main

2024-06-10

Importing dataset and libraries

Firstly, we are going to fix all features in our dataset, we will start from the feature “Item_Fat_Content”.

We can see that there are different names for the same value:

```
item_fat_content_uniques <- unique(df$Item_Fat_Content)#unique values of item_fat_content

print(item_fat_content_uniques)
```

```
## [1] "Low Fat" "Regular" "low fat" "LF"      "reg"
```

As we can see, we have 2 different names for “Regular Fat” that are “Regular” and “reg” and 3 different names for “Low Fat” that are “Low fat”, “low fat”, “LF”

To solve this problem, we can edit the column “Item_Fat_Content” in this way:

all records that start with R or r, will be renamed “Regular Fat”, and other will be renamed “Low Fat”.

Then we will print the uniques name of the column after the edit

```
df$Item_Fat_Content <- ifelse(grepl("^[Rr]", df$Item_Fat_Content), "Regular Fat", "Low Fat")
item_fat_content_uniques <- unique(df$Item_Fat_Content)

print(item_fat_content_uniques)
```

```
## [1] "Low Fat"     "Regular Fat"
```

Now we should work on the column “Outlet_Size” since analyzing the dataset manually it’s clear that there are many missing values. First of all, we should compute the percentage of missing values.

```
#conteggio records senza Outlet_Size:
null<-sum(nchar(df$Outlet_Size) ==0)

#numero di records
total<- nrow(df)

print(paste("Number of rows with missing valuess on Outlet_Size:",null))
```

```
## [1] "Number of rows with missing valuess on Outlet_Size: 2410"
```

```
#percentuale di righe senza outlet_size
percentuale_stringhe_vuote <- (null / total) * 100

#stampa percentuale di righe senza outlet_size

print(paste("Percentuale di records con stringhe vuote in 'Outlet_Size':", percentuale_stringhe_vuote,
```

```
## [1] "Percentuale di records con stringhe vuote in 'Outlet_Size': 28.2764284876217 %"
```

We want to find out if there are relations between the column "Outlet_Type" and the column "Outlet_Size" creating a table to see the number of combination between the outlet size and the outlet type.

```
# Creazione di una tabella di riepilogo
outlet_summary <- df %>%
  filter(Outlet_Size %in% c("Small", "Medium", "High")) %>%  # Filtra per includere solo le righe con i
  group_by(Outlet_Type, Outlet_Size) %>%  # Raggruppa per tipo e dimensione del negozio
  summarise(Count = n(), .groups = 'drop')  # Calcola il conteggio e rimuove il raggruppamento automati

# Visualizzazione della tabella di riepilogo
print(outlet_summary)
```

```
## # A tibble: 6 x 3
##   Outlet_Type       Outlet_Size Count
##   <chr>             <chr>       <int>
## 1 Grocery Store     Small         528
## 2 Supermarket Type1 High          932
## 3 Supermarket Type1 Medium        930
## 4 Supermarket Type1 Small        1860
## 5 Supermarket Type2 Medium        928
## 6 Supermarket Type3 Medium        935
```

From this tab, we can see that all entries that are "Grocery Store" are small, and all entries that are "Supermarket Type2" or "Supermarket Type3" are Medium. Knowing this, we can substitute blank values of "Outlet_Size" of grocery with small, of type2 and type 3 with medium.

```
# Aggiornamento della colonna 'Outlet_Size'
df <- df %>%
  mutate(Outlet_Size = case_when(
    Outlet_Type == "Grocery Store" ~ "Small",
    #Outlet_Type %in% c("Supermarket Type2", "Supermarket Type3") ~ "Medium",
    TRUE ~ Outlet_Size  # Mantiene il valore originale per tutte le altre condizioni
  ))

# Visualizza le modifiche per confermare
# Creazione di una tabella di riepilogo
outlet_summary <- df %>%
  filter(Outlet_Size %in% c("Small", "Medium", "High")) %>%  # Filtra per includere solo le righe con i
  group_by(Outlet_Type, Outlet_Size) %>%  # Raggruppa per tipo e dimensione del negozio
  summarise(Count = n(), .groups = 'drop')  # Calcola il conteggio e rimuove il raggruppamento automati

# Visualizzazione della tabella di riepilogo
print(outlet_summary)
```

```
## # A tibble: 6 x 3
##   Outlet_Type       Outlet_Size Count
##   <chr>             <chr>       <int>
## 1 Grocery Store     Small        1083
## 2 Supermarket Type1 High          932
## 3 Supermarket Type1 Medium        930
```

```
## 4 Supermarket Type1 Small        1860
## 5 Supermarket Type2 Medium        928
## 6 Supermarket Type3 Medium        935
```

```
#conteggio records senza Outlet_Size:
null<-sum(nchar(df$Outlet_Size) ==0)

#numero di records
total<- nrow(df)

print(paste("Number of rows with missing valuess on Outlet_Size:",null))
```

```
## [1] "Number of rows with missing valuess on Outlet_Size: 1855"
```

```
#percentuale di righe senza outlet_size
percentuale_stringhe_vuote <- (null / total) * 100

#stampa percentuale di righe senza outlet_size

print(paste("Percentuale di records con stringhe vuote in 'Outlet_Size':", percentuale_stringhe_vuote,
```

```
## [1] "Percentuale di records con stringhe vuote in 'Outlet_Size': 21.7646368649537 %"
```

Dopo aver riempito grocery store, supermarket type2 e supermarket type3, il numero di na è sceso dal 28% al 21%, passando da 2410 valori mancanti a 1855.

