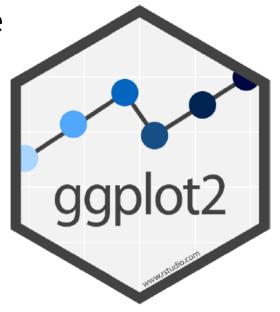
Visualize Data

ggplot2

One of the earliest packages in the tidyverse

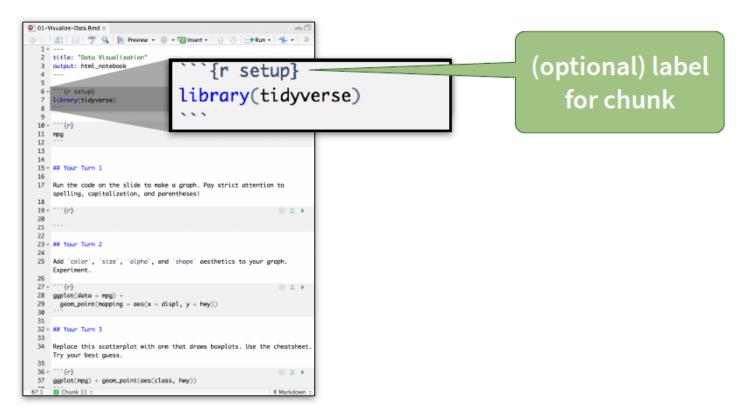
Implements the "grammar of graphics"





Setup

The setup chunk is always run once before anything else



About the data

Monthly averages of water quality data (e.g., temperature, salinity, dissolved oxygen, etc.) from 6 National Estuarine Research Reserves

```
'``{r loaddata}
wq ← read.csv(here::here('data', 'monthly_wq.csv'))
...
```

Be sure to check out the data

You can use:

QUIZ

What relationship do you expect to see between temperature (temp) and dissolved oxygen (do_mgl)?



YOUR TURN 1

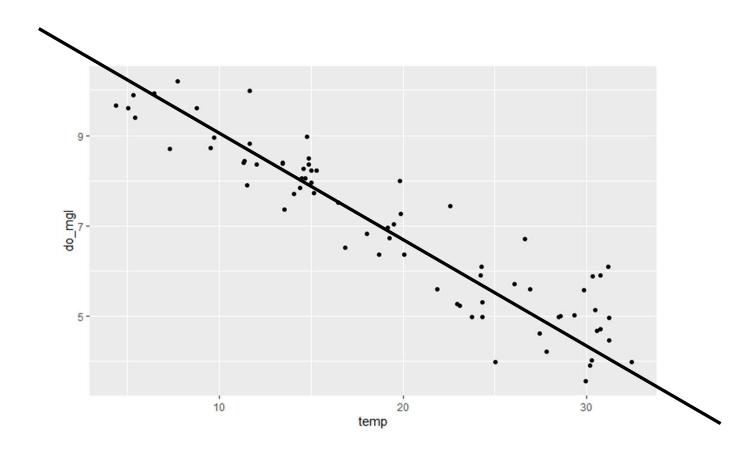
Run this code in your notebook to make a graph.

Pay careful attention to spelling, capitalization, and parentheses!

```
ggplot(data = wq) +
    geom_point(mapping = aes(x = temp, y = do_mgl))
```

I'm working on it I'm stuck! Help! I'm done!





Be sure to always put the '+'
at the end of a line, never at
the start.

ggplot(data = wq) +
 geom_point(mapping = aes(x = temp, y = do_mgl))

- 1.Initialize the figure with ggplot()
- 2.Add layers with geom_ functions

A template

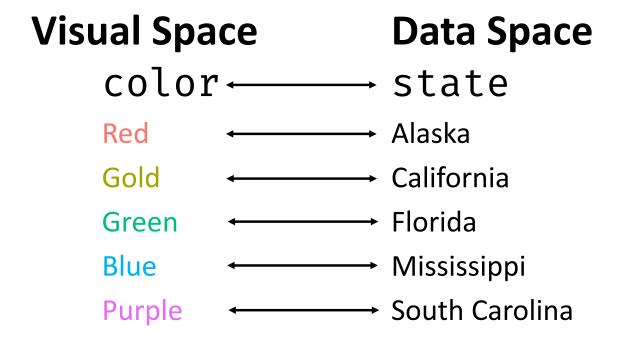
So, what are mappings?

Aesthetics are the visual properties of the objects in your plot.



Mappings describe how aesthetics should *relate to variables* in the data.

Mappings describe how aesthetics should relate to variables in the data.



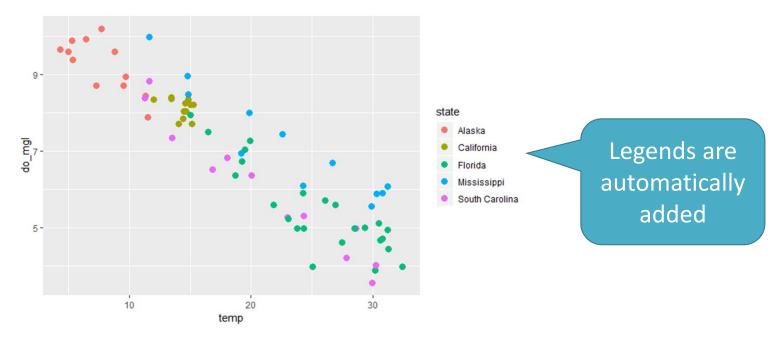
```
ggplot(data = wq) +
  geom_point(mapping = aes(x = temp, y = do_mgl, color = state))

ggplot(data = wq) +
  geom_point(mapping = aes(x = temp, y = do_mgl, size = state))

ggplot(data = wq) +
  geom_point(mapping = aes(x = temp, y = do_mgl, shape = state))
```

Aesthetic property Variable to map to

ggplot(data = wq) +
 geom_point(mapping = aes(x = temp, y = do_mgl, color = state))



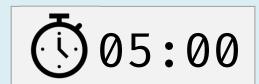
CERF 2019 | The Next Step with R | Cressman and Dunnigan

YOUR TURN 2

In the next chunk, add color, size, alpha, and shape aesthetics to your graph. Play around with the data!

