HW 02 - Airbnb listings in Edinburgh

Max Asmar

Figure 1: Photo by Madeleine Kohler on Unsplash

Once upon a time, people traveled 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 neighborhood.

Getting started

Learning Objectives

In this assignment, you will practice:

- Creating visualizations with `ggplot2`
- Using faceting to compare distributions across groups
- Working with pipelines to filter and summarize data
- Interpreting distributions using visualizations and summary statistics
- Choosing appropriate visualization types for different questions

---

Navigate to your `homework-instructions` repo in JupyterHub and open the `hw-02-airbnb-edi.Rmd` file. Knit the document to make sure it compiles without errors.

Warm up

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

Complete these steps:

1. Update the YAML, changing the author name to your name
2. **Knit** the document
3. **Commit** your changes with message: `"Updated author name"`
4. **Push** to GitHub
5. Verify your changes are visible in your GitHub repo (check both .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 visualization 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:

```
library(tidyverse)
library(dsbox)
```

Note: If you get an error that dsbox is not installed, run this in your Console first:

```
install.packages("devtools")
devtools::install_github("tidyverse/dsbox")
```

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.

```
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 can access it from the Help menu in RStudio.

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

   There are 13245 observations in the dataset.

   Hint: In your answer, type a backtick, then r, then space, then `nrow(edibnb)`, then close with another backtick. When you knit, this will show the actual number instead of the code.

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

Each observation in the dataset represents an airbnb in Edinburgh and contains information about its location, price, and features as variables.

After completing Exercises 1-2:

Knit → Commit with message `"Completed Exercises 1-2"` → Push

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

```
names(edibnb)
```

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.

Before you create your first plot, let's understand the structure of ggplot2 code. Here's the basic template:

```
ggplot(data = dataset_name, mapping = aes(x = variable)) +
  geom_histogram(binwidth = 50) +
  facet_wrap(~grouping_variable)
```

Breaking it down:

- `ggplot()` starts the plot and defines the data and aesthetic mappings
- `geom_histogram()` adds the geometric layer (histogram bars)
- `facet_wrap()` creates separate plots for each group
- Each layer is connected with `+`

---

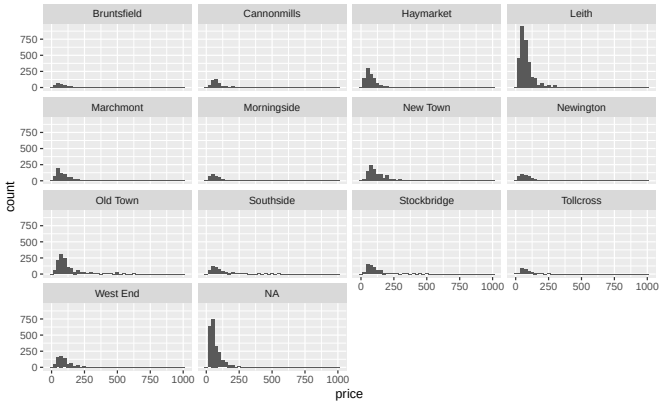
3. Create a faceted histogram where each facet represents a neighborhood and displays the distribution of Airbnb prices in that neighborhood.

a) Experiment with binwidth

Try binwidths of 25, 50, and 100. Which gives you the most useful view of the data?

```
# Try binwidth = 25
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 25) +
  facet_wrap(~neighbourhood)
```

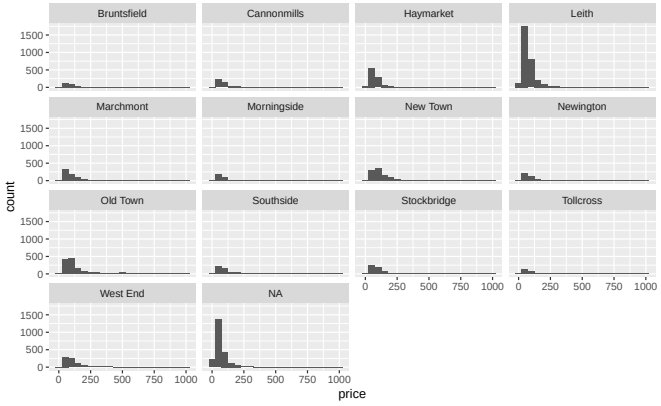
```
## Warning: Removed 199 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
# Try binwidth = 50
```

