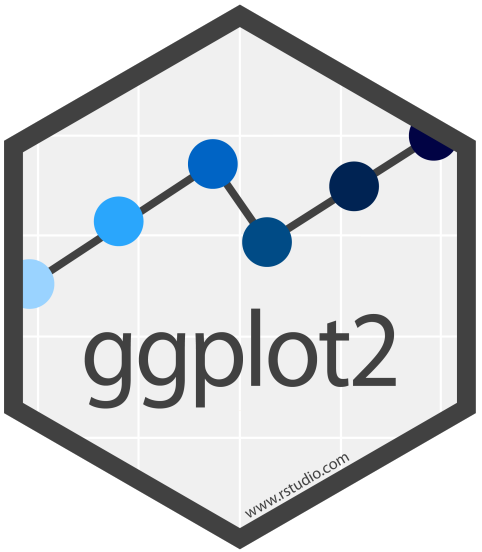
Detailed visualization with `ggplot()`

Basic Plots with `ggplot2`

There are built-in functions in R and standalone packages that allow for visualizations.

However, we will be using the ggplot2 package in this course, which is part of the tidyverse:

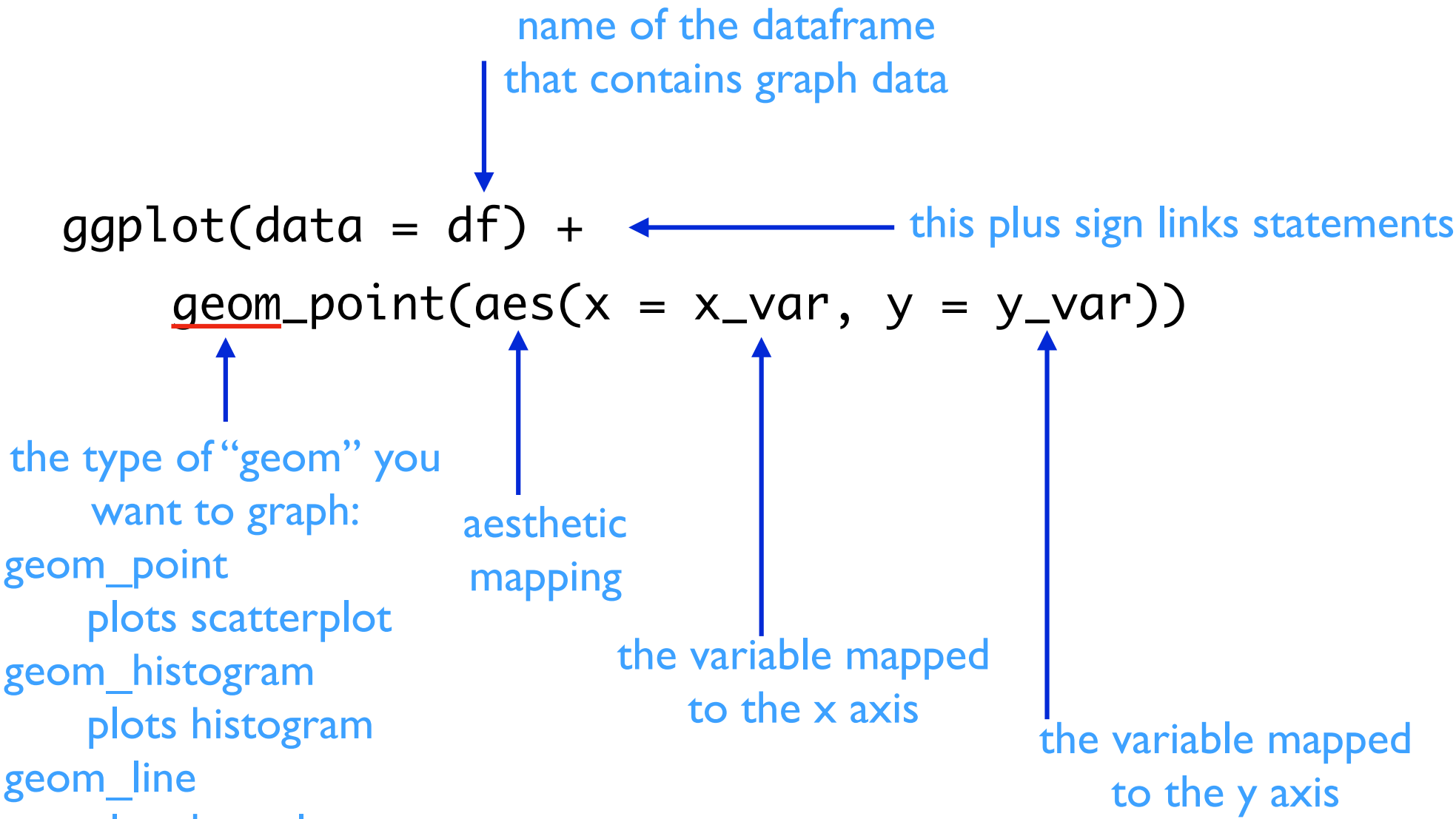
Why use `ggplot2`?



- Produces publication-ready graphs.
- Uses an underlying *grammar* to build graphs layer-by-layer.
- Easy to use without understanding the grammar.
- Once comfortable, allows the user to build graphs from concepts rather than memorizing options.

ggplot2 Structure

Every ggplot2 graph has the same structure, or grammar:



need geom to make a plot

What is a geom?

Each `geom` allows the user to plot a different *geometric object* in a layer on the canvas.

Complex plots in `ggplot2` can be constructed by layering more than one geom on top of each other (e.g. `geom_point()` and `geom_line()` are often used together)

The `geom` chosen determines the type of graph. Different geoms will produce different layers:

- `geom_point()` creates points (scatter plots)
- `geom_boxplot()` creates box plots
- `geom_histogram()` creates histograms

So many geoms!

`geom_tile()` for heatmaps

`geom_bar()` `geom_col()` for bar graphs

`geom_histogram()` for histograms

`geom_point()` for points or scatterplots

`geom_line()` for connecting observations with lines

`geom_pointrange()` for graphing points with error bars

`geom_ribbon()` for creating ribbons shaded area defined by two lines

`geom_boxplot()` for creating box plots

`geom_text()` for adding text labels

`geom_polygon()` for creating shapes (and maps)

No `geom_pie()` will ever exist!

depending on the purpose of the plot, you might want to include certain things
regardless ggplot has options for both

Exploration vs. Communication

Exploration

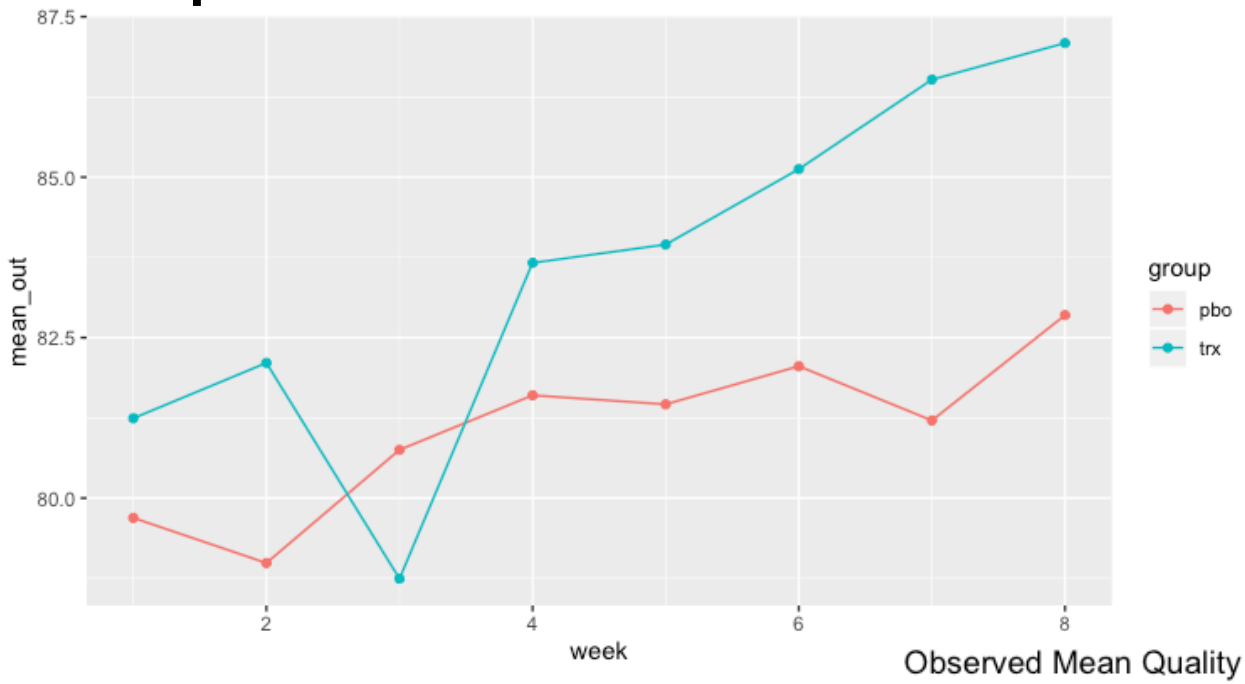
- Quick plots created during exploration of data
- Aesthetics are functional but not polished
- Labeling must communicate, but may be temporary
- Used to check distributions and evaluate assumptions

Communication

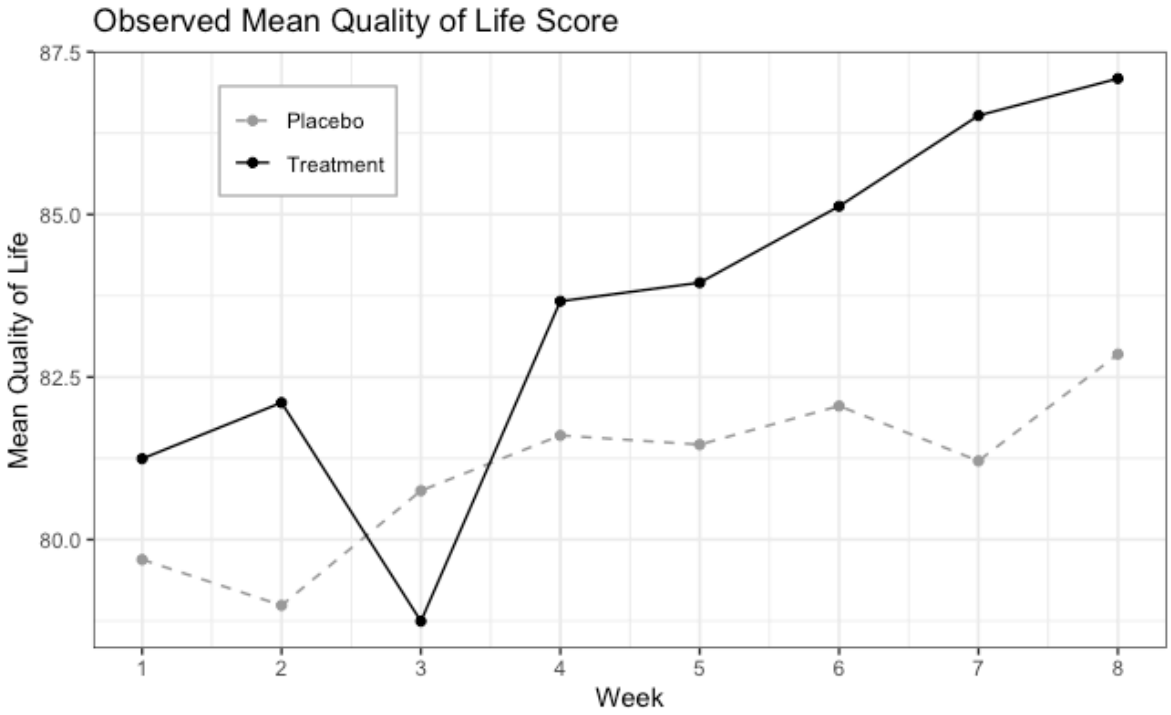
- Polished visualizations
- Choices are made to most clearly communicate data to an audience
- Precise labeling and annotation
- Thoughtful choices of aesthetics
- Publication or presentation quality

Both should maintain integrity and should accurately reflect the data!