What happens when you map aesthetics to discrete and continuous variables?

What happens when you apply more than one aesthetic?



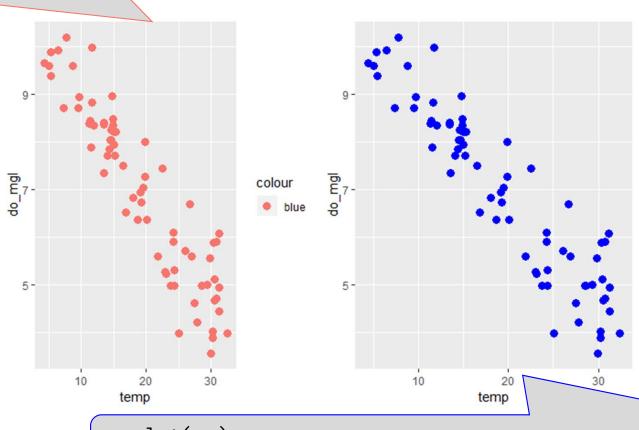
You can also *set* the aesthetic properties of your geom manually.

```
ggplot(wq) +
   geom_point(aes(x = temp, y = do_mgl, color = "blue"))
```

```
ggplot(wq) +
    geom_point(aes(x = temp, y = do_mgl), color = "blue")
```

VS

```
ggplot(wq) +
    geom_point(aes(x = temp, y = do_mgl, color = "blue"))
```



ggplot(wq) +
 geom_point(aes(x = temp, y = do_mgl), color = "blue")

CERF 2019 | The Next Step with R | Cressman and Dunnigan

TL;DR

• Inside of aes(): maps an aesthetic to a variable

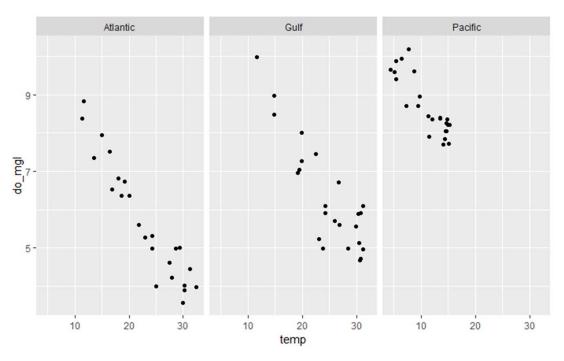
```
ggplot(wq) +
   geom_point(aes(x = temp, y = do_mgl, color = state))
```

• Outside of aes(): sets an aesthetic to a value

```
ggplot(wq) +
  geom_point(aes(x = temp, y = do_mgl), color = "blue")
```

What if I don't want to add additional variables as aesthetics?

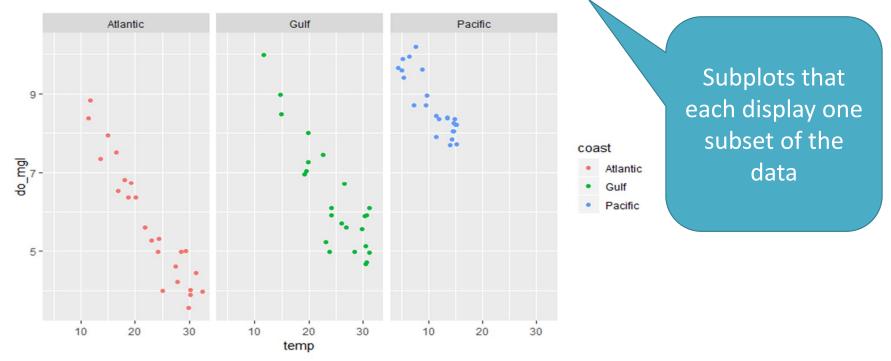
If they are categorical variables, you can use **facets**!



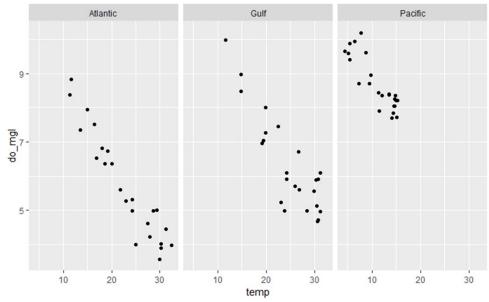
Subplots that each display one subset of the data

What if I don't want to add additional variables as aesthetics?

If they are categorical variables, you can use **facets**!

