Interpretation of Data Assignment II

A00275664

2023-12-01

Preparing Datasets for Analysis

Tidy the Column Names

```
colnames(PropertyDataset1) <- gsub(" ", "_", colnames(PropertyDataset1))
colnames(PropertyDataset2) <- gsub(" ", "_", colnames(PropertyDataset2))
```

Check for Inconsistencies

PropertyDataset1

Check the new dataset's variables are consistent and understandable

```
PropertyDataset1$raw_address[1:6] #Not consistent, need to capitalise data values
unique(PropertyDataset1$num_bedrooms) #Should be changed to a numeric format
unique(PropertyDataset1$num_bathrooms) #Should be changed to a numeric format
unique(PropertyDataset1$parking) # Not consistent, inputs need to be replaced to Yes/No
str(PropertyDataset1) # check data formats
```

PropertyDataset2

```
PropertyDataset2$Address[1:10] #Not consistent title case
PropertyDataset2$`Price_( )`[1:10] #symbol not recognised
PropertyDataset2$`Description_of_Property`[1:10] #Inconsistent title case
unique(PropertyDataset2$County) #Inconsistent title case and language
unique(PropertyDataset2$`Not_Full_Market_Price`) # Inconsistent title case
unique(PropertyDataset2$VAT_Exclusive) # Inconsistent title case
PropertyDataset2$Property_Size_Description[1:10] # Inconsistent language and title case
str(PropertyDataset2) # Check the structure of dataset
```

Format Modification

PropertyDataset1

PropertyDataset1a$sale_date

To begin cleaning the first Property dataset, we will convert the column data types to their appropriate formats for analysis. The *lubridate* package is used to change the **sale_date** format from a character to date format

```
# Converts from a character format to date format
PropertyDataset1a<-PropertyDataset1
PropertyDataset1a$sale_date<- dmy(PropertyDataset1a$sale_date)
str(PropertyDataset1a$sale_date)
```

```
##  Date[1:40000], format: "2020-05-29" "2019-12-17" "2017-02-09" "2018-12-19" "2017-03-31" ...
```

PropertyDataset1a$num_bedrooms

Next, **num_bedrooms** format will be modified to a numeric data format. Advantages for this include

- Consistency within variables
- Easier manipulation of data
- Better variety for data visualisation options

```
# Replacing the new dataframe PropertyDataset1a num_bedrooms Character values with Numeric values
PropertyDataset1a <- PropertyDataset1a %>%
  mutate(num_bedrooms = case_when(num_bedrooms == "1 Bedroom" ~ "1",
                                  num_bedrooms == "2 Bedrooms" ~ "2",
                                  num_bedrooms == "3 Bedrooms" ~ "3",
                                  num_bedrooms == "4 Bedrooms" ~ "4",
                                  num_bedrooms == "5 Bedrooms" ~ "5",
                                  num_bedrooms == "6 Bedrooms" ~ "6",
                                  num_bedrooms == "7 Bedrooms" ~ "7",
                                  num_bedrooms == "8 Bedrooms" ~ "8",
                                  num_bedrooms == "11 Bedrooms" ~ "11",
                                  num_bedrooms == "14 Bedrooms" ~ "14",
                                  TRUE ~ as.character(num_bedrooms)))

# Converting the column to a Numeric data type
PropertyDataset1a$num_bedrooms<- as.numeric(PropertyDataset1a$num_bedrooms)
#This warning from R alerts the user that it could not alter some of the  data values in the
#column, which results in NA appearing in the cells.
```

```
# Completed successfully. Compare the changes made
PropertyDataset1a$num_bedrooms[1:10]
```

```
##  [1] NA  3 NA NA  4 NA NA NA  2 NA
```

