

# Introduction to R for Biologists

Day 2 – Data visualization with ggplot2

Developed by Rachael Cox

# Refresher from Day 1

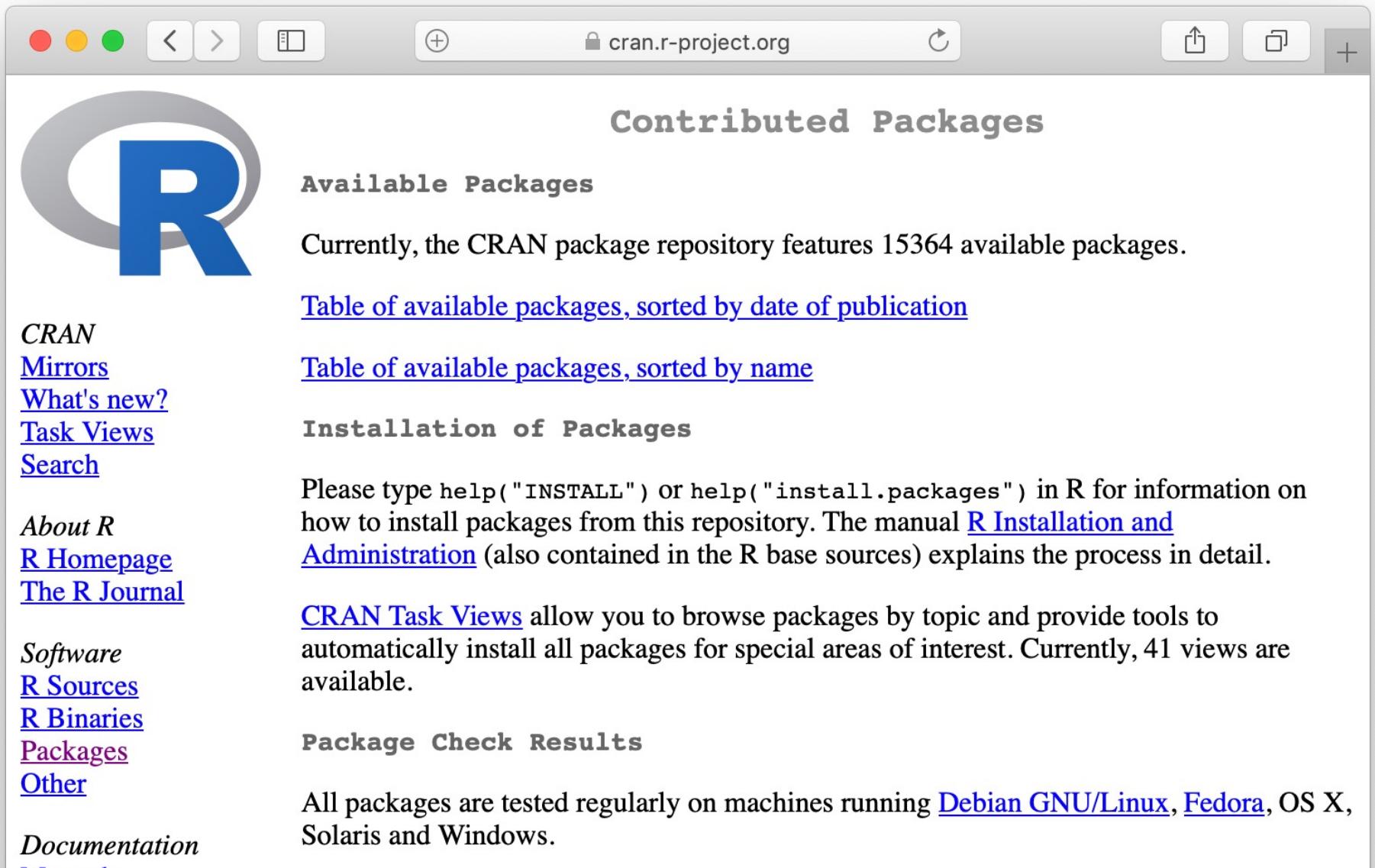
- Rmarkdown: knitting and running code blocks
- Variables: objects, vectors, and strings
- Functions: Arguments, library(), and (?)
  - `Read_csv`

# Day 2 Outline

- Lecture
- What is ggplot?
  - Examples of what ggplot can do
  - How to "build" figures in ggplot
- Workbook Problems 1.1 and 1.2 (20min)
  - Solutions
- Lecture
  - Advanced Examples
- Break (15min)
- Workbook Problems 2 (20min)
  - Solutions
- Workbook Problems 3 (20min)
  - Solutions
- Workbook Problems 4 (20min)

# Extending R through packages: There's a package for everything

# R packages are available on CRAN (Comprehensive R Archive Network)



The screenshot shows a web browser window with the URL `cran.r-project.org` in the address bar. The page title is "Contributed Packages". On the left, there is a large "R" logo. The main content area has several sections: "Available Packages" (with a note about 15364 packages), links to "Table of available packages, sorted by date of publication" and "Table of available packages, sorted by name", "Installation of Packages" (with instructions for R users), "CRAN Task Views" (described as allowing browsing by topic and automatic installation of packages for specific areas of interest), and "Package Check Results" (noted as being tested on Debian GNU/Linux, Fedora, OS X, Solaris, and Windows). On the far left, a sidebar lists various CRAN resources: CRAN Mirrors, What's new?, Task Views, Search, About R, R Homepage, The R Journal, Software, R Sources, R Binaries, Packages, Other, and Documentation.

## Contributed Packages



### Available Packages

Currently, the CRAN package repository features 15364 available packages.

[Table of available packages, sorted by date of publication](#)

[Table of available packages, sorted by name](#)

### Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual [R Installation and Administration](#) (also contained in the R base sources) explains the process in detail.

### CRAN Task Views

[CRAN Task Views](#) allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 41 views are available.

### Package Check Results

All packages are tested regularly on machines running [Debian GNU/Linux](#), [Fedora](#), OS X, Solaris and Windows.

*CRAN*  
[Mirrors](#)  
[What's new?](#)  
[Task Views](#)  
[Search](#)  
  
[About R](#)  
[R Homepage](#)  
[The R Journal](#)  
  
[Software](#)  
[R Sources](#)  
[R Binaries](#)  
[Packages](#)  
[Other](#)  
  
[Documentation](#)

# You can install packages using install.packages( ) in RStudio

```
Console ~/ ↗
Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> install.packages("ggplot2") 
% Total % Received % Xferd Average Speed Time Time Time Current
          Dload Upload Total Spent Left Speed
0      0    0      0      0      0      0 --:--:-- --:--:-- --:--:-- 0 38 1932k
38 751k  0      0  1529k      0  0:00:01 --:--:-- 0:00:01 1527k 100 1932k
0      0  2918k      0 --:--:-- --:--:-- 2918k

The downloaded binary packages are in
/var/folders/q8/wptgtbdn1pz0cfgrz39gq00m0000gn/T//RtmpvQgw1u/downloaded_packages
> |
```

# ggplot2: A grammar of graphics

Traditional plotting: You **are** a painter

- Manually place individual graphical elements

ggplot2: You **employ** a painter

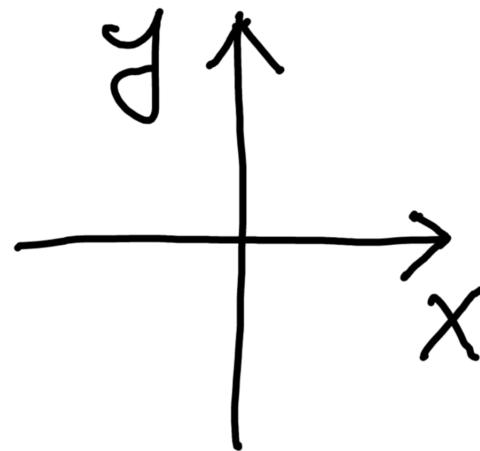
- Describe conceptually how data should be visualized

