Final Practice Problems

YSC2210 - Data Aanalysis and Visualisation with R

Michael T. Gastner

Academic Year 2021/2, Semester 2

1 Exam rules

Important: please read the following rules for the exam.

The exam is open-book and open-Internet with the exception that **any form of real-time communication** (e.g. email, Facebook or user forums) **and file exchanges** (e.g. Dropbox or RStudio Server Pro) **are strictly forbidden**. Do not look at another person's monitor. You must record your screen from the beginning of the exam until after you submitted your solutions to Canvas. You can only use one electronic device during the exam, and the screen of this device must be visible in the screen recording. **It is forbidden to share code with others**—before, during and after the exam—if this code can serve as help for other students during the exam (including students in future editions of this course). If your solution is based on ideas from a book, a website or code you shared with another student prior to the exam (e.g. your team's homework solutions), you must clearly state the source in your submission.

Violations of these rules result in appropriate disciplinary procedures.

2 These problems are not representative of the exam's difficulty

I offer these problems for your practice only. They are not intended to be representative of the problems on the exam. However, these problems may aid you in your preparation.

3 R Markdown style

- Load all the packages you need for your solutions at the beginning of your R Markdown file, for example:

  ```
  library(lubridate)
  library(tidyverse)
  library(tmap)
  ```

  Show the code, but do not show package start-up messages in the knitted file.
- Adhere to the tidyverse style (https://style.tidyverse.org/).
- Use code chunk options `fig.width`, `fig.height` and `out.width` to adjust figure dimensions. All parts of the figures should be clearly legible without appearing disproportionately large compared to the font size of the running text in the knitted R Markdown file.
- If knitting the full document takes a long time, you can use the chunk option `cache=TRUE`. You can also submit separate R Markdown files for each problem; however, please do not forget to upload all files to Canvas before the end of the exam.

4 Florentine marriage network

A classic data set in social network analysis is the 'Florentine Marriage Network' (figure 1). It consists of 16 nodes, each representing an influential family in Renaissance Florence (1282–1500). An edge between two families implies that there was at least one marriage between members of these two families. Marriages were often strategic decisions to build economic and political coalitions between families.

The data are from J. F. Padgett and C. K. Ansell (1993): "Robust Action and the Rise of the Medici", Am. J. Sociol. 98:1259–1319.

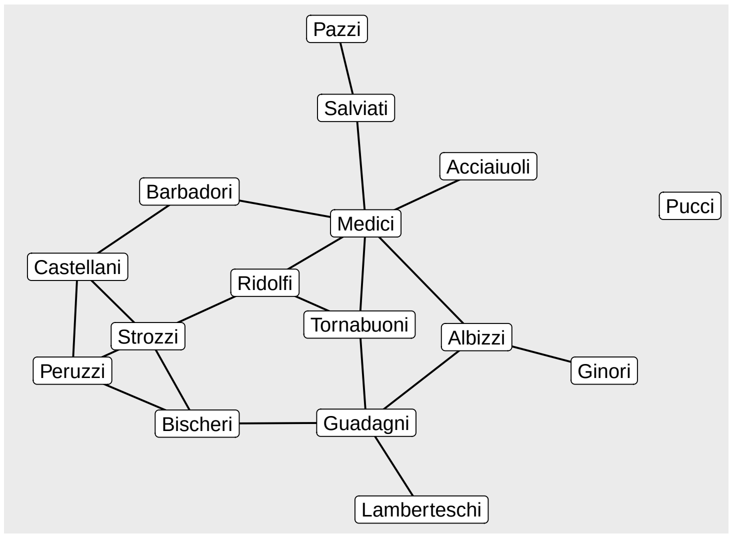

Figure 1: Florentine marriage network.

You can download the network data (`florence_nodes.csv` and `florence_edges.csv`) from Canvas.

(a) Create a `tbl_graph` from these data. Note that the network is **undirected**.

Use R to answer the following questions.

(b) How many edges are in the network?

(c) What is the density of the network?

(d) What is the transitivity of the network?

(e) What is the diameter of the network?

(f) What is (are) the shortest path(s) from the Guadagni family to the Barbadori family?

(g) Which node(s) has (have) the highest degree?

(h) Which node(s) has (have) the highest betweenness centrality?

(i) Which node(s) has (have) the highest eigenvector centrality?

(j) How many components does the network have?

