Data Cleaning and Exploration

Atiq Ur Rehman

2025-03-01

Introduction

Data cleaning and exploration are fundamental steps in any data analysis project. Raw data often contains inconsistencies such as **missing values, duplicate records, incorrect formats, and outliers**. Cleaning the data ensures its accuracy and reliability, while exploratory data analysis (EDA) helps us understand its structure, patterns, and relationships.

In this document, we will be working with a **real estate dataset** containing information on **properties sold in Canberra**. The goal of this analysis is to:

- Identify and handle **missing values**.
- Convert **data types** appropriately (e.g., dates, categorical variables).
- Filter and refine the dataset for meaningful insights.
- Explore key variables such as **property type, sale price, land size, and number of bedrooms**.
- Summarize and visualize trends using **ggplot2** and **dplyr**.

By the end of this process, we will have a **clean and structured dataset** ready for further analysis, modeling, and visualization.

Load Required Libraries

```
library(tidyverse)
library(lubridate)
library(tidygeocoder)
```

Note on data collection

We scraped data from the **AU House Prices** website. Since there were no stated restrictions on copying or sharing the data, we proceeded with collecting it. The dataset includes all

properties sold in Canberra since 1993. While having more historical data is beneficial, older records tend to have more missing values. Nonetheless, we still included them in our collection. The code for web scraping this data is in an R script named **version5.R**, which can be used to extract the data.

---

data loading

I divided the data into two parts. There is nothing special about this; I just wanted to download one part first and the other later. When web scraping, websites can sometimes block requests if they receive too many at once or over an extended period. To avoid this, I set a delay of 2 seconds between requests and divided the suburbs in Canberra into two groups. This allowed me to download some data first and then retrieve the remaining data separately.

```
# Load data from CSV files
sold_houses_remaining <- read.csv("../Data/sold_houses_Remaining.csv",
                                  header = TRUE)

sold_houses_franklin <- read.csv("../Data/sold_houses_till_Franklin.csv",
                                 header = TRUE)

# Combine both datasets into a single dataframe
canberra_sold_houses <- bind_rows(sold_houses_remaining,
                                  sold_houses_franklin)
```

Data cleaning

```
# Convert 'sold_date' column to Date format
canberra_sold_houses <- canberra_sold_houses %>%
  mutate(sold_date = dmy(sold_date))
```

Missing values

The dataset contains a total of 158,126 observations. However, some variables have missing values, as shown in the table below. The land size variable has the highest number of missing values, likely because it is not always recorded in property listings. Other variables, such as agency, sold date, garage, and baths, also have a significant number of missing values. In contrast, suburb and full address have no missing values.

```r
missing_values <- canberra_sold_houses %>%
    summarise(across(everything(),
                     ~ sum(is.na(.)))) %>%
    pivot_longer(cols = everything(),
                 names_to = "Variable",
                 values_to = "Missing_Count") %>%
    arrange(desc(Missing_Count))  # Sort by most missing values

knitr::kable(missing_values,
             caption = "Number of Missing Values per Variable")
```

Table 1: Number of Missing Values per Variable

Variable	Missing_Count
land_size	116280
agency	67087
sold_date	65969
garage	58083
baths	52309
beds	41688
price	11047
type_of_property	20
suburb	0
full_address	0

Removing all missing values

The remaining observations are

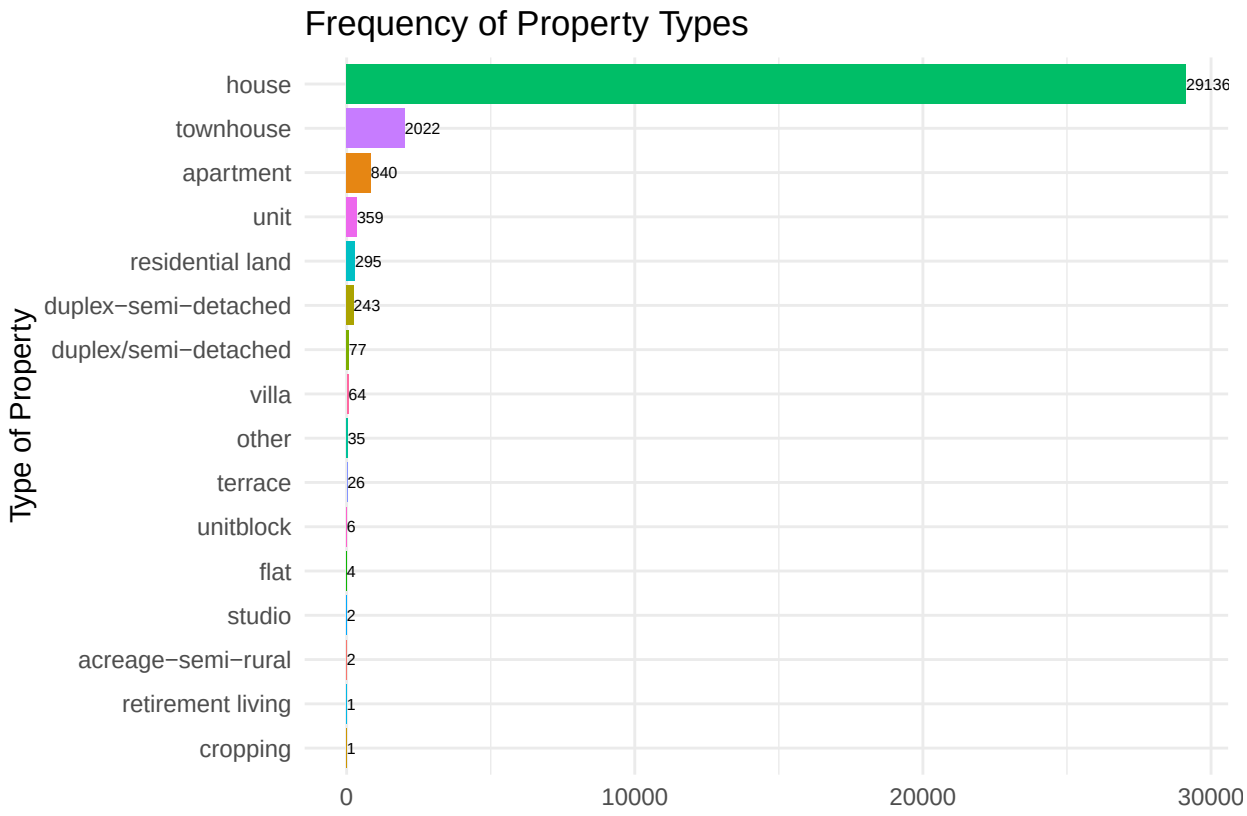
```r
canberra_sold_houses <-  canberra_sold_houses %>%
  drop_na()

canberra_sold_houses %>%
  summarise(total_samples = n())
```

```
##   total_samples
## 1         33113
```

