Aide-mémoire

Ewen Gallic

5/27/23

Table of contents

Preface

This “ebook” serves as a quick reference guide for various programming tasks, numerical tools, and procedures. It contains code snippets and explanations related to R, LaTeX, Markdown, Quarto, and other relevant tools.

The creation of this document follows a discussion I had with Olivier L’Haridon who suggested the idea. Having been won over by the idea, I took the liberty of revisiting it. The primary motivation behind creating this project was to address a common issue I encountered in my programming and numerical work—having to search for specific procedures and codes that I don’t frequently use. I noticed that valuable time was often wasted trying to locate the exact location of these procedures within my notebooks, leading to delays and frustration.

By creating this “Aide-mémoire”, I aim to centralize and organize all the necessary information, codes, and procedures that I frequently need but might not remember off the top of my head. This way, I can quickly access and retrieve the required information whenever I encounter a similar task in my work.

I believe that sharing this guide on GitHub and making it available online may be of interest for those who have already some bacground knowledge in R, Markdown, and LaTeX and face the similar problems.

Part I

R

1 Colours Palettes

Here are some colours that I often use in R plots or in LaTeX documents. For each color, I provide the hexadecimal value and the RGB values. For some palettes, I gave names to the colors, so I provide some definitions in R and in LaTeX.

For LaTeX, the following package needs to be loaded to define our own colors:

```
\usepackage[dvipsnames]{xcolor}
```

Cool stuff for colour picking: https://davidmathlogic.com/colorblind/

1.1 Colours that are ok for color blindness and for printers

1.1.1 Palette 1

```
#a6cee3
```

```
rgb(166, 206, 227)
```

```
#1f78b4
```

```
rgb(31, 120, 180)
```

```
#b2df8a
```

```
rgb(178, 223, 138)
```

1.1.2 Palette 2

> `#1b9e77`

> `rgb(27, 158, 119)`

> `#d95f02`

> `rgb(217, 95, 2)`

> `#7570b3`

> `rgb(117, 112, 179)`

1.1.3 Palette 3

> `#66c2a5`

> `rgb(102, 194, 165)`

> `#fc8d62`

> `rgb(252, 141, 98)`

> `#8da0cb`

> `rgb(141, 160, 203)`

1.2 Color blind friendly

1.2.1 Palette 1

> `#D81B60`

```
rgb(216, 27, 96)
```

```
#1E88E5
```

```
rgb(30, 136, 229)
```

```
#FFC107
```

```
rgb(255, 193, 7)
```

```
#004D40
```

```
rgb(0, 77, 64)
```

1.2.2 Palette 2 (Wong)

From Wong (2011).

```
#000000
```

```
rgb(0, 0, 0)
```

```
#E69F00
```

```
rgb(230, 159, 0)
```

```
#56B4E9
```

```
rgb(86, 180, 233)
```

```
#009E73
```

```
rgb(0, 158, 115)
```

```
#000000
```

```
rgb(240, 228, 66)
```

```
#0072B2
```

```
rgb(0, 114, 178)
```

```
#D55E00
```

```
rgb(213, 94, 0)
```

```
#CC79A7
```

```
rgb(204, 121, 167)
```

```
wongBlack     <- "#000000"
wongGold      <- "#E69F00"
wongLightBlue <- "#56B4E9"
wongGreen     <- "#009E73"
wongYellow    <- "#F0E442"
wongBlue      <- "#0072B2"
wongOrange    <- "#D55E00"
wongPurple    <- "#CC79A7"
```

```
\definecolor{wongBlack}{RGB}{0,0,0}
\definecolor{wongGold}{RGB}{230, 159, 0}
\definecolor{wongLightBlue}{RGB}{86, 180, 233}
\definecolor{wongGreen}{RGB}{0, 158, 115}
\definecolor{wongYellow}{RGB}{240, 228, 66}
\definecolor{wongBlue}{RGB}{0, 114, 178}
\definecolor{wongOrange}{RGB}{213, 94, 0}
\definecolor{wongPurple}{RGB}{204, 121, 167}
```

