

Visualization in R using ggplot2

Angela Zoss

2/20/19

<https://github.com/amzoss/ggplot2-S19b>

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>

Why visualize in R?

- Quickly explore data
- Save time switching to another tool
- Use charts to inspire new analyses and vice versa
- Reproducibility

Why care about reproducibility?

- Open science makes review easier
- Increasingly a requirement
- Saves you a lot of time trying to figure out what you did last time!

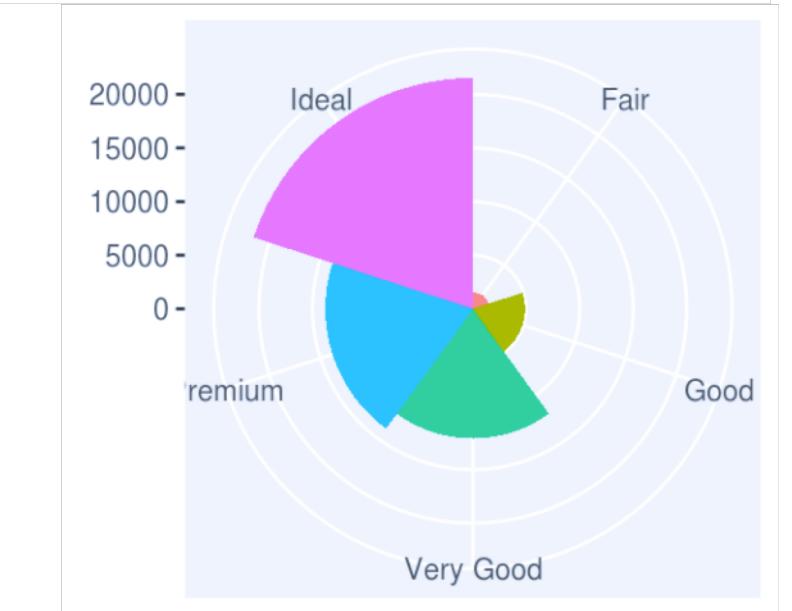
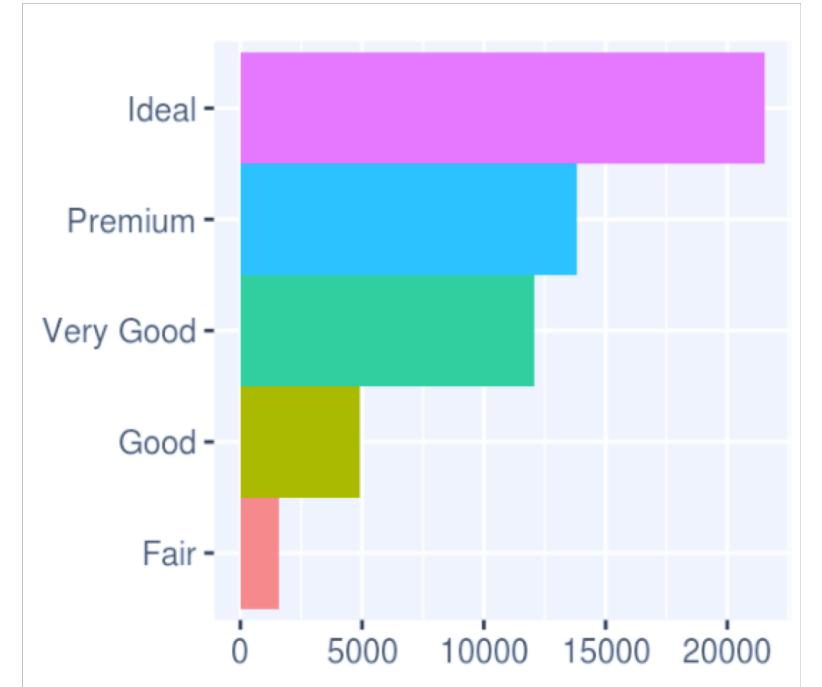
*“Your closest collaborator is **you** six months ago,
but you don’t reply to emails.”*

- *Mark Holder*

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.).

Wilkinson, Leland. (2005). *The grammar of graphics (2nd ed)*. New York: Springer.

Why ggplot2 instead of base R?

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

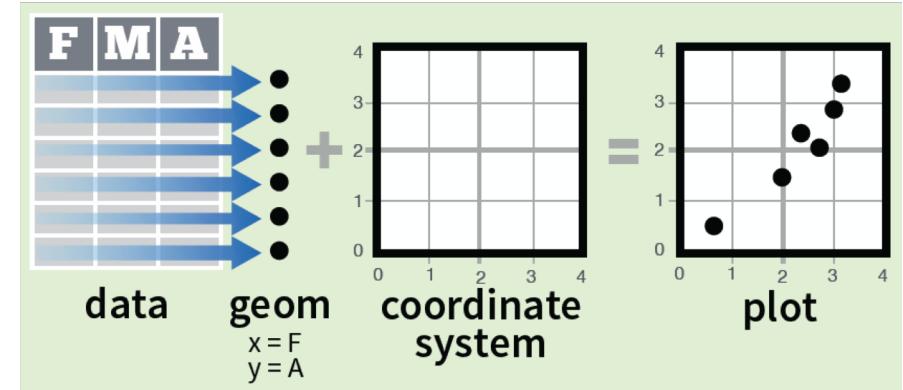
“Why I use ggplot2”, David Robinson

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

ggplot2: Elements

Basic elements in any ggplot2 visualization

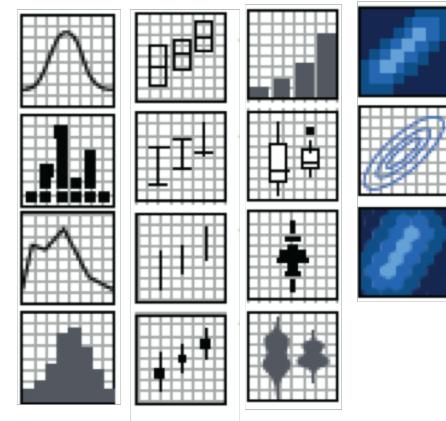
- **data**
- **aesthetics**
(variable mappings)
- **geom**
(chart type or shape)
- coordinate system
(the arrangement of the marks;
most geoms use default, cartesian)



<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

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**

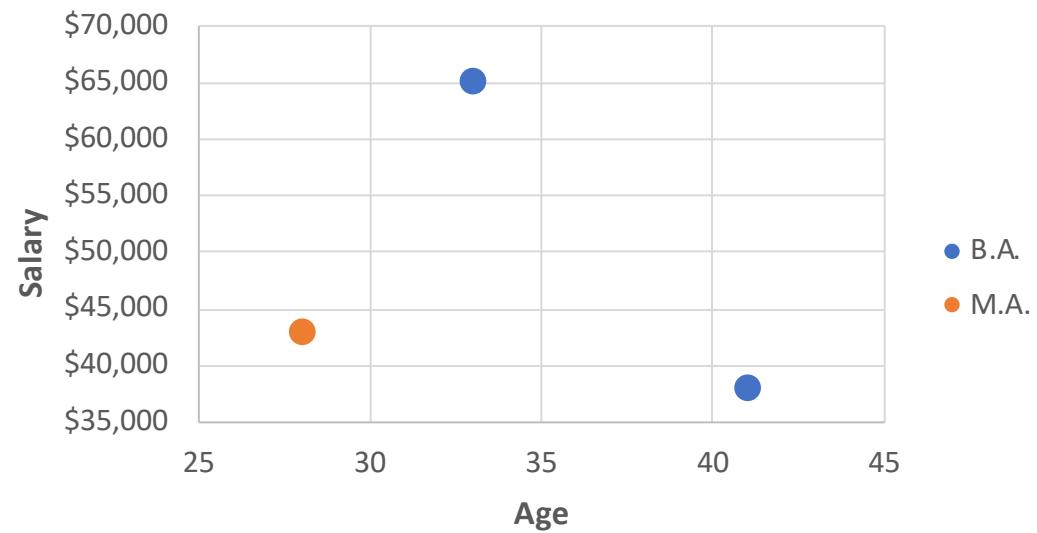
```
geom_... ( mapping = aes(variable mappings) ,  
          non-variable adjustments )
```

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.

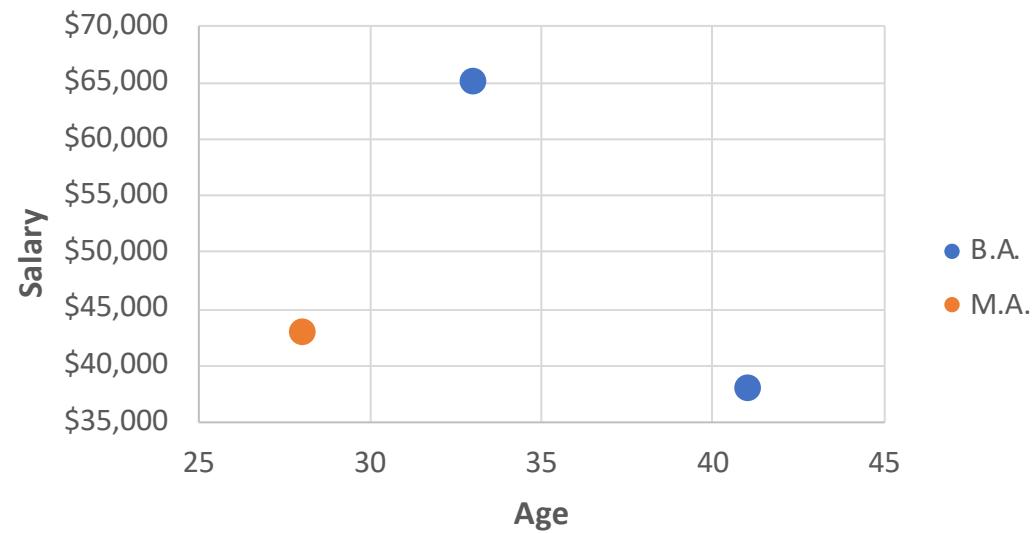
```
ggplot(employees)
```



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.

