RentAnalytics

2024-05-02

Exploratory Data Analysis of UCIs “Apartment for Rent Classified”

1. Data Loading

```
# Set the working directory to the folder containing your data
setwd("C:/code/Courses/DataAnalytics/Project/RentAnalytics")
```

```
df <- read.csv('apartment_data_final.csv')
```

2. Initial Data Exploration

```
data <- df
```

```
dim(data)
```

```
## [1] 99125    14
```

Data Summary

```
str(data)
```

```
## 'data.frame':    99125 obs. of  14 variables:
##  $ bathrooms   : num  1 1.5 2 1 1 1.5 2 2 1 2 ...
##  $ bedrooms    : int  1 3 3 2 1 2 2 2 2 2 ...
##  $ fee         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ has_photo   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ price       : num  2195 1250 1395 1600 975 ...
##  $ square_feet : int  542 1500 1650 820 624 965 1120 947 600 1005 ...
##  $ state       : chr  "CA" "VA" "NC" "CA" ...
##  $ latitude    : num  33.9 37.1 35.8 38.4 35.1 ...
##  $ longitude   : num  -118.4 -76.5 -78.6 -122 -106.6 ...
##  $ studio      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ dogs_allowed: int  1 1 0 1 1 1 0 1 0 0 ...
##  $ cats_allowed: int  0 1 0 1 1 1 0 1 1 0 ...
##  $ us_region   : chr  "West" "South" "South" "West" ...
##  $ us_division : chr  "Pacific" "South Atlantic" "South Atlantic" "Pacific" ...
```

Calculating the percentage of NA values in each column and sorting them in descending

```
sort(colMeans(is.na(data)), decreasing = TRUE)
```

```
##     bedrooms    bathrooms        price          fee    has_photo  square_feet
## 1.240858e-03 5.750315e-04 1.008827e-05 0.000000e+00 0.000000e+00 0.000000e+00
##        state     latitude    longitude       studio dogs_allowed cats_allowed
## 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
##    us_region  us_division
## 0.000000e+00 0.000000e+00
```

Identifying Variables with Missing Values

```
vars_with_na <- names(data)[colSums(is.na(data)) > 0]
```

```
na_percentage <- colMeans(is.na(data[vars_with_na]))
```

```
na_percentage
```

```
##    bathrooms     bedrooms        price
## 5.750315e-04 1.240858e-03 1.008827e-05
```

Aggregating Data for Analysis

This analysis helps understand regional price differences.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Calculating the average price by region and sorting the results
average_price_by_region <- data %>%
  group_by(us_region) %>%
  summarise(average_price = mean(price, na.rm = TRUE)) %>%
  arrange(desc(average_price))

# Printing the results
print(average_price_by_region)
```

```
## # A tibble: 4 × 2
##   us_region average_price
##   <chr>             <dbl>
## 1 Northeast         1988.
## 2 West              1851.
## 3 South             1336.
## 4 Midwest           1109.
```

```
library(dplyr)

# Average price by state
state_avg_price <- df %>%
  group_by(state) %>%
  summarise(average_price = mean(price, na.rm = TRUE))
print(state_avg_price)
```

```
## # A tibble: 51 × 2
##    state average_price
##    <chr>         <dbl>
##  1 AK            1051.
##  2 AL             960.
##  3 AR             875.
##  4 AZ            1119.
##  5 CA            2463.
##  6 CO            1554.
##  7 CT            1266.
##  8 DC            2112.
##  9 DE            1155.
## 10 FL            1574.
## # ℹ 41 more rows
```

```
# Average price by presence of photos
photo_avg_price <- df %>%
  group_by(has_photo) %>%
  summarise(average_price = mean(price, na.rm = TRUE))
print(photo_avg_price)
```

```
## # A tibble: 2 × 2
##   has_photo average_price
##       <int>         <dbl>
## 1         0         1618.
## 2         1         1516.
```

```
# Average price by region and division
region_division_avg_price <- df %>%
  group_by(us_region, us_division) %>%
  summarise(average_price = mean(price, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'us_region'. You can override using the
## `.groups` argument.
```

```
print(region_division_avg_price)
```

```
## # A tibble: 9 × 3
## # Groups:   us_region [4]
##   us_region us_division        average_price
##   <chr>     <chr>                      <dbl>
## 1 Midwest   East North Central         1170.
## 2 Midwest   West North Central         1015.
## 3 Northeast Middle Atlantic            1959.
## 4 Northeast New England                2015.
## 5 South     East South Central         1062.
## 6 South     South Atlantic             1433.
## 7 South     West South Central         1172.
## 8 West      Mountain                   1361.
## 9 West      Pacific                    2313.
```

```
# Average price for studios
studio_avg_price <- df %>%
  group_by(studio) %>%
  summarise(average_price = mean(price, na.rm = TRUE))
print(studio_avg_price)
```

```
## # A tibble: 2 × 2
##   studio average_price
##    <int>         <dbl>
## 1      0         1528.
## 2      1         1447.
```

```
# Average price considering pet policy
pet_avg_price <- df %>%
  group_by(dogs_allowed, cats_allowed) %>%
  summarise(average_price = mean(price, na.rm = TRUE))
```

```
## `summarise()` has grouped output by 'dogs_allowed'. You can override using the
## `.groups` argument.
```

```
print(pet_avg_price)
```

```
## # A tibble: 4 × 3
## # Groups:   dogs_allowed [2]
##   dogs_allowed cats_allowed average_price
##          <int>        <int>         <dbl>
## 1            0            0         1565.
## 2            0            1         2057.
## 3            1            0         1395.
## 4            1            1         1465.
```

- **Group by state and calculate average price**
  This would allow you to see the average price of properties in each state.

- **Group by whether the property has photos and calculate average price** This would help you understand if having photos affects the price.
- **Group by US region and division and calculate average price** How price varies across different regions and divisions of the US.
- **Group by two categories—dogs_allowed and cats_allowed and calculate average price** Helps to understand the impact of pet-friendliness on rental costs.

Exploring Boolean and Numerical Variables

```
bool_vars <- names(data)[sapply(data, function(x) length(unique(x)) == 2)]

# Display the first few rows of these columns
head(data[bool_vars])
```

```
##   fee has_photo studio dogs_allowed cats_allowed
## 1   0         1      0            1            0
## 2   0         1      0            1            1
## 3   0         1      0            0            0
## 4   0         1      0            1            1
## 5   0         1      0            1            1
## 6   0         1      0            1            1
```

```
# Identify numerical variables, excluding boolean ones
num_vars <- names(data)[sapply(data, is.numeric) & !names(data) %in% bool_vars]

print(paste('Number of numerical variables: ', length(num_vars)))
```

```
## [1] "Number of numerical variables:  6"
```

```
head(data[num_vars])
```

```
##   bathrooms bedrooms price square_feet latitude longitude
## 1       1.0        1  2195         542  33.8520 -118.3759
## 2       1.5        3  1250        1500  37.0867  -76.4941
## 3       2.0        3  1395        1650  35.8230  -78.6438
## 4       1.0        2  1600         820  38.3622 -121.9712
## 5       1.0        1   975         624  35.1038 -106.6110
## 6       1.5        2  1250         965  35.1038 -106.6110
```

Geospatial Visualization

Map of the world and plots the geospatial data on it, visualizing the locations of the apartments on a global map. This helps understand the geographical distribution of the dataset’s entries.

```
library(sf)
```

```
## Warning: package 'sf' was built under R version 4.3.2
```

```
## Linking to GEOS 3.11.2, GDAL 3.7.2, PROJ 9.3.0; sf_use_s2() is TRUE
```

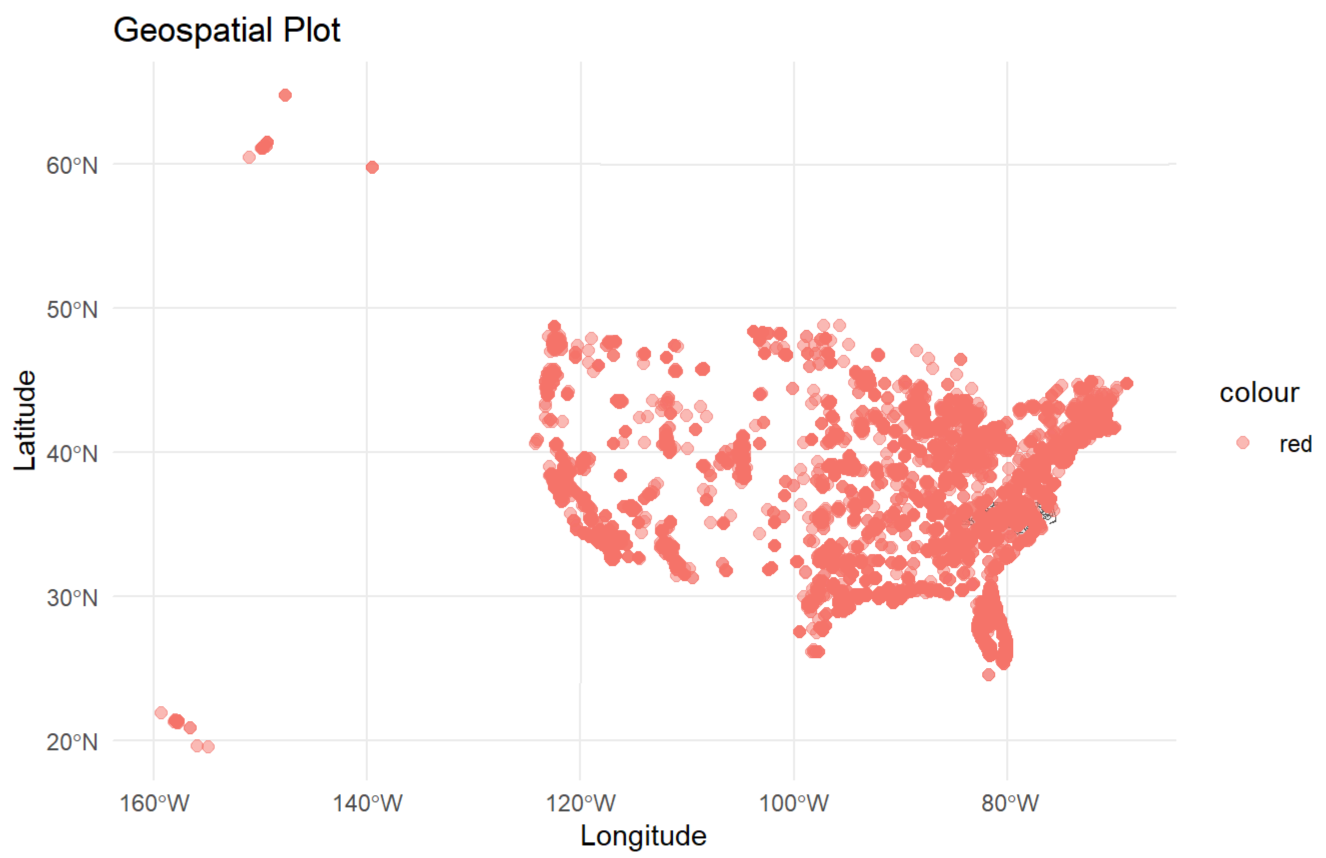
```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```r
# Create an sf object (adjust as per your previous transformations)
data_sf <- st_as_sf(data, coords = c("longitude", "latitude"), crs = 4326, agr = "constant")

# Read the world map (adjust the file path to your specific file)
world <- st_read(system.file("shape/nc.shp", package="sf"))  # Update with your path
```

```
## Reading layer `nc' from data source
##   `C:\Users\singh\AppData\Local\R\win-library\4.3\sf\shape\nc.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 100 features and 14 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## Geodetic CRS:  NAD27
```

```r
# Plotting with adjustments
ggplot() +
  geom_sf(data = world, fill = "gray90") +  # Adjusting world map color
  geom_sf(data = data_sf, aes(color = 'red'), shape = 19, size = 2, alpha = 0.5) +  # Smalle
r, transparent red points
  theme_minimal() +
  labs(title = "Geospatial Plot", x = "Longitude", y = "Latitude")
```



Identifying Discrete Variables

- Identifies discrete numerical variables in the dataset, defined here as those numerical variables that have fewer than 20 unique values and are not ‘id’ or ‘price’.