Exploration



Communication



Advanced ggplot2 Structure

Additional ggplot2 statements can be added to a ggplot to add geom layers, add labels, change aesthetic scales, add preset themes, and alter legends.

```
ggplot(data = mean_summary) +
  geom_point(aes(x = week, y = mean_out, color = group)) +
  geom_line(aes(x = week, y = mean_out, color = group,
                linetype = group)) +
  theme_bw() +
  scale_color_manual(values = c("pbo" = "grey60", "trx" = "black"),
                     labels = c("pbo" = "Placebo", "trx" = "Treatment")) +
  scale_linetype_manual(values = c("pbo" = "dashed", "trx" = "solid"),
                        labels = c("pbo" = "Placebo", "trx" = "Treatment")) +
  scale_x_continuous(breaks = 1:8, labels = 1:8) +
  theme(legend.position = c(0.2,0.85),
        legend.background = element_rect(color = "grey70"),
        legend.title = element_blank(),
        legend.margin = margin(c(0,5,5,5))) +
  labs(title = "Observed Mean Quality of Life Score", x = "Week",
       y = "Mean Quality of Life", color = "Group", linetype = "Group")
```

adding a line geom layer

applying a preset theme for all none data elements

manually setting scales

setting breaks (ticks) and tick labels

creating a clean legend inside plot

creating titles and labels for the axes and graph

ggplot2 Layers

Instead of creating a single plot function with many options, the ggplot2 package works by adding layers that reflect concepts

So far we have introduced the data, aesthetics, and geometry layers.

But to create more customized plots, we can leverage additional ggplot2 layers to alter the scales, legends, themes, and create facets.

Mapping Aesthetics

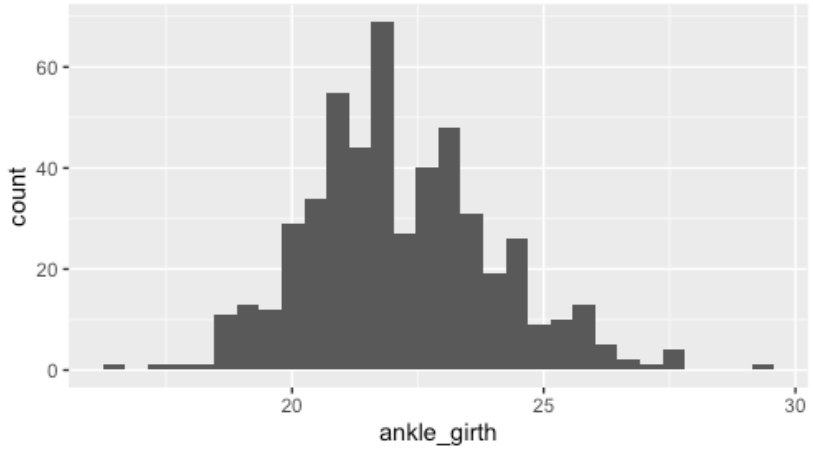
Aesthetic mappings specify how variables from a data frame are mapped to visual properties.

The most basic aesthetic mappings are **x** and **y**, which identify the variables that will be mapped to the x and y axes.

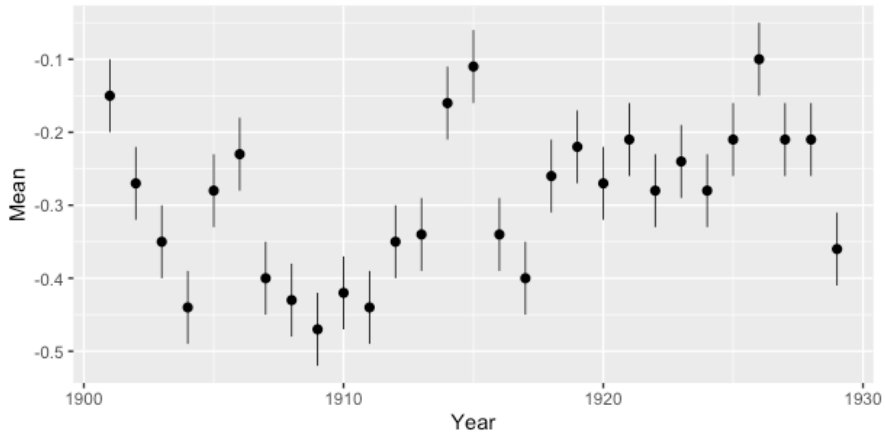
In order to visualize multiple variables at once, other aesthetics can be used: as color, fill, shape, linetype, alpha, size.

Each geom has different **required** and optional aesthetics:

`geom_histogram()` only requires x



`geom_pointrange()` requires x, y, ymin, and ymax

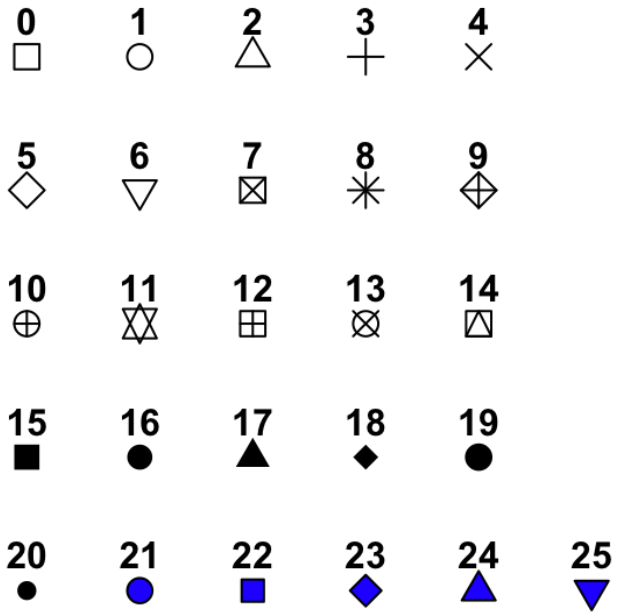


depending on the type of geom you have different requirements.

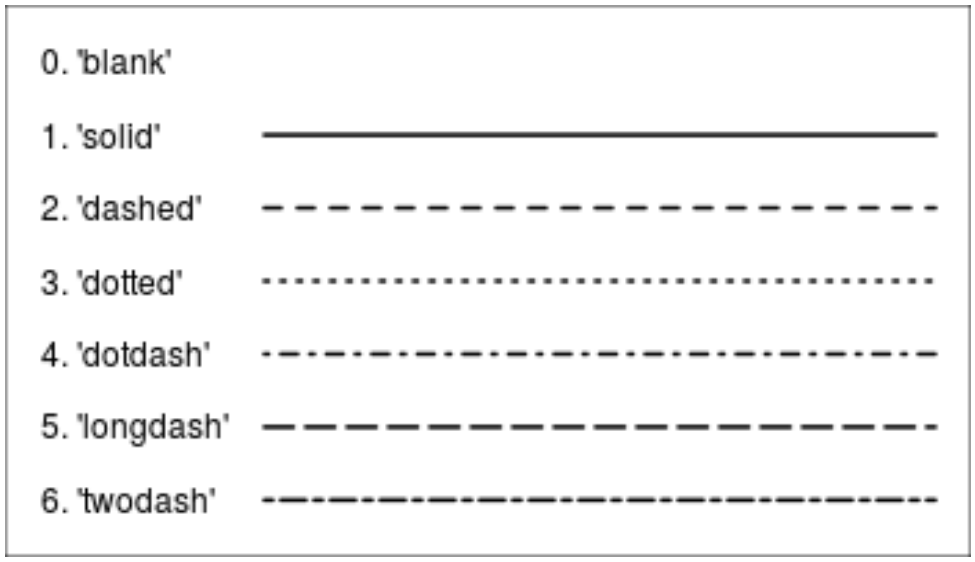
Main Aesthetics

shape

cant fill 0 - 14



linetype



can make things see through

alpha (opacity)

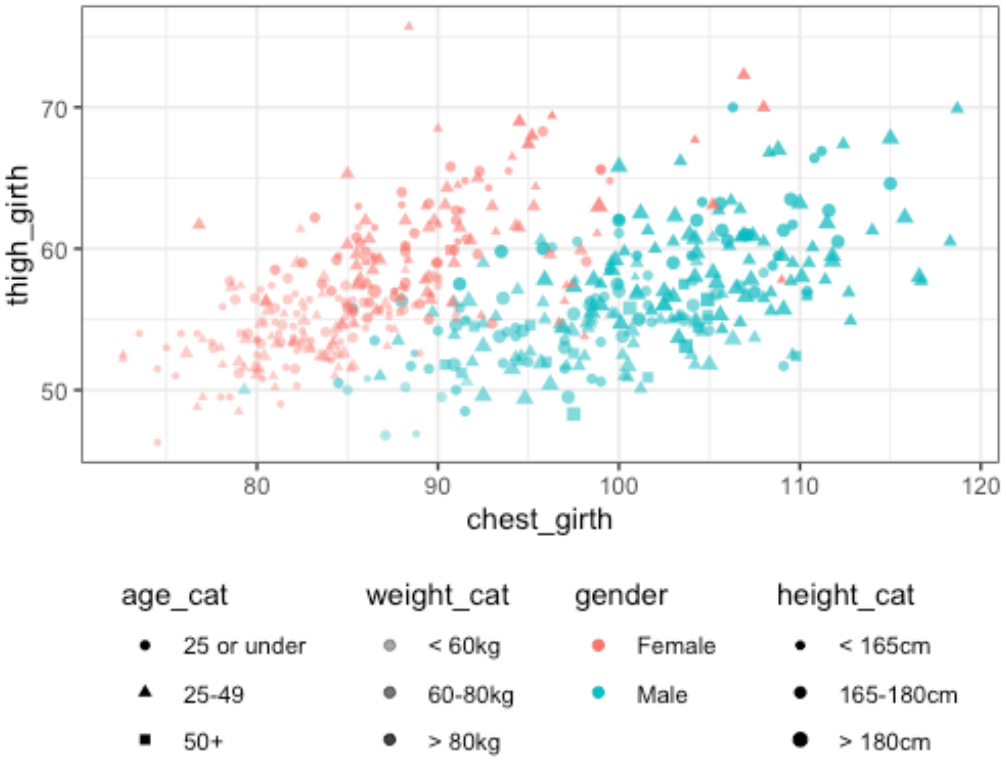
size

outside color inside color

color or fill

Maximum Encoding

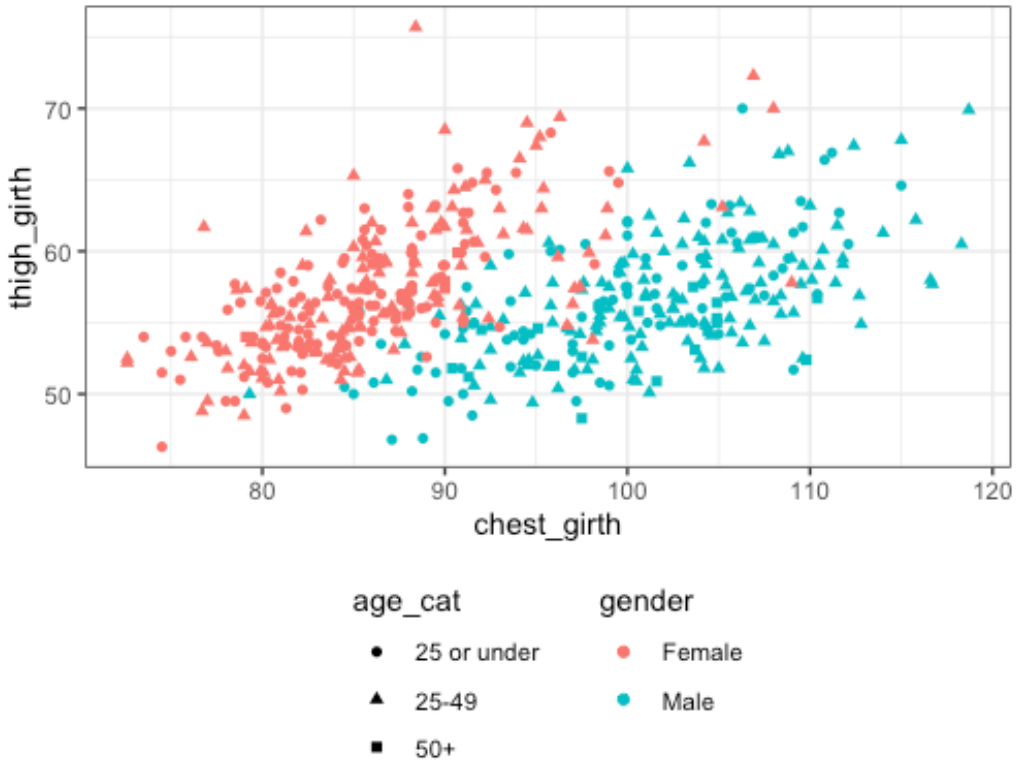
Mapping aesthetics in `ggplot()` is a powerful tool, but most visualization experts argue that people can only effectively process 3 - 4 mappings in a single visualization. **Don't overdo it!**



Mapping color, shape, size, alpha, x, and y

Too much to process!

not really gonna come away with anything



Mapping color, shape, x, and y

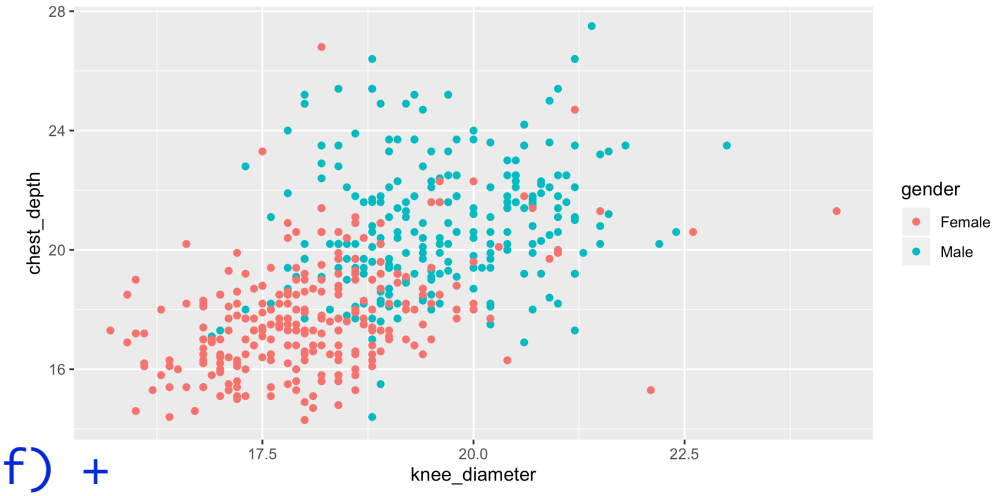
Clearer Communication!