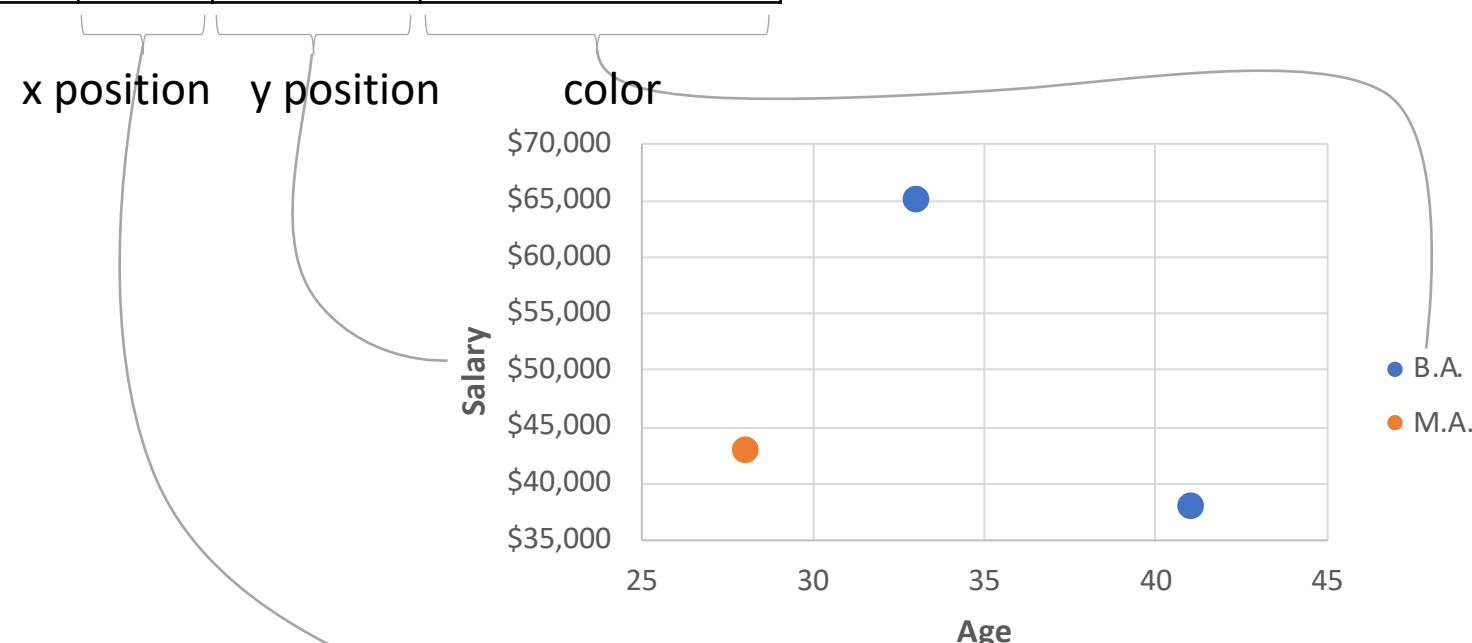
```
ggplot(employees) +  
  geom_point()
```



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.

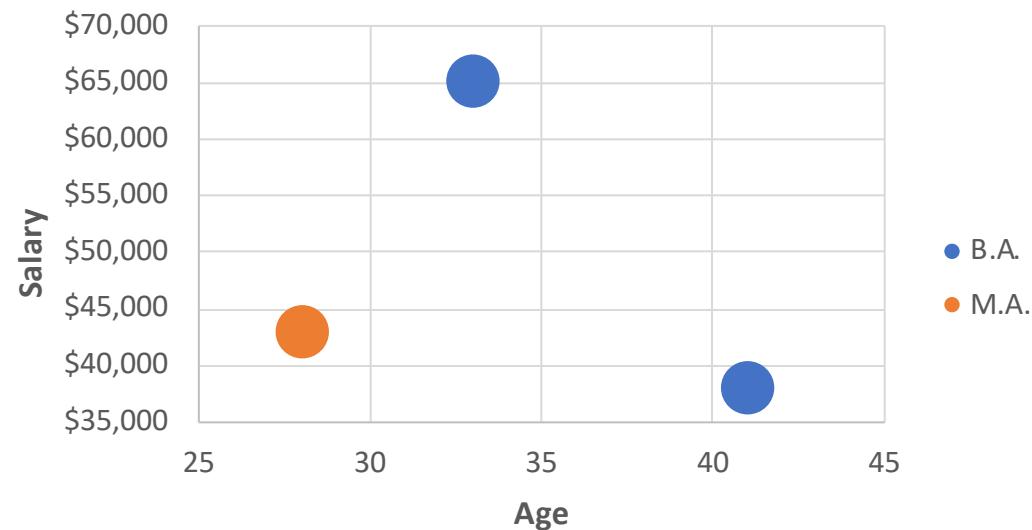
```
ggplot(employees) +  
  geom_point(  
    aes(x=age,  
        y=salary,  
        color=degree))
```



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)
```



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 )
```

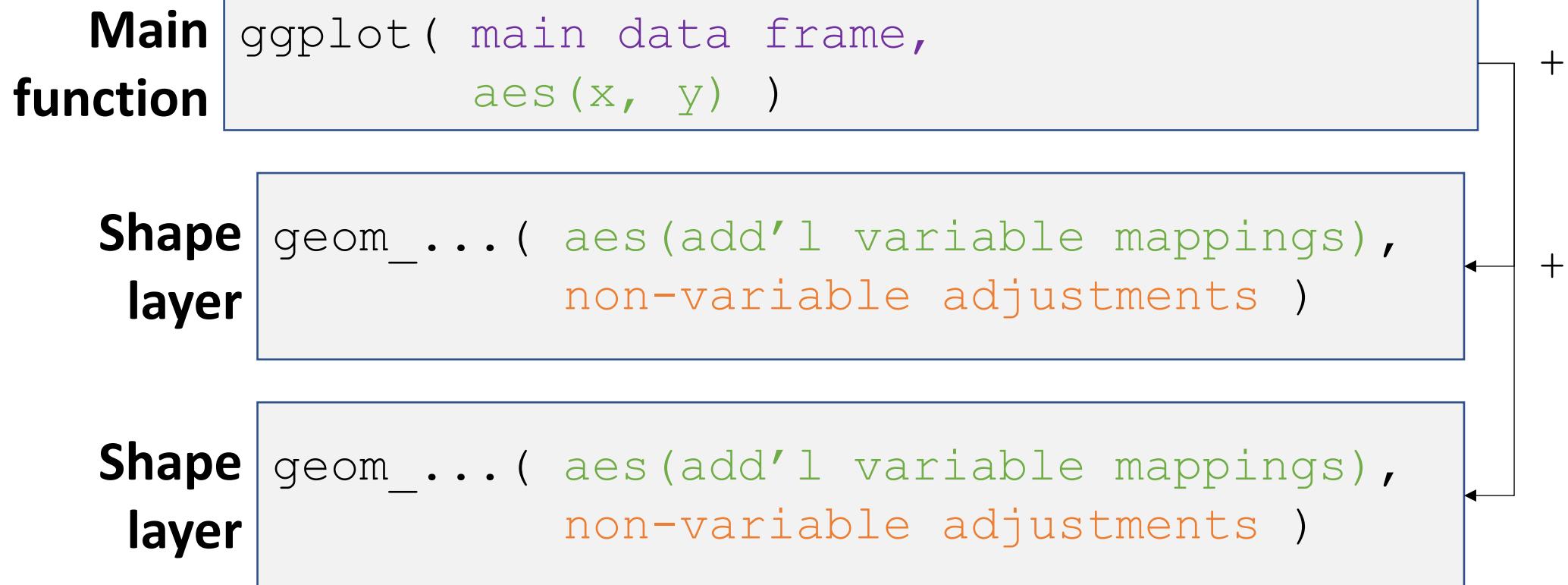
+

Shape layer

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

←

General inheritance strategy



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-S19b>

With Git installed

In RStudio:

- Project → New project
- Version Control
- Git
 - Paste in GitHub URL
 - Project directory name: **ggplot2-S19b**
 - 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.

