Week 2 Assignment

Adam Douglas

9/7/2018

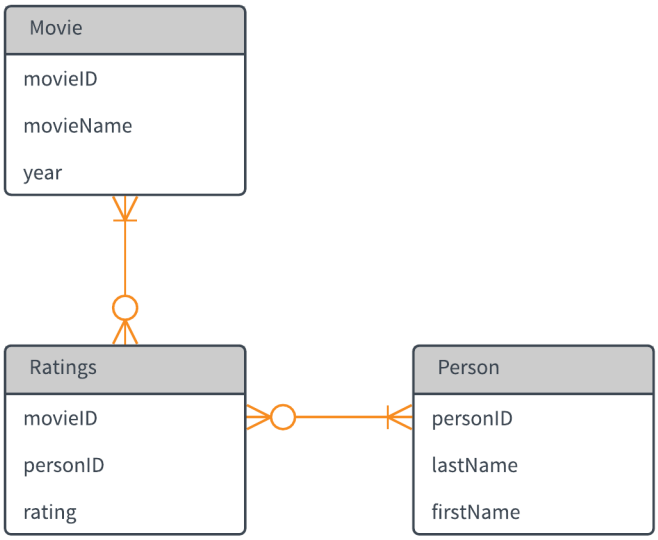
The Database

We begin by creating a database to store our movies, raters, and ratings in. For this exercise, I have chosen PostgreSQL. Not only is it a popular open source choice and a robust DBMS, but it is also one of the DBMS I have not used yet so I wanted to give it a try.

The Structure

The structure of the database will be simple enough. One table with movies in it, one with people who see one or more of the movies, and an associative table with each person's ratings. Below is a basic ER diagram outlining the general structure:

```
knitr::include_graphics("db.png")
```



The Code

Once we have our structure set, we begin creating the tables themselves. The code used to create the tables in the PostgreSQL database system and load the data into them is included in a separate file called `Assignment 2.sql`. To give an idea of what the structure looks like from a code perspective, below are the table creation statements used:

> CREATE TABLE public.movies ("movieID" integer NOT NULL, "movieName" character varying(200), year integer, CONSTRAINT movies_pkey PRIMARY KEY ("movieID"))

CREATE TABLE public.person (“personID” integer NOT NULL, “lastName” character varying(20), “firstName” character varying(40), CONSTRAINT person_pkey PRIMARY KEY (“personID”))

CREATE TABLE public.ratings (“movieID” integer, “personID” integer, rating integer, PRIMARY KEY (“movieID”, “personID”), CONSTRAINT fk_movie FOREIGN KEY (“movieID”) REFERENCES public.movies (“movieID”), CONSTRAINT fk_person FOREIGN KEY (“personID”) REFERENCES public.person (“personID”), CONSTRAINT ck_rating CHECK (rating >= 1 AND rating <= 5))

Note that the ratings table has a few constraints on it. First it uses foreign keys to ensure that any `movieID` or `personID` put into it has a match in their respective tables. Also, we use a check constraint to ensure that ratings are from 1 - 5 stars.

The Data

Now that we have the data loaded into our database, we will need to query it to do any kind of analysis. We begin by loading the package we'll use to connect to the PostgreSQL database and setting up some basic variables with information about the database itself (such as user, password, what port it is running on, etc.):

```
# Load the package to connect
library(RPostgreSQL)

# Variables to create the connection
host <- "localhost"
usr <- "postgres"
pass <- params$password
port <- 5432
database <- "movies"
```

Now we create the actual driver and connection:

```
# Creates the actual connection object
conn <- dbConnect(RPostgreSQL::PostgreSQL(),host = host, dbname = database, user = usr, password = pass
```

Now that we have the connection object, let's test to make sure it works:

```
# A test SQL query
testSQL <- dbSendQuery(conn, "select count(*) from public.movies")

# Retreive the results
test <- dbFetch(testSQL)
```

We see from the results that we have 6 records in the `movies` table, which is what we expect.