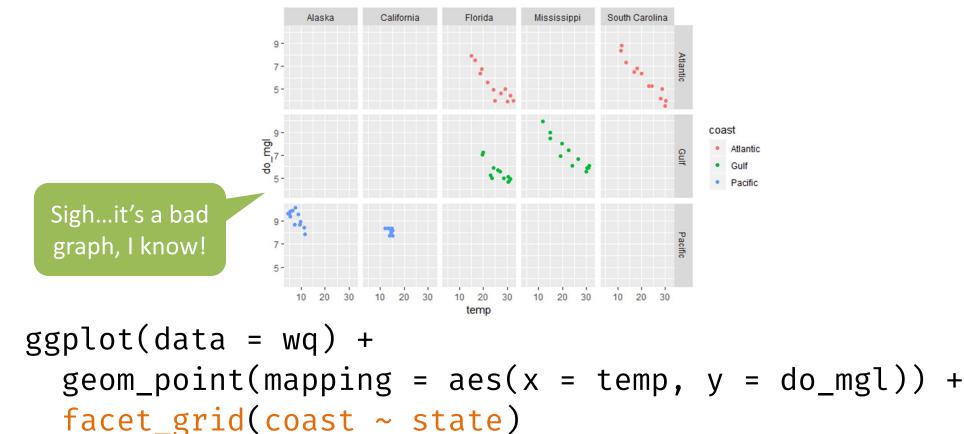


facet_wrap() ← for a single variable



```
ggplot(data = wq) +
  geom_point(mapping = aes(x = temp, y = do_mgl)) +
  facet_wrap(~ coast, nrow = 1)
```

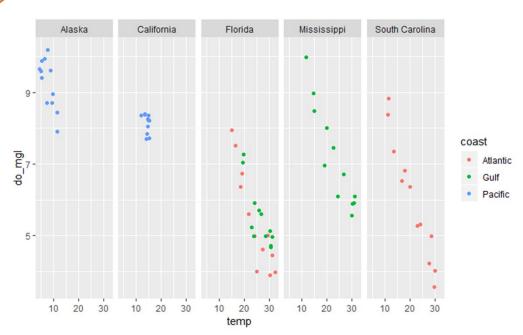
facet_grid() for a combo of 2 variables



facet_grid() ← for a combo of 2 variables

ggplot(data = wq) +
 geom_point(mapping = aes(x = temp, y = do_mgl)) +
 facet_grid(. ~ state)

Use this in place if you would prefer not to facet in the rows or columns dimension

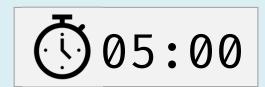


YOUR TURN 3

Using the 'wq' data, plot temperature (temp) by month as a scatterplot and facet by state and coast.

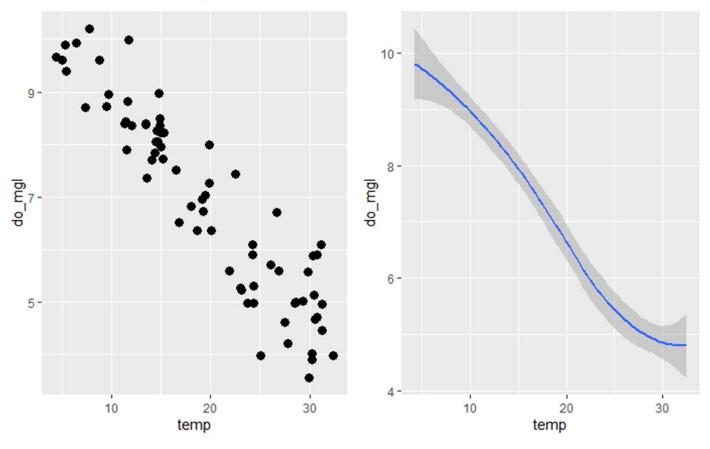
Play around with facet_grid() and facet_wrap().

What kinds of things do you notice?



Geoms

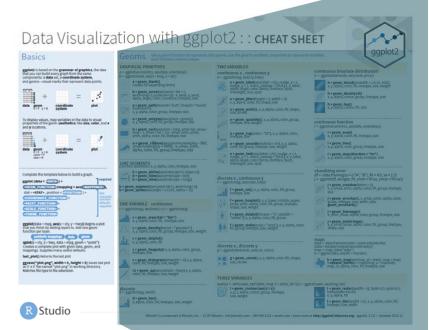
How are these plots similar? Different?