Type of houses

In the given dataset, the **type_of_property** variable contains 16 distinct categories, some of which are closely related shown in the figure below. To streamline the data for analysis and improve clarity, we merged similar property types based on their characteristics and usage in the real estate market. The goal was to reduce redundancy while preserving meaningful distinctions between different property types.

```r
canberra_sold_houses %>%
  group_by(type_of_property) %>%
  summarise(frq_property = n()) %>%
  arrange(desc(frq_property)) %>%
  ggplot(aes(x = reorder(type_of_property, frq_property),
             y = frq_property,
             fill = type_of_property)) +
  geom_col() +
  geom_text(aes(label = frq_property),
            hjust = 0,  # Adjust position to avoid overlap
            size = 2) +  # Adjust text size
  coord_flip() +  # Flip for better readability
  labs(title = "Frequency of Property Types",
       x = "Type of Property",
       y = "") +
  theme_minimal() +
  theme(legend.position = "none")
```



1. Merging Similar Property Types

Several categories in the dataset referred to the same type of property but had variations in naming. For example:

- **duplex-semi-detached** and **duplex/semi-detached** were merged into **duplex-semi-detached** to maintain consistency in naming.
- **apartment**, **unit**, **flat**, and **studio** were grouped into a single category **apartment/unit** since these all represent small, self-contained dwellings in multi-unit buildings.
- **unitblock** was also merged into **apartment/unit**, assuming that individual units within a block were recorded separately.

2. Grouping Related Property Types

Some property types shared structural similarities and were logically grouped together:

- **townhouse** and **terrace** were combined into **townhouse/terrace**, as both are multi-story attached dwellings with similar living styles.
- **villa** could either be merged with **townhouse/terrace** or kept separate, depending on the need for finer granularity in analysis.

3. Categorizing Land and Niche Property Types

Properties related to **land usage** were also consolidated:

- **cropping** and **acreage-semi-rural** were merged into **rural land**, as they represent large land areas used for agriculture or semi-rural living.
- **residential land** was kept as a separate category since it represents land intended for housing development rather than agriculture.

Lastly, property types such as **retirement living** and **other** need to be grouped under a single category, **Other**, for completeness. However, since these property types as well as **rural land** are rare and do not have a significant market presence in Canberra, they are need to be removed from the dataset.

Final Categorization

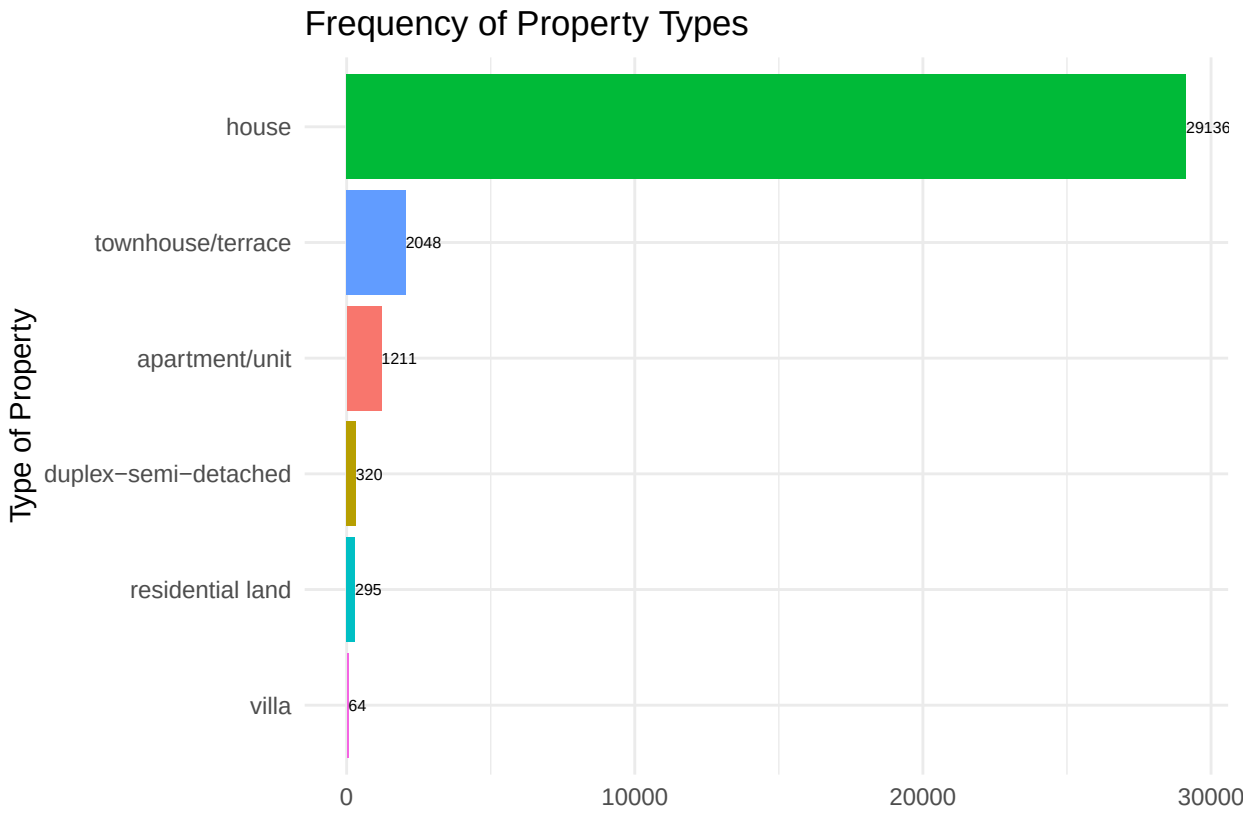
The following table summarizes the original categories and their respective merged groups:

