

Are you a student? Take the library survey!

<https://library.duke.edu>

The screenshot shows the top navigation bar of the Duke University Libraries website. On the left is the "DUKE UNIVERSITY LIBRARIES" logo. To the right are links for "My Accounts", "Ask a Librarian", and a search bar containing "Search articles, books" with a magnifying glass icon. Below the search bar are links for "Search & Find", "Using the Library", "Research Support", "Course Support", "Libraries", and "About".

A green callout box contains the text: "Have ideas for improving Duke University Libraries? Take a short survey for a chance to win a \$150 Amazon gift card!" A red arrow points from the top-left towards this box. In the top right corner of the box is a small "x" icon.

The main search interface features a large search bar with the placeholder "Find articles, books, journals & more" and a "search" button. Below the search bar are tabs for "All", "Books & Media", and "Articles", with "Books & Media" currently selected. Other navigation links include "Advanced Search", "eBooks", "Online Journal Titles", "Research Databases", "Digitized Collections", and "More".

A box displays "Perkins & Bostock Hours" for "Thu 1/30" with "Open 24hrs" and "Public: 8am - 7pm". A link "All Libraries & Hours »" is also present.

News, Events & Exhibits



A thumbnail image for an "Upcoming Library Events" section, showing a group of people in what appears to be a library setting.

Upcoming Library Events

Intro to QGIS
Thu, Jan 30, 10:00am

Quick Links

- Ask a Librarian
- Research Guides

Visualization in R using ggplot2

Angela Zoss

1/30/2020

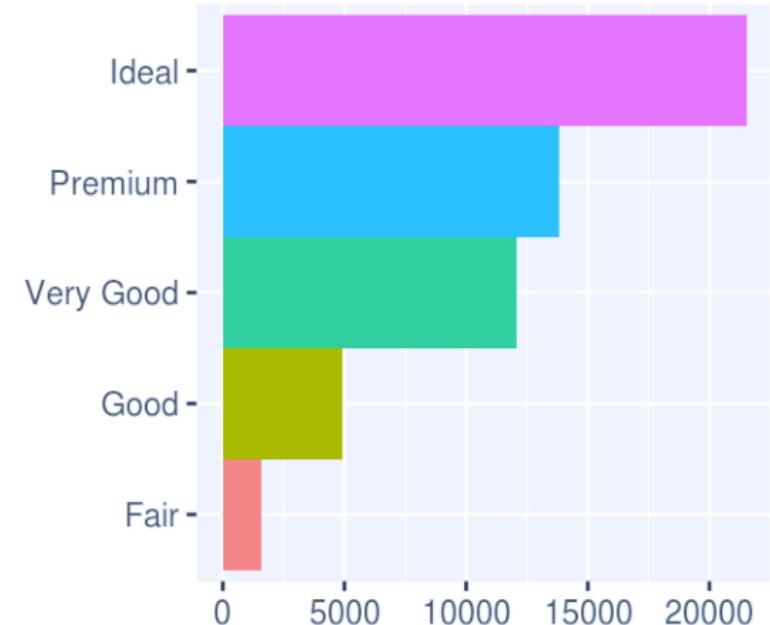
<https://github.com/amzoss/ggplot2-S20>

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

ggplot2

What is ggplot2?

an R package designed to create plots based on a theory of the grammar of graphics.



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/ggplot2-S20>

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

In RStudio:

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

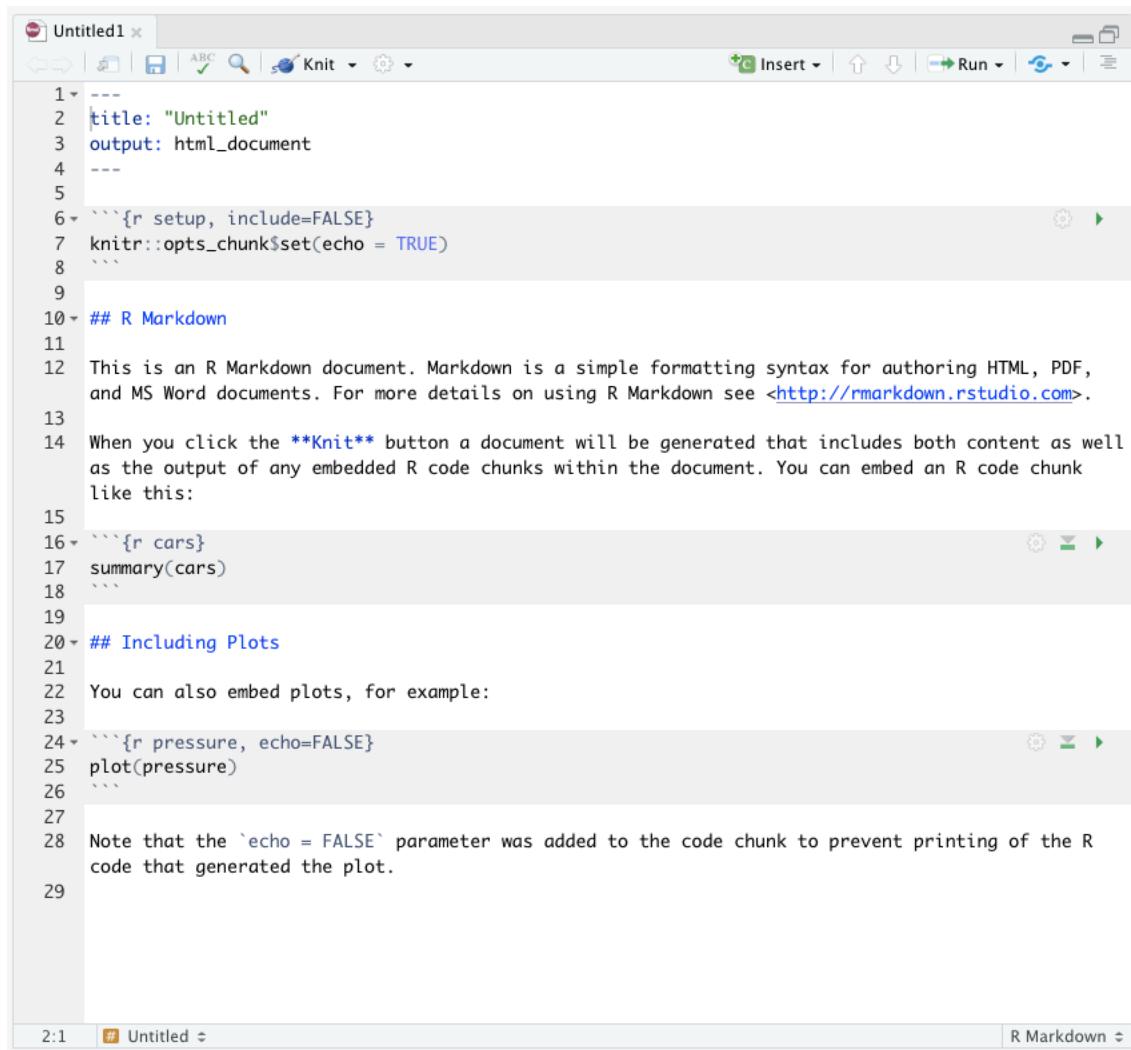
Why R Markdown?

- Plots show up inline
- Easier to incorporate explanatory text and materials
- Like to be able to easily run one chunk at a time

Caution: Running things out of order can mean your code won't work again later. Clear your environment often and run code chunks in order to be safe.

R Markdown files

- First few lines are called YAML header, set up some properties of file
- R code goes inside code chunks
- Text in Markdown syntax goes in between code chunks
- Use the “play” button to run individual code chunks
- Knit or run all to run the entire document



```
Untitled1 x
ABC Knit ▾
Insert ▾ Run ▾

1 ---  
2 title: "Untitled"  
3 output: html_document  
4 ---  
5  
6 ```{r setup, include=FALSE}  
7 knitr::opts_chunk$set(echo = TRUE)  
8 ````  
9  
10 ## R Markdown  
11  
12 This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF,  
and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.  
13  
14 When you click the **Knit** button a document will be generated that includes both content as well  
as the output of any embedded R code chunks within the document. You can embed an R code chunk  
like this:  
15  
16 ```{r cars}  
17 summary(cars)  
18 ````  
19  
20 ## Including Plots  
21  
22 You can also embed plots, for example:  
23  
24 ```{r pressure, echo=FALSE}  
25 plot(pressure)  
26 ````  
27  
28 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R  
code that generated the plot.  
29
```

R Markdown test

- File → New File → R Markdown
- Click OK to accept defaults
- Type inside the first few lines to edit the YAML header (edit title, add author, etc.)
- Add a new R code chunk at the end of the file using Insert → R
- Type some R code inside the code chunk:
library(tidyverse)
- Run the new code chunk



```
29
30  ````{r}
31
32  library(tidyverse)
33
34  ...
35
```

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.