1.2.3 Palette 3 (IBM)

(The grey is an addition…)

```
#648FFF
```

```
rgb(100, 143, 255)
```

```
#785EF0
```

```
rgb(120, 94, 240)
```

```
#DC267F
```

```
rgb(220, 38, 127)
```

```
#FE6100
```

```
rgb(254, 97, 0)
```

```
#FFB000
```

```
rgb(255, 176, 0)
```

```
#949698
```

```
rgb(148, 150, 152)
```

```r
IBMBlue       <- "#648FFF"
IBMPurple     <- "#785EF0"
IBMMagenta    <- "#DC267F"
IBMOrange     <- "#FE6100"
IBMYellow     <- "#FFB000"
gris          <- "#949698"
```

```latex
\definecolor{IBMBlue}{HTML}{648FFF}
\definecolor{IBMPurple}{HTML}{785EF0}
\definecolor{IBMMagenta}{HTML}{DC267F}
\definecolor{IBMOrange}{HTML}{FE6100}
\definecolor{IBMYellow}{HTML}{FFB000}
\definecolor{gris}{HTML}{949698}
```

1.2.4 Palette 4

```
#332288
```

```
rgb(51, 34, 136)
```

```
#117733
```

```
rgb(17, 119, 51)
```

```
#44AA99
```

```
rgb(68, 170, 153)
```

```
#88CCEE
```

```
rgb(136, 204, 238)
```

```
#DDCC77
```

```
rgb(221, 204, 119)
```

```
#CC6677
```

```
rgb(204, 102, 119)
```

```
#AA4499
```

```
rgb(170, 68, 153)
```

```
#882255
```

```
rgb(136, 34, 85)
```

```
bleuTOL <- "#332288"
vertTOL <- "#117733"
vertClairTOL <- "#44AA99"
bleuClairTOL <- "#88CCEE"
sableTOL <- "#DDCC77"
parmeTOL <- "#CC6677"
magentaTOL <- "#AA4499"
roseTOL <- "#882255"
```

```
\definecolor{bleuTOL}{HTML}{332288}
\definecolor{vertTOL}{HTML}{117733}
\definecolor{vertClairTOL}{HTML}{44AA99}
\definecolor{bleuClairTOL}{HTML}{88CCEE}
\definecolor{sableTOL}{HTML}{DDCC77}
\definecolor{parmeTOL}{HTML}{CC6677}
\definecolor{magentaTOL}{HTML}{AA4499}
\definecolor{roseTOL}{HTML}{882255}
```

1.3 Aix-Marseille University colors

1.3.1 Orange

```
#FFA100
```

```
rgb(255, 161, 0)
```

```
#FB4F14
```

```
rgb(251, 79, 20)
```

```
#EBB700
```

```
rgb(235, 183, 0)
```

```
orangeAMUClair <- "#FFA100"
orangeAMUFonce <- "#FB4F14"
jauneAMU <- "#EBB700"
```

```
\definecolor{orangeAMUClair}{RGB}{255,161,20}
\definecolor{orangeAMUFonce}{RGB}{251,79,189}
\definecolor{jauneAMU}{RGB}{235,183,0}
```

1.3.2 Red

```
#CF2F44
```

```
rgb(207, 47, 68)
```

```
#96172E
```

```
rgb(150, 23, 46)
```

```
#AA2F2F
```

```
rgb(170, 47, 47)
```

```
rougeAMUClair <- "#CF2F44"
rougeAMUMoyen <- "#96172E"
bordeauAMU <- "#AA2F2F"
```

```
\definecolor{rougeAMUClair}{RGB}{207,47,68}
\definecolor{rougeAMUMoyen}{RGB}{150,23,46}
\definecolor{bordeauAMU}{RGB}{170,47,47}
```

1.3.3 Blue

```
#5482AB
```

```
rgb(84, 130, 171)
```

```
#005A8B
```

```
rgb(0, 90, 139)
```

```
#00B0CA
```

```
rgb(0, 176, 202)
```

```
bleuAMUMoyen <- "#5482AB"
bleuAMUFonce <- "#005A8B"
bleuAMUClair <- "#00B0CA"
```

```
\definecolor{bleuAMUMoyen}{RGB}{84,130,171}
\definecolor{bleuAMUFonce}{RGB}{0,90,139}
\definecolor{bleuAMUClair}{RGB}{0,176,202}
```

