Data 607 Assignment 2

Matthew Tillmawitz

The goal of this assignment is to create a SQL database to store the movie ratings of several individuals. Given the small sample size a simple solution is presented and a discussion of how this solution could be expanded upon is included. The SQL database chosen was SQLite due to the ease with which a simple in memory database can be set up in the code. If we needed to connect to a database on an external host or authenticate with the database we would simply add the required parameters to the `dbConnect()` function call. The simplest way to create a table using RSQLite is actually to create a dataframe first, then write it to the database which results in a slightly redundant example of reading data from a database into a dataframe. Such examples can actually be quite useful when practicing iterative development however, as one can be certain they have correctly implemented their connection, authentication, read, and write functions when moving on to more sophisticated solutions. The simple solution is as follows:

First ensure you have RSQLite installed by running

```
install.packages("RSQLite")
```

Followed by the solution in question

```
library(DBI)

input_dataframe <- data.frame(viewer_name = c("Emily", "John", "Sam", "Roshni", "Mike", "Justin"),
                              inside_out_2 = c(4, NA, NA, 4, NA, 4),
                              deadpool_and_wolverine = c(2, 5, 3, 3, 5, 5),
                              dune_part_2 = c(4, 4, 2, 3, 2, 3),
                              twisters = c(NA, 5, NA, NA, 5, 2),
                              godzilla_x_kong = c(NA, 5, NA, NA, 5, NA),
                              the_fall_guy = c(3, NA, 3, 3, NA, 3))

con <- dbConnect(RSQLite::SQLite(), ":memory:")
dbWriteTable(con, "movie_ratings", input_dataframe)

output_dataframe <- dbReadTable(con, "movie_ratings")
dbDisconnect(con)
output_dataframe
```

```
##   viewer_name inside_out_2 deadpool_and_wolverine dune_part_2 twisters
## 1       Emily            4                       2           4       NA
## 2        John           NA                       5           4        5
## 3         Sam           NA                       3           2       NA
## 4      Roshni            4                       3           3       NA
## 5        Mike           NA                       5           2        5
## 6      Justin            4                       5           3        2
##   godzilla_x_kong the_fall_guy
## 1              NA            3
## 2               5           NA
```

```
## 3              NA            3
## 4              NA            3
## 5               5           NA
## 6              NA            3
```

The empty value NA was used when a given viewer had not seen a movie as it can be simply converted to a database safe type without the use of additional code. It has the added benefit of being easily handled by the summary functions in R when conducting analyses of the read dataframes. If we were to significantly expand the data set to include more movies or if we wanted to store additional information about the viewers or movies we would want to split the data into several normalized tables. An example of how to construct the tables using SQL is provided below for context.

```sql
create table viewers(
  viewer_id int,
  first_name varchar(20),
  last_name varchar(20),
  age int
);

create table movies(
  movie_id int,
  title varchar(50),
  release date,
  runtime int
);

create table viewer_ratings(
  viewer_id int,
  movie_id int,
  score int
);
```

Normalizing in this way would prevent unnecessary data duplication, keeping the database manageable from both a size and data cleanliness perspective. The scores for any viewer or movie can be easily retrieved by doing a join across tables on the viewer_id or movie_id fields.