

Visualization in R using ggplot2

Angela Zoss

9/19/19

<https://github.com/amzoss/ggplot2-F19>

Try right now:

Open RStudio

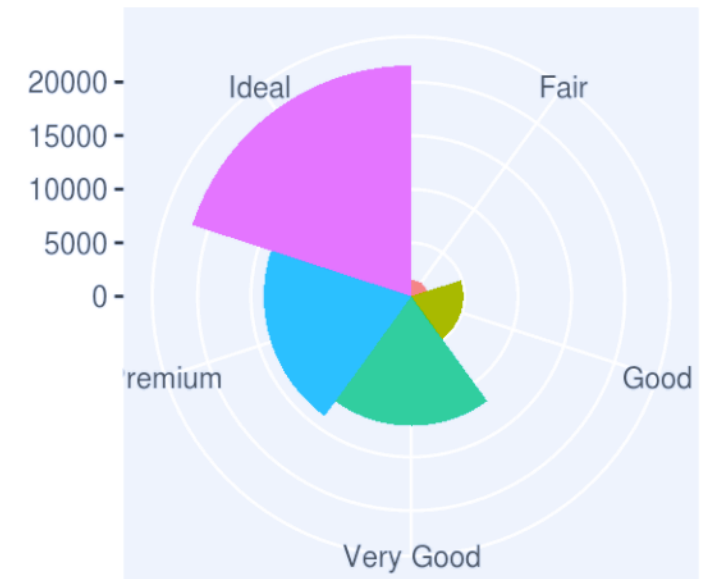
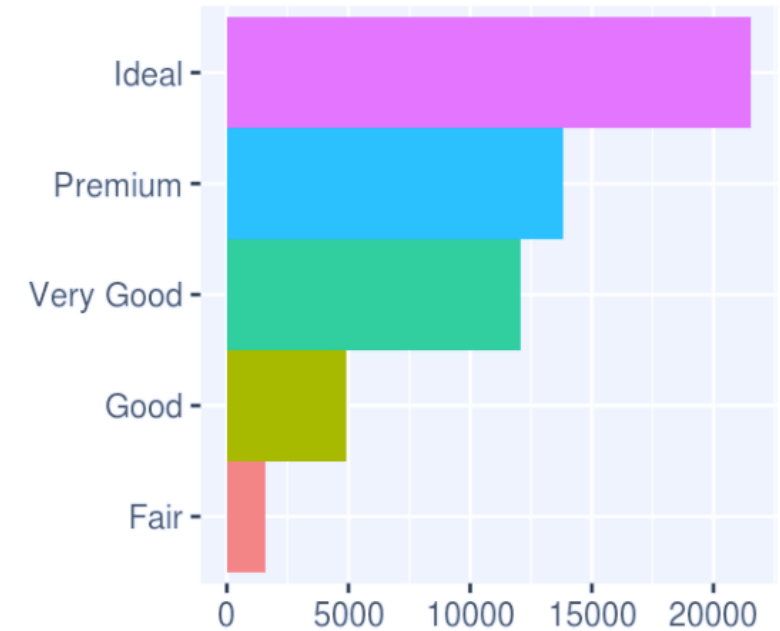
Try running “library(tidyverse)”

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.



Grammar of graphics

1. DATA: a set of data operations that create variables from datasets
2. TRANS: variable transformations (e.g., rank)
3. SCALE: scale transformations (e.g., log)
4. COORD: a coordinate system (e.g., polar)
5. ELEMENT: graphs (e.g., points) and their aesthetic attributes (e.g., color)
6. GUIDE: one or more guides (axes, legends, etc.).

Why ggplot2 instead of base R?

- nice defaults
- easy faceting
- (arguably) more natural syntax
- can switch chart types more easily

“Why I use ggplot2”, David Robinson

<http://varianceexplained.org/r/why-i-use-ggplot2/>

R vs. Excel, Tableau, etc.

Questions to ask:

- Are you already using R? Why switch?
- Are you going to have to share this process or reproduce it? Try R!
- Is it a quick project, or will others work on it? Maybe Excel is fine.
- Do you need to try a bunch of charts quickly, build interactive components, etc? Tableau might be more powerful and faster.

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-F19>

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

In RStudio:

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

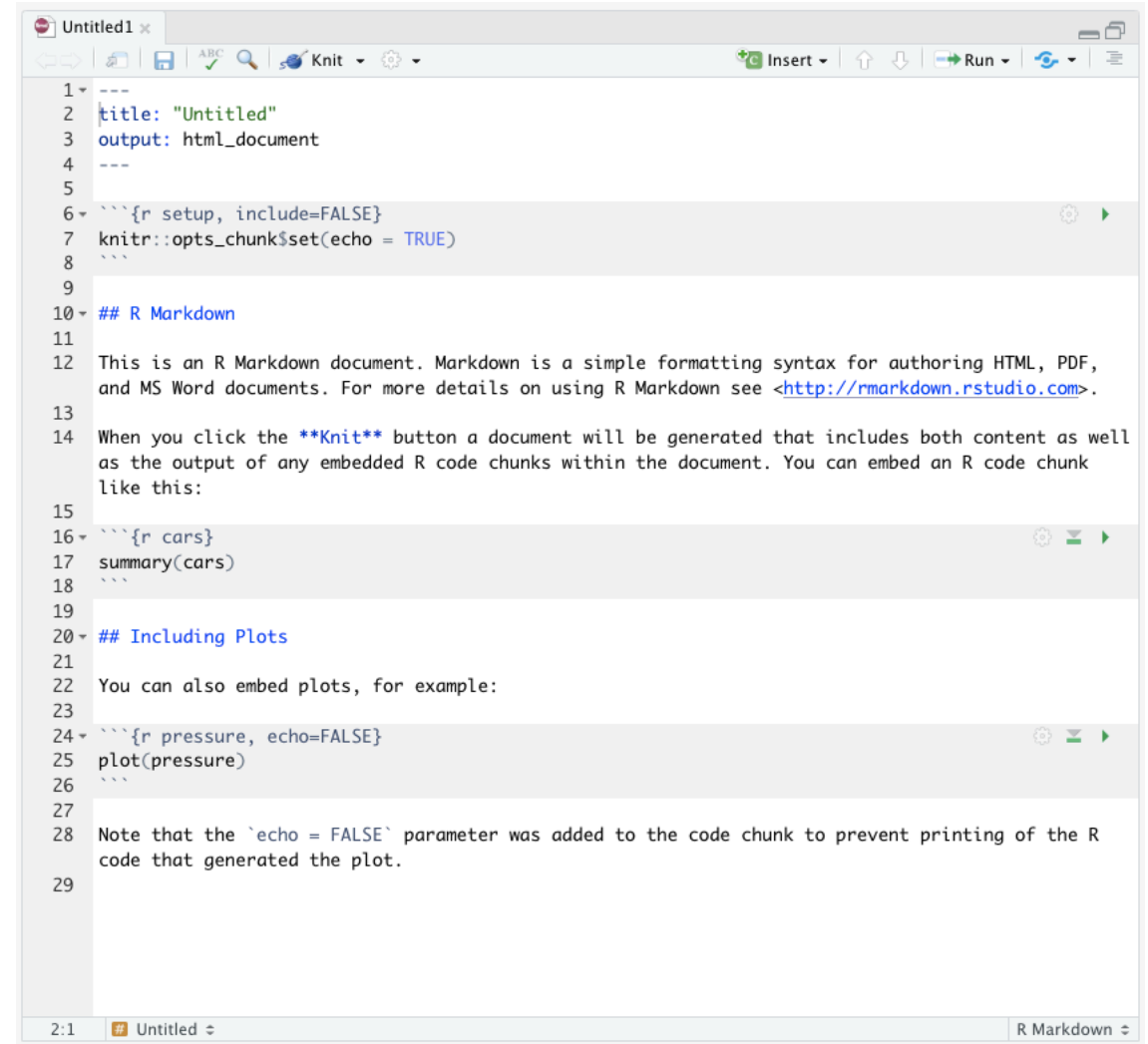
Why R Markdown?

