INTRODUCTION TO DATA ANALYSIS IN R - DAY 1

Randi L. Garcia, PhD

DATIC Introduction to R Workshop

Session 1: June 7th and 8th

Session 2: June 21st and 22nd

SMITH COLLEGE

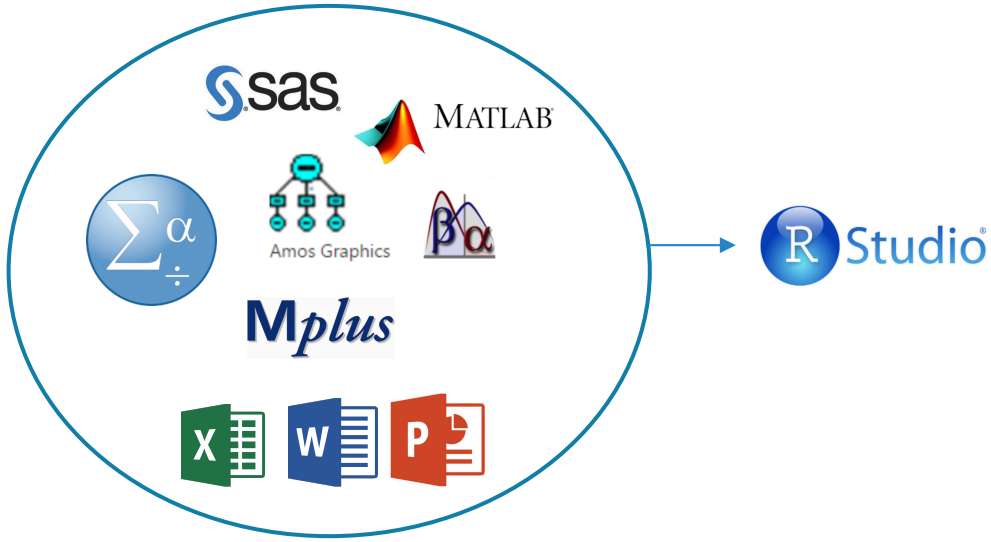
Introductions

- Me
  - Randi L. Garcia
    - Assistant Professor in Psychology and Statistical & Data Sciences at Smith College
  - Research interests
  - Data analysis software experiences

- You…
  - Who are you, where are you coming from?
  - What brings you here? What do you hope to get out of this workshop?

Why Learn to use R?

- Many of the reasons you mentioned…
  - High cost of SPSS, especially for students
  - Reproducibility
- My personal reasons:
  - It can do everything in one program
  - The R programming language versus SPSS syntax
  - Ability to create fully reproducible results, including automating results in manuscripts
  - Many teaching reasons



Schedule

DAY 1

- RStudio environment, packages, and R Markdown
- Making figures
- Data cleaning
- Descriptive stats, correlations, reliability, creating scale scores

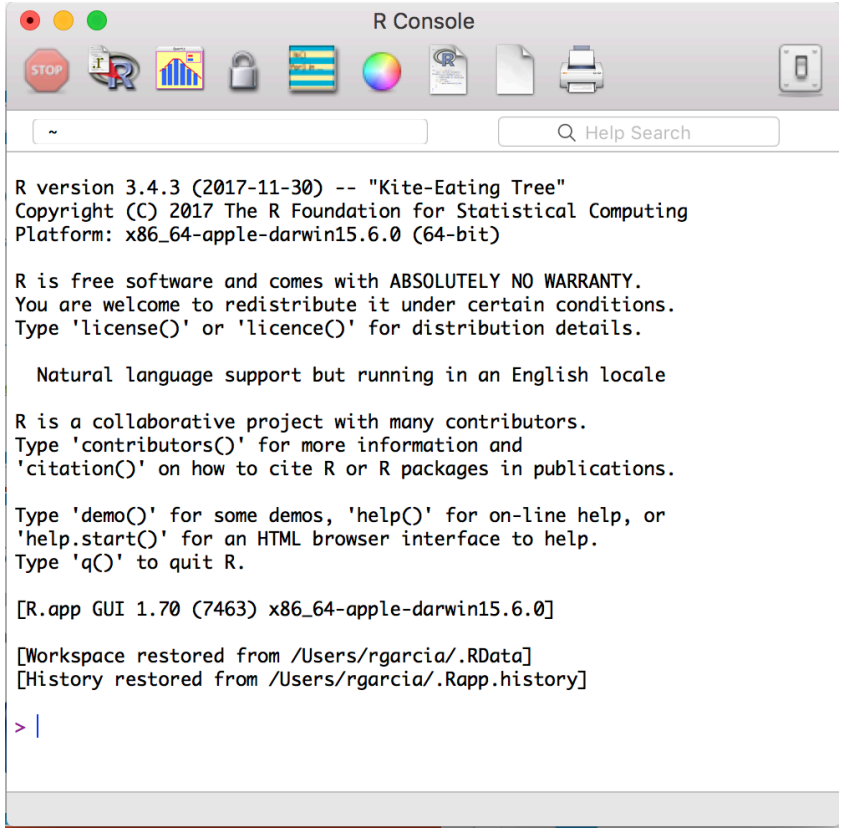
DAY 2

- T-test, ANOVA, and regression
- Preparing APA style manuscripts
- Exploratory Factor Analysis (EFA) and Confirmatory Factor Analysis (CFA)
- Path Analysis and Structural Equation Modeling (SEM)

