Visualization for Data Science in R

Angela Zoss

Data Matters 2022

https://www.angelazoss.com/RVis-2Day/

Schedule, Day 2

Session	Topics	Duration
Session 1	ggplot2 review, advanced techniques	9:30 a.m. – 10:35 a.m.
Morning break		10:35 a.m. – 10:50 a.m.
Session 2	Simple interactive plots	10:50 a.m. – 11:55 a.m.
Lunch		11:55 a.m. – 1:10 p.m.
Session 3	Intro to Shiny	1:10 p.m. – 2:15 p.m.
Afternoon break		2:15 p.m. – 2:30 p.m.
Session 4	Shiny examples and practice	2:30 p.m. – 3:35 p.m.
Q&A		3:35 p.m. – 3:40 p.m.

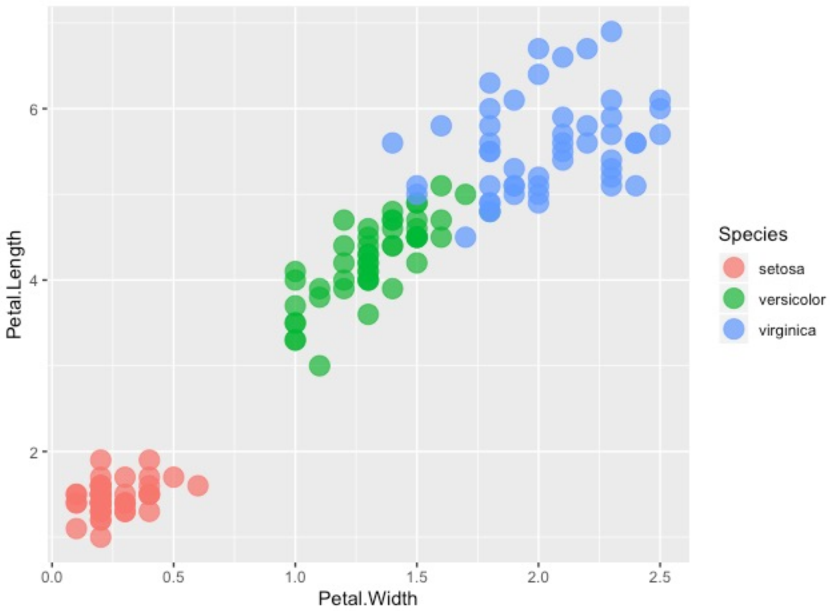
Day 1 Review

Example plot

“iris”

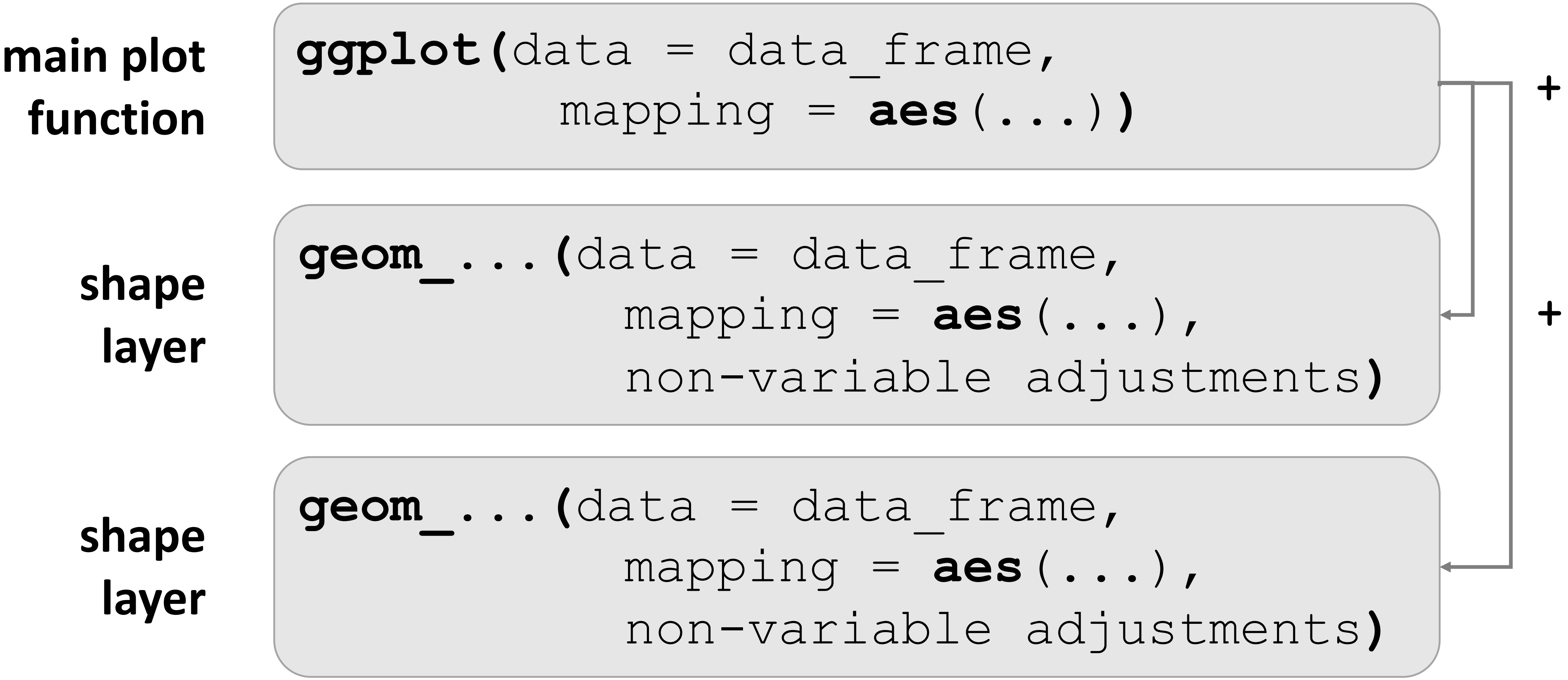
Petal.Width	Petal.Length	Species
0.3	1.4	setosa
1.3	4.0	versicolor
2.1	5.7	virginica

```
ggplot(data=iris) +
  geom_point(
     mapping=aes(x=Petal.Width,
                           y=Petal.Length,
                           color=Species),
     size=5, alpha=.75)
```



General pattern

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



main plot function

```
ggplot(data = data_frame,
          mapping = aes(...))
```

+

shape layer

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

+

shape layer

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

geom vs. scale vs. theme

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)

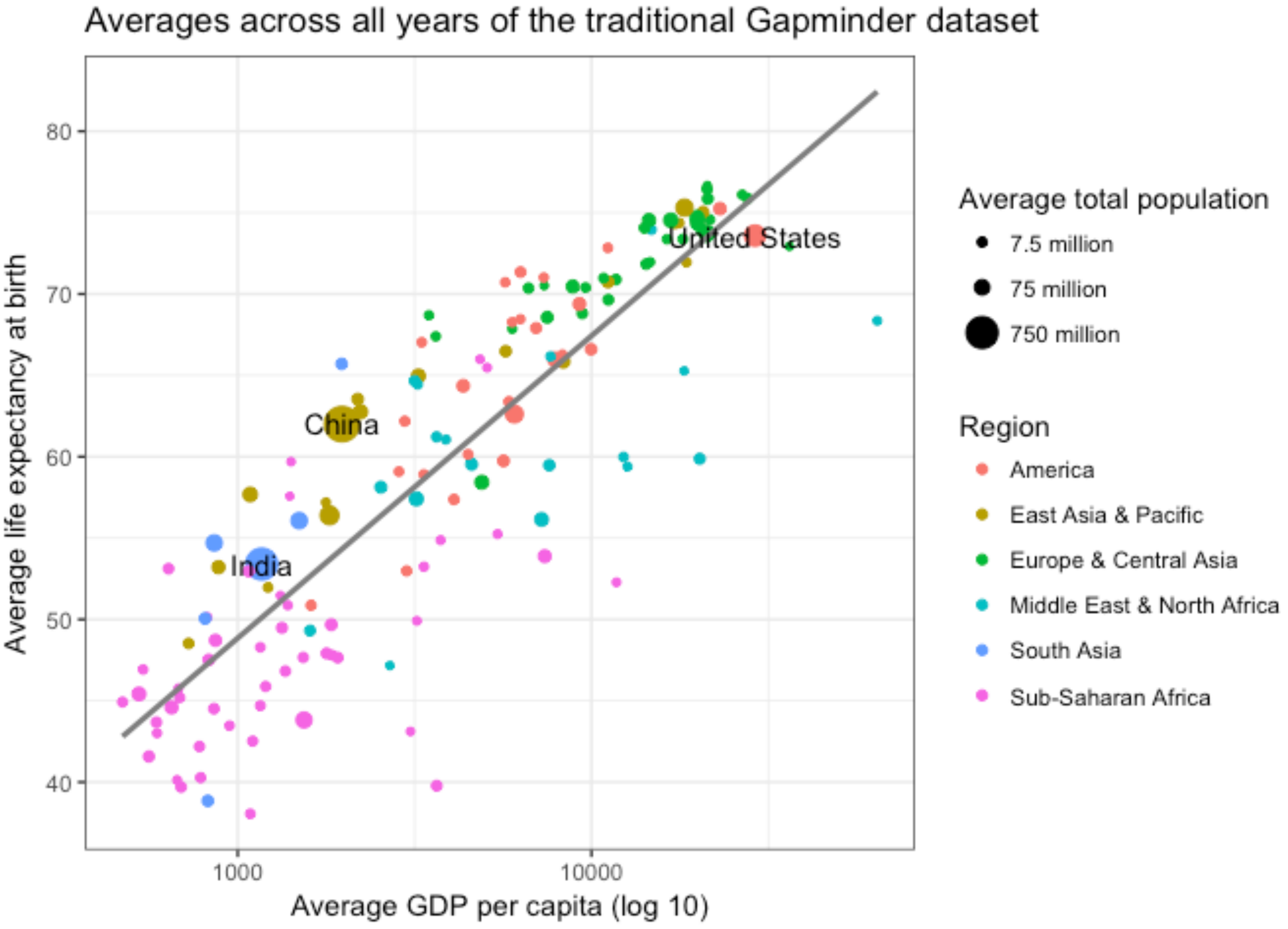
You will be adding a **scale**!

Changing the **look and feel** of the chart?

You will be adding or making changes to a **theme!**

Exercise 1: Gapminder Data

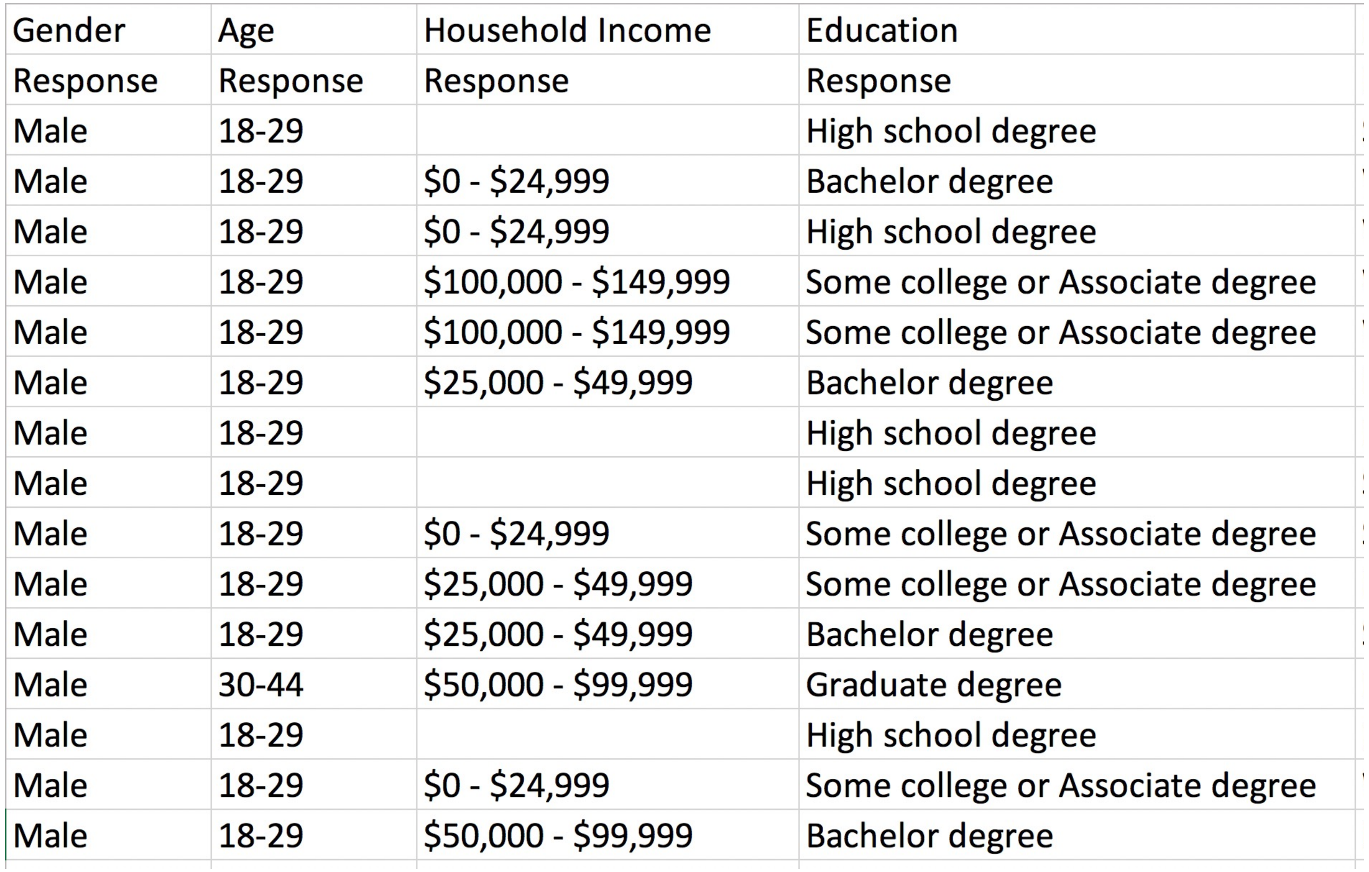
http://www.gapminder.org/


Averages across all years of the traditional Gapminder dataset
Average life expectancy at birth
80
70
60
50
40
1000
10000
Average GDP per capita (log 10)
China
India
United States
Average total population
● 7.5 million
● 75 million
● 750 million
Region
● America
● East Asia & Pacific
● Europe & Central Asia
● Middle East & North Africa
● South Asia
● Sub-Saharan Africa