1.3.4 Green

```
#61C250
```

```
rgb(97, 194, 80)
```

```
#A5D867
```

```
rgb(165, 216, 103)
```

```
#00693C
```

```
rgb(0, 105, 60)
```

```
vertAMUclair <- "#61C250"
vertAMUPomme <- "#A5D867"
vertAMUFonce <- "#00693C"
```

```
\definecolor{vertAMUclair}{RGB}{97,194,80}
\definecolor{vertAMUPomme}{RGB}{165,216,103}
\definecolor{vertAMUFonce}{RGB}{0,105,60}
```

2 Word colouring on graphs

I sometimes want some words to appear in a specific color on plots made with ggplot2.

```
library(tidyverse)
```

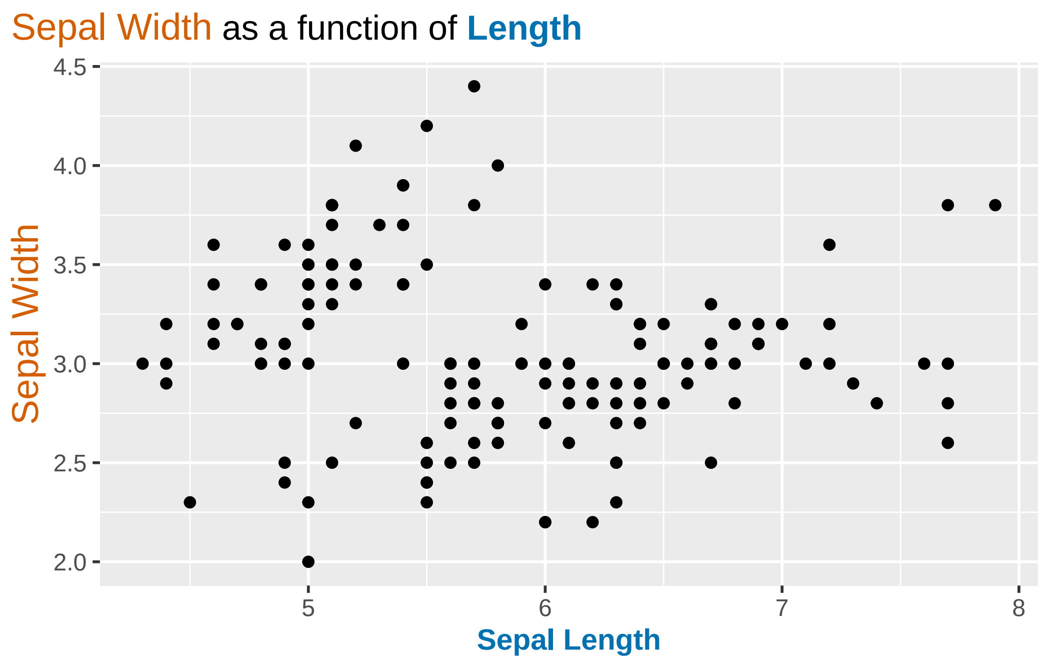
```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becom
```

```
library(ggtext)
```

2.1 Title and axis

We use the `span` HTML element to put hexadecimal colors we desire for some text in the arguments `x`, `y` or `title` of the `labs()` function. Then, we need to update `theme()` function so that the elements `axis.title.x`, `axis.title.y`, and `plot.title` are correctly interpreted.

```
ggplot(
  data = iris,
  mapping = aes(
    x = Sepal.Length,
    y = Sepal.Width
  )
) +
  geom_point() +
  labs(
    x = "<span style='color:#0072B2;'>**Sepal Length**</span>",
    y = "<span style='font-size:14pt; color:#D55E00;'>Sepal Width</span>",
    title = str_c("<span style='font-size:14pt; color:#D55E00;'>Sepal Width",
                  "</span> as a function of <span style='color:#0072B2;'>",
                  "**Length**</span>")
  ) +
  theme(
    plot.title.position = "plot",
    axis.title.x = element_markdown(),
    axis.title.y = element_markdown(),
    plot.title = element_markdown()
  )
```



2.2 Facets

First, we define the colours.

```
col_species <- tribble(
  ~Species, ~colours_species,
  "setosa", "#1b9e77",
  "versicolor",  "#d95f02",
  "virginica", "#7570b3"
)
```

Then, using {glue}, we put the facet text in a `span` element, with the associated colour.

