Introduction to Reproducible Data Science with R

Institute for Research in Statistics and its Applications at the University of Minnesota

Contents

Home

With the increasing availability of data with broad applications (and the sheer size of some of these data), it is more important than ever to be able to elucidate trends, decisions, and stories from data. Our team will offer a hands-on introduction to data science and statistics using the free and publicly available software R. Assuming no background in software or statistics, we will introduce you to some of the most useful, modern, and popular data analysis techniques.

TOPICS COVERED:

- Features of R and best practices for reproducible data science
- Constructing visualizations, wrangling data, and producing simple numerical summaries
- A gentle introduction to statistical modeling

Welcome & Set-Up

Welcome to IRSA's "Introduction to Reproducible Data Science with R"! As you get settled in, please take the following steps to get set up for the workshop:

1. Introduce yourself to & meet the other students at your table. Though workshop helpers will be floating around the room, we encourage you to support one another as you learn R.
2. Ensure you have completed the steps in the Participant checklist: https://z.umn.edu/Rworkshop19-checklist
3. Open the workshop manual: https://irsaatumn.github.io/RWorkshop19/
4. Open RStudio.
5. Within RStudio, type the following into the console (lower left panel):

```
library("rmarkdown")
library("knitr")
library("ggplot2")
library("dplyr")
library("fivethirtyeight")
library("car")
library("glmbb")
library("pacman")
library("RColorBrewer")
```

   If you get any errors, this likely means that you have not yet installed the packages listed in the participant checklist. If this is the case, type the following into the console:

```
install.packages(c("RColorBrewer", "knitr", "car", "ggplot2", "fivethirtyeight", "glmbb", "p
```

Chapter 1

Introduction to R and RStudio

Author: Alicia Hofelich Mohr

This workshop is motivated by the increasing need for tools that can be used to elucidate trends, decisions, and stories from data. This practice is broadly referred to as "data science":

source: Wickham & Grolemund: R for Data Science

This workflow is often **iterative** and to ensure we can trust the outcomes, the process between input and output should be **transparent** and **repeatable**.

Workshop Outline & Goals

Our goal is to provide a hands on introduction to navigating the data science pipeline with R. You will walk away with a solid foundation upon which you can build for your own research.

Day 1:

- Introduction to R & RStudio
- Reproducibility and R Markdown
- Data Visualization
- Simple Data Wrangling and Summaries

Day 2:

- Linear Regression

- Logistic Regression

1.1 Getting Started

No matter what data you are working with, you need *software* to explore and construct inferences from these data. In this workshop, we'll use the **R** statistical software. Why R?

- it's free
- it's open source
- it's flexible / useful for a wide variety of applications
- it has a huge online community
- it can be used to create reproducible documents, apps, books, etc. (In fact, *this* document was constructed within RStudio.)

R and RStudio

Before this workshop, you were asked to download/update both R and RStudio.

What's the difference?

https://mirror.las.iastate.edu/CRAN/

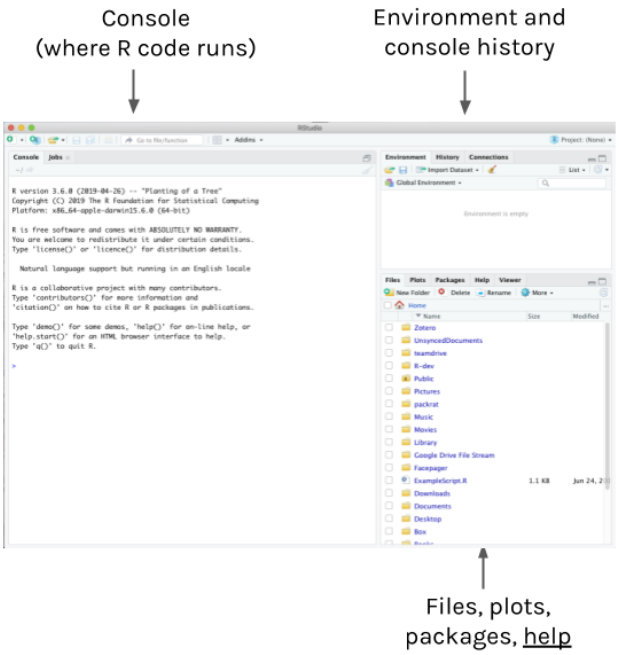
- Actual program (engine)
- Can create and run scripts directly in R
- Text editing on a Mac

https://www.rstudio.com/products/rstudio/

- Integrated Development Environment (IDE)
- Text editing for Mac and PC
- Easy integration with R Markdown, Shiny, git, etc

1.2 R Studio Overview

Here's what you will see when you first open R Studio:



1.2.1 The R console

The R console is where R commands are executed and most output will be displayed. You can type commands directly into the console and run them by pressing the "enter" or "return" key.

Using R as a calculator

We can use R as a basic calculator. Try typing and running each of the following into the console.

```
2 + 3
2 * 3
2^3
(2 + 3)^2
2 + 3^2
```

Assignment

We can assign and store R output as *objects* in R's environment. This allows us to use or reference the output later.

Store the result of `2 + 3^2` as `my_result`

```
my_result <- 2 + 3^2
```

Tip: press 'option/alt and -' at the same time to create the assign arrow

Once you create an object, you will see it appear in the "Environment" pane.

Check out the result in the console

```
my_result
```

Do something with the results

```
my_result + 5
```

Update the object

```
my_result <- my_result + 5

my_result
```

Object names can not include spaces or start with numbers!

```
my result <- 2 + 3^2

1result <- 2 + 3^2
```

Functions

R also has built in *functions* that take in *arguments* and return an output: `function(arguments)`. Arguments can be objects, numbers, or text.

```
sqrt(9)

sqrt(my_result)
```

The sum() function calculates the sum of the listed numbers.

Does the order of arguments matter?

```
sum(2, 3)
sum(3, 2)
```

What does the rep() function do? Does the order of arguments matter?

```
rep(2, 3)
rep(3, 2)
```

Arguments have names

```
rep(x = 3, times = 2)
rep(times = 2, x = 3)
```

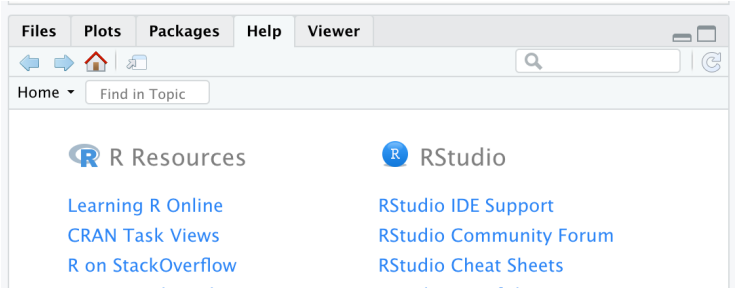
Is R case sensitive? eg: Can we spell rep() as Rep()?

```
Rep(2, 3)
```

1.2.2 The Help window

How do you know what a function does, what arguments are available, and what they are named? Look in the help window!

You can find the help page for a function by typing the function name in the search bar



Or by typing "?" followed by the function name in the console

```
?rep()
```

If you are not sure of what a function would be called or whether it exists, a great option is to Google it.

Some helpful sites include:

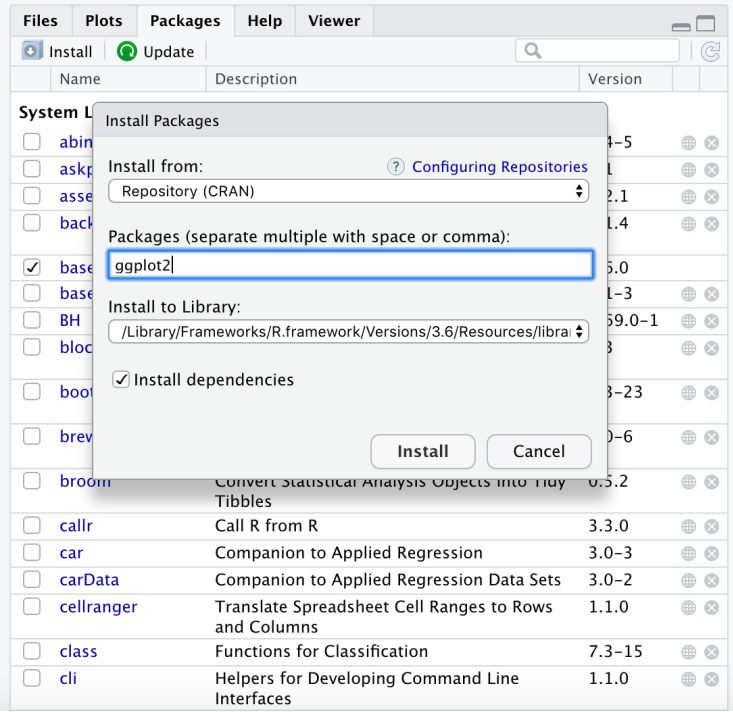
- Stack Overflow: http://stackoverflow.com/questions/tagged/r
- R bloggers: http://www.r-bloggers.com/
- Quick-R: http://www.statmethods.net/

1.2.3 The Package window

Functions such as `rep()` and `sum()` (along with many others) are built into R's base. These sets of functions are often referred to as "base R".

However, because R is open source, anyone can contribute extra functions to R (which allow you to do more things) - these extra functions are bundled into "packages" and are typically hosted in the CRAN repository.

To install the package from CRAN to your computer, you can click "Install" in the package tab, which opens a pop-up window.



Or use `install.packages()` in the console

```
install.packages("ggplot2")
```

You only need to install packages once for each computer/version of R.

To access the functions in R, you need to load the packages with the `library()` command each time start a new R session.

```
library(ggplot2)
```

Note: There are *MANY* ways to do the same thing in R.

For example, Cronbach's alpha (a common measure of internal scale consistency) is not built into base R.

- psych package: alpha()
- psy package: cronbach()
- ltm package: cronbach.alpha()
- fmsb package: CronbachAlpha()
- epiDisplay package: alpha()

1.2.4 The Environment window

Objects we create in R are stored in the Environment tab. If this gets cluttered and you want to "start fresh", you can clear the environment:

Click on the "broom" icon:

Or type into the console:

```
rm(list=ls())
```

1.2.5 The History window

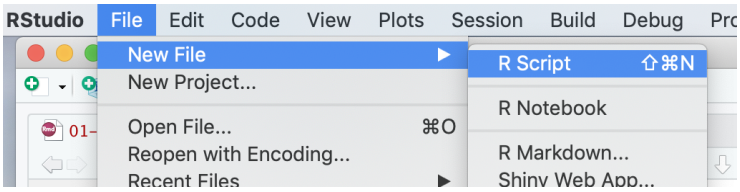
This window provides a list of all the commands you have entered in the console

While this can serve as a nice reminder of what you have done, think about how well you will remember what all this means tomorrow. Or six months from now.

For procedures you care about (for example, anything with data), you will want to capture what you do in an *R script*.

1.3 R script

To open a new script, select File –> New File –> R Script



R scripts

- Contain R code for a given analysis/project (just as you would type it in the console)
- Include comments (R ignores lines starting with #)
- Need to be written in order (i.e., code to create an object needs to come before code that uses the object)
- Can be run line by line ("run" or "cmd/cntl+enter") or all at once ("source")

Let's prep our R script for data exploration. You can use comments to describe what lines of code do, or as breaks to set apart information about the script from the script itself.

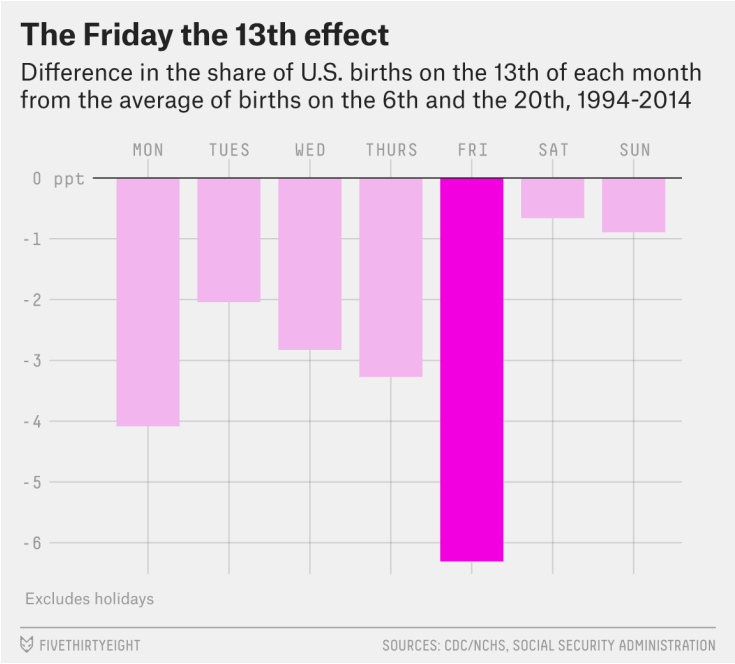
```
########################################
## Intro to R Workshop
##
##  2019-08-14
########################################
```

```r
# Calculate 3 squared
3^2

# Save result to use later
my_result <- 3^2
```

You can run code from a script in the console by pressing "cmd+enter" (Mac) or "cntl+enter" (PC)

1.3.1 Working with data

The follow data were used in FiveThirtyEight's article "Some People Are Too Superstitious To Have A Baby On Friday The 13th", which analyzes rates of scheduled births on the 13th of the month.



This analysis uses the following data:

Show 10 entries Search:

	year	month	date_of_month	date	day_of_week	births
1	2000	1	1	2000-01-01	Sat	9083
2	2000	1	2	2000-01-02	Sun	8006
3	2000	1	3	2000-01-03	Mon	11363
4	2000	1	4	2000-01-04	Tues	13032
5	2000	1	5	2000-01-05	Wed	12558
6	2000	1	6	2000-01-06	Thurs	12466
7	2000	1	7	2000-01-07	Fri	12516
8	2000	1	8	2000-01-08	Sat	8934
9	2000	1	9	2000-01-09	Sun	7949
10	2000	1	10	2000-01-10	Mon	11668

Showing 1 to 10 of 5,479 entries Previous 1 2 3 4 5 … 548 Next

Data Structure

Tidy data tables have two key features:

1. Each row represents a single **observational unit** of the sample.
2. Each column represents a **variable**, i.e., an attribute of the cases.
3. There are no extras in the dataset - no row summaries, column summaries, data entry notes, comments, graphs, etc. All comments about the data collection, variables, etc should be provided in a separate **codebook**.

Question: What are the units of observation in the above data? What are the variables?

Importing Data

You can read data into R from a file on your computer, a dataset on the internet, or use data that are included in a package in R.

Luckily for us, the Friday the 13th data are already stored within R in the `fivethirtyeight` package.

IF AND ONLY IF you did not install the `fivethirtyeight` package before the workshop, you can do so now by typing the following code **in the console**:

```
install.packages("fivethirtyeight", dependencies = TRUE)
```

You only need to install a package one time for a given computer and version of R. However, each time you want to use functions or data in a package, you will need to load the package into R.

Put the following code **in your script** (*hint: future you will thank present you if you include the comments as well)*:

```
#load required package
library(fivethirtyeight)

#load in data
data(US_births_2000_2014)
```

To learn more about this data, you can access **codebook** information:

```
#See codebook information about the data
?US_births_2000_2014
```

Examining data structure in R

Before we do any analysis, we have to understand the structure of our data. Try each of the following by typing them into your script and running them:

```
# View the data table in a separate tab
View(US_births_2000_2014)

# Check out the first rows in the console
head(US_births_2000_2014)

# Obtain the data dimensions: rows x columns
dim(US_births_2000_2014)

# Get the variable names
names(US_births_2000_2014)

# Look at summary information for each variable
summary(US_births_2000_2014)
```

Examining specific variables

```
# Access a single variable using "$"
US_births_2000_2014$year
US_births_2000_2014$month

# Determine levels/categories of categorical variables
levels(factor(US_births_2000_2014$year))
levels(factor(US_births_2000_2014$day_of_week))
```

Importing data from other sources

R can take in data in many different formats, including from proprietary statistical program such as SPSS, SAS, or Stata. However, the most commonly used files are comma delimited.