```
# Load necessary library
library(dplyr)

# Identify discrete variables from the set of numerical variables, excluding 'id' and 'price'
discrete_vars <- num_vars[num_vars != "id" & num_vars != "price" & sapply(data[num_vars], function(x) length(unique(x)) < 20)]

# Print the number of discrete variables
print(paste('Number of discrete variables: ', length(discrete_vars)))
```

```
## [1] "Number of discrete variables:  2"
```

```
# Display the first few rows of these discrete variables
head(data[discrete_vars])
```

```
##   bathrooms bedrooms
## 1       1.0        1
## 2       1.5        3
## 3       2.0        3
## 4       1.0        2
## 5       1.0        1
## 6       1.5        2
```

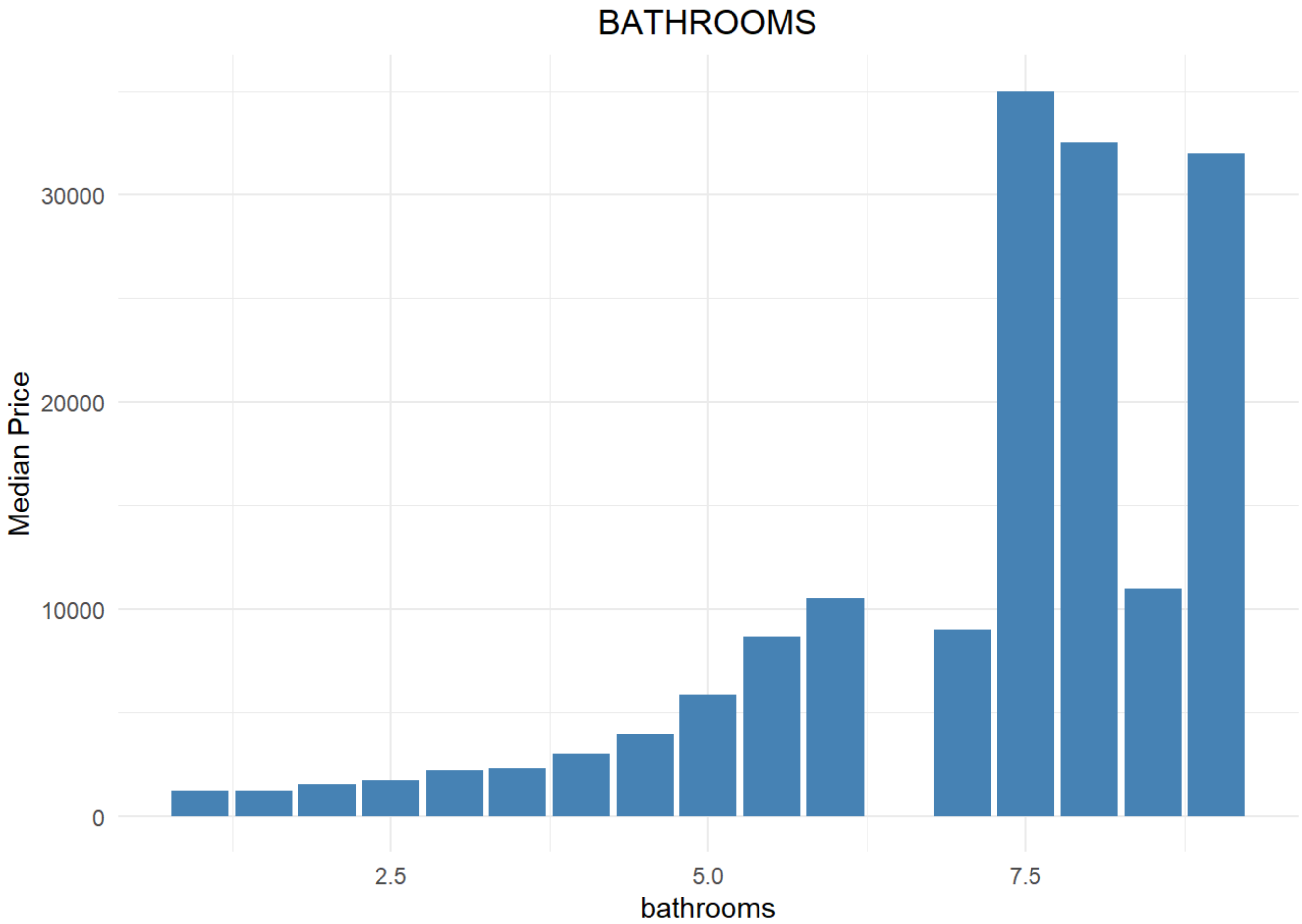
Visualizing Discrete Variables

```
library(ggplot2)
library(dplyr)

analyse_discrete <- function(df, var) {
  # Create a variable symbol from the string
  var_sym <- rlang::sym(var)

  # Creating a summary of median prices by the discrete variable
  grs <- df %>%
    group_by(!!var_sym) %>%
    summarise(price_median = median(price, na.rm = TRUE)) %>%
    ungroup()

  # Plotting the results
  p <- ggplot(grs, aes(x = !!var_sym, y = price_median)) +
    geom_bar(stat = "identity", fill = "steelblue") +
    theme_minimal() +
    labs(title = toupper(var), x = var, y = "Median Price") +
    theme(plot.title = element_text(hjust = 0.5)) # Center the plot title

  print(p)
}
```

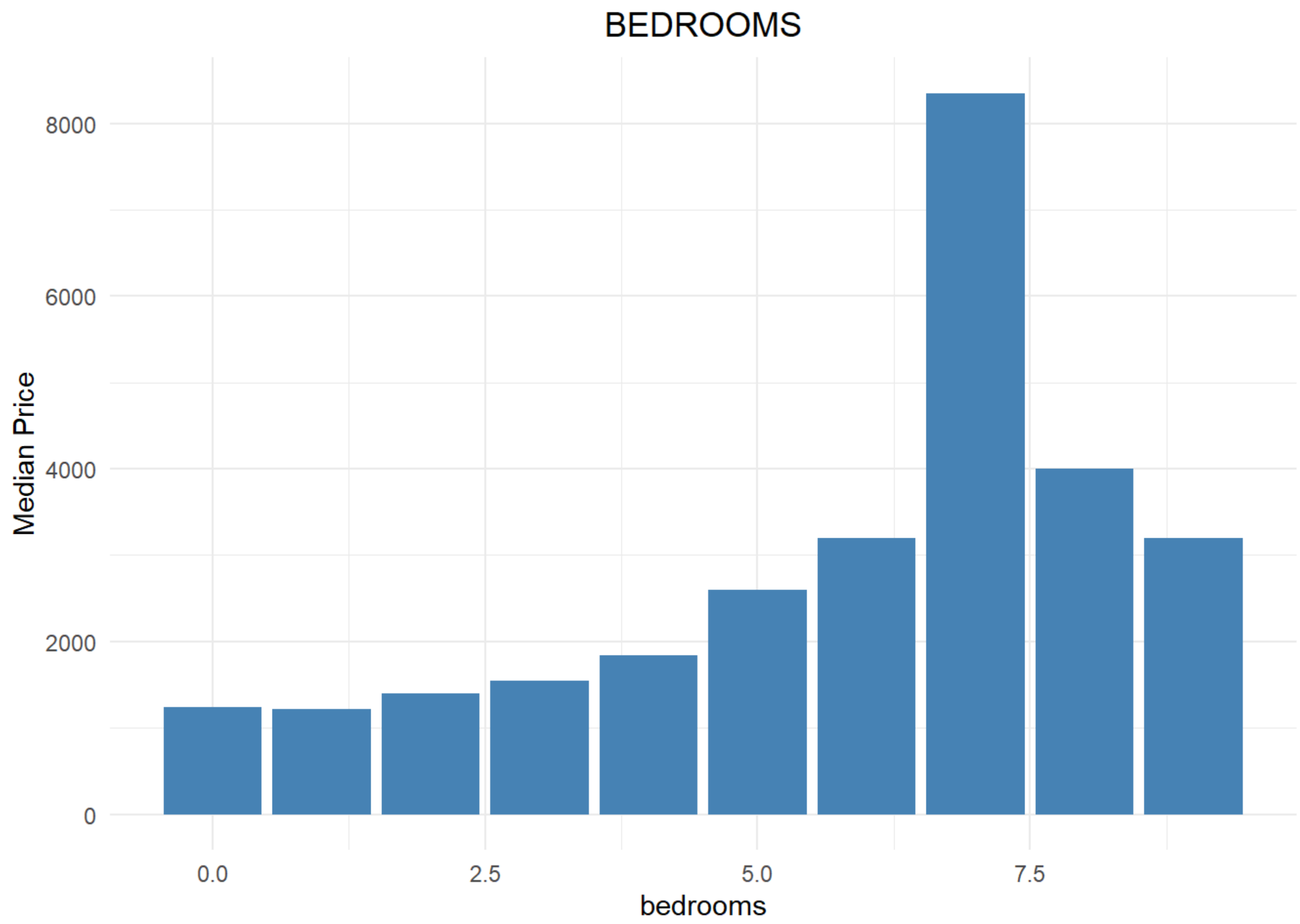
- This function visualizes the relationship between each discrete variable and the 'price' using a bar plot, showing the median price for each category of the discrete variable.

