Rverse

PhD student Giulia Nicole Baldrighi

2024-03-20

Handling projects for reproducibility

R projects

Setting a new project

- New folder in an accessible place of laptop
- copy my repository (RVerse) to have the necessary material
- - go to the link on GitHub
- - download the material within the folder you will use for working
- - create and save a project (snapshot) within your folder

File –> New project …

File Type

- **.Rproj** This is your project file in RStudio. This automatically sets your working directory to this containing folder
- **.R** This is your script file. It contains your previously saved code. Like a Word Document.
- **.RHistory** R keeps track of all the commands you use in a session.

Rproject

- Valuable and reproducible workflow that will serve in the future
- Import the data (Environment -> Import Dataset)
- You can then save and leave everything as in a snapshot.

Open/close the project

- Top right button (open/close)
- It is good practice to look at your pathway sometimes `getwd()`
- When you open your project you do not need to set the working directory (Session is already set)

```
setwd("C:/Users/gnbal/Desktop/GitHub/RVerse/data")
```

Good workflow

- Save your scripts (with informative names) in the project, edit them, run them in bits or as a whole.
- Restart R frequently to make sure you've captured everything in your scripts.
- Only ever use relative paths, not absolute paths.

More information at:

(https://r4ds.had.co.nz/workflow-projects.html)

Reading the data

- **External data**: import dataset (text, csv, excel …) with `read()`
- **Internal data**: import dataset (rda) with `data()`
- Otherwise objects are included in the libraries.

Example of external:

```
data <- read.csv("C:/Users/gnbal/Desktop/GitHub/RVerse/data/Lars data.csv.gz")
```

```
head(data[1:10,1:5])
```

```
##                   Date ElapsedTime Glucose Temperature Activity
## 1 2017-10-03T09:51:00Z         240   11.37       33.23     8.06
## 2 2017-10-03T09:52:00Z         300   11.23       33.20     5.06
## 3 2017-10-03T09:53:00Z         360   10.98       32.97    23.06
## 4 2017-10-03T09:54:00Z         420   10.90       32.78    17.06
## 5 2017-10-03T09:55:00Z         480   10.92       32.68     3.06
## 6 2017-10-03T09:56:00Z         540   10.95       32.63    14.06
```

Example of internal:

```
data("mtcars")
```

```
head(data[1:10,1:5])
```

```
##                   Date ElapsedTime Glucose Temperature Activity
## 1 2017-10-03T09:51:00Z         240   11.37       33.23     8.06
## 2 2017-10-03T09:52:00Z         300   11.23       33.20     5.06
## 3 2017-10-03T09:53:00Z         360   10.98       32.97    23.06
## 4 2017-10-03T09:54:00Z         420   10.90       32.78    17.06
## 5 2017-10-03T09:55:00Z         480   10.92       32.68     3.06
## 6 2017-10-03T09:56:00Z         540   10.95       32.63    14.06
```

Reading directly from library

```
library(dplyr)
library(tidyverse)
head(diamonds)
```

```
## # A tibble: 6 × 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Tidyverse

- summarise data using: `group_by()`, `summarise()`, and `mutate()`
- reshape data between the wide and long formats: `pivot_wider()` and `pivot_longer()`
- `select()` columns and `arrange()` (sort) rows.
- other important: `filter()` and `count()`

More information at: (https://argoshare.is.ed.ac.uk/healthyr_book/exercises.html)

group_by()

```
library(dplyr)
library(tidyverse)
```

```
head( diamonds %>%
  group_by(clarity) )
```

```
## # A tibble: 6 × 10
## # Groups:   clarity [5]
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Saving the df in an object

```
data<- diamonds %>%
  group_by(clarity)
```

```
head(data)
```

```
## # A tibble: 6 × 10
## # Groups:   clarity [5]
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.23 Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5  0.31 Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

group_by() %>% summarise()

```
diamonds %>%
  group_by(clarity) %>%
  summarize(m = mean(price))
```

```
## # A tibble: 8 × 2
##   clarity     m
##   <ord>   <dbl>
## 1 I1      3924.
## 2 SI2     5063.
## 3 SI1     3996.
## 4 VS2     3925.
## 5 VS1     3839.
## 6 VVS2    3284.
## 7 VVS1    2523.
## 8 IF      2865.
```

Logical operators

To use filtering effectively, you have to know how to select the observations that you want using the comparison operators. R provides the standard suite: >, >=, <, <=, != (not equal), and == (equal).

filter()

When you run that line of code, dplyr executes the filtering operation and returns a new data frame. dplyr functions never modify their inputs, so if you want to save the result, you'll need to use the assignment operator, <-:

filter()

```
filter(variable, variable2 == 1)

jan1 <- filter(variable, month == 1, day == 1)

nov_dec <- filter(flights, month %in% c(11, 12))

filter(flights, !(arr_delay > 120 | dep_delay > 120))

filter(flights, arr_delay <= 120, dep_delay <= 120)
```

count()

```
flights %>% count(year, month, day, flight) %>% filter(n > 1)
```

select()

```
flights2 <- flights %>% select(year:day, hour, origin, dest, tailnum, carrier)
```

Working on *Lars data*

Libraries

library(tidyverse)

library(ggplot2)

library(nycflights13)

How to access variable type, number of observations and variables?

- type
- length

Answer the following questions

1. How many different mice were profiled in the experiment?
2. How many observations per mouse? Was there an equal number of observations for each mouse?
3. Is there any missing data? How much? Is there a mouse with more missing data than other mice? Or a week, a day, a ZT time with more missing data?
4. Across how many weeks was the experiment performed?
5. What is the mean, SD, and range in glucose value across weeks and days between the two different groups of mice?