geoms

```
ggplot(data = <DATA>) +
     <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Every geom requires a mapping argument



You can also find the supported aesthetics by visiting the ?help for each geom!

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Geoms Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))

a + geom_blank() (Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1,

x, y, alpha, color, group, linetype, size

LINE SEGMENTS common aesthetics: x, y, alpha, color, linetype, size

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

c + geom_area(stat = "bin")

x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian") x, y, alpha, color, fill, group, linetype, size, weight

c + geom_freqpoly() x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5) x, y, alpha, color, fill, Tinetype, size, weight

c2 + geom_qq(aes(sample = hwy)) x, y, alpha, color, fill, linetype, size, weight

d + geom_bar() x, alpha, color, fill, linetype, size, weight

ONE VARIABLE continuous c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_dotplot()

x, y, alpha, color, fill

discrete

d <- ggplot(mpg, aes(fl))</pre>

b + geom_abline(aes(Intercept=0, slope=1))

b + geom_hline(aes(yintercept = lat)) b + geom_vline(aes(xintercept = long))

a + geom_polygon(aes(group = group)) x, y, alpha, color, fill, group, linetype, size

xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)

b + geom_rect(aes(xmin = long, ymin=lat, xmax=long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

b <- ggplot(seals, aes(x = long, y = lat))

TWO VARIABLES

continuous x, continuous y e <- ggplot(mpg, aes(cty, hwy))



e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, Tamilly, fontface, hjust, lineheight, size, ylust



e + geom_point(), x, y, alpha, color, fill, shape, size, stroke



e + geom_quantile(), x, y, alpha, color, group, linetype, Size, weight



e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, Size e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight



discrete x, continuous y

f <- ggplot(mpg, aes(class, hwy)) f + geom_col(), x, y, alpha, color, fill, group,





f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group



f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x, discrete y

g <- ggplot(diamonds, aes(cut, color))



g + geom_count(), x, y, alpha, color, fill, shape, size, stroke

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))



h + geom_bln2d(binwidth = c(0.25, 500)) x, y, alpha, color, fill, linetype, size, weight

ggplot2



h + geom_density2d() x, y, alpha, colour, group, linetype, size



h + geom_hex() x, y, alpha, colour, fill, size



I <- ggplot(economics, aes(date, unemploy))



I + geom_area() x, y, alpha, color, fill, linetype, size

I + geom_line() x, y, alpha, color, group, linetype, size

I + geom_step(direction = "hv") x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2) j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

J+geom_crossbar(fatten = 2) x,y,ymax,ymin, alpha, color, fill, group, linetype, size

J+geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also geom_errorbarh())

J+geom_linerange() x, ymin, ymax, alpha, color, group, linetype, size

J+geom_pointrange() x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

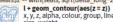
maps data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests))) map <- map data("state") k <- ggplot(data, aes(fill = murder))



k + geom_map{aes(map_ld = state), map = map) + expand_limits(x = map\$long, y = map\$lat), map_ld, alpha, color, fill, linetype, size

THREE VARIABLES

seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))i <- ggplot(seals, aes(long, lat))



x, y, z, alpha, colour, group, linetype, size, weight



l + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, Interpolate=FALSE) x, y, alpha, fill

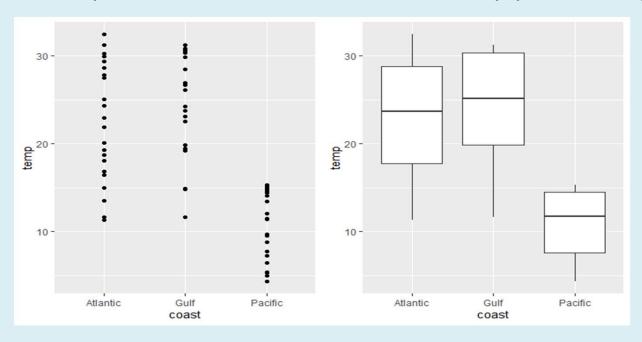


I + geom_tile(aes(fill = z)), x, y, alpha, color, fill,

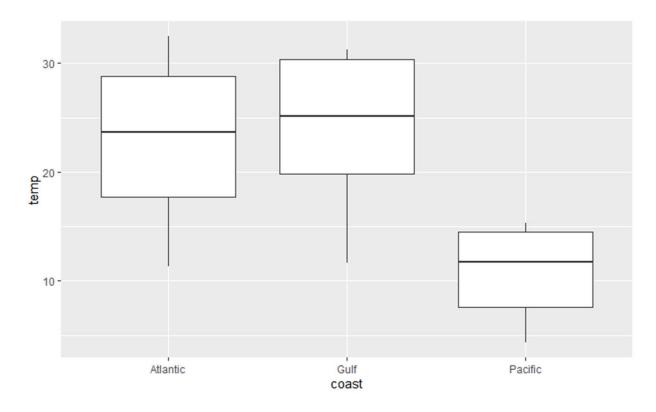
CERF 2019 | The

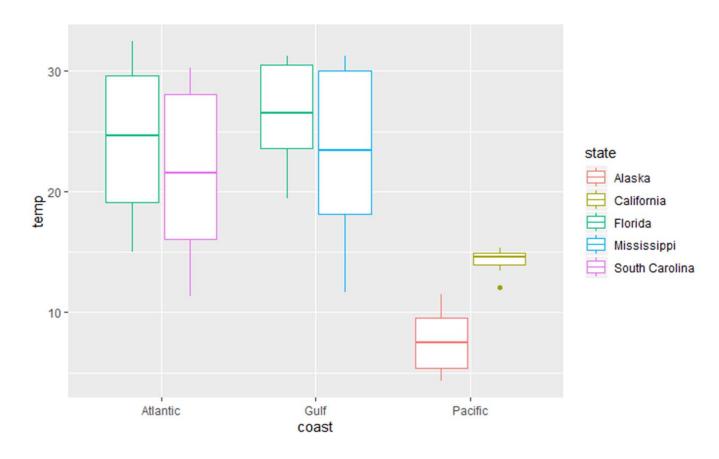
YOUR TURN 4

With a partner, decide how to replace the scatterplot with one that draws boxplots. Use the cheatsheet and try your best guess!





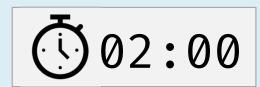


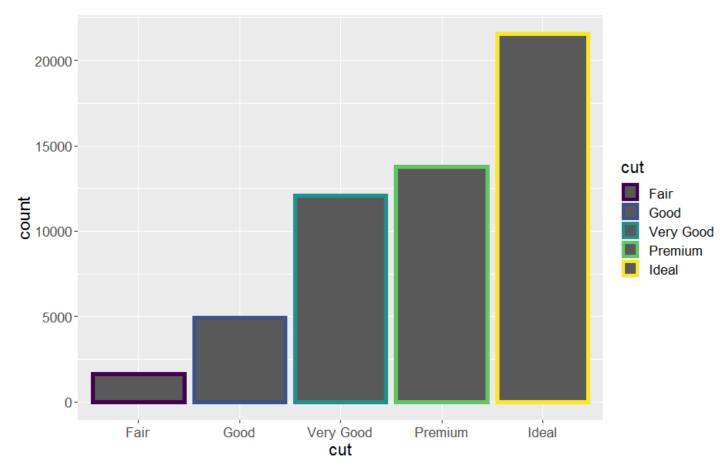


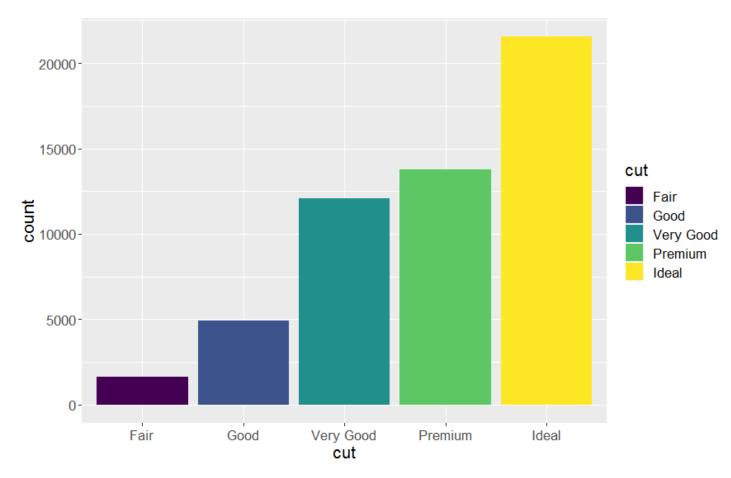
YOUR TURN 5

Load the `diamonds` dataset and examine it. This dataset is provided in the `ggplot2` package.

Create a bar chart of cut colored by cut. Use the cheatsheet.



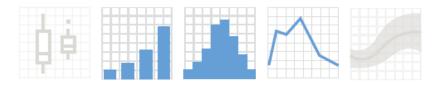




Hold up...count is not a variable in `diamonds`!

Many graphs plot the raw values of your dataset while others calculate new values to plot:

 Bar charts, histograms, and frequency polygons bin data and plot bin counts (# of points that fall into each bin)



Hold up...count is not a variable in `diamonds`!

Many graphs plot the raw values of your dataset while others calculate new values to plot:

• Smoothers, geom_smooth(), fit a model to your data and plot the predictions from the model



Hold up...count is not a variable in `diamonds`!

Many graphs plot the raw values of your dataset while others calculate new values to plot:

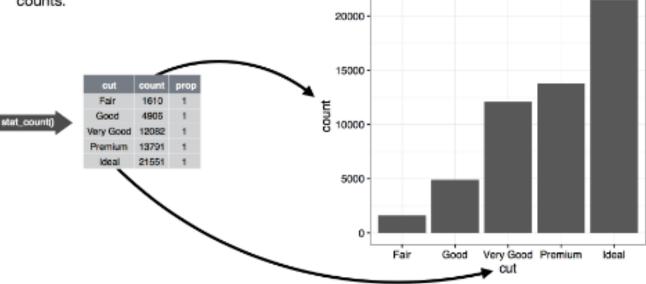
 Boxplots compute a summary of the distribution and display it in a formatted box



Hold up...count is not a variable in `diamonds`!

 geom_bar() begins with the diamonds data set geom_bar() transforms the data with the "count" stat, which returns a data set of cut values and counts. geom_bar() uses the transformed data to build the plot. cut is mapped to the x axis, count is mapped to the y axis.







stat

Short for "statistical transformation" - the algorithm that is used to create new values for a graph.

One of the five components of each layer!

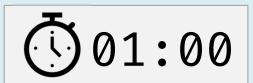
Learn what stat a geom uses by inspecting the default value for the stat argument.

Ex: ?geom_bar

YOUR TURN 6

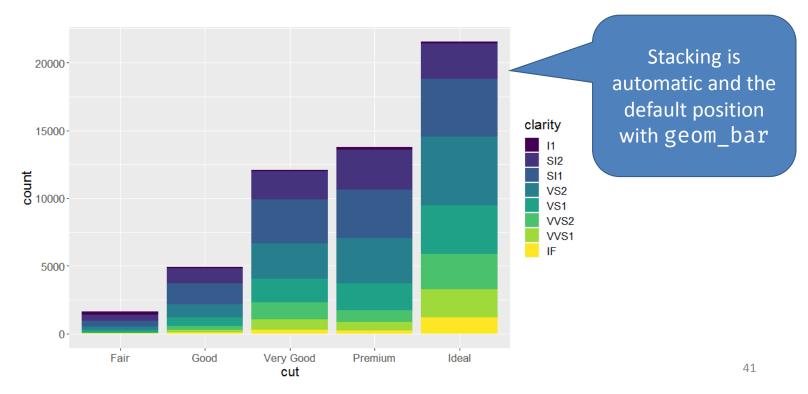
Determine what the default stat is for each of the following geoms:

- `geom_line`
- `geom_density`
- `geom_smooth`

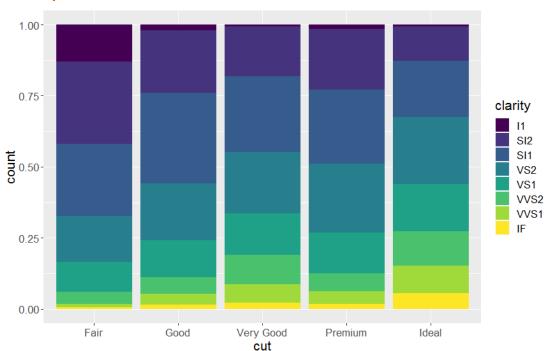


Position: how overlapping objects are arranged

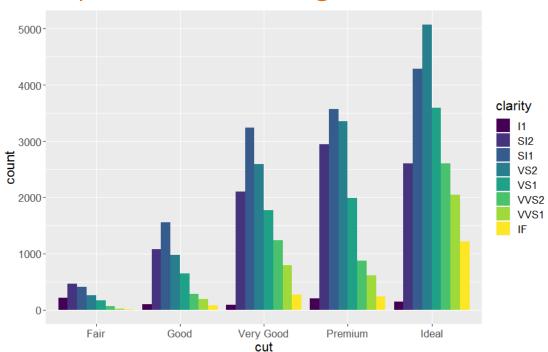
```
ggplot(data = diamonds) +
   geom_bar(mapping = aes (x = cut, fill = clarity))
```



Position: how overlapping objects are arranged



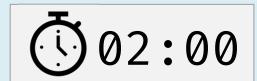
Position: how overlapping objects are arranged



YOUR TURN 7

Predict what this code will do, then run it.