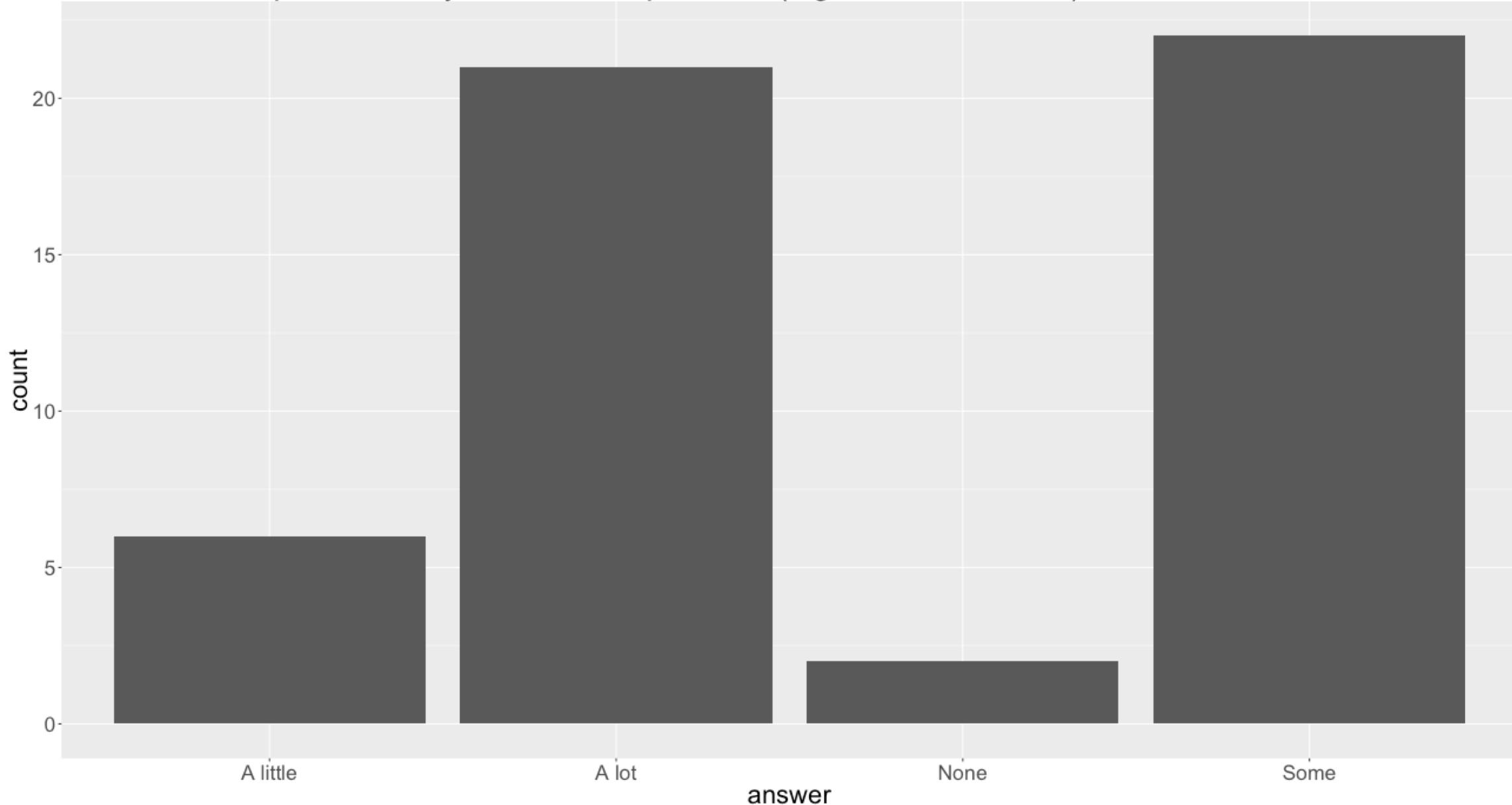
Game of Thrones character ratings

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

Principles for Effective Visualizations

Principle 1: Order matters

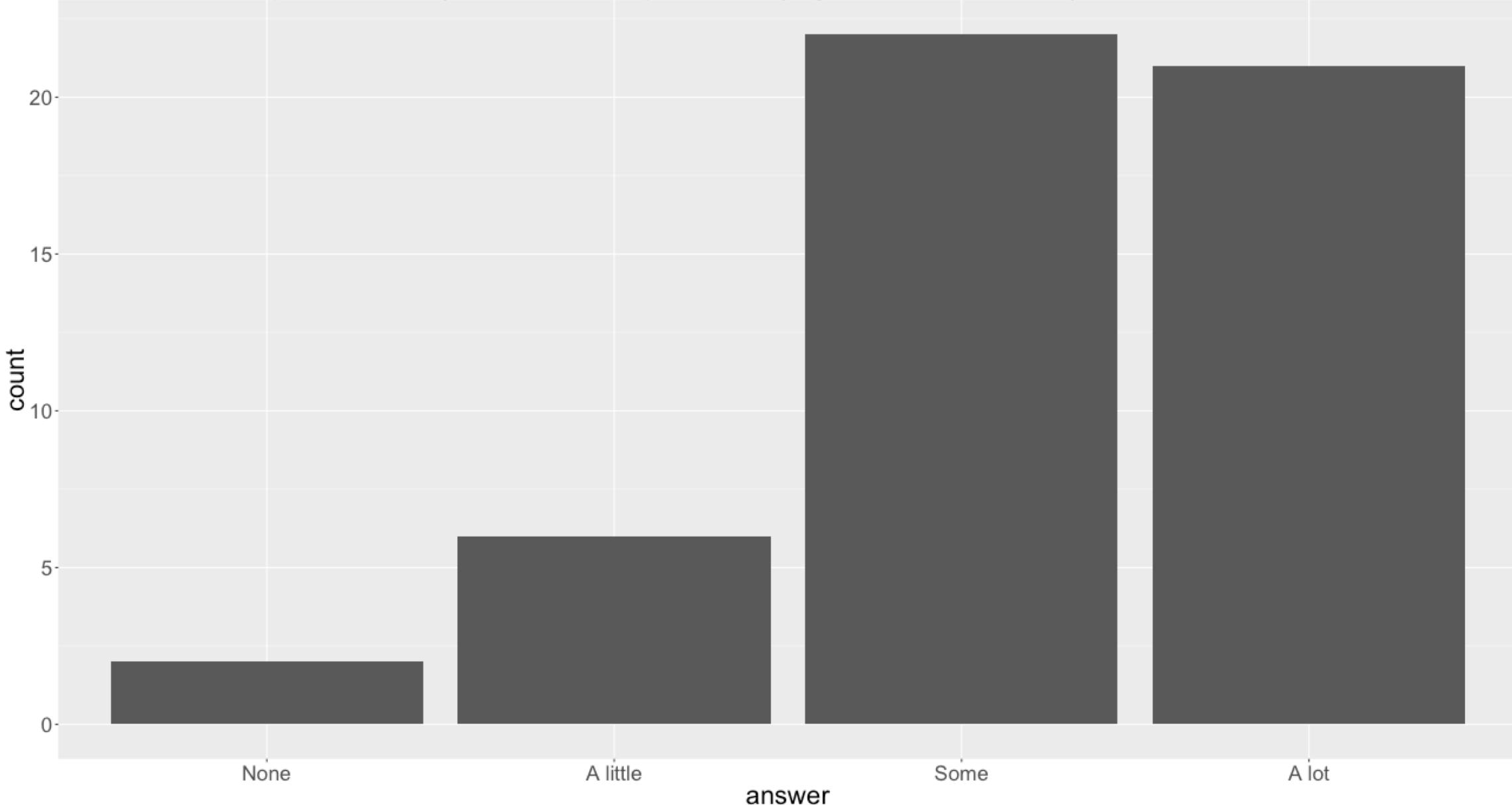
How much experience do you have as a producer (e.g., reader, follower) of network science research?



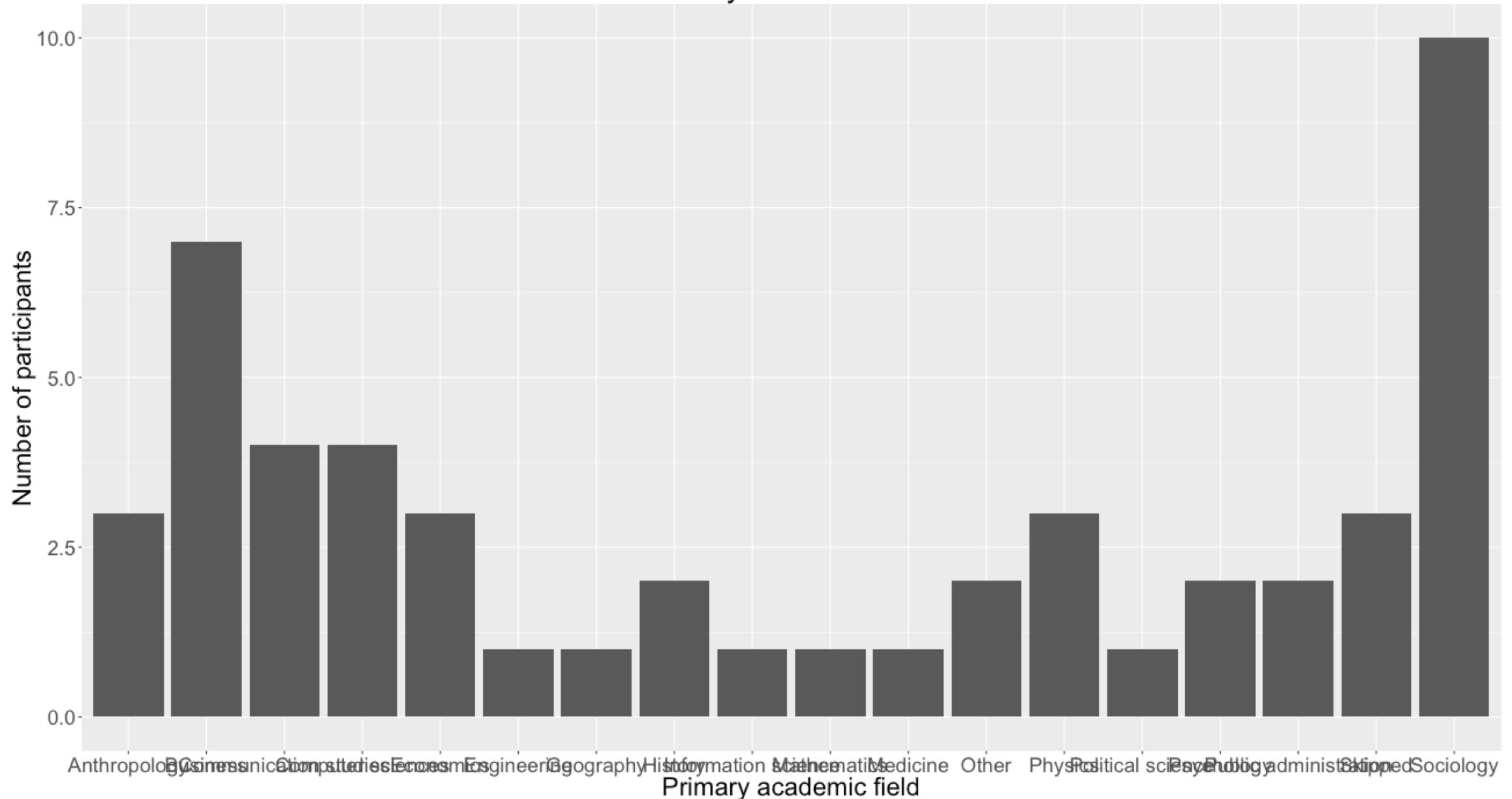
Order by meaning

```
data$answer <-
  factor(data$answer,
         levels=c("None", "A little", "Some", "A lot"),
         ordered = TRUE)
```

