Perth School Distance and House Price GWR

2025-05-09

1. Load packages and dataframe

1.1 Load packages

```
# Data processing
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
```

```
library(tidyr)
library(readr)
```

```
## Warning: package 'readr' was built under R version 4.3.3
```

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v purrr     1.0.1
## v ggplot2   3.4.2     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
```

```
## -- 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 errors
```

```
#Data visualization
library(patchwork)
```

```
## Warning: package 'patchwork' was built under R version 4.3.3
```

2.1 Load Data and Initial Inspection

Check the dataframe types and values

```
glimpse(house)
```

```
## Rows: 33,656
## Columns: 19
## $ ADDRESS          <chr> "1 Acorn Place", "1 Addis Way", "1 Ainsley Court", "1~
## $ SUBURB           <chr> "South Lake", "Wandi", "Camillo", "Bellevue", "Lockri~
## $ PRICE            <int> 565000, 365000, 287000, 255000, 325000, 409000, 40000~
## $ BEDROOMS         <int> 4, 3, 3, 2, 4, 4, 3, 4, 4, 3, 4, 3, 3, 4, 4, 3, 4, 4,~
## $ BATHROOMS        <int> 2, 2, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,~
## $ GARAGE           <chr> "2", "2", "1", "2", "2", "1", "2", "2", "3", "8", "6"~
## $ LAND_AREA        <int> 600, 351, 719, 651, 466, 759, 386, 468, 875, 552, 253~
## $ FLOOR_AREA       <int> 160, 139, 86, 59, 131, 118, 132, 158, 168, 126, 241, ~
## $ BUILD_YEAR       <chr> "2003", "2013", "1979", "1953", "1998", "1991", "2014~
## $ CBD_DIST         <int> 18300, 26900, 22600, 17900, 11200, 27300, 28200, 4170~
## $ NEAREST_STN      <chr> "Cockburn Central Station", "Kwinana Station", "Chall~
## $ NEAREST_STN_DIST <int> 1800, 4900, 1900, 3600, 2000, 1000, 3700, 1100, 2500,~
## $ DATE_SOLD        <chr> "09-2018\n", "02-2019\n", "06-2015\n", "07-2018\n", "~
## $ POSTCODE         <int> 6164, 6167, 6111, 6056, 6054, 6112, 6112, 6169, 6022,~
## $ LATITUDE         <dbl> -32.11590, -32.19347, -32.12058, -31.90055, -31.88579~
## $ LONGITUDE        <dbl> 115.8424, 115.8596, 115.9936, 116.0380, 115.9478, 116~
## $ NEAREST_SCH      <chr> "LAKELAND SENIOR HIGH SCHOOL", "ATWELL COLLEGE", "KEL~
## $ NEAREST_SCH_DIST <dbl> 0.8283386, 5.5243244, 1.6491782, 1.5714009, 1.5149216~
## $ NEAREST_SCH_RANK <int> NA, 129, 113, NA, NA, NA, NA, NA, NA, 29, NA, 39, NA,~
```

Check the early values

```
head(house)
```

```
##               ADDRESS       SUBURB  PRICE BEDROOMS BATHROOMS GARAGE LAND_AREA
## 1       1 Acorn Place   South Lake 565000        4         2      2       600
## 2         1 Addis Way        Wandi 365000        3         2      2       351
## 3     1 Ainsley Court      Camillo 287000        3         1      1       719
## 4     1 Albert Street     Bellevue 255000        2         1      2       651
## 5       1 Aman Place    Lockridge 325000        4         1      2       466
## 6 1 Amethyst Crescent Mount Richon 409000        4         2      1       759
##   FLOOR_AREA BUILD_YEAR CBD_DIST              NEAREST_STN NEAREST_STN_DIST
## 1        160       2003    18300 Cockburn Central Station             1800
## 2        139       2013    26900          Kwinana Station             4900
## 3         86       1979    22600          Challis Station             1900
## 4         59       1953    17900          Midland Station             3600
## 5        131       1998    11200       Bassendean Station             2000
## 6        118       1991    27300         Armadale Station             1000
##   DATE_SOLD POSTCODE  LATITUDE LONGITUDE                  NEAREST_SCH
## 1 09-2018\n     6164 -32.11590  115.8424  LAKELAND SENIOR HIGH SCHOOL
## 2 02-2019\n     6167 -32.19347  115.8596               ATWELL COLLEGE
## 3 06-2015\n     6111 -32.12058  115.9936 KELMSCOTT SENIOR HIGH SCHOOL
## 4 07-2018\n     6056 -31.90055  116.0380 SWAN VIEW SENIOR HIGH SCHOOL
## 5 11-2016\n     6054 -31.88579  115.9478                KIARA COLLEGE
## 6 03-2013\n     6112 -32.15380  116.0237  ARMADALE SENIOR HIGH SCHOOL
```

```
##   NEAREST_SCH_DIST NEAREST_SCH_RANK
## 1        0.8283386               NA
## 2        5.5243244              129
## 3        1.6491782              113
## 4        1.5714009               NA
## 5        1.5149216               NA
## 6        1.2272192               NA
```

The house price data consist of sold houses throughout the years. I need to review the time range.

```
summary(house)
```

```
##    ADDRESS             SUBURB              PRICE            BEDROOMS
##  Length:33656       Length:33656       Min.   :  51000   Min.   : 1.000
##  Class :character   Class :character   1st Qu.: 410000   1st Qu.: 3.000
##  Mode  :character   Mode  :character   Median : 535500   Median : 4.000
##                                        Mean   : 637072   Mean   : 3.659
##                                        3rd Qu.: 760000   3rd Qu.: 4.000
##                                        Max.   :2440000   Max.   :10.000
##
##    BATHROOMS         GARAGE            LAND_AREA        FLOOR_AREA
##  Min.   : 1.000   Length:33656       Min.   :    61   Min.   :  1.0
##  1st Qu.: 1.000   Class :character   1st Qu.:   503   1st Qu.:130.0
##  Median : 2.000   Mode  :character   Median :   682   Median :172.0
##  Mean   : 1.823                      Mean   :  2741   Mean   :183.5
##  3rd Qu.: 2.000                      3rd Qu.:   838   3rd Qu.:222.2
##  Max.   :16.000                      Max.   :999999   Max.   :870.0
##
##   BUILD_YEAR           CBD_DIST     NEAREST_STN        NEAREST_STN_DIST
##  Length:33656       Min.   :  681   Length:33656       Min.   :   46
##  Class :character   1st Qu.:11200   Class :character   1st Qu.: 1800
##  Mode  :character   Median :17500   Mode  :character   Median : 3200
##                     Mean   :19777                      Mean   : 4523
##                     3rd Qu.:26600                      3rd Qu.: 5300
##                     Max.   :59800                      Max.   :35500
##
##   DATE_SOLD            POSTCODE       LATITUDE        LONGITUDE
##  Length:33656       Min.   :6003   Min.   :-32.47   Min.   :115.6
##  Class :character   1st Qu.:6050   1st Qu.:-32.07   1st Qu.:115.8
##  Mode  :character   Median :6069   Median :-31.93   Median :115.9
##                     Mean   :6089   Mean   :-31.96   Mean   :115.9
##                     3rd Qu.:6150   3rd Qu.:-31.84   3rd Qu.:116.0
##                     Max.   :6558   Max.   :-31.46   Max.   :116.3
##
##  NEAREST_SCH        NEAREST_SCH_DIST   NEAREST_SCH_RANK
##  Length:33656       Min.   : 0.07091   Min.   :  1.00
##  Class :character   1st Qu.: 0.88057   1st Qu.: 39.00
##  Mode  :character   Median : 1.34552   Median : 68.00
##                     Mean   : 1.81527   Mean   : 72.67
##                     3rd Qu.: 2.09722   3rd Qu.:105.00
##                     Max.   :23.25437   Max.   :139.00
##                                        NA's   :10952
```

The "DATE_SOLD" column is a character so I need to transform it to numeric and extract the year first.

```r
## Extract year
house$YEAR_SOLD<- substr(house$DATE_SOLD, 4, 8)

## Change it into numeric
house$YEAR_SOLD <- as.numeric(house$YEAR_SOLD)
```

```r
summary(house$YEAR_SOLD)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1988    2015    2017    2017    2019    2020
```

- The sold range is from 1988 to 2020.

```r
year_count <- aggregate(data.frame(count = house$YEAR_SOLD), list(value = house$YEAR_SOLD), length)

year_count
```

```
##    value count
## 1   1988     9
## 2   1989     8
## 3   1990     7
## 4   1991    12
## 5   1992    11
## 6   1993    12
## 7   1994    21
## 8   1995    22
## 9   1996    17
## 10  1997    22
## 11  1998    41
## 12  1999    32
## 13  2000    36
## 14  2001    52
## 15  2002    85
## 16  2003    86
## 17  2004    74
## 18  2005   115
## 19  2006   133
## 20  2007   143
## 21  2008   141
## 22  2009   242
## 23  2010   271
## 24  2011   362
## 25  2012   615
## 26  2013  1178
## 27  2014  1947
## 28  2015  3050
## 29  2016  3901
## 30  2017  4844
## 31  2018  5621
## 32  2019  5285
## 33  2020  5261
```

Display the mode

```
year_count[which.max(year_count$count),]
```

```
##    value count
## 31  2018  5621
```

- Most of the entry comes from houses sold in 2018.

2. Clean Data

Group up data for EDA analysis

```
all_cols <- colnames(house)

all_cols
```

```
##  [1] "ADDRESS"          "SUBURB"           "PRICE"            "BEDROOMS"
##  [5] "BATHROOMS"        "GARAGE"           "LAND_AREA"        "FLOOR_AREA"
##  [9] "BUILD_YEAR"       "CBD_DIST"         "NEAREST_STN"      "NEAREST_STN_DIST"
## [13] "DATE_SOLD"        "POSTCODE"         "LATITUDE"         "LONGITUDE"
## [17] "NEAREST_SCH"      "NEAREST_SCH_DIST" "NEAREST_SCH_RANK" "YEAR_SOLD"
```

2.1. Check Duplicate Entry

Check duplicated entry

```
sum(duplicated(house))
```

```
## [1] 0
```

- There is no duplicated row entry in the dataframe.

I also need to check for duplicated ID. In this case, it can happen if the same house is sold twice in the dataframe. The dataframe doesn't have a necessary ID column, but it can be represented by the full address (combination of ADDRESS and SUBURB)

```
duplicate_indices <- duplicated(house[, c("ADDRESS", "SUBURB")]) | duplicated(house[, c("ADDRESS", "SUBU

duplicate_rows <- house[duplicate_indices, ]

sum_of_duplicates <- sum(duplicate_rows$value)

print(sum_of_duplicates)
```

```
## [1] 0
```

- There's no repeated address.

Coordinates can also represent As the ID.

```r
## Create coords argument
house <- house %>%
  mutate(COORDS = paste(LONGITUDE, LATITUDE, sep = ","))

## Check repeated coords
coord_counts <- house %>%
  group_by(COORDS) %>%
  summarise(n = n())

duplicate_coords <- coord_counts %>%
  filter(n > 1)

print(duplicate_coords)
```

```
## # A tibble: 250 x 2
##    COORDS                      n
##    <chr>                   <int>
##  1 115.70489,-31.68901         2
##  2 115.70772,-32.29483         2
##  3 115.70784,-32.29384         2
##  4 115.71232,-32.29617         2
##  5 115.73531,-32.2729          2
##  6 115.73897,-31.78222         2
##  7 115.7433,-31.81293          2
##  8 115.74389,-32.29335         2
##  9 115.7441699,-31.77928516    2
## 10 115.7446259,-31.78765901    7
## # i 240 more rows
```

I found 250 rows of repeated coordinates. I need to inspect the row entry to get much better context.

```r
duplicate_rows <- house %>%
  filter(COORDS %in% duplicate_coords$COORDS)

head(duplicate_rows, 10)
```

```
##                  ADDRESS       SUBURB  PRICE BEDROOMS BATHROOMS GARAGE
## 1          1 Corbin Gate      Madeley 427000        3         2      2
## 2          1 Dermer Road Beaconsfield 530000        3         2      2
## 3      1 Fallow Crescent    Spearwood 475000        4         2      2
## 4       1 Fortune Street     Balcatta 460000        3         2      2
## 5          1 Hamlyn Glen        Kiara 405000        3         1      3
## 6      1 Ivythorne Grove        Kiara 210000        3         1      2
## 7  1/1 Coolgardie Street   West Perth 480000        2         2      2
## 8      1/1 French Street     Ashfield 455000        3         2      2
## 9     1/101 Epsom Avenue    Redcliffe 407000        3         1      2
## 10     1/11 Adamson Road    Brentwood 555000        3         1      2
##    LAND_AREA FLOOR_AREA BUILD_YEAR CBD_DIST              NEAREST_STN
## 1        618        134       2005    16300        Greenwood Station
## 2       1319        133       NULL    15800        Fremantle Station
## 3        427        142       1994    19600 Cockburn Central Station
## 4        322        109       1997     8100         Stirling Station
```

```
## 5        975        126       1993    10600         Ashfield Station
## 6        534        125       1992    10500         Ashfield Station
## 7        813         87       2008     1100        City West Station
## 8        345        146       2009     8200         Ashfield Station
## 9       3643        101       1990     7500         Ashfield Station
## 10       774         82       1989    10300       Bull Creek Station
##    NEAREST_STN_DIST DATE_SOLD POSTCODE  LATITUDE LONGITUDE
## 1              4200 03-2020\n     6065 -31.80811  115.8265
## 2              3100 12-2017\n     6162 -32.07189  115.7687
## 3              7700 01-2019\n     6163 -32.11469  115.7777
## 4              2100 07-2018\n     6021 -31.88513  115.8244
## 5              3400 07-2012\n     6054 -31.88204  115.9342
## 6              3400 09-2013\n     6054 -31.88213  115.9338
## 7               493 03-2020\n     6005 -31.94894  115.8477
## 8               306 11-2017\n     6054 -31.91533  115.9340
## 9              3000 04-2015\n     6104 -31.93973  115.9372
## 10              389 12-2014\n     6153 -32.04427  115.8516
##                           NEAREST_SCH NEAREST_SCH_DIST NEAREST_SCH_RANK
## 1           KINGSWAY CHRISTIAN COLLEGE        1.0966909               50
## 2                     FREMANTLE COLLEGE        0.4478663              128
## 3                          PORT SCHOOL        3.5247258               NA
## 4          BALCATTA SENIOR HIGH SCHOOL        1.5724518              135
## 5                        KIARA COLLEGE        0.5180013               NA
## 6                        KIARA COLLEGE        0.5340347               NA
## 7  ST GEORGE'S ANGLICAN GRAMMAR SCHOOL        1.0533886               NA
## 8          CYRIL JACKSON SENIOR CAMPUS        0.8414539               NA
## 9                BELMONT CITY COLLEGE        2.1465059               57
## 10                ALL SAINTS' COLLEGE        1.5666411               21
##    YEAR_SOLD                   COORDS
## 1       2020     115.82654,-31.808107
## 2       2017       115.7687,-32.07189
## 3       2019 115.7777208,-32.11469111
## 4       2018      115.82445,-31.88513
## 5       2012 115.9341911,-31.88204218
## 6       2013      115.93384,-31.88213
## 7       2020 115.8477325,-31.94893602
## 8       2017 115.9340203,-31.91532844
## 9       2015 115.9371684,-31.93973337
## 10      2014      115.85158,-32.04427
```

It doesn't seem to come from repeated observation on the same house, as the address is different.

Check Repeated Coordinates Percentages

```
print((nrow(duplicate_rows)/nrow(house))*100)
```

```
## [1] 1.747088
```

- There is 1.74% of observation that have repeated coordinates.
- The duplicated entries could be caused by entry error, but since the exact cause is not yet known then the duplicated observation will be omitted.

Remove duplicate coordinates

```r
house <- house %>%
  anti_join(duplicate_rows, by="COORDS")
```

2.2. Check for NA entries

Check NA entries

```r
sum(is.na(house))
```

```
## [1] 10774
```

- There's 10952 Na entries.

```r
print("Columns with NA value")
```

```
## [1] "Columns with NA value"
```

```r
for (col in all_cols) {
  n_na <- sum(is.na(house[[col]]))
  if (n_na > 0) {
  print(paste0("Number and percentages of Na in ", col))
  print(n_na)
  print((n_na/(nrow(house)))*100)
  }
}
```

```
## [1] "Number and percentages of Na in NEAREST_SCH_RANK"
## [1] 10774
## [1] 32.58135
```

- The NA's value is only present in the "NEAREST_SCH_RANK".
- According to the data source, "NEAREST_SCH_RANK" missing values is because many of the schools are still not ranked.
- NA value made up 32% of the total entry in "NEAREST_SCH_RANK", which is substantial if all observation with NA gets omitted.

For further analysis: - To get the effect of ranked and unranked school, a new dummy will be generated indicated that the nearest school have a ranking ("NEAREST_SCH_IS_RANKED"). - There will be two dataframe: the 'house1' that doesn't include "NEAREST_SCH_RANK" column, and house_a that omitted the NA's observation.

Generate the dummy variable

```r
house$NEAREST_SCH_IS_RANKED <- ifelse(is.na(house$NEAREST_SCH_RANK), 0, 1)

## Add to the all columns list
new_char <- "NEAREST_SCH_IS_RANKED"
all_cols <- append(all_cols, new_char)

# Remove the NEAREST_SCH_RANK
remove_char <- "NEAREST_SCH_RANK"

all_cols <- all_cols[!(all_cols %in% remove_char)]
```

Save dataframe

```
## Without column
house1 <- select(house, -NEAREST_SCH_RANK)

## Without NA's
house_a <- house[complete.cases(house),]
```

Recheck NA

```
print("Columns with NA value")
```

```
## [1] "Columns with NA value"
```

```
for (col in all_cols) {
  n_na <- sum(is.na(house1[[col]]))
  if (n_na > 0) {
  print(paste0("Number and percentages of Na in ", col))
  print(n_na)
  print((n_na/(nrow(house)))*100)
  }
}
```

2.3. Check empty string value (“ “)

```
print("Columns with empty string value")
```

```
## [1] "Columns with empty string value"
```

```
for (col in all_cols) {
  n_empty <- sum(house1[[col]]=="")
  if (n_empty > 0) {
  print(paste0("Number and percentages of empty string value in ", col))
  print(n_empty)
  print((n_empty/(nrow(house1)))*100)
  }
}
```

- There’s no empty string value

2.3. Check “NULL” entry

```
print("Columns with NULL string value")
```

```
## [1] "Columns with NULL string value"
```

```
for (col in all_cols) {
  n_null <- sum(is.null(house1[[col]]))
  if (n_null > 0) {
  print(paste0("Number and percentages of NULL value in ", col))
  print(n_null)
  print((n_null/(nrow(house1)))*100)
  }
}
```

- There is no NULL value.

```
print("Columns with NULL string value")
```

```
## [1] "Columns with NULL string value"
```

```
for (col in all_cols) {
  n_snull <- sum(house1[[col]]=="NULL")
  if (n_snull > 0) {
  print(paste0("Number and percentages of null string value in ", col))
  print(n_snull)
  print((n_snull/(nrow(house1)))*100)
  }
}
```

```
## [1] "Number and percentages of null string value in GARAGE"
## [1] 2443
## [1] 7.387807
## [1] "Number and percentages of null string value in BUILD_YEAR"
## [1] 3088
## [1] 9.338333
```

- "GARAGE" and "BUILD_YEAR" have "NULL" entries.
- 9.37% of the total data is with missing year built info.
- The "GARAGE" NULL entry could signify the lack of garage or 0.

"GARAGE" unique value check

```
unique(house1$GARAGE)
```

```
##  [1] "2"    "1"    "3"    "8"    "6"    "4"    "NULL" "5"    "7"    "9"
## [11] "10"   "12"   "32"   "14"   "16"   "11"   "13"   "17"   "18"   "21"
## [21] "20"   "99"   "26"   "22"   "50"   "31"
```

- There is a lack of zero which supports teh aforementioned NULL = 0 argument.

Convert null to 0

```
house1$GARAGE[house1$GARAGE == "NULL"] <- 0
```

Additionally, convert garage to numeric

```
house1$GARAGE <- as.numeric(house1$GARAGE)
```

The "BUILD_YEAR"'s NULL is also likely to be caused by missing information and typically should be removed. Almost 10% of the data will be omitted, so it's better to save the previous copy.

```
house2 <- house1[house1$BUILD_YEAR != "NULL",]
```

3. Check Data Format

3.1. Check column types and correction

3.1.1. Checking column types

```
sapply(house2, class)
```

```
##               ADDRESS                SUBURB                 PRICE
##           "character"           "character"             "integer"
##              BEDROOMS             BATHROOMS                GARAGE
##             "integer"             "integer"             "numeric"
##             LAND_AREA            FLOOR_AREA            BUILD_YEAR
##             "integer"             "integer"           "character"
##              CBD_DIST           NEAREST_STN      NEAREST_STN_DIST
##             "integer"           "character"             "integer"
##             DATE_SOLD              POSTCODE              LATITUDE
##           "character"             "integer"             "numeric"
##             LONGITUDE           NEAREST_SCH      NEAREST_SCH_DIST
##             "numeric"           "character"             "numeric"
##             YEAR_SOLD                COORDS NEAREST_SCH_IS_RANKED
##             "numeric"           "character"             "numeric"
```

- The "DATE_SOLD" should be converted to date. But the column is only made up from month and year. Converting it to date would be futile since the column don't have full date information. But the year could be used to generate building age when sold.

3.1.1. Format correction

The "BUILD_YEAR" and "YEAR_SOLD" column to extract insight on building age sold.

