Research Methods Coding Tutorial

Ben Silver

9/9/2024

Welcome!

In this tutorial, we'll practice loading data from a csv file, inspecting it, running descriptive statistics, inferential statistics, and plotting. If you find the tutorial difficult, don't worry! Practice makes perfect, and you won't be graded on this. When you eventually analyze your data from your group project, you'll likely do very similar things to what we're doing today.

Annotation: This coding tutorial has several purposes: First, it is designed to introduce students to basic coding concepts in a low-pressure environment. Second, it is sort of a "least-common denominator" of coding for Psychology research. I've tried to reduce all the many and varied steps - data loading, data cleaning, data exploration, statistical analysis, and visualization - into their most essential elements. Third, this tutorial should serve as a resource and/or example for students completing their own coding projects in the class. When coding, it's common to copy and paste from other resources that are doing similar things - that's actually one of the best ways to learn and feel more confident as a coder. In creating this tutorial, I tried to highlight functions and steps that students might be most likely to need for their own project.

This document is an R Markdown file, which is a special kind of file often used for teaching purposes. It allows us to combine regular text (such as this) with chunks of code, like you can find in the gray box below. Click the green "Play" button on the right side to run the code.

```
print("Hi there")
```

```
## [1] "Hi there"
```

The first thing we'll want to do is load the `tidyverse` package.

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ dplyr     1.1.4     ✔ readr     2.1.5
## ✔ forcats   1.0.0     ✔ stringr   1.5.1
## ✔ ggplot2   3.5.1     ✔ tibble    3.2.1
## ✔ lubridate 1.9.3     ✔ tidyr     1.3.1
## ✔ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

The dataset that we're going to work with today is from the website fivethirtyeight, and contains data on the Bechdel test. It was created by a cartoonist named Allison Bechdel. From fivethirtyeight: "Bechdel said that if a movie can satisfy three criteria — there are at least two named women in the picture, they have a conversation with each other at some point, and that conversation isn't about a male character — then it passes "The Rule," whereby female characters are allocated a bare minimum of depth."

Our goal is to see if there is a difference in both domestic box office performance (DV1) and critical performance (DV2) between movies that pass the Bechdel test and those that don't (IV).

Annotation: My choice to use this dataset is intentional, and aligns with my teaching philosophy. I try to make classroom content as relevant to the outside world as possible. Rather than use an obscure or esoteric psychology research dataset, I use something more accessible and relevant - a database about representation of women in movies - to prove to students that learning to code can have widespread benefits and uses beyond their college assignments.

Data exploration

Let's load in the data.

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

```
## Rows: 1794 Columns: 34
## ── Column specification ────────────────────────────────────────────────────
## Delimiter: ","
## chr (24): imdb, title, test, clean_test, binary, domgross, intgross, code, d...
## dbl  (7): year, budget, budget_2013, period_code, decade_code, metascore, im...
## num  (1): imdb_votes
## lgl  (2): response, error
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

There are a few ways we can explore our data. The first is by clicking on the white graph-thing on the right side of the environment window. This will show you your data in a separate tab.

There are also several R functions that give you some information about your data. One of these is `str()`. `str()` tells you the following things about an object:

- the type of object that IntroSurvey is
- the number of observations and number of variables (dimensions) of IntroSurvey
- a list of each variable and its class (interval, factor, numeric, etc.)
- for each variable, a list of all values