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

Complete the template below to build a graph.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(<mapping> = aes(<MAPPINGS>),  
  stat = <STAT>, position = <POSITION>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTIONS> +  
  <SCALE_FUNCTIONS> +  
  <THEME_FUNCTIONS>
```

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

```
aesthetic mappings    data    geom  
geom_point(x=cyl, 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'x5' 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, unemployed))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, nudge_y = -1, check_overlap = TRUE), x, y, yend, alpha, angle, color, curvature, linetype, size, linemetre = 1)

a + geom_path(linend = "butt", linejoin = "round", x, y, alpha, color, group, linetype, size)

a + geom_rect(aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1), x, y, alpha, color, group, linetype, size)

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

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

a + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

a + geom_text(aes(label = cyl), nudge_x = 1, nudge_y = -1, check_overlap = TRUE), x, y, label, alpha, angle, color, fontfamily, fontface, hjust, linheight, size, vjust

TWO VARIABLES

continuous x, continuous y

c + geom_label(aes(label = cyl), nudge_x = 1, nudge_y = -1, check_overlap = TRUE), x, y, label, alpha, angle, color, fontfamily, fontface, hjust, linheight, size, vjust

c + geom_liner(hight = 2, width = 2), x, y, alpha, color, fill, shape, size, stroke

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

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

c + geom_rug(sides = "bl"), x, y, alpha, color, fill, group, linetype, size, weight

c + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

c + geom_text(aes(label = cyl), nudge_x = 1, nudge_y = -1, check_overlap = TRUE), x, y, label, alpha, angle, color, fontfamily, fontface, hjust, linheight, size, vjust

continuous bivariate distribution

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, unemployed))

i + geom_line(), x, y, alpha, color, fill, linetype, size

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

visualizing error

j <- data.frame(grp = c("A", "B"), fit = 4.5, se = 1.2)

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

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

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

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

maps

data <- data.frame(murder = USArrests\$Murder, state = tolowerrownames(USArrests))
state <- data[, c(1, 2, 3, 4)]
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map)
+ expand_limits(x = mapLong, y = mapLat), 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_raster(aes(fill = z), interpolate = FALSE), x, y, alpha, fill

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

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

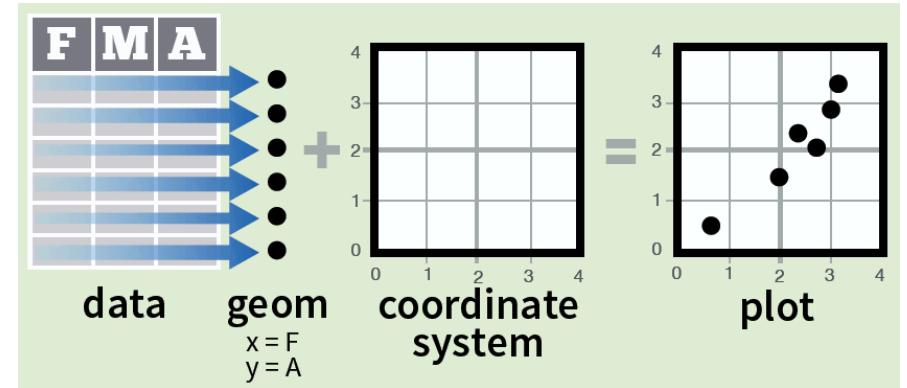


<https://www.rstudio.com/resources/cheatsheets/#ggplot2>

ggplot2: making a basic plot

Basic elements in any ggplot2 visualization

- **data**
- **aesthetics**
(variable mappings)
- **geom**
(chart type or shape)
- coordinate system
(the arrangement of the marks;
most geoms use default, cartesian)



<http://bit.ly/ggplot2-cheatsheet>

Template for a simple plot

Main
function

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

Shape
layer

```
geom_... ( non-variable adjustments )
```

Step-by-step

1. Set the data

**Main
function**

```
ggplot( data = data frame )
```

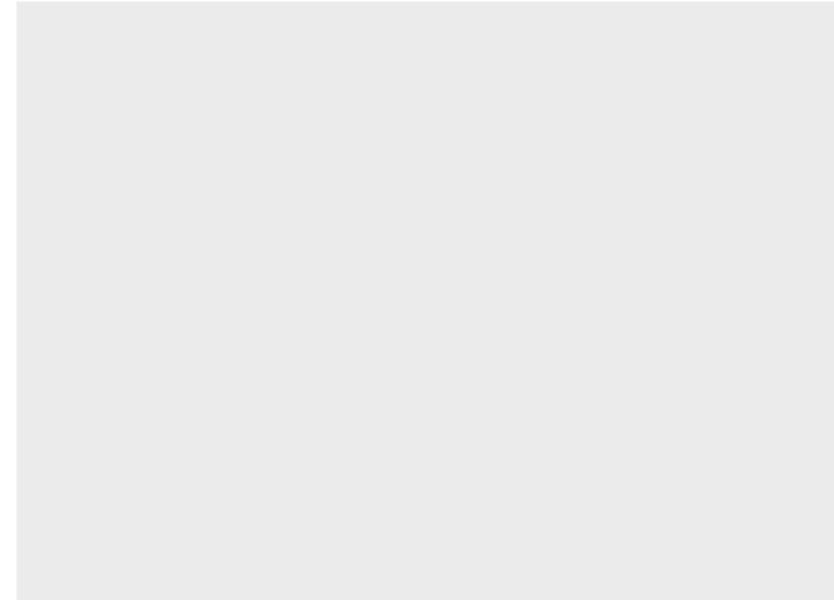
**Shape
layer**

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)
```



Step-by-step

1. Set the data
2. Map variables to aesthetics

Main function

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

Shape layer

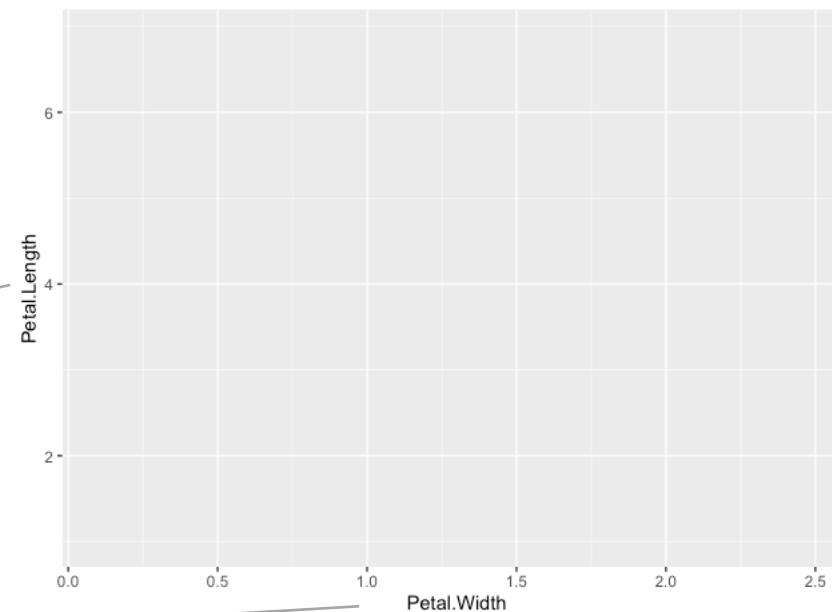
2. Map variables to aesthetics

“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))
```



Step-by-step

1. Set the data
2. Map variables to aesthetics
3. Choose a shape layer

Main
function

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

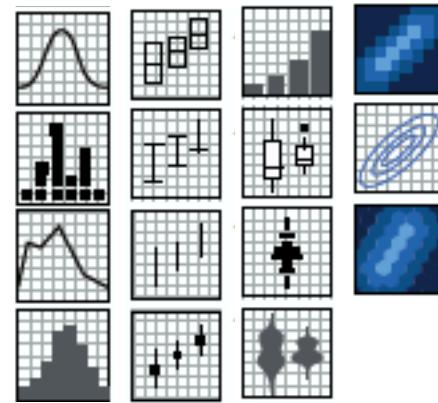
Shape
layer

```
geom_...()
```

+

Types of geoms

- geom_bar()
- geom_point()
- geom_histogram()
- geom_map()
- etc.