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

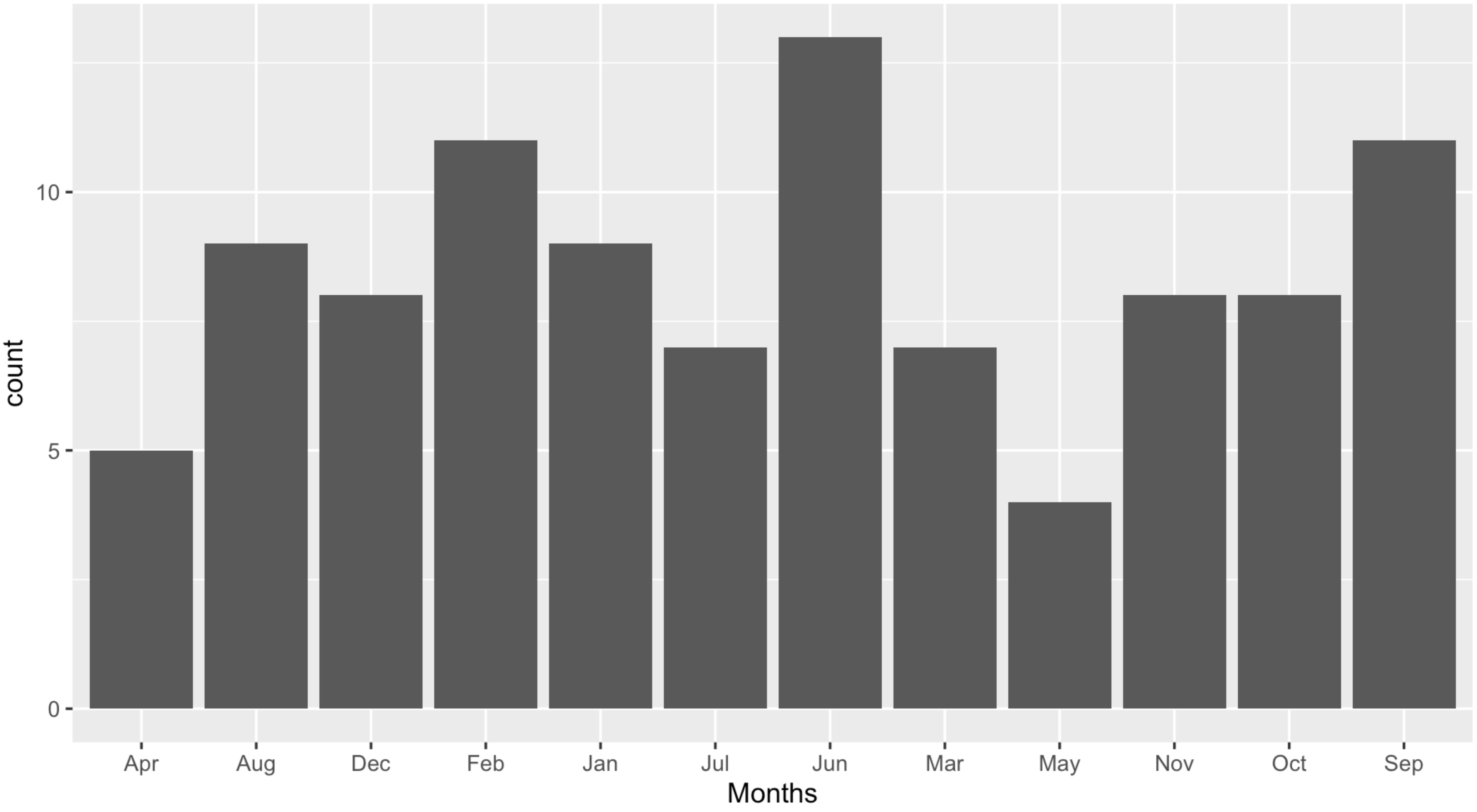
Problems with text variables: Ordering

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”

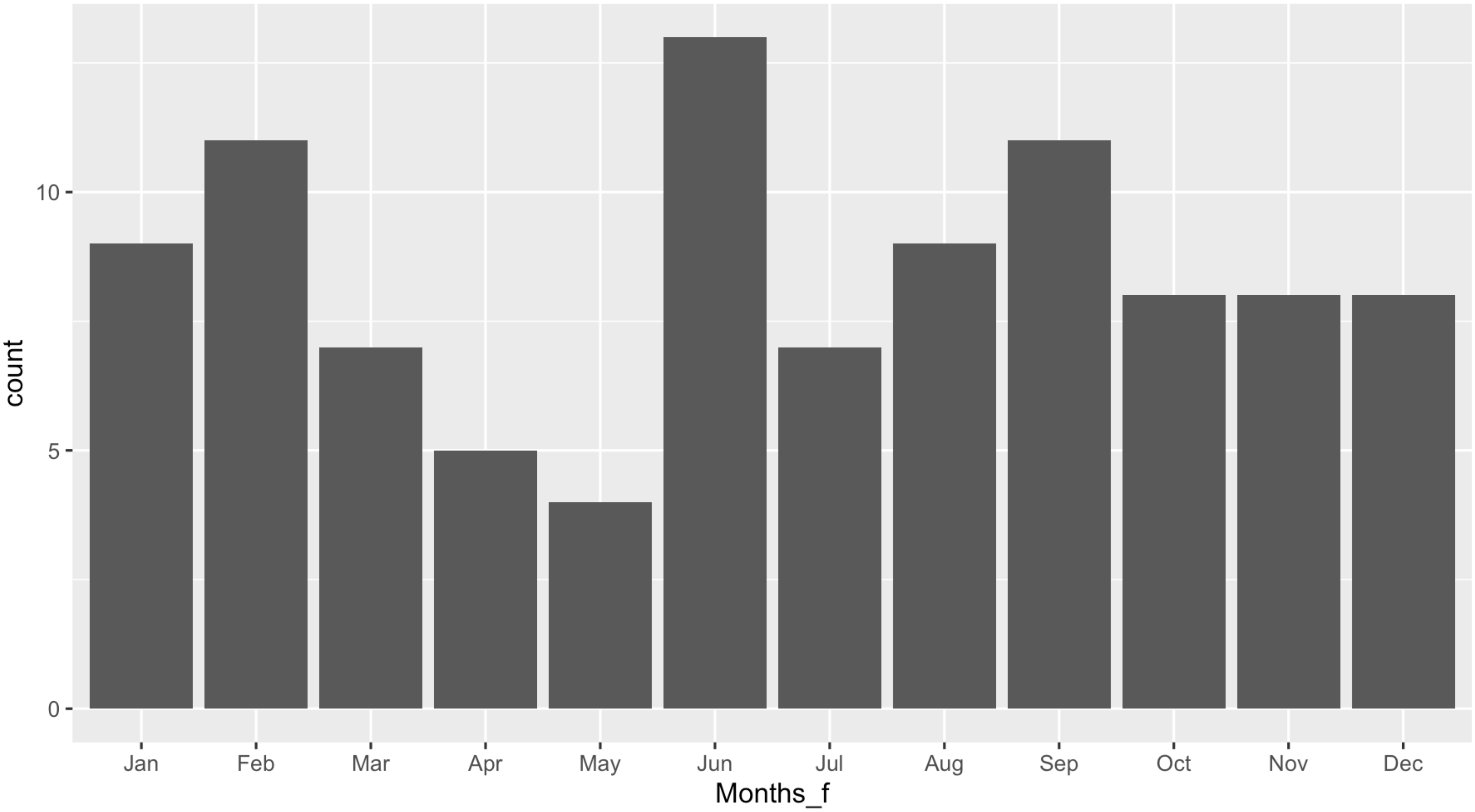
https://r4ds.had.co.nz/factors.html

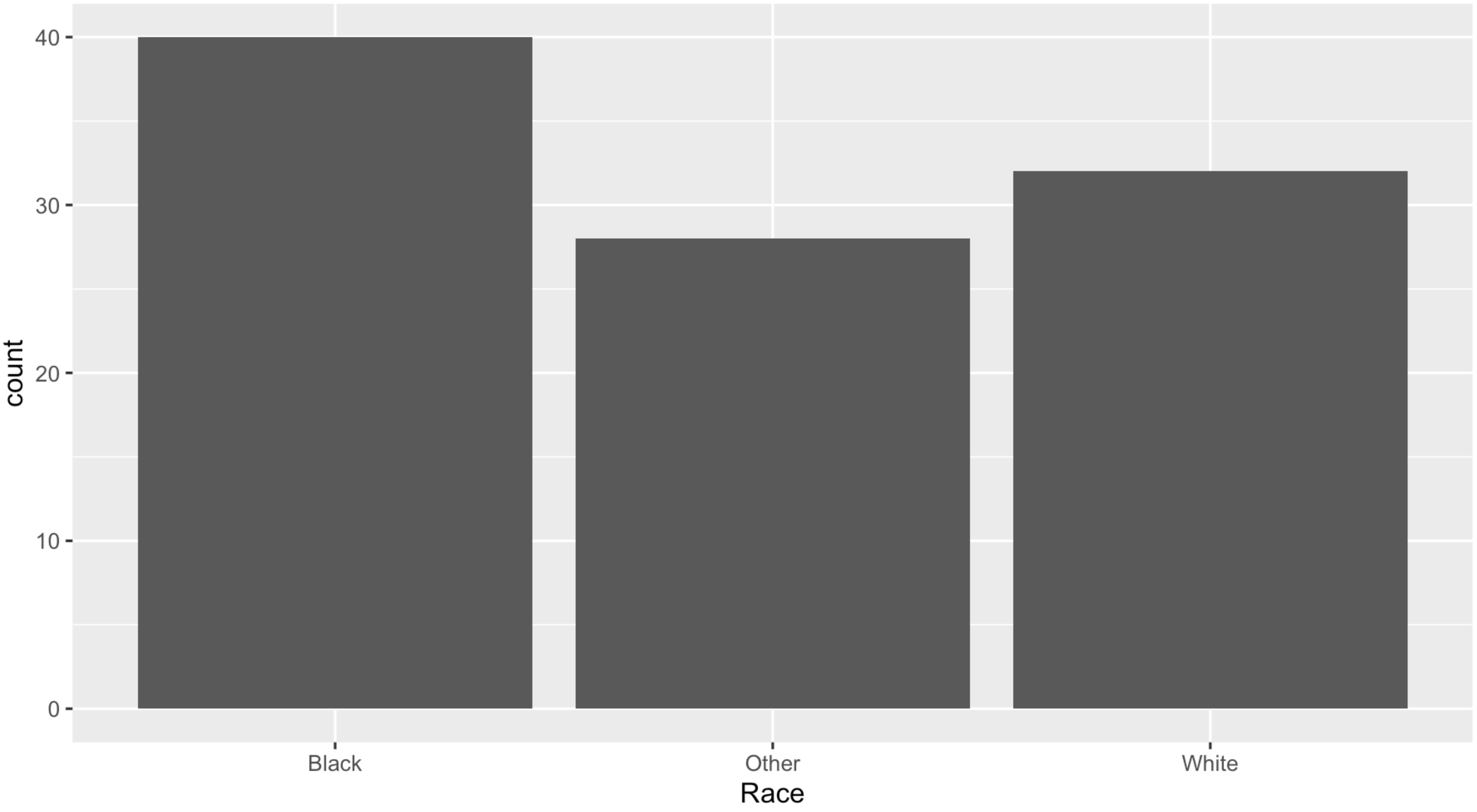
```
> x1 <- c("Dec", "Apr", "Jan",
"Mar")

> 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
```


