Automating web scraping with R

Alex Sanchez and Francesc Carmona
Genetics Microbiology and Statistics Department
Universitat de Barcelona

October 2022

Outline

1) Introduction

2) User defined functions

3) Changing the execution flow

4) References and Resources

Introduction

Introduction

- We have introduced R as a *a language (a tool), to manage and analyze data*.
- It is also a *programming language*
    - It is simple and versatile
    - The user can create new functions that adapt to their needs
    - It is widely used (2nd most widely used in Data Science)
    - Users provide the community with a high variety of solutions ("packages")
    - As a programming language it is not, however, very efficient

Example 1: Why we need programming

- It is very common that one has to do repetitive tasks on a the same type of datasets, e.g.
    - Data produced periodically o,
    - Data from multiple sources but have the same structure.
- For example, given a file with information about the cities of a given province, we are required to produce a simpler version:
    - With less columns
    - Without spaces or accents in the column names
    - With the appropriate data types for each column

Example 1: Transformations

```r
library(dplyr); library(janitor)
```

```
## 
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
## 
##     filter, lag

## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union

## 
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
## 
##     chisq.test, fisher.test
```

```r
MunicipisBARC <- read.csv("datasets/MunicipisBARC.csv", quote='"')
MunicipisBreu <- MunicipisBARC %>% select (2,9,10,27,28) %>%
```

Example 1: Before and after transform

```
summary(MunicipisBARC[,c(2,9,10,27,28)])
```

```
summary(MunicipisBreu)
```

```
##     Codi.INE    Codi.de.comarca Nom.de.la.comarca  N##bre.dcodbiIaets  codi_de_comarca       nom_de_la_comar
##  Min.   :8001   Min.   : 3.00   Length:311         M##. Min.   :8700   24     : 47      Osona          : 47
##  1st Qu.:8078   1st Qu.: 7.00   Class :character   1## Q1st Qu.:7808   41     : 39      Vallès Oriental: 39
##  Median :8156   Median :17.00   Mode  :character   M##iaMedian328950   6      : 33      Anoia          : 33
##  Mean   :8166   Mean   :19.73                      M##n Mean 1837566   7      : 30      Bages          : 30
##  3rd Qu.:8236   3rd Qu.:24.00                      3## Q3rd Q1108036   11     : 30      Baix Llobregat : 30
##  Max.   :8905   Max.   :42.00                      M##. Max163678905   14     : 30      Berguedà       : 30
##     Extensió                                        ##                 (Other):102      (Other)        :102
##  Min.   :  0.40                                     ##     extensio
##  1st Qu.: 11.01                                     ##  Min.   :  0.40
##  Median : 21.17                                     ##  1st Qu.: 11.01
##  Mean   : 24.90                                     ##  Median : 21.17
##  3rd Qu.: 34.12                                     ##  Mean   : 24.90
##  Max.   :102.90                                     ##  3rd Qu.: 34.12
                                                        ##  Max.   :102.90
                                                        ##
```

`codi_de_comarca` and `nom_de_la_comarca` are now factors.

Repeating the transformation

- How should we proceed if these changes had to be applied repeteadly to many distinct files (with same structure)

- One solution may consists of:
  - providing some way to encapsulate all steps needed to do the transformation
  - in such a way that they can be easily applied to a file everytime they are required.

- This is an example of a "function" (a type of sub-program in R) that we would use to automate this preprocessing.

User defined functions

Functions are named expressions

- A function is a set of statements organized together to perform a specific task.
- R has a large number of in-built functions.
- Users can create their own functions, for those situations where they wish to apply the same set of instructions more than once.

```
function_name ← function(arg_1, arg_2, ...) {
    sentence 1
    ...
    sentence n
    return(result)
}
```

Go here for more information on functions.

A preprocessing function

We can encapsulate preprocessing in a function:

```
preprocessa ← function(nomArxiuMunicipi){
  require(dplyr); require(janitor)
  unMunicipi ← read.csv(nomArxiuMunicipi)
  unMunicipiBreu ← unMunicipi %>% select (2,9,10,27,28) %>%
  janitor::clean_names() %>%
  mutate (across(c("codi_de_comarca", "nom_de_la_comarca"), as.factor))
  return(unMunicipiBreu)
}
```

And use it whenever is required. Assuming we had the files "MunicipisBARC.csv", "MunicipisGIRO.csv" and "MunicipisLLEI.csv" available we would do:

```
BCNBreu ← preprocessa("MunicipisBARC.csv")
GiroBreu ← preprocessa("MunicipisGIRO.csv")
LleidaBreu ← preprocessa("MunicipisLLEI.csv")
```

Scraping a recipes site

- Imagine we are scraping a recipes site
- The code below extracts (without cleaning it) a recipe for brownies.

```r
library(rvest)
brownies <- read_html("https://www.allrecipes.com/recipe/25080/mmmmm-brownies/")
ingredients <- brownies %>%
  html_elements( "#mntl-structured-ingredients_1-0") %>%
  html_text2() %>% stringr::str_split("\\n\\n")

xpath4Directions <- '//*[(@id = "recipe__steps_1-0")]'

directions <-  brownies %>%
  html_elements( xpath=xpath4Directions) %>%
  html_text2() %>% stringr::str_split("\\n\\n")
```

- Selectors were obtained inspecting the page source code with SelectorGadget or Google Developper tools.