Table 2: Final Property Type Categorization

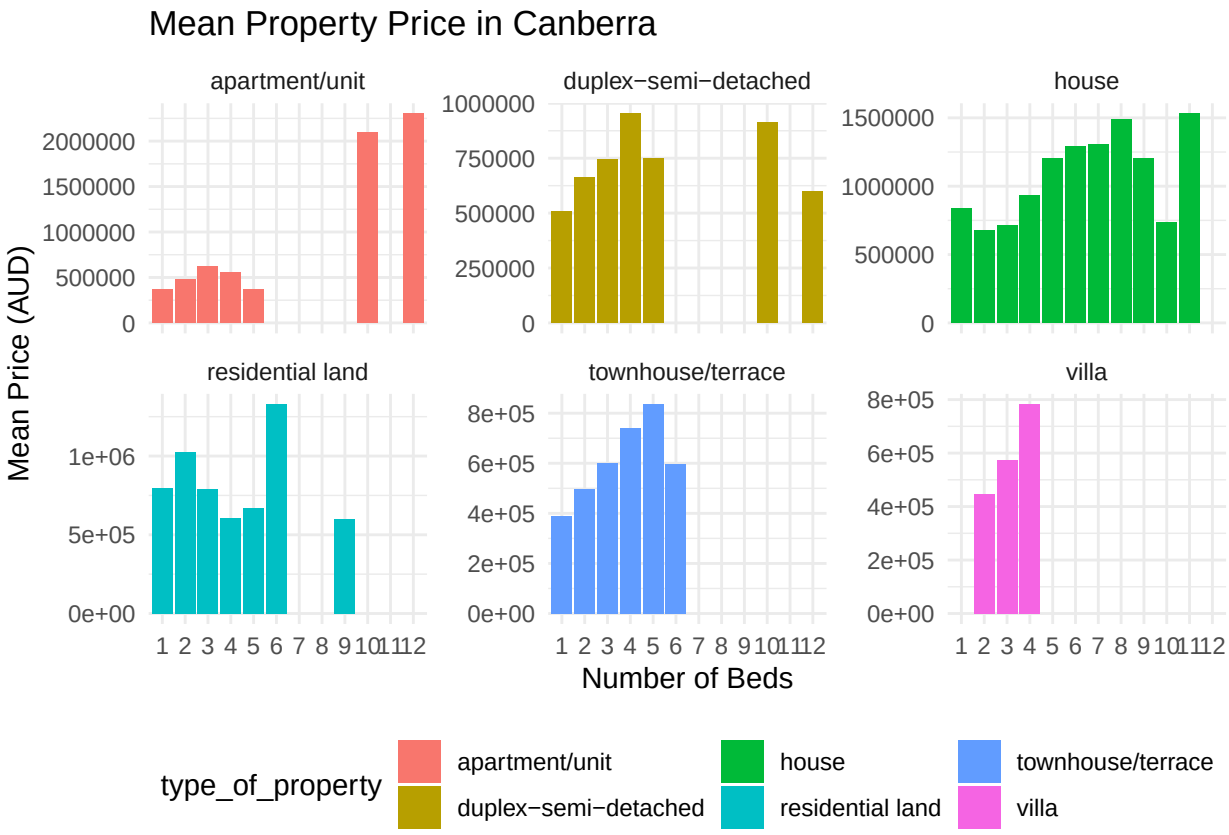
New.Category	Merged.from
House	House
Townhouse/Terrace	Townhouse, Terrace
Apartment/Unit	Apartment, Unit, Flat, Studio, Unitblock
Duplex-Semi-Detached	Duplex-semi-detached, Duplex/Semi-detached
Villa	Villa (or merged with Townhouse/Terrace)
Residential Land	Residential land
Rural Land	Cropping, Acreage-semi-rural
Other	Other, Retirement living

```r
canberra_sold_houses <- canberra_sold_houses %>%
  mutate(
    type_of_property = case_when(
      type_of_property %in% c("duplex-semi-detached", "duplex/semi-detached") ~ "duplex-
      type_of_property %in% c("apartment", "unit", "flat", "studio", "unitblock") ~ "apa
      type_of_property %in% c("townhouse", "terrace") ~ "townhouse/terrace",
      type_of_property %in% c("cropping", "acreage-semi-rural") ~ "rural land",
      type_of_property == "residential land" ~ "residential land",
      type_of_property %in% c("retirement living", "other") ~ "other",
      TRUE ~ type_of_property  # Keep other categories unchanged
    )
  )
```

```r
filtrd_houses <- canberra_sold_houses %>%
  group_by(type_of_property) %>%
  filter(!type_of_property %in% c("other", "rural land")) %>%
  summarise(frq_property = n()) %>%
  arrange(desc(frq_property))

filtrd_houses %>%
  ggplot(aes(x = reorder(type_of_property, frq_property),
             y = frq_property,
             fill = type_of_property)) +
  geom_col() +
  geom_text(aes(label = frq_property),
            hjust = 0,  # Adjust position to avoid overlap
            size = 2) +  # Adjust text size
  coord_flip() +  # Flip for better readability
  labs(title = "Frequency of Property Types",
       x = "Type of Property",
       y = "") +
  theme_minimal() +
  theme(legend.position = "none")
```




```
canberra_sold_houses %>%
  filter(!type_of_property %in% c("other", "rural land")) %>%
  group_by(type_of_property, beds) %>%
  summarise(Mean = mean(price, na.rm = TRUE)) %>%
  arrange(desc(Mean)) %>%
  ggplot(aes(x = factor(beds), y = Mean, fill = type_of_property)) +
  geom_bar(stat = "identity") +
  facet_wrap(~type_of_property, scales = "free_y") +  # Creates subplots by property ty
  labs(title = "Mean Property Price in Canberra",
       x = "Number of Beds",
       y = "Mean Price (AUD)") +
  theme_minimal() +
  theme(legend.position = "bottom")
```