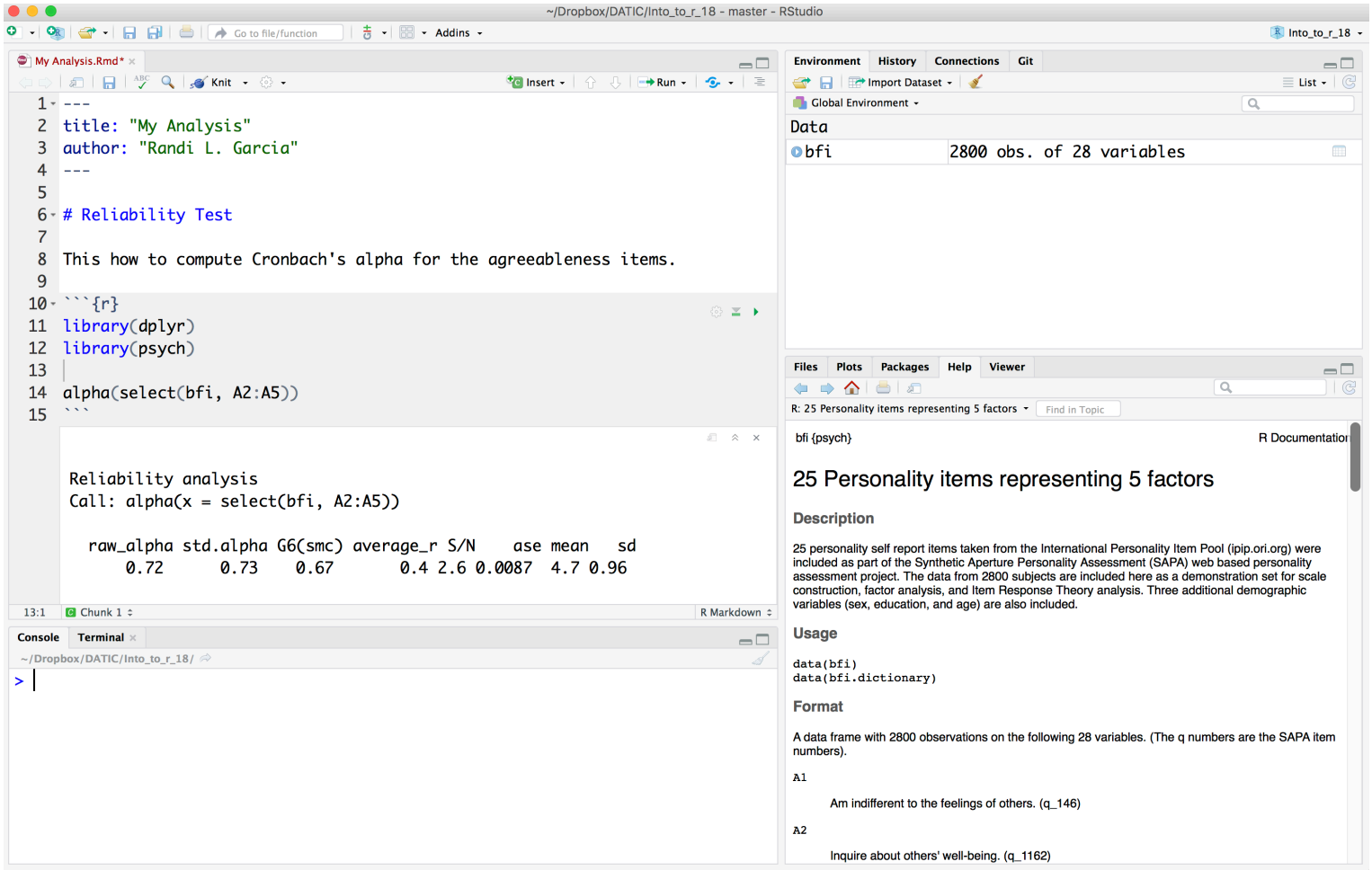
DAY 1

- RStudio environment, packages, and R Markdown
- Making figures
- Data cleaning
- Descriptive stats, correlations, reliability, creating scale scores

R and RStudio



R and RStudio



OPEN R STUDIO

Let's Use R Studio!

- Bookmark this website:

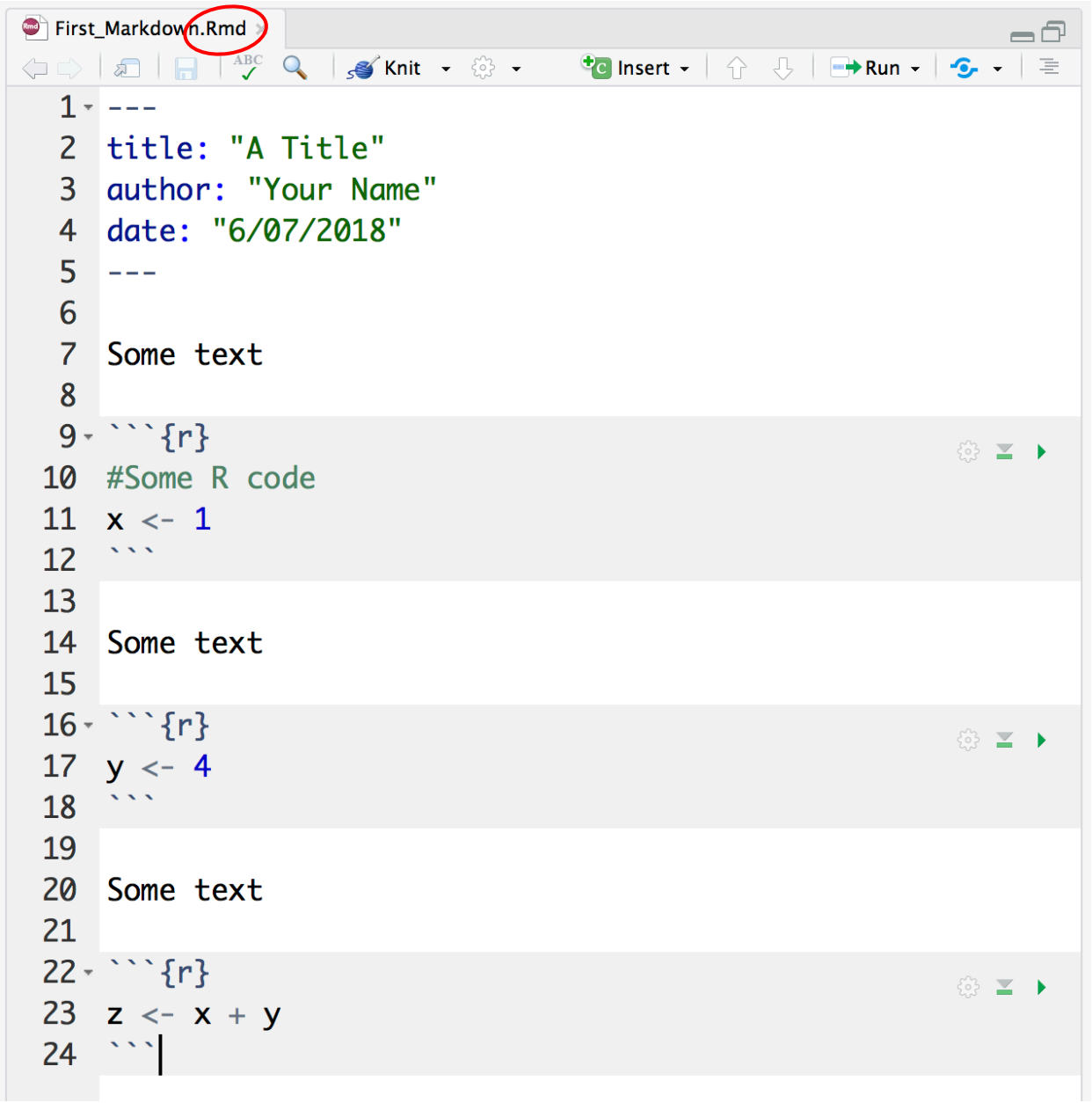
bit.ly/intro-r-website

- Download ALL materials, including R-code, here:

bit.ly/intro-r-materials

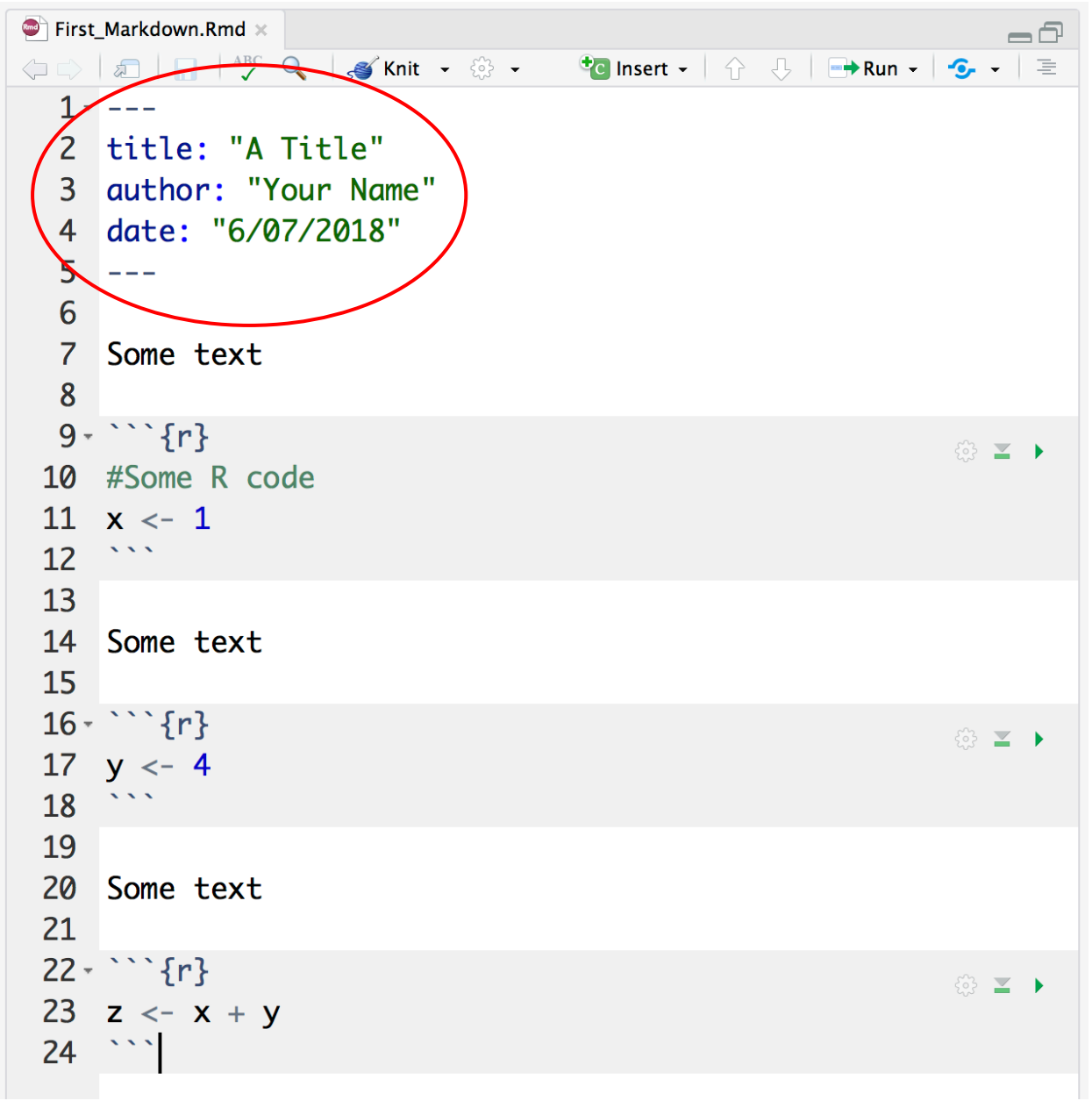
R Markdown is where your analyses live!

