Module 1 Introduction to Literate Programming with Quarto

Principle Assignment

Jae Jung

2025-01-14

Overview

Learning outcomes

1. Install R and RStudio.
2. Describe the layout and menus of RStudio.
3. Compare and contrast different types of codebook in RStudio - R Scripts, Rmd, and Quarto.
4. Describe the operational basics necessary to perform in every projects in R
5. Explain typical workflow for a Quarto project in RStudio.
6. Explain what the reproducible research is and the role of literate coding.
7. Describe major authoring tools you can implement to make your Quarto document effective.
8. Describe some major computational options you can implement on Quartor documents.
9. List Quarto's capabilities that makes it an alternative to MS Office.
10. List online resources available in R Community when you need help.

The textbook chapters to cover:

- Ch01: Introduction to R for Data Science
- Ch28: Quarto
- Ch02: Workflow: basics
- Ch04: Workflow: Code style
- Ch06: Workflow: Scripts and projects
- Ch08: Getting help
- Ch01: Data visualization

R & R Studio

- Brief History
- Installation
- R Studio IDE menu
- Three major file types in RStudio

RStudio IDE

- Four quadrants
- Menu
  - Tools > Global Options
  - Output location options

Steps in Preparing for a project

Start a new project

- Projects are the containers for all of your scripts

1. Press new project
2. Select a new or existing directory depending on your needs
3. Select your folder that contains your scripts
4. Press create project

- You should see your .proj file and others in the files panel

Start a codebook and save it

R Scripts

Rmd file

Quarto File

Markdown vs. HTML

Markdown and HTML are both markup languages used for creating formatted content, but they have distinct characteristics. Here's a summary of their similarities and differences:

Feature	Markdown	HTML
Syntax complexity	Simple and easy to read	More complex with specific tags
Learning curve	Low, quick to learn	Steeper, requires more time to master
Readability	Highly readable, even in raw form	Less readable in raw form
Flexibility	Limited formatting options	Highly flexible with extensive formatting
Output	Primarily static content	Dynamic web pages and applications
Supported elements	Basic formatting (headings, lists, links)	Wide range of elements (forms, multimedia)
Conversion	Easily converts to HTML	Cannot be directly converted to Markdown
Collaborative editing	Well-suited for collaboration	Less ideal for collaborative editing
Customization	Limited styling options	Extensive styling with CSS
Use cases	Documentation, simple content	Complex web development, detailed layouts
Browser support	Requires conversion to HTML	Natively supported by browsers

Feature	Markdown	HTML
Extensibility	Limited, varies by flavor	Highly extensible with JavaScript

Start literate coding

Literate Programming

Basic Operations

In the following code chunk, you will learn some basic operations in R.

```
```{r}
#| label: Basic-Operation
1+1
2*2 # *: multiplication
2^3 # ^: use carrot to raise the base to the power of the following number.

#creating an object
message <- "Hello WOrld!" # assigning elements to a variable. Do not break between less-than si

message = "Hello WOrld!" # equal sign also works.

#to print,
print(message)

#to print, print function is necessary. You can just type the object and run it.
message

# An object can be any type: e.g., strings and numbers
number <- 7

max(2,5,90,30) # maximum

min(2,5,90,30) # minimum
```
```

```
[1] 2
[1] 4
[1] 8
[1] "Hello WOrld!"
[1] "Hello WOrld!"
[1] 90
[1] 2
```

Coding Styles

````
```{r}
#| code-fold: false

#install.packages("tidyverse")
#install.packages("palmerpenguins")
library(tidyverse)
library(palmerpenguins)
```
````

Base R way of coding

````
```{r}
#| code-fold: false

head(penguins)
mean(penguins$bill_length_mm, na.rm = TRUE)
```
````

```
# A tibble: 6 x 8
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
  <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
1 Adelie  Torgersen           39.1          18.7               181        3750
2 Adelie  Torgersen           39.5          17.4               186        3800
3 Adelie  Torgersen           40.3          18                 195        3250
4 Adelie  Torgersen           NA            NA                  NA          NA
5 Adelie  Torgersen           36.7          19.3               193        3450
6 Adelie  Torgersen           39.3          20.6               190        3650
```

```
# i 2 more variables: sex <fct>, year <int>
[1] 43.92193
```