Mapping vs. Setting Aesthetics

Mapping an aesthetic will cause it to vary based on the levels of a variable in the dataframe:

Mapping occurs within the aes() function:

links colors to value

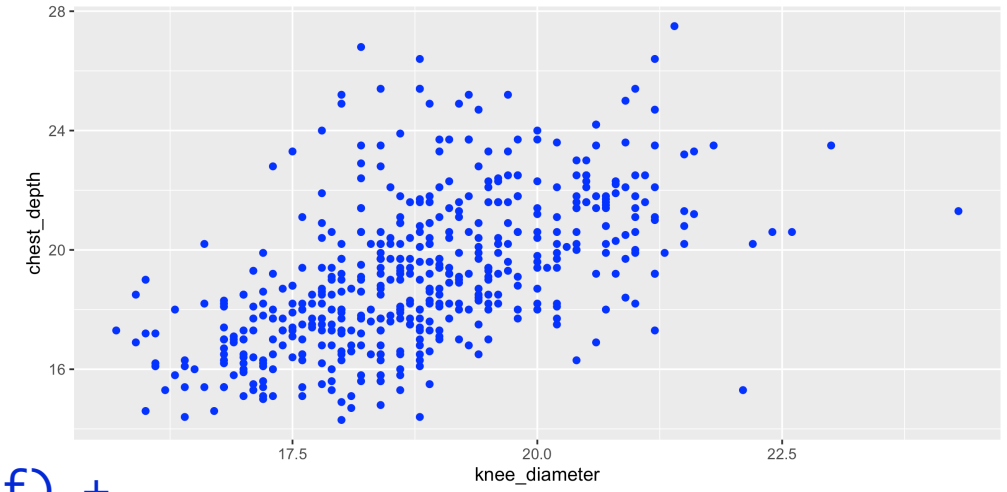



```
ggplot(data = df) +
   geom_point(aes(x = xvar, y = yvar, color = gender))
```

Setting an aesthetic will set that aesthetic to a specific value for the entire graph:

Setting occurs outside of the aes() function:

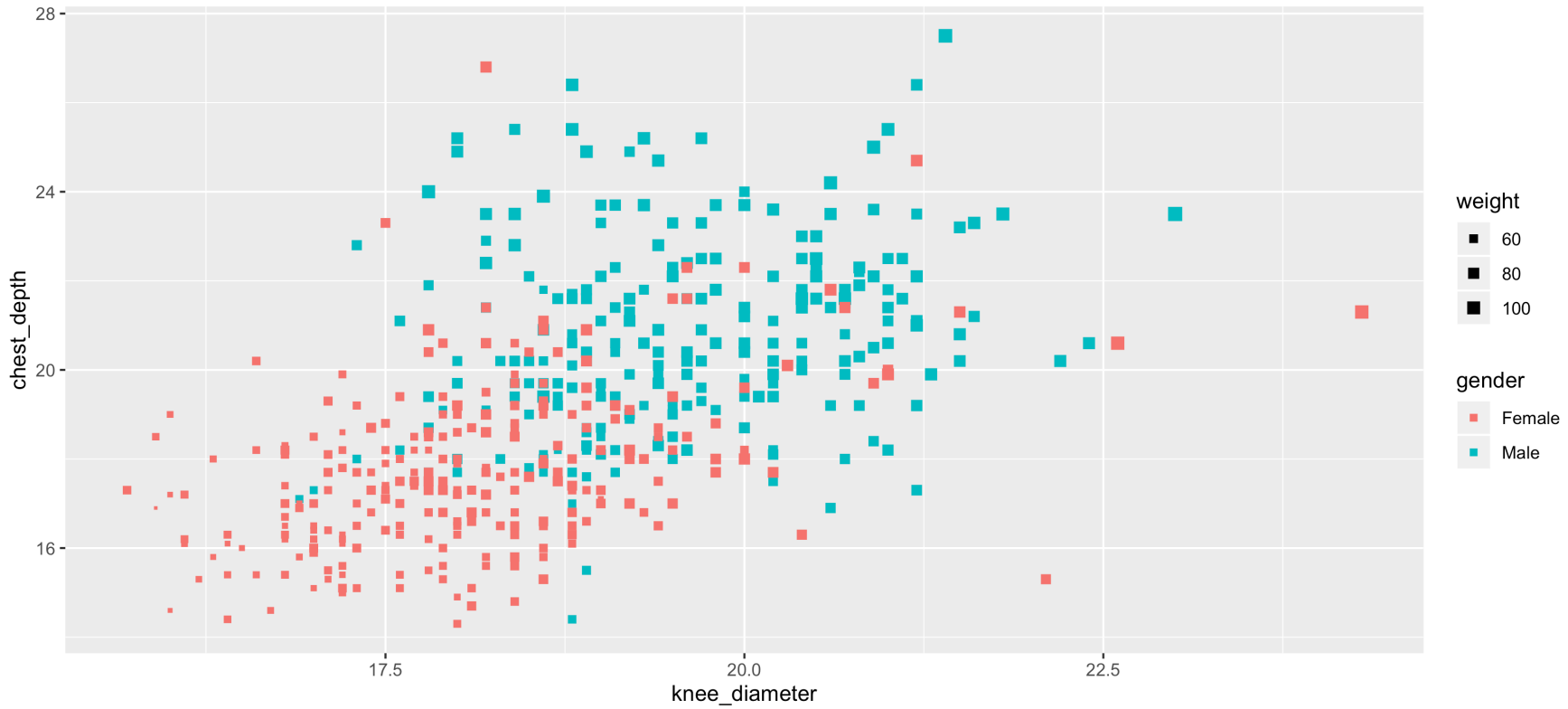
sets everything to color




```
ggplot(data = df) +
   geom_point(aes(x = xvar, y = yvar), color = "blue")
```

Mapping with Setting Aesthetics

The user can combine map **and** set aesthetics, as long as they use separate aesthetics. For example, you can map **color** to one variable, **size** to a second variable, and then set the **shape** of the points in a scatterplot




```
ggplot(data = df) +
   geom_point(aes(x = xvar, y = yvar, color = gender, size = weight), shape = 15)
```

you wouldn't want to map and set the same thing

Customizing aesthetics : Scales

Aesthetic mapping tells R that a variable should be mapped to an aesthetic — but it does not say **how** that should happen. R goes with default setting.

For example, if we map a variable to shape `aes(shape = var)` we do not explicitly say *which* shapes should be used.

Similarly, `aes(color = var)` does not specify *which* colors.

To customize details of the aesthetic scales, add scale statements, which all have a similar naming scheme: `scale_<aesthetic>_<type>`

`scale_color_continuous`

`scale_shape_discrete`

`scale_linetype_manual`

`scale_x_continuous`

`scale_y_discrete`

`scale_alpha_manual`

not good for continous but its good for everything else

Scale options

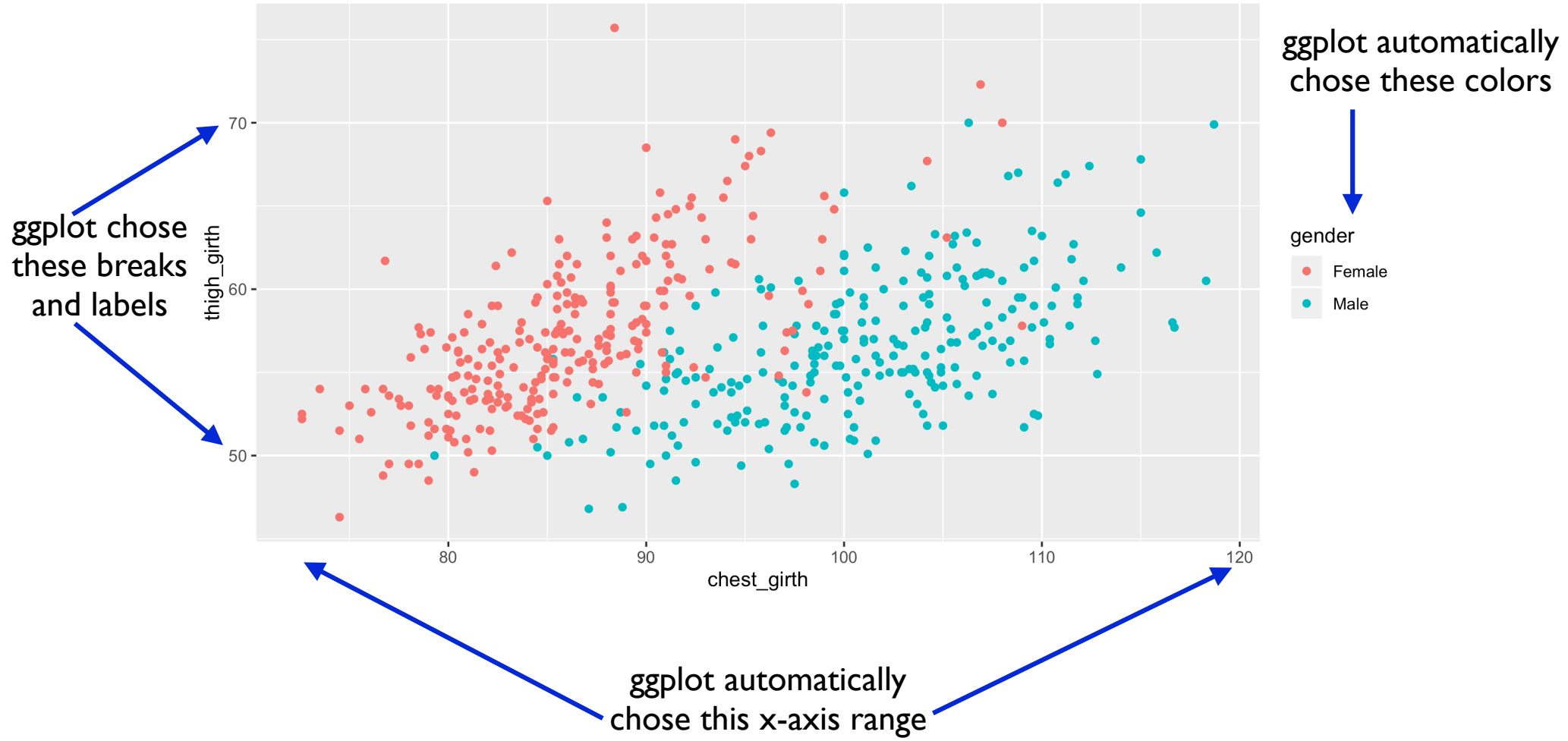
Most scale statements include the following common options for customization of aesthetic scales:

- **name** - sets the axis or legend title same as label
- **values** - manually set the aesthetic mappings
- **limits** - the minimum and maximum of the scale or axis
- **breaks** - the points along the scale or axis where labels should appear
- **labels** - the labels that appear at each break

Helpful note: In RStudio you can type `scale_` followed by TAB to get the whole list of available scale statements.