- A file of type ".Rmd"



R Markdown is where your analyses live!

- A file of type ".Rmd"
- Starts with some basic information in the "YAML header"



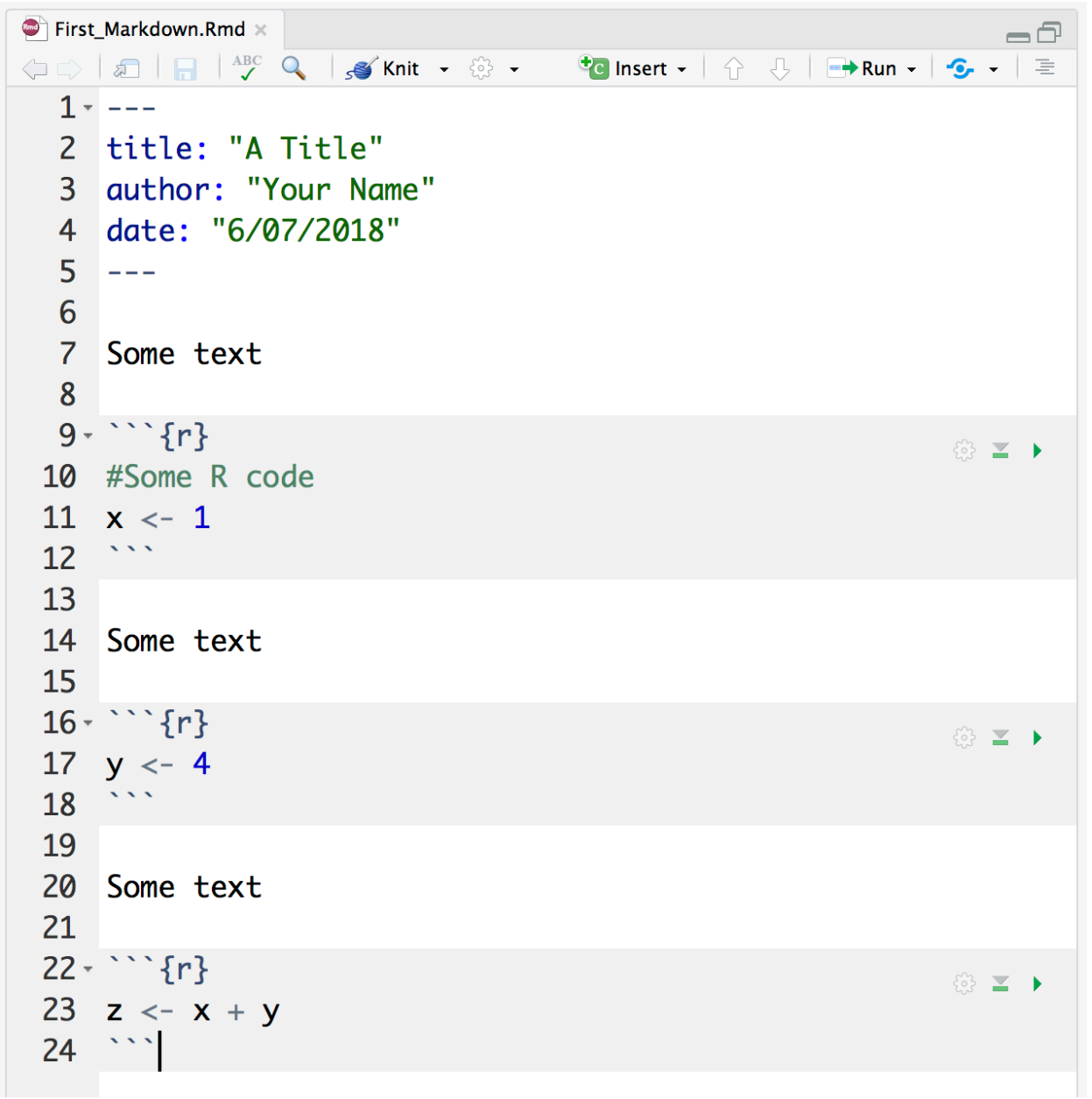
R Markdown is where your analyses live!

- A file of type ".Rmd"
- Starts with some basic information in the "YAML header"
- A series of text and "code chunks":

```
```{r}


```
```

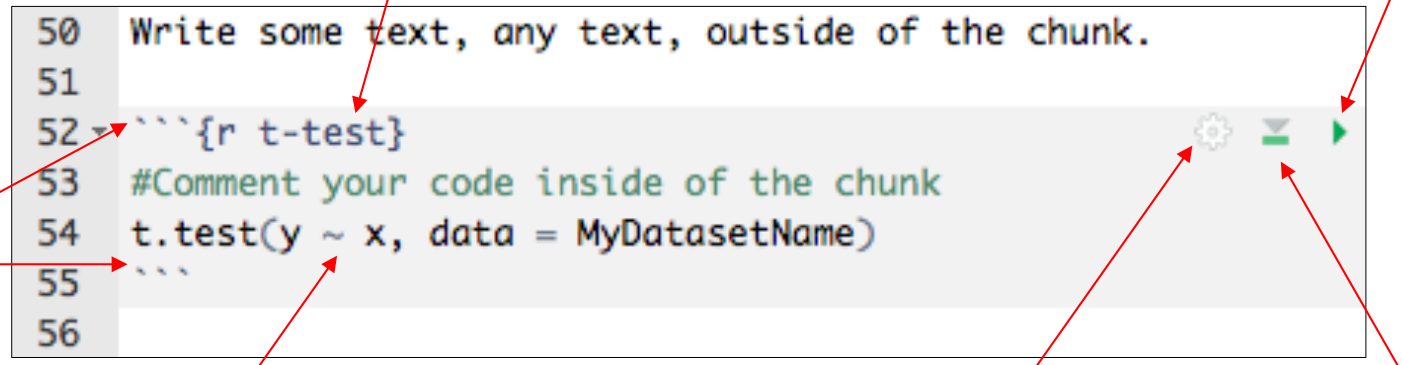
- We will need to install some stuff…



Anatomy of a Code Chunk

Run all of the code in this chunk

Giving your chunk a name helps find it later



"Bookends" to signify code is starting and ending

The R code goes between the bookends

Chunk options (more on that later)

Run all of code in the chunks above

R STUDIO

Intro_to_R.Rmd

packages_descriptive_stats.Rmd

Which R?