# Most confusing key concept: aesthetic mapping

Maps data values to visual elements of the plot

# A few examples of aesthetics

position



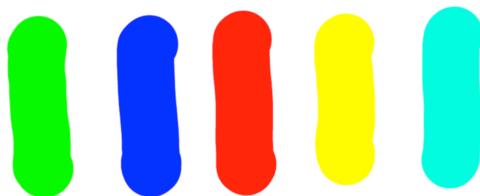
shape



size



color

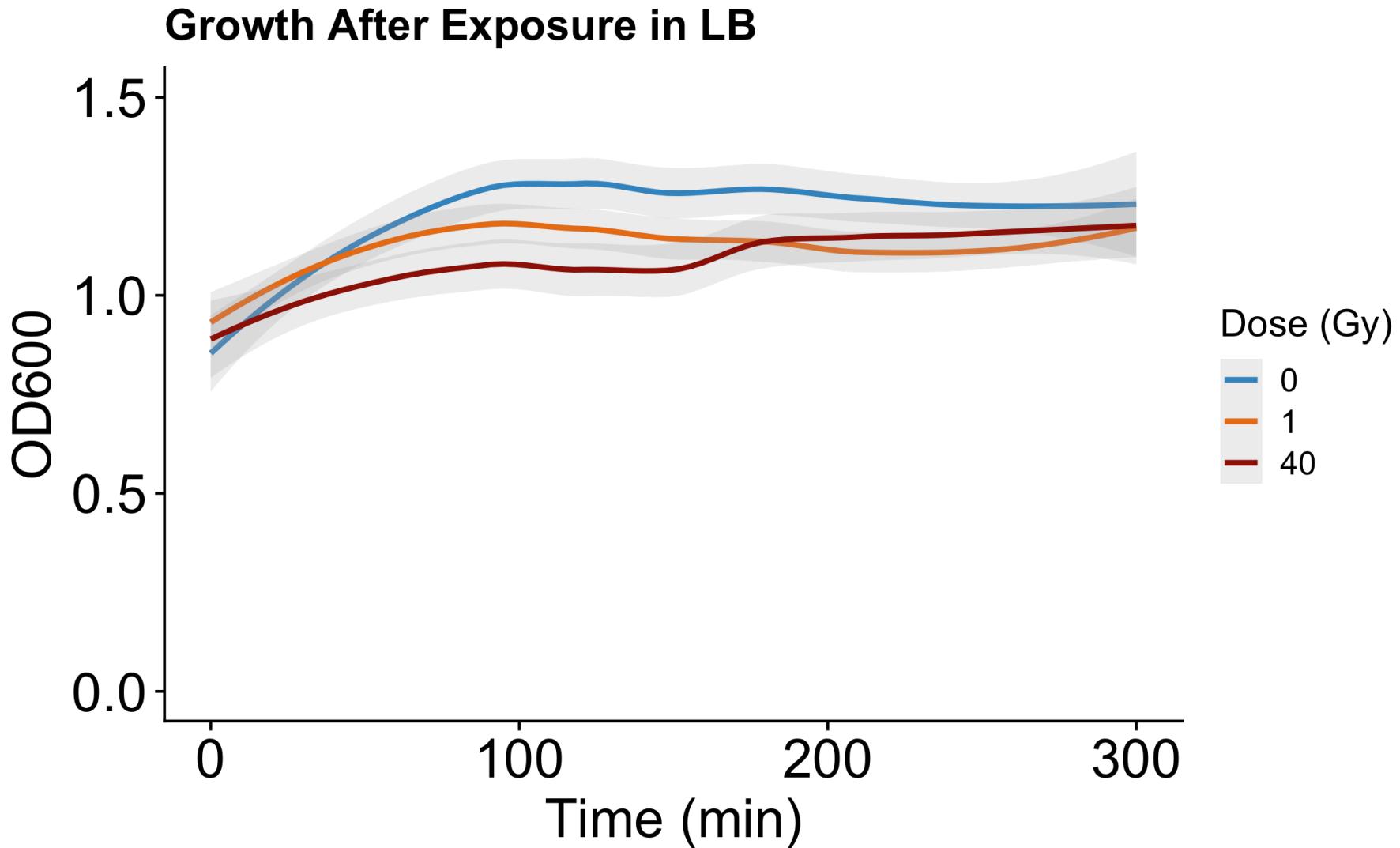


# Where there's a figure, there's a geom\_ for that!

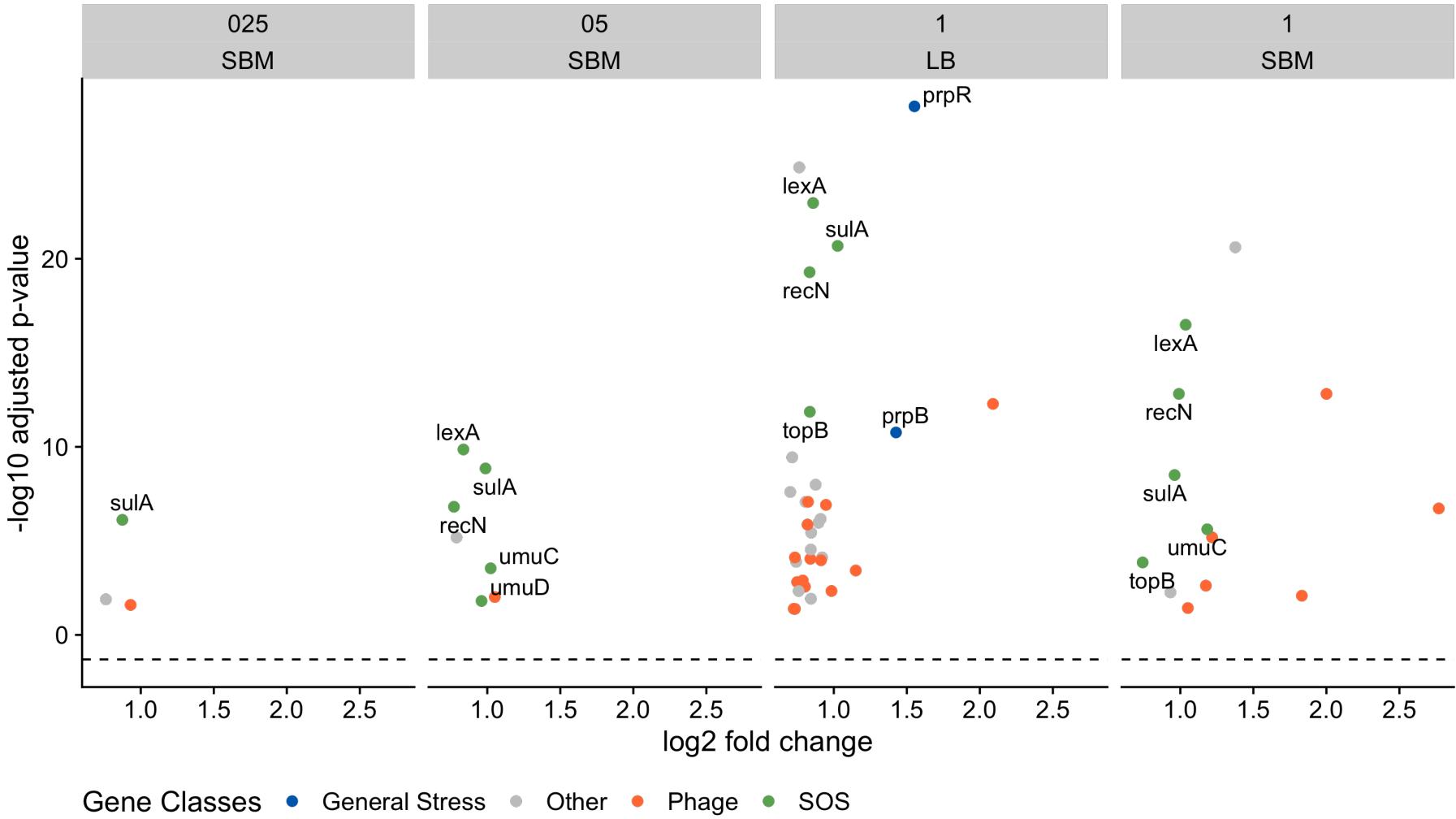
50 examples of what ggplot can do

<http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html>

# Personal Example: Geom\_smooth



# Personal Example: Geom\_point



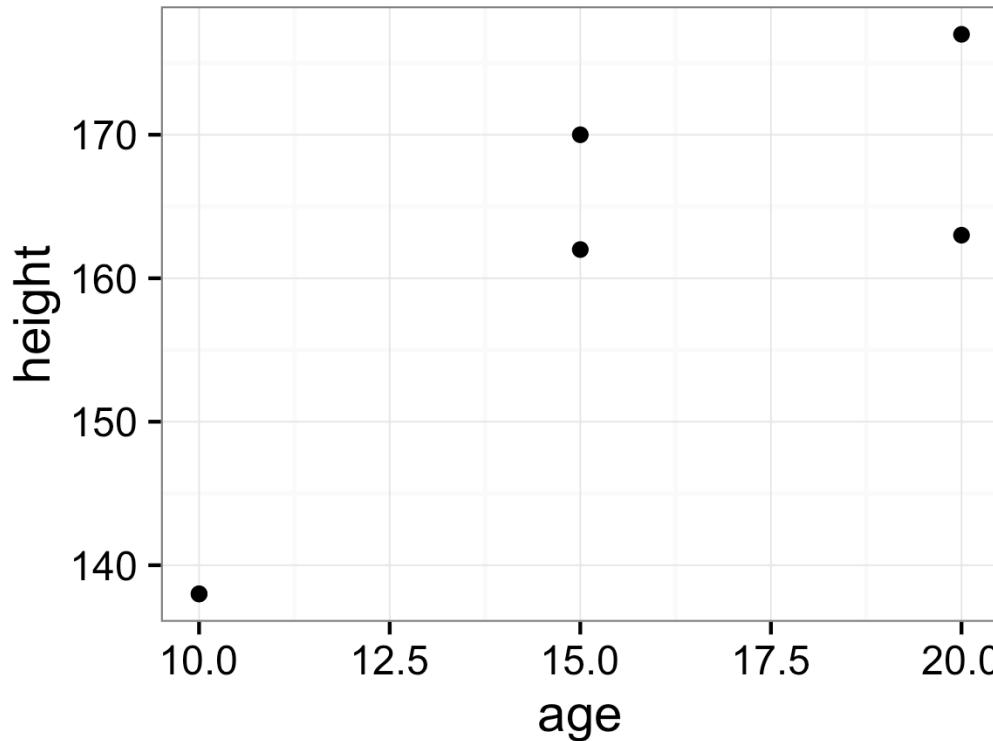
# Let's go over a simple example: mean height and weight of boys/girls ages 10-20

age (yrs)	height (cm)	weight (kg)	sex
10	138	32	M
15	170	56	M
20	177	71	M
10	138	33	F
15	162	52	F
20	163	53	F

Data from: <http://www.cdc.gov/growthcharts/>

Map age to x, height to y, visualize using points

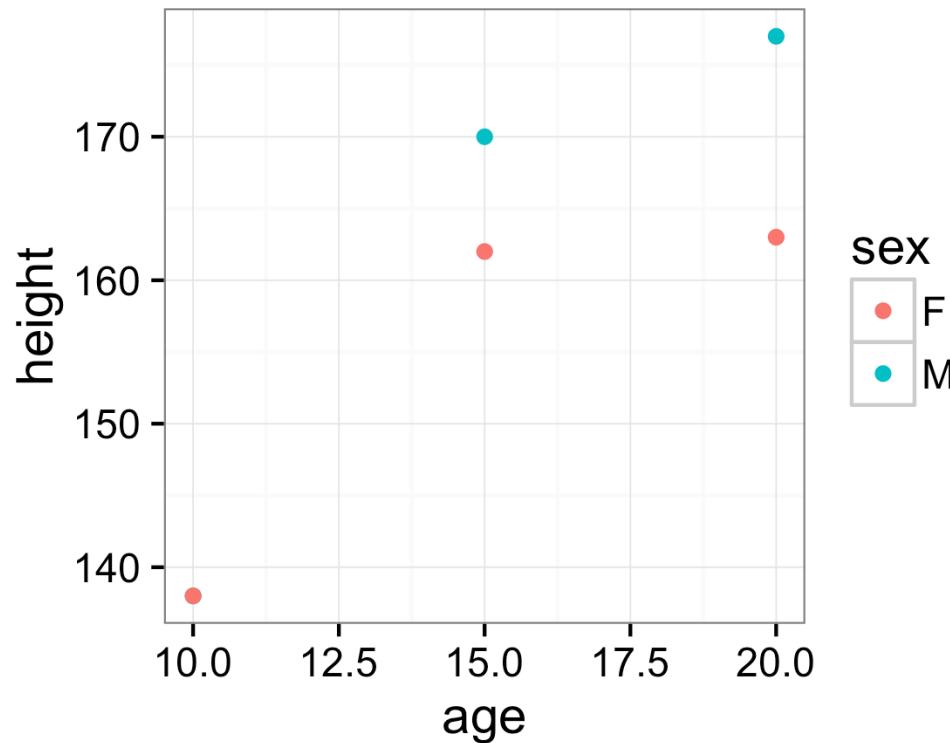
```
ggplot(data, aes(x=age, y=height)) +  
  geom_point()
```



# Let's color the points by sex

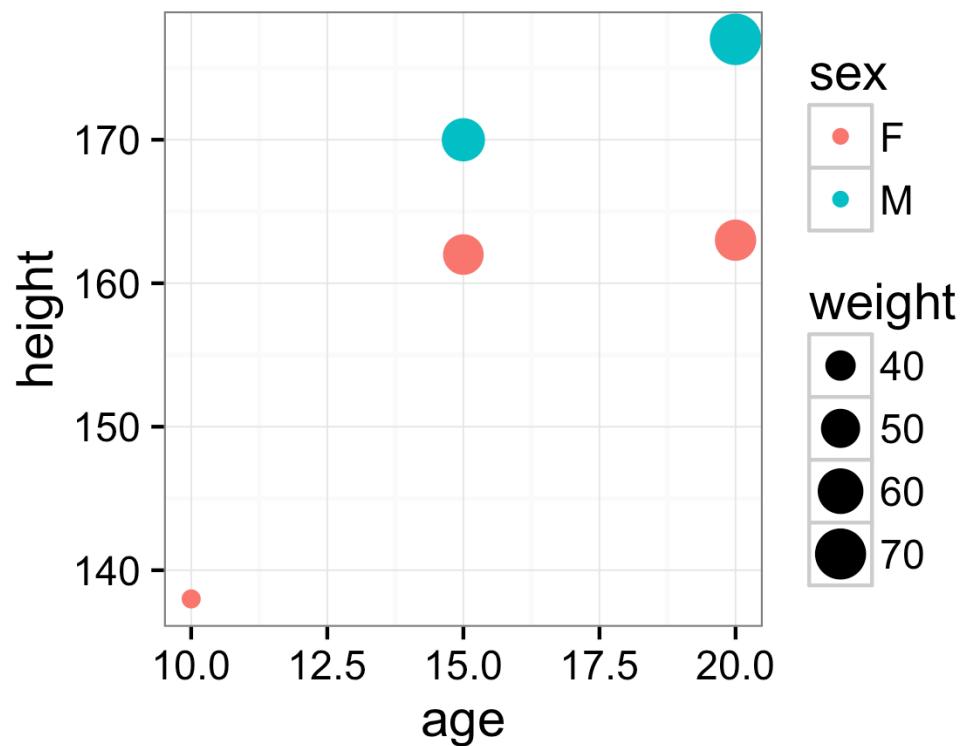
```
ggplot(data, aes(x=age, y=height,  
color=sex)) + geom_point()
```