The scraped recipe

```
show(ingredients)
```

```
## [[1]]
##  [1] "Ingredients"                        "½ cup white sugar"
##  [3] "2 tablespoons butter"               "2 tablespoons water"
##  [5] "1 ½ cups semisweet chocolate chips" "2 large eggs, beaten"
##  [7] "½ teaspoon vanilla extract"         "⅔ cup all-purpose flour"
##  [9] "½ teaspoon salt"                    "¼ teaspoon baking soda"
```

```
show(directions)
```

```
## [[1]]
## [1] "Directions"
## [2] "Preheat the oven to 325 degrees F (165 degrees C
## [3] "Combine sugar, butter, and water in a medium sau
## [4] "Bake in the preheated oven until top is dry and
## [5] "dotdash meredith food studios"
```

A function to scrape recipes

- Proceed similarly as before:
  - abstracting the process and
  - turning what is different every time (URL) into arguments

```r
scrape_recipes ← function(URL) {
  aDessert ← read_html(URL)
  ingredients ← aDessert %>%
    html_elements( "#mntl-structured-ingredients_1-0") %>%
    html_text2() %>% stringr::str_split("\\n\\n")

  xpath4Directions ← '//*[(@id = "recipe__steps_1-0")]'

  directions ←  aDessert %>%
    html_elements( xpath=xpath4Directions) %>%
    html_text2()%>% stringr::str_split("\\n\\n")
  return(list(Ingredientes=ingredients, Receta=directions))
}
```

The scraped recipe (2)

```r
library(rvest)
recipeURL ← "https://www.allrecipes.com/recipe/25080/mmmmm-brownies/"
brownies ← scrape_recipes (recipeURL)
```

```r
show(brownies[["Ingredientes"]])
```

```
## [[1]]
##  [1] "Ingredients"                        "½ cup white sugar"
##  [3] "2 tablespoons butter"               "2 tablespoons water"
##  [5] "1 ½ cups semisweet chocolate chips" "2 large eggs, beaten"
##  [7] "½ teaspoon vanilla extract"         "⅔ cup all-purpose flour"
##  [9] "½ teaspoon salt"                    "¼ teaspoon baking soda"
```

```r
show(show(brownies[["Receta"]]))
```

```
## [[1]]
## [1] "Directions"
## [2] "Preheat the oven to 325 degrees F (165 degrees C
## [3] "Combine sugar, butter, and water in a medium sau
## [4] "Bake in the preheated oven until top is dry and
## [5] "dotdash meredith food studios"
##
## NULL
```

Changing the flow

Changing the flow of execution

- R, as most ordinary programming languages, is executed lineally, that is from the first to last line.

- Sometimes this needs to be changed.

  - Taking alternative flows according to certain conditions
  - Repeating some instructions while certain condition holds, or a fixed number of times,...

- This can be acomplished using *Flow Control Structures*

Loop controlled by a counter: `for`

- Loops are used in programming to repeat a specific block of code made by one or more instructions.
- Syntax of `for` loops:

```
for (val in sequence)
{
statement
}
```

- `sequence` is a vector and `val` takes on *each of its values* during the loop.
- In each iteration, `statement` is evaluated.

Example of `for` loop

- A `for` loop can be used to preprocess a list of selected files
- Assume we have the list of four files to be processed, and ***we know they have the same structure***.
- To process them all in one step do (not run):

```
llistaMunicipis ← c("MunicipisBAR.csv", "MunicipisGIR.csv",
                    "MunicipisLLE.csv", "MunicipisTAR.csv" )
for (nomFitxerMunicipis in llistaMunicipis) {
  municipisProvincia ← preprocessa("nomFitxerMunicipis")
  summary(municipisProvincia)
}
```

An alternative way to run the loop:

```
for (i in 1:length(llistaMunicipis) {
  municipisProvincia ← preprocessa(llistaMunicipis[i])
```

Exercise

- Create a `for` loop that reads all filenames in your datasets directory (or the directory you decide) and prints the name of the file and the column names in the screen.

Scraping multiple recipes

- Imagine we want to process not one but many desserts' recipes from the web "https://www.allrecipes.com/".
- This can be done using a simple for loop:

```r
recipe_urls <- c("https://www.allrecipes.com/recipe/25080/mmmmm-brownies/",
                 "https://www.allrecipes.com/recipe/27188/crepes/",
                 "https://www.allrecipes.com/recipe/22180/waffles-i/")
listOfRecipes <- list()
for (i in 1:length(recipe_urls)) {
  listOfRecipes[i] <- scrape_recipes(recipe_urls[i])
}
```

- Notice that the resulting scraped recipes are now stored in a *list* that will be eventually processed by the user.

Exercise

- Write a simple function to print one recipes obtained using the function `scrape_recipes`
- Use this function to print all the recipes collected in the list "listOfRecipes" -