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



Primary academic field

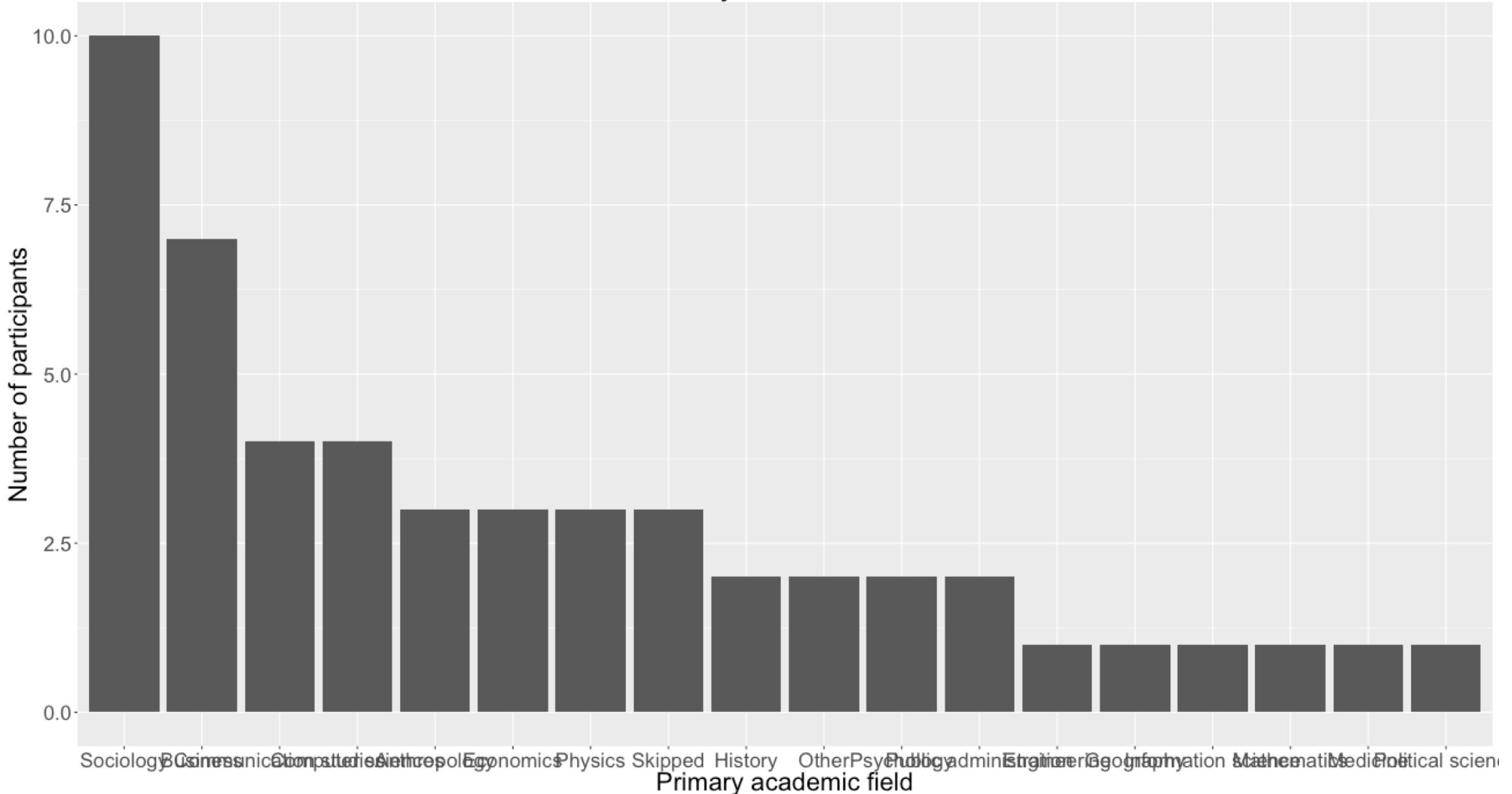


Order by value

```
data$academic_field <-  
  factor(data$academic_field,  
         levels=names(  
           sort(  
             table(  
               data$academic_field),decreasing=TRUE)))
```

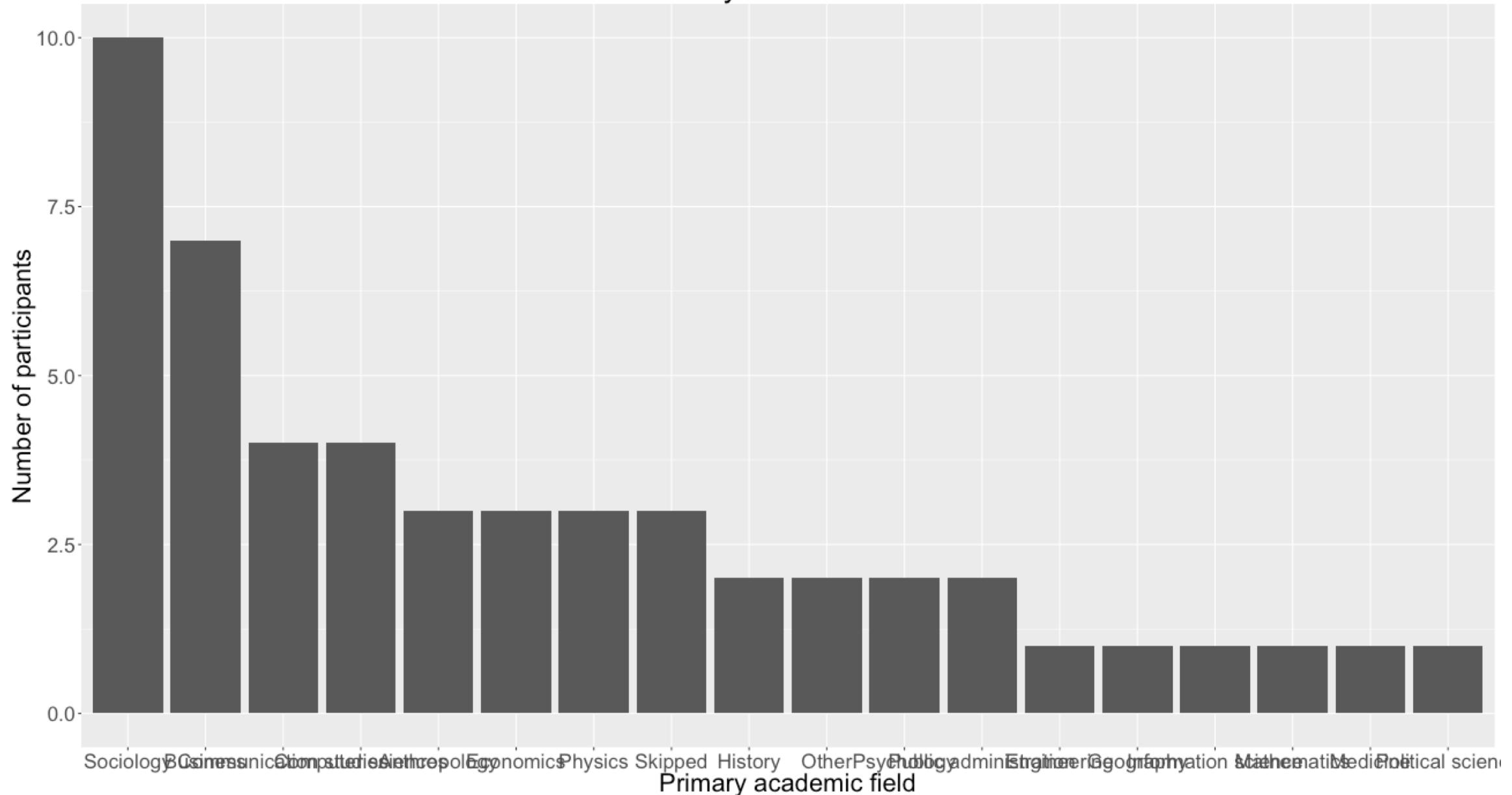
```
data$academic_field <-  
  fct_infreq(as_factor(data$academic_field))
```

