Statistikos laboratorinis darbas Nr. 2

VU

2025-03-27

Contents

1. Aprašykite turimus duomenis, nurodykite duomenų šaltinį.

Duomenys atsisiųsti iš https://github.com/valdas-v1/lithuanian-real-estate-listings. Duomenys buvo surinkti 2024 m. vasarį iš https://www.aruodas.lt/ puslapio. Duomenų rinkinyje yra informacija apie parduodamus ir nuomojamus butus, garažus, namus, sklypus ir patalpas.

```
# Set the path to the data directory
data_dir <- "C:/Users/zabit/Documents/GitHub/Statistikos-lab-2/data"

# Get all folder names inside the data directory
folders <- list.dirs(data_dir, full.names = FALSE, recursive = FALSE)

# Print all folder names
print(folders)
```

```
##  [1] "apartments"           "apartments_rent"      "garages_parking"
##  [4] "garages_parking_rent" "house_rent"           "houses"
##  [7] "land"                 "land_rent"            "premises"
## [10] "premises_rent"
```

```
# Check if all folders have the file "all_cities_20240214.csv"
all_have_file <- TRUE
folders_with_file <- 0
folders_missing_file <- character(0)

for (folder in folders) {
  file_path <- file.path(data_dir, folder, "all_cities_20240214.csv")
  if (file.exists(file_path)) {
    folders_with_file <- folders_with_file + 1
  } else {
    all_have_file <- FALSE
    folders_missing_file <- c(folders_missing_file, folder)
  }
}
```

```
# Read the CSV files into a list of dataframes
csv_data_list <- list()
```

```r
for (folder in folders) {
  file_path <- file.path(data_dir, folder, "all_cities_20240214.csv")
  if (file.exists(file_path)) {
    # Try reading the file
    tryCatch({
      df <- read.csv(file_path)
      csv_data_list[[folder]] <- df
      cat("Read:", folder, "with", nrow(df), "rows and", ncol(df), "columns\n")
    }, error = function(e) {
      cat("Error", folder, ":", conditionMessage(e), "\n")
    })
  }
}
```

```
## Read: apartments with 7721 rows and 38 columns
## Read: apartments_rent with 3208 rows and 38 columns
## Read: garages_parking with 497 rows and 28 columns
## Read: garages_parking_rent with 307 rows and 27 columns
## Read: house_rent with 310 rows and 40 columns
## Read: houses with 7284 rows and 39 columns
## Read: land with 6322 rows and 27 columns
## Read: land_rent with 104 rows and 27 columns
## Read: premises with 1556 rows and 37 columns
## Read: premises_rent with 2739 rows and 37 columns
```

```r
# First print the number of rows in each dataset
for (type in names(csv_data_list)) {
  if (!is.null(csv_data_list[[type]])) {
    cat(type, ": ", nrow(csv_data_list[[type]]), " rows\n", sep="")
  }
}
```

```
## apartments: 7721 rows
## apartments_rent: 3208 rows
## garages_parking: 497 rows
## garages_parking_rent: 307 rows
## house_rent: 310 rows
## houses: 7284 rows
## land: 6322 rows
## land_rent: 104 rows
## premises: 1556 rows
## premises_rent: 2739 rows
```

```r
# Remove rows with extreme prices (price > 50,000,000 or price < 20)
for (type in names(csv_data_list)) {
  if (!is.null(csv_data_list[[type]]) && "price" %in% colnames(csv_data_list[[type]])) {
    # Count rows with extreme prices
    extreme_high <- sum(csv_data_list[[type]]$price > 50000000, na.rm = TRUE)
    extreme_low <- sum(csv_data_list[[type]]$price < 20, na.rm = TRUE)
    extreme_prices <- extreme_high + extreme_low

    if (extreme_prices > 0) {
```

```
      # Store original row count
      original_count <- nrow(csv_data_list[[type]])

      # Remove rows with extreme prices, keep rows where price is within range or NA
      csv_data_list[[type]] <- csv_data_list[[type]][(csv_data_list[[type]]$price >= 20 &
                                                      csv_data_list[[type]]$price <= 50000000) |
                                                      is.na(csv_data_list[[type]]$price), ]

      # Verify how many rows were removed
      new_count <- nrow(csv_data_list[[type]])
      removed_count <- original_count - new_count

      cat(type, ": Removed ", removed_count, " rows with extreme prices (", extreme_high,
          " high, ", extreme_low, " low)\n", sep="")
    } else {
      cat(type, ": No rows with extreme prices\n", sep="")