Default Scales

```
ggplot(data = body_measures) +
  geom_point(aes(x = chest_girth, y = thigh_girth, color = gender))
```

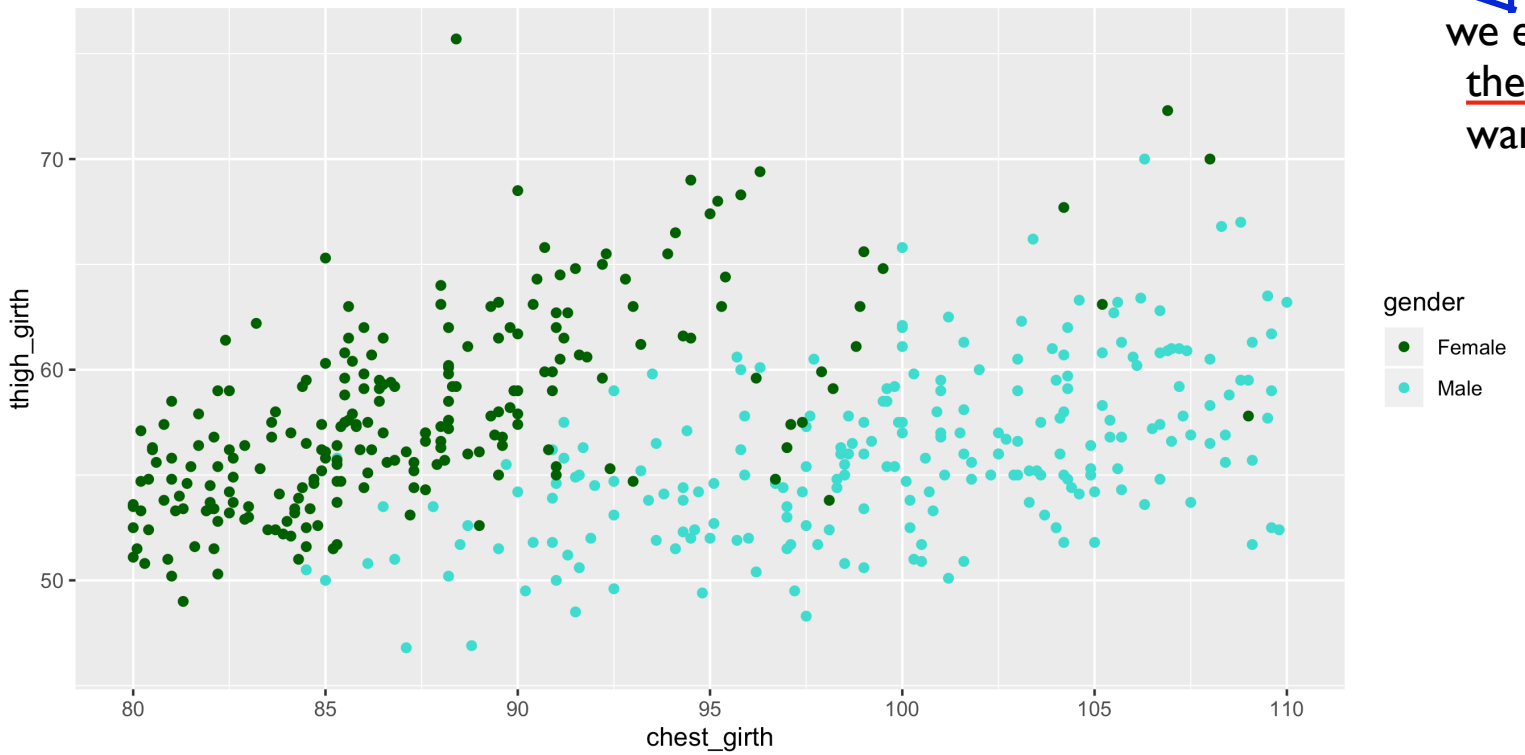


Custom Scales

```
ggplot(data = body_measures) +
  geom_point(aes(x = chest_girth, y = thigh_girth, color = gender)) +
  scale_color_manual(values = c("Female" = "darkgreen", "Male" = "turquoise")) +
  scale_x_continuous(limits = c(80, 110),
                     breaks = seq(from = 80, to = 110, by = 5))
```

we explicitly set the limits of the x-axis and what the breaks are along the x-axis

we explicitly set the colors we want for each level

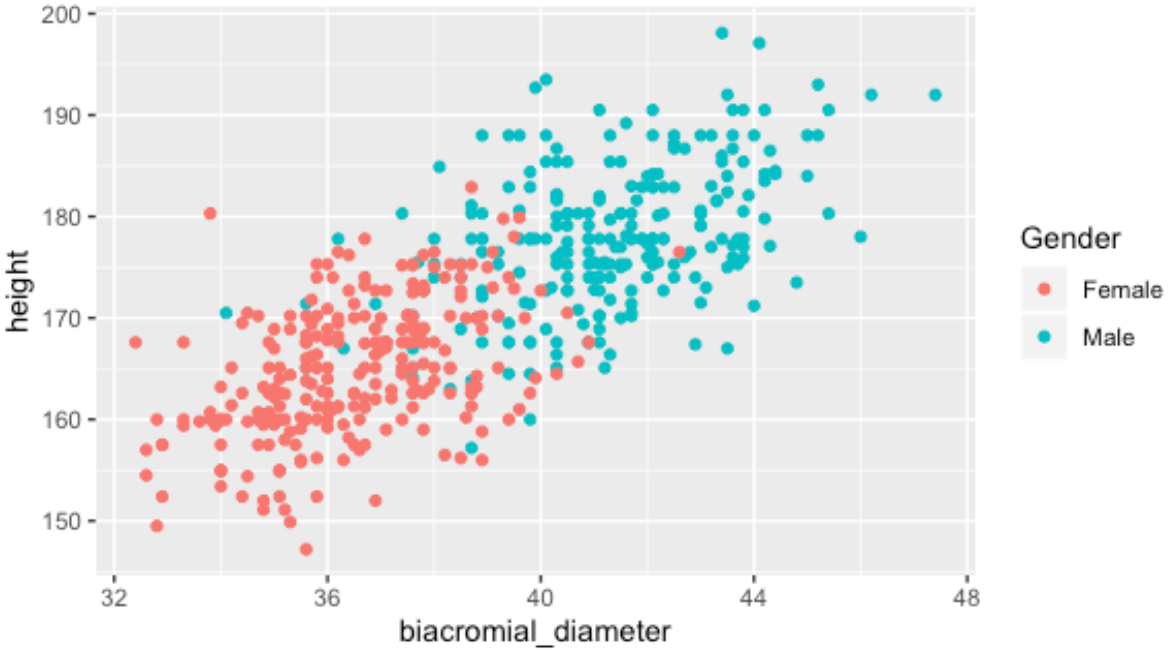


NOTE: When you set limits on x or y axes you may lose observations from your graph. R will warn you

```
Warning message:
Removed 59 rows containing missing values (geom_point).
```

Legends

When the user maps an aesthetic to a variable, R automatically generates a legend to help a viewer understand the mapping:



The easiest way to change the position of a legend is with a theme statement:

```
theme(legend.position = "bottom")

theme(legend.position = "none")
```

Changing the name of a legend can be done using several different statements:

these all do the same thing

```
labs(color = "Gender")

scale_color_manual(name = "Gender")

guides(color = guide_legend(title = "Gender"))
```

Legends

All three methods will produce a plot with a color legend that is title “Gender”

```
ggplot(data = body_measures) +
  geom_point(aes(x = biacromial_diameter,
                 y = height, color = gender)) +
  labs(color = “Gender”)

ggplot(data = body_measures) +
  geom_point(aes(x = biacromial_diameter,
                 y = height, color = gender)) +
  scale_color_manual(name = “Gender”)

ggplot(data = body_measures) +
  geom_point(aes(x = biacromial_diameter,
                 y = height, color = gender)) +
  guides(color = guide_legend(title = “Gender”))
```

Faceting

We often want to create separate visualizations for subsets of our data. `ggplot2` uses **facets** to split graphs by the levels of variables. Facets are created through `facet_wrap()` and `facet_grid()` statements.

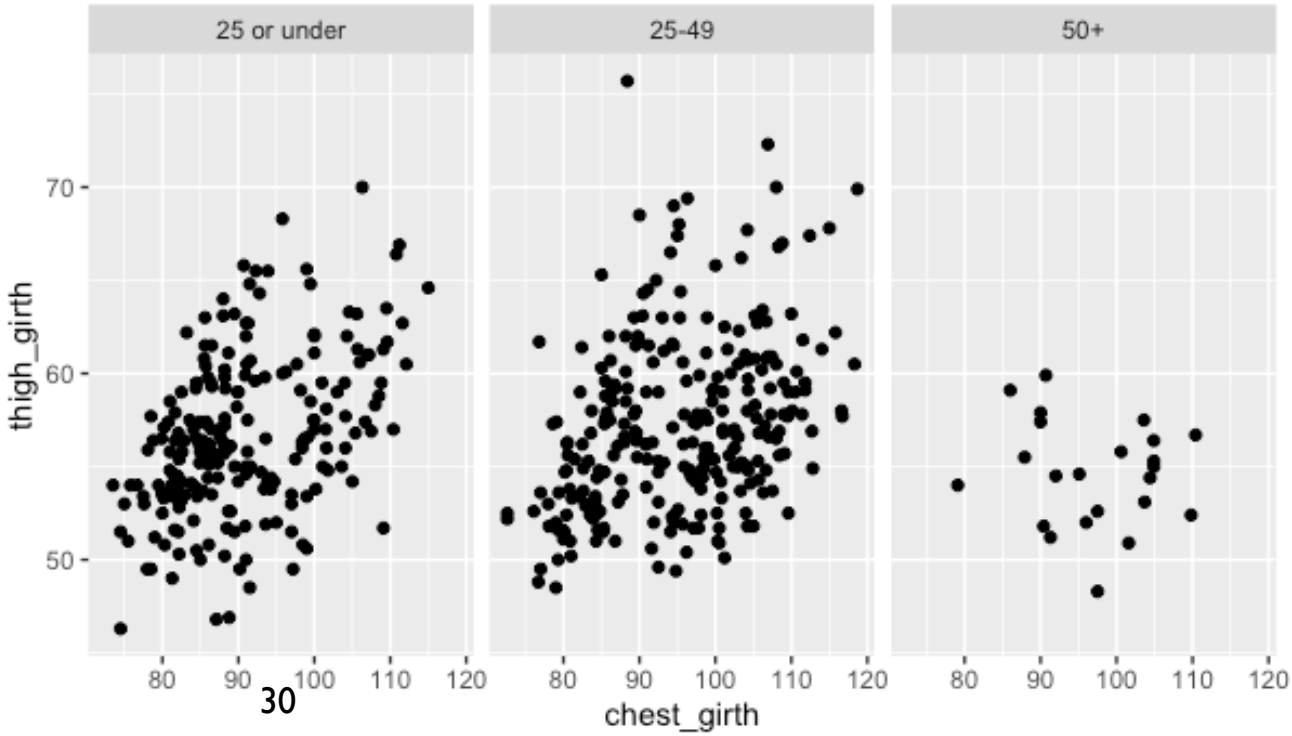
`facet_wrap()` separates plots by levels of a single variable:

```
ggplot(data = body_measures) +
  geom_point(aes(x = chest_girth, y = thigh_girth)) +
  facet_wrap(~age_cat)
```

the variable to facet graphs by

this splits graphs by age categories

columns: age categories



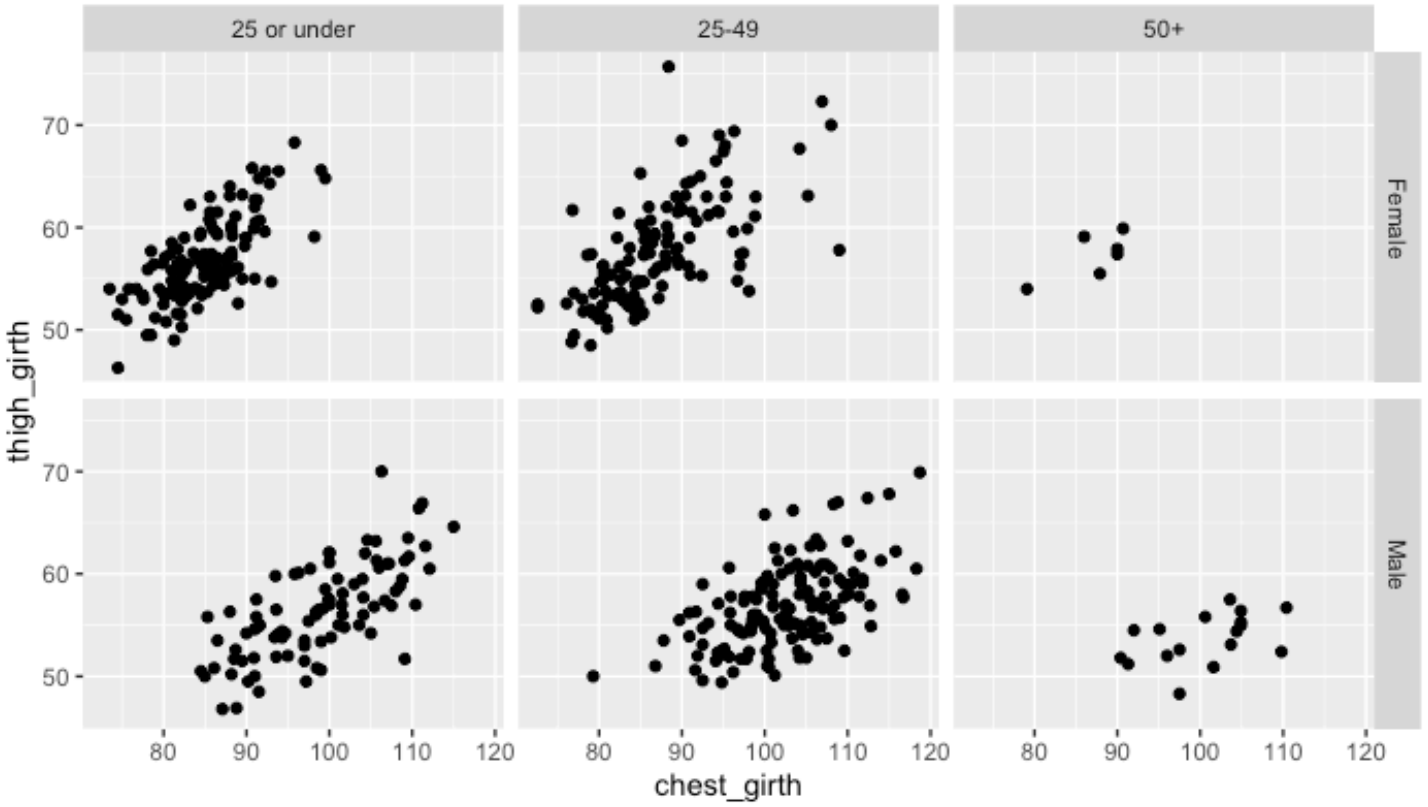
Faceting

`facet_grid()` creates a grid of graphs crossed by two variables:

```
ggplot(data = body_measures_age_cat) +
  geom_point(aes(x = chest_girth, y = thigh_girth)) +
  facet_grid(gender ~ age_cat)
```