```
house2$AGE_SOLD <- house2$YEAR_SOLD - as.numeric(house2$BUILD_YEAR)
```

3.2. Group up numeric and character columns

To ease the exploratory data analysis process, the columns will be grouped up first based on their types. ### 3.2.1. Group up numeric column

```r
num_cols <- names(house2)[sapply(house2,is.numeric)]
char_cols <- names(house2)[sapply(house2,is.character)]
print("Columns who are numeric")
```

```
## [1] "Columns who are numeric"
```

```r
print(num_cols)
```

```
##  [1] "PRICE"                "BEDROOMS"             "BATHROOMS"
##  [4] "GARAGE"               "LAND_AREA"            "FLOOR_AREA"
##  [7] "CBD_DIST"             "NEAREST_STN_DIST"     "POSTCODE"
## [10] "LATITUDE"             "LONGITUDE"            "NEAREST_SCH_DIST"
## [13] "YEAR_SOLD"            "NEAREST_SCH_IS_RANKED" "AGE_SOLD"
```

```r
print("Columns who are character")
```

```
## [1] "Columns who are character"
```

```r
print(char_cols)
```

```
## [1] "ADDRESS"     "SUBURB"      "BUILD_YEAR"  "NEAREST_STN" "DATE_SOLD"
## [6] "NEAREST_SCH" "COORDS"
```

Check their uniqueness percentages

```r
sapply(house2, function(x){
  distinct_count <- n_distinct(x)
  row_count <- nrow(house2)
  result <- (distinct_count / row_count) * 100
  result <- round(result, 5)
  return(result)
})
```

```
##               ADDRESS                SUBURB                 PRICE
##              99.76651               1.06404               7.34823
##              BEDROOMS             BATHROOMS                GARAGE
##               0.03336               0.02668               0.08672
##             LAND_AREA            FLOOR_AREA            BUILD_YEAR
##              13.23215               1.69446               0.41361
##              CBD_DIST           NEAREST_STN      NEAREST_STN_DIST
##               1.96464               0.22682               3.87258
##             DATE_SOLD              POSTCODE              LATITUDE
##               1.14743               0.38025              89.75317
##             LONGITUDE           NEAREST_SCH      NEAREST_SCH_DIST
##              86.48099               0.53369             100.00000
##             YEAR_SOLD                COORDS NEAREST_SCH_IS_RANKED
##               0.11007             100.00000               0.00667
##              AGE_SOLD
##               0.50367
```

3.2. Group up columns based on its characteristics

The next step is to group up variables based on their characteristics: number of unique values and uniqueness.

Group up ID cols - These Column will be omitted for further data analysis.

```
ID_cols <- c("ADDRESS","SUBURB","LONGITUDE","LATITUDE","POSTCODE", "DATE_SOLD",
             "BUILD_YEAR")
```

3.2.1. Group up numeric variables

Group up Discrete variables - Discrete variables are often used to categorize observation into specific category. The category can be based on ranking arrangement (Ordinal) and can additionally allowed for statistical test (ratio). - Due to its categorical characteristic, I will categorize discrete variables if it have less than 20 unique values.

```
get_disc <- function(data, col_names) {
  match_cols <- character(0)
  for (col_name in col_names) {
    if(col_name %in% names(data)) {
      num_unique <- length(unique(data[[col_name]]))
      if (num_unique > 2 && num_unique < 20) {
        match_cols <- c(match_cols, col_name)
      }
    }
  }
  return(match_cols)
}

disc_cols <- get_disc(house2, num_cols)
print(disc_cols)
```

```
## [1] "BEDROOMS"  "BATHROOMS"
```

- BEDROOMS and BATHROOMS are actually continuous as they don't represent grouping. But due to the low unique values they are categorized as discrete.

Group up continuous variables - Continuous variables are characterized by its high uniqueness and can be used for statistical analysis.

```
get_cont <- function(data, col_names) {
  match_cols <- character(0)
  for (col_name in col_names) {
    if (col_name %in% names(data)) {
      if (length(unique(data[[col_name]])) > 20) {
        match_cols <- c(match_cols, col_name)
      }
    }
  }
  return(match_cols)
}

cont_cols <- get_cont(house2, num_cols)
print(cont_cols)
```

```
##  [1] "PRICE"            "GARAGE"           "LAND_AREA"        "FLOOR_AREA"
##  [5] "CBD_DIST"         "NEAREST_STN_DIST" "POSTCODE"         "LATITUDE"
##  [9] "LONGITUDE"        "NEAREST_SCH_DIST" "YEAR_SOLD"        "AGE_SOLD"
```

Add BEDROOMS and BATHROOMS to the continuous variable

```
add <- c("BEDROOMS","BATHROOMS")

cont_cols <- append(cont_cols, add)
```

Group up binary variables - Binary variables is characterized by only having two values, which are 1 and 0.

```
get_dum <- function(data, col_names) {
  match_cols <- character(0)
  for (col_name in col_names) {
    if (col_name %in% names(data)) {
      if (length(unique(data[[col_name]])) == 2) {
        match_cols <- c(match_cols, col_name)
      }
    }
  }
  return(match_cols)
}

bin_cols <- get_dum(house1, num_cols)
print(bin_cols)
```

```
## [1] "NEAREST_SCH_IS_RANKED"
```

3.2.2. Group up character column

Group up categorical columns - All character columns are basically categorical. But in this analysis, I define it as a column that assign specific category for each entry. - I characterized categorical column as having limited unique values count, which is less than 20.

```
get_cat <- function(data, col_names) {
  match_cols <- character(0)
  for (col_name in col_names) {
    if(col_name %in% names(data)) {
      if (length(unique(data[[col_name]])) < 20) {
        match_cols <- c(match_cols, col_name)
      }
    }
  }
  return(match_cols)
}
cat_cols <- get_cat(house2, char_cols)
print("Character type category columns")
```

```
## [1] "Character type category columns"
```

```r
print(cat_cols)
```

```
## character(0)
```

- There is no categorical column.

Group up description characters - Description columns are characters columns that have high uniqueness.

```r
get_desc <- function(data, col_names) {
  match_cols <- character(0)
  for (col_name in col_names) {
    if (col_name %in% names(data)) {
      if (length(unique(data[[col_name]])) > 20) {
        match_cols <- c(match_cols, col_name)
      }
    }
  }
  return(match_cols)
}

desc_cols <- get_desc(house2, char_cols)
print("Character type discrete columns")
```

```
## [1] "Character type discrete columns"
```

```r
print(desc_cols)
```

```
## [1] "ADDRESS"     "SUBURB"      "BUILD_YEAR"  "NEAREST_STN" "DATE_SOLD"
## [6] "NEAREST_SCH" "COORDS"
```

- There are several description columns, which most of it are already ID variable. But due to the data can't be compute mathematically, then these variable won't be used for further regression model.

Group up dummy variables

```r
dum_cols <- get_dum(house2, char_cols)
print(dum_cols)
```

```
## character(0)
```

- There are no dummy variables.

Remove ID variable from other variables group.

```r
remove_columns <- function(ori_cols, remove_cols){
  return(ori_cols[!ori_cols %in% remove_cols])
}

## Numerical variable
```

```r
cont_cols <- remove_columns(cont_cols, ID_cols)
disc_cols <- remove_columns(disc_cols, ID_cols)

## Character variable
desc_cols <- remove_columns(desc_cols, ID_cols)
```

4. EDA

4.1. Numerical Variable EDA

Check numerical variable summary

```r
summary(house2[cont_cols])
```

```
##      PRICE             GARAGE         LAND_AREA        FLOOR_AREA
##  Min.   :  52000   Min.   : 0.000   Min.   :    61   Min.   :  1.0
##  1st Qu.: 410000   1st Qu.: 2.000   1st Qu.:   507   1st Qu.:131.0
##  Median : 540000   Median : 2.000   Median :   683   Median :173.0
##  Mean   : 640626   Mean   : 2.043   Mean   :  2678   Mean   :184.4
##  3rd Qu.: 770000   3rd Qu.: 2.000   3rd Qu.:   832   3rd Qu.:224.0
##  Max.   :2440000   Max.   :99.000   Max.   :999999   Max.   :849.0
##     CBD_DIST     NEAREST_STN_DIST NEAREST_SCH_DIST     YEAR_SOLD
##  Min.   :  747   Min.   :   52    Min.   : 0.07091   Min.   :1988
##  1st Qu.:11200   1st Qu.: 1800    1st Qu.: 0.87828   1st Qu.:2015
##  Median :17500   Median : 3200    Median : 1.33763   Median :2017
##  Mean   :19732   Mean   : 4508    Mean   : 1.79636   Mean   :2017
##  3rd Qu.:26500   3rd Qu.: 5300    3rd Qu.: 2.08389   3rd Qu.:2019
##  Max.   :59800   Max.   :35500    Max.   :23.25437   Max.   :2020
##     AGE_SOLD         BEDROOMS        BATHROOMS
##  Min.   :-23.00   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 12.00   1st Qu.: 3.000   1st Qu.: 1.000
##  Median : 22.00   Median : 4.000   Median : 2.000
##  Mean   : 26.97   Mean   : 3.673   Mean   : 1.827
##  3rd Qu.: 38.00   3rd Qu.: 4.000   3rd Qu.: 2.000
##  Max.   :147.00   Max.   :10.000   Max.   :16.000
```

- There are 2 concerning column:
  - "FLOOR_AREA" has a minimal value of 1, which is too small for a house.

Floor Area Check

```r
print(house2[house2$FLOOR_AREA < 10,])
```

```
##               ADDRESS     SUBURB  PRICE BEDROOMS BATHROOMS GARAGE LAND_AREA
## 2085   11 Bracken Way Bibra Lake 520000        4         2      1       700
## 3905 12 Vernon Street      Trigg 680000        3         1      2       728
##      FLOOR_AREA BUILD_YEAR CBD_DIST              NEAREST_STN NEAREST_STN_DIST
## 2085          1       1986    17000 Cockburn Central Station             3800
## 3905          1       2015    12800          Warwick Station             4200
```

```
##      DATE_SOLD POSTCODE  LATITUDE LONGITUDE                     NEAREST_SCH
## 2085 05-2020\n     6163 -32.10254  115.8285             PERTH WALDORF SCHOOL
## 3905 05-2014\n     6029 -31.86999  115.7639 ST MARY'S ANGLICAN GIRLS' SCHOOL
##      NEAREST_SCH_DIST YEAR_SOLD                  COORDS NEAREST_SCH_IS_RANKED
## 2085        1.5390971      2020       115.8285,-32.10254                     0
## 3905        0.8065437      2014 115.7639273,-31.86999197                     1
##      AGE_SOLD
## 2085       34
## 3905       -1
```

- Although other variable value is normal, the floor are of 1 (sqm) is unlikely and most likely to be entry error.

Remove suspicious floor area

```
house2 <- house2[house2$FLOOR_AREA > 10, ]
```

Building Age check

```
#print(house2[house2$AGE_SOLD < 0,])
head(house2[house2$AGE_SOLD < 0,], 10)
```

```
##                 ADDRESS              SUBURB   PRICE BEDROOMS BATHROOMS GARAGE
## 59      1 Bristow Street              Haynes  177000        3         1      0
## 144    1 Eastfield Court            Ferndale  250000        3         1      1
## 224        1 Hartle Lane           Brookdale  311850        4         2      2
## 343     1 Mcdermott Road Kwinana Town Centre  145700        3         2      0
## 489         1 Silky Lane               Wandi  303000        4         1      0
## 778 10 Barrington Avenue      Champion Lakes  185000        4         2      2
## 822     10 Bosbaan Place      Champion Lakes  196000        4         2      2
## 837    10 Bristow Street              Haynes  167000        3         2      2
## 845    10 Bungalow Court    Peppermint Grove 2400000        4         5      4
## 872     10 Cerulean Road               Karnup  179000        3         1      2
##     LAND_AREA FLOOR_AREA BUILD_YEAR CBD_DIST         NEAREST_STN
## 59        412        150       2015    26200    Armadale Station
## 144       955        396       2017    11400 Queens Park Station
## 224       471        154       2014    28000    Armadale Station
## 343       206        120       2015    32600     Wellard Station
## 489       437        199       2016    27600     Kwinana Station
## 778       420        191       2010    22200   Kelmscott Station
## 822       451        180       2013    22300   Kelmscott Station
## 837       340        151       2015    26200    Armadale Station
## 845       444        420       2012    10200   Cottesloe Station
## 872       405        159       2015    55700    Mandurah Station
##     NEAREST_STN_DIST DATE_SOLD POSTCODE  LATITUDE LONGITUDE
## 59              2400 08-2013\n     6112 -32.15937  115.9881
## 144             2700 07-2016\n     6148 -32.03099  115.9354
## 224             1900 10-2013\n     6112 -32.17006  116.0063
## 343             2400 08-2013\n     6167 -32.24286  115.8161
## 489             4300 02-2015\n     6167 -32.19984  115.8608
## 778             2000 10-2009\n     6111 -32.11564  115.9929
## 822             2200 09-2012\n     6111 -32.11843  115.9910
## 837             2400 02-2014\n     6112 -32.15944  115.9889
```

```
## 845              937 04-2007\n     6011 -32.00224  115.7683
## 872             9200 09-2014\n     6176 -32.44582  115.7645
##                              NEAREST_SCH NEAREST_SCH_DIST YEAR_SOLD
## 59                  DALE CHRISTIAN SCHOOL        1.0032399      2013
## 144                      FOUNTAIN COLLEGE        1.5711536      2016
## 224           ARMADALE SENIOR HIGH SCHOOL        1.2149886      2013
## 343                        GILMORE COLLEGE        0.5263698      2013
## 489                        GILMORE COLLEGE        6.0801532      2015
## 778          KELMSCOTT SENIOR HIGH SCHOOL        1.8334426      2009
## 822          KELMSCOTT SENIOR HIGH SCHOOL        1.9163643      2012
## 837                 DALE CHRISTIAN SCHOOL        0.9267401      2014
## 845 ST HILDA'S ANGLICAN SCHOOL FOR GIRLS        0.3154539      2007
## 872                       COMET BAY COLLEGE        3.3419427      2014
##                         COORDS NEAREST_SCH_IS_RANKED AGE_SOLD
## 59  115.9880817,-32.15937438                       0       -2
## 144    115.9353606,-32.03099                       0       -1
## 224 116.0063044,-32.17006445                       0       -1
## 343 115.8161455,-32.24286002                       1       -2
## 489 115.8607572,-32.19984019                       1       -1
## 778 115.9928519,-32.11564044                       1       -1
## 822 115.9910315,-32.11842915                       1       -1
## 837 115.9888948,-32.15943529                       0       -1
## 845 115.7683182,-32.00224362                       1       -5
## 872 115.7644814,-32.44582454                       1       -1
```

```
print("Number of negative value house age")
```

```
## [1] "Number of negative value house age"
```

```
sum(house2$AGE_SOLD < 0)
```

```
## [1] 561
```

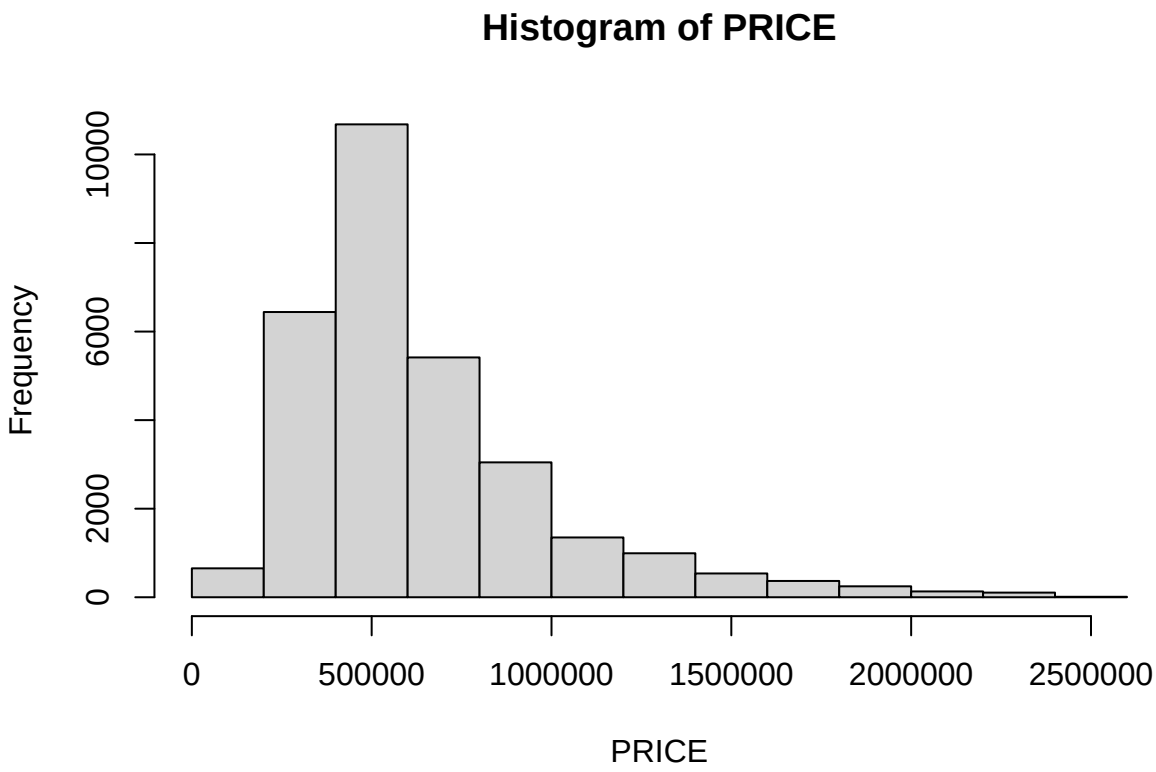
- There are 561 entries that have negative age value. This could mean that the land is bought first and later the home is constructed.
- But for now as the nature of the age entry is still ambiguous, "AGE_SOLD" will be omitted.
- This also applies for the "YEAR_SOLD" entry as it is also ambiguous.

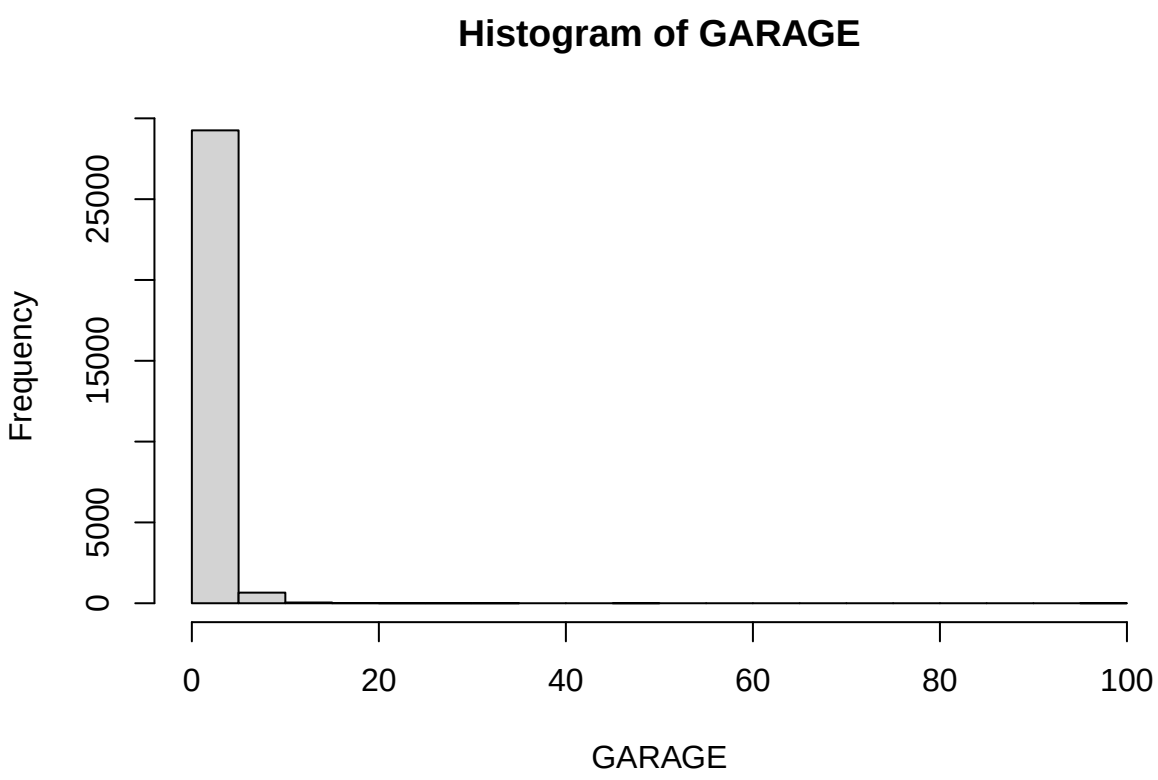
Omit "AGE_SOLD" column.

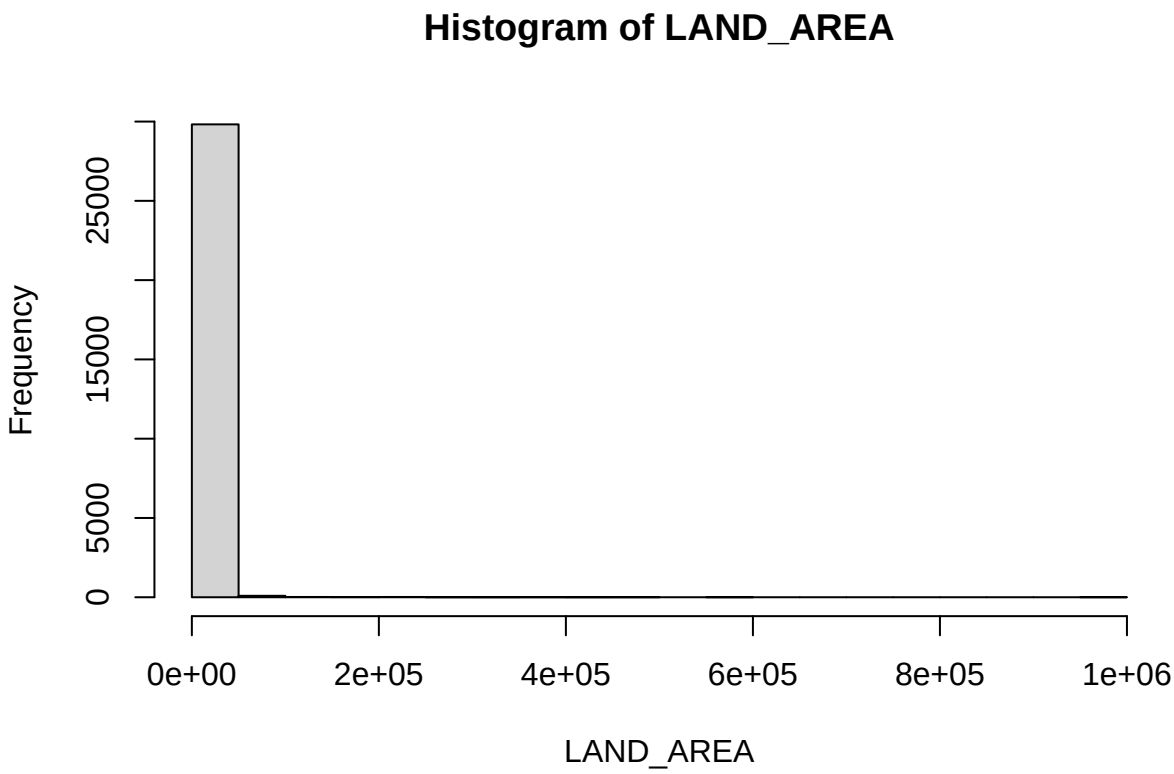
```
## Remove from column
house2 <- select(house2, -c(AGE_SOLD, YEAR_SOLD))