Now we will fill the remaining missing values with n/a

```
# Aggiornamento della colonna 'Outlet_Size' per riempire le stringhe vuote
df <- df %>%
  mutate(Outlet_Size = if_else(nchar(Outlet_Size) == 0, "NA", Outlet_Size))
```

Now, we want to convert as factor the columns "Item_Fat_Content" giving 0 to "Low Fat" and 1 to "Regular". And we also want to convert the column Outlet_Size: 1->Small 2->Medium 3->Large

```
# Conversione di 'Outlet_Size' in valori numerici
df <- df %>%
  mutate(Outlet_Size = case_when(
    Outlet_Size == "Small" ~ 1,
    Outlet_Size == "Medium" ~ 2,
    Outlet_Size == "High" ~ 3,
    TRUE ~ NA_real_  # Imposta NA per qualsiasi altro valore non specificato
  ))

df <- df%>%
  mutate(Item_Fat_Content = case_when(
    Item_Fat_Content == "Low Fat" ~ 1,
    Item_Fat_Content == "Regular Fat" ~ 2,
    TRUE ~ NA_real_
  ))
head(df)
```

```
##   Item_Identifier Item_Weight Item_Fat_Content Item_Visibility
## 1           FDA15       9.300                1      0.01604730
## 2           DRC01       5.920                2      0.01927822
## 3           FDN15      17.500                1      0.01676007
## 4           FDX07      19.200                2      0.00000000
## 5           NCD19       8.930                1      0.00000000
## 6           FDP36      10.395                2      0.00000000
##                 Item_Type Item_MRP Outlet_Identifier Outlet_Establishment_Year
## 1                   Dairy 249.8092            OUT049                      1999
## 2             Soft Drinks  48.2692            OUT018                      2009
## 3                    Meat 141.6180            OUT049                      1999
## 4 Fruits and Vegetables 182.0950            OUT010                      1998
## 5               Household  53.8614            OUT013                      1987
## 6            Baking Goods  51.4008            OUT018                      2009
##   Outlet_Size Outlet_Location_Type        Outlet_Type Item_Outlet_Sales
## 1           2               Tier 1 Supermarket Type1         3735.1380
## 2           2               Tier 3 Supermarket Type2          443.4228
## 3           2               Tier 1 Supermarket Type1         2097.2700
## 4           1               Tier 3     Grocery Store          732.3800
## 5           3               Tier 3 Supermarket Type1          994.7052
## 6           2               Tier 3 Supermarket Type2          556.6088
```

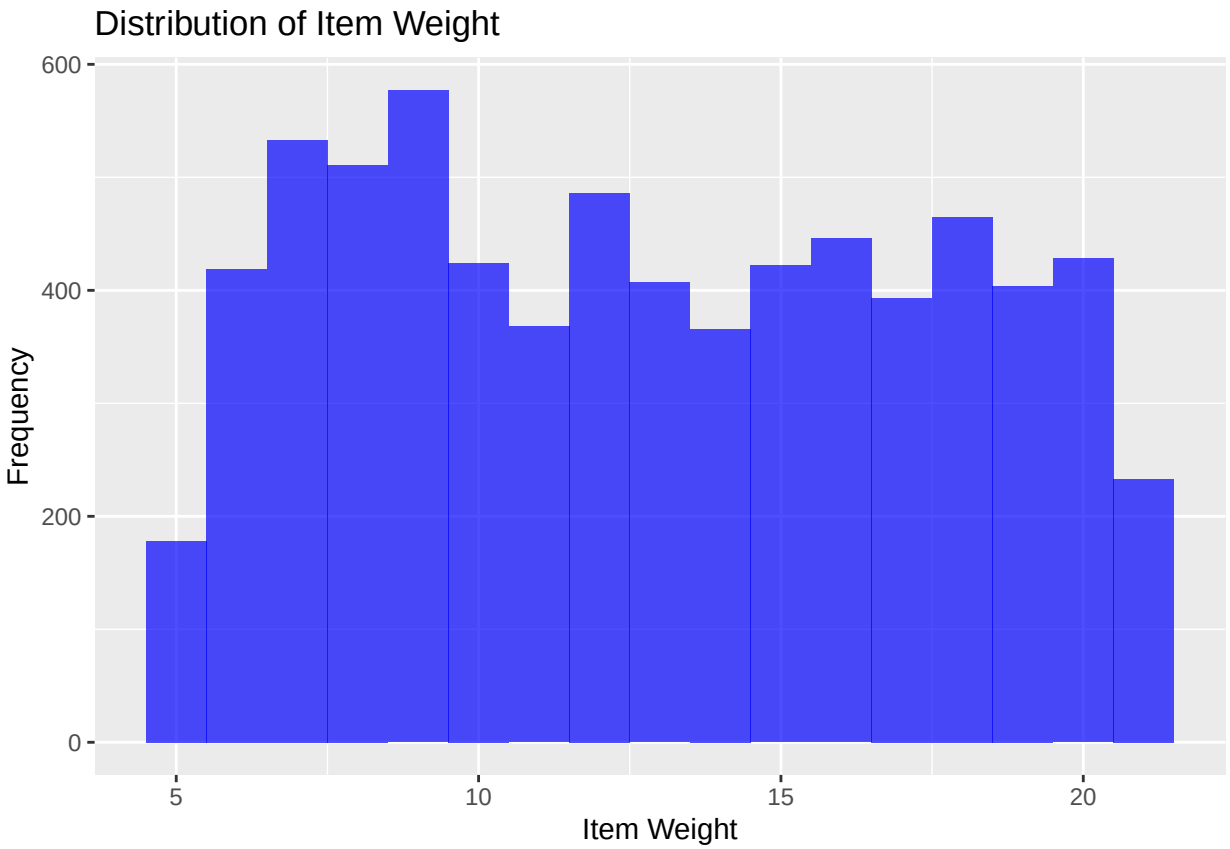
```r
# Find the number of zero 'Item_Visibility' values for each 'Outlet_Identifier'
zero_visibility_counts <- aggregate(Item_Visibility ~ Outlet_Identifier, data = df, function(x) sum(x ==

# Rename the column for better understanding
names(zero_visibility_counts)[2] <- "Zero_Item_Visibility_Count"

# Display the result
print(zero_visibility_counts)
```

```
##    Outlet_Identifier Zero_Item_Visibility_Count
## 1             OUT010                        29
## 2             OUT013                        59
## 3             OUT017                        57
## 4             OUT018                        65
## 5             OUT019                        30
## 6             OUT027                        60
## 7             OUT035                        54
## 8             OUT045                        58
## 9             OUT046                        61
## 10            OUT049                        53
```

```r
# Load necessary libraries
library(ggplot2)
library(gridExtra)

# Histogram and Box Plot for Item_Weight
p1 <- ggplot(df, aes(x = Item_Weight)) +
  geom_histogram(binwidth = 1, fill = 'blue', alpha = 0.7) +
  ggtitle("Distribution of Item Weight") +
  xlab("Item Weight") +
  ylab("Frequency")
```

