

Visualization in R using ggplot2

Angela Zoss

3/20/19

<https://github.com/amzoss/ggplot2-DF19>

Set up environment

- R
- RStudio
- tidyverse package

Try right now:

Open RStudio

Try running “library(tidyverse)”

Tell me about any errors

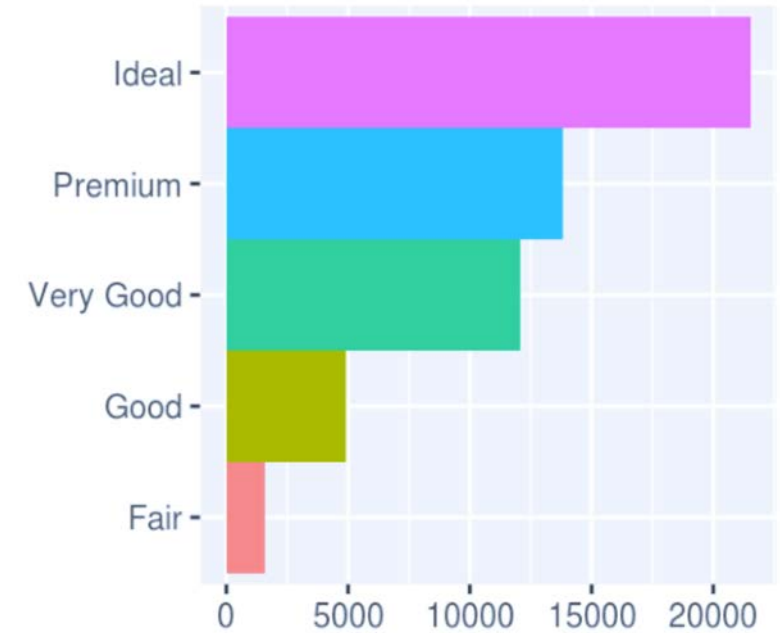
Don't have it installed?

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

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

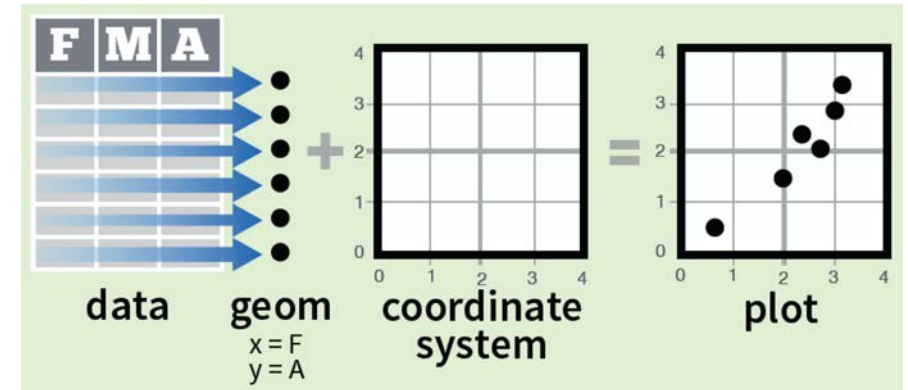
“Why I use ggplot2”, David Robinson

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

ggplot2: Elements

Basic elements in any ggplot2 visualization

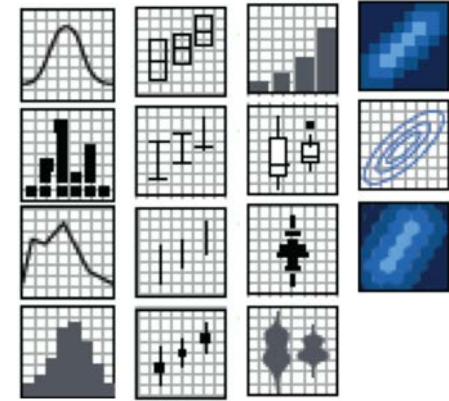
- **data**
- **aesthetics**
(variable mappings)
- **geom**
(chart type or shape)
- **coordinate system**
(the arrangement of the marks;
most geoms use default, cartesian)



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

Types of geoms

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



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

Note: some geoms also include data summary functions.
e.g., the “bar” geom will count data points in each category.

ggplot2: Basic syntax

Required functions:

Main function
ggplot()

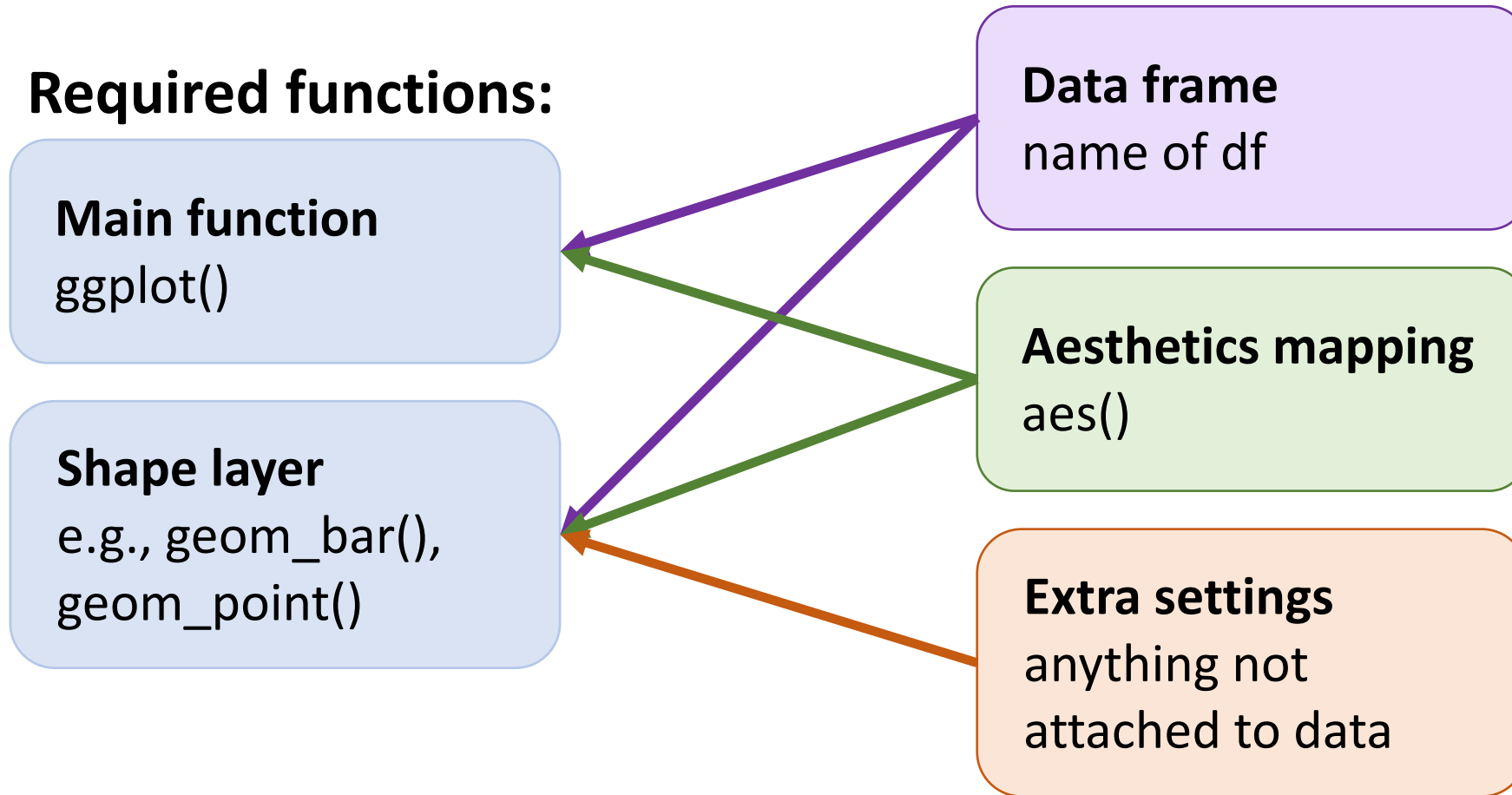
Shape layer
e.g., geom_bar(),
geom_point()

Required elements:

Data frame
name of df

Aesthetics mapping
aes()

Extra settings
anything not
attached to data



Template for a simple plot

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

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

Template for a simple plot

**Main
function** `ggplot(data = data frame)` +

**Shape
layer** `geom_... (mapping = aes(variable mappings),
non-variable adjustments)`

Step-by-step

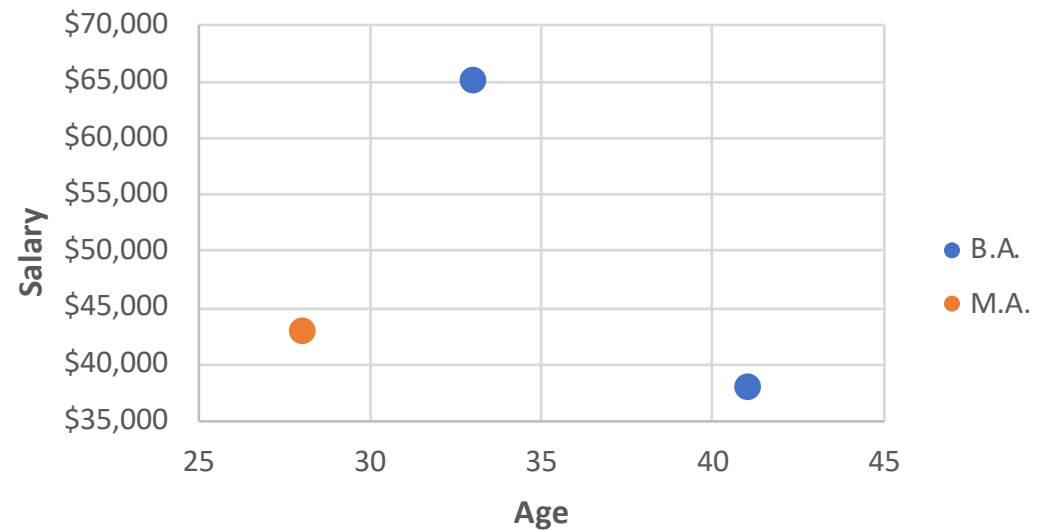
1. Set the data
2. Choose a shape layer
3. Map variables to aesthetics
4. Add non-variable adjustments

1. Set the data

“employees”

Name	Age	Salary	Highest Degree
Jane Smith	33	\$65,000	B.A.
Abby Jones	28	\$43,000	M.A.
Bridget Carden	41	\$38,000	B.A.

