Data Visualization and Presentation with R and RStudio

Yiheng Wu, Jeff Werst, Matt Steele

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")
```

```r
# 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

```r
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

```r
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

```r
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")
```

```r
scotus.line +
  geom_line(color = "coral", size = 2) +
  scale_x_date(date_breaks = "1 month", date_labels = "%b %Y") +  # e.g., "Jan 2023"
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
    x = NULL
  ) +
  theme_minimal()


scotus.line +
  geom_line(color = "coral", size = 2) +
  scale_x_date(date_breaks = "3 months", date_labels = "%b %Y") +  # e.g., "Jan 2023", "Apr 2023"
  labs(
    title = "SCOTUS Approval",
    subtitle = "2023",
    caption = "polls from YouGov",
    y = "Approval",
```

```
    x = NULL
  ) +
  theme_minimal()
```

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()
```

```
demo.hist +
  geom_histogram(binwidth=1, fill = "skyblue", alpha=0.8) +
  theme_light()+
  scale_y_continuous(expand = c(0, 0))+
  labs(
    title = "Histogram Showing the Count of Population at Each Age",
    x = "Age",
    y = "Count"
  )
```

```
#install.packages("gridExtra")
library(gridExtra)

#install.packages("patchwork")
library(patchwork)



# Create plots
p1 <- ggplot(demo, aes(age)) +
  geom_histogram(binwidth = 1, fill = "skyblue", alpha = 0.8) +
  theme_light() +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Population Count by Age",subtitle='For each Age (binwidth=1)', x = "Age", y = "Count")
```

```r
p2 <- ggplot(demo, aes(age)) +
  geom_histogram(binwidth = 5, fill = "skyblue", alpha = 0.8) +
  theme_light() +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Population Count by Age",subtitle='5-Year Intervals (binwidth=5)', x = "Age", y = "Count")

p3 <- ggplot(demo, aes(age)) +
  geom_histogram(binwidth = 10, fill = "skyblue", alpha = 0.8) +
  theme_light() +
  scale_y_continuous(expand = c(0, 0)) +
  labs(title = "Population Count by Age",subtitle='10-Year Intervals (binwidth=10)', x = "Age", y = "Count")

# Arrange in 1 row and 3 columns
# grid.arrange(p1, p2, p3, nrow = 1)

p1 + p2 + p3
p1
```

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

```r
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

```r
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

```r
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

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

#scale_y_continuous(labels = label_number(scale_cut = cut_short_scale()))
#scale_y_continuous(labels = scales::label_number_si())
```

```
#install.packages('plotly')
library(plotly)

demo_summary <- demo |>
  group_by(carcat, ed) |>
  summarise(
    total_income = sum(income, na.rm = TRUE),
    n_people = n(),
    .groups = "drop"
```

```
  )

p <- ggplot(demo_summary, aes(
  x = fct_reorder(carcat, total_income, .fun = sum),
  y = total_income,
  fill = ed,
  text = paste0(
    "Car category: ", carcat,
    "<br>Education: ", ed,
    "<br>Sum of people: ", n_people,
    "<br>Sum of income: $", comma(total_income)
  )
)) +
  geom_col() +
  scale_fill_brewer(palette = "Pastel1") +
  scale_y_continuous(labels = label_comma()) +
  labs(x = "Car Category", y = "Income", fill = "Education")

ggplotly(p, tooltip = "text")
```