PropertyDataset1a$num_bathrooms

Now, we will do the same to **num_bathrooms**

```
# View some example values of the column
PropertyDataset1a$num_bathrooms[1:10]
#Display unique variables to replace
unique(PropertyDataset1a$num_bathrooms)

# Replacing the dataframe PropertyDataset1a num_bedrooms Character values with Numeric values
PropertyDataset1a <- PropertyDataset1a %>%
  mutate(num_bathrooms = case_when(num_bathrooms == "1 Bathroom" ~ "1",
                                   num_bathrooms == "2 Bathrooms" ~ "2",
                                   num_bathrooms == "3 Bathrooms" ~ "3",
                                   num_bathrooms == "4 Bathrooms" ~ "4",
                                   num_bathrooms == "5 Bathrooms" ~ "5",
                                   num_bathrooms == "6 Bathrooms" ~ "6",
                                   num_bathrooms == "7 Bathrooms" ~ "7",
                                   num_bathrooms == "8 Bathrooms" ~ "8",
                                   num_bathrooms == "15 Bathrooms" ~ "15",
                                   TRUE ~ as.character(num_bathrooms)))

# Converting the column to a Numeric data type
PropertyDataset1a$num_bathrooms<- as.numeric(PropertyDataset1a$num_bathrooms)
```

```
#Completed successfully. Compare the changes made
PropertyDataset1a$num_bedrooms[1:10]
```

```
##  [1] NA  3 NA NA  4 NA NA NA  2 NA
```

PropertyDataset1a$parking

Replace **parking** variables with consistent and meaningful data values

```
#Display unique variables to replace
unique(PropertyDataset1a$parking)
```

```
## [1] "Has" "No"  ""    "has"
```

```
# Replacing the dataframe PropertyDataset1a parking Character values with consistent values
PropertyDataset1a <- PropertyDataset1a %>%
  mutate(parking = case_when(parking == 'Has' ~ 'Yes',
                             parking == 'No' ~ 'No',
                             parking == 'has' ~ 'Yes',
                             !is.na(parking) ~ 'No',
                             TRUE ~ as.character(parking1))) %>%
  select(-parking1)  # Remove 'parking1' column
```

PropertyDataset1a$raw_address

Format **raw_address** to title_case using *stringr* package

```
PropertyDataset1a$raw_address <- str_to_upper(PropertyDataset1a$raw_address)
```

Review Updated Data Variables

Finally, ensure data type and variables are consistent and correct

```
unique(PropertyDataset1a$num_bedrooms)  # No character variables are observed
```

```
##  [1] NA  3  4  2  1  5  6 11  7 14  8
```

```
unique(PropertyDataset1a$num_bathrooms)  #No character variables are observed
```

```
##  [1] NA  2  3  1  4  5  7 15  8  6
```

```
unique(PropertyDataset1a$parking)  # Consistent variables created
```

```
## [1] "Yes" "No"
```

```
PropertyDataset1a$raw_address[1:5]  # Consistent sentence case
```

```
## [1] "APARTMENT NO. 7, LIBRARY CORNER, MANORHAMILTON"
## [2] "4 WOODLAND PARK, RUSH, DUBLIN"
## [3] "50 MARBLE CREST, KILKENNY, KILKENNY"
## [4] "18 MARINA COURT, ATHY, KILDARE"
## [5] "61 ROSEHILL, NEWPORT, CO TIPPERARY"
```

```
str(PropertyDataset1a[, c(4, 10, 11)])
```

```
## 'data.frame':    40000 obs. of  3 variables:
##  $ sale_date    : Date, format: "2020-05-29" "2019-12-17" ...
##  $ num_bedrooms : num  NA 3 NA NA 4 NA NA NA 2 NA ...
##  $ num_bathrooms: num  NA 2 NA NA 3 NA NA NA 1 NA ...
```

```
# sale_date, num_bedrooms and num_bathrooms all now have
# their updated data format
```

PropertyDataset2

PropertyDataset2a$Date_of_Sale_(dd/mm/yyyy)

```
# Rename Date_of_Sale_(dd/mm/yyyy) column for easier analysis
PropertyDataset2a<- PropertyDataset2
colnames(PropertyDataset2a)[colnames(PropertyDataset2a) == 'Date_of_Sale_(dd/mm/yyyy)'] <- 'Date_of_Sal
```

PropertyDataset2a$Address

Using *stringr* package, we convert tidy up the columns to title case and delete replace blanks cells with NA for further cleaning later

```
#Converting the Address column to Title case
PropertyDataset2a$Address = str_to_title(PropertyDataset2a$Address)
```

PropertyDataset2a$Price

```
#Rename Price column for easier readability
colnames(PropertyDataset2a)[5] <- 'Price'
#Converting Price column to numeric & removing unrecognised characters
PropertyDataset2a$Price <- as.numeric(gsub("[^0-9.]", "", PropertyDataset2a$Price))
```

