Data Visualization in R

Dominic Bordelon, Research Data Librarian, University of Pittsburgh Library System

September 19, 2022

# Data Visualization in R

## Agenda

1. About the presenter
2. ggplot2 and the grammar of graphics
3. Scatter plots (x-y relationships)
4. Histograms (distributions)
5. Visualizing amounts and proportions
6. Line graphs
7. Labeling your plot
8. Themes
9. Saving your work
10. Resources to keep learning

Today we are using the following packages:

#install.packages(c("tidyverse", "naniar", "palmerpenguins", "RColorBrewer", "ggthemes"))  
library(tidyverse) # includes ggplot2  
library(naniar) # visualize missingness  
library(palmerpenguins) # penguin data  
library(RColorBrewer) # ColorBrewer palettes  
library(ggthemes) # extra themes

Valuable cheat sheets: [ggplot2](https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-visualization.pdf) by RStudio, [ggplot2 Theme Elements](https://henrywang.nl/wp-content/uploads/2020/04/ggplot2-themes.pdf) by Henry Wang

## About the presenter

### Services

* Consultations
* Presentations
* Skills workshops

### Areas of Support

* data management planning for research projects
* making sense of and navigating software tools related to research: R, python, git and GitHub
* advising on Open (Science, Access, Source, etc.)

### Interests

* computer programming education
* computer programming for research; research software engineering (RSE)
* science and technology studies (STS)
* ecology (I am a part-time returning undergraduate)

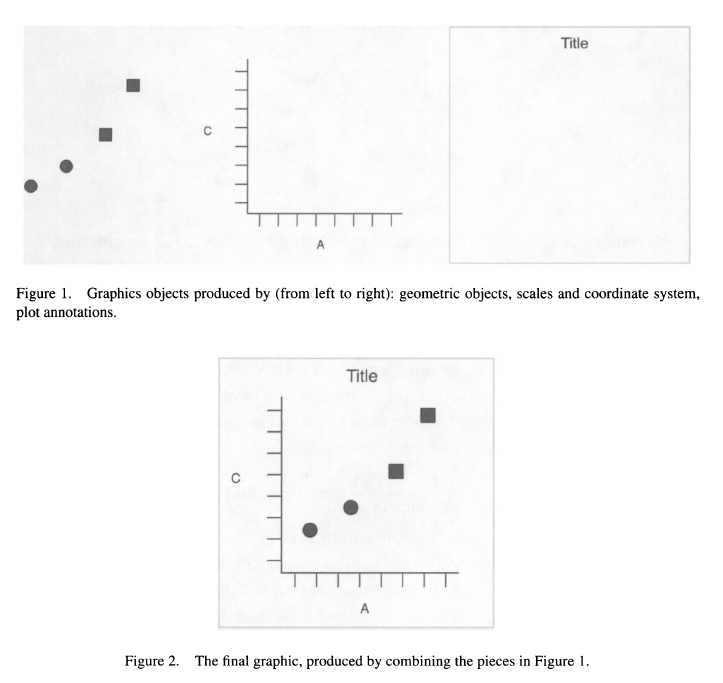
## “But I don’t use R…”

If you do your analysis in Python or Excel (for example), you can still plot the results in R. Simply save your results as an .xlsx or .csv file.

Alternatively, data viz can be your entry to learning more about R!

## ggplot2 and the grammar of graphics

ggplot2 is conceptually based in a “layered grammar of graphics” as described in Hadley 2010:



Wickham, Hadley. 2010. “A Layered Grammar of Graphics.” Journal of Computational and Graphical Statistics 19 (1): 3–28. <https://doi.org/10.1198/jcgs.2009.07098.>

### The “tidy” data assumption

ggplot2 assumes that your data are “tidy,” meaning:

* one observation per row
* one variable per column
* one value per cell

The data we’re using today are already tidy. If you’d like to learn more about this concept, you can join me for [Data Wrangling](https://pitt.libcal.com/calendar/today/rdatawrangling) next week, or check out [Wickham (2014)](https://doi.org/10.18637/jss.v059.i10).