```
str(movies)
```

```
## spc_tbl_ [1,794 × 34] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ year         : num [1:1794] 2013 2012 2013 2013 2013 ...
##  $ imdb         : chr [1:1794] "tt1711425" "tt1343727" "tt2024544" "tt1272878" ...
##  $ title        : chr [1:1794] "21 &amp;amp; Over" "Dredd 3D" "12 Years a Slave" "2 Guns"
...
##  $ test         : chr [1:1794] "notalk" "ok-disagree" "notalk-disagree" "notalk" ...
##  $ clean_test   : chr [1:1794] "notalk" "ok" "notalk" "notalk" ...
##  $ binary       : chr [1:1794] "FAIL" "PASS" "FAIL" "FAIL" ...
##  $ budget       : num [1:1794] 1.30e+07 4.50e+07 2.00e+07 6.10e+07 4.00e+07 2.25e+08
9.20e+07 1.20e+07 1.30e+07 1.30e+08 ...
##  $ domgross     : chr [1:1794] "25682380" "13414714" "53107035" "75612460" ...
##  $ intgross     : chr [1:1794] "42195766" "40868994" "158607035" "132493015" ...
##  $ code         : chr [1:1794] "2013FAIL" "2012PASS" "2013FAIL" "2013FAIL" ...
##  $ budget_2013  : num [1:1794] 13000000 45658735 20000000 61000000 40000000 ...
##  $ domgross_2013: chr [1:1794] "25682380" "13611086" "53107035" "75612460" ...
##  $ intgross_2013: chr [1:1794] "42195766" "41467257" "158607035" "132493015" ...
##  $ period_code  : num [1:1794] 1 1 1 1 1 1 1 1 1 ...
##  $ decade_code  : num [1:1794] 1 1 1 1 1 1 1 1 1 ...
##  $ imdb_id      : chr [1:1794] "1711425" "1343727" "2024544" "1272878" ...
##  $ plot         : chr [1:1794] NA NA "In the antebellum United States, Solomon Northu
p, a free black man from upstate New York, is abducted and sold into slavery." "A DEA ag
ent and a naval intelligence officer find themselves on the run after a botched attempt
to infiltrate a"| __truncated__ ...
##  $ rated        : chr [1:1794] NA NA "R" "R" ...
##  $ response     : logi [1:1794] NA NA TRUE TRUE TRUE TRUE ...
##  $ language     : chr [1:1794] NA NA "English" "English, Spanish" ...
##  $ country      : chr [1:1794] NA NA "USA, UK" "USA" ...
##  $ writer       : chr [1:1794] NA NA "John Ridley (screenplay), Solomon Northup (base
d on \"Twelve Years a Slave\" by)" "Blake Masters (screenplay), Steven Grant (based on t
he Boom! Studios graphic novels by)" ...
##  $ metascore    : num [1:1794] NA NA 97 55 62 29 28 55 48 33 ...
##  $ imdb_rating  : num [1:1794] NA NA 8.3 6.8 7.6 6.6 5.4 7.8 5.7 5 ...
##  $ director     : chr [1:1794] NA NA "Steve McQueen" "Baltasar Kormákur" ...
##  $ released     : chr [1:1794] NA NA "08 Nov 2013" "02 Aug 2013" ...
##  $ actors       : chr [1:1794] NA NA "Chiwetel Ejiofor, Dwight Henry, Dickie Gravois,
Bryan Batt" "Denzel Washington, Mark Wahlberg, Paula Patton, Edward James Olmos" ...
##  $ genre        : chr [1:1794] NA NA "Biography, Drama, History" "Action, Comedy, Cri
me" ...
##  $ awards       : chr [1:1794] NA NA "Won 3 Oscars. Another 131 wins & 137 nomination
s." "1 win." ...
##  $ runtime      : chr [1:1794] NA NA "134 min" "109 min" ...
##  $ type         : chr [1:1794] NA NA "movie" "movie" ...
##  $ poster       : chr [1:1794] NA NA "http://ia.media-imdb.com/images/M/MV5BMjExMTEzO
DkyN15BMl5BanBnXkFtZTcwNTU4NTc4OQ@@._V1_SX300.jpg" "http://ia.media-imdb.com/images/M/MV
5BNTQ5MTgzNDg4OF5BMl5BanBnXkFtZTcwMjAyODEzOQ@@._V1_SX300.jpg" ...
##  $ imdb_votes   : num [1:1794] NA NA 143446 87301 43608 ...
##  $ error        : logi [1:1794] NA NA NA NA NA NA ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   year = col_double(),
##   ..   imdb = col_character(),
##   ..   title = col_character(),
```

```
##   ..   test = col_character(),
##   ..   clean_test = col_character(),
##   ..   binary = col_character(),
##   ..   budget = col_double(),
##   ..   domgross = col_character(),
##   ..   intgross = col_character(),
##   ..   code = col_character(),
##   ..   budget_2013 = col_double(),
##   ..   domgross_2013 = col_character(),
##   ..   intgross_2013 = col_character(),
##   ..   period_code = col_double(),
##   ..   decade_code = col_double(),
##   ..   imdb_id = col_character(),
##   ..   plot = col_character(),
##   ..   rated = col_character(),
##   ..   response = col_logical(),
##   ..   language = col_character(),
##   ..   country = col_character(),
##   ..   writer = col_character(),
##   ..   metascore = col_double(),
##   ..   imdb_rating = col_double(),
##   ..   director = col_character(),
##   ..   released = col_character(),
##   ..   actors = col_character(),
##   ..   genre = col_character(),
##   ..   awards = col_character(),
##   ..   runtime = col_character(),
##   ..   type = col_character(),
##   ..   poster = col_character(),
##   ..   imdb_votes = col_number(),
##   ..   error = col_logical()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

In an R markdown file, the output will always appear within the file. If you are running a regular script, than your output will appear in the console below.

