Data Visualization

Matt Steele

2023-11-10

GGPLOT2

visualize your data with ggplot

- ggplot package

Grammar of Graphics

the grammar of graphics is implemented through a layered approach to building plots.

Required

1. Data: The raw data that you want to visualize.
2. Aesthetic Mapping (aes()): Defines how variables in the data map to visual properties like position, color, size, shape, etc.
3. Geometric Objects (geom_*): Represent the actual geometric shapes on the plot (e.g., points, lines, bars).

Optional

4. Coordinate System (coord_*):Defines the coordinate system for the plot (e.g., Cartesian, polar).
5. Faceting (facet_*): Divides the plot into subplots based on one or more categorical variables.
6. Themes (theme_*): Customize the appearance of the plot, including titles, labels, and overall aesthetics.

Line graph

lets create a line graph that tracks approval ratings for the Supreme Court of the United States over time.

Data

```
# load scotus approval data
```

```
scotus <- read_csv("scotus_approval.csv")

# let's just view the pollster YouGov using the filter function

scotus_yg <- scotus |>
  filter(pollster == "YouGov")

scotus_yg
```

Aesthetic Mapping (aes())

next we are going to set the variables that will use by using the ggplot function along with the aes function

```
# set the parameters and coordinates

scotus.line <- ggplot(scotus_yg, aes(date, per_yes))

# calling that object will give us an empty plot


scotus.line
```

Geometric Objects (geom_*)

choose the plot that we want to use for our visualization using the geom_* element

- Geom Cheatsheet

```
We combine elements in ggplot using the (+) operator
```

```
scotus.line +
  geom_line()
```

Theme: Color, Geom Size, Transparancy

fill = or **color =**	change colors
size =	change size

alpha = change transparency

```
# add color

scotus.line +
  geom_line(color = "coral")

# change the size

scotus.line +
  geom_line(color = "coral", size = 2)
```

Theme: Labels

the labs element will allow you to add or change labels in the plot

```
scotus.line +
  geom_line(color = "coral", size = 2) +
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
    x = NULL
  )
```

Themes: Built-in

ggplot has built-in themes with pre-set settings for you

```
scotus.line +
  geom_line(color = "coral", size = 2) +
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
    x = NULL
  ) +
```

```
  theme_minimal()
```

Themes: Customize

the theme element will allow you to customize the appearance of axes, legends, and labels

```
scotus.line +
  geom_line(color = "coral", size = 2) +
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
    x = NULL
  ) +
  theme_minimal() +
  theme(plot.title = element_text(size = 20, color = "navy"))
```

Theme: Scales

the scales element allows you to fine-tune and adjust the mapping/scale of labels, breaks, and legends

- the **scale_x_date** element allows you to adjust your date elements on the x axis
- Date Formats - strptime

```
scotus.line +
  geom_line(color = "coral", size = 2) +
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
    x = NULL
  ) +
  theme_minimal() +
  theme(plot.title = element_text(size = 20, color = "navy")) +
  scale_x_date( date_breaks = "6 weeks",
                date_labels = "%b %d")
```

Smoothed Lines

You can reduce overplotting using **loess** or **linear regression lines** with the geom_smooth or stat_smooth element

```
scotus.line +
  geom_smooth(color = "coral", size = 2) +
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
    x = NULL
  ) +
  theme_minimal() +
  theme(plot.title = element_text(size = 20, color = "coral")) +
  scale_x_date( date_breaks = "6 weeks",
                date_labels = "%b %d")
```

Export your plot

The ggsave function will export the most recent plot called in a fille type specified by the user

```
ggsave("scotus_approval.png", plot = my_plot, width = 6, height = 4, dpi = 300)
```

Additionally you can use the export options in RStudio's Plot tab in the Misc Pane

---

Histogram Graph

the histogram geom allows you to see the distribution of a continuous (dbl or num) variable

```
# load demographics data frame

demo <- read_csv("demographics.csv")


# let's look at the distribution of the age variable by creating a histogram

demo.hist <- ggplot(demo, aes(age))

demo.hist +
  geom_histogram()
```

Binning

the binning argument allows you to group continuous data into discrete intervals or bins

```
# number of bins to use

demo.hist +
  geom_histogram(bins = 10)

# length of a bins

demo.hist +
  geom_histogram(binwidth = 15)
```

Theme: Color, Geom Size, Transparancy

fill = or **color =**	change colors
size =	change size
alpha =	change transparency

```
demo.hist +
  geom_histogram(bins = 25, color = "coral", fill = "skyblue", alpha = .5) +
  theme_light()
```

Order of Elements

The order that the elements appear on the plot is dictated by its position in your code.