Now, we will aquire the data. Rather than writing a single query combining the tables, I'm going to pull each one into its own object and work with them within R. I am doing this for two reasons:

1. I am not sure yet what sort of analysis I want to perform on the data. If I have the data as a result of a SQL statement, I need to decide what I want when I execute that SQL. This way I can adapt as I go by filtering the data in its raw form within R.
2. It gives me an opportunity to work with `dplyr` and other tools to format and tidy data.

```
# Get the movies
movieSQL <- dbSendQuery(conn, "select * from public.movies")
movies <- dbFetch(movieSQL)
```

```
# Get the people
personSQL <- dbSendQuery(conn, "select * from public.person")
person <- dbFetch(personSQL)
# Get the ratings
ratingsSQL <- dbSendQuery(conn, "select * from public.ratings")
ratings <- dbFetch(ratingsSQL)
```

Now that we have our data, we can close the database connection:

```
dbDisconnect(conn)
```

The Analysis

First let's look at the data we've gathered:

```
str(movies)
```

```
## 'data.frame':    6 obs. of  3 variables:
##  $ movieID  : int  1 2 3 4 5 6
##  $ movieName: chr  "Avengers: Infinity War" "BlacKkKlansman" "Crazy Rich Asians" "Won't You Be My Ne
##  $ year     : int  2018 2018 2018 2018 2018 2017
```

```
str(person)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ personID : int  1 2 3 4 5
##  $ lastName : chr  "Douglas" "Douglas" "Douglas" "Douglas" ...
##  $ firstName: chr  "Adam" "April" "Amber" "Tyler" ...
```

```
str(ratings)
```

```
## 'data.frame':    21 obs. of  3 variables:
##  $ movieID : int  1 1 1 2 2 2 3 3 3 4 ...
##  $ personID: int  1 4 5 1 2 3 2 3 5 1 ...
##  $ rating  : int  5 4 5 4 3 4 4 5 5 5 ...
```

We have both int variables and character variables, which is what we expected from the database.

First, let's look at who is doing how many ratings. Before we can answer this, and other, questions, we need to combine these datasets into a single tidy dataset. We will do this by using joining functions courtesy of the `dplyr` package, which is part of the Tidyverse set of packages; the syntax is similar to SQL.

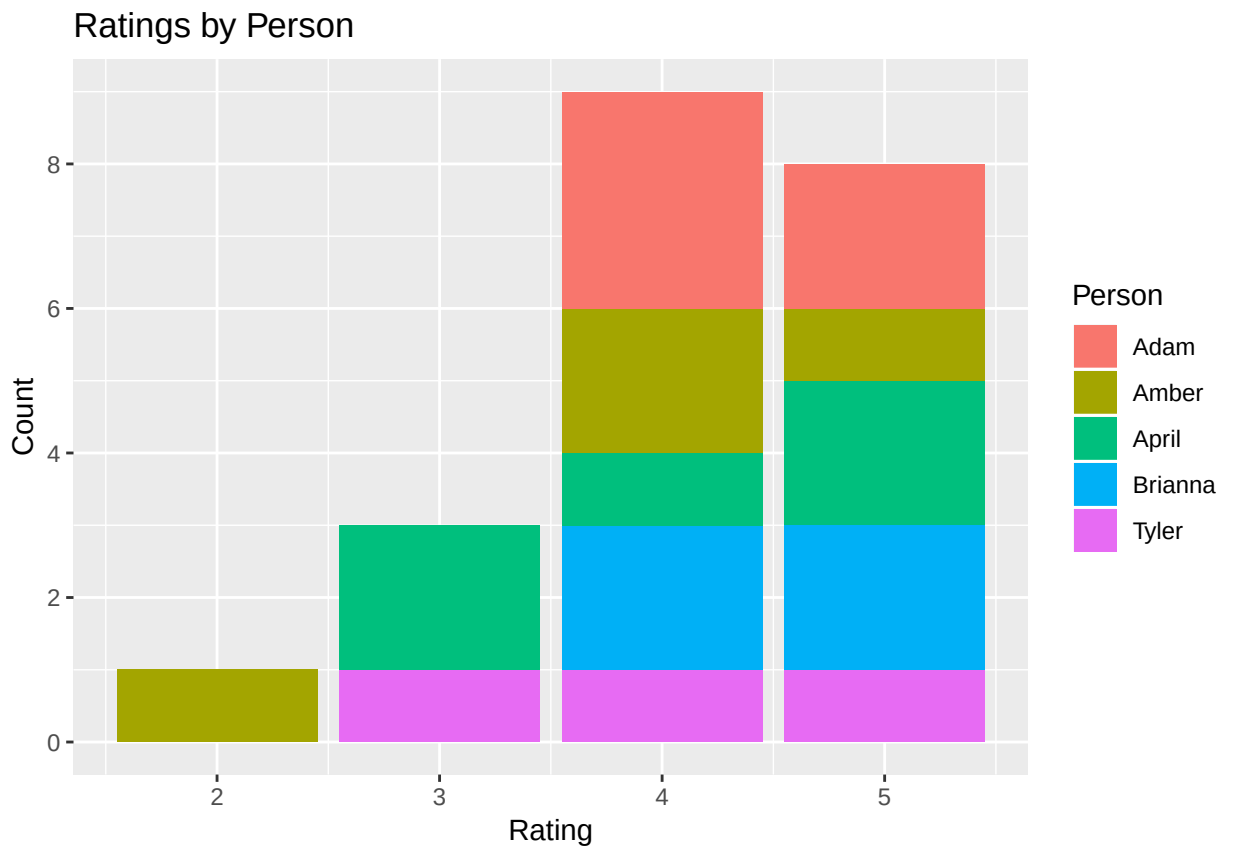
Let's look at everyone's ratings, regardless of the movie:

```
# We combine the datasets and select only the fields we want
ratingsByPerson <- movies %>% left_join(ratings, by="movieID") %>% inner_join(person, by="personID") %>%

glimpse(ratingsByPerson)
```

```
## Observations: 21
## Variables: 2
## $ firstName <chr> "Adam", "Tyler", "Brianna", "Adam", "April", "Amber"...
## $ rating    <int> 5, 4, 5, 4, 3, 4, 4, 5, 5, 5, 5, 4, 3, 4, 4, 3, 2, 4...
```

```
ggplot(ratingsByPerson, aes(rating, fill=firstName)) + geom_bar() + scale_y_continuous(breaks=seq(0,10,
```



