Data Visualization

Fall R ’23

Dominic Bordelon, Research Data Librarian, ULS

Agenda

1. The grammar of graphics
2. Plotting distribution
3. Scatter plots
4. Comparing categories
5. Heat maps of two-way tables
6. Time series
7. Small multiples
8. Labeling, theming, and annotation

About the trainer

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

R and RStudio Drop-In Hour

Bring your R questions!
If we don't know the answer,
we'll help you look for it.
Students, post-docs, faculty, and staff are all welcome.

For R users in the Pitt community

Tuesdays, 4:30-5:30 pm
Fall semester 2023

Hillman Library, Rm. 255

https://bit.ly/pitt-r-office-23

Service provided by:
University Library System,
Digital Scholarship Services

Scan QR code
for more info

The ggplot2 package

“ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. 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")
```

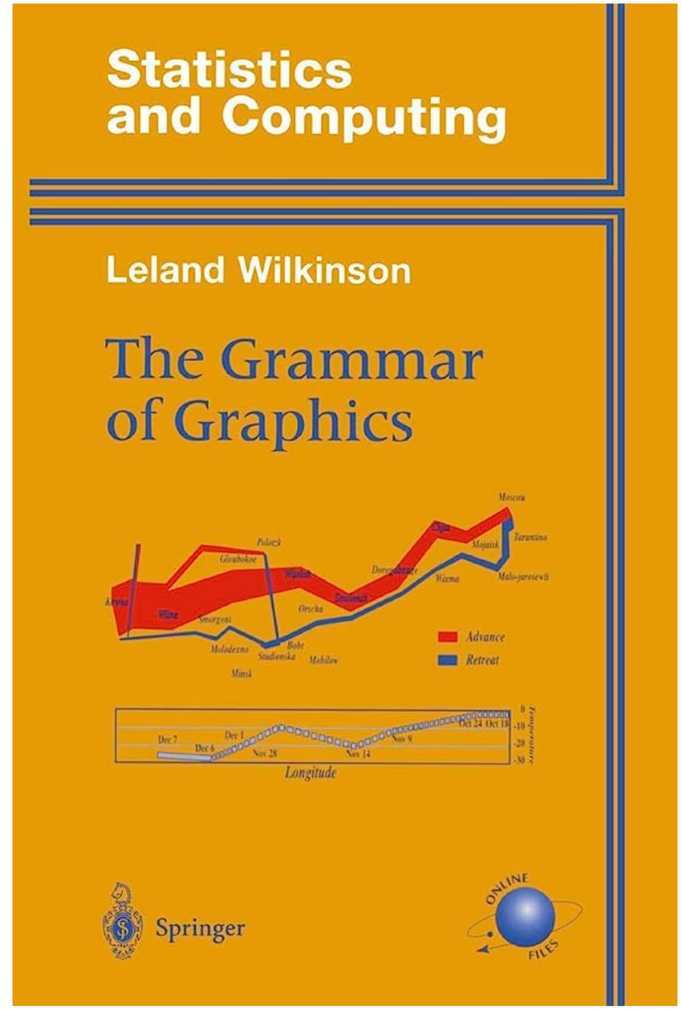
```
library(palmerpenguins)