count
10
5
0
Apr
Aug
Dec
Feb
Jan
Jul
Jun
Mar
May
Nov
Oct
Sep
Months

Order by meaning

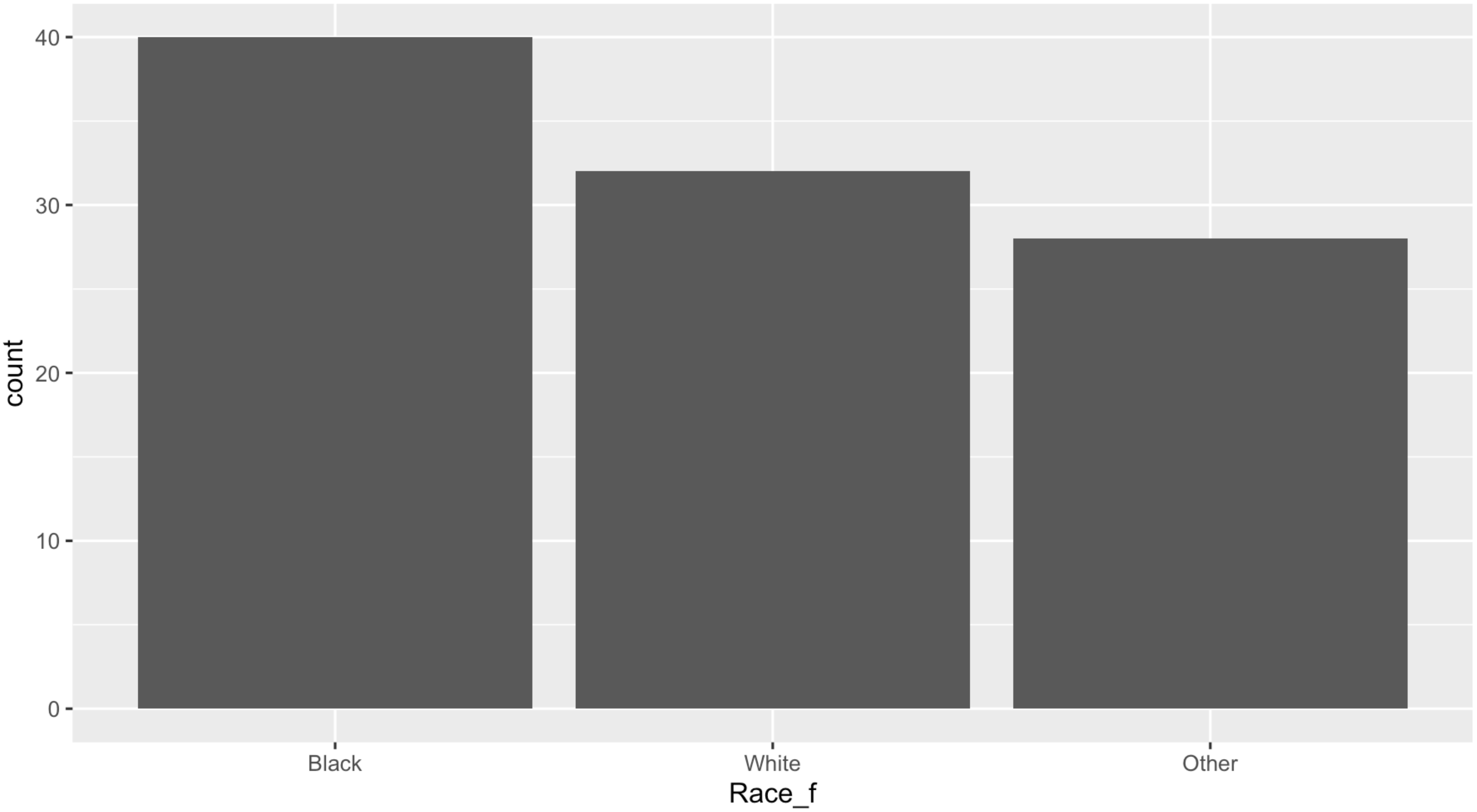
```
month_levels <- c( "Jan", "Feb", "Mar", "Apr",
"May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",
"Dec" )


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


count
10
5
0
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Months_f


40
30
20
10
0
count
Black
Other
White
Race

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()
```


40
30
20
10
0
count
Black
White
Other
Race_f

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)

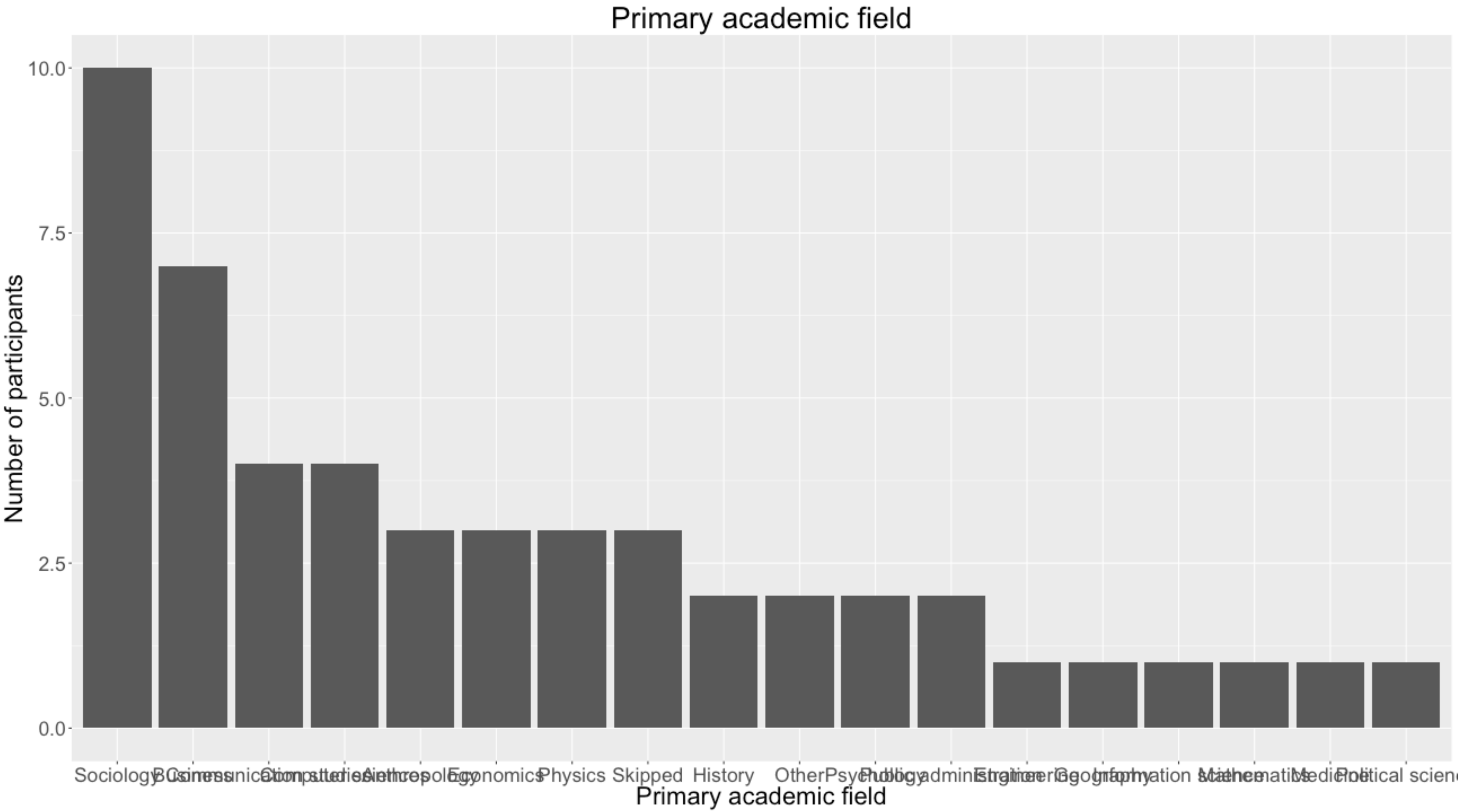
- Converts string variables to factors by default
- Can either:
  - Include `stringsAsFactors=FALSE`
  - Use `read_csv()` instead

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

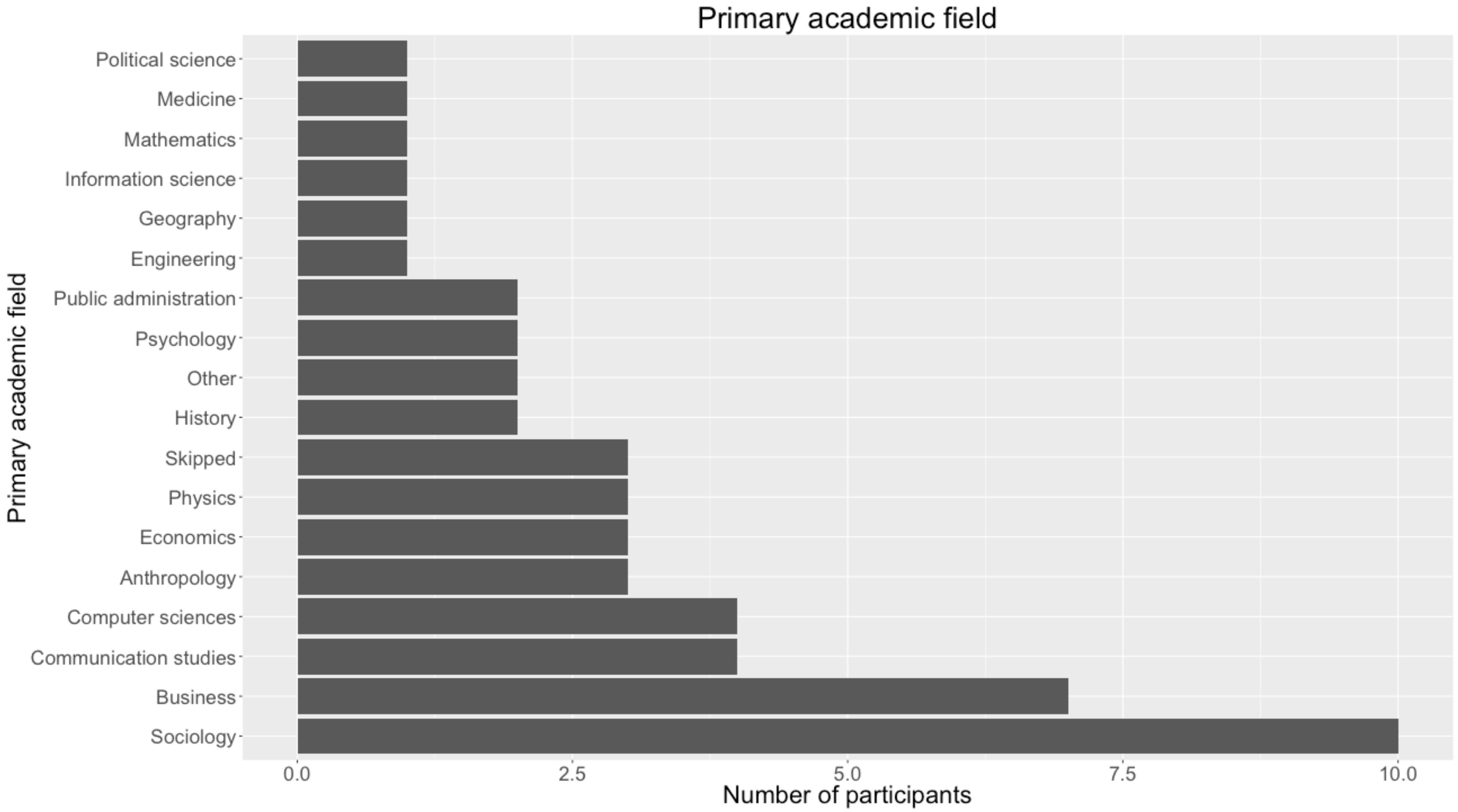

Primary academic field
Number of participants
10.0
7.5
5.0
2.5
0.0
Sociology
Business
Communication studies
Computer science
Anthropology
Economics
Physics
Skipped
History
Other
Psychology
Public administration
Engineering
Geography
Information science
Mathematics
Medicine
Political scien
Primary academic field