- 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



The screenshot shows a text editor window titled 'Untitled1' with a toolbar at the top containing icons for undo, redo, save, search, and a 'Knit' button. The document content is as follows:

```
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,
13 and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.
14
15 When you click the Knit button a document will be generated that includes both content as well
16 as the output of any embedded R code chunks within the document. You can embed an R code chunk
17 like this:
18
19 ```{r cars}
20 summary(cars)
21 ```
22
23 ## Including Plots
24
25 You can also embed plots, for example:
26
27 ```{r pressure, echo=FALSE}
28 plot(pressure)
29 ```
30
31 Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R
32 code that generated the plot.
```

The status bar at the bottom indicates '2:1' and 'Untitled', and the bottom right corner shows 'R Markdown'.

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> +
<SCALE FUNCTION> +
<THEME FUNCTION>

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

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, yend, 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)) ~ x, y, alpha, color, fill, group, linetype, size

a + **geom_ribbon**(aes(ymin = unemploy - 900, ymax = unemploy + 900)) ~ x, y, 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(f))

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, stroke

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

continuous x, discrete y

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

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

THREE VARIABLES

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

i + **geom_contour**(aes(z = z))
x, y, z, alpha, colour, group, linetype, size, weight

i + **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, linetype, size, width

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, y, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh**())


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

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

maps

data <- data.frame(murder = USArrests\$Murder, state = tolower(row.names(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



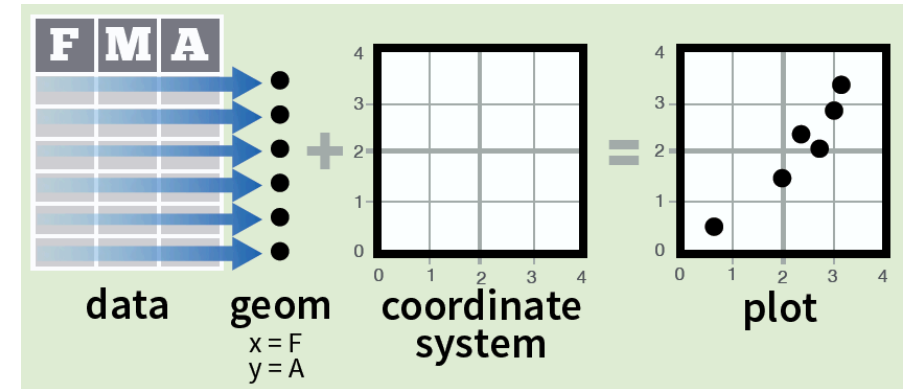
RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-448-1212 • rstudio.com • Learn more at <http://ggplot2.tidyverse.org> • ggplot2 2.1.0 • Updated: 2016-11

<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 )
```

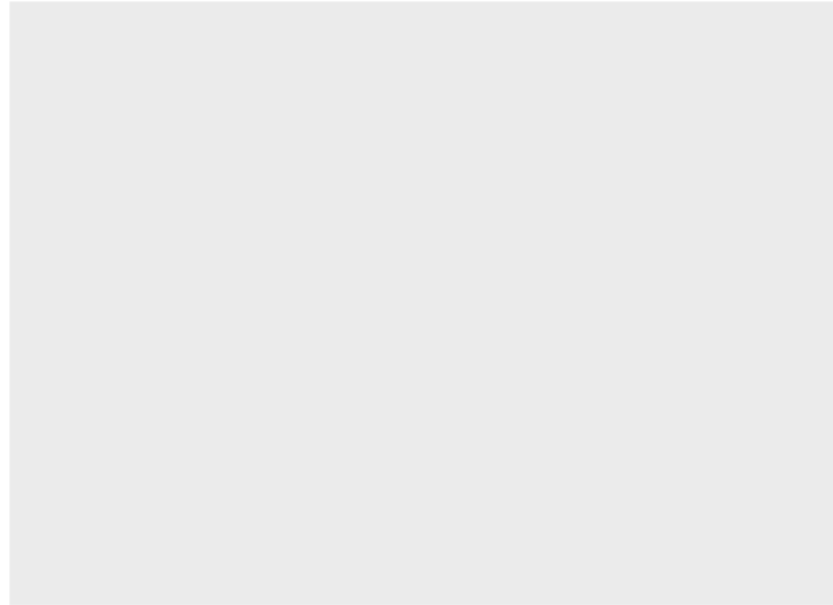
**Shape
layer**

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"

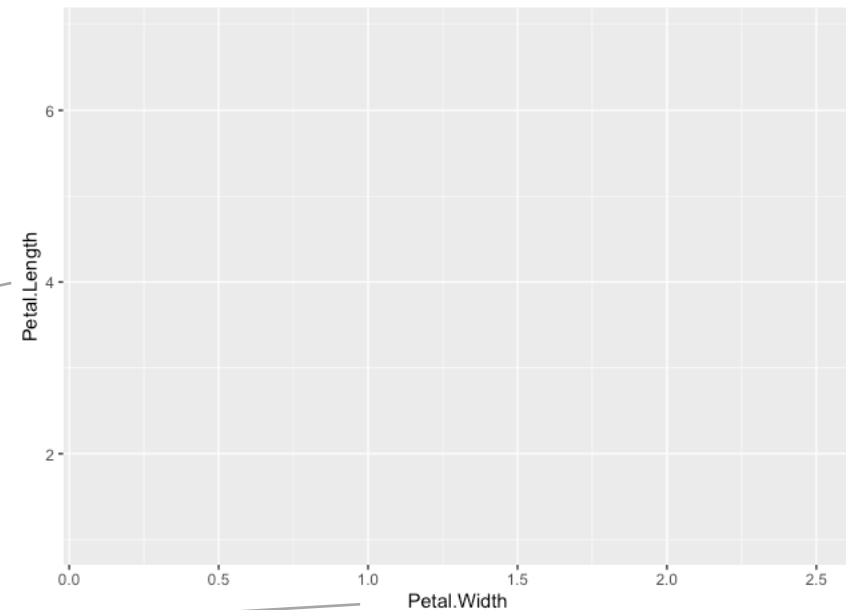
Petal.Width	Petal.Length	Species
0.3	1.4	setosa
1.3	4.0	versicolor
2.1	5.7	virginica

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) )
```

