

# Getting Started with R

R for the Rest of Us



# Installation





# Install R

The first thing you need to do is download the R software. Go to the [Comprehensive R Archive Network \(aka “CRAN”\) website](#) and download the software for your operating system (Windows, Mac, or Linux).



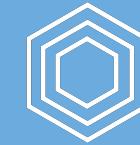
# Working Directly in R

R for the Rest of Us



# Working Directly in R

1. Enter:  $2 + 2$
2. Hit return
3. View result



## Your Turn

1. Download and Install R
2. Open R
3. Use any mathematical operators (+, -, /, and \*) to create an expression and make sure it works as expected



# RStudio

---

R: Engine



RStudio: Dashboard



---

Courtesy [Modern Dive](#)

R for the Rest of Us



# RStudio

If you use RStudio, you'll have a graphical user interface, the ability to see all of your stored information, and much more.

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Displays the code for "Interactive figures.R". The code uses the leaflet package to create a map of Washington and Oregon. It filters data based on participation ("Yes" or "No") and adds markers for each school, color-coded by participation status (blue for Yes, red for No). A legend in the map view allows users to switch between these two categories.
- Environment Browser:** Shows the global environment with various datasets and objects listed.
- Map View:** A map of the Pacific Northwest (Washington, Oregon, and parts of Idaho and Canada) with markers placed at the locations of schools. A tooltip on the map indicates the participation status for each location.
- Console:** Shows the command history and output from running the R code.

R for the Rest of Us



## Download RStudio

Download RStudio at the [RStudio website](#). Ignore the various versions listed there. All you need is the latest version of RStudio Desktop.

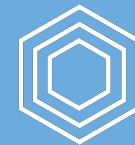


# Tour of RStudio

The screenshot shows the RStudio interface with four main panels:

- Scripts**: Shows an R script with code for installing packages, loading tidyverse and skimr, importing faketucky data, and examining it.
- Environment/History**: Shows the Global Environment tab with a list of objects and their types (e.g., `intro-to-r-slides.html`, `intro-to-r-slides.Rmd`, `setup.Rmd`, `style.css`).
- Files/Plots/Packages/Help**: Shows the Files tab with a file tree and details for each file (size, modified date).
- Console/Terminal**: Shows the R console output, including the R startup message and help information.

R for the Rest of Us



## Your Turn

1. Download and Install RStudio
2. Open RStudio
3. Working in the console pane, use any mathematical operators (+, -, /, and \*) to create an expression and make sure it works as expected



# Projects



# Projects

Projects allow you to keep a collection of files all together, including:

- R scripts
- RMarkdown files (more on those soon)
- Data files
- And much more!



# Sample Project

The screenshot shows the RStudio interface with four main panes:

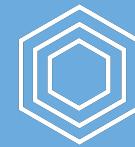
- Scripts:** Displays an R script with code for installing packages, loading tidyverse and skimr, importing foketucky data, and examining data.
- Environment/History:** Shows the Global Environment tab with a list of objects.
- Files/Plots/Packages/Help:** Shows a file browser with files like intro-to-r-slides.html, intro-to-r-slides.Rmd, setup.Rmd, and style.css.
- Console/Terminal:** Displays the R startup message and help information.

R for the Rest of Us



# How to Create a Project

1. File -> New Project
2. Quit RStudio, Double-click .Rproj file to reopen project



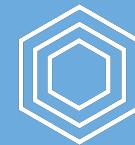
## Your Turn

1. Create a new project (doesn't matter if it's in a new or existing directory)
2. Quit RStudio, double-click the .Rproj file and reopen your project



# Download Course Project





# Your Turn

Enter the following into the RStudio console pane:

```
install.packages("usethis")
library(usethis)
use_course("http://bit.ly/getting-started-with-r")
```



# Files in R





# File Types

There are **two main file types** that you'll work with:

## R scripts (.R)

Text is assumed to be executable R code unless you comment it (more on this soon)

```
# This is a comment  
  
data <- read_csv("data.csv")
```