PropertyDataset2a$Description of Property

```
# Modifying format to title case
PropertyDataset2a$Description_of_Property = str_to_title(PropertyDataset2a$Description_of_Property)

# Replacing misspelled descriptions
PropertyDataset2a$Description_of_Property[PropertyDataset2a$Description_of_Property == 'Teach/?Ras?N C?
```

PropertyDataset2a$County

```
#Changing the County column to title case
PropertyDataset2a$County = str_to_title(PropertyDataset2a$County)
unique(PropertyDataset2a$County)
```

```
##  [1] "Dublin"          "Laois"           "Meath"           "Kilkenny"
##  [5] "Limerick"        "Carlow"          "Cork"            "Clare"
##  [9] "Sligo"           "Cavan"           "Tipperary"       "Wicklow"
## [13] "Roscommon"       "Wexford"         "Mayo"            "Donegal"
## [17] "Longford"        "Galway"          "Offaly"          "Kildare"
## [21] "Waterford"       "Louth"           "Kerry"           "Westmeath"
## [25] "Monaghan"        "Leitrim"         "Gaillimh"        "County Cavan"
## [29] "Ros Comain"      "Corcaigh"        "Co. Carlow"      "Tipp"
## [33] "Co. Offaly"      "County Dublin"   "County Louth"    "County Westmeath"
```

```
#Replacing duplicate data with set inputs
PropertyDataset2a <- PropertyDataset2a %>%
  mutate(County = case_when(County == 'Co.  Carlow' ~ 'Carlow',
                            County == 'County Westmeath' ~ 'Westmeath',
                            County == 'Gaillimh' ~ 'Galway',
                            County == 'Tipp' ~ 'Tipperary',
                            County == 'Co. Offaly' ~ 'Offaly',
                            County == 'Ros Comain' ~ 'Roscommon',
                            County == 'County Dublin' ~ 'Dublin',
                            County == 'Corcaigh' ~ 'Cork',
                            County == 'County Louth' ~ 'Louth',
                            County == 'County Cavan' ~ 'Cavan',
                            TRUE ~ as.character(County)))
```

PropertyDataset2a$Not_Full_Market_Price

```
#Changing the Not_Full_Market_Price column to title case
PropertyDataset2a$Not_Full_Market_Price = str_to_title(PropertyDataset2a$Not_Full_Market_Price)
```

PropertyDataset2a$Vat_Exclusive

```
#Changing the Vat_Exclusive column to title case
PropertyDataset2a$VAT_Exclusive = str_to_title(PropertyDataset2a$VAT_Exclusive)
```

PropertyDataset2a$Property_Size_Description

```
#Changing the Property_Size_Description column to title case
PropertyDataset2a$Property_Size_Description = str_to_title(PropertyDataset2a$Property_Size_Description)

#Replacing data values with the most common language format
PropertyDataset2a$Property_Size_Description[PropertyDataset2a$Property_Size_Description == 'Níos Mó Ná

PropertyDataset2a$Property_Size_Description[PropertyDataset2a$Property_Size_Description == 'N?Os L? N? 
```

Null Values In a Dataset

Removing null values from a dataframe is also an important part of data cleaning, however it also can affect the remaining data in the dataset when too many null values are present.

```
PropertyDataset1b<- na.omit(PropertyDataset1a)
```

Due to this, I will not remove the null values in the datasets

Creating New Columns

PropertyDataset1a

```
#Creating a Month column for a more detailed analysis
PropertyDataset1a$Month <- month(PropertyDataset1a$sale_date)
```

##PropertyDataset2a

```
#Creating Three new columns for address to provide more exploratory details
PropertyDataset2a <- separate(PropertyDataset2a, Address, into = c("Street", "City", "Area"), sep = ","
```

```
## Warning: Expected 3 pieces. Missing pieces filled with `NA` in 77539 rows [4, 5, 6, 10,
## 11, 14, 16, 17, 19, 21, 22, 25, 26, 27, 29, 32, 33, 35, 37, 38, ...].
```

```
# The warnings tell that some data does not have Area values and have been replaced with NA instead
```