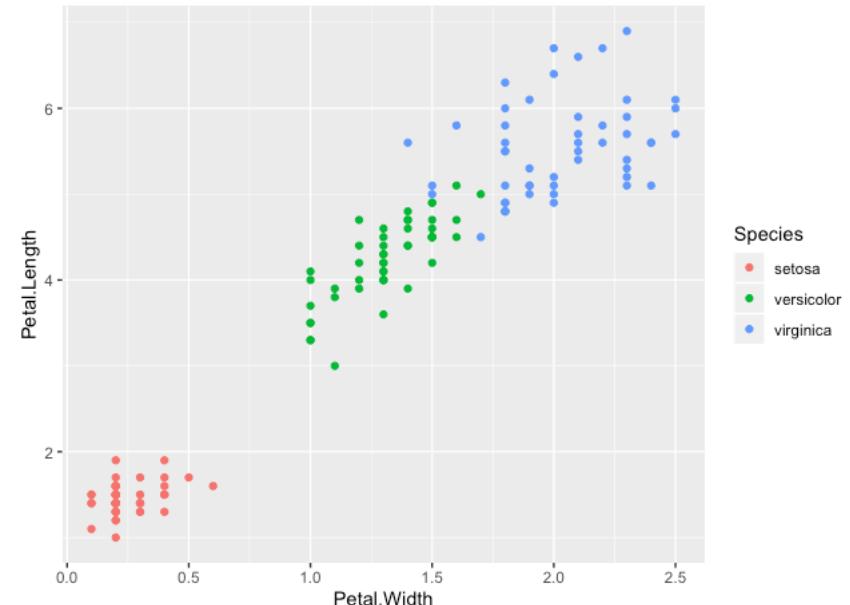
<http://bit.ly/ggplot2-cheatsheet>

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()
```



Step-by-step

1. Set the data
2. Map variables to aesthetics
3. Choose a shape layer
4. Add non-variable adjustments

Main function

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

Shape layer

```
geom_... ( non-variable adjustments )
```

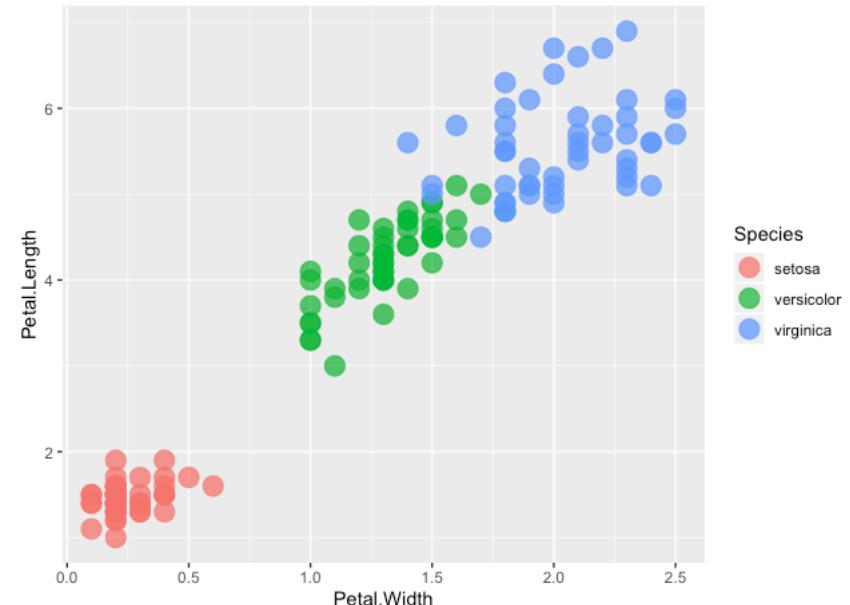
+

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)
```



Dataset 1: Game of Thrones character ratings

[https://www.nytimes.com/interactive/2017/08/09/upshot/game-of-thrones-
chart.html](https://www.nytimes.com/interactive/2017/08/09/upshot/game-of-thrones-chart.html)

ggplot2: inheritance

Template for a simple plot

Main
function

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

Shape
layer

```
geom_... ( non-variable adjustments )
```

Behind the scenes

Main
function

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

Shape
layer

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

Inheritance

data and aesthetics will carry through from main function to shape layers

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 )
```

+

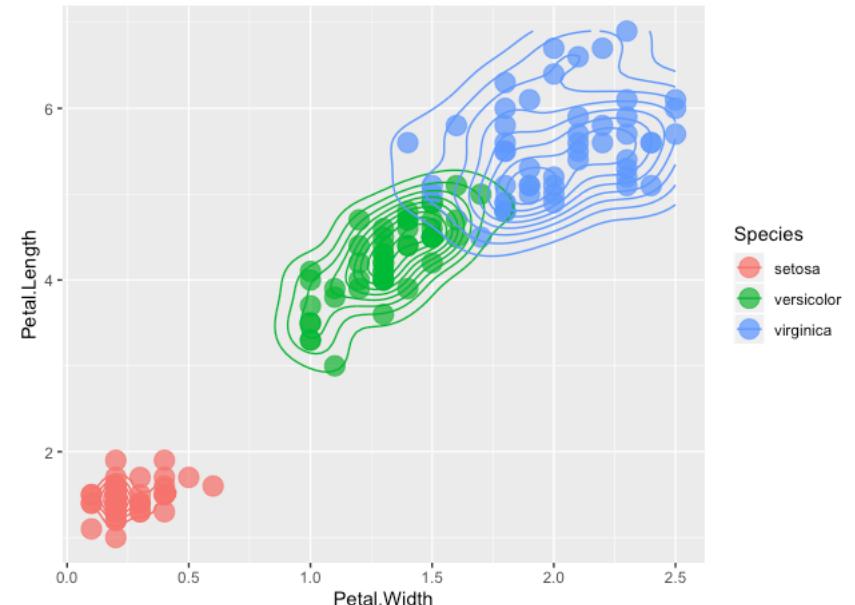
+

+

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()
```