Something strange is happening in the data. When looking at the apartment/unit bar plot, we see some listings with 11 or 12 rooms, which doesn't make sense. It's unlikely that an apartment or unit in Australia would have that many rooms. I checked one of the addresses and found that the listing actually refers to an entire building with multiple units, not a single apartment.

Since buying an entire apartment block is very rare, I've decided to remove listings with 11 or 12 bedrooms, as well as properties with 8, 9, or 10 bedrooms. These cases are uncommon and don't represent a significant part of the real estate market. In total, 21 sample excluded, as shown below.

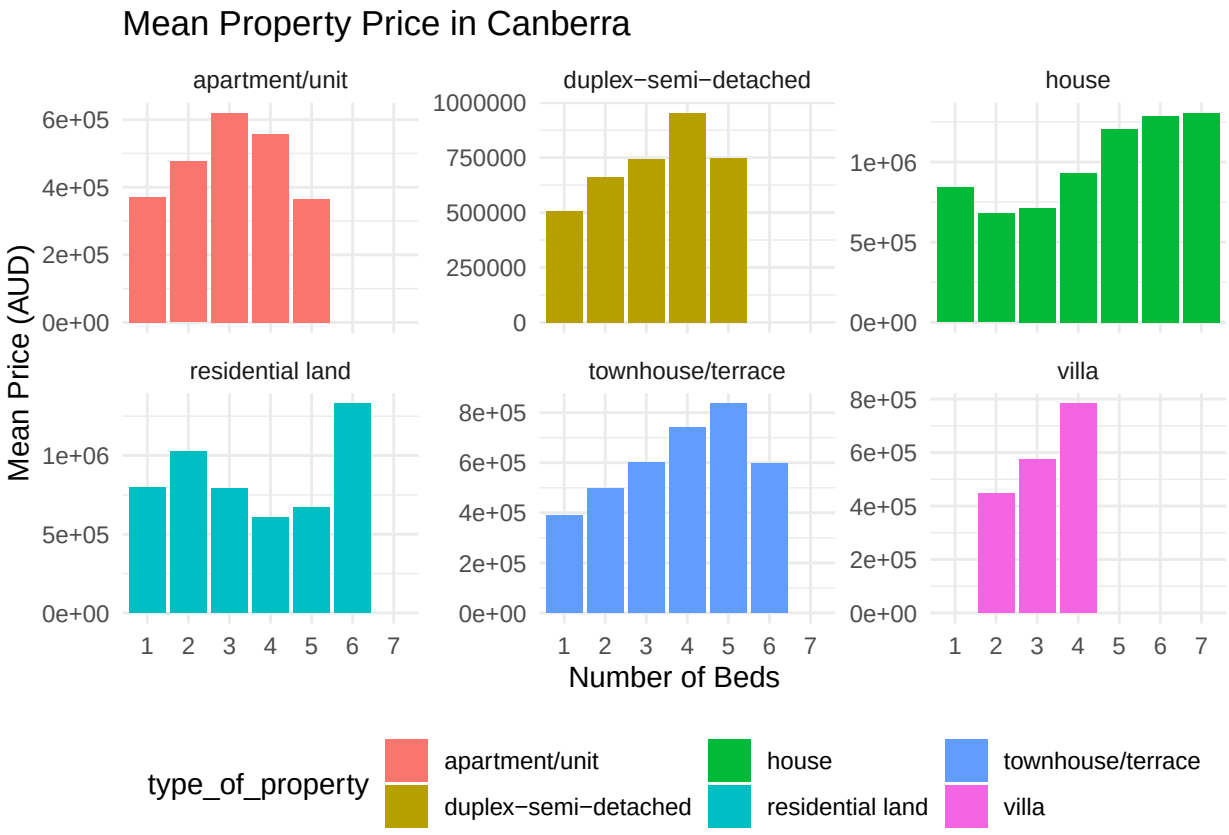
```
canberra_sold_houses %>%
  filter(!type_of_property %in% c("other", "rural land") & beds %in% 8:12) %>%
  summarise(total_excluded_beds = n())
```

```
##   total_excluded_beds
## 1                  21
```

The remaining are

```
canberra_sold_houses %>%
  filter(!type_of_property %in% c("other", "rural land") & beds %in% 1:7) %>%
  group_by(type_of_property, beds) %>%
  summarise(Mean = mean(price, na.rm = TRUE)) %>%
  arrange(desc(Mean)) %>%
```

```r
ggplot(aes(x = factor(beds), y = Mean, fill = type_of_property)) +
geom_bar(stat = "identity") +
facet_wrap(~type_of_property, scales = "free_y") +  # Creates subplots by property ty
labs(title = "Mean Property Price in Canberra",
     x = "Number of Beds",
     y = "Mean Price (AUD)") +
theme_minimal() +
theme(legend.position = "bottom")
```



Adding Latitude and Longitude

We need to add the latitude and longitude of each house because they are useful for spatial analysis, making both visualization and prediction better. With these coordinates, we can find the nearest schools, markets, universities, sports grounds, and bus stops. This matters because property prices are often higher for homes close to markets and other important places.