(k) What is (are) the largest clique(s) in the network?

5 Data wrangling: English Premier League

The file `epl_results.csv` on the Canvas assignment page shows all the results of English Premier League games during the 2019–2020 season. The data were scraped from https://en.wikipedia.org/wiki/2019%E2%80%9320_Premier_League#Results. Here are the first few rows:

```
## # A tibble: 6 x 21
##   home   ARS   AVL   BOU   BHA   BUR   CHE   CRY   EVE   LEI   LIV   MCI   MUN  
##   <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 Arsen~ -     3-2   1-0   1-2   2-1   1-2   2-2   3-2   1-1   2-1   0-3   2-0  
## 2 Aston~ 1-0   -     1-2   2-1   2-2   1-2   2-0   2-0   1-4   1-2   1-6   0-3  
## 3 Bourn~ 1-1   2-1   -     3-1   0-1   2-2   0-2   3-1   4-1   0-3   1-3   1-0  
## 4 Brigh~ 2-1   1-1   2-0   -     1-1   1-1   0-1   3-2   0-2   1-3   0-5   0-3  
## 5 Burnl~ 0-0   1-2   3-0   1-2   -     2-4   0-2   1-0   2-1   0-3   1-4   0-2  
## 6 Chels~ 2-2   2-1   0-1   2-0   3-0   -     2-0   4-0   1-1   1-2   2-1   0-2  
## # ... with 8 more variables: NEW <chr>, NOR <chr>, SHU <chr>, SOU <chr>,
## #   TOT <chr>, WAT <chr>, WHU <chr>, WOL <chr>
```

The first column, named `home`, shows the home team. The remaining column names are three-letter abbreviations for the away team. You can assume that the n-th team name in the first column matches the n-th three-letter abbreviation (i.e. ARS is Arsenal, AVL is Aston Villa etc.). Every team played against each other team, once at home and once away.

The objective is to create the following league table:

...

```
## # A tibble: 20 x 5
##      pos team                       pts goal_diff goals_for
##    <int> <chr>                    <int>     <int>     <int>
##  1     1 Liverpool                   99        52        85
##  2     2 Manchester City             81        67       102
##  3     3 Manchester United           66        30        66
##  4     4 Chelsea                     66        15        69
##  5     5 Leicester City              62        26        67
##  6     6 Tottenham Hotspur           59        14        61
##  7     7 Wolverhampton Wanderers     59        11        51
##  8     8 Arsenal                     56         8        56
##  9     9 Sheffield United            54         0        39
## 10    10 Burnley                     54        -7        43
## 11    11 Southampton                 52        -9        51
## 12    12 Everton                     49       -12        44
## 13    13 Newcastle United            44       -20        38
## 14    14 Crystal Palace              43       -19        31
## 15    15 Brighton & Hove Albion      41       -15        39
## 16    16 West Ham United             39       -13        49
## 17    17 Aston Villa                 35       -26        41
## 18    18 Bournemouth                 34       -25        40
## 19    19 Watford                     34       -28        36
## 20    20 Norwich City                21       -49        26
```

The columns have the following meaning:

- `pos`: position in the table from best to worst. The position is determined by the cumulative points. If two or more teams have earned equally many points, then the goal difference is used as a tie breaker. If points *and* goal difference are equal, the tie is broken by the number of goals scored.
- `team`: name of the team.
- `pts`: cumulative points.

- `goal_diff`: goal difference.
- `goals_for`: cumulative goals scored by the team.

You can confirm your partial results by looking at the league table at https://en.wikipedia.org/wiki/2019%E2%80%9320_Premier_League#League_table.

(a) Import the CSV as a tibble called `epl`. Interpret the string `"-"` on the diagonal as `NA`.

(b) Make a tibble called `teams` with:

- the team's name as first column, called `name`.
- the team's three-letter abbreviation as second column, called `abbr`.

We will need this tibble later to replace abbreviations by full names.

(c) Pivot `epl` into longer format so that the home team's name is in the first column and the away team's abbreviation in the second column. Here are the first few rows:

```
## # A tibble: 6 x 3
##   home    away  result
##   <chr>   <chr> <chr>
## 1 Arsenal ARS   <NA>
## 2 Arsenal AVL   3-2
## 3 Arsenal BOU   1-0
## 4 Arsenal BHA   1-2
## 5 Arsenal BUR   2-1
## 6 Arsenal CHE   1-2
```

(d) Remove all rows that contain missing values.