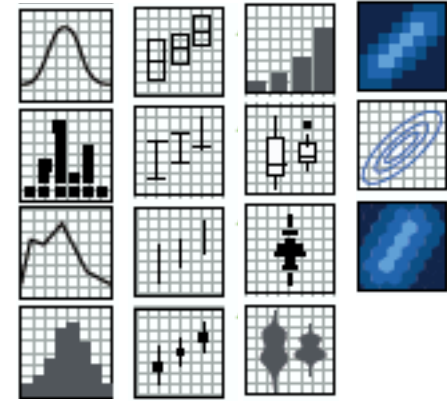
+

**Shape
layer**

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

Types of geoms

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



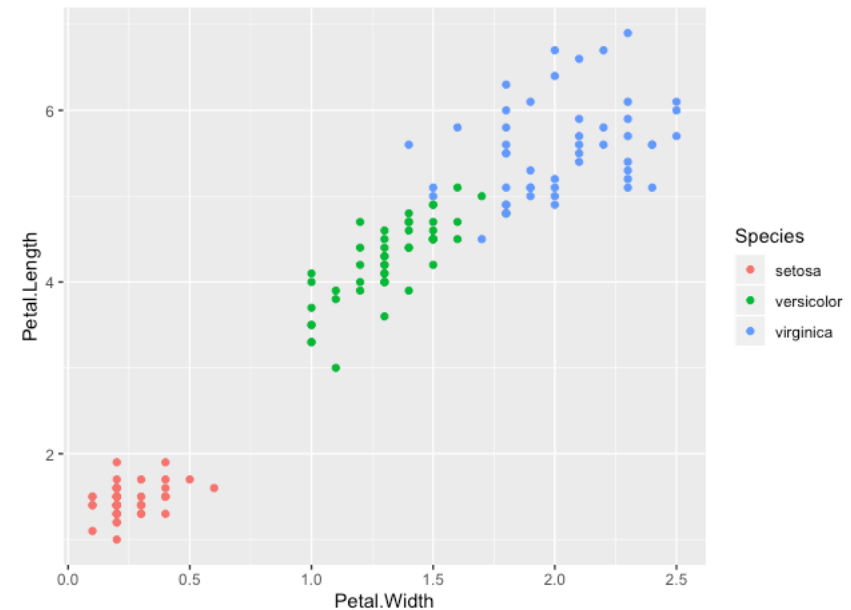
<http://bit.ly/ggplot2-cheatsheet>

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**

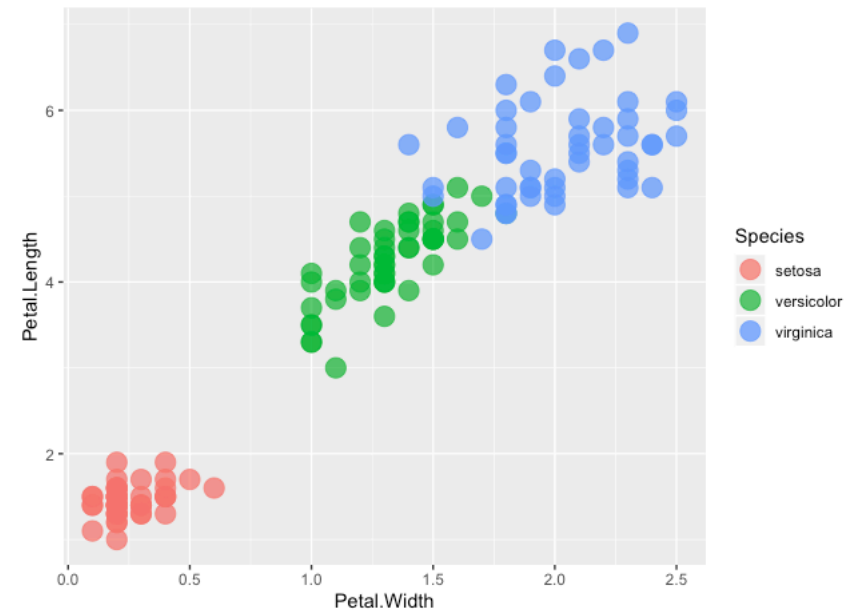
```
geom_... ( non-variable adjustments )
```

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)
```



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 )
```