```
# Apply the function to each discrete variable
for (var in discrete_vars) {
  analyse_discrete(data, var)
}
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```




```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_bar()`).
```



Identifying and Analyzing Continuous Variable Distributions

- Identifies continuous variables, considered as those not already classified as discrete and excluding ‘id’.

```
cont_vars <- num_vars[!(num_vars %in% c(discrete_vars, "id"))]

print(paste('Number of continuous variables: ', length(cont_vars)))
```

```
## [1] "Number of continuous variables:  4"
```

```
head(data[cont_vars])
```

```
##   price square_feet latitude longitude
## 1  2195         542  33.8520 -118.3759
## 2  1250        1500  37.0867  -76.4941
## 3  1395        1650  35.8230  -78.6438
## 4  1600         820  38.3622 -121.9712
## 5   975         624  35.1038 -106.6110
## 6  1250         965  35.1038 -106.6110
```

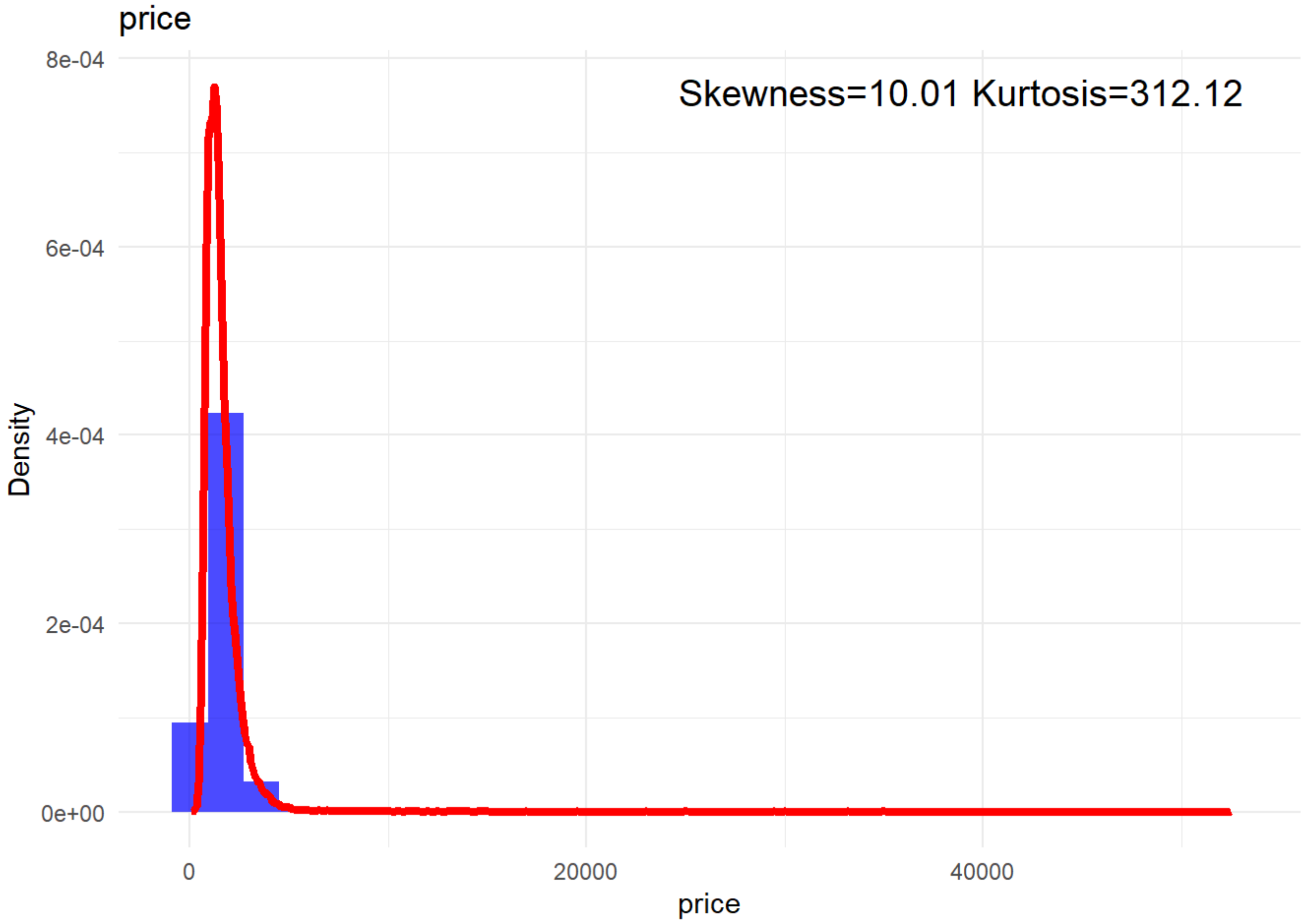
```r
library(ggplot2)
library(dplyr)
library(moments)

analyse_continuous <- function(df, var) {
  # Ensure the variable is a symbol for tidy evaluation
  var_sym <- rlang::sym(var)

  # Remove NA values and prepare data
  df <- df %>%
    filter(!is.na(!!var_sym))

  # Create the distribution plot
  p <- ggplot(df, aes(x = !!var_sym)) +
    geom_histogram(aes(y = ..density..), bins = 30, fill = "blue", alpha = 0.7) +
    geom_density(color = "red", size = 1.5) +
    labs(title = var, x = var, y = "Density") +
    theme_minimal()

  # Calculate skewness and kurtosis
  skewness <- moments::skewness(df[[var]])
  kurtosis <- moments::kurtosis(df[[var]])

  # Add annotations for skewness and kurtosis
  p <- p + annotate("text", x = Inf, y = Inf, label = sprintf("Skewness=%.2f Kurtosis=%.2f",
skewness, kurtosis),
                    hjust = 1.1, vjust = 2, size = 5, color = "black")

  # Print the plot
  print(p)
}
```

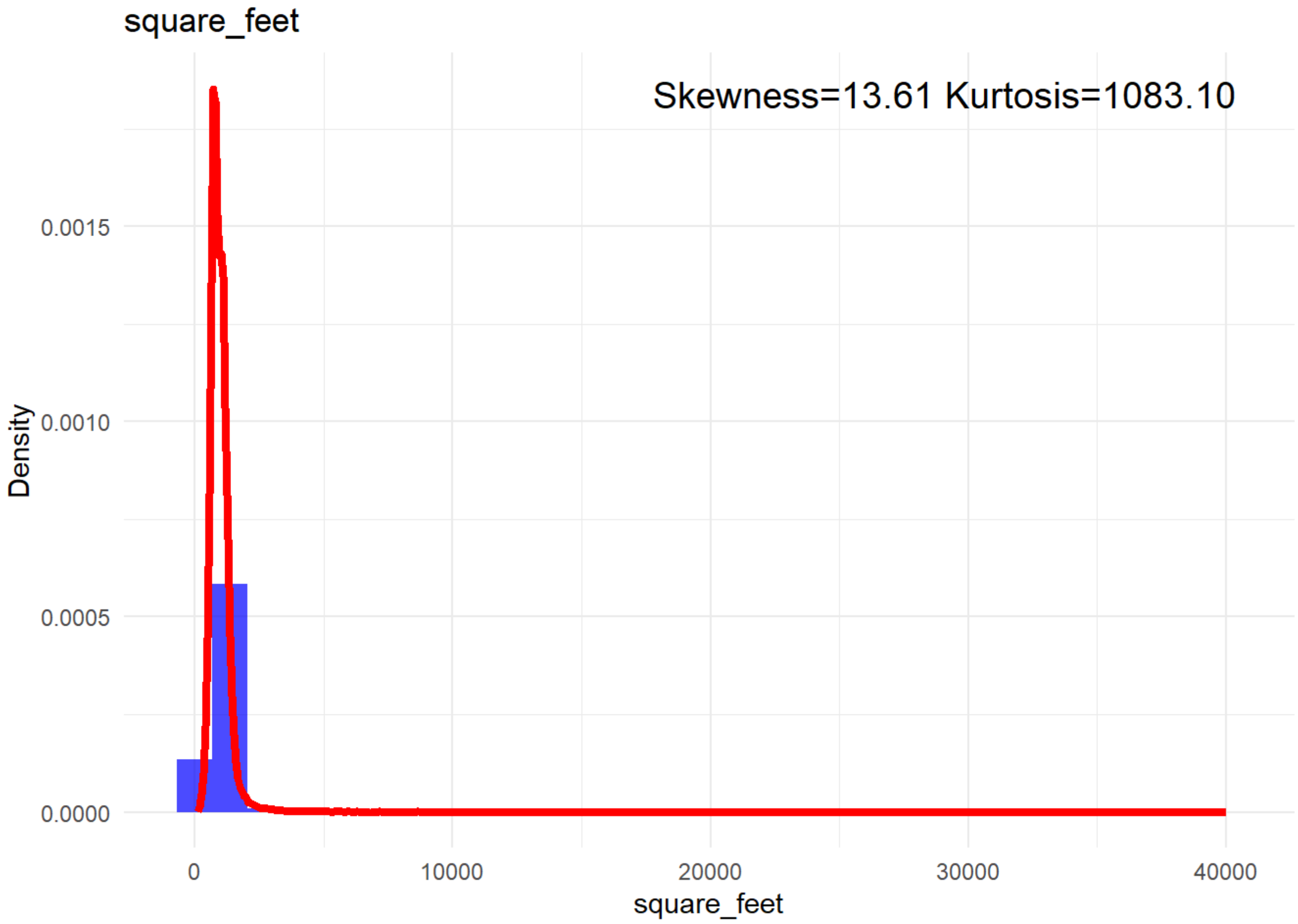
- This function visualizes the distribution of each continuous variable using histograms, including metrics like skewness and kurtosis for deeper insights into each distribution's shape.