Looking at the distribution of ratings, it appears that I (Adam) tend to rate high (only 4's or 5's) and my wife (April) tends to rate more evenly across the 3-4 range. Perhaps I need to choose the movies less often.

Now let's look at how individual movies fared overall:

```
# Like before, we combine the datasets and select only the fields we want
ratingsByMovie <- movies %>% left_join(ratings, by="movieID") %>% select(movieName, rating)

glimpse(ratingsByMovie)
```

```
## Observations: 21
## Variables: 2
## $ movieName <chr> "Avengers: Infinity War", "Avengers: Infinity War", ...
## $ rating    <int> 5, 4, 5, 4, 3, 4, 4, 5, 5, 5, 5, 4, 3, 4, 4, 3, 2, 4...
```

Let's get a sense of how the movies fared in terms of typical ratings:

```
ratingsByMovie %>% group_by(movieName) %>% summarize(AvgRating = mean(rating), NumRatings = n()) %>% ar
```

```
## # A tibble: 6 x 3
##   movieName                                AvgRating NumRatings
##   <chr>                                        <dbl>      <int>
## 1 Avengers: Infinity War                        4.67          3
## 2 Crazy Rich Asians                             4.67          3
## 3 Three BIillboards Outside Ebbing, Missouri    4.67          3
## 4 Won't You Be My Neighbor?                     4.2           5
## 5 BlacKkKlansman                                3.67          3
## 6 A Quiet Place                                 3.25          4
```

"Avengers: Infinity War", "Crazy Rich Asians", and "Three Billboards Outside Ebbing, Missouri" are the most popular when one looks at mean ratings. The Mr. Roger's documentary "Won't You Be My Neighbor" had the most appeal amongst the family, as it had the largest number of ratings overall (and got a 4.2 mean rating!).

Finally, let's look at how each person rated each individual movie:

```
# Once again, we select just what we need
ratingsByMovieByPerson <- movies %>% left_join(ratings, by="movieID") %>% inner_join(person, by="personI

glimpse(ratingsByMovieByPerson)
```

```
## Observations: 21
## Variables: 3
## $ movieName <chr> "Avengers: Infinity War", "Avengers: Infinity War", ...
## $ firstName <chr> "Adam", "Tyler", "Brianna", "Adam", "April", "Amber"...
## $ rating    <int> 5, 4, 5, 4, 3, 4, 4, 5, 5, 5, 5, 4, 3, 4, 4, 3, 2, 4...
```