Tidyverse way of coding

````
```{r}
#| code-fold: false

penguins |>
  head()
penguins |>
  pull(bill_length_mm) |>
  mean(na.rm = TRUE)
```
````

```
# A tibble: 6 x 8
  species island    bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
  <fct>   <fct>              <dbl>         <dbl>             <int>       <int>
1 Adelie  Torgersen           39.1          18.7               181        3750
2 Adelie  Torgersen           39.5          17.4               186        3800
3 Adelie  Torgersen           40.3          18                 195        3250
4 Adelie  Torgersen           NA            NA                  NA          NA
5 Adelie  Torgersen           36.7          19.3               193        3450
6 Adelie  Torgersen           39.3          20.6               190        3650
# i 2 more variables: sex <fct>, year <int>
[1] 43.92193
```

Pipe Operator

> **Tip**
>
> Note that R has always multiple ways to accomplish the same goal. `|>` is called `native pipe operator`. It works the same as `%>%`, which came from `mgrittr` package that revolutionized the way we code in R, paving the trend for the modern data science in R.
> While the data frame, df, in Figure 3 was included inside ggplot function, the df in Figure 4 was the first appear in

the code.

> The pipe operator is one important difference between base R and Tidyverse in how we code.

Quarto

- Quarto unifies the functionality of many packages from the R Markdown ecosystem (rmarkdown, bookdown, distill, xaringan, etc.) into a single consistent system as well as extends it with native support for multiple programming languages like Python and Julia in addition to R.
- In a way, Quarto reflects everything that was learned from expanding and supporting the R Markdown ecosystem over a decade.

Quarto Work Flow Basics

- How to start it and save it (e.g., test.qmd)
- Rendering it:
- Source vs. visual tab interface
- r4ds: https://r4ds.hadley.nz/quarto#quarto-basics

Three Areas in Quarto File

- Detailed tutorials availabe at the official Quarto site

Yaml header

> An (optional) YAML header demarcated by three dashes (—) on either end.

```
---
title: "Module 1 Introduction to R, RStudio, and Qurto"
author: "Jae Jung"
date: '2025-01-14 00:51:27'
```

```
format:
  html:
    toc: true
    toc-depth: 4
    embed-resources: true
editor: visual
execute:
  freeze: auto
---
```

Code chunk

````
```{r}
#| label: demo-code-chunk
#| include: true

#install.packages("tidyverse")
#install.packages("palmerpenguins")
library(tidyverse)
library(palmerpenguins)
```
````

Markdown text

- Text area is all the canvas area within qmd file other than Yaml header and code chun areas.
- Quarto uses markdown syntax for text.
  - If using the **visual editor**, you won't need to learn much markdown syntax for authoring your document, as you can use the menus and shortcuts to add a header, bold text, insert a table, etc.
  - If using the **source editor**, you can achieve these with markdown expressions like ##, **bold**, etc.
- You can use text area for typing pros as you would normally do in MS Word or Google Doc.

  - Text with formatting, including section headers, hyperlinks, an embedded image, and an inline code chunk.

- You can also style it.
  - Bold
  - Italicize
  - Headings: h1, h2, h3

> Coding Tips
>
> Note: It is possible to type the code and run in the text area.
>
> - However, your code in the text area won't be read and rendered into a document unless your codes are inside a code chunk. - During the rendering, RStudio will be in auto piolt mode and will treat everything in the text area a text except for in-line coding, which will be demonstrated later.
> - Thus, do **not *code*** in the **text area**.

