# Advanced ggplot2 techniques

Angela Zoss

10/25/19

https://github.com/amzoss/adv-ggplot2-F19

Try right now:
Open RStudio
Try running “library(tidyverse)”
Tell me about any errors

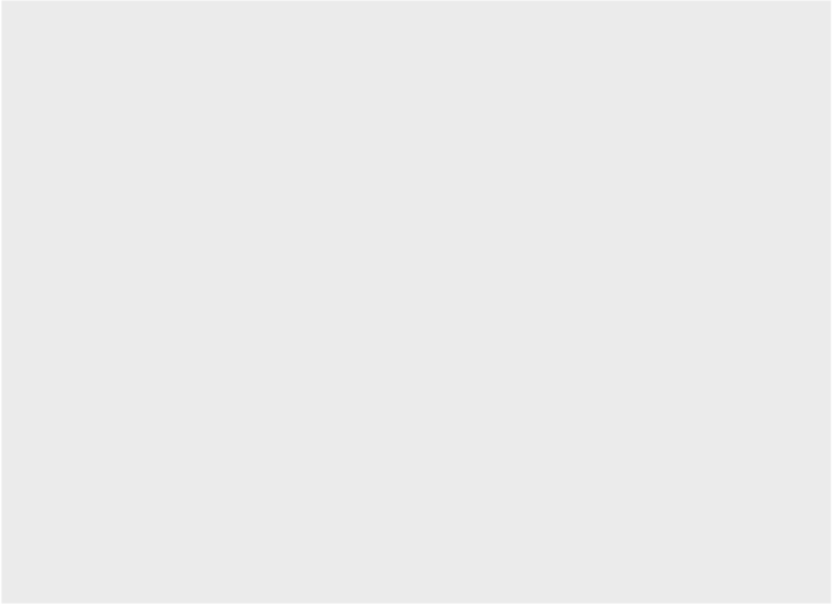
# ggplot2 refresher

# 1. Set the data

“iris”

| Petal.Width | Petal.Length | Species |
|---|---|---|
| 0.3 | 1.4 | setosa |
| 1.3 | 4.0 | versicolor |
| 2.1 | 5.7 | virginica |

```
ggplot(data=iris)
```



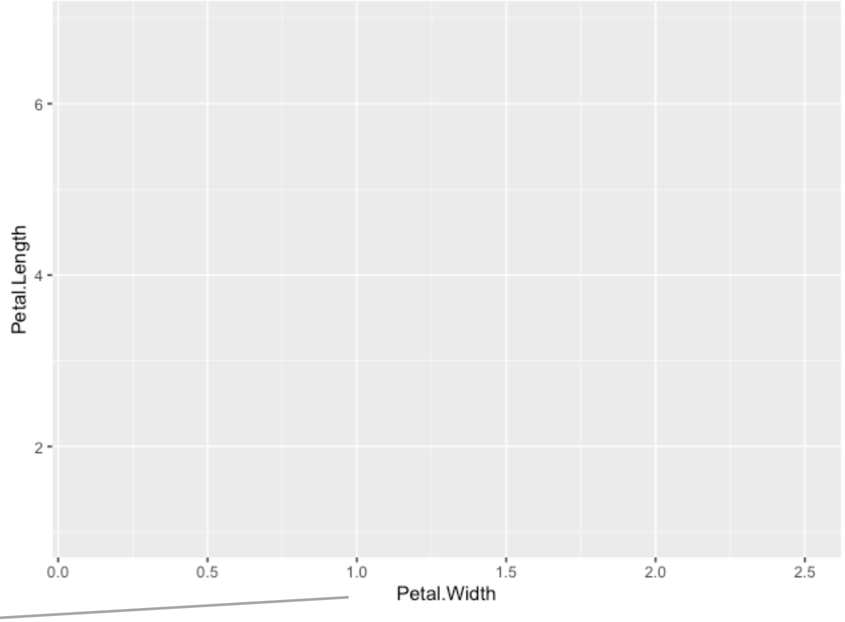
# 2. Map variables to aesthetics

“iris”

| Petal.Width | Petal.Length | Species |
|---|---|---|
| 0.3 | 1.4 | setosa |
| 1.3 | 4.0 | versicolor |
| 2.1 | 5.7 | virginica |

x position y position color

```
ggplot(data=iris,
        aes(x=Petal.Width, y=Petal.Length,
            color=Species))
```