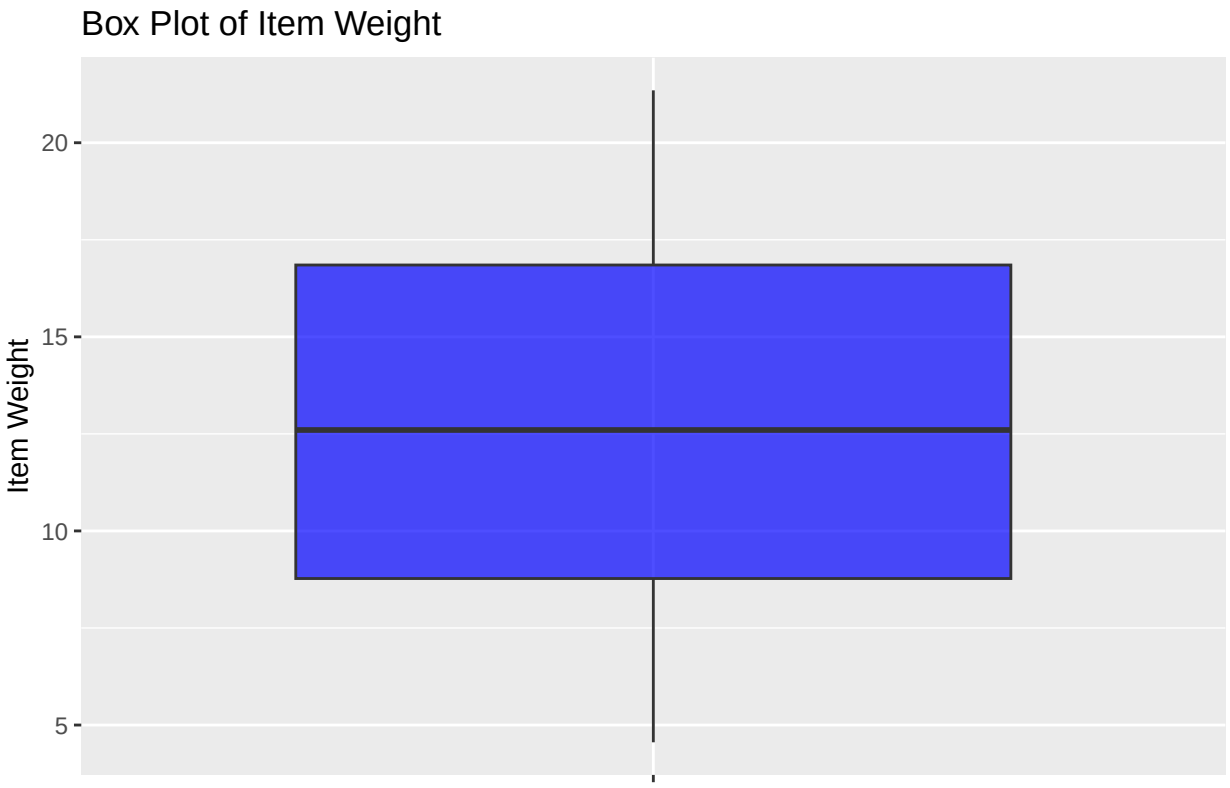
```r
p2 <- ggplot(df, aes(x = "", y = Item_Weight)) +
  geom_boxplot(fill = 'blue', alpha = 0.7) +
  ggtitle("Box Plot of Item Weight") +
  xlab("") +
  ylab("Item Weight")

# Histogram and Box Plot for Item_Visibility
p3 <- ggplot(df, aes(x = Item_Visibility)) +
  geom_histogram(binwidth = 0.01, fill = 'green', alpha = 0.7) +
  ggtitle("Distribution of Item Visibility") +
  xlab("Item Visibility") +
  ylab("Frequency")

p4 <- ggplot(df, aes(x = "", y = Item_Visibility)) +
  geom_boxplot(fill = 'green', alpha = 0.7) +
  ggtitle("Box Plot of Item Visibility") +
  xlab("") +
  ylab("Item Visibility")

# Histogram and Box Plot for Item_MRP
p5 <- ggplot(df, aes(x = Item_MRP)) +
  geom_histogram(binwidth = 5, fill = 'red', alpha = 0.7) +
  ggtitle("Distribution of Item MRP") +
  xlab("Item MRP") +
  ylab("Frequency")

p6 <- ggplot(df, aes(x = "", y = Item_MRP)) +
  geom_boxplot(fill = 'red', alpha = 0.7) +
  ggtitle("Box Plot of Item MRP") +
  xlab("") +
  ylab("Item MRP")

# Histogram and Box Plot for Item_Outlet_Sales
p7 <- ggplot(df, aes(x = Item_Outlet_Sales)) +
  geom_histogram(binwidth = 100, fill = 'purple', alpha = 0.7) +
  ggtitle("Distribution of Item Outlet Sales") +
  xlab("Item Outlet Sales") +
  ylab("Frequency")

p8 <- ggplot(df, aes(x = "", y = Item_Outlet_Sales)) +
  geom_boxplot(fill = 'purple', alpha = 0.7) +
  ggtitle("Box Plot of Item Outlet Sales") +
  xlab("") +
  ylab("Item Outlet Sales")

# Print the plots individually
print(p1)
```

```
## Warning: Removed 1463 rows containing non-finite outside the scale range
## (`stat_bin()`).
```