Primary academic field



Principle 2:
Put long categories on y-axis

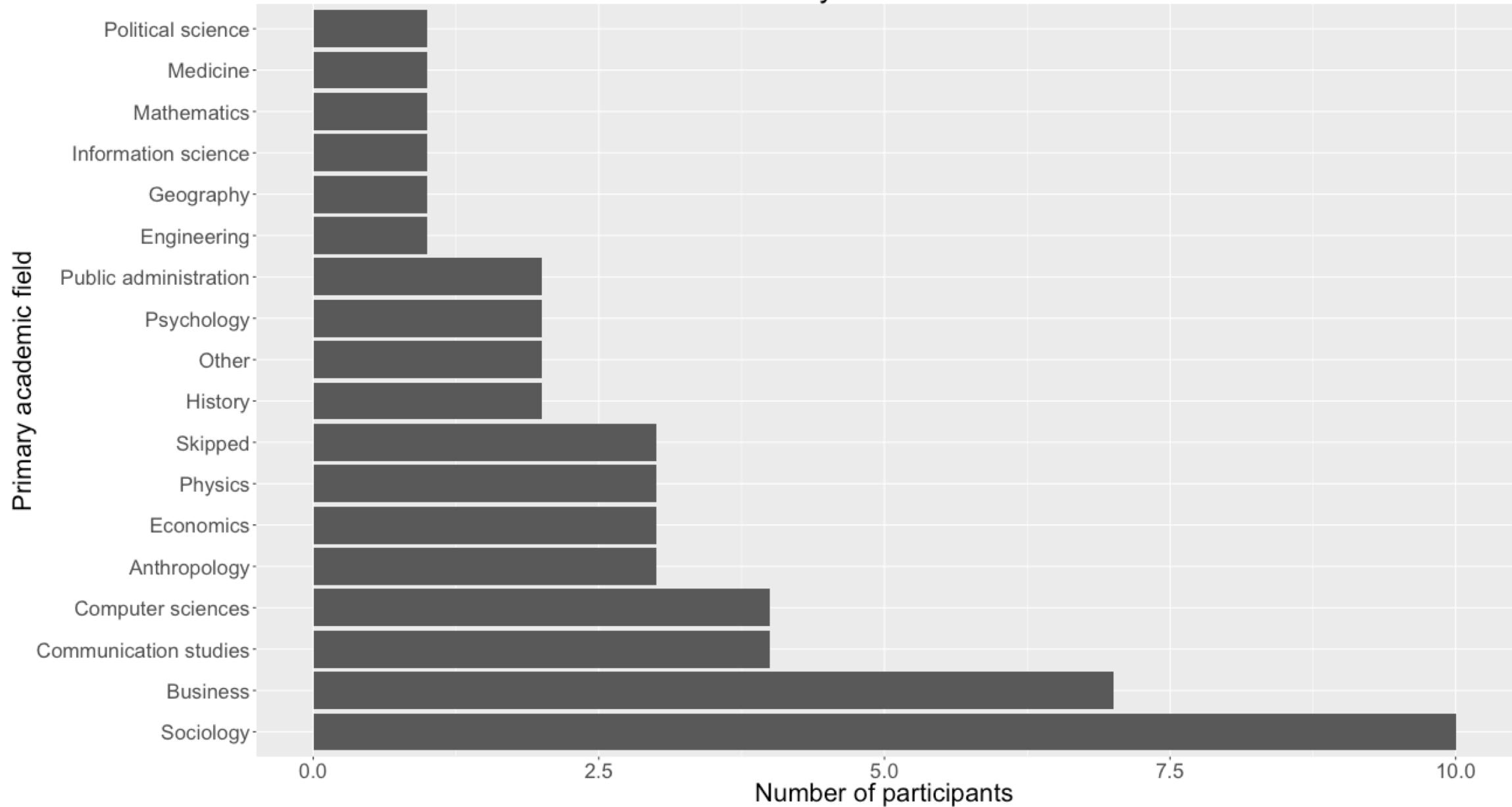
Primary academic field



Flip the axes

```
coord_flip()
```

Primary academic field

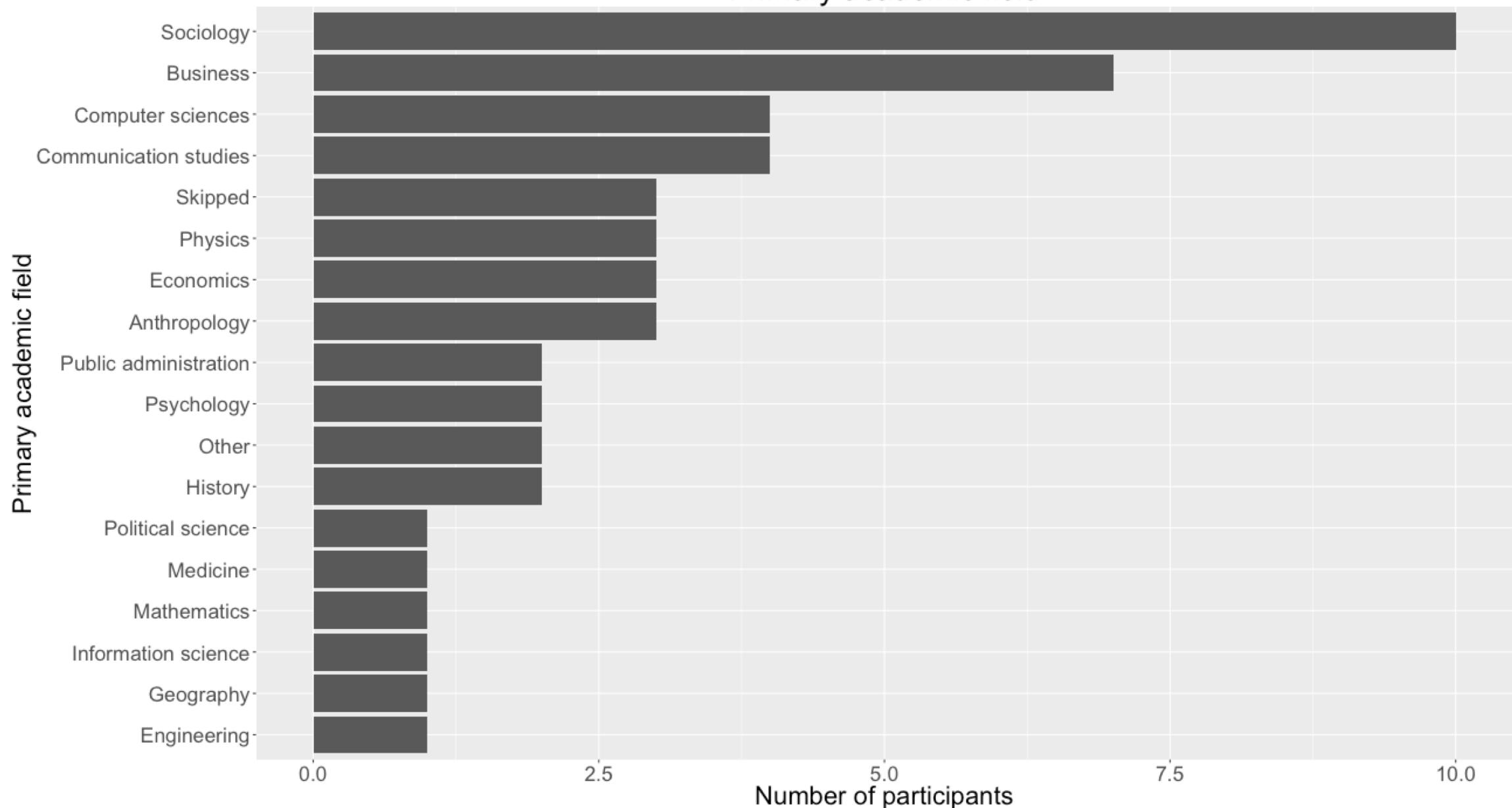


Oops!

```
data$academic_field <-  
  factor(data$academic_field,  
         levels=names(  
             sort(  
                 table(data$academic_field),  
                 decreasing=TRUE) ))
```

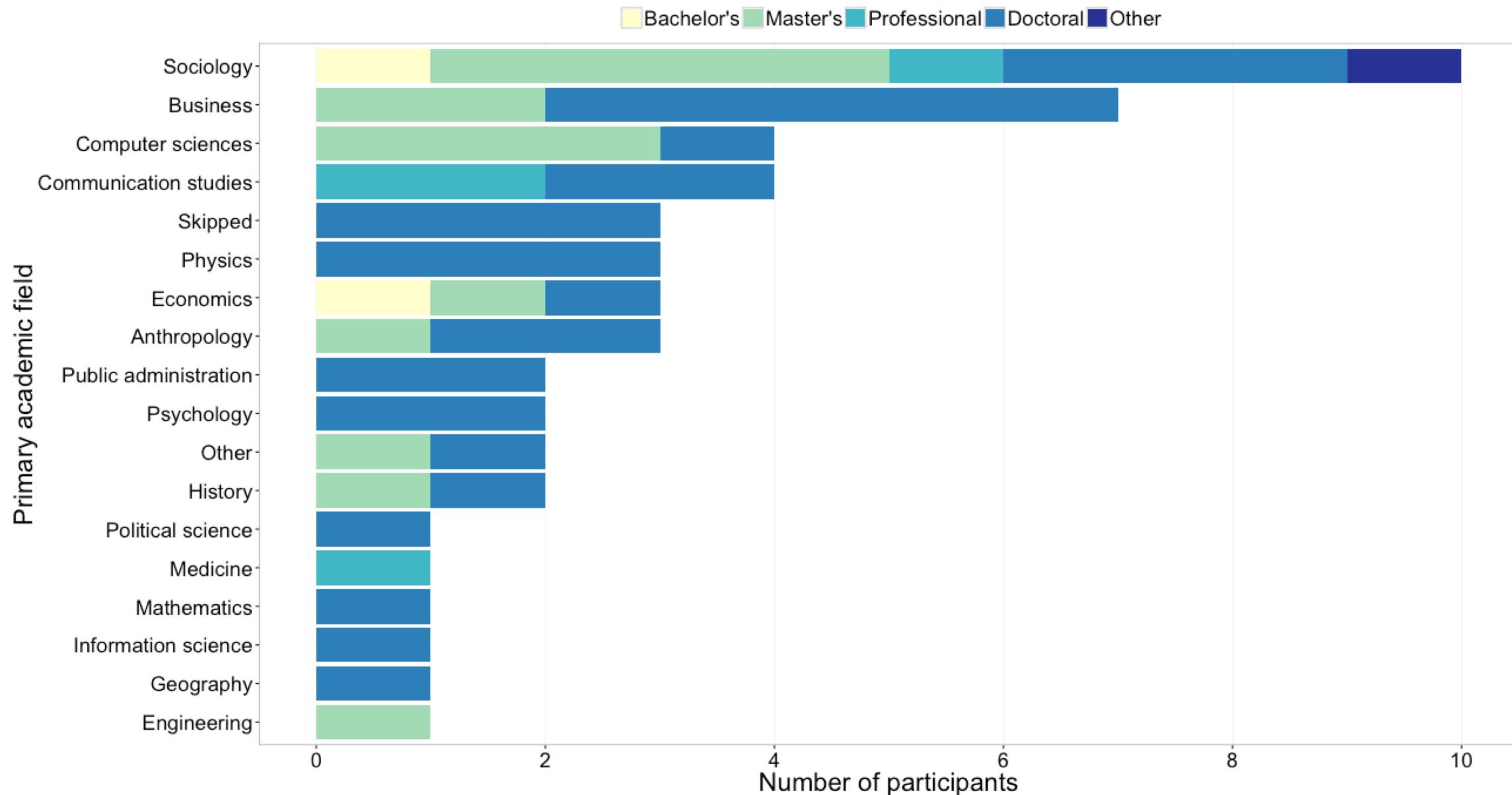
```
data$academic_field <-  
  fct_rev(fct_infreq(as_factor(data$academic_field)))
```