- The first elements in the code appear at the bottom of the plot and the last elements appear on the top of you plot

Mutliple Geoms

We can add multiple geoms into a plot by adding theme as their own element

the geom_vline/geom_hline element allows you to add a reference line to your plot

```
# add a reference line

demo.hist +
  geom_vline(xintercept = 40, color = "navy", size = 3) +
  geom_histogram(bins = 25, color = "coral", fill = "skyblue", alpha = .5) +
  theme_light()
```

Faceting

the facet_grid or facet_wrap element will allow you to break your plot out by categorical variables

```
demo.hist +
  geom_vline(xintercept = 40, color = "navy", size = 3) +
  geom_histogram(bins = 25, color = "coral", fill = "skyblue", alpha = .5) +
  theme_light() +
  facet_wrap(facets = vars(inccat), nrow = 3)
```

Bar Graph

the geom_bar element allows you create a bar chart uses the number of cases of each group in a categorical variable

```
demo.bar <- ggplot(demo, aes(carcat))

demo.bar +
  geom_bar()
```

the geom_col element allows you to create a bar chart using a categorical and continuous variable

```
demo.col <- ggplot(demo, aes(carcat, income))

demo.col +
  geom_col()
```

Reorder Plot

you can order the bar graph using the fct_reorder function from Forcats

```
demo.col <- ggplot(demo, aes(fct_reorder(carcat, income), income))

demo.col +
  geom_col()
```

Add Additional Variable

you can use the fill argument in aes to map an additional variable onto individual bars

```
demo.col +
  geom_col(aes(fill = ed))
```

Add Color Palletes

The scale_fill_brewer function will allow you to add pre-built palettes to your plot

- Color Brewer

```
demo.col +
  geom_col(aes(fill = ed)) +
  scale_fill_brewer(palette = "Pastel1")
```

Scales

the scales element allows you to fine-tune and adjust the mapping/scale of labels, breaks, and legends

- the **scale_y_continuous** or **scale_x_continuous** along with label_number elements allows you to adjust a numeric axis

```
demo.col +
  geom_col(aes(fill = ed)) +
  scale_fill_brewer(palette = "Pastel1") +
  scale_y_continuous(labels = scales::label_number_si())
```

---
title: "Introduction to R Markdown"
author: "Matt Steele"
date: "`r Sys.Date()`"
output:
  html_document: default
  word_document: default

Additional Resources

- [R Markdown for RStudio](https://rmarkdown.rstudio.com/)
- [R Markdown Cheatsheet](https://doi.org/10.1093/oso/9780197582756.003.0009)
- [R Markdown Reference Guide](https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf?_ga=2.116454790.1075794466.1676401806-680139860.1644522425)
- [R Markdown Definitive Guide](https://bookdown.org/yihui/rmarkdown/)
- [R Markdown Cookbook](https://bookdown.org/yihui/rmarkdown-cookbook/)

About R Markdown

R Markdown allows you to blend formatted prose with code to create reproducible scientific documents that can be outputted in a HTML, PDF, and MS Word document.

Clicking on the **Knit** button in the editor toolbar will generate a document that includes both the content as well as the output of any embedded R code chunks within the document.

- Global Options
- Markdown Quick Reference (Help)

Why Bother?

1. Encourages you to document your analysis
2. Provides a non-proprietary format that you can easily store, preserve, document with metadata, and retrieve at later dates.
3. Reproducibility means that you can share the document with colleagues and peers to check errors or to collaborate easily. R Markdown even allows for multiple coding languages to be used in a single document.
4. Create reports/documents that are dynamically generated from you data and can be easily revised. R Markdown documents are dynamic and an errors or issues with the coding can be made with little work on the user's end.

- No longer do you need to re-code and re-paste

YAML Header

  - YAML AIN'T MARKUP LANGUAGE

This is the metadata area for your document and it also determines how the document is rendered when you knit it.
It's default fields are **title**, **author**, **date**, and **output**.
But you can add more fields.

[Available fields for YAML](https://cran.r-project.org/web/packages/ymlthis/vignettes/yaml-fieldguide.html)

About YAML:

- White spaces matter: indents indicate the contents are *child* of the level above
  - Spaces not tabs
- Boolean operators: true/false is lowercase
- true/false \~ yes/no
- Entries can include executable code
  - "`r Sys.Date()`"
- Most common outputs are *html_document*, *pdf_document*, and *word_document*
  - [Full listing of available formats](https://rmarkdown.rstudio.com/formats.html)
  - For example, if you are interested in creating an interactive dashboard you would want to use the flexboard package output

```{r HTML help, eval=FALSE}
# Help with HTML header options

?html_document


```

```{r PDF help, include=FALSE, eval=FALSE}

#Help with PDF header options

?pdf_document


```

```{r MS DOC help, include=FALSE, eval=FALSE}

#Help with MS Word header options

?word_document

```

Formating Options

The following will provide ways for you to format your text/prose within the document that you are editing

```{r}

#| label: formatting
#| eval: false


# Header 1
```

