## Layers on Layers on Layers

### **Last Time**

#### **Multipanel Plots**

- Faceting
- Combining separate plots into 1 cohesive figure

### **Adding Stuff**

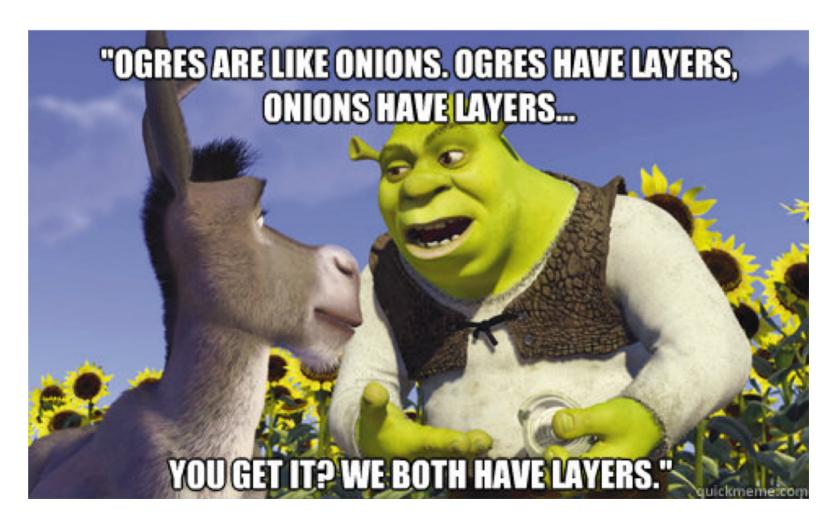
- Lines
- Text

## This time

#### Miscellaneous & Remaining Details

- Bar plots & Error bars
- Jittering
- Adding layers
- Misc.

## Layers on layers on layers



- Usually bar plots reflect the *mean* of a group (or at least some summary statistic)
- Just like vertical/horizontal lines, we'll need to transform our raw data into summary statistics
- If you decide to work with summary stats, you have a few options:
  - You can calculate the summaries within the ggplot code
  - Calculate the summaries as their own data.frame, then call it from inside the ggplot function
  - I personally suggest the latter

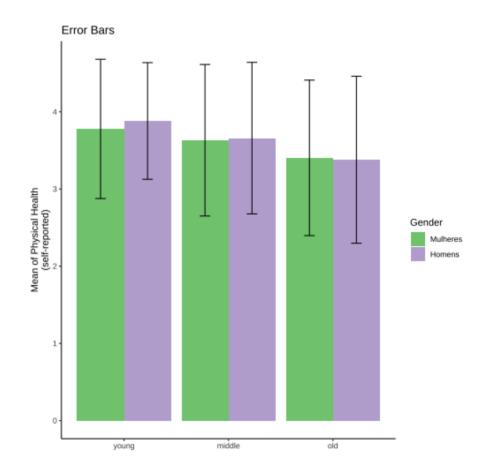
- When it comes to error bars, you need to decide what your bars will reflect and calculate the appropriate statistic!
  - 1 standard deviation
  - 95% confidence intervals
  - 1 standard error of the mean
- For now, we'll stick to +/- 1 standard deviation

#### Step 1: Summarize your data

```
## # A tibble: 6 x 6
## # Groups: age_category [3]
  age_category sex means sds sdLower sdUpper
##
  <fct> <fct> <dbl> <dbl> <dbl> <dbl>
##
## 1 young Female 3.78 0.902 2.88 4.68
## 2 young Male 3.88 0.755 3.13 4.64
## 3 middle Female 3.63 0.981 2.65 4.61
## 4 middle Male 3.66 0.981 2.68 4.64
          Female 3.40 1.01 2.40 4.41
## 5 old
## 6 old
          Male 3.38 1.08
                            2.30
                                     4.46
```

#### Now we can plot

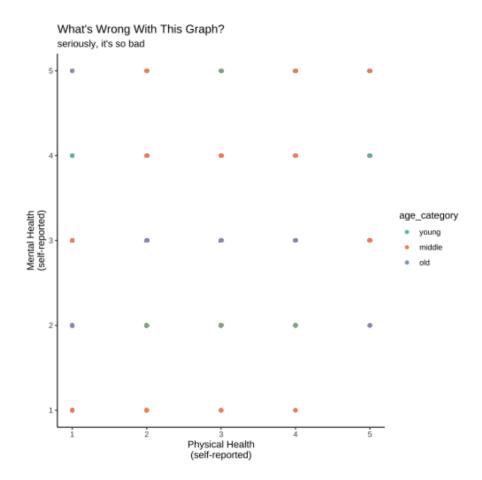
```
ggplot(data = summaryStats,
      aes(x = age_category,
          v = means
          fill = sex)) +
 geom_col(position = position_dodge(width=
 geom_errorbar(aes(ymin = sdLower,
                   ymax = sdUpper),
                position = position_dodge(w
                width = .2) +
 theme_classic() +
 scale_fill_brewer(palette = "Accent",
                    labels = c("Mulheres",
                               "Homens")) +
 labs(title = "Error Bars",
      x = "",
      y = "Mean of Physical Health\n(self-
       fill = "Gender")
```



# **Jittering**

Why does this plot look so weird?