- There are >10,000 packages in R
- This can feel overwhelming for new users
  - To make matters worse, "R people" are opinionated about which packages are "best"
  - There is NO consensus! Eventually you'll be able to decide for yourself, for now, I'll decide for you…
- We are going to learn *some* of the **tidyverse** packages in this workshop
  - Hadley Wickham

Making Figures with ggplot2

- As with everything else, there are lots of ways to make figures in R
  - Base R
  - Lattice graphics
  - The ggplot2 package

```{r}
library(ggplot2)
```

- We'll be learning the ggplot2 package.
  - It makes beautiful visualizations
  - It's popular so there is a lot of help on the internet and companion packages
  - It works well with all of the tidyverse packages

Making Figures with ggplot2

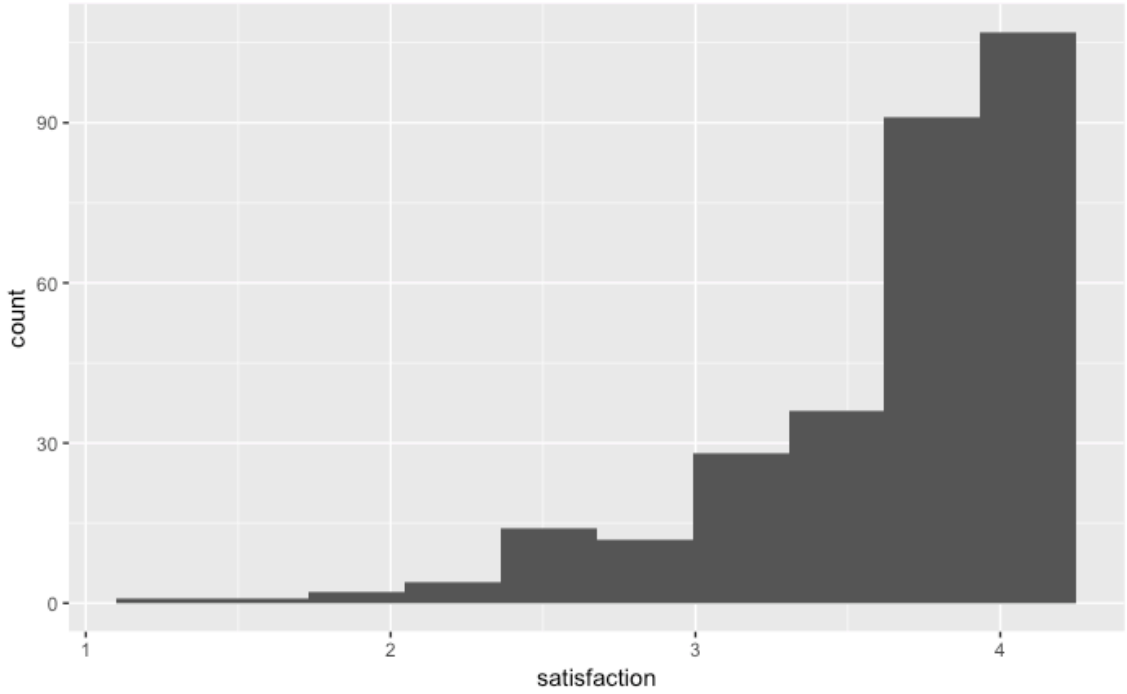
- The easiest figures are made with the `qplot()` function
  - The q stands for quick!

Guesses which kind of figure you want based on the variable(s) type

Customize it!

```
```{r}
qplot(satisfaction, data = acitelli, bins = 10)
```
```

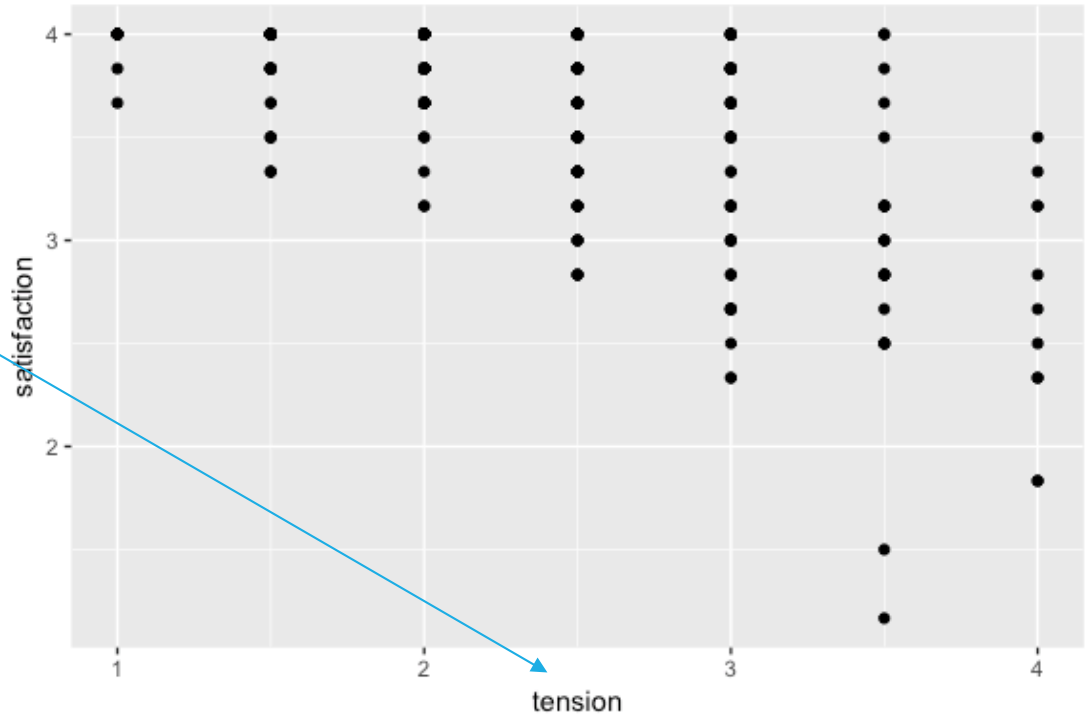
It needs to know the data, but no dollar signs!



Making Figures with ggplot2


```r
qplot(x = tension, y = satisfaction, data = acitelli)
```

- qplot: "Two numerical variables? Oh, you probably want a scatter plot…"



Making Figures with `ggplot2`

- The `qplot()` function is good for quick visualizations
  - Good for probably 80% of what you'd want to do while analyzing data

- But, you'll use the `ggplot()` function for anything more involved, like for making figures for publication

- The `ggplot2` packages uses the "**g**rammar of **g**raphics"