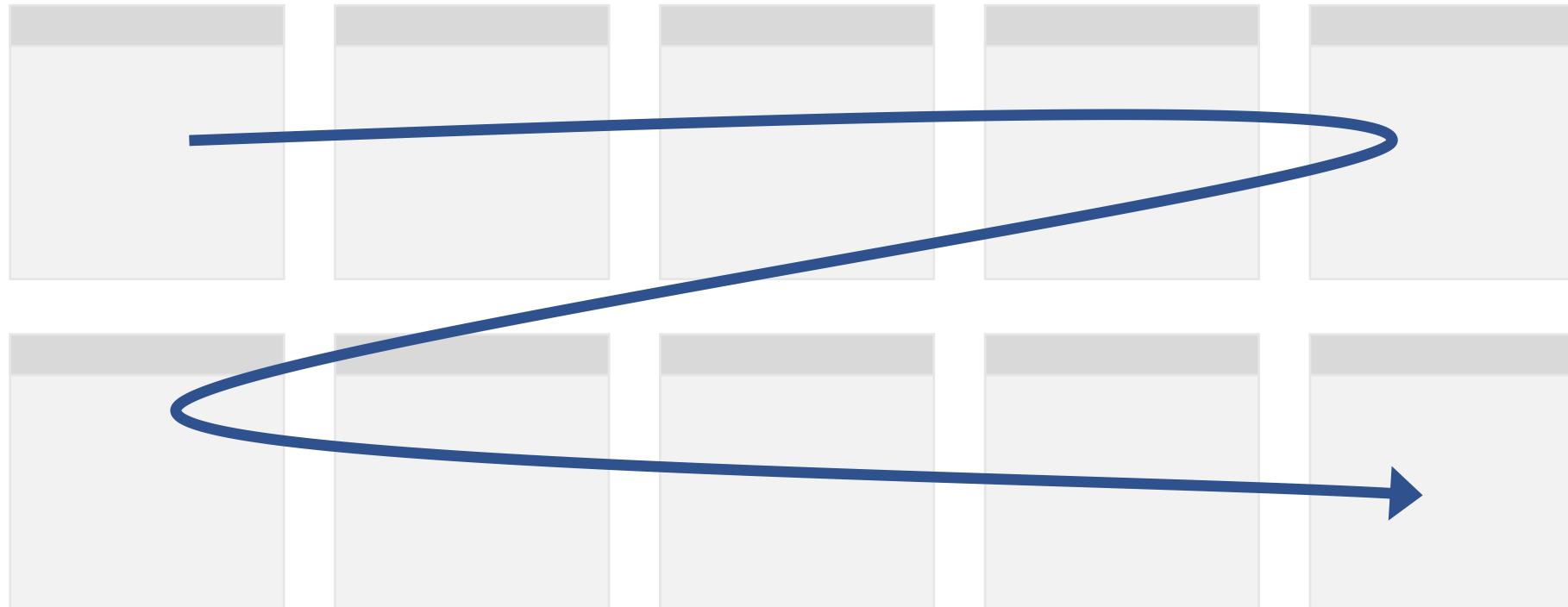
Dataset 2: Star Wars character data

<https://dplyr.tidyverse.org/reference/starwars.html>

Creating repeated charts

facet_wrap()

```
+ facet_wrap(vars(variable))
```

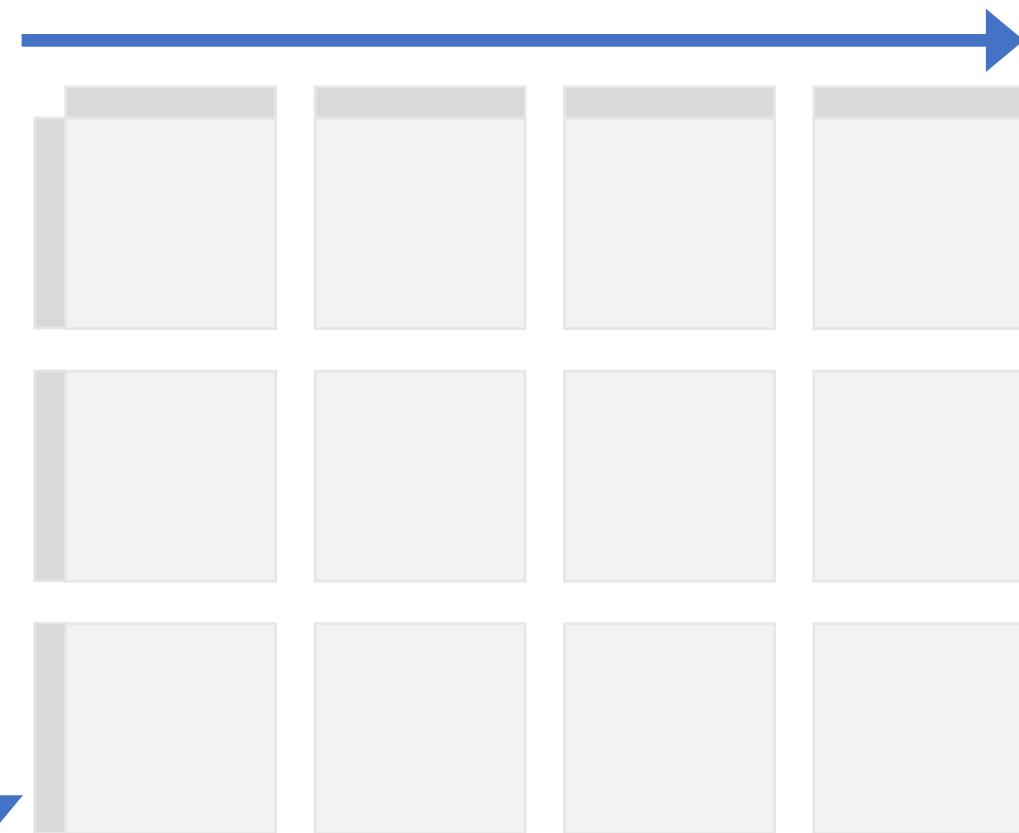


facet_grid()

```
+ facet_grid(rows=vars(yvar,  
cols=vars(xvar))
```

Another categorical variable

One categorical variable



Inheritance

data and aesthetics will carry through from main function to shape layers

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 )
```

+

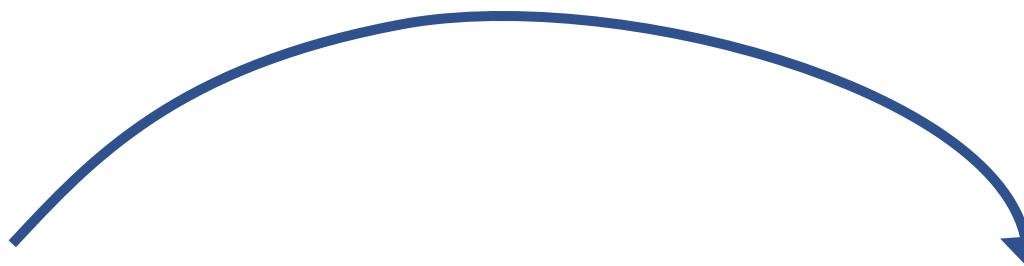
+

+

Helpful data manipulation

About %>%

- Loads automatically with tidyverse
- Used throughout tidyverse (except for ggplot2)
- Pushes data from the left into the function on the right



data_frame %>% function(args)

A blue curved arrow originates from the left side of the word "data_frame" and points to the right side of the word "function".

filter

Select a subset of rows

```
data %>% dplyr::filter(name == "John")
```

same as

```
dplyr::filter(data, name == "John")
```

<https://www.rstudio.com/resources/cheatsheets/#dplyr>

select

Select a subset of columns (many options!)

```
data %>% dplyr::select(id, name, age)
```

```
data %>% dplyr::select(-count)
```

<https://www.rstudio.com/resources/cheatsheets/#dplyr>

drop_na

Remove rows with NA values, either in any column or in specified columns

```
data %>% drop_na()
```

```
data %>% drop_na(age)
```

<https://www.rstudio.com/resources/cheatsheets/#import>

Working with text variables

Text variables

In R, “character” variables

Gender	Age	Household Income	Education
Response	Response	Response	Response
Male	18-29		High school degree
Male	18-29	\$0 - \$24,999	Bachelor degree
Male	18-29	\$0 - \$24,999	High school degree
Male	18-29	\$100,000 - \$149,999	Some college or Associate degree
Male	18-29	\$100,000 - \$149,999	Some college or Associate degree
Male	18-29	\$25,000 - \$49,999	Bachelor degree
Male	18-29		High school degree
Male	18-29		High school degree
Male	18-29	\$0 - \$24,999	Some college or Associate degree
Male	18-29	\$25,000 - \$49,999	Some college or Associate degree
Male	18-29	\$25,000 - \$49,999	Bachelor degree
Male	30-44	\$50,000 - \$99,999	Graduate degree
Male	18-29		High school degree
Male	18-29	\$0 - \$24,999	Some college or Associate degree
Male	18-29	\$50,000 - \$99,999	Bachelor degree

Problems with text variables:
Ordering

Factors

- Default ordering for categories: **alphabetical**
- Converting to factor allows you to:
 - Specify “levels” for a categorical variable
 - Specify the order of those levels
 - Specify whether the factor is “ordered”

<https://r4ds.had.co.nz/factors.html>

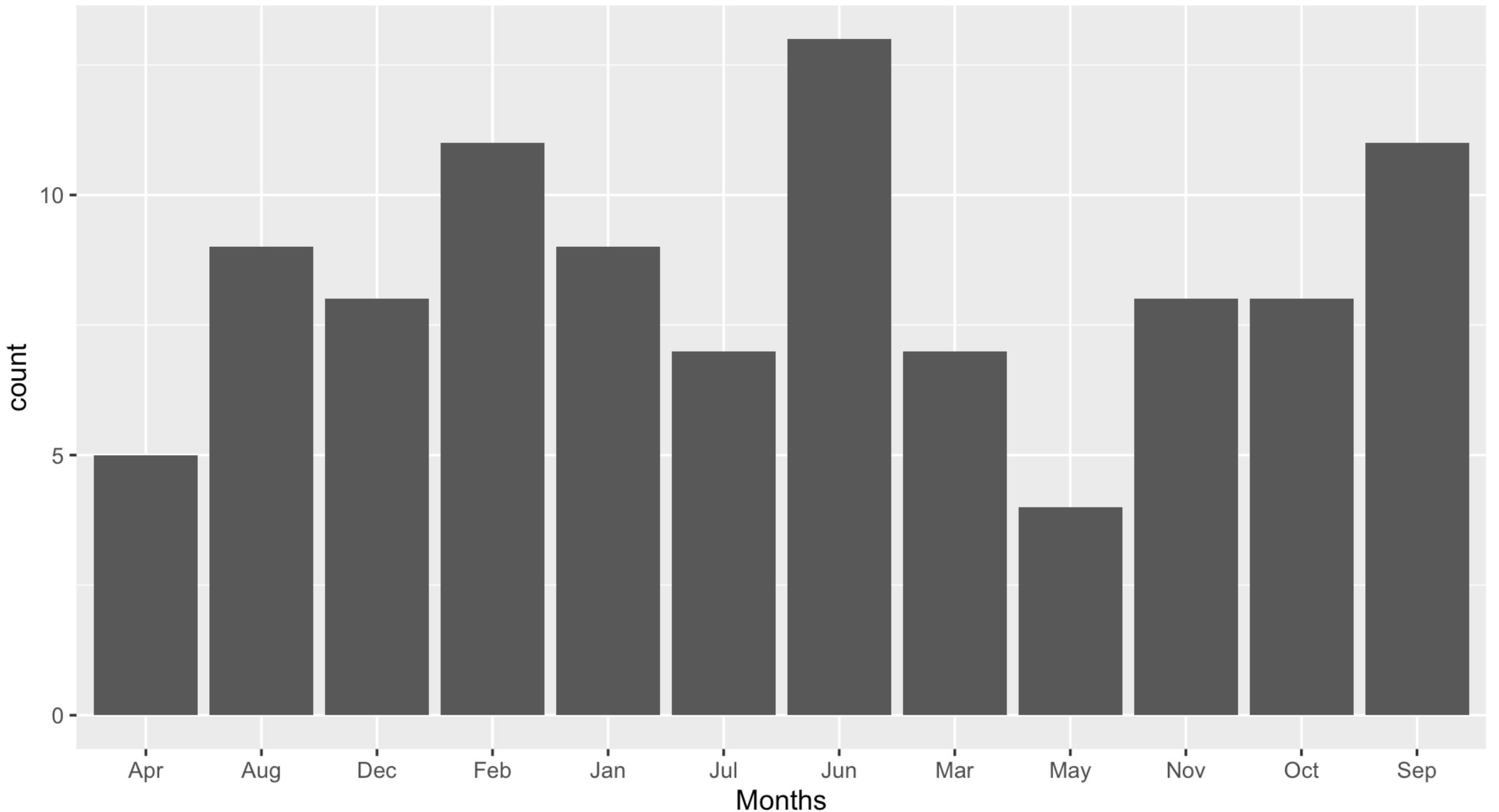
```
> x1 <- c("Dec", "Apr", "Jan",  
"Mar")
```