## RMarkdown files (.Rmd)

Text is assumed to be text unless you put it in a code chunk (more on this soon)

```
...{r}  
ggplot(tfff_basemap) +  
  geom_sf(fill = "#eeeeee",  
          alpha = .5) +  
  coord_sf(datum = NA) +  
  theme(axis.title = element_blank()) +  
  geom_jitter(data = data,  
              aes(longitude, latitude),  
              size = 2,  
              stroke = 1,  
              # shape = 21,  
              fill = "white",  
              color = tfff_orange) +  
  theme(text = element_text(lineheight = .1,  
                           family = "Calibri"),  
        panel.background = element_rect(fill = "transparent"))  
...
```



# R Scripts

Create new script file: File -> New File -> R Script

The screenshot shows the RStudio interface. The top menu bar has 'File' selected, with 'New File' highlighted. A sub-menu dropdown is open, showing options like 'New File...', 'Open File...', 'Save', and 'Close'. The main workspace shows a data frame named 'satisfaction\_data'. The bottom-left pane is a terminal window displaying R code for generating an HTML presentation. The bottom-right pane shows a file browser with several files listed.

```
1:4 [Top Level] R Script
Console Terminal
~/Google Drive/Work/Projects/R for Eval/R/ presentation.Rmd
List of 1
$ eval: logi FALSE
!..... ordinary text without R code
output file: presentation.knit.md
/usr/local/bin/pandoc +RTS -K512m -RTS presentation.utf8.md --to html4 --from markdown+autolink_bare_uris+ascii_identifiers+tex_math_single_backslash+smart --output presentation.html --email-obfuscation none -V 'mathjax-url=https://cdn.bootcss.com/mathjax/2.7.1/MathJax.js?config=TeX-MML-AM_CHTML' --standalone --section-divs --template /Library/Frameworks/R.framework/Versions/3.5/Resources/library/xaringan/rmarkdown/templates/xaringan/resources/default.html --no-highlight --css style.css --include-in-header /var/folders/pd/stkbswg1471cgy69q26s6r080000gn/T//Rtmpn6hFre/rmarkdown-str40f174140c.html --include-before-body /var/folders/pd/stkbswg1471cgy69q26s6r080000gn/T//Rtmpn6hFre/xaringan40f3c8d338.md --include-after-body /var/folders/pd/stkbswg1471cgy69q26s6r080000gn/T//Rtmpn6hFre/xaringan40f79a8da26.js --variable title-slide=true --variable math=true
Output created: presentation.html
```

R for the Rest of Us



# How to Run Code

Run the code: control + enter on Windows, command + enter on Mac  
keystrokes or use Run button



# Comments

Do them for others, and for your future self.

```
# This is a comment  
head(data, n = 5)
```



# Packages





# Packages

Packages add functionality that is not present in base R.

They're where much of the power of R is found.

---

R: A new phone



R Packages: Apps you can download



---

Courtesy [Modern Dive](#)



# Packages We'll Use



tidyverse

The [tidyverse](#) is a collection  
of packages.

We'll use [readr](#) to import data.



# Packages We'll Use

## skimr

[skimr](#) provides easy summary statistics.





# Install Packages

The syntax to install packages is as follows.

```
install.packages("tidyverse")
install.packages("skimr")
```

The package name must be in quotes.

Packages should be installed **once per computer** (i.e. once you've installed a package, you don't need to do it again on the same computer).



# Load Packages

To load packages, use the following syntax:

```
library(tidyverse)  
library(skimr)
```

Package names don't need to be quoted here (though they can be).

Packages should be loaded **once per session** (i.e. every time you start working in R, you need to load any packages you want to use).



# Your Turn

1. Open the project you downloaded before (it should be on your desktop)
2. Open the exercises.R file
3. Install the tidyverse and skimr packages using the install.packages function
4. Load the tidyverse and skimr packages using the library function



# Import Data



# Import Data

Let's read data from a CSV file.

```
faketucky <- read_csv("data/faketucky.csv")
```

We now have a data frame/tibble called `faketucky` that we can work with in R.