row ~ column

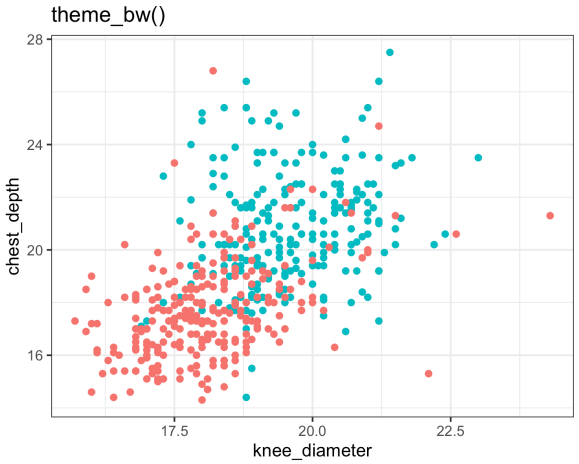
columns: age categories

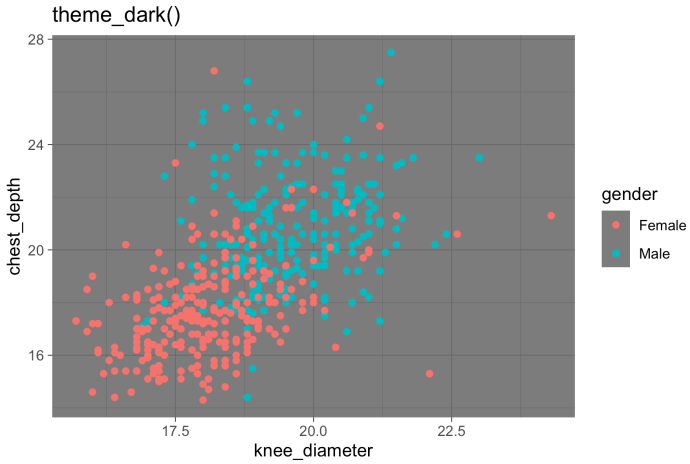


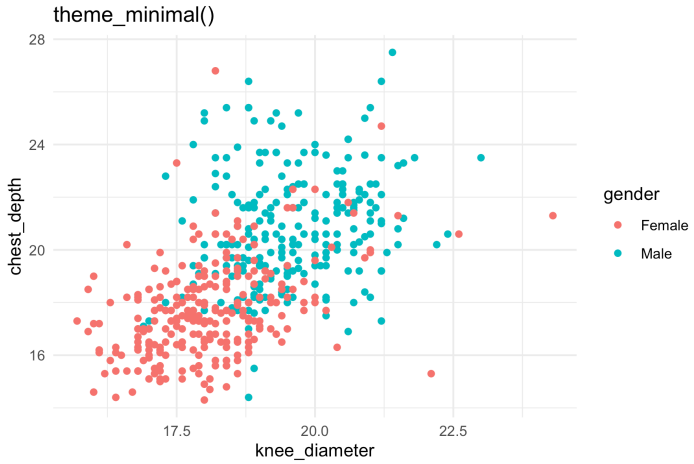
rows: gender

ggplot2 Themes

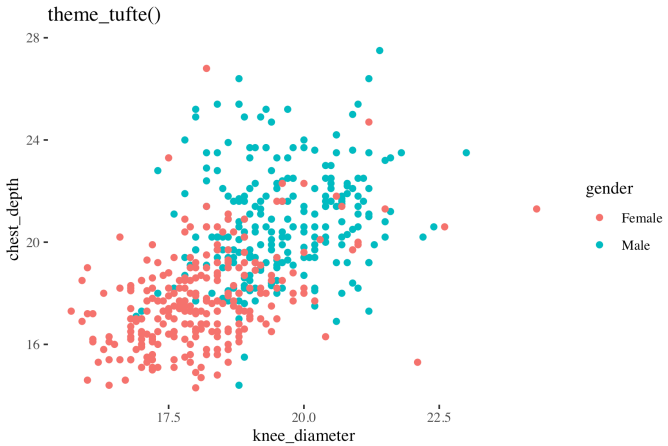
There are many options for `ggplot2` graphs. Some sets of options have been combined into preset themes to emulate popular data visualization styles:

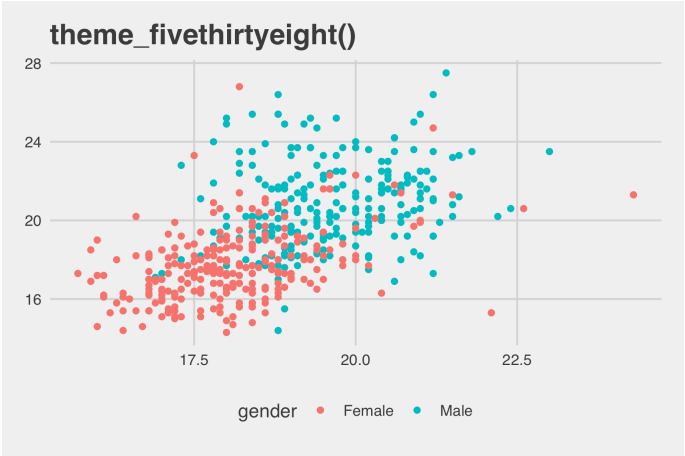


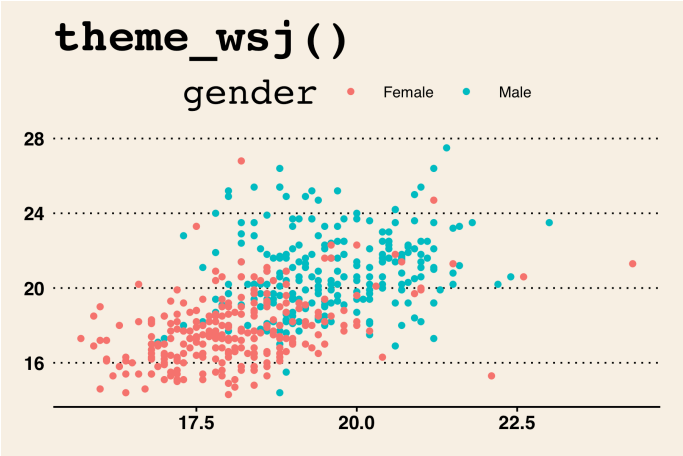




The `ggthemes` package includes additional available themes:







Theme Statements

In addition to preset themes, every element of a `ggplot()` graph can be customized using standalone `theme()` statements.

These statements allow for changing non-data plot elements like:

- Label fonts and font sizes
- Plot background
- Legend appearance
- Facet label background

The format for theme statements depends on the type of theme option you are interested in adjusting:

changing size

For text: `theme(axis.text = element_text(size = 14))`

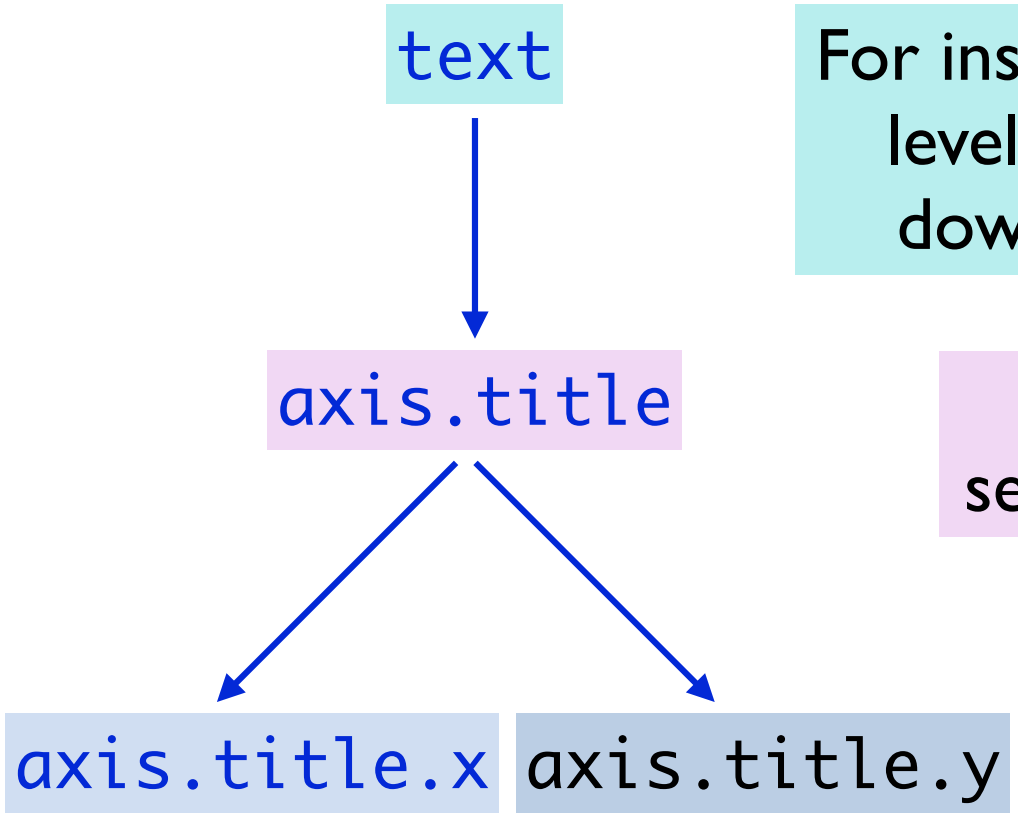
For boxes: `theme(plot.background = element_rect(fill = "grey"))`

For axis lines: `theme(axis.line = element_line(color = "red"))`

To remove an element: `theme(legend.title = element_blank())`

Theme Inheritance

There is a theme option for nearly every aspect of a ggplot graph, but they are organized in a hierarchy



For instance, you can change the high-level text settings and it will filter down to all the text in the graph

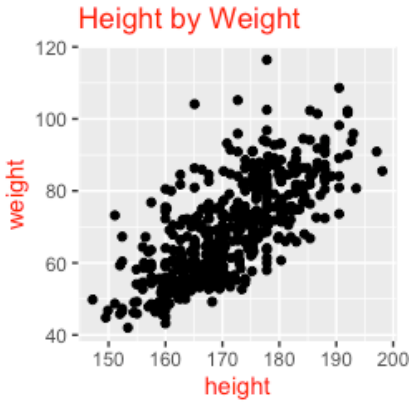
Or you can change the text settings for the title of both axes

Or you can change the text settings for just the x-axis title

Theme Inheritance

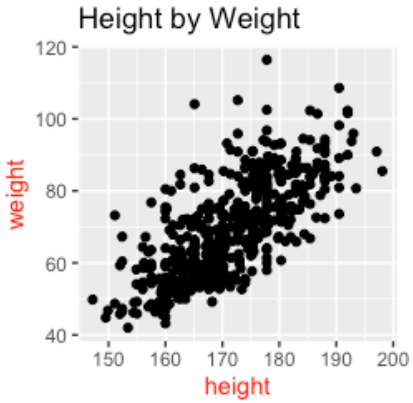
text

```
ggplot(data = body_measures) +
  geom_point(aes(x = height, y = weight)) +
  labs(title = "Height by Weight") +
  theme(text = element_text(color = “red”))
```