### The “code template” for a plot

Minimal:

# ggplot(data = DATA) +  
# GEOM\_FUNCTION(mapping = aes(MAPPINGS))

The ggplot() function creates a basic plot object. DATA is a data frame whose contents you want to plot. Then we add one or more layer functions, using the + operator. You need at least one geom\_ layer in order to see a result. Examples of geoms are geom\_histogram() and geom\_point() (for points, e.g. scatter plots).

The aes() function creates one or more aesthetic mappings, i.e., a relation between a variable in the data (e.g., time elapsed) and a visual feature/property of the plot (e.g., the x axis). Mappings take the format aesthetic = variable, e.g., x = time\_elapsed or color = species (the categorical variable species is indicated using different colors).

A more complete template:

# ggplot(data = DATA, mapping = aes(MAPPINGS)) +  
# GEOM\_FUNCTION(mapping = aes(MAPPINGS),  
# stat = STAT,  
# position = POSITION)  
# COORD\_FUNCTION() +  
# FACET\_FUNCTION() +   
# SCALE\_FUNCTION() +  
# THEME\_FUNCTION()

Now let’s take a look at some popular types of plots and how to create them in ggplot.

## Scatter plots (x-y relationships)

data(penguins) # load the "penguins" dataset into our session  
  
glimpse(penguins) # what does our data frame look like?  
# you can also read more at:   
?penguins

ggplot(data = penguins) +  
 geom\_point(aes(x=body\_mass\_g, y=bill\_length\_mm))  
  
ggplot(data = penguins) +  
 geom\_miss\_point(aes(x=body\_mass\_g, y=bill\_length\_mm))

### Adding a z variable

# discrete, color:  
ggplot(penguins, aes(body\_mass\_g, bill\_length\_mm, color=species)) +  
 geom\_point()  
  
# discrete, shape:  
ggplot(penguins, aes(body\_mass\_g, bill\_length\_mm, shape=species)) +  
 geom\_point()  
  
# continuous, bubble:  
ggplot(penguins, aes(body\_mass\_g, bill\_length\_mm, size=bill\_depth\_mm)) +  
 geom\_point()

### Handling overplotting

glimpse(diamonds)  
  
ggplot(diamonds, aes(x, price)) + geom\_point()  
  
# modifying alpha (opacity):  
ggplot(penguins, aes(body\_mass\_g, bill\_length\_mm)) + geom\_point(alpha=0.5)  
ggplot(diamonds, aes(x, price)) +   
 geom\_point(alpha=0.3)  
  
# adding a regression line with confidence interval:  
ggplot(penguins, aes(body\_mass\_g, bill\_length\_mm)) +   
 geom\_point() +  
 geom\_smooth()  
ggplot(diamonds, aes(x, price)) +  
 geom\_point() +  
 geom\_smooth()  
  
# hex bins:  
ggplot(penguins, aes(body\_mass\_g, bill\_length\_mm)) +  
 geom\_hex()  
ggplot(diamonds, aes(x, price)) +  
 geom\_hex()

## Histograms (distributions)

# histogram:  
ggplot(penguins, aes(x=bill\_length\_mm)) +  
 geom\_histogram()  
  
# density plot:  
ggplot(penguins, aes(x=bill\_length\_mm)) +  
 geom\_density()  
  
# histogram and density together:  
ggplot(penguins, aes(x=bill\_length\_mm)) +  
 geom\_histogram(aes(y=..density..)) +  
 geom\_density()

### Grouped distributions: violins, boxplots, overlapping densities

# overlapping density plots:  
ggplot(penguins, aes(x=bill\_length\_mm, fill=species)) +  
 geom\_density(alpha=0.2)  
  
# violin plot:  
ggplot(penguins, aes(y=bill\_length\_mm, x=species)) +  
 geom\_violin()  
  
# box plot:  
ggplot(penguins, aes(y=bill\_length\_mm, x=species)) +  
 geom\_boxplot()  
  
# comparison using faceting:  
ggplot(penguins, aes(y=bill\_length\_mm)) +  
 geom\_histogram() +  
 facet\_wrap(vars(species))