    }
  } else if (!is.null(csv_data_list[[type]])) {
    cat(type, ": No price column found\n", sep="")
  }
}
```

```
## apartments: No rows with extreme prices
## apartments_rent: No rows with extreme prices
## garages_parking: No rows with extreme prices
## garages_parking_rent: No rows with extreme prices
## house_rent: No rows with extreme prices
## houses: No rows with extreme prices
## land: No rows with extreme prices
## land_rent: Removed 2 rows with extreme prices (0 high, 2 low)
## premises: Removed 65 rows with extreme prices (64 high, 1 low)
## premises_rent: Removed 146 rows with extreme prices (113 high, 33 low)
```

```
for (type in names(csv_data_list)) {
  if (!is.null(csv_data_list[[type]])) {
    cat(type, ": ", nrow(csv_data_list[[type]]), " rows\n", sep="")
  }
}
```

```
## apartments: 7721 rows
## apartments_rent: 3208 rows
## garages_parking: 497 rows
## garages_parking_rent: 307 rows
## house_rent: 310 rows
## houses: 7284 rows
## land: 6322 rows
## land_rent: 102 rows
## premises: 1491 rows
## premises_rent: 2593 rows
```

```
# Output summary of data
cat("\nSummary of all loaded datasets:\n")
```

```
##
## Summary of all loaded datasets:
```

```
for (folder_name in names(csv_data_list)) {
  cat("\nDataset from folder:", folder_name, "\n")
  cat("Number of rows:", nrow(csv_data_list[[folder_name]]), "\n")
  cat("Number of columns:", ncol(csv_data_list[[folder_name]]), "\n")
  cat("Column names:", paste(colnames(csv_data_list[[folder_name]]), collapse = ", "), "\n")
}
```

```
##
## Dataset from folder: apartments
## Number of rows: 7721
## Number of columns: 38
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: apartments_rent
## Number of rows: 3208
## Number of columns: 38
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: garages_parking
## Number of rows: 497
## Number of columns: 28
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: garages_parking_rent
## Number of rows: 307
## Number of columns: 27
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: house_rent
## Number of rows: 310
## Number of columns: 40
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: houses
## Number of rows: 7284
## Number of columns: 39
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: land
## Number of rows: 6322
## Number of columns: 27
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: land_rent
## Number of rows: 102
## Number of columns: 27
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: premises
## Number of rows: 1491
## Number of columns: 37
```

```
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
##
## Dataset from folder: premises_rent
## Number of rows: 2593
## Number of columns: 37
## Column names: listing_id, type_id, price, region, microdistrict, street, coordinates, images, descri
```

```r
# Real estate types to analyze
real_estate_types <- c("apartments", "houses", "land", "premises")

for (type in real_estate_types) {
  cat("\n----------\n", toupper(type), "PRICE SUMMARY\n")

  # Check if this real estate type exists in our list
  if (type %in% names(csv_data_list)) {
    df <- csv_data_list[[type]]

    # Check if price column exists
    if ("price" %in% colnames(df)) {
      # Extract price data
      prices <- df$price

      # Generate summary statistics
      cat("Number of observations:", length(prices), "\n")
      cat("Summary statistics:\n")
      print(summary(prices))

      # Additional statistics
      cat("\nStandard deviation:", sd(prices, na.rm = TRUE), "\n")
    } else {
      cat("No 'price' column found in", type, "dataset.\n")
    }
  } else {
    cat("No data available for", type, "real estate type.\n")
  }
}
```

```
##
## ----------
##  APARTMENTS PRICE SUMMARY
## Number of observations: 7721
## Summary statistics:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      43   64000  107558  143718  172000 2500000
##
## Standard deviation: 146129.7
##
## ----------
##  HOUSES PRICE SUMMARY
## Number of observations: 7284
## Summary statistics:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     200   55000  140000  183734  235000 4200000
##
```