- Comma delimited: `read.csv()`
- Tab delimited: `read.delim()`
- Excel files: `library(readxl)`; `read_excel()`
- SPSS files: `library(haven)`; `read_spss()`
- Stata files: `library(haven)`; `read_dta()`
- SAS files: `library(haven)`; `read_sas()`

To read in a data file, assign the output of the `read.csv()` function to an object name.

```
# Example csv data on women's shoes
shoes <- read.csv(file="https://z.umn.edu/shoe-data")
```

You can read in files from your computer by replacing the web address with a file path.

```
# Example csv from computer
shoes <- read.csv(file="~/Documents/MyProject/data/womens-shoes.csv")
```

Exercises

First, let's look at the `shoes` data above. This data contains price information for various brands of women's shoes.

1. Load the data from the website into your environment with the `read.csv` command, if you haven't already.
2. View the dataset in a separate tab
3. Check out the first 6 cases of the data
4. How many rows/columns does this dataset have?
5. Access the variable `prices.amountMax` alone. What is the mean maximum price of shoes in the dataset?

Next, let's look at the `bechdel` data in the `fivethirtyeight` package. This data was used in the fivethirtyeight.com's article "The Dollar-And-Cents Case Against Hollywood's Exclusion of Women" which analyzes movies that do/don't pass the **Bechdel test**.

A movie passes the test if it meets the following criteria:

- there are ≥ 2 female characters
- the female characters talk to each other
- at least 1 time, they talk about something other than a male character

1. Load the data into your environment and examine the codebook information.
2. View the dataset in a separate tab.
3. Look at the summary information for the data.
4. What are the units of observation (rows)?
5. What are the names of the variables?
6. Access the variable `clean_test` alone. What are the levels of this variable?
7. Access the variable `budget_2013` alone. What is the largest amount spent on a movie in this dataset?

Solutions:

Shoes data:

```
#1. Load the data from the website into your environment with the `read.csv` command, if you have
shoes <- read.csv(file="https://z.umn.edu/shoe-data")

#2. View the dataset in a separate tab
View(shoes)

#3. Check out the first 6 cases of the data
head(shoes)

#4. How many rows/columns does this dataset have?
dim(shoes)

#5. Access the variable `prices.amountMax` alone. What is the mean maximum price of shoes in the
shoes$prices.amountMax

#can look at the summary information
summary(shoes$prices.amountMax)

#or use the mean command
mean(shoes$prices.amountMax)
```

Bechdel data:

```
#1. Load the data into your environment and examine the codebook information.
library(fivethirtyeight)
data("bechdel")
?bechdel


#2. View the dataset in a separate tab.
View(bechdel)


#3. Look at the summary information for the data.
summary(bechdel)


#4. What are the units of observation (rows)?
# each row = a movie


#5. What are the names of the variables?
names(bechdel)
```

```
#6. Access the variable `clean_test` alone. What are the levels of this variable?
bechdel$clean_test
levels(factor(bechdel$clean_test))

#7. Access the variable `budget_2013` alone. What is the largest amount spent on a mov
bechdel$budget_2013

#can look at summary
summary(bechdel$budget_2013)

#or use max() function
max(bechdel$budget_2013)
```

Chapter 2

Reproducible Reports with R Markdown

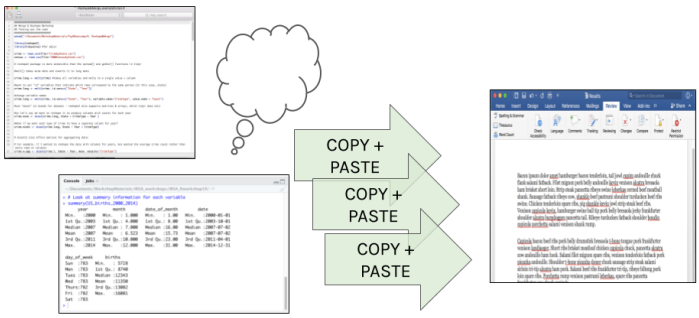
Author: Alicia Hofelich Mohr

2.1 Overview of R Markdown

A not-so-great workflow

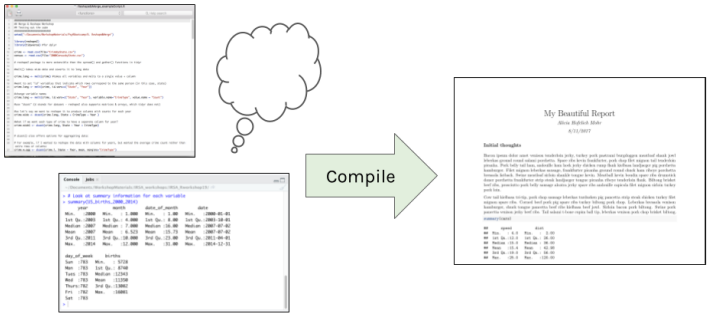
While R scripts contain all the R code you produce in a format that can be easily re-run, they do not include the output or results of your analysis. Comments can be used to describe reactions or interpretations of results, but they are not the best way to present this interpretation in a readable way.

Therefore, to use R for data analysis and reporting, you may find yourself engaging in a workflow that involves a lot of copying and pasting (or transcribing) between your R script, console, and a document editor.



A better way

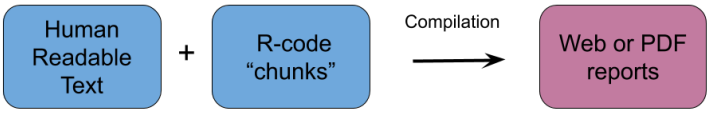
What if there was a better way to run R code, capture output, and present interpretation together?



There is - R Markdown!

Reproducible Reports

R Markdown is a considered a *literate programming* tool, which is a form of coding that combines text meant for a human to understand with snippets or sections of code for a computer to execute.



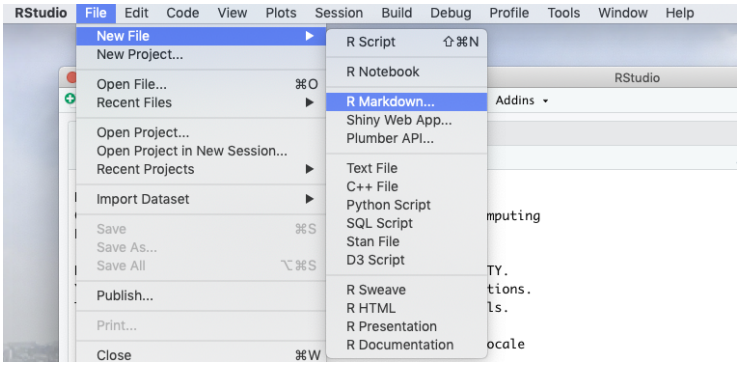
R Markdown uses **Markdown** as its human readable text language. Markdown is a straightforward text-to-HTML tool that can format text for the web. This is interspersed with **code chunks** R can read and execute.

If you are familiar with the text editing language LaTeX, there is a fully LaTeX option for reproducible reports in R using Sweave.

	R Markdown (.Rmd)	R Sweave (.Rnw)
Human-readable text	Markdown	LaTeX
R code chunks	```{r chunk_name, options…} # R code goes here! ```	<<chunk_name, options…>>= # R code goes here! @
How do I compile it?	knitr	knitr or Sweave
Default output	HTML	PDF

In R Studio

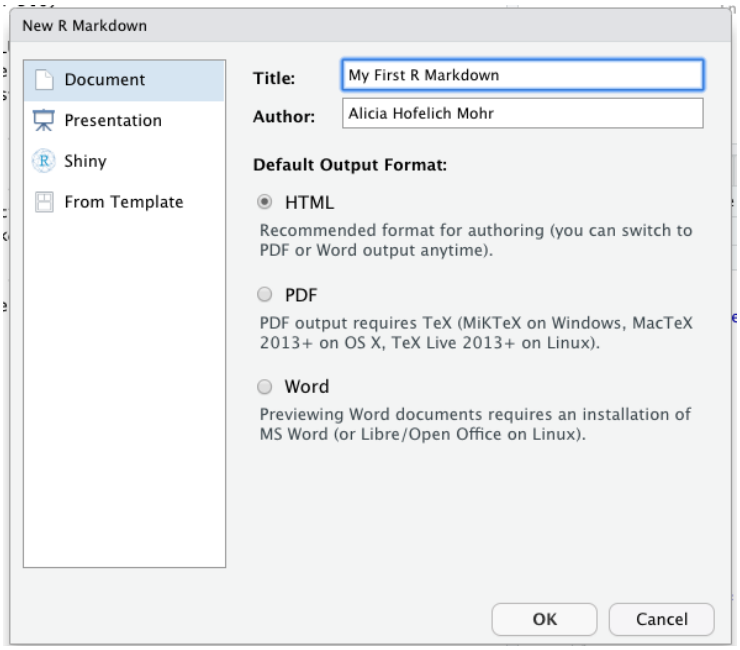
To open a new R Markdown file, go to File –> New File –> R Markdown



If you have not yet installed the `rmarkdown` package, you may see a window prompting you to install that package and any related dependencies. If you see

this, click to install the required packages.

Once you have all the packages installed, you will see a pop-up window with various options for the type of R Markdown document and the output. There are many different options for each - in fact, this website was created using R Markdown!

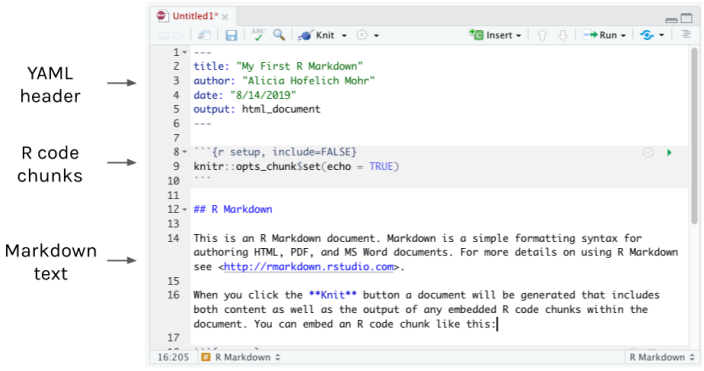


For this workshop, we will stick with the default options and click ok.

The .Rmd File

R will automatically open an R Markdown (.Rmd) file with some example text and code already included.

An R Markdown file has the following components - YAML, R-code chunks, and Markdown text.



To compile the code and text together, press the "knit" button at the top of the file. This will by default open the HTML report in a new window and save a .html file in the folder with the .rmd file.

2.2 YAML

The YAML section contains the front matter of your document - title, authors, date, etc. It also controls the formatting of the report output.

```
1 ---
2 title: "My First R Markdown"
3 author: "Alicia Hofelich Mohr"
4 date: "8/14/2019"
5 output: html_document
6 ---
```

For example, you can add a table of contents to your document:

```
---
title: "My First R Markdown"
author: "Alicia Hofelich Mohr"
date: "8/14/2019"
output:
  html_document:
    toc: true
---
```

Or add options to make the table of contents float as you scroll:

```
---
title: "My First R Markdown"
author: "Alicia Hofelich Mohr"
date: "8/14/2019"
output:
  html_document:
    toc: true
    toc_float: true
---
```

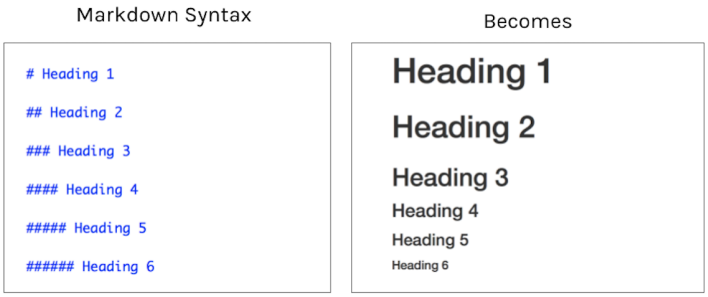
There are many other adjustments you can make to the document. The Definitive Guide to R Markdown website is a great place to learn about these options.

2.3 Markdown

Markdown is an easy to use text formatting language that can be converted to HTML. The markdown portion of the document is where you add in headings, narrative information about the analysis, and any interpretations.

Headings

Headings are indicated with `#`, with the number of `#` corresponding to the header level.



Try it - Add a new first and third level heading to your example R Markdown file.

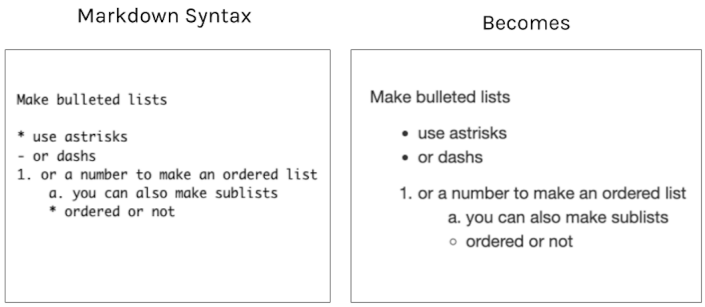
Text formatting

For plain text, just type as is.

- `**` or `__` around words for **bold** text
- `*` for `_` around words for *italic* text
- `[linked text](link-url)` for hyperlinks

Try it - make a word in the first paragraph of your example R Markdown file bold; make another one italic.

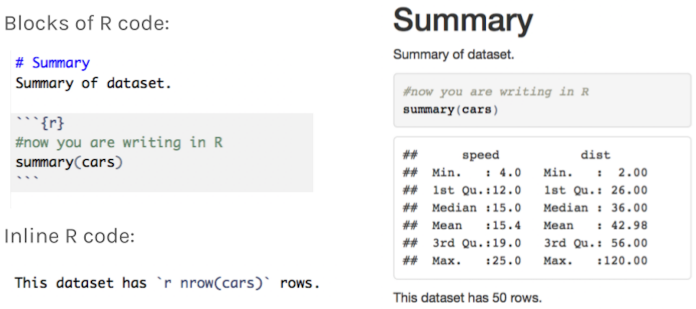
You can also create bulleted lists in Markdown:



Try creating a bulleted list in your example R Markdown document.

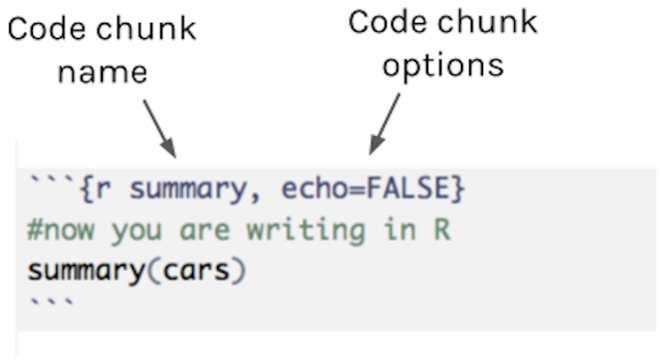
2.4 R chunks

To run and display R code and its output, you'll need to add in an R code chunk.



Chunk options

You can specify how you want the R code and output to appear inside the R code chunk.



