Data Programming in R Final Project

Hrishita Bapuram 22204557

2022-12-22

The data set used for the purpose of the project is sourced from the data repository of Central Statistics Office (CSO), Ireland (https://www.cso.ie) *. The data contains, for the year of 2021 in County Dublin, the monthly data of the volume of houses sold along with their total sale value and Mean and median price of houses along with information on the type of buyer (Household Buyer who can be first time owner occupiers or former owner occupiers or non occupiers and Non-Household buyers) as well the status of the dwelling (New or existing)*

Data Loading and Cleaning

```
library(tidyverse)
library(ggplot2)
library(patchwork) #makes arranging multiple plots easier
```

Loading the necessary libraries:

```
#loading the data using read.csv
df1 <- read.csv("housing2021.csv")

#show the structure of the data
str(df1)
```

Loading the data and observing its structure:

```
## 'data.frame':    384 obs. of  9 variables:
##  $ Statistic.Label : chr  "Volume of Sales" "Volume of Sales" "Volume of Sales" "Volume of Sales" ..
##  $ Month           : chr  "2021 January" "2021 January" "2021 January" "2021 January" ...
##  $ County          : chr  "Dublin" "Dublin" "Dublin" "Dublin" ...
##  $ Dwelling.Status : chr  "New" "New" "New" "New" ...
##  $ Stamp.Duty.Event: chr  "Executions" "Executions" "Executions" "Executions" ...
##  $ Type.of.Buyer   : chr  "Household Buyer - First-Time Buyer Owner-Occupier" "Household Buyer - For
##  $ Type.of.Sale    : chr  "Market" "Market" "Market" "Market" ...
##  $ UNIT            : chr  "Number" "Number" "Number" "Number" ...
##  $ VALUE           : num  59 39 7 147 216 389 54 237 65 72 ...
```

We see that there are several redundant variables since they all have the same value for example,

1. County (which has all values labeled Dublin),
2. Stamp.Duty.Event (all the recorded sales in this data set are ones with executed registrations and not filed)
3. Type.of.Sale (all sales are Market sales and not off-market sales)

Getting rid of the unnecessary variables

```
#selecting variables of interest using select() function in dplyr
df2 <- df1 %>% select(Statistic.Label, Month, Dwelling.Status, Type.of.Buyer, VALUE)
```

We see that the variable Month has two parts - the year (2021 for all observation hence, redundant) and the month name. We need only the month name.

```
#separating month and year from month column and getting rid of year since redundant information
df3 <- df2 %>% separate(Month, c("Year", "Month"), sep = " ") %>% select(-Year)
```

The three statistics or numeric measures are listed under one column with their respective values in the VALUE column. We have to separate this column into three individual columns.

```
#converting the 3 statistics as separate columns using spread()
df4 <- df3 %>% spread(Statistic.Label, VALUE)

#checking how the data looks through the first 6 observations
head(df4)
```

```
##   Month Dwelling.Status                                    Type.of.Buyer
## 1 April        Existing Household Buyer - First-Time Buyer Owner-Occupier
## 2 April        Existing           Household Buyer - Former Owner-Occupier
## 3 April        Existing                    Household Buyer - Non-Occupier
## 4 April        Existing                                Non-Household Buyer
## 5 April             New Household Buyer - First-Time Buyer Owner-Occupier
## 6 April             New           Household Buyer - Former Owner-Occupier
##   Mean Sale Price Median Price Value of Sales Volume of Sales
## 1          390184       355000           94.8             243
## 2          542274       422500          235.3             434
## 3          410656       341250           39.4              96
## 4          318649       250000           73.9             232
## 5          477730       427528           31.5              66
## 6          733453       566365           44.7              61
```

Converting the Categorical Variables into factors with their respective variables.

```
#converting categorical variable Month to ordered factor
df4$Month <- factor(df4$Month, levels = c("January","February","March","April",
                                          "May","June","July","August", "September",
                                          "October", "November", "December"),
                    ordered = TRUE,
                    labels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
                               "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"))
```

```
#converting categorical variable Dwelling Status to factor with 2 levels
df4$Dwelling.Status <- factor(df4$Dwelling.Status, levels = c("Existing", "New"))
```

```
#converting categorical variable Type of Buyer to factor with 4 levels
df4$Type.of.Buyer <- factor(df4$Type.of.Buyer,
                            levels = c("Household Buyer - First-Time Buyer Owner-Occupier",
                                       "Household Buyer - Former Owner-Occupier",
                                       "Household Buyer - Non-Occupier",
                                       "Non-Household Buyer"),
                            labels = c("HB_FOwnOcc", "HB_OwnOcc",
                                       "HB_NonOcc", "NHB"))
```

```
#exploring the structure of the transformed dataset
str(df4)
```

```
## 'data.frame':    96 obs. of  7 variables:
##  $ Month          : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..: 4 4 4 4 4 4 4 4 8 8 ...
##  $ Dwelling.Status: Factor w/ 2 levels "Existing","New": 1 1 1 1 2 2 2 2 1 1 ...
##  $ Type.of.Buyer  : Factor w/ 4 levels "HB_FOwnOcc","HB_OwnOcc",..: 1 2 3 4 1 2 3 4 1 2 ...
##  $ Mean Sale Price: num  390184 542274 410656 318649 477730 ...
##  $ Median Price   : num  355000 422500 341250 250000 427528 ...
##  $ Value of Sales : num  94.8 235.3 39.4 73.9 31.5 ...
##  $ Volume of Sales: num  243 434 96 232 66 61 10 143 348 534 ...
```

Renaming the columns to shorter names for ease of reference while coding:

```
new_colnames <- c("month", "dwell_status", "buyer_type",
                  "mean_price", "median_price", "sale_value", "sale_volume")
#assigning new column names
colnames(df4) <- new_colnames
str(df4)
```

```
## 'data.frame':    96 obs. of  7 variables:
##  $ month       : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..: 4 4 4 4 4 4 4 4 8 8 ...
##  $ dwell_status: Factor w/ 2 levels "Existing","New": 1 1 1 1 2 2 2 2 1 1 ...
##  $ buyer_type  : Factor w/ 4 levels "HB_FOwnOcc","HB_OwnOcc",..: 1 2 3 4 1 2 3 4 1 2 ...
##  $ mean_price  : num  390184 542274 410656 318649 477730 ...
##  $ median_price: num  355000 422500 341250 250000 427528 ...
##  $ sale_value  : num  94.8 235.3 39.4 73.9 31.5 ...
##  $ sale_volume : num  243 434 96 232 66 61 10 143 348 534 ...
```

There are 3 price related columns in the data set - Mean Price, Median Price and Sale Value. The first two are in Euros and the last one is in Millions of Euros. Converting all of them to millions of euros will creating uniformity in unit and scale.

```
#converting mean_price and median_price to millions of euros

df4$mean_price = df4$mean_price/1000000
df4$median_price = df4$median_price/1000000
```

```
str(df4) #final structure of the cleaned dataset
```

Exploring the final structure of the data set.

```
## 'data.frame':    96 obs. of  7 variables:
##  $ month       : Ord.factor w/ 12 levels "Jan"<"Feb"<"Mar"<..: 4 4 4 4 4 4 4 4 8 8 ...
##  $ dwell_status: Factor w/ 2 levels "Existing","New": 1 1 1 1 2 2 2 2 1 1 ...
##  $ buyer_type  : Factor w/ 4 levels "HB_FOwnOcc","HB_OwnOcc",..: 1 2 3 4 1 2 3 4 1 2 ...
##  $ mean_price  : num  0.39 0.542 0.411 0.319 0.478 ...
##  $ median_price: num  0.355 0.422 0.341 0.25 0.428 ...
##  $ sale_value  : num  94.8 235.3 39.4 73.9 31.5 ...
##  $ sale_volume : num  243 434 96 232 66 61 10 143 348 534 ...
```

Following are the variables in the final dataset:

1. `month` - This is an ordered factor with labels denoting the month of the sale of houses.
2. `dwell_status` - This is a categorical variable separating the data on the basis of whether the houses sold were newly constructed (never lived-in before), i.e., `New` or if the houses sold were existing structures, i.e., `Existing`
3. `buyer_type` - This is also a categorical variable that differentiates the sales transactions on the basis of the type of buyer for each house. The four categories for this variable are as follows:
    - `HB_FOwnOcc` : Household Buyers that are First Time Owner-Occupiers. These are people who buying a home for the first time with the intention of owning and living on the property.
    - `HB_OwnOcc` : Household Buyers that are Former Owner-Occupiers. This is the same as the previous category but the buyers in this category have owned homes before the current purchase.
    - `HB_NonOcc` : These are household buyers that are Non-Occupiers, i.e., they are not intending to live in nor lived in the home purchased.
    - `NHB`: Non-Household Buyers. This category of buyers consists of non-household or commercial entities that have purchased the home either with the intention of investing for returns or as a place to conduct business.
4. `mean_price` - This is the average value of the houses sold (in Millions of Euros)
5. `median_price` - This is the middle value or the central price value of the houses sold (in Millions of Euros). This value indicates that 50% of the houses sold and bought are below this threshold.
6. `sale_volume` - This denotes the Quantity or amount of houses sold under each categorical combination discussed (i.e., for a particular month, for a particular dwell_status and for a particular buyer_type)
7. `sale_value` - In millions of Euros, this denotes the total sale value of all the houses sold for a particular configuration of the categorical variables.