Making Figures with ggplot2

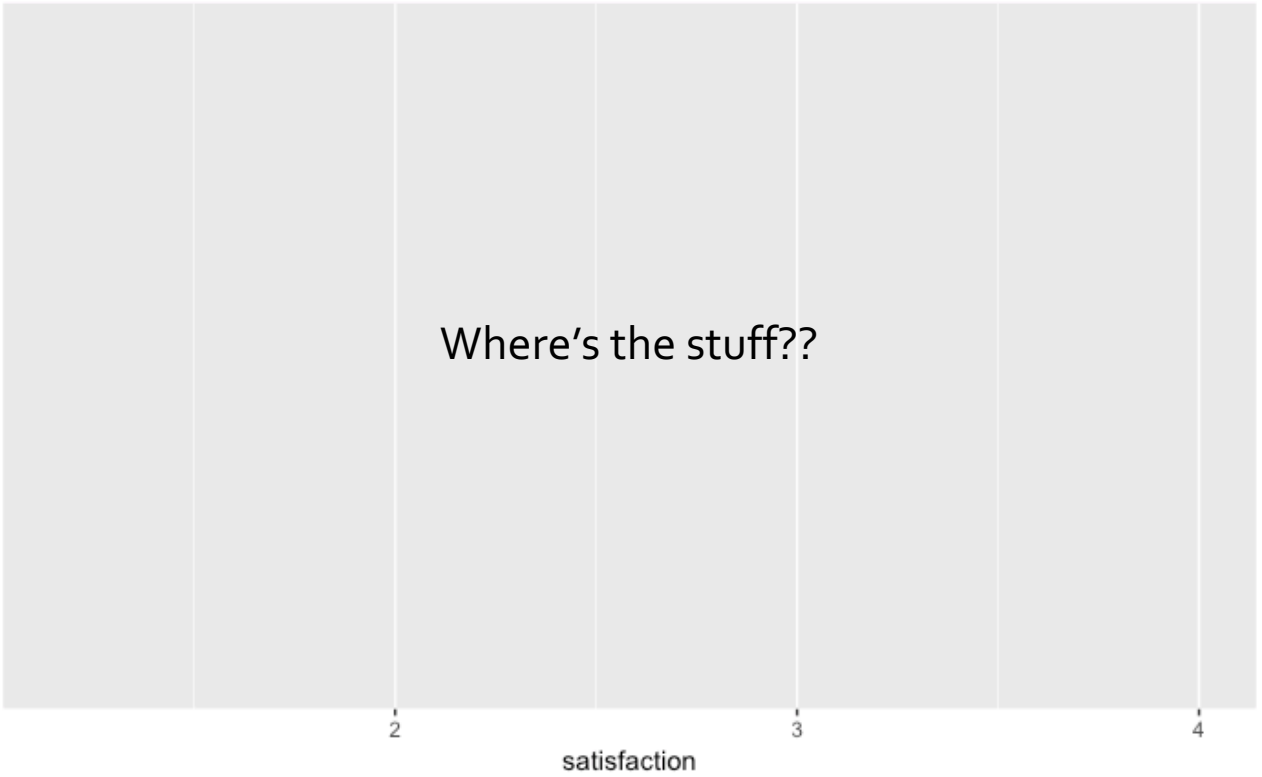
- We independently specify pieces of the graph using the "**g**rammar of **g**raphics"
- Building blocks:
  - **Data**
  - **Geometric objects** (the actual things we'll draw: points, lines, boxplot, histograms, etc.)
  - **Aesthetic mappings** (what and where we'll draw: x-axis, y-axis, color, fill, shape, size, linetype, etc.)
  - **Statistics** (implied or specified computing to be done)
  - **Scales** (range of values, colors, or shapes)
  - **Facets** (the panes—there can be more than 1)
  - **Guides** (legends—what the humans see)

```{r}
ggplot()
```


```{r}
ggplot(acitelli, aes(x = satisfaction))
```

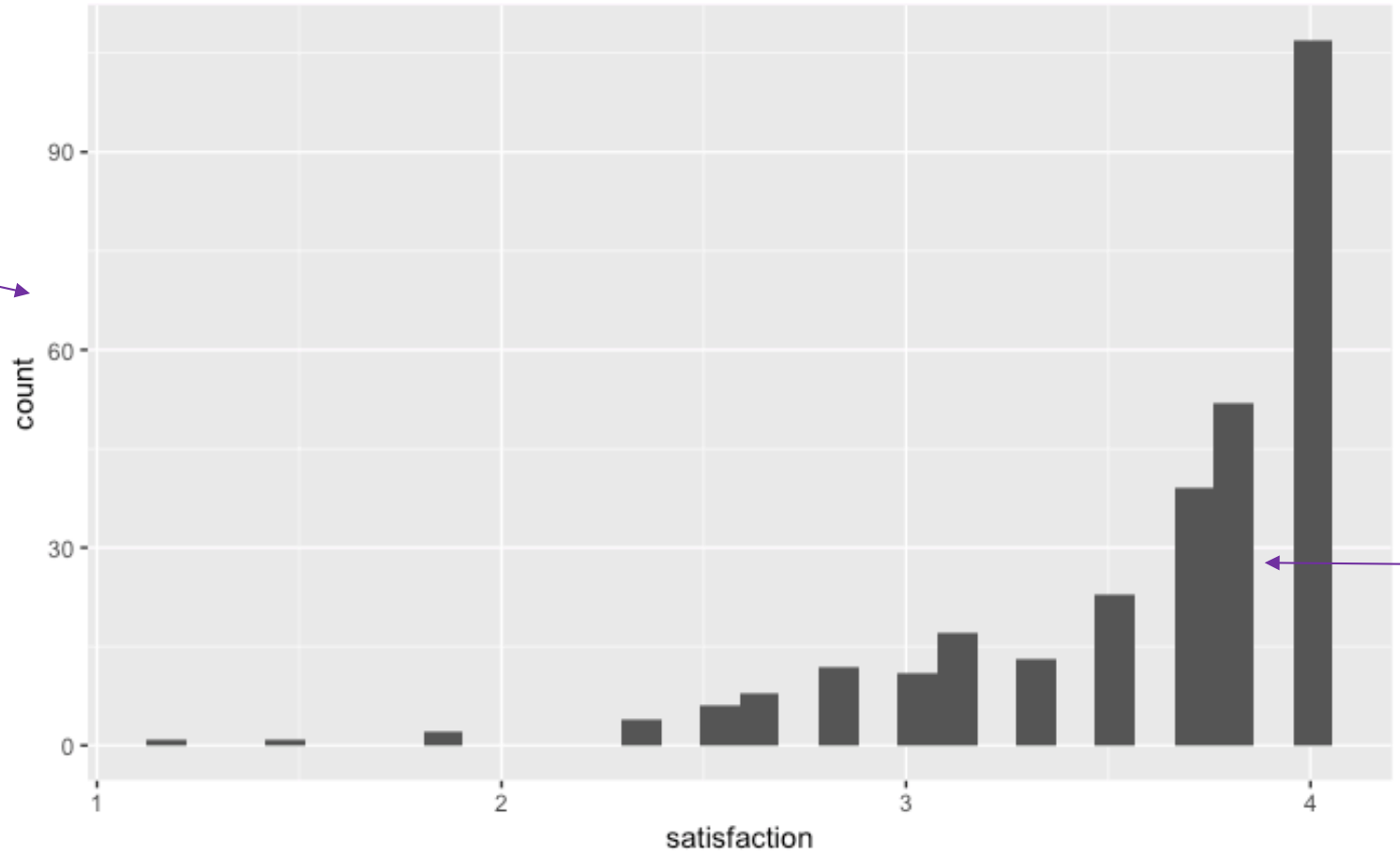
The data comes first

Specify "aesthetic mappings" with the `aes()` function



```{r}
ggplot(acitelli, aes(x = satisfaction)) +
  geom_histogram()
```