Using the `cars` chunk in the example R Markdown file, test what each option does:

- `echo=FALSE`
- `results='hide'`
- `comment=""`

Chunk options

option	default	effect
`eval`	TRUE	Whether to evaluate the code and include its results
`echo`	TRUE	Whether to display code along with its results
`warning`	TRUE	Whether to display warnings
`error`	FALSE	Whether to display errors
`message`	TRUE	Whether to display messages
`tidy`	FALSE	Whether to reformat code in a tidy way when displaying it
`results`	"markup"	"markup", "asis", "hold", or "hide"
`cache`	FALSE	Whether to cache results for future renders
`comment`	"##"	Comment character to preface results with
`fig.width`	7	Width in inches for plots created in chunk
`fig.height`	7	Height in inches for plots created in chunk

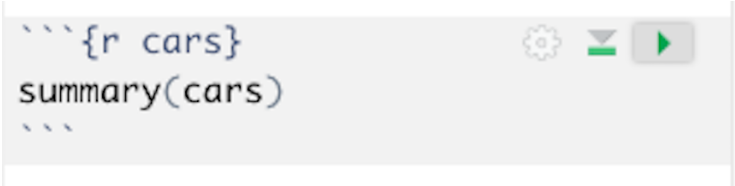
2.5 Working in R Markdown

Working in R Markdown and R scripts have several *similarities*:

- Within R chunks, run a line of R code by clicking on "Run" or pressing "cmd/cntrl + enter".
- Order matters - need to create objects before computing on them and load packages before using functions from them.
- Need to have objects present in your environment in order for R code to work in console

However there are some *differences*:

- When "knit", all the R code is run in separate session
- Objects created in one chunk are available to use in later chunks
- You can run all the code in a single chunk by pressing the green triangle button on the right side of the chunk



Exercises

Revisit the `bechdel` data exploration exercises from the previous session.

This time, we will create an R Markdown document with these findings. You can start fresh in a new R Markdown document, or use the existing one you have been modifying.

- Start by creating separate R chunks for each exercise question (*skip question 2, as View() does not work well in R Markdown*)
- Write the question above each chunk in Markdown
- Put the R code for the exercise in an R code chunk (remove the comments - that will be in the Markdown above)
- Write a sentence interpreting the result below each chunk in Markdown
- Knit the document to produce your report

Chapter 3

Understanding data: Visualization

Author: Alicia Johnson

Getting started

As you get settled in, please open a new Rmd (R markdown document) and prepare the data:

```
# Import data from the fivethirtyeight package
library(fivethirtyeight)
data("US_births_2000_2014")

# Load packages
library(ggplot2)  # for visualizing data
library(dplyr)    # for transforming data

# Define a dataset with only 2014 births
# (will discuss this syntax in the next module)
only_2014 <- US_births_2000_2014 %>%
  filter(year == 2014)
```

3.1 Goal: Understanding Through Visualizations

In this module we'll focus on the **"Visualise"** step in the data science workflow:

source: Wickham & Grolemund: R for Data Science

Motivation

We investigated the basic structure of the **tidy** births data set in the previous activity:

Show 4 entries Search:

	year	month	date_of_month	date	day_of_week	births
1	2000	1	3	2000-01-03	Mon	11363
2	2000	1	4	2000-01-04	Tues	13032
3	2000	1	5	2000-01-05	Wed	12558
4	2000	1	6	2000-01-06	Thurs	12466

Showing 1 to 4 of 5,477 entries Previous 1 2 3 4 5 … 1370 Next

Now we can start to ask some **research questions** to better **understand** the data:

- What's the *typical* number of births per day?
- To what degree do the number of births *vary* from day to day?
- Is there a *relationship* between the number of births and time of year? Between the number of births and day of week?
- Is there any evidence that people are too superstitious to give birth on Friday the 13th?

Visualizing the data is the first natural step in answering these questions. Why?

- Visualizations help us understand what we're working with: What are the scales of our variables? Are there any outliers, i.e. unusual cases? What are the patterns among our variables?
- This understanding will inform our next steps: What statistical tool / model is appropriate?
- Once our analysis is complete, visualizations are a powerful way to communicate our findings and tell a story.

3.2 tidyverse: ggplot

We'll construct visualizations using the `ggplot()` function in the `ggplot2` package. This package is part of the broader **tidyverse**, a "collection of R packages designed for data science" which "share an underlying design philosophy, grammar, and data structures"[1].

[1] https://www.tidyverse.org/

Though the `ggplot()` learning curve can be steep, its "grammar" is intuitive and generalizable once mastered. The best way to learn about `ggplot()` is to just play around. **Don't worry about memorizing the syntax.** Rather, focus on the *patterns* and *potential* of their application. There's a helpful cheat sheet for future reference:

GGPLOT CHEAT SHEET

3.3 Univariate visualizations

Our primary variable of interest in the birthday data is the **quantitative** `births` variable which summarizes the number of U.S. births on each individual day between 2000 and 2014. Before examining the relationship between `births` and any other variable, it's important to develop an understanding of the **univariate** patterns. We'll explore 2 of many methods for visualizing quantitative variables: histograms & density plots.

3.3.1 Histograms

A **histogram** is constructed by: (1) dividing up the observed range of the variable into 'bins' of equal width; and (2) counting up the number of cases that fall into each bin. Try out the code below.

1. **Construct a histogram for the 2014 `births`**
   Try out the code below that builds up from a simple to a customized histogram. At each step determine how each piece of code contributes to the plot and, for future reference, record this in a `#` comment.

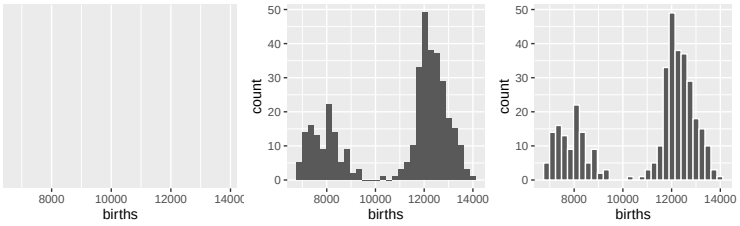
```
# ???
ggplot(only_2014, aes(x = births))

# ???
ggplot(only_2014, aes(x = births)) +
  geom_histogram()

# ???
ggplot(only_2014, aes(x = births)) +
  geom_histogram(color = "white")
```

2. **Examine the histogram**
   Based on the histogram alone...
   - Examine the *trend*: roughly, what's the *typical* number of births per day?

- Examine the *variability*: roughly, what was the smallest number of births? The largest?
- Describe the *shape* of the histogram? Do you have any guesses about what explains this shape?
- How wide are the `births` bins?
- What was the most common bin? On roughly how many days did the number of births fall into this bin?

3. **You try**
   Construct a histogram of `births` using the entire `US_births_2000_2014` which spans from 2000 to 2014. How, if at all, do the trends and variability in `births` across all of these years compare to those in 2014 alone?

In summary, you constructed:



3.3.2 Density plots

A **density plot** is essentially a smooth version of the histogram. Instead of sorting cases into discrete bins, the "density" of cases is calculated across the entire range of values. The greater the number of cases, the greater the density! The density is then scaled so that the area under the density curve **always equals 1** and the area under any fraction of the curve represents the fraction of cases that lie in that range.

4. **Construct a density plot for the 2014 `births`**
   Take note of what each step does.

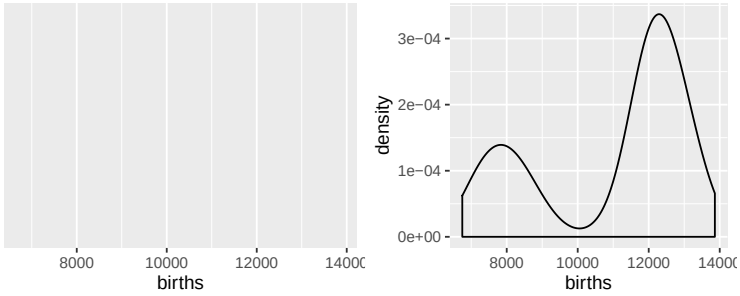
```
# ???
ggplot(only_2014, aes(x = births))

# ???
ggplot(only_2014, aes(x = births)) +
    geom_density()
```

5. **Examine the density plot for the 2014 `births`**

- Does this visualization provide any information that the histogram doesn't?

- Which do you prefer for visualizing the `births` data, the density plot or histogram? Why?

In summary, you constructed:



3.3.3 Pause

We'll move onto the next section together. If you finish early, take this time to revisit the `ggplot()` syntax. Take some notes on the common steps / patterns utilized in building a `ggplot`.

3.4 Visualizing relationships

Consider the data on just the first 6 days of 2014:

Show 6 entries Search:

	year	month	date_of_month	date	day_of_week	births
1	2014	1	1	2014-01-01	Wed	8018
2	2014	1	2	2014-01-02	Thurs	11171
3	2014	1	3	2014-01-03	Fri	12317
4	2014	1	4	2014-01-04	Sat	8199
5	2014	1	5	2014-01-05	Sun	7174
6	2014	1	6	2014-01-06	Mon	11400

Showing 1 to 6 of 6 entries Previous 1 Next

6. **Pause and think**
   We saw some goofy patterns in the `births` data. Perhaps this might be explained by day of year, month, or day of week. However, before constructing graphics of the relationships among these variables, we need to understand what features these graphics should have. Challenge yourself to think about how we might visualize the relationships among the following sets of variables:

   - `births` vs `date`
   - `births` vs `day_of_week`
   - `births` vs `date` *and* `day_of_week` (in 1 plot)

3.4.1 Quantitative vs quantitative

7. **Scatterplots of 2 quantitative variables**
   Let's start by visualizing the relationship between the number of daily `births` and the date or day of year for the 2014 births data. We can think of both as *quantitative variables*. In order to visualize their numerical scales, both variables need an axis. On these axes, we can represent each row (day) by a dot.

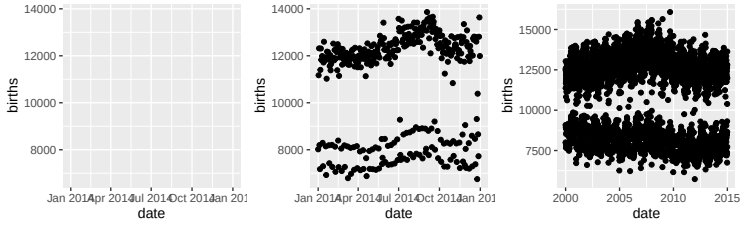
```
# Just a graphics frame
ggplot(only_2014, aes(y = births, x = date))

# Add a scatterplot layer
ggplot(only_2014, aes(y = births, x = date)) +
  geom_point()
```

8. **You try**
   Construct a scatterplot of `births` vs `date` using the 2000-2014 `US_births_2000_2014`.

9. **Examine the scatterplots**
   - Do you observe any *seasonal* trends? If so, at what time of year do `births` tend to be the highest? The lowest?

- Do you observe any birth trends across 2000 to 2014? When were the most babies born? The fewest?
- Any other strange patterns? What do you think might explain these patterns?

In summary:



3.4.2 Quantitative vs categorical

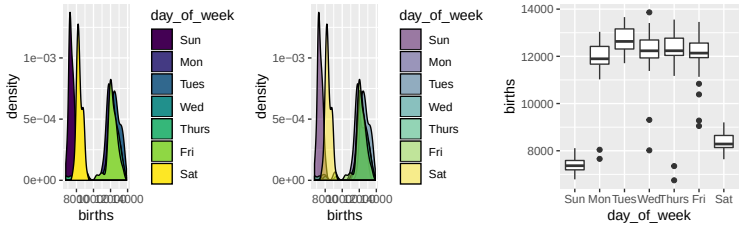
Next, consider the relationship between `births` (quantitative) and `day_of_week` (categorical). Since `day_of_week` is categorical, visualizing this variable requires a *visual grouping* mechanism. We'll consider a couple approaches.

10. **Construct side-by-side plots**
    Again focus on the general patterns in the syntax and how we've changed the earlier code.

```
# Density plots by group
ggplot(only_2014, aes(x = births, fill = day_of_week)) +
  geom_density()

# ???
ggplot(only_2014, aes(x = births, fill = day_of_week)) +
  geom_density(alpha = 0.5)

# Or use boxplots!
ggplot(only_2014, aes(x = day_of_week, y = births)) +
  geom_boxplot()
```

11. **Interpreting the side-by-side plots**
    - What do the side-by-side plots reveal about birth patterns?
    - Which day tends to see the fewest births? The greatest?
    - There are some outliers here. What do you suspect explains these outliers?

In summary:



3.4.3 Quantitative vs quantitative vs categorical

12. **Three variables**
    If `day_of_week` and `date` both explain some of the variability in `births`, why not include both in our analysis?! Let's. Note how the code changes.

```
# births vs date
ggplot(only_2014, aes(y = births, x = date)) +
  geom_point()

# births vs date AND day_of_week
ggplot(only_2014, aes(y = births, x = date, color = day_of_week)) +
  geom_point()
```

13. **Visualize the trend**
    Finally, we can add a visual summary of the trend in `births` by date from 2000-2014. This is visual summary represents a **model** (which you'll discuss tomorrow).

```
# add trend lines
ggplot(US_births_2000_2014, aes(y = births, x = date, color = day_of_week)) +
  geom_point() +
  geom_smooth()

# change the smoothness
ggplot(US_births_2000_2014, aes(y = births, x = date, color = day_of_week)) +
  geom_point() +
  geom_smooth(span = 0.1)

# You try: keep the smooths but get rid of the dots
```

14. **Summarize**
    Summarize what you've learned about birth patterns.
    - In what month do the most babies tend to be born? The least?
    - On what day of week do the most babies tend to be born? The least?

- In what year were the most babies born? The least?
- What explains the outliers in the dataset?

3.5 Exercises

NOTE: Please do not feel pressured to blaze through these exercises. There are more here than you will finish during the workshop – this is simply meant to give you more to practice when you leave.

In the exercises, you'll return to the Bechdel test data:

```
library(fivethirtyeight)
data(bechdel)
dim(bechdel)
## [1] 1794   15
```

Solutions are provided below.

3.5.1 Questions

1. **Visualizing a categorical variable**
   The categorical `binary` variable records whether each film in the dataset passes or fails the Bechdel test:

   ```
   levels(factor(bechdel$binary))
   ```

   A bar chart provides a simple visualization of this variable. Summarize what you learn from this plot: roughly how many of the films pass / fail the test?

   ```
   # Bar plot
   ggplot(bechdel, aes(x = binary)) +
     geom_bar()
   ```

2. **Visualize potential "predictors"**
   Our ultimate goal will be to better understand which films pass / fail the Bechdel test. For example, can this be explained the year in which it was made? The film's budget? The amount of money it makes at the box office? Before we can answer these questions, it's important to understand the scales of these variables. For each variable below, construct and summarize a **univariate visualization**.
   a. `year`: the year in which the film was released

   b. `budget_2013`: the film's budget (in 2013 $)

c. `domgross_2013`: the film's total domestic gross (in 2013 $)

3. **binary vs x**
   To answer each question below, construct and summarize a visualization between the given pair of variables. The hints below apply in each scenario.
   a. Is there a relationship between the `year` in which a film is made and whether is passes the Bechdel (`binary`)?

   b. Do movies that pass the Bechdel test tend to receive more or less funding (as measured by `budget_2013`) than those that fail the test?

   c. Do movies that pass the Bechdel test tend to make more or less money (as measured by `domgross_2013`) than those that fail the test?

   Hints
   - Is `binary` categorical or quantitative? What about the other variable? Thus what visualization tool might you use?

   - Fill in the following:

```
ggplot(___, aes(x = ___, y = ___)) +
  ___()
ggplot(___, aes(x = ___, fill = ___)) +
  ___()
```

4. **binary vs x1 and x2**
   For each question below, construct and summarize a helpful visualization that includes a `geom_smooth()`.
   a. How have the budgets (`budget_2013`) of films that pass / fail the Bechdel test (`binary`) changed and compared over time (`year`)? HINT: Put `year` on the x-axis.

   b. How have the gross profits (`domgross_2013`) of films that pass / fail the Bechdel test (`binary`) changed and compared over time (`year`)? HINT: Put `year` on the x-axis.
   c. How does the "return on investment" compare for films that pass / fail the Bechdel test? Specifically, examine the relationship between `domgross_2013` and `budget_2013` for films that pass / fail the Bechdel test (`binary`). HINT: Put `budget_2013` on the x-axis.