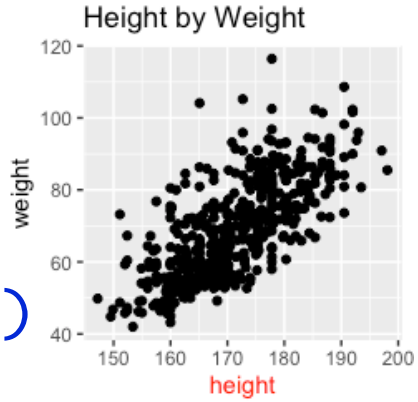
axis.title

```
ggplot(data = body_measures) +
  geom_point(aes(x = height, y = weight)) +
  labs(title = "Height by Weight") +
  theme(axis.title = element_text(color = “red”))
```



axis.title.x

```
ggplot(data = body_measures) +
  geom_point(aes(x = height, y = weight)) +
  labs(title = "Height by Weight") +
  theme(axis.title.x = element_text(color = “red”))
```



Using Color

R allows the user to use colors based on a color name

`color = "red"` OR `color = "firebrick2"`

Great R color name guide: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

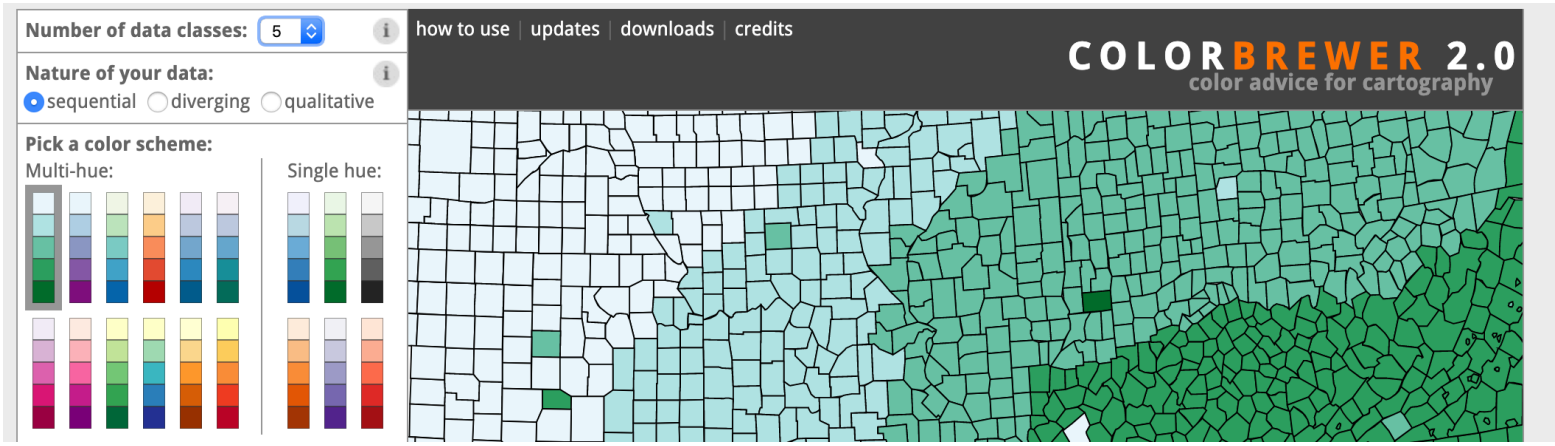
Users can also specify a color using HTML hex codes

`color = "#297A31"` OR `color = "#8F9AC0"`

https://htmlcolorcodes.com/

There are other online resources for picking color scales, such as Color Brewer which was developed for making beautiful maps.

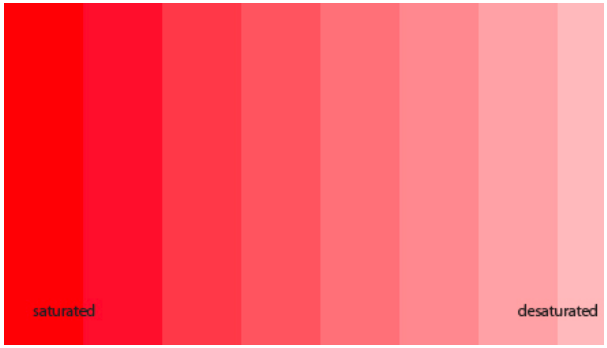
http://colorbrewer2.org/



Using Color

categorical

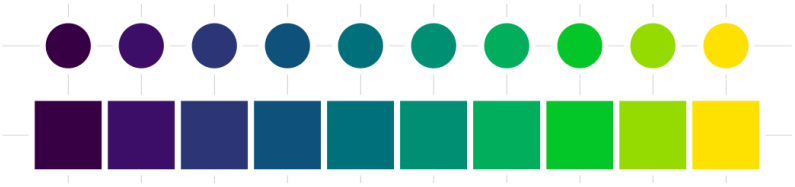
Saturation is most often used for continuous variables



Hue is best used for 'discrete' variables

The `viridis` package provides color scales which are:

- perceptually uniform (the difference between colors steps is equal)
- can be printed in both color and black-and-white
- are color-blind friendly*

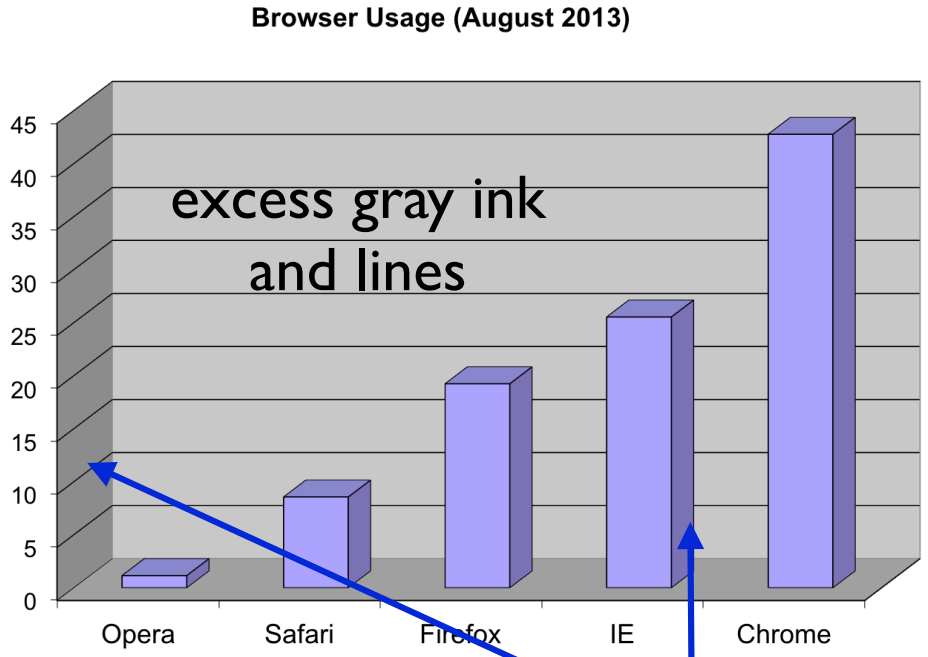


*Color blindness affects approximately 1 in 12 men (8%) and 1 in 200 women (0.5%) in the world. There are an estimated 300 million colorblind people in the world http://www.colourblindawareness.org/

Chartjunk

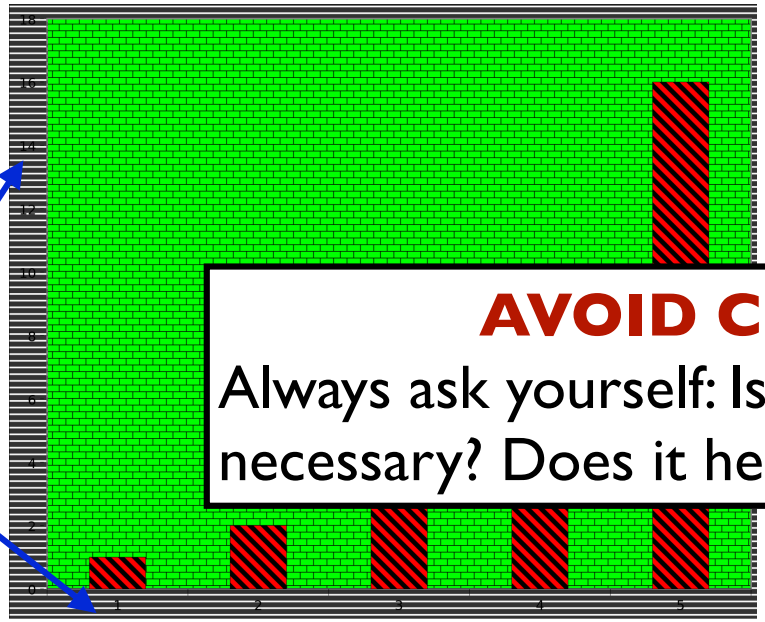
The visualization guru Edward Tufte coined the term *chartjunk*:

- Confusing visual elements
- Noisy unnecessary backgrounds
- Visuals which make it *harder* for the viewer to see the data
- Too much ink devoted to non-data elements



The textures obscure the purpose of this graph.

It is almost impossible to read the actual values!