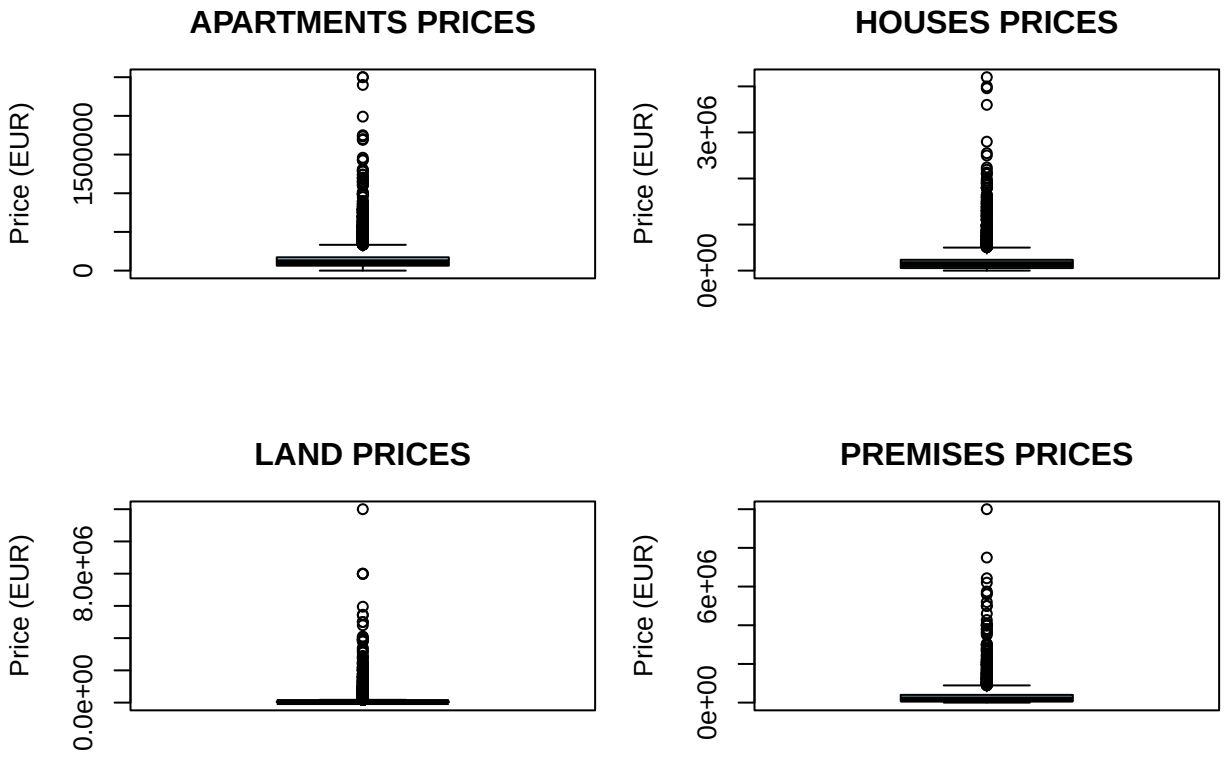
```
## Standard deviation: 223884.9
##
## ----------
##  LAND PRICE SUMMARY
## Number of observations: 6322
## Summary statistics:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      100    18000    35000   115389    79900 12000000
##
## Standard deviation: 386437.4
##
## ----------
##  PREMISES PRICE SUMMARY
## Number of observations: 1491
## Summary statistics:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##      490    70000   165000   413170   399850 10000000
##
## Standard deviation: 762212.4
```

```r
# Load ggplot2 for better visualization
if(!require(ggplot2)) install.packages("ggplot2")
```

```
## Loading required package: ggplot2
```

