Data Visualization

Fall R ’23

Dominic Bordelon, Research Data Librarian, ULS

## Agenda

1. [The grammar of graphics](#the-grammar-of-graphics)
2. [Plotting distribution](#plotting-distribution)
3. [Scatter plots](#scatter-plots)
4. [Comparing categories](#comparing-categories)
5. [Heat maps of two-way tables](#heat-maps-of-two-way-tables)
6. [Time series](#time-series)
7. [Small multiples](#small-multiples)
8. [Labeling, theming, and annotation](#labeling-theming-and-annotation)

## About the trainer

**Dominic Bordelon, Research Data Librarian**  
University Library System, University of Pittsburgh  
[dbordelon@pitt.edu](mailto:dbordelon@pitt.edu)

Services for the Pitt community:

* Consultations
* Training (on-request and via public workshops)
* Talks (on-request and publicly)
* Research collaboration

Support areas and interests:

* Computer programming fundamentals, esp. for data processing and analysis
* Open Science and Data Sharing
* Data stewardship/curation
* Research methods; science and technology studies

## Fall R Series

| # | Date | Title |
| --- | --- | --- |
| 1 | 8/29 | Getting Started with Tabular Data |
| 2 | 9/5 | Working with Data Frames |
| **3** | **9/12** | **Data Visualization** |
| 4 | 9/19 | Inference and Modeling Intro |
| 5 | 9/26 | Machine Learning Intro |

|  |
| --- |
|  |

## The ggplot2 package

“ggplot2 is a system for declaratively creating graphics, based on [The Grammar of Graphics](https://www.amazon.com/Grammar-Graphics-Statistics-Computing/dp/0387245448/ref=as_li_ss_tl). You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.”

|  |
| --- |
|  |

## …and using penguins examples

install.packages("palmerpenguins")

library(palmerpenguins)  
  
# load palmerpenguins' data into your environment:  
data(penguins)  
names(penguins)

[1] "species" "island" "bill\_length\_mm"   
[4] "bill\_depth\_mm" "flipper\_length\_mm" "body\_mass\_g"   
[7] "sex" "year"

palmerpenguins is one of many examples of an R package which functions as a downloadable data set.

## The grammar of graphics

* A plot is constructed in layers:
  + data
  + aesthetics (axes, encodings)
  + scale (axis labels, color coding)
  + geometric objects (bar, scatter, heatmap tiles, etc.)
  + facets
  + statistical summaries (e.g., highlighted mean; smoother)
  + annotations
  + coordinate system (Cartesian, polar, or map projection)
  + theme

|  |
| --- |
| Wilkinson 1999 |

## Aesthetic mapping

An aesthetic mapping, created with aes(), associates an aesthetic property of the plot with a variable in your data.

penguins example: I want a scatter plot of flipper length on x and bill length on y. The aesthetic mapping would be:  
aes(x = flipper\_length\_mm, y = bill\_length\_mm)

Common aesthetics:

* x, y
* color, fill
* linetype
* size
* shape (of points)

|  |
| --- |
| Commonly used aesthetics (source: Wilke 2023) |

# Plotting distribution

## Histograms and density plots

geom\_histogram(mapping, binwidth) and geom\_density(mapping)

A histogram “bins” a variable’s values and charts how many values are in each bin, giving a sense of central tendency and spread. A density plot works similarly, but it produces a smooth curve while sacrificing countable units.

# distribution of body mass:  
penguins %>%   
 ggplot() +  
 geom\_histogram(mapping = aes(x = body\_mass\_g))  
  
# remove NA values, and specify bin width:  
penguins %>%   
 drop\_na(body\_mass\_g) %>%   
 ggplot() +  
 geom\_histogram(mapping = aes(x = body\_mass\_g),  
 binwidth = 100)  
  
# density plot:  
penguins %>%   
 ggplot() +  
 geom\_density(aes(x = body\_mass\_g))  
  
# layering histogram and density plot  
# note that 1) mapping is specified in ggplot(), and inherited by geoms; 2) histogram's y mapping needs adjusting to be compatible with density plot  
penguins %>%   
 drop\_na(body\_mass\_g) %>%   
 ggplot(aes(x = body\_mass\_g)) +  
 geom\_histogram(mapping=aes(y=after\_stat(density)), binwidth = 100) +  
 geom\_density()

## Comparing distributions: violin, box

geom\_violin(), geom\_boxplot(). Mapping a categorical variable onto an axis will make a violin or box plot for each level of the variable.