Now, we move on to the next step of the analysis process, which is Exploratory Data Analysis to understand the individual behavior of each variable and it's interaction with other variables.

Analysis

```
#printing the numerical summary statistics of all the variables
summary(df4)
```

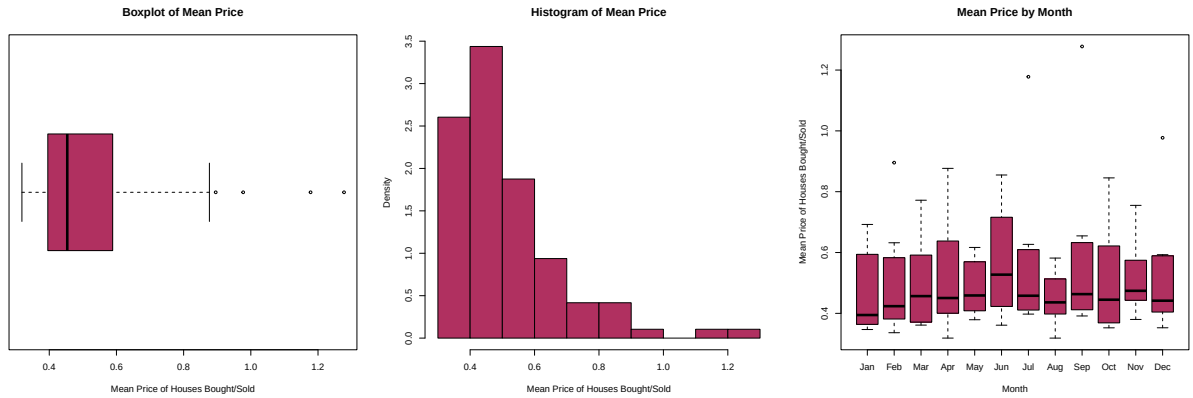
Numerical Summary of all variables:

```
##      month      dwell_status      buyer_type   mean_price      median_price
##  Jan    : 8   Existing:48    HB_FOwnOcc:24   Min.   :0.3186   Min.   :0.2500
##  Feb    : 8   New     :48    HB_OwnOcc :24   1st Qu.:0.3966   1st Qu.:0.3433
##  Mar    : 8                  HB_NonOcc :24   Median :0.4536   Median :0.4050
##  Apr    : 8                  NHB       :24   Mean   :0.5157   Mean   :0.4310
##  May    : 8                                  3rd Qu.:0.5871   3rd Qu.:0.4672
##  Jun    : 8                                  Max.   :1.2775   Max.   :1.2775
##  (Other):48
##    sale_value      sale_volume
##  Min.   :  1.90   Min.   :  2.0
##  1st Qu.: 32.33   1st Qu.: 64.0
##  Median : 53.60   Median :114.5
##  Mean   : 90.15   Mean   :188.0
##  3rd Qu.: 99.83   3rd Qu.:255.0
##  Max.   :397.30   Max.   :670.0
##
```

- **In terms of the categorical variables, we see that there are an equitable distribution of observations for the three variables `month`, `dwell_status`, `buyer_type`.**
- **Mean Price is distributed between 0.32 Million Euros and 1.28 Million Euros. That is, the average price of all the houses sold/bought in 2021 range between 320,000 EUR and 1,280,000 EUR. 50% of the values lie below 500,000 EUR and the upper 25% of the houses are priced between 0.58 Million Euros and 1.23 Million Euros. There seems to be high variation in mean prices as they increase.**
- **This can be attributed to the fact that means are sensitive to outliers and are therefore more prone to instability (or high variation) in the presence of even a few large numbers (in our case the presence of maybe a few prices bought for a very high price that skewed the mean).**
- **Studying the behavior of the median price especially for this period will help us understand if this drastic shifts in mean prices are due to the emergence of a pattern of a lot of high priced houses or just a few skewing the mean. Median prices of houses bought/sold in 2021 are distributed between 0.25 and 1.28 Million Euros.**
- **The total sale value of houses for a particular category, month and type of buyer range between 1.9 Million Euros and 397.3 Million Euros**
- **While 50% of the houses bought had a total value below 53.6 Million Euros and the upper 50% have very high variance. Since the mean sale value is 90.15 and much higher than the median sale value of 53.6 million euros, we can also confirm that there is right skewness in the data, potentially due to the sale of a small number of very large valued houses.Sale Volume displays the same pattern of skewness as Sale value.**

Understanding structure of each variable MEAN PRICE

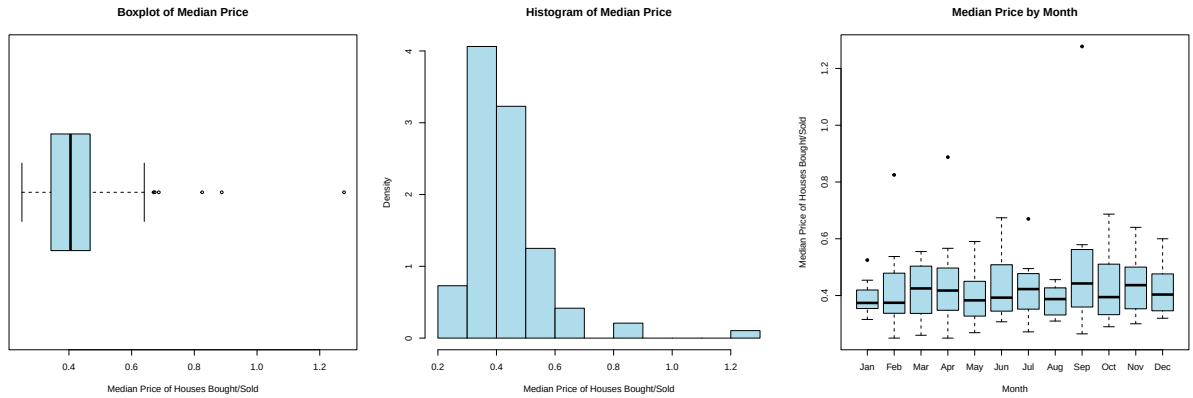
```
par(mfrow = c(1,3))
boxplot(df4$mean_price,
        horizontal = TRUE, col = "maroon",
        main = "Boxplot of Mean Price",
        xlab = "Mean Price of Houses Bought/Sold")
hist(df4$mean_price, freq = FALSE ,col = "maroon",
     main = "Histogram of Mean Price",
     xlab = "Mean Price of Houses Bought/Sold",
     ylab = "Density")
plot(df4$month, df4$mean_price, col = "maroon",
     main = "Mean Price by Month",
     xlab = "Month",
     ylab = "Mean Price of Houses Bought/Sold")
```



As discussed through the numerical summary, we see that there is high variance in the upper 50% of the data, attributed to the few number of highly priced houses that skewed the mean price of houses. We see that although the variability changes month-on-month, the mean price of houses sold do not change between months, except for a small increase in month of June.

MEDIAN PRICE

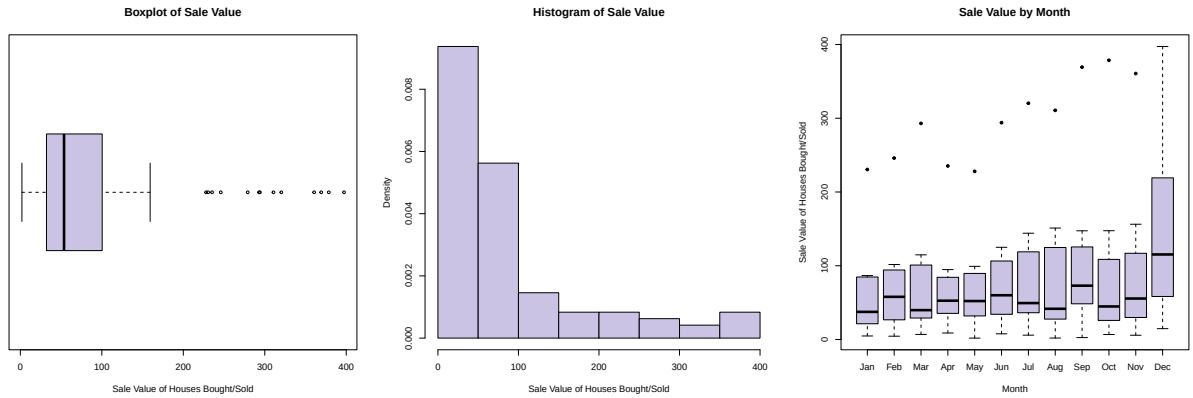
```
par(mfrow = c(1,3))
boxplot(df4$median_price,
        horizontal = TRUE, col = "#AFDCEB",
        main = "Boxplot of Median Price",
        xlab = "Median Price of Houses Bought/Sold")
hist(df4$median_price, freq = FALSE ,col = "#AFDCEB",
     main = "Histogram of Median Price",
     xlab = "Median Price of Houses Bought/Sold",
     ylab = "Density")
plot(df4$month, df4$median_price, col = "#AFDCEB",
     main = "Median Price by Month",
     xlab = "Month",
     ylab = "Median Price of Houses Bought/Sold", pch = 19)
```



As discussed through the numerical summary, we see that there are a few outliers that cause the skewness. The values for median price of house sales do not appear to differ by months significantly. The pattern is very similar to that of mean price of houses. It will be useful to study the intercation between mean house prices and median house prices.