5. **Learn something new**
   It's impossible to memorize everything you'll ever need to know about `ggplot()`. Luckily, there's a strong online community of R users (eg: stackoverflow). Utilize a web search to make the following plot changes.

   a. Add more meaningful axis labels and a title to your plot of `domgross_2013` vs `budget_2013`. For example, change `budget_2013` to "budget (in 2013 dollars)". Example search term: "change axis labels ggplot2 stack overflow".

b. In the same plot, change the points from black to red. Example search term: change geom_point color ggplot2

6. **Extra visualization practice**
Bike sharing is becoming a more popular means of transportation in many cities. The dataset we will analyze in this assignment comes from Capital Bikeshare, the bike-sharing service for the Washington DC area. The dataset originally comes from the UCI Machine Learning Repository and the version we'll work with is made available at Macalester Prof Shuman's website:

```
bikes <- read.csv("https://www.macalester.edu/~dshuman1/data/155/bike_share.csv")
```

Codebook

variable	meaning
`date`	date in format YYYY-MM-DD
`season`	winter, spring, summer, or fall
`year`	2011 or 2012
`month`	3-letter month abbreviation
`day_of_week`	3-letter abbreviation for day of week
`weekend`	TRUE if the case is a weekend, FALSE otherwise
`holiday`	is the day a holiday? (yes or no)
`temp_actual`	temperature in degrees Fahrenheit
`temp_feel`	what the temperature *feels* like in degrees Fahrenheit
`humidity`	fraction from 0 to 1 giving the humidity level
`windspeed`	wind speed in miles per hour
`weather_cat`	`categ1`: clear to partly cloudy `categ2`: mist + some clouds `categ3`: light precipitation to th
`riders_casual`	count of daily rides by casual users (non-registered users)
`riders_registered`	count of daily rides by registered users
`riders_total`	count of total daily rides (`riders_casual + riders_registered`)

Utilize visualizations to better understand how `riders_registered` might depend upon `windspeed`, `temp_feel`, `weekend`, or `season`. For example, consider the following prompts.

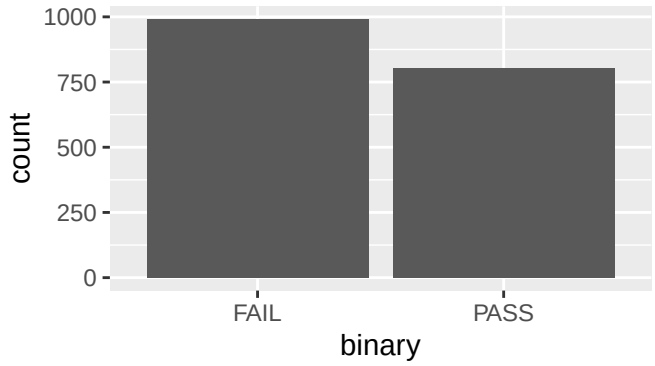
- Examine the following relationships. Which of the variables appears to be the best predictor of the number of riders?
  - `riders_registered` vs `windspeed`
  - `riders_registered` vs `temp_feel`
  - `riders_registered` vs `weekend`
  - `riders_registered` vs `season`.

- Examine how `riders_registered` depends upon `windspeed` and `weekend`.
- **Learn something new:** How can you examine the relationship between `riders_registered` and *two* categorical variables: `season` and `weekend`?
- **Learn something new:** How can you examine the relationship between `riders_registered` and *two* quantitative variables: `windspeed` and `temperature`?

3.5.2 Solutions

1. More films fail than pass the Bechdel test (~1000 vs ~800).

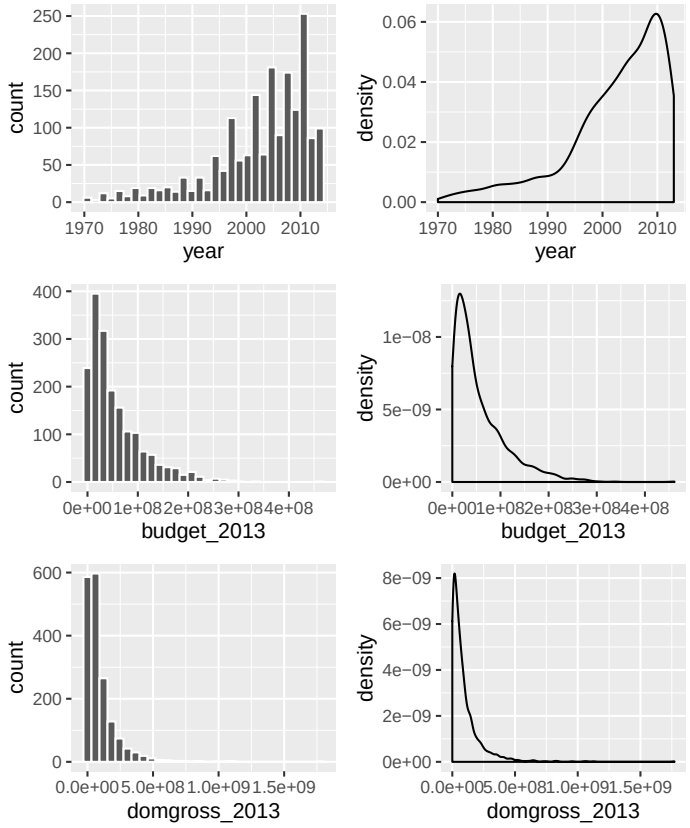
```
ggplot(bechdel, aes(x = binary)) +
  geom_bar()
```



2. .

```
# a
ggplot(bechdel, aes(x = year)) +
  geom_histogram(color = "white")
ggplot(bechdel, aes(x = year)) +
  geom_density()

# b
ggplot(bechdel, aes(x = budget_2013)) +
  geom_histogram(color = "white")
ggplot(bechdel, aes(x = budget_2013)) +
  geom_density()

# c
ggplot(bechdel, aes(x = domgross_2013)) +
  geom_histogram(color = "white")
```

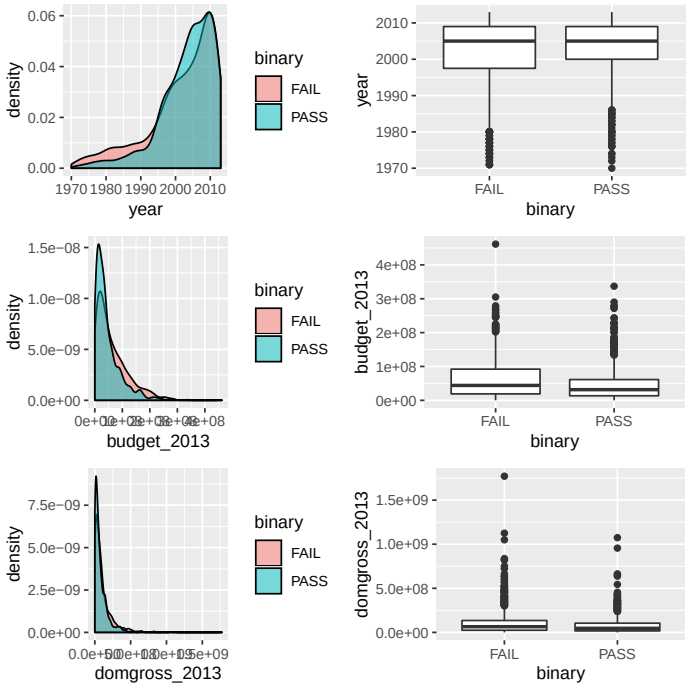
```
ggplot(bechdel, aes(x = domgross_2013)) +
  geom_density()
```



3. Over time, more films have been passing the Bechdel test. Further, films that fail the Bechdel tend to have bigger budgets and to make more money.

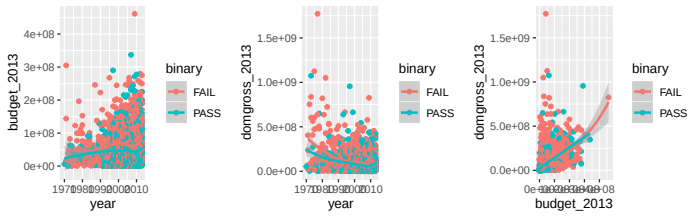
```
# a
ggplot(bechdel, aes(x = year, fill = binary)) +
  geom_density(alpha = 0.5)
ggplot(bechdel, aes(x = binary, y = year)) +
  geom_boxplot()

# b
ggplot(bechdel, aes(x = budget_2013, fill = binary)) +
  geom_density(alpha = 0.5)
ggplot(bechdel, aes(x = binary, y = budget_2013)) +
  geom_boxplot()

# c
```

```
ggplot(bechdel, aes(x = domgross_2013, fill = binary)) +
  geom_density(alpha = 0.5)
ggplot(bechdel, aes(x = binary, y = domgross_2013)) +
  geom_boxplot()
```



4. .

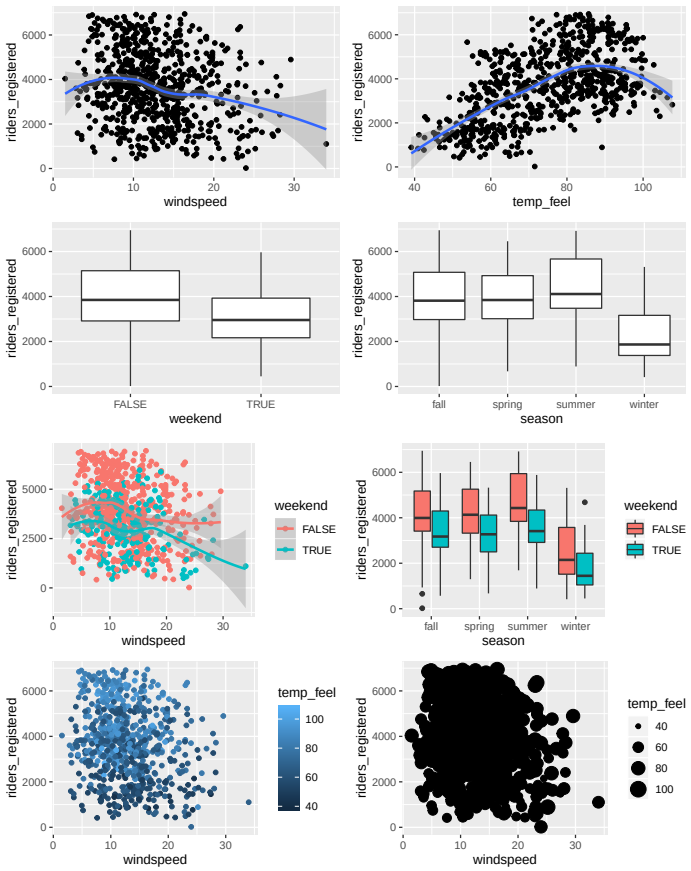
```
# a
ggplot(bechdel, aes(x = year, y = budget_2013, color = binary)) +
  geom_point() +
  geom_smooth()

# b
ggplot(bechdel, aes(x = year, y = domgross_2013, color = binary)) +
  geom_point() +
  geom_smooth()

# c
ggplot(bechdel, aes(x = budget_2013, y = domgross_2013, color = binary)) +
  geom_point() +
  geom_smooth()
```



5. Here are some resources:
   a. https://stackoverflow.com/questions/10438752/adding-x-and-y-axis-labels-in-ggplot2
   b. https://ggplot2.tidyverse.org/reference/geom_point.html
6. Choose your own adventure! Here are some ideas:

```
bikes <- read.csv("https://www.macalester.edu/~dshuman1/data/155/bike_share.csv")

# rides vs windspeed
ggplot(bikes, aes(y = riders_registered, x = windspeed)) +
    geom_point() +
    geom_smooth()

# rides vs temperature
ggplot(bikes, aes(y = riders_registered, x = temp_feel)) +
    geom_point() +
    geom_smooth()

# rides by weekend
ggplot(bikes, aes(y = riders_registered, x = weekend)) +
    geom_boxplot()

# rides by season
ggplot(bikes, aes(y = riders_registered, x = season)) +
    geom_boxplot()

# rides by windspeed and weekend
ggplot(bikes, aes(y = riders_registered, x = windspeed, color = weekend)) +
    geom_point() +
    geom_smooth()

# EXTRA: rides by season and weekend
ggplot(bikes, aes(y = riders_registered, x = season, fill = weekend)) +
    geom_boxplot()

# EXTRA: rides by windspeed and temperature
ggplot(bikes, aes(y = riders_registered, x = windspeed, color = temp_feel)) +
    geom_point()
```

```
ggplot(bikes, aes(y = riders_registered, x = windspeed, size = temp_feel)) +
    geom_point()
```



3.6 Resources

Data visualization is an art and science. To learn more:

- Hadley Wickham, the author of the `ggplot2` package has a free online book.
- Nathan Yau does a great job of discussing and presenting the core principles of data visualization. His flowing data blog is a great resource for inspiration and his books, Visualize This and Data Points, are great resources for digging deeper into the principles.
- Stack Overflow is a great online community board to learn more about `ggplot()`.

Chapter 4

Understanding data: Transformations

Author: Alicia Johnson

Getting started

As you get settled in, please open a new Rmd (R markdown document) and prepare the data:

```
# Import data from the fivethirtyeight package
library(fivethirtyeight)
data("candy_rankings")

# Rename the dataset to something shorter
candy <- candy_rankings

# Load packages
library(ggplot2)  # for visualizing data
library(dplyr)    # for transforming data
```

4.1 Goal: Understanding Through Transformations

In this module we'll focus on the **"Transform"** step within the data science workflow:

source: Wickham & Grolemund: R for Data Science

In the previous modules, we imported and visualized tidy data. These visualizations provided us with important insights into the patterns in our data. Data transformations are also fundamental to understanding data:

- We can utilize **data transformations** to:
  - define & explore new variables;
  - filter & focus on subsets of the data; and
  - reorder the data.
- Simple numerical **summaries** calculated via data transformations complement, formalize, and support the patterns observed in our visual summaries.

4.2 tidyverse: dplyr

We'll construct data transformations using the `dplyr`, a package within the broader tidyverse. Like `ggplot`, the `dplyr` "grammar" is intuitive and generalizable once mastered. The best way to learn about `dplyr` is to just play around. Don't worry about memorizing the syntax. Rather, focus on the *patterns* and *potential* of their application. There's a helpful cheat sheet for future reference:

DPLYR CHEAT SHEET

In the `dplyr` grammar, there are 5 *verbs* (actions):

verb	action
`arrange()`	reorder the *rows*
`filter()`	take a subset of *rows*
`select()`	take a subset of *columns*
`mutate()`	create a new variable, ie. *column*
`summarize()`	calculate a numerical summary of a variable, i.e. *column*

The general syntax for applying these verbs is below, where we call "`%>%`" a "pipe":

```
my_dataset %>%
  verb(___)
```

Just as we can add layers to a `ggplot` utilizing "`+`", we can implement sequential data transformations utilizing "'%>%":

```
my_dataset %>%
  verb1(___) %>%
  verb2(___)
```

4.3 Transforming rows: `arrange()` and `filter()`

The fivethirtyeight article The Ultimate Halloween Candy Power Ranking analyzes the data from this experiment which presented subjects with a series of head-to-head candy matchups and asked them to indicate which candy they preferred. You **imported** these **tidy** data above and named it `candy`.