Identifying Outliers in the Dataset

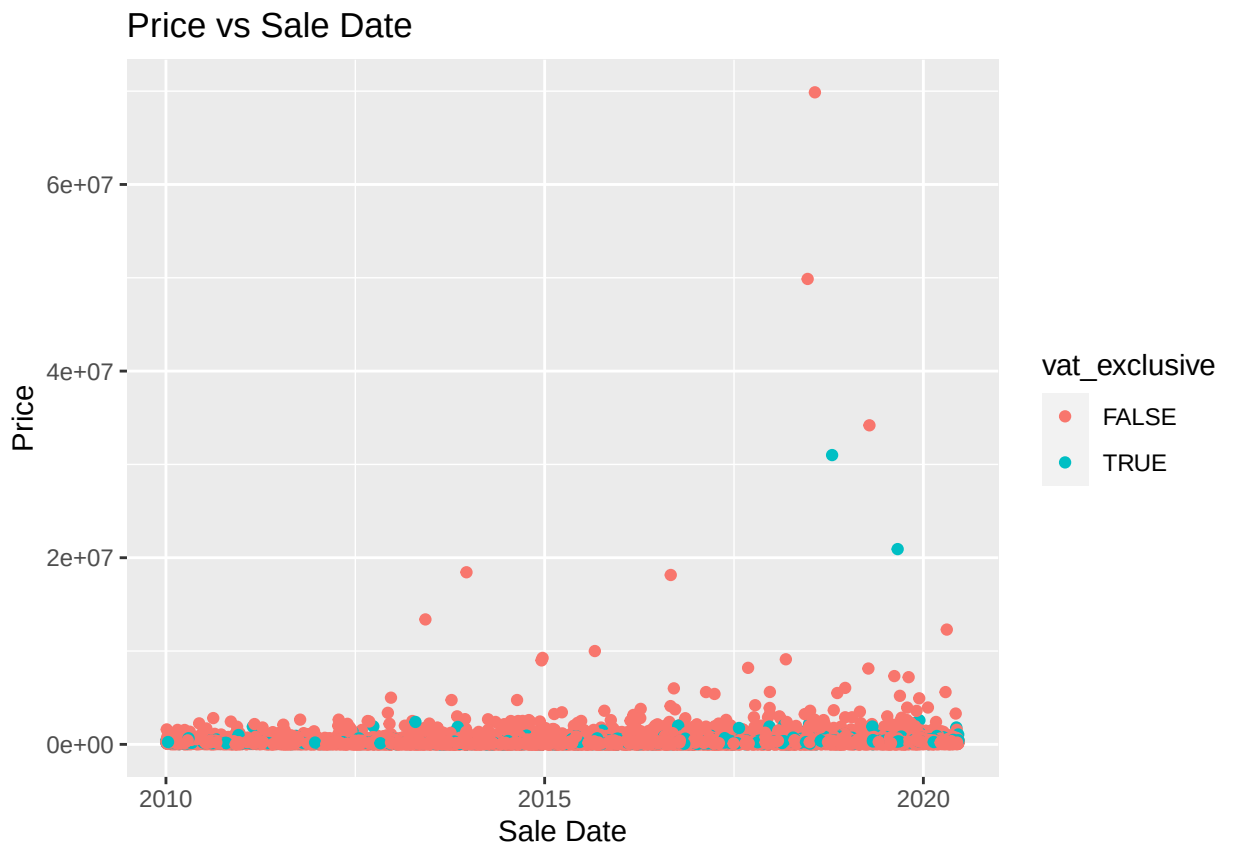
PropertyDataset1a

The data plotted below is quite bunched together and not easy to gather a readings from. Outliers are easily identifiable spaced out far from the average price range, whilst the majority of clusters sit in the bottom

```
mean(PropertyDataset1a$price)
```

```
## [1] 253913
```

```
ggplot(PropertyDataset1a, aes(sale_date, price)) +
  geom_point(aes(colour = vat_exclusive)) +
  labs(x = "Sale Date",
       y = "Price",
       title = "Price vs Sale Date")
```