SALE VALUE

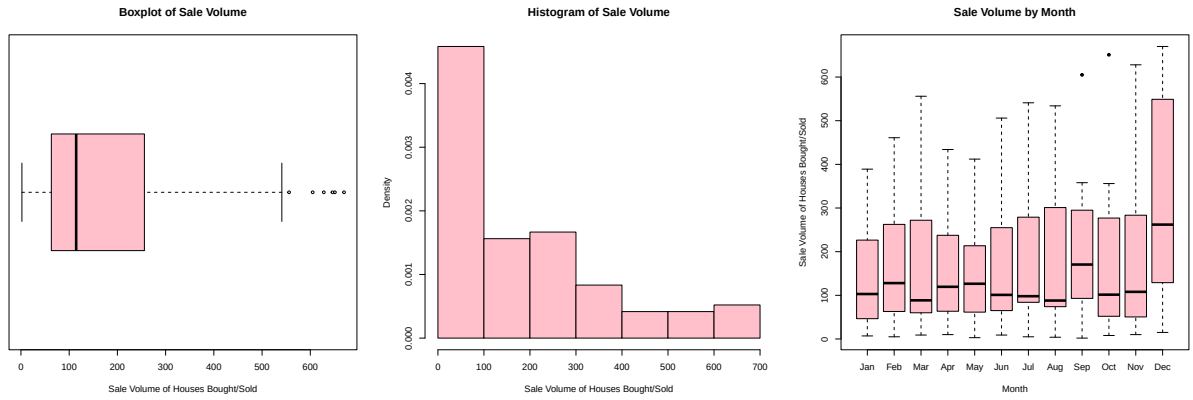
```
par(mfrow = c(1,3))
boxplot(df4$sale_value,
        horizontal = TRUE, col = "#CBC3E3",
        main = "Boxplot of Sale Value",
        xlab = "Sale Value of Houses Bought/Sold")
hist(df4$sale_value, freq = FALSE ,col = "#CBC3E3",
     main = "Histogram of Sale Value",
     xlab = "Sale Value of Houses Bought/Sold",
     ylab = "Density")
plot(df4$month, df4$sale_value, col = "#CBC3E3",
     main = "Sale Value by Month",
     xlab = "Month",
     ylab = "Sale Value of Houses Bought/Sold", pch = 19)
```



We can confirm the findings about sales value from the numerical summaries through the plots above. There is skewness to the right and a large number of outliers. Additionally, we see a spike in Sale value in the month of December where the median sales volume is higher than most of the months' 3rd Quantile benchmark.

SALE VOLUME

```
par(mfrow = c(1,3))
boxplot(df4$sale_volume,
        horizontal = TRUE, col = "pink",
        main = "Boxplot of Sale Volume",
        xlab = "Sale Volume of Houses Bought/Sold")
hist(df4$sale_volume, freq = FALSE ,col = "pink",
     main = "Histogram of Sale Volume",
     xlab = "Sale Volume of Houses Bought/Sold",
     ylab = "Density")
plot(df4$month, df4$sale_volume, col = "pink",
     main = "Sale Volume by Month",
     xlab = "Month",
     ylab = "Sale Volume of Houses Bought/Sold", pch = 19)
```

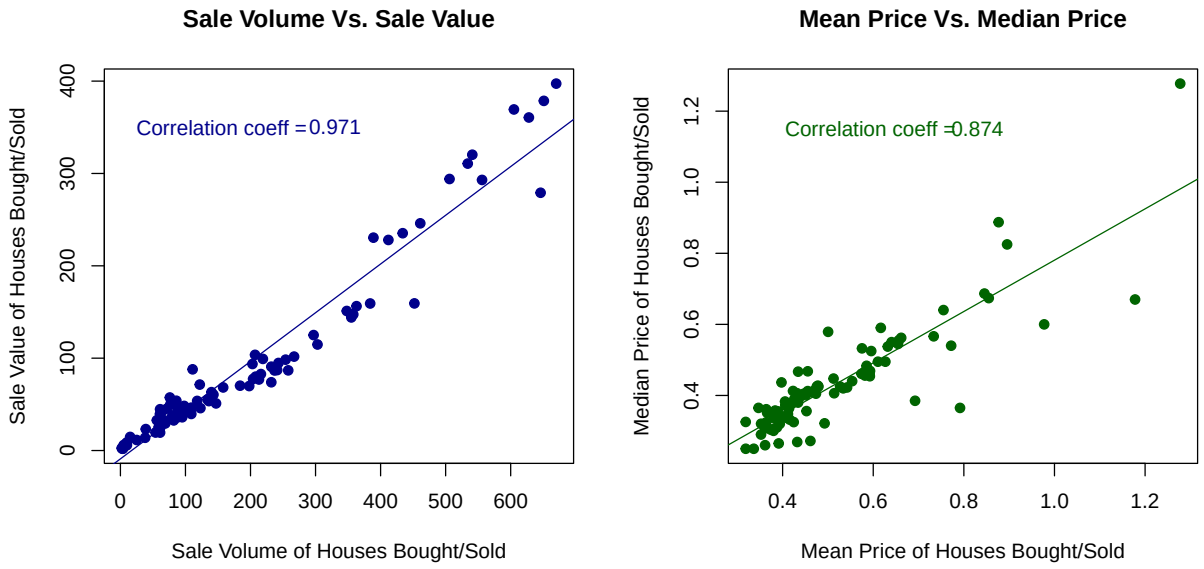


The data for sales volume corresponds to that of the findings from the sales value analysis. The month of December in 2021 appears to have recorded the largest number of house sales which can explain the higher sales value as well. There is a high degree of positive correlation between the two variables sales_value and sales_volume.

Since the variables Mean_price and Median price and the variables Sale Value and Sale Volume appear to be correlated to each other, it will be useful to understand the extend of the relationship between the variables.

Plotting scatter plots along with the lines of best fit for each pair of variables:

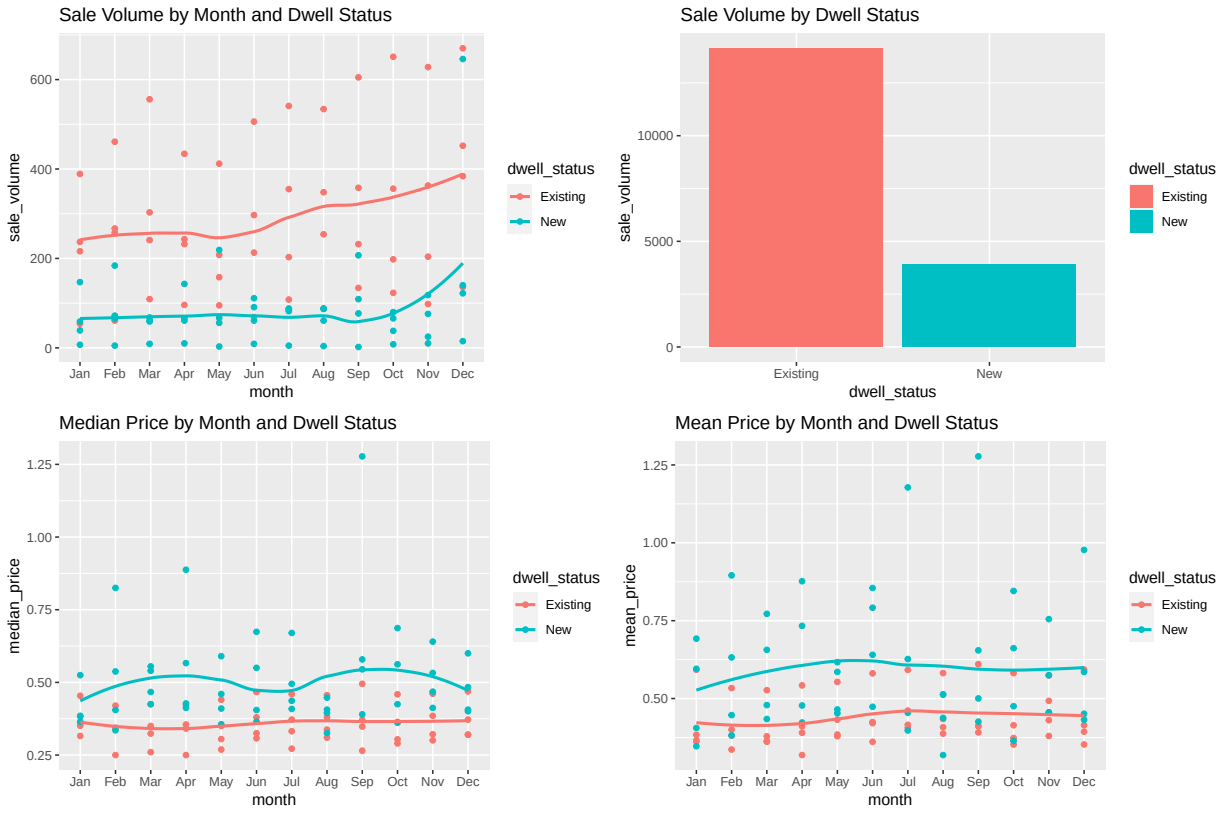
```
par(mfrow = c(1,2))
plot(df4$sale_volume, df4$sale_value, main = "Sale Volume Vs. Sale Value",
     xlab = "Sale Volume of Houses Bought/Sold",
     ylab = "Sale Value of Houses Bought/Sold", col = "darkblue", pch = 19)
abline(lm(df4$sale_value ~ df4$sale_volume), col = "darkblue")
text(x=160,y=350,"Correlation coeff = ",col = "darkblue")
text(x=330,y=350,round(cor(df4$sale_value,df4$sale_volume),3),col = "darkblue")
plot(df4$mean_price, df4$median_price, main = "Mean Price Vs. Median Price",
     xlab = "Mean Price of Houses Bought/Sold",
     ylab = "Median Price of Houses Bought/Sold", col = "darkgreen", pch = 19)
abline(lm(df4$median_price ~ df4$mean_price), col = "darkgreen")
text(x=0.6,y=1.15,"Correlation coeff = ", col = "darkgreen")
text(x=0.83,y=1.15,round(cor(df4$mean_price,df4$median_price),3),col="darkgreen")
```