(e) Replace the team's abbreviation in the column called `away` by the team's full name. (Here you need the tibble called `teams` we created earlier.)

(f) Split the value in the column called `result` into `home_goals` and `away_goals`. Note that the first number in the `result` string stands for the goals scored by the home team. Here are the first few rows:

```
## # A tibble: 6 x 4
##   home    away                   home_goals away_goals
##   <chr>   <chr>                       <int>      <int>
## 1 Arsenal Aston Villa                     3          2
## 2 Arsenal Bournemouth                     1          0
## 3 Arsenal Brighton & Hove Albion          1          2
## 4 Arsenal Burnley                         2          1
## 5 Arsenal Chelsea                         1          2
## 6 Arsenal Crystal Palace                  2          2
```

(g) Add two columns called `home_pts` and `away_pts`. The winning team (i.e. the team that scored more goals) in a game earns 3 points, the losing team 0 points. If the result is a draw, both teams get 1 point each. Here are the first few rows:

```
## # A tibble: 6 x 6
##   home    away                   home_goals away_goals home_pts away_pts
##   <chr>   <chr>                       <int>      <int>    <int>    <int>
## 1 Arsenal Aston Villa                     3          2        3        0
## 2 Arsenal Bournemouth                     1          0        3        0
## 3 Arsenal Brighton & Hove Albion          1          2        0        3
## 4 Arsenal Burnley                         2          1        3        0
## 5 Arsenal Chelsea                         1          2        0        3
## 6 Arsenal Crystal Palace                  2          2        1        1
```

(h) Pivot `epl` into longer format so that the team name is in the first column called `team`, and the second column `home_or_away` shows whether the team played at home or away. Here are the first few rows:

```
## # A tibble: 6 x 6
##   team                  home_or_away home_goals away_goals home_pts away_pts
##   <chr>                 <chr>             <int>      <int>    <int>    <int>
## 1 Arsenal               home                  3          2        3        0
## 2 Aston Villa           away                  3          2        3        0
## 3 Arsenal               home                  1          0        3        0
## 4 Bournemouth           away                  1          0        3        0
## 5 Arsenal               home                  1          2        0        3
## 6 Brighton & Hove Albion away                 1          2        0        3
```

(i) Change `epl` such that it contains columns `team`, `pts`, `goals_for`, `goals_against` (i.e. remove the information about whether a team played at home or away). Here are the first few rows:

```
## # A tibble: 6 x 4
##   team                     pts goals_for goals_against
##   <chr>                  <int>     <int>         <int>
## 1 Arsenal                    3         3             2
## 2 Aston Villa                0         2             3
## 3 Arsenal                    3         1             0
## 4 Bournemouth                0         0             1
## 5 Arsenal                    0         1             2
## 6 Brighton & Hove Albion     3         2             1
```

(j) Replace `pts`, `goals_for`, and `goals_against` by their cumulative values for each team and add a column called `goal_diff` for the goal difference (i.e. `goals_for` minus `goals_against`). Here are the first few rows:

```
## # A tibble: 6 x 4
##   team                     pts goal_diff goals_for
##   <chr>                  <int>     <int>     <int>
## 1 Arsenal                   56         8        56
## 2 Aston Villa               35       -26        41
## 3 Bournemouth               34       -25        40
## 4 Brighton & Hove Albion    41       -15        39
## 5 Burnley                   54        -7        43
## 6 Chelsea                   66        15        69
```

(k) Sort `epl` to obtain the league table shown at the start of this problem. Ensure that ties in the points are handled correctly.

6 String handling: corporate ipsum

An ipsum is a placeholder text commonly used to demonstrate the visual form of a document without relying on meaningful content. The text file `corporate_ipsum.txt` on Canvas contains automatically generated text for the corporate world.[1]

(a) Download the file to your working directory. Import the text as a character vector called `ipsum`. Each element in `ipsum` should correspond to one word.
(b) Remove punctuation symbols at the start and end of all elements in `ipsum`.
(c) Turn all characters in `ipsum` into lower case characters.
(d) Remove all stop words from `ipsum`. Use the list of stopwords returned by `tm::stopwords()`.
(e) Remove empty strings.

[1]Data from https://www.cipsum.com/.

(f) Make a word cloud with all (non-stop) words in `ipsum` that appear at least 20 times.