+

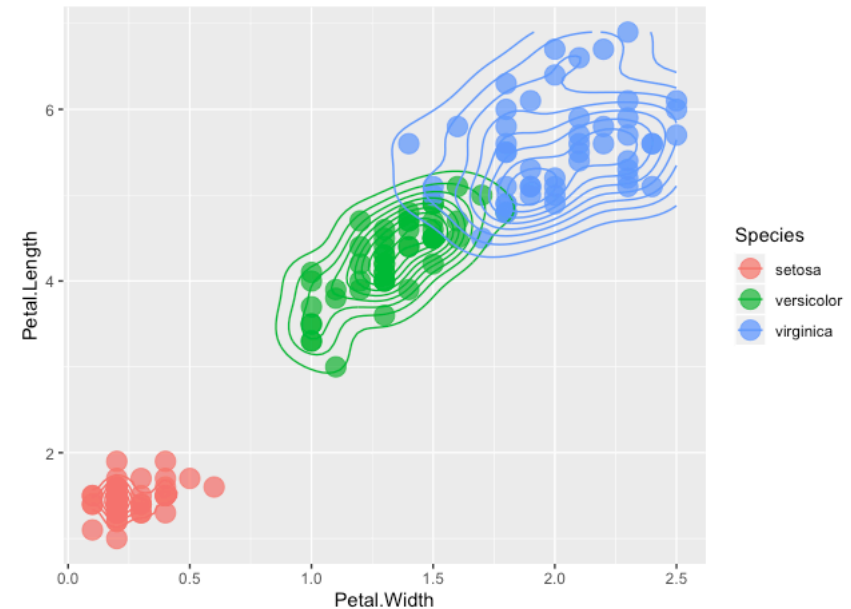
+

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()
```



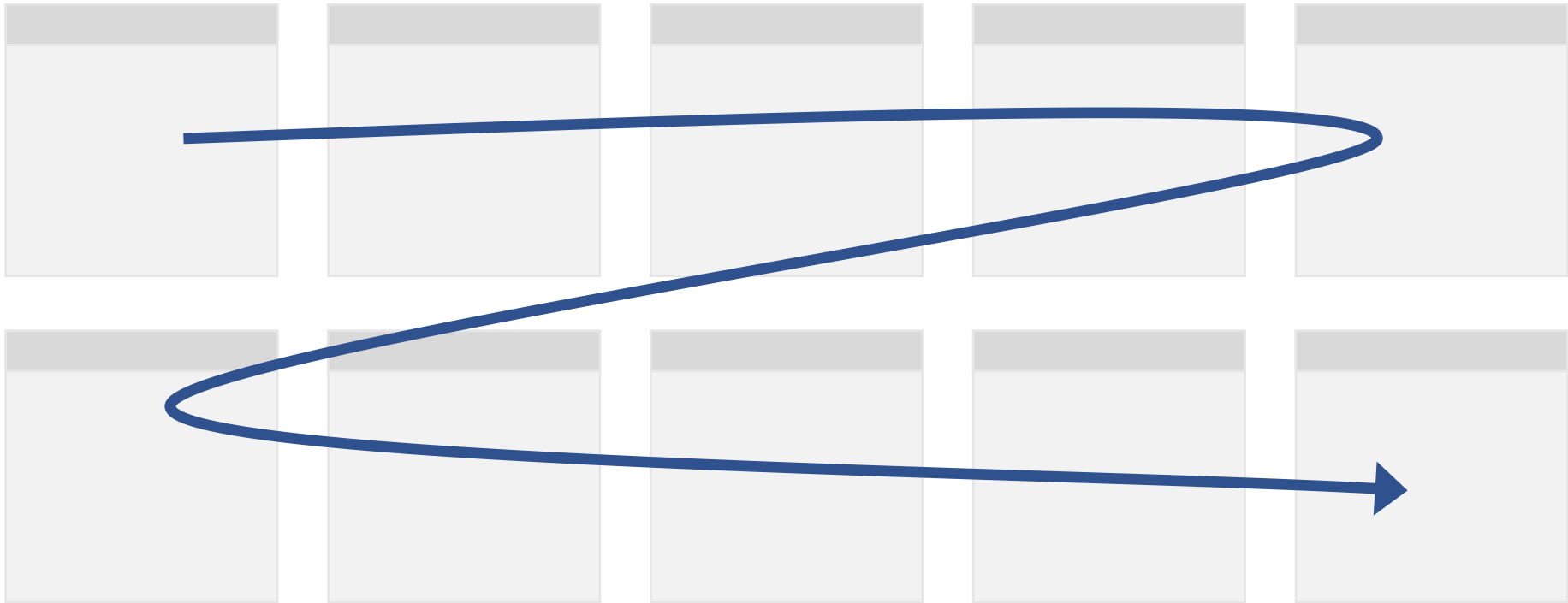
Dataset 1: Game of Thrones character ratings

<https://www.nytimes.com/interactive/2017/08/09/upshot/game-of-thrones-chart.html>

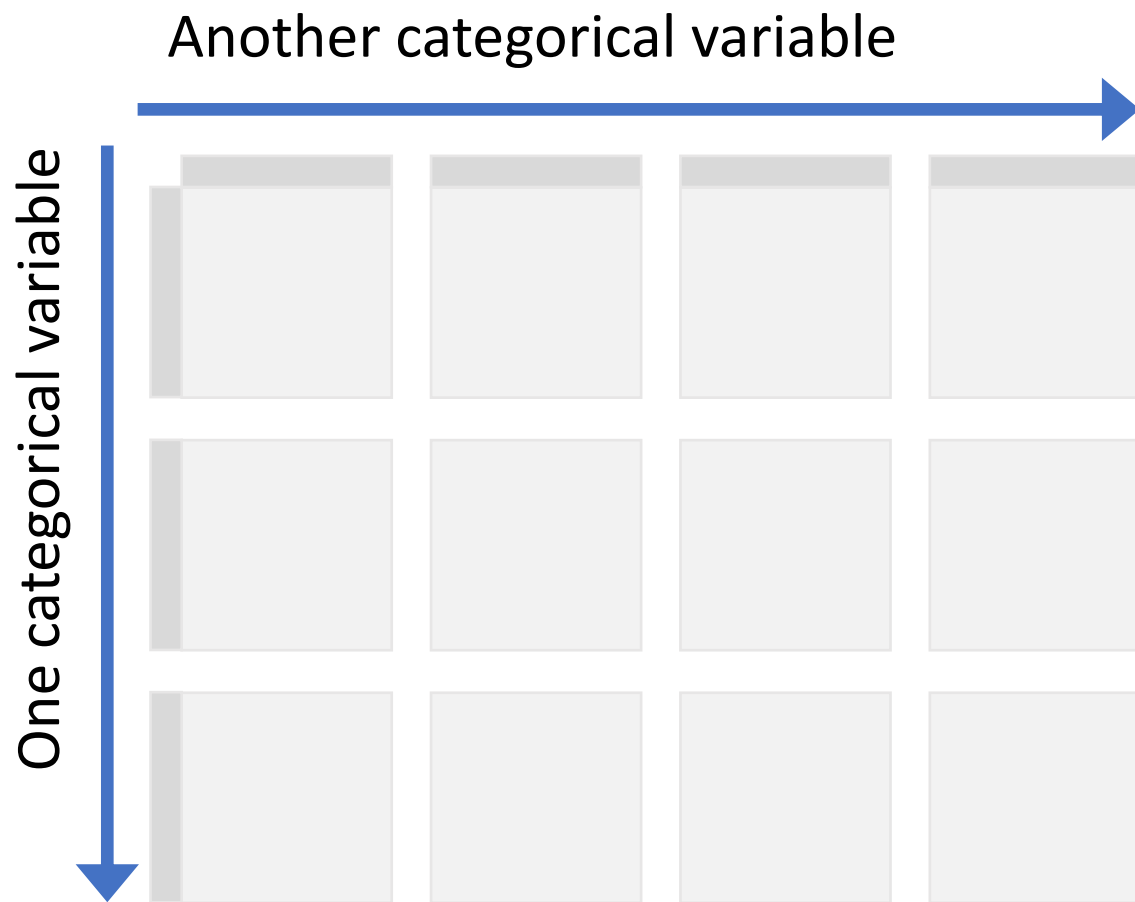
Creating repeated charts

facet_wrap()

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



facet_grid()

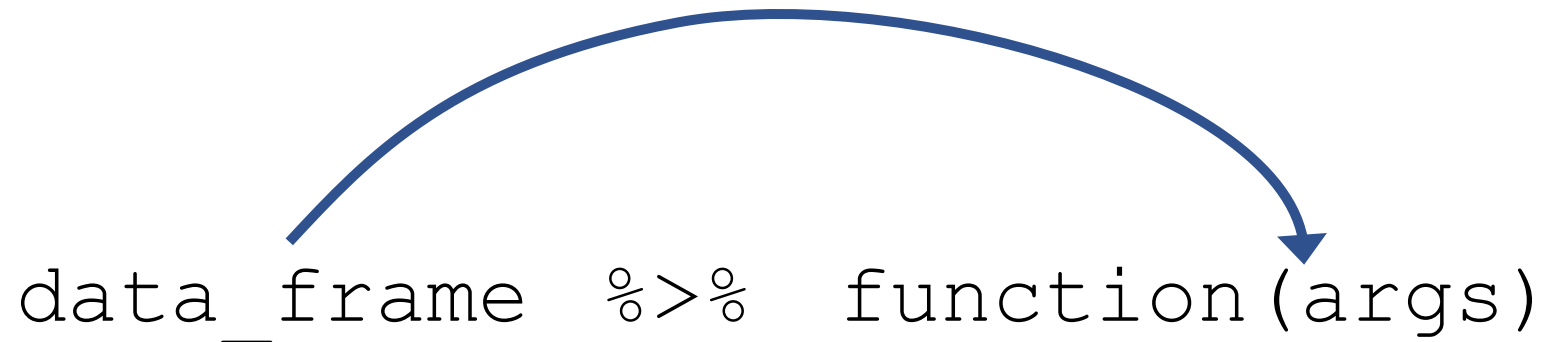


```
+ facet_grid(rows=vars(yvar,  
                    cols=vars(xvar))
```

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



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>

Dataset 2:

Star Wars character data

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

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

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
```

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)

Note about read.csv (base R)

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

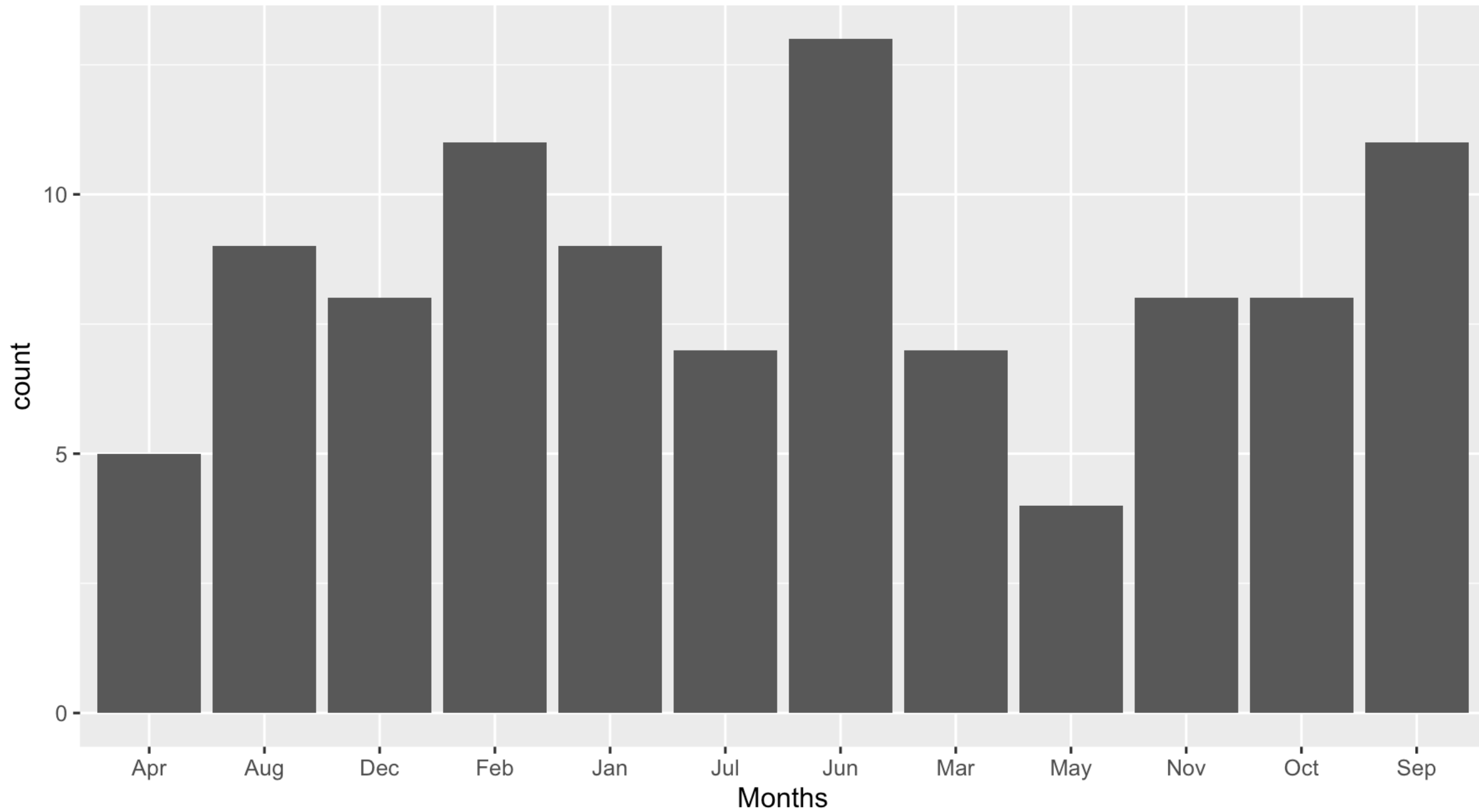
Factoring resources

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](#)