```
ggplot(wq) +
  geom_point(aes(temp, do_mgl)) +
  geom_smooth(aes(temp, do_mgl))
```



Global vs. Local

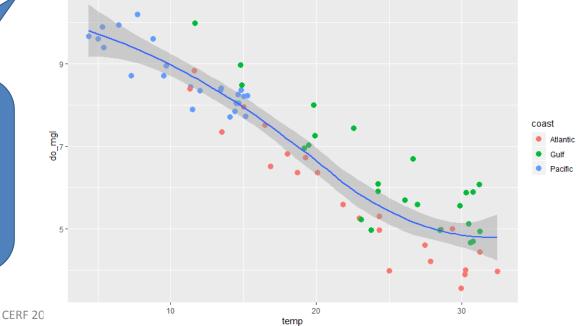
ggplot(data = wq, mapping = aes(x = temp, y = do_mgl)) +
 geom_point() +
 geom_smooth()

Mappings and data that
 appear in ggplot() will
 apply globally to each layer

Global vs. Local

ggplot(data = wq, mapping = aes(x = temp, y = do_mgl)) +
 geom_point(mapping = aes(color = coast)) +
 geom_smooth()

Mappings applied in geom_function will add or override the global mappings for that layer only



46

1. Pick your data set

ggplot(data = <DATA>) +

1. Pick your data set