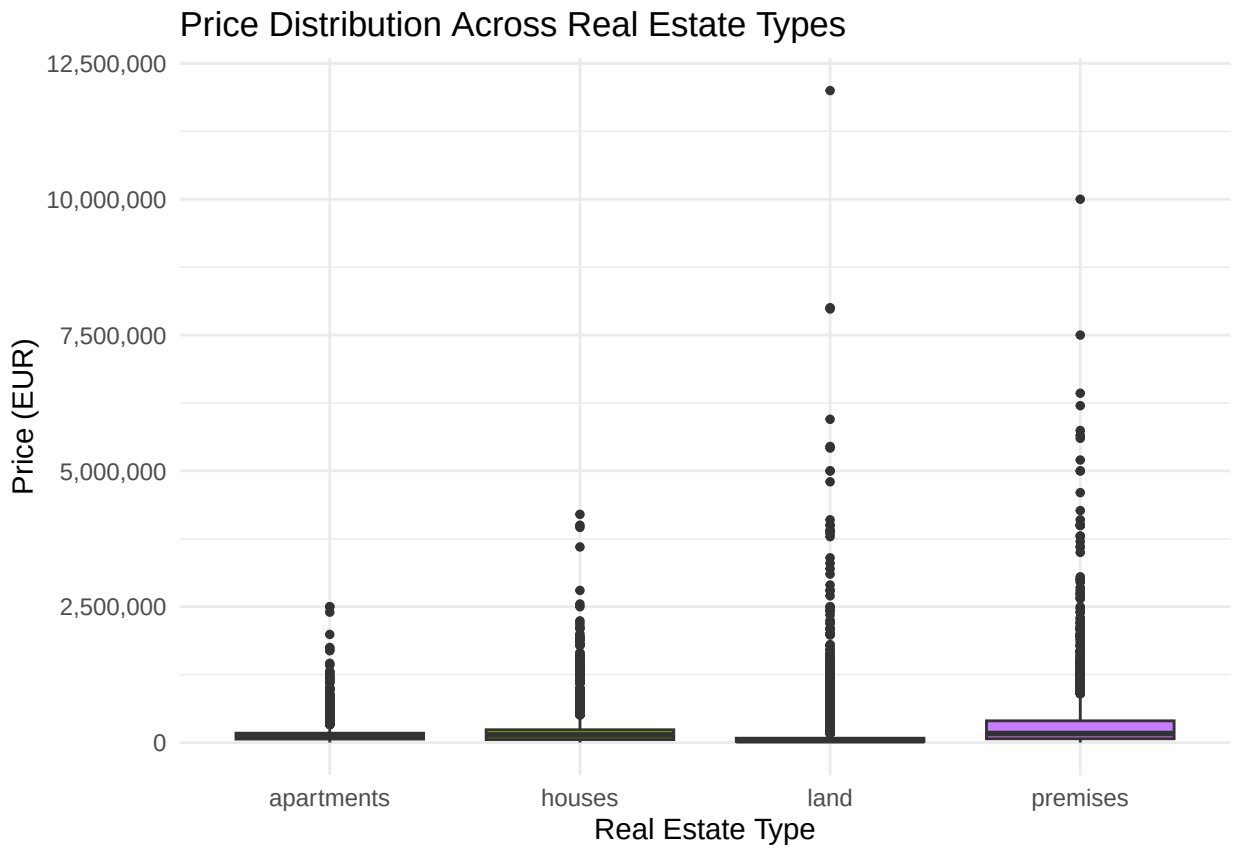
```r
library(ggplot2)

# Setting up a 2x2 panel for the plots
par(mfrow = c(2, 2), mar = c(4, 4, 3, 1))

# Real estate types to analyze
real_estate_types <- c("apartments", "houses", "land", "premises")

# Loop through each type and create box plots
for (type in real_estate_types) {
  if (type %in% names(csv_data_list) && "price" %in% colnames(csv_data_list[[type]])) {
    # Get price data
    prices <- csv_data_list[[type]]$price

    # Basic box plot
    boxplot(prices, main = paste(toupper(type), "PRICES"),
            ylab = "Price (EUR)", col = "lightblue",
            outline = TRUE, # Show outliers
            na.rm = TRUE)

    # Alternative log-scale box plot (prices often have skewed distributions)
    # Uncomment below if needed
    # boxplot(log10(prices[prices > 0]), main = paste(toupper(type), "PRICES (Log Scale)"),
    #         ylab = "Log10(Price)", col = "lightgreen", outline = TRUE, na.rm = TRUE)
  } else {
    # Create an empty plot with a message if data isn't available
    plot(1, type = "n", xlab = "", ylab = "",
         main = paste(toupper(type), "- No data available"))
```

```
    text(1, 1, "No price data available", col = "red")
  }
}
```




```
# Reset the plot layout
par(mfrow = c(1, 1))

# Alternative: Create a more sophisticated plot with ggplot2
# This creates a single plot with all real estate types for comparison
price_data <- data.frame()

for (type in real_estate_types) {
  if (type %in% names(csv_data_list) && "price" %in% colnames(csv_data_list[[type]])) {
    # Extract prices and create a data frame
    temp_data <- data.frame(
      price = csv_data_list[[type]]$price,
      type = rep(type, length(csv_data_list[[type]]$price))
    )
    price_data <- rbind(price_data, temp_data)
  }
}

# Create combined box plot if we have data
if (nrow(price_data) > 0) {
  ggplot(price_data, aes(x = type, y = price, fill = type)) +
```

```
    geom_boxplot(outlier.size = 1) +
    scale_y_continuous(labels = scales::comma) +
    labs(title = "Price Distribution Across Real Estate Types",
         x = "Real Estate Type",
         y = "Price (EUR)") +
    theme_minimal() +
    theme(legend.position = "none")
}
```