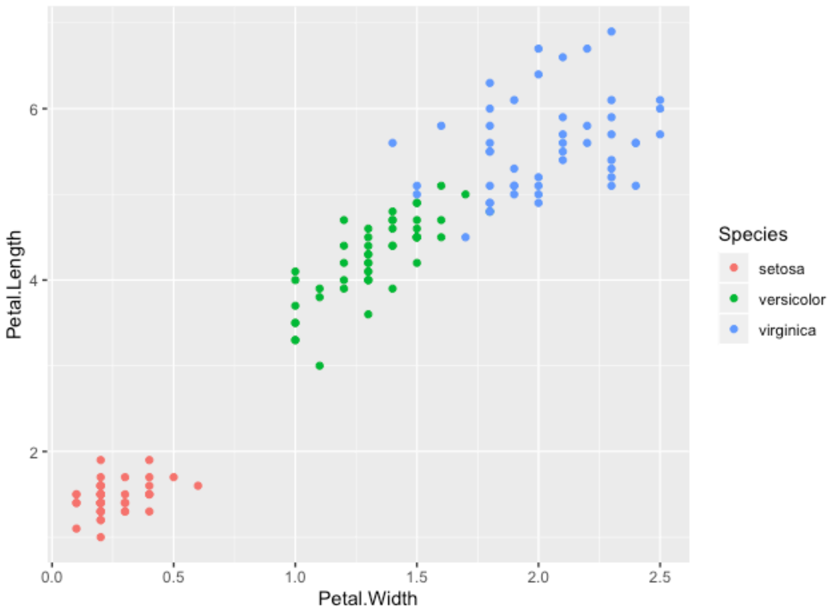


# 3. Choose a shape layer

“iris”

| Petal.Width | Petal.Length | Species |
|---|---|---|
| 0.3 | 1.4 | setosa |
| 1.3 | 4.0 | versicolor |
| 2.1 | 5.7 | virginica |

```
ggplot(data=iris,
          aes(x=Petal.Width, y=Petal.Length,
               color=Species)) +
geom_point()
```

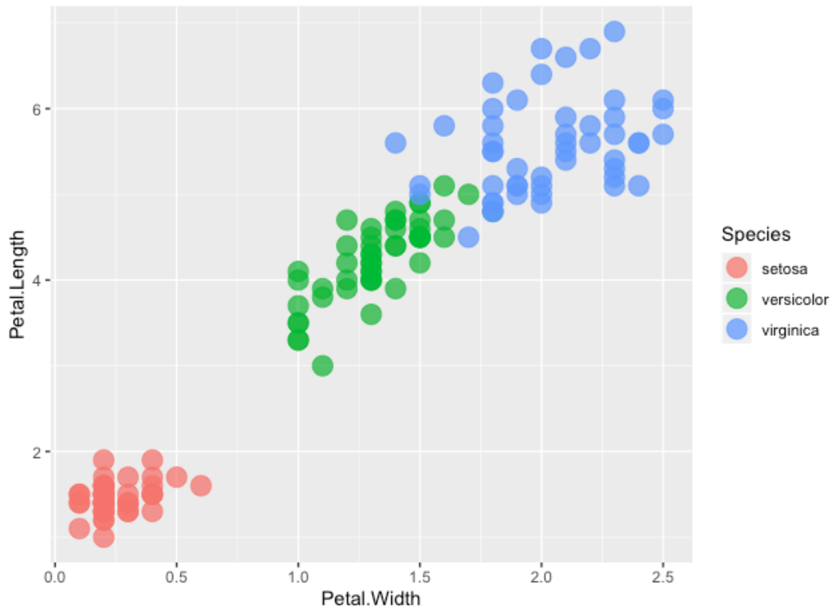


# 4. Add non-variable adjustments

“iris”