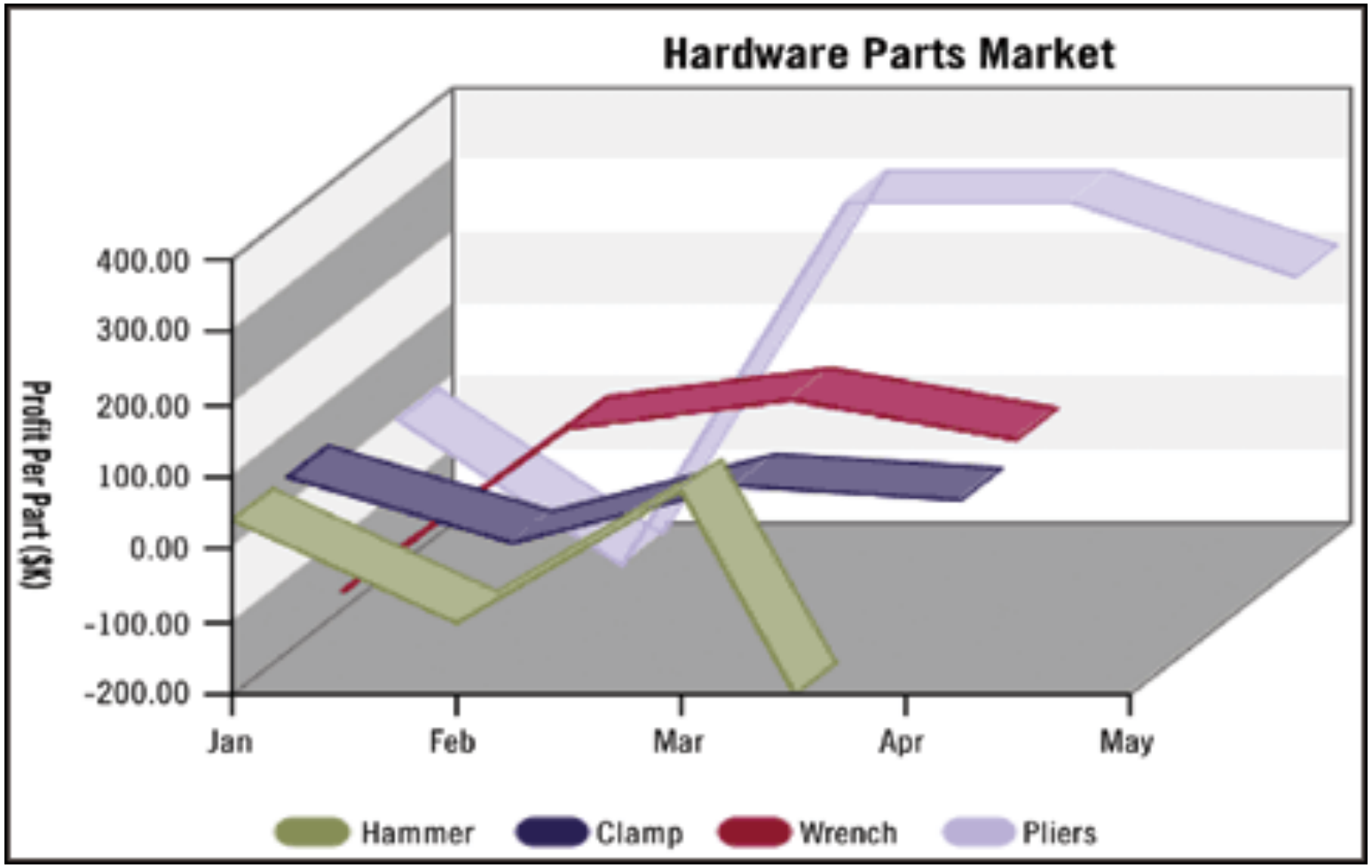


AVOID CHARTJUNK

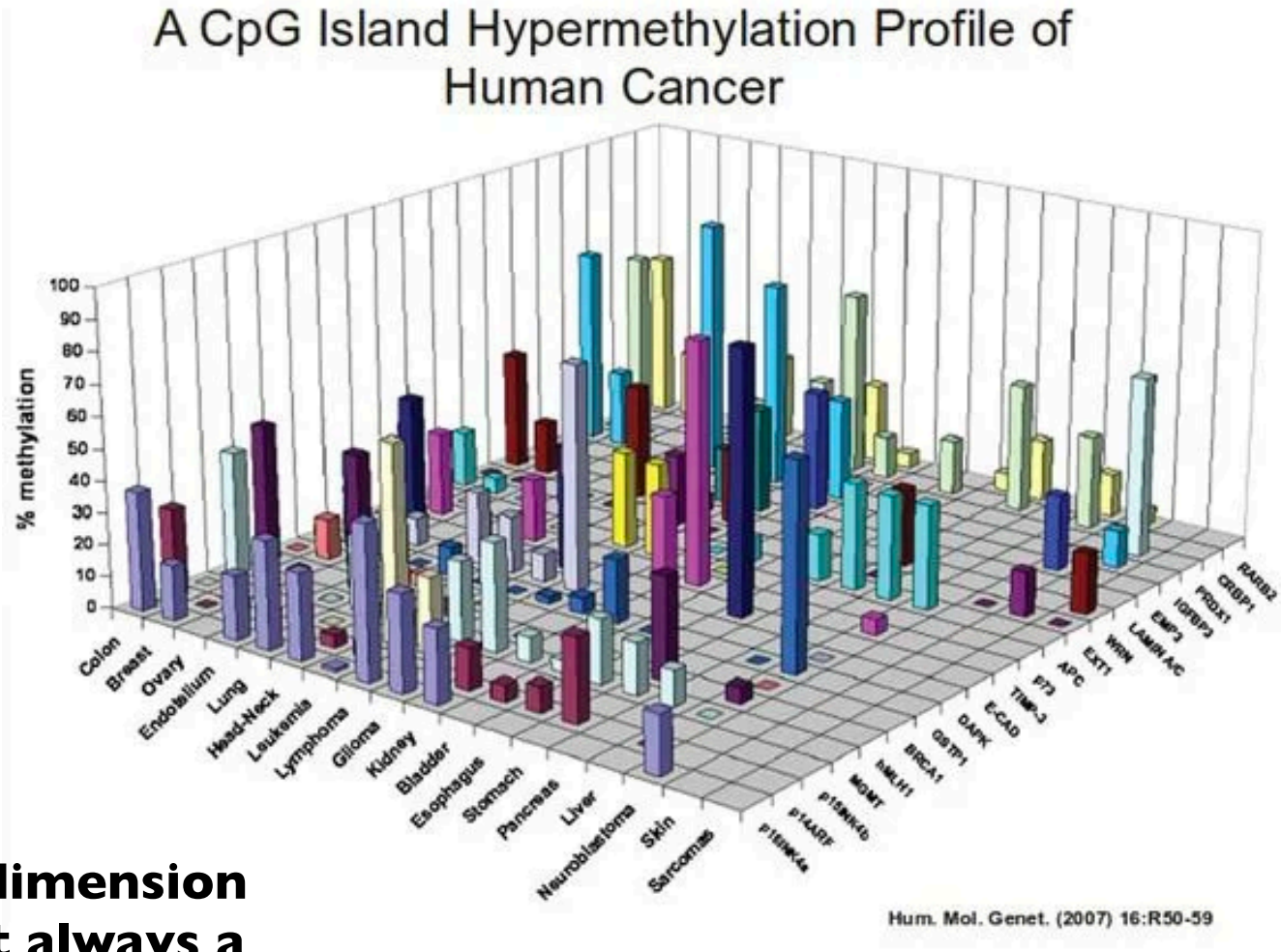
Always ask yourself: Is this visual element necessary? Does it help communicate the data?

when you dont need a dimension to convey information

Meaningless Third Dimension

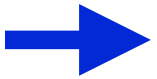


Meaningful Third Dimension



A third dimension is almost always a bad idea.

If you are ever tempted to use a third dimension…



Think about a heat map (geom_tile) or facets instead

Bizarre Scales

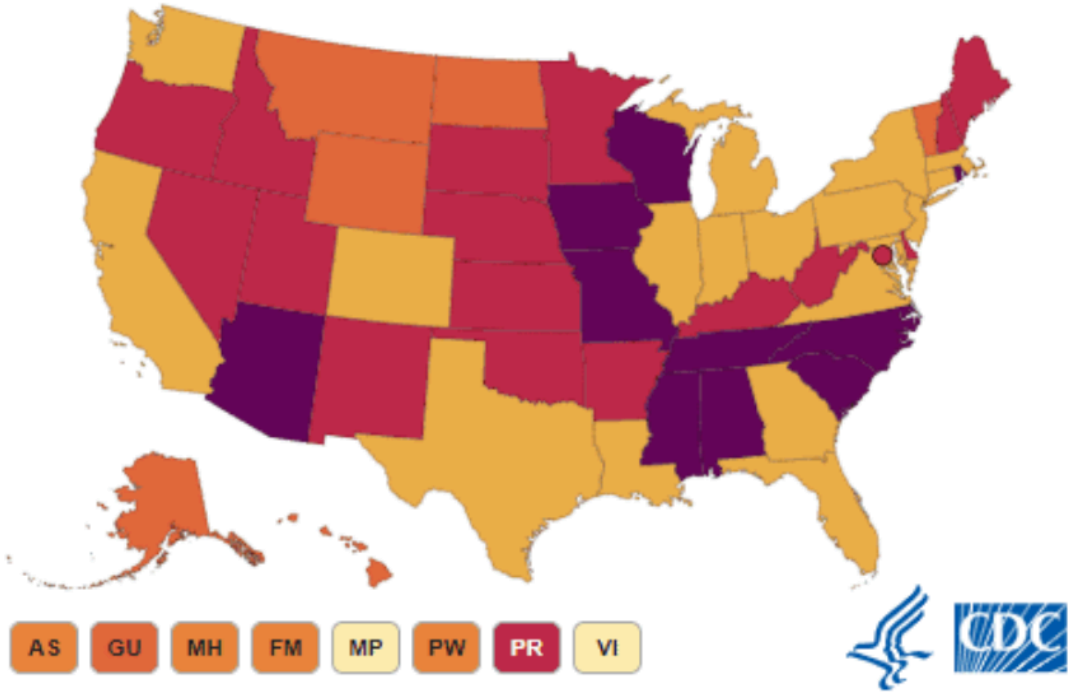
Joint Statistical Meetings locations:

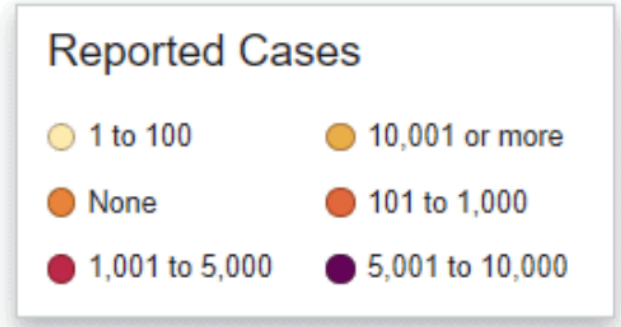
Bizarre Scales 2020

We are doomed to repeat the same errors:

most saturated should be greatest number

18 states report more than 10,000 cases of COVID-19.

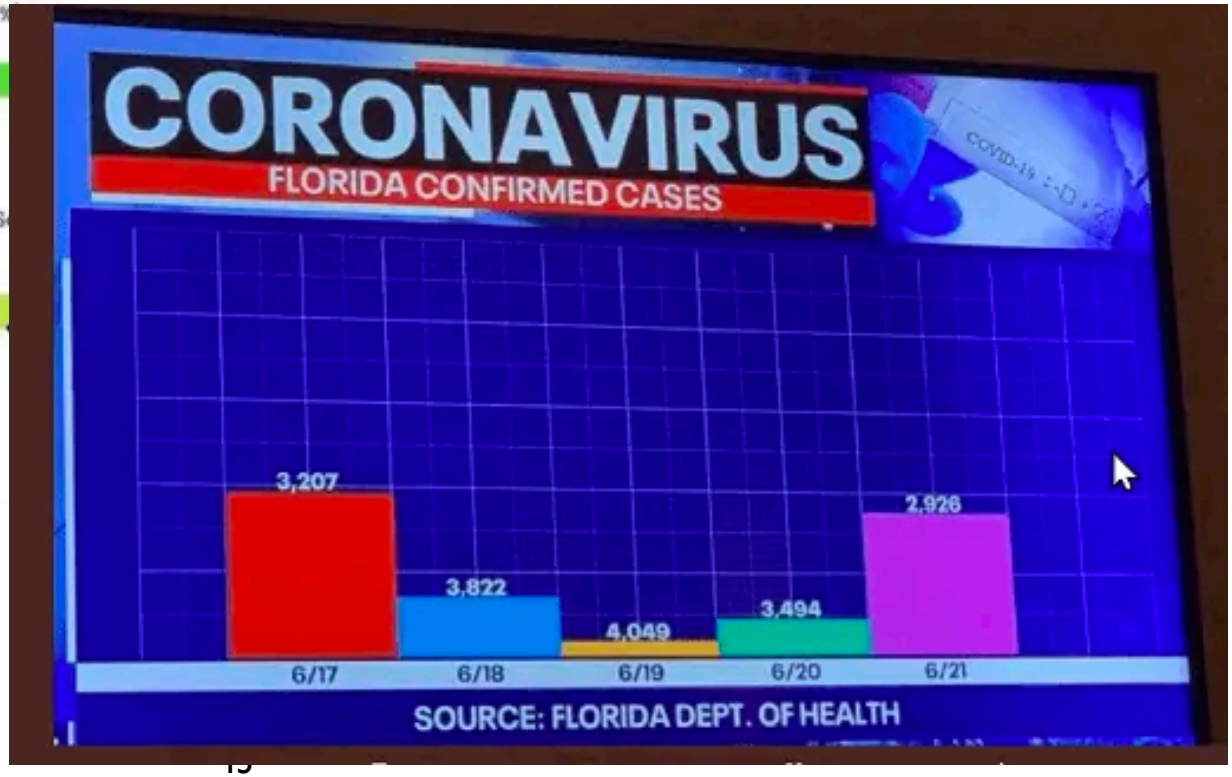




Obvious Misrepresentation

Clearly trying to communicate a politically motivated narrative about Joe Biden!