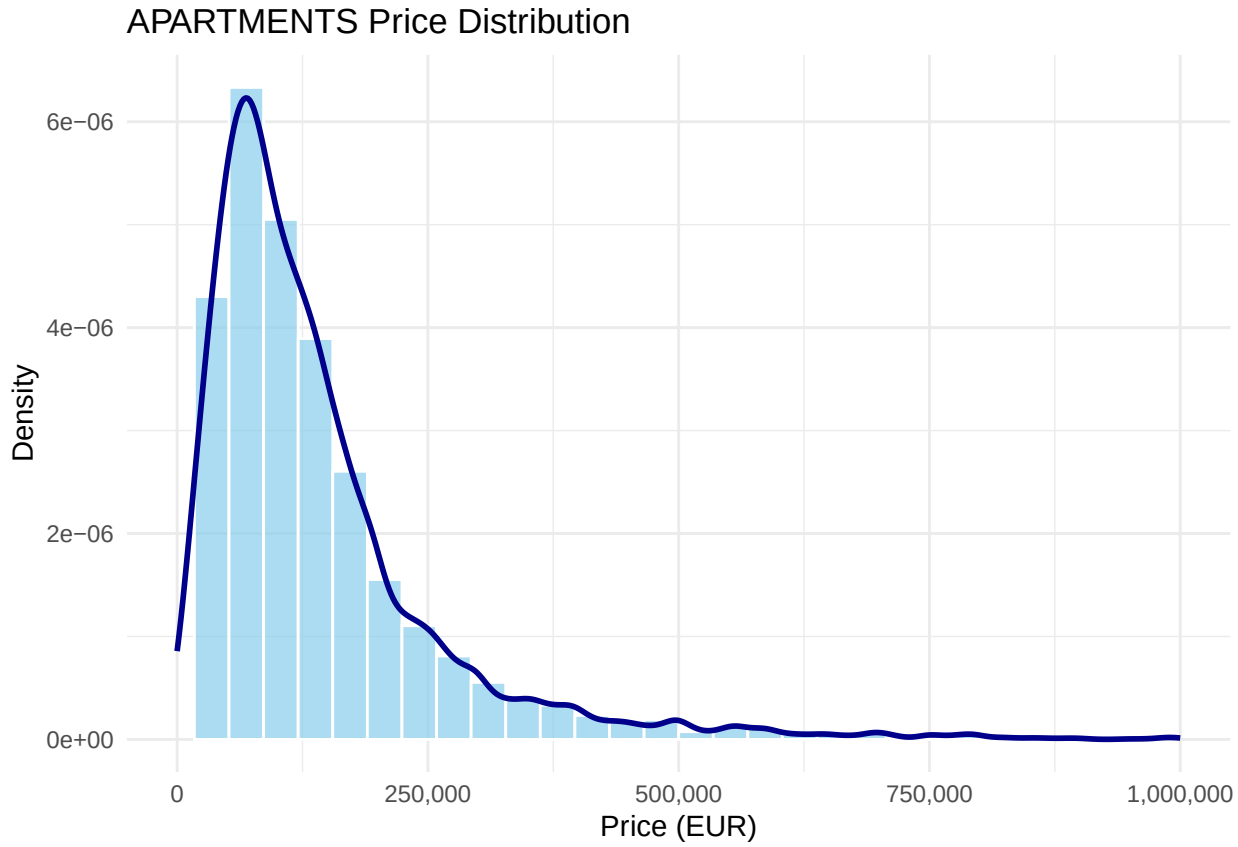

```
# Reset the plot layout
par(mfrow = c(1, 1))

# Alternative: Create more sophisticated histograms with ggplot2 that use density curves
# This might be better for highly skewed real estate price data
for (type in real_estate_types) {
  if (type %in% names(csv_data_list) && "price" %in% colnames(csv_data_list[[type]])) {
    # Get price data and create a data frame
    df <- data.frame(price = csv_data_list[[type]]$price)

    # Create plot object with fixed deprecated features
    p <- ggplot(df, aes(x = price)) +
      geom_histogram(aes(y = after_stat(density)),
                     bins = 30,
                     fill = "skyblue",
                     color = "white",
```

```
                    alpha = 0.7) +
      geom_density(color = "darkblue", linewidth = 1) + # Fixed: size -> linewidth
      labs(title = paste(toupper(type), "Price Distribution"),
           x = "Price (EUR)",
           y = "Density") +
      theme_minimal() +
      scale_x_continuous(labels = scales::comma, limits = c(0, 1000000)) +
      coord_cartesian(xlim = c(0, 1000000))

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