 (Hint, how many observations do we have in the midus data set?)



## **Jittering**

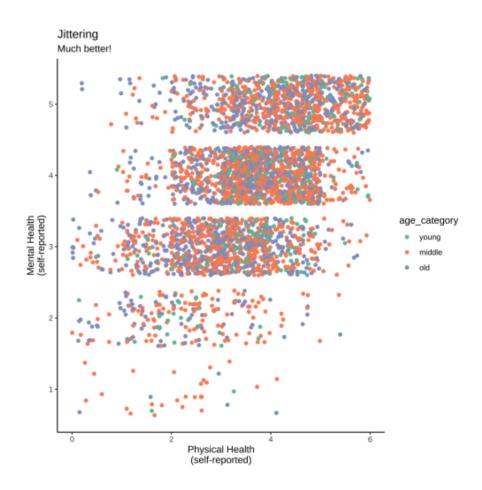
The way we can get around this is with **jittering** 

- A **jitter** is a slight irregular movement, variance, or unsteadiness
- Someone is jittery means someone is *shaky*, usually with nervousness

We can **jitter** the points on the x-axis to randomly shift them. This let's us see *all* of the points.

We are adding in some random variance to make things more visible.

# **Jittering**



## Layers on Layers on Layers

- By now you've hopefully realized that you can add as many layers as you'd like to your ggplot
- This means you can use multiple shapes or geom\_s from the same data on the same plot
- Be careful!
  - You are layering geoms on top of each other
  - Order matters depending on what you're doing!

# Layers on Layers: Line Graph Example

```
ggplot(data = summaryStats,
       aes(x = age_category,
           v = means,
           group = sex)) +
  geom_point(aes(color = sex,
                 shape = sex).
             size = 8) +
  geom_errorbar(aes(ymin = sdLower,
                   vmax = sdUpper,
                    color = sex),
                width = .1,
                linetype = "dashed") +
 ylim(c(2,5)) +
 geom_line(aes(color = sex)) +
  theme_classic() +
 labs(y = "Mean Physical Health",
      x = "",
      color = "",
      shape = "",
      title = "Line Graph")
```

# Layers on Layers: Line Graph Example

#### Order switch

```
ggplot(data = summaryStats,
       aes(x = age_category,
           y = means,
           group = sex)) +
  geom errorbar(aes(ymin = sdLower,
                    ymax = sdUpper,
                    color = sex),
                width = .1,
                linetype = "dashed") +
  ylim(c(2,5)) +
  geom line(aes(color = sex)) +
  geom_point(aes(color = sex,
                 shape = sex).
             size = 8) +
 theme_classic() +
  labs(y = "Mean Physical Health",
      x = "",
      color = "",
      shape = "",
       title = "Line Graph")
```

- When combining mutliple geoms, the order matters!
- Look what happens when you switch geom\_violin and geom\_point:

```
ggplot(data = midus,
   aes(x = age_category,
        y = BMI)) +
geom_jitter(aes(color = age_category)
        width = .2,
        alpha = .3) +
geom_violin(aes(color = age_category)
        fill = "white") +
theme_classic() +
labs(title = "Distribution of BMI",
   subtitle = "Per Age Category",
   x = "",
   color = "")
```

- Adding a 3rd geom
- Not quite right...
- This is a great example of jittering!

```
ggplot(data = midus,
    aes(x = age_category,
        y = BMI)) +
geom_jitter(aes(color = age_category)
        width = .2,
        alpha = .7) +
geom_violin(aes(fill = age_category),
        alpha = .3) +
geom_boxplot(fill = "white",
        width = .2) +
theme_classic() +
labs(title = "Distribution of BMI",
    subtitle = "Per Age Category",
    x = "",
    color = "")
```

 Where should I look to figure out how to fix this?

```
ggplot(data = midus,
   aes(x = age_category,
       \vee = BMI)) +
geom jitter(aes(color = age category)
         width = .2,
         alpha = .7) +
geom violin(aes(fill = age category),
          alpha = .3) +
geom_boxplot(fill = "white",
           width = .2,
           outlier.shape = NA) +
theme classic() +
labs(title = "Distribution of BMI",
   subtitle = "Per Age Category",
   x = "",
   color = "",
   fill = "")
```

# **Last Remaining Thoughts**

- Idea of "addition" & "piping"
- Exporting plots
- Debugging
- Fun!