# R is Case Sensitive

R is **case sensitive** so choose one of the following for all objects and **be consistent**.

Option	Example
snake_case	student_data
camelCase	studentData
periods.in.names	student.data



# Directories

If the data file is in the working directory, you only need to specify its name.

```
faketucky <- read_csv("faketucky.csv")
```

If the data file is not in the working directory, you need to specify full path name.

When specifying the path name use of forward slash (“/”) not backslash (“\”).

```
faketucky <- read_csv("data/faketucky.csv")
```



# Where Does our Data Live?

Data we have imported is available in the environment/history pane.

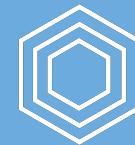
The screenshot shows the RStudio interface with two main panes highlighted:

- Console/Terminal**: On the left, it displays the R startup message and basic usage instructions.
- Environment/History**: On the right, it shows the "Global Environment" tab with a list of objects and their details. The list includes:

Name	Type	Size	Modified
Icon	File	0 B	Feb 12, 2019, 7:20 AM
images	Folder		
intro-to-r-slides.html	File	15.1 KB	Feb 13, 2019, 9:39 AM
intro-to-r-slides.Rmd	File	9.5 KB	Feb 13, 2019, 9:39 AM
libs	Folder		
setup.Rmd	File	2.4 KB	Feb 12, 2019, 2:57 PM
style.css	File	2.3 KB	Feb 12, 2019, 3:07 PM

Large, semi-transparent text overlays are present in the center of the interface:  
Scripts  
Environment/  
History  
Files/Plots/  
Packages/Help

R for the Rest of Us



# Your Turn

1. Open the exercises.R file
2. Import the faketucky data into a data frame called `faketucky`.
3. Make sure you see `faketucky` in your environment/history pane.



# Objects and Functions





# Objects and Functions

To understand computations in R, two slogans are helpful:

Everything that exists is an **object**, and

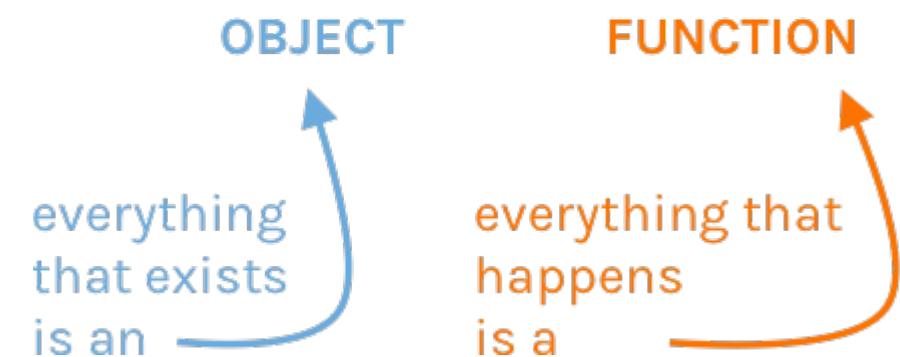
Everything that happens is a **function** call.

John Chambers, quoted in [Hadley Wickham's Advanced R](#).



# Objects and Functions

```
faketucky <- read_csv("data/faketucky.csv")
```





# Assignment Operator

```
faketucky <- read_csv("data/faketucky.csv")
```

ASSIGNMENT  
OPERATOR

We assign the result of the **read\_csv function** to the **faketucky object**



# Examine Our Data



# Examine Our Data

There are many ways to look at our data. We'll talk about a few.