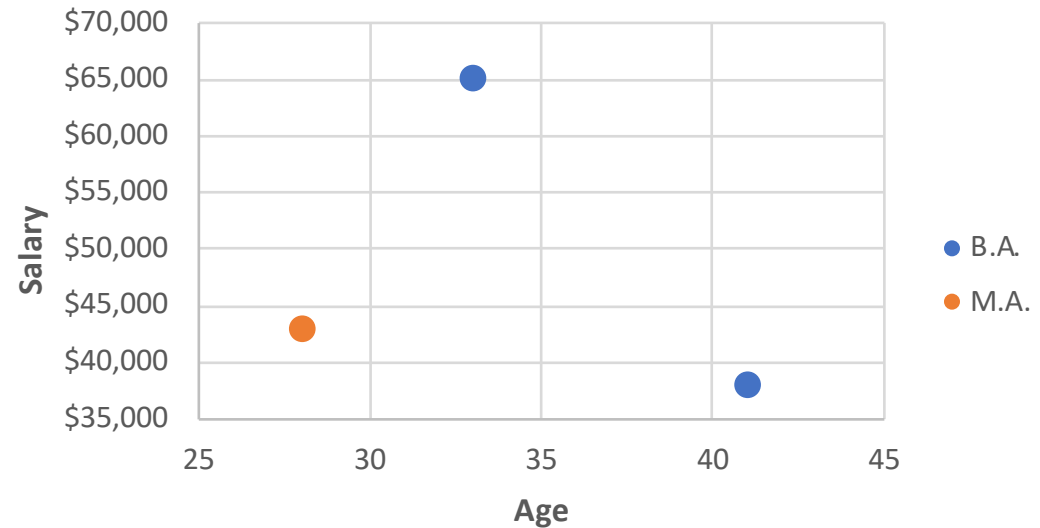
```
ggplot(employees)
```



2. Choose a shape layer

Name	Age	Salary	Highest Degree
Jane Smith	33	\$65,000	B.A.
Abby Jones	28	\$43,000	M.A.
Bridget Carden	41	\$38,000	B.A.

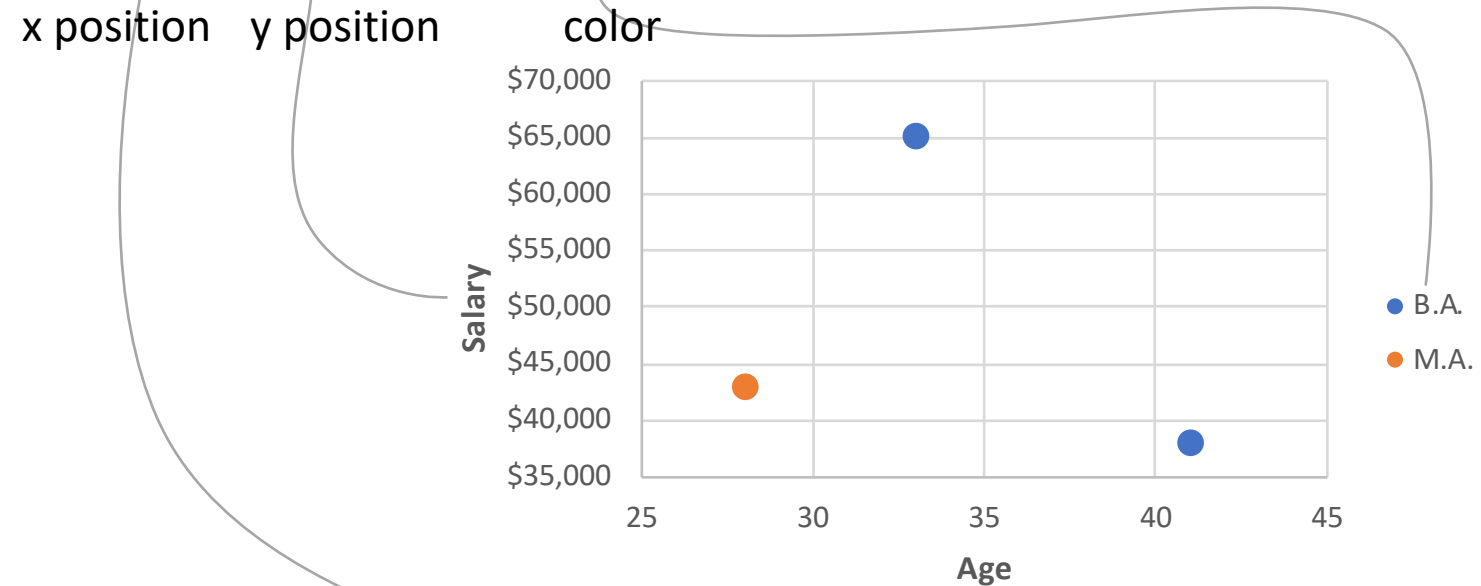
```
ggplot(employees) +  
  geom_point()
```



3. Map variables to aesthetics

Name	Age	Salary	Highest Degree
Jane Smith	33	\$65,000	B.A.
Abby Jones	28	\$43,000	M.A.
Bridget Carden	41	\$38,000	B.A.

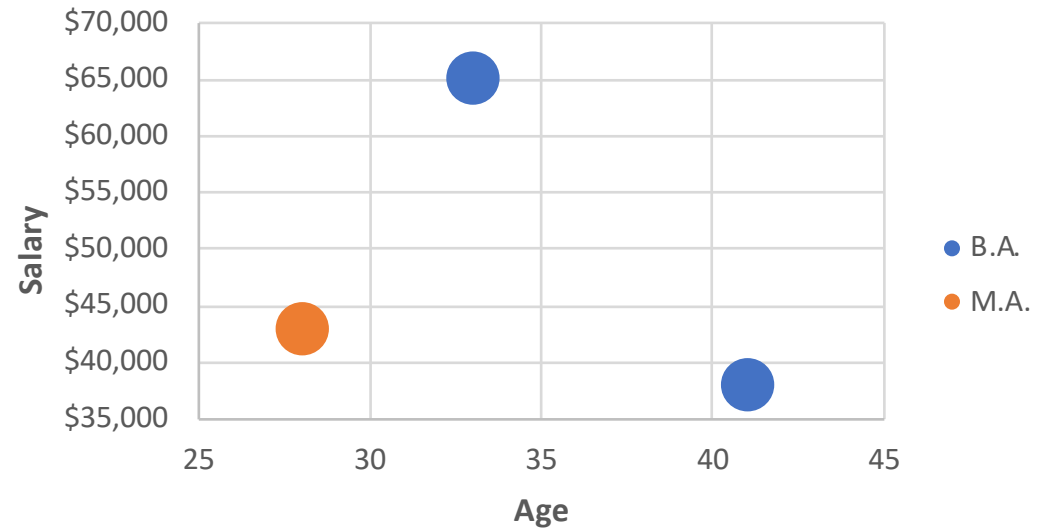
```
ggplot(employees) +  
  geom_point(  
    aes(x=age,  
        y=salary,  
        color=degree))
```



4. Add non-variable adjustments

Name	Age	Salary	Highest Degree
Jane Smith	33	\$65,000	B.A.
Abby Jones	28	\$43,000	M.A.
Bridget Carden	41	\$38,000	B.A.

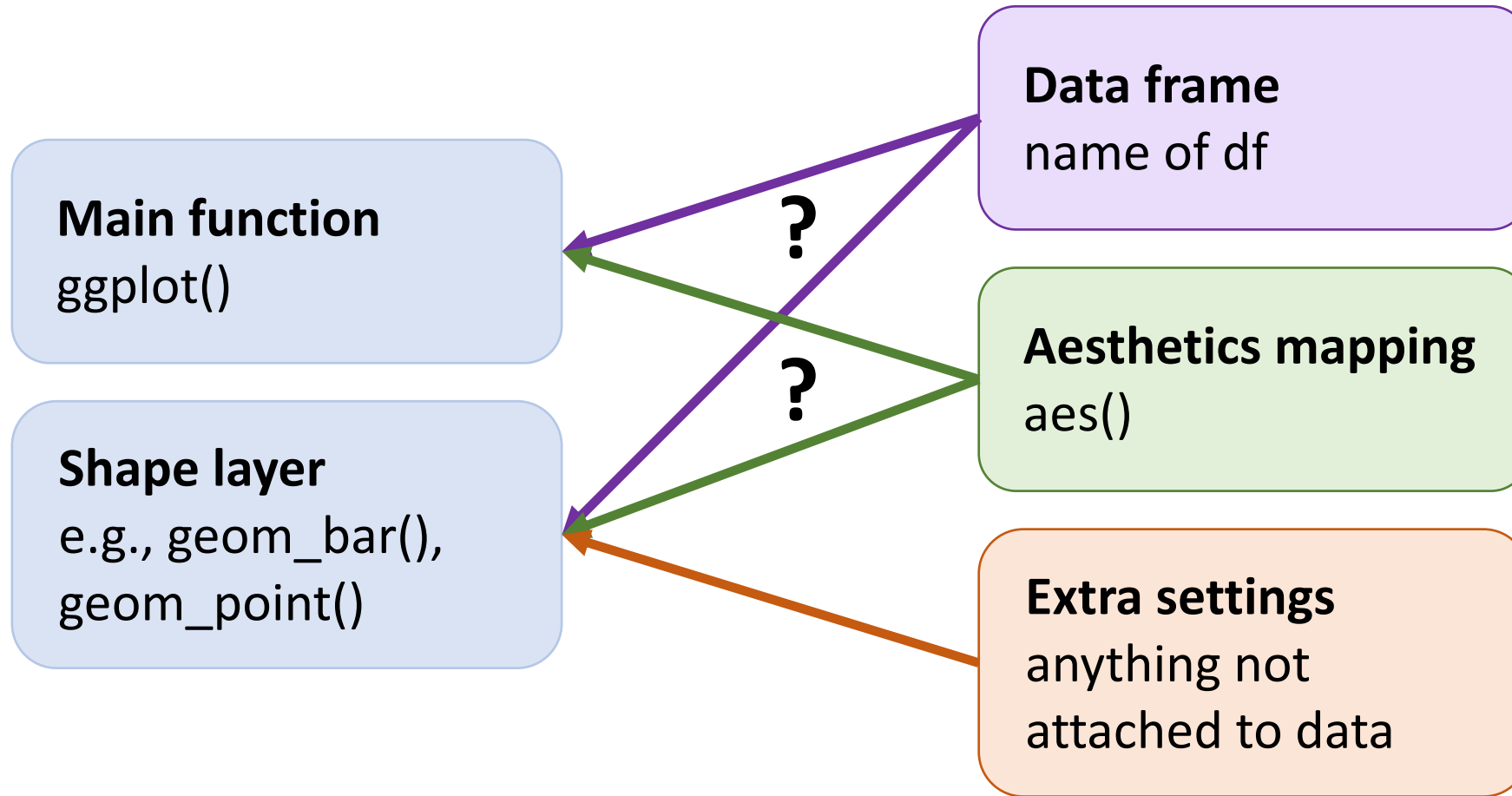
```
ggplot(employees) +  
  geom_point(  
    aes(x=age,  
        y=salary,  
        color=degree),  
    size=10)
```