## Remove from continuous variable list
cont_cols <- cont_cols[!cont_cols %in% c("AGE_SOLD", "YEAR_SOLD")]
```

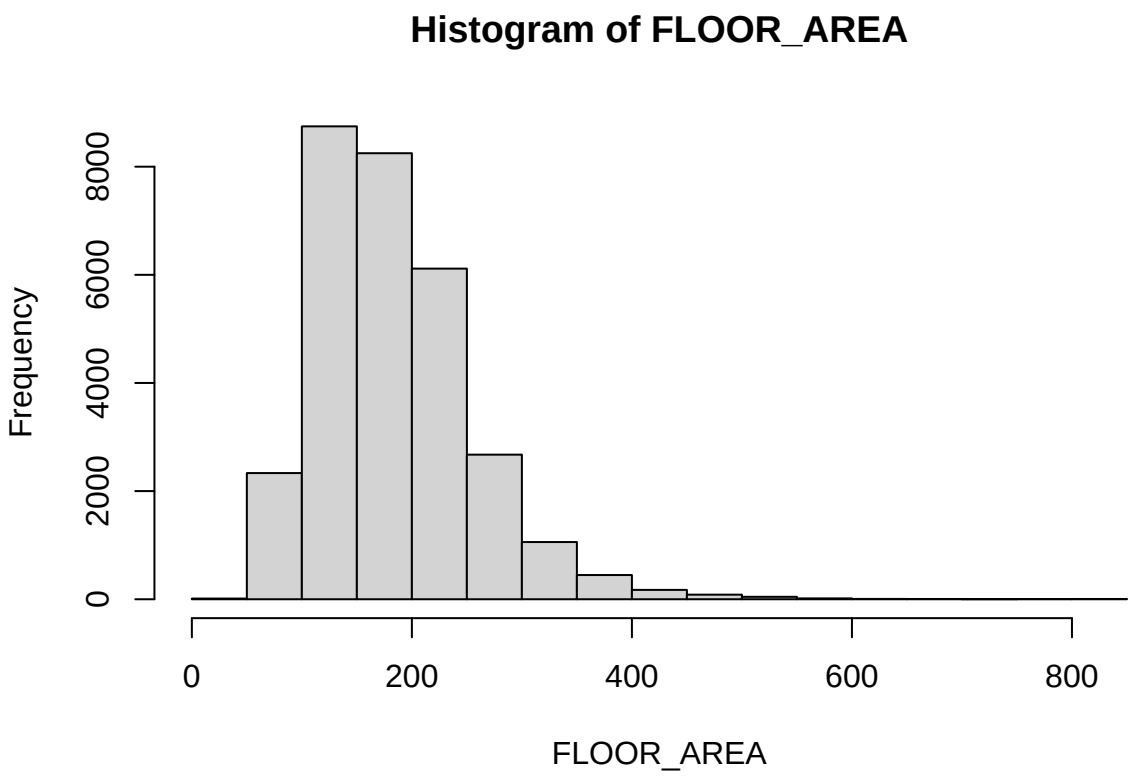
4.1.1. Check Distribution

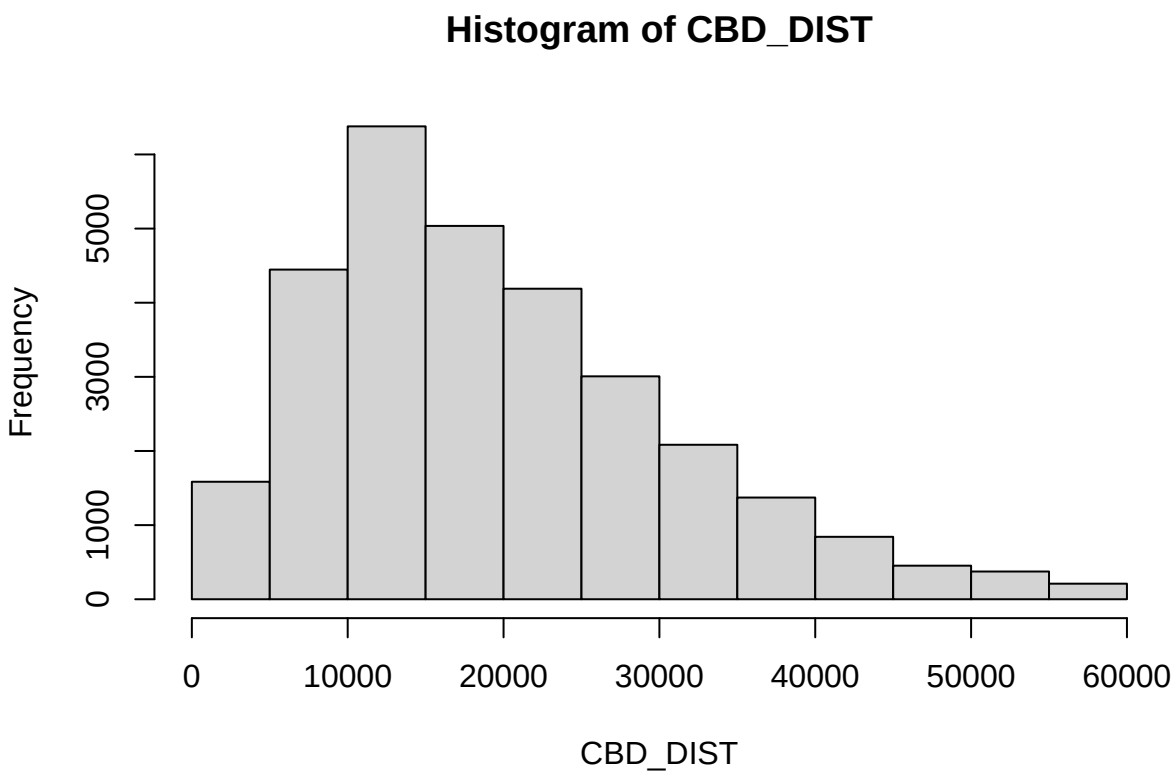
```
for (col in cont_cols) {
  hist(house2[[col]], main = paste("Histogram of", col), xlab = col)
}
```

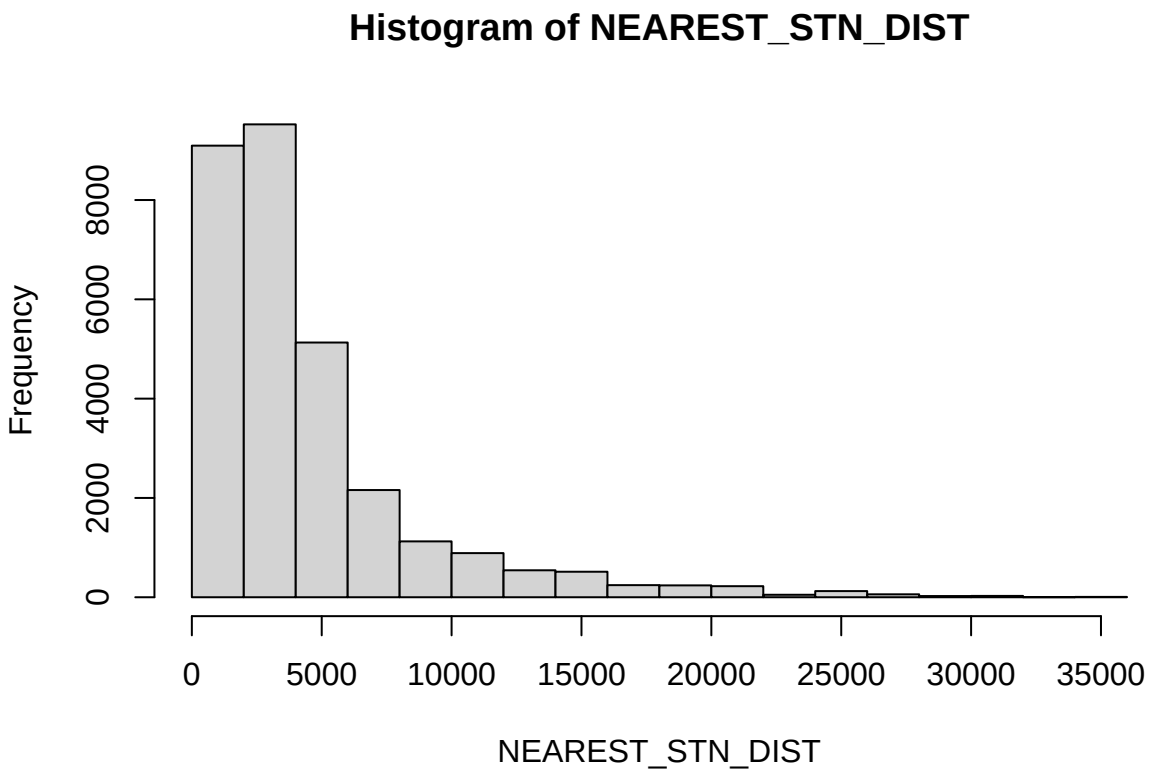

Histogram of PRICE
Frequency
0
2000
6000
10000
0
500000
1000000
1500000
2000000
2500000
PRICE

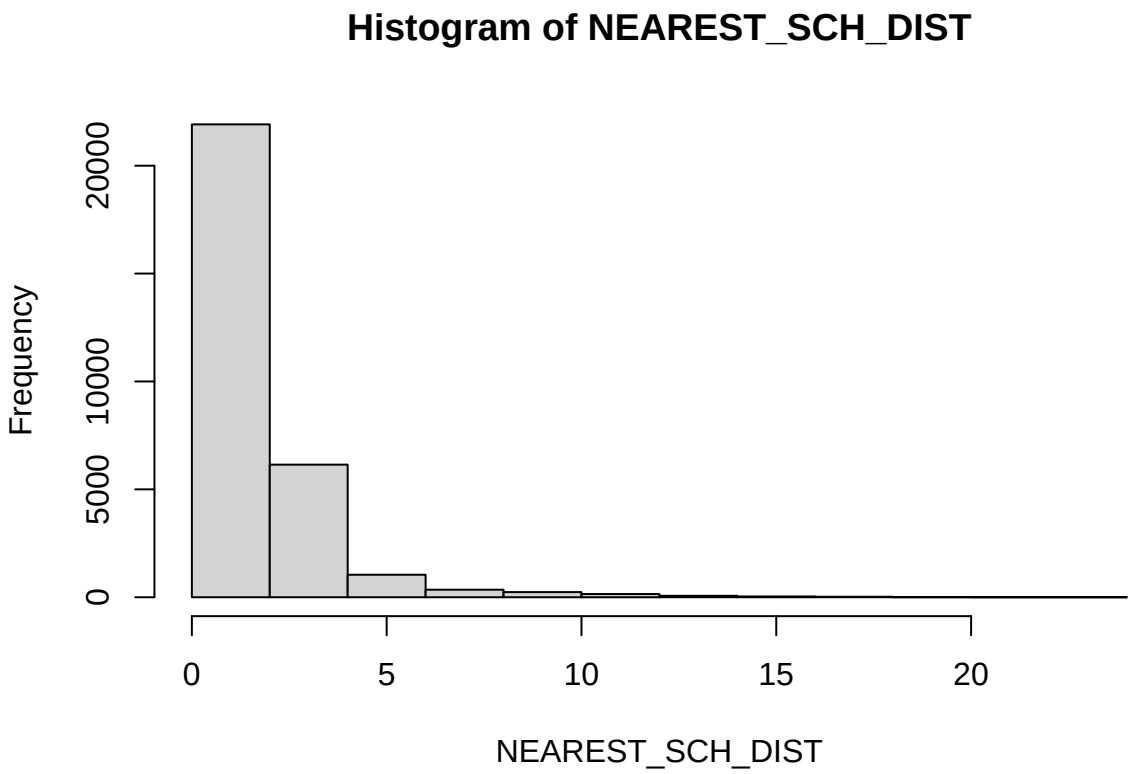

Histogram of GARAGE
Frequency
0
5000
15000
25000
0
20
40
60
80
100
GARAGE

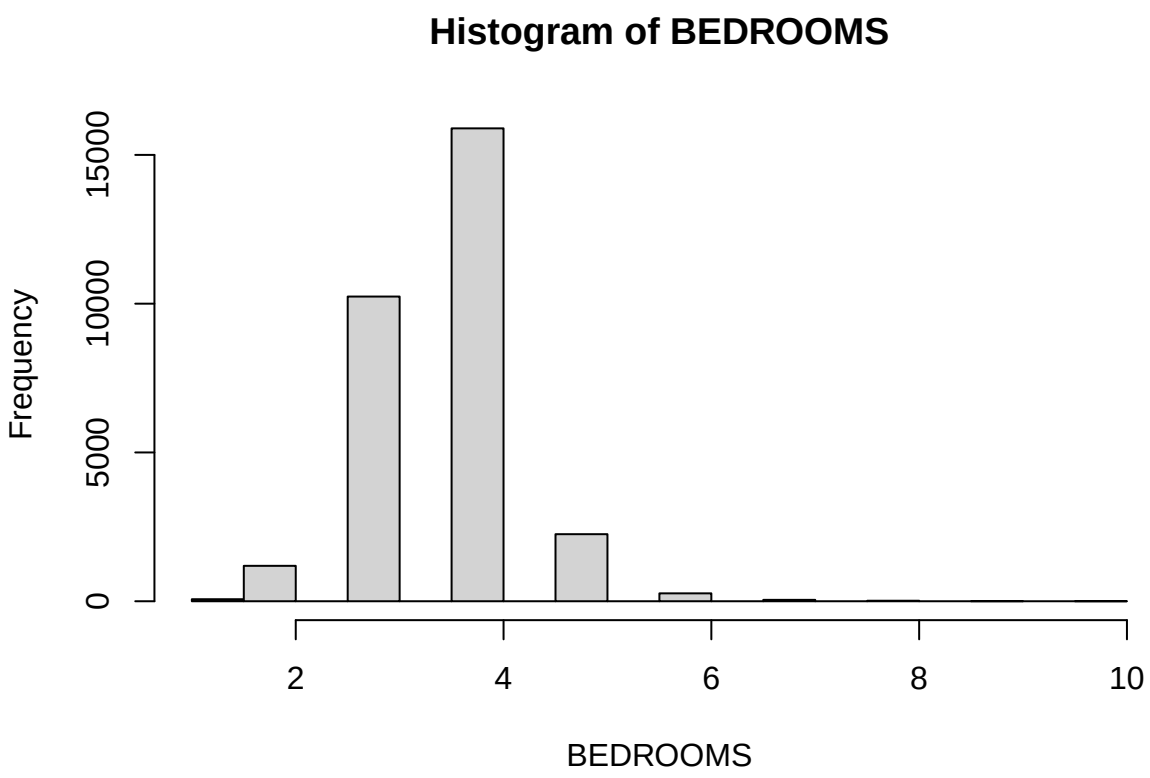

Histogram of LAND_AREA
Frequency
0
5000
15000
25000
0e+00
2e+05
4e+05
6e+05
8e+05
1e+06
LAND_AREA

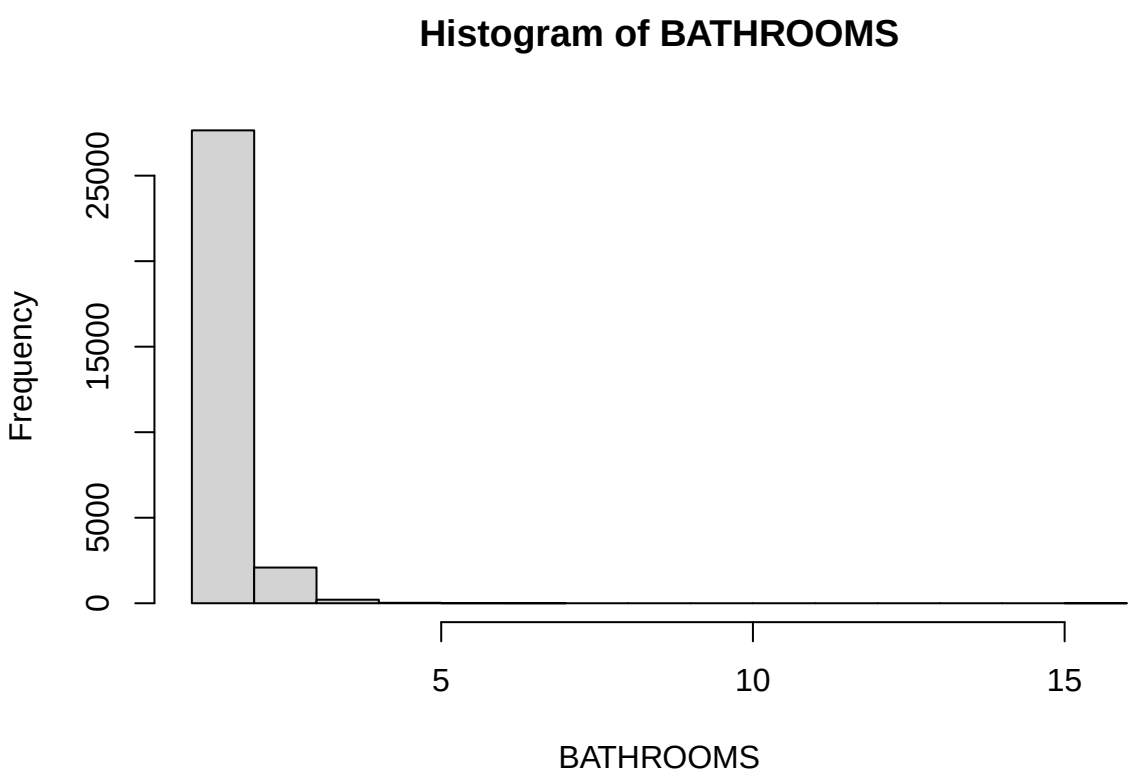

Histogram of FLOOR_AREA
Frequency
0
2000
4000
6000
8000
0
200
400
600
800
FLOOR_AREA


Histogram of CBD_DIST
Frequency
0
1000
3000
5000
0
10000
20000
30000
40000
50000
60000
CBD_DIST


Histogram of NEAREST_STN_DIST
Frequency
0
2000
4000
6000
8000
0
5000
10000
15000
20000
25000
30000
35000
NEAREST_STN_DIST

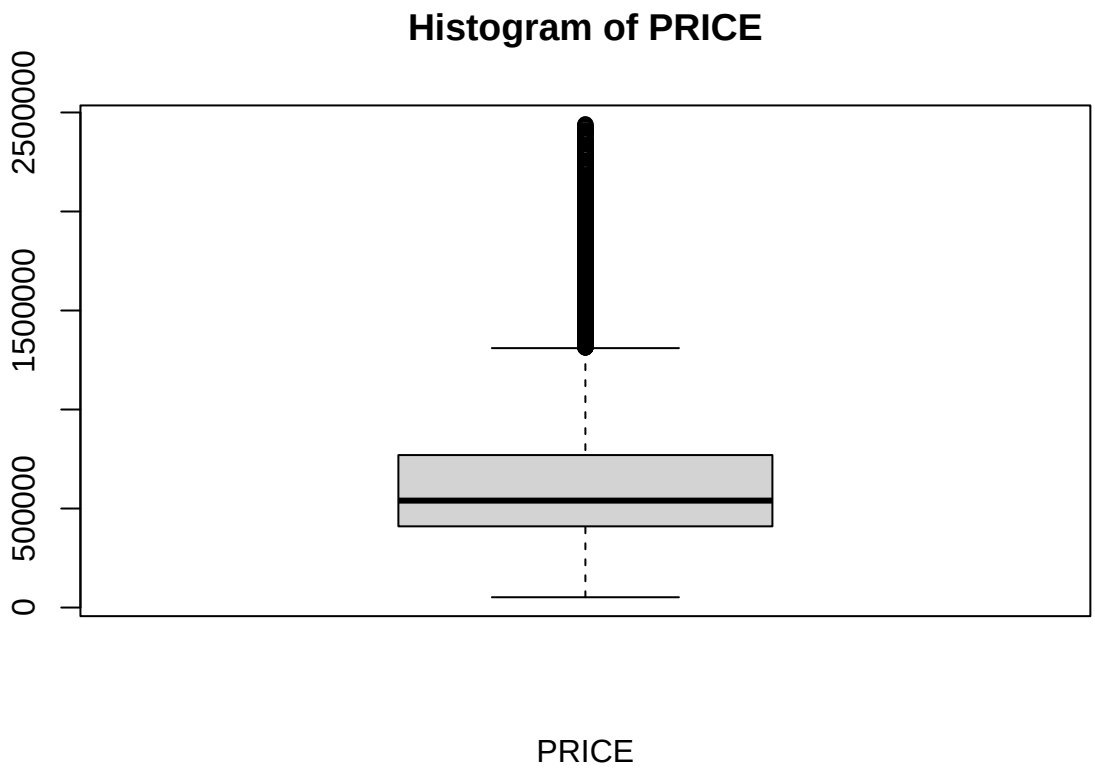

Histogram of NEAREST_SCH_DIST
Frequency
0
5000
10000
20000
0
5
10
15
20
NEAREST_SCH_DIST

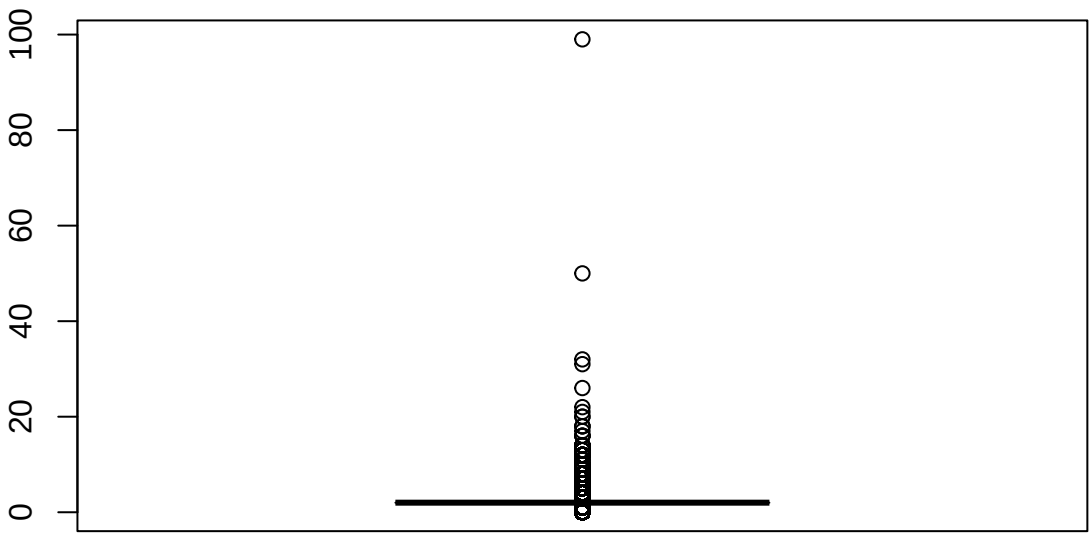

Histogram of BEDROOMS
Frequency
0
5000
10000
15000
2
4
6
8
10
BEDROOMS

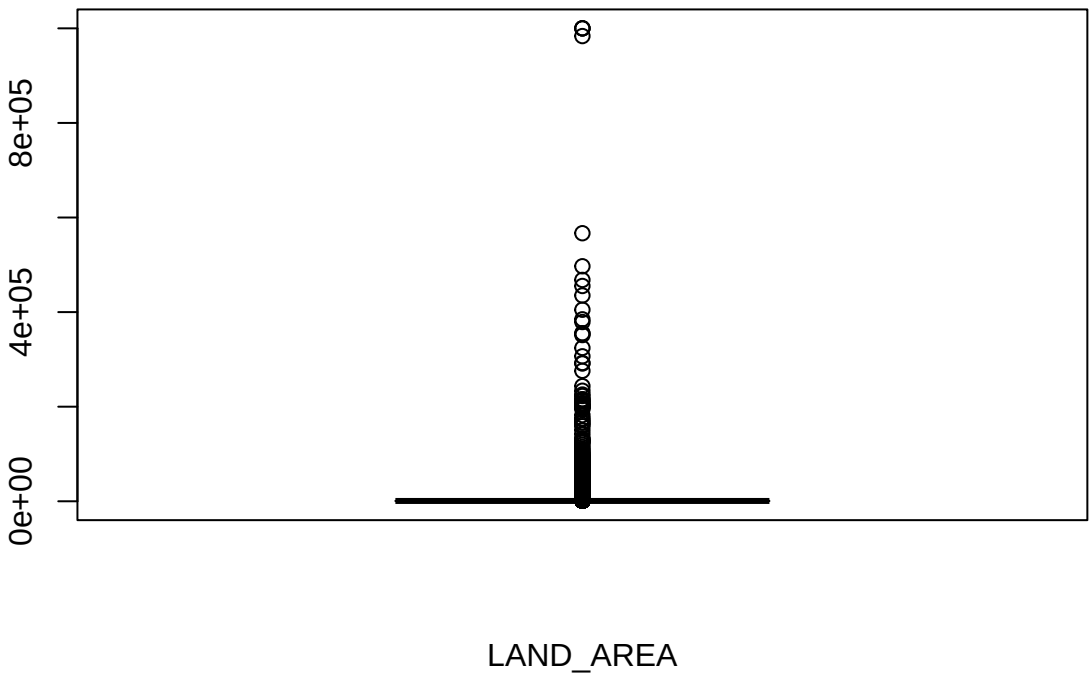


Check for the quartilr distribution

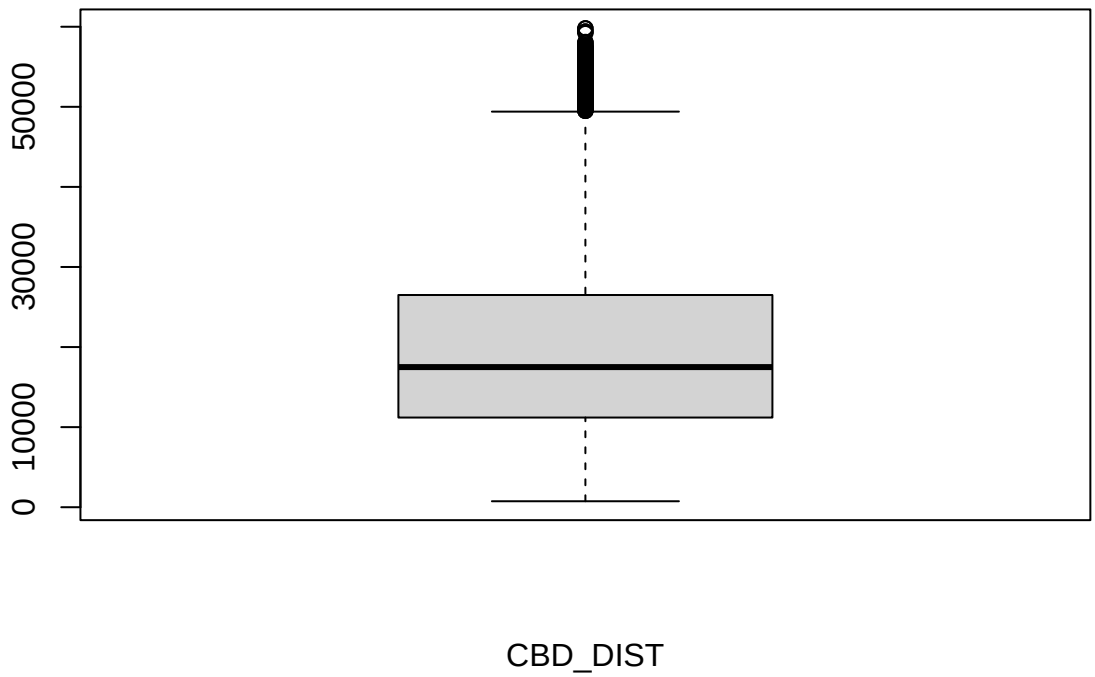
```
for (col in cont_cols){
  boxplot(house2[[col]], main = paste("Histogram of", col), xlab = col)
}
```

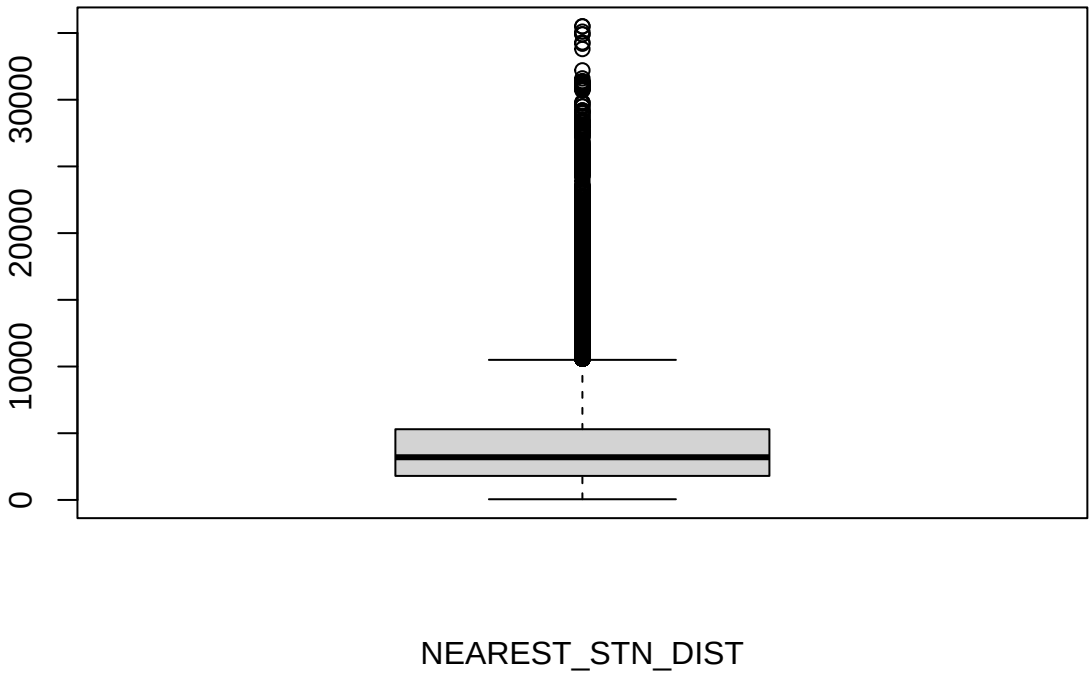

Histogram of PRICE
2500000
1500000
500000
0
PRICE

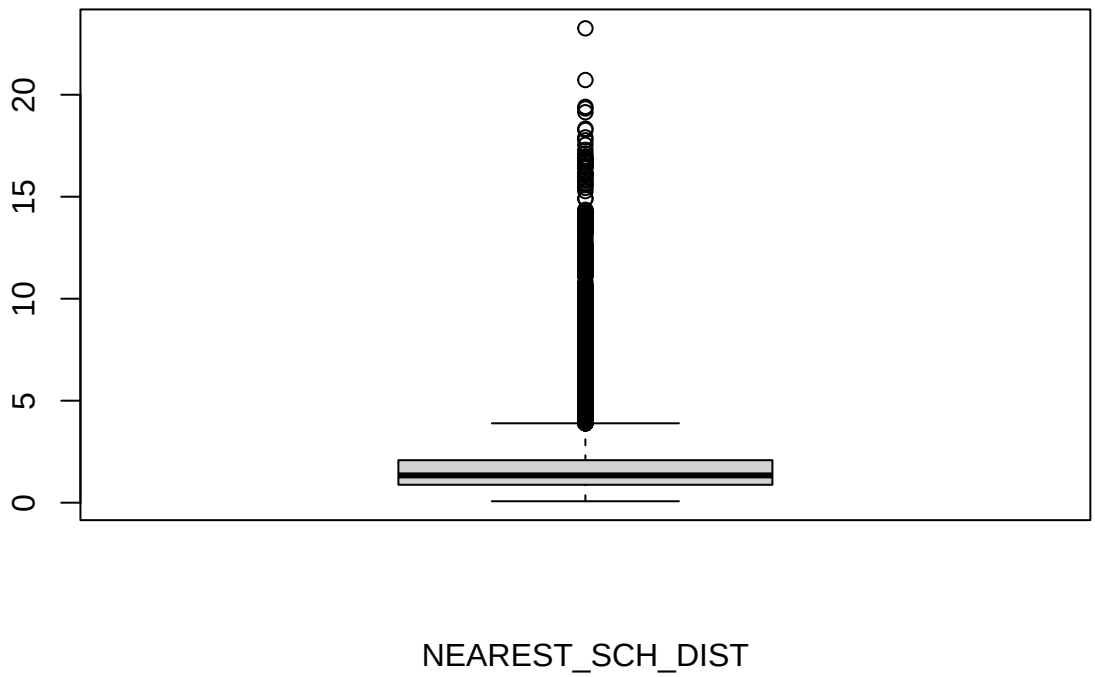

Histogram of GARAGE
0
20
40
60
80
100
GARAGE

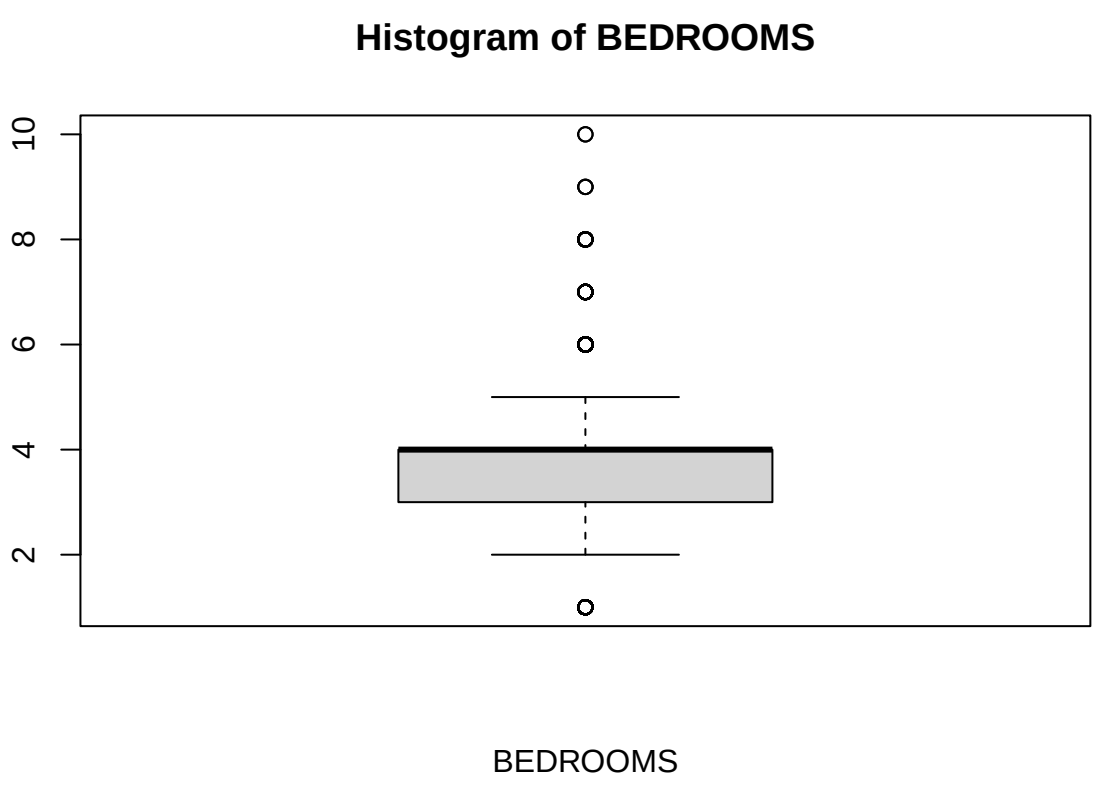

Histogram of LAND_AREA
0e+00
4e+05
8e+05
LAND_AREA


Histogram of FLOOR_AREA
0
200
400
600
800
FLOOR_AREA


Histogram of CBD_DIST
0
10000
30000
50000
CBD_DIST


Histogram of NEAREST_STN_DIST
0
10000
20000
30000
NEAREST_STN_DIST


Histogram of NEAREST_SCH_DIST
0
5
10
15
20
NEAREST_SCH_DIST


Histogram of BEDROOMS
10
8
6
4
2
BEDROOMS

BATHROOMS

- All of the continuous variables are positively skewed and have outliers. - I need to confirm it with normal distribution test.

Normal distribution test