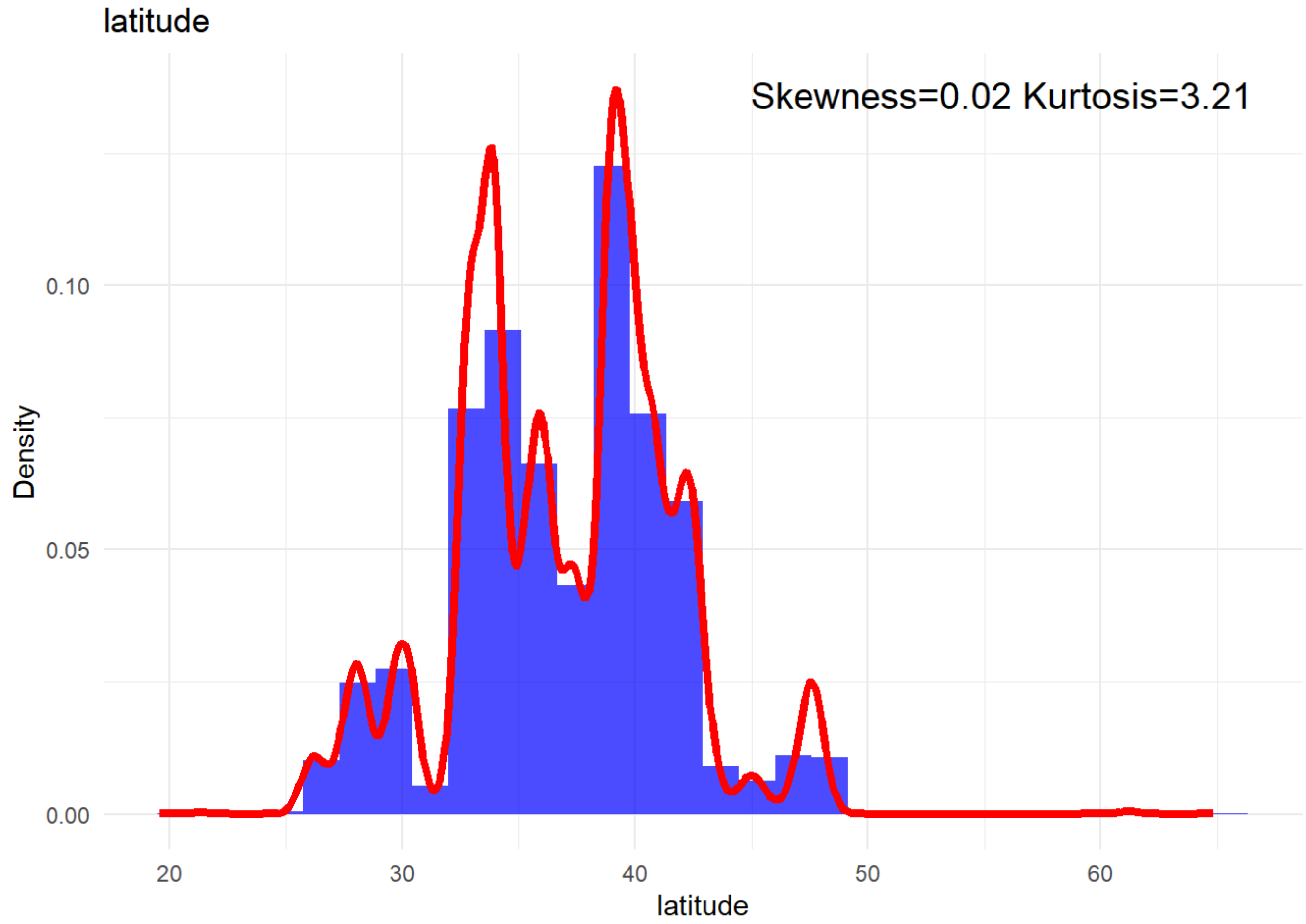
```r
# Apply the function to each continuous variable
for (var in cont_vars) {
  analyse_continuous(data, var)
}
```

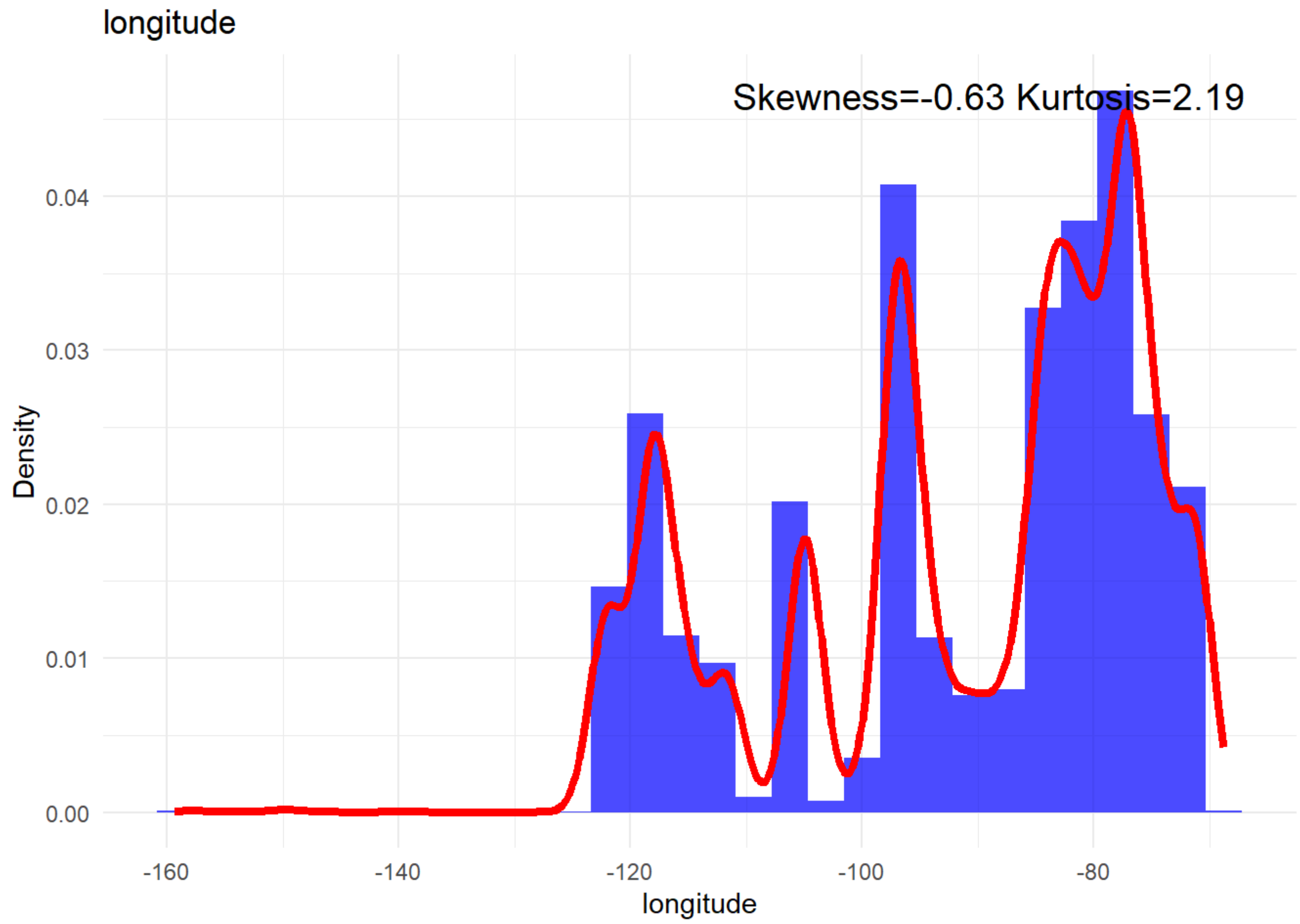
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## ℹ Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


price
Skewness=10.01 Kurtosis=312.12
Density
8e-04
6e-04
4e-04
2e-04
0e+00
0
20000
40000
price


square_feet
Skewness=13.61 Kurtosis=1083.10
Density
0.0015
0.0010
0.0005
0.0000
0
10000
20000
30000
40000
square_feet