Qurto Interface

Visual editor

- Easier to those who are familiar with MS Word or Google Doc.
- `ctrl/commnad + /`
- Adding table by hand is cumbersome; use the visual mode as in Section .
- Adding an figure/image

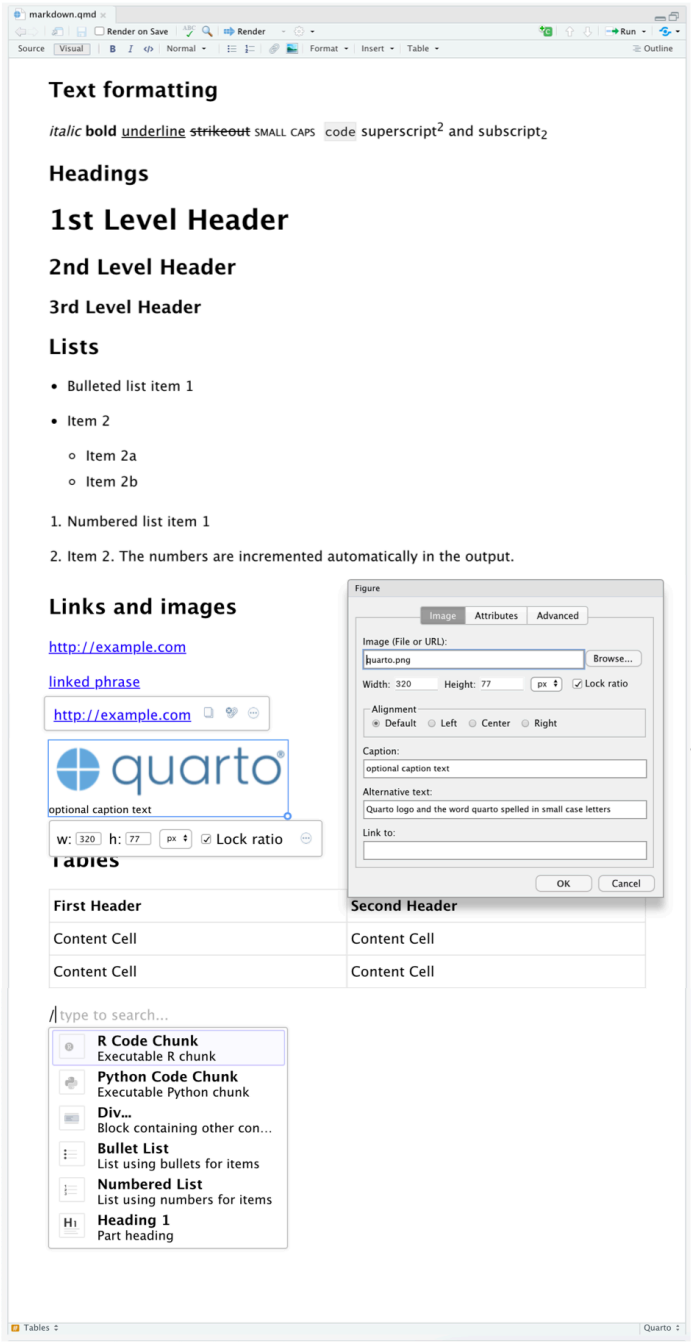


Figure 1: fig-visual-editor

Source editor

- Easier to those who are familiar with R Script file or Rmd file.
- Useful for debugging any Quarto syntax errors since it's often easier to catch these in plain text.
- Handy reference sheet available at the RStudio menue: *Help > Markdown Quick Reference*
- The guide below shows how to use Pandoc's Markdown for authoring Quarto documents in the source editor.

Quarto Document types

HTML

Note: Practice creating documents in each form.

pdf

- In order to create PDFs you will need to install a recent distribution of LaTeX.
- Use TinyTeX (which is based on TexLive), which you can install with the following command:

  ```
  Terminal

  quarto install tinytex
  ```

MS Word

```
---
title: "Testing for Word document"
format: docx
editor: visual
---
```

revealjs presentation

```
---
title: "Testing for Presentation"
format: revealjs
editor: visual
---
```

Dashboard

```
---
title: "Testing for Presentation"
format: dashboard
editor: visual
---
```

Multiple formats

Some documents you create will have only a single output format, however in many cases it will be desirable to support multiple formats. Let's add the `html` and `docx` formats to our document and modify some options specific to each format.

```
---
title: "Housing Prices"
author: "YOur Name"
highlight-style: pygments
format:
  html:
    code-fold: true
    html-math-method: katex
  pdf:
    geometry:
      - top=30mm
      - left=30mm
  docx: default
---
```

Rendering to all formats.