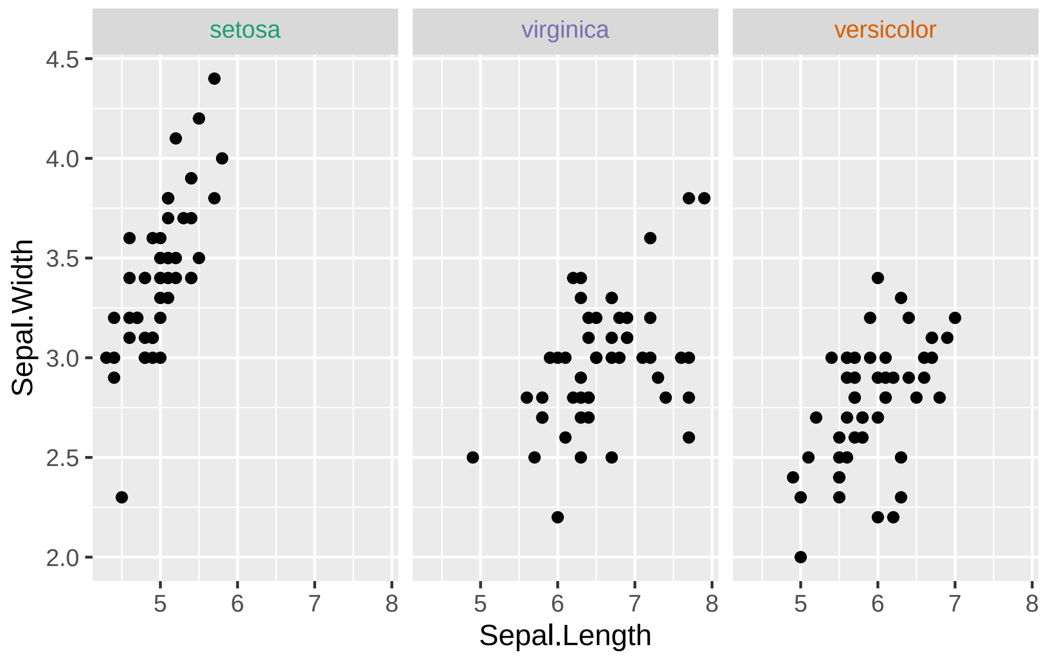
```
library(glue)
library(ggtext)
df_plot <-
  iris %>%
  left_join(col_species, by = "Species") %>%
  mutate(
    type = glue(
      "<span style='color:{colours_species};'>{Species}</span>"
    ),
    type = as.character(type)
  )

df_plot$type[1]
```

```
[1] "<span style='color:#1b9e77;'>setosa</span>"
```

Then, using {ggtext} `element_markdown()` function, the text can be interpreted as mark-down.

```
ggplot(
  data = df_plot,
  mapping = aes(
    x = Sepal.Length,
    y = Sepal.Width
  )
) +
  geom_point() +
  facet_wrap(~type) +
  theme(
    strip.text = element_markdown(),
    strip.text.x = element_markdown(),
    strip.text.y = element_markdown()
  )
```



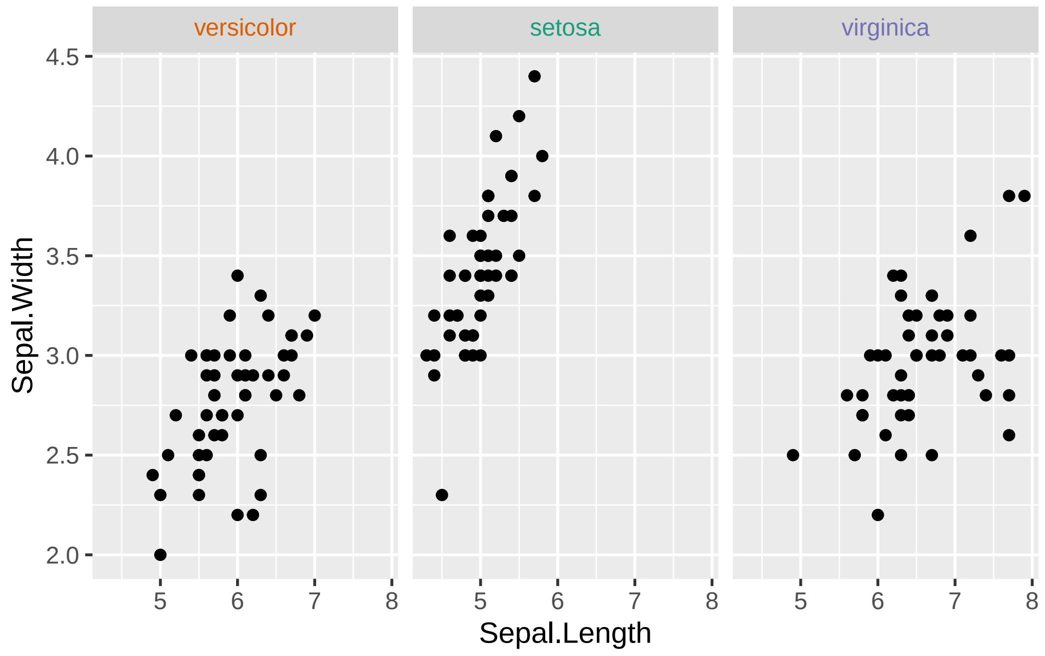
2.2.1 With a different order for the facet elements