A violin plot mirrors a density plot across the axis. A box plot marks interquartile range (white box), median (line inside the box), tails, and outliers (individual points). (where outlier has a distance from the median)

# distribution of body mass across species:  
penguins %>%   
 ggplot() +  
 geom\_violin(aes(x = body\_mass\_g,  
 y = species))  
  
penguins %>%   
 ggplot() +  
 geom\_boxplot(aes(x = body\_mass\_g,  
 y = species))

# Scatter plots

## geom\_point() plots points

A scatter plot compares continuous variables and on a Cartesian plane. geom\_point() can take several aesthetics (size, shape, color), but be careful not to overload your plot with information.

penguins %>%   
 ggplot() +  
 geom\_point(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm))  
  
# encode species as color:  
penguins %>%   
 ggplot() +  
 geom\_point(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm,  
 color = species))

## geom\_smooth() adds a smoother to a scatter plot

A natural next step to a scatter plot is to fit a regression model. geom\_smooth() does this job; specify the modeling function you want with the method argument. The default methods are loess, local polynomial regression fitting (when ), or gam, generalized additive model with restricted maximum likelihood (when ). See ?stats::loess and ?mgcv::gam for more info on these modeling functions.

(If you have fit your own model outside of ggplot, you should instead use predict() to generate points to plot with geom\_line().)

# using default loess:  
penguins %>%   
 ggplot(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm)) +  
 geom\_point() +  
 geom\_smooth()  
  
# using a linear model, lm():  
penguins %>%   
 ggplot(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm)) +  
 geom\_point() +  
 geom\_smooth(method="lm")  
  
# encode species as color:  
penguins %>%   
 ggplot(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm,  
 color = species)) +  
 geom\_point() +  
 geom\_smooth(method="lm")

# Comparing categories

## geom\_bar() makes bar charts of *counts*

penguins %>%   
 ggplot(aes(x = species)) +  
 geom\_bar()  
  
# note 2 variables, y axis, and dodge position  
# we also reorder the factors for aesthetic reasons   
# ...and note unexpected colors!  
penguins %>%  
 mutate(species = fct\_rev(fct\_infreq(species)),  
 sex = fct\_infreq(sex)) %>%   
 ggplot(aes(y = species, fill = sex)) +  
 geom\_bar(position = position\_dodge2(reverse=TRUE, preserve="single"))

## Bar charts of calculated statistics

Perform summary calculations using dplyr, then use geom\_bar() with stat = "identity" to plot the numbers as-is (default is "count"). ⚠ When plotting a summary statistic, one should also include error! geom\_errorbar(), geom\_errorbarh(), geom\_linerange()

Example: mean body mass for each species and gender, with standard error

penguins %>%  
 drop\_na(body\_mass\_g) %>%  
 group\_by(species) %>%   
 summarize(mean\_mass\_g = mean(body\_mass\_g),  
 standard\_error = sd(body\_mass\_g)/sqrt(n())  
 ) %>%   
 ggplot(aes(x = species, y = mean\_mass\_g)) +  
 geom\_bar(stat = "identity") +  
 geom\_errorbar(aes(ymax = mean\_mass\_g + standard\_error,  
 ymin = mean\_mass\_g - standard\_error), width = 0.5, color="blue")  
  
# note 2 variables, y axis, and dodge position  
# we also reorder the factors for aesthetic reasons   
# ...and note unexpected colors!  
penguins %>%  
 drop\_na(body\_mass\_g) %>%   
 group\_by(species, sex) %>%   
 summarize(mean\_mass\_g = mean(body\_mass\_g),  
 standard\_error = sd(body\_mass\_g)/sqrt(n())  
 ) %>%   
 ggplot(aes(y = species, fill = sex, x = mean\_mass\_g)) +  
 geom\_bar(stat="identity",  
 position = position\_dodge2()) +  
 geom\_errorbarh(aes(xmin = mean\_mass\_g - standard\_error,  
 xmax = mean\_mass\_g + standard\_error),  
 position = position\_dodge2())

## Stacked proportional bars and pies

Sometimes we want to compare *proportions* of category membership. With position = "fill", each bar has the same height, and the fill aesthetic works proportionately.

penguins %>%   
 ggplot(aes(x = species, fill=sex)) +  
 geom\_bar(position = "fill") +  
 scale\_y\_continuous(labels = scales::percent)

A pie chart in ggplot2 terms is a proportional bar cast onto a polar coordinate system.

penguins %>%   
 mutate(species = fct\_rev(fct\_infreq(species))) %>%   
 ggplot(aes(x = "", fill=species)) +  
 geom\_bar(position = "fill",  
 color = "white",  
 width = 1) +  
 coord\_polar("y", start=0) +  
 theme\_void()

# Heat maps of two-way tables

## Two-way tables

The *two-way table* or *contingency table* format shows the *conditional distribution* of observations among two categorical variables. In penguin terms, how many penguins of each species were observed on each of three islands?

penguins %>%   
 janitor::tabyl(species, island)

species Biscoe Dream Torgersen  
 Adelie 44 56 52  
 Chinstrap 0 68 0  
 Gentoo 124 0 0

## Heat maps of two-way tables

A graphical version of the previous table is given by geom\_tile(). However, you will need to group and summarize the data first.

penguins %>%   
 group\_by(species, island) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_tile(aes(y=species, x=island, fill=n))

# Time series

## geom\_line() connects point observations

Time series data are the most common application for line graphs. Since line graphs expect a single for each , I recommend using dplyr to generate the table that you need. The examples use n() as the summarizing function, but one could also use sum() of a variable (for example).