| Petal.Width | Petal.Length | Species |
|---|---|---|
| 0.3 | 1.4 | setosa |
| 1.3 | 4.0 | versicolor |
| 2.1 | 5.7 | virginica |

```
ggplot(data=iris,
        aes(x=Petal.Width, y=Petal.Length,
            color=Species)) +
geom_point(size=5, alpha=.75)
```

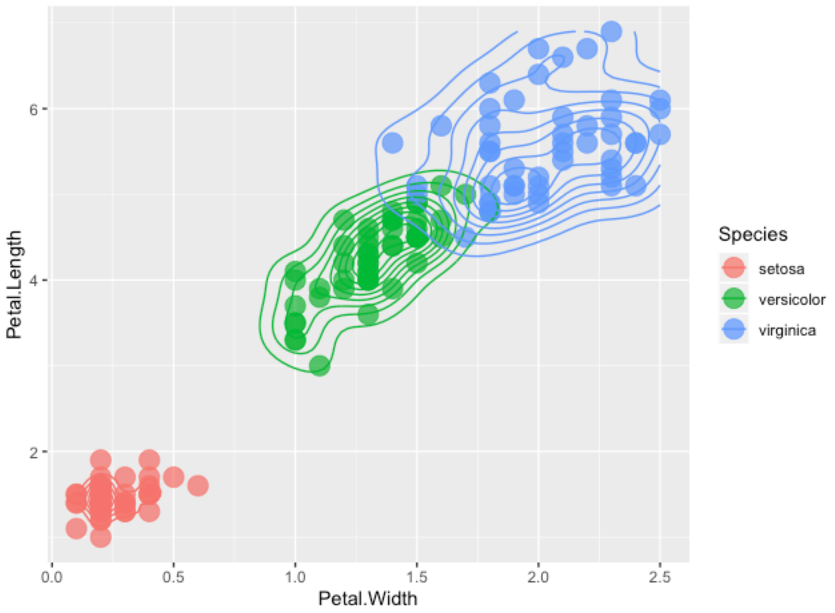


# Adding a new shape layer: geom_density2d()

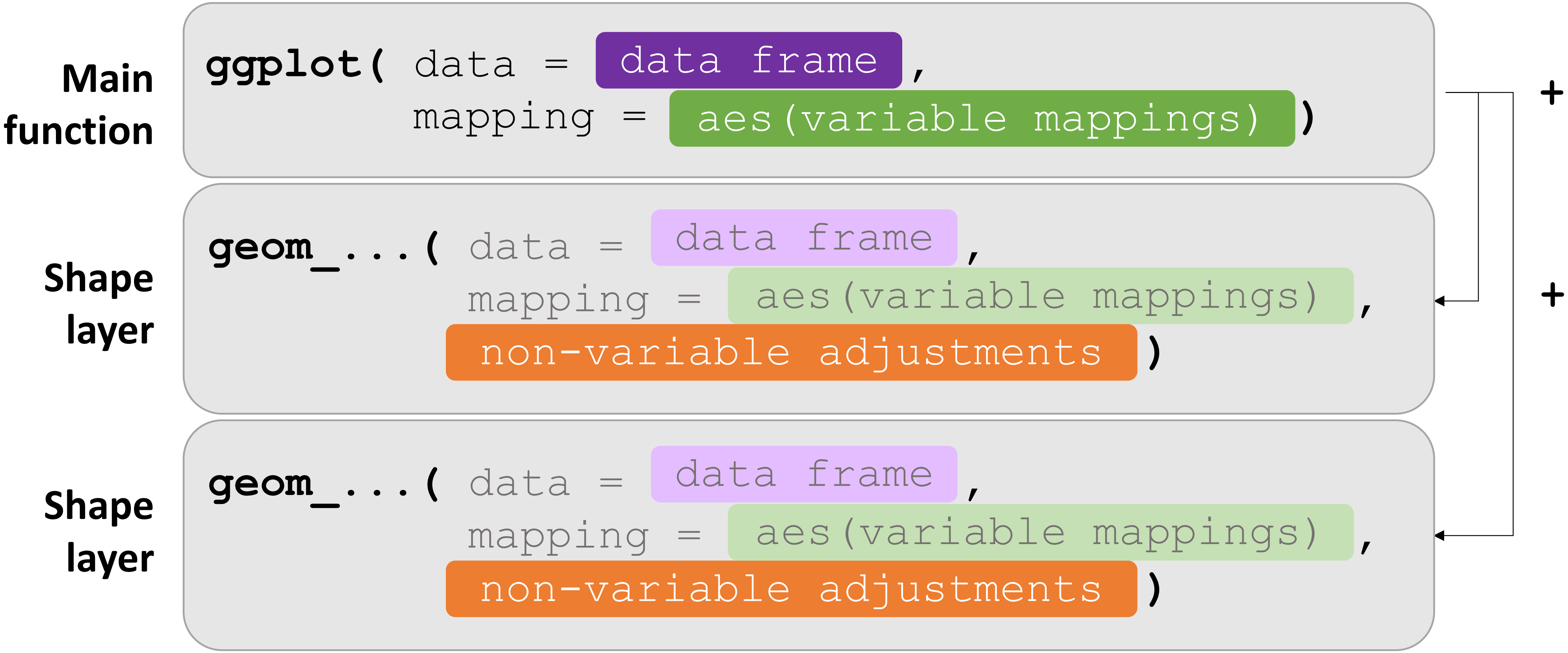
“iris”

| Petal.Width | Petal.Length | Species |
|---|---|---|
| 0.3 | 1.4 | setosa |
| 1.3 | 4.0 | versicolor |
| 2.1 | 5.7 | virginica |

```
ggplot(data=iris,
          aes(x=Petal.Width, y=Petal.Length,
               color=Species)) +
geom_point(size=5, alpha=.75) +
geom_density2d()
```



# General syntax



**Main function**

```
ggplot( data = data frame,
        mapping = aes(variable mappings))
```

+

**Shape layer**

```
geom_...( data = data frame,
          mapping = aes(variable mappings),
          non-variable adjustments)
```

+

**Shape layer**

```
geom_...( data = data frame,
          mapping = aes(variable mappings),
          non-variable adjustments)
```

