HW 01 - Airbnb listings in Edinburgh

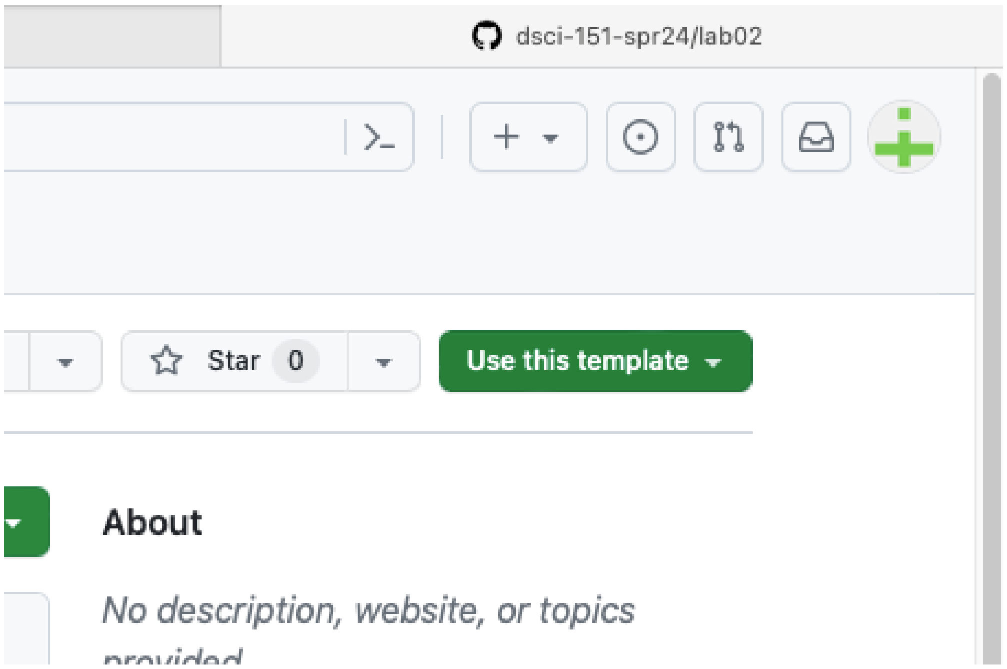
Figure 1: Photo by Madeleine Kohler on Unsplash

Once upon a time, people travelled all over the world, and some stayed in hotels and others chose to stay in other people's houses that they booked through Airbnb. Recent developments in Edinburgh regarding the growth of Airbnb and its impact on the housing market means a better understanding of the Airbnb listings is needed. Using data provided by Airbnb, we can explore how Airbnb availability and prices vary by neighbourhood.

Getting started

Go to Homework

Then make your own copy of this by clicking on `Use this template`:



Grab the URL of the repo, and clone it in RStudio. Refer to Lab 01 if you would like to see step-by-step instructions for cloning a repo into an RStudio project.

First, open the R Markdown document `hw-01-airbnb-edi.Rmd` and Knit it. Make sure it compiles without errors. The output will be in the file markdown `.md` file with the same name.

Warm up

Before we introduce the data, let's warm up with some simple exercises.

- Update the YAML, changing the author name to your name, and **knit** the document.
- Commit your changes with a meaningful commit message.
- Push your changes to GitHub.
- Go to your repo on GitHub and confirm that your changes are visible in your Rmd **and** md files. If anything is missing, commit and push again.

Packages

We'll use the **tidyverse** package for much of the data wrangling and visualisation and the data lives in the **dsbox** package. These packages are already installed for you. You can load them by running the following in your Console:

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
library(dsbox)
data(edibnb)
```

Data

The data can be found in the **dsbox** package, and it's called `edibnb`. Since the dataset is distributed with the package, we don't need to load it separately; it becomes available to us when we load the package.

You can view the dataset as a spreadsheet using the `View()` function. Note that you should not put this function in your R Markdown document, but instead type it directly in the Console, as it pops open a new window (and the concept of popping open a window in a static document doesn't really make sense…). When you run this in the console, you'll see the following **data viewer** window pop up.

```r
View(edibnb)
```

You can find out more about the dataset by inspecting its documentation, which you can access by running `?edibnb` in the Console or using the Help menu in RStudio to search for `edibnb`. You can also find this information here.

Exercises

```
**Hint:** The Markdown Quick Reference sheet has an example of inline R code that might be helpful. You
```

1. How many observations (rows) does the dataset have? Instead of hard coding the number in your answer, use inline code.

There are 13245 rows in the dataset.

2. Run `View(edibnb)` in your Console to view the data in the data viewer. What does each row in the dataset represent?

Each row in the dataset represents an airbnb listing in Edinburgh.

Knit, commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.

Each column represents a variable. We can get a list of the variables in the data frame using the `names()` function.