In ggplot2, have to flip the axes

```
+ coord_flip()
```

or

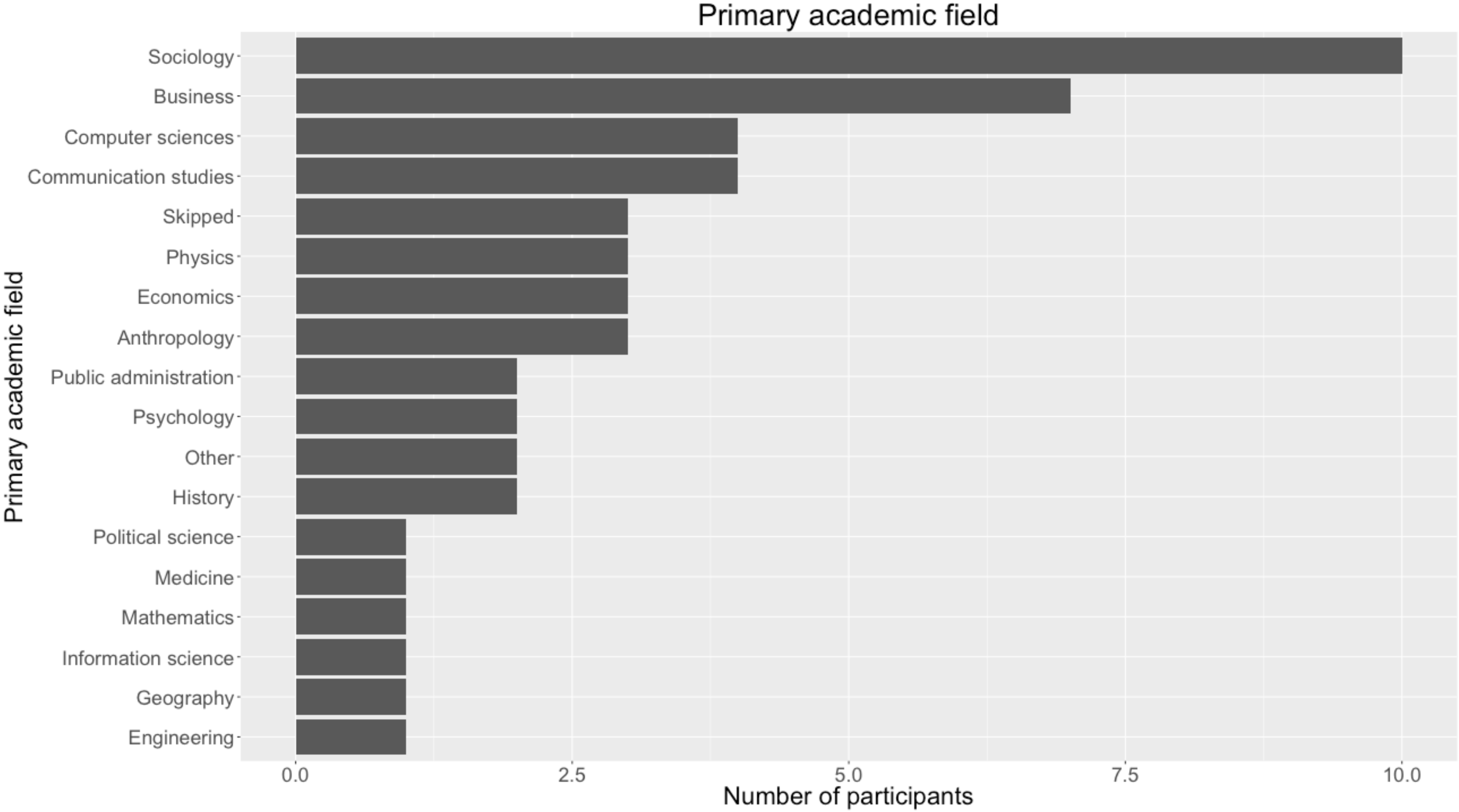
```
ggplot(df, aes(y=cat_variable)) +
    geom_bar()
```


Primary academic field
Political science
Medicine
Mathematics
Information science
Geography
Engineering
Public administration
Psychology
Other
History
Skipped
Physics
Economics
Anthropology
Computer sciences
Communication studies
Business
Sociology
Primary academic field
0.0
2.5
5.0
7.5
10.0
Number of participants

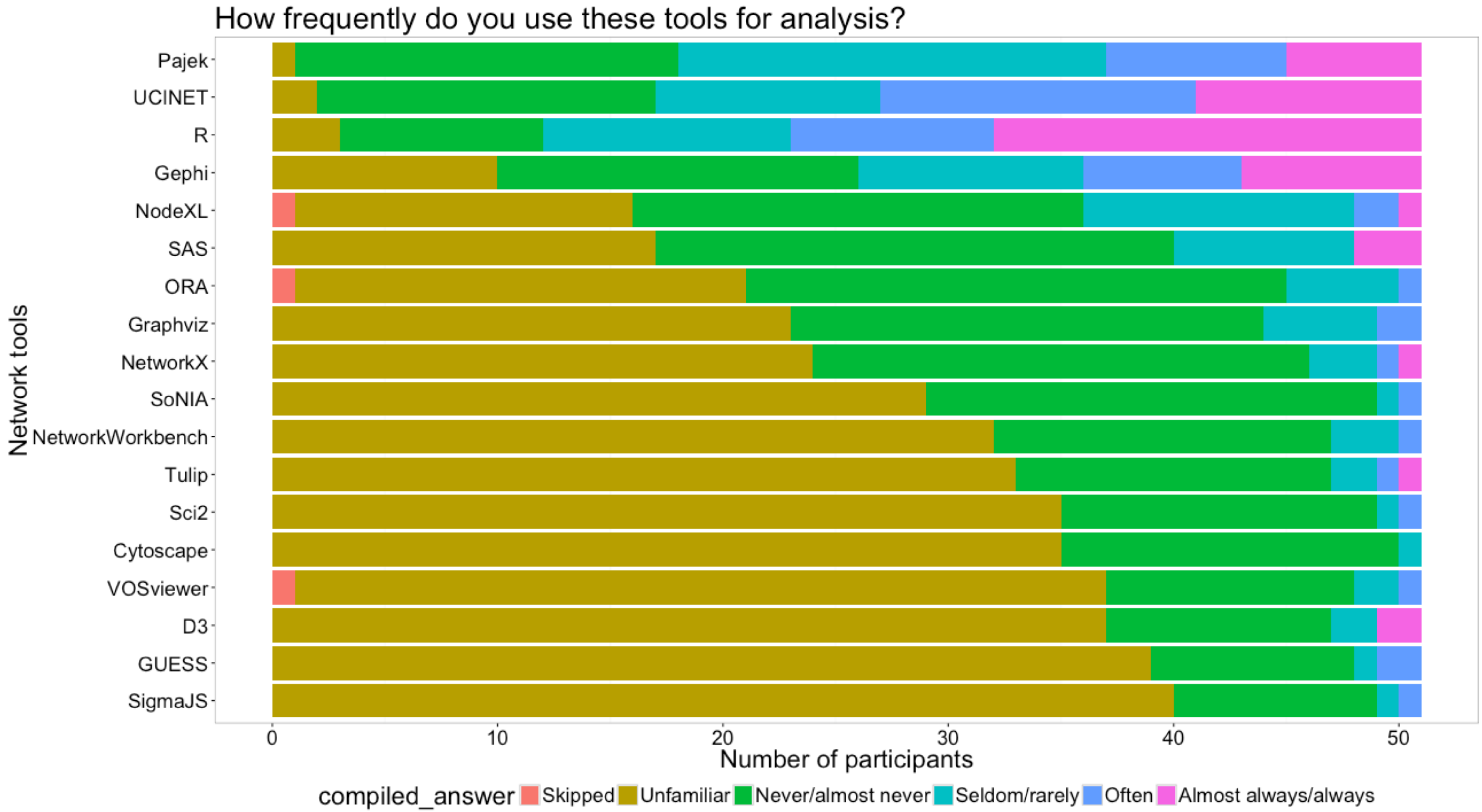
When you flip axes, you sort the other way

```
academic_field %>%
    as_factor() %>%
    fct_infreq() %>%
    fct_rev()
```

Have to reverse the order of the levels

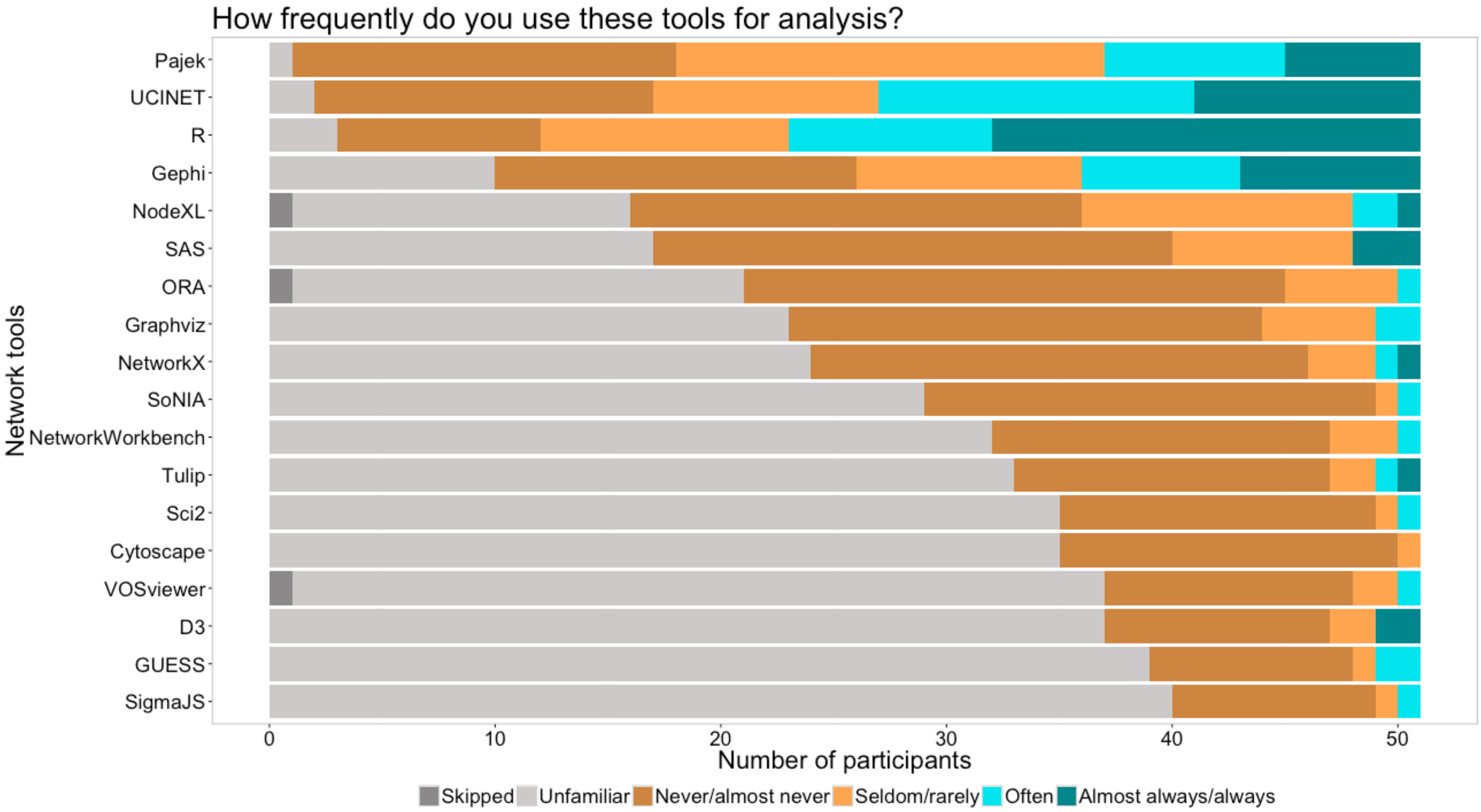

Primary academic field
Primary academic field
Sociology
Business
Computer sciences
Communication studies
Skipped
Physics
Economics
Anthropology
Public administration
Psychology
Other
History
Political science
Medicine
Mathematics
Information science
Geography
Engineering
0.0
2.5
5.0
7.5
10.0
Number of participants

Problems with text variables: Arbitrary colors


How frequently do you use these tools for analysis?
Network tools
Pajek
UCINET
R
Gephi
NodeXL
SAS
ORA
Graphviz
NetworkX
SoNIA
NetworkWorkbench
Tulip
Sci2
Cytoscape
VOSviewer
D3
GUESS
SigmaJS
0
10
20
30
40
50
Number of participants
compiled_answer
Skipped
Unfamiliar
Never/almost never
Seldom/rarely
Often
Almost always/always

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?
Network tools
Pajek
UCINET
R
Gephi
NodeXL
SAS
ORA
Graphviz
NetworkX
SoNIA
NetworkWorkbench
Tulip
Sci2
Cytoscape
VOSviewer
D3
GUESS
SigmaJS
0
10
20
30
40
50
Number of participants
Skipped
Unfamiliar
Never/almost never
Seldom/rarely
Often
Almost always/always