I am not going to run this code here as it takes long time. I have save the data and I will import it again here.

```r
clean_data <- canberra_sold_houses %>%
  filter(!type_of_property %in% c("other", "rural land") & beds %in% 1:7)
```

```r
# Initialize an empty dataframe to store results
geocoded_results <- data.frame()

# Define batch size
batch_size <- 1000
num_batches <- ceiling(nrow(clean_data) / batch_size)

# Loop through batches
for (i in 1:num_batches) {
  # Define start and end index for each batch
  start_index <- ((i - 1) * batch_size) + 1
  end_index <- min(i * batch_size, nrow(clean_data))  # Ensure it doesn't exceed data s

  # Select the batch
  batch <- clean_data %>%
    slice(start_index:end_index) %>%
    filter(!is.na(full_address))  # Remove rows with missing addresses

  # Retry mechanism for each batch
  success <- FALSE
  while (!success) {
    try({
      geocoded_batch <- batch %>%
        geocode(full_address, method = 'arcgis',
                lat = latitude, long = longitude)

      # Append successfully geocoded batch to results
      geocoded_results <- bind_rows(geocoded_results, geocoded_batch)

      # Print progress
      print(paste("Batch", i, "successfully processed"))

      # If no error occurs, mark success as TRUE
      success <- TRUE

    }, silent = TRUE)

    if (!success) {
      print(paste("Retrying batch", i, "..."))
      Sys.sleep(5)  # Wait for 5 seconds before retrying (optional)
    }
  }
}
```

```r
clean_data <- geocoded_results
```

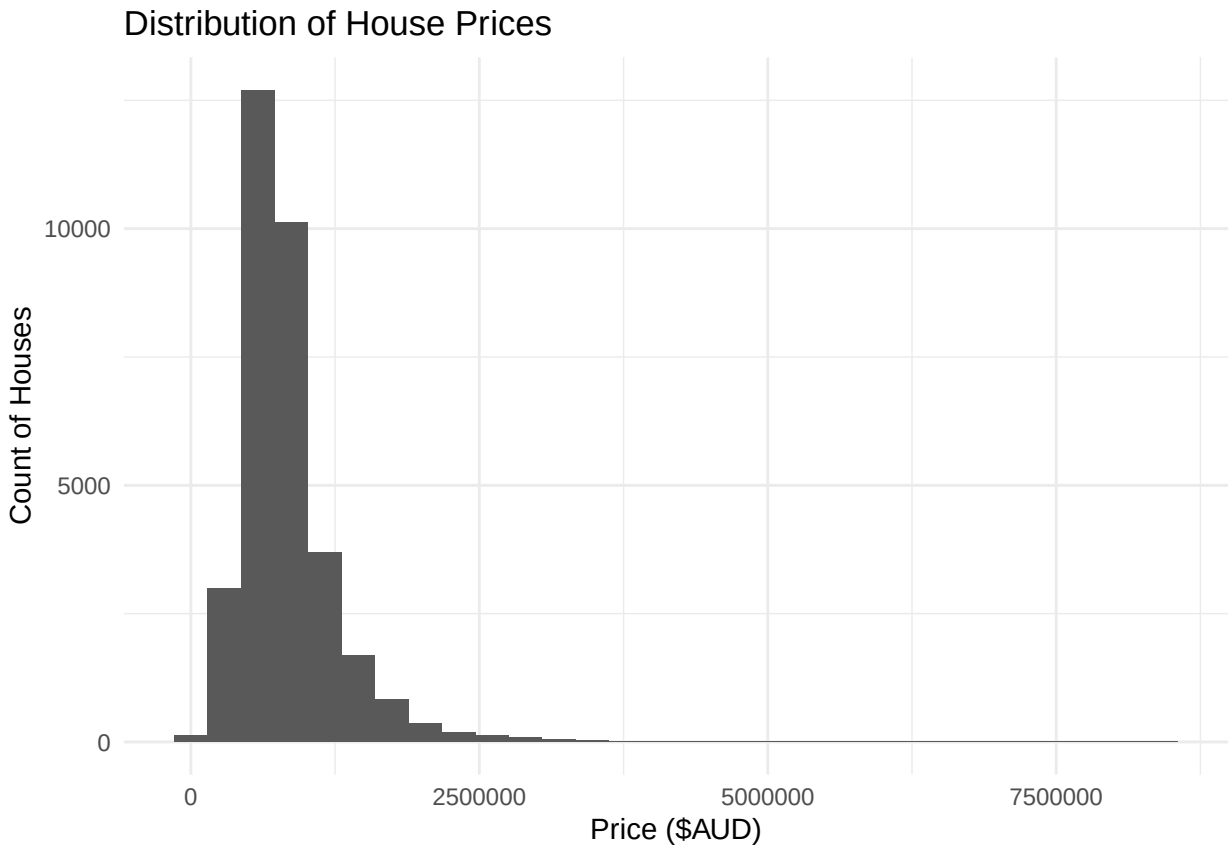
```r
clean_data <- read.csv("../Data cleaning/clean_data.csv", header = TRUE)
```