- **With Sale Value and Sale Volume, we see that two variables share a very strong positive correlation (Correlation Coefficient is 0.971) and we see that this pattern carries on even onto the higher values of Sales Volume and Sales Value.**
- **With Mean Price and Median Price, we see that the two variables share a strong degree of positive correlation only at the lower values of Mean and Median House Prices. For higher values of Mean Price, the Median Price doesn't see to increase just that much.**
- **This can be attributed to the fact that the measure of mean is sensitive to outliers whereas median is not. Therefore, when the value of mean price is higher, that might not necessarily mean high median price since the high mean can be due to a few high Mean Prices of houses.**

```
par(mfrow = c(2,2))
#Plot of sale volume by month and dwell status
pA1 <- ggplot(df4) +
  geom_point(aes(month, sale_volume, color = dwell_status)) +
  geom_smooth(aes(month, sale_volume, group = dwell_status, color = dwell_status), se = FALSE)+
  labs(title = "Sale Volume by Month and Dwell Status")

#Plot of Median Price by month and dwell status
pA2 <- ggplot(df4) +
  geom_point(aes(month, median_price, color = dwell_status)) +
  geom_smooth(aes(month, median_price, group = dwell_status, color = dwell_status), se = FALSE)+
  labs(title = "Median Price by Month and Dwell Status")

#Plot of Mean Price by month and dwell status
pA3 <- ggplot(df4) +
  geom_point(aes(month, mean_price, color = dwell_status)) +
  geom_smooth(aes(month, mean_price, group = dwell_status, color = dwell_status), se = FALSE)+
  labs(title = "Mean Price by Month and Dwell Status")

#column chart of Dwell Status by sale volume
pA4 <- ggplot(df4) +
  geom_col(aes(dwell_status, sale_volume, fill = dwell_status))+
  labs(title = "Sale Volume by Dwell Status")

(pA1 + pA4) / (pA2 + pA3) #patchwork package enables arranging plots easily
```

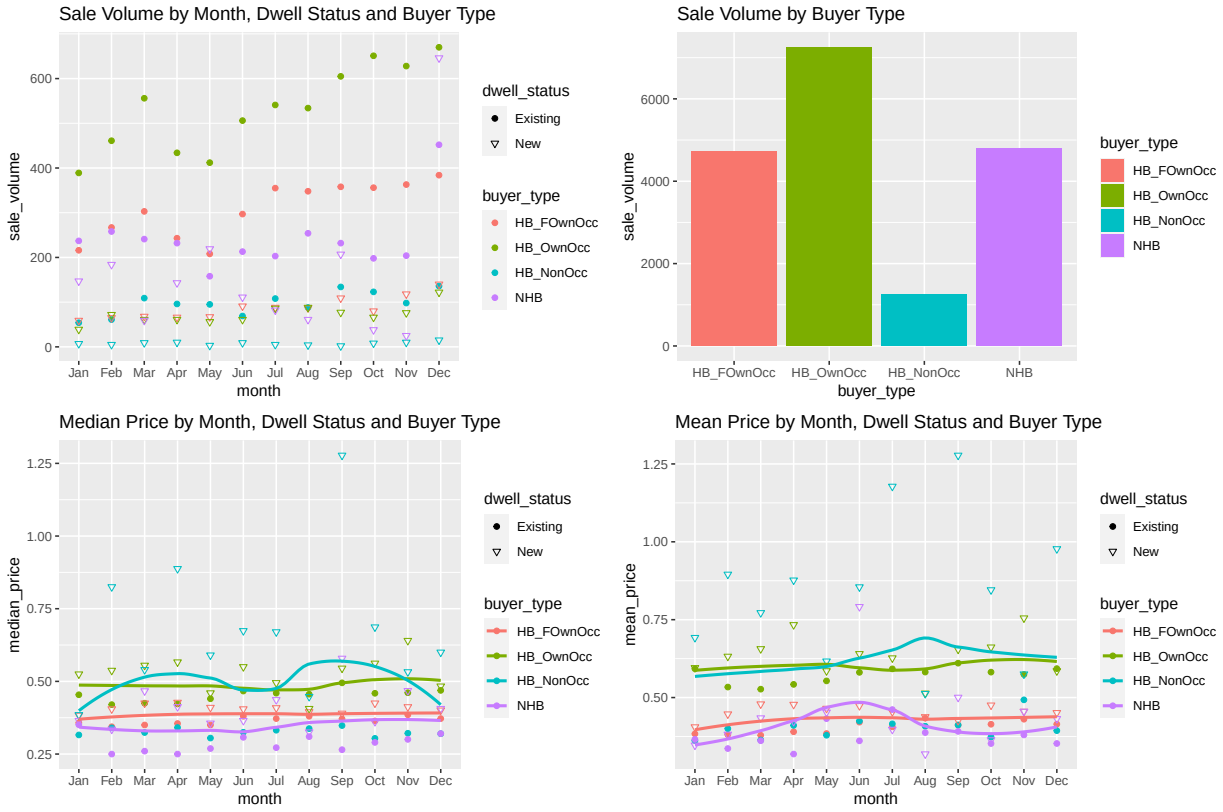
Understanding Interactions



- **We see that when it comes to the categorical variable `dwell_status`, We see that there isn't much difference in terms of months except for the surge towards the last few months starting August for existing houses and September for New houses when sales were increasing. On the whole, Sales Volume was higher for existing houses than for New houses, that is, most of the houses sold were existing.**
- **For the entire year, we see that volume of sales of existing houses was nearly 3 times that of new houses.**
- **In terms of pricing, with the excepting of small differences between the months of June and August, pricing has remaining consistent for all the remaining months of the year. When it comes to the type of houses sold, New houses were sold for a higher price throughout the year.**

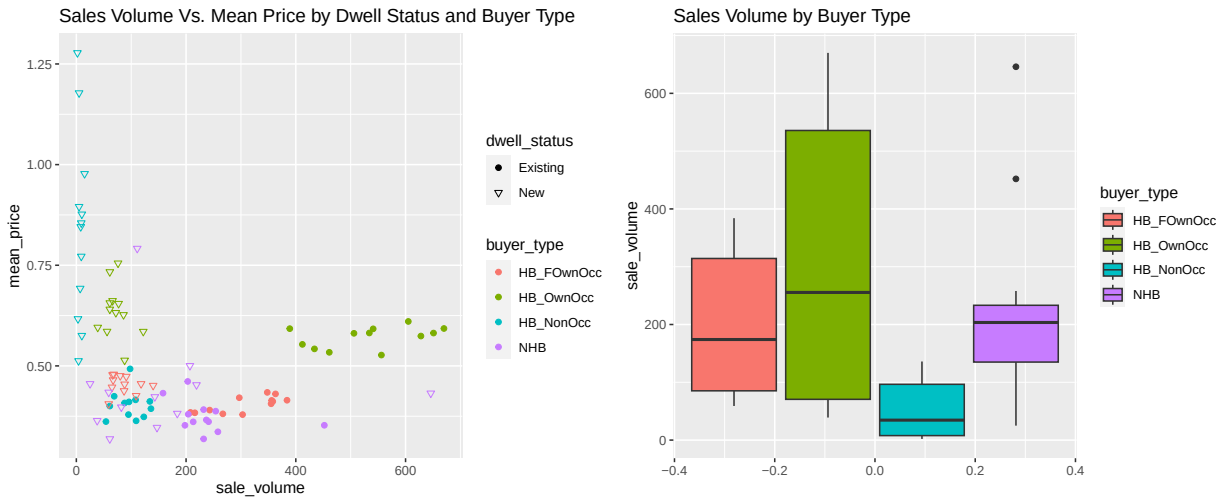
```
#Plot of sale volume by month, buyer type and dwell status
pB1 <- ggplot(df4) +
  geom_point(aes(month, sale_volume, color = buyer_type, shape = dwell_status)) +
  scale_shape_manual(values = c(19,25)) +
  labs(title = "Sale Volume by Month, Dwell Status and Buyer Type")

#plot of median price by month and buyer type
pB2 <- ggplot(df4) +
  geom_point(aes(month, median_price, color = buyer_type, shape = dwell_status)) +
  scale_shape_manual(values = c(19,25)) +
  geom_smooth(aes(month, median_price, group = buyer_type, color = buyer_type), se = FALSE)+
  labs(title = "Median Price by Month, Dwell Status and Buyer Type")
```

```
#Plot of Mean Price by Month, Dwell Status and Buyer Type
pB3 <- ggplot(df4) +
  geom_point(aes(month, mean_price, color = buyer_type, shape = dwell_status)) +
  scale_shape_manual(values = c(19,25)) +
  geom_smooth(aes(month, mean_price, group = buyer_type, color = buyer_type), se = FALSE)+
  labs(title = "Mean Price by Month, Dwell Status and Buyer Type")

#column chart of Buyer type by sale volume
pB4 <- ggplot(df4) +
  geom_col(aes(buyer_type, sale_volume, fill = buyer_type))+
  labs(title = "Sale Volume by Buyer Type")

(pB1 + pB4) / (pB2 + pB3) #arranging using patchwork library
```