```
## Warning: Removed 28 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

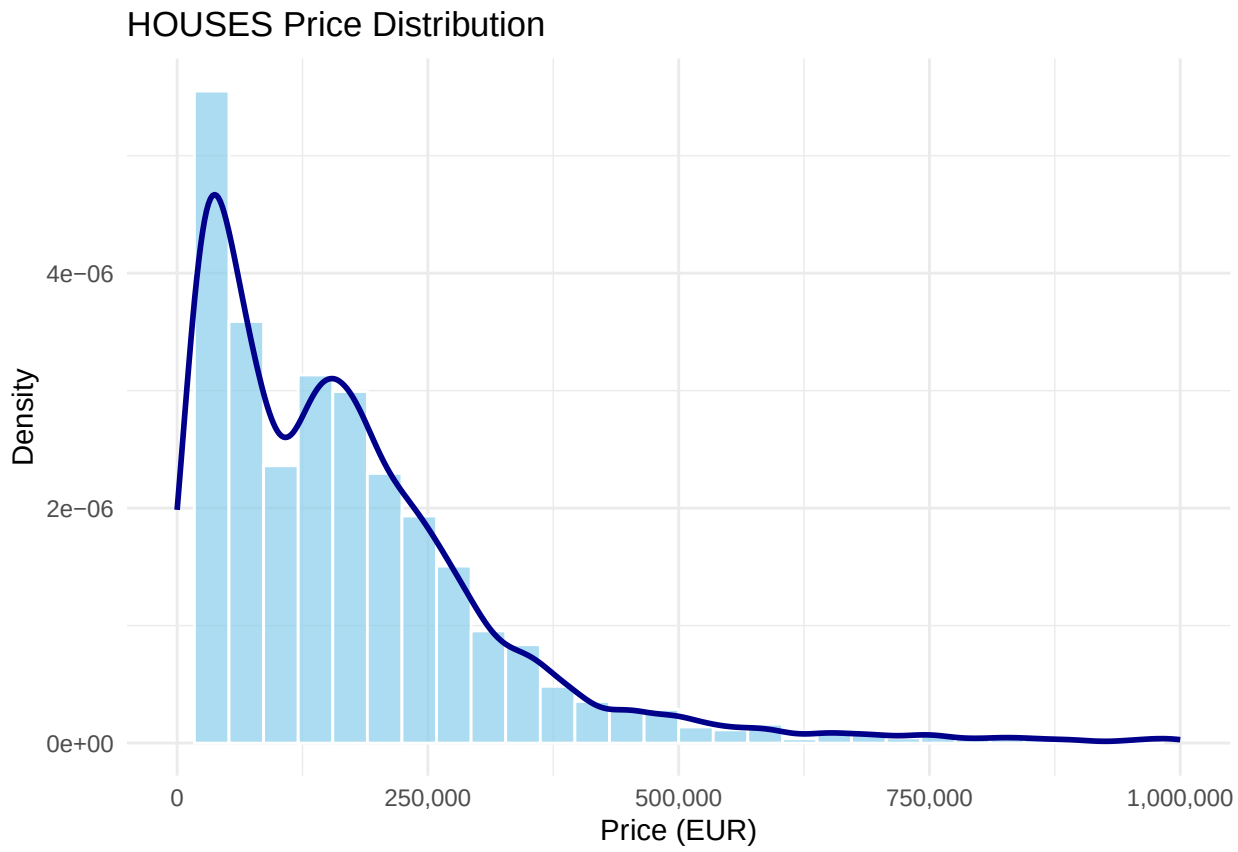
```
## Warning: Removed 28 rows containing non-finite outside the scale range
## (`stat_density()`).
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_bar()`).
```