Sample Projects

Morning Break

Designing tools for data exploration

Supporting data exploration

Output

Picking the right visual elements

Input

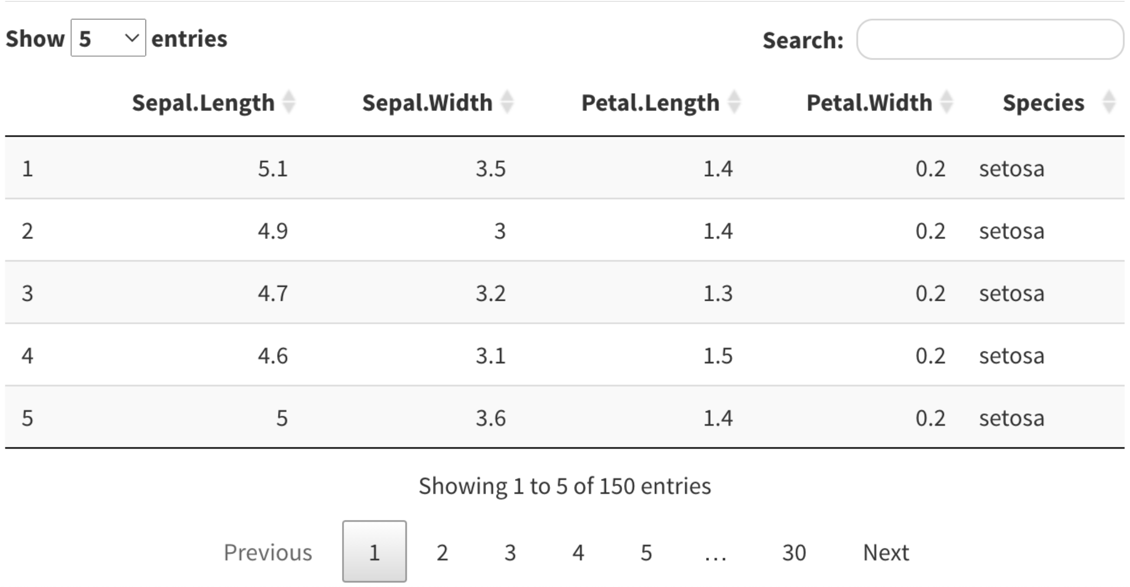
Giving users the right controls

Layout

Arranging everything in the right place

Interactive components

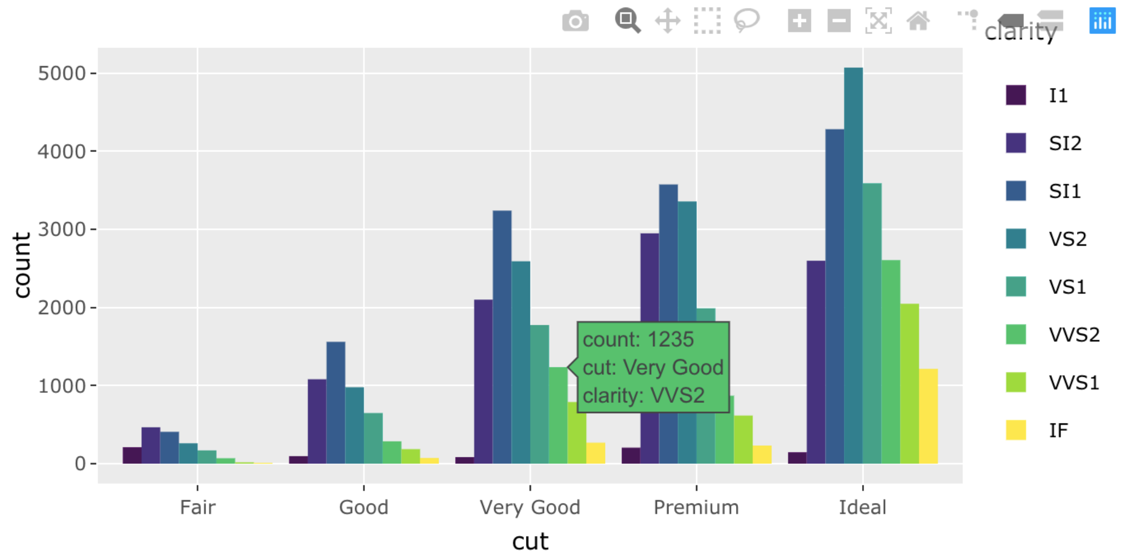
- Start with the basic info
- Show more or less on demand

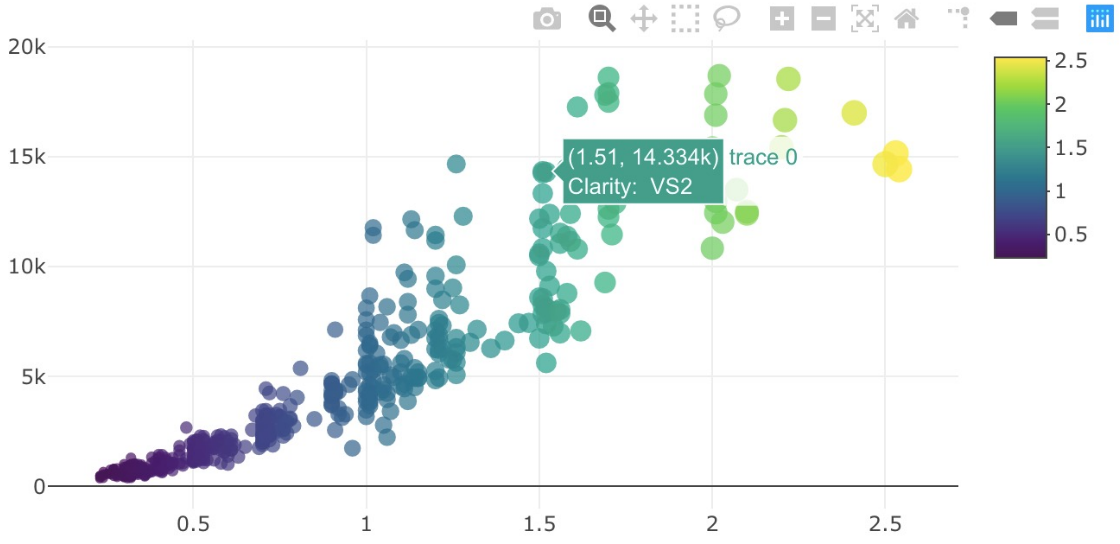


Show 5 entries Search:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5	3.6	1.4	0.2	setosa

Showing 1 to 5 of 150 entries



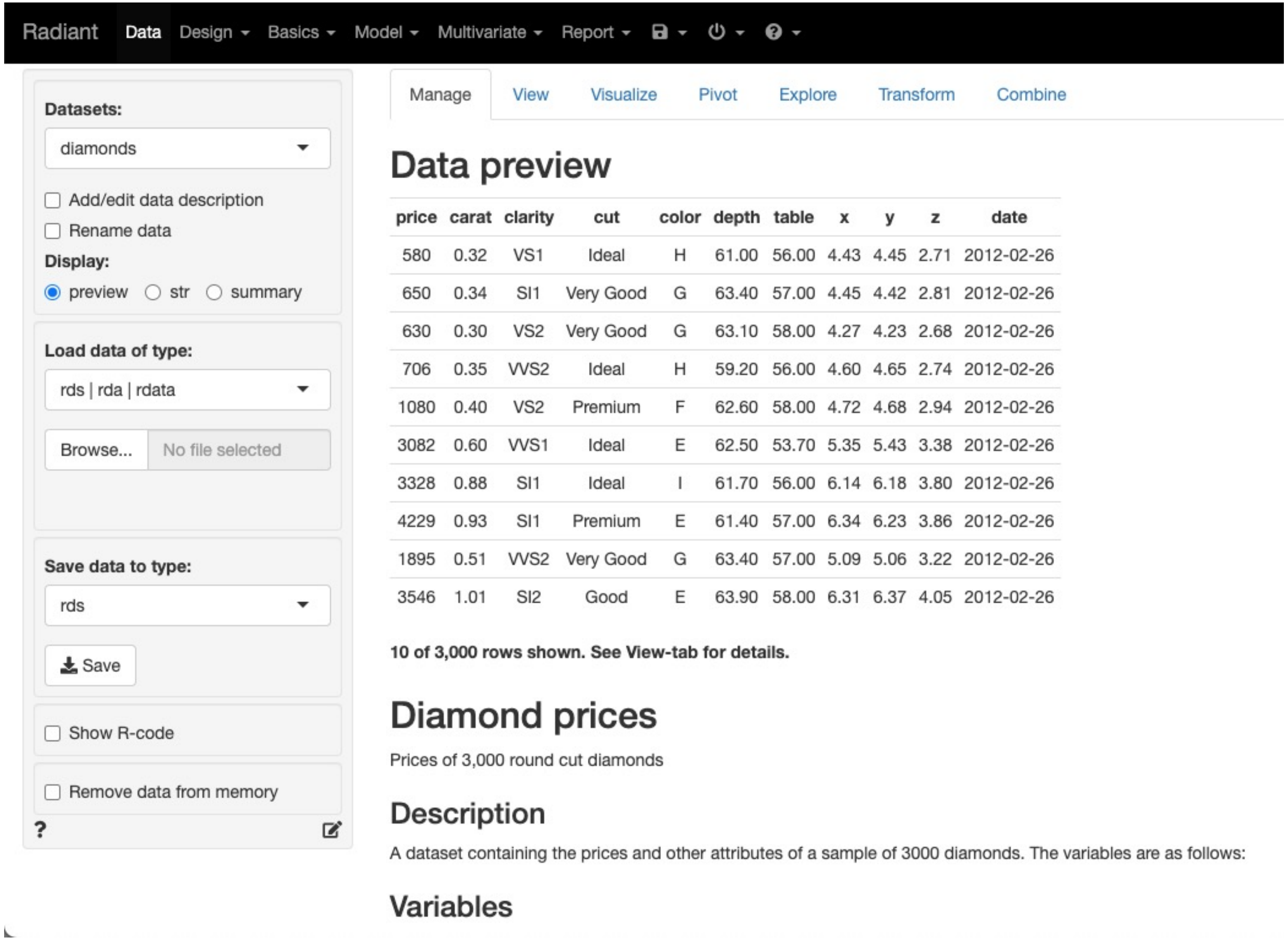


https://www.htmlwidgets.org/

http://gallery.htmlwidgets.org/

Responding to user input

- Generalized workflows
- Custom subsetting
- Changing parameters
- Personalizing output



Datasets: diamonds

□ Add/edit data description

□ Rename data

Display: ● preview ○ str ○ summary

Load data of type: rds | rda | rdata

Browse... No file selected

Save data to type: rds

Save

□ Show R-code

□ Remove data from memory

Data preview

price	carat	clarity	cut	color	depth	table	x	y	z	date
580	0.32	VS1	Ideal	H	61.00	56.00	4.43	4.45	2.71	2012-02-26
650	0.34	SI1	Very Good	G	63.40	57.00	4.45	4.42	2.81	2012-02-26
630	0.30	VS2	Very Good	G	63.10	58.00	4.27	4.23	2.68	2012-02-26
706	0.35	VVS2	Ideal	H	59.20	56.00	4.60	4.65	2.74	2012-02-26
1080	0.40	VS2	Premium	F	62.60	58.00	4.72	4.68	2.94	2012-02-26
3082	0.60	VVS1	Ideal	E	62.50	53.70	5.35	5.43	3.38	2012-02-26
3328	0.88	SI1	Ideal	I	61.70	56.00	6.14	6.18	3.80	2012-02-26
4229	0.93	SI1	Premium	E	61.40	57.00	6.34	6.23	3.86	2012-02-26
1895	0.51	VVS2	Very Good	G	63.40	57.00	5.09	5.06	3.22	2012-02-26
3546	1.01	SI2	Good	E	63.90	58.00	6.31	6.37	4.05	2012-02-26

10 of 3,000 rows shown. See View-tab for details.

Diamond prices

Prices of 3,000 round cut diamonds

Description

A dataset containing the prices and other attributes of a sample of 3000 diamonds. The variables are as follows:

Variables

https://shiny.rstudio.com/

https://shiny.rstudio.com/gallery/radiant.html

Interactive components

Why make charts interactive?