- **We know from the previous set of plots that the sale volume was higher for existing houses, in comparison to New houses as indicated by the first scatter plot in this grid. Secondly, when it comes to Household Buyers who are former Owner-Occupiers, the sale volume is particularly high for Existing houses and lower for New houses. The interaction between New houses and HB_OwnOcc seems to be particularly significant.**
- **On the whole, sales volume was the highest for Household Buyers who are Former Owner-Occupiers and the lowest for Household Buyers that are Non-Occupiers.**
- **The median and mean prices are higher for Non-Household buyers and Household Buyers that are former Owner Occupiers, and lower for the other two categories. The median price was the lowest and consistent throughout the year for Non-Household Buyers.**
- **In terms of mean price, the value was higher in June for Non-Household Buyers owing to the the mean price of a New house of approximately 0.76 Million Euros.**

```r
#Boxplot of sales Volume by Buyer type
pC2 <- ggplot(df4) +
  geom_boxplot(aes(y = sale_volume, fill = buyer_type))+
  labs(title = "Sales Volume by Buyer Type")

pC1 <- ggplot(df4) +
  geom_point(aes(sale_volume, mean_price, color = buyer_type, shape = dwell_status)) +
  scale_shape_manual(values = c(19,25))+
  labs(title = "Sales Volume Vs. Mean Price by Dwell Status and Buyer Type")

pC1|pC2 #using patchwork library
```



- **The largest sale volume is observed for Household Buyers that are Former Owner Occupiers. We can explain the large variability because of the interaction between dwell_status and this particular buyer_type (Fewer New houses were sold to HB_OwnOcc than Existing houses).**
- **Although the smallest sales volume is for Household Buyers who are Non-Occupiers we see that the variability is particularly high for New houses sold to this category.**
- **We also see a distinct cluster being formed for the two types of houses (New and Existing) for the First time Household Buyers who are Owner-Occupiers**
- **Lastly while the clusters for Non-Household Buyers investing in New houses is more variable, the cluster for buyers of existing houses in the same buyer category is more tightly packed and therefore their overall variability is much lower.**

R Package - `DataExplorer`

- **DataExplorer is an R package created by Boxuan Cui (source: https://github.com/boxuancui/DataExplorer) and is aimed at automating the EDA or Exploratory Data Analysis process. Most of the functions in this package, aim at automating the multi-step process of making relevant plots, identifying and visualizing missing values, densities of variables to name a few.**
- **Additionally, the package also generates a pre-formatted report with just one function, very useful to novice data analysts and even experienced ones to get an entire report will all basic EDA with just one command.**
- **Apart from EDA and customisable visualization, the package also comes with a few feature engineering functions including PCA (Principal Component Analysis) to modify and and format variables/features to better suit the next part of analysis/model fitting.**

```
library(DataExplorer) #loading the package
```

The first important function is the use of the `create_report` function which will generate a full length report on HTML. For the sake of brevity, I will use other functions in the package that are a part of the report (in create_report). With `create_report`, the function also allows the user to pick a response variable and then generates a specific report with the given response variable. For further customization of the report, the `configure_report` function allows the user to pick the specific plots and graphs that go in the report as well as the theme and format of the same.

```
#commenting the code for `create_report`

#configure_report(add_introduce = TRUE, add_plot_missing = TRUE, add_plot_histogram = TRUE)
#create_report(df4, y = "mean_price")
```

The functions in DataExplorer will be demonstrated using the Sleep Dataset that is in the VIM package of R. Loading the same.

```
library(VIM)
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## VIM is ready to use.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/statistikat/VIM/issues
```

```
##
## Attaching package: 'VIM'
```

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

```
data("sleep")
d2 <- sleep
```

The `introduce()` function provides a snapshot of the input dataset, with information about dimensions of the data, the number of discrete and continuous columns and missing values amongst others.

Demonstrating the use of this function using the `Air Quality` dataset that is inbuilt in R

```
introduce(d2)
```

```
##   rows columns discrete_columns continuous_columns all_missing_columns
## 1   62      10                0                  10                   0
##   total_missing_values complete_rows total_observations memory_usage
## 1                   38            42                620         7184
```

As we can see above data set has 62 observations with 10 columns, and it has no discrete values columns. There are 38 missing values in the data set. We can study the same using the next very useful function in Data Explorer

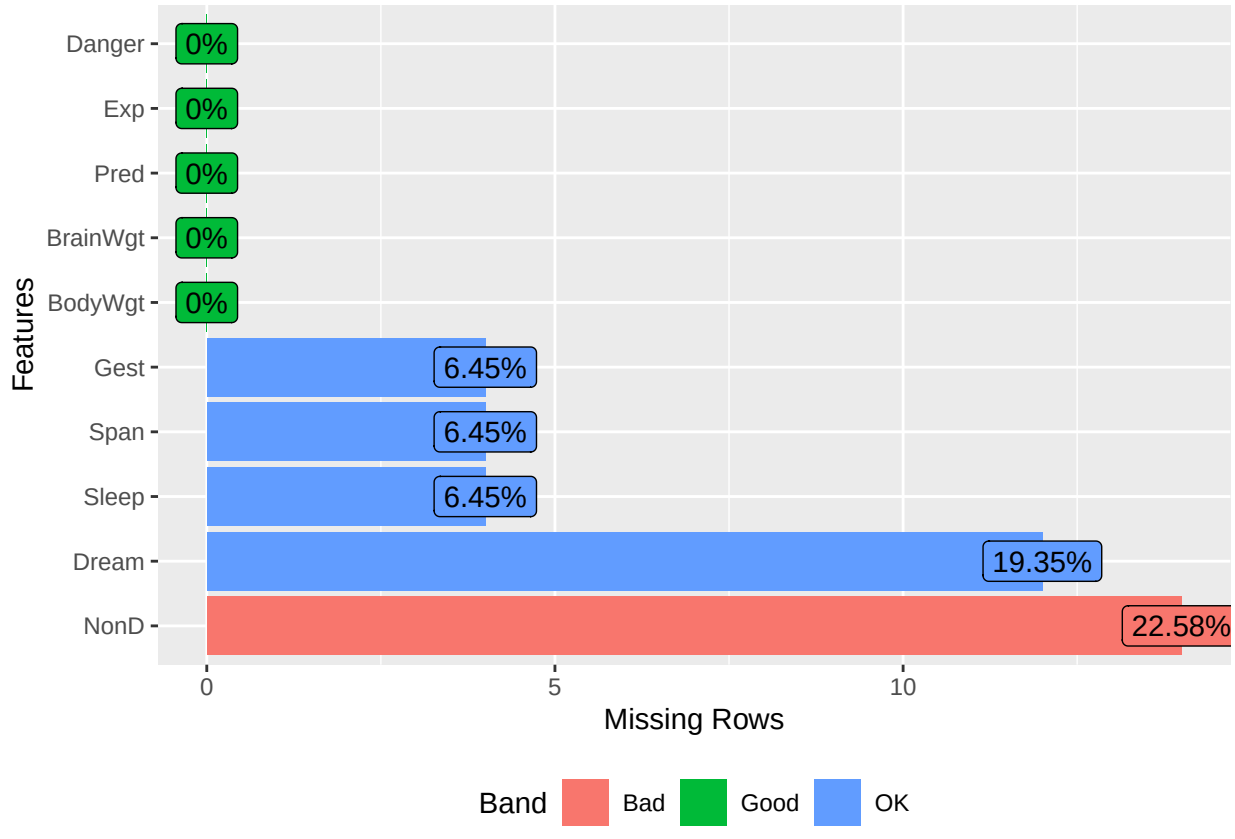
To understand various aspects of the missing values in a data set we use the `profile_missing()` function from Data Explorer

```
profile_missing(d2)
```

```
##     feature num_missing pct_missing
## 1   BodyWgt           0  0.00000000
## 2  BrainWgt           0  0.00000000
## 3      NonD          14  0.22580645
## 4     Dream          12  0.19354839
## 5     Sleep           4  0.06451613
## 6      Span           4  0.06451613
## 7      Gest           4  0.06451613
## 8      Pred           0  0.00000000
## 9       Exp           0  0.00000000
## 10   Danger           0  0.00000000
```

In the above table, we can see the number of missing values corresponding to each variable and then percentage of missing values as well. We can also visualize the above using the `plot_missing()` function