Template for a simple plot

**Main
function** `ggplot(data = data frame)` +

**Shape
layer** `geom_... (mapping = aes(variable mappings),
non-variable adjustments)`



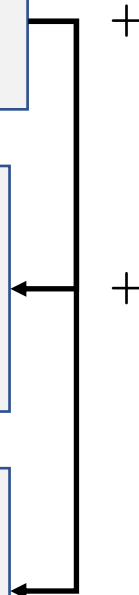
Inheritance

data and aesthetics will
carry through
from top to bottom

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

```
geom_... ( aes(add'l variable mappings),  
           non-variable adjustments )
```

```
geom_... ( aes(add'l variable mappings),  
           non-variable adjustments )
```



Inheritance

data and aesthetics will
carry through
from top to bottom

**Main
function**

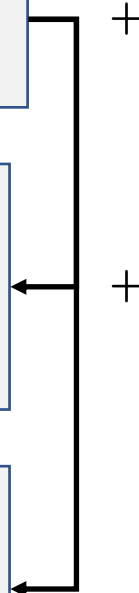
```
ggplot( data = data frame,  
        mapping = aes(variable mappings) )
```

**Shape
layer**

```
geom_... ( aes(add'l variable mappings),  
            non-variable adjustments )
```

**Shape
layer**

```
geom_... ( aes(add'l variable mappings),  
            non-variable adjustments )
```



General inheritance strategy

**Main
function**

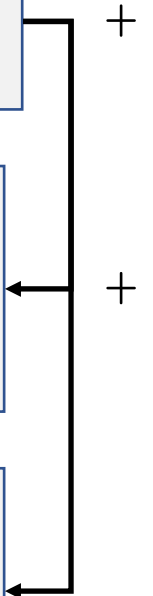
```
ggplot( main data frame,  
        aes(x, y) )
```

**Shape
layer**

```
geom_... ( aes(add'l variable mappings),  
           non-variable adjustments )
```

**Shape
layer**

```
geom_... ( aes(add'l variable mappings),  
           non-variable adjustments )
```



Working in RStudio

Set up environment

- R
- RStudio
- tidyverse package

Try right now:

Open RStudio

Try running “library(tidyverse)”

Tell me about any errors

Don't have it installed?

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

Get workshop files

URL: <https://github.com/amzoss/ggplot2-DF19>

With Git installed

In RStudio:

- Project → New project
- Version Control
- Git
 - Paste in GitHub URL
 - Project directory name:
ggplot2-DF19
 - Subdirectory: you choose
- Create Project

Without Git installed

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

In RStudio:

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

Using RStudio

- Projects
- Rmarkdown
- Cheat sheets

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

Why Rmarkdown?

- 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.

