Eurovision 2021: SQL and R

Jake Simon

9/20/2021

Libraries

First, let's take a look at the libraries used for this program:

```
# Libraries for HTML reading in R
library(xml2)
```

```
## Warning: package 'xml2' was built under R version 3.6.2
```

```
library(rvest)

# Libraries for SQL
library(RSQLite)
library(sqldf)
```

```
## Loading required package: gsubfn
```

```
## Loading required package: proto
```

```
library(DBI)
```

Reading the HTML into R

First, we want to choose the year of the contest we will be importing and exporting. This will make it easy to convert and clean each year by simply changing the year selected.

```
year <- 2021
```

Next, we will take the `read_html` function, and import our contest's html file as a 2-D list.

```
# Reads the html file into the R program, and splits the table from the other data:
esc <- read_html(paste0("html database/Eurovision Song Contest Database ",
                        year, ".html")) %>% html_table()

# The table is the second element from the html_table function.
esc_table <- esc[[2]]

esc_finals <- read_html(paste0("html finals database/Eurovision Song Contest Database ",
                        year, ".html")) %>% html_table()

esc_finals_table <- esc_finals[[2]]

# Remove the empty columns that are there for some reason:
esc_table <- esc_table[,-1]
esc_finals_table <- esc_finals_table[,-4]
```

```r
# We also need to rename the Order column "No." to "Song_Order", to avoid syntax errors:
colnames(esc_finals_table)[3] = "Song_Order"
```

Afterwards, we will apply the following SQL code to clean and export the data.

```r
# This connects us to the SQLite file for our dataset:
esc_db <- dbConnect(RSQLite::SQLite(), "esc-db.sqlite")

# Now, we can take that table, and add it to the SQL database:
dbWriteTable(esc_db, "ESC", esc_table, overwrite=T)
dbWriteTable(esc_db, "ESC_FINALS", esc_finals_table, overwrite=T)

# Let's add a new column to the table called "Points_Earned".
dbSendQuery(esc_db, '
            ALTER TABLE ESC_FINALS
            ADD COLUMN Percent_Earned DOUBLE')
```

```
## <SQLiteResult>
##   SQL
##             ALTER TABLE ESC_FINALS
##             ADD COLUMN Percent_Earned DOUBLE
##   ROWS Fetched: 0 [complete]
##        Changed: 0
```

```r
# We'll take our new column, and calculate the percentage of points earned by country.
# Note: The denominator, 24 * (# total entries-1), is the max. points a country could score.
dbSendStatement(esc_db, '
            UPDATE ESC_FINALS
            SET Percent_Earned =
              (Points+0.0) / (12 * 2.0 * ((SELECT COUNT(*) FROM ESC)-1))')
```

```
## Warning: Closing open result set, pending rows
```

```
## <SQLiteResult>
##   SQL
##             UPDATE ESC_FINALS
##             SET Percent_Earned =
##               (Points+0.0) / (12 * 2.0 * ((SELECT COUNT(*) FROM ESC)-1))
##   ROWS Fetched: 0 [complete]
##        Changed: 26
```

```r
# Now, let's take the data and subset it by the following columns:
le_table <- dbGetQuery(esc_db, '
              SELECT Place, Points, Percent_Earned, Song_Order, Country,
              (SELECT COUNT(*) FROM ESC) AS Participants
              FROM ESC_FINALS')
```

```
## Warning: Closing open result set, pending rows
```

```r
# Since this is R, we can assign the SQL table to an R table, and export that as a
# .csv file for our Python project:
write.csv(le_table, file=paste0("csv database/", year, " Eurovision Final Results.csv"))

# Now, let's disconnect from the SQLite server, and head on over to Python!
dbDisconnect(esc_db)
```