Looks like there's a ton of movies in this list. What can we do to figure out the range of years it encompasses?

```
range(movies$year)
```

```
## [1] 1970 2013
```

Let's stick with more recent movies to make our data a little bit more manageable. We're going to use a tidyverse function called `filter()` to filter out movies from before the year 2000. Take note of the tidyverse pipe syntax.

```
movies_new <- movies %>%
  filter(year >= 2000)
```

Descriptive statistics

Let's begin to get some more meaningful information about the data. First, how many movies do we have in each group? (passing and failing the Bechdel test)

```
movies_new %>% count(binary)
```

binary <chr>	n <int>
FAIL	672
PASS	606

2 rows

Looks like our groups are pretty evenly-sized, which is nice.

How would we calculate the overall mean metacritic score, across all movies?

```
mean(movies$metascore)
```

```
## [1] NA
```

Uh oh! Why are we getting an answer of 'NA'? In R, NA means not available, and it is automatically assigned to any missing data when we load it in. If you inspect the data, you can see there are some metacritic scores with a value of 'NA'. What do we do about this? Luckily, the `mean` function has an easy solution.

```
mean(movies_new$metascore, na.rm = T)
```

```
## [1] 57.87975
```

That's better. Let's say we also wanted to get the mean domestic gross, in 2013 dollars. Before we do that though, go back to your `str()` output and look at what the data type is for the `domgross_2013` variable. For some reason, it's being read as a character class, not a number class. Why might that be?

```
movies_new %>% count(domgross_2013)
```

domgross_2013 <chr>	n <int>
#N/A	15
100004670	1
10019987	1
100347776	1
100674928	1

domgross_2013 <chr>	n <int>
101179654	1
101470202	1
101802906	1
101853441	1
102008450	1

1-10 of 1,264 rows Previous **1** 2 3 4 5 6 … 127 Next

Weird! That is not the same type of NA that R assigns to missing data - that's just a strange character that was already in the data when we loaded it in. That's going to mess with our analyses - let's remove all the rows where we don't have a `domgross_2013 score` , and then let's tell R to re-evaluate the column as numbers, rather than characters.

```
movies_new <- movies_new %>%
  filter(domgross_2013 != "#N/A") %>%
  mutate(domgross_2013 = as.double(domgross_2013))
```

Now if we try to view the mean, we should get a real number.

```
mean(movies_new$domgross_2013)
```

```
## [1] 78878054
```

Annotation: Every dataset has its own weird quirks, and so it's hard to automate the data cleaning process. Students often have a preconception about coding, that good coders have every function memorized. But that's not actually true. Good coders know how to formulate questions and break down complex problems into smaller, more tackle-able chunks. In this tutorial, I'm trying to highlight the thought process that goes into cleaning data, rather than the exact issues that might come up or the exact functions that might be needed. In this way, I hope to give students practice in the more abstract skills that are crucial when coding, rather than having them memorize specific functions.

Inferential statistics

Let's run a t test. First, let's see if there is a difference in metacritic score between our Bechdel groups. We're going to use the `t.test` function, but we're going to use it slightly differently than we have in the past. Rather than have two separate columns for each group, we're going to have one column with all of the data, and another column that indicates what group each data point is from.

```
t.test(metascore ~ binary, data = movies_new)
```

```
##
##  Welch Two Sample t-test
##
## data:  metascore by binary
## t = 1.0881, df = 1090.9, p-value = 0.2768
## alternative hypothesis: true difference in means between group FAIL and group PASS is
not equal to 0
## 95 percent confidence interval:
##  -0.8857123  3.0911465
## sample estimates:
## mean in group FAIL mean in group PASS
##           58.42373           57.32101
```

Looks like there's no difference in critic scores of movies regardless of whether they pass the Bechdel test. What about how they perform at the US box office?

```
t.test(domgross_2013 ~ binary, data = movies_new)
```