- Easier for data exploration
  - Drill-down to data subsets of interest
  - Details on demand
  - Customize look-and-feel of chart
- Can be more engaging for users

Visual information seeking mantra

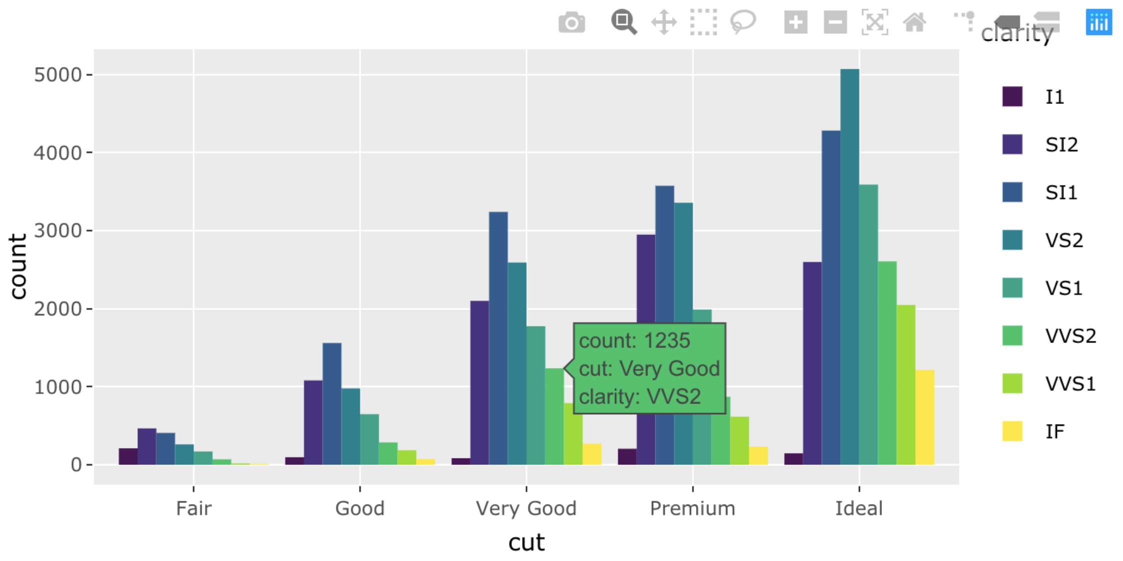
Overview first,
zoom and filter,
then details-on-demand

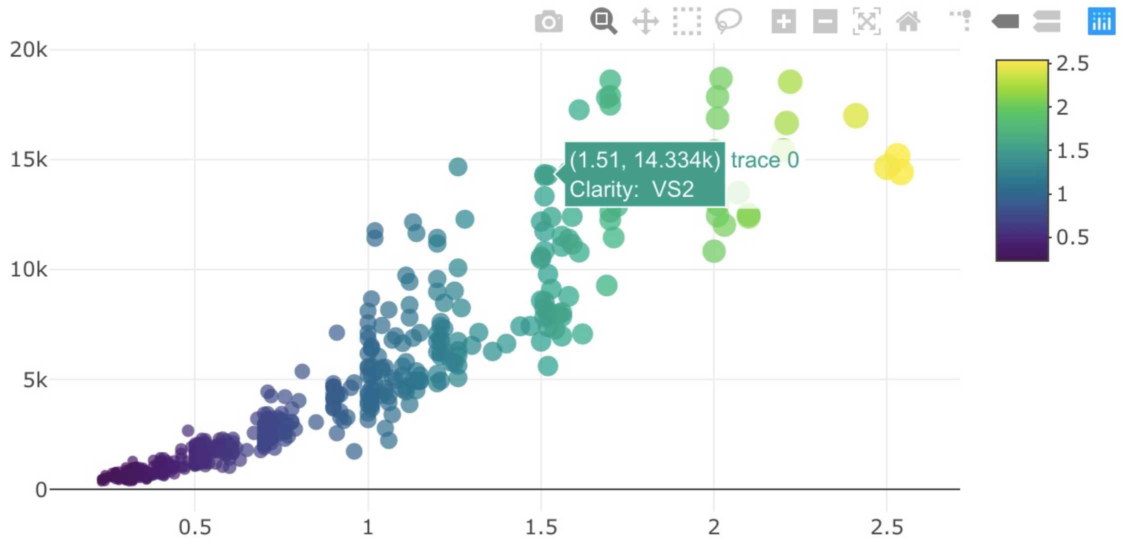
Shneiderman, B. (1996). The eyes have it: A task by data type taxonomy for information visualization.
In *VL '96 Proceedings of the 1996 IEEE Symposium on Visual Languages.*

Interactivity in R Markdown

- R Markdown gets compiled into HTML
- Some R packages can create interactive elements by converting R output to HTML/JavaScript code in the final document
- We will use the **plotly** package to create interactive charts

http://www.htmlwidgets.org/





Other interactive chart packages

- ggiraph for extending ggplot2 with interactive geoms
- rCharts for an R version of Polycharts, NVD3, and MorrisJS
- rBokeh for an R version of Bokeh
- altair for an R version of Altair
- leaflet for interactive maps

Exercise 2: Make yesterday’s charts interactive

plotly

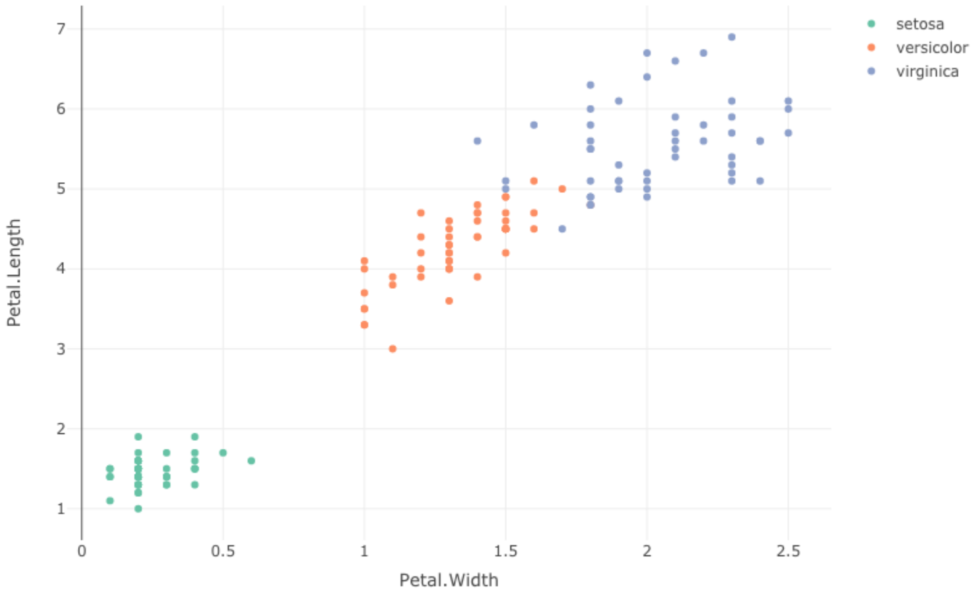
- Create plots that are interactive right away, either in R Markdown or in a website version
- Can either convert ggplot2 charts to plotly or build natively with plotly syntax

Basic plotly syntax

- Main plot function: plot_ly()
- Set the data: data = [data name]
- No aesthetics function, just list aesthetics pairings
- For each variable name, need "~" in front
- Default plot type is scatter; for others, add: type = "[plot type]"

Plotly for R Reference

```
plot_ly(data = iris,
   x = ~Petal.Width,
   y = ~Petal.Length,
   color = ~Species,
   type="scatter")
```



Publishing interactive plotly charts

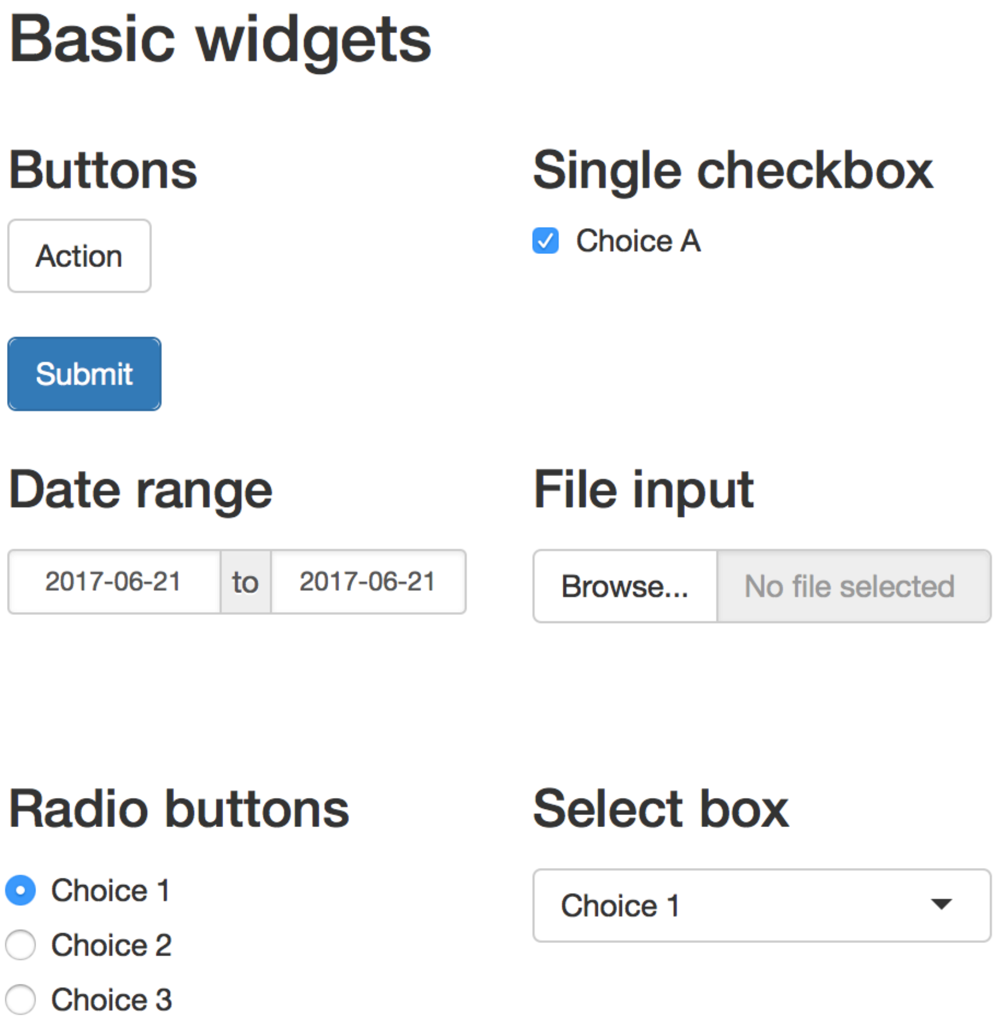
- Write R Markdown in RStudio
- Make sure “output” at top is “html_document”
- Use knitr to knit to HTML
- Publish HTML to:
  - RPubs (click the “Publish” button in RStudio)
  - GitHub (setup a GitHub Pages repository and add the HTML files)
  - Any website you already have that can publish HTML

Lunch

Responding to user input

Input controls to guide exploration

- For more complex data exploration, you may need input from the user
- Input controls can gather different kinds of information from the user, from free text to buttons to date ranges
- Simple input processing can happen within a standalone website, but for complex data processing, the input may need to feed back into a real R calculation (Shiny)



Basic widgets

Buttons

Action

Submit

Single checkbox

☑ Choice A

Date range

2017-06-21 to 2017-06-21

File input

Browse... No file selected

Radio buttons

● Choice 1

○ Choice 2

○ Choice 3

Select box

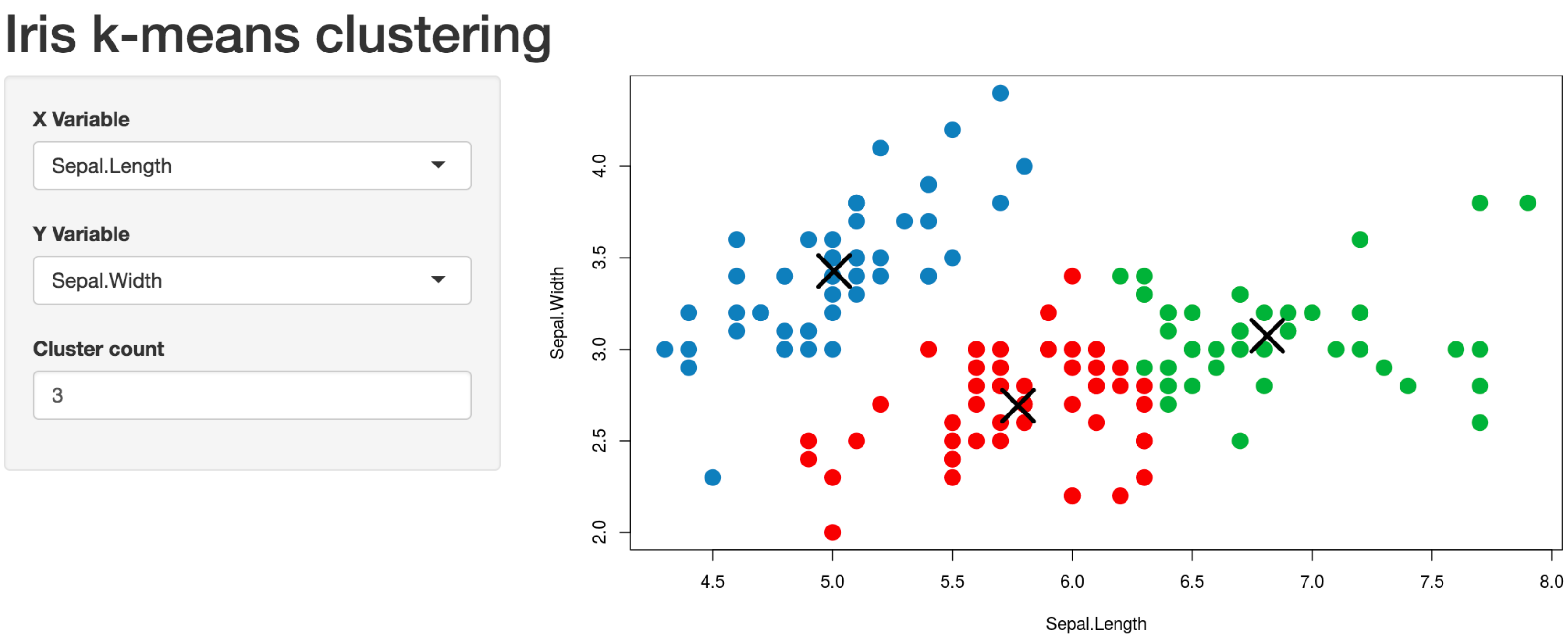
Choice 1 ▾

https://rstudio.github.io/crosstalk/ https://shiny.rstudio.com/

Shiny

What is Shiny?