Design Principles for Text Variables

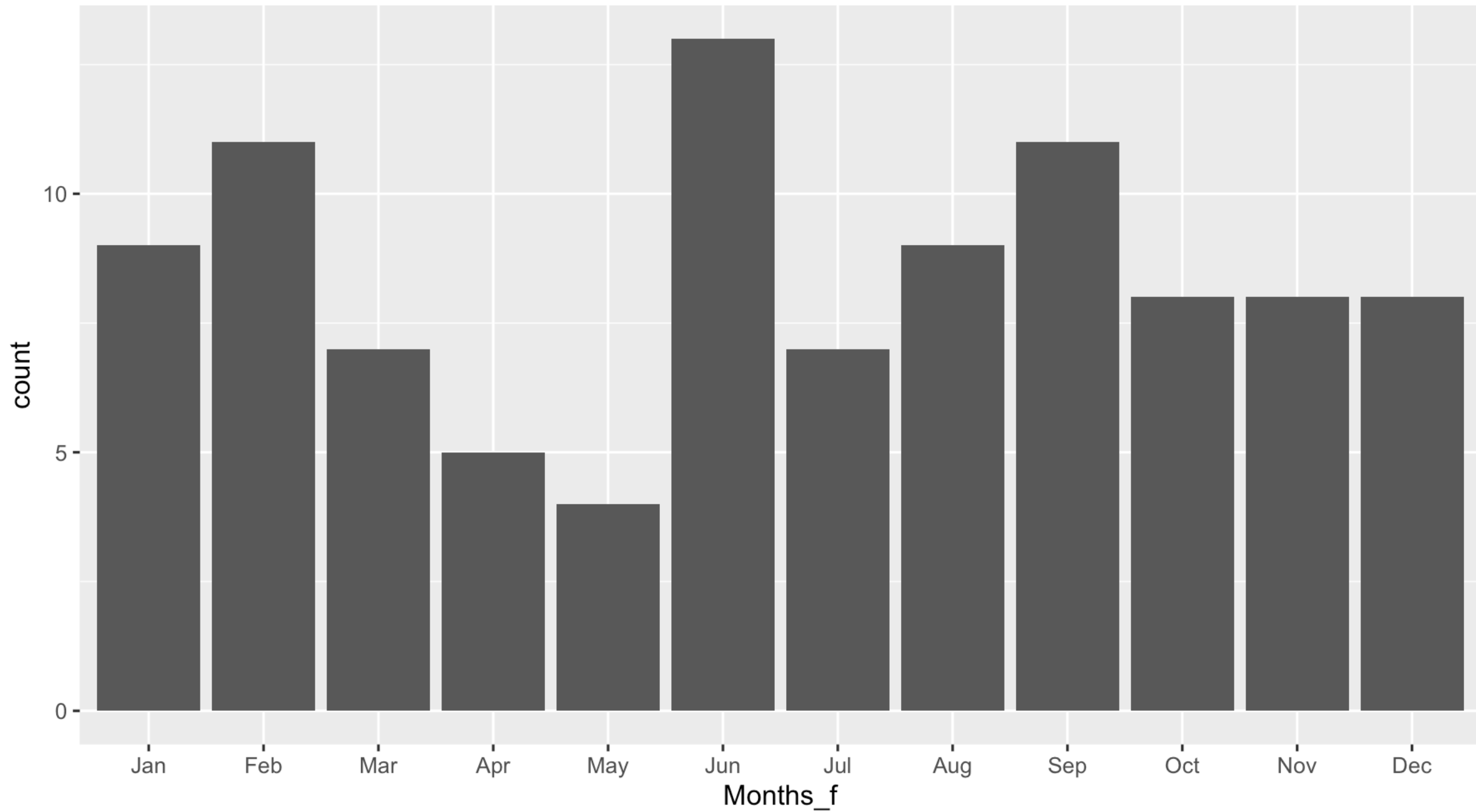
Principle 1: Order matters

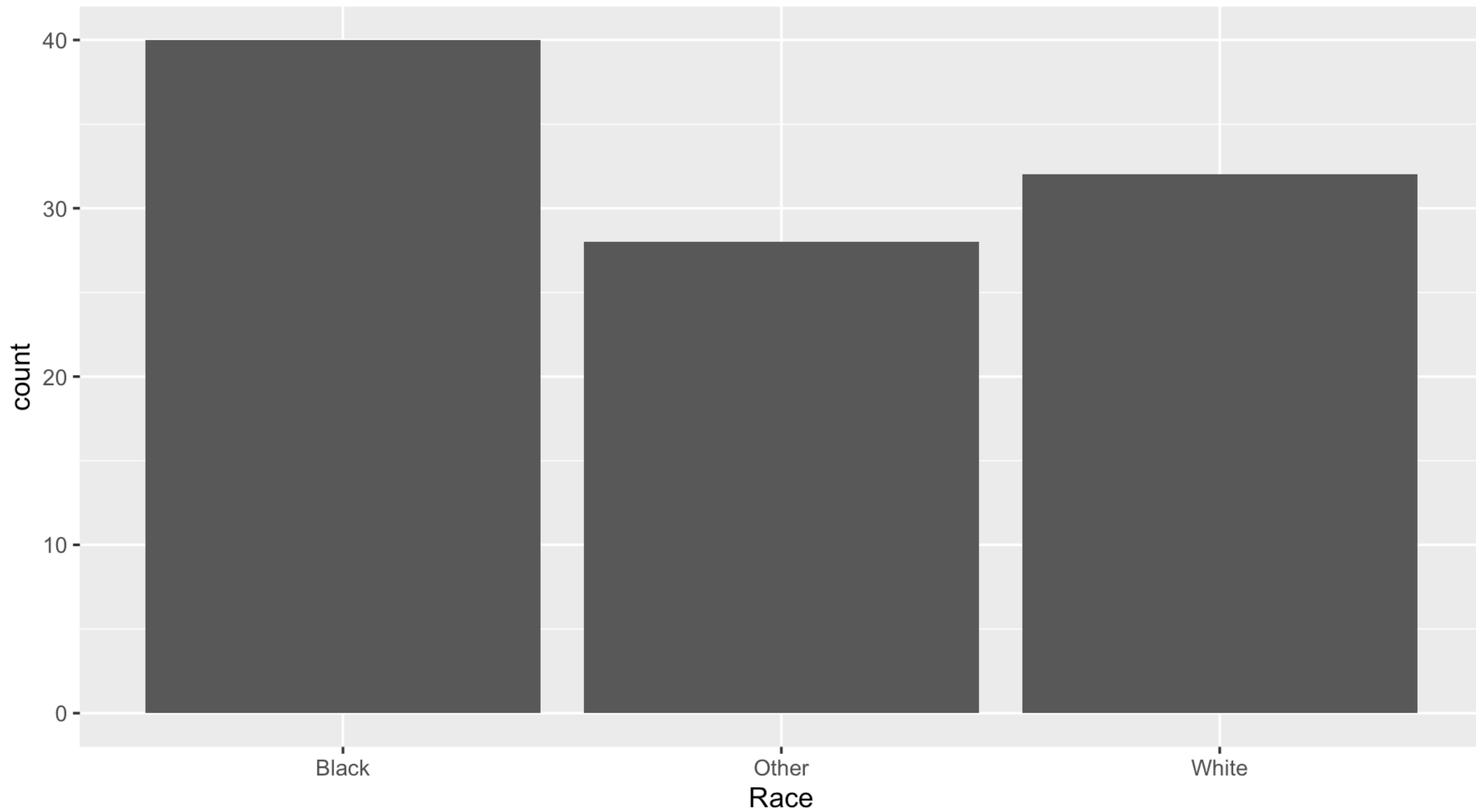


Order by meaning

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

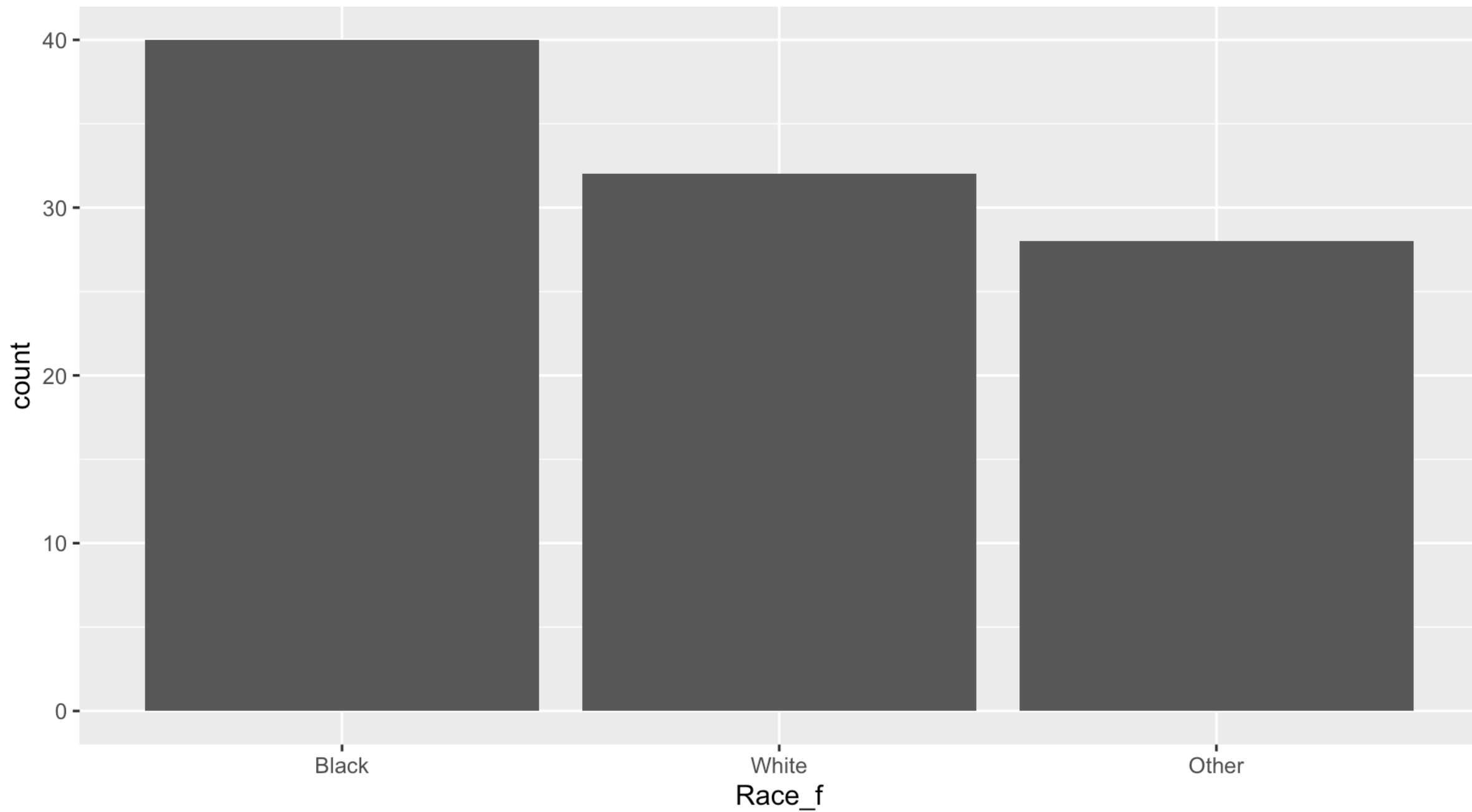
```
data <- data %>%  
  mutate(Months_f = Months %>%  
    as_factor() %>%  