# observations over time  
penguins %>%   
 group\_by(year) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_line(aes(x=year, y=n))  
  
# a more interesting example, using tropical storm/hurricane data  
?storms  
storms %>%   
 group\_by(year) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_line(aes(x=year, y=n)) +  
 labs(y = "Storm observations")

## Categories over time

Consider geom\_line() for comparison, or geom\_area() for showing cumulative distribution.

# observations over time  
penguins %>%   
 group\_by(year, species) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_line(aes(x=year, y=n, linetype=species))  
  
# storms by hurricane category:  
# note: ~5-7 is the greatest number of categories you can color code before readers start having difficulty interpreting!  
storms %>%   
 mutate(category = as\_factor(category)) %>%  
 drop\_na(category) %>%   
 group\_by(year, category) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_line(aes(x=year, y=n, color=category)) +  
 labs(y = "Storm observations")  
  
storms %>%   
 mutate(category = as\_factor(category)) %>%  
 drop\_na(category) %>%   
 group\_by(year, category) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_area(aes(x=year, y=n, fill=category)) +  
 labs(y = "Storm observations")

# Small multiples

## Small multiples

Small multiples AKA faceting replicate a plot in columns and/or rows, with one plot for each level of some categorical variable.

Want to combine multiple plots into one figure? Check out the [patchwork](https://github.com/thomasp85/patchwork) package.

# one line graph for each storm category  
storms %>%   
 mutate(category = as\_factor(category)) %>%  
 drop\_na(category) %>%   
 group\_by(year, category) %>%   
 summarize(n = n()) %>%   
 ggplot() +  
 geom\_line(aes(x=year, y=n, color=category)) +  
 facet\_wrap(vars(category), ncol=3, nrow=2) +  
 labs(y = "Storm observations")  
  
# a scatter for each island, with groups colored  
penguins %>%   
 ggplot(aes(x=flipper\_length\_mm,   
 y=bill\_length\_mm,  
 color=species)) +  
 geom\_point() +  
 facet\_wrap(vars(island), nrow=2, ncol=2)

# Labeling, theming, and annotation

## labs() sets labels

penguins %>%   
 drop\_na(flipper\_length\_mm, bill\_length\_mm) %>%   
 ggplot(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm,  
 color = species)) +  
 geom\_point() +  
 geom\_smooth(method="lm") +  
 labs(title = "Gentoo tend to have the longest flippers",  
 x = "Flipper length (mm)",  
 y = "Bill length (mm)",  
 color = "Species",  
 caption = "Source: Gorman KB, Williams TD, Fraser WR (2014)")

## theme\_ layers change the plot’s look

Start typing theme\_ in RStudio to see available options, or install and attach the ggthemes package for more. There is also a theme() function for adjusting specific aspects of the plot.

Lastly, annotate() can add annotations to the plot, using the coordinate system.

penguins %>%   
 drop\_na(body\_mass\_g) %>%   
 ggplot() +  
 geom\_histogram(mapping = aes(x = body\_mass\_g),  
 binwidth = 100) +  
 annotate(geom="text", x = 6200, y = 7,  
 label="Biggest\n penguin") +  
 annotate(geom="segment", x = 6200, y = 5,  
 xend = max(penguins$body\_mass\_g, na.rm=TRUE),   
 yend = 1.5) +  
 theme\_bw()

## We can’t forget to save!

ggsave(filename, plot) where plot is a ggplot() object you have assigned. .png, .svg, or .pdf in the filename determines the output type.

Other arguments to use: width, height, units

my\_plot <- penguins %>%   
 drop\_na(flipper\_length\_mm, bill\_length\_mm) %>%   
 ggplot(aes(x = flipper\_length\_mm,   
 y = bill\_length\_mm,  
 color = species)) +  
 geom\_point()  
  
ggsave("my\_plot.png", my\_plot)

## Visualize *responsibly*

* Consider the validity and reliability of your measures
* Always cite any data you visualize which you did not collect yourself
* Design for accessibility
* Study best practices, and some of the pitfalls/abuses of viz, to keep your plot honest

|  |
| --- |
| Wilke 2019, available free-to-read at <https://clauswilke.com/dataviz/> |

# Wrap up

## Session in review

Today we learned about:

* the grammar of graphics and aesthetic mapping
* lots of plot types! for a variety of applications
* some easy ways of customizing plots

Join us next week for inference and modeling!