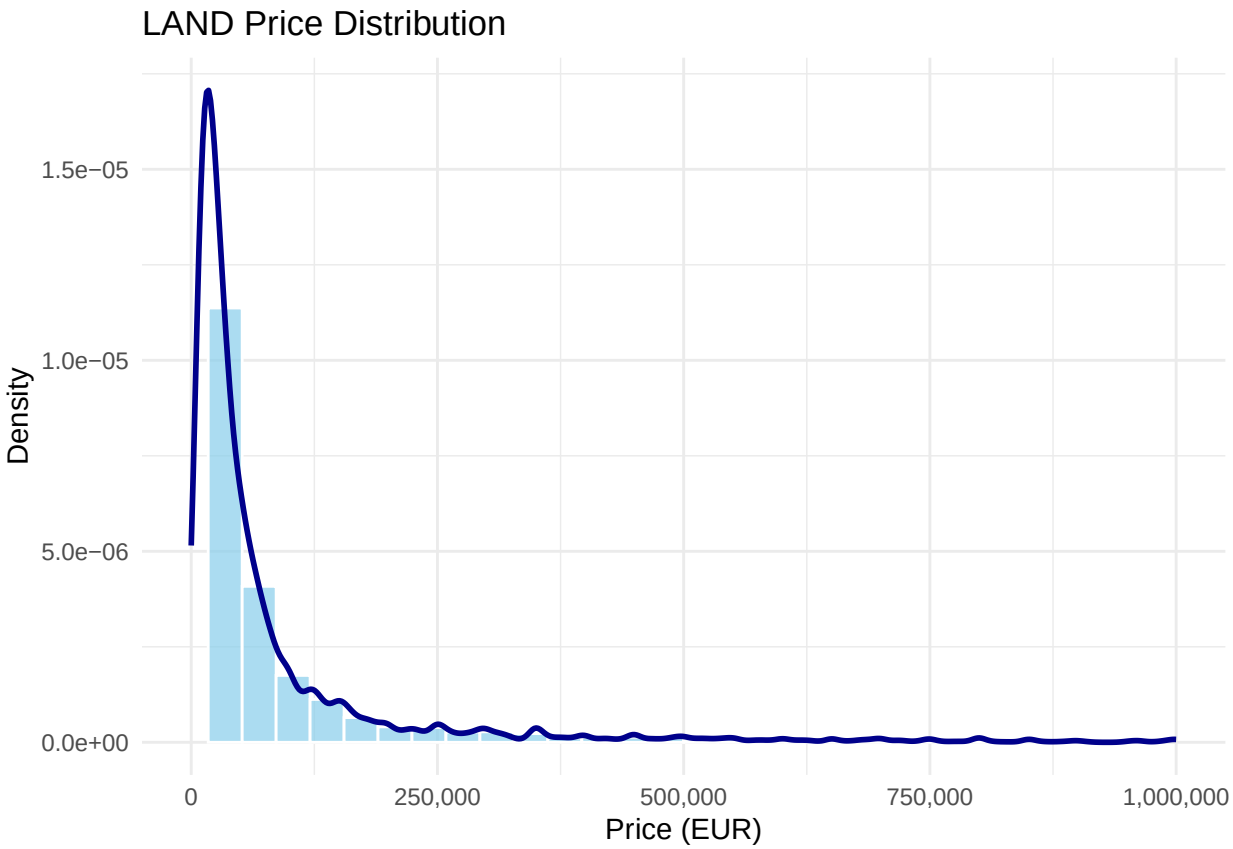



```
## Warning: Removed 83 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

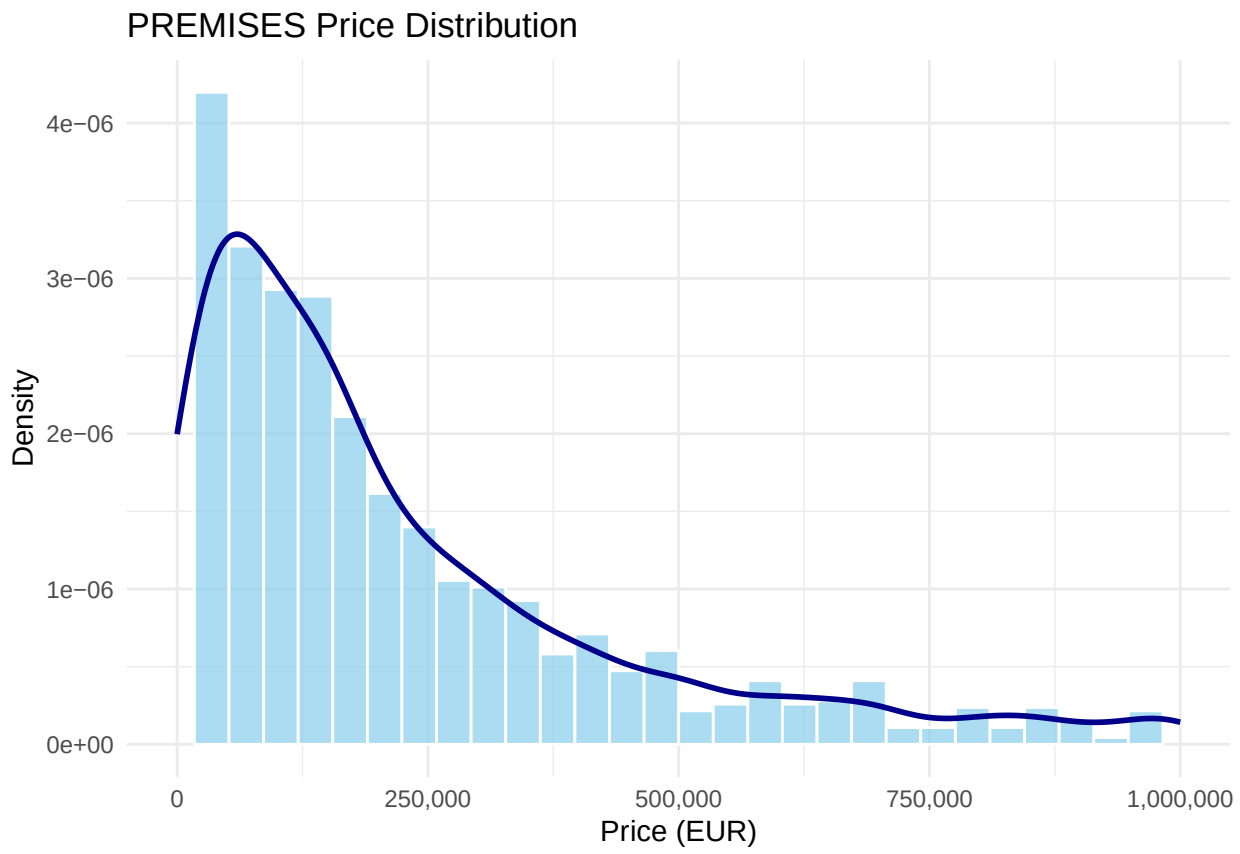
```
## Warning: Removed 83 rows containing non-finite outside the scale range
## (‘stat_density()‘).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_bar()‘).
```