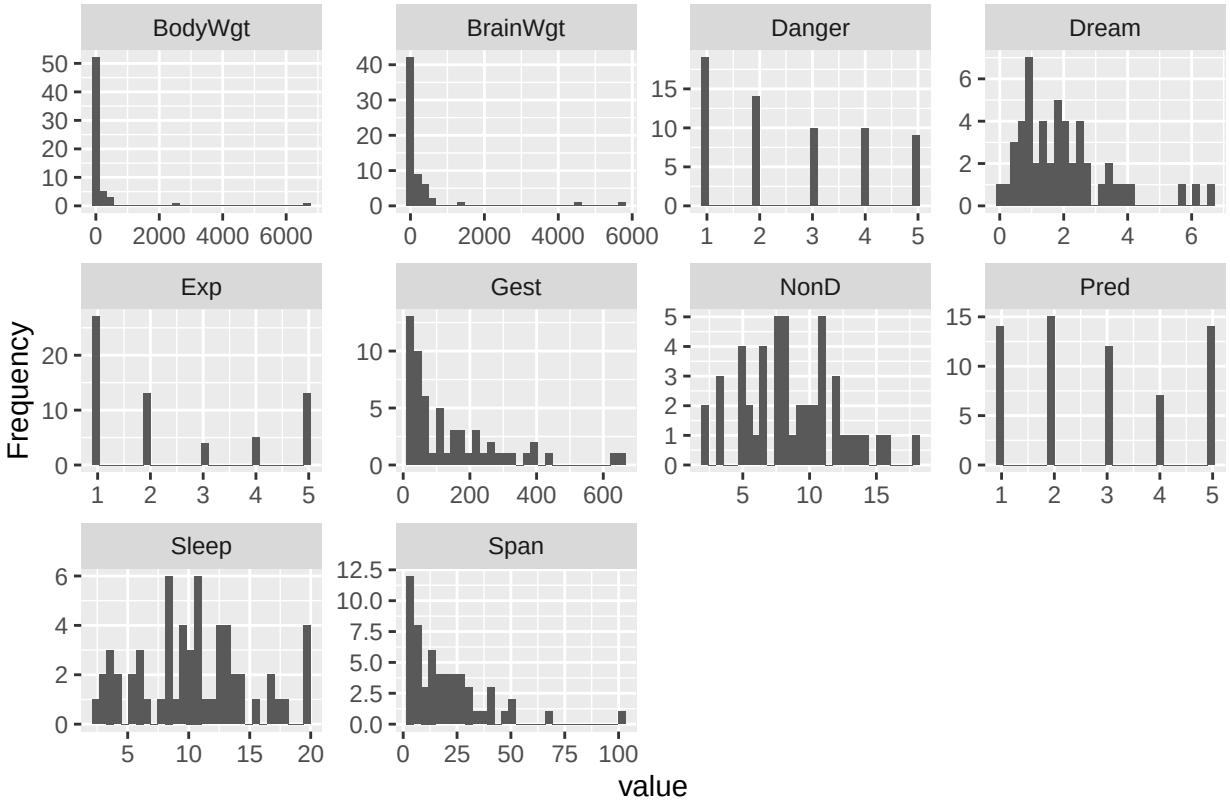
```
plot_missing(d2, group = list("Good" = 0.05, "OK" = 0.2, "Bad" = 0.4, "Remove" = 1))
```



The "Band" option allows us to set a threshold for how many missing values are okay. This is also customizable using the argument group as shown

The next most useful feature/function in this package is the wide array of plotting capabilities. Following are the examples of a few:

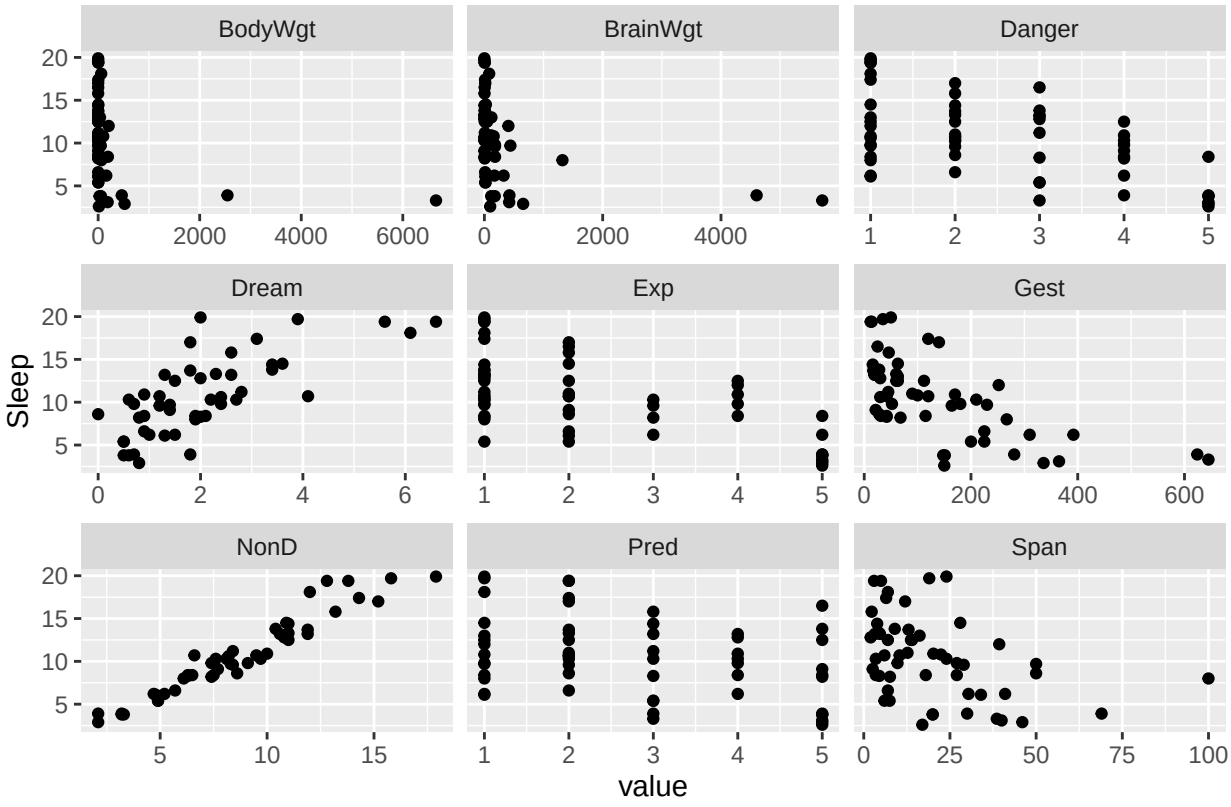
```
plot_histogram(d2) #Generates a histogram for all the variables in the data set
```



Histograms reveal information about the shape of the distribution of a particular variable and are of particular importance to the analyst if the model he/she plans on fitting have any underlying assumptions about the data. For example, requirement of normality in linear regression. Knowing the shape can help up also check if there are any transformations necessary for the variable to better suit further analysis including model fitting.

```
plot_scatterplot(d2, by = "Sleep") #This plots all the variables against a target variable specified. T
```

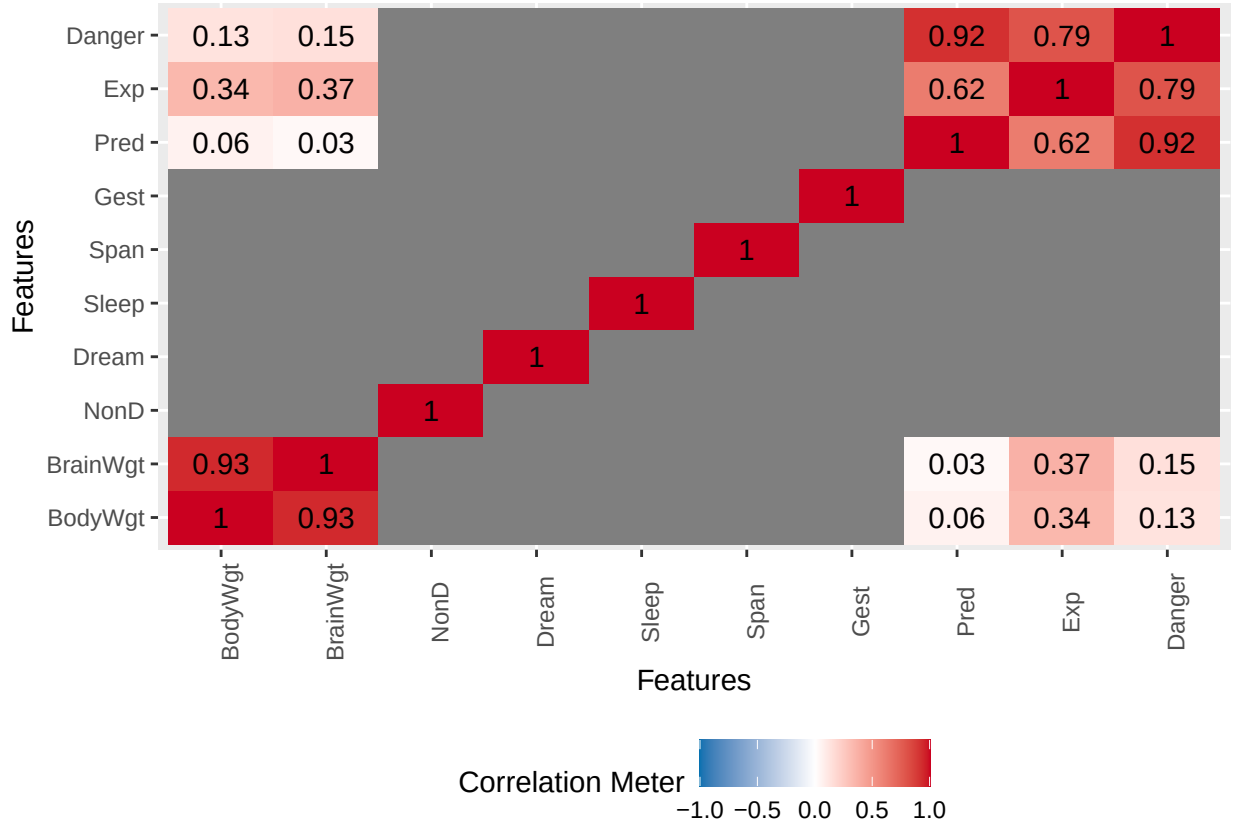
```
## Warning: Removed 64 rows containing missing values (`geom_point()`).
```