```
> factor(x1)  
[1] Dec Apr Jan Mar  
Levels: Apr Dec Jan Mar
```

```
> month_levels <- c( "Jan", "Feb",  
"Mar", "Apr", "May", "Jun", "Jul",  
"Aug", "Sep", "Oct", "Nov", "Dec" )
```

```
> y1 <- factor(x1,  
                levels = month_levels)  
> y1  
[1] Dec Apr Jan Mar
```

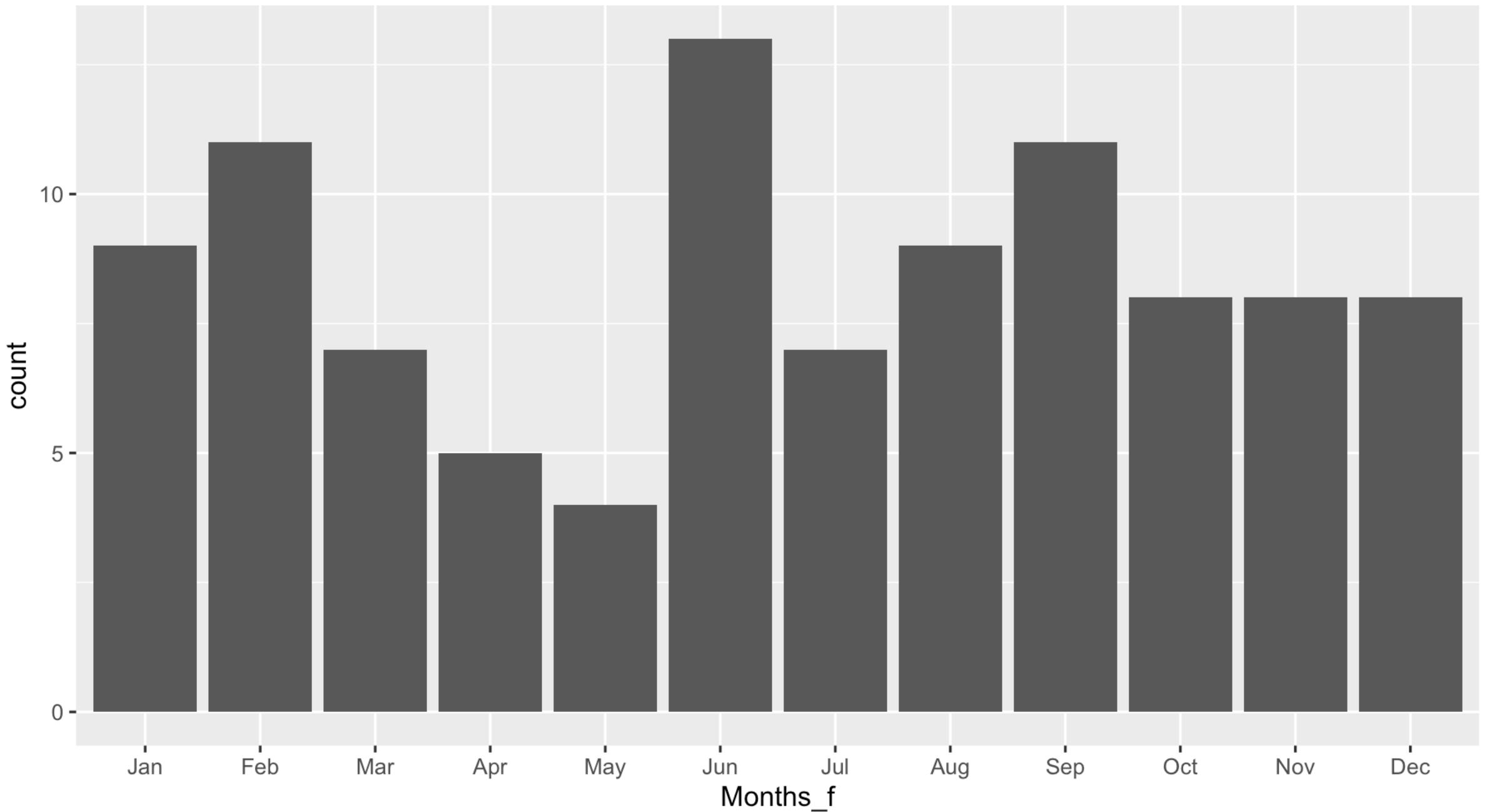
```
Levels: Jan Feb Mar Apr May Jun Jul  
Aug Sep Oct Nov Dec
```

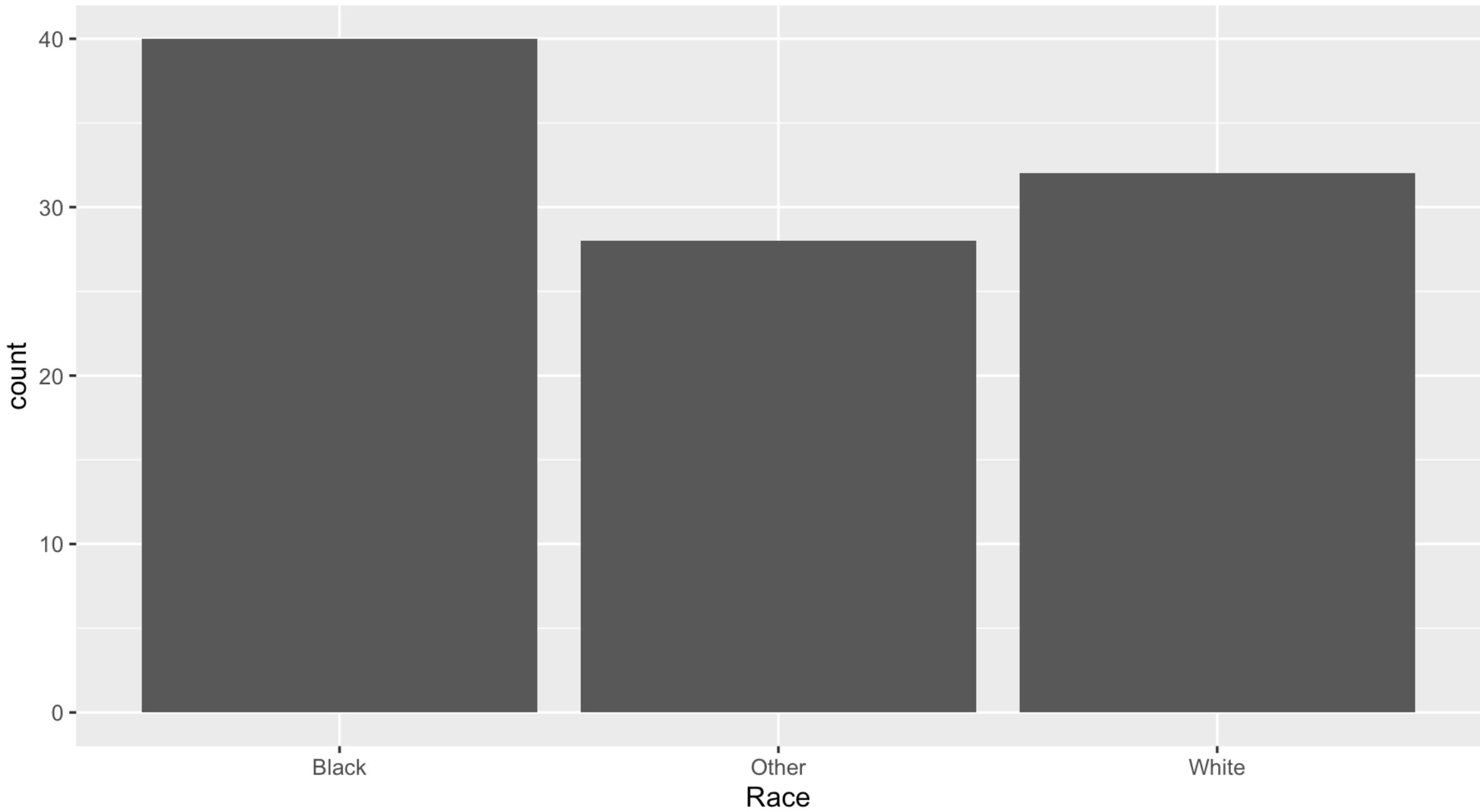


Order by meaning