# faketucky

If you type the name of your data frame (i.e. `faketucky`), R will output the following:

```
faketucky
```

```
## # A tibble: 57,855 x 12
##   student_id first_high_scho... school_district male race_ethnicity
##       <dbl> <chr>           <chr>          <dbl> <chr>
## 1       1622 Jackson         Jackson        1 Multiple/Nati...
## 2       1877 Jackson         Jackson        1 White
## 3       1941 Jackson         Jackson        1 White
## 4       3442 Jackson         Jackson        0 White
## 5       4623 Jackson         Jackson        1 White
## 6       4913 Jackson         Jackson        1 White
## 7       5754 Jackson         Jackson        1 White
## 8       6293 Jackson         Jackson        0 White
## 9       7010 Jackson         Jackson        1 White
## 10      8343 Jackson         Jackson        1 White
## # ... with 57,845 more rows, and 7 more variables:
## #   free_and_reduced_lunch <dbl>, percent_absent <dbl>, gpa <dbl>,
## #   act_reading_score <dbl>, act_math_score <dbl>,
## #   received_high_school_diploma <dbl>, enrolled_in_college <dbl>
```



# head

head shows us the first X rows.

```
head(faketucky, 5)
```

```
## # A tibble: 5 x 12
##   student_id first_high_scho... school_district male race_ethnicity
##       <dbl> <chr>           <chr>          <dbl> <chr>
## 1      1622 Jackson         Jackson        1 Multiple/Nati...
## 2      1877 Jackson         Jackson        1 White
## 3      1941 Jackson         Jackson        1 White
## 4      3442 Jackson         Jackson        0 White
## 5      4623 Jackson         Jackson        1 White
## # ... with 7 more variables: free_and_reduced_lunch <dbl>,
## #   percent_absent <dbl>, gpa <dbl>, act_reading_score <dbl>,
## #   act_math_score <dbl>, received_high_school_diploma <dbl>,
## #   enrolled_in_college <dbl>
```



# head

```
head(faketucky, 5)
```

faketucky and the number 5 here are **arguments**.



# tail

tail shows us the last X rows.

```
tail(faketucky, 5)
```

```
## # A tibble: 5 x 12
##   student_id first_high_scho... school_district male race_ethnicity
##       <dbl> <chr>           <chr>          <dbl> <chr>
## 1     89364 Lookout Point    Lookout Point    1 Hispanic
## 2     94764 Lookout Point    Lookout Point    0 White
## 3     97538 Lookout Point    Lookout Point    1 White
## 4    101342 Lookout Point    Lookout Point    1 White
## 5    104903 Lookout Point    Lookout Point    1 Hispanic
## # ... with 7 more variables: free_and_reduced_lunch <dbl>,
## #   percent_absent <dbl>, gpa <dbl>, act_reading_score <dbl>,
## #   act_math_score <dbl>, received_high_school_diploma <dbl>,
## #   enrolled_in_college <dbl>
```



# View

`View` (note capital V) opens the RStudio viewer (or click on a data frame in the environment pane).

```
View(faketucky)
```

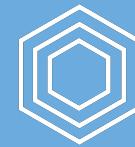


# skimr

The `skimr` package provides more detailed information about our data frame. It is also broken up by the type of variable.

```
skim(faketucky)
```

```
## Skim summary statistics
## n obs: 57855
## n variables: 12
##
## — Variable type:character —
##           variable missing complete      n  min  max empty n_unique
## first_high_school_attended     0    57855 57855     4   14     0     393
## race_ethnicity                 794    57061 57855     5   24     0       5
## school_district                  0    57855 57855     4   13     0     171
##
## — Variable type:numeric —
##           variable missing complete      n      mean        sd p0
## act_math_score                  0    57855 57855 257.85 420.77  1
## act_reading_score                0    57855 57855 258.78 420.65  2
## enrolled_in_college              0    57855 57855 255.74 435.54  0
## free_and_reduced_lunch            0    57855 57855 15.45 120.82  0
## gpa                             0    57855 57855 40.22 189.95  0
```



# Your Turn

1. Open the file exercises.R
2. Follow the instructions to use the `head()`, `tail()`, `View()`, and `skim()` functions to examine your data.



We've Got  
Issues!