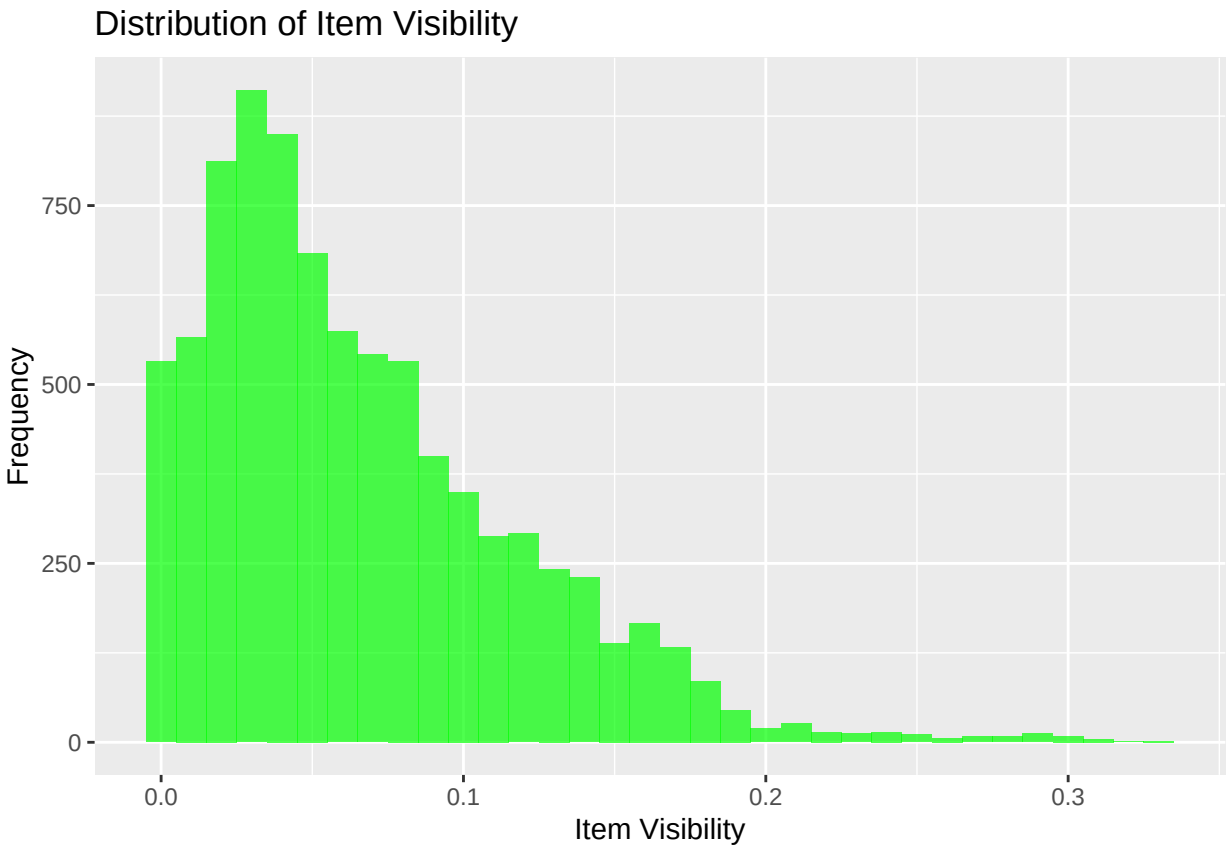



```
print(p2)
```

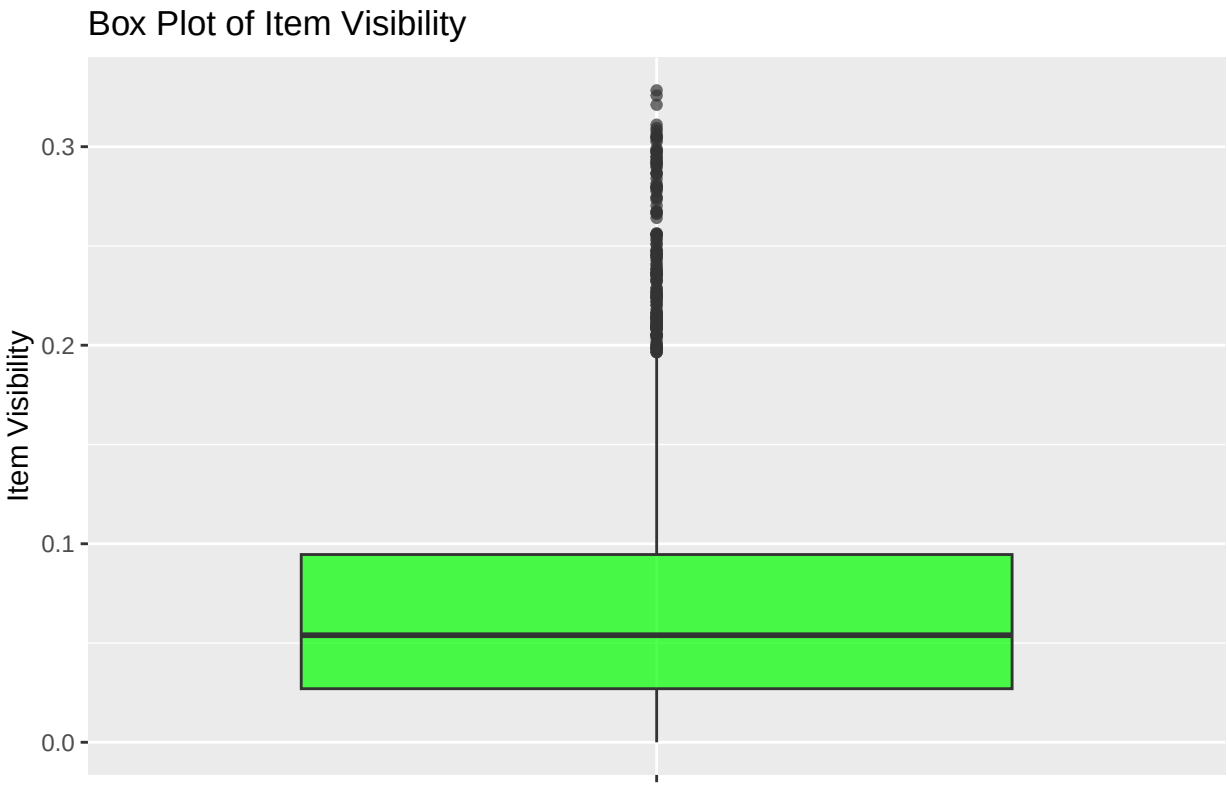
```
## Warning: Removed 1463 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



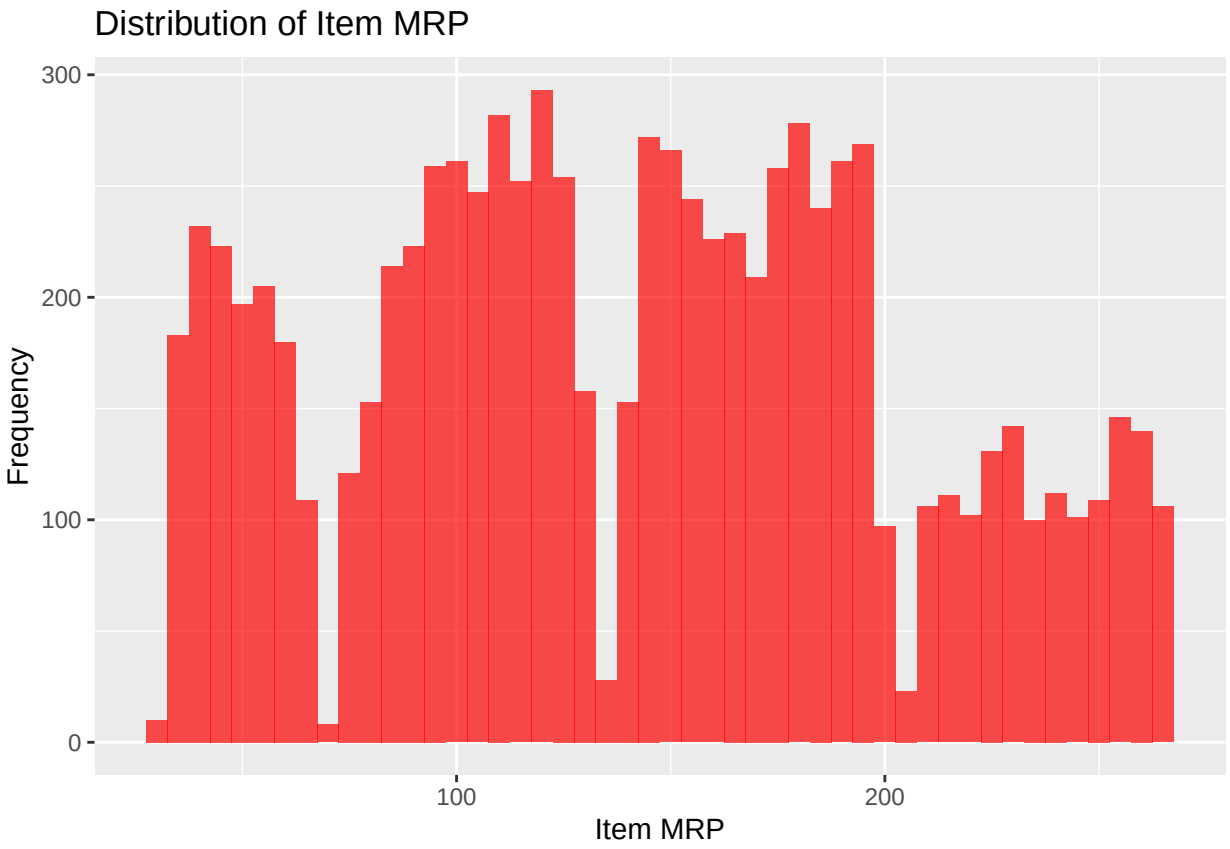