```
month_levels <- c( "Jan", "Feb", "Mar", "Apr",
"May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
"Dec" )

data <- data %>%
  mutate(Months_f = Months %>%
    as_factor() %>%
    fct_relevel(month_levels) )
```

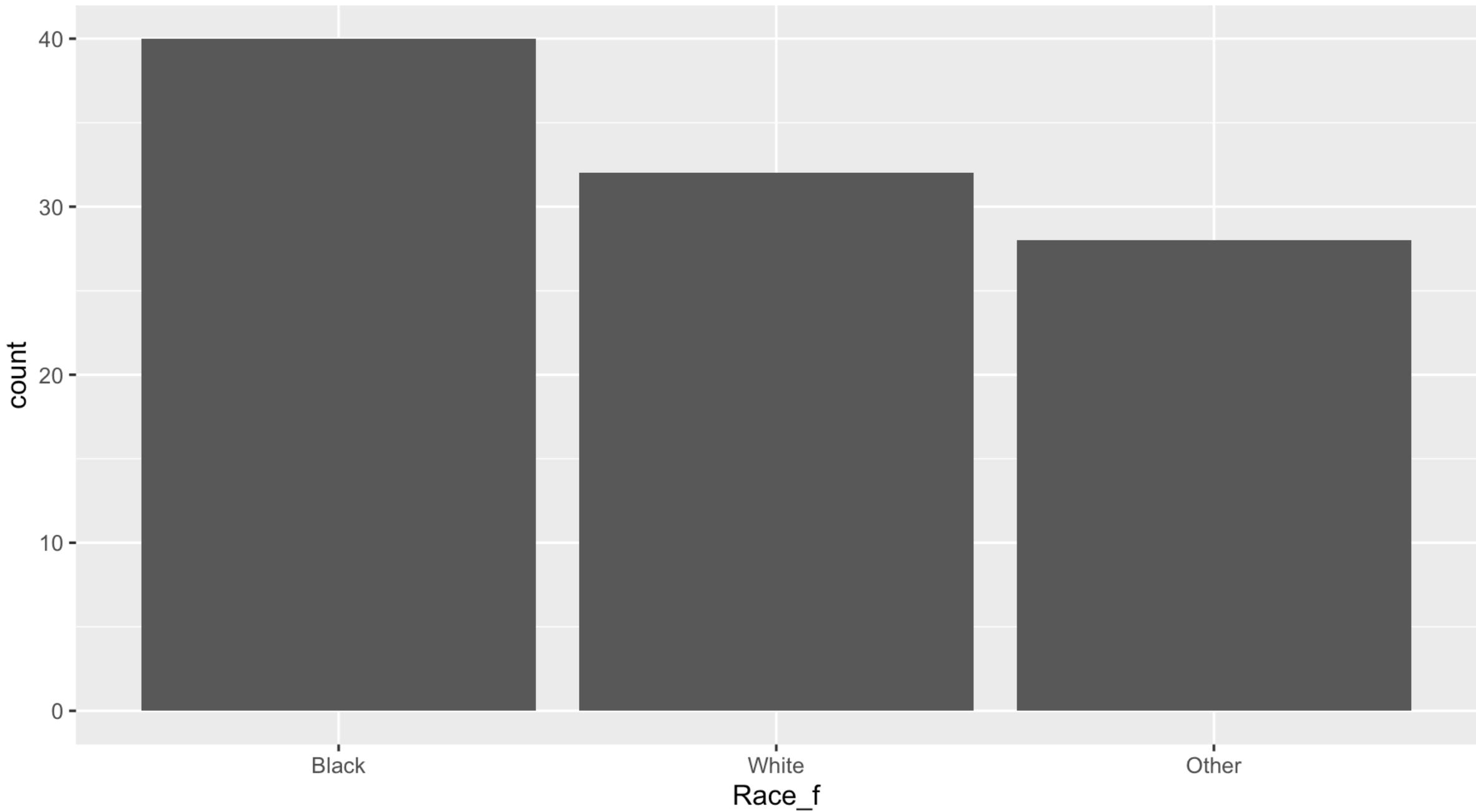




Order by value (usingforcats)

```
demo <- data %>%
  mutate(Race_f = Race %>%
          as_factor() %>%
          fct_infreq()))

ggplot(data,
       aes(Race %>%
             as_factor() %>%
             fct_infreq()) ) +
  geom_bar()
```



forcats package: helpful functions

- `as_factor(char_var)`:
convert a character variable to a factor
- `fct_infreq(factor)`:
take factor levels and set the order according to
(inverse) category frequency
- `fct_reorder(factor, num_var)`:
sort factor levels by a second, numerical variable
(like a pre-calculated count or average)

<https://www.rstudio.com/resources/cheatsheets/#forcats>

Note about `read.csv` (base R)

- Converts string variables to factors by default
- Can either:
 - Include `stringsAsFactors=FALSE`
 - Use `read_csv()` instead

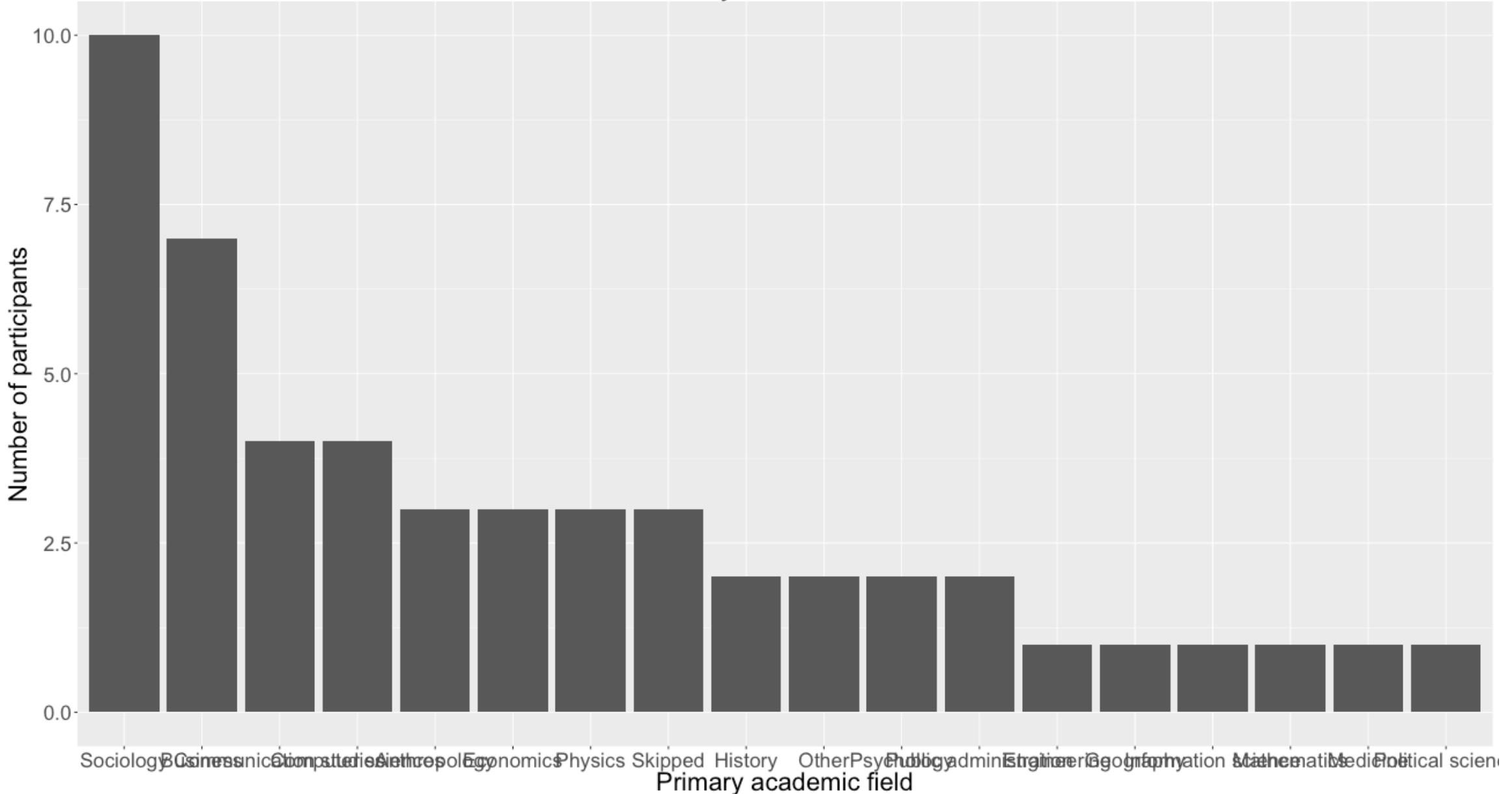
Factoring resources

From Amelia McNamara:

- RStudioConf 2019 slides:
[Working with Categorical Data in R Without Losing Your Mind](#)
- [Wrangling Categorical Data in R article](#)
- [Wrangling Categorical Data in R repository](#)

Problems with text variables:
Long category names

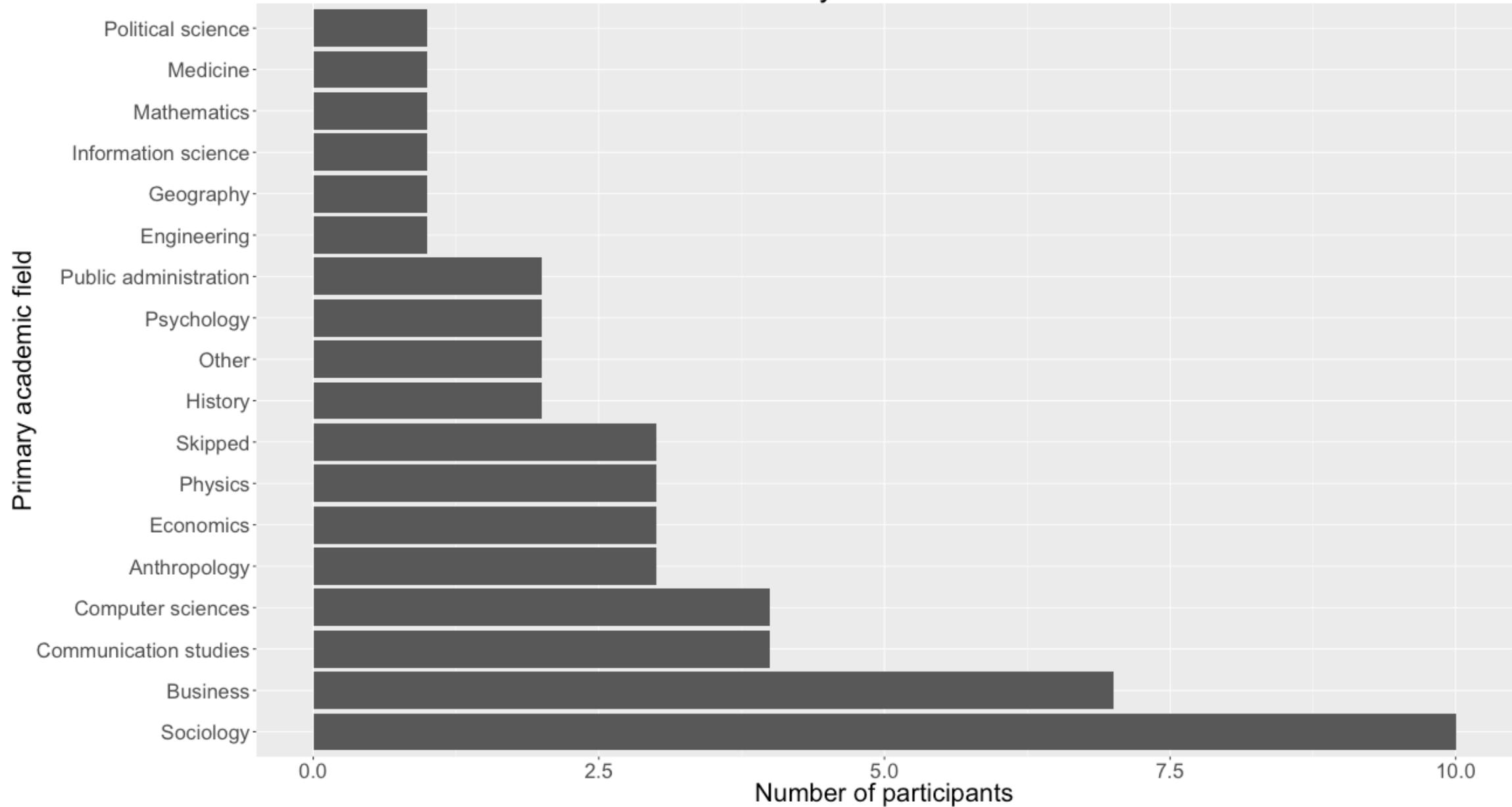
Primary academic field



In ggplot2, have to flip the axes