★ NOTE: “color”  
aesthetic is for  
coloring points  
& lines;  
“fill” aesthetic is  
for coloring bars  
& distributions



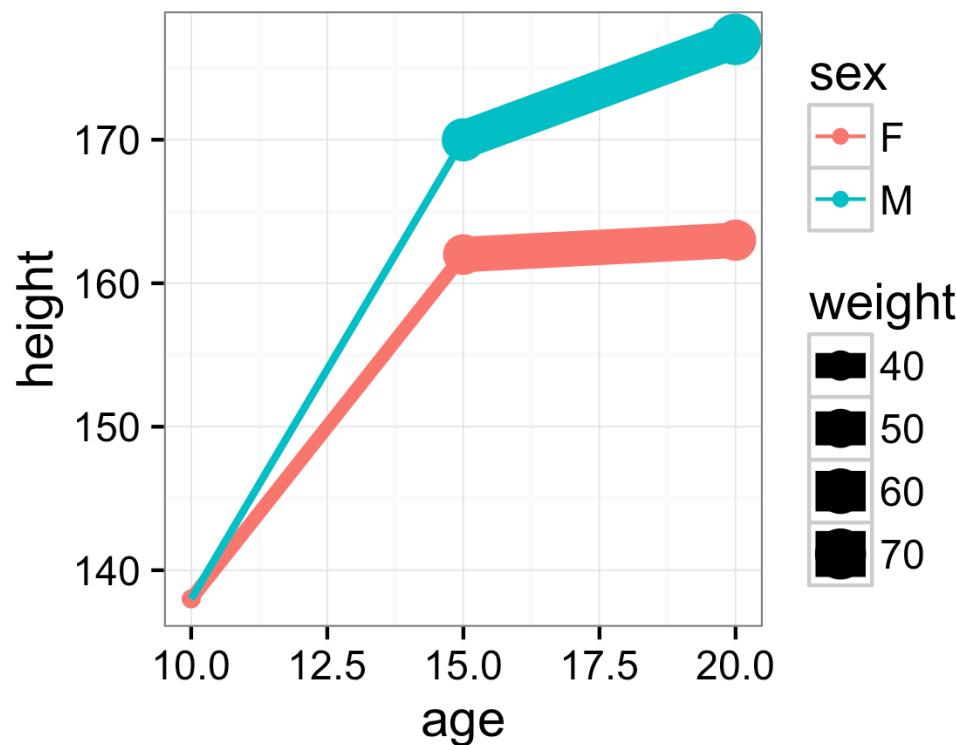
# And change point size by weight

```
ggplot(data, aes(x=age, y=height,  
color=sex, size=weight)) + geom_point()
```



# And connect the points with lines

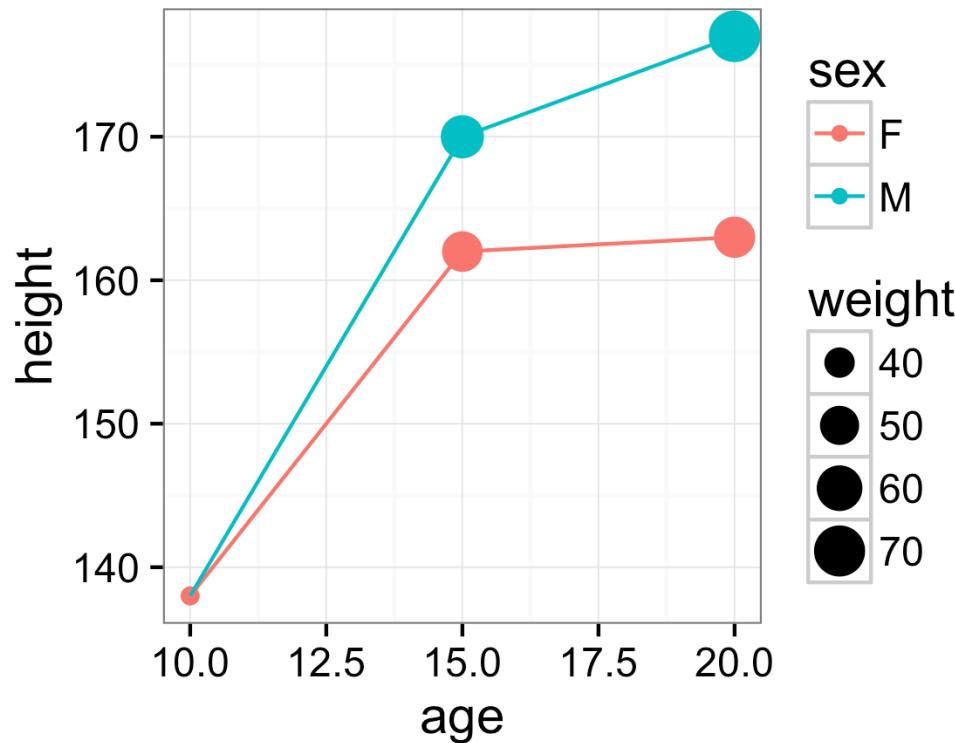
```
ggplot(data, aes(x=age, y=height,  
color=sex, size=weight)) +  
  geom_point() + geom_line()
```



Oops!

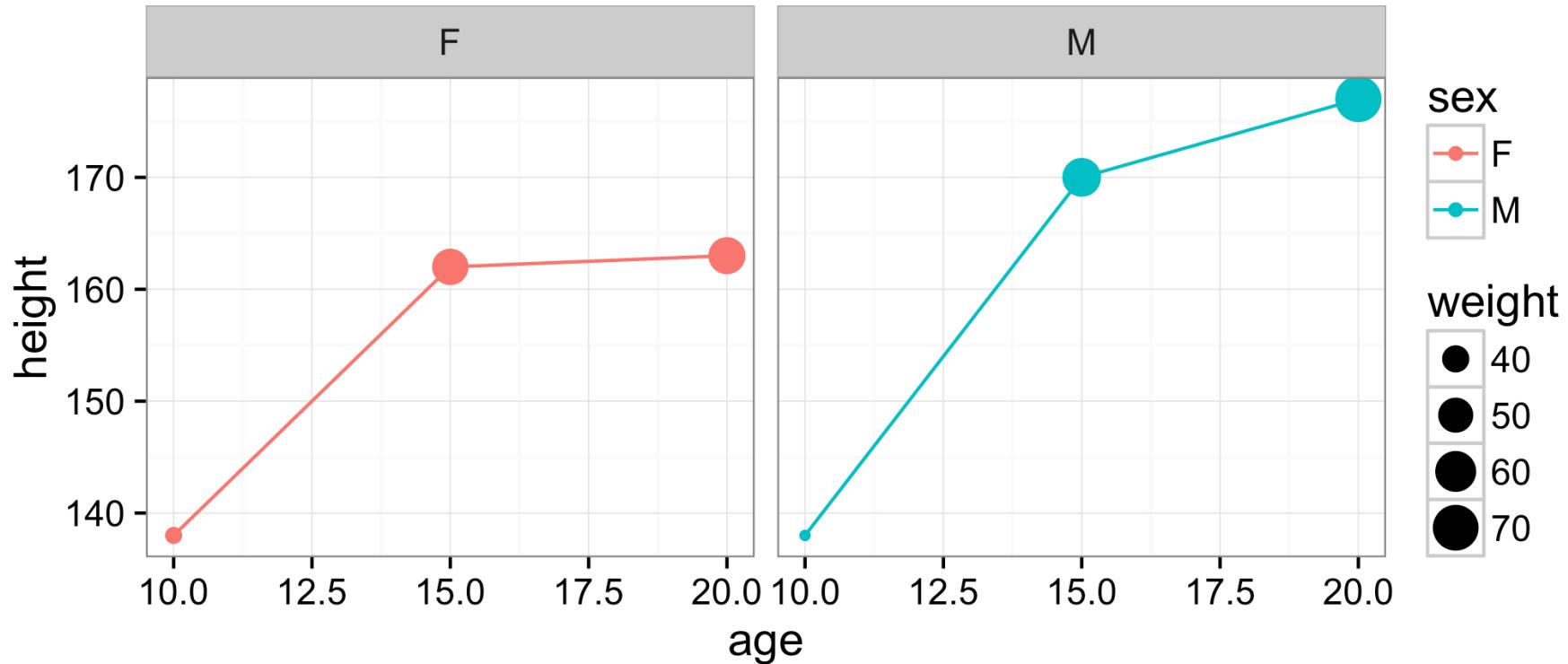
# The weight-to-size mapping should only be applied to points

```
ggplot(data, aes(x=age, y=height,  
color=sex)) + geom_point(aes(size=weight)) +  
geom_line()
```



# We can also make side-by-side plots (called facets)

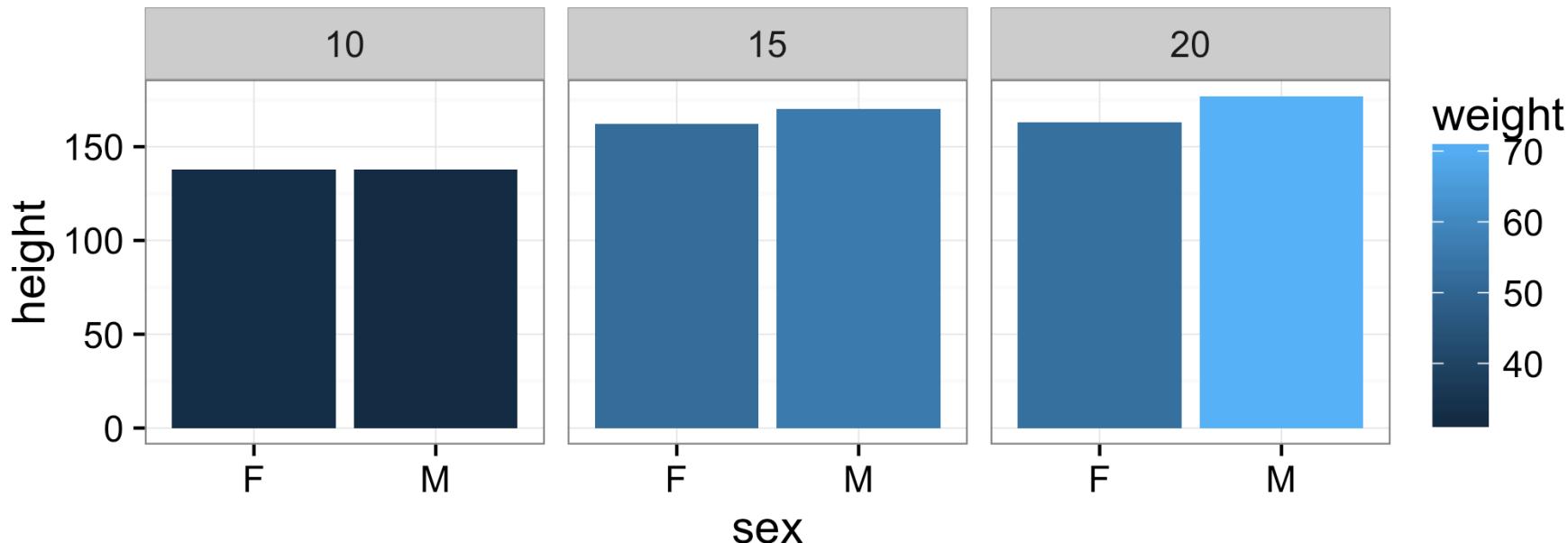
```
ggplot(data, aes(x=age, y=height,  
color=sex)) + geom_point(aes(size=weight)) +  
geom_line() + facet_wrap(~sex)
```



