7316 - INTRODUCTION TO R

Module 3: Correction of the Exercise

Teacher: Mickaël Buffart (mickael.buffart@hhs.se)

TABLE OF CONTENTS

1 Scraping data from a website 1
2 String manipulation 2
3 More string manipulations 3
4 Writing a function that can be re-used 4
5 Automating this process 4

In this assignment, you will create your own dataset by scraping a website, manipulating strings, merging the dataset with another dataset and finally write your own routine to analyze the data.

1 SCRAPING DATA FROM A WEBSITE

Go on the website https://www.beeradvocate.com/beer/top-rated/. BeerAdvocate is a website that crowd-sources beer reviews and ratings.

1. Save the URL as a string.

```
url <- "https://www.beeradvocate.com/beer/top-rated/"
```

2. using `read_html()` and `html_table()` from the `rvest` package to extract the table with the reviews (use the header option)

```
data <- rvest::html_table(rvest::read_html(url), header = TRUE)
```

3. view the structure of the table.

```
str(data)
```

4. Clean up the table:

- Extract the data.frame from the list

```
df <- data[[1]]
```

- Drop variables that contain no information ("You")

```
# Remove the "You" variable that contains no data
df <- df[, !names(df) %in% c("You")]
```

- Change the class of the variables to their appropriate format. *Hint:* You will need to remove the commas from the numbers > 1000 in the Ratings variable to get

them into a format that can be converted into a numerical. You can use `str_replace()` or `gsub()`

```
# Change the number of ratings to a numerical variable. it is currently a char
acter. First remove the commas.
df$Ratings <- gsub(",", "", df$Ratings)

# Convert to integer
df$Ratings <- as.integer(df$Ratings)
```

- Rename the variables with appropriate names.

```
# Rename the columns
names(df) <- c("rank", "name", "n_ratings", "avg_rating")
```

You should in the end have a table with 4 variables. We will deal with the information in the string in the next section.

2 STRING MANIPULATION

We now have a table with 4 variables but the "Name" variable contains information that we would like to have separately. But first we must take care of missing values. Some beers do not have information about he alcohol content. We can identify them by looking for the percentage sign in the string variable.

5. Find all observations that contain a % sign. Filter the data according to this criterion.

```
# Filter observations with % sign
df_tmp <- df[grepl("%", df$name, fixed = TRUE),]
```

6. Split the alcohol content from the string, trim it and store it into a separate, numeric variable. You can use `strsplit()` or `gsub()` to split the string along `"|"`. Use a double escape symbol `"\\|"`, otherwise R cannot interpret the character you want to split by.

```
# Get the data containing alcohol info
df$alcohol <- df$name

# Then, let's use regular expressions
# The alcohol level is between | and % in the string
df$alcohol <- gsub(".*\\| ([^%]+).*", "\\1", df$alcohol)
```

The regular expression above means:

- `.*`: anything
- `\\|`: a bar `|` followed by a space
- `([^%]+)`: a series a characters to memorize until the first `%`; `^%` means the first `%`
- `.*`: anything
- `\\1`: matches with the characters memorized above

But this is not yet fully cleaned because two observations contains no `%` information, as identified in Q5. Those should be `NA`. When transforming the string to numeric, they will become `NA` automatically, because they contains characters:

```
# We will get a warning: the two strings that are not the percentage of alcoho
l will be coerced into NA
df$alcohol <- as.numeric(df$alcohol)
```

```
# Double checking that we only have 2 missing values:
sum(is.na(df$alcohol))
## [1] 2
```

7. Unlist the resulting list.

Because we used `gsub()`, we do not have a list. To have a list, we should have used `strsplit()`, as also suggested in Q6:

```
tmp <- strsplit(df$name, split = "\\|")

# Before unlisting, let's make sure all vectors are the same length, otherwise
#   the list will not match our dataframe
#   We should have only vectors of length 2, because we tried to split the
#   string into 2, with \\|

for (i in 1:length(tmp)) {
  if (length(tmp[[i]]) == 1) {
    # if length == 1, we fill the second element with NA
    tmp[[i]] <- c(tmp[[i]], NA)
  } else if (length(tmp[[i]]) > 2) {
    # if length > 2, we have a problem: too long
    print(pastr("TOO LONG:", i))
  } # else: nothing to do
}

# OK: no warning, we can unlist
tmp <- unlist(tmp)
```

8. save the length of the unlisted vector as a variable `n`.

```
n <- length(tmp)
```

9. create a n/2 x 2 matrix and fill it with the unlisted data.

```
tmp <- matrix(tmp,
              nrow  = n / 2,
              ncol  = 2,
              byrow = TRUE)
```

Now you should have a table with two variables that can be further manipulated before we put it back into the original dataset.