# Working in RStudio

# Using RStudio

- Projects
- R Markdown
- Cheat sheets

https://www.rstudio.com/resources/cheatsheets/#rmarkdown

# Don’t have it installed?

https://vm-manage.oit.duke.edu/containers

# Create a new project with workshop files

URL: https://github.com/amzoss/adv-ggplot2-F19

- Click green button to download ZIP
- Unzip files on your laptop

**In RStudio:**

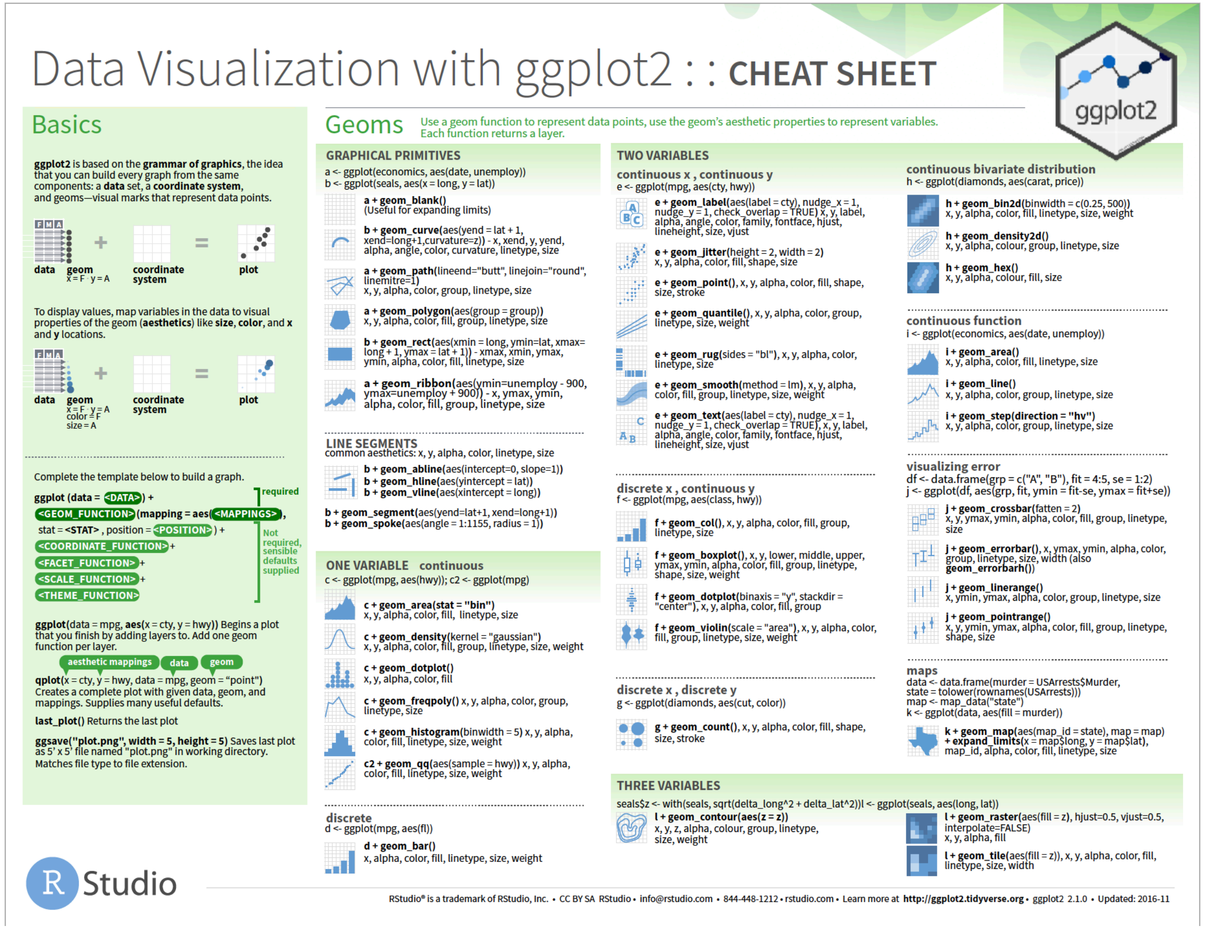
- Project → New project…
- Existing directory
- Select unzipped folder
- Create Project