An interactive interface onto an R program
(requires a special server to publish)



Iris k-means clustering

X Variable: Sepal.Length

Y Variable: Sepal.Width

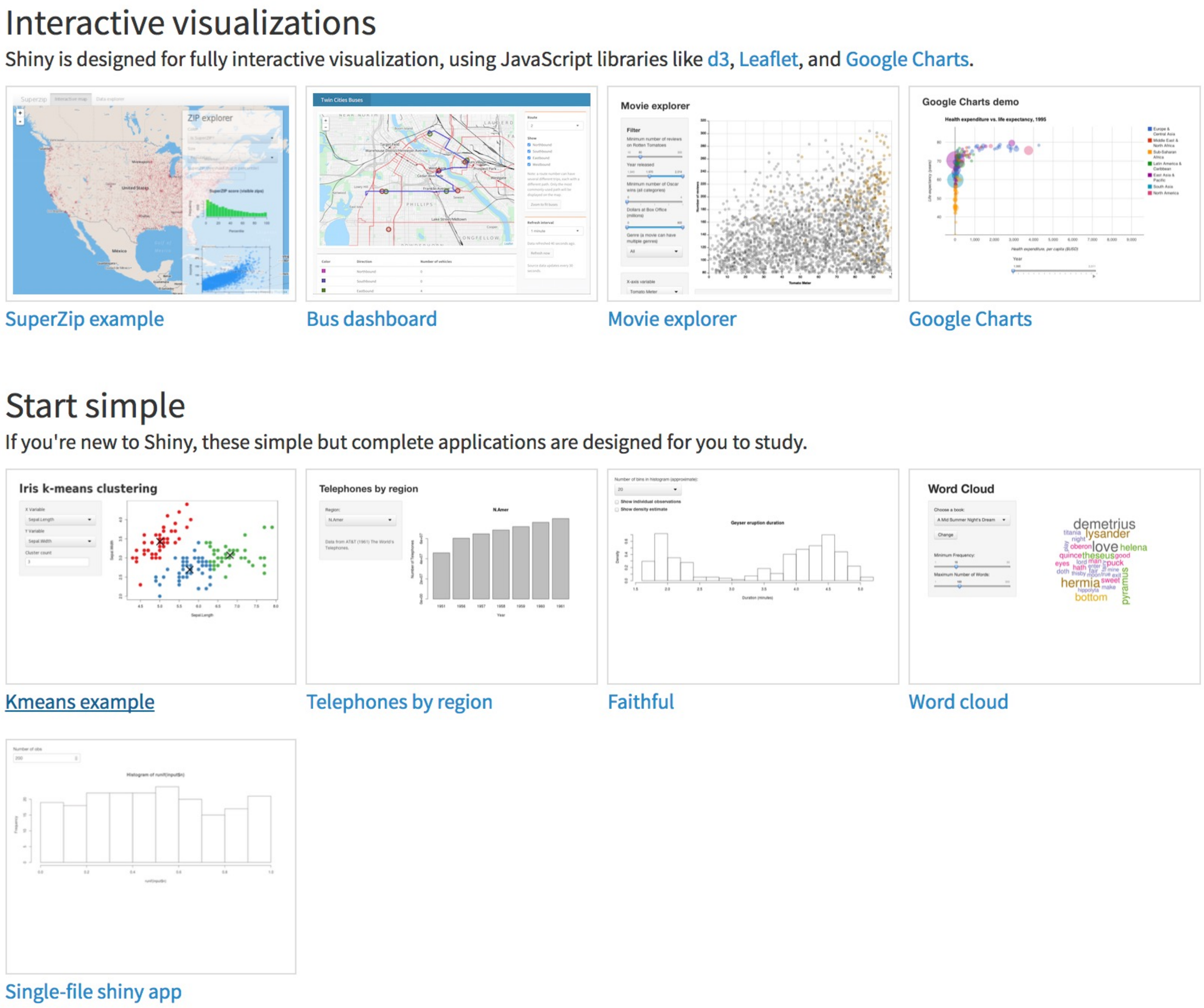
Cluster count: 3

http://shiny.rstudio.com/

How can you use Shiny for visualization?

- Use Shiny to control some kind of simulation interactively, then visualize the results
- Use Shiny to change components within the chart (e.g., switch the mappings)
- Use Shiny to filter data to subsets to highlight patterns
- Change type of regression, plot results

Shiny examples



Interactive visualizations

Shiny is designed for fully interactive visualization, using JavaScript libraries like d3, Leaflet, and Google Charts.

SuperZip example

Bus dashboard

Movie explorer

Google Charts

Start simple

If you're new to Shiny, these simple but complete applications are designed for you to study.

Kmeans example

Telephones by region

Faithful

Word cloud

Single-file shiny app

https://shiny.rstudio.com/gallery/

Shiny Apps

How do you build a Shiny app?

User Interface (UI)

the website people will see and interact with, including inputs and (placeholders for) outputs

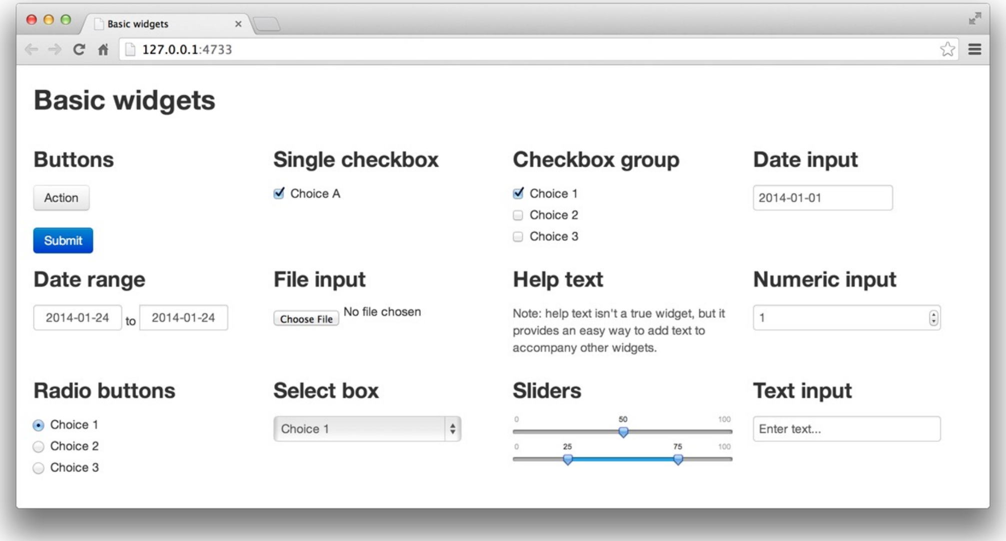
Server

takes values from the inputs, does some calculations, and fills in the outputs

Components

Some kind of **input widget**

(e.g., selectInput, sliderInput)



Basic widgets

Buttons

Action

Submit

Single checkbox

☑ Choice A

Checkbox group

☑ Choice 1
□ Choice 2
□ Choice 3

Date input

2014-01-01

Date range

2014-01-24 to 2014-01-24

File input

Choose File No file chosen

Help text

Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Numeric input

1

Radio buttons

● Choice 1
○ Choice 2
○ Choice 3

Select box

Choice 1

Sliders

Text input

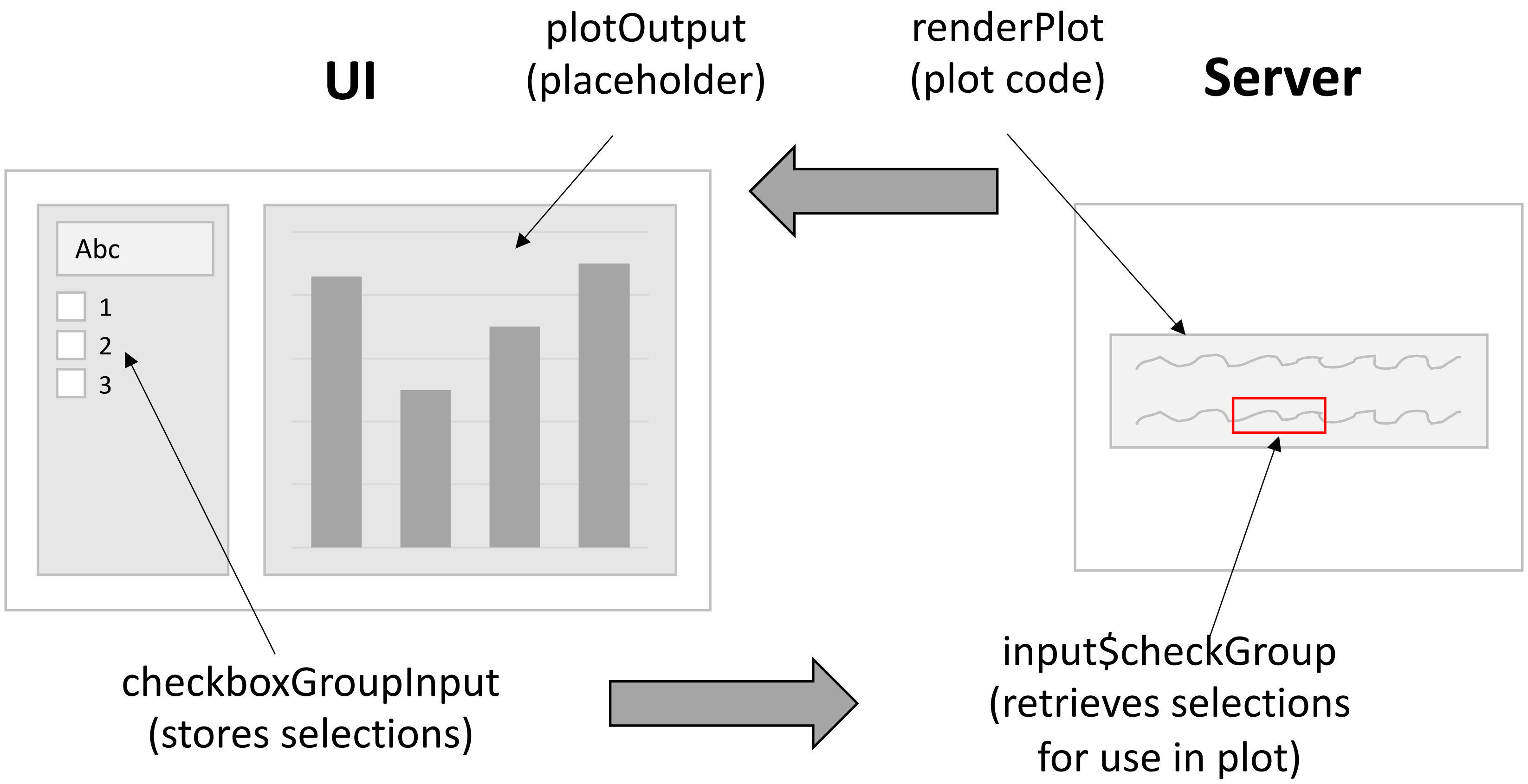
Enter text...