```
ggplot(data = <DATA>) +
     <GEOM FUNCTION>()
```

2. Choose the best **geom** to display cases

1. Pick your **data** set

2. Choose the best **geom** to display cases

3. **Map** aesthetic properties to variables.

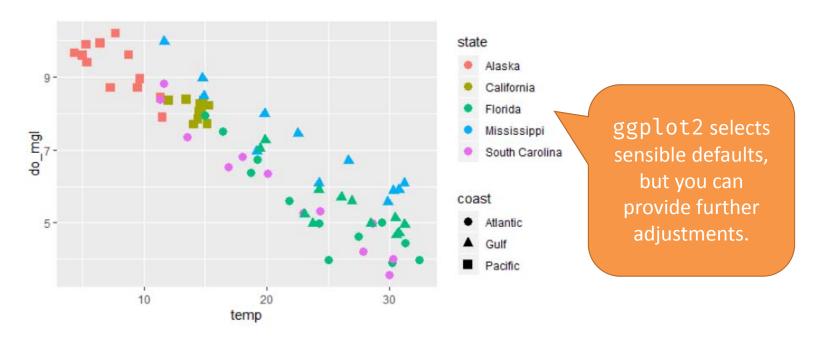
How about a break?

15 minutes

What else can I do?

Scales, themes, and saving your plot.

Scales control the way your data is mapped to your geom and are required for every aesthetic.



```
ggplot(wq, aes(x = temp, y = do_mgl)) +
  geom_point(aes(shape = coast, color = state), size = 3)
```

The specific scale function you use is dependent on the type of scale