```r
names(edibnb)
```

```
##  [1] "id"                   "price"                "neighbourhood"
##  [4] "accommodates"         "bathrooms"            "bedrooms"
##  [7] "beds"                 "review_scores_rating" "number_of_reviews"
## [10] "listing_url"
```

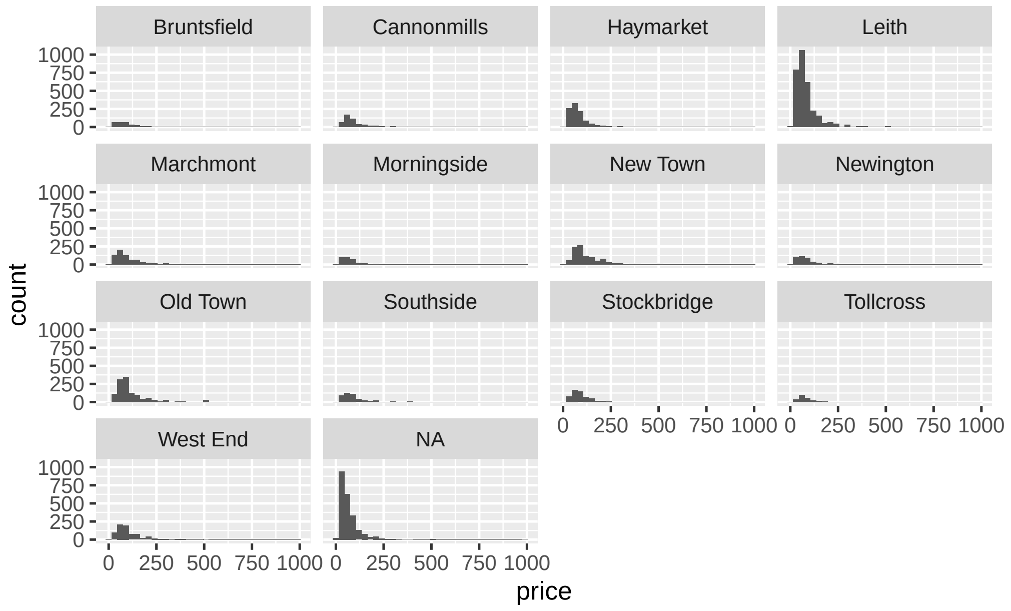
You can find descriptions of each of the variables in the help file for the dataset, which you can access by running `?edibnb` in your Console.

```
**Note:** The plot will give a warning about some observations with non-finite values for price being r
```

3. Create a faceted histogram where each facet represents a neighbourhood and displays the distribution of Airbnb prices in that neighbourhood. Think critically about whether it makes more sense to stack the facets on top of each other in a column, lay them out in a row, or wrap them around. Along with your visualisation, include your reasoning for the layout you chose for your facets.

```r
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 30) +
  facet_wrap(~neighbourhood)
```

```
## Warning: Removed 199 rows containing non-finite outside the scale range
## (`stat_bin()`).
```



The data makes more sense to wrap rather than stacked or all in a row as there are too many neighbohoods to see the data clearly in a row. The same goes for a single column. There data would be too condensed to see the trends.

Let's de-construct this code:

- `ggplot()` is the function we are using to build our plot, in layers.
- In the first layer we always define the data frame as the first argument. Then, we define the mappings between the variables in the dataset and the **aes**thetics of the plot (e.g. x and y coordinates, colours, etc.).
- In the next layer we represent the data with **geom**etric shapes, in this case with a histogram. You should decide what makes a reasonable bin width for the histogram by trying out a few options.
- In the final layer we facet the data by neighbourhood.