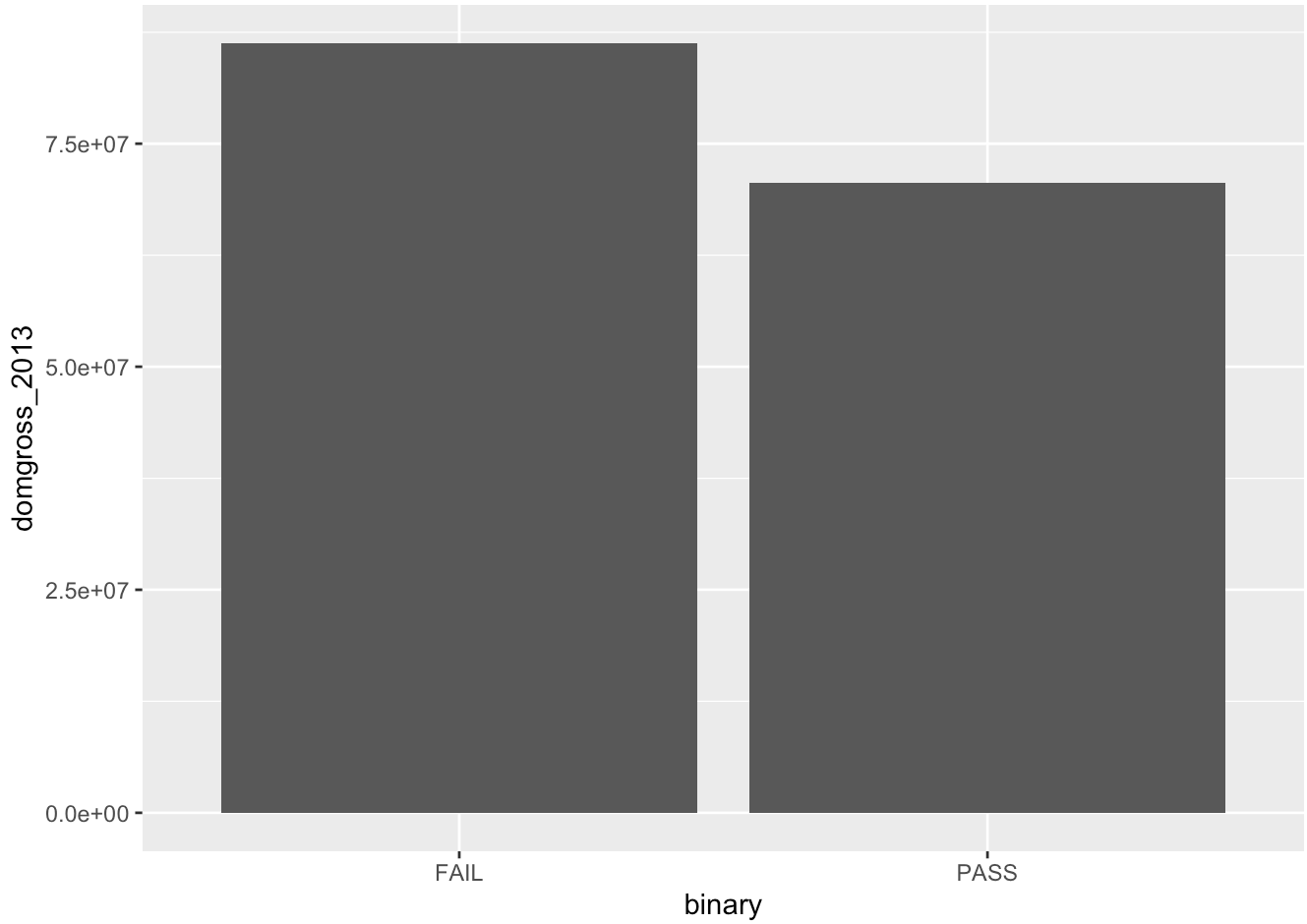
```
##
##  Welch Two Sample t-test
##
## data:  domgross_2013 by binary
## t = 3.0467, df = 1257, p-value = 0.002362
## alternative hypothesis: true difference in means between group FAIL and group PASS is
not equal to 0
## 95 percent confidence interval:
##   5579040 25757348
## sample estimates:
## mean in group FAIL mean in group PASS
##           86284160           70615966
```

What can we conclude from these results? Looks like the domestic gross for movies that pass the Bechdel test is significantly lower (how much lower?) than movies that fail the Bechdel test, even though critics rate them similarly. That's kind of sad, if you ask me.