1. **Basics**

```
# Take a glimpse at the data.  What's the unit of observation?

# How much data do we have?
```

Solution

```
# Take a glimpse at the data.  What's the unit of observation?
head(candy)

# How much data do we have?
dim(candy)
```

2. **arrange() rows**
Summarize what the `arrange()` function does.

```
# ???
candy %>%
  arrange(winpercent)
candy %>%
  arrange(winpercent) %>%
  head()

# ???
candy %>%
  arrange(desc(winpercent))
candy %>%
  arrange(desc(winpercent)) %>%
  head()

# Store the new dataset
candy_arranged <- candy %>%
  arrange(winpercent)

# You try: arrange from least to most expensive


# You try: arrange from most to least expensive
```

Solution

```
# arrange candy by popularity in ascending order
candy %>%
  arrange(winpercent)
candy %>%
  arrange(winpercent) %>%
  head()

# arrange candy by popularity in descending order
candy %>%
  arrange(desc(winpercent))
candy %>%
  arrange(desc(winpercent)) %>%
  head()

# Store the new dataset
candy_arranged <- candy %>%
  arrange(winpercent)

# You try: arrange from least to most expensive
candy %>%
  arrange(pricepercent) %>%
  head()

# You try: arrange from most to least expensive
candy %>%
  arrange(desc(pricepercent)) %>%
  head()
```

4. **filter() rows**
   We're not always interested in *all* rows of a dataset. For example, here we might be interested in studying only chocolate, not *all* candy. `filter()` allows us to keep only certain *rows* that meet a given criterion. To write these criteria, we must specify the *variable* by which we want to filter the data and the *value(s)* of that variable that we want to keep. Here are some general rules to try out below:

   - If variable `x` is *quantitative*:
     `x == 1`, `x < 1`, `x <= 1`, `x > 1`, `x >= 1`, `x != 1`
   - If variable `x` is *categorical / factor*:
     `x == "a"`, `x != "a"`
   - If variable `x` is *logical* (`TRUE` / `FALSE`):
     `x == TRUE`, `x == FALSE`

```
# Keep the sad candies that won less than 30%
candy %>%
```

```
  filter(___)

# Keep only Dots
candy %>%
  filter(___)

# Keep only chocolate-y candies
candy %>%
  filter(___)
```

Solution

```
# Keep the sad candies that won less than 30%
candy %>%
  filter(pricepercent < 30)

# Keep only Dots
candy %>%
  filter(competitorname == "Dots")

# Keep only chocolate-y candies
candy %>%
  filter(chocolate == TRUE)
```

5. **Combining `arrange()` and `filter()`**
We can perform multiple, sequential `arrange()` and `filter()` operations.

```
# Keep only chocolate-y, peanut butter-y candies. Do this in 3 lines.
candy %>%
  filter(___)  %>%
  filter(___)

# Keep only chocolate-y, peanut butter-y candies. Do this in 2 lines.
candy %>%
  filter(___, ___)

# Arrange fruity candy from most to least popular
candy %>%
  ___(___) %>%
  ___(___)

# Can you rearrange the operations above to arrange fruity candy from most to least popular?
candy %>%
  ___(___) %>%
  ___(___)
```

Solution

```
# Keep only chocolate-y, peanut butter-y candies. Do this in 3 lines.
candy %>%
  filter(chocolate == TRUE)  %>%
  filter(peanutyalmondy == TRUE)

# Keep only chocolate-y, peanut butter-y candies. Do this in 2 lines.
candy %>%
  filter(chocolate == TRUE, peanutyalmondy == TRUE)

# Arrange fruity candy from most to least popular
candy %>%
  filter(fruity == TRUE) %>%
  mutate(desc(winpercent))

# Can you rearrange the operations above to arrange fruity candy from most to leas
candy %>%
  mutate(desc(winpercent)) %>%
  filter(fruity == TRUE)
```

4.4 Transforming columns: `select()` and `mutate()`

6. `select()` columns
   There are often more variables (columns) in a dataset than we're interested in. Removing the superfluous variables can make data analysis more computationally efficient and less overwhelming.

```
# Create a dataset which only contains candy name & winpercent
candy %>%
  select(competitorname, winpercent) %>%
  head()

# You try: create a dataset which only contains candy name, chocolate status, & w

# You try: can you guess how to remove only the nougat variable?
```

Solution

```
# Create a dataset which only contains candy name & winpercent
candy %>%
  select(competitorname, winpercent) %>%
  head()
```

```
# You try: create a dataset which only contains candy name, chocolate status, & winpercent
candy %>%
  select(competitorname, winpercent, chocolate) %>%
  head()


# You try: can you guess how to remove only the nougat variable?
candy %>%
  select(-nougat) %>%
  head()
```

7. `mutate()` columns
We can create new variables (columns) by *mutating* existing columns. This is helpful when we want to change the scale of a variable, combine variables into new measurements, etc.

```
# Create price_percentile variable which transforms pricepercent to 0-100 scale
candy %>%
  mutate(price_percentile = ___) %>%
  head()

# You try: create a meaningless_sum variable which is the sum of sugarpercent and priceperce
```

Solution

```
# Create price_percentile variable which transforms pricepercent to 0-100 scale
candy %>%
  mutate(price_percentile = pricepercent * 100) %>%
  head()

# You try: create a meaningless_sum variable which is the sum of sugarpercent and priceperce
candy %>%
  mutate(meaningless_sum = sugarpercent + pricepercent) %>%
  head()
```

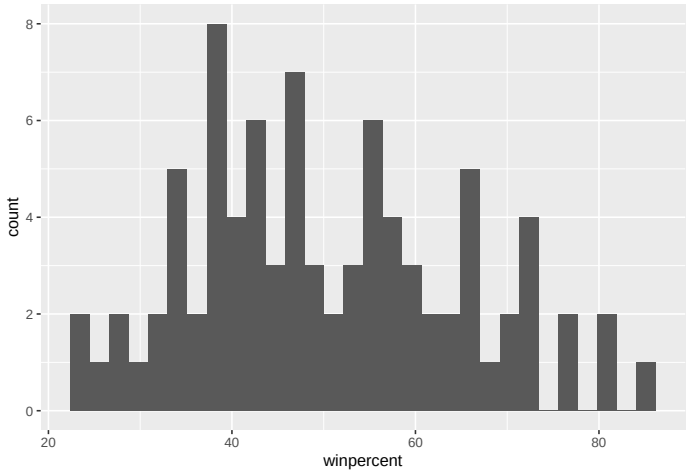
4.5 Simple numerical summaries: `summarize()`

8. **Review: univariate visualizations**
Construct a univariate visualization of `winpercent`. Summarize the trend and variability in the `winpercent` from candy to candy.

```
# try on your own first!
```

Solution

```
ggplot(candy, aes(x = winpercent)) +
  geom_histogram()
```



9. **Univariate numerical summaries of trend and variability**
The visualization above allows us to *eyeball* the trend and variability in `winpercent`. We can support these observations with rigorous numerical summaries.

```
# Trend: calculate the mean winpercent
candy %>%
  summarize(mean(winpercent))

# Trend: calculate the mean & median winpercent
candy %>%
  summarize(mean(winpercent), median(winpercent))

# Variability: calculate the min & max winpercent
candy %>%
  summarize(___, ___)
```

Solution

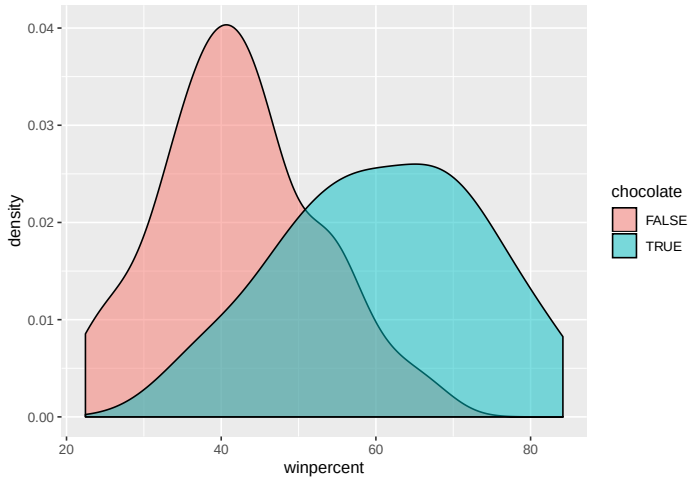
```
# Variability: calculate the min & max winpercent
candy %>%
  summarize(min(winpercent), max(winpercent))
```

10. **Review: multivariate visualizations**
Construct a visualization of the relationship between `winpercent` and `chocolate`. Summarize your observations.

```
# try on your own first!
```

Solution

```
ggplot(candy, aes(x = winpercent, fill = chocolate)) +
  geom_density(alpha = 0.5)
```



11. **Numerical summaries by group**
The visualization above illuminates how the trend and variability in `winpercent` differs depending upon `chocolate` status. We can calculate separate numerical summaries for these groups.

```
# Trend: calculate the mean winpercent by chocolate group
candy %>%
  group_by(chocolate) %>%
  summarize(mean(winpercent))

# (You try) Variability: calculate the min & max winpercent by fruit group
```

Solution

```
# (You try) Variability: calculate the min & max winpercent by fruit group
candy %>%
  group_by(fruity) %>%
  summarize(min(winpercent), max(winpercent))
```

4.6 Exercises

Practice, practice, practice is important when learning the `dplyr` verbs. Try out the following exercises with solutions below. Again, there are more exerices here than you will finish during the workshop.

4.6.1 Questions

1. **Revisiting movies**
   Recall the `bechdel` data:

   ```
   data("bechdel")
   ```

   To simplify our analysis, create a new dataset named `new_bechdel` which contains only the following variables: `title`, `year`, `binary`, `clean_test`, `budget_2013`, `domgross_2013`. HINT: Which verb is helpful here: `arrange()`, `filter()`, `select()`, `mutate()`, `summarize()`

   ```
   new_bechdel <- ___
   ```

2. **Transforming budget and gross**
   Film budgets and profits can be huge. To make these values easier to explore, create two new variables, `budget_mil` and `gross_mil`, which transform `budget_2013` and `domgross_2013` to the millions scale (instead of $ scale). Be sure to store these in the `new_bechdel` dataset. HINT: Which verb is helpful here: `arrange()`, `filter()`, `select()`, `mutate()`, `summarize()`

   ```
   new_bechdel <- ___
   ```

3. **Most vs least expensive. Most vs least profitable**
   a. Identify the 6 movies with the largest budgets and the 6 movies with the smallest budgets.

   b. Identify the 6 movies with the largest gross and the 6 movies with the smallest gross.

   c. Among films that *pass* the Bechdel test, which are the 6 with the largest gross? HINT: You'll need to combine 2 `dplyr` verbs!

4. **Which films?**
   Identify which films were...
   a. Made in 2013.

   b. Made in 2013 and didn't pass the "at least 2 women characters" criterion of the Bechdel test. HINT: This is indicated by the `nowomen` level of the `clean_test` variable.

   c. Made in 2013 and grossed more than $300,000,000.

5. **Min, Median, Max**
   a. Among all films in the dataset, calculate the minimum, median, and maximum budget.

   b. Among all films in the dataset, calculate and compare the minimum,

median, and maximum budgets for movies that pass and fail the Bechdel test.

c. Among films made in the 1970s, calculate and compare the minimum, median, and maximum budgets for movies that pass and fail the Bechdel test.

d. Among films made in the 2010s, calculate and compare the minimum, median, and maximum budgets for movies that pass and fail the Bechdel test.

e. Compare your results for parts c and d. What does this reveal about changes in the movie business?

6. **Challenge: Friday the 13th**
In this final exercise, you'll need to combine some of the `dplyr` verbs in a more open-ended scenario. Hints are provided below. Recall the `US_births_2000_2014` births data:

```
data("US_births_2000_2014")
```

We'll use these data to determine whether there's any evidence that people are superstitious about giving birth on Friday the 13th. Specifically:

a. Visualize the birth patterns among Fridays that fall on the 13th day of the month and Fridays that fall off the 13th day.

b. Calculate and compare the median number of births on Fridays that fall on the 13th vs Fridays that fall off the 13th.

c. Summarize your findings. Mainly, did you find ample evidence of superstition?

HINTS:

- We're only interested in births that occur on Fridays.
- There's currently no variable that lumps Friday birthdays into two categories: those that fall on the 13th day of the month and those that don't.

4.6.2 Solutions

1. .

```
data("bechdel")
new_bechdel <- bechdel %>%
  select(title, year, binary, clean_test, budget_2013, domgross_2013)
```

2. .

```
new_bechdel <- new_bechdel %>%
    mutate(budget_mil = budget_2013 / 1000000, gross_mil = domgross_2013 / 1000000
```

3. .

```
# a
new_bechdel %>%
    arrange(desc(budget_mil)) %>%
    head()
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 Avat~  2009 FAIL   men          461435929     825707158       461.
## 2 Pira~  2007 PASS   ok           337063045     347647302       337.
## 3 Shaft  1971 FAIL   notalk       305063707     404702718       305.
## 4 Tita~  1997 PASS   ok           290247625     955890356       290.
## 5 John~  2012 PASS   ok           279025606      74128153       279.
## 6 The ~  2012 FAIL   notalk       279025606     454699213       279.
## # ... with 1 more variable: gross_mil <dbl>
new_bechdel %>%
    arrange(budget_mil) %>%
    head()
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 Prim~  2004 FAIL   notalk            8632        523811    0.00863
## 2 El M~  1992 FAIL   nowomen          11622       3388636    0.0116
## 3 In t~  1997 FAIL   notalk           36281       4184879    0.0363
## 4 Funn~  2002 PASS   ok               38855         99819    0.0389
## 5 Slac~  1991 PASS   ok               39349       2100070    0.0393
## 6 Cler~  1994 FAIL   notalk           42435       4830398    0.0424
## # ... with 1 more variable: gross_mil <dbl>

# b
new_bechdel %>%
    arrange(desc(gross_mil)) %>%
    head()
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 Star~  1977 FAIL   notalk        42274609    1771682790       42.3
## 2 Jaws   1975 FAIL   notalk        51937204    1125306085       51.9
## 3 The ~  1973 PASS   ok            62926730    1074306128       62.9
## 4 E.T.~  1982 FAIL   dubious       25339314    1050038377       25.3
## 5 Tita~  1997 PASS   ok           290247625     955890356      290.
```

```
## 6 The ~  1973 FAIL   notalk        28841418     837011132       28.8
## # ... with 1 more variable: gross_mil <dbl>
new_bechdel %>%
    arrange(gross_mil) %>%
    head()
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 Perr~  2009 FAIL   nowomen        7165829           899     7.17
## 2 Pont~  2008 FAIL   nowomen        1623384          4183     1.62
## 3 Supp~  2012 FAIL   men              60878          4989     0.0609
## 4 Trop~  2007 PASS   ok             7345604          9824     7.35
## 5 St. ~  2007 PASS   ok            12808396         16853    12.8
## 6 The ~  2011 PASS   ok             3107072         18642     3.11
## # ... with 1 more variable: gross_mil <dbl>

# c
new_bechdel %>%
    filter(binary == "PASS") %>%
    arrange(desc(gross_mil)) %>%
    head()
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 The ~  1973 PASS   ok             62926730    1074306128       62.9
## 2 Tita~  1997 PASS   ok            290247625     955890356      290.
## 3 Star~  1999 PASS   ok            160823133     663632711      161.
## 4 Grea~  1978 PASS   ok             21424236     649203517       21.4
## 5 Jura~  1993 PASS   ok            101584911     638063379      102.
## 6 Shre~  2004 PASS   ok             86323501     544117065       86.3
## # ... with 1 more variable: gross_mil <dbl>
```