If you would like to render to all formats, you can do so with the quarto package, which provides an R interface to the Quarto CLI. - For example, to render the current document, use quarto::quarto_render(). You can also specify the name of the document you want to render as well as the output format(s).

````
```{r}
#| warning: false
#| eval: false
quarto::quarto_render(
  "test.qmd",
  output_format = c("pdf", "docx")
)
```
````

Quarto Markdown Basics

Text formatting

1. *italic*
2. **bold**
3. ~~strikeout~~
4. `code`
5. superscript^2
6. subscript$_2$
7. Using spans
    - underline
    - SMALL CAPS
    - can change the font color to red

Headings

```
# 1st Level Header

## 2nd Level Header

### 3rd Level Header
```

Note: There must be a space between the last pound and the first letter of the headings.

Lists

Unnumbered list

- Bulleted list item 1
  - sub-item 1
  - sub-itme 2
    - sub-sub-item 1
- Item 2
  - Item 2a
  - Item 2b

Numbered List

1. Numbered list item 1
2. Item 2. The numbers are incremented automatically in the output.

   i) sub-item 1

      a. sub-sub-item 1

Links and images

https://www.cpp.edu/cba/customer-insights-lab/index.shtml

Center for Customer Insights and Digital Marketing

Figure 2: CCIDM Logo

Callout Blocks

> **Note**
>
> - Callouts are markdown divs that have special callout attributes.

> **Important**
>
> - We can insert a callout using the Insert Anything tool.

Blockquote

> Blockquote highlights the prose by making it bigger.

Panel Tabset

- An example is provided in section Section .

Tables

- Using visual mode

Table 2: MSDM Program Curriculum[1]

	Digital Marketing Strategy Emphasis	Marketing Analytics Emphasis
Major Required	20 units	20 units
Major Electives	14 units	14 units
Emphasis	10 units	10 units
Recommended Emphasis	4 units	4 units
Other		
Total Units	***34 units***	***34 units***

Cell output

> By default, the code and its output are displayed within the rendered document.

- showing code: `echo: true`
- Hiding code: `echo: false`
- Can be done globally at Yaml or locally at cell.
  - **Global setting**

Echo: true/false

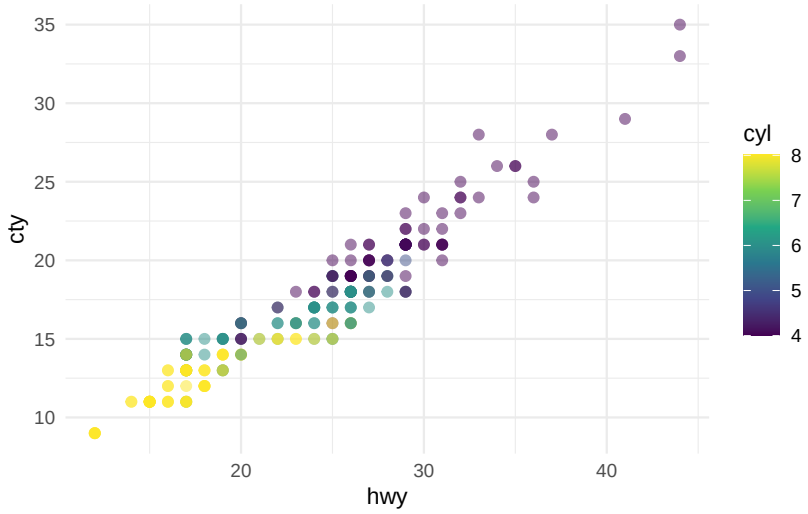
```
---
title: "Quarto Computations"
execute:
  echo: false
---
```

[1]This is a table caption added when a table was created in the visual mode.