# We've Got Issues!

Several variables have max values of 999. This seems suspicious!

```
> skim(faketucky)
Skim summary statistics
n obs: 57855
n variables: 12

— Variable type:character —
      variable missing complete   n  min  max empty n_unique
first_high_school_attended     0    57855 57855  4  14     0     393
race_ethnicity                 794   57061 57855  5  24     0       5
school_district                  0    57855 57855  4  13     0     171

— Variable type:numeric —
      variable missing complete     n   mean     sd p0    p25    p50    p75   p100 hist
act_math_score                  0    57855 57855 257.85 420.77  1    16     20     33    999 ─────────
act_reading_score                0    57855 57855 258.78 420.65  2    16     22     34    999 ─────────
enrolled_in_college              0    57855 57855 255.74 435.54  0     0     1     999    999 ─────────
free_and_reduced_lunch            0    57855 57855 15.45 120.82  0     0     1     1     999 ─────────
gpa                             0    57855 57855 40.22 189.95  0    2.02    2.7    3.36    999 ─────────
male                            0    57855 57855  0.76 15.54  0     0     1     1    999 ─────────
percent_absent                   0    57855 57855 10.68 46.19  0    3.27    6.28   11.35  3153 ─────────
received_high_school_diploma      0    57855 57855  0.74  0.44  0     0     1     1     1 ─────────
student_id                       0    57855 57855 55922.15 32332.71  1 27909.5  56070  83872.5 111990 ─────────
```



# We've Got Issues!

Several variables show up as numeric, but we know they're **not** actually numeric.

```
> skim(faketucky)
Skim summary statistics
n obs: 57855
n variables: 12

— Variable type:character —
      variable missing complete   n  min  max empty n_unique
first_high_school_attended     0    57855 57855  4  14    0    393
race_ethnicity                 794   57061 57855  5  24    0      5
school_district                0    57855 57855  4  13    0    171

— Variable type:numeric —
      variable missing complete     n    mean      sd p0    p25    p50    p75    p100 hist
act_math_score                  0    57855 57855 257.85 420.77  1    16    20    33    999 ─────────
act_reading_score                0    57855 57855 258.78 420.65  2    16    22    34    999 ─────────
enrolled_in_college              0    57855 57855 255.74 435.54  0    0    1    999    999 ─────────
free_and_reduced_lunch            0    57855 57855 15.45 120.82  0    0    1    1    999 ─────────
gpa                             0    57855 57855 40.22 189.95  0    2.02   2.7   3.36   999 ─────────
male                            0    57855 57855  0.76 15.54  0    0    1    1    999 ─────────
percent_absent                   0    57855 57855 10.68 46.19  0    3.27   6.28  11.35  3153 ─────────
received_high_school_diploma      0    57855 57855  0.74  0.44  0    0    1    1    1 ─────────
student_id                       0    57855 57855 55922.15 32332.71 1 27909.5 56070  83872.5 111990 ─────────
```



# Let's Import Our Data Again

We need to do two things:

1. Tell `read_csv` how to handle **missing data**
2. Make sure `read_csv` assigns the correct **data type** to each variable



# Missing Data

Here's how we imported our data the first time:

```
faketucky <- read_csv("data/faketucky.csv")
```

To tell R which data is missing, simply add an argument to the `read_csv` function as follows:

```
faketucky <- read_csv("data/faketucky.csv",
                        na = "999")
```



# Missing Data

If we skim our data again, we'll see that there are no longer 999 values.

```
skim(faketucky)
```



# Data Types

When you run `read_csv` you'll see the following message, which tells you the data types that have been assigned to your data frame.

```
Parsed with column specification:  
cols(  
  student_id = col_double(),  
  first_high_school_attended = col_character(),  
  school_district = col_character(),  
  male = col_double(),  
  race_ethnicity = col_character(),  
  free_and_reduced_lunch = col_double(),  
  percent_absent = col_double(),  
  gpa = col_double(),  
  act_reading_score = col_double(),  
  act_math_score = col_double(),  
  received_high_school_diploma = col_double(),  
  enrolled_in_college = col_double()  
)
```



# Data Types

- **Double/Numeric** (e.g. 2.5)
- **Character** (e.g. "Male")

There are [many other data types in R](#) that we won't discuss in this course.