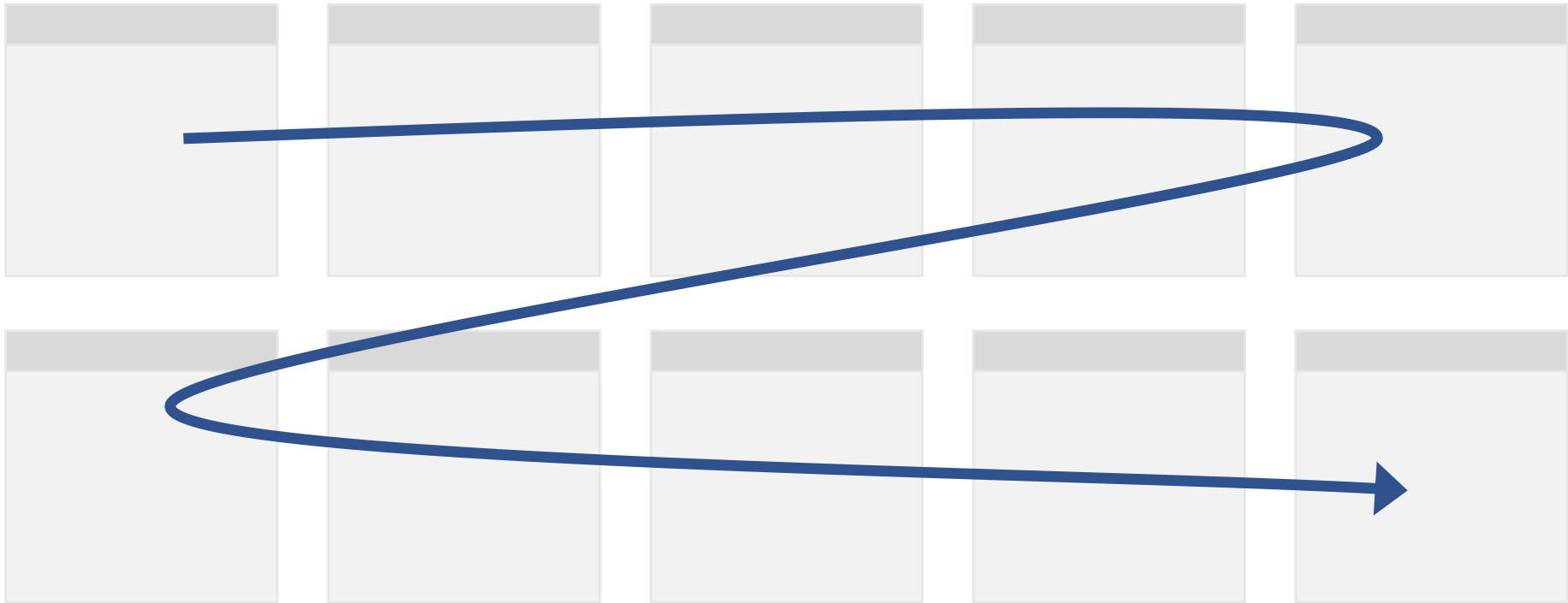
Game of Thrones character ratings

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

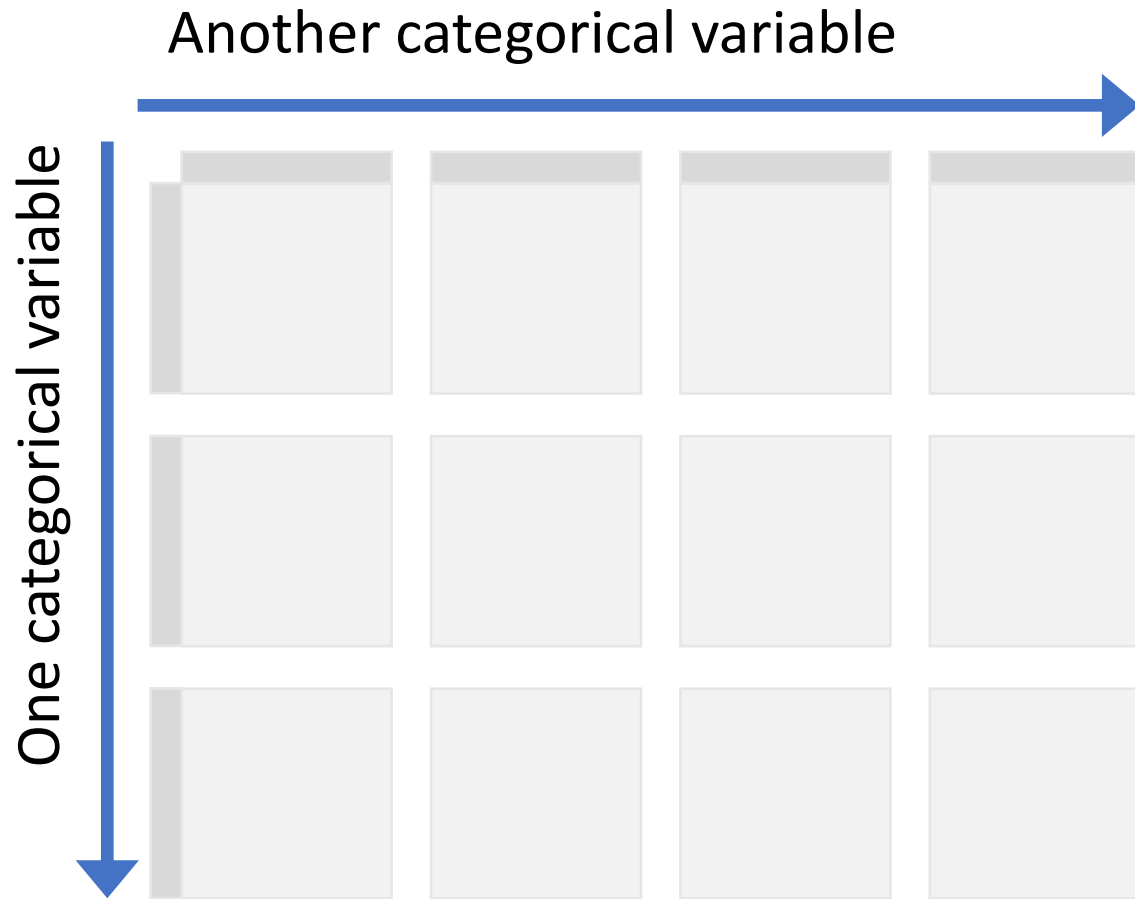
Creating repeated charts

Facet_wrap

```
+ facet_wrap(~variable)
```



Facet_grid

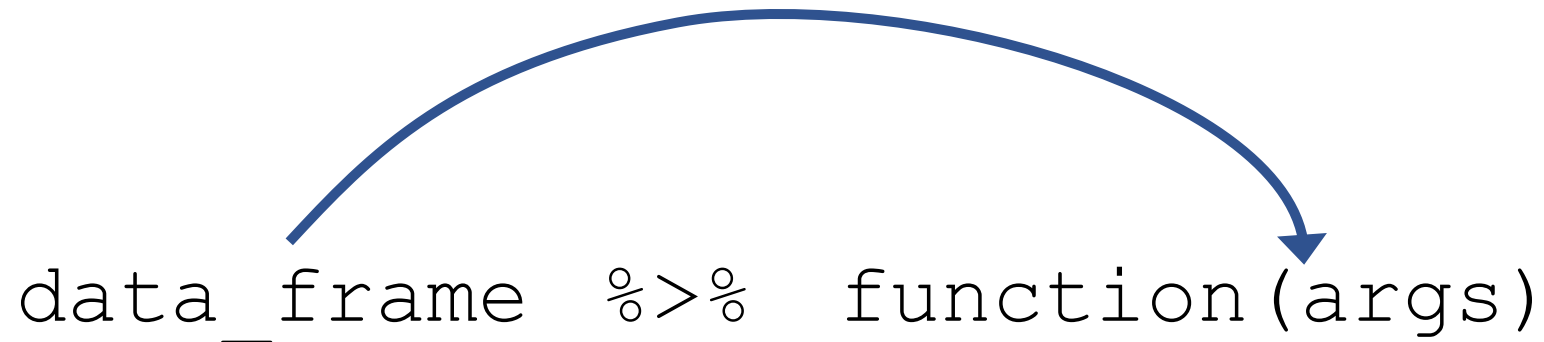


```
+ facet_grid(yvar~xvar)
+ facet_grid(.~xvar)
+ facet_grid(yvar~.)
```

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")
```

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

select

Select a subset of columns

```
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>

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

- Specify “levels” for a categorical variable
- Specify the order of those levels
- Specify whether the factor is “ordered”

```
month_levels <- c( "Jan", "Feb",  
  "Mar", "Apr", "May", "Jun", "Jul",  
  "Aug", "Sep", "Oct", "Nov", "Dec" )  
  
y1 <- factor(x1,  
             levels = month_levels)
```

forcats package: helpful functions

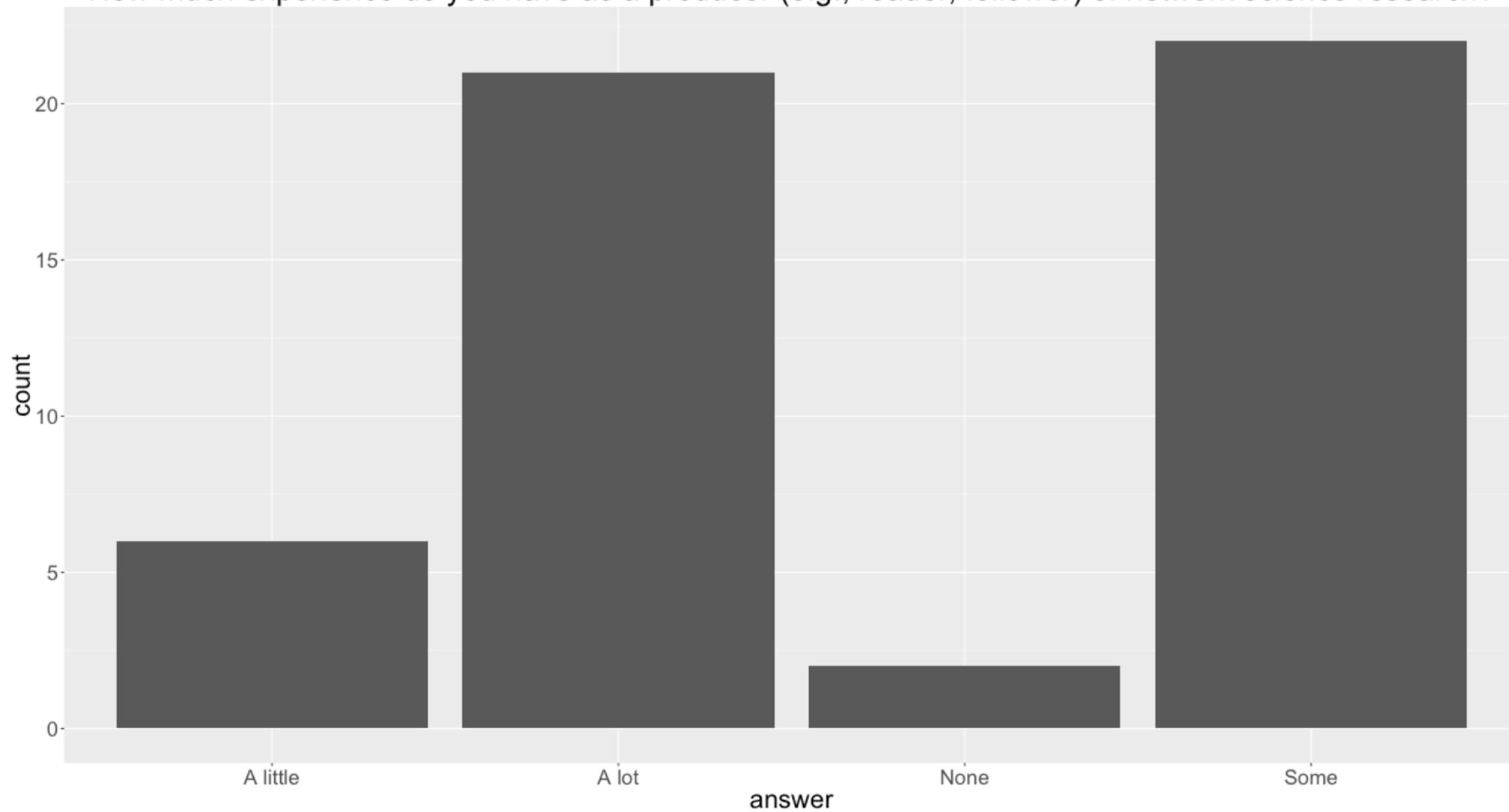
- `as_factor(char_var)`: convert a character variable to a factor
- `fct_infreq(factor)`: sort a factor by frequency, descending
- `fct_reorder(factor, num_var)`: sort a factor by a second, numerical variable

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

Principles for Effective Visualizations

Principle 1: Order matters

How much experience do you have as a producer (e.g., reader, follower) of network science research?



Order by meaning

1. Treat variable
as factor



```
data$answer <- fct_relevel(as_factor(data$answer),  
  c("None", "A little",  
    "Some", "A lot"))
```