4. .

```
# a
new_bechdel %>%
    filter(year == 2013) %>%
    head()   # just for simplicity
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 21 &~  2013 FAIL   notalk         13000000      25682380         13
## 2 12 Y~  2013 FAIL   notalk         20000000      53107035         20
## 3 2 Gu~  2013 FAIL   notalk         61000000      75612460         61
## 4 42     2013 FAIL   men            40000000      95020213         40
## 5 47 R~  2013 FAIL   men           225000000      38362475        225
```

```
## 6 A Go~  2013 FAIL   notalk        92000000      67349198         92
## # ... with 1 more variable: gross_mil <dbl>

# b
new_bechdel %>%
    filter(year == 2013, clean_test == "nowomen")
## # A tibble: 6 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 Clou~  2013 FAIL   nowomen       78000000     119640264         78
## 2 Grav~  2013 FAIL   nowomen      110000000     271814796        110
## 3 Jack~  2013 FAIL   nowomen      195000000      65187603        195
## 4 Runn~  2013 FAIL   nowomen       30000000      19316646         30
## 5 The ~  2013 FAIL   nowomen       19200000            NA         19.2
## 6 Tran~  2013 FAIL   nowomen       16000000       2322593         16
## # ... with 1 more variable: gross_mil <dbl>

# c
new_bechdel %>%
    filter(year == 2013, gross_mil > 300)
## # A tibble: 4 x 8
##   title  year binary clean_test budget_2013 domgross_2013 budget_mil
##   <chr> <int> <chr>  <ord>            <int>         <dbl>      <dbl>
## 1 Desp~  2013 PASS   ok            76000000     368065385         76
## 2 Froz~  2013 PASS   ok           150000000     393050114        150
## 3 Iron~  2013 FAIL   dubious      200000000     408992272        200
## 4 The ~  2013 PASS   ok           130000000     424088260        130
## # ... with 1 more variable: gross_mil <dbl>
```

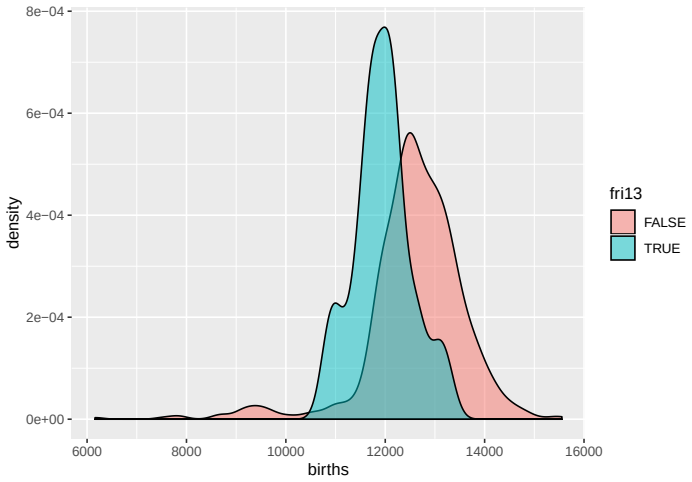
5. .

```
# a
new_bechdel %>%
    summarize(min(budget_mil), median(budget_mil), max(budget_mil))
## # A tibble: 1 x 3
##   `min(budget_mil)` `median(budget_mil)` `max(budget_mil)`
##               <dbl>                <dbl>             <dbl>
## 1           0.00863                 37.0              461.

# b
new_bechdel %>%
    group_by(binary) %>%
    summarize(min(budget_mil), median(budget_mil), max(budget_mil))
## # A tibble: 2 x 4
##   binary `min(budget_mil)` `median(budget_mil)` `max(budget_mil)`
##   <chr>              <dbl>                <dbl>             <dbl>
```

```
## 1 FAIL             0.00863                 44.0             461.
## 2 PASS             0.0389                  31.5             337.

# c
new_bechdel %>%
    filter(year < 1980) %>%
    group_by(binary) %>%
    summarize(min(budget_mil), median(budget_mil), max(budget_mil))
## # A tibble: 2 x 4
##   binary `min(budget_mil)` `median(budget_mil)` `max(budget_mil)`
##   <chr>              <dbl>                <dbl>             <dbl>
## 1 FAIL              0.384                 20.7              305.
## 2 PASS              0.0669                22.2               62.9

# d
new_bechdel %>%
    filter(year >= 2010) %>%
    group_by(binary) %>%
    summarize(min(budget_mil), median(budget_mil), max(budget_mil))
## # A tibble: 2 x 4
##   binary `min(budget_mil)` `median(budget_mil)` `max(budget_mil)`
##   <chr>              <dbl>                <dbl>             <dbl>
## 1 FAIL              0.0609                46.6              279.
## 2 PASS              0.0534                26.9              279.
```

6. .

```
data("US_births_2000_2014")

# Filter out all but Fridays and define Friday the 13th variable
new_births <- US_births_2000_2014 %>%
    filter(day_of_week == "Fri") %>%
    mutate(fri13 = (date_of_month == 13))

# a
ggplot(new_births, aes(x = births, fill = fri13)) +
    geom_density(alpha = 0.5)
```




```
# b
new_births %>%
    group_by(fri13) %>%
    summarize(median(births))
## # A tibble: 2 x 2
##   fri13 `median(births)`
##   <lgl>            <int>
## 1 FALSE            12656
## 2 TRUE             11994
```

4.7 Resources

Garrett Grolemund and Hadley Wickham wrote a free, online book that spans the tidyverse, including `ggplot2` and `dplyr`.

Chapter 5

Linear Regression

Author: Christina Knudson

5.1 Goals

In this module we'll focus on the **"Model"** and **"Communicate"** steps in the data science workflow. We'll also call on your understanding of data visualization and transformation.

source: Wickham & Grolemund: R for Data Science

This chapter will cover how to do the following:

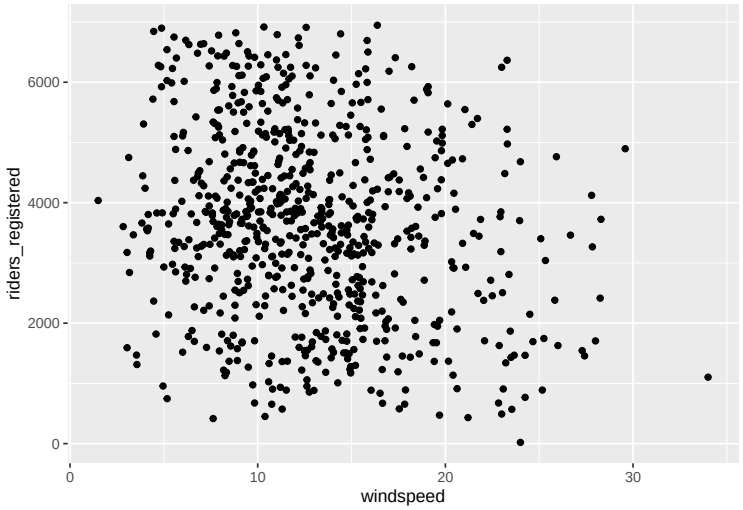
- Fit simple linear regression models
- Plot and interpret the regression results
- Make predictions from regression models
- Test whether two quantitative variables have a linear relationship

R's **lm** (linear model) function will be the primary tool used in the chapter.

5.2 Visualize the Data

Recall the bike-share data from the chapter on data visualization. Riding on a windy day can be challenging, but is it actually associated with fewer registered riders? Let's start by visualizing the data. Since the wind-speed is the predictor, we put it on the x-axis. Let's use the number of registered riders as the response variable.

```
library(ggplot2)
bikes <- read.csv("https://www.macalester.edu/~dshuman1/data/155/bike_share.csv")
ggplot(bikes, aes(y = riders_registered, x = windspeed)) +
    geom_point()
```



There may be a slight trend, but this plot doesn't provide a clear answer, but we can dig in with regression. Our goal is to fit a line to characterize this linear relationship between the number of registered riders and the windspeed.

5.3 Notation and Setup

Recall the equation for a line with intercept a and slope b is

$$y = a + bx. \tag{5.1}$$

The general equation for a simple linear regression model has the following form:

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

where x_i is the quantitative predictor, β_0 is the unknown regression intercept, β_1 is the unknown regression slope, and $\hat{y}_i$ is the predicted response given x_i.

We estimate β_0 and β_1 using data. This estimation allows us to characterize the linear relationship between x_i and y_i in the simple linear regression setting.

Let's write the regression equation for our bike-share example. The variable name "number of registered riders" is a bit long, so let's shorten it down to "nriders." The number of registered riders is our response variable, so we put

that on the left-hand side of the equality. The windspeed is our predictor, so we write our regression equation as follows:

$$\widehat{\text{nriders}}_i = \beta_0 + \beta_1 \text{windspeed}_i.$$

Our goal is to use the bike-share data set to estimate β_0 and β_1.

5.4 Fit a Simple Linear Regression Model

To fit a linear regression model, we use the **lm** function:

```
bikemod <- lm(riders_registered ~ windspeed,  data = bikes)
```

The first input is the regression formula (Response ~ Predictor) and the next argument is the name of the data set. The **lm** command estimates the regreesion coefficients and calculates other quantitaties. To find the regression coefficients (i.e. the estimates of β_0 and β_1), we can use the **coef** command. The only input for this command is the model.

```
coef(bikemod)
## (Intercept)   windspeed
##  4490.09761   -65.34145
```

We can now enter these estimates into our linear regression equation:

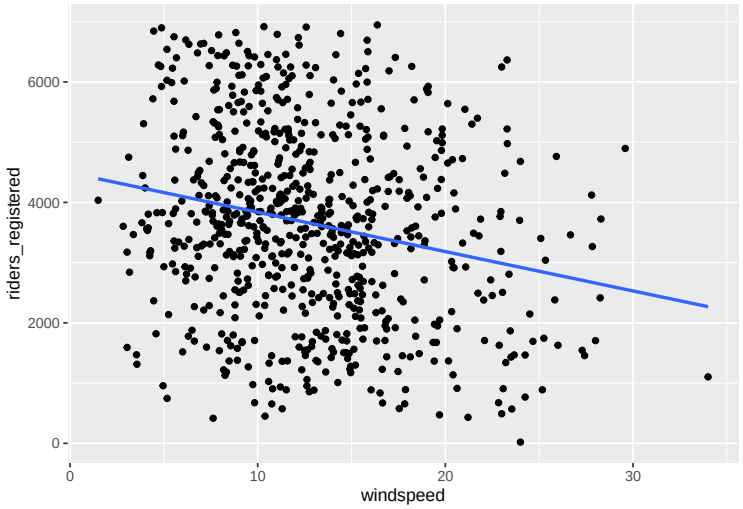
$$\hat{\text{nriders}}_i = 4490.10 + -65.34 \text{ windspeed}_i$$

.

5.5 Plot the Regression Line

It's easy to include the regression line on a scatterplot by adding a `geom_smooth()` to the `ggplot()`:

```
ggplot(bikes, aes(y = riders_registered, x = windspeed)) +
    geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



We can see the line is downward sloping. We'll discuss that in a moment.

5.6 Interpret the Model

First, let's focus on the sign of predictor's coefficient (-65.34). This is a **negative** number. This tells us that there are **fewer** registered riders on windier days. That is, the line is downward sloping. If the predictor's coefficient were **zero**, there would be **no** linear relationship between the windspeed and the number of registered riders, and the line would be horizontal. If the predictor's coefficient were **positive**, then windier days would have **more** registered riders, and the line would be upward sloping.

Next, let's look at the magnitude of the predictor's coefficient (-65.34). We can interpret this by thinking back to the interpretation of the slope of a line. Every one mile per hour increase in the wind speed is associated with about 65 fewer registered riders. One mile per hour is an imperceptible difference for most people, so let's think about a 5 mph increase in wind speed. A five mph increase in windspeed is associated with about 323 fewer registered riders.

Finally, let's look at the y-intercept. The y-intercept is the value of the line when the predictor is 0. We can interpret the y-intercept only if 0 is a plausible value for the predictor AND if we have values around 0 in our data set. The lowest windspeed in our dataset is 1.5 mph, so it's questionable whether we can safely interpret this. If we decide to interpret it, we'd say that on a day with absolutely no wind (that is, the wind is 0 mph), we expect about 4490 registered riders.

5.7 Calculate Point Predictions

5.7.1 The Type-It-In Method

Let's use our model to predict the number of riders on a day with 15 mph winds. All we need to do is plug 15mph into the variable windspeed in our regression equation

$\widehat{\text{nriders}}_i = 4490.10 + -65.34 \text{ windspeed}_i = 4490.10 + -65.34 * 15 = 3510.$

```
4490.10  + -65.34 * 15
## [1] 3510
```

Thus, we predict 3510 riders on a day with 15mph winds.

5.7.2 More Reproducible Methods

You can certainly add numbers in R's console, but a safer (more reproducible) method is to take advantage of some of R's functions, such as **predict**. Before you can use **predict**, you need to create a new data frame with the predictor value. We want to predict for a windspeed of 15 mph, so we specify that in our data frame.

```
new_windspeed <- data.frame(windspeed = 15 )
new_windspeed
##   windspeed
## 1        15
```

Now we can use the **predict** command. The first input is the model that we want to use for the prediction. We want to use the regression model in bikemod. The second input is the new data frame.

```
predict(bikemod, new_windspeed )
##        1
## 3509.976
```

5.8 Bonus: Test the Linear Relationship

This section is a bit more advanced, so don't stress over it. We will cover this in much more detail in the upcoming regression workshop.

Based on the data collected, we found that windier days have fewer riders. Did this result pop up by random chance in this data set, or do windier days truly have fewer riders in general? We can answer this question by conducting a test.

Several methods exist for testing the linear relationship between two variables. One of the easiest ways to conduct our test is to use the **summary** function. The only input for this function is our model.

```
summary(bikemod)
##
## Call:
## lm(formula = riders_registered ~ windspeed, data = bikes)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3575.7 -1126.8   -48.3  1089.2  3525.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4490.10     149.66  30.002  < 2e-16 ***
## windspeed     -65.34      10.86  -6.015 2.84e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1524 on 729 degrees of freedom
## Multiple R-squared:  0.04728,    Adjusted R-squared:  0.04598
## F-statistic: 36.18 on 1 and 729 DF,  p-value: 2.844e-09
```

The summary outputs a wealth of information, which can be kind of overwhelming for those without much experience with statistics. To test whether our two variables in a simple linear regression have a linear relationship, look at the number at the very bottom right (labeled p-value).

For our bike example, this number is 2.844e-09, or 2.844×10^{-9}. Because the p-value is quite small (and smaller than the "usual" threshold of .05), we conclude that there is indeed a linear relationship between the windspeed and the number of registered riders.

If the p-value were large (perhaps larger than a threshold such as .05), we would say that no linear relationship existed between the windspeed and the number of registered riders.

As a final note, the p-value in the last line matches a p-value in the second row of the coefficients table. In the simple linear regression setting (where you have a single predictor), these numbers match and you can pull the p-value from either place. If/when you move into multiple linear regression, these numbers will no longer match. You can learn more about this on your own or in our upcoming regression workshop.

5.9 Exercises

The first two practice problems uses the bike share data. The last two practice problems use the bechdel data.