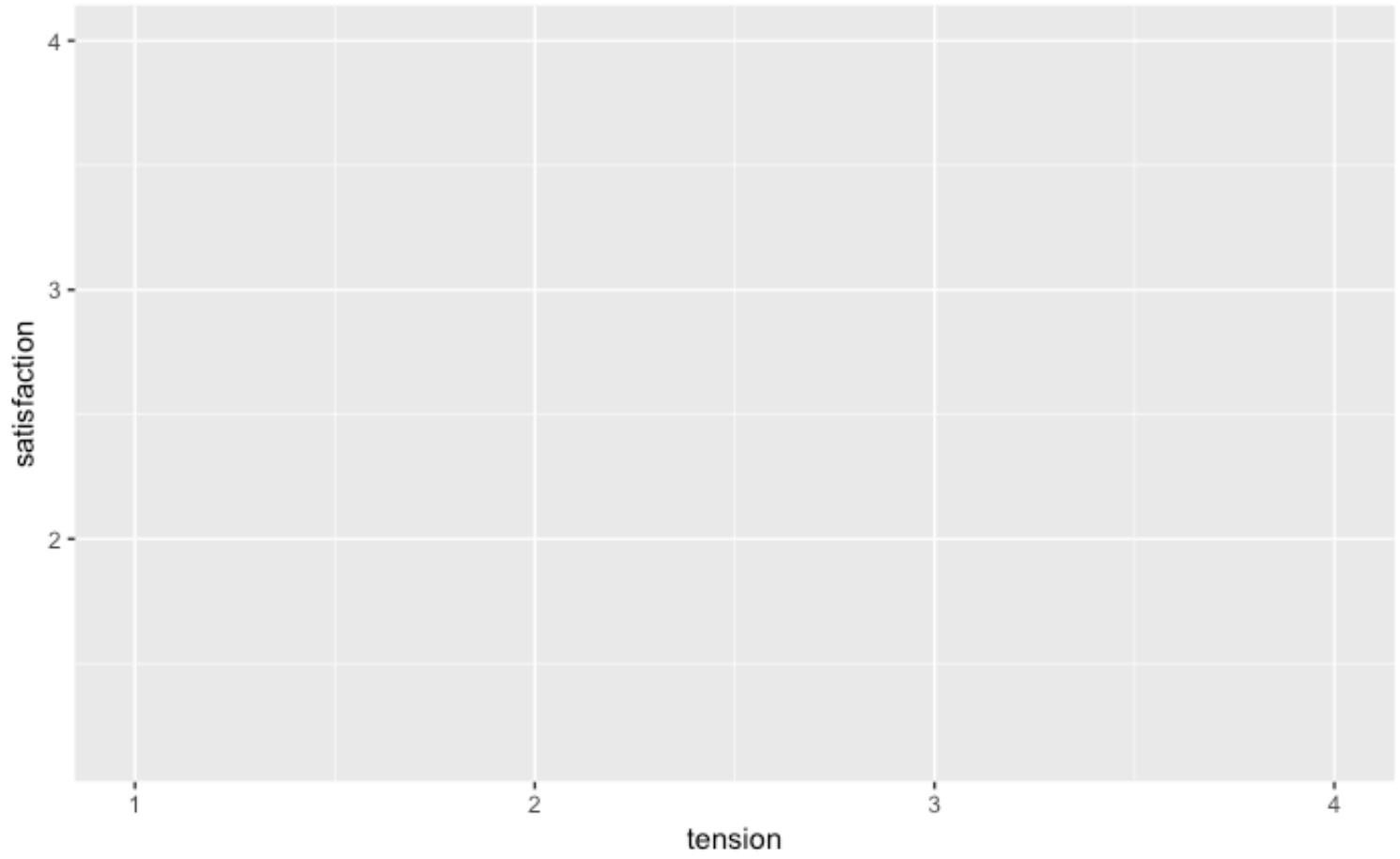
Gotta add some geom's



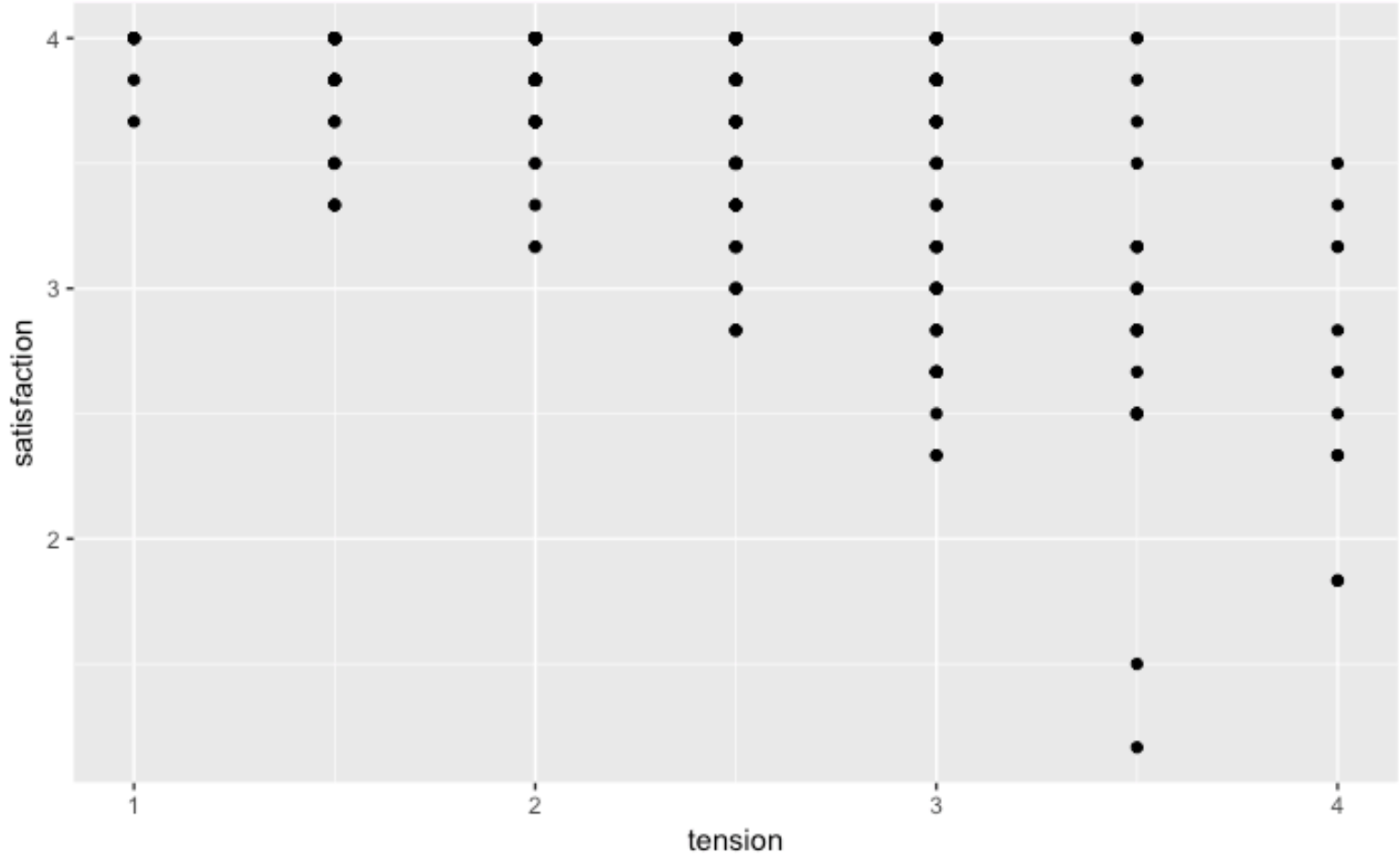
Statistic

Geometric object

```{r}
ggplot(acitelli, aes(x = tension, y = satisfaction))
```



```{r}
ggplot(acitelli, aes(x = tension, y = satisfaction)) +
  geom_point()
```