Primary academic field

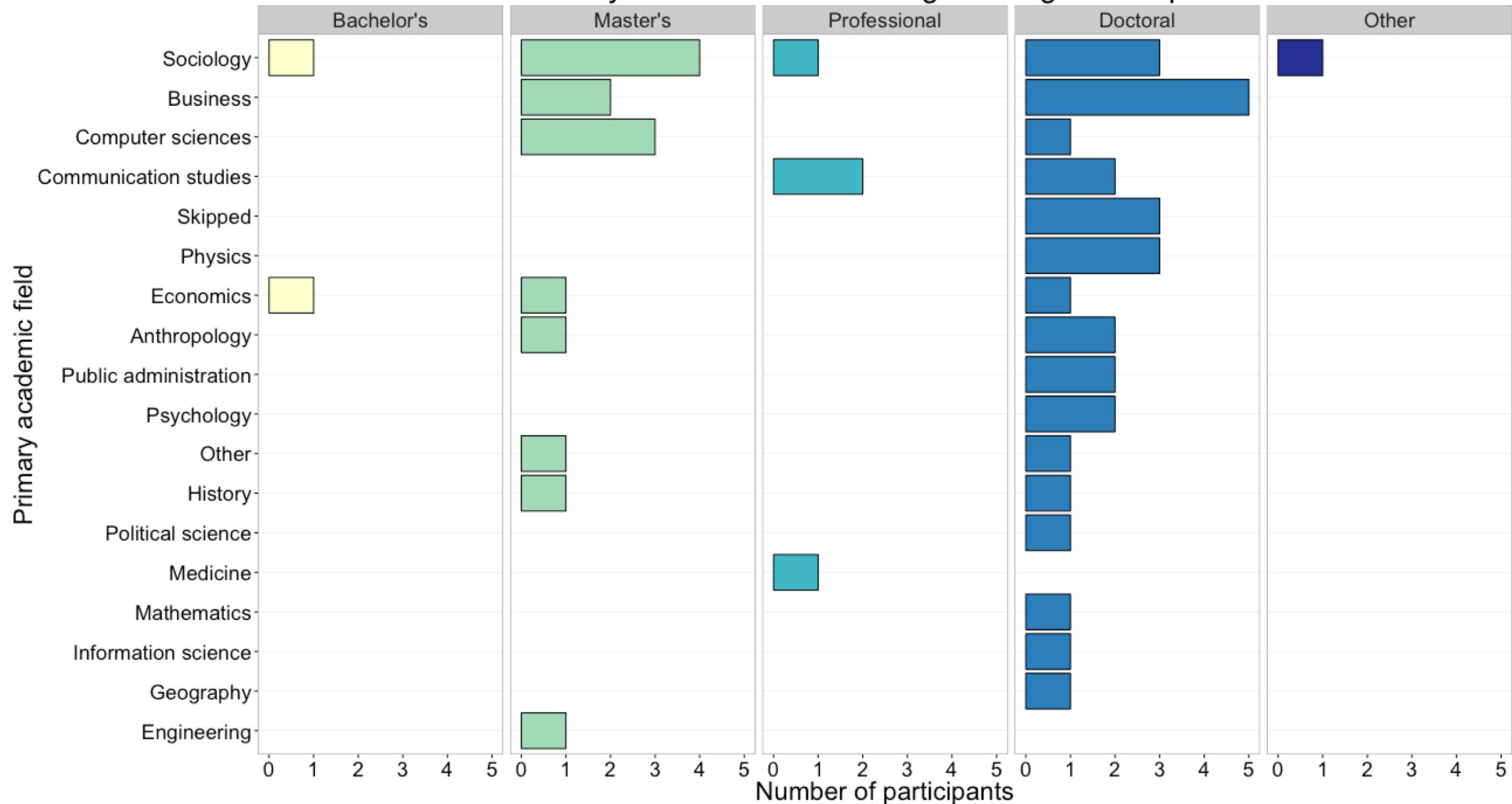


Principle 3: Pick a purpose

Primary academic field and highest degree completed



Primary academic field and highest degree completed



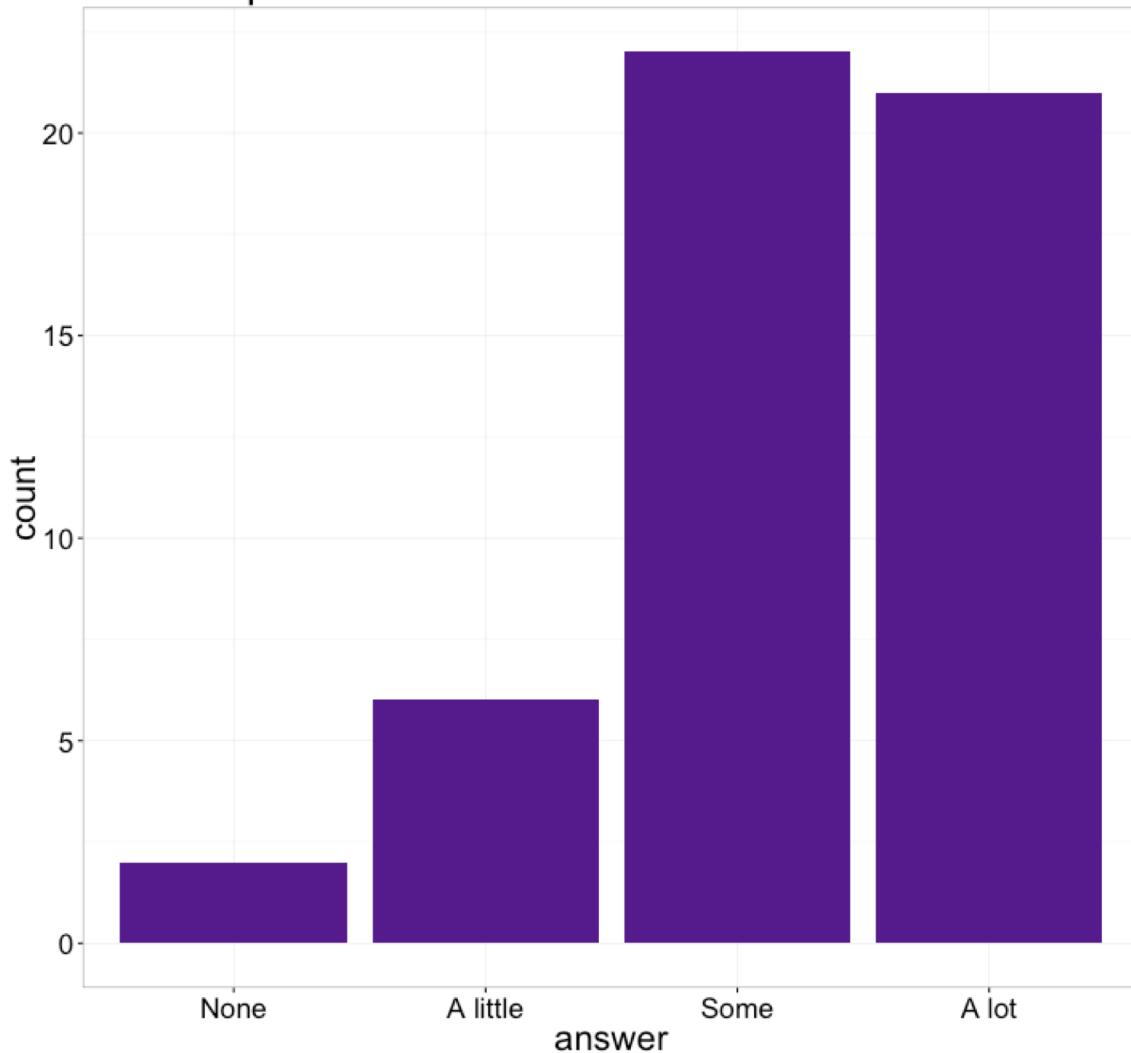
Different placement helps with different comparisons

```
fill=highest_degree
```

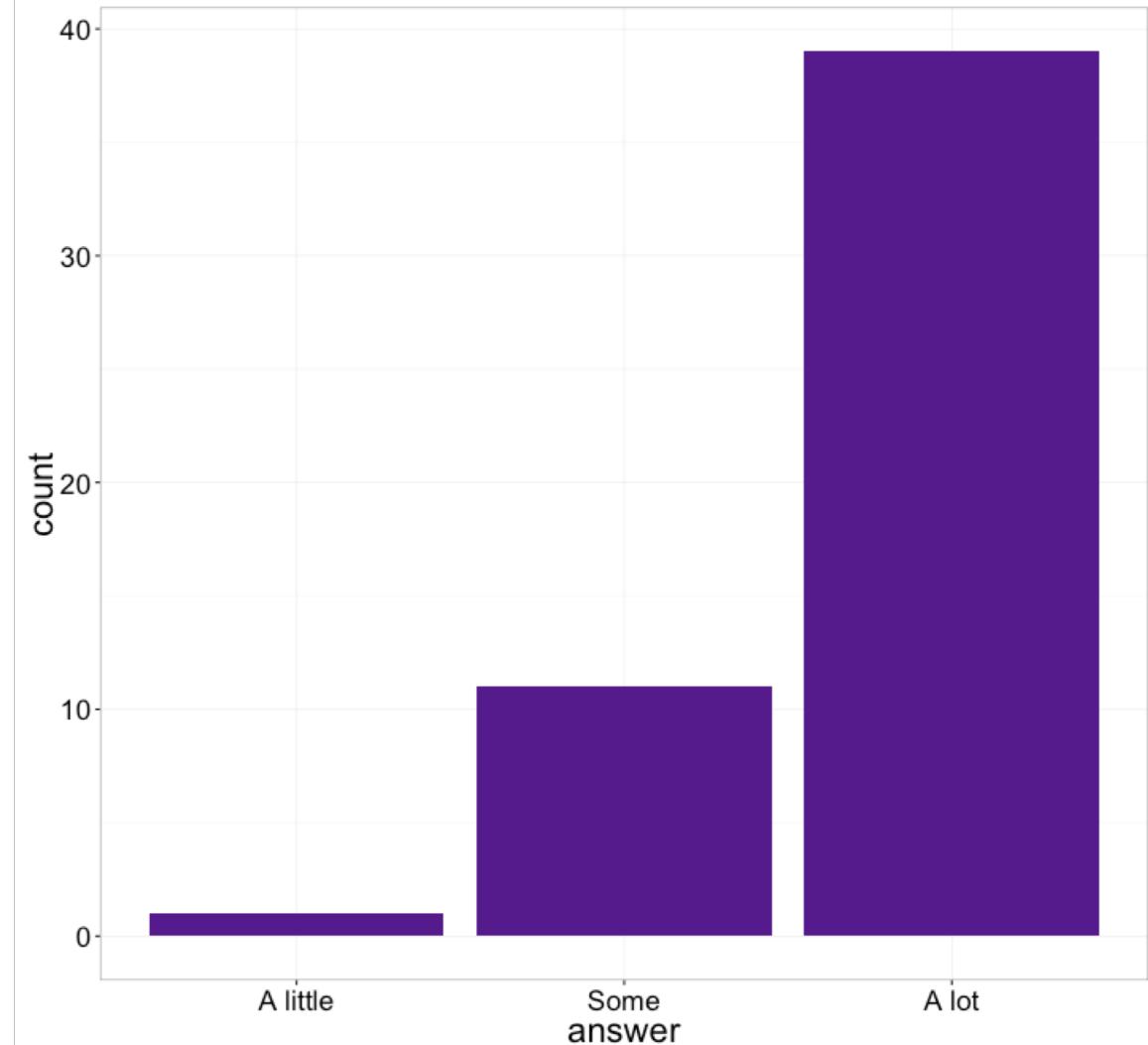
```
facet_grid(.~highest_degree)
```

Principle 4:
Keep scales consistent

How much experience do you have as a producer of network science research?