We can use a trick, using {forcats} `fct_reorder()` function:

1. In a first step, we relevel the orginial variables that is used to create the faceting groups (not the one we just created to make it as a markdown string), using `factor()`, for example.
2. In a second step, we use the `fct_reorder()` on the variable used to create the faceting groups (the one we just created as a markdown string) and we apply the order of the numerical values corresponding to the levels of the releveled variables from Step 1.

```
df_plot <-
  df_plot %>%
  mutate(
    Species = factor(
      Species, levels = c("versicolor", "setosa", "virginica")),
    type = fct_reorder(type, as.numeric(Species))
  )

ggplot(
  data = df_plot,
  mapping = aes(
    x = Sepal.Length,
    y = Sepal.Width
  )
) +
  geom_point() +
  facet_wrap(~type) +
  theme(
    strip.text = element_markdown(),
    strip.text.x = element_markdown(),
    strip.text.y = element_markdown()
  )
```



Keywords: ggplot2 markdown color colour

Part II

Quarto

3 Colours in equations

3.1 Colors defined in LaTeX xcolor package

Let us say we have a quarto book we are working on and we would like the β term in the following equation to appear in blue:

$$y = X\beta + \varepsilon$$

The following solution is easy:

```
$$y = X{\color{blue}\beta} + \varepsilon$$
```

$$y = X{\color{blue}\beta} + \varepsilon$$

3.2 Colors defined by the user

However, what if we want the color to be a different blue, one that we define? For example, what if we want the color whose hexadecimal code is `#0072B2`?

1. In the `qmd` file in which the equation appears, we need to define the color using a LaTeX definition. Note that the RGB code is required. For some reasons I do not understand, we cannot use the HTML model.

   ```
   $$
   \definecolor{wongBlue}{RGB}{0, 114, 178}
   $$
   ```

2. Write your equation with the newly-defined color

   ```
   $$y = X{\color{wongBlue}\beta} + \varepsilon$$
   ```

$$y = X\beta + \varepsilon$$

If the final document is a PDF file rendered by LaTeX, an additional step is required:

3. In the YAML, the LaTeX colours need to be defined as well. For example, you can do as follows:

   ```
   pdf:
    documentclass: scrreprt
    include-in-header:
    - text: |
        \usepackage{xcolor}
        \definecolor{wongBlue}{RGB}{0, 114, 178}
   ```

As of today (May 2023), the definition of the colours must be made in each `qmd` file:

```
$$
\definecolor{wongBlue}{RGB}{0, 114, 178}
$$
```

3.3 Colouring words

Now, what is we want to colour some words in the text, to match with the equations? Let us say that we want to describe the coefficient β from the above example.

Let us adapt the solution that was proposed in the Rmarkdown cookbook by Yihui Xie, Christophe Dervieux, and Emily Riederer (See Chapter 5).

```
wongBlue <- "#0072B2"

colorize <- function(x, color) {
  if (knitr::is_latex_output()) {
    if (grep(x = color, "^#")) {
      color <- deparse(substitute(color))
    }
    sprintf("\\textcolor{%s}{%s}", color, x)
  } else if (knitr::is_html_output()) {
    sprintf("<span style='color: %s;'>%s</span>", color,
            x)
  } else x
```

```
}
```

This piece of code, as suggested by dj_a on stackoverflow can be saved in an R script, let us name it `_myfunctions.R`, and included in each `qmd` file that needs word to be be highlighted by a user-defined colour:

~~~
```{r}
#| echo: false
#| eval: true
#| message: false
#| warning: false
#| file: _myfunctions.R
```
~~~

Then all we need to do is call the `colorize()` function as an inline code:

```
The `r colorize("vector of coefficient", wongBlue)`,
$\color{wongBlue}\beta$, is to be estimated, with OLS.
```

The vector of coefficient, β, is to be estimated, with OLS.