(g) Use **dplyr** functions to identify the 10 most frequent (non-stop) words in `ipsum`. Sort these 10 words in descending order of frequency. Also include their frequencies in the output tibble.

7 Choropleth map: automated teller machines per 100,000 adults

In the following exercises, we want to make a choropleth map that shows the number of Automated Teller Machines (ATMs) per 100,000 adults (figure 2).

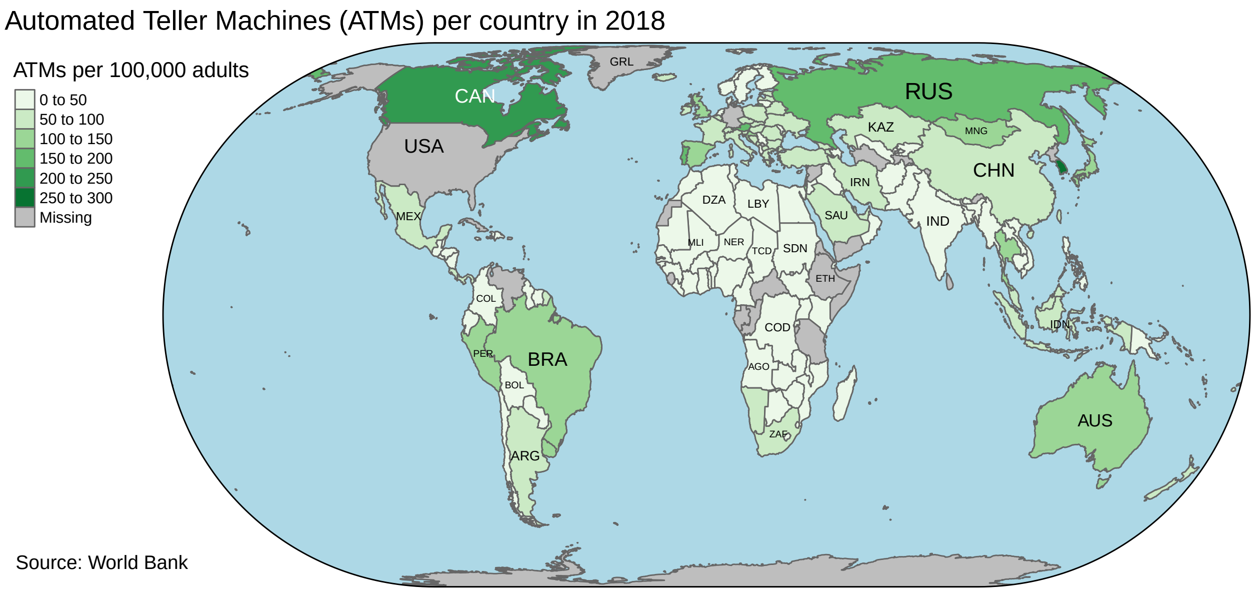

Figure 2: Choropleth map.

(a) Download the file `API_FB.ATM.TOTL.P5_DS2_en_excel_v2_3932325.xls`[2] from Canvas. Import the data into R as a tibble called `atm`.

(b) Only keep columns in `atm` for:

- country name.
- country code.
- the year 2018.

(c) Show that there are 47 countries with missing values for the number of ATMs per 100,000.

(d) Why is the number of ATMs per 100,000 adults a suitable variable for a choropleth map?

(e) Download the following files from Canvas:

- `wb_countries_admin0_10m.zip` (country borders)
- `wb_land_10m.zip` (land masses)

Import their data as `sf` objects called `country_borders` and `land`.

(f) Norway and France have invalid country codes in the column `ISO_A3` of `country_borders`. Change them to `NOR` and `FRA` respectively.

(g) Merge `country_borders` and `atm` so that `atm` becomes an `sf` object with:

- three-letter country code.
- a column for the number of ATMs per 100,000 adults per country.

[2]Source: https://data.worldbank.org/indicator/FB.ATM.TOTL.P5. Accessed on 26 April 2022.

- the geospatial boundaries of the corresponding country.

