# Problem Solving Task 5, MXB262 2020

10/05/2020

## Problem solving task 5, for submission

Problem solving task 5 is due Sunday 17th May at midnight, and is worth 5% of your final grade.

You must upload an .Rmd file with your answers below plus a knitted version (preferably pdf, but html is fine if your computer doesn't like pdf). P

Note that you will be graded on the effective data visualisation choices you make in addition to your technical ability to produce the plots (i.e.: make it nice, and an effective communication tool).

## TASK

### Question 1.

Create two maps using the data contained in `rnaturalearth`. One must have a map extent that is smaller than the entire world (e.g. only shows one continent, or is bounded in some other way). Both must have polygons that are colours according to some aspect of the data. In the workshop you created the entire world coloured by the square root of the total population, so you cannot use population data for this question.

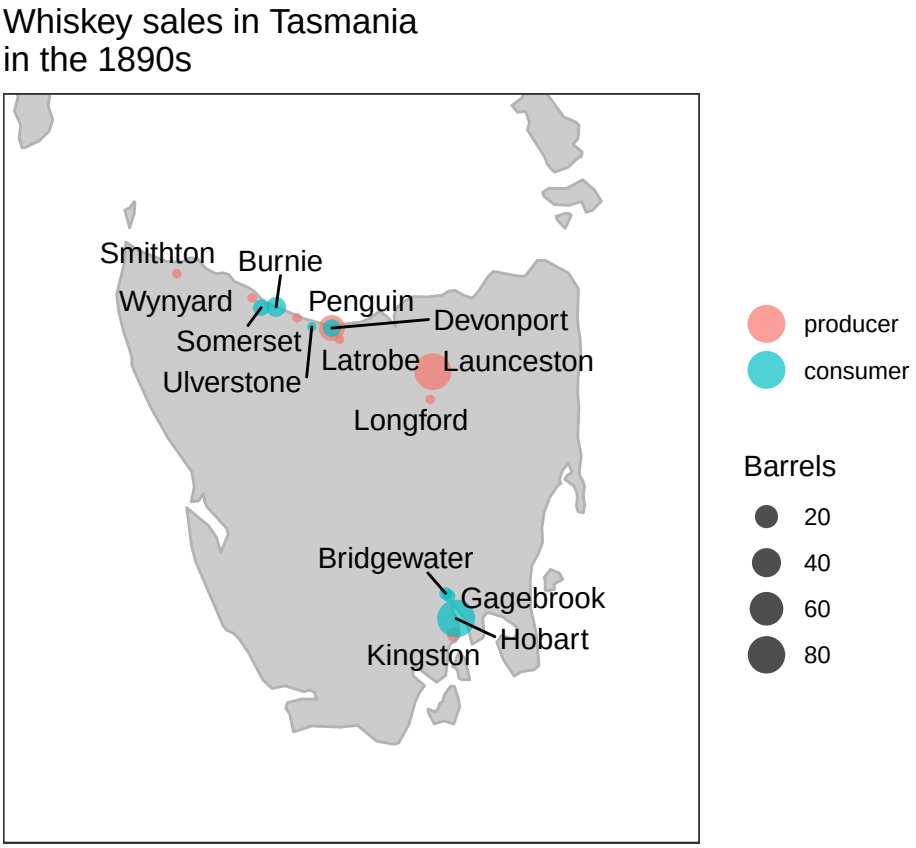
For each of your plots, represent that same data using at least one other plot type that we learned earlier in the unit, very briefly (one sentence) identify a target audience and an intended message for your visualisation.

### Question 2.

Reproduce the following map using the locations and transport datasets provided to you in the workshop files on Blackboard. In the dataset, one line represents the sale of one barrel of whiskey (this is made up data!).

```
library(tidyverse)
library(sp)
library(sf)
library(rnaturalearth)
library(ggrepel)
```

```
sales <- read_csv("data/whiskey_sales_tasmania.csv")
locations <- read_csv("data/tas_locations.csv")
```




This is a difficult question, for which you'll need to draw from your past experiences in this unit wrangling data and changing elements of the `ggplot` outputs. If you cannot get all the way to the final figure, for partial marks you can submit the closest figure you can get to and briefly explain: (1) the roadblock you could not pass and (2) how you tried to get around it.

## TIPS - Some Essential Data Wrangling

The following code will be very useful to you when trying to extract the relevant information from the whiskey dataset. This is actually a relly cool and useful technique when dealing with data in R, so you may find it useful throughout your other work.

First, I'll just create data of flights of 3 staff members of a large organisation. Don't worry too much about how this is done (the tibble, tribble part), you've already got the input data for this problem

```
flights <- tibble::tribble(
   ~departure, ~staff_ID, ~destination,
     "London",      734,      "Paris",
      "Paris",      734,  "Barcelona",
      "Paris",      734,  "Barcelona",
      "Paris",      734,     "Havana",
      "Paris",      734,  "Barcelona",
      "Paris",      734,     "Havana",
      "Paris",      734,     "Havana",
      "Paris",      734,     "Havana",
      "Paris",      735,  "Barcelona",
      "Paris",      734,     "Havana",
      "Paris",      734,     "Havana",
      "Paris",      734,     "Havana",
```

```
      "Paris",      734,     "Havana",
      "Paris",      738,     "Havana",
      "Paris",      734,     "Havana",
      "Paris",      734,     "Havana",
  "Barcelona",      735,     "Madrid",
      "Paris",      734,     "London",
  "Barcelona",      735,     "London",
      "Paris",      734,     "London",
      "Paris",      735,     "London",
      "Paris",      738,     "London",
      "Paris",      734,     "London",
      "Paris",      735,     "London",
      "Paris",      735,     "London",
      "Paris",      735,     "London",
      "Paris",      735,     "London",
      "Paris",      735,     "London",
      "Paris",      734,     "London",
  "Barcelona",      734,     "London",
      "Paris",      734,     "London",
  "Barcelona",      735,     "London",
      "Paris",      734,     "London",
      "Paris",      735,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
  "Barcelona",      738,     "London",
   "Kingston",      738,  "Barcelona",
      "Paris",      734,     "London",
      "Paris",      738,  "Barcelona",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
  "Barcelona",      738,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London",
      "Paris",      738,     "London",
      "Paris",      734,     "London",
      "Paris",      734,     "London"
  )
```

Now, it is possible that instead of wanting to know about every single flight, we instead want to count the number of flights coming to and from each city:

```
# Count the number of departures per city
departures <- flights %>%
  group_by(departure) %>%
  count() %>%
  rename(city = departure) %>%
  add_column(direction = "departure") %>%
  ungroup()

# Count the number of arrival flights per city
```

```
destinations <- flights %>%
  group_by(destination) %>%
  count() %>%
  rename(city = destination) %>%
  add_column(direction = "arrivals") %>%
  ungroup()
```

There are two ways to combine two datasets like the one here. First, we can combine multiple rows by using `rbind()`, which keeps the same column headings. Here we can use that to combine our departure and destination counts into one dataset, and specify that the `direction` column is an important sorting factor for our data:

```
travel_counts <- rbind(departures, destinations) %>% # Bind the rows of the two data frames
  mutate(direction = as_factor(direction)) # and change direction column to factor
```

And we can also combine to datasets by column, as long as they have one column in common. Explore this for yourself by typing `?left_join` into your R console to get to the help menu. This will be useful for you because although we have provided two different datasets, you need them all in one in order to plot them.