```
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 50) +
  facet_wrap(~neighbourhood)
```

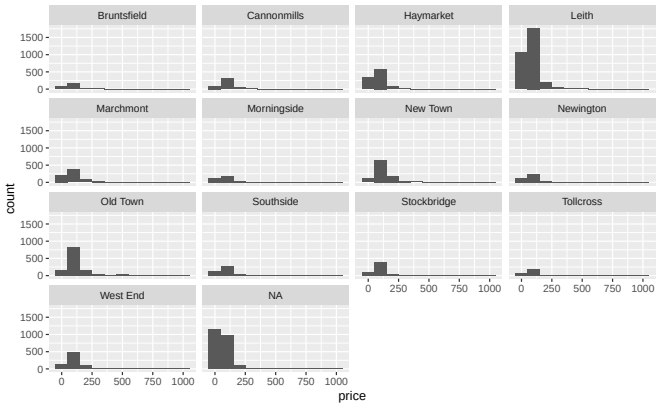
```
## Warning: Removed 199 rows containing non-finite outside the scale range
## (`stat_bin()`).
```




```
# Try binwidth = 100
```

```
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 100) +
  facet_wrap(~neighbourhood)
```

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



Which binwidth did you choose and why?

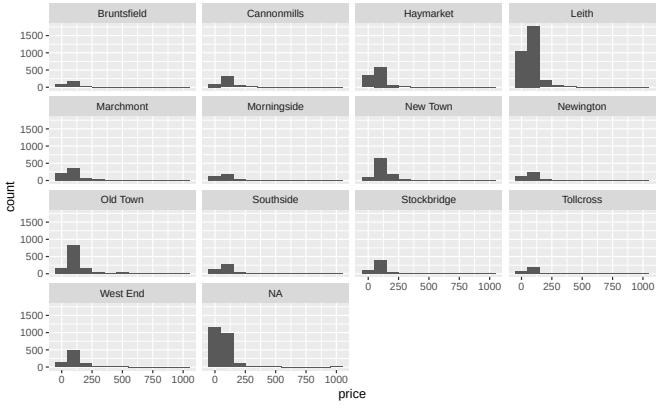
I chose the binwidth to be 100 because it allowed me to see most clearly the distribution of Airbnb prices, even though it sacrificed detail. There are 14 plots on the overall grid, which makes it harder to view each one individually. I addressed this by increasing binwidth size.

b) Experiment with facet layout*

Try these three options:

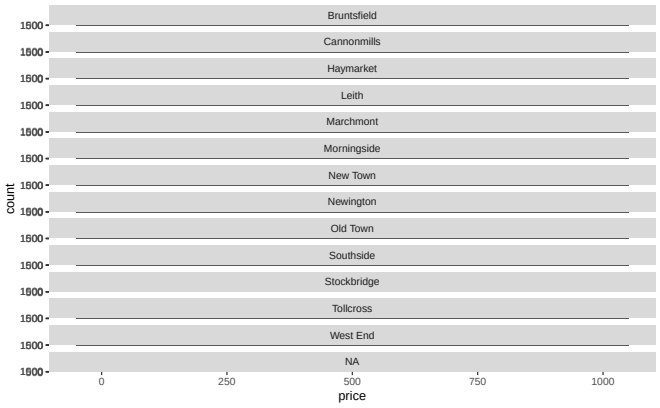
```
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 100) +
  facet_wrap(~neighbourhood)
```

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



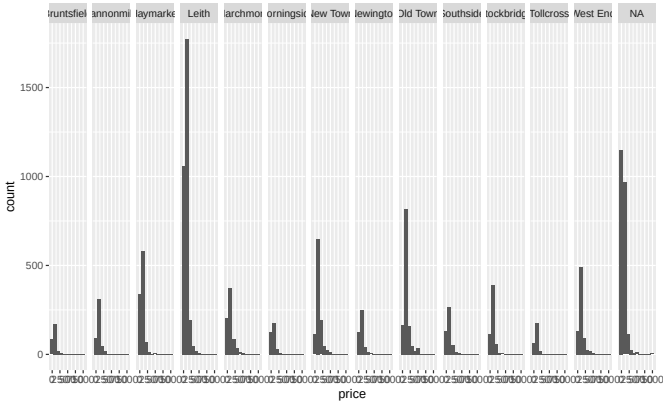

```
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 100) +
  facet_wrap(~neighbourhood, ncol = 1)
```

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