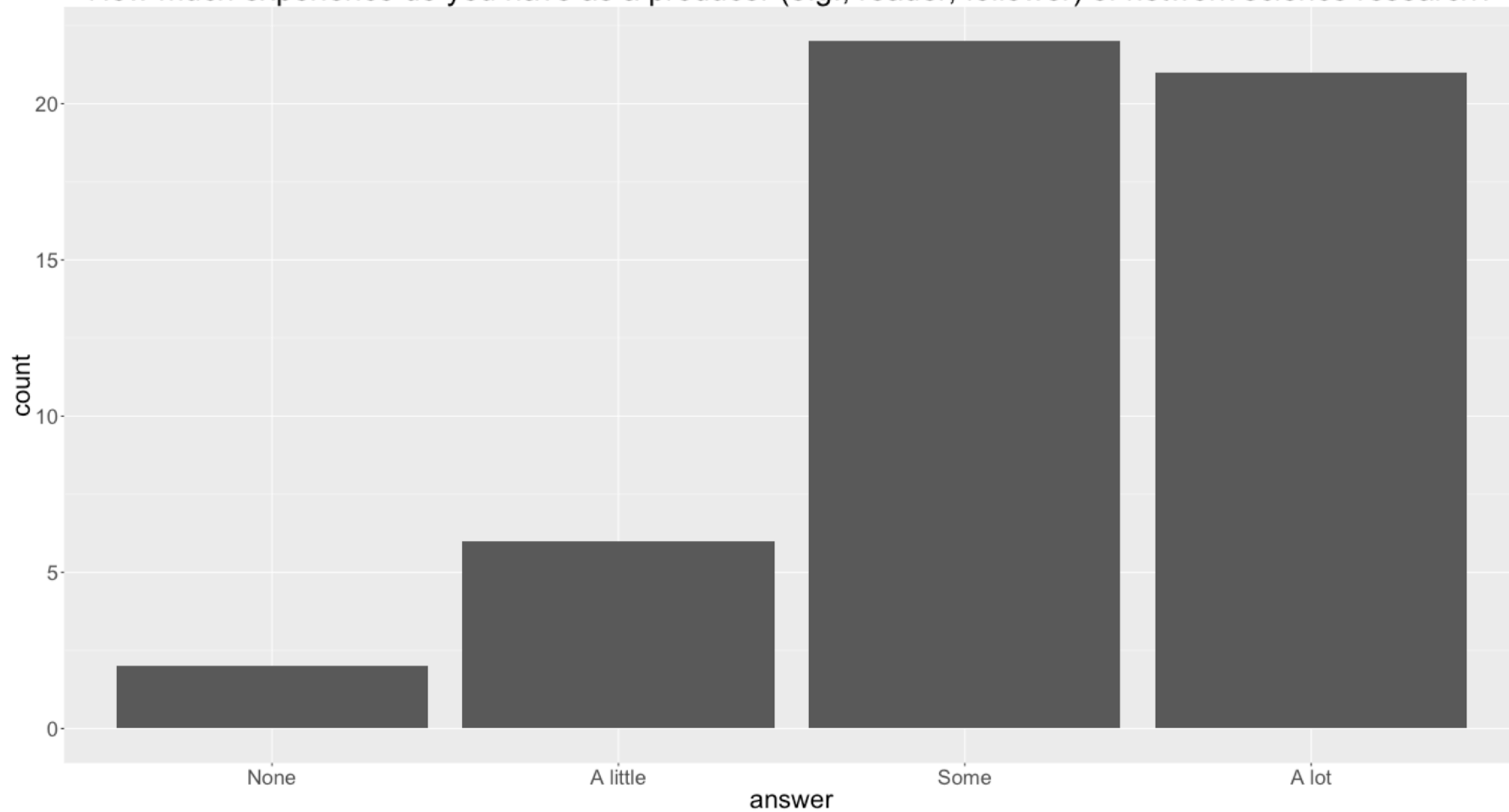


2. Reorder
levels manually

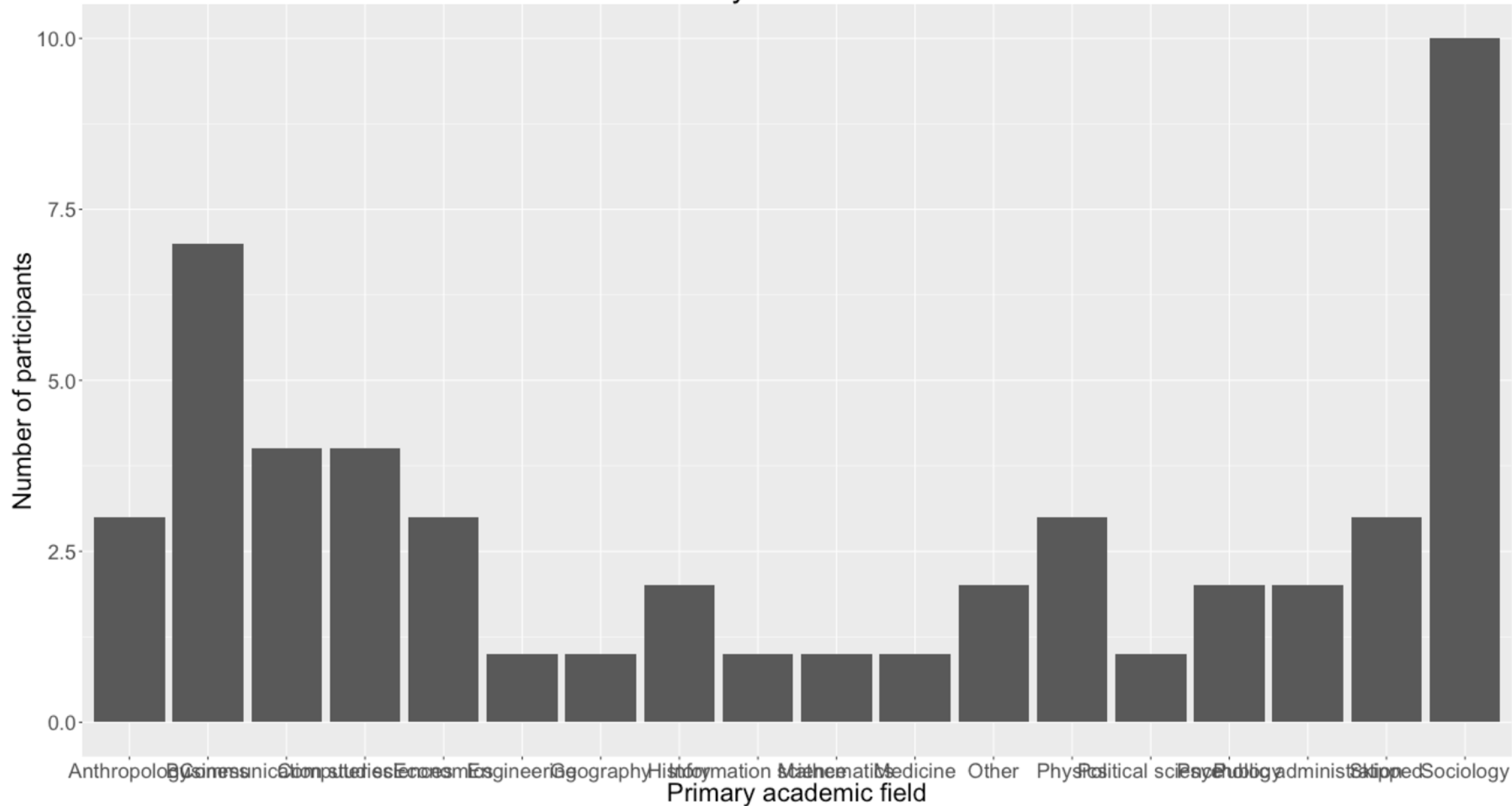


3. Specify levels
in order

How much experience do you have as a producer (e.g., reader, follower) of network science research?



Primary academic field



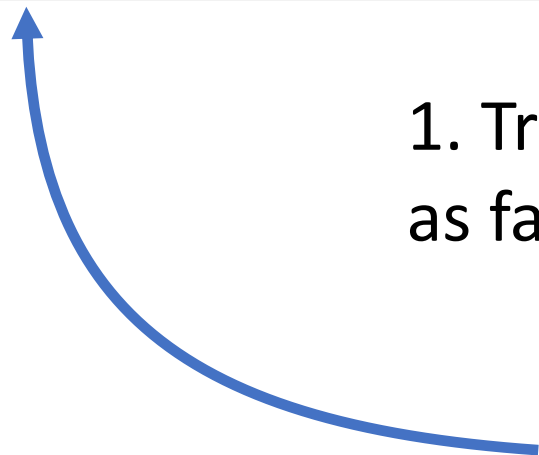
Order by value (using forcats)