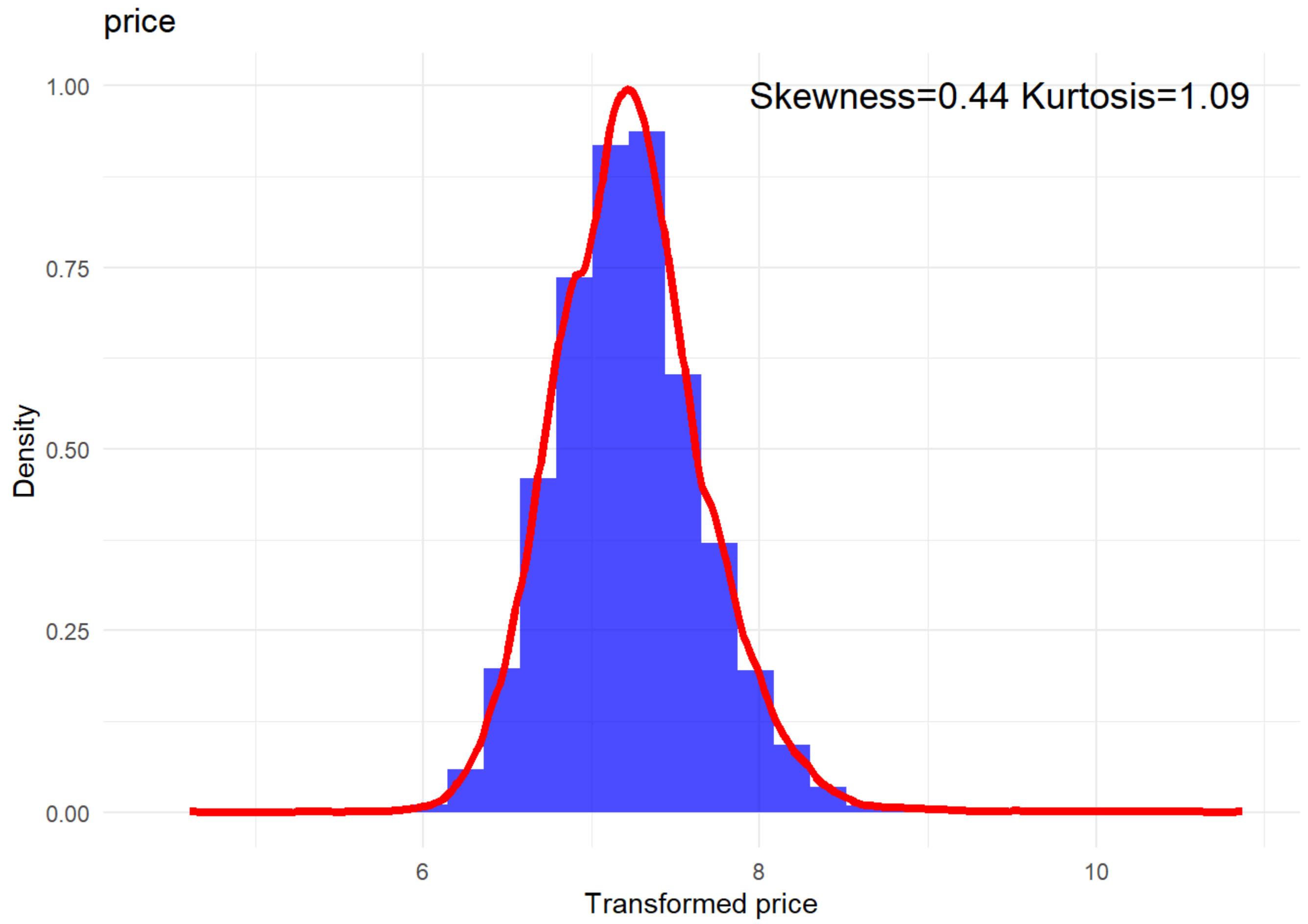


Logarithmic Transformation on Continuous Variables

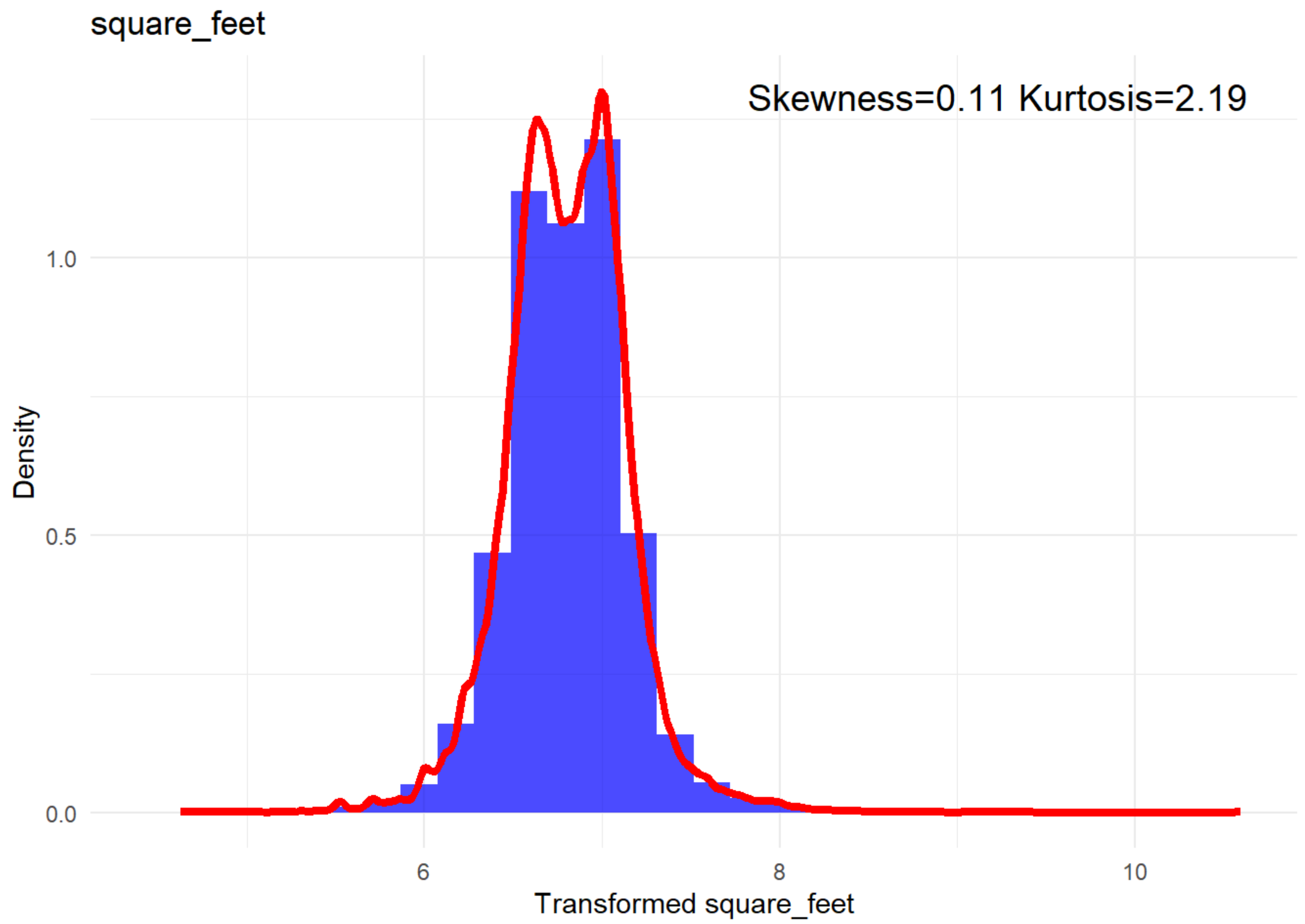
```r
library(ggplot2)
library(dplyr)
library(moments)  # For skewness and kurtosis

analyse_transformed_continuous <- function(df, var) {
  # Ensure the variable is a symbol for tidy evaluation
  var_sym <- rlang::sym(var)

  # Remove NA values
  df <- df %>%
    filter(!is.na(!!var_sym))

  # Skip transformation for 'latitude' or 'longitude'
  if (var %in% c('latitude', 'longitude')) {
    message(paste("Skipping transformation for", var))
  } else {
    # Apply logarithmic transformation with +1 to handle zero and negative values
    df <- df %>%
      mutate(!!var_sym := log1p(!!var_sym))
  }

  # Create the distribution plot
  p <- ggplot(df, aes(x = !!var_sym)) +
    geom_histogram(aes(y = ..density..), bins = 30, fill = "blue", alpha = 0.7) +
    geom_density(color = "red", size = 1.5) +
    labs(title = var, x = paste("Transformed", var), y = "Density") +
    theme_minimal()

  # Calculate skewness and kurtosis
  skewness_val <- skewness(df[[var]], na.rm = TRUE)
  kurtosis_val <- kurtosis(df[[var]], na.rm = TRUE) - 3  # Adjust kurtosis to match Python's
definition

  # Add annotations for skewness and kurtosis
  p <- p + annotate("text", x = Inf, y = Inf, label = sprintf("Skewness=%.2f Kurtosis=%.2f",
skewness_val, kurtosis_val),
                    hjust = 1.1, vjust = 2, size = 5, color = "black")

  # Print the plot
  print(p)
}
```

- Applies a logarithmic transformation to each continuous variable (except for geographical coordinates like latitude and longitude) and visualizes their new distributions to often normalize data and reduce skewness.

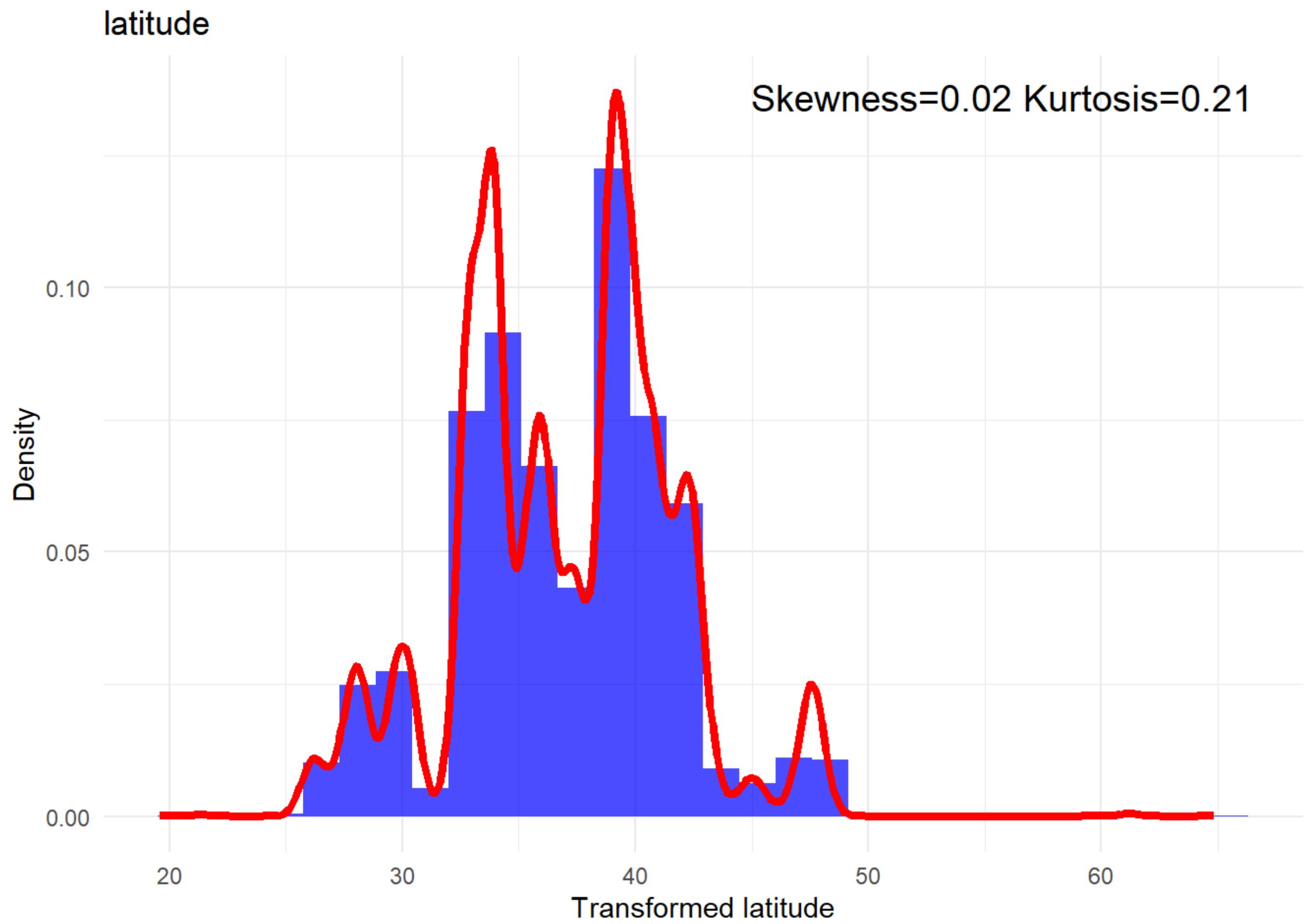
```r
# Apply the function to each continuous variable
for (var in cont_vars) {
  analyse_transformed_continuous(data, var)
}
```