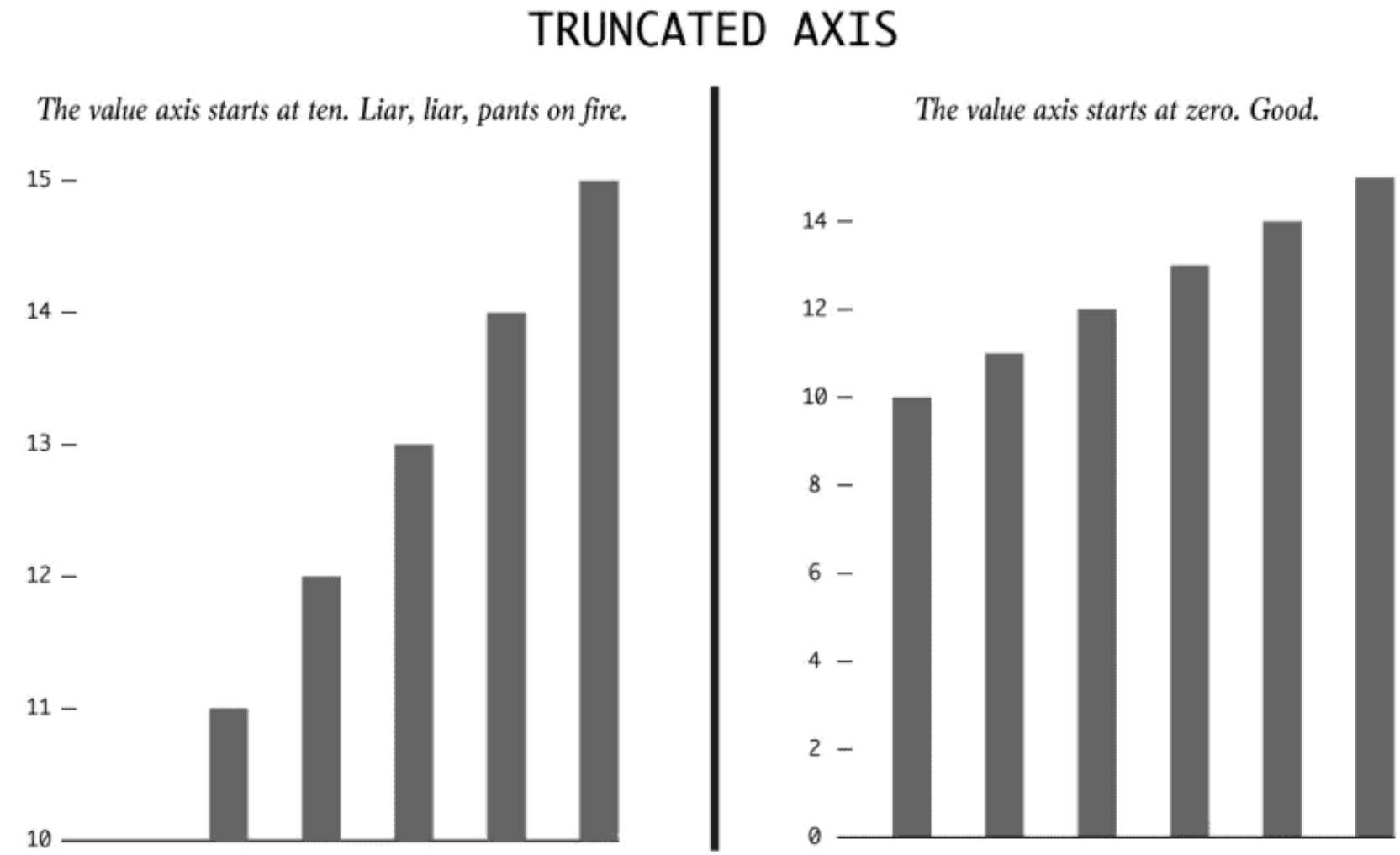
sizes should reflect value



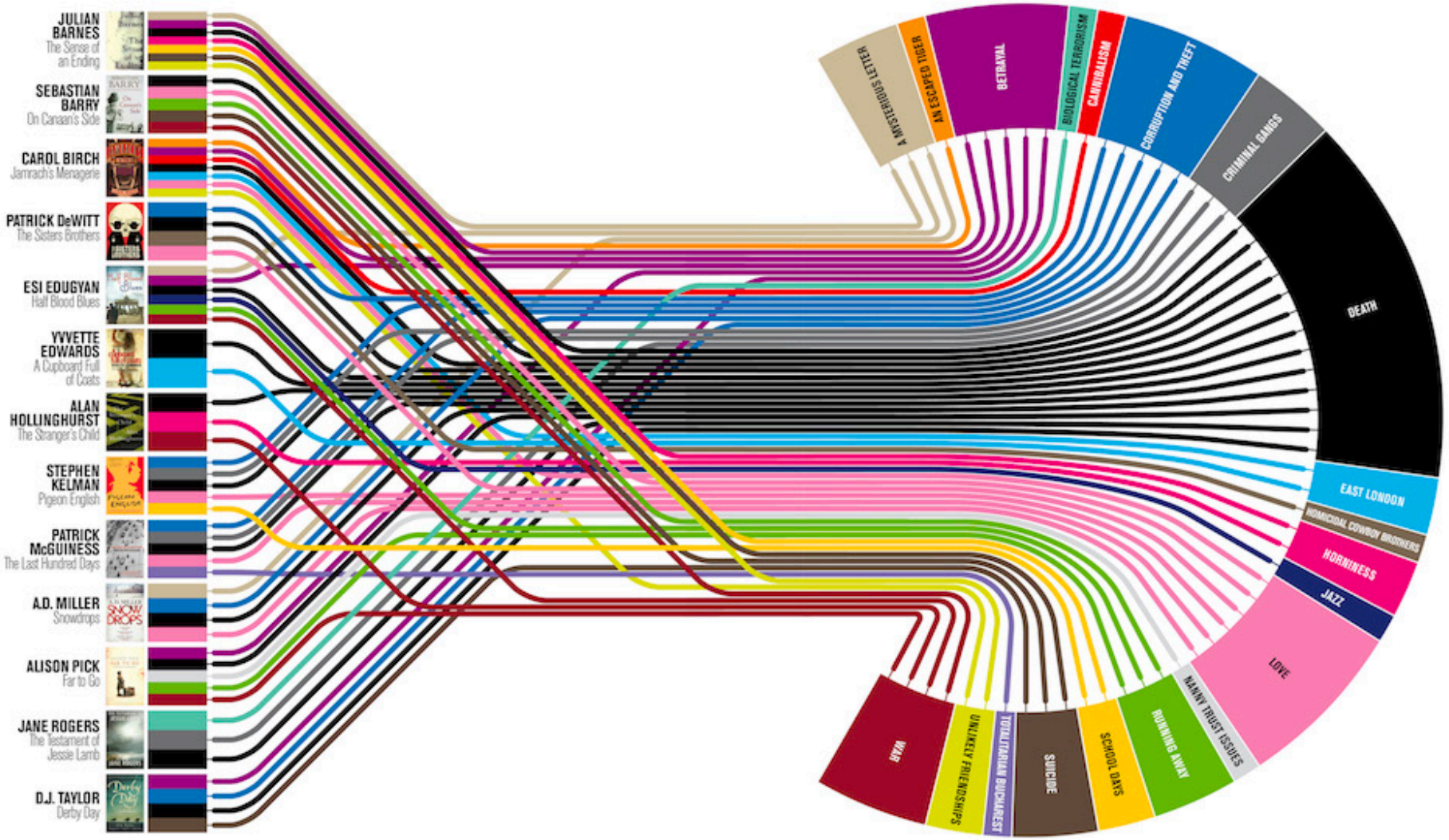
WHAT??

Subtle Misrepresentation

do not change the scale purposefully to misrepresent effects



Crippling Complexity

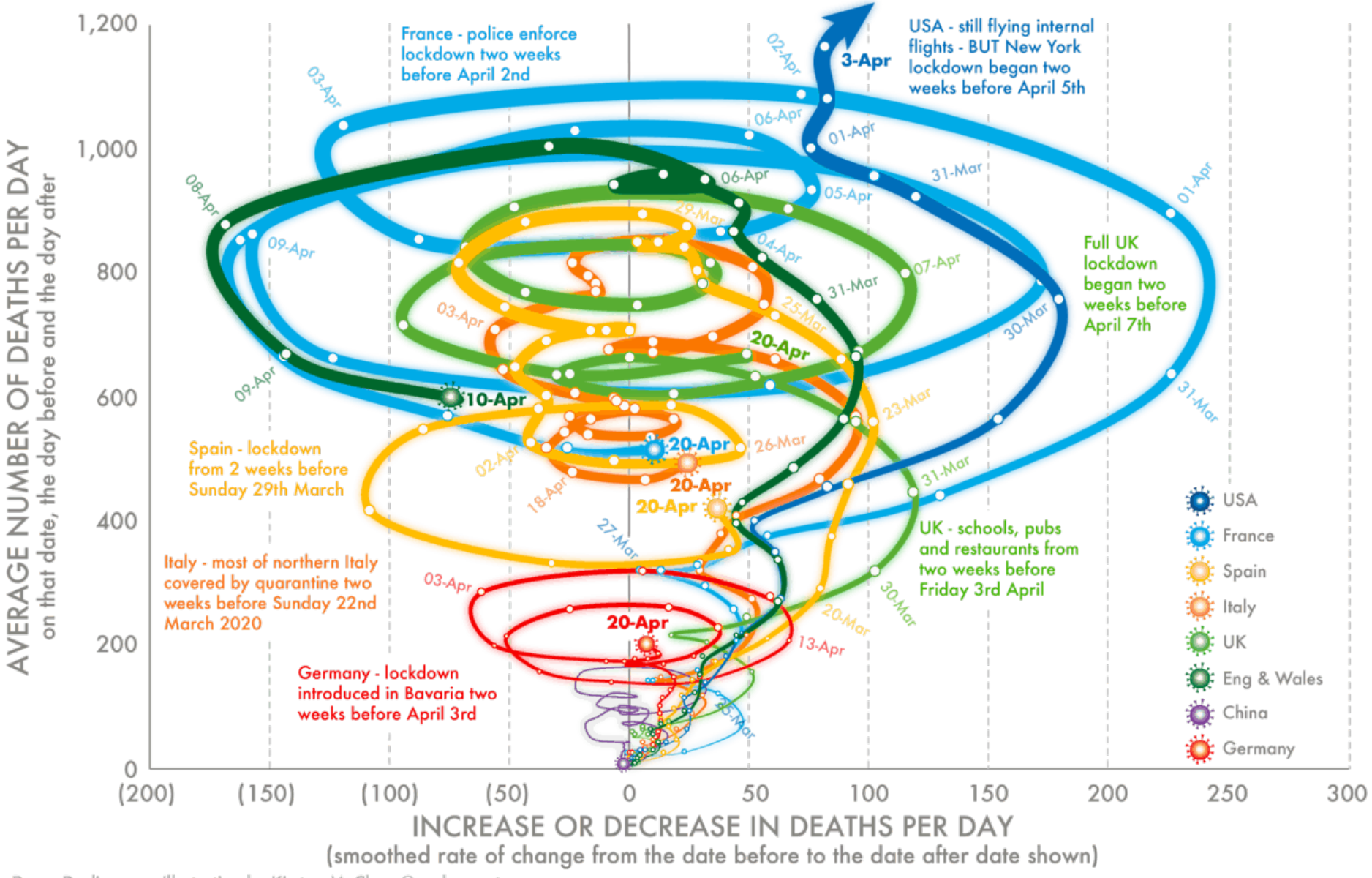


Plot lines

What makes a prize-winning novel? As Julian Barnes wins the Booker Prize, Delayed Gratification's Johanna Kamradt charts the themes of this year's longlisters.

Illustration: Christian Tate

Misguided Design



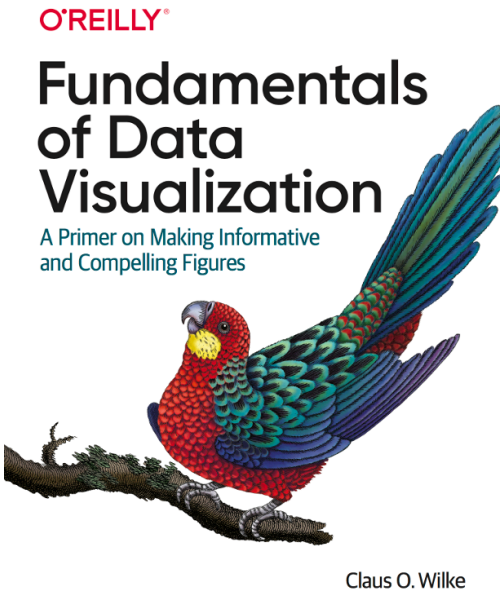
DannyDorling.org. Illustration by Kirsten McClure @orpheuscat

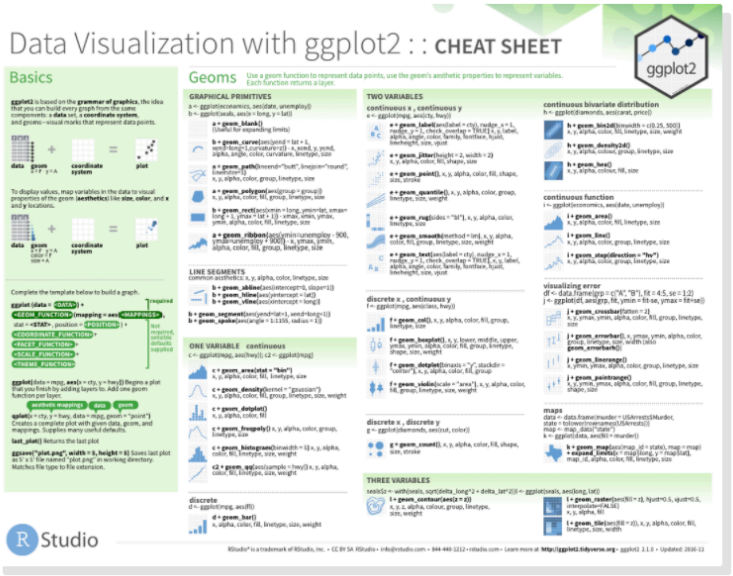
Visualization Resources

Claus O. Wilke's

Fundamentals of Data Visualization

https://clauswilke.com/dataviz/





Check out their R Markdown Cheatsheet too!

https://rstudio.com/resources/cheatsheets/

Visualization = Communication

Visualizations are a form of communication — they tell a story and highlight elements of a data set.

- Preserve data integrity and do not intentionally mislead your audience
- Use the type of visualization that best communicates your message using the type of data you have
- Don't overload your audience, stick to a 1-2 clear takeaways you want to communicate
- Eliminate chart junk as much as possible and *don't use three-dimensional plots*
- Consider the scale of your axes and whether you are being unintentionally misleading