```
+ coord_flip()
```

Primary academic field



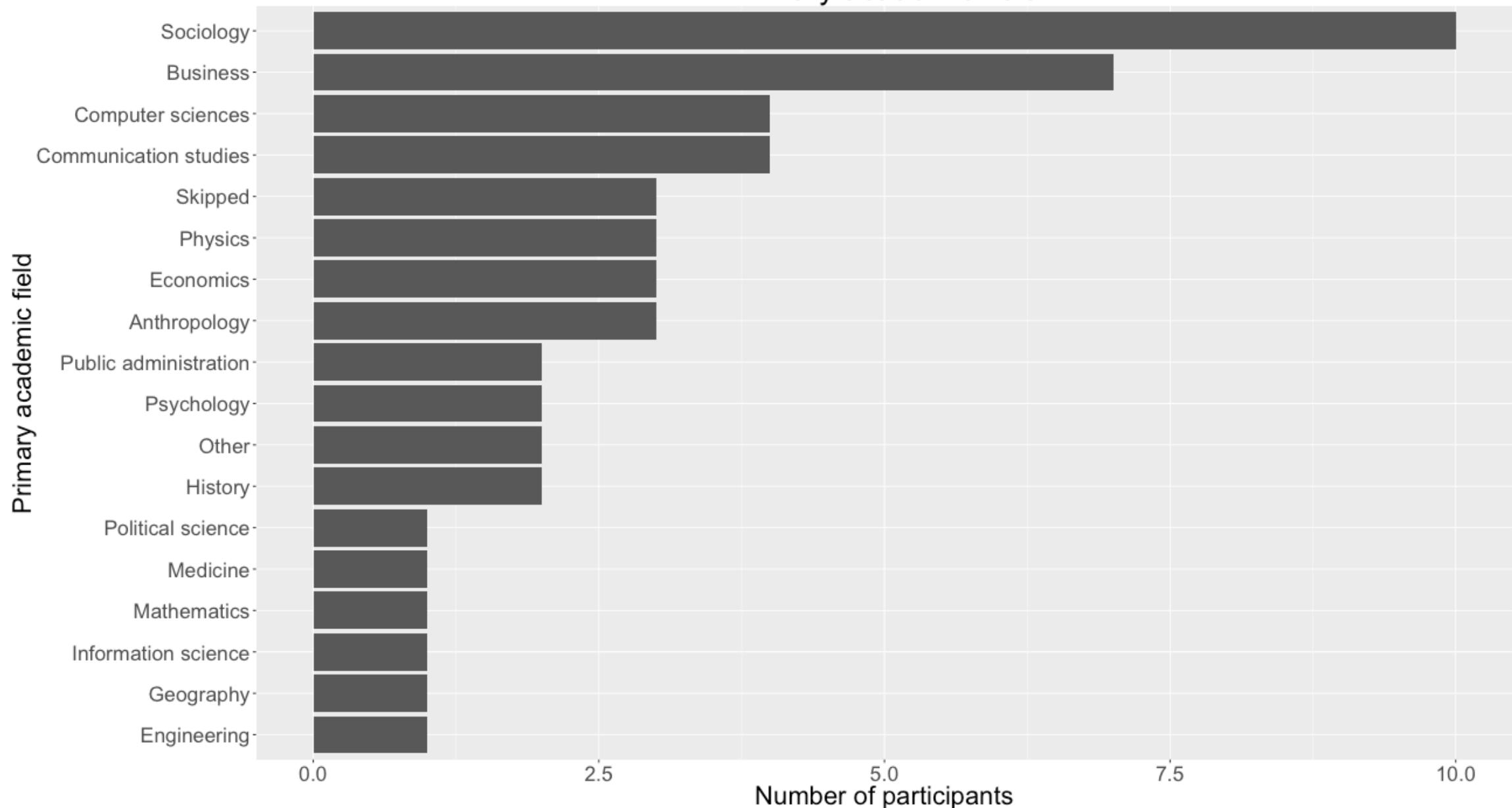
When you flip axes, you sort the other way