- **Local setting**

To override code hiding global setting at Yaml, add the `echo: true` cell option as shown below in the chunk labelled `scatterplot`.

```
ggplot(mpg, aes(x = hwy, y = cty, color = cyl)) +
  geom_point(alpha = 0.5, size = 2) +
  scale_color_viridis_c() +
  theme_minimal()
```



> **Warning**
>
> `echo: true` prints all the codes but not the cell setting such as label and echo.

Echo: fenced

- Great for teaching or publication purpose
- Code chunk printed except for `echo: fenced`

````
```{python}
#| ouput: false
#| code-overflow: wrap
```
````

```
1 + 1
```

2

Data Visualization Workflow

Information about mpg data set

- There are a lot of built-in data set ready for use in R.
- mpg is one of the data set

```
```{r data info}
# ?mpg  # help
# help(mpg) # another way of looking up help
```
```

Load up data and set-up

```
```{r prep}
library(tidyverse)
library(GGally)

mpg # print mpg dataset
df <- mpg # assigning the data set to a new name
head(df)
theme_set(theme_light()) #set the graphics theme to a light style for this R session.
```
```

```
# A tibble: 234 x 11
   manufacturer model      displ  year   cyl trans drv     cty   hwy fl    class
   <chr>        <chr>      <dbl> <int> <int> <chr> <chr> <int> <int> <chr> <chr>
 1 audi         a4           1.8  1999     4 auto~ f        18    29 p     comp~
 2 audi         a4           1.8  1999     4 manu~ f        21    29 p     comp~
 3 audi         a4           2    2008     4 manu~ f        20    31 p     comp~
 4 audi         a4           2    2008     4 auto~ f        21    30 p     comp~
```

```
 5 audi         a4           2.8  1999     6 auto~ f        16    26 p     comp~
 6 audi         a4           2.8  1999     6 manu~ f        18    26 p     comp~
 7 audi         a4           3.1  2008     6 auto~ f        18    27 p     comp~
 8 audi         a4 quattro   1.8  1999     4 manu~ 4        18    26 p     comp~
 9 audi         a4 quattro   1.8  1999     4 auto~ 4        16    25 p     comp~
10 audi         a4 quattro   2    2008     4 manu~ 4        20    28 p     comp~
# i 224 more rows
# A tibble: 6 x 11
  manufacturer model displ  year   cyl trans      drv     cty   hwy fl    class
  <chr>        <chr> <dbl> <int> <int> <chr>      <chr> <int> <int> <chr> <chr>
1 audi         a4      1.8  1999     4 auto(l5)   f        18    29 p     compa~
2 audi         a4      1.8  1999     4 manual(m5) f        21    29 p     compa~
3 audi         a4      2    2008     4 manual(m6) f        20    31 p     compa~
4 audi         a4      2    2008     4 auto(av)   f        21    30 p     compa~
5 audi         a4      2.8  1999     6 auto(l5)   f        16    26 p     compa~
6 audi         a4      2.8  1999     6 manual(m5) f        18    26 p     compa~
```

Research Hypothesis or Research Questions

- H1: City mileage will be negatively associated with the size of cylinder. or
- RQ1: Would city mileage differ by cylinder size of vehicles? If so, how?