# Data Types

```
faketucky <- read_csv("data/faketucky.csv",
                      na = "999",
                      col_types = list(enrolled_in_college = col_character,
                                      free_and_reduced_lunch = col_character,
                                      male = col_character(),
                                      received_high_school_diploma = col_
```



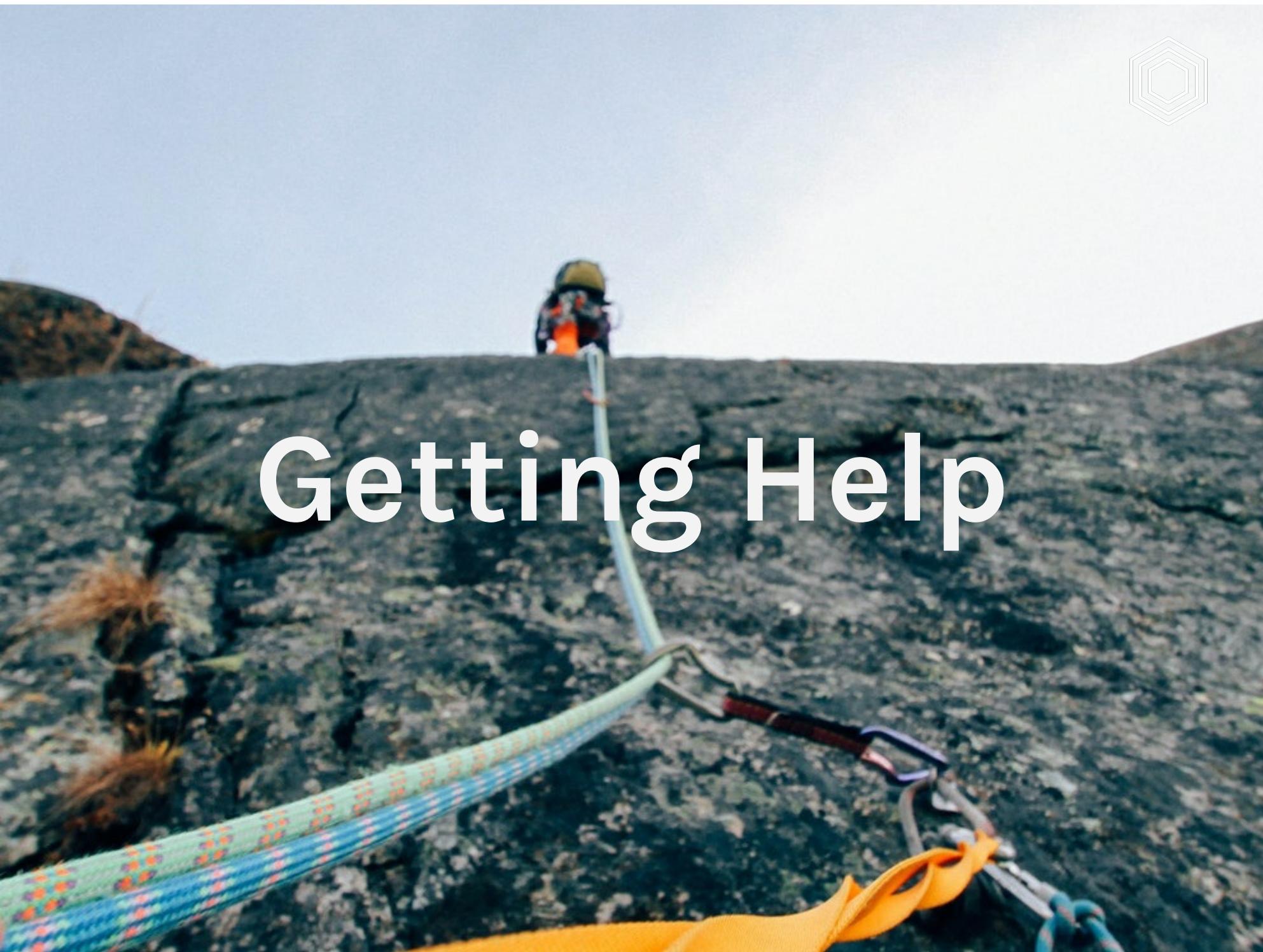


# Your Turn

1. Open the file `exercises.R` and change the code so that you correctly import the `faketucky` data frame, telling `read_csv` which data is missing and explicitly defining column types where necessary.
2. After you make changes to how you import your data, rerun the code in the Examine Data section to make sure everything worked!



# Getting Help





# ?function

Use the ? to get help about anything you're confused about

```
?read_csv
```



# Tidyverse Website

Tidyverse



## Tidyverse packages

### Installation and use

- Install all the packages in the tidyverse by running `install.packages("tidyverse")`.
- Run `library(tidyverse)` to load the core tidyverse and make it available in your current R session.

Learn more about the tidyverse package at <http://tidyverse.tidyverse.org>.



# Package Vignettes

## Using Skimr

*Elin Waring*

**2019-01-13**

`skimr` is designed to provide summary statistics about variables. It is opinionated in its defaults, but easy to modify.

In base R, the most similar functions are `summary()` for vectors and data frames and `fivenum()` for numeric vectors:

```
summary(iris)

##   Sepal.Length   Sepal.Width    Petal.Length    Petal.Width
##   Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##   1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##   Median :5.800   Median :3.000   Median :4.350   Median :1.300
```



# Twitter

Sam Way and 12 others Retweeted