## Visualizing amounts and proportions

Bars: counts of cases

ggplot(penguins, aes(x=species)) +  
 geom\_bar()  
  
ggplot(penguins, aes(y=species)) +  
 geom\_bar()  
ggplot(diamonds, aes(y=clarity)) +  
 geom\_bar()

Bars of summary statistics

ggplot(penguins, aes(x=species, y=bill\_length\_mm)) +  
 geom\_bar(stat="summary", fun="mean")   
ggplot(diamonds, aes(x=price, y=clarity)) +  
 geom\_bar(stat="summary", fun="mean")  
  
ggplot(penguins, aes(y=species, x=bill\_length\_mm)) +  
 geom\_bar(stat="summary", fun="mean") +  
 scale\_y\_discrete(limits=c("Chinstrap", "Gentoo", "Adelie"))

### Adding error bars

Although geoms are available for error bars, you must perform the calculations yourself. This means that you need to modify the data frame—adding appropriate variables (columns) for mean, standard deviation, etc.—prior to plotting. We haven’t learned how to do that yet, but we will next week.

When you manually calculate summary statistics, such as the mean, rather than asking ggplot to do it, you will also need to change geom\_bar’s stat argument to "identity" instead of "summary". You are telling ggplot, “please plot the exact numbers I give you, rather than a summary of them.”

penguins %>%   
 drop\_na(bill\_length\_mm) %>%   
 group\_by(species) %>%   
 summarize(bill\_mean = mean(bill\_length\_mm),  
 bill\_sd = sd(bill\_length\_mm)) %>%   
 ggplot(aes(x=species, y=bill\_mean)) +  
 geom\_bar(stat="identity") +  
 geom\_errorbar(mapping=aes(ymax=bill\_mean+bill\_sd,   
 ymin=bill\_mean-bill\_sd),   
 width=0.2)  
  
diamonds %>%   
 drop\_na(price, clarity) %>%   
 group\_by(clarity) %>%   
 summarize(n=n(),  
 price\_mean = mean(price),  
 price\_sd = sd(price),  
 price\_se = price\_sd/sqrt(n)) %>%   
 ggplot(aes(x=price\_mean, y=clarity)) +  
 geom\_bar(stat="identity") +  
 geom\_errorbarh(aes(xmin=price\_mean-price\_se,   
 xmax=price\_mean+price\_se),   
 height=0.2)

### Grouped bars

Choose an additional variable, to see and compare subgroups.

ggplot(penguins, aes(x=species, y=bill\_length\_mm, fill=island)) +  
 geom\_bar(stat="summary", fun="mean", position="dodge")  
  
ggplot(diamonds, aes(x=cut, fill=clarity, y=price)) +  
 geom\_bar(stat="summary", fun="mean", position="dodge")

### Stacked bars and pies

Stacked bars show each group’s relative proportion in the whole. You can also opt to standardize bars to the same height. To make a stacked bar, add a variable mapped to the fill aesthetic, with a position of stack.

# stacked bar chart  
ggplot(penguins, aes(x=island, fill=species)) +  
 geom\_bar()  
   
ggplot(diamonds, aes(y=clarity, fill=cut)) +  
 geom\_bar()

The familiar pie chart, in ggplot2 terms, is a stacked bar chart with polar coordinates.

ggplot(penguins, aes(x="", fill=species)) +  
 geom\_bar(width=1, color="white") +  
 coord\_polar(theta="y", start=0, direction=-1) +  
 theme\_void()  
  
ggplot(diamonds, aes(x="", fill=clarity)) +  
 geom\_bar(width=1, color="white") +  
 coord\_polar(theta="y", start=0, direction=-1) +  
 theme\_void()