(h) Make a choropleth map with **tmap**. The map should have the following properties:

- Use the land masses as the map's bottom layer, the countries as the top layer.
- The map projection is Eckert IV.
- Colours are a sequential ColorBrewer palette. (I leave it up to you which palette you choose.)
- Choose matching shades of grey for the bottom layer and for missing values in the top layer.
- Label the largest countries with their country code.
- Show the earth's boundary as a black curve around the globe.
- Fill the globe with a light blue background colour, but keep the background outside the globe white.
- Add a main title and a legend title.
- Credit the World Bank as source of these data.
- Place the legend close to the map without obstructing any part of the globe.
- Adjust the figure dimensions with the R Markdown options `fig.width` and, if necessary, `out.width` so that all map elements (polygons, legends and labels) are easy to read without appearing disproportionately large.

8 COVID-19 in Mainland China and Singapore (lubridate and ggplot2)

At the end of 2019, an unusually large number of pneumonia cases in the Chinese city Wuhan raised concern about a new infectious respiratory disease. The disease was given the name COVID-19 and quickly developed into a pandemic. Johns Hopkins University has published daily observations of COVID-19 cases since 22 January 2020. The spreadsheet `covid_19_data.csv` on Canvas is a collection of these data.[3] The column `Confirmed` is the cumulative number of confirmed cases until the corresponding date.[4] The objective of this exercise is to compare the increase in COVID-19 cases in Mainland China and Singapore.

(a) Import the data as a tibble called `covid_19` and keep only those rows in which `Country/Region` is either `"Mainland China"` or `"Singapore"`.

(b) Use functions from the **lubridate** package to convert the column `ObservationDate` into a `Date` column.

(c) Summarise the total number of confirmed cases in `covid_19` by `ObservationDate` and `Country/Region`.

(d) Plot the time series of COVID-19 cases (figure 3). Here is a checklist:

- Give meaningful titles to the plot and the axes.
- Give credit to Johns Hopkins University as source of the data.
- You must adjust figure dimensions with R Markdown chunk options (e.g. `fig.width`) so that the labels are easily legible without appearing disproportionately large.

(e) Use tidyverse functions to find the first dates when the number of cases exceeded 10 000 in Mainland China and Singapore, respectively.

(f) Find the number of days between

- the date when the number of cases exceeded 10 000 in Mainland China and
- the date when the number of cases exceeded 10 000 in Singapore.

(g) How many days elapsed between $> 10\,000$ and $> 20\,000$ cases in Mainland China? How many days elapsed in Singapore? Use R to find the answers.

[3]Data downloaded from https://www.kaggle.com/sudalairajkumar/novel-corona-virus-2019-dataset, accessed on 2021-Apr-25.

[4]There are dates when `Confirmed` decreases, which is impossible if these are the *cumulative* cases. For the sake of simplicity, let us ignore these inconsistencies.

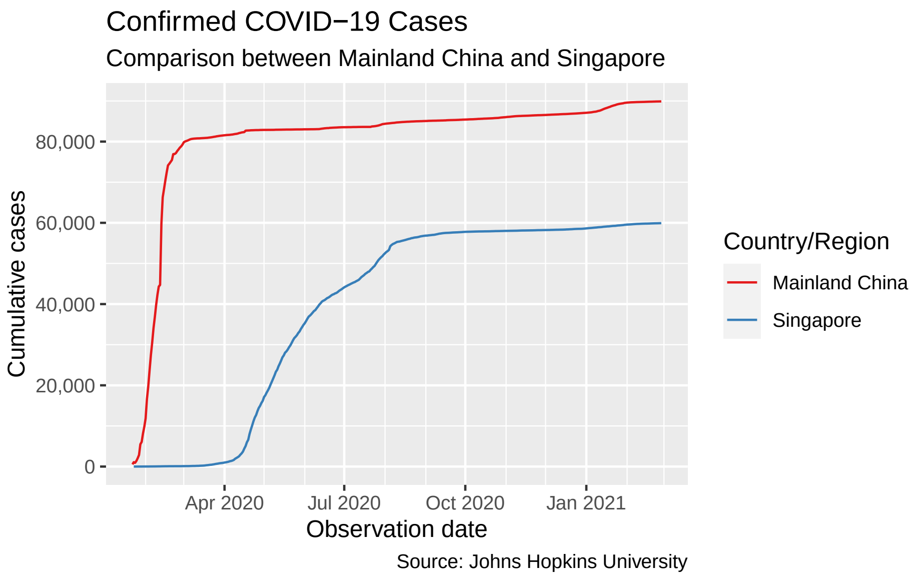

Figure 3: Time series plot of cumulative COVID-19 cases.