```
library("tseries")
```

```
## Warning: package 'tseries' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

For the test, I use Jarque-Bera due to the high data points.

```
for (col in cont_cols) {
  res = jarque.bera.test(house2[[col]])
  print(col)
  print(res)
}
```

```
## [1] "PRICE"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
```

```
## X-squared = 33966, df = 2, p-value < 2.2e-16
##
## [1] "GARAGE"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 676026936, df = 2, p-value < 2.2e-16
##
## [1] "LAND_AREA"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 3539747662, df = 2, p-value < 2.2e-16
##
## [1] "FLOOR_AREA"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 32573, df = 2, p-value < 2.2e-16
##
## [1] "CBD_DIST"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 4186.3, df = 2, p-value < 2.2e-16
##
## [1] "NEAREST_STN_DIST"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 94401, df = 2, p-value < 2.2e-16
##
## [1] "NEAREST_SCH_DIST"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 612269, df = 2, p-value < 2.2e-16
##
## [1] "BEDROOMS"
##
##  Jarque Bera Test
##
## data:  house2[[col]]
## X-squared = 6151.6, df = 2, p-value < 2.2e-16
##
## [1] "BATHROOMS"
##
##  Jarque Bera Test
```

```
##
## data:  house2[[col]]
## X-squared = 219718, df = 2, p-value < 2.2e-16
```

- All of the continuous variables deosn't have normal distribution (bivariate).

Check for outliers percentage (Based on IQR)

```
for (col in cont_cols){

  var <- house2[[col]]
  #Calculate IQE, Upper, and Lower Bound
  iqr <- IQR(var)
  q1 <- quantile(var, 0.25)
  q3 <- quantile(var, 0.75)

  # Calculate Upper and lower bound
  lower_bound <- q1 - (1.5*iqr)
  upper_bound <- q3 + (1.5*iqr)

  # Count character that is outisde the bounds
  outliers <- which(var < lower_bound | var > upper_bound)

  # Count percentages
  print(col)
  res = (length(outliers) / length (house2[[col]]))*100
  print(res)
}
```

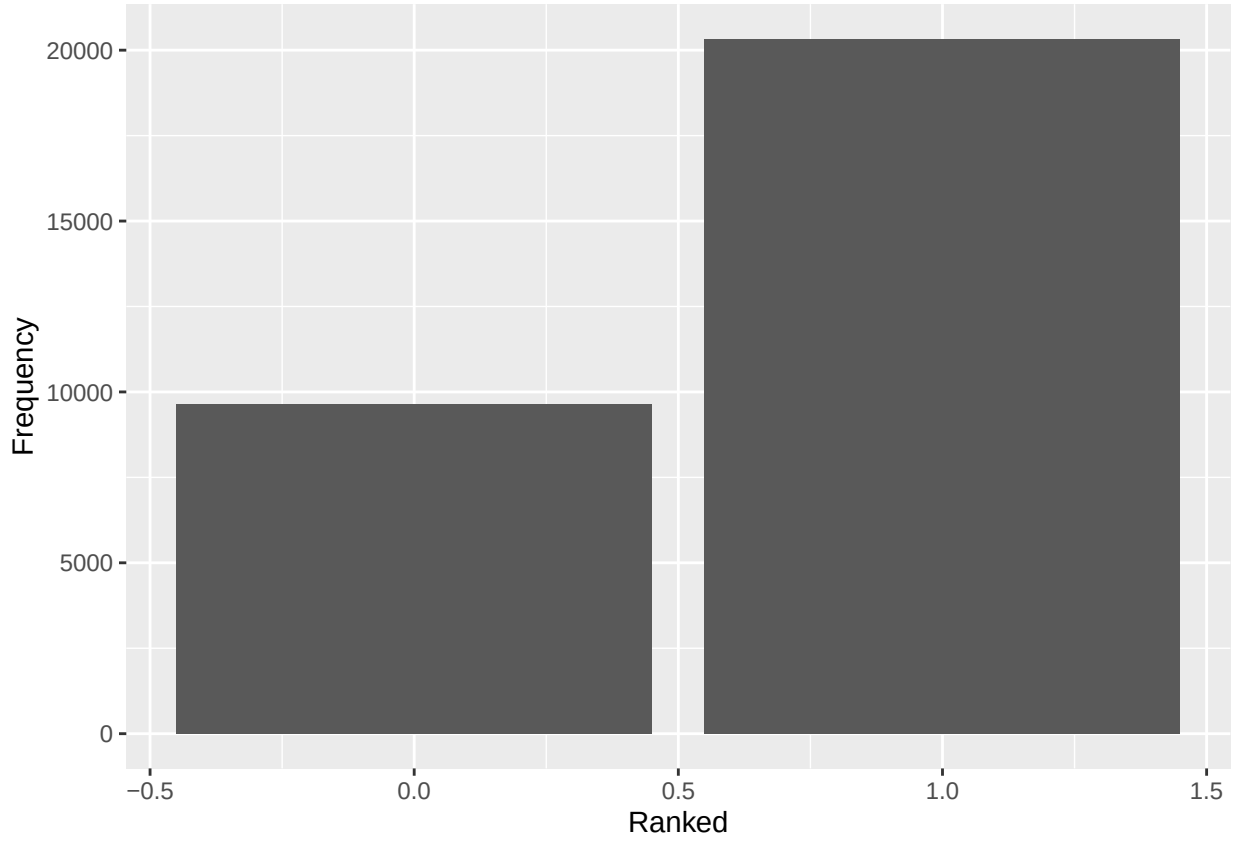
```
## [1] "PRICE"
## [1] 5.957702
## [1] "GARAGE"
## [1] 38.28808
## [1] "LAND_AREA"
## [1] 14.087
## [1] "FLOOR_AREA"
## [1] 2.158249
## [1] "CBD_DIST"
## [1] 1.99146
## [1] "NEAREST_STN_DIST"
## [1] 8.896524
## [1] "NEAREST_SCH_DIST"
## [1] 6.678231
## [1] "BEDROOMS"
## [1] 1.340983
## [1] "BATHROOMS"
## [1] 0.8105944
```

- There are variable that have more than %5 of outlier: PRICE, GARAGE, LAND_AREA, NEAREST_STN_DIST, and NEAREST_SCH_DIST.

The continuous variable have: - Positive value - 0 value (Garage) - Outliers

Dummy variable distribution

```
house2 %>%
  ggplot(aes(x= NEAREST_SCH_IS_RANKED)) +
  geom_bar() +
  labs(y = "Frequency",  x = "Ranked")
```



- Most of the schools are ranked.

4.2. Plot columns distribution

"PRICE" is the dependent variable and main variable of intrest in this analysis. I need to recheck the distribution.