## Line graphs

Line graphs tend to work best with identity data.

ggplot(economics, aes(x=date, y=unemploy)) +  
 geom\_line()  
  
penguins %>%   
 drop\_na(body\_mass\_g) %>%   
 group\_by(species, year) %>%   
 summarize(mass\_mean=mean(body\_mass\_g)) %>%   
 ggplot(aes(x=year, y=mass\_mean, color=species)) +  
 geom\_line() +  
 geom\_point() +  
 scale\_x\_continuous(breaks=2007:2009)  
  
storms %>%   
 drop\_na(wind) %>%   
 group\_by(year) %>%   
 summarize(n=n(),  
 wind\_mean=mean(wind),   
 wind\_sd=sd(wind),   
 wind\_se=wind\_sd/sqrt(n)) %>%   
 ggplot(aes(x=year, y=wind\_mean)) +  
 geom\_line() +  
 geom\_point()   
# + geom\_errorbar(aes(ymin=wind\_mean-wind\_se,  
# ymax=wind\_mean+wind\_se),  
# width=0.75)

## Labeling your plot

penguins %>%   
 drop\_na(body\_mass\_g, bill\_length\_mm, species) %>%   
 ggplot(aes(x=body\_mass\_g,   
 y=bill\_length\_mm,   
 color=species)) +  
 geom\_point() +  
 geom\_smooth(method="lm") +  
 labs(title="Penguin bill length correlates with body mass",  
 x="Body mass (g)",  
 y="Bill length (mm)",  
 color="Species")

## Themes

Themes style the plot overall. A theme will always be a function. To use a theme with your plot, use + and the function call, for example, ...geom\_point() + theme\_bw() . Here are some sources of themes:

* themes included with ggplot2, e.g., theme\_minimal()
* the [ggthemes](https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/) package

penguins %>%   
 drop\_na(body\_mass\_g, bill\_length\_mm, species) %>%   
 ggplot(aes(x=body\_mass\_g,   
 y=bill\_length\_mm,   
 color=species)) +  
 geom\_point() +  
 geom\_smooth(method="lm", se=FALSE) +  
 labs(title="Penguin bill length correlates with body mass",  
 x="Body mass (g)",  
 y="Bill length (mm)",  
 color="Species") +  
 theme\_bw() +  
 scale\_color\_brewer(type="qual")  
  
ggplot(diamonds, aes(y=clarity, fill=cut)) +  
 geom\_bar() +  
 theme\_bw() +  
 scale\_fill\_brewer(type="seq")