Exploration

The price distribution for all property types shows some values that stand out from the bulk, which may seem unusual. However, I have verified these observations with other real estate websites, and they are indeed real. This could be due to factors such as location, larger land size, or other property features. We will explore this further in the analysis.

```r
clean_data %>%
  ggplot(aes(x = price)) +
  geom_histogram() +  # Set bin width and colors
  labs(
    title = "Distribution of House Prices",
    x = "Price ($AUD)",
    y = "Count of Houses"
  ) +
  theme_minimal()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

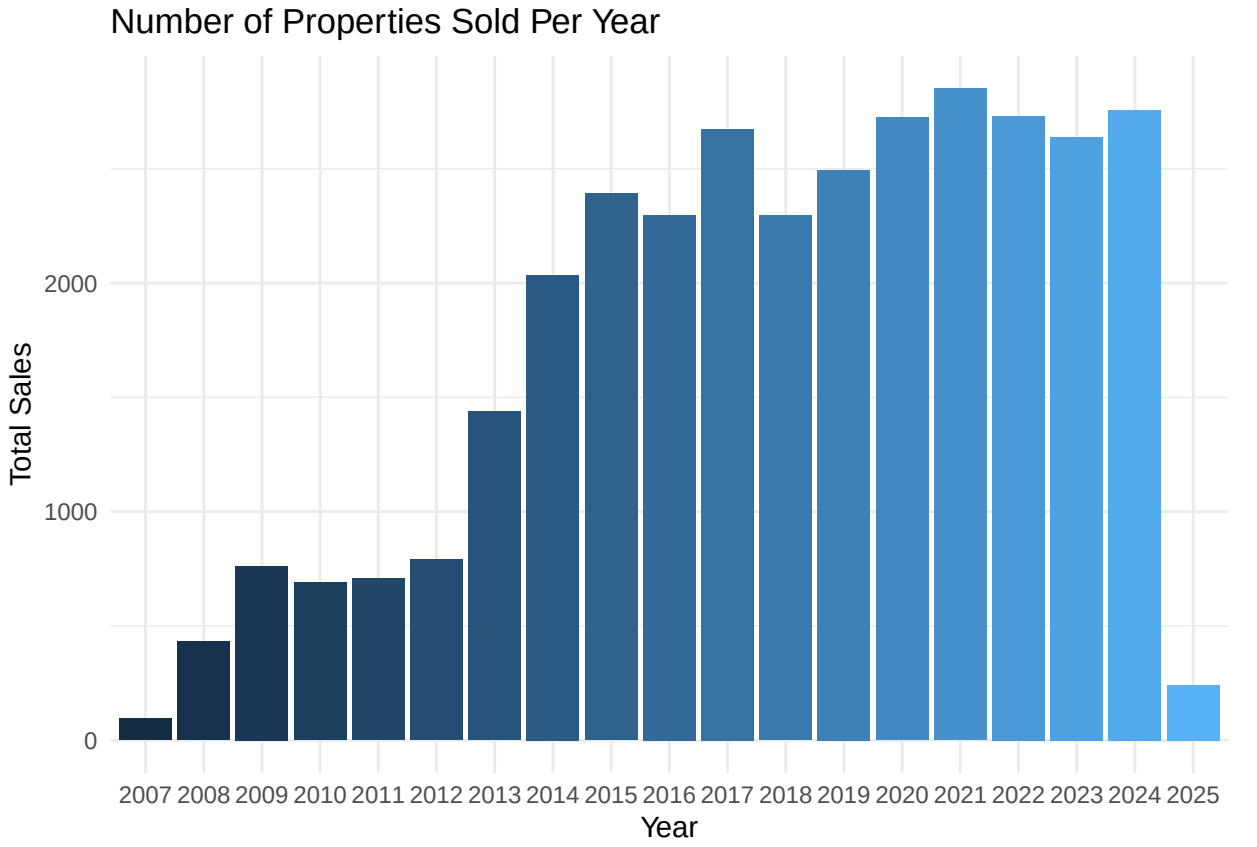


Number of observations in each year

- The number of sales starts **very low in 2007 (95 sales)** but gradually increases.
- From **2013 onward**, sales counts are significantly higher, suggesting **better data availability** and more consistent sample sizes.
- The **peak sales period** is between **2015 and 2024**, with counts consistently exceeding **2000 per year**, indicating a strong dataset for analysis.
- **Recent Drop in 2025:** Sales count is currently **243**, which is much lower than previous years, suggesting that **data collection for this year may still be incomplete**.

```
clean_data %>%
  mutate(sale_year = year(sold_date)) %>%  # Extract year from 'sold_date'
  group_by(sale_year) %>%
  summarise(total_sales = n(), .groups = "drop") %>%  # Count number of sales per year
  arrange(sale_year) %>%  # Ensure chronological order
  ggplot(aes(x = factor(sale_year), y = total_sales, fill = sale_year)) +  # Use 'facto
  geom_col() +
  labs(
    title = "Number of Properties Sold Per Year",
    x = "Year",
    y = "Total Sales"
  ) +
```