The scatter plots are a good way to understand the relationship amongst various variables in the data set. the shape of the points and the extent of their scatter or clustering can reveal new information about their relationship. The above scatter plot depicts the relationship between the variable Sleep and all other variables in the data set. For example, the variable NonD shares a very strong positive linear relationship wiht sleep and the variables Span and Gest share a weak negative relationship with Sleep. These plots can also be useful to make judgements about any transformations if necessary

```
plot_correlation(d2) #plots the correlation matrix with significant values
```

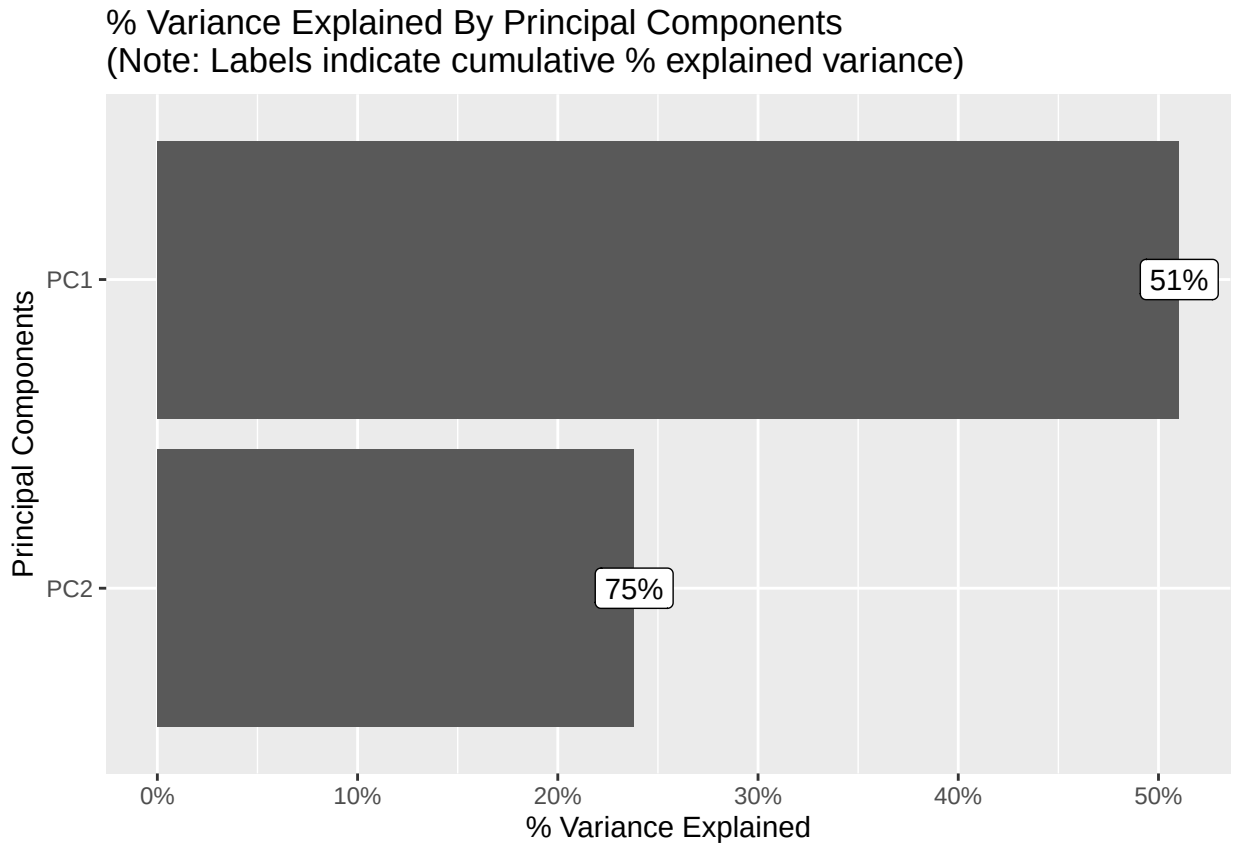
```
## Warning: Removed 70 rows containing missing values (`geom_text()`).
```

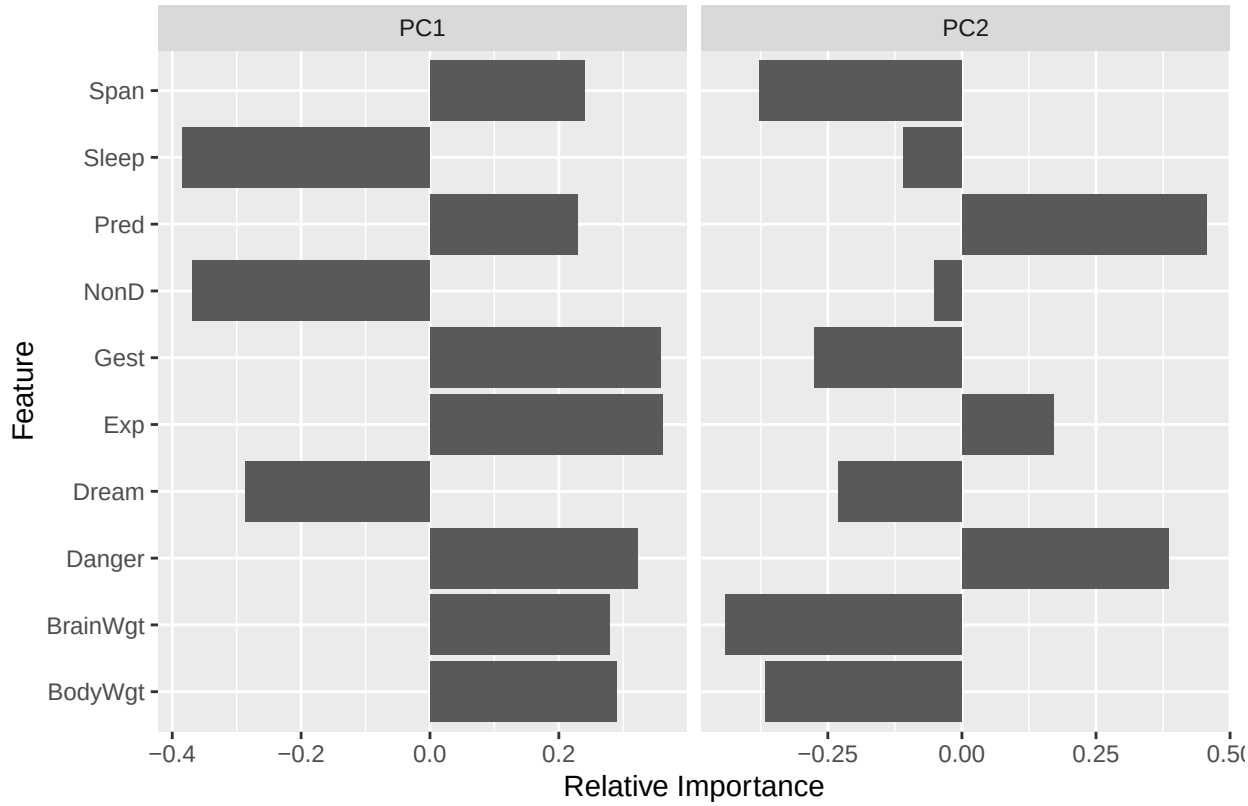


The Correlation plot gives a visual depiction of the the strength linear relationship between each pair of variables in the data set. This linear relationship can be positive (one goes up when the other goes up) or negative (one goes up when the other goes down). For example, in our data, we see that Brain Weight and Body Weight are highly positively correlated.

Lastly, following is a small demonstration of PCA or Principal Component Analysis that is generally performed when there is high Multicollinearlity (correlation between predictor variables) or a large number of predictor variables.

```
d3 <- d2 %>% drop_na() #removing the missing values before performing PCA
plot_prcomp(d3) #generates 2 plots
```





The first plot shows the amount of variance individually explained by each Principal Component (which is a a linear combination of the variables in the data set) as well as the cumulative amount of variation explained. In the example above, the first Principal Component (PC1) accounts for 51% of variation in the data and PC1 and PC2 together account of 75% of the variation in the data.