Visualize the data and assess the output

Density Plot

- is for continuous variable.
  - In Figure 3, `cty` is a continuous variable.
- alpha argument determines the thickness of the color; smaller the number the more transparent it would be.
- fill argument have a different effect depending on whether we add it inside or outside aes() function.
- factor() transforms the data into a factor, categorical data type.

```{r Density plot}
#| label: fig-cty-by-cyl-density
#| fig-cap: The impact of cylinder size on the city mileage using density plot

g <- ggplot(df, aes(x = cty)) # there is only one variable, "cty"
g + geom_density(aes(fill=factor(cyl)), alpha=0.8) +
  labs(title = "Density Plot",
       subtitle = "City Mileage Grouped by Number of Cylinders",
       caption = "Source: mpg dataset",
       x= "city Mileage",
       fill="# Cylinders")
```

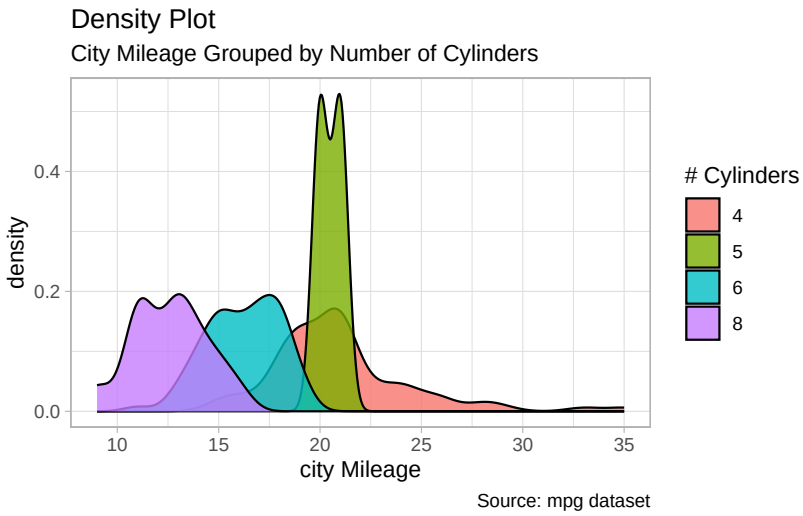


Figure 3: The impact of cylinder size on the city mileage using density plot

Histogram

- is for continuous variable.
  - in Figure 4, the cty is continuous variable.

```{r Histogram}
#| label: fig-cty-by-cyl-histogram
#| fig-cap: The impact of cylinder size on the city mileage using histogram
```

```
df |>
ggplot(aes(x = cty, fill=factor(cyl))) +
geom_histogram(color = "white", alpha=0.8) +
  facet_wrap(~ factor(cyl), ncol = 1) +
  labs(title = "City Mileage Grouped by Number of Cylinders",
       subtitle = "Histogram",
       caption = "Source: mpg dataset",
       x = "City Mileage",
       y = "Frequency / # of Cars",
       fill = "# of Cylinder"
       )
```

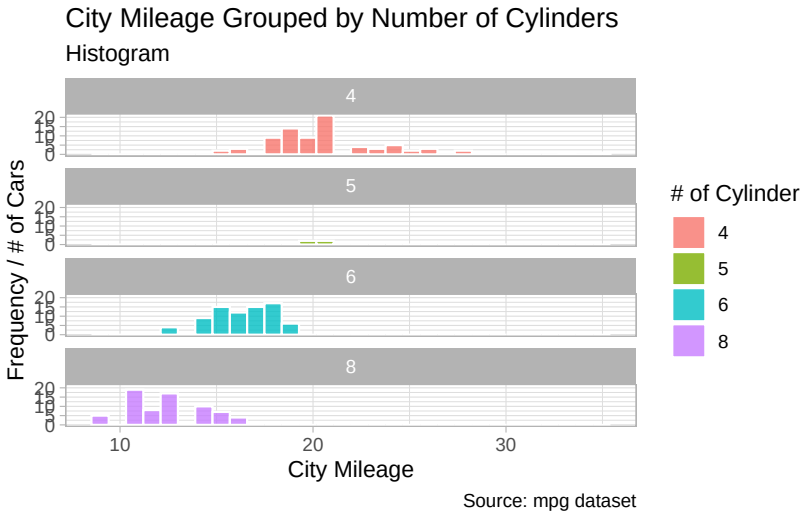


Figure 4: The impact of cylinder size on the city mileage using histogram

Summary Insights from both plots

> **!** Important
>
> As can be seen in both Figure 3 and Figure 4, cylinder size is negatively related to the city mileage.