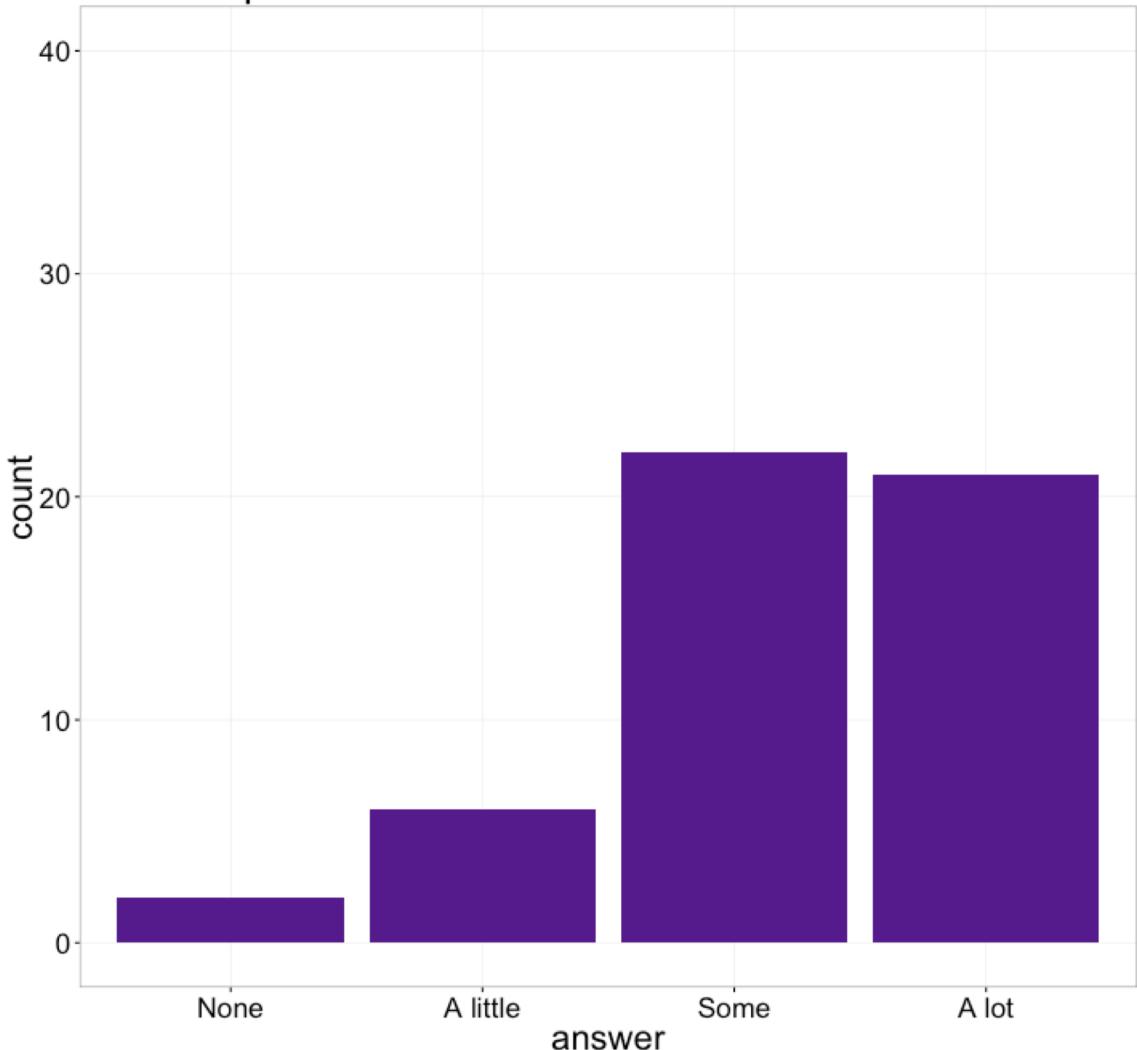
How much experience do you have as a consumer of network science research?



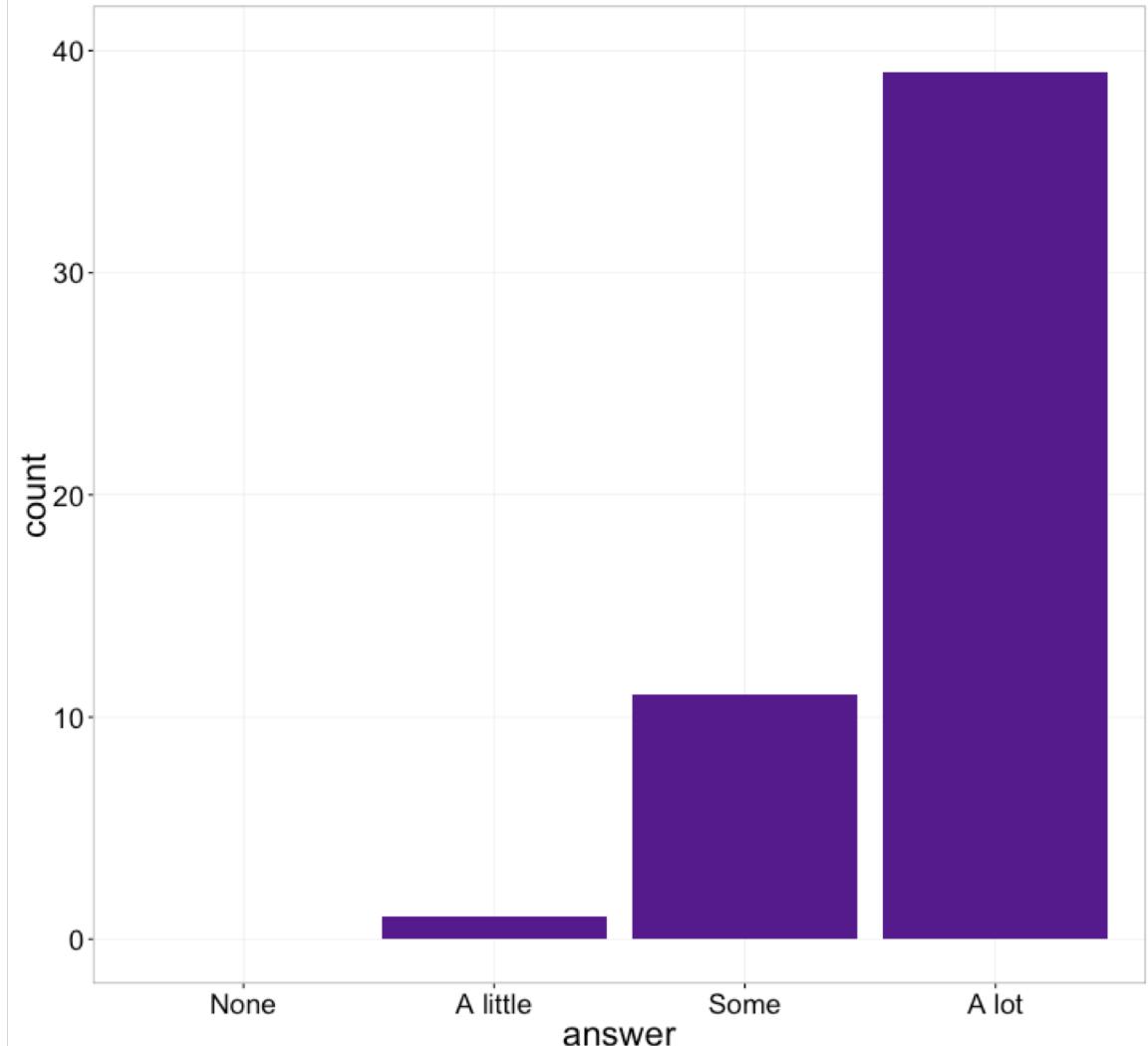
Keep all categories, manually set axes

```
scale_x_discrete(drop=FALSE)
scale_y_continuous(limits=c(0,40),
                   breaks=c(0,10,20,30,40),
                   minor_breaks=NULL)
```

How much experience do you have as a producer of network science research?

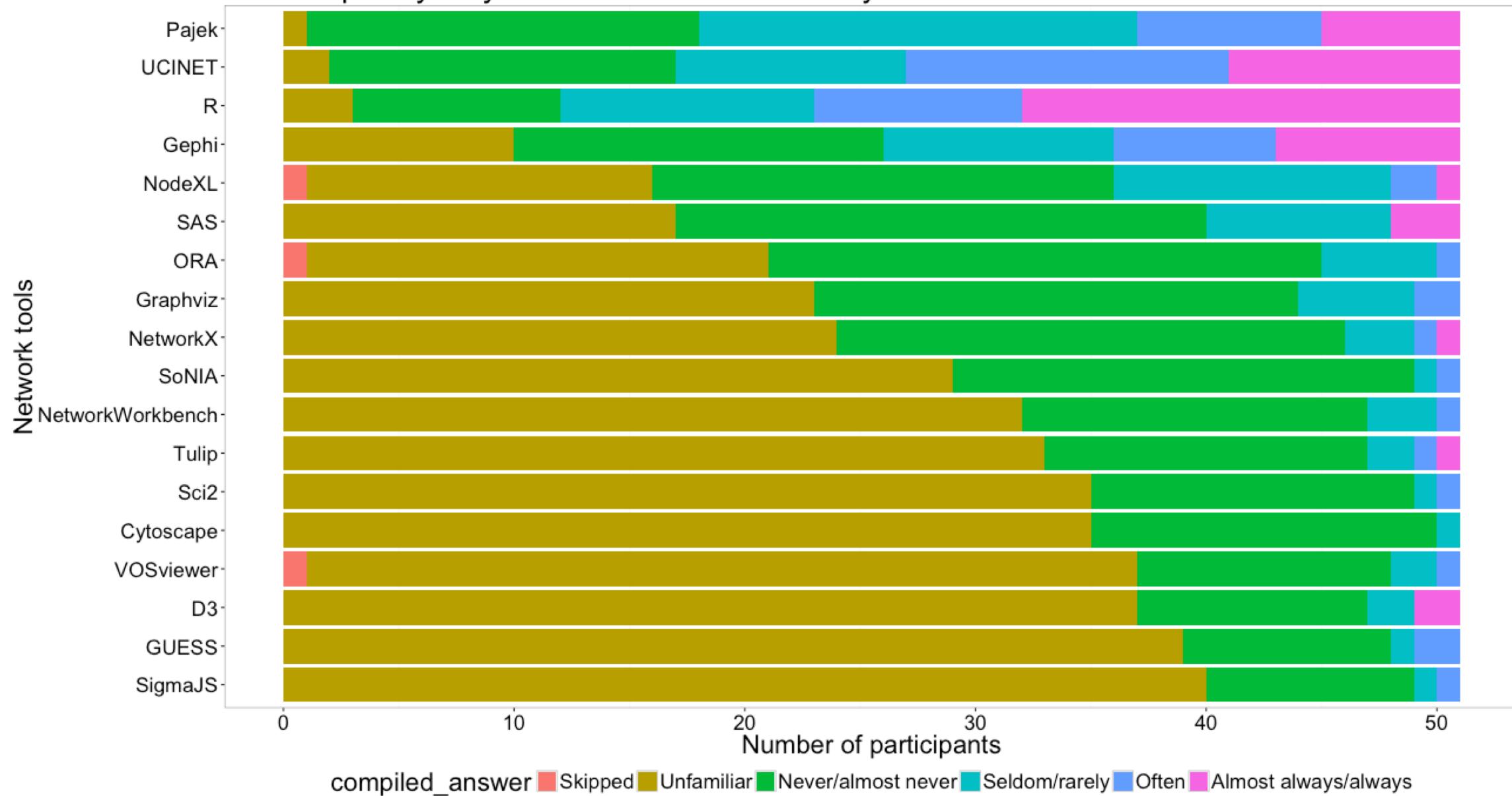


How much experience do you have as a consumer of network science research?