The Second plot illustrates the weightage or relative importance given to the individual features in each of the Principal Components. We see that in PC1, Sleep has the highest negative influence on the PC and the variables Gest, Exp and Danger have the highest positive influence on the PC.

Apart from this the Data Explorer package also allows the user to create Dummy variables, set values for the missing observation, add, drop and modify columns along with the wide range of plotting capabilities.

This package is therefore extremely useful, with vast applications and can save a lot of time for users in their data analysis process.

Class and Methods

Creating a function to perform a correlation analysis between the numeric variables in a data set

```
correl <- function(df){
  mat <- (df %>% select(where(is.numeric)) %>% drop_na() %>% cor())
  #need to include only numeric variables in calculation. Also removing NA values if any.
  print(mat) #printing the output of the function as a matrix before changing its class
  class(mat) = "correlation" #assigning the class to the output object
  return(mat) #returning the object of class "correlation"
}
```

- **The above function accepts a data set, selects the numeric variables, filters out the missing values and calculates the correlation coefficient between each pair of variables using the `cor()` function in R.**
- **The function then prints the output of the the `cor()` function which is of the class `matrix` and results in the correlation matrix. The function then changes the class of the function to `correlation` and returns the converted object.**

```
#applying the function to the cleaned dataset df4 from part 1 and 2
cormat <- correl(df4) #naming the returned value of the function as cormat
```

```
##               mean_price median_price  sale_value sale_volume
## mean_price    1.00000000    0.8739588 -0.07992121  -0.2219669
## median_price  0.87395879    1.0000000 -0.12542014  -0.2424175
## sale_value   -0.07992121   -0.1254201  1.00000000   0.9707428
## sale_volume  -0.22196690   -0.2424175  0.97074282   1.0000000
```

- **The printed matrix above is the default output of print function when applied to matrices. We will no define the print function for the class `correlation`**
- **The print function will result in a nested list of values, each of which will give the values of correlation coefficients of the variable paired with the other variables in the data set.**

```
print.correlation <- function(cormatrix){
  #list of correlation coefficients with other numeric variables
  l <- lapply(rownames(cormatrix),function(y) cormatrix[cormatrix[,y]!=1,y])
  names(l) <- rownames(cormatrix)
  #reassigning names of list to access specific variables and their correlation
  return(l)
}
```

```
print(cormat) #testing the function on the output obtained above using correl() function
```

```
## $mean_price
## median_price   sale_value  sale_volume
##   0.87395879  -0.07992121  -0.22196690
##
## $median_price
##  mean_price  sale_value sale_volume
##   0.8739588  -0.1254201  -0.2424175
##
## $sale_value
##   mean_price median_price  sale_volume
##  -0.07992121  -0.12542014   0.97074282
##
## $sale_volume
##   mean_price median_price   sale_value
##   -0.2219669   -0.2424175    0.9707428
```

```
#checking if accessing specific variables' correlation coefficients is possible
print(cormat)$median_price #Correlation Coefficients of Median Price with the remaining variables
```

```
##  mean_price  sale_value sale_volume
##   0.8739588  -0.1254201  -0.2424175
```

```
print(cormat)$sale_volume['mean_price'] #Correlation coefficient of Sale Volume and Mean Price
```

```
## mean_price
## -0.2219669
```

We see above that we can access the necessary values for a specific pair of variable

Constructing the summary function to produce the following outputs:

1. **List of all the Numeric Variables**
2. **Nested List of Correlation Coefficients (Similar to Print function)**
3. **The Variable pair with the largest correlation coefficent along with the value**

```
summary.correlation <- function(cormatrix){

  #list of numeric variables
  NumericVariables <- rownames(cormatrix)

  #list of correlation coefficients with other numeric variables
  l <- lapply(rownames(cormatrix),function(y) cormatrix[cormatrix[,y]!=1,y])
  names(l) <- rownames(cormatrix)

  #most correlated pair
  m = unlist(l)[which(abs(unlist(l)) == max(abs(unlist(l))))][1]
  max.corr = list(variables = strsplit(names(m), split = "\\.")[[1]], value = as.vector(m))

  summaryl <- list(NumericVariables, l, max.corr)
  names(summaryl) <- c("Num.Var", "Corr.Pairs", "Max.Corr")

  return(summaryl)
}
```

```
#testing the function method on `cormat`
summary(cormat)
```

```
## $Num.Var
## [1] "mean_price"   "median_price" "sale_value"   "sale_volume"
##
## $Corr.Pairs
## $Corr.Pairs$mean_price
## median_price   sale_value  sale_volume
##   0.87395879  -0.07992121  -0.22196690
##
## $Corr.Pairs$median_price
##  mean_price  sale_value sale_volume
##   0.8739588  -0.1254201  -0.2424175
##
## $Corr.Pairs$sale_value
##   mean_price median_price  sale_volume
##  -0.07992121  -0.12542014   0.97074282
##
## $Corr.Pairs$sale_volume
##   mean_price median_price   sale_value
##   -0.2219669   -0.2424175    0.9707428
##
##
## $Max.Corr
## $Max.Corr$variables
## [1] "sale_value"  "sale_volume"
##
## $Max.Corr$value
## [1] 0.9707428
```

It is also possible to extract specific parts of the summary output as illustrated below

```
summary(cormat)$Max.Corr$variables #pair of variables with largest correlation
```

```
## [1] "sale_value"  "sale_volume"
```

```
summary(cormat)$Corr.Pairs$sale_volume #correlation coefficients associated with Sale Volume
```

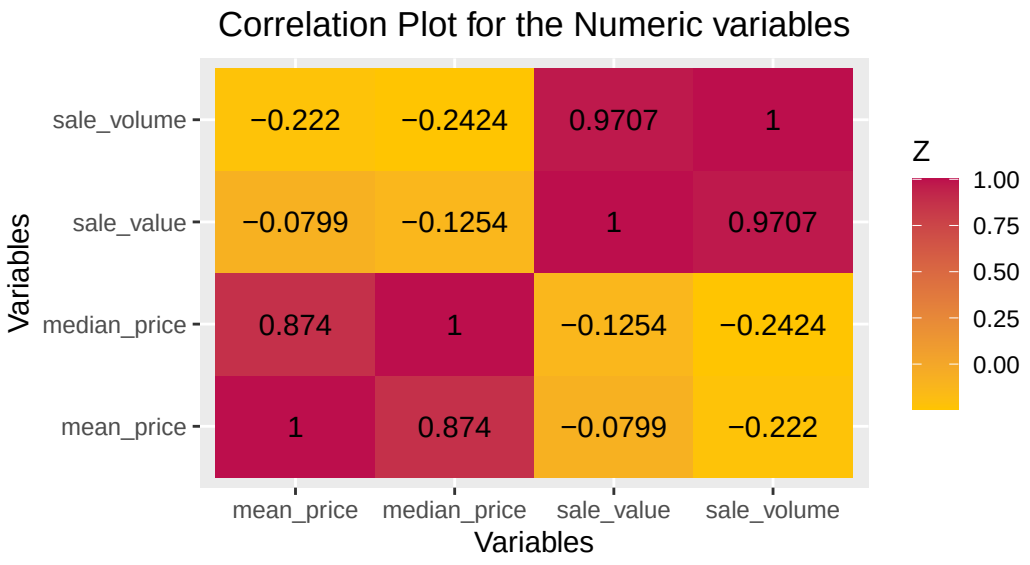
```
##   mean_price median_price   sale_value
##   -0.2219669   -0.2424175    0.9707428
```

We now define the plot function for this class. We construct a correlation plot using the geom_tile() function of ggplot2

```
plot.correlation <- function(cormatrix){
  n = length(rownames(cormatrix))
  var = rownames(cormatrix)
  X = NULL #defining null variables to append with paramters of the graph
  Y = NULL
  Z = NULL
  for(i in 1:n){
    for(j in 1:n){
      X = append(X, var[i])
      Y = append(Y, var[j])
      Z = append(Z, cormatrix[i,j])
    }
  }
  ggplot()+
    geom_tile(aes(x=X, y=Y, fill=Z))+ #using the correlation coefficients to define fill of tiles
    scale_fill_gradient(high = "#BC0E4C",low = "#FFC501", guide = "colorbar")+
    geom_text(aes(X,Y,label = round(Z,4)))+ #adding labels with respective values of corr. coeff.
    labs(title = "Correlation Plot for the Numeric variables", x = "Variables", y = "Variables")+
    theme(plot.title = element_text(hjust = 0.5))
}
```

Testing the plot function

```
plot(cormat) #testing using the output of correl() function using df4 dataset
```



The above graph provides the correlation plot along with the correlation coefficients rounded upto 4 decimal places

---

References: Additional Packages Used:

1. Pedersen T (2022). *patchwork: The Composer of Plots*. R package version 1.1.2, https://CRAN.R-project.org/package=patchwork. (Package Patchwork)

2. Cui B (2020). *DataExplorer: Automate Data Exploration and Treatment*. R package version 0.8.2, https://CRAN.R-project.org/package=DataExplorer (Package DataExplorer)