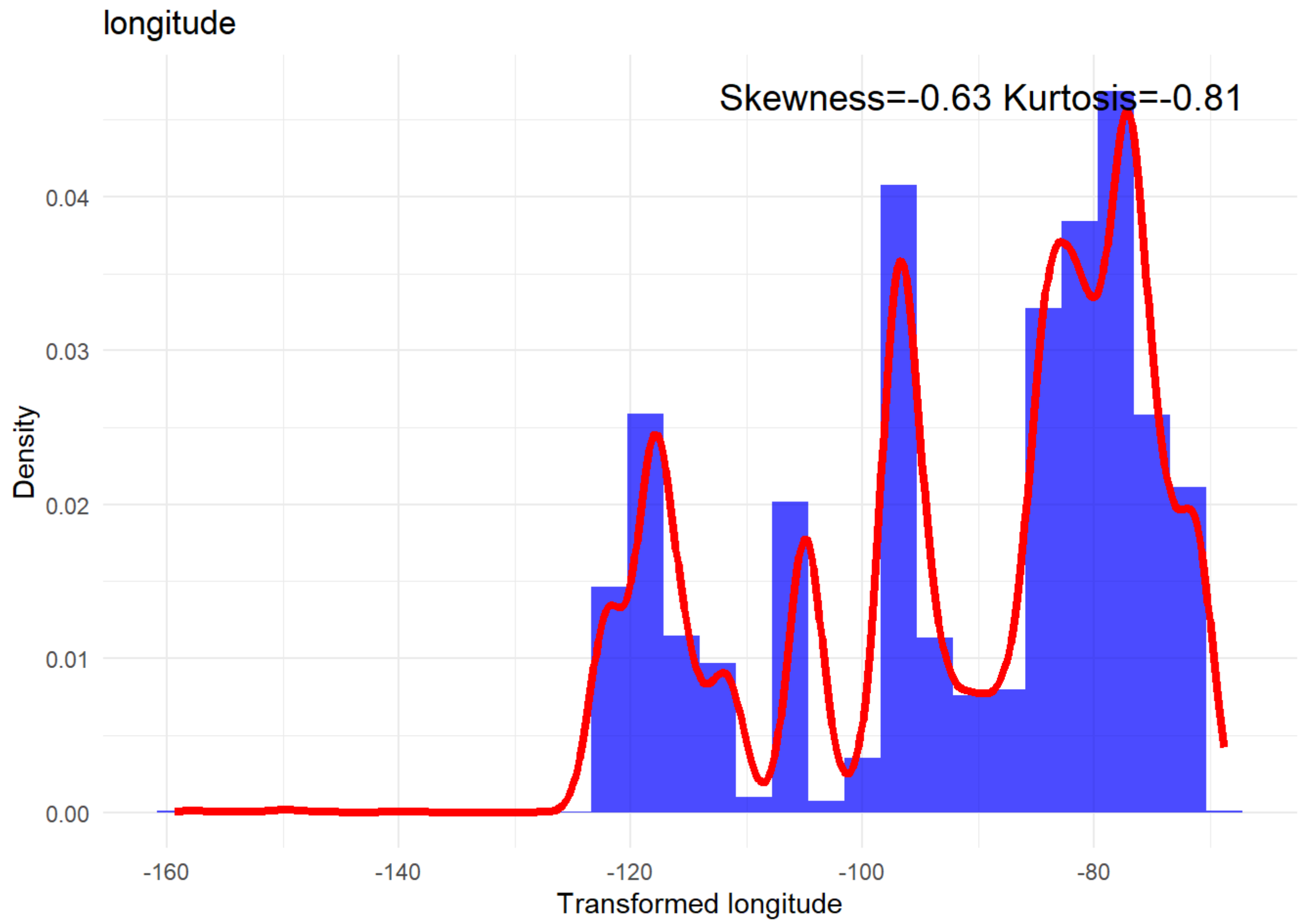



```
## Skipping transformation for latitude
```



```
## Skipping transformation for longitude
```





Visualizing Outliers in Continuous Variables

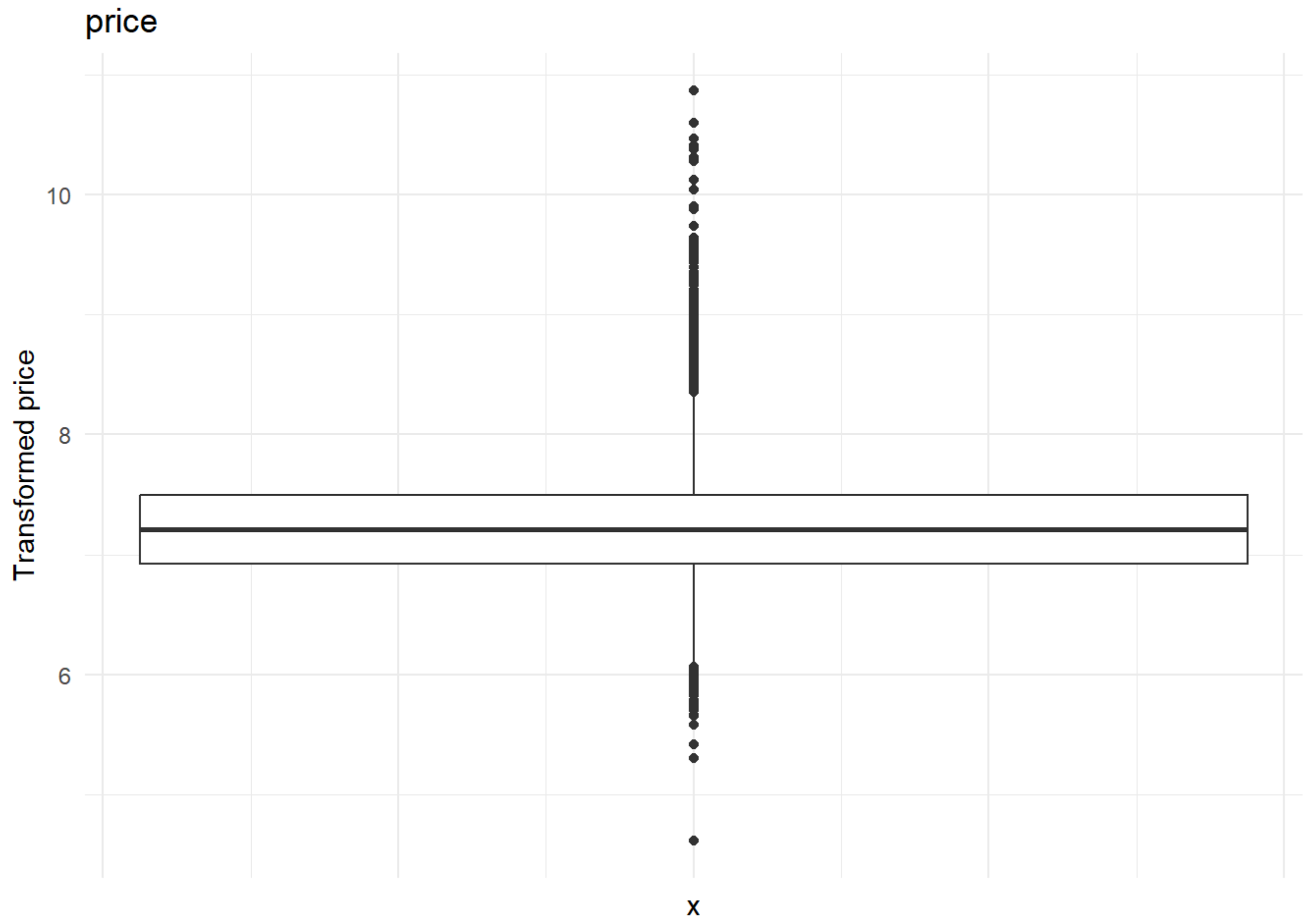
```r
# Removing rows where the 'price' column has NA values
df <- na.omit(df, cols = "price")
```

```r
library(ggplot2)
library(dplyr)

find_outliers <- function(df, var) {
  # Ensure the variable is a symbol for tidy evaluation
  var_sym <- rlang::sym(var)

  # Skip transformation for 'latitude' or 'longitude'
  if (var %in% c('latitude', 'longitude')) {
    message(paste("Skipping", var))
  } else {
    # Apply logarithmic transformation to handle zero and negative values
    df <- df %>%
      mutate(!!var_sym := log1p(!!var_sym))
  }

  # Plotting the boxplot
  p <- ggplot(df, aes_string(x = "1", y = as.character(var_sym))) +
    geom_boxplot() +
    labs(title = var, y = paste("Transformed", var)) +
    theme_minimal() +
    theme(axis.text.x=element_blank(), axis.ticks.x=element_blank())  # Hide x-axis details

  # Print the plot
  print(p)
}
```