# Now let's facet by age, color by weight, and use bars (columns) to plot height

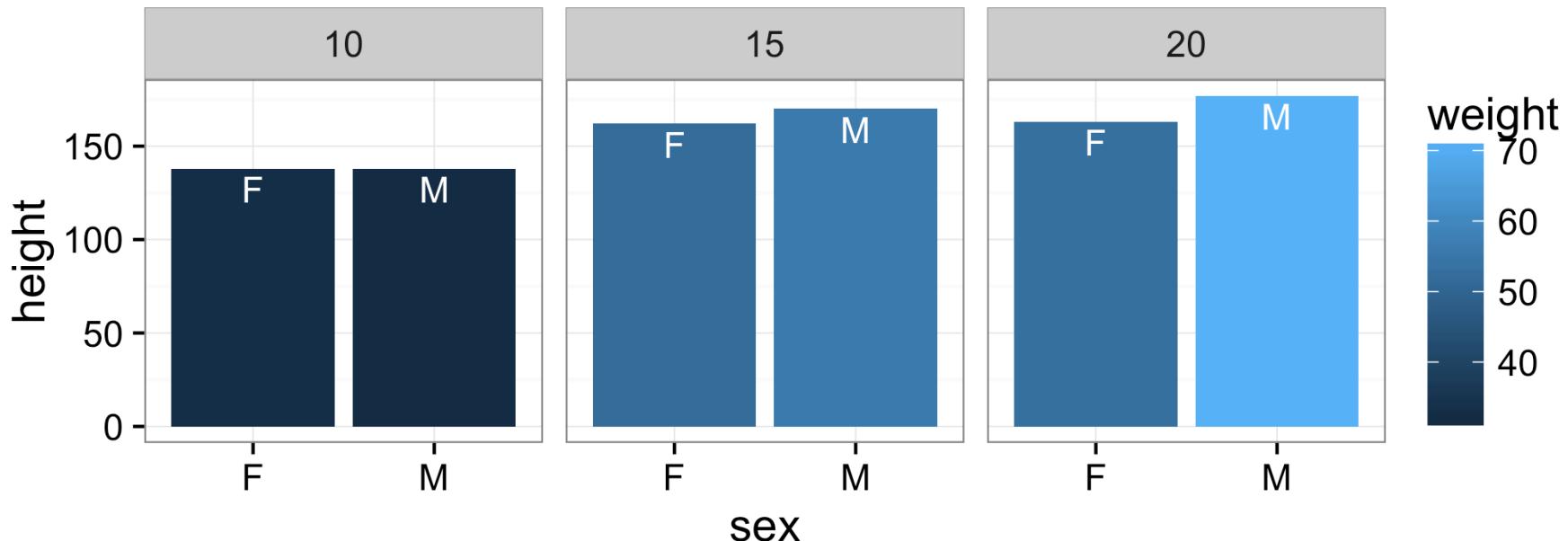
```
ggplot(data, aes(x=sex, y=height, fill=weight)) +  
  geom_col() + facet_wrap(~age)
```

NOTE: “fill” ★  
aesthetic is for  
coloring bars &  
distributions



# Let's plot the sex also at the top of the bar

```
ggplot(data, aes(x=sex, y=height, fill=weight)) +  
  geom_col() +  
  geom_text(aes(label=sex), vjust=1.3, color='white') +  
  facet_wrap(~age)
```



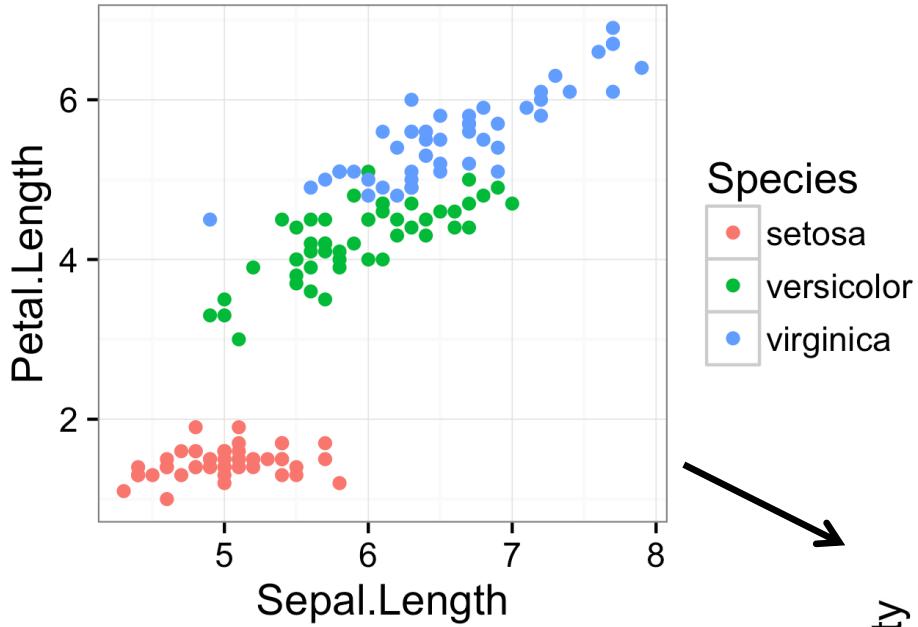
All the geoms with all their options are described on the ggplot2 web page

<https://ggplot2.tidyverse.org/reference/>

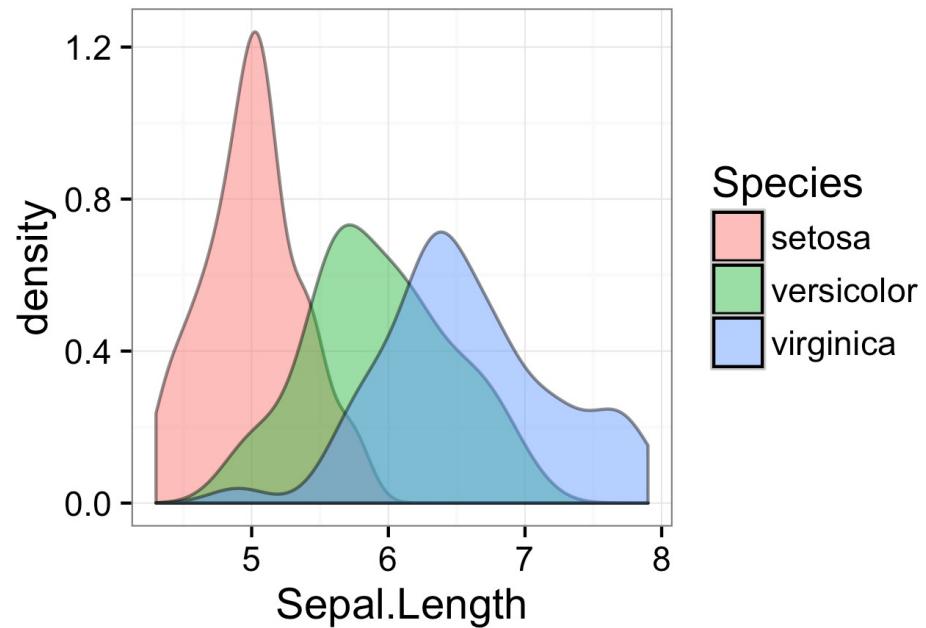
# Demonstration Time!

**Work on Section #1.1 and 1.2**

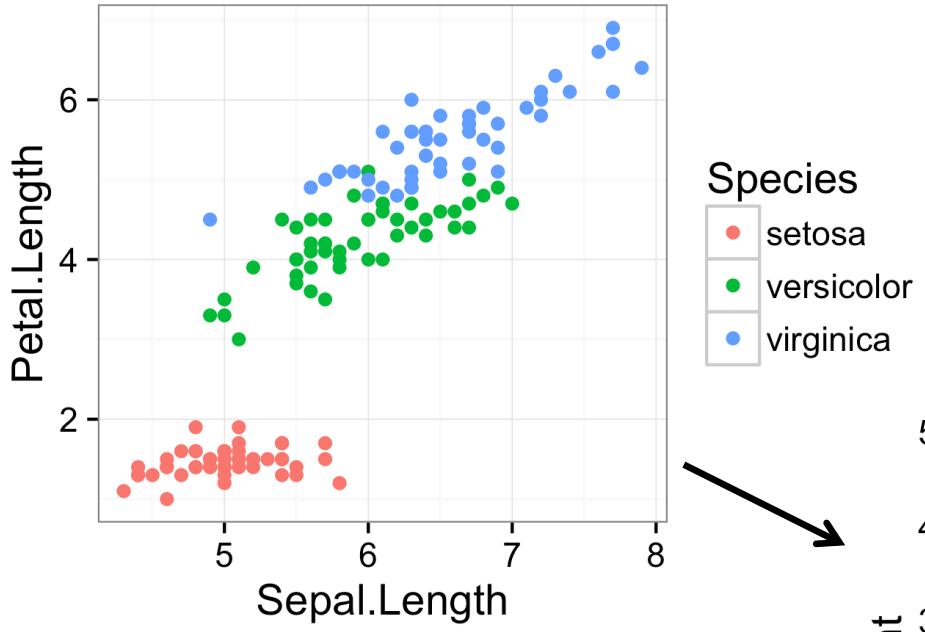
# We often need to do statistical transformations before plotting



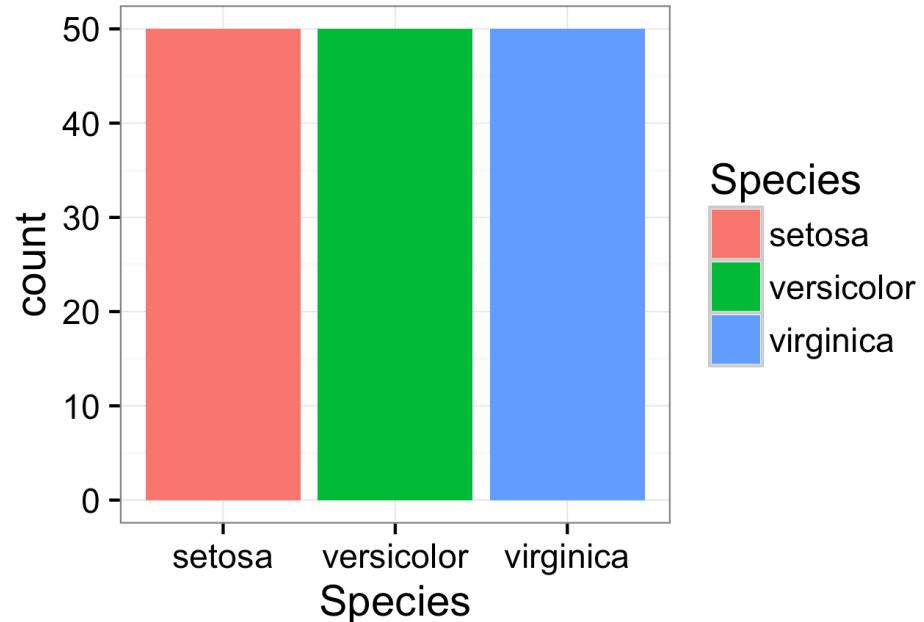
density of  
data points



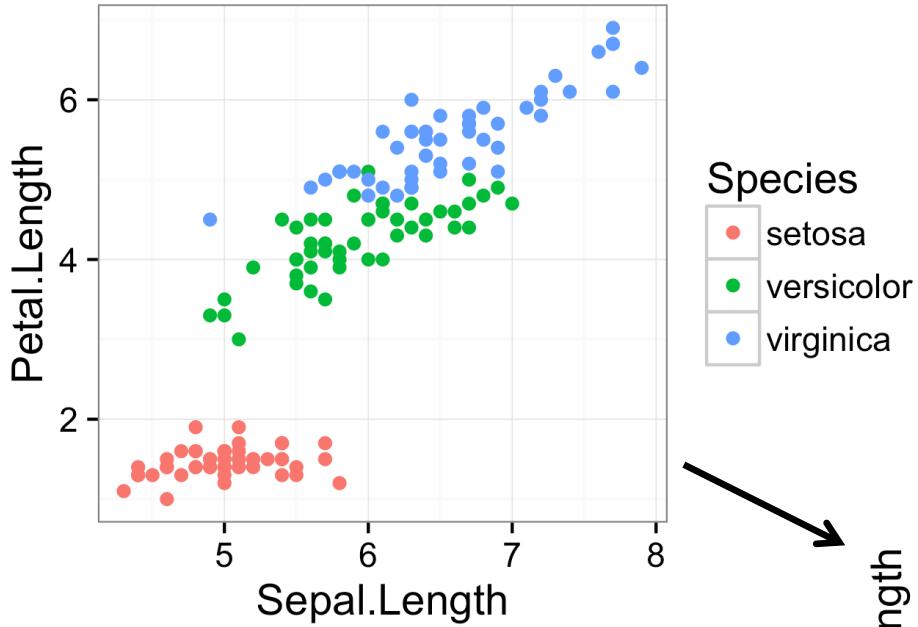
# We often need to do statistical transformations before plotting