```
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 100) +
  facet_wrap(~neighbourhood, nrow = 1)
```

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



- `facet_wrap(~neighbourhood)` - wraps into a grid
- `facet_wrap(~neighbourhood, ncol = 1)` - stacks vertically
- `facet_wrap(~neighbourhood, nrow = 1)` - arranges horizontally

Which layout did you choose and why?

I chose the facet that wraps the neighborhood into a grid. I chose this because the vertical stack was not very intuitive to read and the horizontal stack was too cluttered for interpretation, for my personal preference.

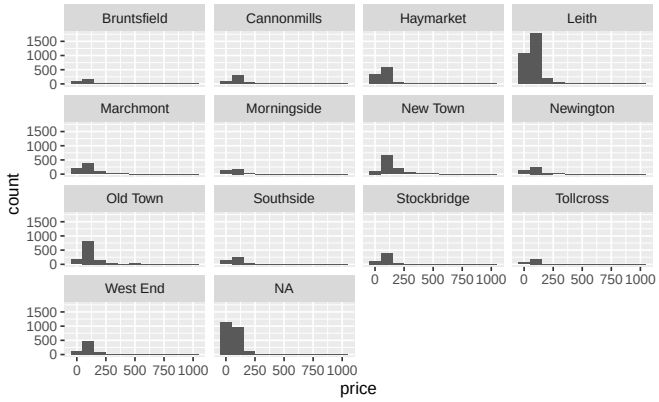
Note: The plot will give a warning about some observations with non-finite values for price being removed. Don't worry about the warning, it simply means that 199 listings in the data didn't have prices available, so they can't be plotted.

c) Create your final visualization

Use your chosen binwidth and layout:

```
ggplot(data = edibnb, mapping = aes(x = price)) +
  geom_histogram(binwidth = 100) +
  facet_wrap(~neighbourhood)
```

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



After completing Exercise 3:

Knit → Commit with message `"Completed Exercise 3"` → Push

4. This question has three parts that work together to explore the top 5 most expensive neighborhoods.

a) Identify the top 5 neighborhoods by median price. Complete the pipeline to find neighborhoods with the highest median prices:

```
top_5 <- edibnb %>%
  group_by(neighbourhood) %>%
  summarise(median_price = median(price, na.rm = TRUE)) %>%
  arrange(desc(median_price)) %>%
  slice_max(median_price, n = 5)
```

```
top_5
```

```
## # A tibble: 7 x 2
##   neighbourhood median_price
##   <chr>                <dbl>
## 1 New Town               100
## 2 Old Town                90
## 3 West End                90
## 4 Stockbridge             85
## 5 Bruntsfield             80
## 6 Marchmont               80
## 7 Southside               80
```

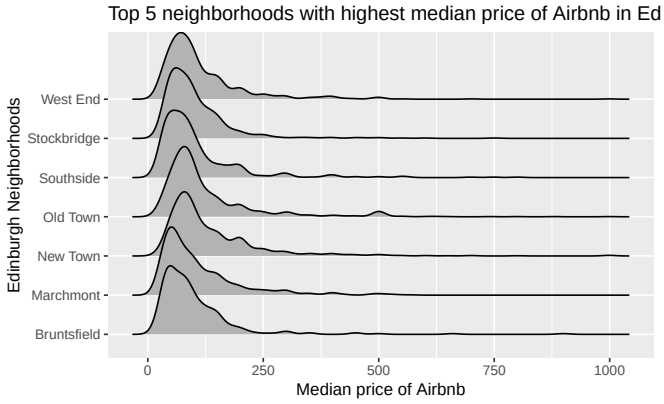
b) Create ridge plots for these neighborhoods.

Ridge plots show distributions stacked vertically, making them easy to compare.

```
library(ggridges)

# Filter for top 5 neighborhoods and create ridge plot
edibnb %>%
  filter(neighbourhood %in% top_5$neighbourhood) %>%
  ggplot(aes(x = price, y = neighbourhood)) +
  geom_density_ridges() +
  labs(
    x = "Median price of Airbnb",
    y = "Edinburgh Neighborhoods",
    title = "Top 5 neighborhoods with highest median price of Airbnb in Edinburgh"
  )
```

```
## Warning: Removed 116 rows containing non-finite outside the scale range
## (`stat_density_ridges()`).
```



c) Calculate the summary statistics.

```
edibnb %>%
  filter(neighbourhood %in% top_5$neighbourhood) %>%
  group_by(neighbourhood) %>%
  summarise(
    min = min(price, na.rm = TRUE),
    mean = mean(price, na.rm = TRUE),
    median = median(price, na.rm = TRUE),
    sd = sd(price, na.rm = TRUE),
    IQR = IQR(price, na.rm = TRUE),
    max = max(price, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 7
##   neighbourhood   min  mean median    sd   IQR   max
##   <chr>         <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 Bruntsfield      10  99.4     80  90.2  72.5   900
```