Plotting

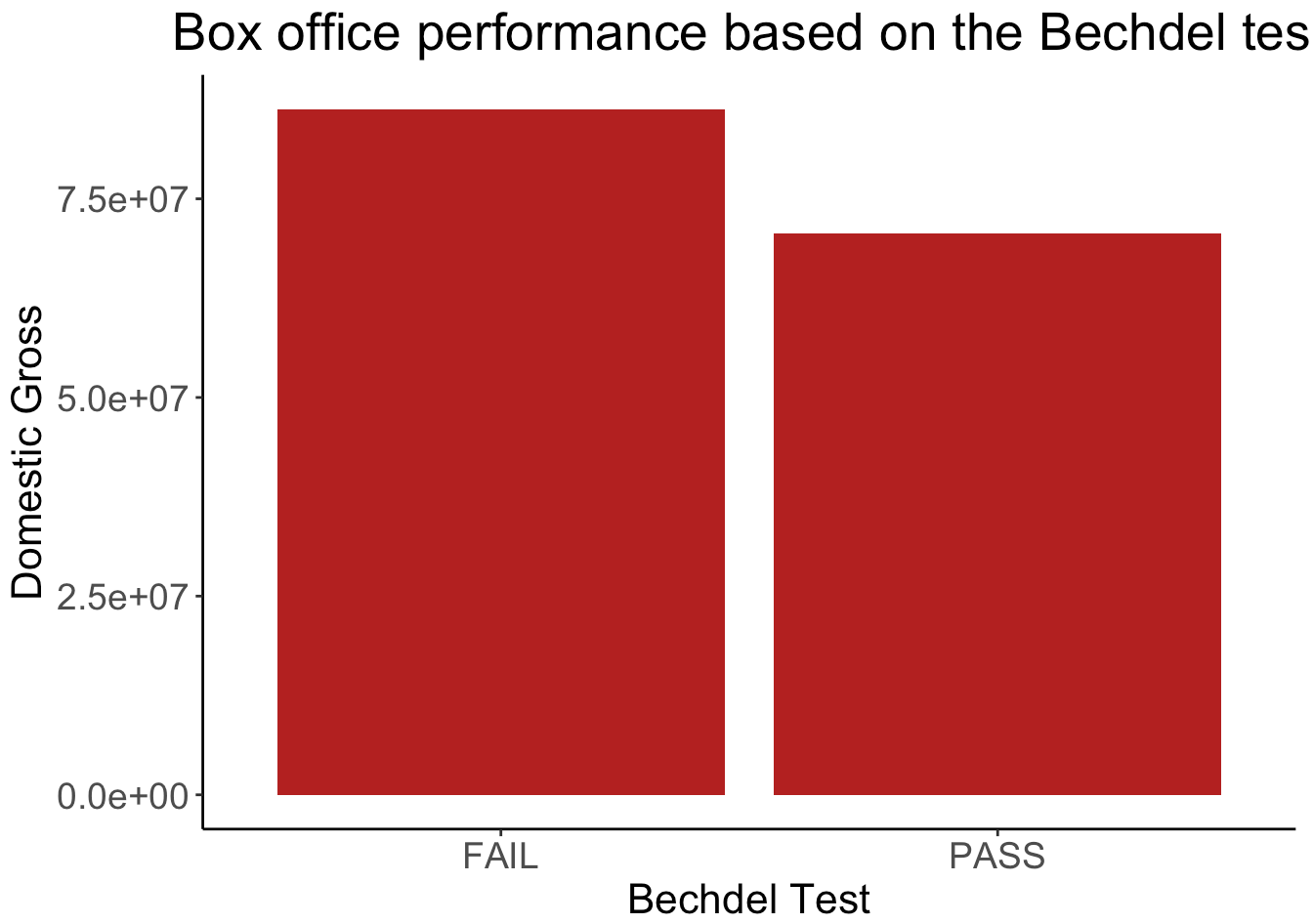
We've got some interesting results here, but showing is always better than telling. Let's plot the domestic gross.

```
ggplot(movies_new, aes(x = binary, y = domgross_2013)) +
  geom_bar(stat = "summary", fun = "mean")
```



Let's make this graph actually look nice.

```
ggplot(movies_new, aes(x = binary, y = domgross_2013)) +
  geom_bar(stat = "summary", fun = "mean", fill = "firebrick") +
  theme_classic() +
  labs(x = "Bechdel Test", y = "Domestic Gross",
       title = "Box office performance based on the Bechdel test") +
  theme(plot.title = element_text(size = 20,hjust = .5),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 14))
```



Annotation: A code chunk like the above represents one of the most difficult aspects of coding tutorials: Trust. By introducing a whole host of new syntax and functions to students that I haven't previously prepared them for, I'm asking them to trust me that these things will make sense with enough practice. Furthermore, I'm trying to emphasize that the specifics don't matter - I want to show them what's possible, so they can know what to search for when they need help later on. Finally, this code chunk has a more practical purpose: I want students to be able to copy parts of it and build on it when they create their own visualizations for their own data.

And that's all a simple data analysis takes: Load your data, clean it up, statistical test, and plot. Not too bad, right??