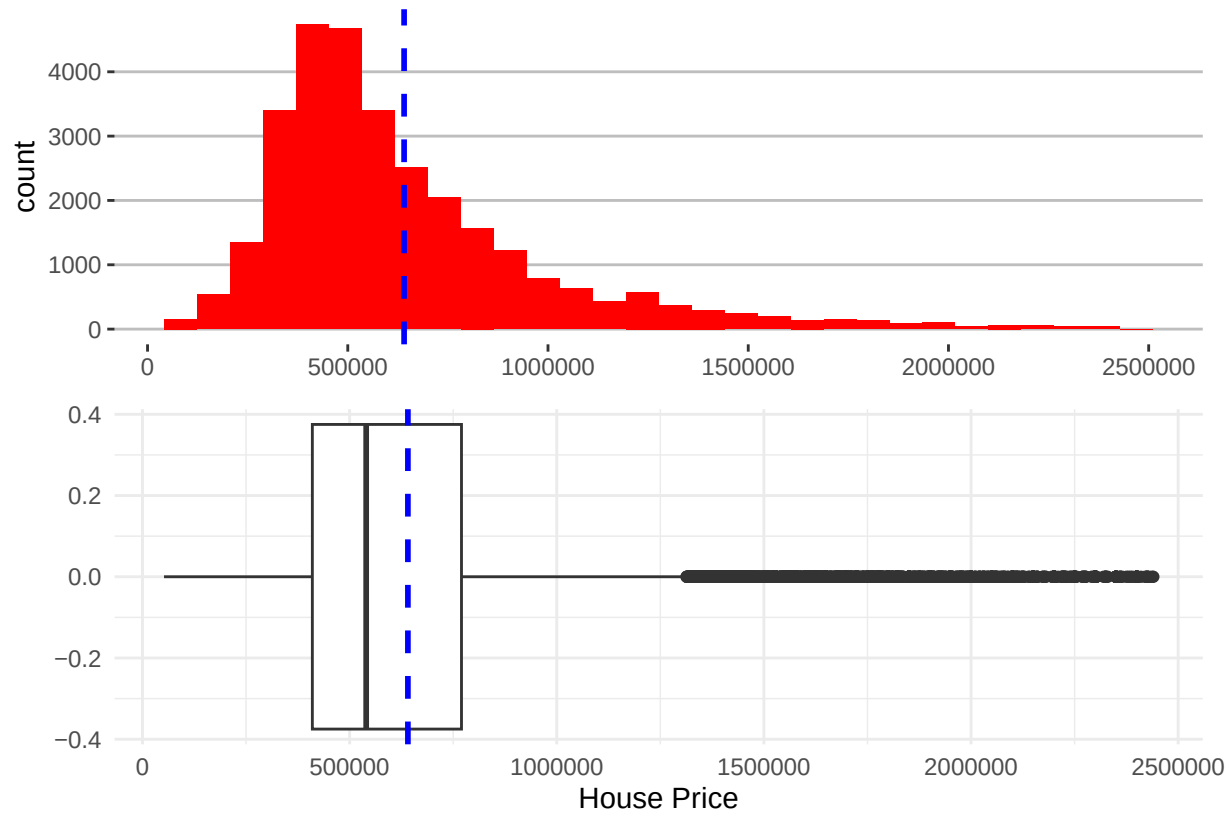
```
## Plot 1. House Price distribution
plot_1a <- ggplot(house2, aes(x= PRICE)) +
  geom_histogram(bins=30, fill = "red") +
  geom_vline(aes(xintercept = mean(PRICE)), color = "blue", linetype = "dashed", size = 1) +
  labs(x = "House Price") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid.major.x = element_blank(),
        panel.grid.minor = element_blank(),
        panel.grid.major.y = element_line(color="grey"),
        strip.background = element_rect(fill = "white", color="grey"))
```

```
## 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.
```

```
plot_1b <- ggplot(house2, aes(x= PRICE)) +
          geom_boxplot() +
          geom_vline(aes(xintercept = mean(PRICE)), color = "blue", linetype = "dashed", size = 1) +
          labs(x = "House Price") +
          theme(panel.background = element_rect(fill = "white"),
            panel.grid.major.x = element_blank(),
            panel.grid.minor = element_blank(),
            panel.grid.major.y = element_line(color="grey"),
            strip.background = element_rect(fill = "white", color="grey"))
```

```
plot_1 <- plot_1a / plot_1b +
            plot_layout(axis_titles = "collect") +
            theme_minimal()

plot_1
```



- It has a positive skewness caused positive upper bound outliers.

```
# Save plot
ggsave(plot_1,
       filename = "plot_1.png",
       device = "png",
       height = 4, width = 8, units = "in")
```

```r
median(house2$PRICE)
```

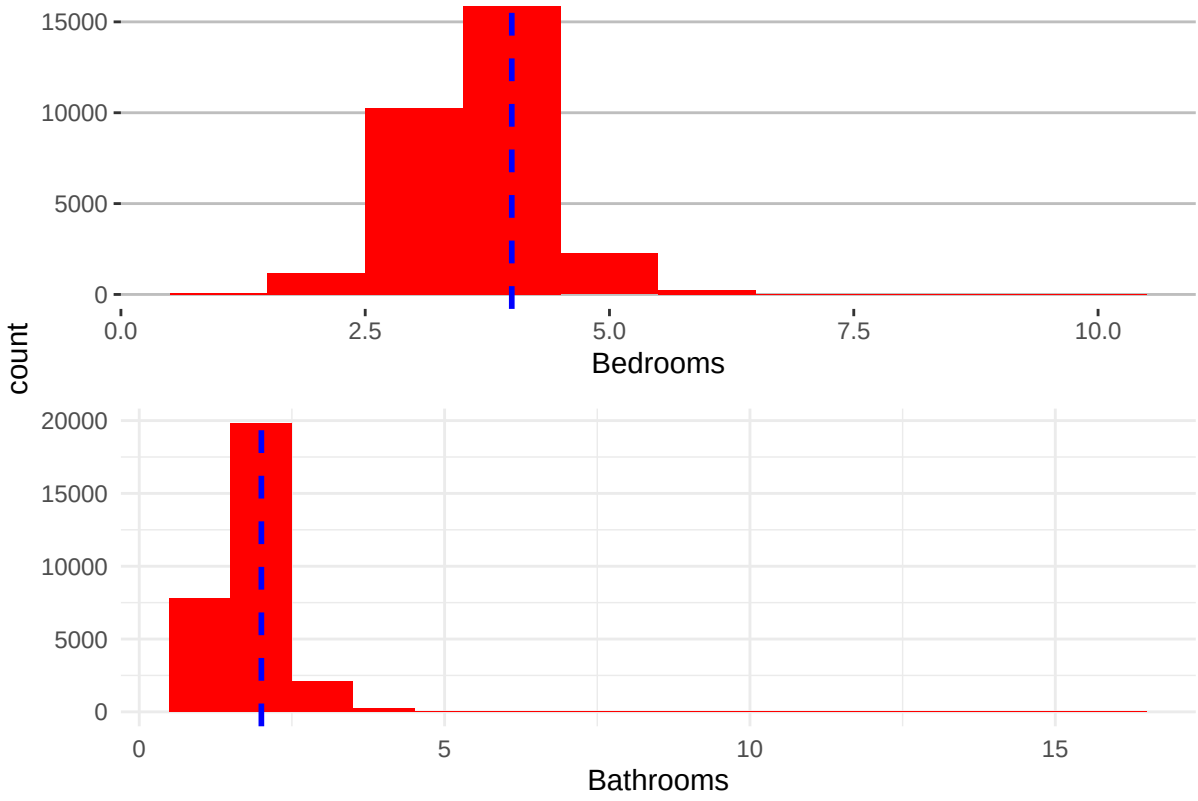
```
## [1] 540000
```

House price outliers is pretty common considering the existence of big houses such as mansions. Let's recheck the bathroom and bedrooms as a proxy for house size.

```r
## Plot 2: Land and Floor Distribution (2 kernel density)
plot_2a <- ggplot(house2, aes(x= BEDROOMS)) +
  geom_histogram(binwidth = 1, fill = "red") +
  geom_vline(aes(xintercept = median(BEDROOMS)), color = "blue", linetype = "dashed", size = 1) +
  labs(x = "Bedrooms") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid.major.x = element_blank(),
        panel.grid.minor = element_blank(),
        panel.grid.major.y = element_line(color="grey"),
        strip.background = element_rect(fill = "white", color="grey"))


plot_2b <- ggplot(house2, aes(x= BATHROOMS)) +
  geom_histogram(binwidth = 1, fill = "red") +
  geom_vline(aes(xintercept = median(BATHROOMS)), color = "blue", linetype = "dashed", size = 1) +
  labs(x = "Bathrooms") +
  theme(panel.background = element_rect(fill = "white"),
        panel.grid.major.x = element_blank(),
        panel.grid.minor = element_blank(),
        panel.grid.major.y = element_line(color="grey"),
        strip.background = element_rect(fill = "white", color="grey"))
```

Plot 2: Bedrooms and bathrooms distribution

```r
plot_2 <- plot_2a / plot_2b +
          plot_layout(axes = "collect") +
          theme_minimal()

plot_2
```

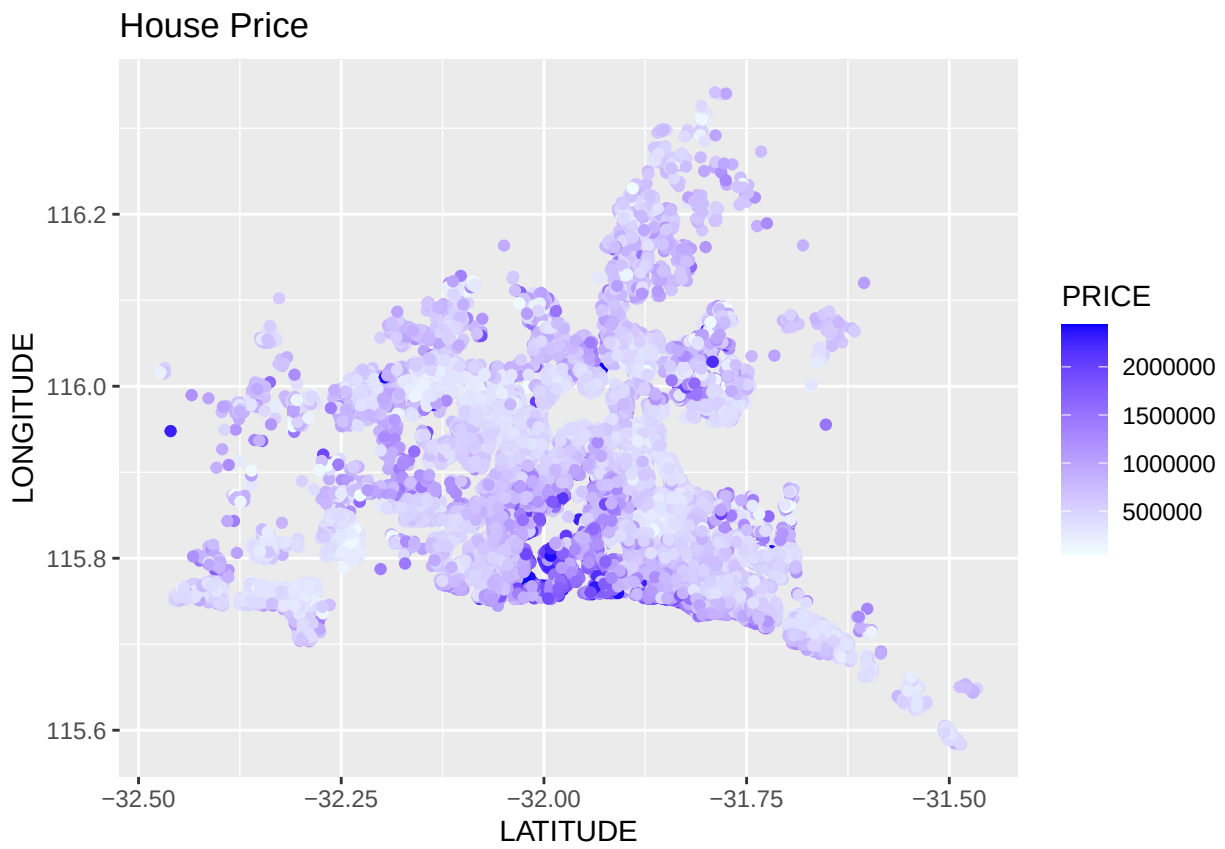



```
# Save plot
ggsave(plot_2,
       filename = "plot_2_Bed_Bath.png",
       device = "png",
       height = 4, width = 8, units = "in")
```

4.2.1. Spatial Distribution

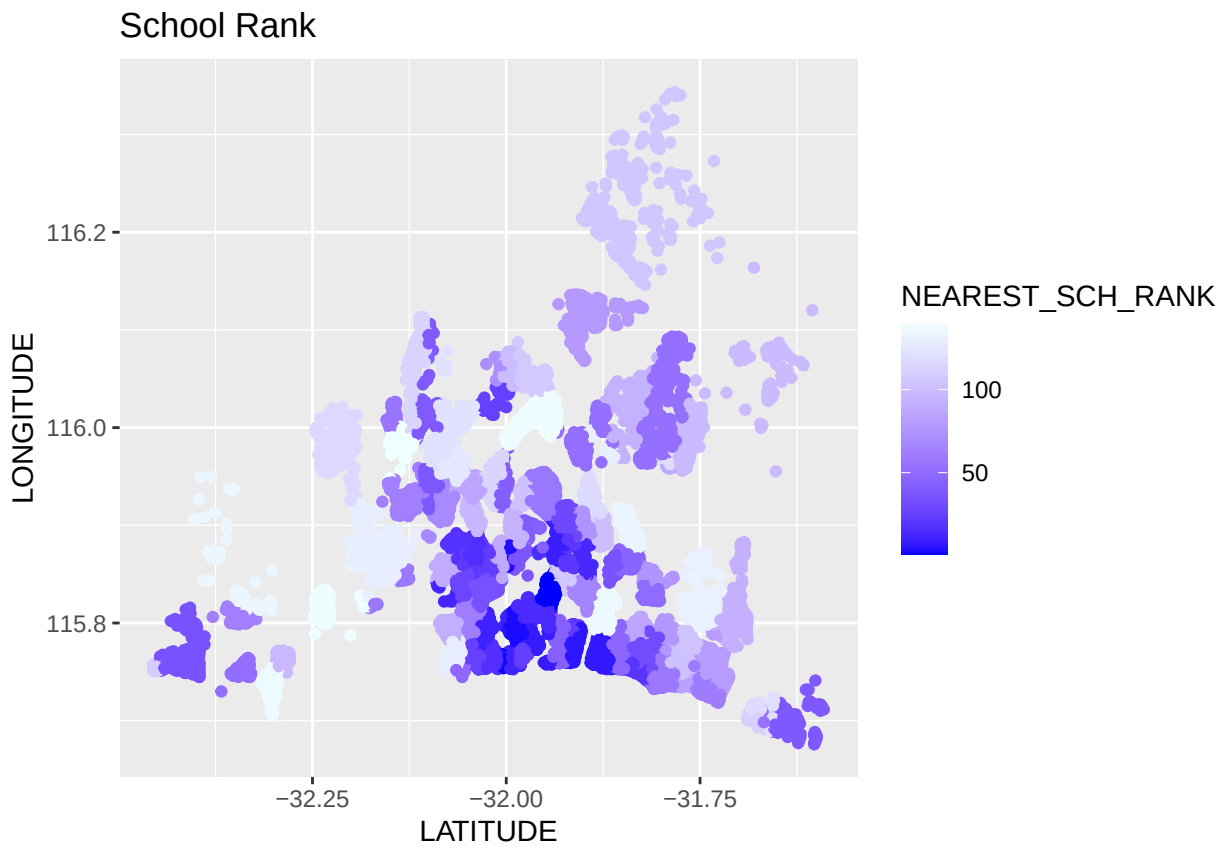
Check house price spatial distribution

```
plot_3a <- ggplot(data = house2) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = PRICE)) +
  scale_colour_gradient(low = "azure", high = "blue") +
  ggtitle("House Price")

plot_3a
```



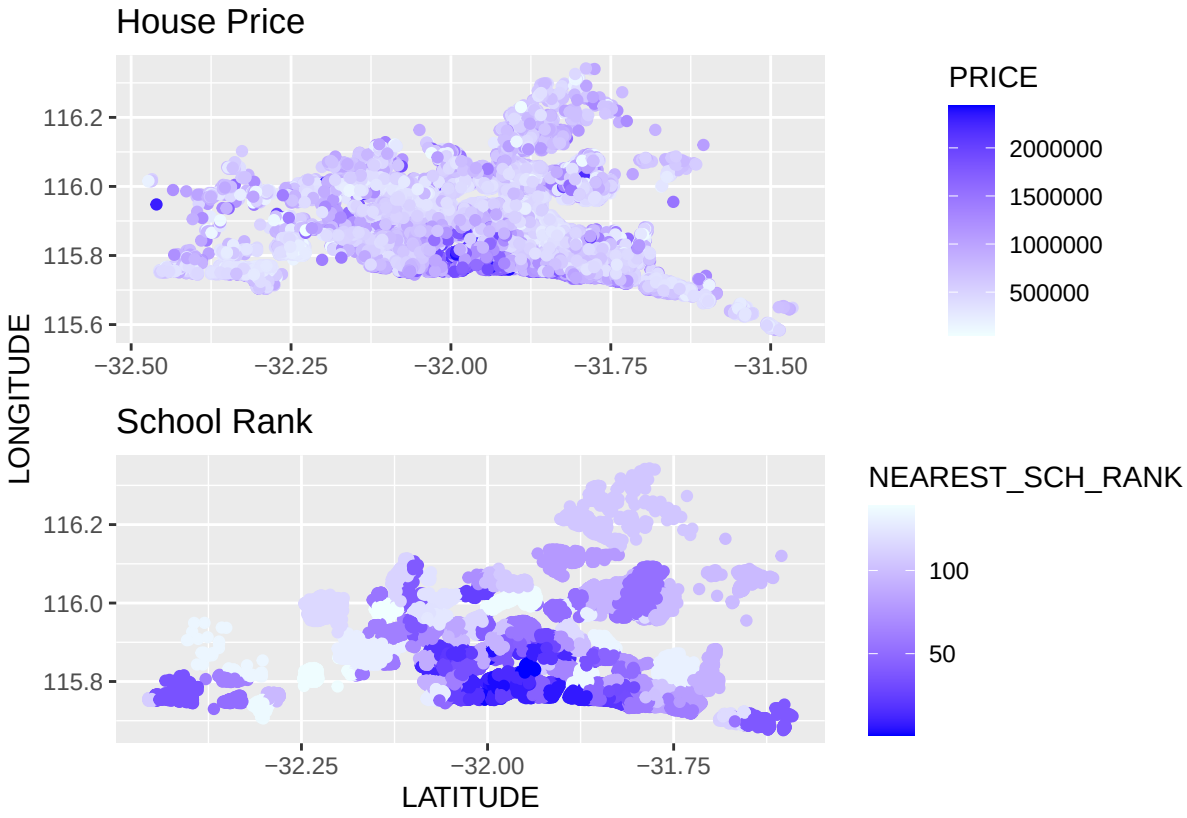
Check school distribution

```
plot_3b <- ggplot(data = house_a) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = NEAREST_SCH_RANK)) +
  scale_colour_gradient(low = "blue", high = "azure") +
  ggtitle("School Rank")

plot_3b
```



Plot 3

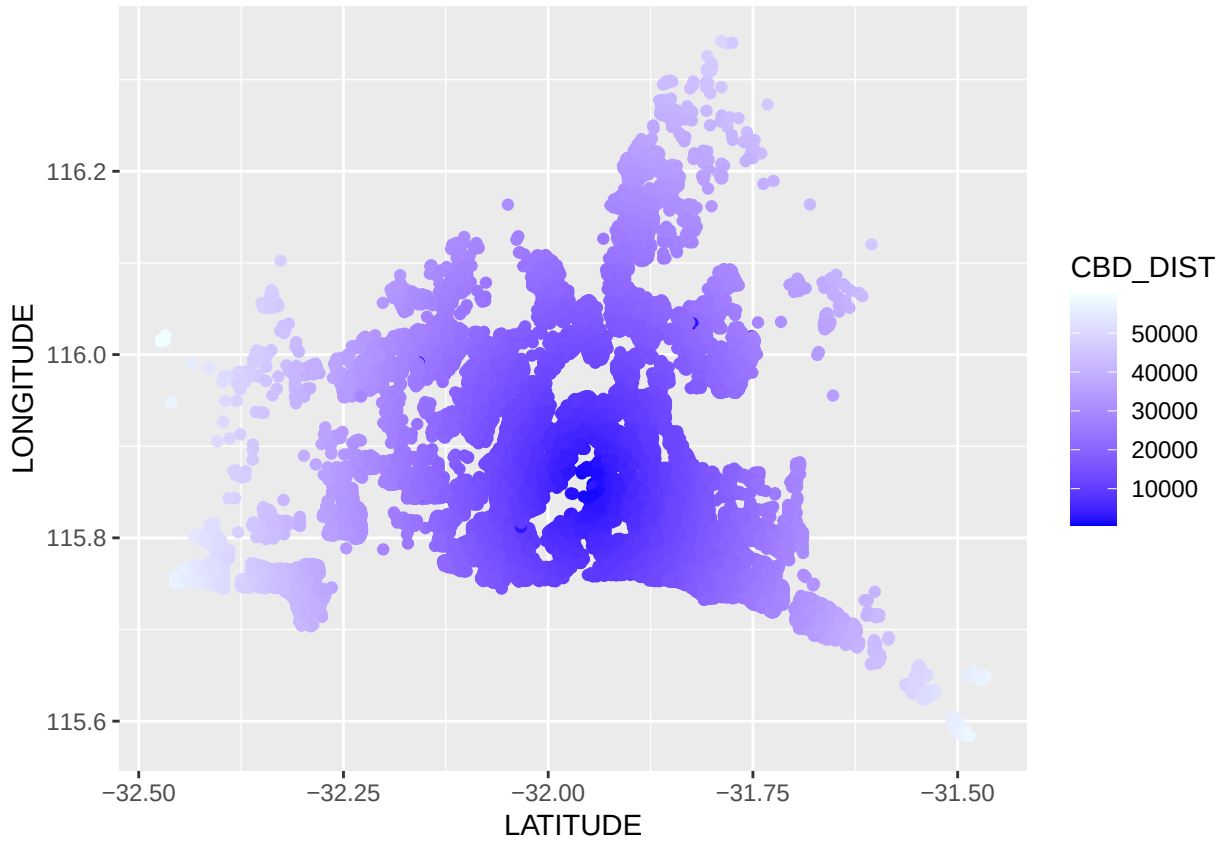
```
plot_3 <- plot_3a / plot_3b +
          plot_layout(axes = "collect")

plot_3
```




```
# Save plot
ggsave(plot_3,
       filename = "plot_3_Spatial_distribution.png",
       device = "png",
       height = 4, width = 8, units = "in")
```

Just incase, let's take quick look into floor size distribution

```
plot_L <- ggplot(data = house2) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = CBD_DIST)) +
  scale_colour_gradient(low = "blue", high = "azure")

plot_L
```



5. Regression Assumption Check and Variable Selection

Load packages

```
## For multicolinearity check
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.3.3
```

```
## corrplot 0.94 loaded
```

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```r
## For assumption check
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
# For error distribution check
library("ggpubr")
```

```
## Warning: package 'ggpubr' was built under R version 4.3.3
```

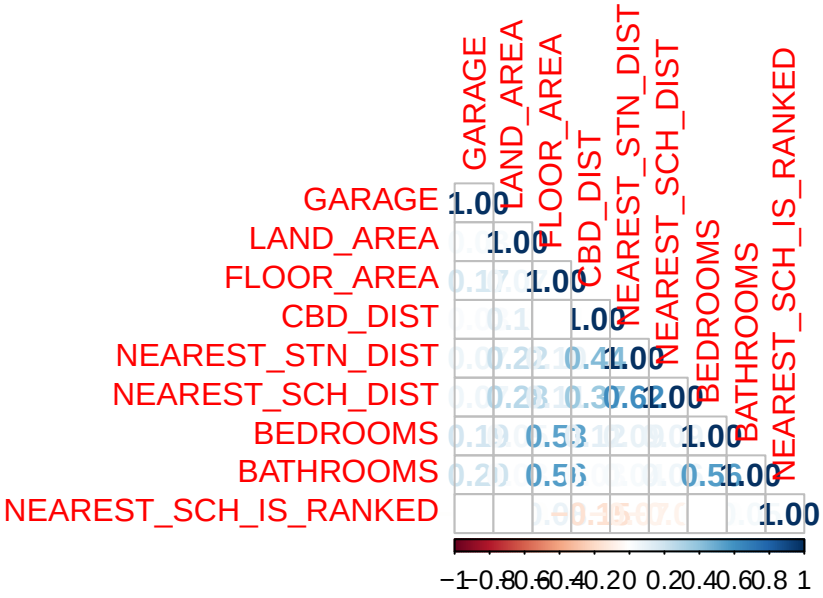
Dependent variable in this analysis is "PRICE".