Bar Plot

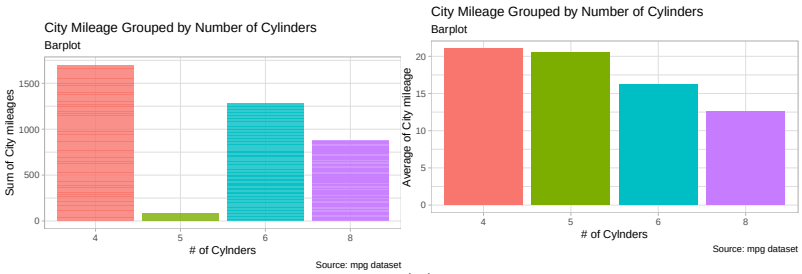
- Bar plot is usually used to count one categorical variable, but it can also be used for categorical x and continuous y using "stat = identity".
- Can we plot the impact of cylinder on city mileage using bar plot?
- The answer is yes, but a caution is needed to avoid a mistake.
- Let's consider Figure 5a and Figure 5b below.

```
```{r Barplot}
#| message: false
#| label: fig-cty-by-cyl-bar
#| fig-cap: The impact of cylinder size on the city mileage using barplots
#| fig-subcap:
#|   - "cty on Y-axis by summing them by cylinder size"
#|   - "cty on Y-axis by averaging them per cylinder size during the wrangling stage"
#| layout-ncol: 2

df |>
ggplot(aes(x = factor(cyl), y = cty)) +
geom_col(aes(fill=factor(cyl)), alpha=0.8, show.legend = FALSE) +
  labs(title = "City Mileage Grouped by Number of Cylinders",
       subtitle = "Barplot",
       caption = "Source: mpg dataset",
       x = "# of Cylnders",
       y = "Sum of City mileages"
       )

df |>
  group_by(cyl) |>  # group data by cyl
  summarize(cty_mean = mean(cty), .groups = "drop") |>  # calculate mean of cty and call it "ct
ggplot(aes(x = factor(cyl), y = cty_mean)) +
geom_col(aes(fill=factor(cyl)), show.legend = FALSE) +
  labs(title = "City Mileage Grouped by Number of Cylinders",
       subtitle = "Barplot",
       caption = "Source: mpg dataset",
       x = "# of Cylnders",
```
```

```
        y = "Average of City mileage"
        )
```



(a) cty on Y-axis by summing them by cylinder size

(b) cty on Y-axis by averaging them per cylinder size during the wrangling stage

Figure 5: The impact of cylinder size on the city mileage using barplots

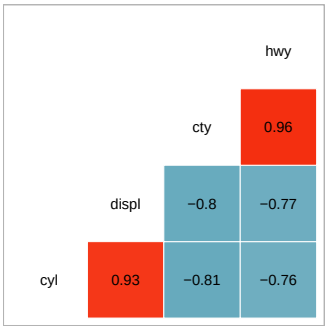
> **Lessons**
>
> When using `geom_bar` or `geom_col` with continuous y variable, make sure y is expressed as an an average, not sum.

Correlation

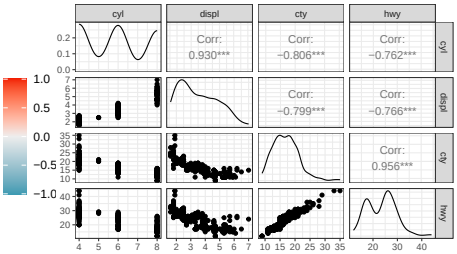
```
```{r}
#| message: false
#| label: fig-GGally
#| fig-cap: Correlations table using GGally package
#| fig-subcap:
#|   - "correlations using ggcorr"
#|   - "correlations using ggpairs"
#| layout-ncol: 2
#| column: page

df %>%
  select(cyl, displ, cty, hwy) %>%
  ggcorr(label = TRUE, label_round = 2)
```
```

```
df %>%
  select(cyl, displ, cty, hwy) %>% # select all continuous variables in the data
  ggpairs()+
  theme_bw()
```