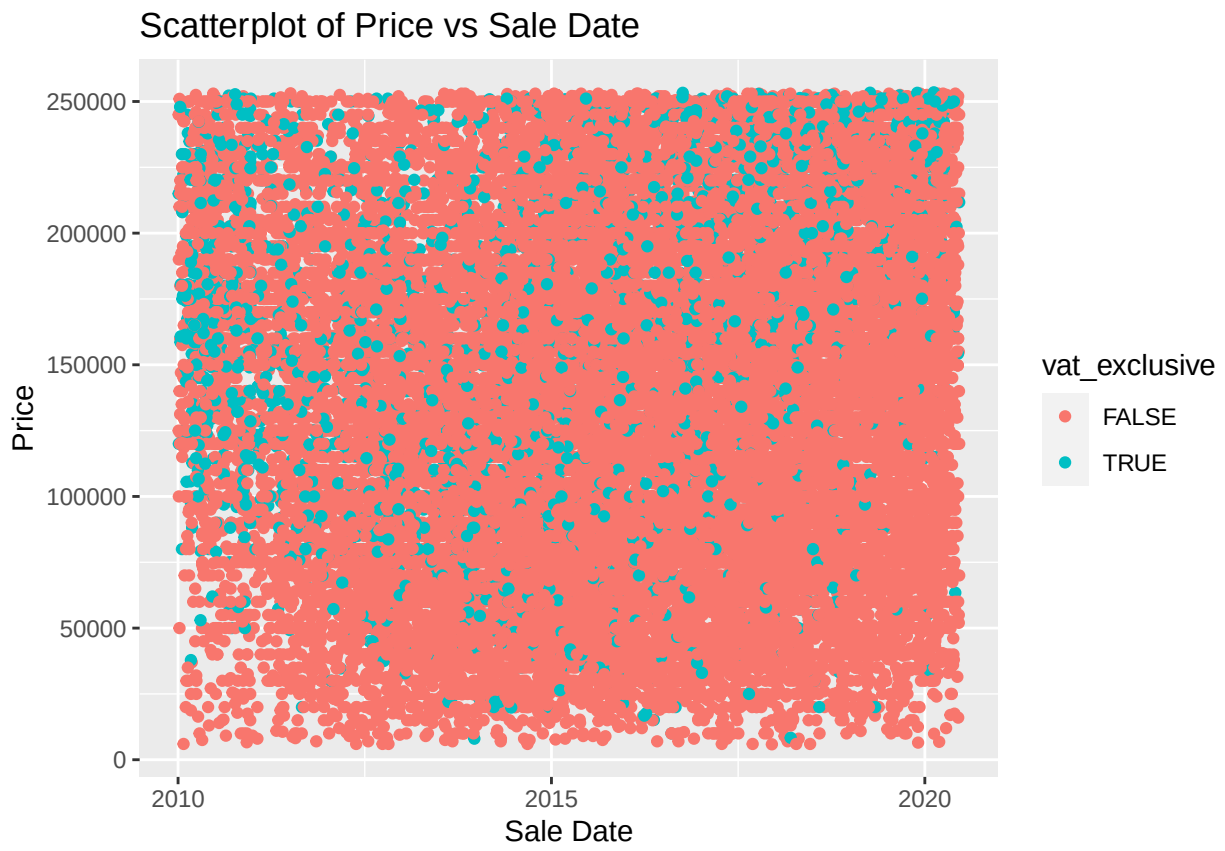
To form a more comprehensible and representative scatterplot, we will remove outliers in the data. Two outliers were removed, however there was a significant change in the visual aspect of the graph. The data is now more spread out. It is easier to identify individual plots on the graph, identify groups and possible trends.

The mean has also reduced to 244532.7 on average paid.

```
#Filter outliers into new dataframe

PropertyData1b <- PropertyDataset1a %>%
  filter(price <= mean(PropertyDataset1a$price))

# Mean to compare

mean(PropertyData1b$price)
```