3 More string manipulations

10. Rename the columns “Name” and “Percentage”

```
tmp <- as.data.frame(tmp)
names(tmp) <- c("name", "percentage")
```

11. Trim the space from the alcohol percentage

```
tmp$percentage <- stringr::str_trim(tmp$percentage)
```

12. Remove the percentage sign using `str_replace()` or `gsub()` and a double escape (as with the `"|"` symbol)

```
tmp$percentage <- gsub("%", "", tmp$percentage, fixed = TRUE)
```

13. convert the Percentage value into a numerical

```
tmp$percentage <- as.numeric(tmp$percentage)
```

14. Replace the old Name variable with the new one and add the Percentage column to the original table.

```
# No need to use merge, as we did not reorder the rows.
# Warning: df contains a "name" column already; we rename to avoid confusion
names(df)[names(df) == "name"] <- "name_old"

df <- cbind(df, tmp)
```

At this point you should have a nice dataset with 5 variables.

4 Writing a function that can be re-used.

Since we want to automate this process, it would be convenient to have a function that performs the step of trimming and turning the percentage into a numerical .

15. Write a function that takes a character vector as an input.
16. Inside the function, trim the left side, remove the percentage sign and turn it into a numeric

```
perc_to_num <- function(x) {
  x <- as.character(x)
  stopifnot(is.character(x))

  x <- stringr::str_trim(x, side = "left")
  x <- gsub("%", "", x, fixed = TRUE)
  x <- as.numeric(x)

  return(x)
}
```

5 Automating this process

If you select different countries for the BeerAdvocate ratings, you will notice that the the URL stays the same but a country code is added.

We now want to extract the ratings for Australia, Austria, Germany, Sweden, the UK and the United States.

17. create a vector with the country codes “de”, “se” and “gb”.

```
country_codes <- c("de", "se", "gb")
```

18. Write a loop that loops through this vector.

```
for (i in 1:length(country_codes)) {
  # 19. In each iteration, the loop should add the country code to the url
  url <- paste0("https://www.beeradvocate.com/beer/top-rated/",
                country_codes[i], "/")

  # and extract the html table, as you did above.
  tmp <- rvest::html_table(rvest::read_html(url), header = TRUE)
  tmp <- tmp[[1]]

  # 20. Rename the columns as "rank", "name", "ratings", "avg", "yours"
  names(tmp) <- c("rank", "name", "ratings", "avg", "yours")

  # 21. If Ratings is read as an integer, convert it into a numeric
  #     (remove the comma, as above). *Hint:* use *is_character()*
  #     in an if-statement.
```

```
  if (is.character(tmp$ratings)) {
      tmp$ratings <- gsub(",", "", tmp$ratings)
      tmp$ratings <- as.integer(tmp$ratings)
  }

  # 22. In addition, you should add the country code as
  #     a variable "Country" to the table.
  tmp$country <- country_codes[i]

  # 23. If you are in the first iteration, save this html table as a data.fram
e,
  #     call it data_combined For each following iteration,
  #     simply append the new data.
  if (i == 1) {
    df <- tmp
  } else {
    df <- rbind(df, tmp)
  }
}
```

24. Remove the "Yours" variable

```
df$yours <- NULL
```

25. Filter out all observations where the string variable does not contain a percentage sign.

```
df <- df[grepl("%", df$name, fixed = TRUE),]
```

26. Perform the same procedure as above to get the alcohol percentage out of the Name string in data_combined

```
df$alcohol <- df$name
df$alcohol <- gsub(".*\\| ([^%]+).*", "\\1", df$alcohol)
df$alcohol <- as.numeric(df$alcohol)

# Then, we remove the alcohol level from names:
#   regex: we replace with nothing anything after |
df$name <- gsub("\\|.*$", "", df$name)
```