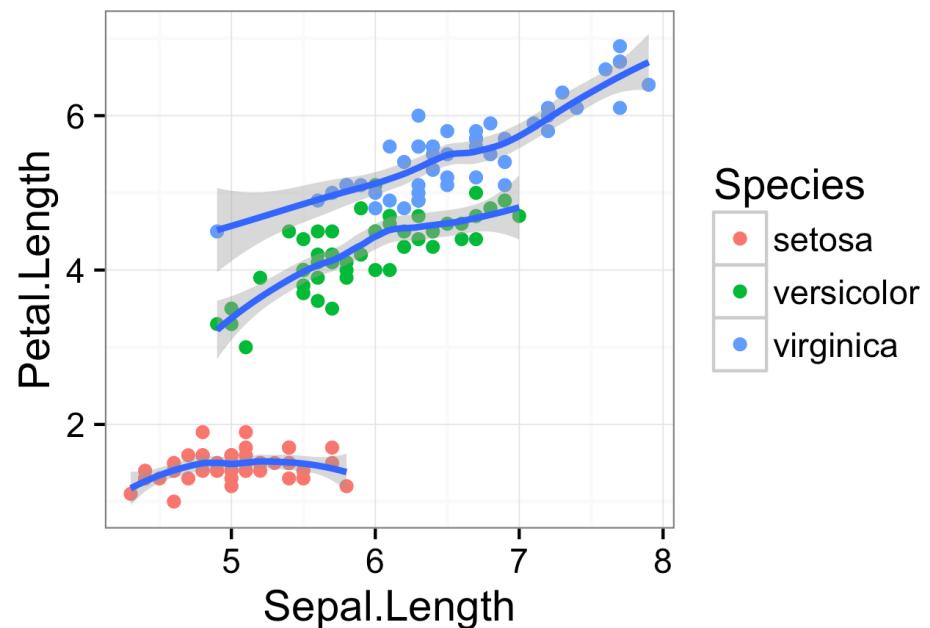
count of number  
of different types



# We often need to do statistical transformations before plotting



statistical smoothing/  
trend lines



# In ggplot2, these transformations are done with stats

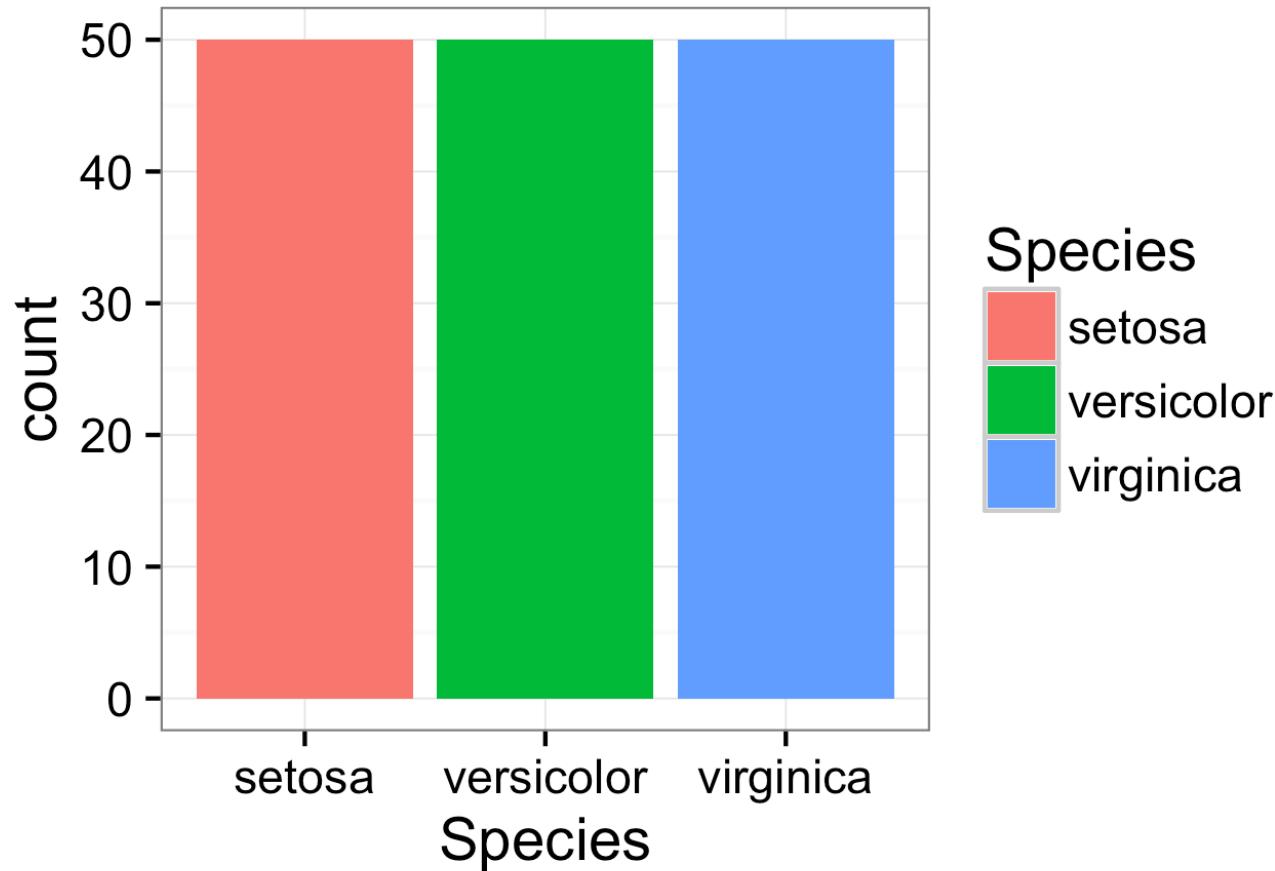
- **stat\_ecdf**  
Empirical Cumulative Density Function
- **stat\_ellipse**  
Plot data ellipses.
- **stat\_function**  
Superimpose a function.
- **stat\_identity**  
Identity statistic.
- **stat\_qq** (geom\_qq)  
Calculation for quantile-quantile plot.
- **stat\_summary\_2d** (stat\_summary2d, stat\_summary\_hex)  
Bin and summarise in 2d (rectangle & hexagons)
- **stat\_unique**  
Remove duplicates.



$$f(x) = x$$

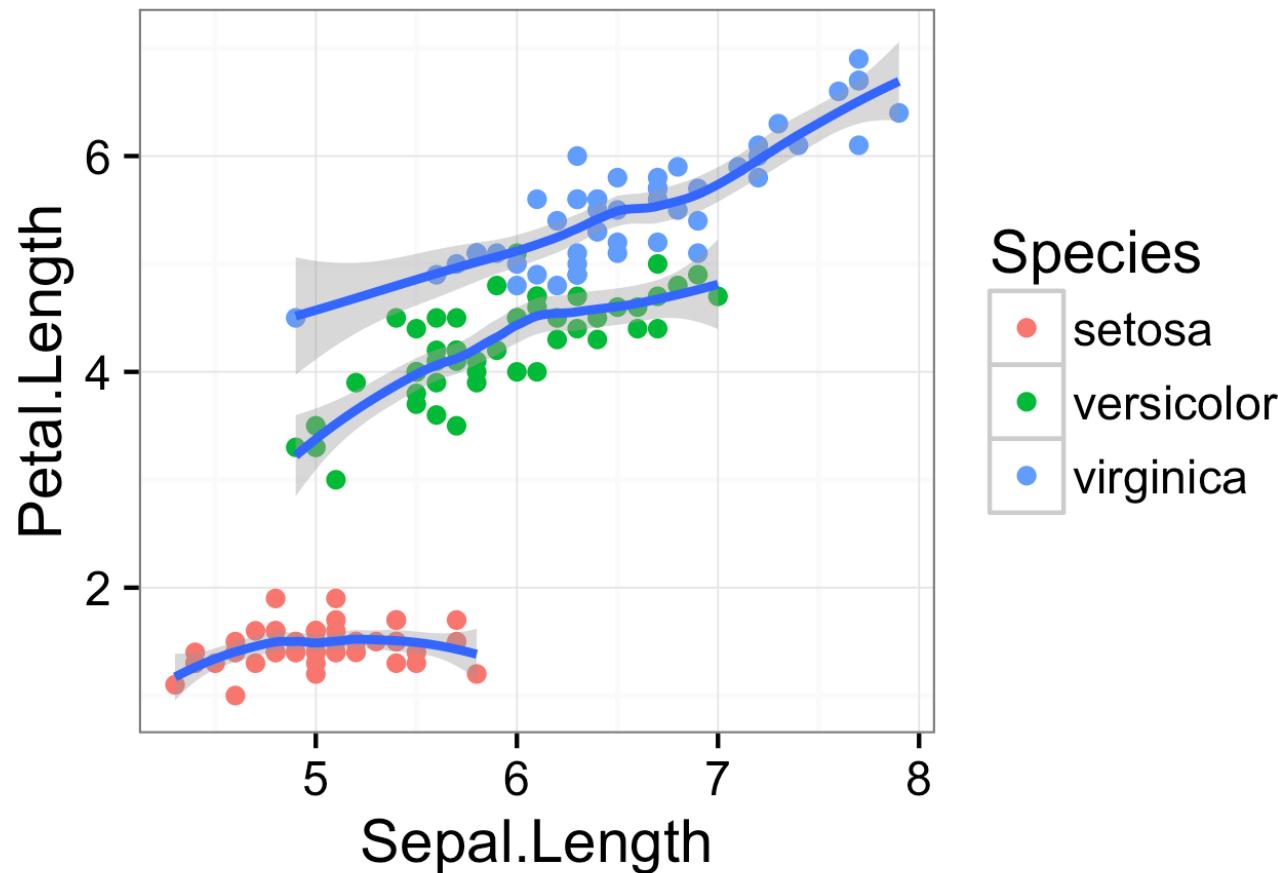
In most cases we just need to call the appropriate geom and it calls a stat

```
ggplot(iris, aes(x=Species, fill=Species)) +  
  geom_bar()
```