Some kind of **render object**

(e.g., renderPlot, renderTable)

renderPlot wraps around something like a ggplot() plot

a plot can read data from the input widget using input$inputId



UI

plotOutput
(placeholder)

renderPlot
(plot code)

Server

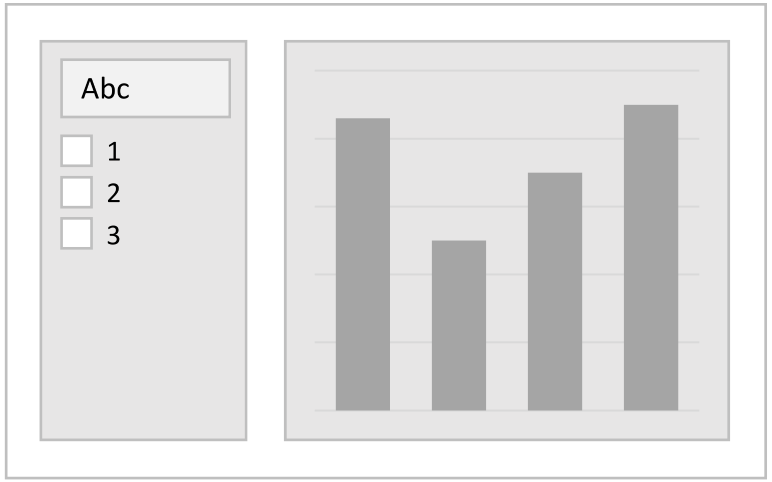
checkboxGroupInput
(stores selections)

input$checkGroup
(retrieves selections
for use in plot)

Step 1: Create the interface

What to put in the UI?

- Layout containers
- Input widgets
- Placeholders for reactive output
- Extra text/HTML elements



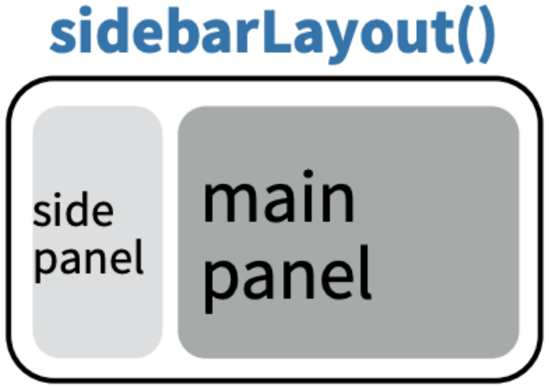
Page layout containers

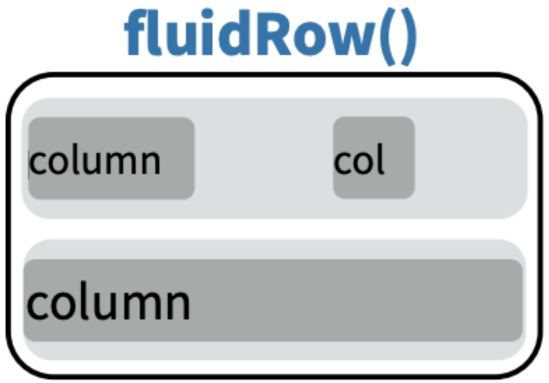
1. fluidPage
   - titlePanel
   - sidebarLayout
     - sidebarPanel
     - mainPanel
   - fluidRow
     - column
     - wellPanel
   - tabsetPanel
   - navlistPanel

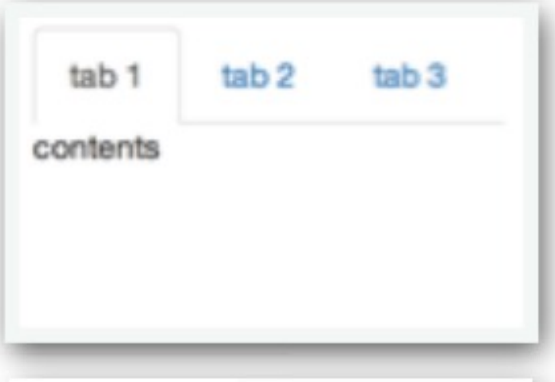
2. fixedPage
   - fixedRow
3. navbarPage
   - tabPanel
   - navbarMenu
     - tabPanel

https://shiny.rstudio.com/articles/layout-guide.html

https://www.rstudio.com/resources/cheatsheets/ (Shiny)

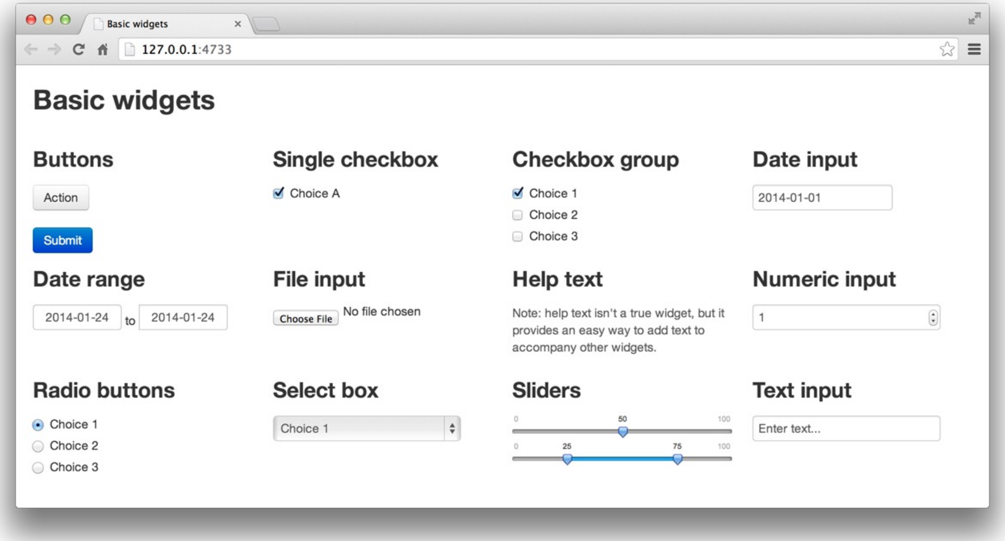






Input widgets

- Button
- Checkboxes
- Date, date range input
- File input
- Numeric input
- Radio buttons
- Drop-down (select) box
- Slider bar
- Text input
- Text



Date input: 2014-01-01

Date range: 2014-01-24 to 2014-01-24

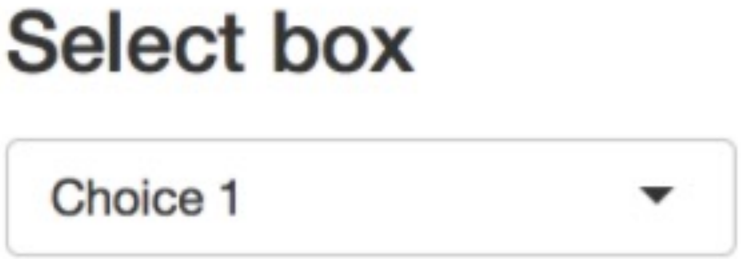
Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Numeric input: 1

Select box: Choice 1

http://shiny.rstudio.com/tutorial/written-tutorial/lesson3/
http://shiny.rstudio.com/gallery/widget-gallery.html

Anatomy of an input widget



Select box

Choice 1

- **inputId** for the widget (internal only)
- **label** (will be visible)
- Check documentation for other required arguments (e.g., selectInput requires choices)

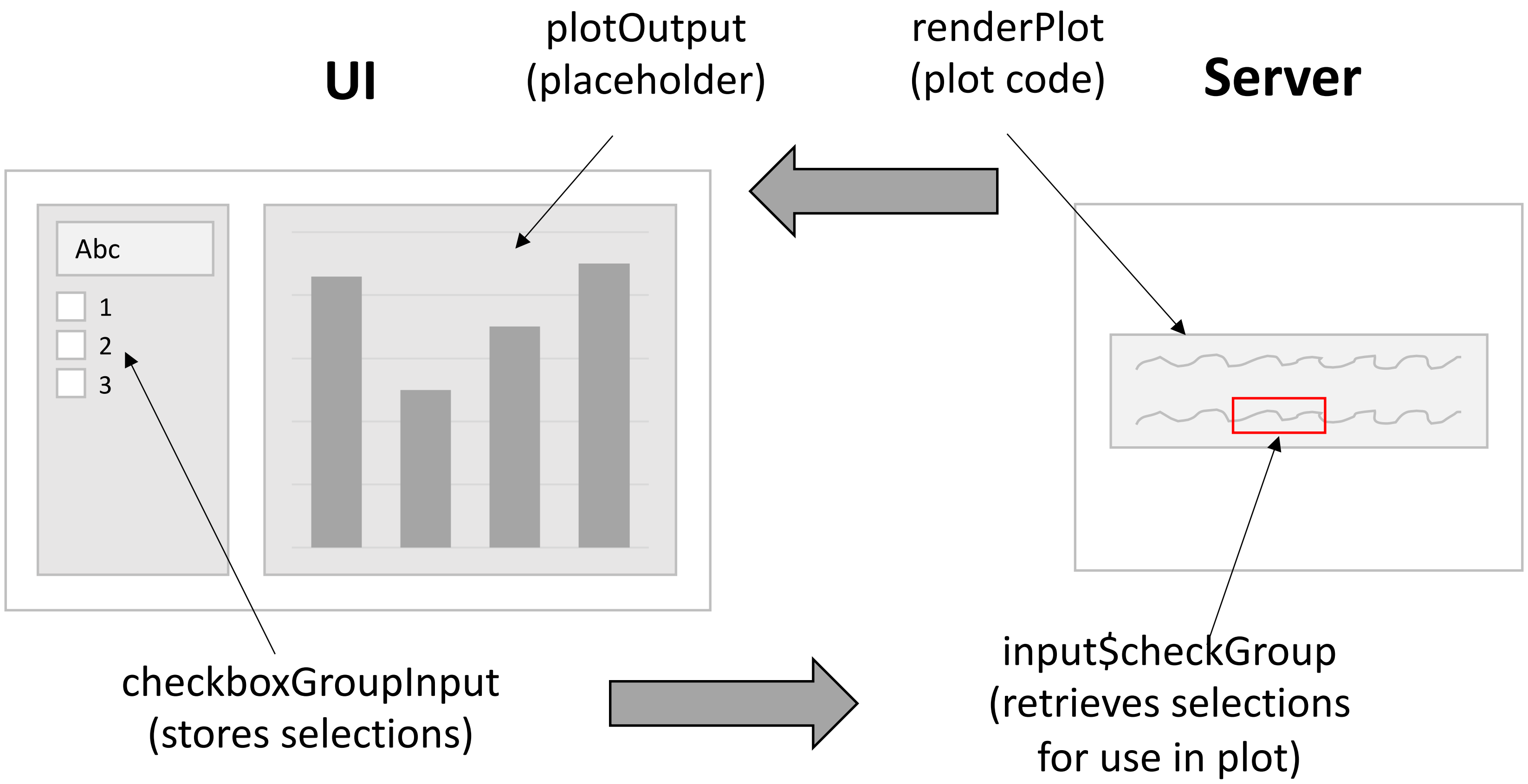
```
selectInput(inputId = "state",
            label = "Choose a state:",
            choices = c("NY", "NJ", "CT", "WA", "OR",
                        "CA", "MN", "WI", "IA")
```

Reactive output objects

UI	Server
htmlOutput	renderUI
imageOutput	renderImage
plotOutput	renderPlot
tableOutput	renderTable
textOutput	renderText
uiOutput	renderUI
verbatimTextOutput	renderPrint

http://shiny.rstudio.com/tutorial/written-tutorial/lesson4/

Step 2: Set up server to create dynamic objects



UI

plotOutput
(placeholder)

renderPlot
(plot code)

Server

checkboxGroupInput
(stores selections)

input$checkGroup
(retrieves selections
for use in plot)

What to put in the server

- R code
- Render objects with same names and types as the ones listed in UI
- Input objects with the same names as the control widgets

http://shiny.rstudio.com/tutorial/lesson4/

UI:

```
sliderInput(“slider1”, ...)

textOutput(“text1”)
```

Server:

```
output$text1 <- renderText({
                  input$slider1
                })
```

Step 3: Test

Running the app

Set options in RStudio:

- Window
- Viewer
- External

Sharing an app

- Shiny Apps
  http://www.shinyapps.io/
- Shiny Server (free – host on your own server)
  https://github.com/rstudio/shiny-server/blob/master/README.md
- Shiny Server Pro (fee)
  https://www.rstudio.com/products/shiny/shiny-server/

Exercise 3:
Explore the default Shiny app

Create a new app

- File → New File → Shiny Web App…
- Set a name
- Use “Single File” application type
- Look through the code
- Click “Run App” to see the default app

Afternoon Break

Exercise 4:
App from scratch

Tips for building your first app

- Start with basic layout elements and static content, like plain text
- Add one output and connect it to something in the server (e.g., plotOutput/renderPlot), but don’t use input$ in the plot yet
- Now create a control and add input$ to the server code
- You can save individual components as variables and then just use the variable names in your layout, if it gets confusing

Shiny resources

- Shiny Gallery
- Shiny Tutorial
- Shiny Articles
- Shiny function reference
- Shinyapps.io
- RStudio::conf 2019 workshop: Introduction to Shiny and R Markdown
- Shiny in Production (slides), Shiny in Production (book)
- Interactive web-based data visualization with R, plotly, and shiny
- Accessing and responding to plotly events in shiny

Thanks for your time this week!

angela.zoss@duke.edu