```r
dep_var <- c("PRICE")

# There are only continuous and binary variable
ind_vars <- c(cont_cols, bin_cols)
ind_vars <- remove_columns(ind_vars, dep_var)

# For variable collection
pca_vars <- c(dep_var, ind_vars)
```

5.1. Variable selection based on correlation

```r
ind_df <- house2 %>%
  as.data.frame() %>% dplyr :: select(all_of(ind_vars))
```

```r
res1 <- cor.mtest(ind_df, conf.level=0.95)
corrmatrix <-cor(ind_df)
corrplot(corrmatrix, method="number", type="lower", p.mat=res1$p, sig.level = 0.05, insig="blank")
```



Check correlation value

```
melted_corr <- melt(corrmatrix)
colnames(melted_corr) <- c("Variable1", "Variable2", "Correlation")

filtered_corr <- subset(melted_corr, abs(Correlation) > 0.5 & abs(Correlation) < 1)
sorted_corr <- filtered_corr[order(-abs(filtered_corr$Correlation)), ]

print(sorted_corr)
```

```
##           Variable1        Variable2 Correlation
## 42 NEAREST_SCH_DIST NEAREST_STN_DIST   0.6201047
## 50 NEAREST_STN_DIST NEAREST_SCH_DIST   0.6201047
## 26        BATHROOMS       FLOOR_AREA   0.5569730
## 66       FLOOR_AREA        BATHROOMS   0.5569730
## 62        BATHROOMS         BEDROOMS   0.5553268
## 70         BEDROOMS        BATHROOMS   0.5553268
## 25         BEDROOMS       FLOOR_AREA   0.5320232
## 57       FLOOR_AREA         BEDROOMS   0.5320232
```

- There are 5 variable that have inner correlation: “NEAREST_SCH_DIST”, “NEAREST_STN_DIST”, “BATHROOMS”, “BEDROOMS”, and “BATHROOMS”.
- Since we are focused on investigating the effect “NEAREST_SCH_DIST” will be kept rather than the “NEAREST_STN_DIST”.
- “FLOOR_AREA” will be kept instead of “BEDROOMS” and “BATHROOMS”, since it offer more explanation of the house.

Omits variables that have strong multicollinearity

```
omit_vars <- c("NEAREST_STN_DIST","BEDROOMS", "BATHROOMS")

cont_cols <- remove_columns(cont_cols, omit_vars)
ind_vars <- remove_columns(ind_vars, omit_vars)

reg_vars <- c(dep_var, ind_vars)
```

Recheck multicollinearity

```
ind_df <- house2 %>%
  as.data.frame() %>% dplyr :: select(all_of(ind_vars))

res1 <- cor.mtest(ind_df, conf.level=0.95)
corrmatrix <-cor(ind_df)

melted_corr <- melt(corrmatrix)
colnames(melted_corr) <- c("Variable1", "Variable2", "Correlation")

filtered_corr <- subset(melted_corr, abs(Correlation) > 0.5 & abs(Correlation) < 1)
sorted_corr <- filtered_corr[order(-abs(filtered_corr$Correlation)), ]

print(sorted_corr)
```

```
## [1] Variable1   Variable2   Correlation
## <0 rows> (or 0-length row.names)
```

- There's no more multicollinearity problem

5.2. Best model Selection using resgression subset method

Load Packages

```
library(olsrr)
```

```
## Warning: package 'olsrr' was built under R version 4.3.3
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.3.3
```

```r
house_reg <- house2 %>%
  select(any_of(reg_vars))
```

```r
lm1 <- lm(house_reg, formula = PRICE ~.)
ols_regress(lm1)
```

```
##                                Model Summary
## ---------------------------------------------------------------------------
## R                            0.682       RMSE                  261637.001
## R-Squared                    0.465       MSE                68453920311.549
## Adj. R-Squared               0.465       Coef. Var                 40.845
## Pred R-Squared               0.464       AIC                   833023.791
## MAE                     181048.731       SBC                   833090.257
## ---------------------------------------------------------------------------
##  RMSE: Root Mean Square Error
##  MSE: Mean Square Error
##  MAE: Mean Absolute Error
##  AIC: Akaike Information Criteria
##  SBC: Schwarz Bayesian Criteria
##
##                                    ANOVA
## --------------------------------------------------------------------------------
##                     Sum of
##                    Squares           DF        Mean Square       F          Sig.
## --------------------------------------------------------------------------------
## Regression    1.785394e+15            6       2.975656e+14    4345.933    0.0000
## Residual      2.052112e+15        29971    68469908348.057
## Total         3.837505e+15        29977
## --------------------------------------------------------------------------------
##
##                                        Parameter Estimates
## ----------------------------------------------------------------------------------------------------
##                  model          Beta    Std. Error    Std. Beta       t        Sig          lower
## ----------------------------------------------------------------------------------------------------
##            (Intercept)    275269.688      5646.314                48.752    0.000    264202.669
##                 GARAGE     11681.809      1059.509        0.047   11.026    0.000      9605.125
##              LAND_AREA         1.098         0.092        0.053   11.956    0.000         0.918
##             FLOOR_AREA      2630.391        21.575        0.528  121.920    0.000      2588.104
##               CBD_DIST       -11.750         0.145       -0.372  -81.070    0.000       -12.034
##      NEAREST_SCH_DIST      9248.329       987.176        0.044    9.368    0.000      7313.422
## NEAREST_SCH_IS_RANKED    101200.299      3288.033        0.132   30.778    0.000     94755.612
## ----------------------------------------------------------------------------------------------------
```

- All of the independent variable are significant.
- With all if the independent variable included, the model can explain 46% of the data.

Check model variation

```r
best_subset <-
  regsubsets(PRICE~., data = house_reg, nbest = 1, nvmax = NULL,
             method="exhaustive")
```

```
summary_best_subset <- summary(best_subset)
as.data.frame(summary_best_subset$outmat)
```

```
##          GARAGE LAND_AREA FLOOR_AREA CBD_DIST NEAREST_SCH_DIST
## 1  ( 1 )                          *
## 2  ( 1 )                          *        *
## 3  ( 1 )                          *        *
## 4  ( 1 )               *          *        *
## 5  ( 1 )      *        *          *        *
## 6  ( 1 )      *        *          *        *                *
##          NEAREST_SCH_IS_RANKED
## 1  ( 1 )
## 2  ( 1 )
## 3  ( 1 )                    *
## 4  ( 1 )                    *
## 5  ( 1 )                    *
## 6  ( 1 )                    *
```

Review the best model selection

```
best_subset <-
  regsubsets(PRICE~., data = house_reg, nbest = 3, nvmax = NULL,
             method="exhaustive")

best_subset
```

```
## Subset selection object
## Call: regsubsets.formula(PRICE ~ ., data = house_reg, nbest = 3, nvmax = NULL,
##     method = "exhaustive")
## 6 Variables  (and intercept)
##                       Forced in Forced out
## GARAGE                    FALSE      FALSE
## LAND_AREA                 FALSE      FALSE
## FLOOR_AREA                FALSE      FALSE
## CBD_DIST                  FALSE      FALSE
## NEAREST_SCH_DIST          FALSE      FALSE
## NEAREST_SCH_IS_RANKED     FALSE      FALSE
## 3 subsets of each size up to 6
## Selection Algorithm: exhaustive
```

Check variable count in best performing model.

```
which.max(summary_best_subset$adjr2)
```

```
## [1] 6
```

- The best performing model includes all of the independent variable.

Review the best model performances

```r
best_model1 <- lm(house_reg, formula = PRICE ~.)

ols_regress(best_model1)
```

```
##                                Model Summary
## ----------------------------------------------------------------------------
## R                            0.682       RMSE                  261637.001
## R-Squared                    0.465       MSE                68453920311.549
## Adj. R-Squared               0.465       Coef. Var                40.845
## Pred R-Squared               0.464       AIC                  833023.791
## MAE                     181048.731       SBC                  833090.257
## ----------------------------------------------------------------------------
##  RMSE: Root Mean Square Error
##  MSE: Mean Square Error
##  MAE: Mean Absolute Error
##  AIC: Akaike Information Criteria
##  SBC: Schwarz Bayesian Criteria
##
##                                    ANOVA
## --------------------------------------------------------------------------------
##                     Sum of
##                    Squares           DF        Mean Square       F          Sig.
## --------------------------------------------------------------------------------
## Regression    1.785394e+15            6       2.975656e+14    4345.933    0.0000
## Residual      2.052112e+15        29971    68469908348.057
## Total         3.837505e+15        29977
## --------------------------------------------------------------------------------
##
##                                        Parameter Estimates
## ----------------------------------------------------------------------------------------------------
##                 model          Beta    Std. Error    Std. Beta       t        Sig          lower
## ----------------------------------------------------------------------------------------------------
##           (Intercept)    275269.688      5646.314                48.752    0.000    264202.669
##                GARAGE     11681.809      1059.509        0.047   11.026    0.000      9605.125
##             LAND_AREA         1.098         0.092        0.053   11.956    0.000         0.918
##            FLOOR_AREA      2630.391        21.575        0.528  121.920    0.000      2588.104
##              CBD_DIST       -11.750         0.145       -0.372  -81.070    0.000       -12.034
##      NEAREST_SCH_DIST      9248.329       987.176        0.044    9.368    0.000      7313.422
## NEAREST_SCH_IS_RANKED    101200.299      3288.033        0.132   30.778    0.000     94755.612
## ----------------------------------------------------------------------------------------------------
```

5.3. Regression assumption check

5.3.1. Multicollinearity recheck (VIF)

```r
car::vif(best_model1)
```

```
##                GARAGE             LAND_AREA            FLOOR_AREA
##              1.031753              1.086467              1.051786
##              CBD_DIST      NEAREST_SCH_DIST NEAREST_SCH_IS_RANKED
##              1.179064              1.247802              1.033105
```

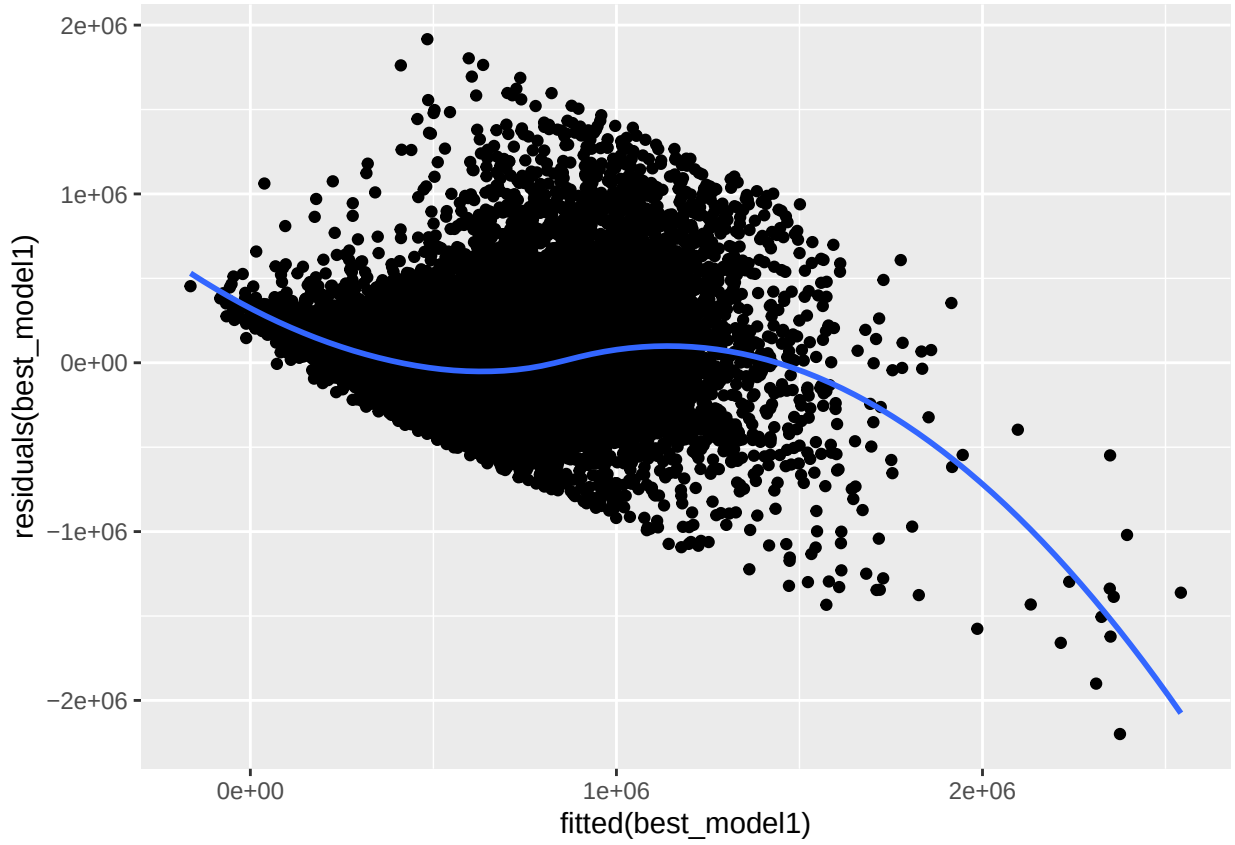
- There's no multicollinearity problem since all VIF value is less than 5.

5.3.2. Homoskedasticity check

Residuals distribution on fitted values

```
ggplot(house_reg, aes(x=fitted(best_model1), y= residuals(best_model1)))+
  geom_point() +
  geom_smooth(method = "loess", se = FALSE)
```

```
## `geom_smooth()` using formula = 'y ~ x'
```



- There residuals distrtibution is not constant especially in the fitted model mean. - This suggest a heteroscedasticity problem.

Breusch-Pagan test

```
ols_test_breusch_pagan(best_model1, rhs = TRUE, multiple = TRUE)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
```

```
##                                        Data
##  ---------------------------------------------------------------------------------------
##  Response : PRICE
##  Variables: GARAGE LAND_AREA FLOOR_AREA CBD_DIST NEAREST_SCH_DIST NEAREST_SCH_IS_RANKED
##
##                Test Summary (Unadjusted p values)
##  ------------------------------------------------------------
##   Variable                    chi2        df          p
##  ------------------------------------------------------------
##   GARAGE                     72.63366     1     1.560942e-17
##   LAND_AREA                2314.99096     1     0.000000e+00
##   FLOOR_AREA               8069.35056     1     0.000000e+00
##   CBD_DIST                 2011.14737     1     0.000000e+00
##   NEAREST_SCH_DIST          177.14688     1     2.034230e-40
##   NEAREST_SCH_IS_RANKED    1387.38440     1    1.158334e-303
##  ------------------------------------------------------------
##   simultaneous            13132.72328     6     0.000000e+00
##  ------------------------------------------------------------
```

- The p value is less than 0.05, which means the null hypothesis (homoscedasticity) is rejected and therefore the model have a heteroscedasticity problem.
- Furthermore these variables don't have heteroscedasticity problems: GARAGE, NEAREST_SCH_DIST, and NEAREST_SCH_IS_RANKED.

5.3.3. Autocorrelation

Durbin-Watson test

```
at_test <- dwtest(best_model1)

print(at_test)
```

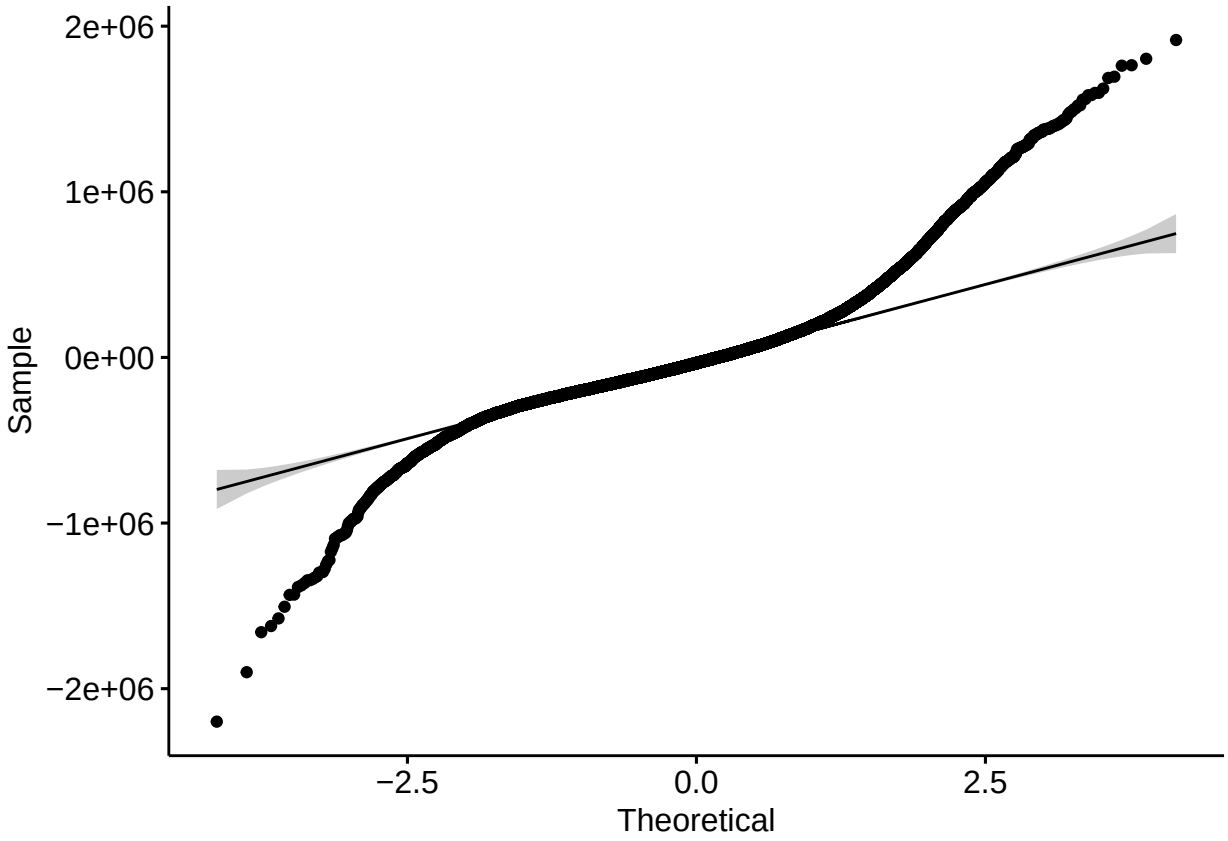
```
##
##  Durbin-Watson test
##
## data:  best_model1
## DW = 1.9742, p-value = 0.01265
## alternative hypothesis: true autocorrelation is greater than 0
```

- Null hypothesis is that there is no significant autocorrelation.
- The null hypotehsis is rejected which means that there is evidence for autocorrelation but the DW value is normal (1.5 - 2) and doesn't support positive autocorrelation.
- Therefore it is concluded that there is no autocorrelation problem

5.3.4. Normally distributed errors (residuals).

Q-Q plot

```
ggqqplot(residuals(best_model1))
```



Shapiro-wilk test

```
jarque.bera.test(residuals(best_model1))
```

```
##
##  Jarque Bera Test
##
## data:  residuals(best_model1)
## X-squared = 44406, df = 2, p-value < 2.2e-16
```

- P-value is less than 0.05, which measn the null hypothesis is rejected and therefore the residuals is therefore not normally distributed.

Recap: - The variable have: - Outliers - Positive Skewed distribution - 0 value - positive value - The model have: - heteroskedasticity problem - Skewed error distribution

6. Spatial checking

Load Package

```
library(GWmodel)
```

```
## Loading required package: robustbase
```

```
## Loading required package: sp

## The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
## which was just loaded, will retire in October 2023.
## Please refer to R-spatial evolution reports for details, especially
## https://r-spatial.org/r/2023/05/15/evolution4.html.
## It may be desirable to make the sf package available;
## package maintainers should consider adding sf to Suggests:.
## The sp package is now running under evolution status 2
##      (status 2 uses the sf package in place of rgdal)

## Loading required package: Rcpp

## Welcome to GWmodel version 2.3-1.
```

```
library(sf)
```

```
## Linking to GEOS 3.11.2, GDAL 3.6.2, PROJ 9.2.0; sf_use_s2() is TRUE
```

```
library(spdep)
```

```
## Loading required package: spData

## To access larger datasets in this package, install the spDataLarge
## package with: `install.packages('spDataLarge',
## repos='https://nowosad.github.io/drat/', type='source')`

##
## Attaching package: 'spData'

## The following object is masked _by_ '.GlobalEnv':
##
##     house
```

6.1. Spatial transformation

Convert the dataframe into sf object that can hold geometry data. - Also making sure that the coordinates to a commonly used 3414 SRID

```
house2.sf <- st_as_sf(house2, coords = c("LONGITUDE", "LATITUDE"),
                          crs=4326) %>%
  st_transform(crs=3414)
```

Also convert the same to model residuals incase it's needed for further analysis.