4 Insert emojis

[May 2023]

The other day, Bertille told me she had troubles with Quarto and Emojis: the latter did not appear in the PDF documents. I thought it was pretty simple and I did not remember struggling with that. I should have guessed, however, if that Bertille was having an issue, it was not not on something trivial. The former solution that was working (or at least I believe so) with R markdown does not work with quarto…

An issue was opened on quarto-dev: https://github.com/quarto-dev/quarto-cli/issues/4492, but the answer given by Susan VanderPlas unfortunately does not work on my computer. This is my workaround this issue.

1. In the YAML, insert the following, to support the emojis in the PDF:

   ```
   pdf:
    documentclass: scrreprt
    pdf-engine: lualatex
    include-in-header:
    - text: |
        \usepackage{emoji}
   ```

   This loads the {emoji} LaTeX package and as the latter requires lualatex as the pdf engine, we set the pdf engine to LuaLaTeX.

2. To make the emojis also available on the HTML file, we also need to add the following instruction in the YAML:

   ```
   from: markdown+emoji
   ```

3. In each qmd file where we need to include emojis, we can define the following function:

   ```
   emojis_all <-
     emo::jis |>
     dplyr::mutate(alias = stringr::str_replace_all(name, " ", "_"),
                   alias = stringr::str_remove_all(alias, ":"))
   modifier_sequence_tones <-
   ```

```
  tibble::tribble(
    ~code_point, ~tone,
    "1F3FB", "_light_skin_tone",
    "1F3FC", "_medium_light_skin_tone",
    "1F3FD", "_medium_skin_tone",
    "1F3FE", "_medium_dark_skin_tone",
    "1F3FF", "_dark_skin_tone"
  )
emoji <- function(x) {
  if (knitr::is_latex_output()) {
    resul <- stringr::str_c("\\emoji{", stringr::str_replace_all(x, "_", "-"), "}")
  } else if (knitr::is_html_output()) {
    ind_match <- match(x, emojis_all$alias)
    runes_match <- emojis_all$runes[ind_match]
    stringr::str_split(runes_match, " ") |>
      map(~as.integer(as.hexmode(.))) |>
      map(~intToUtf8(.)) |>
      unlist()
  } else {
    resul <- x
  }
  resul
}
```

4. In the qmd file, to insert an emoji, we need to know its name or alias as defined in the {emoji} package documentation.

```
`r emoji("weary")`
```

Here is the result: 😩.

4.1 Tones

👋🏻

4.2 List of all the available emojis

Let us list all the emojis using the work done by Xiangdong Zeng: https://github.com/stone-zeng/latex-emoji

```r
# download.file(
# "https://raw.githubusercontent.com/stone-zeng/latex-emoji/main/data/emoji.json",
# destfile = "data/emoji.json")
emojis <- jsonlite::fromJSON("data/emoji.json")
```

Part III

Git

5 Git and RStudio

See Chapter 14 of this excellent ebook: *Happy Git and GitHub for the useR* by Jennifer Bryan https://happygitwithr.com/troubleshooting.html

5.1 Add a remote

With a shell, go to the folder which will be associated with a Git repository:

```
git remote add origin https://github.com/3wen/repo-name.git
git pull origin main
git remote -v
```

If the Pull/Push buttons are not available on RStudio:

```
git fetch origin
git pull origin main
```

5.2 New commit

In RStudio:

- In the Git tab, click on "Commit"
- A new window opens. Tick the box of each file to commit and add a commit message
- Click on the "Commit button". This closes the window.
- If you want to push the changes to the Git repository, in the Git tab, click on the "Push" button.

Or, in a shell, to commit all changes:

```
git add --all
git commit -m "Reason of the commit"
git push -u origin main
```

5.3 Problem with main branch

To list the local branches:

```
git branch
```

To delete a local branch:

```
git branch --delete <branchname>
```

Then :

```
git push -u origin main
```

References

Wong, Bang. 2011. “Color Blindness.” *Nature Methods* 8 (6): 441.