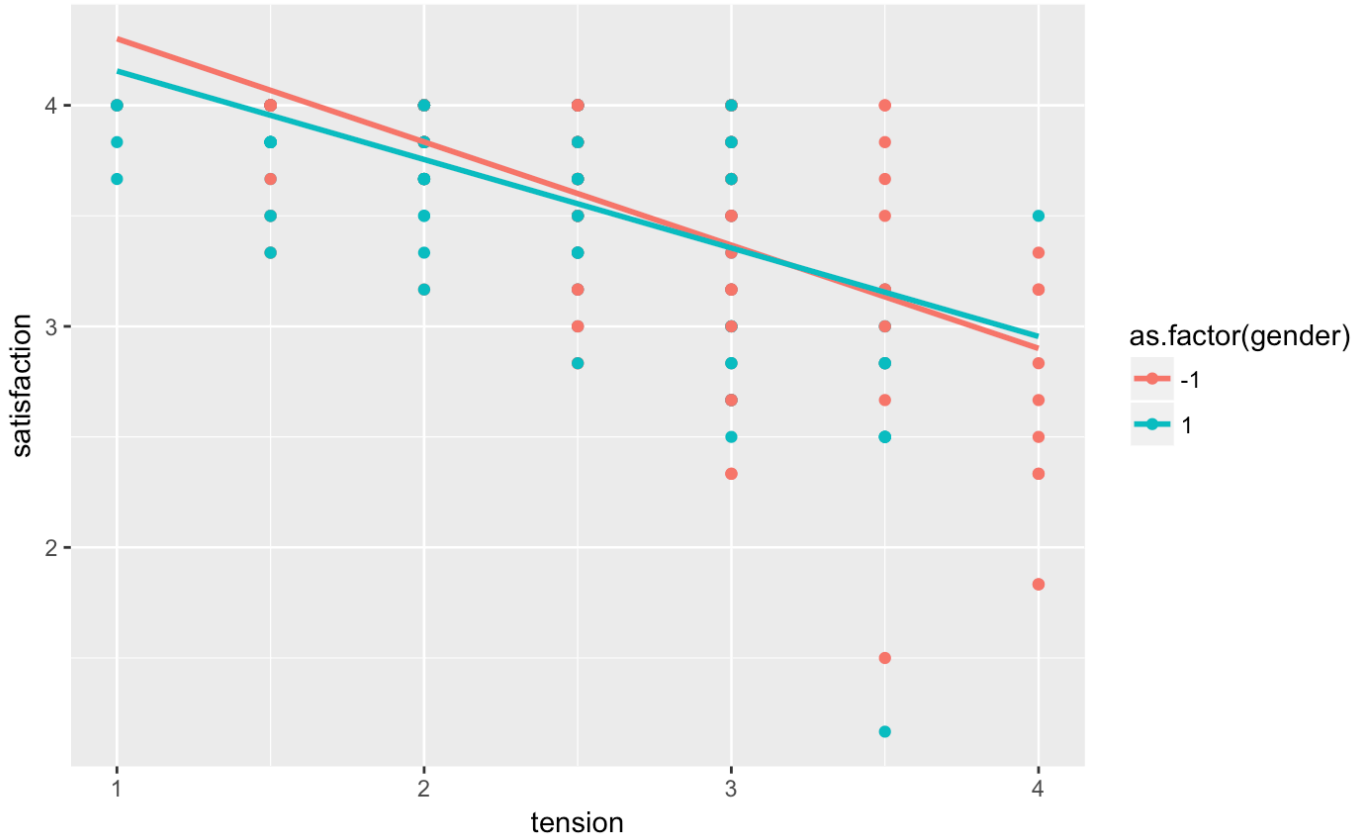


```{r}
ggplot(acitelli, aes(x = tension,
                     y = satisfaction,
                     color = as.factor(gender))) +
  geom_point() +
  geom_smooth(method = "lm", se = 0)
```

Map to color!

Layer on those geoms!

- What do you think would happen if we mapped color to self_pos, a numerical variable?

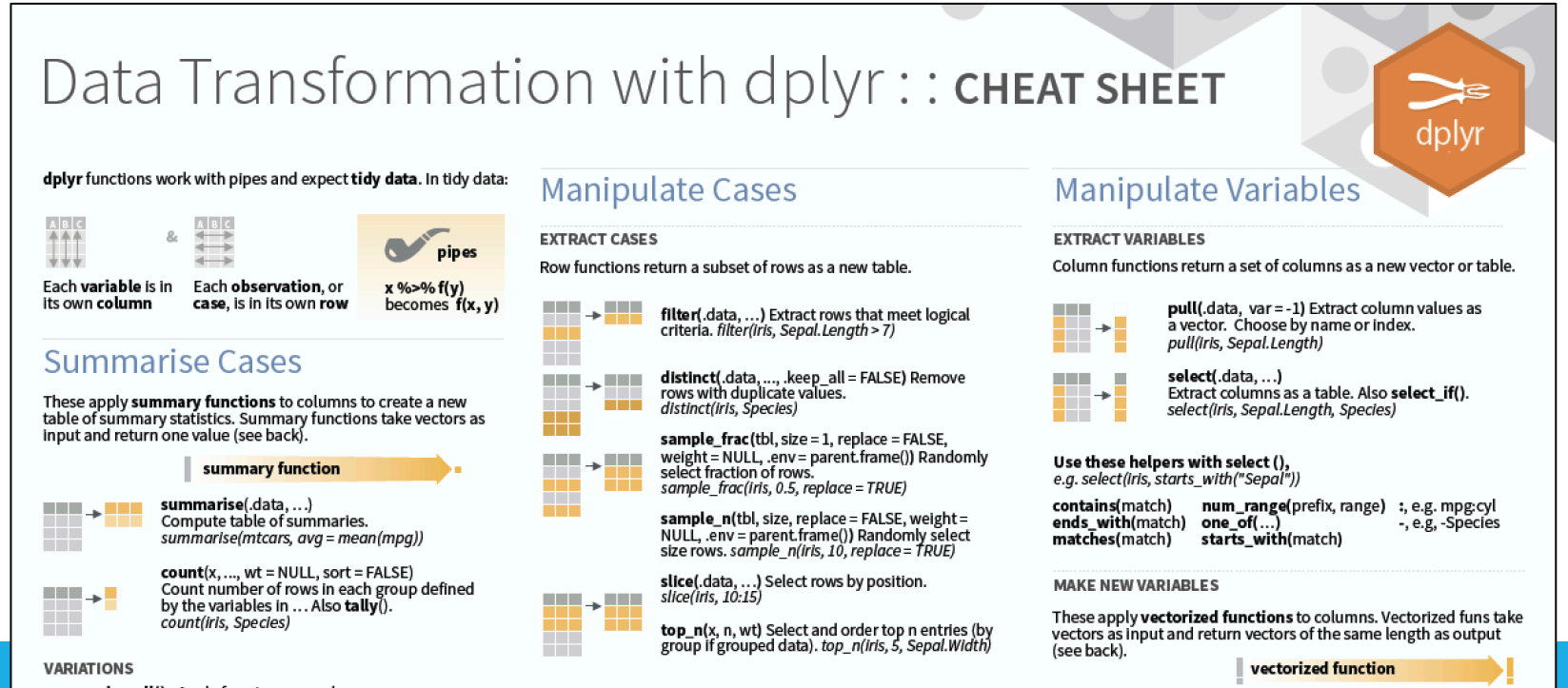


R MARKDOWN

Intro_to_ggplot2.Rmd

Data Cleaning

- The package we'll use for data cleaning is called `dplyr`, which is part of the `tidyverse`, also written by Hadley Wickham
- Find all the cheatsheets here: https://www.rstudio.com/resources/cheatsheets/



Data Transformation with dplyr : : **CHEAT SHEET**

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes
x %>% f(y) becomes f(x, y)

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …) Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE) Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()) Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …) Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt) Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1) Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …)
Extract columns as a table. Also **select_if**().
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match) **num_range**(prefix, range) **:**, e.g. mpg:cyl
ends_with(match) **one_of**(…) **-**, e.g, -Species
matches(match) **starts_with**(match)

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

vectorized function

Data Cleaning

- The five data verbs
  - `filter()`
  - `mutate()`
  - `arrange()`
  - `select()`
  - `summarize()`

- And also…
  - `group_by()`
  - `rename()`
  - `full_join()`, `right_join()`, `left_join()`, `inner_join()`
  - `gather()`
  - `spread()`

Data Cleaning

- Each verb performs familiar operations on a dataset
  - Each function takes and dataset and returns a dataset

Verb	What is does	…in SPSS
mutate()	Creates new variables	COMPUTE (or transform in menu)
filter()	Filters for specific cases	FILTER (or select data in menu)
arrange()	Sorts using some logic	SORT
select()	Subsets for only certain variables	DROP
summarize()	Create a summary table	Descriptive statistics
group_by()	Groups dataset by a categorical variable	Like split file in menu

Data Cleaning

- We will use the pipe operator to combine verbs!

