Problem Set #1

Emorie D Beck

10/9/23

Overview

Welcome to Data Cleaning and Management using R! This problem set is intended to give you some practice becoming familiar with using R. In this problem set, I'm asking you to: create an R project; render to pdf; load and investigate an R data frame that is stored on the web; and apply some basic functions to atomic vectors.

- Note: Change the values of the YAML header above to your name and the date.

```r
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.4.2     v purrr   1.0.2
v tibble  3.2.1     v dplyr   1.1.3
v tidyr   1.2.1     v stringr 1.5.0
v readr   2.1.2     v forcats 0.5.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

Question 1: Creating an R project

Create an R project

- Create a folder where you want to save files associated with problem set 1. Let's call that folder "problemset1", but you can name it whatever you want.
    - For instance, it could be psc290-fq23 » problem-sets » problemset1.
- In RStudio, click on "File" » "New Project" » "Existing Directory" » "Browse".

- Browse to find and select your problem set 1 folder.
- Click on "Create Project".
  - An R project file has the extension ".Rproj".
  - The name of the file should be "problemset1.Rproj", or whatever you named the folder.

Save this problemset1.qmd file anywhere in the folder named problemset1.

- Use this naming convention "lastname-firstname-ps#" for your .qmd files (e.g. beck_emorie_ps1.qmd).
  - If you want, you can change the name of this file to include your first and last name.
- Run the `getwd()` function and the `list.files()` function in the code chunk below.
- What is the output? Why?

```
getwd() # list current directory
list.files() # list files in current directory
# mine may be different from yours because I have the data
```

ANSWER: `getwd()` lists the current directory path, which is important for relative file paths. `list.files()` lists all the files in current working directory (listed in `getwd()`).

Question 2: Render to pdf

- At the top of this .qmd file, type in your first and last name in the appropriate place in the YAML header (e.g. "Hadley Wickham").
- In the date field of the YAML header, insert the date within quotations (any date format is fine).
- Now click the "Render" button near the top of your RStudio window (icon with blue yarn ball).
  - Alternatively you can use the shortcut: **Cmd/Ctrl + Shift + k**.
  - *Note*: One goal of this assignment is to make sure you are able to render to a PDF without running into errors.

Question 3: Load .Rdata directly with url and then investigate the data frame

1. This question asks you to load a dataframe by specifying the `url()` function within the `load()` function.

- Url link for data frame: https://github.com/emoriebeck/psc290-data-FQ23/raw/main/05-assignments/01-ps1/pwe-ps1-small.RData

  - Hint: to load .Rdata use the `load()` and `url()` functions because you are using a link. follow this approach: `load(url("url_link"))`.
    - Note: the `url_link` is put within quotes

Load the dataframe within this code chunk below.

```
#?load
load(url("https://github.com/emoriebeck/psc290-data-FQ23/raw/main/05-assignments/01-ps1/pw
```

2. Print the data frame `df_pwe` by typing its name.

```
df_pwe
```

```
# A tibble: 100 x 45
   country introelapse testelapse surveyelapse TIPI1 TIPI2 TIPI3 TIPI4 TIPI5
   <chr>         <dbl>      <dbl>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 US             1118        103          111     2     6     1     7     7
 2 GB                8        116          128     3     2     6     1     5
 3 US                9        142          217     5     6     7     5     5
 4 US                2         79           90     2     5     1     2     3
 5 VN              545        210          276     3     1     6     5     5
 6 IN             4397        625           79     4     5     2     6     7
 7 NO                2        181          136     3     5     3     6     7
 8 NONE              4        208          198     5     6     2     5     7
 9 PL               13        191          266     6     7     4     3     5
10 IN               37        394          380     7     1     7     1     1
# i 90 more rows
# i 36 more variables: TIPI6 <dbl>, TIPI7 <dbl>, TIPI8 <dbl>, TIPI9 <dbl>,
#   TIPI10 <dbl>, VCL1 <dbl>, VCL2 <dbl>, VCL3 <dbl>, VCL4 <dbl>, VCL5 <dbl>,
#   VCL6 <dbl>, VCL7 <dbl>, VCL8 <dbl>, VCL9 <dbl>, VCL10 <dbl>, VCL11 <dbl>,
#   VCL12 <dbl>, VCL13 <dbl>, VCL14 <dbl>, VCL15 <dbl>, VCL16 <dbl>,
#   education <dbl>, urban <dbl>, gender <dbl>, engnat <dbl>, age <dbl>,
#   screenw <dbl>, screenh <dbl>, hand <dbl>, religion <dbl>, ...
```

3. Use the `typeof()` function to investigate the type of data frame `df_pwe`.

```
typeof(df_pwe) # tibble
```

```
[1] "list"
```

4. Apply the `length()` function to the data frame `df_pwe`. What does this output mean in your own words?

```r
length(df_pwe)
```

```
[1] 45
```

ANSWER: `length()` captures the number of columns in the data frame. It's better practice to use `ncol()` and `nrow()`, which have more interpretable behavior.

5. Use the `str()` function to investigate the structure of the data frame `df_pwe`.