```
library(fivethirtyeight)
data(bechdel)
```

5.9.1 Problem 1

Find the linear relationship between the perceived temperature and the number of registered riders. Then predict the number of riders on a day that feels like 80 degrees F.

5.9.2 Problem 2

Find the linear relationship between the humidity and the number of total riders. Then predict the total number of riders when humidity is at .5 (50 percent).

5.9.3 Problem 3

Use linear regression to predict worldwide (international) gross profits based on domestic gross profits. Use the 2013 adjusted sales. Given the domestic box office gross sales for **12 Years a Slave**, predict the movie's international gross sales. Here's the domestic gross for **12 Years a Slave** in 2013 dollars:

```
bechdel[3,"domgross_2013"]
## # A tibble: 1 x 1
##   domgross_2013
##           <dbl>
## 1      53107035
```

5.9.4 Problem 4

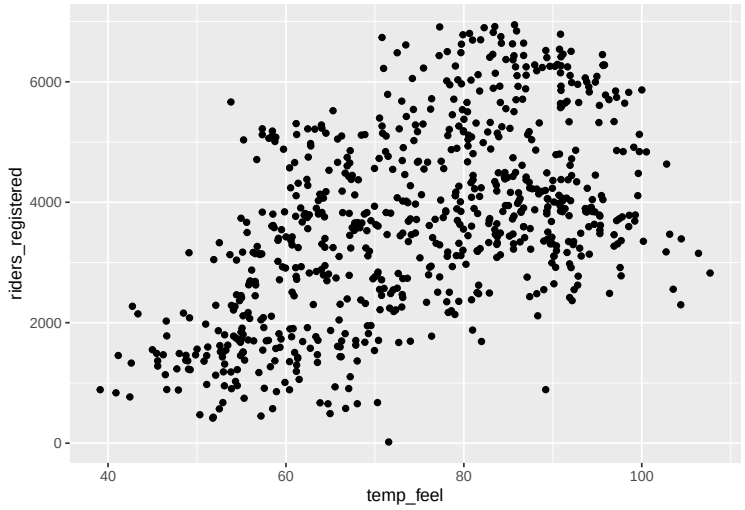
The `intgross` and `intgross_2013` variables represent the worldwide gross profits of films. To find the non-US (ie foreign) gross profits, you can subtract the domestic gross profits from the international gross profits. Do this (that is, use the **mutate** function) and use linear regression to predict the non-US (foreign) gross profits for **12 Years a Slave** based on the film's domestic gross profits.

5.10 Solutions

5.10.1 Problem 1

Start with data visualization.

```
ggplot(bikes, aes(y = riders_registered, x = temp_feel)) +
    geom_point()
```



Next, create the regression model.

```
biketemp <- lm(riders_registered ~ temp_feel, data = bikes)
coef(biketemp)
## (Intercept)   temp_feel
##  -667.91568    57.89236
```

Then, write the regression equation.

$$\widehat{\text{riders}}_i = -667.91568 + 57.89236\,\text{temp}_i$$

For every degree increase in temperature (in F), we expect about 58 additional registered riders. Now, let's predict the number of registered riders for a day that feels like 80 degrees.

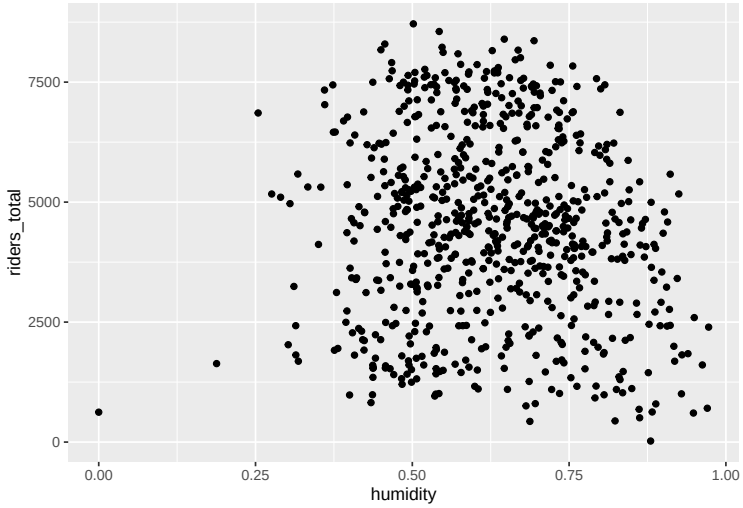
```
eighty <- data.frame(temp_feel = 80)
predict(biketemp, eighty )
##        1
## 3963.473
```

We expect about 3963 riders.

5.10.2 Problem 2

Start with data visualization.

```
ggplot(bikes, aes(y = riders_total, x = humidity)) +
    geom_point()
```



There may be a downward trend, but there's a lot of variability.

Next, create the regression model.

```
bikehumid <- lm(riders_total ~ humidity, data = bikes)
coef(bikehumid)
## (Intercept)    humidity
##    5363.986   -1369.081
```

Then, write the regression equation.

$$\widehat{\text{riders}}_i = 5363.986 + -1369.081\,\text{humidity}_i$$

We need to be a little careful with the interpretation here, because humidity values are between 0 and 1. We can't increase humidity by an entire unit! Let's increase humidity by .1 (meaning 10 percentage points. A 10 percentage point increase in humidity is associated with 137 fewer total riders.

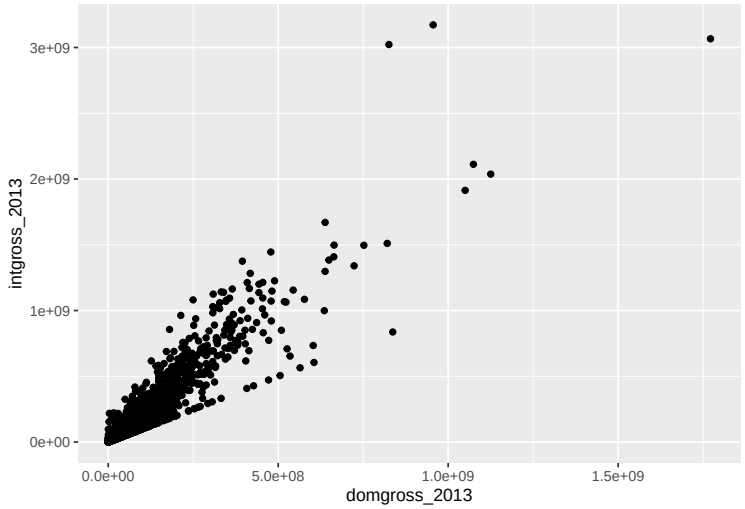
```
newdf <- data.frame(humidity = .50)
predict(bikehumid, newdf )
##        1
## 4679.446
```

We expect about 3963 riders. To be safe, check this number on your plot to make sure that you're getting something reasonable.

5.10.3 Problem 3

Begin by visualizing the data:

```
ggplot(bechdel, aes(y = intgross_2013, x = domgross_2013)) +
    geom_point()
```



Next, create the regression model.

```
moviemod <- lm(intgross_2013 ~ domgross_2013, data = bechdel)
coef(moviemod)
##   (Intercept) domgross_2013
## -364528.19765       2.09019
```

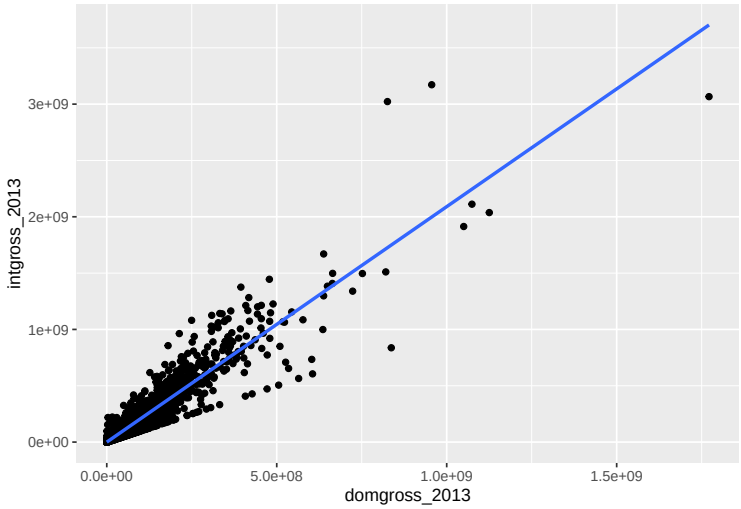
Then, write the regression equation.

$$\widehat{\text{int gross}}_i = -364528.19765 + 2.09019\,\text{domestic}_i$$

Unsurprisingly, there is a positive association between domestic and worldwide gross. A one dollar increase in the domestic gross is associated with a two dollar (and nine cent) increase in the worldwide gross, in 2013 US dollars.

For fun, let's add the line to our plot.

```
ggplot(bechdel, aes(y = intgross_2013, x = domgross_2013)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE)
```



Now, let's predict for **12 Years a Slave**. The domestic gross for this movie was 53107035. Plugging that value into the regression equation produces the predicted worldwide gross profits in US dollars:

```
twelve <- bechdel[3, "domgross_2013"]
predict(moviemod, twelve )
##         1
## 110639278
```

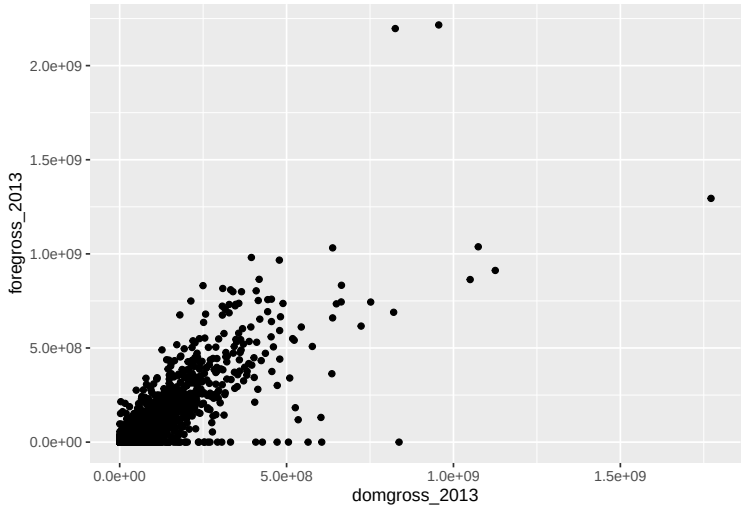
5.10.4 Problem 4

Begin by creating the new variable.

```
library(dplyr)
bechdel <- bechdel %>%
  mutate(foregross_2013 = intgross_2013 - domgross_2013)
```

Second, visualize the data.

```
ggplot(bechdel, aes(y = foregross_2013, x = domgross_2013)) +
    geom_point()
```



Third, create the regression model.

```
foremod <- lm (foregross_2013 ~ domgross_2013, data = bechdel)
coef(foremod)
##   (Intercept) domgross_2013
## -364528.19765       1.09019
```

Then, write the regression equation.

$$\widehat{\text{foreign}}_i = -364528.19765 + 1.09019\,\text{domestic}_i$$

Let's interpret this regression equation for fun. First, can we interpret the intercept? No, because none of the films in our data-set have a domestic gross profit of 0 USD, as you can see in the summary below.

```
bechdel %>% select(domgross_2013) %>% summary()
##  domgross_2013
##  Min.   :8.990e+02
##  1st Qu.:2.055e+07
##  Median :5.599e+07
##  Mean   :9.517e+07
##  3rd Qu.:1.217e+08
##  Max.   :1.772e+09
##  NA's   :18
```

Fine, we can't interpret the intercept. Let's interpret the slope then. First of all, the slope is positive, which makes sense: films that make money in the US probably tend to make money in non-US markets, too. (If the slope were negative, this would mean that films that are more lucrative in the US are less lucrative in foreign markets.) Now let's check out the magnitude of the slope:

1.09. For every additional dollar of profit in the US, we expect the film to make an additional 1.09 USD profit in foreign markets.

Next, predict the foreign gross profit in 2013 dollars for **12 Years a Slave**:

```
predict(foremod, twelve)
##        1
## 57532243
```

Chapter 6

Logistic Regression

Author: Christina Knudson

6.1 Introduction

The previous chapter covered linear regression, which models a quantitative response variable. This chapter covers logistic regression, which is used to model a binary response variable. Here are some examples of binary responses:

- Whether or not a person wears a helmet while biking
- Whether or not a dog is adopted
- Whether or not a beer is given an award
- Whether or not a tree survives a storm

6.2 Goals

In this module we'll focus on the **"Model"** and **"Communicate"** steps in the data science workflow.

source: Wickham & Grolemund: R for Data Science

This chapter will cover how to do the following:

- Fit a logistic regression model in R with quantitative and/or categorical predictors.
- Interpret logistic regression models.
- Calculate probabilities.
- Test the significance of regression coefficients.

R's **glm** (generalized linear model) function will be the primary tool used in the chapter.

6.2.1 Horseshoe Crab Data

Throughout this module, we will refer to the horseshoe crab data. Some female crabs attract many males while others do not attract any. The males that cluster around a female are called "satellites." In order to understand what influences the presence of satellite crabs, researchers selected female crabs and collected data on the following characteristics:

- the color of her shell
- the condition of her spine
- the width of her carapace shell (in centimeters)
- the number of male satellites
- the weight of the female (in grams)

In today's example, we will use the width of a female's shell to predict the probability of her having one or more satellites. Let's start by loading the data.

```
crabs <- read.csv("http://www.cknudson.com/data/crabs.csv")
head(crabs)
##    color spine width satell weight y
## 1 medium   bad  28.3      8   3050 1
## 2   dark   bad  22.5      0   1550 0
## 3  light  good  26.0      9   2300 1
## 4   dark   bad  24.8      0   2100 0
## 5   dark   bad  26.0      4   2600 1
## 6 medium   bad  23.8      0   2100 0
```

You can learn more about this data set here.

6.3 Model Basics

6.3.1 Notation and Setup

Recall that a simple linear regression model has the following form:

$$\hat{y}_i = \beta_0 + \beta_1 x_i$$

where x_i is the predictor, β_0 is the unknown regression intercept, β_1 is the unknown regression slope, and $\hat{y}_i$ is the predicted response given x_i.

In order to model a binary response variable, we need to introduce p_i, the probability of something happening. For example, this might be the probability of a person wearing a helmet, the probability of a dog being adopted, or the

probability of a beer winning an award. Then our logistic regression model has the following form:

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_i.$$

Recall that we estimated β_0 and β_1 to characterize the linear relationship between x_i and y_i in the simple linear regression setting. In the logistic regression setting, we will estimate β_0 and β_1 in order to understand the relationship between x_i and p_i.

As a final introductory note, we define the odds as $\frac{p_i}{1-p_i}$.

6.3.2 Fit a Logistic Regression Model

To fit a logistic regression model, we can use the **glm** function:

```
mod <- glm(y ~ width, family = binomial, data = crabs)
```

The first input is the regression formula (Response ~ Predictor), the second input indicates we have a binary response, and the third input is the data frame. To find the regression coefficients (i.e. the estimates of β_0 and β_1), we can use the **coef** command

```
coef(mod)
## (Intercept)       width
## -12.3508177   0.4972306
```

We can now enter these estimates into our logistic regression equation, just as we did in the simple linear regression setting. Our logistic regression equation is

$$\log\left(\frac{p_i}{1-p_i}\right) = -12.3508 + 0.4972 \text{ width}_i,$$

where width_i is the width of a female crab's carapace shell and p_i is her probability of having one or more satellites.

6.3.3 Interpret the Model

To do some basic interpretation, let's focus on the predictor's coefficient: 0.4972. This is a **positive** number. This tells us that wider crabs have **higher** chances of having one or more satellites. If the predictor's coefficient were **zero**, there would be **no** linear relationship between the width of a female's shell and her log odds of having one or more satellites. If the predictor's coefficient were **negative**, then wider crabs would have **lower** chances of having one or more satellites.

6.3.4 Calculate Probabilities

Let's use our model for a female crab with a carapace shell that is 25 centimeters in width. We start by simply substituting this crab's width into our regression equation:

$$\begin{aligned}\log\left(\frac{p_i}{1-p_i}\right) &= -12.3508177 + 0.4972306\ \text{width}_i \\ &= -12.3508177 + 0.4972306 * 25 \\ &= 0.0799473.\end{aligned}$$

Now that you understand the idea of prediction, let's use our **predict** command, which we learned about in the regression chapter. We can use it for logistic regression. Again, create a data frame with a value for the predictor, and then use the **predict** command. The first input is the model and the second is the new crab's data frame.