```
US Coast

Atlantic

Gulf

Pacific

US State

Alaska

California

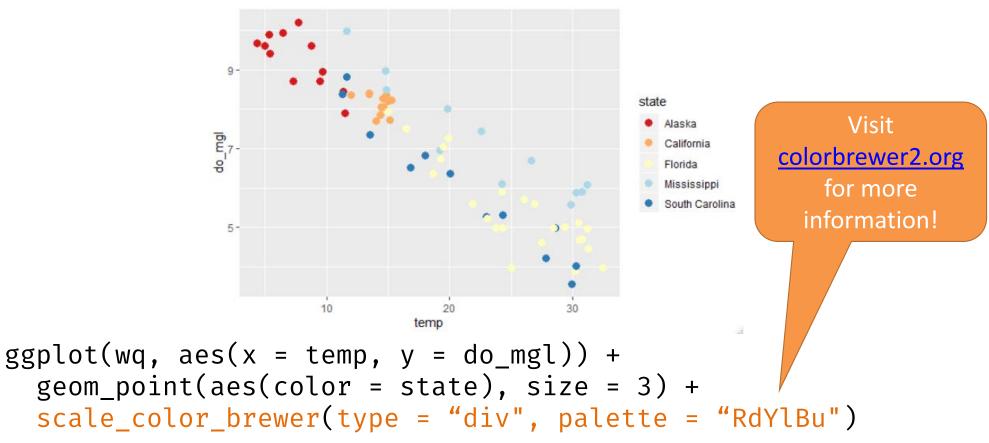
Florida

Mississippi

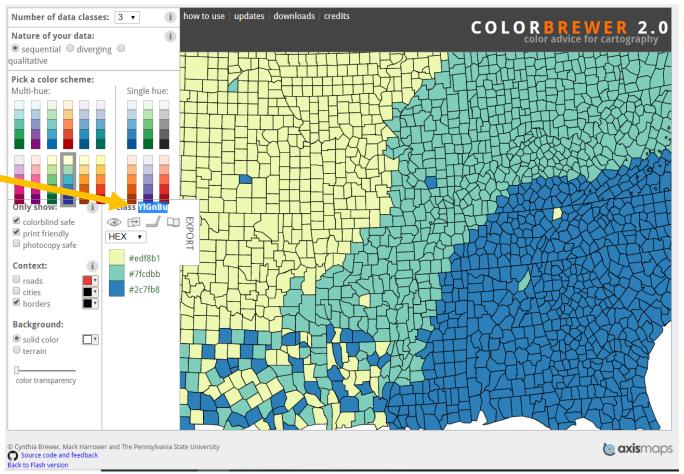
South Carolina
```

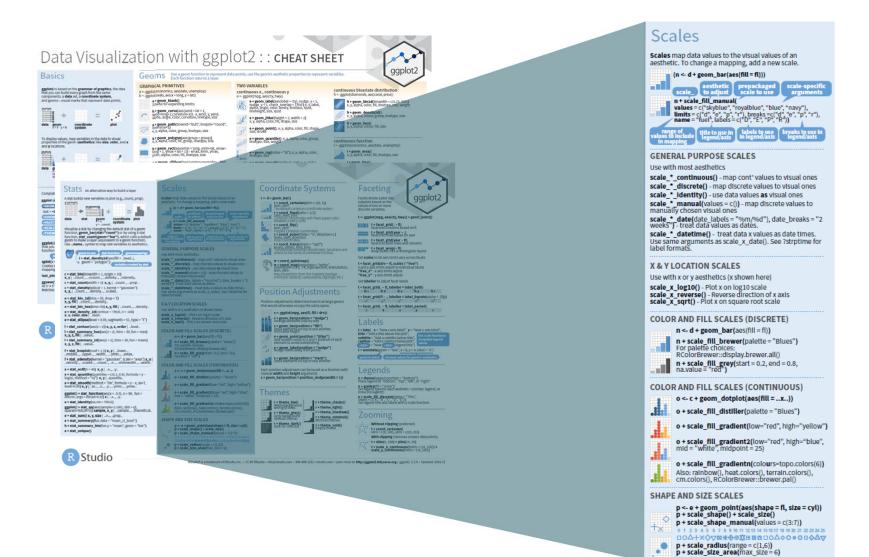
```
ggplot(wq, aes(x = temp, y = do_mgl)) +
   geom_point(aes(shape = coast, color = state), size = 3) +
   scale_shape_discrete(name = "US Coast") +
   scale_color_discrete(name = "US State") +
   scale_y_continuous(breaks = c(2, 4, 6, 8, 10)) +
   scale_x_continuous(breaks = c(5, 10, 15, 20, 25, 30, 35))
```

scale_color_brewer()

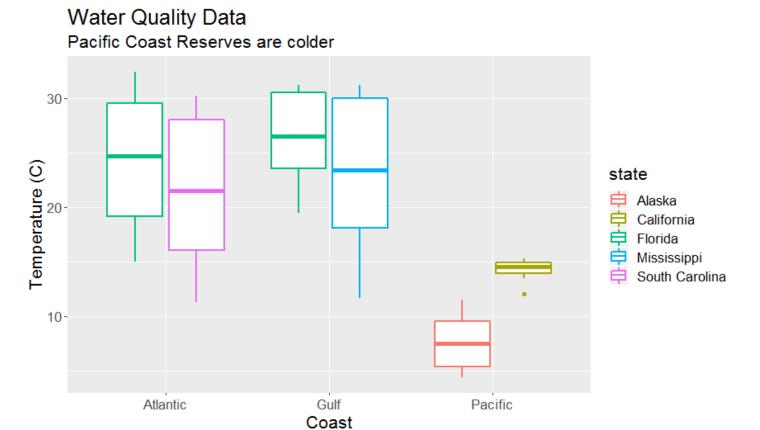


www.colorbrewer2.org

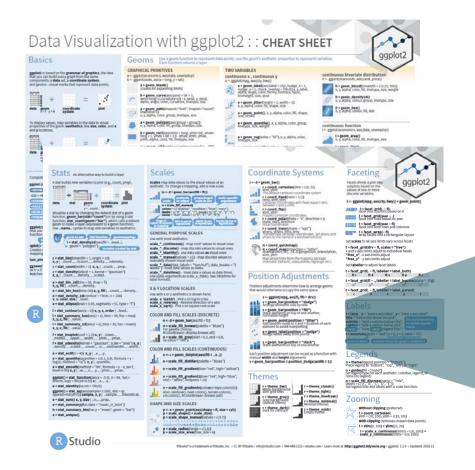


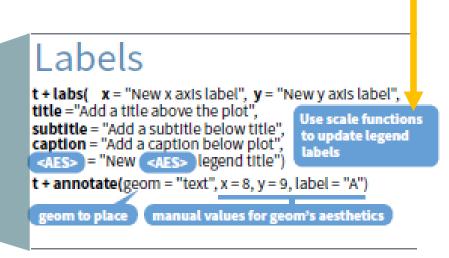


Titles and Captions



Titles and Captions



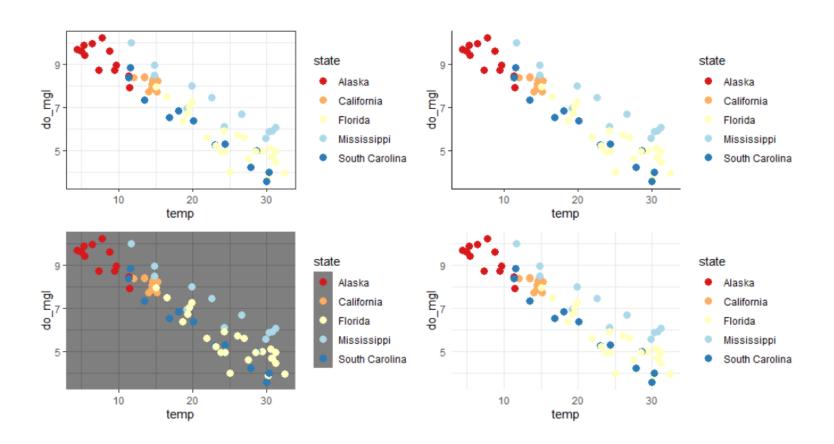


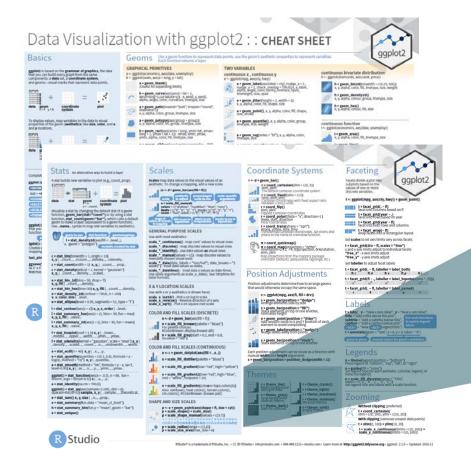
Themes: the non-data elements of your plot.

Do not affect how the data is:

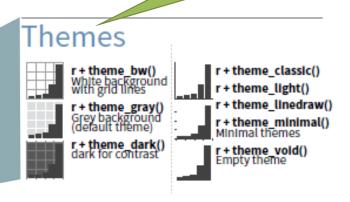
- rendered by geoms, or
- transformed by scales.

Help you make the plot aesthetically pleasing or match an existing style guide (fonts, ticks, panel strips, backgrounds, etc).





The ggthemes package by Jeffrey Arnold provides even more complete themes!



Structure of the theme system

Three components:

- 1. theme() function
- 2. Theme elements (e.g., plot.title element which controls the appearance of the plot title)
- 3. Element function (of which there are four)
 - 1. element_text()
 - 2. element_line()
 - 3. element_rectangle()
 - 4. element_blank()

Structure of the theme system

Explore all the different elements by going to the help page ?theme

ggThemeAssist

Saving your plot

YOUR TURN 8

What does this command return?

getwd()



The Working Directory

R associates itself with a folder (i.e., directory) on your computer.

- This is the "working directory"
- R will look for files here
- R will save files here

The files pane of RStudio
IDE will display your
working directory

Saving plots