```
ggplot(data = edibnb, aes(x = price)) +
  geom_histogram(binwidth = 20)
```

Knit, *commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards.*

4. Use a single pipeline to identity the neighbourhoods with the top five median listing prices. Then, in another pipeline filter the data for these five neighbourhoods and make ridge plots of the distributions of listing prices in these five neighbourhoods. In a third pipeline calculate the minimum, mean, median, standard deviation, IQR, and maximum listing price in each of these neighbourhoods. Use the visualisation and the summary statistics to describe the distribution of listing prices in the neighbourhoods. (Your answer will include three pipelines, one of which ends in a visualisation, and a narrative.

create a table

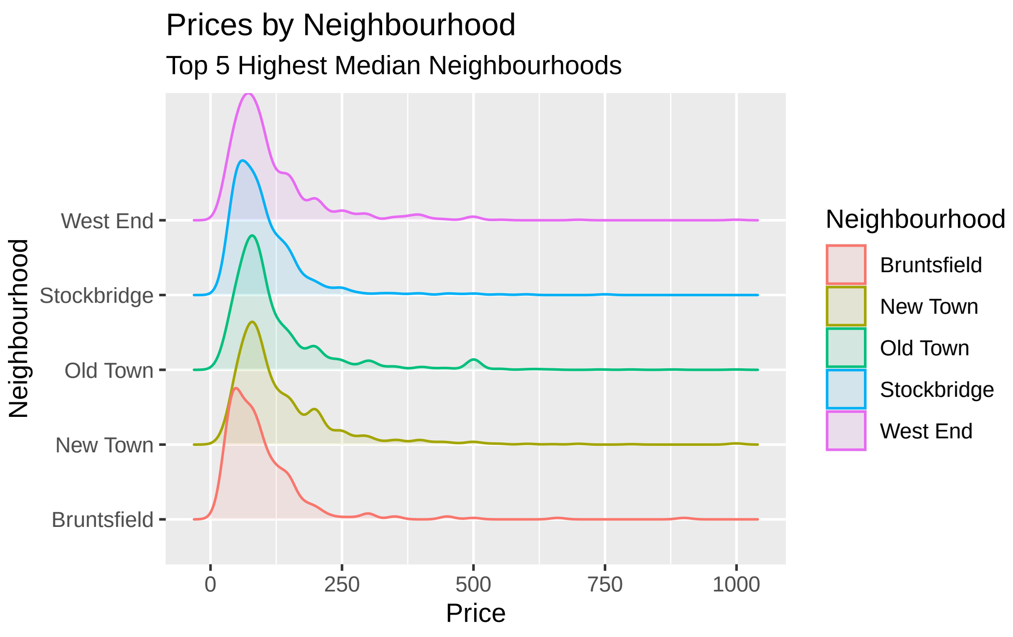
```
edibnb %>%
  group_by(neighbourhood) %>%
  summarise(median(price, na.rm = TRUE))
```

```
## # A tibble: 14 x 2
##    neighbourhood `median(price, na.rm = TRUE)`
##    <chr>                             <dbl>
##  1 Bruntsfield                          80
##  2 Cannonmills                          79
##  3 Haymarket                            68
##  4 Leith                                66
##  5 Marchmont                            80
##  6 Morningside                          67
##  7 New Town                            100
##  8 Newington                            75
##  9 Old Town                             90
## 10 Southside                            80
## 11 Stockbridge                          85
## 12 Tollcross                            75
## 13 West End                             90
## 14 <NA>                                 50
```

```
#New Town = 100, West End = 90, Old Town = 90, Stockbridge = 85
#Bruntsfield, Marchmont, and Southside = 80

edibnb_top_5 <- edibnb %>%
  filter(neighbourhood == "New Town" | neighbourhood == "West End" | neighbourhood == "Old Town" | neig

library(ggridges)

ggplot(data = edibnb_top_5, mapping = aes(x = price, y = neighbourhood, color = neighbourhood, fill = ne
  geom_density_ridges(alpha = 0.10) +
  labs(
    title = "Prices by Neighbourhood", subtitle = "Top 5 Highest Median Neighbourhoods",
    x = "Price",
    y = "Neighbourhood", color = "Neighbourhood", fill = "Neighbourhood")
```

```
## Picking joint bandwidth of 13.8
```

```
## Warning: Removed 104 rows containing non-finite outside the scale range
## (`stat_density_ridges()`).
```