    fct_relevel(month_levels))
```





Order by value (using forcats)

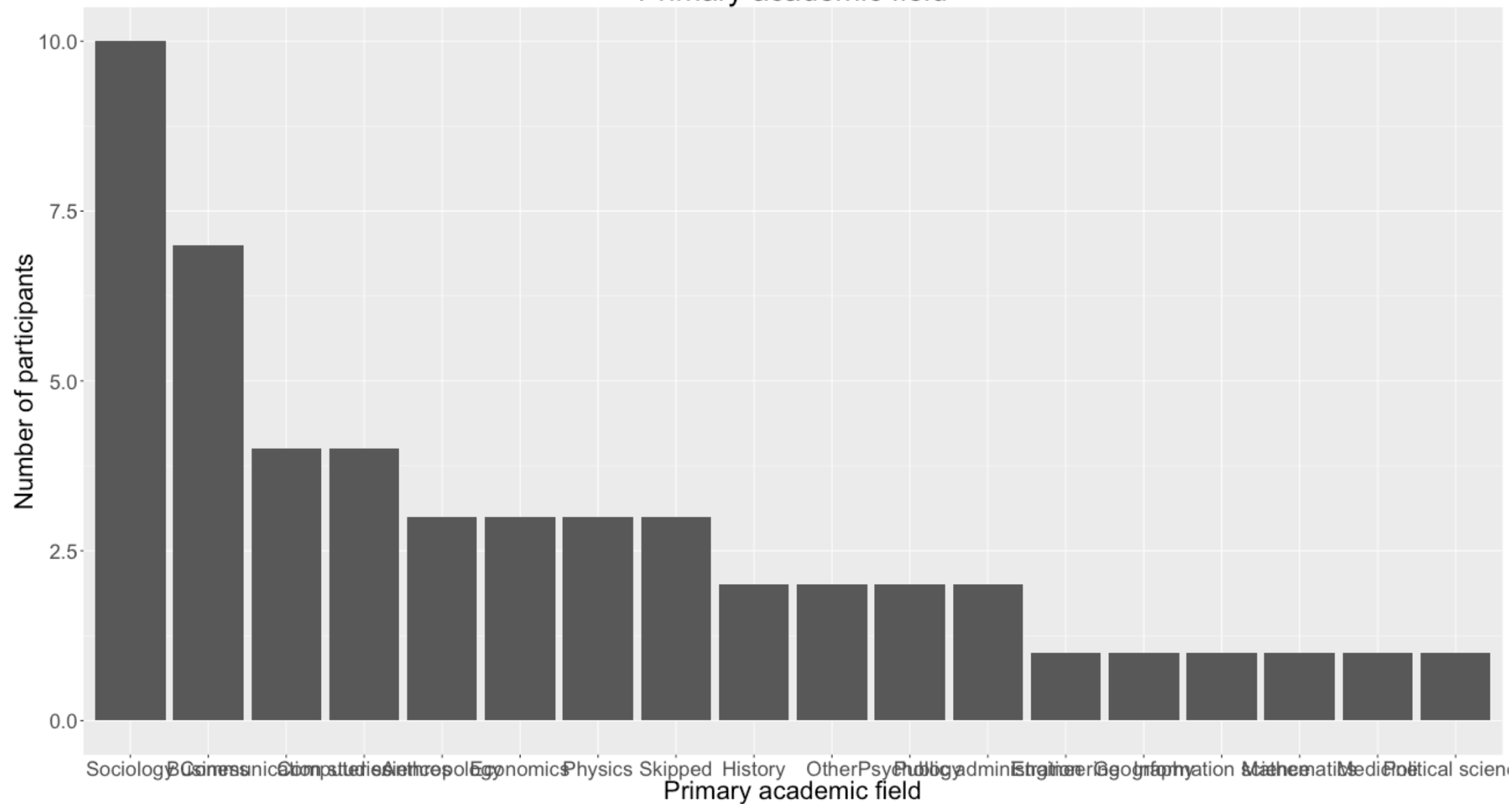
```
demo <- data %>%  
  mutate(Race_f = Race %>%  
           as_factor() %>%  
           fct_infreq())  
  
ggplot(data,  
       aes(Race %>%  
           as_factor() %>%  
           fct_infreq())) +  
  geom_bar()
```



Principle 2:

