)
# In addition tidyverse has a built-in pipe operation
library(tidyverse)
```

```r
# Initialization of the website, where to scrape
url <- "https://data.worldbank.org/indicator/SP.ADO.TFRT"

# We simulate a Browser to extract raw_data

b <- ChromoteSession$new()

b$Page$navigate(url)
raw_data <- b$Runtime$evaluate("document.querySelector('html').
   outerHTML")$result$value %>%
  read_html() %>%
  html_nodes(".item") %>%
  html_text()
b$close()
# We print the collected raw_data
raw_data


# 2. We modify the view of the dataset to make nicer.
# a. Countrycodes


# We will convert names of countries into their short codes using
# the countryside package
library(countrycode)
# First of all we get rid of any numbers using "[[:digit:]]*"
   regular expression
country <- gsub("[[:digit:]]*", "", raw_data)
# Convert country names to their codes
country_iso <- countrycode(country, origin = 'country.name',
   destination = 'iso3c')
# We print the resulting list of codes
country_iso
# However, country_iso contains NA's. Next we check order numbers
    of non-NA elements
ind <- which(!is.na(country_iso)) # remove missing iso
# and print the indeces of such counrties
ind


#b. Further modifications with raw_data

# we replace spaces with "a"
raw_data2 <- gsub("\\s","a",raw_data)
# and print the result
raw_data2
# we replace any non-word charcter with "a"
raw_data2 <- gsub("\\W","a",raw_data2)
# and print the result
raw_data2
```

```r
# we remove all letters and first 4 digital naumbers
data <- gsub("[[:alpha:]]*\\d{4}", "", raw_data2)
# and print the result
data
# and convert elements to number
data <- as.numeric(data)
# We print the result. Some of elements are NA
data


# c. We make a nice dataset

# Using tibble library we create a dataset of a type tibble
library(tibble)
fert_rate <- tibble(ISO = country_iso[ind], value = data[ind])
# and print the resulting datset
fert_rate
```