```
print(p3)
```


Distribution of Item Visibility
Frequency
750
500
250
0
0.0
0.1
0.2
0.3
Item Visibility

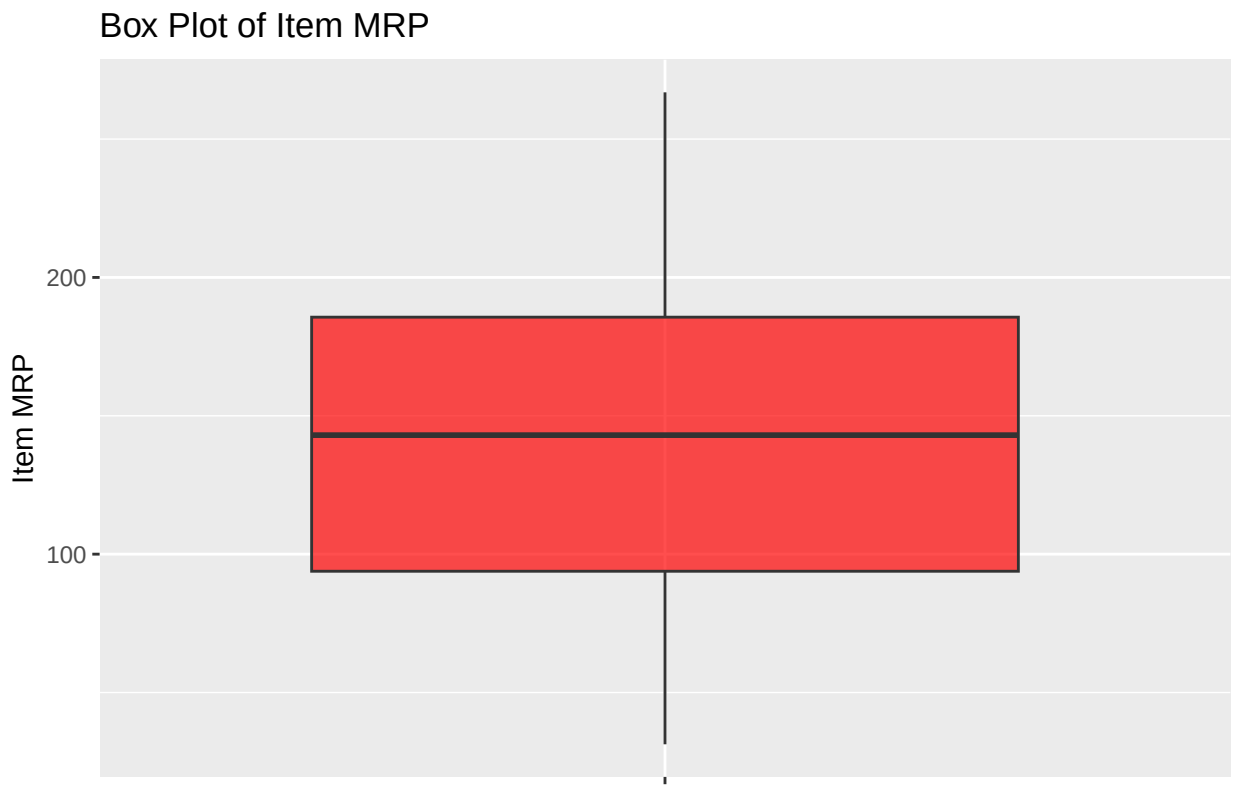
```
print(p4)
```



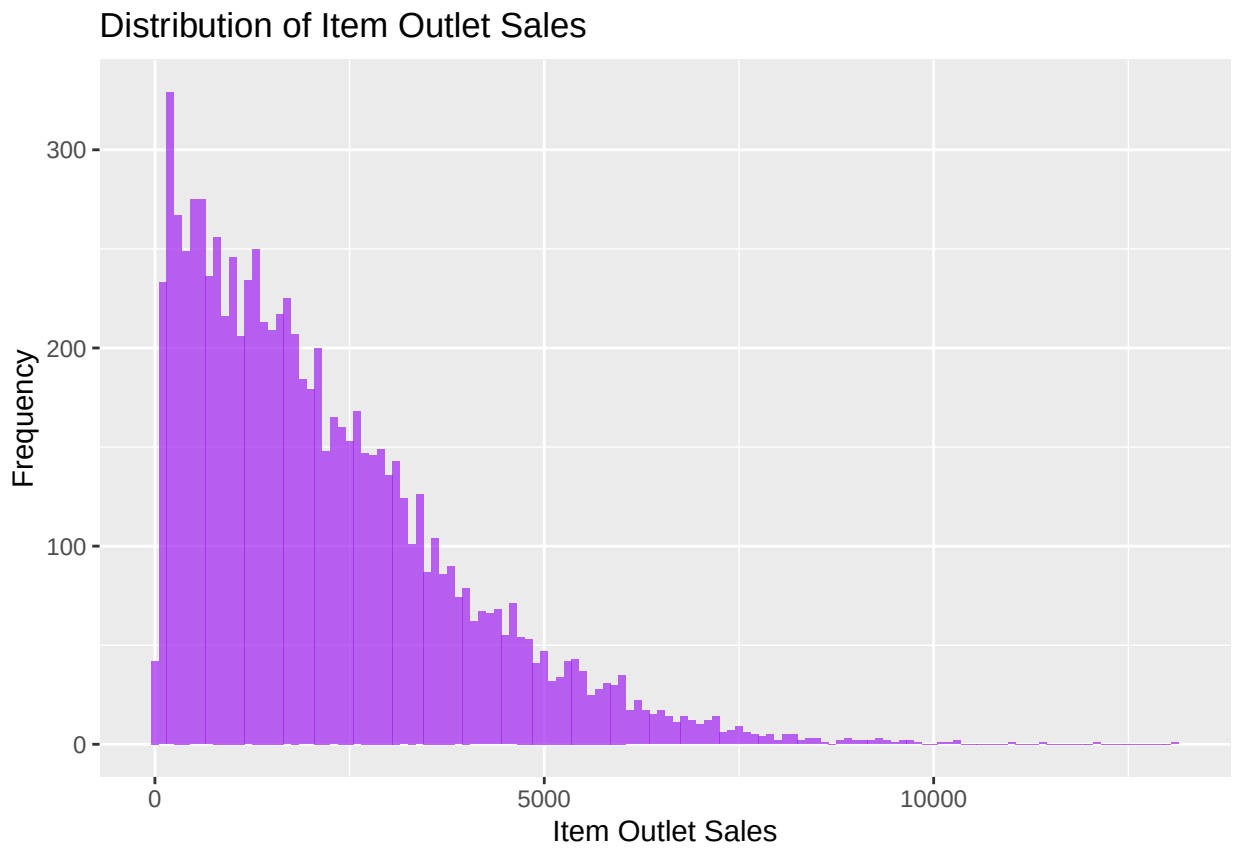