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

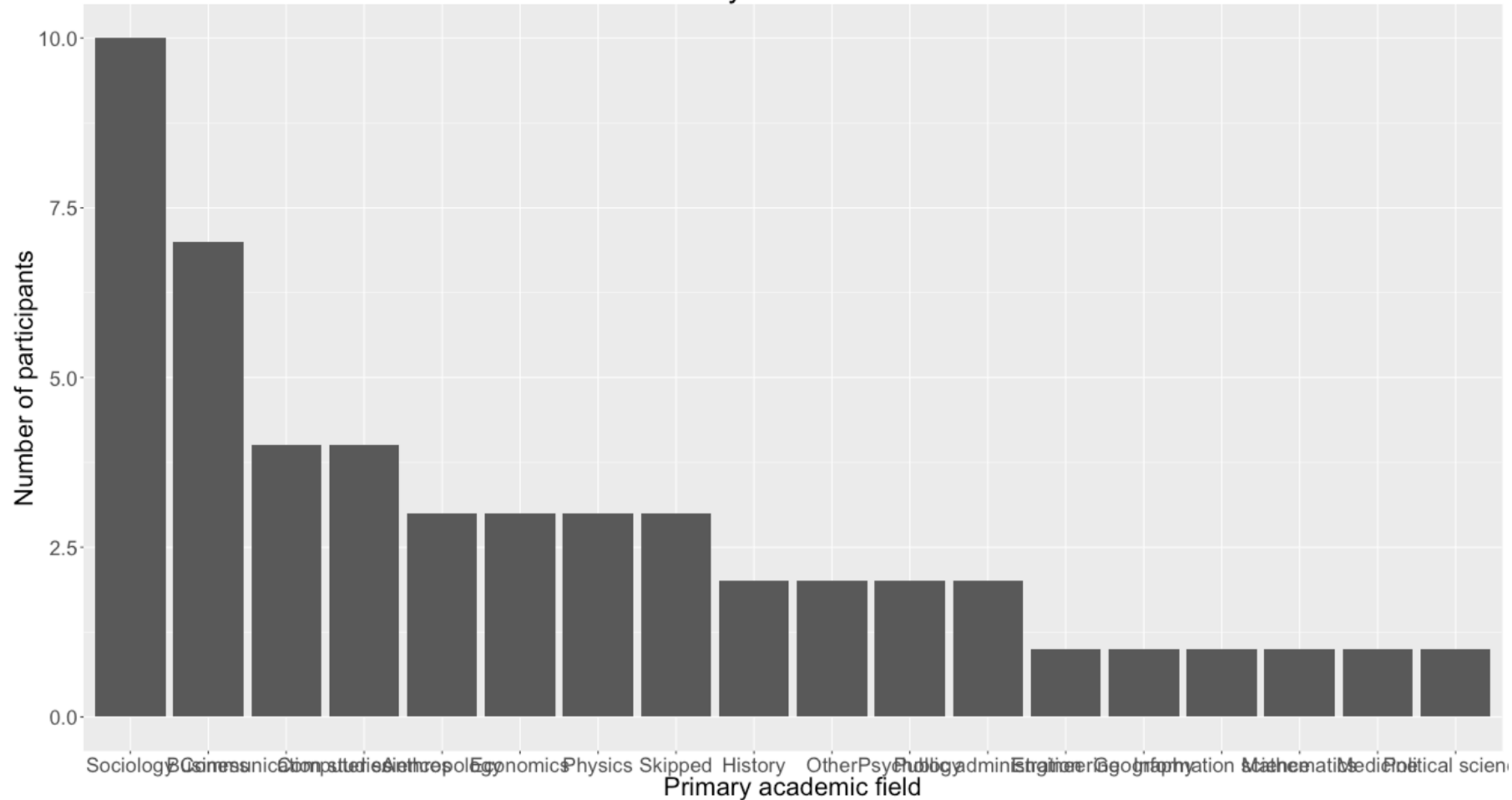
1. Treat variable
as factor



2. Order factor by
inverse frequency



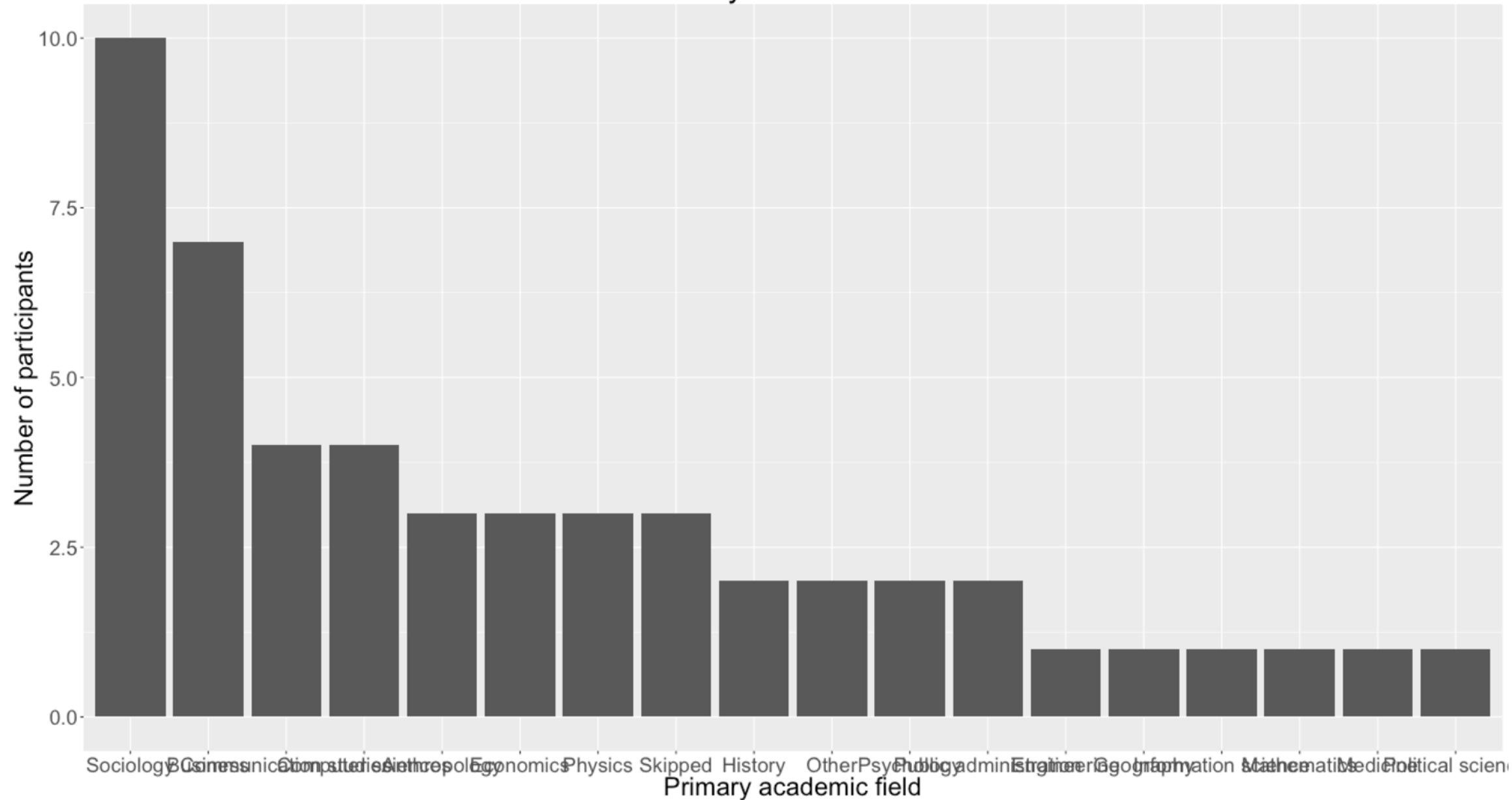
Primary academic field



Principle 2:

Put long categories on y-axis

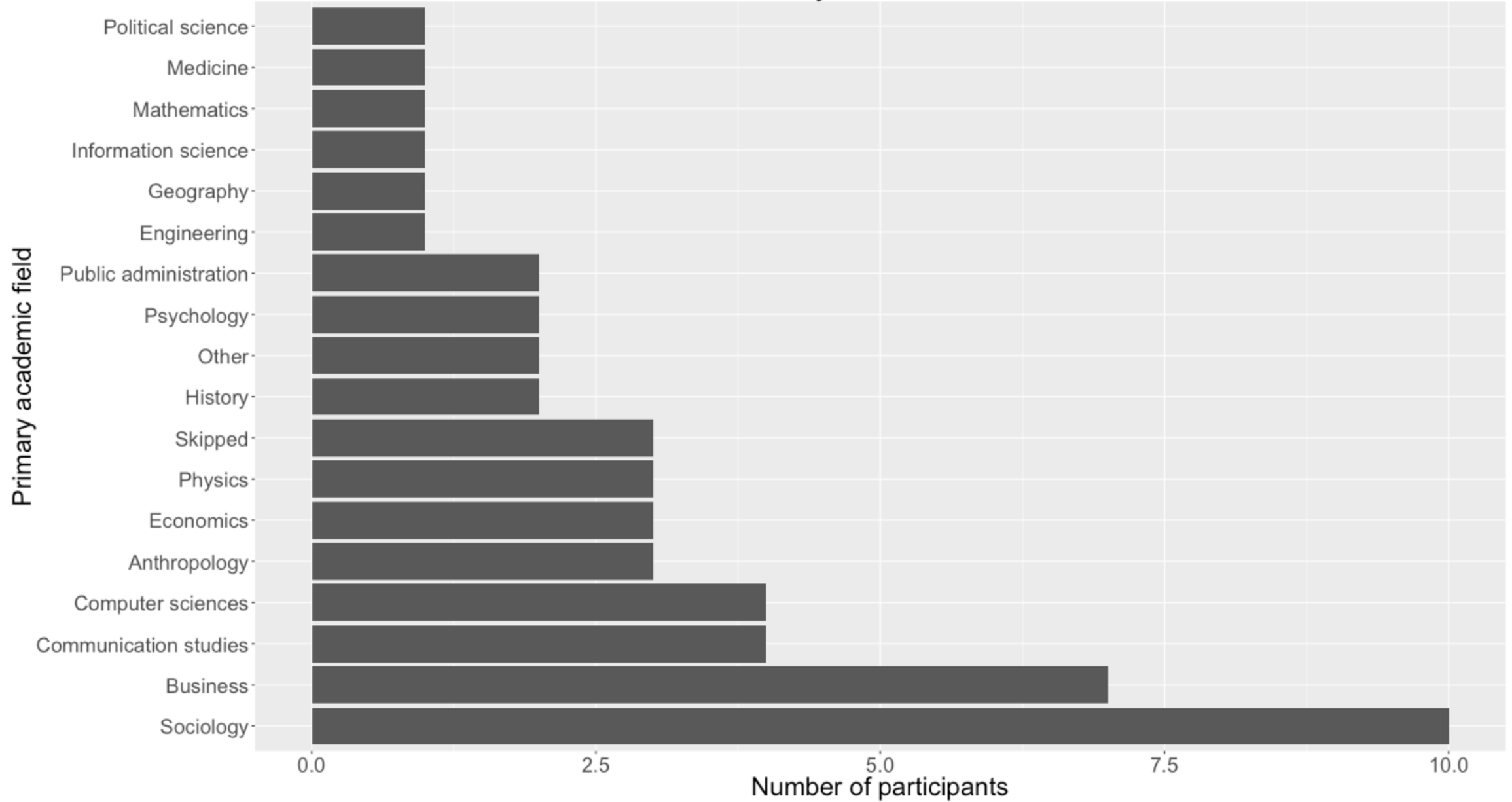
Primary academic field



Flip the axes

```
+ coord_flip()
```

Primary academic field



Oops!