```r
str(df_pwe)
```

```
tibble [100 x 45] (S3: tbl_df/tbl/data.frame)
 $ country      : chr [1:100] "US" "GB" "US" "US" ...
 $ introelapse : num [1:100] 1118 8 9 2 545 ...
 $ testelapse  : num [1:100] 103 116 142 79 210 625 181 208 191 394 ...
 $ surveyelapse: num [1:100] 111 128 217 90 276 79 136 198 266 380 ...
 $ TIPI1       : num [1:100] 2 3 5 2 3 4 3 5 6 7 ...
 $ TIPI2       : num [1:100] 6 2 6 5 1 5 5 6 7 1 ...
 $ TIPI3       : num [1:100] 1 6 7 1 6 2 3 2 4 7 ...
 $ TIPI4       : num [1:100] 7 1 5 2 5 6 6 5 3 1 ...
 $ TIPI5       : num [1:100] 7 5 5 3 5 7 7 7 5 1 ...
 $ TIPI6       : num [1:100] 7 6 5 5 6 4 5 3 1 6 ...
 $ TIPI7       : num [1:100] 7 5 1 1 7 5 5 7 5 7 ...
 $ TIPI8       : num [1:100] 7 2 1 7 1 6 5 1 6 1 ...
 $ TIPI9       : num [1:100] 1 6 6 5 6 3 5 6 6 7 ...
 $ TIPI10      : num [1:100] 1 5 7 3 3 2 1 2 4 4 ...
 $ VCL1        : num [1:100] 1 1 1 1 1 1 1 0 1 0 ...
 $ VCL2        : num [1:100] 1 1 1 1 1 0 1 1 0 0 ...
 $ VCL3        : num [1:100] 0 1 1 0 0 0 0 0 0 0 ...
 $ VCL4        : num [1:100] 1 1 1 0 1 1 1 1 1 0 ...
 $ VCL5        : num [1:100] 1 1 1 1 1 1 1 0 1 1 ...
 $ VCL6        : num [1:100] 0 0 0 0 0 0 0 1 0 0 ...
 $ VCL7        : num [1:100] 0 1 0 0 1 0 0 0 0 0 ...
 $ VCL8        : num [1:100] 0 1 0 0 0 1 1 1 1 0 ...
 $ VCL9        : num [1:100] 0 0 0 0 0 0 0 0 0 0 ...
 $ VCL10       : num [1:100] 1 1 1 0 1 1 1 1 1 1 ...
 $ VCL11       : num [1:100] 0 1 0 1 1 0 1 1 1 0 ...
 $ VCL12       : num [1:100] 0 0 0 0 0 0 0 0 0 0 ...
 $ VCL13       : num [1:100] 1 1 1 1 0 1 1 0 0 0 ...
 $ VCL14       : num [1:100] 1 1 1 0 1 1 1 1 0 0 ...
 $ VCL15       : num [1:100] 1 1 1 1 1 1 1 1 1 0 ...
 $ VCL16       : num [1:100] 1 1 1 1 1 1 1 1 1 1 ...
 $ education   : num [1:100] 2 3 2 2 2 3 1 2 2 4 ...
 $ urban       : num [1:100] 1 1 2 2 3 3 2 3 3 1 ...
 $ gender      : num [1:100] 2 2 2 1 2 1 2 2 2 1 ...
 $ engnat      : num [1:100] 1 1 1 1 2 2 2 2 2 2 ...
 $ age         : num [1:100] 19 64 18 25 19 22 32 21 19 26 ...
 $ screenw     : num [1:100] 1093 768 1366 1536 1152 ...
 $ screenh     : num [1:100] 615 1024 768 864 720 ...
 $ hand        : num [1:100] 1 1 1 1 1 1 1 1 1 1 ...
 $ religion    : num [1:100] 1 2 2 1 3 8 1 2 2 8 ...
 $ orientation : num [1:100] 1 1 1 1 1 1 1 1 5 2 ...
 $ race        : num [1:100] 16 16 16 13 11 11 16 16 16 11 ...
 $ voted       : num [1:100] 2 2 1 1 2 1 2 1 1 1 ...
 $ married     : num [1:100] 1 3 1 1 1 1 1 1 1 2 ...
 $ familysize  : num [1:100] 2 2 3 3 2 2 2 2 1 3 ...
 $ major       : chr [1:100] NA "Psychology" "Chemistry" NA ...
```

6. Use the `names` function to list the names of the elements (variables) within `df_pwe`.