Put long categories on y-axis

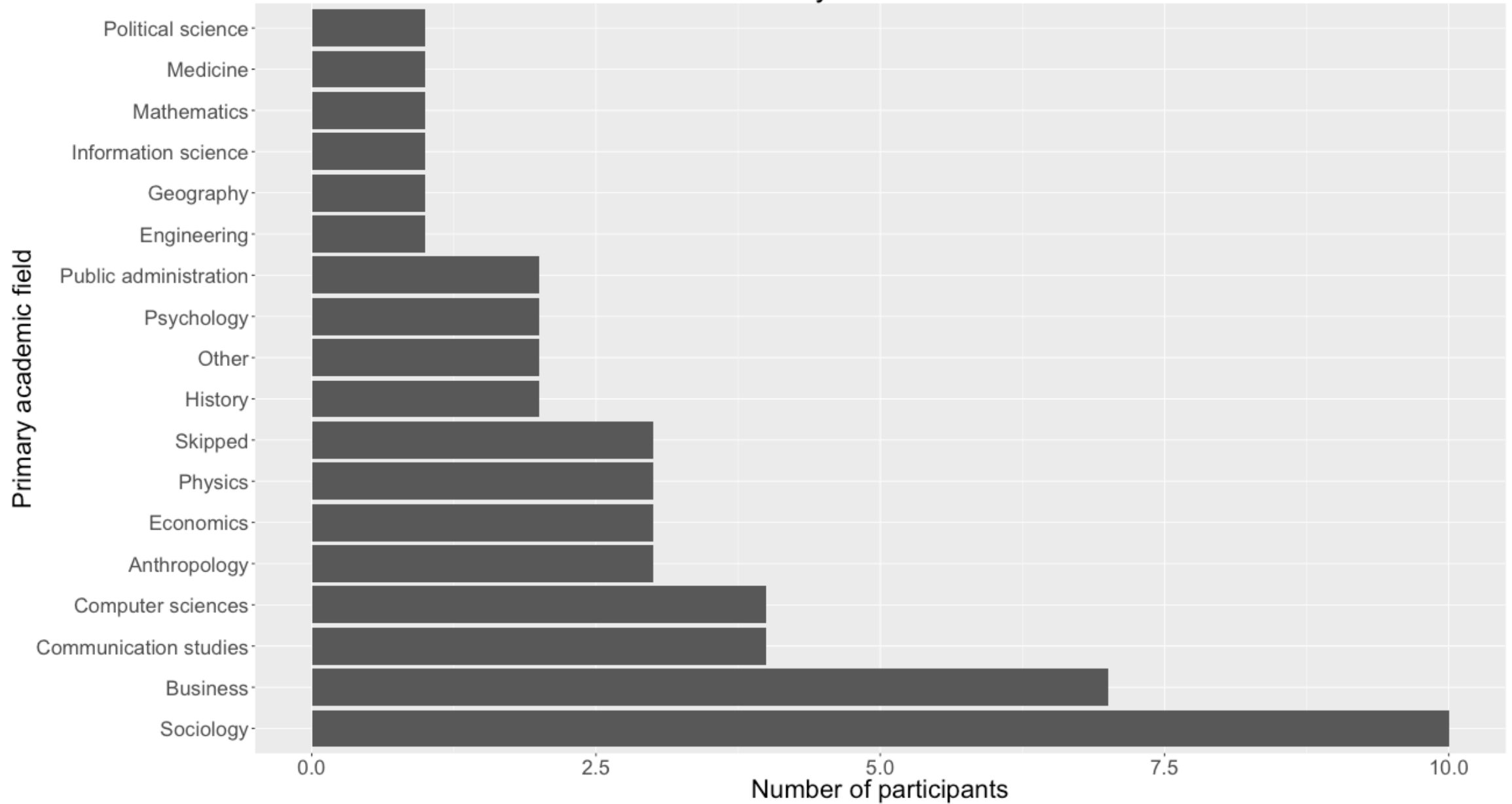
Primary academic field



Flip the axes

```
+ coord_flip()
```

Primary academic field



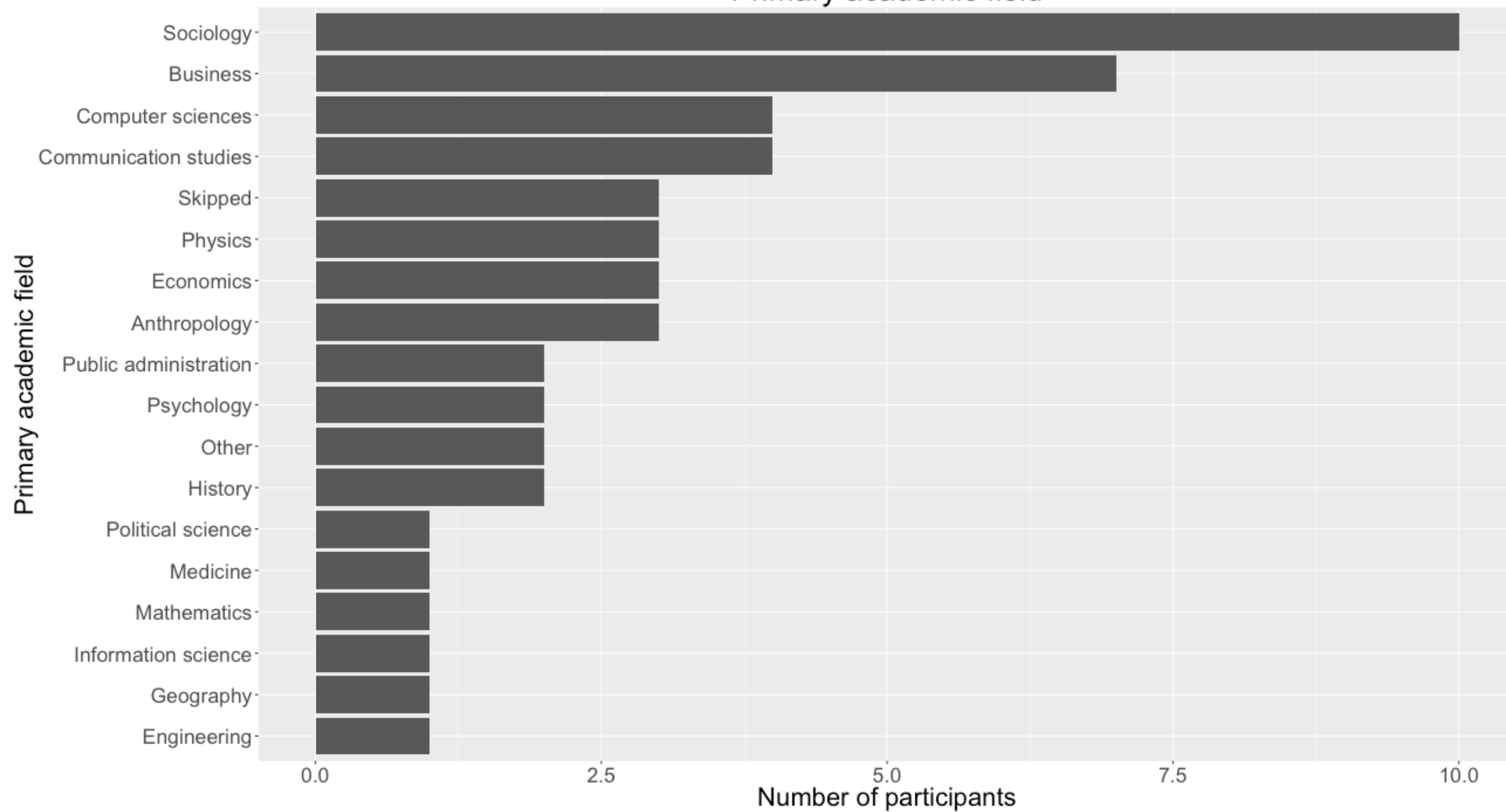
Oops!