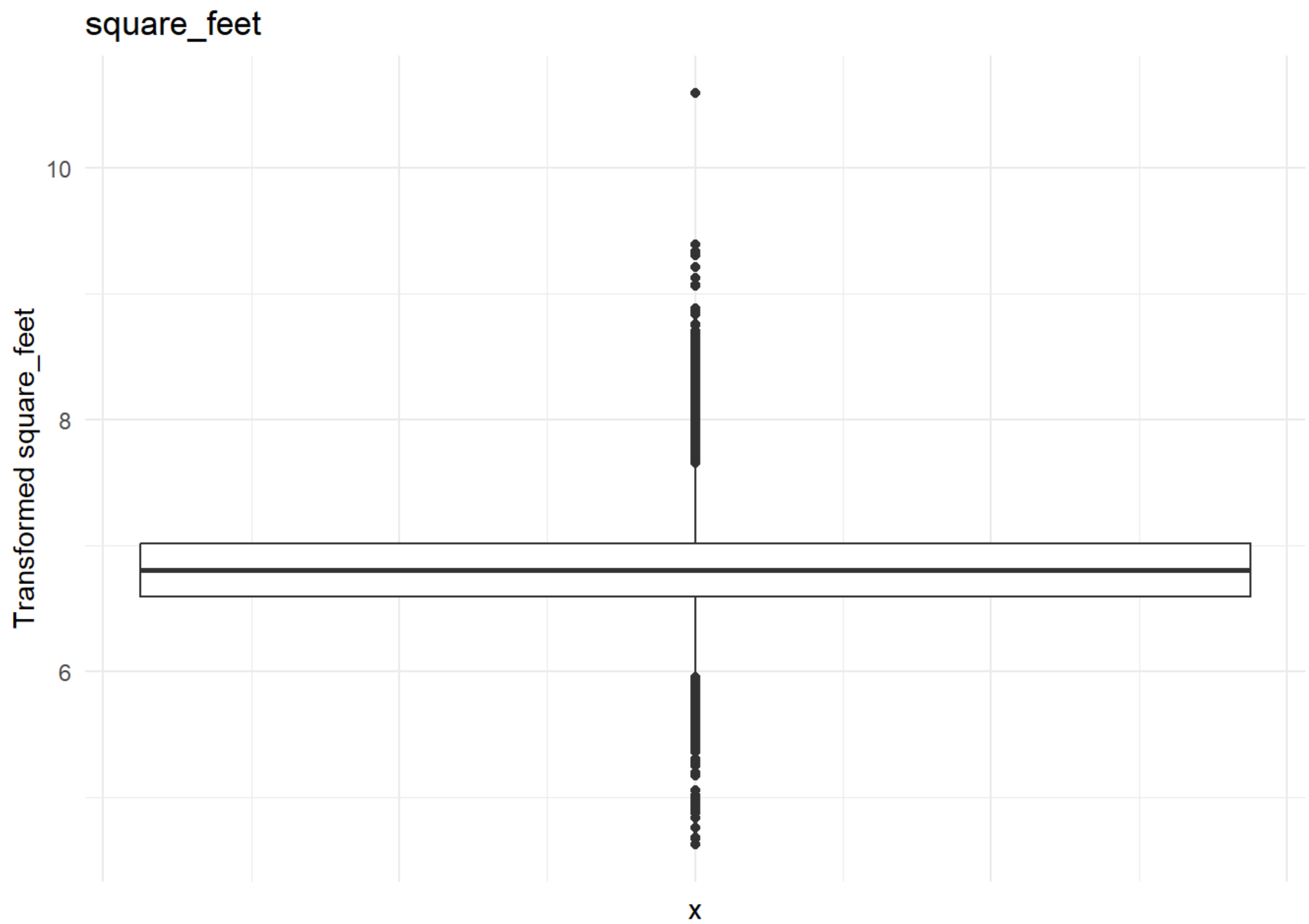
```r
# Apply the function to each continuous variable
for (var in cont_vars) {
  find_outliers(data, var)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## ℹ Please use tidy evaluation idioms with `aes()`.
## ℹ See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

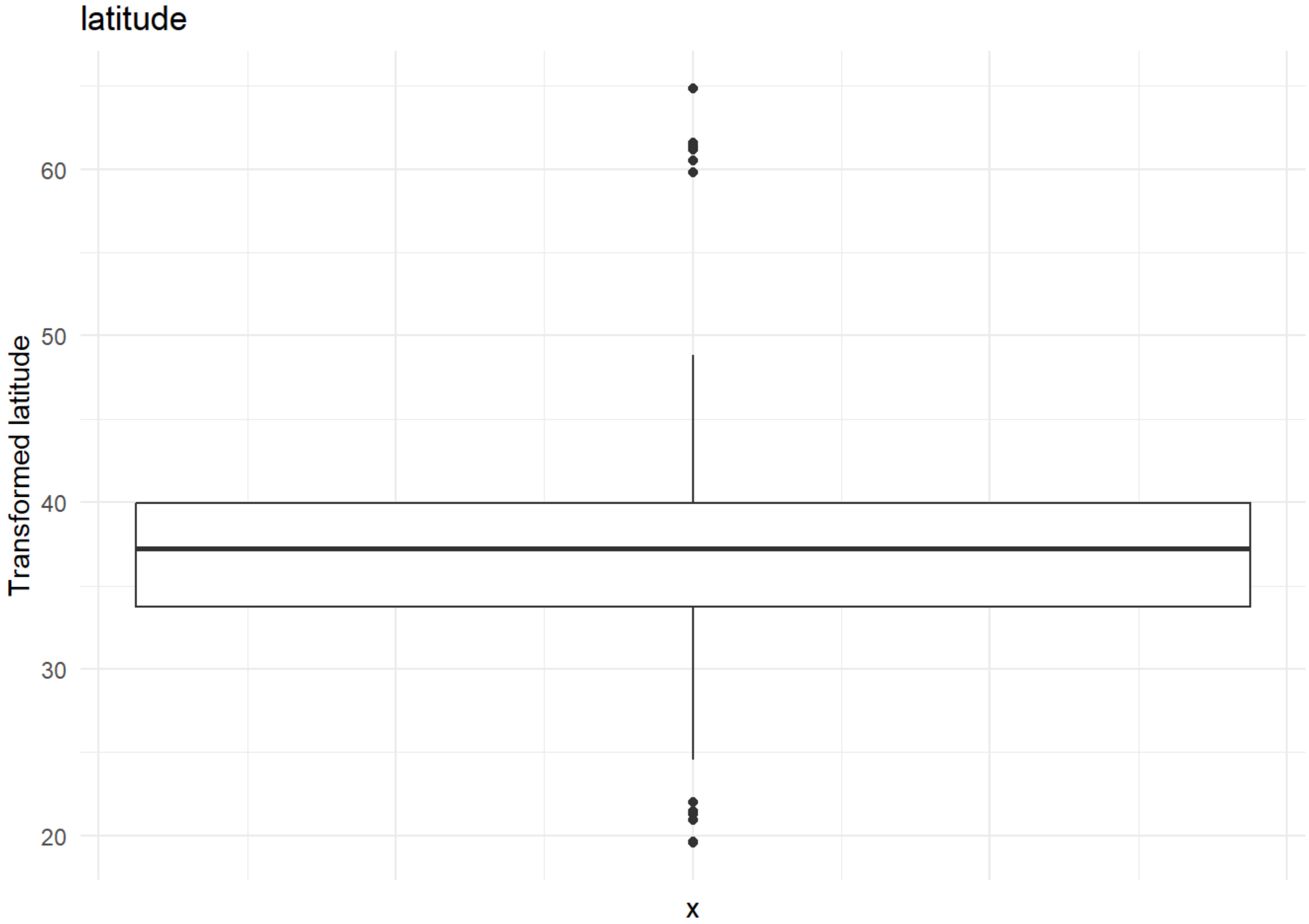
```
## Warning: Removed 1 row containing non-finite outside the scale range
## (`stat_boxplot()`).
```

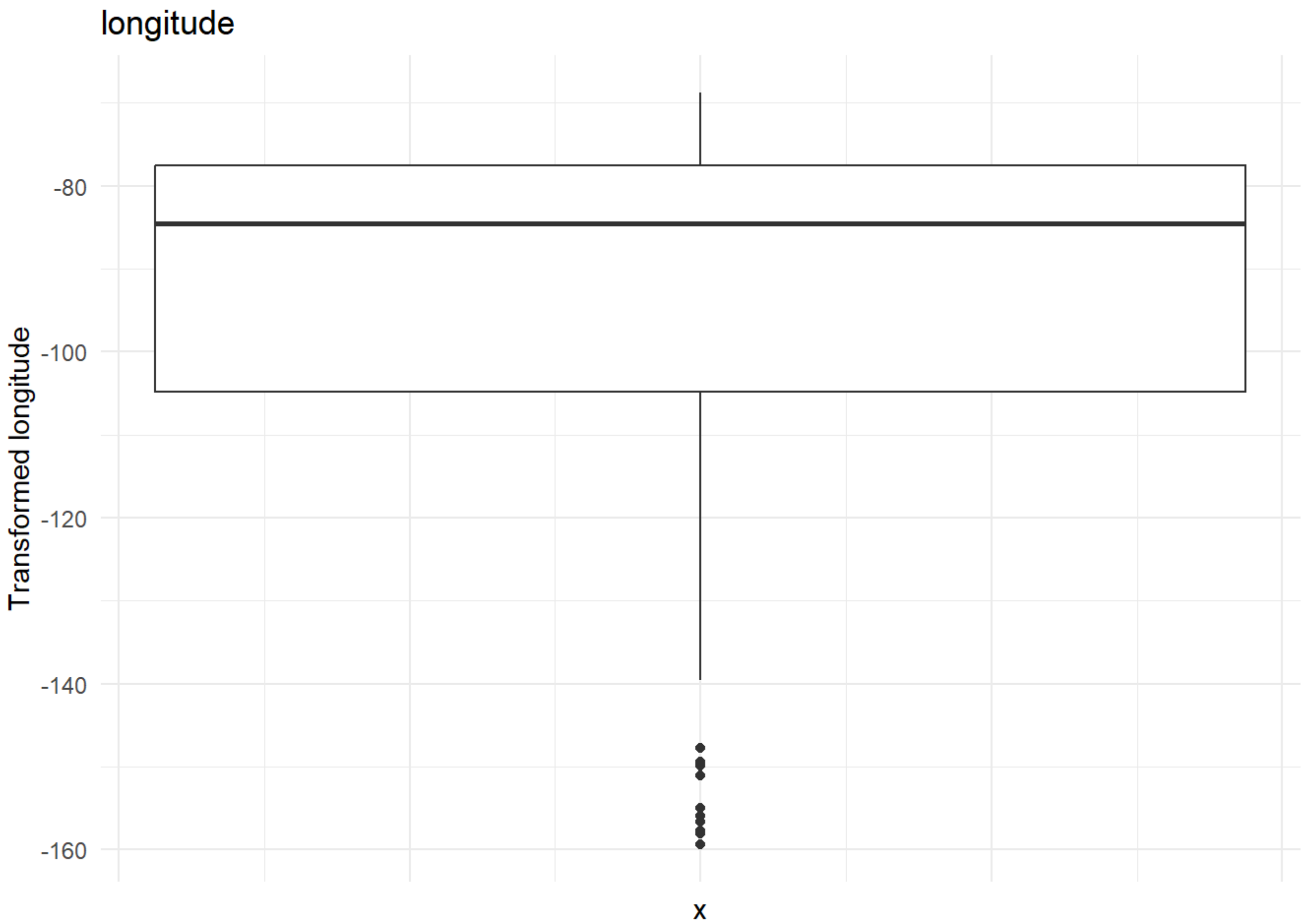



```
## Skipping latitude
```



```
## Skipping longitude
```

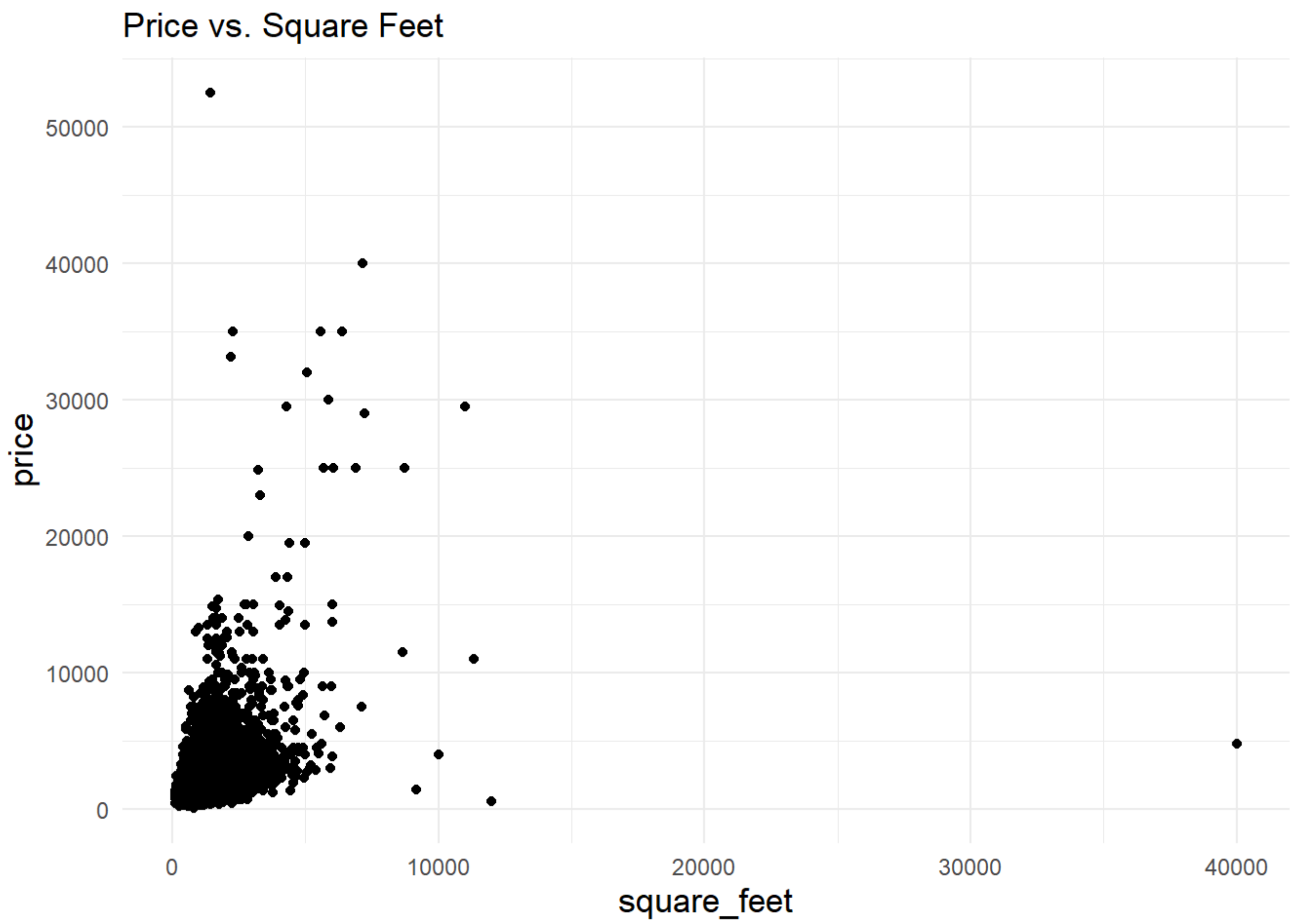




Outlier Detection