```
## Warning: Removed 93 rows containing non-finite outside the scale range
## (‘stat_bin()‘).

## Warning: Removed 93 rows containing non-finite outside the scale range
## (‘stat_density()‘).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_bar()‘).
```



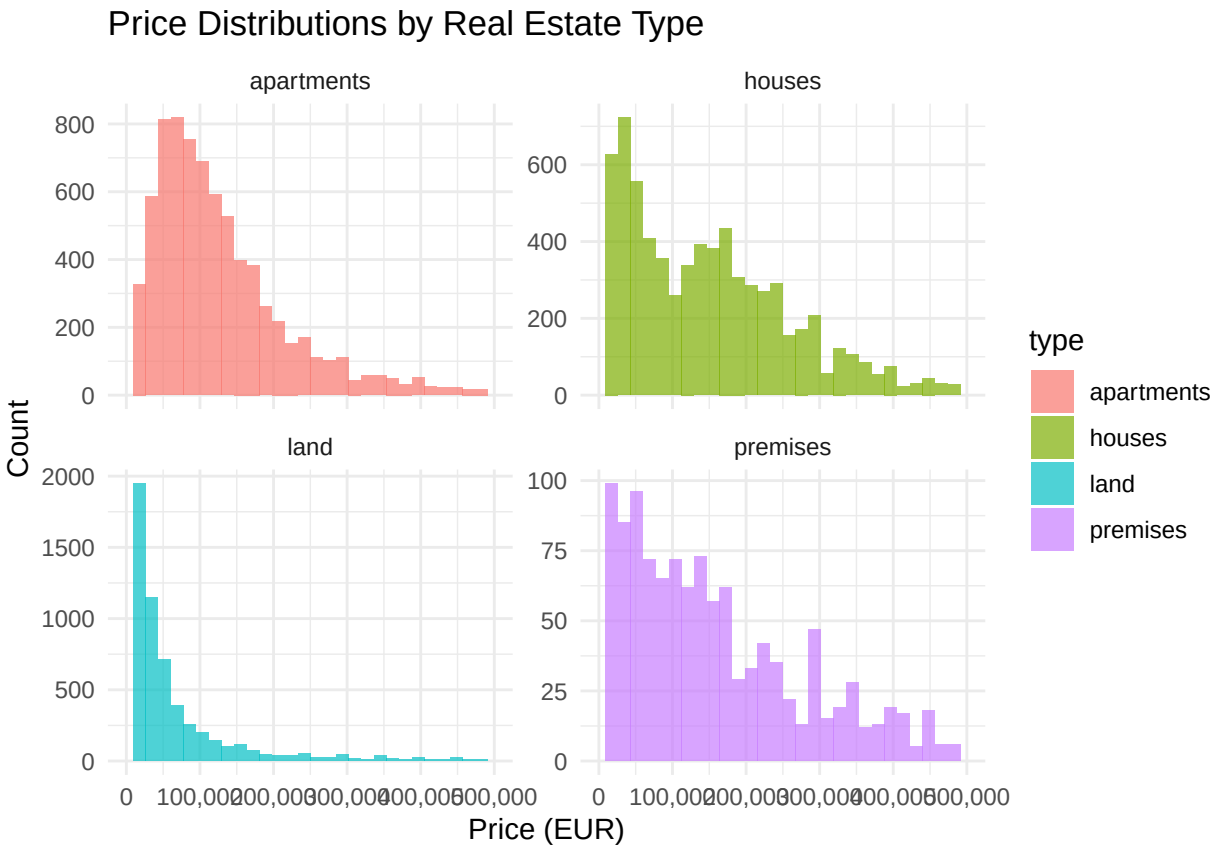

```
## Warning: Removed 144 rows containing non-finite outside the scale range
## (‘stat_bin()‘).

## Warning: Removed 144 rows containing non-finite outside the scale range
## (‘stat_density()‘).

## Warning: Removed 2 rows containing missing values or values outside the scale range
## (‘geom_bar()‘).
```




```
# Create faceted histograms in one figure (comparing all types)
if (nrow(price_data) > 0) {
  ggplot(price_data, aes(x = price, fill = type)) +
    geom_histogram(bins = 30, alpha = 0.7) +
    facet_wrap(~type, scales = "free_y") +
    labs(title = "Price Distributions by Real Estate Type",
         x = "Price (EUR)",
         y = "Count") +
    theme_minimal() +
    scale_x_continuous(labels = scales::comma, limits = c(0, 500000)) +
    coord_cartesian(xlim = c(0, 500000))
}
```

```
## Warning: Removed 1079 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```

```
## Warning: Removed 8 rows containing missing values or values outside the scale range
## (‘geom_bar()‘).
```



2. Išbrėžkite turimų duomenų grafikus (parinkite tinkamiausius). Manau kokių 4 užtektų
3. Apskaičiuokite pagrindines skaitines charakteristikas kiekybiniams kintamiesiems. Vidurkis (Mean), Mediana (Median), Moda (Mode), Dispersija (Variance), Standartinis nuokrypis (Standard Deviation), Kvartiliai (Quartiles), Tarpkvartilinis plotis (Interquartile Range, IQR), Diapazonas (Range, max-min) Kiekybiniai duomenys: kaina ("price"), aukštų skaičius, peržiūrų skaičius, namo, buto dydis ("area")
4. Sudarykite dažnių lenteles kategoriniams kintamiesiems.
5. Suformuluokite bent 6 tyrimo hipotezes iš savo duomenų rinkinio
6. Užrašykite kokius testus parinkote savo tyrimo hipotezėms. Hipotezės turi būti skirtos skirtingų testų naudojimui. Jei reikia susikurkite naujus kintamuosius iš turimų duomenų.
7. Patikrinkite, ar kintamieji tenkina būtinas sąlygas testų taikymui. Jei netenkina, atlikite duomenų trans- formacijas.
8. Atlikite statistinį tyrimą savo suformuluotoms hipotezėms.
9. Pateikite tyrimo atsakymą.