```
## 2 Marchmont        10 111.      80  90.7  90     570
## 3 New Town         12 136.     100 109.   86.5   999
## 4 Old Town         15 128.      90 110.   76     999
## 5 Southside        10 107.      80  99.3  70     800
## 6 Stockbridge      21 104.      85  77.6  66     750
## 7 West End         19 116.      90  93.3  80     999
```

Using both the visualization and summary statistics, describe the distribution of listing prices in these neighborhoods. Consider:

- Which neighborhood has the highest typical prices?
- Which has the most variation (spread) in prices?
- Are there any notable patterns or outliers?

New Town has the highest typical prices of Airbnbs in Edinburgh because it has the highest mean and median prices of 136 and 100 GBP respectively. Old town has the most variation in prices because it has the highest standard deviation in the summary statistics (110). One outlier I noticed is that Marchmont has the lowest maximum price at 570 GBP, which is 27% lower than the next lowest maximum. On the ridge plot, all of the neighborhoods tend to have a similar distribution where there is a peak at around 125 GBP.

After completing Exercise 4:

Knit → Commit with message `"Completed Exercise 4"` → Push

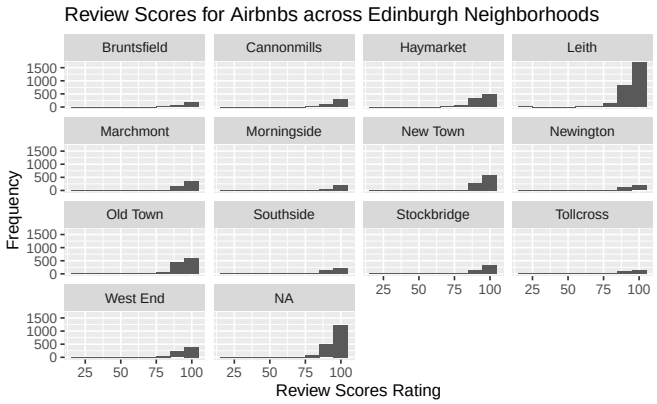
5. Create a visualization that will help you compare the distribution of review scores (`review_scores_rating`) across neighborhoods. You get to decide what type of visualization to create and there is more than one correct answer!

Suggested plot types to consider:

- Box plots: `geom_boxplot()`
- Violin plots: `geom_violin()`
- Faceted histograms: `geom_histogram()` + `facet_wrap()`
- Ridge plots: `geom_density_ridges()` (requires `library(ggridges)`)

```
edibnb %>%
  ggplot(aes(x = review_scores_rating)) +
  geom_histogram(binwidth=10) +
  facet_wrap(~neighbourhood) +
  labs(
    x = "Review Scores Rating",
    y = "Frequency",
    title = "Review Scores for Airbnbs across Edinburgh Neighborhoods"
  )
```

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



Interpretation (2-3 sentences):

- How do Airbnb guests rate properties in general (are ratings generally high, low, or varied)?
- How do the neighborhoods compare to each other?
- Are there any neighborhoods that stand out as having particularly high or low ratings?

In general, Airbnb guests rate properties highly, with most ratings falling in between 75-100. Leith and Old Town have the highest quantity of very positive ratings around 90-100. The Southside neighborhood has a smaller quantity of ratings but has a higher proportion of lower ratings.

After completing Exercise 5:

Knit → Commit with message `"Completed Exercise 5"` → Push

Common Errors and Troubleshooting

Visualization errors:

- **Error: "object not found"** → Make sure you loaded the tidyverse package with `library(tidyverse)`
- **Facets are too small to read** → Try different layouts with `ncol` or `nrow` arguments in `facet_wrap()`
- **Warning about non-finite values** → This is normal for this dataset (some prices are missing)
- **Error: "could not find function 'geom_density_ridges'"** → Load the ggridges package: `library(ggridges)`

Pipeline errors:

- **Error: "could not find function '%>%'"** → Load the tidyverse package
- **`group_by()` not working as expected** → Make sure you use `summarise()` after `group_by()`
- **Can't filter for top 5 neighborhoods** → Make sure you're using `top_5$neighbourhood` to reference the neighborhood names

Inline code issues:

- **Inline code shows as text** → Make sure you're using backticks around your R code (backtick-r-space-your-code-backtick)
- **Error when knitting** → Check that the variable/function you're referencing exists

General issues:

- **Plot not showing** → Make sure the code chunk doesn't have `eval = FALSE`
- **Changes not showing after knitting** → Save your file first (Ctrl+S or Cmd+S)

To submit:

1. Click the **Knit** dropdown menu (next to the Knit button)
2. Select **Knit to tufte_handout** to generate a PDF
3. Download the PDF file from your Files pane
4. Upload the PDF to Canvas

Final check: Visit your GitHub repo to confirm all your commits are there!