Header 2

Header 3

Header 4

Header 5

Header 6

Italics - *I am italic - mamma mia*

Bold - **I am bold**

Hyperlink - You can learn more about [RMarkdown here](https://rmarkdown.rstudio.com/)

Image - ![Spongebob](spongebob.jpg)

Footnotes - [^1]: This sentence is a footnote

Block quote

> "You miss 100% of the shots you do not take. - Wayne Gretsky" - Michael Scott

Unordered lists:

- apple
- pear
- orange
- bear
  # - orange bear
  # - apple pear

Ordered lists:

1. Apple
2. Pear
3. Orange Bear

```

---

# Document Editors

You can change the way that you edit the document by using the **Source** or **Visual** tab on the editor toolbar.

### Source
```

- Allows you view the document in code view

Visual

- Allows you to view the document with markups
- Allows basic WSYWIG

Code Chunks

Code chunks allow you to include code from multiple languages into your narration.

You can insert a chunk code by:

- CTRL + ALT + I (PC)

- COMMAND + OPTIONS + I (MAC)

- Use **Add Chunk** command in editor toolbar

Let's add a code chunk that allows us to see the data set mtcars

Running a Code Chunk

You can run a code chunk by:

- CTRL + SHIFT + ENTER (PC)
- COMMAND + SHIFT + ENTER (MAC)
- Run button in Code Chunk
- Run button in editor toolbar

Customize Chunk Code

Chunk Cog Wheel

- Allows you to rename the chunk so it can be easily located
- Allows you to set message and warning displays
- Allows you to adjust plot sizes

Let's rename our code chunk above

Manual Entry

```{r}
#| label: example manual entry
#| include: true

# I would encourage users to manually enter their labels. It is clearer for another user to view and cleaner for your presentation
```

```

#### Include

Include allows you to include or not include the chunk code in the final product when knitted.

> include =

**Let's create a chunk code that sets our current working directory but does not display the code or output in our final product using include. Hint: Set the working directory with the command - setwd()**

```{r}

```

Eval

Eval tells RStudio to either run or not run a code chunk when the document is knitted

> eval =

Let's install the CRAN package Tidyverse. But since this is a one time operation, let's preface that this code is not run when the document is knitted.

```{r}

```

Message

Some commands, like loading a package, will display messages after the code is run.
You can choose whether or not you want the message to be displayed in the knitted documents

> message =

Let's load the tidyverse package because we will need functions in it to run future code in the report. However, let's set it so the load message does not appear when the document is knitted but the code is displayed so a person who we are collaborating with can see that we are using that package.

```{r}

```

Echo

Echo allows you to show the output of the code that has been run, but not to show the code chunk when the document is knitted

> echo =

Let's get the results of a line of code without displaying the code in the report.

```{r}

```

Inline Code

You can include coding within the body of your work using inline code using the backtick (\`) button on your keyboard

Let's include inline code with the mean of the mpg variable in the mtcars dataset as well as the number of observations of the variable.

The average miles per gallon from the cars dataset is `r mean(mtcars$mpg)` based on `r nrow(mtcars)` observations.

Plots

In addition to adding code and outputs of the code, you can also set up data visualization to be displayed in your documents.

Here we will add a histogram of the dataset for the variable mpg.
And we will use R Markdown to determine the size of the figure as well as give it a captions.
Additionally, as we have learned already, we will use echo=FALSE to display only the output and not the code.

```{r}

#| label:
#| echo: true
#| message: false
#| fig.align: 'center'
#| fig.width: 10
#| fig.cap: "Figure 6.2: MPG Distribution"

library(tidyverse)
mtcars.hist <- ggplot(mtcars, aes(x=mpg))
mtcars.hist +
  geom_histogram(bins = 5, color = "yellow", fill = "skyblue") +
    labs(x = "Miles Per Gallon",
        y = NULL) +
  theme_classic()

```

Citations

R Markdown allows you to insert citations as well as work with citation managers such as [Zotero](https://databases.lib.wvu.edu/connect/1498075110) and [CiteDrive](https://www.citedrive.com/).
Once a citation is added to the document, it will automatically populate in a bibliography at the end of the document.

Insert Citations into your document:
```

- Visual Mode: Insert \> Citation
- Source Mode: [@auerbach2021] or (See [@grolemund])
- Visual Mode: \@ will show you available citations

When a Citation is generated:

- A new .bib file will be created in the current working directory and will be attached to the document in the YAML header

- The default format for the citations is **Chicago Turabian**.
  If you want to change the format you will need to download the proper .csl file and add it to your working directory and add a csl field to your YAML header

  - [Zotero Library](https://www.zotero.org/styles)

  - [Citation Visual Editor](https://editor.citationstyles.org/about/)

Let's add APA 7th Ed. Citation Format to our Working Directory and YAML header

Let's try and find and enter the citation for the [following article](https://pubmed.ncbi.nlm.nih.gov/34303462/)

- 10.1016/j.jvs.2021.03.055

References