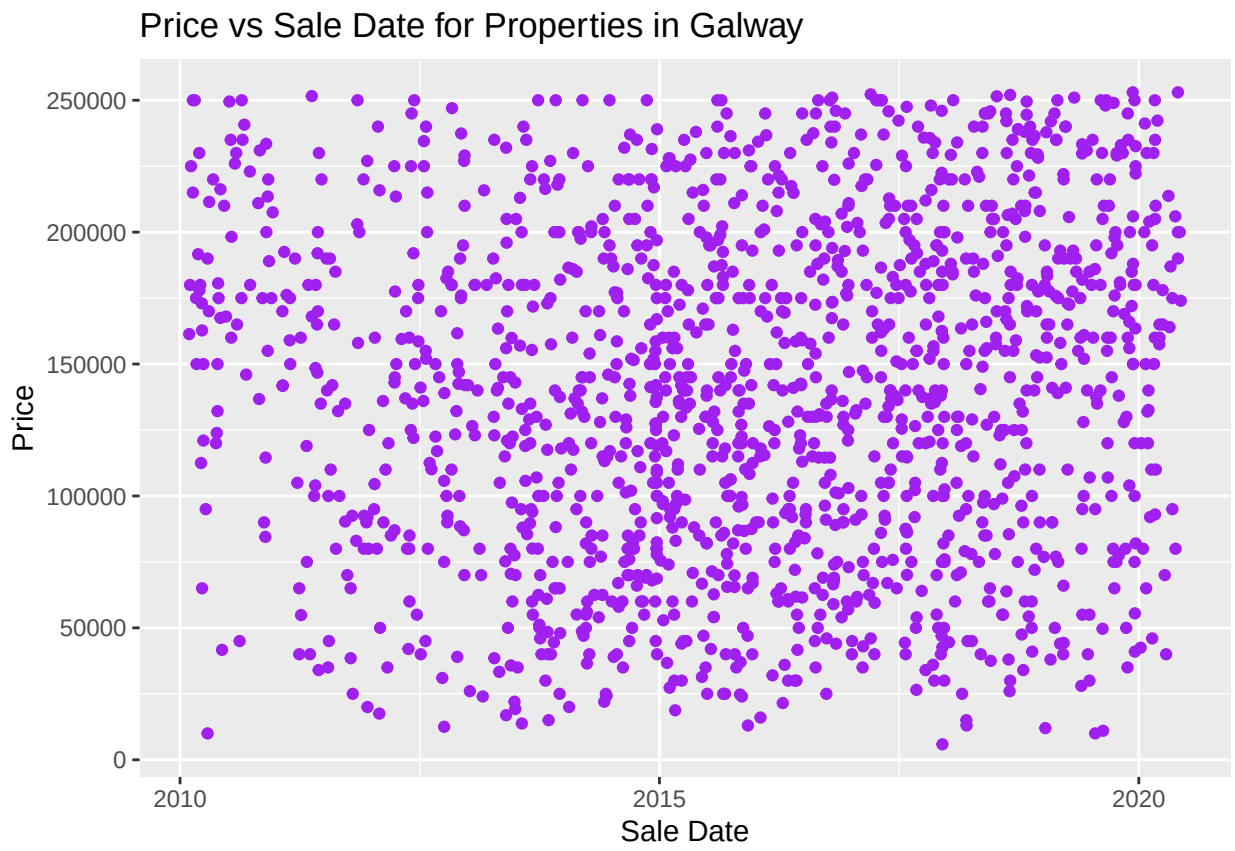
```
## [1] 139291.2
```

```
# Scatterplot without outliers

ggplot(PropertyData1b, aes(x = sale_date, y = price)) +
  geom_point(aes(colour = vat_exclusive)) +
  labs(title = "Scatterplot of Price vs Sale Date", x = "Sale Date", y = "Price")
```



The scatterplot graph is not easy to analyse regardless of filtering outliers. Due to the large amount of data in this dataset, it would need to be filtered further to be easily understandable. Here is an example.