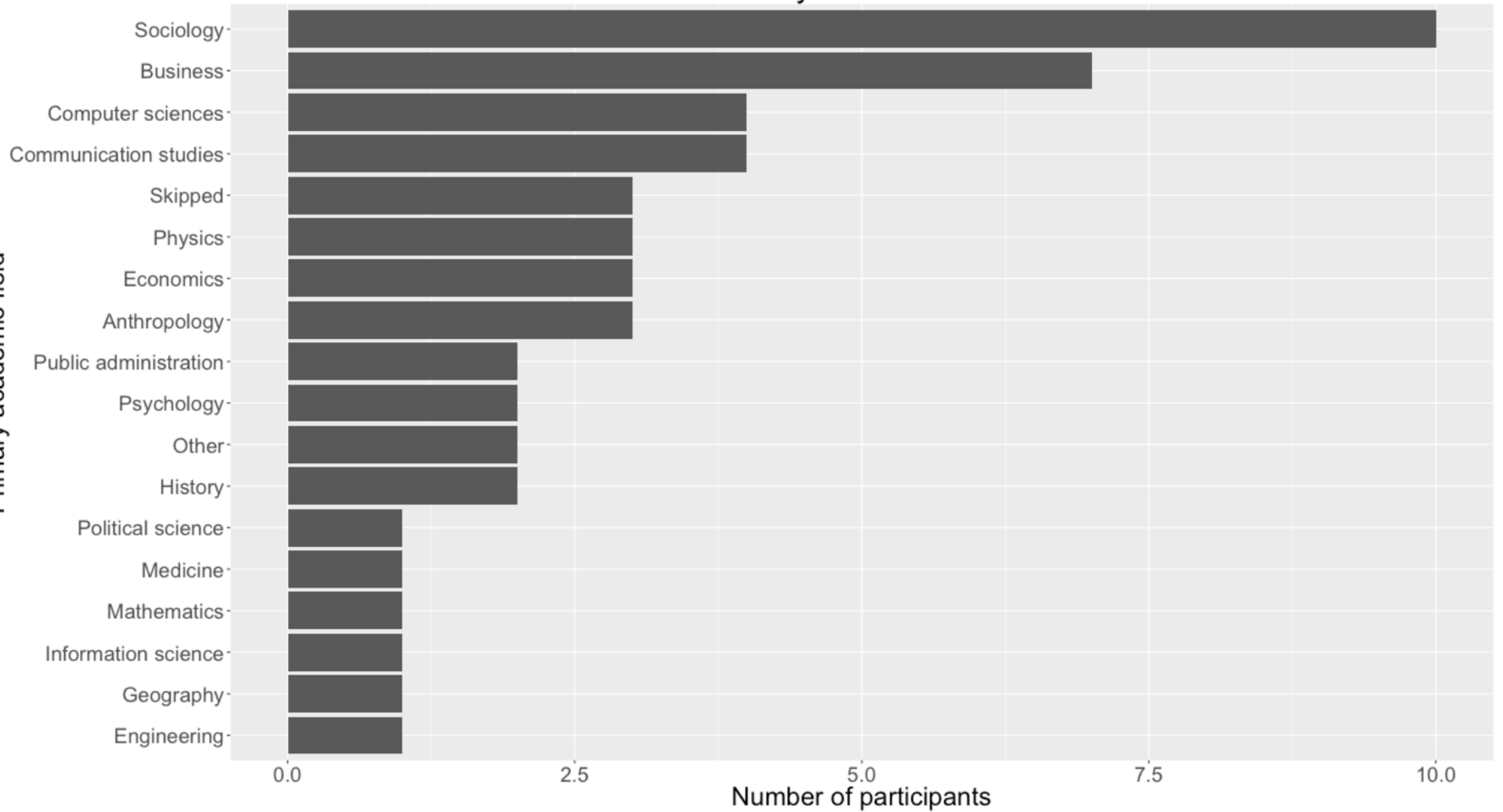
```
data$academic_field <-  
  fct_rev(fct_infreq(  
    as_factor(data$academic_field)))
```



Have to reverse the
order of the levels

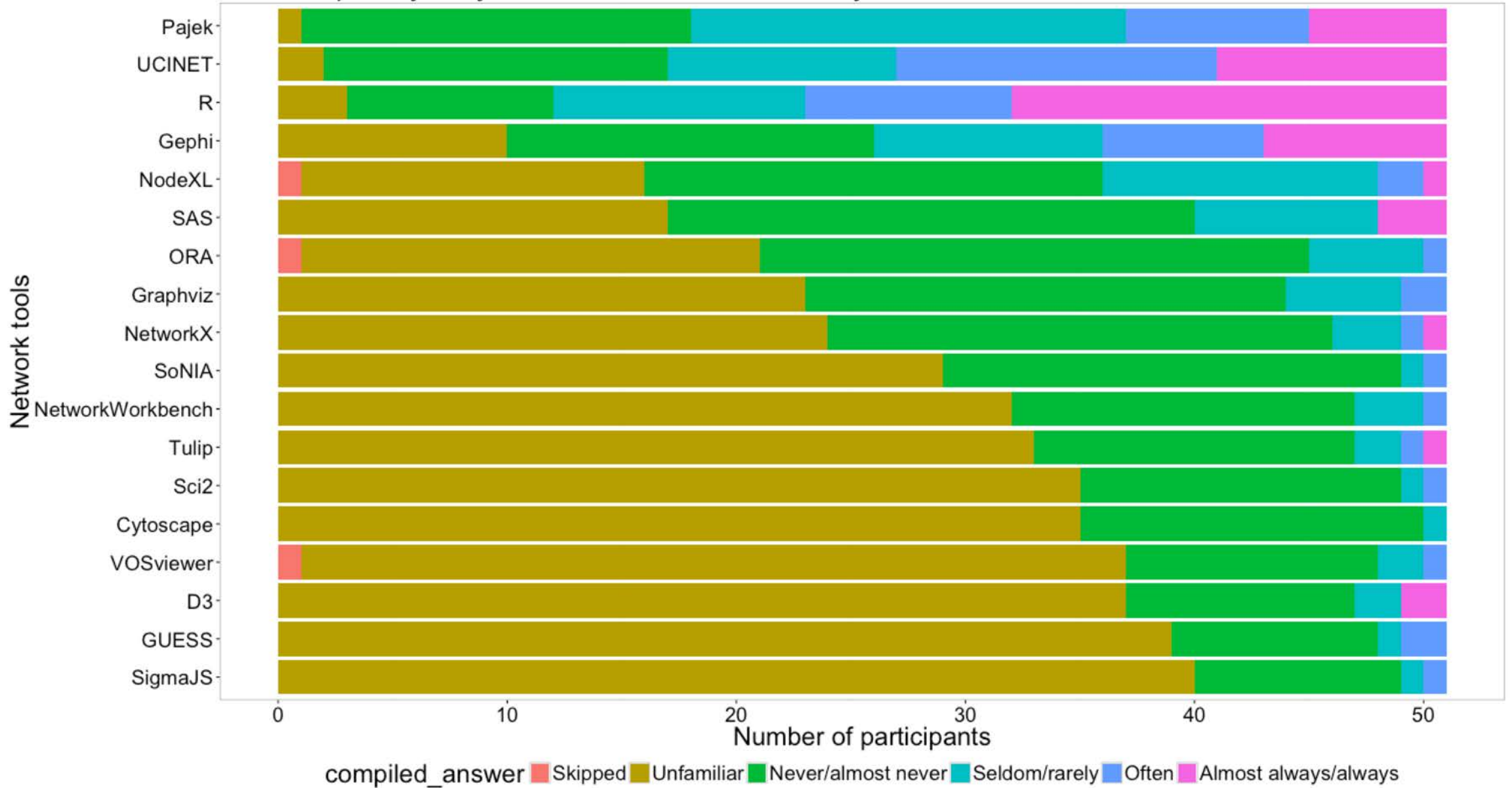
Primary academic field

Primary academic field



Principle 3:
Select meaningful colors

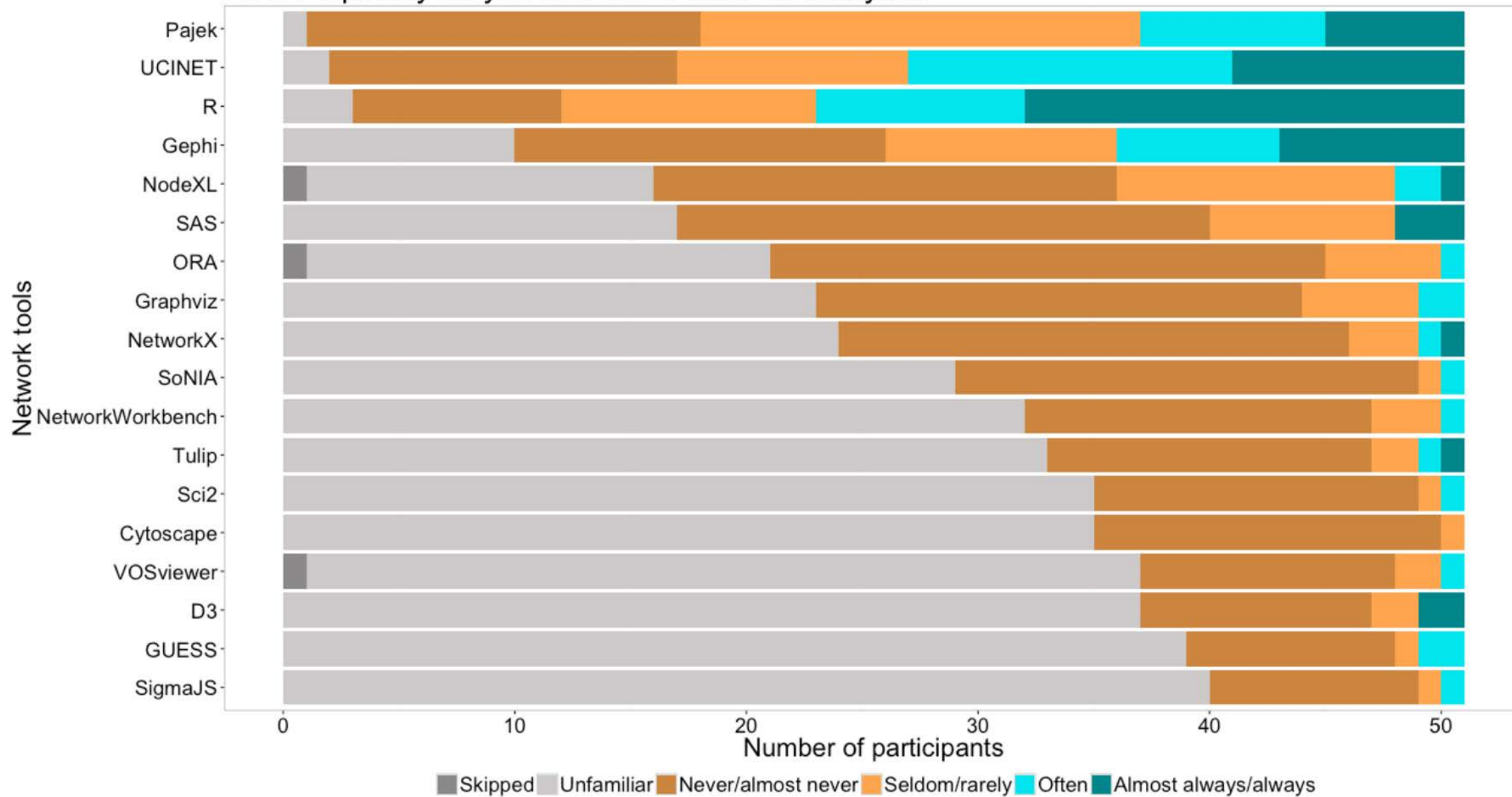
How frequently do you use these tools for analysis?