```
newcrab <- data.frame(width =25)
predict(mod, newcrab)
##          1
## 0.07994696
```

We could say that the log odds of a 25 cm female having satellites is about 0.07995, but let's make this more interpretable to everyday humans by translating this to a probability. We use a little algebra to solve for p_i:

$$\begin{aligned}\log\left(\frac{p_i}{1-p_i}\right) &= 0.0799473 \\ \Rightarrow \left(\frac{p_i}{1-p_i}\right) &= \exp(0.0799473) \\ \Rightarrow p_i &= \frac{\exp(0.0799473)}{1+\exp(0.0799473)} \\ &= 0.5199762\end{aligned}$$

Here is our interpretation: the probability of a 25 cm wide female crab having one of more satellites is about 0.51998.

6.3.5 Test a Regression Coefficient

Recall that an essential question in regression is "Does this predictor actually have a linear relationship the response?" In the context of the crabs, we want to know whether the carapace width really does have a linear relationship with the log odds of satellites.

Recall the **summary** command, which was introduced in the linear regression setting. We can use it in the logistic regression setting, too. Below is the summary for our crab model.

```
summary(mod)
##
## Call:
## glm(formula = y ~ width, family = binomial, data = crabs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0281  -1.0458   0.5480   0.9066   1.6942
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -12.3508     2.6287  -4.698 2.62e-06 ***
## width         0.4972     0.1017   4.887 1.02e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.76  on 172  degrees of freedom
## Residual deviance: 194.45  on 171  degrees of freedom
## AIC: 198.45
##
## Number of Fisher Scoring iterations: 4
```

Although the summary provides many details about our model, we focus for now on the p-value in the second row of the coefficients table of the summary output.

Our p-value is 1.02e-06 (or 1.02×10^{-6}). We compare our p-value to a threshold (such as 0.05); **because our p-value is smaller than our threshold of** .05**, we have evidence that the log odds of satellites have a significant linear relationship with the carapace width.**

If the p-value were large, we would not have evidence to say that there is a linear relationship between a female's carapace width and her log odds of having satellites.

6.4 Half-Time Exercises

6.4.1 Female Horseshoe Crab Weight

Continue using the horseshoe crab data to investigate the relationship between a female's weight and the log odds of her having satellites.

- Create a logistic regression using the female's weight as the predictor and whether she has satellites as the response variable.
- Write down the regression equation.
- Do heavier females having higher or lower chances of having satellites?
- Consider a female weighing 2000 grams. What is the probability that she has one or more satellites?
- Is there a linear relationship between a female crab's weight and her log odds of satellites? Find the p-value and compare it to a threshold of 0.05.

6.4.2 Boundary Water Blowdown

The Boundary Water Canoe Area experienced wind speeds over 90 miles per hour on July 4, 1999. As a result, many trees were blown down. The data set below contains information on the diameter of each tree (**D**) and whether the tree died (**y**). **y** is 1 if the tree died and 0 if the tree survived. You can learn more about this data set here.

```
blowdown <- read.csv("http://www.cknudson.com/data/blowdown.csv")
```

Use this data set to understand the relationship between a tree's diameter and its log odds of death.

- Create a logistic regression using the tree's diameter the predictor and whether it died as the response variable.
- Write down the regression equation.
- Did thicker trees having higher or lower chances of dying?
- Consider a tree 20 cm in diameter. What is the probability that it died?
- Is there a linear relationship between the tree's diameter and its log odds of dying? Find the p-value and compare it to the threshold of 0.05.

6.4.3 Beer

The Minnesota beer data has 44 beers measured on 7 variables:

1) *Brewery*: Name of the brewery (**factor** with 8 levels)
2) *Beer*: Name of the beer (**factor** with 44 levels)
3) *Description*: Description of the beer (**factor** with 37 levels)
4) *Style*: Style of the beer (**factor** with 3 levels)
5) *ABV*: Alcohol by volume (**numeric**)
6) *IBU*: International bitterness units (**integer**)
7) *Rating*: Beer Advocate rating (**integer**)
8) *Good*: 1 if the beer has a score of at least 90 and 0 otherwise

Data obtained from Beer Advocate and the websites of the eight breweries.

- Use logistic regression to decide whether beers with higher **ABV** (alcohol by volume) are more likely to have a rating of at least 90. (Hint: Use the variable *Good* as your response.)
- Choose your favorite beer off the list and calculate its probability of having a score of at least 90.

```
beer <- read.csv("http://www.cknudson.com/data/MNbeer.csv")
```

6.4.4 State Colors

Recall the election data set introduced by Alicia Johnson. The variable **Red** has been added to the data set to indicate whether the state's color is red (1 if red, 0 otherwise).

- Is per capita income related to whether a state is red?
- If a state has a high per capita income, is it more or less likely to be red?

```
election <- read.csv("https://www.macalester.edu/~ajohns24/data/IMAdata1.csv")
election$Red <- as.numeric(election$StateColor == "red")
```

6.5 Half-Time Solutions

6.5.1 Crab Weight

We create the model and look at the summary to find the p-value for the coefficient on weight The p-value is small (1.45×10^{-6}) so we can conclude that a female crab's weight **does** have a significant linear relationship with the log odds of her having one or more satellites.

```
weightmod <- glm(y ~ weight, family = binomial, data = crabs)
summary(weightmod)
##
## Call:
## glm(formula = y ~ weight, family = binomial, data = crabs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1108  -1.0749   0.5426   0.9122   1.6285
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.6947264  0.8801975  -4.198 2.70e-05 ***
## weight       0.0018151  0.0003767   4.819 1.45e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 225.76  on 172  degrees of freedom
## Residual deviance: 195.74  on 171  degrees of freedom
## AIC: 199.74
##
## Number of Fisher Scoring iterations: 4
```

Now we can use our regression equation to calculate the probability of a 2000 gram female having one or more satellites.

```
newcrab2000 <- data.frame(weight = 2000)
logodds <- predict(weightmod, newcrab2000)
exp(logodds)/(1+exp(logodds))
##         1
## 0.4838963
```

The probability of a 2000 gram female having satellites is 0.4838963.

6.5.2 Boundary Water Blowdown

We create the model, isolate the coefficients, and find the summary using the code below.

```
treemod <- glm(y ~ D, data = blowdown, family = binomial)
coef(treemod)
## (Intercept)           D
## -1.70211206  0.09755846
summary(treemod)
##
## Call:
## glm(formula = y ~ D, family = binomial, data = blowdown)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.6309  -0.9616  -0.7211   1.1495   1.7172
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.702112   0.082798  -20.56   <2e-16 ***
## D            0.097558   0.004846   20.13   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 5057.9  on 3665  degrees of freedom
## Residual deviance: 4555.6  on 3664  degrees of freedom
## AIC: 4559.6
##
## Number of Fisher Scoring iterations: 4
```

First, the coefficient on the diameter is positive. This tells us that larger trees were more likely to die. Moreover, the relationship between a trees diameter and its log odds of death is statistically significant: the p-value is quite small (smaller than 2×10^{-16}).

Finally, the probability of death for a tree that was 20 cm in diameter is calculated below.

```
newtree <- data.frame(D = 20)
(logodds <- predict(treemod, newtree))
##         1
## 0.2490571
exp(logodds)/(1+exp(logodds))
##         1
## 0.5619444
```

6.5.3 Beer

We create the model and look at the summary to find the p-value for the coefficient on ABV. The p-value is small (0.00747) so we can conclude that ABV **does** have a significant linear relationship with the log odds of a beer earning a score of at least 90.

```
beermod <- glm(Good ~ ABV, family = binomial, data = beer)
summary(beermod)
##
## Call:
## glm(formula = Good ~ ABV, family = binomial, data = beer)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3847  -0.8206  -0.4663   0.7230   2.1320
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.7881     3.3959  -2.882  0.00395 **
## ABV           1.4662     0.5481   2.675  0.00747 **
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 51.564  on 43  degrees of freedom
## Residual deviance: 42.108  on 42  degrees of freedom
## AIC: 46.108
##
## Number of Fisher Scoring iterations: 5
```

6.5.4 State Colors

We create the model and look at the summary to find the p-value for the coefficient on per capita income. The p-value is very small (smaller than 2.2×10^{-16}) so we can conclude that per capita income **does** have a significant linear relationship with the log odds of a state being red.

```
electionmod <- glm(Red ~  per_capita_income, family = binomial, data = election)
summary(electionmod)
##
## Call:
## glm(formula = Red ~ per_capita_income, family = binomial, data = election)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.617  -1.146  -0.753   1.157   2.073
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)        1.649e+00  1.722e-01   9.575   <2e-16 ***
## per_capita_income -7.339e-05  7.214e-06 -10.173   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4352.6  on 3142  degrees of freedom
## Residual deviance: 4237.1  on 3141  degrees of freedom
## AIC: 4241.1
##
## Number of Fisher Scoring iterations: 4
```

Also, because the coefficient on per capita income is negative, we know that states with **higher** per capita incomes are **less** likely to be red.

6.6 Beyond the Basics

6.6.1 Interpret the Model (Again!)

We can look beyond just whether the predictor's coefficient (β_1) is positive or negative. Exponentiating both sides of the regression equation produces

$$\frac{p_i}{1-p_i} = \exp(\beta_0 + \beta_1 x_i) = \exp(\beta_0)\exp(\beta_1 x_i).$$

To understand the relationship between our predictor and the log odds, let's see what happens when we increase x_i by one unit. That is, let's replace x_i with $x_i + 1$.

$$\frac{p_i}{1-p_i} = \exp(\beta_0)\exp(\beta_1(x_i+1)) = \exp(\beta_0)\exp(\beta_1 x_i)\exp(\beta_1).$$

That is, our odds for predictor value x_i are $\exp(\beta_0)\exp(\beta_1 x_i)$ while the odds for predictor value $x_i + 1$ are $\exp(\beta_0)\exp(\beta_1 x_i)\exp(\beta_1)$. These two odds differ by a factor of $\exp(\beta_1)$. That is, the ratio of the two odds is $\exp(\beta_1)$:

$$\frac{\exp(\beta_0)\exp(\beta_1 x_i)\exp(\beta_1)}{\exp(\beta_0)\exp(\beta_1 x_i)} = \exp(\beta_1).$$

Therefore, we can say that a one unit increase in the predictor is associated with a $\exp(\beta_1)$ multiplicative change in the odds. Let's try this with the horseshoe crab carapace width model.

```
exp(coef(mod))
##  (Intercept)        width
## 4.326214e-06 1.644162e+00
```

A one centimeter increase in carapace width is associated with a 1.64 multiplicative change in the odds of having satellites. Alternatively, imagine two female crabs that have carapace widths that differ by exactly 1 cm. The odds of the larger crab having satellites is approximately 1.64 times the odds of the smaller crab having satellites.

6.6.2 Incorporate a Categorical Predictor

Rather than using a quantitative predictor, we can use a categorical predictor. Let's try using the spine condition in our horseshoe crab data. Crabs were categorized according to whether they had two good spines (*good*), two worn or broken spines (*bad*), or one worn/broken spine and one good spine (*middle*).

We will fit the model using the following **glm** setup.

```
spinemod <- glm(y ~ 0 + spine, data = crabs, family = binomial)
coef(spinemod)
##    spinebad   spinegood spinemiddle
##   0.5955087   0.8602013  -0.1335314
```

This produces three log odds, one for each group (bad, good, and middle). The coefficients represent the log odds of satellites for each group. This tells us that

- the log odds of satellites for a female with two bad spines is 0.596
- the log odds of satellites for a female with two good spines is 0.86
- the log odds of satellites for a female with one good spine is -0.134

Again, most people understand probabilities better than odds or log odds, so let's transform to probabilities:

```
a <- exp(coef(spinemod))
probs <- a/(1+a)
probs
##    spinebad   spinegood spinemiddle
##   0.6446281   0.7027027   0.4666667
```

This tells us that

- the probability of satellites for a female with two bad spines is 0.645
- the probability of satellites for a female with two good spines is 0.703
- the probability of satellites for a female with one good spine is 0.467

If we eliminate the **0+** in the **glm** formula, we create a model with a reference group; this will compare the bad spine group to the other two groups.

```
spinemod2 <- glm(y ~ spine, data = crabs, family = binomial)
coef(spinemod2)
## (Intercept)   spinegood spinemiddle
##   0.5955087   0.2646926  -0.7290401
exp(coef(spinemod2))
## (Intercept)   spinegood spinemiddle
##   1.8139535   1.3030303   0.4823718
```

This model tells us that

- the odds of satellites for a female with two bad spines is 1.8.
- the odds of satellites for a female with two good spines is 1.3 times the odds of satellites for a female with two bad spines.
- the odds of satellites for a female with one good spine and one bad spine is .48 times the odds of satellites for a female with two bad spines.

Sometimes people have a hard time interpreting odds ratios below 1. In this case, it is useful to flip the ratio.

```
1/exp(coef(spinemod2))
## (Intercept)   spinegood spinemiddle
##   0.5512821   0.7674419   2.0730897
```

Therefore, we can reword the third bullet to the following: the odds of satellites for a female with two bad spines is 2 times the odds of satellites for a female with one bad spine and one good spine.

6.7 More Problems

6.7.1 Interpret the Slopes

Revisit the models you created in the half-time problems and interpret the slopes.

6.7.2 State Colors Again

Return to the election data. Find the probability of a state being red based on its income bracket.

6.7.3 Beer Again

Return to the beer data. For each style of beer, calculate the probability of receiving a rating of 90 or higher.

6.8 More Solutions

6.8.1 Interpret the Slopes

6.8.1.1 Crab Weight

```
logodds <- coef(weightmod)[2]
exp(logodds)
##   weight
## 1.001817
```

A one gram increase in a crab's weight is associated with a multiplicative change of 1.0018168 in the odds of her having satellites.

A 100 gram increase in a crab's weight is associated with a multiplicative change of 1.1990319 in her odds of having satellites.

6.8.1.2 Boundary Water Blowdown

```
exp(coef(treemod)[2])
##        D
## 1.102476
```

A one centimeter increase in a tree's diameter was associated with a 1.1024759 multiplicative change in the odds of it blowing down.

A ten centimeter increase in a tree's diameter was associated with a 2.6527174 multiplicative change in the odds of it blowing down.

```
exp(coef(treemod)[2]*10)
##        D
## 2.652717
```

6.8.1.3 Beer

```
exp(coef(beermod)[2])
##      ABV
## 4.332636
```

A one percentage point increase in a beer's ABV is associated with a 1.1024759 multiplicative change in its log odds of having a rating of 90 or higher.

6.8.1.4 State Colors

We could look at a one dollar increase in per capita income, but that is not very useful because a dollar is not very meaningful. Let's instead look at a one thousand dollar increase in per capita income.

```
exp((coef(electionmod)[2]*1000))
## per_capita_income
##          0.929239
```

A one thousand dollar increase in a state's average per capita income is associated with a multiplicative change of 0.929239 in its odds of being red.

6.8.2 State Colors Again

```
lowhighmod <- glm(Red ~ IncomeBracket, data = election, family = binomial)
exp(coef(lowhighmod))
##      (Intercept) IncomeBracketlow
##        0.6197836        1.8217084
```

The odds of a high income state being red is about .62. The odds of a low income state being red are about 1.82 times the odds of a high income state being red. That is, low income states are more likely to be red than high income states.

6.8.3 Beer Again

```
beermod2 <- glm(Good ~ 0+ Style, data=beer, family = binomial)
logodds <- coef(beermod2)
probs <- round(exp(logodds)/(1+exp(logodds)),3)
probs
##   StyleAle   StyleIPA StyleLager
##      0.278      0.412      0.000
```

Ales have a 0.278 probability of having a rating of 90 or higher.

IPAs have a 0.412 probability of having a rating of 90 or higher.

The poor lagers have no chance of having a rating of 90 or higher. This is just based on the data in our data set. Surely if we looked at ALL lagers all over the world, we'd eventually find one that deserves a rating of 90 or higher.