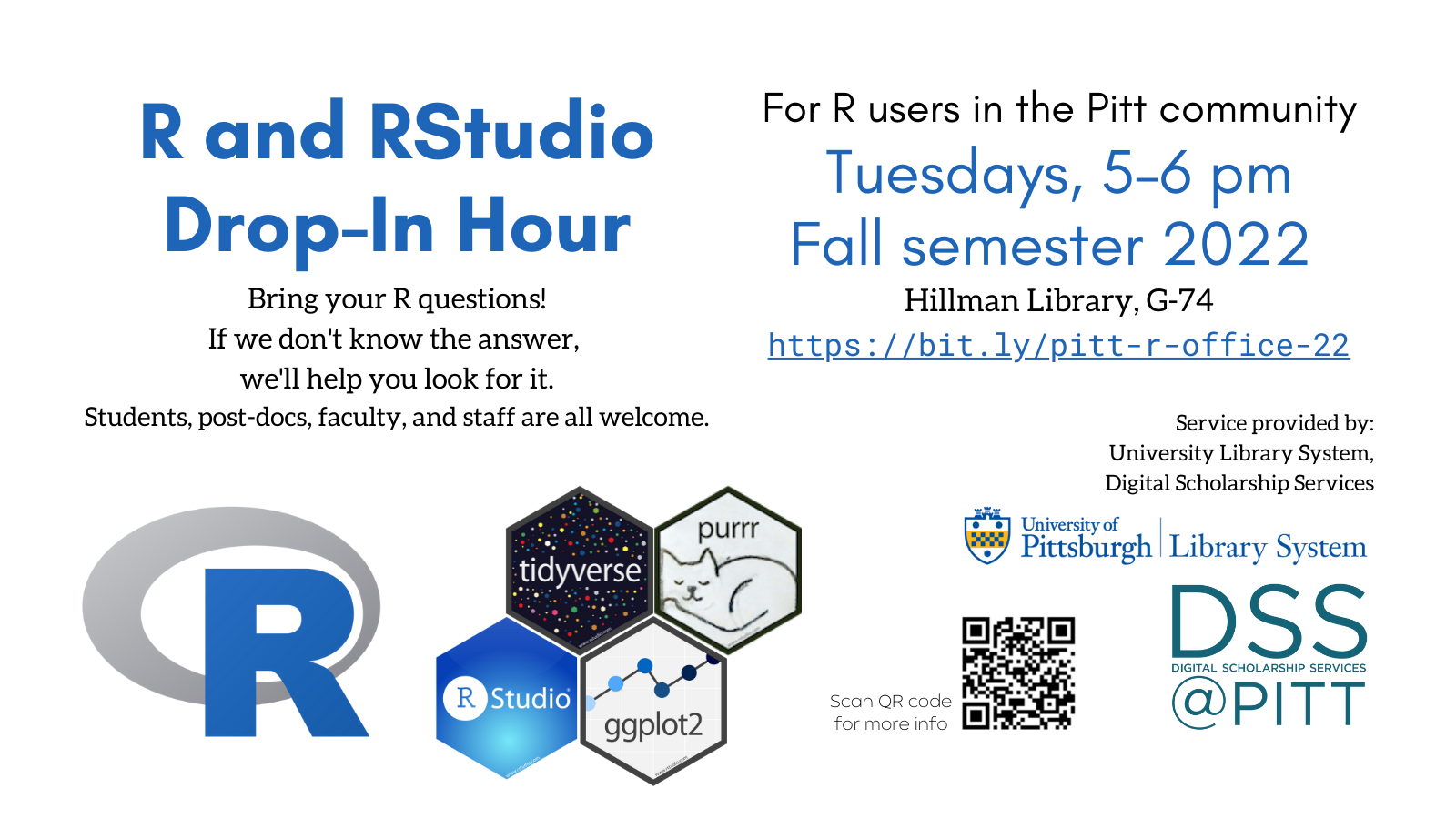
## Saving your work

* Using notebooks (.Rmd or .qmd files) is highly recommended!
* You can save plots as image files (PNG, SVG, PDF, etc.) using ggsave()
* Any R data structure (plot object, data frame…) can also be saved to a file using saveRDS()

pens\_plot <- penguins %>%   
 drop\_na(body\_mass\_g, bill\_length\_mm, species) %>%   
 ggplot(aes(x=body\_mass\_g,   
 y=bill\_length\_mm,   
 color=species)) +  
 geom\_point() +  
 geom\_smooth(method="lm", se=FALSE) +  
 labs(title="Penguin bill length correlates with body mass",  
 x="Body mass (g)",  
 y="Bill length (mm)",  
 color="Species") +  
 theme\_bw() +  
 scale\_color\_brewer(type="qual")  
  
ggsave("penguins.png", pens\_plot)

## Resources to continue learning

* Next week: [Data Wrangling in R](https://pitt.libcal.com/calendar/today/rdatawrangling?hs=a)
* [O’Reilly ebooks](https://learning.oreilly.com/library/view/temporary-access/)
* [The Big Book of R](https://www.bigbookofr.com/)
* Check out [#TidyTuesday](https://twitter.com/search?q=%23tidytuesday) on Twitter



[dbordelon@pitt.edu](mailto:dbordelon@pitt.edu)