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



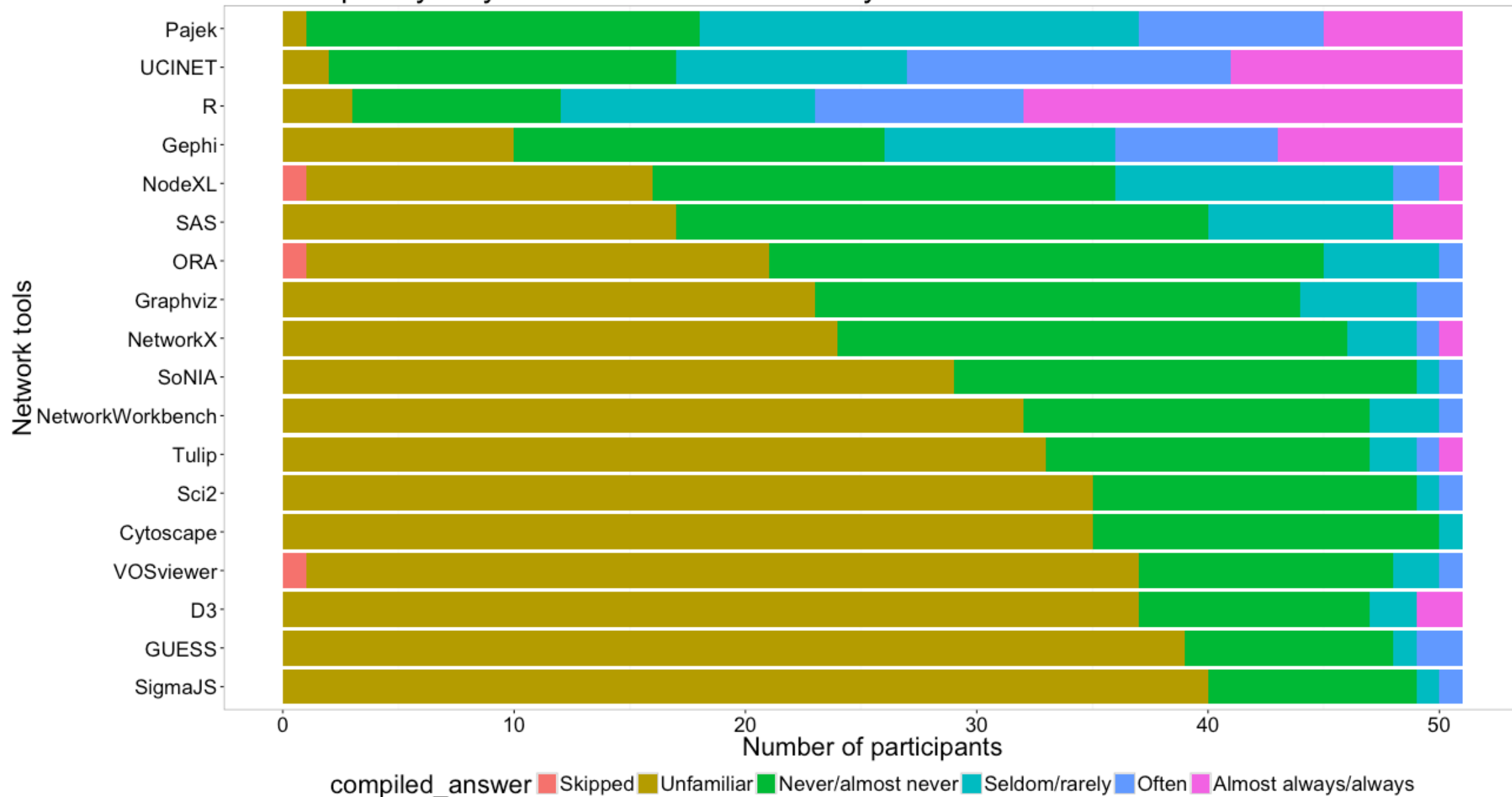
Have to reverse the
order of the levels

Primary academic field



Principle 3:
Select meaningful colors

How frequently do you use these tools for analysis?

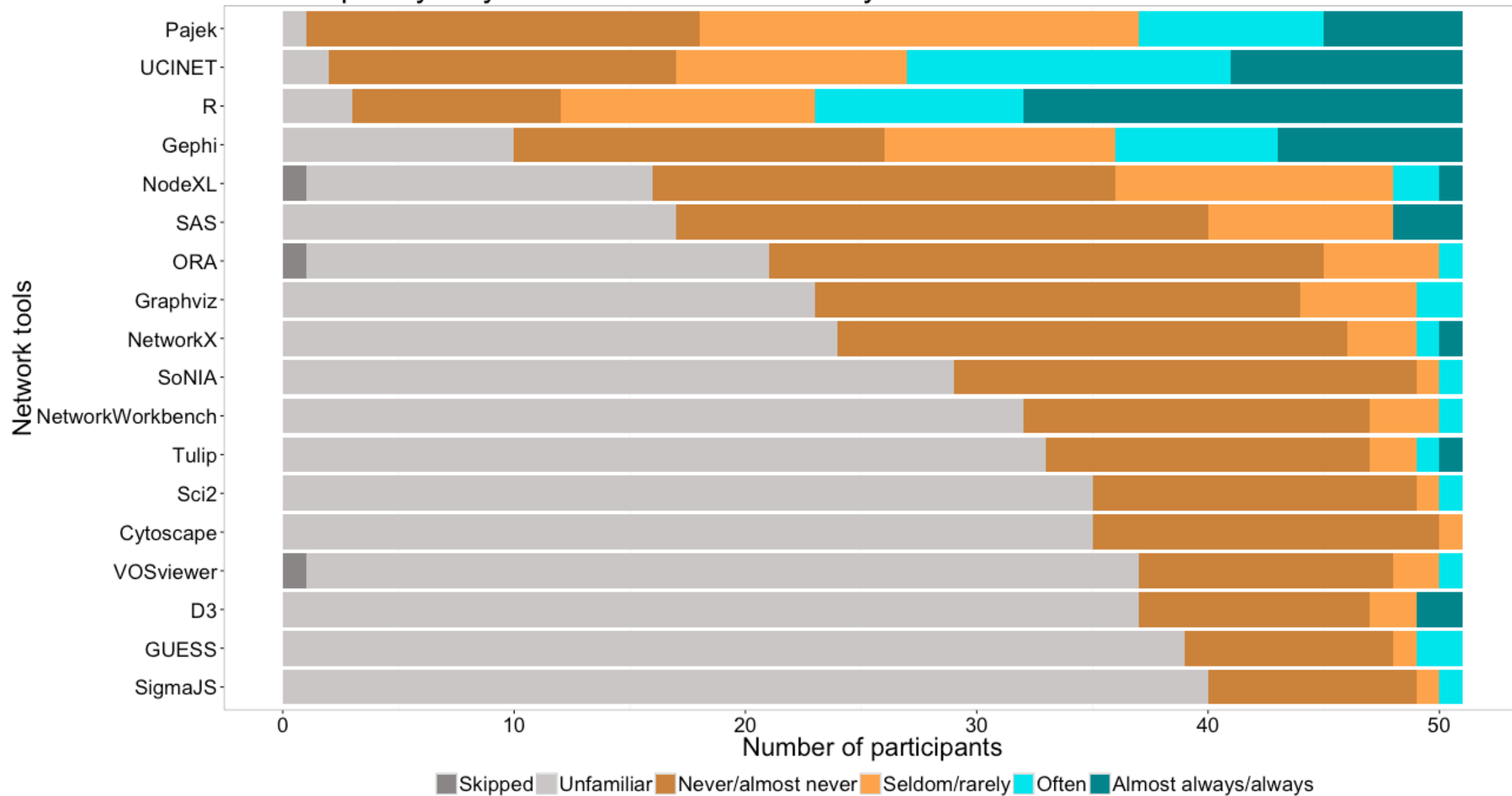


Select colors manually, or use alternate palette

```
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?



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

Panopto Figures and Posters March 4, 2016 in DVS Training Help Sign in



Search this recording

Discussion Sign in to ask a question or share a comment

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

0:03 -1:22:45 1x Speed Quality Hide

Good Posters
• A focused message
• Graphics and images that tell a story
• Use text sparingly
• Well-colored and easy to follow
1:32

Causal Inference
4:32

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.
10:32

<http://bit.ly/DVSvideos>

Questions?

angela.zoss@duke.edu