(a) correlations using ggcorr

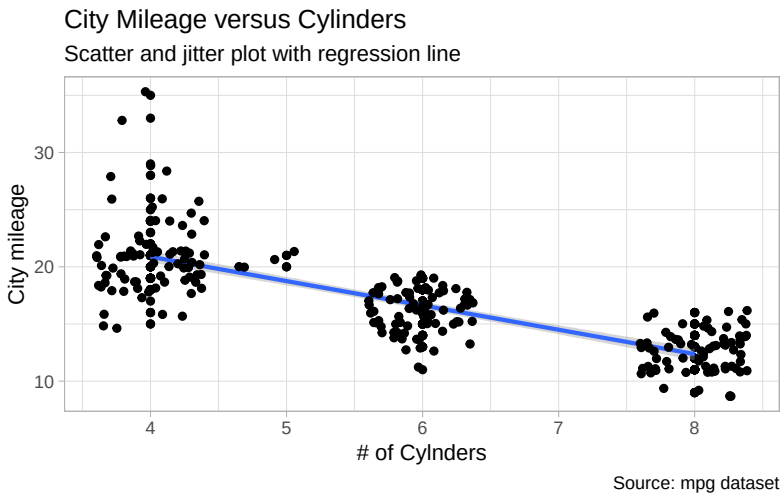


(b) correlations using ggpairs

Figure 6: Correlations table using GGally package

Scatter plot with regression line

```
```{r geom point}
df %>%
  select(cyl, cty) %>%
  ggplot(aes(x = cyl, y = cty))+
  geom_point() +
  geom_smooth(method = "lm") +
  geom_jitter()+
  labs(title = "City Mileage versus Cylinders",
       subtitle = "Scatter and jitter plot with regression line",
       caption = "Source: mpg dataset",
       x = "# of Cylnders",
       y = "City mileage",
       )
```
```



gt and gtsummary tables

````
```{r}
#| label: tbl-mpg-regression
#| tbl-cap: Regression of cty on cyl

library(gtsummary)
library(gt)

df |>
  lm(cty ~ cyl, data = _) |>
  tbl_regression(
  ) |>
  add_n() |>
  modify_header(label = "**Variables**") |>
  as_gt() |>
  tab_header(title = md("**Impact of Cylinder Size on City Mileage (cty)**"),
             subtitle = md("with `mpg` data"))

m_reg <- df |>
  lm(cty ~ cyl, data = _)
```
````

Table 3: Regression of cty on cyl

Impact of Cylinder Size on City Mileage (cty)
with `mpg` data

Variables	N	Beta	95% CI[1]	p-value
cyl	234	-2.1	-2.3, -1.9	<0.001

[1]CI = Confidence Interval

Conclusion

> **!** Important
>
> As shown in all the correct charts above, it appears that the number of cylinders is negatively related to city mileages. One may proceeds to use an inferential statistics as in Table 3 to draw a formal conclusion.
>
> > According to the correlation analysis above - Figure 6a and Figure 6b, the cylinder and the city mileage are highly negatively correlated (r = -0.81). The regressing city miles on cylinder size shows statistically significant negative impact of cylinder size on the city miles such that one unit increase in cylinder size leads to the -2.13 miles decrease in the city mileage.

Resources in R Community

Resources for this module

- R for Data Science (2nd eds): https://r4ds.hadley.nz/
- Quarto Guide: https://quarto.org/docs/get-started/hello/rstudio.html
- Quarto Cheatsheet: <https://rstudio.github.io/cheatsheets/html/quarto.html>

- Quarto presentation Workshop: https://rstudio-conf-2022.github.io/get-started-quarto/materials/05-presentations.html#/presentations

Resources in R Community

- Tidy Tuesday
- R Facebook Groups
  - R Statistical Software Group
- Stackoverflow
  - Good for general questions
- YouTube – *Most effective*
- X
- other social media