```
## Collect residuals into a dataframe.
best_model1.output <- as.data.frame(best_model1$residuals)

house2.res.sf <- cbind(house2.sf,
                       best_model1$residuals)
house2.res.sf
```

```
## Simple feature collection with 29978 features and 19 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 1148035 ymin: -3772865 xmax: 1216891 ymax: -3656164
## Projected CRS: SVY21 / Singapore TM
## First 10 features:
##                ADDRESS       SUBURB  PRICE BEDROOMS BATHROOMS GARAGE LAND_AREA
## 1        1 Acorn Place   South Lake 565000        4         2      2       600
## 2          1 Addis Way        Wandi 365000        3         2      2       351
## 3      1 Ainsley Court      Camillo 287000        3         1      1       719
## 4      1 Albert Street     Bellevue 255000        2         1      2       651
## 5         1 Aman Place    Lockridge 325000        4         1      2       466
## 6  1 Amethyst Crescent Mount Richon 409000        4         2      1       759
## 7        1 Ardara Lane      Hilbert 400000        3         2      2       386
## 8       1 Arnside Bend      Waikiki 370000        4         2      2       468
## 9     1 Arrochar Court    Hamersley 565000        4         2      3       875
## 10    1 Arundel Street    Bayswater 685000        3         2      8       552
##    FLOOR_AREA BUILD_YEAR CBD_DIST              NEAREST_STN NEAREST_STN_DIST
## 1         160       2003    18300 Cockburn Central Station             1800
## 2         139       2013    26900          Kwinana Station             4900
## 3          86       1979    22600          Challis Station             1900
## 4          59       1953    17900          Midland Station             3600
## 5         131       1998    11200       Bassendean Station             2000
## 6         118       1991    27300         Armadale Station             1000
## 7         132       2014    28200         Armadale Station             3700
## 8         158       2013    41700          Warnbro Station             1100
## 9         168       1983    12100          Warwick Station             2500
## 10        126       1999     5900        Bayswater Station              508
##    DATE_SOLD POSTCODE                  NEAREST_SCH NEAREST_SCH_DIST
## 1  09-2018\n     6164  LAKELAND SENIOR HIGH SCHOOL        0.8283386
## 2  02-2019\n     6167               ATWELL COLLEGE        5.5243244
## 3  06-2015\n     6111 KELMSCOTT SENIOR HIGH SCHOOL        1.6491782
## 4  07-2018\n     6056 SWAN VIEW SENIOR HIGH SCHOOL        1.5714009
## 5  11-2016\n     6054                KIARA COLLEGE        1.5149216
## 6  03-2013\n     6112  ARMADALE SENIOR HIGH SCHOOL        1.2272192
## 7  05-2016\n     6112        DALE CHRISTIAN SCHOOL        2.4857309
## 8  03-2019\n     6169  SOUTH COAST BAPTIST COLLEGE        0.4915731
## 9  06-2020\n     6022   WARWICK SENIOR HIGH SCHOOL        1.2219276
## 10 10-2019\n     6053    CHISHOLM CATHOLIC COLLEGE        0.9362433
##                       COORDS NEAREST_SCH_IS_RANKED best_model1.residuals
## 1        115.84245,-32.1159                     0             52212.924
## 2  115.8595535,-32.19347028                     1           -135854.407
## 3   115.993579,-32.12057813                     1            -77852.903
## 4  116.0380088,-31.90054675                     0             -3745.708
## 5       115.94778,-31.88579                     0           -201134.612
## 6   116.0236578,-32.1538014                     0            120259.194
## 7  115.9886473,-32.17997786                     0             62097.347
## 8  115.7619682,-32.31766224                     0            140687.106
## 9  115.8227005,-31.84775987                     0            -57304.960
## 10      115.90705,-31.91788                     1            -56292.299
##                     geometry
## 1  POINT (1164968 -3731000)
## 2  POINT (1165613 -3739869)
## 3  POINT (1179308 -3733164)
```

```
## 4  POINT (1186368 -3709009)
## 5  POINT (1177935 -3706375)
## 6  POINT (1181747 -3737213)
## 7  POINT (1178076 -3739761)
## 8  POINT (1154757 -3752719)
## 9  POINT (1166463 -3700769)
## 10 POINT (1173638 -3709527)
```

Since the data is point based, it is based to convert it to SpatialPoints (sp) object.

```
house2.sp <- as_Spatial(house2.res.sf)
```

6.1.1. Spatial Heterogeneity check

Load Packages

```
library(fastDummies)
```

```
## Warning: package 'fastDummies' was built under R version 4.3.3
```

I'm going check spatial heterogeneity by looking if every suburb have its own effect on house price. But first, I need to established suburb dummy and run the regression without alpha.

```
## generate district dummy
house2.h <- dummy_cols(house2,
                   select_columns = "SUBURB")
```

Run the Regression

```
d.h_model <- lm(PRICE ~ GARAGE + LAND_AREA + FLOOR_AREA + CBD_DIST +
                  NEAREST_SCH_IS_RANKED + NEAREST_SCH_DIST + SUBURB - 1, data = house2)
```

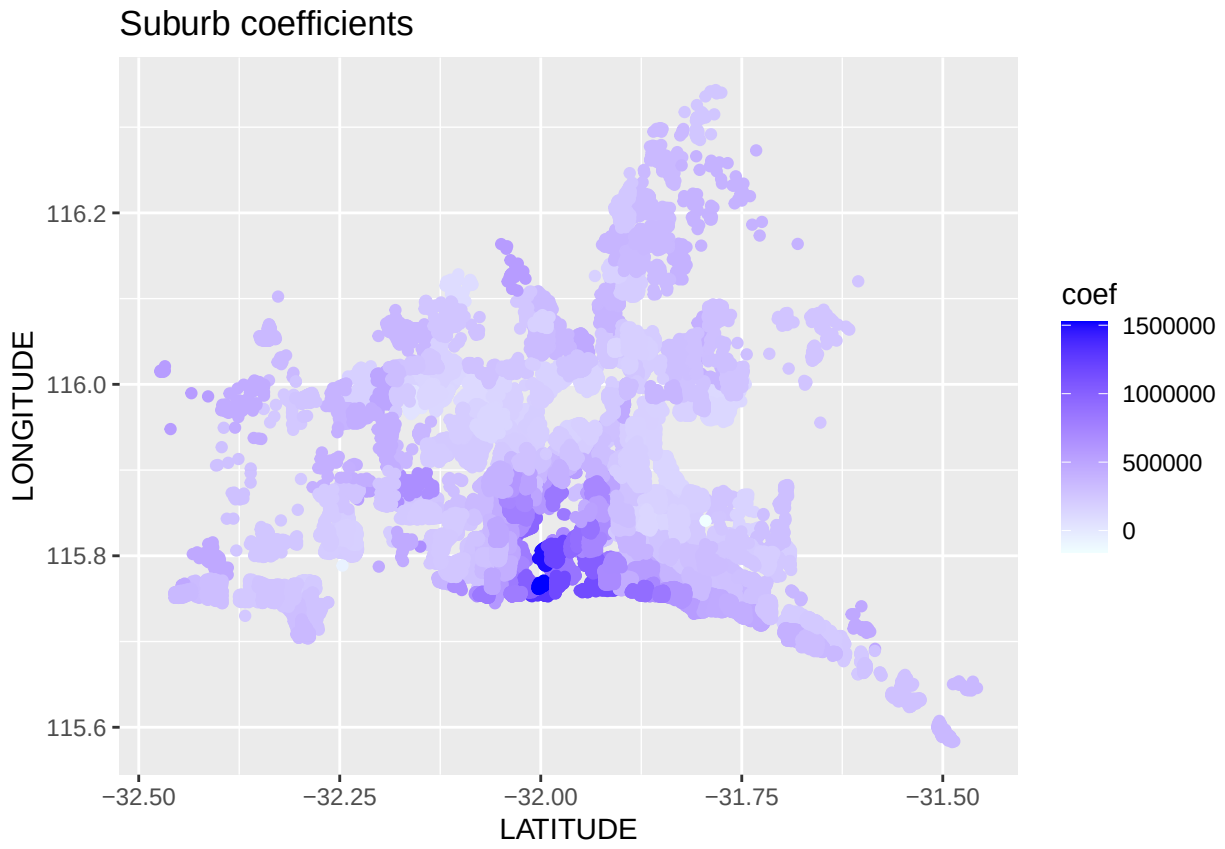
Extract individuals suburb coefficients

```
#extract suburb coef
sub.names <- d.h_model$coefficients %>%
  as.data.frame() %>%
  row.names() %>%
  str_replace("SUBURB","")

#Label other dataframe column
sub.coef <- data.frame(
  coef = d.h_model$coefficients,
  SUBURB = sub.names,
  row.names = sub.names)

# Merge with original dataframe
house.h <- merge(house, sub.coef, by='SUBURB', all.y=TRUE)
house.h$coef <- as.integer(house.h$coef)
```

Plot Suburb coef

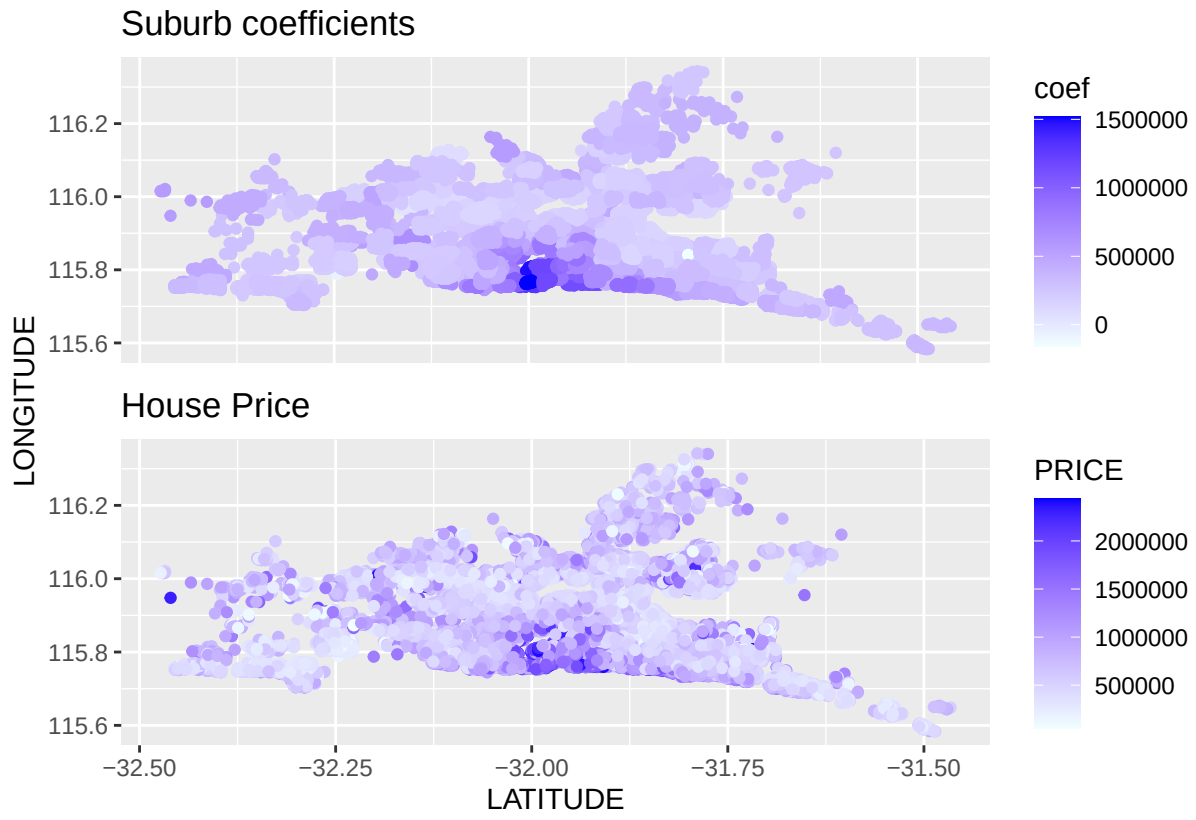
```
plot_sub.h <- ggplot(data = house.h) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = coef)) +
  scale_colour_gradient(low = "azure", high = "blue") +
  ggtitle("Suburb coefficients")

plot_sub.h
```

```
## Warning: Removed 6 rows containing missing values (‘geom_point()‘).
```




```
plot_h <- plot_sub.h / plot_3a +
          plot_layout(axes = "collect")

plot_h
```

```
## Warning: Removed 6 rows containing missing values (‘geom_point()‘).
```



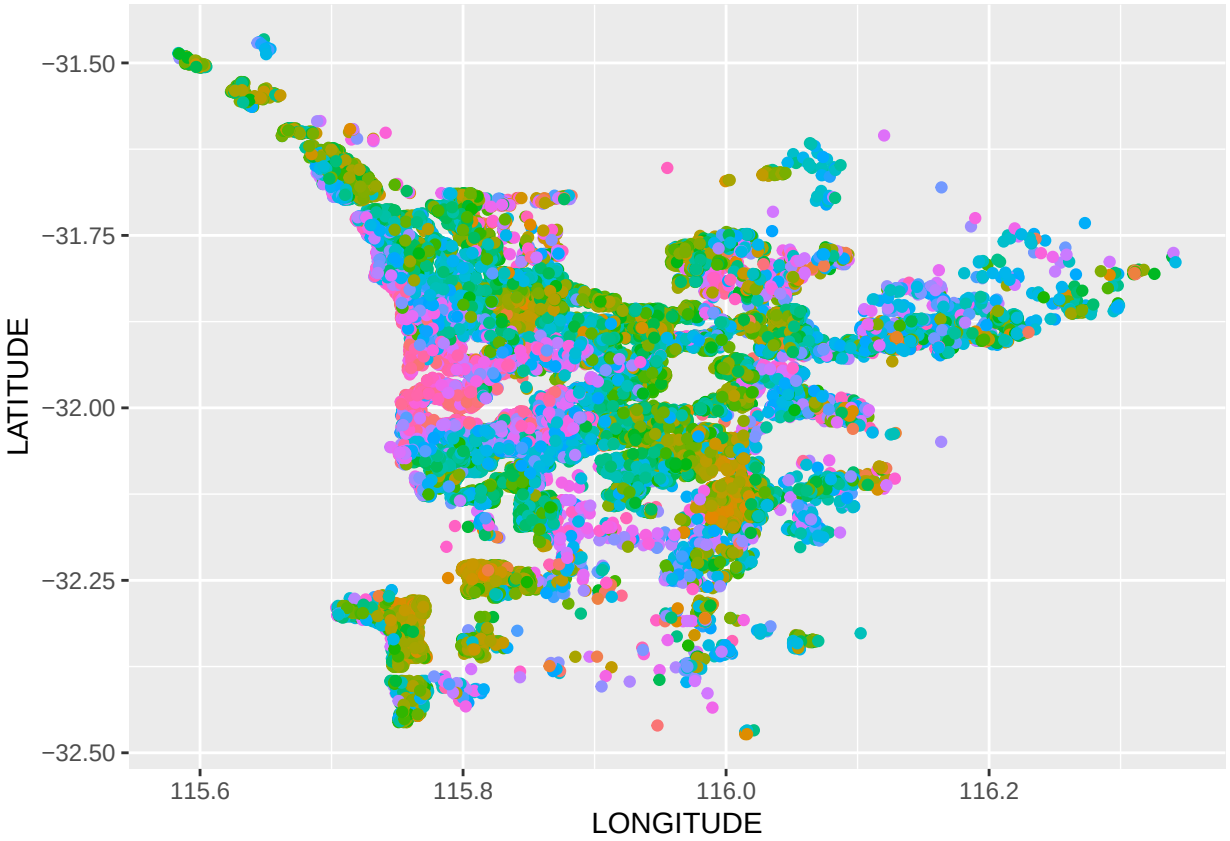

```
## Save plot
ggsave(plot_h,
       filename = "plot_h_spatial_het.png",
       device = "png",
       height = 4, width = 8, units = "in")
```

```
## Warning: Removed 6 rows containing missing values (‘geom_point()‘).
```

7.2. Establish Spatial Weight Matrices

Spatial Distribution Check

```
ggplot() +
  geom_point(aes(x=LONGITUDE, y=LATITUDE, color = factor(PRICE)), data=house2,
             show.legend=FALSE)
```



- There few noticeable points that are distributed far from the rest.

7.2.1. Checking for optimum distance

Let's see the best distance according to the optimum number of dot distribution.

Isolate the coords to a new df

```
coords.h2 <- house2[c("LONGITUDE","LATITUDE")]
head(coords.h2)
```

```
##   LONGITUDE  LATITUDE
## 1  115.8424 -32.11590
## 2  115.8596 -32.19347
## 3  115.9936 -32.12058
## 4  116.0380 -31.90055
## 5  115.9478 -31.88579
## 6  116.0237 -32.15380
```

Check the maximal and minal distance

```
k1 <- knn2nb(knearneigh(coords.h2))
k1dists <- unlist(nbdists(k1, coords.h2, longlat = TRUE))
summary(k1dists)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.003813 0.036410 0.061069 0.086877 0.104117 5.712935
```

The maximal distance between point per observation is 5.713 km, meanwhile the minimum is 3.9 meters.

Make weight matrices

```
wm_d1 <- dnearneigh(coords.h2, 0 , 5.713, longlat = TRUE)
wm_d1
```

Spatial Weight - In this instance the weight is W (row-standardized).

```
nb_lw <- nb2listw(wm_d1, style = 'W')
summary(nb_lw)
```

7.2.2. Moran's I test

To check for spatial dependence, I use Moran's I Test

```
model1.moran <- lm.morantest(best_model1, nb_lw)

model1.moran
```

```
##
##  Global Moran I for regression residuals
##
## data:
## model: lm(formula = PRICE ~ ., data = house_reg)
## weights: nb_lw
##
## Moran I statistic standard deviate = 1215.6, p-value < 2.2e-16
## alternative hypothesis: greater
## sample estimates:
## Observed Moran I      Expectation         Variance
##     3.246610e-01    -1.005493e-04     7.137572e-08
```

P value is less than 0.05 which signs spatial correlations. The global Moran Value is more than 0, which signs clustered distribution.

7.4. Bulding Fixed Bandwidth

Load formula

```
best_model.f <- (PRICE~GARAGE+ LAND_AREA + FLOOR_AREA + CBD_DIST
                 + NEAREST_SCH_DIST + NEAREST_SCH_IS_RANKED)
```

```
bw.fixed <- bw.gwr(best_model.f, data=house2.sp, approach = "CV", kernel = "gaussian"
                   , adaptive = FALSE, longlat = TRUE)
```

```
## Take a cup of tea and have a break, it will take a few minutes.
##           -----A kind suggestion from GWmodel development group
## Fixed bandwidth: 83754.03 CV score: 2.055707e+15
## Fixed bandwidth: 51773.19 CV score: 2.055682e+15
## Fixed bandwidth: 32007.94 CV score: 2.055621e+15
## Fixed bandwidth: 19792.35 CV score: 2.055473e+15
## Fixed bandwidth: 12242.69 CV score: 2.055167e+15
## Fixed bandwidth: 7576.753 CV score: 2.054893e+15
## Fixed bandwidth: 4693.042 CV score: 2.056669e+15
## Fixed bandwidth: 9358.984 CV score: 2.054953e+15
## Fixed bandwidth: 6475.273 CV score: 2.055061e+15
## Fixed bandwidth: 8257.504 CV score: 2.054893e+15
## Fixed bandwidth: 8678.232 CV score: 2.05491e+15
## Fixed bandwidth: 7997.48 CV score: 2.054887e+15
## Fixed bandwidth: 7836.777 CV score: 2.054887e+15
## Fixed bandwidth: 7737.456 CV score: 2.054888e+15
## Fixed bandwidth: 7898.16 CV score: 2.054887e+15
## Fixed bandwidth: 7936.097 CV score: 2.054887e+15
## Fixed bandwidth: 7874.714 CV score: 2.054887e+15
## Fixed bandwidth: 7912.651 CV score: 2.054887e+15
## Fixed bandwidth: 7889.204 CV score: 2.054887e+15
## Fixed bandwidth: 7903.695 CV score: 2.054887e+15
## Fixed bandwidth: 7907.116 CV score: 2.054887e+15
## Fixed bandwidth: 7901.581 CV score: 2.054887e+15
## Fixed bandwidth: 7905.001 CV score: 2.054887e+15
## Fixed bandwidth: 7905.809 CV score: 2.054887e+15
## Fixed bandwidth: 7904.502 CV score: 2.054887e+15
## Fixed bandwidth: 7905.31 CV score: 2.054887e+15
## Fixed bandwidth: 7904.811 CV score: 2.054887e+15
## Fixed bandwidth: 7905.119 CV score: 2.054887e+15
## Fixed bandwidth: 7904.929 CV score: 2.054887e+15
## Fixed bandwidth: 7905.046 CV score: 2.054887e+15
## Fixed bandwidth: 7905.074 CV score: 2.054887e+15
## Fixed bandwidth: 7905.029 CV score: 2.054887e+15
## Fixed bandwidth: 7905.019 CV score: 2.054887e+15
## Fixed bandwidth: 7905.036 CV score: 2.054887e+15
## Fixed bandwidth: 7905.025 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
## Fixed bandwidth: 7905.033 CV score: 2.054887e+15
## Fixed bandwidth: 7905.031 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
## Fixed bandwidth: 7905.031 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
## Fixed bandwidth: 7905.032 CV score: 2.054887e+15
```

```
bw.fixed
```

```
## [1] 7905.032
```

```
gwr.fixed <- gwr.basic(formula = best_model.f, data=house2.sp, bw=bw.fixed, kernel="gaussian",
              longlat=TRUE)

gwr.fixed
```

```
##    ************************************************************************
##    *                       Package   GWmodel                              *
##    ************************************************************************
##    Program starts at: 2025-05-12 02:55:49.335544
##    Call:
##    gwr.basic(formula = best_model.f, data = house2.sp, bw = bw.fixed,
##     kernel = "gaussian", longlat = TRUE)
##
##    Dependent (y) variable:  PRICE
##    Independent variables:  GARAGE LAND_AREA FLOOR_AREA CBD_DIST NEAREST_SCH_DIST NEAREST_SCH_IS_RANKE
##    Number of data points: 29978
##    ************************************************************************
##    *                    Results of Global Regression                      *
##    ************************************************************************
##
##    Call:
##     lm(formula = formula, data = data)
##
##    Residuals:
##      Min       1Q   Median       3Q      Max
## -2198974  -150385   -34551   100799  1916496
##
##    Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
##    (Intercept)            2.753e+05  5.646e+03  48.752   <2e-16 ***
##    GARAGE                 1.168e+04  1.060e+03  11.026   <2e-16 ***
##    LAND_AREA              1.098e+00  9.180e-02  11.956   <2e-16 ***
##    FLOOR_AREA             2.630e+03  2.157e+01 121.920   <2e-16 ***
##    CBD_DIST              -1.175e+01  1.449e-01 -81.070   <2e-16 ***
##    NEAREST_SCH_DIST       9.248e+03  9.872e+02   9.368   <2e-16 ***
##    NEAREST_SCH_IS_RANKED  1.012e+05  3.288e+03  30.778   <2e-16 ***
##
##    ---Significance stars
##    Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##    Residual standard error: 261700 on 29971 degrees of freedom
##    Multiple R-squared: 0.4652
##    Adjusted R-squared: 0.4651
##    F-statistic:  4346 on 6 and 29971 DF,  p-value: < 2.2e-16
##    ***Extra Diagnostic information
##    Residual sum of squares: 2.052112e+15
##    Sigma(hat): 261645.7
##    AIC:  833023.8
##    AICc:  833023.8
##    BIC:  803194.7
##    ************************************************************************
##    *          Results of Geographically Weighted Regression               *
##    ************************************************************************
##
```

