Midterm

This midterm consists of 3 questions. All aspects of the midterm must be handed in through crowdmark. The last question is a project: you should choose one of two possible projects for this questions.

Question 1: Essay

Has data science changed the world? Write a roughly 800 word (before referneces) argumentative essay exploring this question. Your essay should have an introduction, a body and a conclusion. Your essay should have a title and a thesis statement. (A thesis statement is one or two sentences that provides a concise and focused summary of the argument that will be made in the body.) Your thesis statement should appear in the introduction in italics. The essay should be double spaced, but all other aspects of the typesetting are up to you (inclusion of section markers, capitalization of title, style for references ...). Please include your name under the title. Provide references for any material you draw on in your argument, or for any facts you report. (You may reference blog posts, you may not reference wikis.) The marking scheme is as follows: 40% quality of argumentation, 40% quality of research and originality, 10% flow and quality of writing. Upload a *p*df version of your essay to *crowdmark*.

(10 points)

Question 2: Databases

Consider the database in the file `stat240.sqlite` provided in this midterm archive. This database contains a table named **citiesA** containing the area of cities and a table named **citiesB** containing the population of cities. This question has 4 parts, which must all be completed. For each part, provide code and output in a single *pdf* file through *crowdmark*. Provide axis-lables and titles for all of your plots.

Question 2, Part I

Connect to the database in the file `stat240.sqlite` and output the names of the tables in the database. For each table, output the names of the columns of the table and the data types of the columns and the number of entries in the table.

(3 points)

Question 2, Part II

Use *SQL* to extract the unique combinations of **province** and **type** from the **citiesP** table, and provide the number of such unique combinations.

(2 points)

Question 2, Part III

Use *SQL* to obtain the number of municipalities within each province in the database. Restrict to location names that are present in both **citiesA** and **citiesP** tables. Provide a plot of the number of resulting municipalities from each province.

(5 points)

Question 2, Part IV

For each location in the **citiesP** table, the columns **Order2011** and **Order2016** represent the popularity of the destination with tourists in the years 2011 and 2016 respectively (the tourist rank orders). Extract the tourist rank order for 2011 (**Order2011**) and for 2016 (**Order2016**) for each location in the **citiesP** table. Provide a scatter plot of the 2011 values against the 2016 values (*i.e.*, a plot with one point per location and the 2011 values on the y-axis and the 2016 values on the x-axis).

(5 points)

Question 3: Project

Complete one of the following two projects. You may choose which of these two projects to undertake. Only hand in your work for one of these two projects. For the project that you choose, hand in the following two parts through *crowdmark*. Partial marks will be awarded if you satisfy some but not all of the project requirements, or if you demonstrate that your code is heading in the right direction.

Question 3: Part I

Provide a *pdf* containing all of the code that you wrote for this project. Begin the code with a comment indicating which project you've done. If your code spans multiple files, put all of the code into a single *pdf* and use comments to indicate the beginning of the files and the file names. You may use use code from outside sources in your project provided that it is released under an open license and provided that you clearly indicate the code with comments.

(10 points)

Question 3: Part II

Write a short report (in prose) arguing that the code you've provided works as required and provide a *pdf* of the report. Include output of *R* sessions in labelled figures. For example, if you choose *Project A*, you could show the `translink` function being called with different arguments, and show the corresponding elements of the `data` list proving that correct bus routes are returned. You could also show that for a given element of the `data` list, parameters to the `translink` function can be specified in order to return that affected bus route.

(10 points)

Project A: Bus Disruptions

Consider the Twitter data in the file `translink.RData` provided in the archive for this midterm. Write an *R* function named `translink` that

takes 4 numeric arguments: a year (a numeric value such as 2020), a month (a numeric value between 1 and 12, inclusive with 1 indicating January), and a day of the month, and an hour of the day (in 24 hour time). The *R* function should return a list with two elements: 1) an element with the name *start* and with a value specifying a character vector enumerating all bus routes that started to have disruptions during the hour indicated by the date and time provided to the function, 2) an element with the name *stop* and with the value specifying a character vector enumerating all bus routes that stopped having a disruption during the hour indicated by the date and time provided to the function. You don't have to care about timezones in this question: you can assume that the time specified by the parameters to the function are in the same time zone as the data in `translink.RData`. Disruptions are defined as starting or stopping only if a Tweet indicating such is present in `translink.RData` (*i.e.*, you don't have to consider times that fall outside of the data provided in `translink.RData`, nor do you have to access Twitter APIs to access relevant tweets that may not be in `translink.RData`). Note that some of the tweets in `translink.RData` may be truncated: you may ignore the truncated portions (this is approximate). Example usage follows.

```
> disruptions = translink(2020, 1, 26, 3)
> disruptions$start
[1] "401" "406"
> disruptions$stop
NULL
```

Project B: Album Ratings

Consider the *Best New Albums section* of the popular music site *Pitchfork*. Write an *R* function to retrieve the text of a review of an album, and also the rating that the reviewer gives to the album (on the scale 0.0 to 10.0). Provide a function named `pitchfork` which takes a single parameter: a URL specified by a string. When this function is passed the URL to a review of an album in the *Best New Albums section* of *Pitchfork*, the `pitchfork`

function should return a list with two elements: 1) an element named `text` with value given by the body of the review as a character string (with all HTML removed), 2) an element named `rating` with value given by the decimal value of the review, as a numeric. You can find URLs for these reviews by navigating to `https://pitchfork.com/reviews/best/albums/` and right clicking on an album and then selecting something like *Copy Link*. Example usage follows. In this listing, the ellipsis indicates cropped *R* output.

```
> url = 'https://pitchfork.com/reviews/albums/jeff-
   parker-suite-for-max-brown/'
> review = pitchfork(url)
> review$text
[1] "Jeff Parker always always writes parts that sound
    unassuming at first listen and unavoidable by the
   fifth.

...

 And together, with the help of some friends,  they
    build a nest from which to watch the world."
> review$rating
[1] 8.4
```