Principle 5:
Select meaningful colors

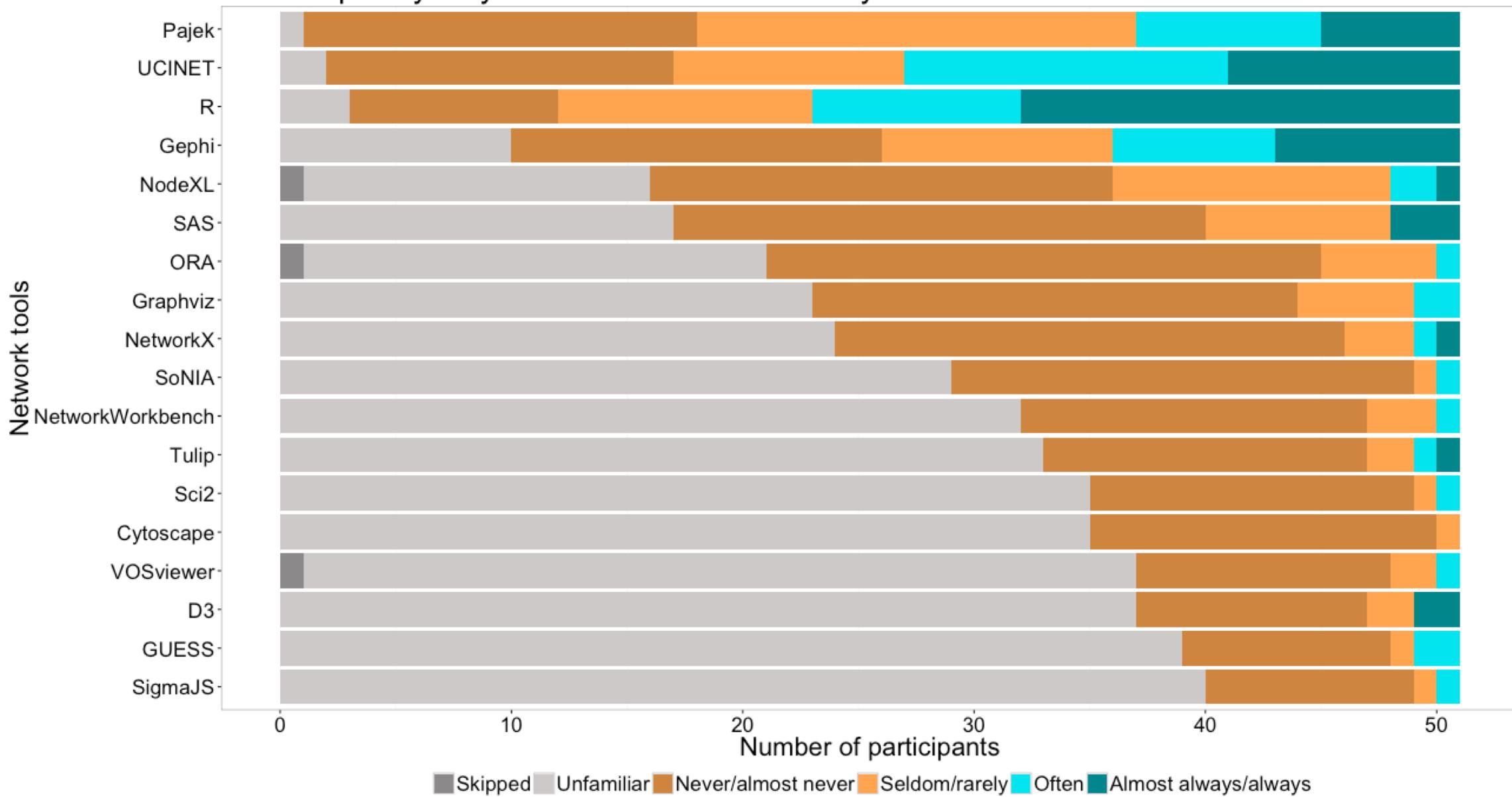
How frequently do you use these tools for analysis?



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/>

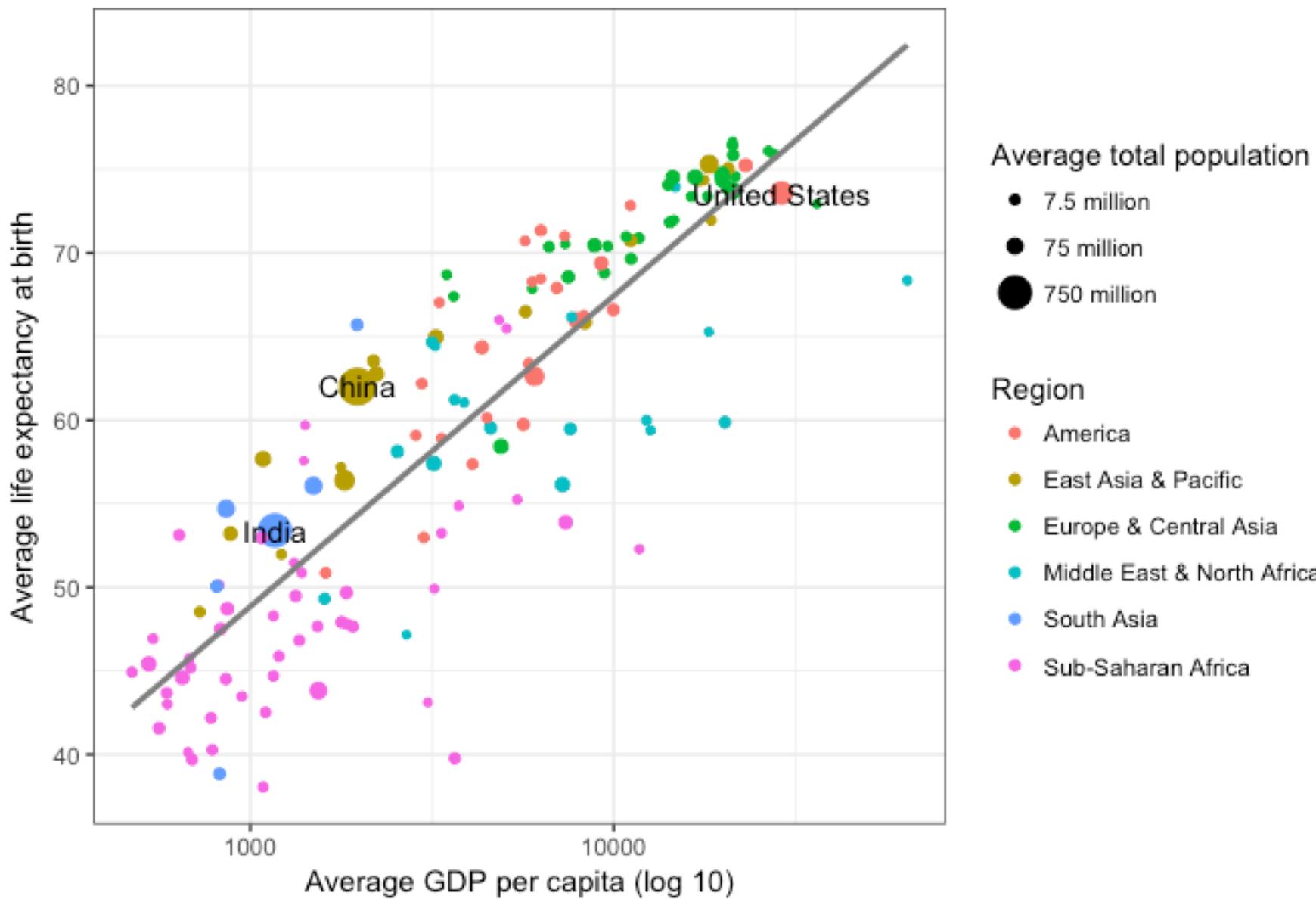
Star Wars character data

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

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's **aesthetics** like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot(data = <DATA>) +  
  <GEOM_FUNCTION>(aes(x = <stat>, y = <stat>)) +  
  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' 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, unemployed))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
a + geom_label(aes(label = cyl), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
x, y, alpha, angle, color, curvature, linetype, size,
linemetre=1)

b + geom_curve(aes(yend = lat + 1,
xend=long+1,curvature=2)) -> x, xend, y, yend,
alpha, angle, color, curvature, linetype, size,
linemetre=1)

a + geom_path(linend="butt", linejoin="round",
x, y, alpha, color, group, linetype, size,
linemetre=1)

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, xmin, ymin, xmax,
ymin, alpha, color, fill, 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

e + geom_text(aes(label = cyl), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
x, y, alpha, angle, color, family, fontface, hjust,
lineheight, size, vjust
```



#### TWO VARIABLES



```
continuous x, continuous y
e <- ggplot(mpg, aes(cty, hwy))

a + geom_label(aes(label = cyl), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
x, y, alpha, angle, color, curvature, fontface, hjust,
lineheight, size, vjust

b + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

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

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

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

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

e + geom_text(aes(label = cyl), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
x, y, alpha, angle, color, family, fontface, hjust,
lineheight, size, vjust
```



#### continuous bivariate distribution



```
h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, colour, 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_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



```
d <- data.frame(grp = c("A", "B"), fit = 4.5, se = 1.0)
j <- ggplot(df, aes(grp, fit, ymin = fit - se, ymax = fit + se))

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

j + geom_errorbar()
x, y, ymax, ymin, alpha, color, fill, 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 = tolowerrownames(USArrests))
murder <- discrete, state <- discrete
k <- ggplot(data, aes(fill = murder))

k + geom_map(aes(map_id = state), map = map)
map_id, alpha, color, fill, linetype, size

+ expand_limits(x = mapLong, y = mapLat),
map_id, alpha, color, fill, linetype, size
```



#### THREE VARIABLES



```
seals$z <- with(seals, sqrt(data.long^2 + data.lat^2))
l <- ggplot(seals, aes(long, lat))

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

d <- ggplot(mpg, aes(ffill))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

discrete
d <- ggplot(mpg, aes(ffill))

d + geom_bar()
x, alpha, color, fill, linetype, size, weight

+ 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 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

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 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 10-second mark, and a 1:22:45 end time, along with buttons for "Speed" (set to 1X), "Quality", and "Hide". At the very bottom, there are four small thumbnail images related to poster design, each with a timestamp: 1:32, 4:32, 7:32, and 10:32.

<http://bit.ly/DVSvideos>

Questions?

angela.zoss@duke.edu