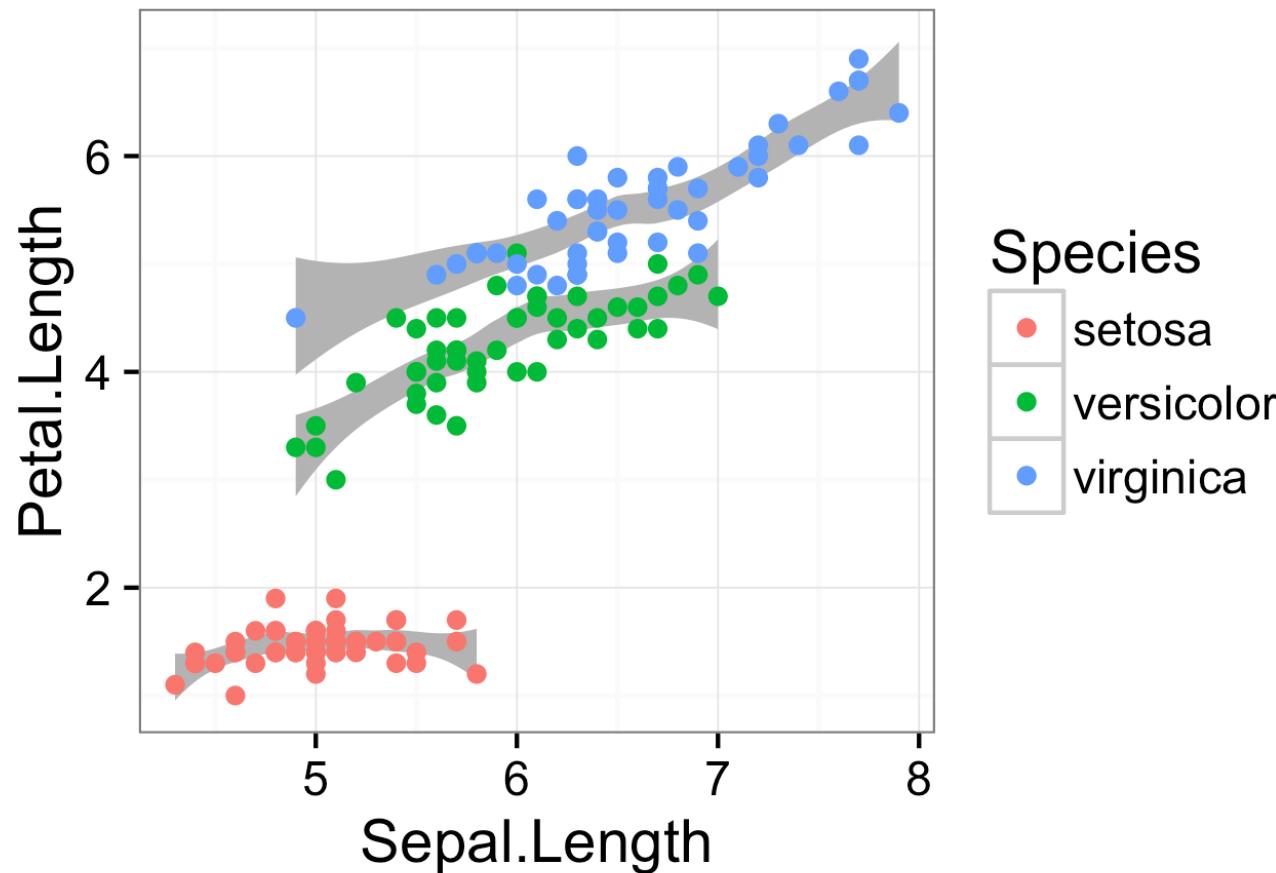
In most cases we just need to call the appropriate geom and it calls a stat

```
ggplot(iris, aes(x=Sepal.Length, y=Petal.Length)) +  
  geom_point(aes(color=Species)) +  
  geom_smooth(aes(group=Species))
```



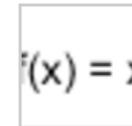
# However, sometimes it can be helpful to call the stat directly

```
ggplot(iris, aes(x=Sepal.Length, y=Petal.Length)) +  
  stat_smooth(aes(group=Species), geom="ribbon", fill='gray70') +  
  geom_point(aes(color=Species))
```

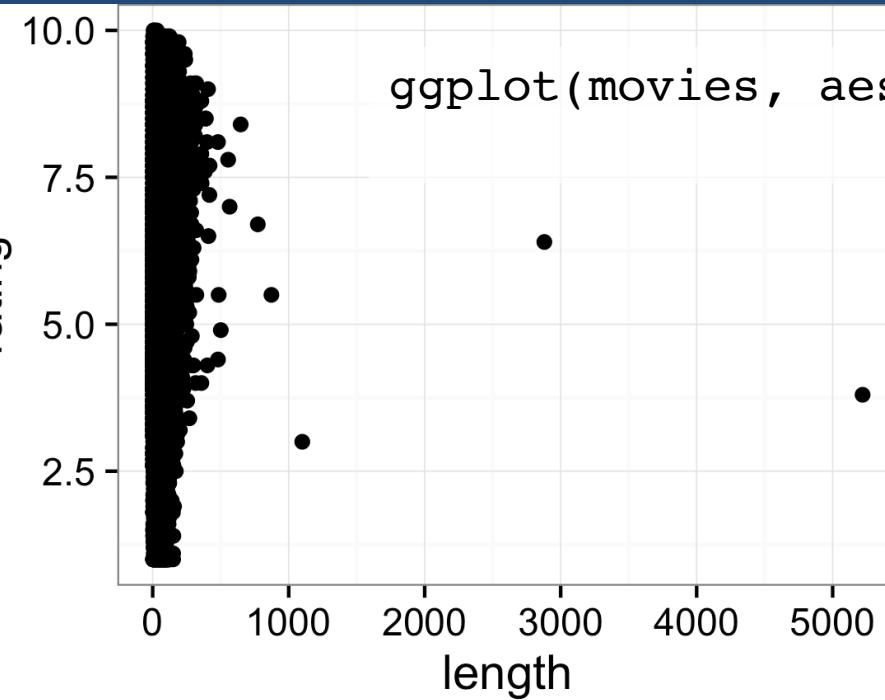


# Scales define how to map data onto aesthetics

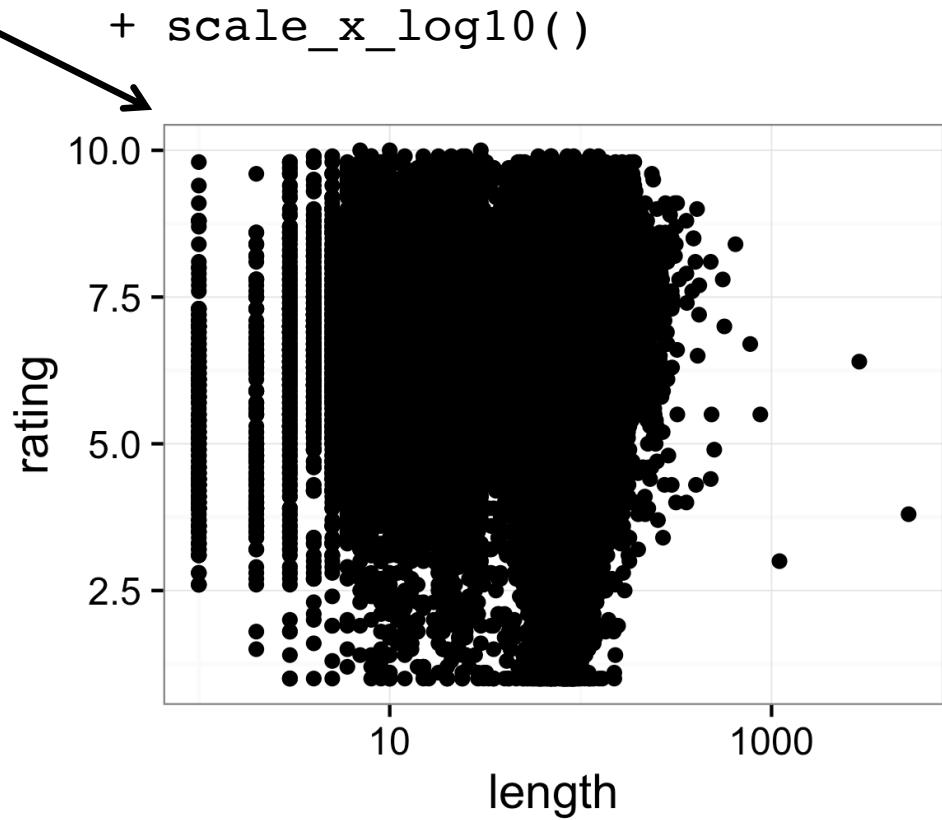
- **scale\_colour\_grey** (scale\_color\_grey, scale\_fill\_grey)  
Sequential grey colour scale.
- **scale\_colour\_hue** (scale\_color\_discrete, scale\_color\_hue, scale\_colour\_discrete, scale\_fill\_discrete, scale\_fill\_hue)  
Qualitative colour scale with evenly spaced hues.
- **scale\_identity** (scale\_alpha\_identity, scale\_color\_identity, scale\_colour\_identity, scale\_fill\_identity, scale\_linetype\_identity, scale\_shape\_identity, scale\_size\_identity)  
Use values without scaling.
- **scale\_manual** (scale\_alpha\_manual, scale\_color\_manual, scale\_colour\_manual, scale\_fill\_manual, scale\_linetype\_manual, scale\_shape\_manual, scale\_size\_manual)  
Create your own discrete scale.
- **scale\_linetype** (scale\_linetype\_continuous, scale\_linetype\_discrete)  
Scale for line patterns.
- **scale\_shape** (scale\_shape\_continuous, scale\_shape\_discrete)  
Scale for shapes, aka glyphs.
- **scale\_size** (scale\_radius, scale\_size\_area, scale\_size\_continuous, scale\_size\_discrete)