# 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

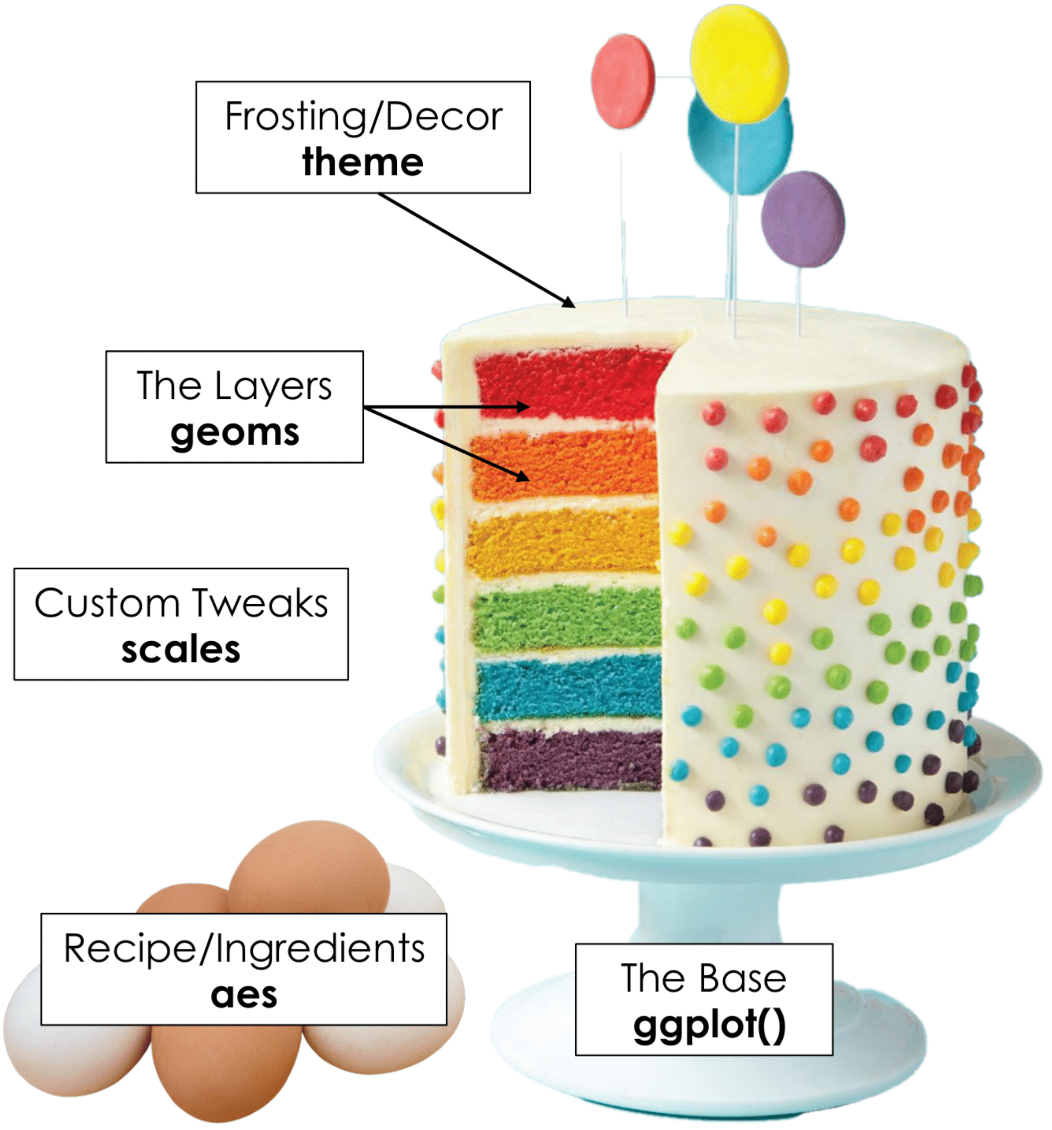
ggplot is a little bit like cake…

We *always* start by setting up the foundation with **ggplot()**

We specify our ingredients (data variables) with an **aes mapping**

We can create *layers* to our plot with **geoms**

We can style our ~~cake~~ ggplot with **themes.** We have out-of-the-box options, or we can go totally custom!



Source: Tanya Shapiro, https://twitter.com/tanya_shapiro/status/1576935152575340544

Aesthetic mapping

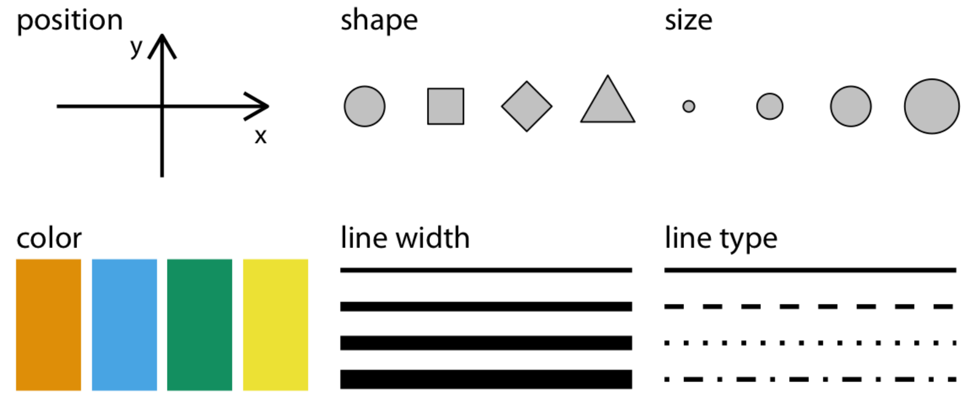
An aesthetic mapping, created with `aes()`, associates an aesthetic property of the plot with a variable in your data.

`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 adjus
penguins %>%
  drop_na(body_mass_g) %>%
  ggplot(aes(x = body_mass_g)) +
```

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 \(\gt 1.5 \times \mathrm{IQR}\) from the median)

```r
# 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 \(x\) and \(y\) 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 $n < 1000$), or `gam`, generalized additive model with restricted maximum likelihood (when $n \geq 1000$). 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()`.)

```r
# 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")

```

Comparing categories

`geom_bar()` makes bar charts of *counts*

```r
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",
```

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 \(y\) for each \(x\), 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 i
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` 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

```r
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!

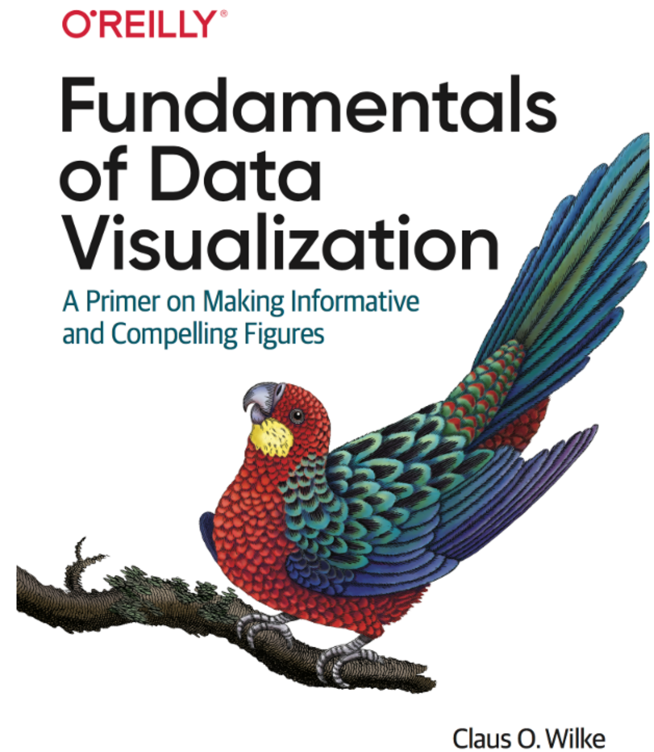
`ggsave(filename, plot)` where `plot` is a `ggplot()` object you have assigned. `.png`, `.svg`, or `.pdf` in the `filename` determines the output type.

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!