Select colors manually, or use alternate palette

```
scale_fill_manual(  
  values=c("snow4", "snow3",  
           "tan3", "tan1",  
           "turquoise2", "turquoise4"))  
  
scale_fill_manual(  
  values=c("#fee391", "#fe9929", "#cc4c02"))  
  
# Also see package RColorBrewer  
scale_fill_brewer(palette="BrBG")
```

How frequently do you use these tools for analysis?



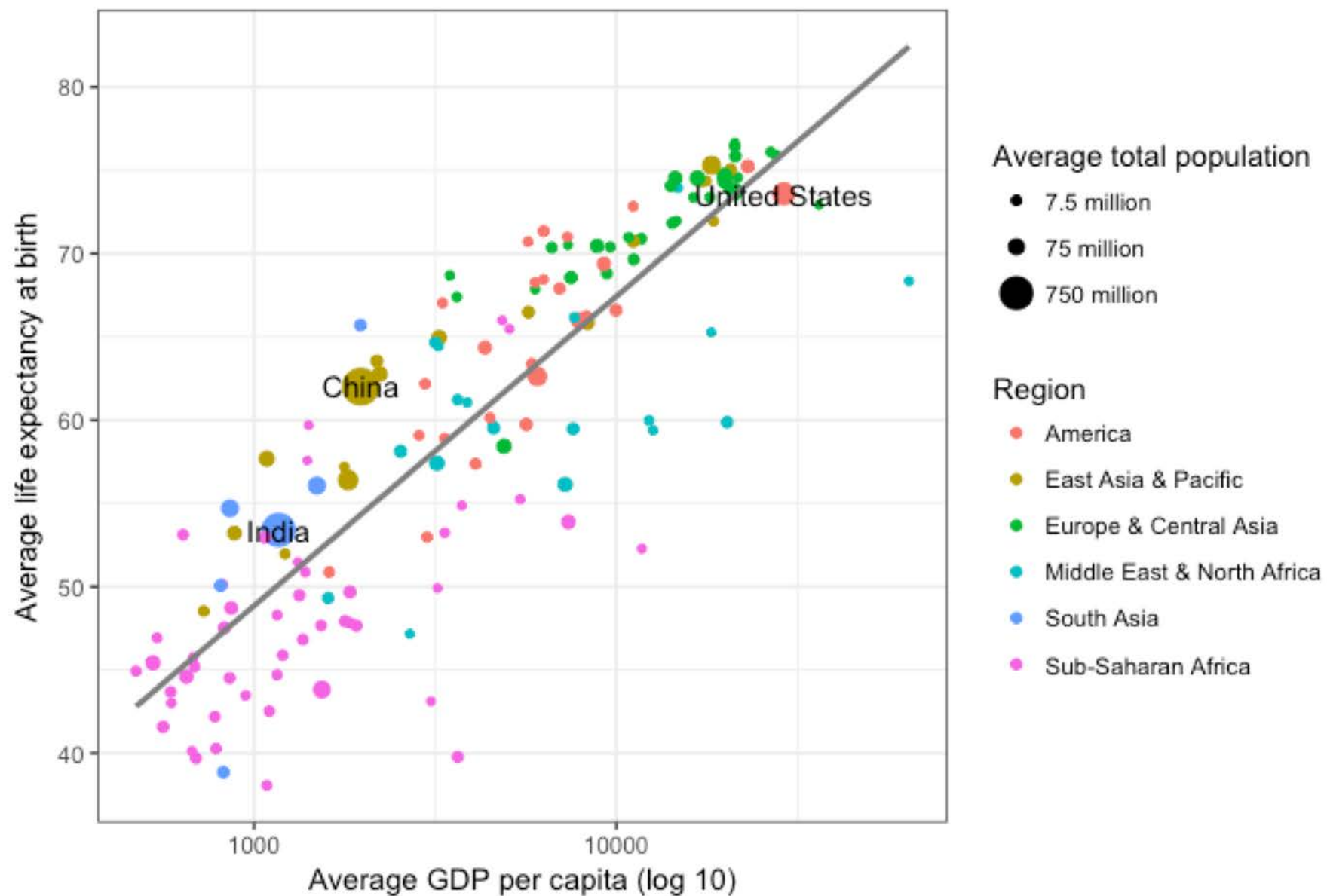
Star Wars opinion survey

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

Gapminder Data

<http://www.gapminder.org/>


Averages across all years of the traditional Gapminder dataset



Saving charts out

ggplot2 Cheat Sheet

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 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.

aplot(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 = 2)) ~ x, yend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(linetype = "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(fit))

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(carat, color))

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

THREE VARIABLES

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

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

h + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5, interpolate = FALSE)
x, y, alpha, fill

h + 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(latten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar() x, ymax, ymin, alpha, color, fill, 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(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

R Studio

RStudio® is a trademark of RStudio, Inc. • CC BY SA RStudio • info@rstudio.com • 844-468-1212 • rstudio.com • Learn more at <http://ggplot2.tidyverse.org> • ggplot2 2.1.0 • Updated: 2016-11

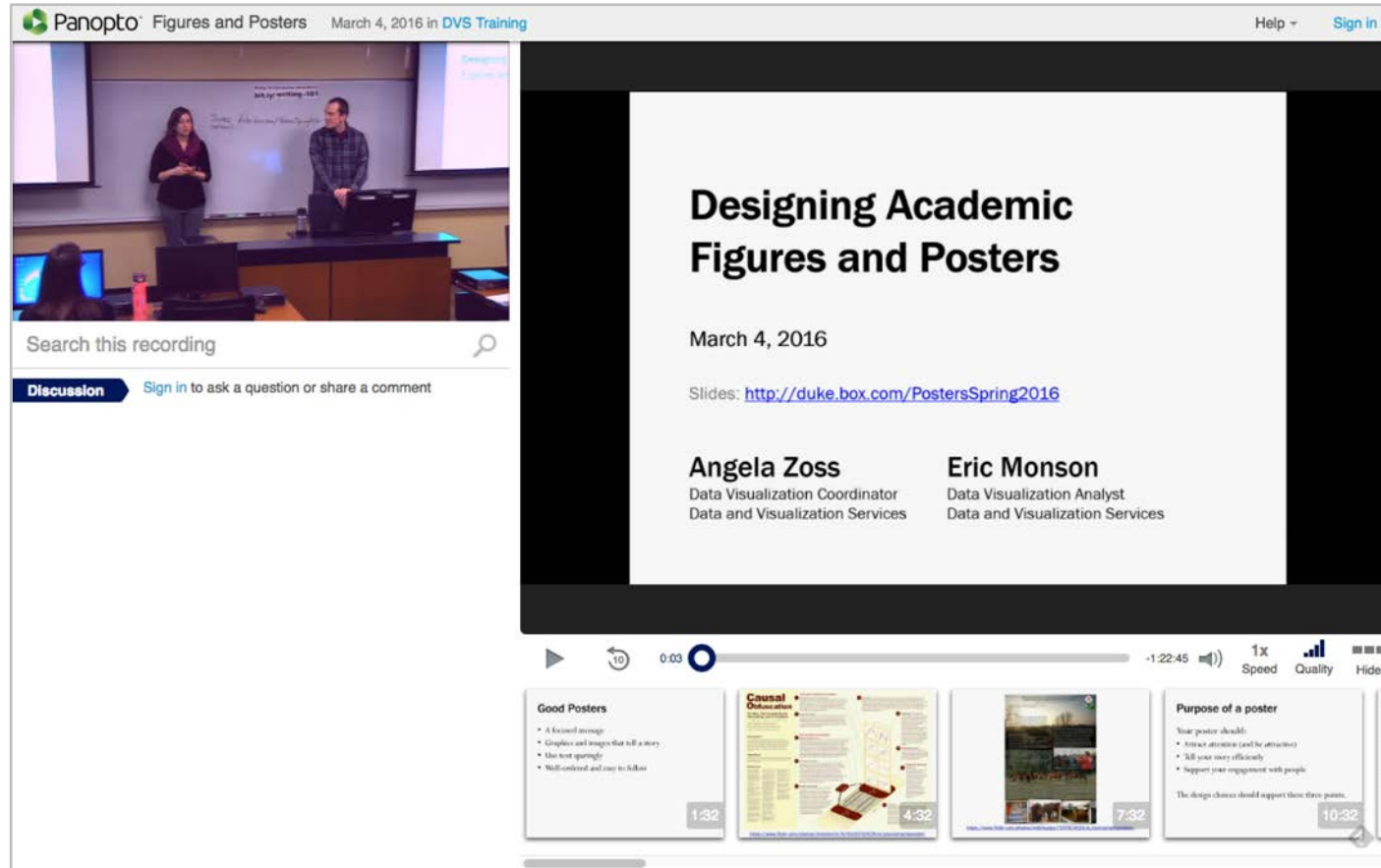
<https://www.rstudio.com/resources/cheatsheets/#ggplot2>

ggplot2 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>

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-organized and easy to follow
1:32

Causal Relationships
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