```
data$academic_field <-  
  fct_rev(fct_infreq(  
    as_factor(data$academic_field) ))
```



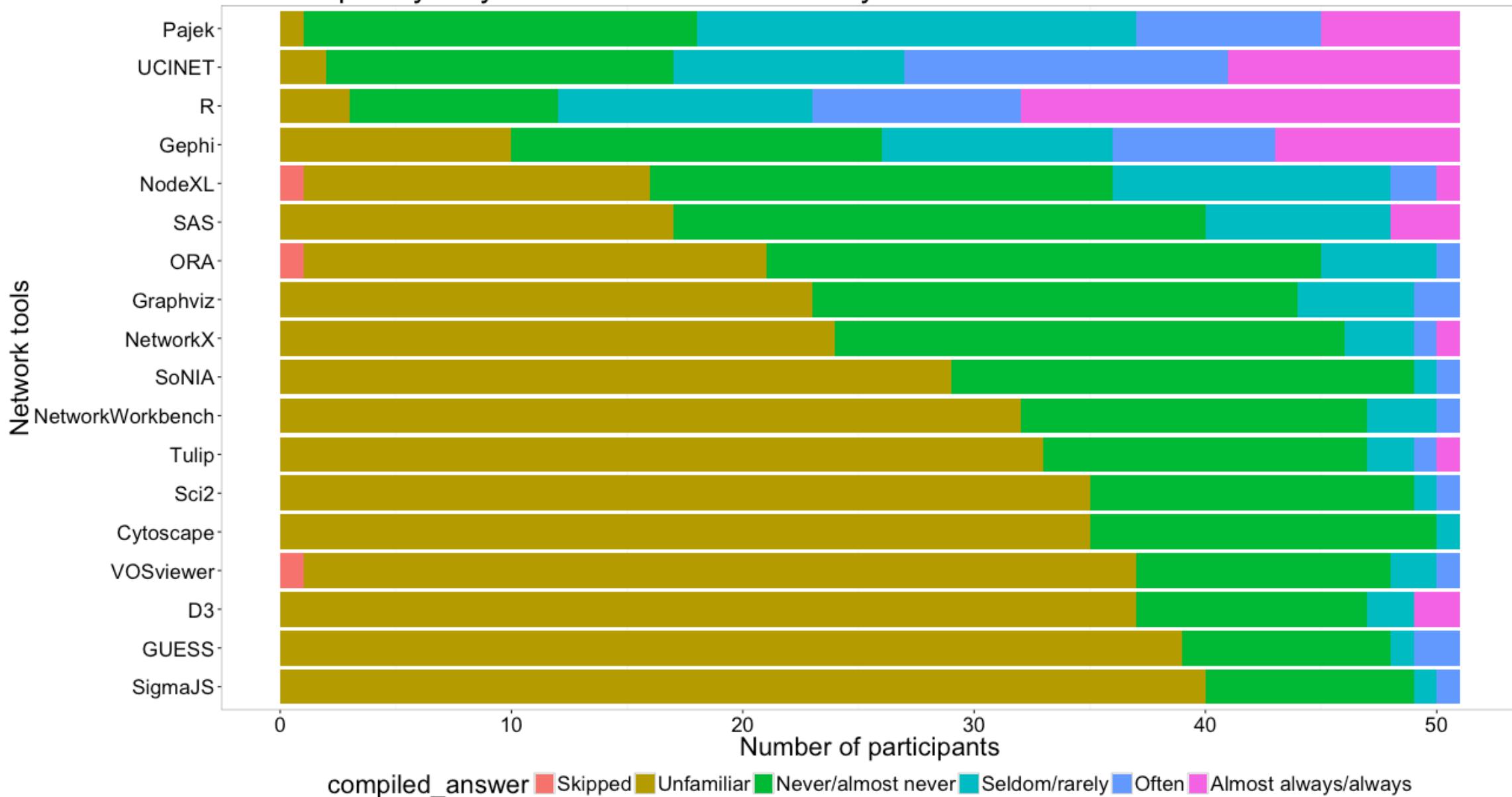
Have to reverse the
order of the levels

Primary academic field



Problems with text variables:
Arbitrary colors

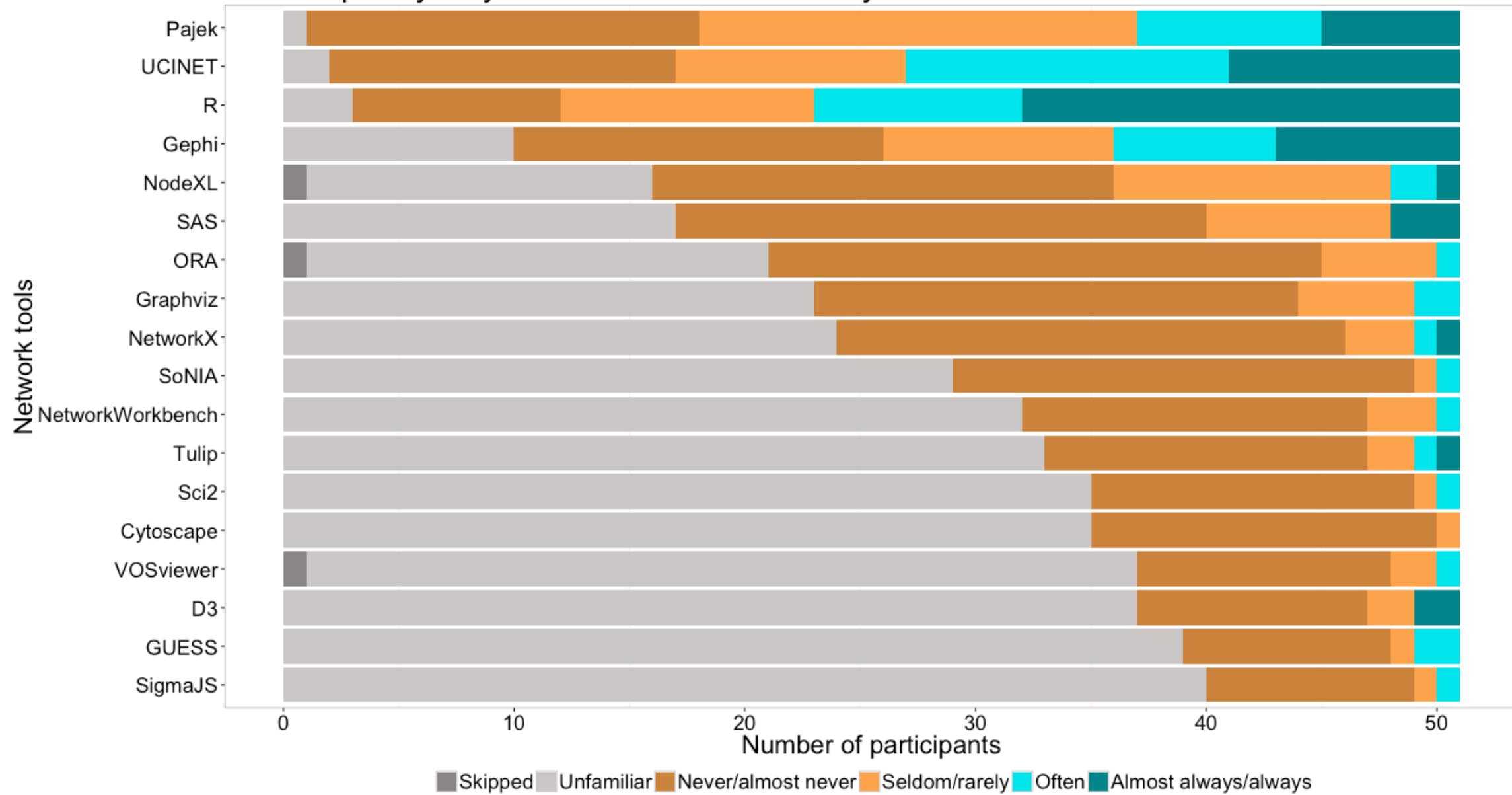
How frequently do you use these tools for analysis?



Select colors manually, or use alternate palette

```
scale_fill_manual(  
  values=c("snow4", "snow3",  
          "tan3", "tan1",  
          "turquoise2", "turquoise4"))  
  
# Also see package RColorBrewer  
scale_fill_brewer(palette="BrBG")
```

How frequently do you use these tools for analysis?



Advanced ggplot2 workshop

[Fall 2020 workshop video](#)

Dataset 3: Star Wars opinion survey

<https://fivethirtyeight.com/features/americas-favorite-star-wars-movies-and-least-favorite-characters/>

Final advice

Additional decisions

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
<http://ggplot2.tidyverse.org/>
- R Graphics Cookbook (recipes for plots)
<http://www.cookbook-r.com/Graphs/index.html>
- R for Data Science (online book that includes ggplot2)
<http://r4ds.had.co.nz/>
- ggplot2: Elegant Graphs for Data Analysis (book by Hadley Wickham)
<http://ggplot2.org/book/>
- ggplot2 cheatsheet (also in RStudio)
<http://bit.ly/ggplot2-cheatsheet>
- [Data Carpentry lesson on ggplot2](#)
- [Data Visualization: A Practical Introduction, by Kieran Healy](#)
- [RStudio “Visualize Data” Primer](#)

Videos of past workshops

The image shows a screenshot of a Panopto video player interface. At the top left is the Panopto logo and the title "Figures and Posters". To the right are links for "Help" and "Sign in". Below the title, there is a thumbnail image of two people, a man and a woman, standing in front of a whiteboard in a classroom setting. The whiteboard has some handwritten text on it. To the right of the thumbnail, the main video area displays the title "Designing Academic Figures and Posters" in large bold letters, followed by the date "March 4, 2016" and a link to "Slides: <http://duke.box.com/PostersSpring2016>". Below the title, two speakers are introduced: "Angela Zoss" and "Eric Monson", both described as Data Visualization Coordinators or Analysts from Data and Visualization Services. The video player includes a control bar at the bottom with a play button, a progress bar showing 0:03, a duration of 1:22:45, a speed control set to 1X, and options for "Quality" and "Hide". Below the control bar, there are four small thumbnail images related to poster design: "Good Posters", "Causal Observation", "Poster Examples", and "Purpose of a poster".

<http://bit.ly/DVSvideos>

Questions?

angela.zoss@duke.edu