### Idea of "addition"

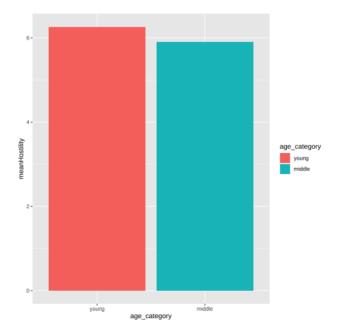
- When you use the + at the end of a line, you are literally telling R that you want to add something -- you're saying "add another layer to the plot"
- As a result, you will see a lot of code, especially in the help documentation, that looks like this:

```
p <- ggplot(data = midus)
p <- p + geom_point(aes(x = age, y = BMI, color = age_category))
p <- p + labs(title = "Plotting by Addition")
p</pre>
```

# Idea of "piping"

- You can also "pipe" ggplot2 code into tidyverse code
- Notice that the pipe %>% changes into a + when you entire ggplot2 code!

```
midus %>%
  filter(age_category != "old") %>%
  group_by(age_category) %>%
  summarize(meanHostility = mean(hostility)) %>%
  ggplot(aes(x = age_category, y = meanHostility)) +
  geom_col(aes(fill = age_category))
```



## **Exporting**

- So you've made a plot. Yay! Now you need to get it out of R and into a file format you can upload (.png, .tiff, .jpf, .pdf etc.)
- Method 1: ggsave() function
  - It should work most of the time
  - Simplier, easier
  - Not as much control
- Method 2: Turn on/off the graphic device
  - Better if you need something specific
  - Specific resolution, compression etc.
  - Slightly more annoying, but not by much

IMPORTANT: You must consider your working directory! Your plot will save to your working directory

## **Exporting**

```
# Method 2: graphic device
# First, you call the device
# Then you plot,
# Then you turn the device off

tiff(filename = "plotSave2.tiff",
    width = 7,
    height = 7,
    units = "in",
    res = 300,
    compression = "lzw")

ggplot(data = data,
    aes(x = x, y = y)) +
    geom_point()

dev.off() # Leave parentheses empty.
```

If you get a pop-up that's like "quartz\_off\_screen" or something, that's OK.

# Debugging

If your code isn't working, and you really think that it should, you might be using a function that comes from a different package.

- Ex: alpha exists in both the ggplot2 and psych packages
- If it gives you trouble, just specify the package like this ggplot2::alpha

If you still want to cry 😥 because your plot still isn't giving you what you want, try:

- Store the plot as an object
- Then look at its inner workings using ggplot\_build(plot0bject) where plot0bject is the name of your plot

Finally, when looking around the internet for help, make sure that the version number is kind of close to the one you're working with (usually doesn't need to be exact). The tidyverse including ggplot2 has been around for some time now, and they have gone through many iterations over the years.

### Want to look at pictures of cute puppies in RStudio?

• pupR package!

```
# use this line of code to install the package
# devtools::install_github("melissanjohnson/pupR")

library(pupR)
pupR()
```

Want to look at pictures of cute puppies in RStudio?

• pupR package!

```
# use this line of code to install the package
# devtools::install_github("melissanjohnson/pupR")

library(pupR)
pupR(dog_type = "basset")
```

For those of you who get bored and like to procrastinate with remarkably dumb things, you can make your own XKCD plot in R!

• Use the xkcd package and the extrafont packages

library(xkcd)
library(extrafont)

```
set.seed(1234) #this makes sure the random numbers generated will be the
df <- data.frame(vacc = rnorm(50, sd = .75), #make fake data</pre>
                  autism = rnorm(50, sd = .55))
xrange \langle -c(-2,2) | # specific for xkcd
yrange \langle -c(-2,2) | \# specific for xkcd \rangle
ratioxy <- diff(xrange) / diff(yrange) # specific for xkcd
mapping <- aes(x, y, # specific for xkcd</pre>
                scale,
                ratioxy,
                angleofspine,
                anglerighthumerus,
                anglelefthumerus,
                anglerightradius,
                angleleftradius,
                anglerightleg,
                angleleftleg,
                angleofneck)
```

This code makes the little stick figure dude. You choose the angles of each line. You can use  $\pi$  charts!

Finally, something you know -- ggplot!

### And here's the actual plot