# ggplot2 Cheat Sheet

Help →

Cheatsheets →

Data Visualization with ggplot2



# Data Visualization with ggplot2 : : CHEAT SHEET

## Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

**data** **geom** x = F · y = A + **coordinate system** = **plot**

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

**data** **geom** x = F · y = A color = F size = A + **coordinate system** = **plot**

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                       required
<GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
 stat = <STAT> , position = <POSITION>) +                      Not
<COORDINATE_FUNCTION>+                                         required,
<FACET_FUNCTION>+                                              sensible
<SCALE_FUNCTION>+                                              defaults
<THEME_FUNCTION>                                               supplied
```

**ggplot**(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings · data · geom

**qplot**(x = cty, y = hwy, data = mpg, geom = “point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

**last_plot()** Returns the last plot

**ggsave("plot.png", width = 5, height = 5)** Saves last plot as 5’ x 5’ file named "plot.png" in working directory. Matches file type to file extension.

## Geoms

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

### GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

**a + geom_blank()**
(Useful for expanding limits)

**b + geom_curve**(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

**a + geom_path(**lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

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

**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

### LINE SEGMENTS

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

**b + geom_abline**(aes(intercept=0, slope=1))
**b + geom_hline**(aes(yintercept = lat))
**b + geom_vline**(aes(xintercept = long))

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

### ONE VARIABLE continuous

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

**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_dotplot()**
x, y, alpha, color, fill

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

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

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

### discrete

d <- ggplot(mpg, aes(fl))

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

### 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, family, fontface, hjust, lineheight, size, vjust

**e + geom_jitter**(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

**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

**e + geom_text**(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

### discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

**f + geom_col()**, x, y, alpha, color, fill, group, linetype, size

**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_bin2d**(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

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

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