```r
library(ggplot2)

# Create the scatter plot
p <- ggplot(data, aes(x = square_feet, y = price)) +
  geom_point() +  # Add points
  labs(x = "square_feet", y = "price", title = "Price vs. Square Feet") +
  theme_minimal() +  # Use a minimal theme
  theme(axis.title.x = element_text(size = 13),  # Customize font size for x label
        axis.title.y = element_text(size = 13))  # Customize font size for y label

# Display the plot
print(p)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



```r
library(dplyr)

out_iqr <- function(df, column) {
  # Calculate the IQR
  q25 <- quantile(df[[column]], 0.25, na.rm = TRUE)
  q75 <- quantile(df[[column]], 0.75, na.rm = TRUE)
  iqr <- q75 - q25

  # Calculate the outlier cutoff
  cut_off <- iqr * 1.5
  lower <- q25 - cut_off
  upper <- q75 + cut_off

  # Output the IQR and bounds
  print(paste("The IQR is", iqr))
  print(paste("The lower bound value is", lower))
  print(paste("The upper bound value is", upper))

  # Calculate the number of outliers
  num_outliers <- sum(df[[column]] < lower | df[[column]] > upper, na.rm = TRUE)

  return(print(paste("Total number of outliers are", num_outliers)))
}
```

- Calculatethe IQR details for ‘price’

```r
out_iqr(df, 'price')
```

```
## [1] "The IQR is 782"
## [1] "The lower bound value is -160"
## [1] "The upper bound value is 2968"
## [1] "Total number of outliers are 4618"
```

- Calculate the IQR details for ‘bedrooms’

```r
out_iqr(df, 'bedrooms')
```

```
## [1] "The IQR is 1"
## [1] "The lower bound value is -0.5"
## [1] "The upper bound value is 3.5"
## [1] "Total number of outliers are 1829"
```

- Calculate he IQR details for ‘bathrooms’

```r
out_iqr(df, 'bathrooms')
```

```
## [1] "The IQR is 1"
## [1] "The lower bound value is -0.5"
## [1] "The upper bound value is 3.5"
## [1] "Total number of outliers are 201"
```

Categorical Variable Analysis

Identifies categorical variables and then counts and sorts the unique values in each of these categorical variables to understand their diversity.

```
library(dplyr)

# Identifying categorical variables (assuming 'O' stands for object type in Python)
cat_vars <- names(data)[sapply(data, function(x) is.character(x))]

# Calculating the number of unique values for each categorical variable
num_unique <- sapply(data[cat_vars], function(x) length(unique(x)))

# Sorting the number of unique values in descending order
sorted_unique <- sort(num_unique, decreasing = TRUE)

# Display the sorted values
sorted_unique
```

```
##       state us_division   us_region
##          51           9           4
```

Rare Label Analysis

- To identify and print categories within each categorical variable that appear in less than 1% of the observations

```
library(dplyr)

analyse_rare_labels <- function(df, var, threshold = 0.01) {
  # Calculate the frequency of each category
  freq <- df %>%
    group_by(!!rlang::sym(var)) %>%
    summarise(Count = n(), .groups = 'drop') %>%
    mutate(Frequency = Count / sum(Count))

  # Identify rare labels
  rare_labels <- freq %>%
    filter(Frequency < threshold)

  # Print or return results
  if (nrow(rare_labels) == 0) {
    message(paste("No rare labels found in", var))
  } else {
    print(rare_labels)
  }

  return(invisible(rare_labels))
}
```

```
# Assuming cat_vars has been defined as shown previously
for (var in cat_vars) {
  print(analyse_rare_labels(data, var, 0.01))
}
```

```
## # A tibble: 30 × 3
##    state Count Frequency
##    <chr> <int>     <dbl>
##  1 AK       58 0.000585
##  2 AL      354 0.00357
##  3 AR      597 0.00602
##  4 CT      509 0.00513
##  5 DC       93 0.000938
##  6 DE        7 0.0000706
##  7 HI       31 0.000313
##  8 IA      372 0.00375
##  9 ID       96 0.000968
## 10 IN      509 0.00513
## # ℹ 20 more rows
## # A tibble: 30 × 3
##    state Count Frequency
##    <chr> <int>     <dbl>
##  1 AK       58 0.000585
##  2 AL      354 0.00357
##  3 AR      597 0.00602
##  4 CT      509 0.00513
##  5 DC       93 0.000938
##  6 DE        7 0.0000706
##  7 HI       31 0.000313
##  8 IA      372 0.00375
##  9 ID       96 0.000968
## 10 IN      509 0.00513
## # ℹ 20 more rows
```

```
## No rare labels found in us_region
```

```
## # A tibble: 0 × 3
## # ℹ 3 variables: us_region <chr>, Count <int>, Frequency <dbl>
```

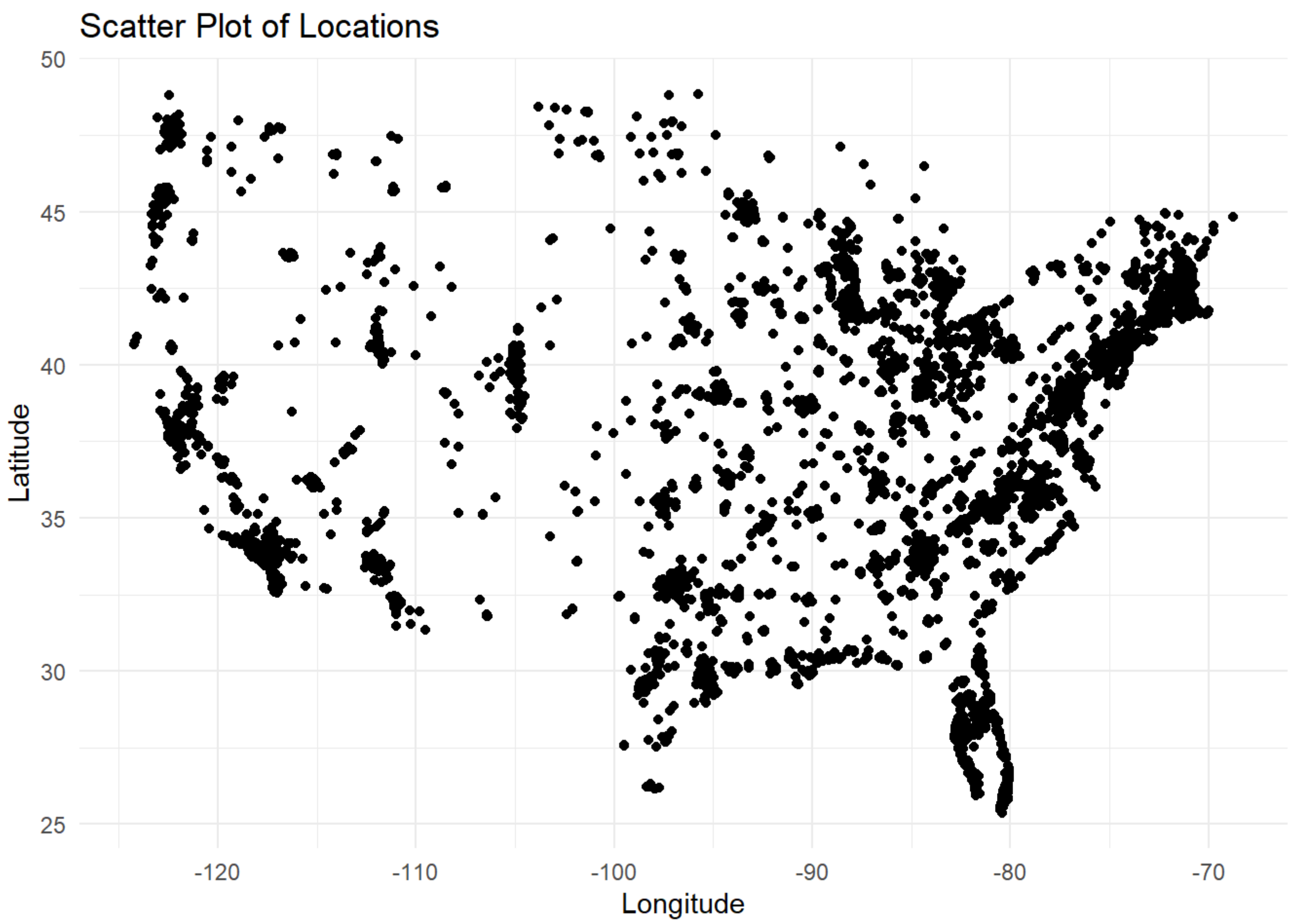
```
## No rare labels found in us_division
```

```
## # A tibble: 0 × 3
## # ℹ 3 variables: us_division <chr>, Count <int>, Frequency <dbl>
```

Geographic Filtering and Visualization

Filters for entries that fall within specific geographic coordinates (likely encompassing the contiguous United States) and a price below $2000.

```
library(dplyr)
library(ggplot2)

# Filter data based on longitude, latitude, and price constraints
df <- data %>%
  filter(longitude > -130, longitude < -60, latitude > 25, latitude < 50, price < 2000)

# Create a scatter plot of longitude vs. latitude
ggplot(df, aes(x = longitude, y = latitude)) +
  geom_point() +  # This adds the scatter plot points
  labs(x = "Longitude", y = "Latitude", title = "Scatter Plot of Locations") +
  theme_minimal()  # Uses a minimal theme for the plot
```



Correlation Analysis

Calculates the correlation matrix for numerical columns in the dataset to identify relationships between different numerical variables.

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.2
```

```
## corrplot 0.92 loaded
```

```
library(reshape2)

# Create a dataframe containing only numeric columns
numeric_df <- data %>%
  select(where(is.numeric))

# Assuming numeric_df is your DataFrame with only numerical columns
corr_matrix <- cor(numeric_df, use = "complete.obs")  # Computes correlation matrix, handling
NA values

# Melt the correlation matrix for ggplot2
melted_corr_matrix <- melt(corr_matrix)

# Plot the heatmap
ggplot(melted_corr_matrix, aes(Var1, Var2, fill = value)) +
  geom_tile() +  # Create tiles for heatmap
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1,
1), space = "Lab", name="Correlation") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels for better
readability
        axis.title = element_blank())  # Remove axis titles
```