Let's take a numerical look at "Avengers: Infinity War", as an example:

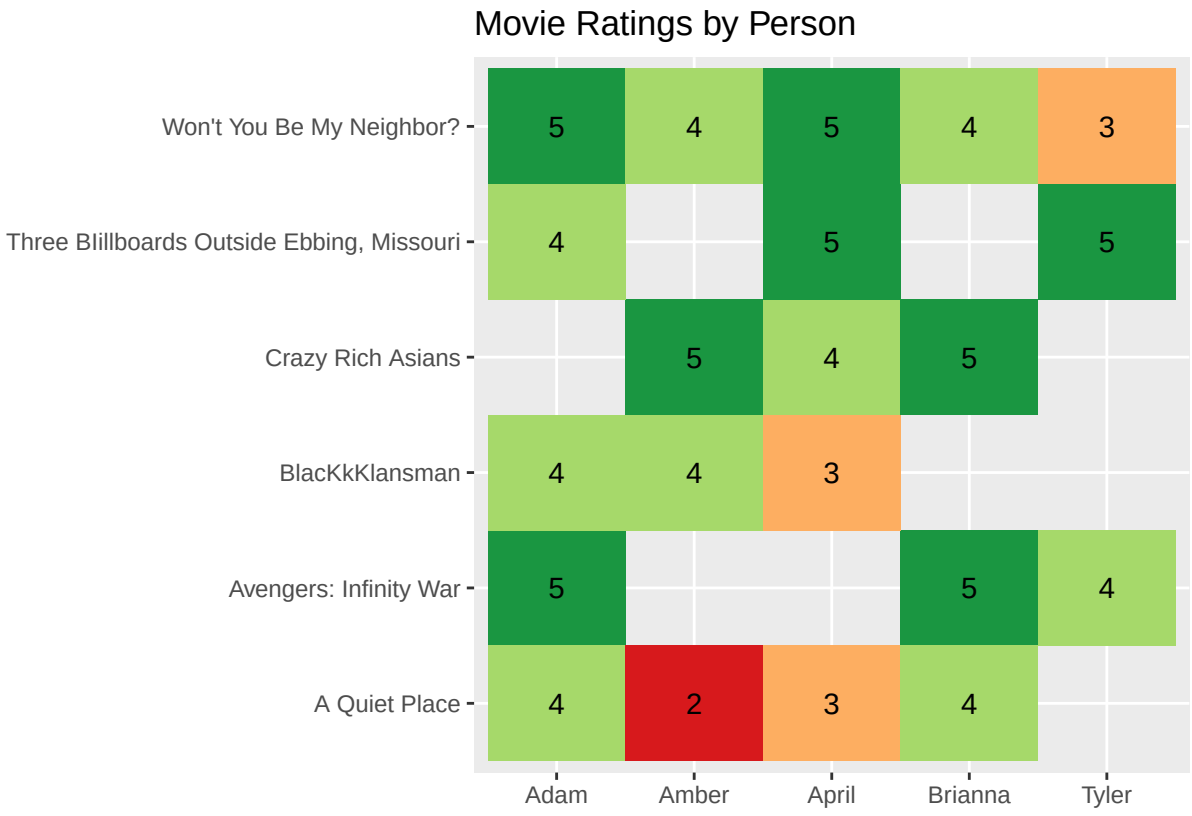
```
ratingsByMovieByPerson %>% filter(movieName == "Avengers: Infinity War")
```

```
##                movieName firstName rating
## 1 Avengers: Infinity War      Adam      5
## 2 Avengers: Infinity War     Tyler      4
## 3 Avengers: Infinity War   Brianna      5
```

Three people rated the movie, and overall it was rated very well.

However, rather than look at each one individually, we can visualize the ratings by person and movie in one graphic:

```
ggplot(ratingsByMovieByPerson, aes(firstName, movieName)) + geom_tile(aes(fill=as.factor(rating))) + sca
```


Movie Ratings by Person
Won't You Be My Neighbor?
Three BIillboards Outside Ebbing, Missouri
Crazy Rich Asians
BlacKkKlansman
Avengers: Infinity War
A Quiet Place
5
4
5
4
3
4
5
5
5
4
5
4
4
3
5
5
4
4
2
3
4
Adam
Amber
April
Brianna
Tyler