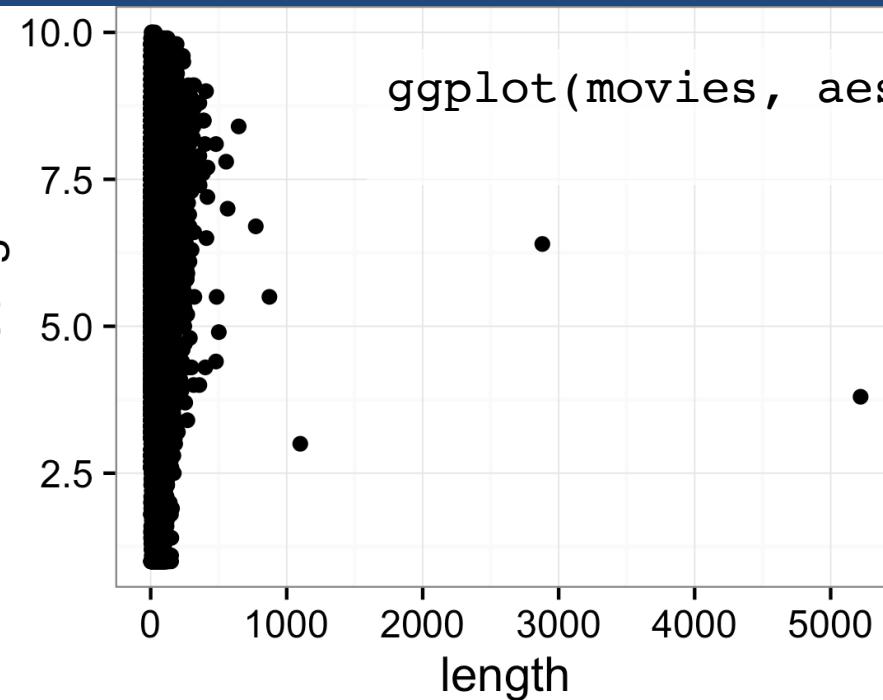
# Example 1: Change scaling of x axis



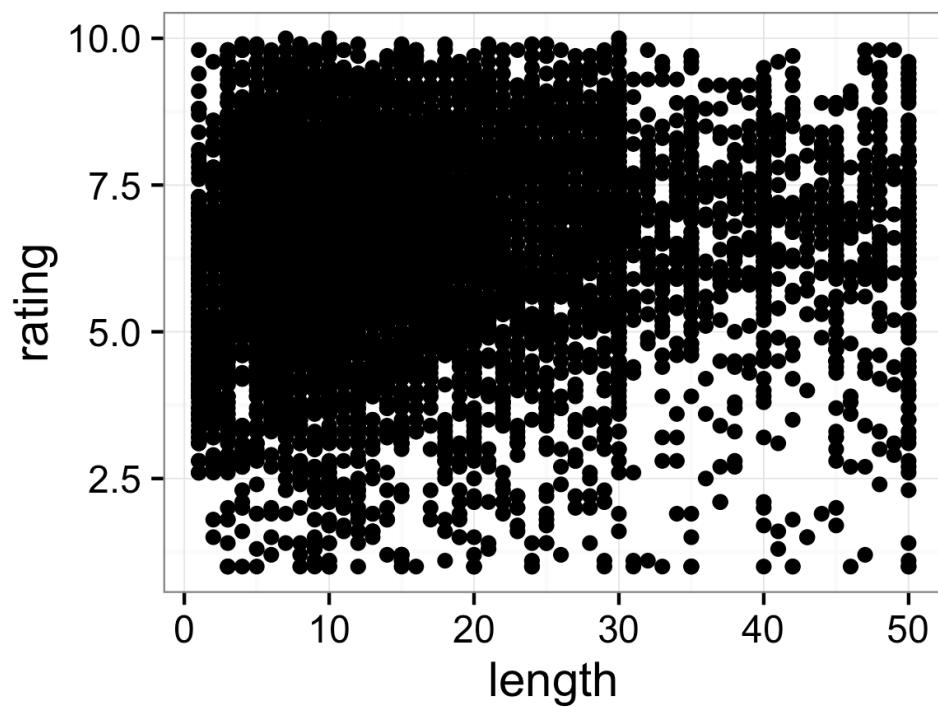
```
+ scale_x_log10()
```



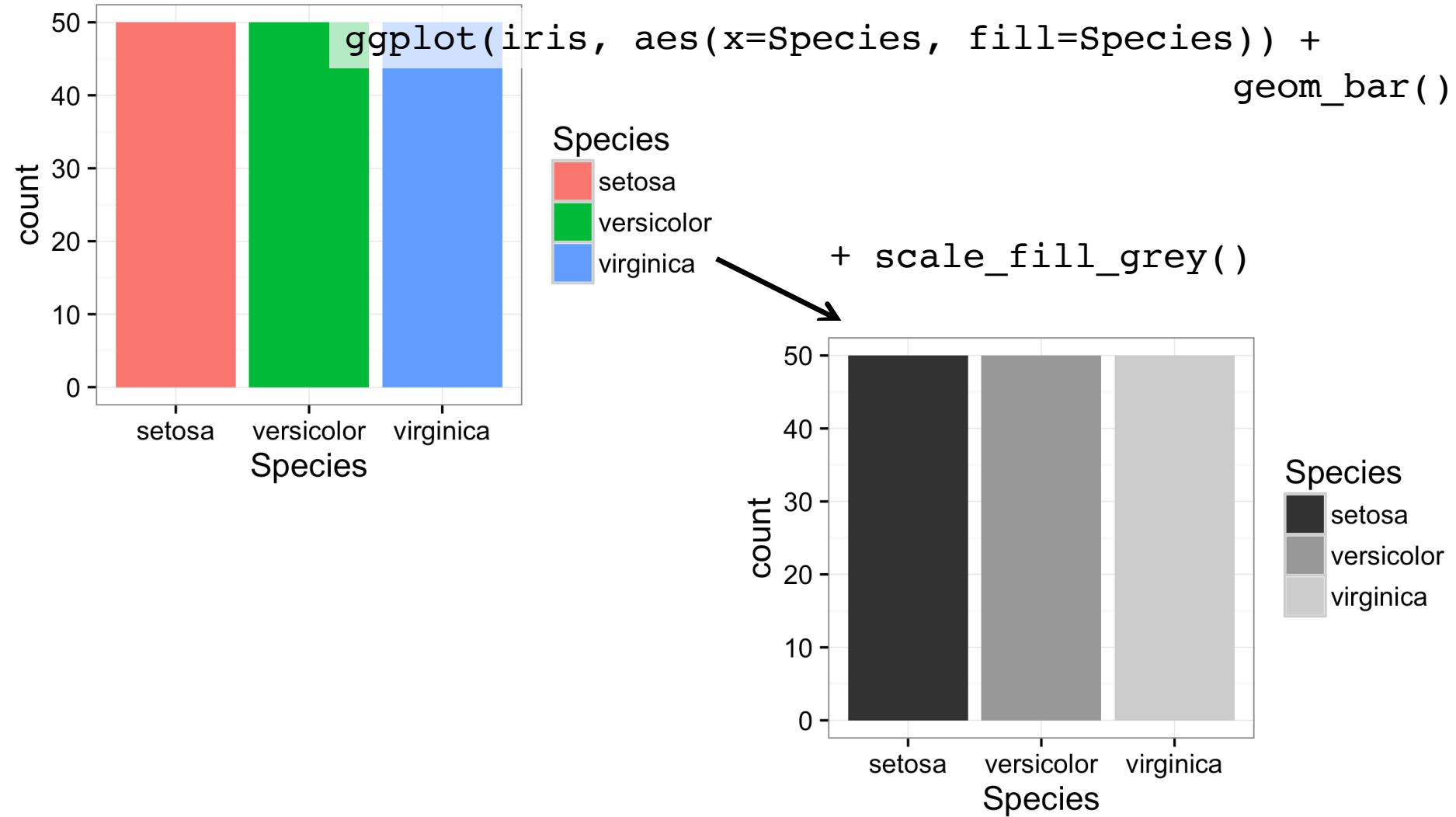
# Example 1: Change scaling of x axis



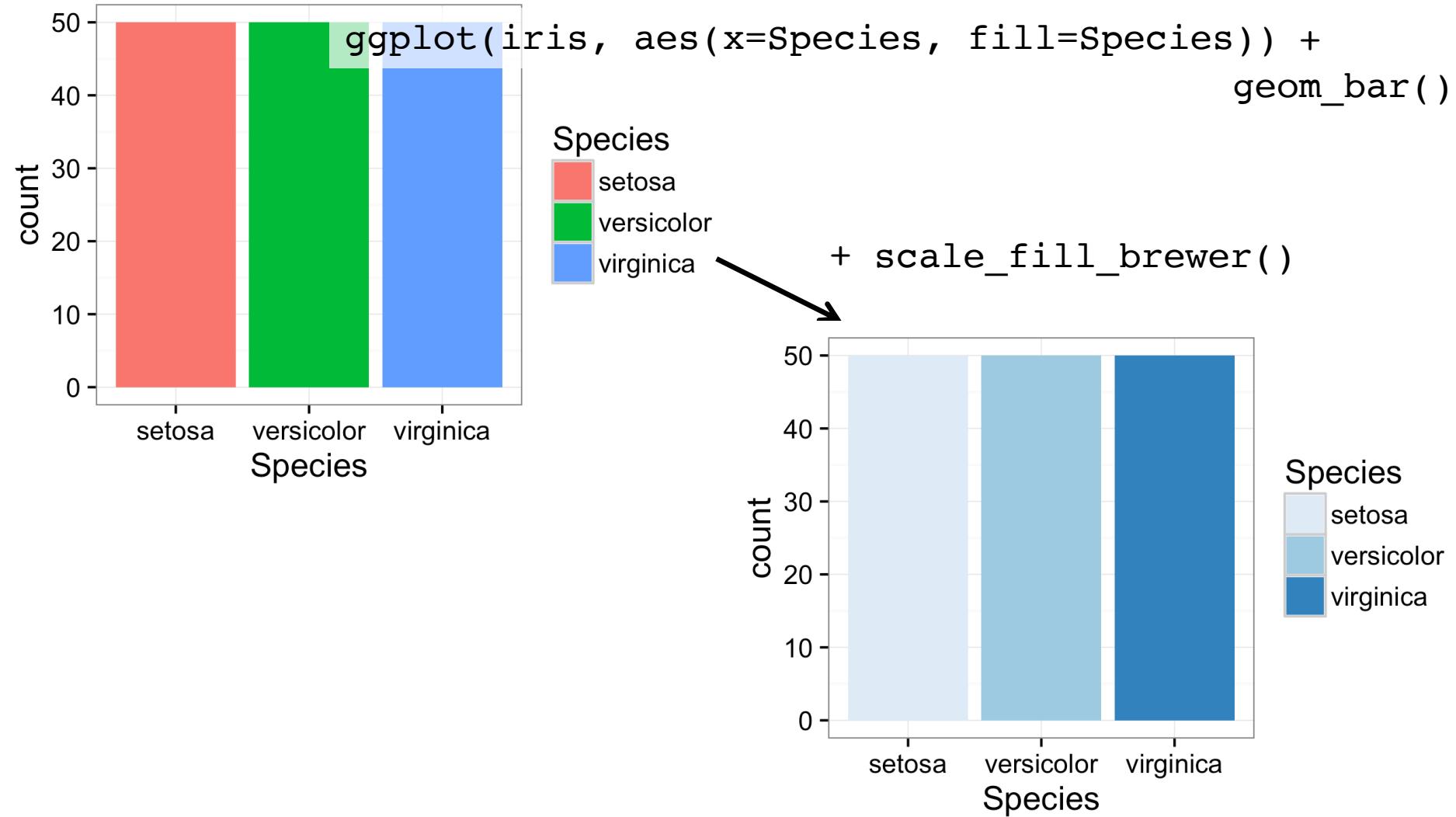
```
+ xlim(1, 50)
```



# Example 2: Change color scaling

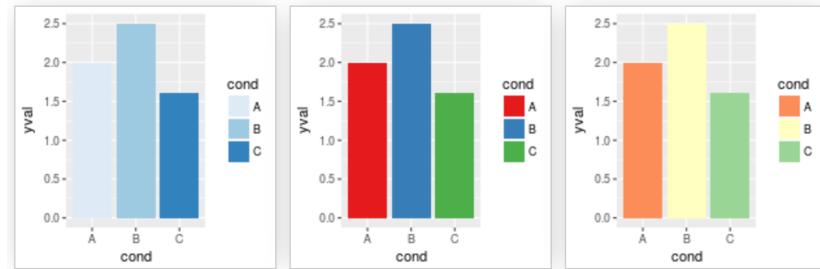
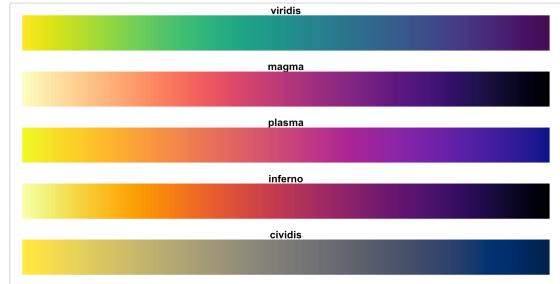


