

Data Visualization with GGPlot

Matt Steele

Resources

- [Tidyverse Documentation](#)
- [O'Reilly Learning Platform](#)
 - R for Data Science, 2nd Edition
 - R Programming for Statistics and Data Science

Recap

- Functions and Arguments
- Objects
- Run Code

Working Directory

The working directory in R is the folder where you are working. Hence, it's the place (the environment) where you have to store your files of your project in order to load them or where your R objects will be saved.

Session > Set Working Directory > Choose Directory

```
1 getwd() # show current directory that you are in
2
3
4 setwd("path/to/your/directory") # sets the working directory
```

GGPlot

visualize your data with ggplot

- ggplot package

```
1 library(tidyverse)
```

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).

Grammar of Graphics

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

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

start by loading the data set we are going to work on

```
1  scotus <- read_csv("scotus_approval.csv")
2
3  scotus
4
5
6  # let's just view the pollster YouGov using the filter function
7
8  scotus_yg <- scotus |>
9    filter(pollster == "YouGov")
10
11  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

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

Geometric Objects (geom_*)

finally we are going to choose the plot that we want to use for our visualization using the `geom_*` element

- [Geom Cheatsheet](#)

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

Theme: Color, Geom Size, Transparency

fill = or color = change geom color

size = change geom size

alpha = change geom transparency

```
1 # add color
2
3 scotus.line +
4   geom_line(color = "coral")
5
6 # change the size
7
8 scotus.line +
9   geom_line(color = "coral", size = 2)
```

Theme: Labels

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

```
1 scotus.line +  
2   geom_line(color = "coral", size = 2) +  
3   labs(  
4     title = "SCOTUS Approval",  
5     subtitle = "2023",  
6     caption = "polls from YouGov",  
7     y = "Approval",  
8     x = NULL  
9   )
```

Themes: Built-in

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

```
1 scotus.line +  
2   geom_line(color = "coral", size = 2) +  
3   labs(  
4     title = "SCOTUS Approval",  
5     subtitle = "2023",  
6     caption = "polls from YouGov",  
7     y = "Approval",  
8     x = NULL  
9   ) +  
10  theme_minimal()
```

Themes: Customize

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

```
1 scotus.line +  
2   geom_line(color = "coral", size = 2) +  
3   labs(  
4     title = "SCOTUS Approval",  
5     subtitle = "2023",  
6     caption = "polls from YouGov",  
7     y = "Approval",  
8     x = NULL  
9   ) +  
10  theme_minimal() +  
11  theme(plot.title = element_text(size = 20, color = "navy"))
```

Theme: Scales

the **scales element** allows you to fine-tune and

- the **scale_x_date** element allows you to adjust
- **Date Formats - strftime**

```
1 scotus.line +  
2   geom_line(color = "coral", size = 2) +  
3   labs(  
4     title = "SCOTUS Approval",  
5     subtitle = "2023",  
6     caption = "polls from YouGov",  
7     y = "Approval",  
8     x = NULL  
9   ) +
```

Smoothed Lines

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

```
1 scotus.line +  
2   geom_smooth(color = "coral", size = 2) +  
3   labs(  
4     title = "SCOTUS Approval",  
5     subtitle = "2023",  
6     caption = "polls from YouGov",  
7     y = "Approval",  
8     x = NULL  
9   ) +  
10  theme_minimal() +  
11  theme(plot.title = element_text(size = 20, color = "coral")) +  
12  scale_x_date( date_breaks = "6 weeks",  
13               date_labels = "%b %d")
```

Export your plot

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

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

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

Histogram Graph

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

- set data, aesthetics, and geom

```
1 # load demographics data frame
2
3 demo <- read_csv("demographics.csv")
4
5 # let's look at the distribution of the age variable by creating a histogram
6
7 demo.hist <- ggplot(demo, aes(age))
8
9 demo.hist +
10   geom_histogram()
```

Binning

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

```
1 # number of bins to use
2
3 demo.hist +
4   geom_histogram(bins = 10)
5
6 # length of a bins
7
8 demo.hist +
9   geom_histogram(binwidth = 15)
```

Theme: Color, Geom Size, Transparency

fill = or color = change geom color

size = change geom size

alpha = change geom transparency

```
1 demo.hist +  
2   geom_histogram(bins = 25,  
3                   color = "coral",  
4                   fill = "skyblue",  
5                   alpha = .5) +  
6   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

Multiple Geoms

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

- Example: the `geom_vline/geom_hline` element allows you to add a reference line to your plot

```
1 # add a reference line
2
3 demo.hist +
4   geom_vline(xintercept = 40, color = "navy", size = 3) +
5   geom_histogram(bins = 25, color = "coral", fill = "skyblue", alpha = .5)
6   theme_light()
```


Faceting

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

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

```
1 demo.bar <- ggplot(demo, aes(carcat))  
2  
3 demo.bar +  
4   geom_bar()
```

Bar Graph

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

```
1 demo.col <- ggplot(demo, aes(carcat, income))  
2  
3 demo.col +  
4   geom_col()
```

Reorder Plot

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

```
1 demo.col <- ggplot(demo, aes(fct_reorder(carcat, income), income))  
2  
3 demo.col +  
4   geom_col()
```

Add Additional Variable

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

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

Theme: Scale Color Palette

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

- Color Brewer

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

Theme: Scale Numeric Axis

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

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