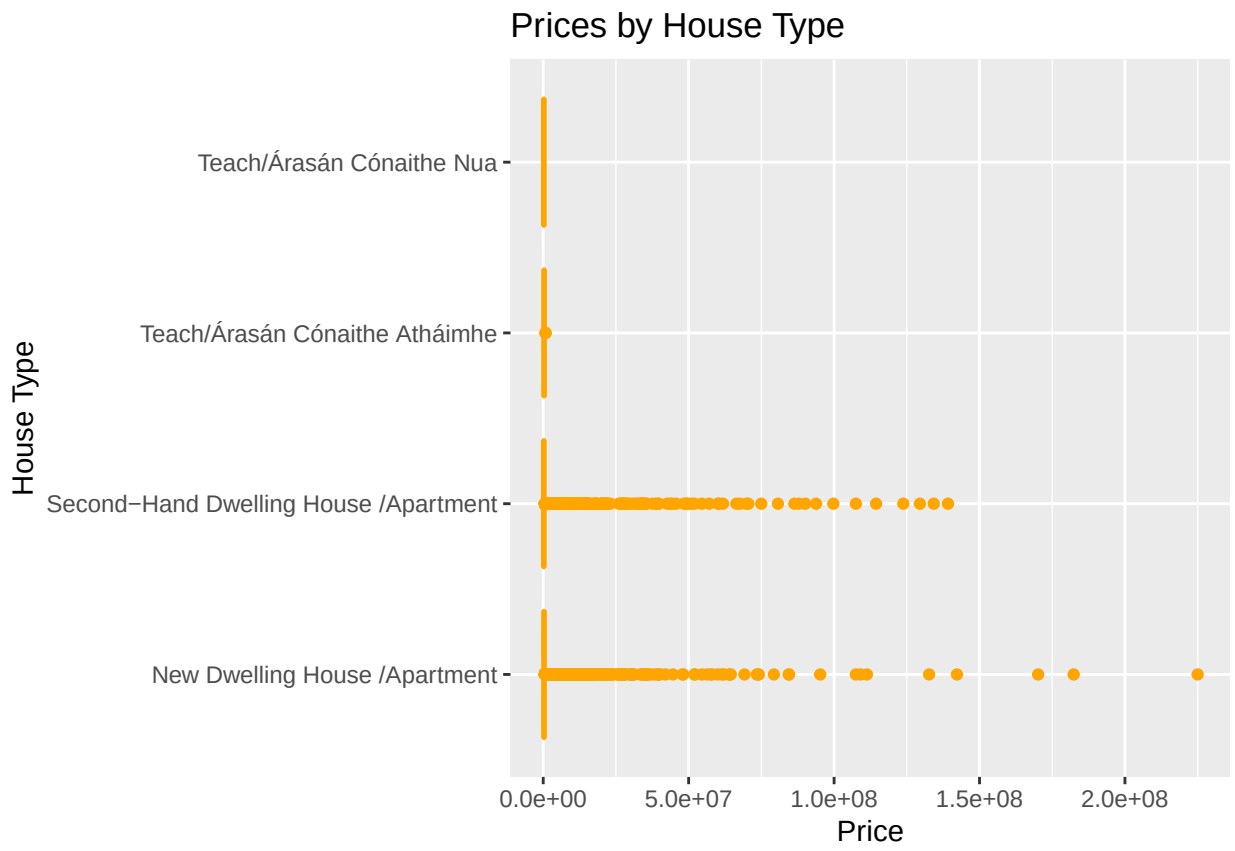
```
# Example of sales date and price for houses in county Galway
gal_prop_data <- PropertyData1b %>%
  filter(county == "Galway")


# Create a scatterplot
ggplot(gal_prop_data, aes(x = sale_date, y = price)) +
  geom_point(colour = "purple") +
  labs(title = "Price vs Sale Date for Properties in Galway",
       x = "Sale Date", y = "Price")
```