```
  theme_minimal() +
  theme(legend.position = "none")
```

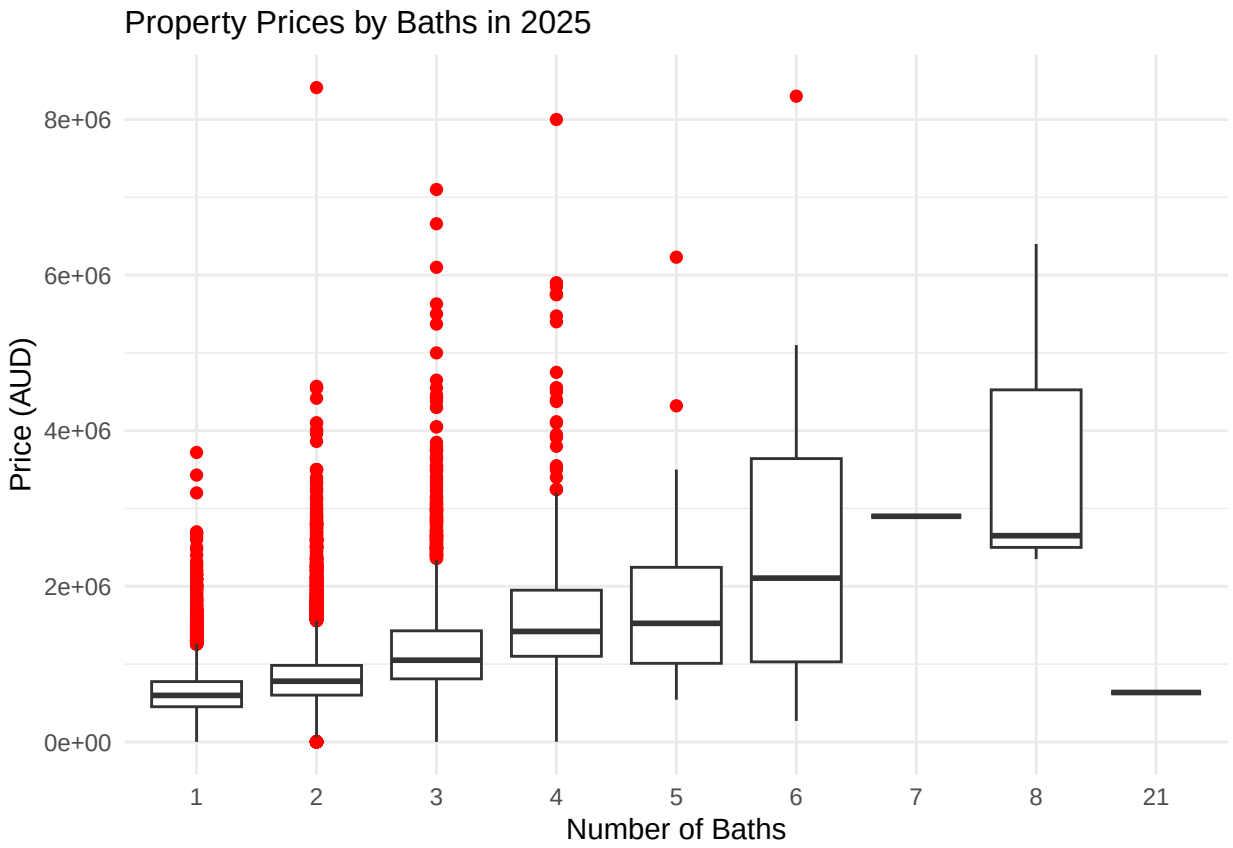


Bivariate Anaylsis

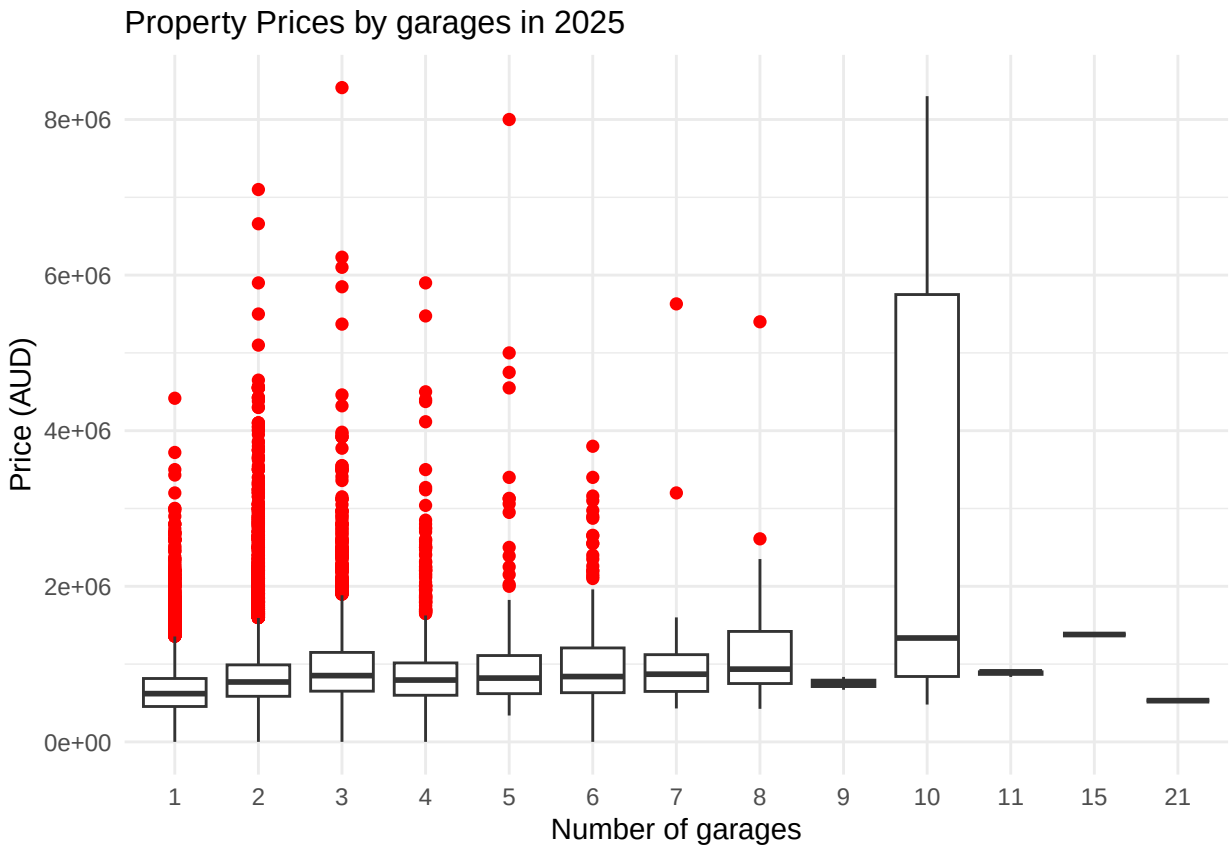
I want to check property prices by number of bedrooms, baths, and garages for just the latest year not all the years.