```r
edibnb_top_5 %>%
  group_by(neighbourhood) %>%
  summarise(min(price, na.rm = TRUE),
            median(price, na.rm = TRUE),
            max(price, na.rm = TRUE),
            IQR(price, na.rm = TRUE),
            mean(price, na.rm = TRUE),
            sd(price, na.rm = TRUE))
```

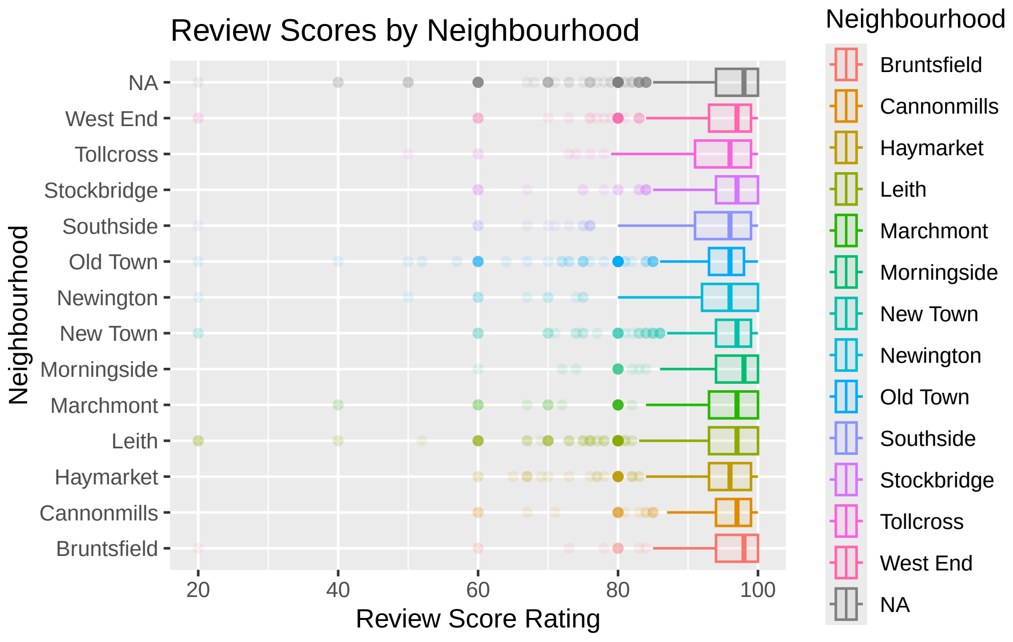
```
## # A tibble: 5 x 7
##   neighbourhood `min(price, na.rm = TRUE)` `median(price, na.rm = TRUE)`
##   <chr>                              <dbl>                         <dbl>
## 1 Bruntsfield                           10                            80
## 2 New Town                              12                           100
## 3 Old Town                              15                            90
## 4 Stockbridge                           21                            85
## 5 West End                              19                            90
## # i 4 more variables: `max(price, na.rm = TRUE)` <dbl>,
## #   `IQR(price, na.rm = TRUE)` <dbl>, `mean(price, na.rm = TRUE)` <dbl>,
## #   `sd(price, na.rm = TRUE)` <dbl>
```

The top 5 highest priced neighbourhoods (West End, Stockbridge, Old Town, New Town and Bruntsfield [technially Bruntsfield, Marchmont, and Southside are all tied for 5th]) have means ranging from 99 - 136. They are skewed to the high end with most having a maximum of at least 900, however, Stockbridge only has a maximum of 750. Bruntsfield has the lowest minimum pricing at 10. The minimums ranged from 10-19. The IQRs were fairly consistent ranging from 73 (Bruntsfield) to 86 (New Town). The standard deviations ranged from 90 (Bruntsfield) to 110 (Old Town). From the visualization we can see that Stockbridge has the most centralized or evenly distributed data and utimately the most consistent pricing. For each neighborhood, the data is unimodal with a prominent peak.

5. Create a visualization that will help you compare the distribution of review scores (`review_scores_rating`) across neighbourhoods. You get to decide what type of visualisation to create and there is more than one correct answer! In your answer, include a brief interpretation of how Airbnb guests rate properties in general and how the neighbourhoods compare to each other in terms of their ratings.

```r
ggplot(data = edibnb, mapping = aes(x = review_scores_rating, y = neighbourhood, color = neighbourhood,
  geom_boxplot(alpha = 0.10) +
  labs(
    title = "Review Scores by Neighbourhood",
    x = "Review Score Rating",
    y = "Neighbourhood", color = "Neighbourhood", fill = "Neighbourhood"
    )
```

```
## Warning: Removed 2177 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



Overall, most neighbourhoods in are rated very well by Airbnb customers with most of the data being above 90. Some neighbourhoods have a few extremely low scores such as West End, Old Tow, Southside, Newington, New Town, Leith, and Bruntsfield. On average, however most neighbourhoods are rated about 95. The lowest and greatst IQR neighbourhoods however appear to be Tollcross, Southside, and Newington. The best rated neighbourhood (by mean) apears to be Morningside.

Knit, *commit, and push your changes to GitHub with an appropriate commit message. Make sure to commit and push all changed files so that your Git pane is cleared up afterwards and review the md document on GitHub to make sure you're happy with the final state of your work.*

Save a pdf version and submit to moodle