```
print(p5)
```


Distribution of Item MRP
300
200
100
0
Frequency
100
200
Item MRP

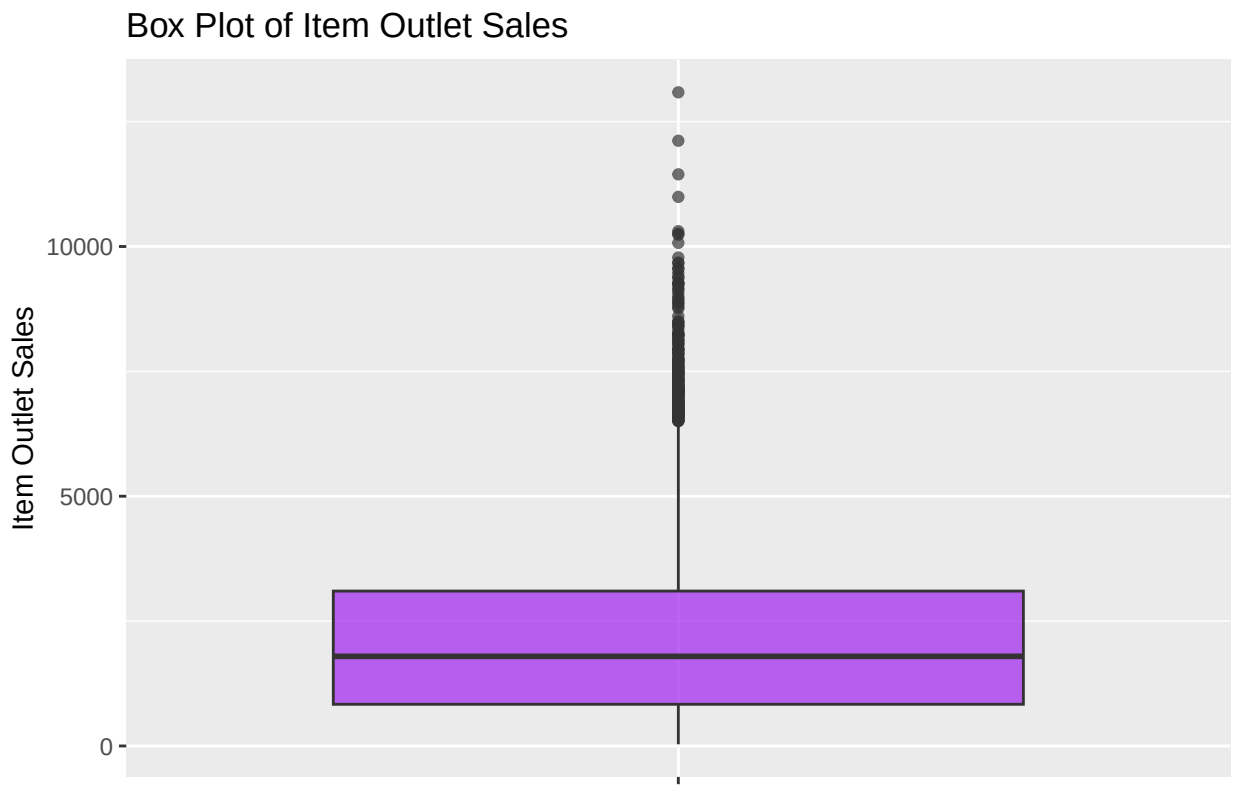
```
print(p6)
```