### continuous function

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_id = state), map = map) **+ expand_limits**(x = map$long, y = map$lat), map_id, alpha, color, fill, linetype, size

### THREE VARIABLES

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

**l + geom_contour(aes(z = z))**
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

**l + geom_tile**(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width

R Studio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at **http://ggplot2.tidyverse.org** • ggplot2 2.1.0 • Updated: 2016-11

https://www.rstudio.com/resources/cheatsheets/#ggplot2

# Summary

Adding something that will appear
inside the **chart coordinate space**?

You will (almost always) be adding a **geom**!

Changing the way a **variable is displayed**?
(e.g., different axis breaks, different color mapping)

You will be adding a **scale**!

Changing the **look and feel** of the chart?

You will be adding or making changes to a **theme!**

# Exercises in RStudio

# Final advice

# Reminder

Adding something that will appear inside the **chart coordinate space**?

You will (almost always) be adding a **geom**!

Changing the way a **variable is displayed**?
(e.g., different axis breaks, different color mapping)

You will be adding a **scale**!

Changing the **look and feel** of the chart?

You will be adding or making changes to a **theme!**

# Debugging code

- Start simple
- If you see an error:
  - read error message for hints
  - check for problems with spelling/punctuation marks
- Get code to run without errors
- Check result to see if it makes sense

- Add a small change
- Get code to run without errors
- Check result to see if it makes sense
- etc.

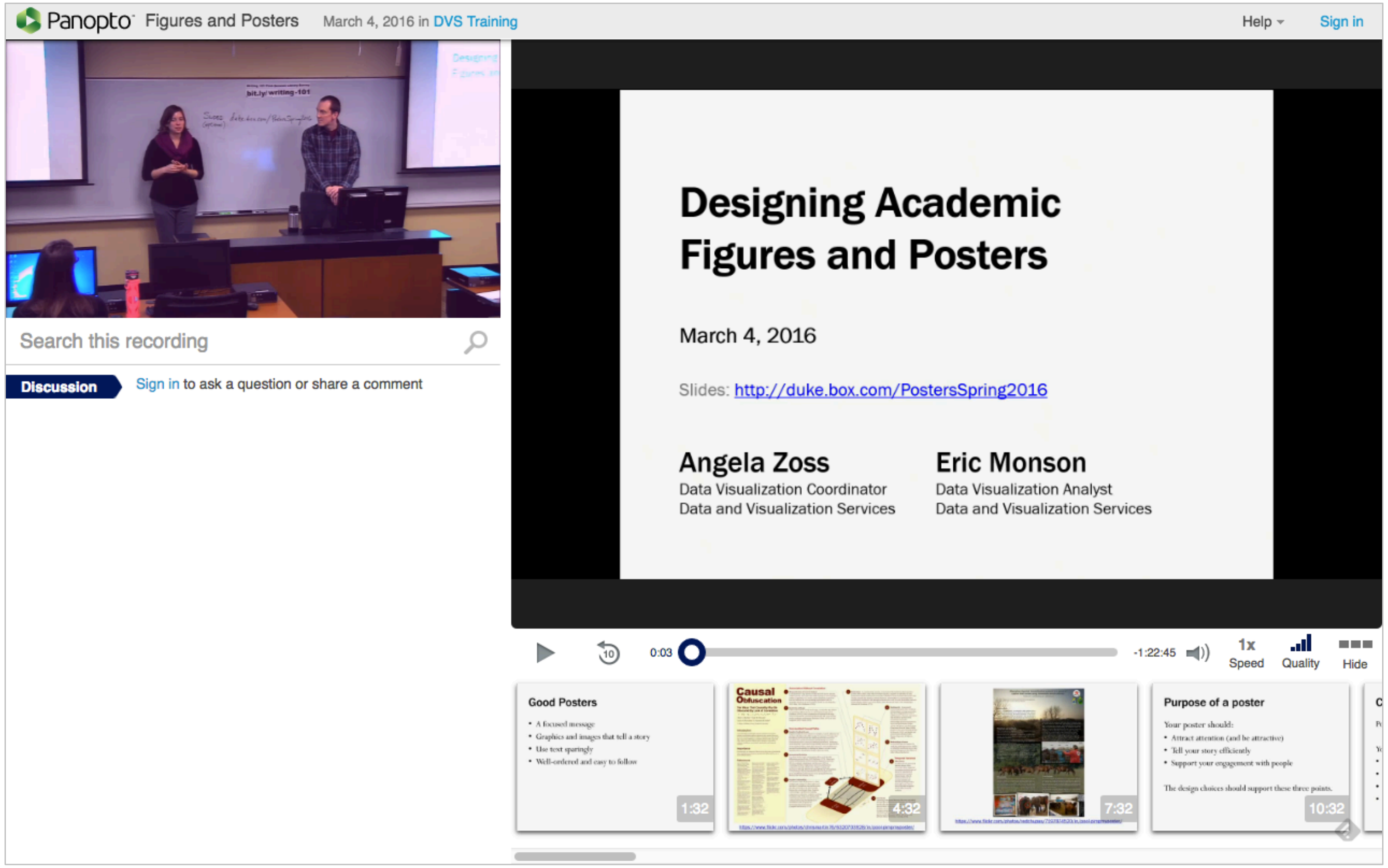
# Other helper packages

- gganonymize to randomize text in ggplot2 figures
- visdat to visualize variable classes and missing data
- ggthemes for additional themes and scales, especially ones that match software defaults (e.g., Tableau)
- esquisse for building ggplot2 charts interactively
- colorblindr for simulating color vision deficiency
- ggpubr for publication-ready plots

# ggplot2 Resources

- General ggplot2 information
- R Graphics Cookbook (recipes for plots)
- R for Data Science (online book that includes ggplot2)
- ggplot2: Elegant Graphs for Data Analysis (book by Hadley Wickham)
- ggplot2 cheatsheet (also in RStudio)
- Data Carpentry lesson on ggplot2
- Data Visualization: A Practical Introduction, by Kieran Healy
- RStudio "Visualize Data" Primer

# Videos of past workshops



**Designing Academic Figures and Posters**

March 4, 2016

Slides: http://duke.box.com/PostersSpring2016

**Angela Zoss**
Data Visualization Coordinator
Data and Visualization Services

**Eric Monson**
Data Visualization Analyst
Data and Visualization Services

Good Posters

- A focused message
- Graphics and images that tell a story
- Use text sparingly
- Well-ordered and easy to follow

Causal Obfuscation

Purpose of a poster

Your poster should:

- Attract attention (and be attractive)
- Tell your story efficiently
- Support your engagement with people

The design choices should support these three points.

[http://bit.ly/DVSvideos](http://bit.ly/DVSvideos)

# Questions?

angela.zoss@duke.edu