```
##    ********************Model calibration information*********************
##    Kernel function: gaussian
##    Fixed bandwidth: 7905.032
##    Regression points: the same locations as observations are used.
##    Distance metric: Great Circle distance metric is used.
##
##    ****************Summary of GWR coefficient estimates:******************
##                                Min.     1st Qu.      Median     3rd Qu.
##    Intercept              2.7123e+05  2.7305e+05  2.7521e+05  2.7757e+05
##    GARAGE                 8.8896e+03  1.0055e+04  1.1724e+04  1.3703e+04
##    LAND_AREA              8.0256e-01  8.7921e-01  1.0949e+00  1.4166e+00
##    FLOOR_AREA             2.6121e+03  2.6215e+03  2.6273e+03  2.6350e+03
##    CBD_DIST              -1.1874e+01 -1.1834e+01 -1.1756e+01 -1.1669e+01
##    NEAREST_SCH_DIST       8.2925e+03  8.6165e+03  9.1992e+03  9.6540e+03
##    NEAREST_SCH_IS_RANKED  1.0032e+05  1.0088e+05  1.0118e+05  1.0143e+05
##                                Max.
##    Intercept             279384.604
##    GARAGE                 15277.222
##    LAND_AREA                  1.610
##    FLOOR_AREA              2652.763
##    CBD_DIST                 -11.617
##    NEAREST_SCH_DIST        9930.819
##    NEAREST_SCH_IS_RANKED 102101.773
##    ************************Diagnostic information*************************
##    Number of data points: 29978
##    Effective number of parameters (2trace(S) - trace(S'S)): 19.49046
##    Effective degrees of freedom (n-2trace(S) + trace(S'S)): 29958.51
##    AICc (GWR book, Fotheringham, et al. 2002, p. 61, eq 2.33): 832986.7
##    AIC (GWR book, Fotheringham, et al. 2002,GWR p. 96, eq. 4.22): 832970.3
##    BIC (GWR book, Fotheringham, et al. 2002,GWR p. 61, eq. 2.34): 803126.7
##    Residual sum of squares: 2.048556e+15
##    R-square value:  0.466175
##    Adjusted R-square value:  0.4658277
##
##    ***********************************************************************
##    Program stops at: 2025-05-12 03:04:53.451176
```

- The model can explain 46.6% of the whole data, which is 0.01% improvement of previous model.
- 0.466175

7.5. Building Adaptive Model

```
bw.adaptive <- bw.gwr(best_model.f, data=house2.sp, approach = "CV", kernel = "gaussian"
                  ,adaptive = TRUE, longlat = TRUE)
```

```
## Take a cup of tea and have a break, it will take a few minutes.
##           -----A kind suggestion from GWmodel development group
## Adaptive bandwidth: 18535 CV score: 2.05517e+15
## Adaptive bandwidth: 11463 CV score: 2.05479e+15
## Adaptive bandwidth: 7092 CV score: 2.055198e+15
## Adaptive bandwidth: 14164 CV score: 2.054956e+15
## Adaptive bandwidth: 9793 CV score: 2.054761e+15
```

```
## Adaptive bandwidth: 8761 CV score: 2.054812e+15
## Adaptive bandwidth: 10431 CV score: 2.05476e+15
## Adaptive bandwidth: 10825 CV score: 2.054765e+15
## Adaptive bandwidth: 10187 CV score: 2.054758e+15
## Adaptive bandwidth: 10036 CV score: 2.05476e+15
## Adaptive bandwidth: 10280 CV score: 2.054758e+15
## Adaptive bandwidth: 10338 CV score: 2.054758e+15
## Adaptive bandwidth: 10244 CV score: 2.054757e+15
## Adaptive bandwidth: 10222 CV score: 2.054758e+15
## Adaptive bandwidth: 10258 CV score: 2.054758e+15
## Adaptive bandwidth: 10236 CV score: 2.054758e+15
## Adaptive bandwidth: 10250 CV score: 2.054758e+15
## Adaptive bandwidth: 10241 CV score: 2.054757e+15
## Adaptive bandwidth: 10246 CV score: 2.054757e+15
## Adaptive bandwidth: 10242 CV score: 2.054758e+15
## Adaptive bandwidth: 10244 CV score: 2.054757e+15
```

- The recommend points are 10244.

Run the adaptive GWR

```
gwr.adaptive <- gwr.basic(best_model.f, data=house2.sp, bw=bw.adaptive,
                          kernel = 'gaussian', adaptive=TRUE, longlat = TRUE)

gwr.adaptive
```

```
##    ***********************************************************************
##    *                       Package   GWmodel                             *
##    ***********************************************************************
##    Program starts at: 2025-05-12 04:53:10.756033
##    Call:
##    gwr.basic(formula = best_model.f, data = house2.sp, bw = bw.adaptive,
##     kernel = "gaussian", adaptive = TRUE, longlat = TRUE)
##
##    Dependent (y) variable:  PRICE
##    Independent variables:  GARAGE LAND_AREA FLOOR_AREA CBD_DIST NEAREST_SCH_DIST NEAREST_SCH_IS_RANKI
##    Number of data points: 29978
##    ***********************************************************************
##    *                    Results of Global Regression                     *
##    ***********************************************************************
##
##    Call:
##     lm(formula = formula, data = data)
##
##    Residuals:
##      Min       1Q   Median       3Q      Max
## -2198974  -150385   -34551   100799  1916496
##
##    Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
##    (Intercept)            2.753e+05  5.646e+03  48.752   <2e-16 ***
##    GARAGE                 1.168e+04  1.060e+03  11.026   <2e-16 ***
##    LAND_AREA              1.098e+00  9.180e-02  11.956   <2e-16 ***
```

```
##    FLOOR_AREA             2.630e+03  2.157e+01 121.920   <2e-16 ***
##    CBD_DIST              -1.175e+01  1.449e-01 -81.070   <2e-16 ***
##    NEAREST_SCH_DIST       9.248e+03  9.872e+02   9.368   <2e-16 ***
##    NEAREST_SCH_IS_RANKED  1.012e+05  3.288e+03  30.778   <2e-16 ***
##
##    ---Significance stars
##    Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##    Residual standard error: 261700 on 29971 degrees of freedom
##    Multiple R-squared: 0.4652
##    Adjusted R-squared: 0.4651
##    F-statistic:  4346 on 6 and 29971 DF,  p-value: < 2.2e-16
##    ***Extra Diagnostic information
##    Residual sum of squares: 2.052112e+15
##    Sigma(hat): 261645.7
##    AIC:  833023.8
##    AICc:  833023.8
##    BIC:  803194.7
##    ***********************************************************************
##    *          Results of Geographically Weighted Regression              *
##    ***********************************************************************
##
##    *********************Model calibration information*********************
##    Kernel function: gaussian
##    Adaptive bandwidth: 10244 (number of nearest neighbours)
##    Regression points: the same locations as observations are used.
##    Distance metric: Great Circle distance metric is used.
##
##    ****************Summary of GWR coefficient estimates:******************
##                                Min.     1st Qu.      Median     3rd Qu.
##    Intercept              2.7084e+05  2.7287e+05  2.7520e+05  2.7762e+05
##    GARAGE                 8.6808e+03  1.0086e+04  1.1727e+04  1.3933e+04
##    LAND_AREA              7.6709e-01  8.6922e-01  1.0971e+00  1.4577e+00
##    FLOOR_AREA             2.6117e+03  2.6193e+03  2.6242e+03  2.6345e+03
##    CBD_DIST              -1.1886e+01 -1.1839e+01 -1.1755e+01 -1.1662e+01
##    NEAREST_SCH_DIST       8.0352e+03  8.5451e+03  9.2002e+03  9.6701e+03
##    NEAREST_SCH_IS_RANKED  1.0028e+05  1.0078e+05  1.0115e+05  1.0143e+05
##                                Max.
##    Intercept             279805.9211
##    GARAGE                 15866.4445
##    LAND_AREA                  1.7475
##    FLOOR_AREA              2650.2960
##    CBD_DIST                 -11.5882
##    NEAREST_SCH_DIST        9991.0023
##    NEAREST_SCH_IS_RANKED 102032.0136
##    ************************Diagnostic information*************************
##    Number of data points: 29978
##    Effective number of parameters (2trace(S) - trace(S'S)): 20.43962
##    Effective degrees of freedom (n-2trace(S) + trace(S'S)): 29957.56
##    AICc (GWR book, Fotheringham, et al. 2002, p. 61, eq 2.33): 832984.2
##    AIC (GWR book, Fotheringham, et al. 2002,GWR p. 96, eq. 4.22): 832967
##    BIC (GWR book, Fotheringham, et al. 2002,GWR p. 61, eq. 2.34): 803130.1
##    Residual sum of squares: 2.048287e+15
##    R-square value:  0.4662453
##    Adjusted R-square value:  0.4658811
```

```
##
##    ************************************************************************
##    Program stops at: 2025-05-12 05:03:56.423448
```

- The Adaptive GWR model can explain 46.6% of the data. Which less than both ols (violated assumption) and fixed gwr.
- 0.4658811

7.7. GWR Comparison Check

Make comparison table

```
Model <- c('Gaussian Fixed Model','Gaussian Adaptive Model')
Adj_R2 <- c(gwr.fixed$GW.diagnostic$gwR2.adj, gwr.adaptive$GW.diagnostic$gwR2.adj)
AIC <- c(gwr.fixed$GW.diagnostic$AIC, gwr.adaptive$GW.diagnostic$AIC)
compare_df <- data.frame(Model, Adj_R2, AIC)
print(compare_df)
```

```
##                      Model    Adj_R2      AIC
## 1    Gaussian Fixed Model 0.4658277 832970.3
## 2 Gaussian Adaptive Model 0.4658811 832967.0
```

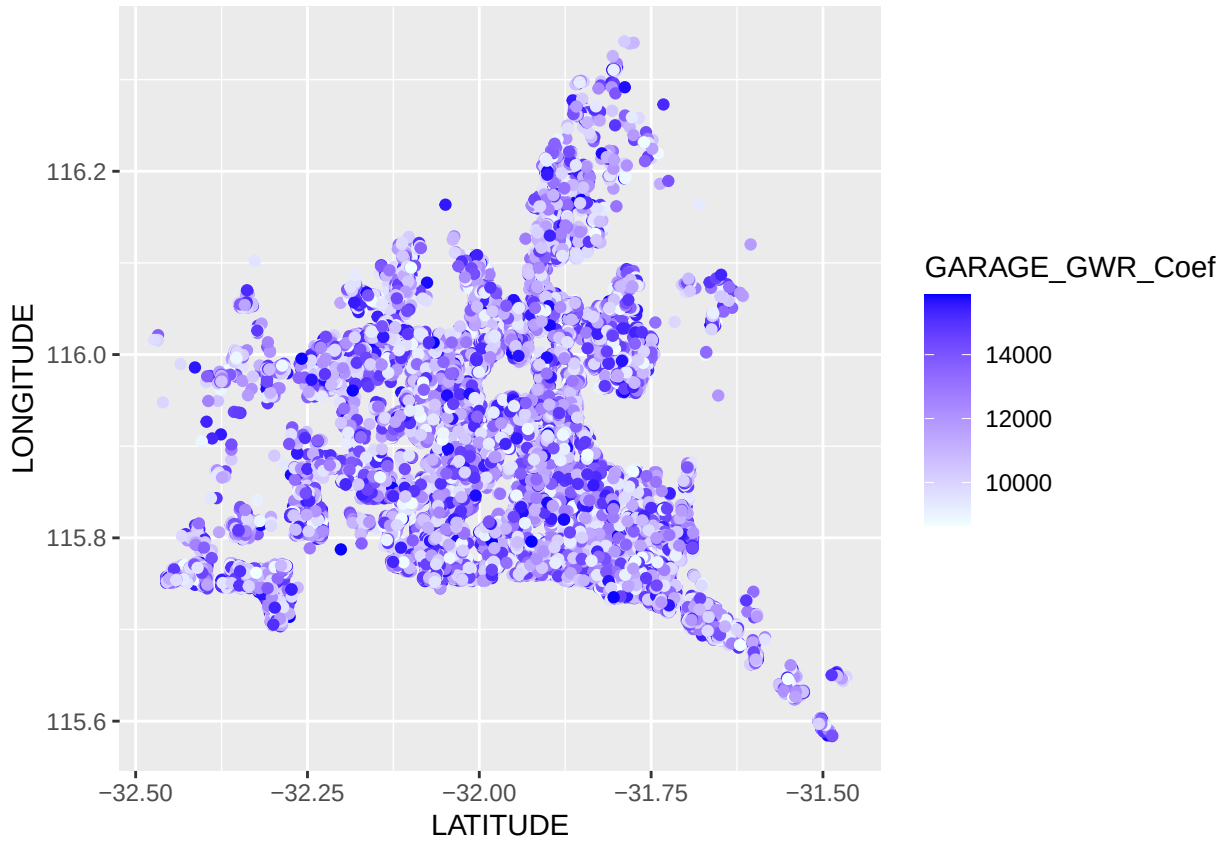
- Both models perform similarly. But the adaptive GWR performs slightly best.

7.8. GWR value spatial spread

```
coef.gar <- gwr.adaptive$SDF$GARAGE
coef.school_distance <- gwr.adaptive$SDF$NEAREST_SCH_DIST
localR2 <- gwr.adaptive$SDF$Local_R2


## Input to copy dataframe
house2_copy<-data.frame(house2)
house2_copy$GARAGE_GWR_Coef<- coef.gar
house2_copy$SCH_DIST_Coef<- coef.school_distance
house2_copy$local_R2<- localR2
```

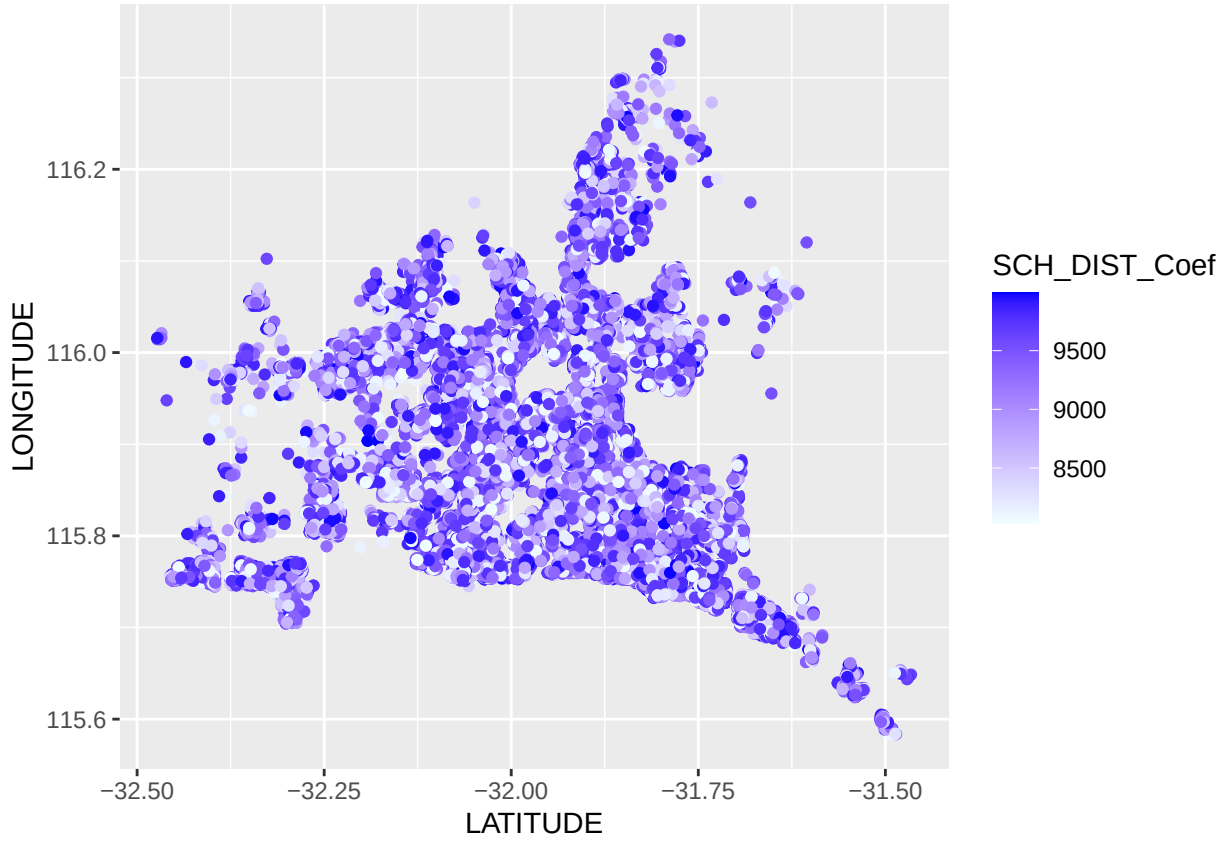
Garage value

```
plot_7a <- ggplot(data = house2_copy) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = GARAGE_GWR_Coef)) +
  scale_colour_gradient(low = "azure", high = "blue")

plot_7a
```



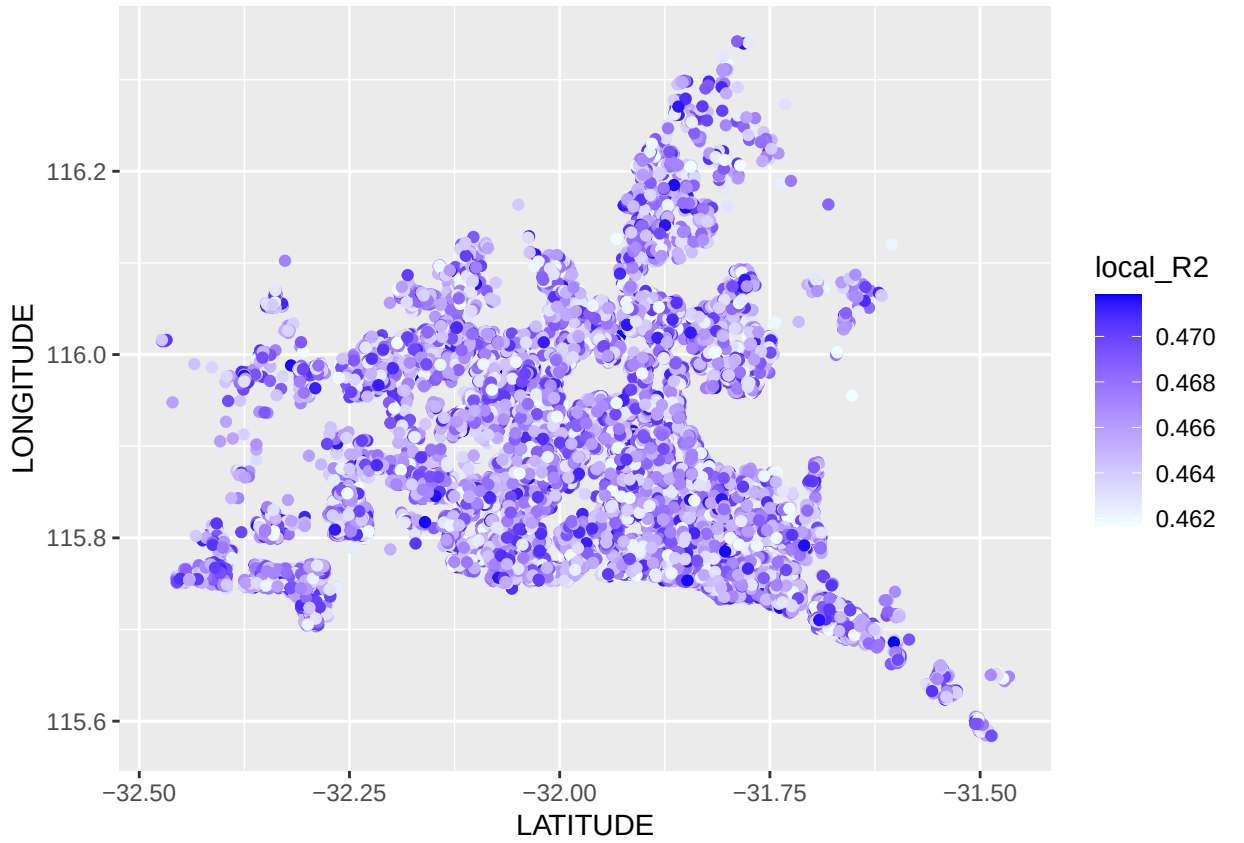
School Distance

```
plot_7b <- ggplot(data = house2_copy) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = SCH_DIST_Coef)) +
  scale_colour_gradient(low = "azure", high = "blue")

plot_7b
```



Local R2

```
plot_7c <- ggplot(data = house2_copy) +
  geom_point(aes(x = LATITUDE, y = LONGITUDE, colour = local_R2)) +
  scale_colour_gradient(low = "azure", high = "blue")

plot_7c
```



There is no obvious clustering, and the value difference is relatively stable. # 8. Model Comparison and conclusion

R squared and AIC

```
Model <- c('OLS','Gaussian Adaptive Model')
Adj_R2 <- c(summary(best_model1)$adj.r.squared, gwr.adaptive$GW.diagnostic$gwR2.adj)
AIC <- c(AIC(best_model1), gwr.adaptive$GW.diagnostic$AIC)
compare_df <- data.frame(Model, Adj_R2, AIC)
print(compare_df)
```

```
##                     Model    Adj_R2      AIC
## 1                     OLS 0.4651415 833023.8
## 2 Gaussian Adaptive Model 0.4658811 832967.0
```

- The adaptive model performs slightly better than OLS