```
ggsave() will save the last plot.
ggsave("my-first-plot.png")
ggsave("my-first-plot.tiff")
```

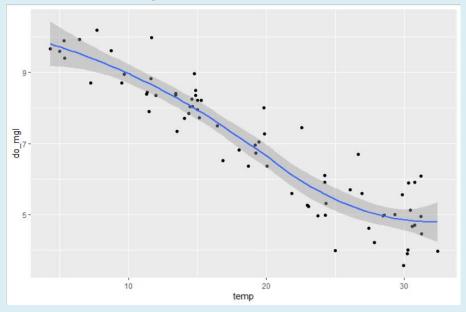
You can further specify size:

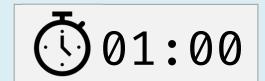
Or even resolution:

```
ggsave("my-first-plot.png", dpi = 300)
```

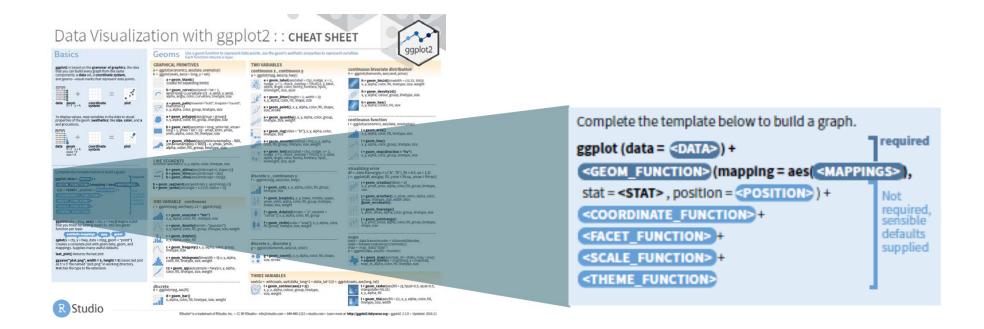
YOUR TURN 9

Save your last plot and then locate it in your files pane. (You may have to refresh your files list).





ggplot2 template



CHALLENGE

Using the diamonds data make a scatterplot of carat by price and assign color by clarity.

Customize the themes and colors of the plot to make it the *ugliest* plot that you can.

Save this plot in your files pane.