```
print(p7)
```


Distribution of Item Outlet Sales
Frequency
300
200
100
0
0
5000
10000
Item Outlet Sales

```
print(p8)
```




```
visibility_sum_per_store <- aggregate(Item_Visibility ~ Outlet_Identifier, data = df, sum)
names(visibility_sum_per_store)[2] <- "Total_Item_Visibility"

# Display the result
print(visibility_sum_per_store)
```

```
##    Outlet_Identifier Total_Item_Visibility
## 1             OUT010              56.30883
## 2             OUT013              55.87986
## 3             OUT017              56.83465
## 4             OUT018              56.62145
## 5             OUT019              57.25704
## 6             OUT027              54.80476
## 7             OUT035              56.97487
## 8             OUT045              56.18078
## 9             OUT046              56.23188
## 10            OUT049              56.54916
```

```
# Caricare il dataset

# Filtrare i record con visibilità maggiore di zero per calcolare le medie
filtered_df <- df[df$Item_Visibility > 0, ]

# Calcolare la media di Item_Visibility per ogni combinazione di Item_Type e Outlet_Size
visibility_avg_per_type_size <- aggregate(Item_Visibility ~ Item_Type + Outlet_Size, data = filtered_df
```

```r
# Creare una chiave unica per facilitare il merge
visibility_avg_per_type_size$key <- with(visibility_avg_per_type_size, paste(Item_Type, Outlet_Size, se
df$key <- with(df, paste(Item_Type, Outlet_Size, sep = "_"))

# Merge tra i record del dataframe originale e le medie calcolate usando un left join
df <- merge(df, visibility_avg_per_type_size, by = "key", all.x = TRUE, suffixes = c("", ".new"))

# Sostituire i valori zero di Item_Visibility con i valori medi calcolati
df$Item_Visibility[df$Item_Visibility == 0] <- df$Item_Visibility.new[df$Item_Visibility == 0]

# Rimuovere le colonne in più create dal merge
df <- df[, !names(df) %in% c("Item_Visibility.new", "key")]

# Visualizzare il dataframe aggiornato
head(df)
```

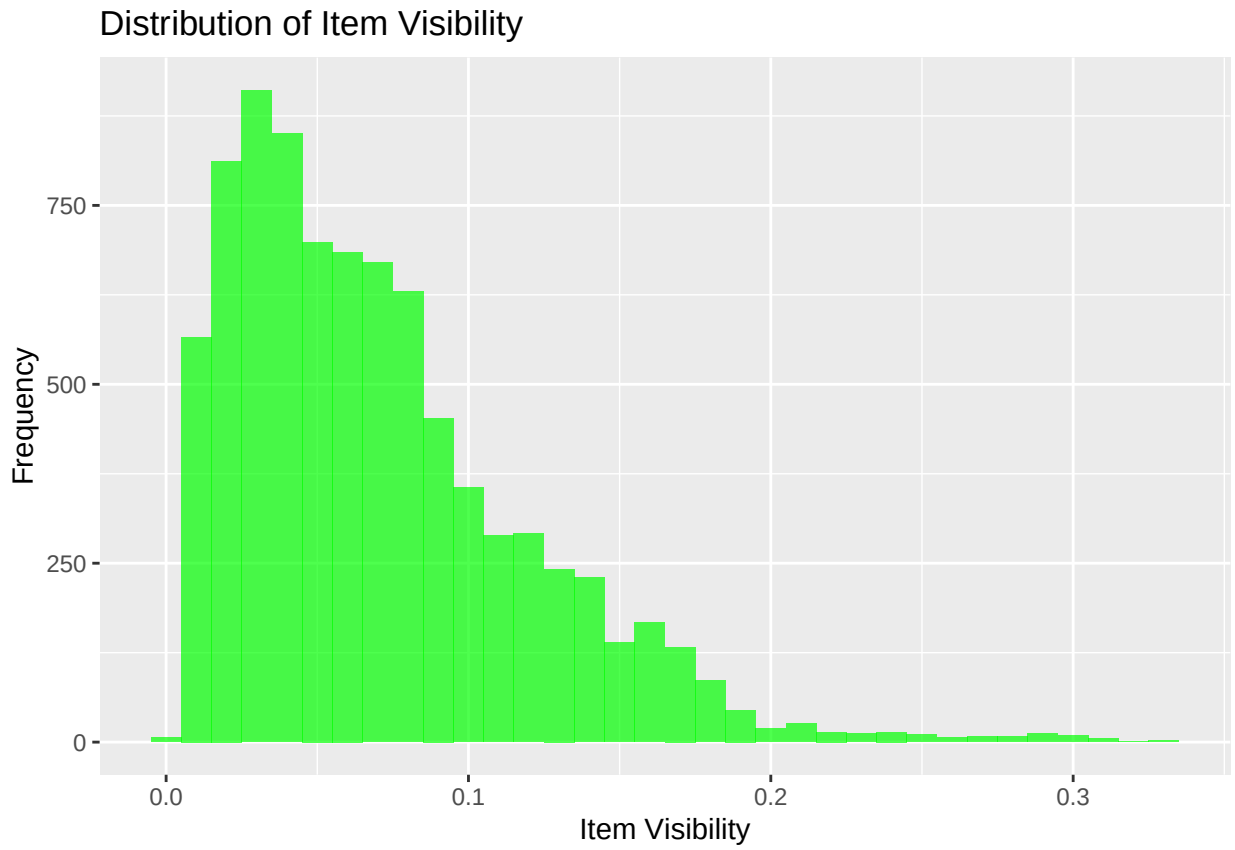
```
##   Item_Identifier Item_Weight Item_Fat_Content Item_Visibility    Item_Type
## 1           FDA11       7.750               1     0.043238822 Baking Goods
## 2           FDA36          NA               1     0.009921107 Baking Goods
## 3           FDQ12      12.650               1     0.035404052 Baking Goods
## 4           FDO12      15.750               1     0.054920146 Baking Goods
## 5           FDM60      10.800               2     0.048143292 Baking Goods
## 6           FDC48       9.195               1     0.015856295 Baking Goods
##   Item_MRP Outlet_Identifier Outlet_Establishment_Year Outlet_Size
## 1  92.5436            OUT046                      1997           1
## 2 183.6924            OUT019                      1985           1
## 3 230.6010            OUT035                      2004           1
## 4 195.8452            OUT035                      2004           1
## 5  40.2138            OUT046                      1997           1
## 6  81.6592            OUT035                      2004           1
##   Outlet_Location_Type       Outlet_Type Item_Outlet_Sales Item_Type.new
## 1               Tier 1 Supermarket Type1         1701.7848  Baking Goods
## 2               Tier 1     Grocery Store          555.2772  Baking Goods
## 3               Tier 2 Supermarket Type1         2067.3090  Baking Goods
## 4               Tier 2 Supermarket Type1         4893.6300  Baking Goods
## 5               Tier 1 Supermarket Type1          690.4346  Baking Goods
## 6               Tier 2 Supermarket Type1         1403.5064  Baking Goods
##   Outlet_Size.new
## 1               1
## 2               1
## 3               1
## 4               1
## 5               1
## 6               1
```