```
latest_year <- max(year(clean_data$sold_date), na.rm = TRUE)

clean_data %>%
  ggplot(aes(x = factor(beds), y = price)) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 2) +
  labs(
    title = paste("Property Prices by Bedrooms in", latest_year),
    x = "Number of Bedrooms",
    y = "Price (AUD)"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(size = 12))
```


```
latest_year <- max(year(clean_data$sold_date), na.rm = TRUE)

clean_data %>%
  ggplot(aes(x = factor(baths), y = price)) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 2) +
  labs(
    title = paste("Property Prices by Baths in", latest_year),
    x = "Number of Baths",
    y = "Price (AUD)"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(size = 12))
```

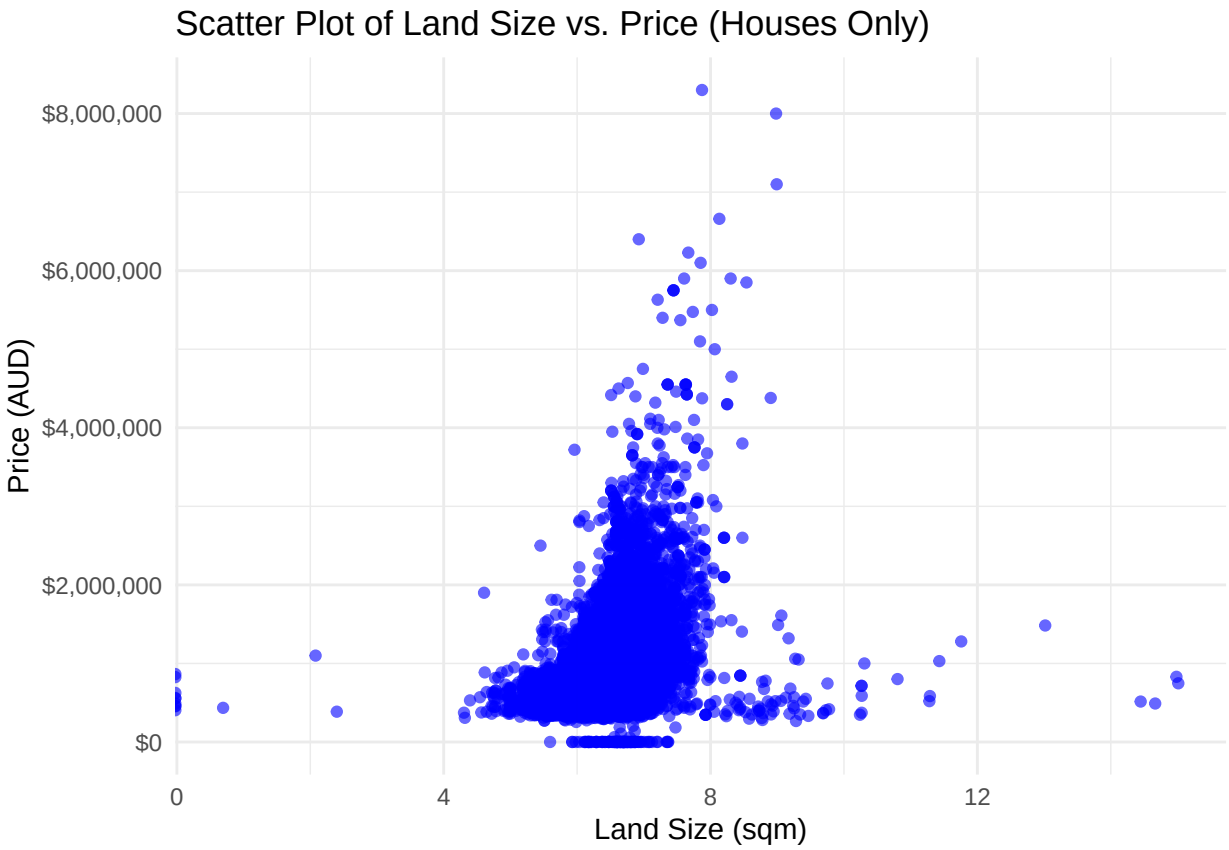



```
latest_year <- max(year(clean_data$sold_date), na.rm = TRUE)

clean_data %>%
  ggplot(aes(x = factor(garage), y = price)) +
  geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 2) +
  labs(
    title = paste("Property Prices by garages in", latest_year),
    x = "Number of garages",
    y = "Price (AUD)"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(size = 12))
```



Correlation Between land size and price

```
clean_data %>%
  filter(type_of_property == "house") %>%
  ggplot(aes(x = log(land_size), y = price)) +
  geom_point(alpha = 0.6, color = "blue") +  # Add transparency and color
  scale_x_continuous(labels = scales::comma_format()) +  # Format land size in thousand
  scale_y_continuous(labels = scales::dollar_format(prefix = "$")) +  # Format price in
  labs(
    title = "Scatter Plot of Land Size vs. Price (Houses Only)",
    x = "Land Size (sqm)",
    y = "Price (AUD)"
  ) +
  theme_minimal()
```


Scatter Plot of Land Size vs. Price (Houses Only)
$8,000,000
$6,000,000
$4,000,000
$2,000,000
$0
Price (AUD)
0
4
8
12
Land Size (sqm)