%>%

Data Cleaning


```r
function(dataset, args)
```

...is the same as:

```r
dataset %>% function(args)
```

Data Cleaning


```r
filter(dataset, age >= 18)
```

...is the same as:

```r
dataset %>% filter(age >= 18)
```

Data Cleaning

- Why the pipe!?!?
- Let's say we want to
  1. Create a scale score, a depression index (bdi), then
  2. Filter for only people 18 or older, then finally
  3. Keep only a smaller dataset with just bdi and say, social support

```
mutate(dataset, bdi = bdi1 + bdi2 + bdi3)
```

Data Cleaning

- Why the pipe!?!?
- Let's say we want to
  1. Create a scale score, a depression index (bdi), then
  2. Filter for only people 18 or older, then finally
  3. Keep only a smaller dataset with just bdi and say, social support

```
filter(mutate(dataset, bdi = bdi1 + bdi2 + bdi3), age >= 18)
```

Data Cleaning

- Why the pipe!?!?
- Let's say we want to
  1. Create a scale score, a depression index (bdi), then
  2. Filter for only people 18 or older, then finally
  3. Keep only a smaller dataset with just bdi and say, social support

```
select(filter(mutate(dataset, bdi = bdi1 + bdi2 + bdi3), age >= 18) bdi, soc_sprt)
```

Data Cleaning

- Instead of reading/writing:

```r
select(filter(mutate(dataset, bdi = bdi1 + bdi2 + bdi3), age >= 18) bdi, soc_sprt)
```

- We can write:

```r
dataset %>%
  mutate(bdi = bdi1 + bdi2 + bdi3) %>%
  filter(age >= 18) %>%
  select(bdi, soc_sprt)
```

Data Cleaning

- Save to a new object:

```r
dataset_small <- dataset %>%
  mutate(bdi = bdi1 + bdi2 + bdi3) %>%
  filter(age >= 18) %>%
  select(bdi, soc_sprt)
```

- Or the same object

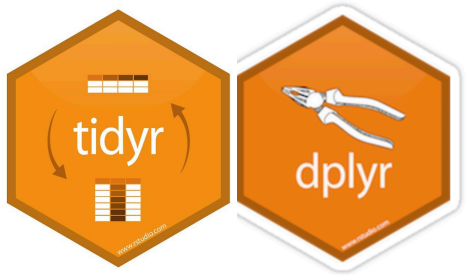
```r
dataset <- dataset %>%
  mutate(bdi = bdi1 + bdi2 + bdi3) %>%
  filter(age >= 18) %>%
  select(bdi, soc_sprt)
```

Little Bunny Foo Foo

```
bop(scoop(hop(foo_foo, through = forest), up = field_mice), on = head)
```

```
foo_foo %>%
  hop(through = forest) %>%
  scoop(up = field_mouse) %>%
  bop(on = head)
```

More Data Cleaning (Day 2)



- There are also verbs for **joining two tables** (in `dplyr`)
  - Adding cases from another dataset
    - `bind_rows()`
  - Adding variables from another dataset
    - `inner_join()`, `right_join()`, `left_join()`, `full_join()`
    - `bind_cols()`

- And verbs for **transforming data** from (in `tidyr` package)
  - Wide-to-long
    - `gather()`
  - Long-to-wide
    - `spread()`

R MARKDOWN FILE

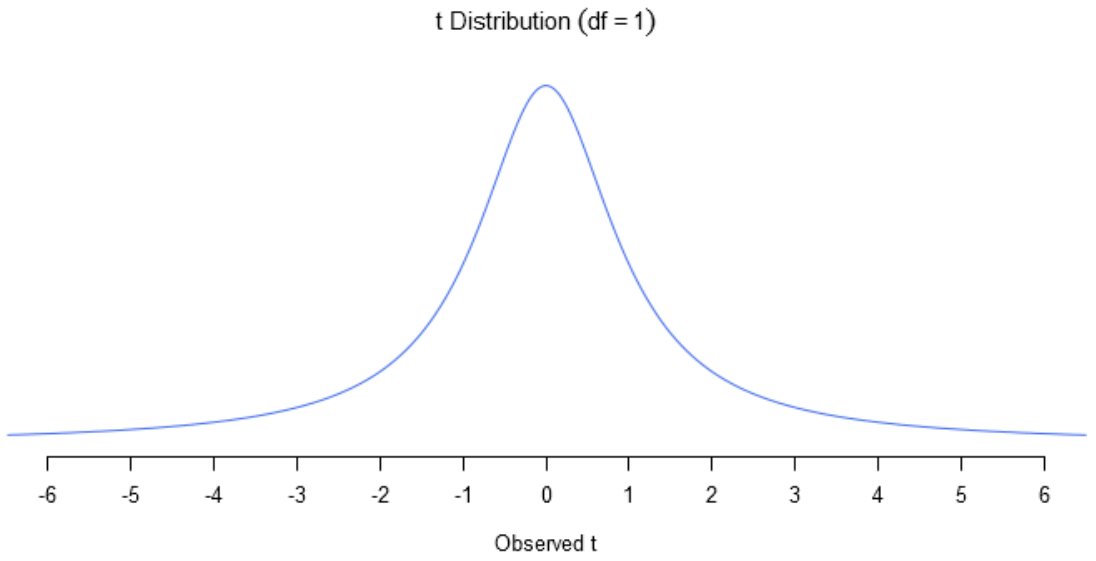
intro_to_dplyr.Rmd

Categorical Variables

- Some stuff you'll need from the `forcats` package

Correlation Matrices, Reliability, and t-Tests

- For correlation matrices and Cronbach's alpha we'll use the package called **psych**
- For t-Tests I recommend you use **mosaic** because it has the formula, then data, syntax (without needing dollar signs)



Correlation Matrices and Reliability

- Correlation Matrix
  - `corr.test()`
  - I like to use this with `select()`:

```
```{r}
corr.test(select(bfi, A1.r, A2, A3, A4, A5))
```
```

vars for matrix

- Reliability
  - `alpha()`
  - Also handy with select:

```
```{r}
alpha(select(bfi, A1.r, A2, A3, A4, A5))
```
```

items for alpha

Creating Scale Scores

- It's best to use the `rowMeans()` function from Base R.
  - Doesn't quite have the same syntax, the data will need to be in the `select()` function.

```
bfi <- bfi %>%
  mutate(agreeable = rowMeans(select(bfi, A1.r, A2, A3, A4, A5), na.rm = TRUE))
```

Student's t-Tests (in `mosaic`)

- One-sample

```r
t.test(~self_pos, data = acitelli)
```

- independent samples

```r
t.test(satisfaction ~ gender, data = acitelli)
```

- paired samples

```r
t.test(~(self_pos-other_pos), data = acitelli)
```

Function Masking

- R is open source and anyone is welcome to contribute a package!
- The package author decides on the names of their functions and there are bound to be redundant function names
- Sometimes it's by design
  - `t.test()` is a function in Base R
  - `t.test()` is a function in `mosaic`
- Sometimes is an unfortunate coincidence
  - `alpha()` is a function in `ggplot2`
  - `alpha()` is a function in `psych`

Function Masking

- Solution 1: Always load psych *after* `dplyr` and `ggplot2`

```
```{r}
library(forcats)
library(dplyr)
library(tidyr)
library(ggplot2)
library(psych)

acitelli <- read.csv("acitelli.csv")
```
```

- Solution 2: Do what you want, but if you get errors, be explicit about which package you want

```
```{r}
psych::alpha(select(bfi, A1.r, A2, A3, A4, A5))
```
```

R MARKDOWN FILE

cor_reliability_ttest.Rmd