# Example 2: Change color scaling



# Some color scaling options in ggplot2

- `scale_color_gradient()`,  
`scale_fill_gradient()`
- `scale_color_discrete()`,  
`scale_fill_discrete()`
- `scale_color_brewer()`,  
`scale_fill_brewer()`
- `scale_color_distiller()`,  
`scale_fill_distiller()`
- `scale_color_colorblind()`,  
`scale_fill_colorblind()`
- `scale_color_manual()`,  
`scale_fill_manual()`



```
palette_pretty <- c("#0072B2", "#E69F00", "#009E24", "#FF0000", "#979797", "#5530AA")
palette_bgy <- c("#FFFFCC", "#A1DAB4", "#41B6C4", "#2C7FB8", "#253494")
palette_wine <- c("#bcb37b", "#e934d", "#8f8023", "#790000", "#5b00b0b")
palette_cb <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442",
  "#0072B2", "#D55E00", "#CC79A7", "#999999")
```

# Themes control non-data display

The `labs()` function lets you change the title, x- and y-axis labels, and color/legend labels:

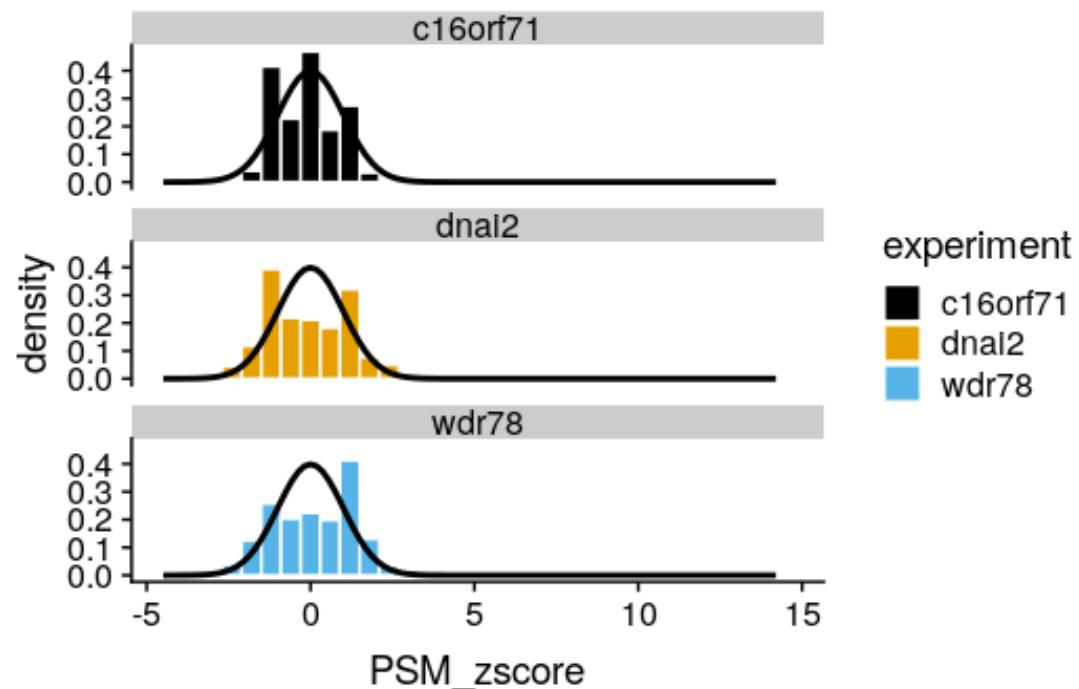
<https://ggplot2.tidyverse.org/reference/labs.html>

Adding `theme()` layers allow you to customize fonts, sizes, and positions of titles, labels, background, gridlines and legends:

<https://ggplot2.tidyverse.org/reference/theme.html>

# Putting it all together, example 1

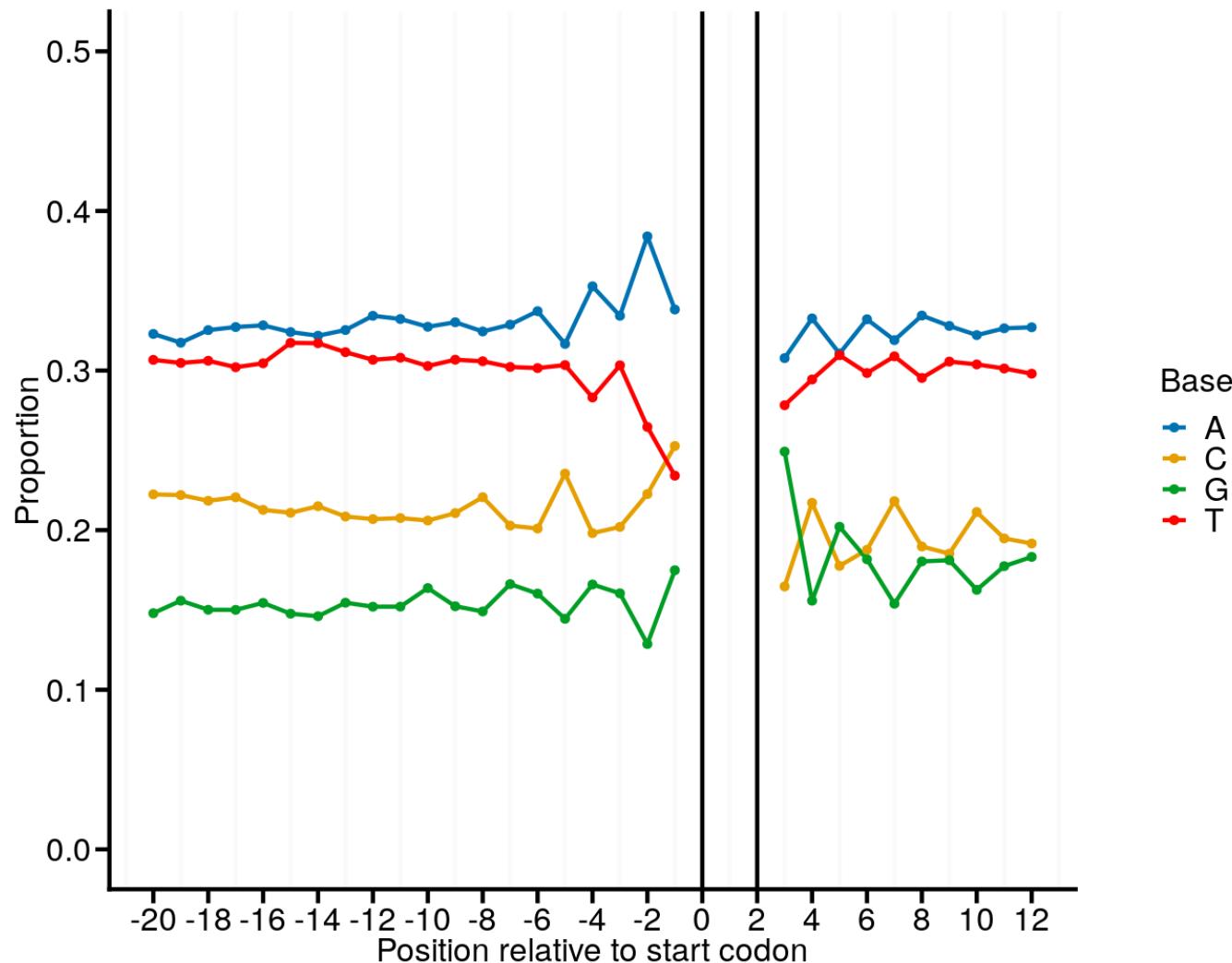
```
ggplot(apms_df, aes(x = PSM_zscore, fill = experiment)) +  
  geom_histogram(binwidth = 0.6, color = "white") +  
  facet_wrap(~experiment, ncol = 1) +  
  stat_function(aes(group = experiment), fun = dnorm, n = 101,  
                args = list(mean = 0, sd = 1), size = 1) +  
  scale_fill_colorblind()
```



# Putting it all together, example 2

```
seq_plot <- seq_df %>%
  ggplot(aes(x = pos, y = prop, group = Base, color = Base)) +
  geom_line() +
  geom_point(size = 0.5) +
  scale_x_continuous(breaks = seq(-20, 13, 2),
                     labels = seq(-20, 13, 2)) +
  scale_color_manual(values = palette_pretty) +
  geom_vline(xintercept = 0) +
  geom_vline(xintercept = 2) +
  background_grid(major = "only_minor", minor = "x") +
  ylim(0,0.5) +
  ylab("Proportion") +
  xlab("Position relative to start codon")
```

# Putting it all together, example 2



# Saving plots

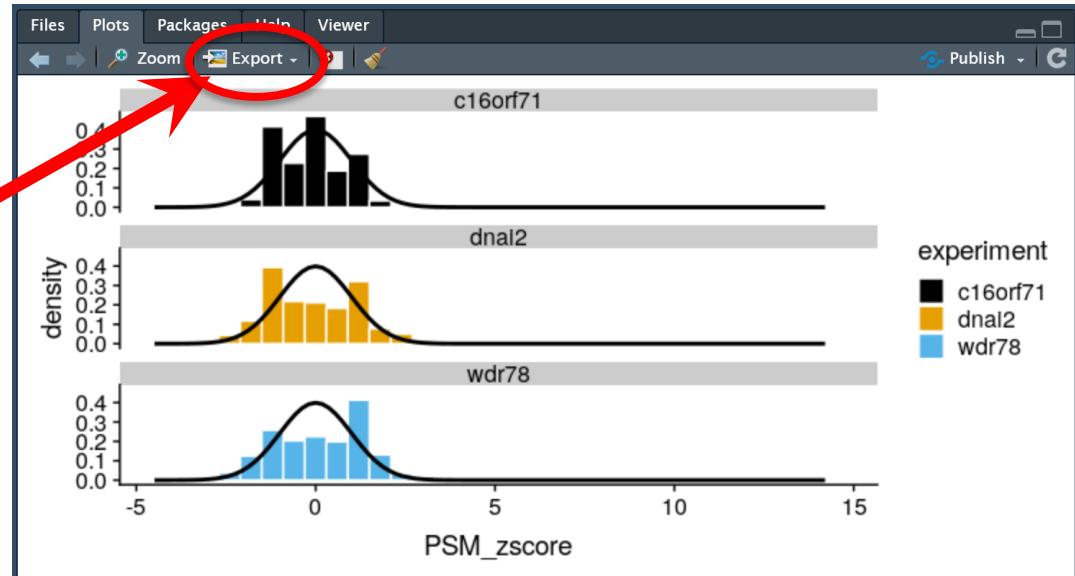
With code:

```
seq_plot %>% ggsave("relative_start_plot.png", ., device =  
"png", width = 4.5, height = 3.5, units = "in")
```

```
seq_plot %>% ggsave("relative_start_plot.pdf", ., device =  
"pdf", width = 4.5, height = 3.5, units = "in")
```

In RStudio; to make plots appear in the bottom-right window:

- Tools > Global Options  
> R Markdown >  
Uncheck “Show output inline”



# Demonstration Time!

Checkout the “**ggplot cheatsheet**” in the Day 2 folder for code examples

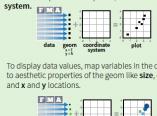
## Data Visualization with ggplot2 Cheat Sheet



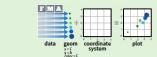
Cheat Sheet

### Basics

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



To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

**aesthetic mappings** + **data** + **geom**  
Creates a **ggplot** object with your data, geom, and mappings. Supplies many defaults.

**ggplot(data = mpg, aes(x = cyl, y = hwy))**  
Creates a plot that you finesse by adding layers to. No defaults, but provides more control than qplot().

**data**  
**ggplot(mpg, aes(hwy, cyl)) +**  
**geom\_point(aes(color = cyl)) +**  
**coord\_cartesian() +**  
**theme\_bw() +**  
**scale\_color\_gradient()**

**add** a new layer to a plot with a geom, “**0**” or **stat**.” “**0**” function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

**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.

additional arguments

additional arguments