**Gina Reynolds @EvaMaeRey** · Feb 11

my **#ggplot2** flipbook project is online! 😎😎😎 Incrementally walks through plotting code (**#MakeoverMonday**, soon **#TidyTuesday** plots). Using **#xaringan** with reveal function; thanks, **@statsgen** **@grrrck**. **#rstats**.

[evamae.rey.github.io/ggplot\\_flipboo...](http://evamae.rey.github.io/ggplot_flipboo...)

```
ggplot(data = dcast +  
  mutate(.byract = if (year > 1) 4  
    else 1, .byract = "Percent change from previous period"), +  
  mutate(.byr = year, .byr =  
  .byr * 100, .byr =  
  .byr * 100, .byr =  
  geom_bar(stat = "identity", width = 0.5) +  
  geom_text(x = 2010, y = 0.5, label = "gapminder")
```

0:07 | 12.9K views

16 / 28

23

208

...

729

✉

...

Show this thread

R for the Rest of Us



# R for Data Science Community

A screenshot of the R for Data Science Online Learning Community website. The header reads "R4DS Online Learning Community". The main content features a large teal hexagonal graphic with a green parrot and the text "R FOR DATA SCIENCE ONLINE LEARNING COMMUNITY". Below this is a navigation bar with links to "HOME", "GET INVOLVED", and "ABOUT".

R4DS Online Learning Community

HOME GET INVOLVED ABOUT

R FOR DATA SCIENCE  
ONLINE LEARNING  
COMMUNITY

r4ds  
an online learning community



## LEARNERS

Develop your skills in R and Data science with a friendly community



## MENTORS

Help build a positive learning community for R users and expand your R debugging skills



## COMMUNITY

Help develop resources for others to feel part of the #rstats community

R for the Rest of Us



# Google



E. Kale Edmiston PhD  
@EKaleEdmiston

Follow



A friend/colleague who is an excellent programmer offhandedly told me the other day that coding is 90% googling error messages & 10% writing code. Until this point, I thought that all the time I spent googling error messages meant I was bad at coding. What a perspective change!

8:12 AM - 4 Jan 2019

151 Retweets 1,069 Likes



27

151



1.1K



R for the Rest of Us