Clusters can be seen in 2015 at approximately the 100-150,000 price mark

PropertyDataset2a

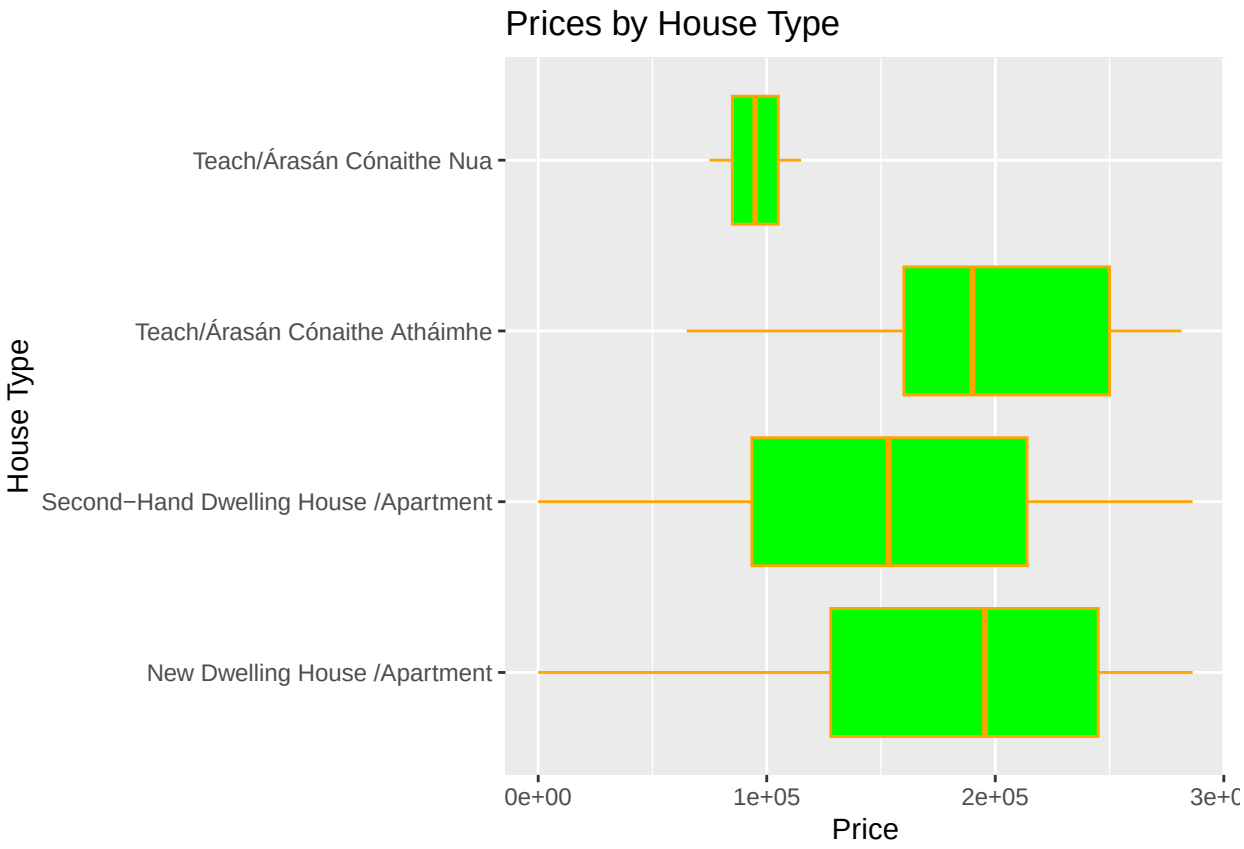
```
#Boxplot with all values included

ggplot(PropertyDataset2a, aes(x = Description_of_Property, y = Price)) +
  geom_boxplot(colour = "orange") +
  labs(title = "Prices by House Type", x = "House Type", y = "Price") +
  coord_flip()
```




```
mean(PropertyDataset2a$Price) #Mean price of a property in Dataset 2
```

```
## [1] 286376.5
```

```
#Filtering out outliers

PropertyDataset2b <- PropertyDataset2a %>%
  filter(Price <= mean(PropertyDataset2a$Price))


# Box plot for filtered PropertyDataset2b

ggplot(PropertyDataset2b, aes(x = Description_of_Property, y = Price)) +
  geom_boxplot(colour = "orange", fill = "green") +
  labs(title = "Prices by House Type", x = "House Type", y = "Price") +
  coord_flip()
```


Prices by House Type
Teach/Árasán Cónaithe Nua
Teach/Árasán Cónaithe Atháimhe
Second−Hand Dwelling House /Apartment
New Dwelling House /Apartment
House Type
0e+00
1e+05
2e+05
3e+0
Price

References

(ssayols 2017)

(Zach 2020)

(“IEEE Xplore Full-Text PDF:” n.d.)

[@!latexe2023]

(“Error While Reading Csv File in r - Stack Overflow,” n.d.)

(inscaven 2016)

file:///C:/Users/erink/Downloads/rmarkdown%20(2).pdf - RMarkdown Cheatsheet, RStudio

“Error While Reading Csv File in r - Stack Overflow.” n.d. https://stackoverflow.com/questions/18444769/error-while-reading-csv-file-in-r.

“IEEE Xplore Full-Text PDF:” n.d. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8768554.

inscaven. 2016. “Answer to ”Gsub r Extract Numeric from String”.” https://stackoverflow.com/a/36772009/22916988.

ssayols. 2017. “Answer to ”Line Breaks in r Markdown Text (Not Code Blocks)”.” https://stackoverflow.com/a/43113246/22916988.

Zach. 2020. “Create New Variables in r with Mutate() and Case_when().” https://www.statology.org/conditional-mutating-r/.