```
names(df_pwe)
```

```
 [1] "country"      "introelapse"  "testelapse"   "surveyelapse" "TIPI1"
 [6] "TIPI2"        "TIPI3"        "TIPI4"        "TIPI5"        "TIPI6"
[11] "TIPI7"        "TIPI8"        "TIPI9"        "TIPI10"       "VCL1"
[16] "VCL2"         "VCL3"         "VCL4"         "VCL5"         "VCL6"
[21] "VCL7"         "VCL8"         "VCL9"         "VCL10"        "VCL11"
[26] "VCL12"        "VCL13"        "VCL14"        "VCL15"        "VCL16"
[31] "education"    "urban"        "gender"       "engnat"       "age"
[36] "screenw"      "screenh"      "hand"         "religion"     "orientation"
[41] "race"         "voted"        "married"      "familysize"   "major"
```

7. Wrap your answer above — `names(data_frame_name)` — within the `typeof()` function. Do the same for the `length()` function, and the `str()` function as well. Interpret what the output means in your own words.

```
typeof(names(df_pwe))
```

```
[1] "character"
```

```
length(names(df_pwe))
```

```
[1] 45
```

ANSWER: `names()` provides a character vector listing the column names of `df_pwe`. Thus, `typeof(names())` will always tell you it's a character vector, and `length(names())` will always provide an integer with the number of columns in the data frame.

Question 4: Applying basic functions to atomic vectors

1. Create an atomic vector object named age with the following values: 3, 6, 41, 43.

```
age <- c(3, 6, 41, 43)
```

2. Apply the `typeof()`, `length()`, and `str()` functions to the object `age`.

```
typeof(age) # double
```

```
[1] "double"
```

```
length(age) # 4
```

```
[1] 4
```

```
str(age)    # num [1:4] 3 6 41 43
```

```
 num [1:4] 3 6 41 43
```

3. Apply the `sum()` function to `age`.

```
sum(age)
```

```
[1] 93
```

4. Apply the `sum()` function to `age` but this time include the argument `na.rm = FALSE`.

```
sum(age, na.rm = F)
```

```
[1] 93
```

5. In general, what is a function "argument name" and what is an "argument value"? What does the argument `na.rm` do?

ANSWER: Argument names are generally the names of function-specific arguments. Argument values are the input to those function-specific arguments. ?function (e.g., ?sum) will help you understand what arguments a function takes and what accurate argument values will be.

6. Create a new object `age2` with the following values: 3, 6, 41, 43, NA. Now calculate the sum of `age2` using the argument `na.rm = FALSE` and then calculate the sum using the argument `na.rm = TRUE`. Explain why the outputs of these two `sum()` functions differ.

```
age2 <- c(3, 6, 41, 43, NA)
sum(age2, na.rm = F)
```

```
[1] NA
```

```
sum(age2, na.rm = T)
```

```
[1] 93
```

ANSWER: Now that there is a missing (`NA`) value, any sums, means, etc. will return as `NA` unless `na.rm = T`. When `na.rm = T`, the missing values are removed (hence the .rm).

7. Create a vector `tf` using the following code: `tf <- c(TRUE,FALSE,TRUE,FALSE,TRUE)`. Next apply the `typeof()`, `length()`, and `str()` functions to the object `tf`. Based on this output, briefly describe the object `tf` in your own words (one sentence is fine).

```
tf <- c(TRUE,FALSE,TRUE,FALSE,TRUE)
typeof(tf)
```

```
[1] "logical"
```

```
length(tf)
```

```
[1] 5
```

```
str(tf)
```

```
 logi [1:5] TRUE FALSE TRUE FALSE TRUE
```

ANSWER: tf is a logical vector containing `TRUE`'s and `FALSE`s.

8. Apply the `sum()` function to the object, using the option to remove `NA` values prior to calculation. What numeric value do mathematical calculations in `R` assign to `TRUE` values and what do they assign to `FALSE` values?

```
sum(tf, na.rm = T)
```

```
[1] 3
```

ANSWER: `TRUE`s are assigned 1's and `FALSE`s are 0's. So the result here indicates that there are 3 trues in the vector.

9. This is the syntax of the `mean()` function that includes both argument names and the default values for arguments: `mean(x, trim = 0, na.rm = FALSE)`.

When using a function, `R` requires you to type the values you assign to each argument, but typing in the argument names is usually optional. Even though it takes a bit more time, I usually like typing in both argument names and argument values, because it forces me to be more conscious about what value I am assigning to which argument, especially when a function is new to me.

Use the `mean()` function to calculate the mean of object `tf` (removing `NA` values prior to calculation). In your function call, include both the argument name and the argument value for each argument (argument value for the `trim` argument can be `0`). Then run the same function, but without typing any argument names.

```
mean(tf, trim = 0, na.rm = T)
```

```
[1] 0.6
```

```
mean(tf, 0, T)
```

```
[1] 0.6
```

Render to pdf and submit problem set

Render to pdf by clicking the "Render" button near the top of your RStudio window (icon with blue yarn ball) or drop down and select "Knit to PDF".

- Go to the Canvas and under the "Assignments", submit to the Problem Set 1 Assignment.
- Submit both .qmd and .pdf files.
- Use this naming convention "lastname_firstname_ps#" for your .qmd and pdf files (e.g. beck-emorie-ps1.rmd & beck-emorie-ps1.pdf).