```r
df <- df %>%
  select(-Item_Type.new, -Outlet_Size.new)
```

```r
# Histogram and Box Plot for Item_Visibility
p3 <- ggplot(df, aes(x = Item_Visibility)) +
  geom_histogram(binwidth = 0.01, fill = 'green', alpha = 0.7) +
  ggtitle("Distribution of Item Visibility") +
```

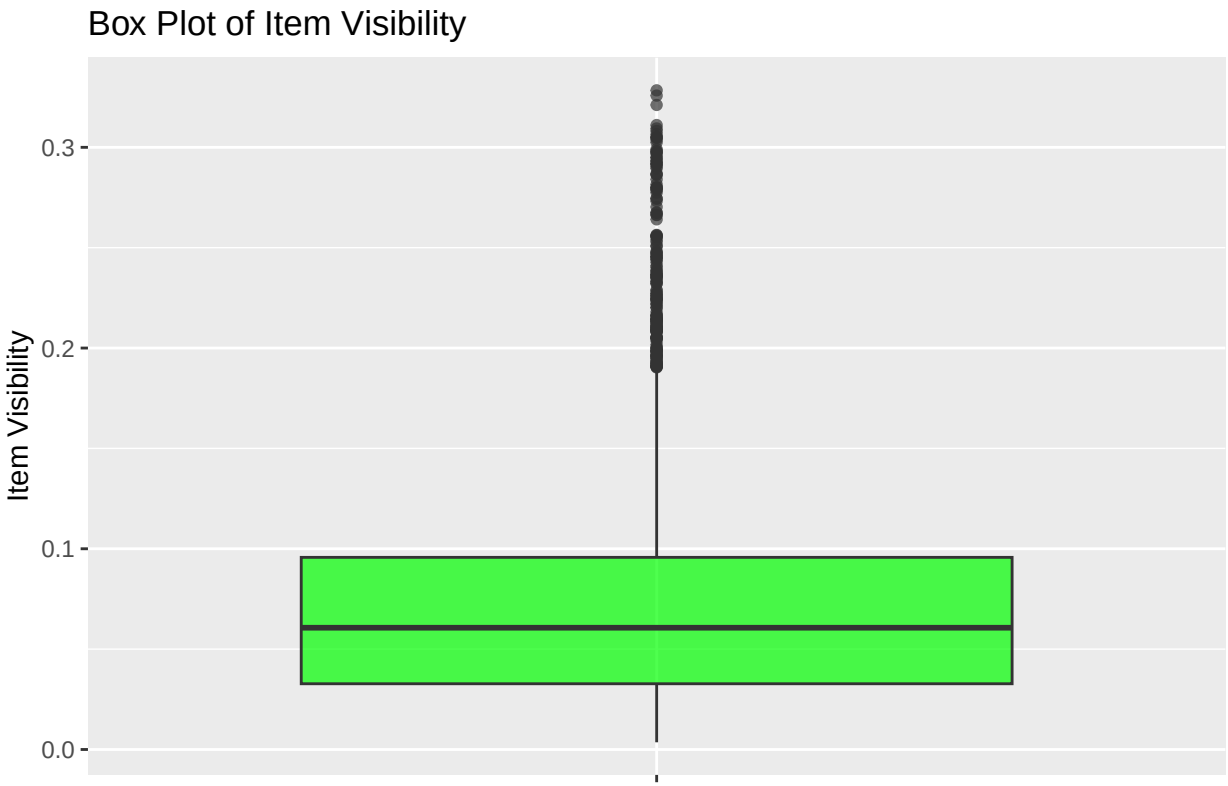
```
  xlab("Item Visibility") +
  ylab("Frequency")

p4 <- ggplot(df, aes(x = "", y = Item_Visibility)) +
  geom_boxplot(fill = 'green', alpha = 0.7) +
  ggtitle("Box Plot of Item Visibility") +
  xlab("") +
  ylab("Item Visibility")
print(p3)
```

```
## Warning: Removed 115 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```




```
print(p4)
```

```
## Warning: Removed 115 rows containing non-finite outside the scale range
## (‘stat_boxplot()‘).
```




```
# Check for entire row duplicates
duplicates_entire_row <- duplicated(df)

# Count the number of entirely duplicated rows
num_entire_row_duplicates <- sum(duplicates_entire_row)
print(paste("Number of entirely duplicated rows:", num_entire_row_duplicates))
```

```
## [1] "Number of entirely duplicated rows: 0"
```

```
# View the duplicated rows
entire_row_duplicate_entries <- df[duplicates_entire_row, ]
print("Entirely duplicated rows:")
```

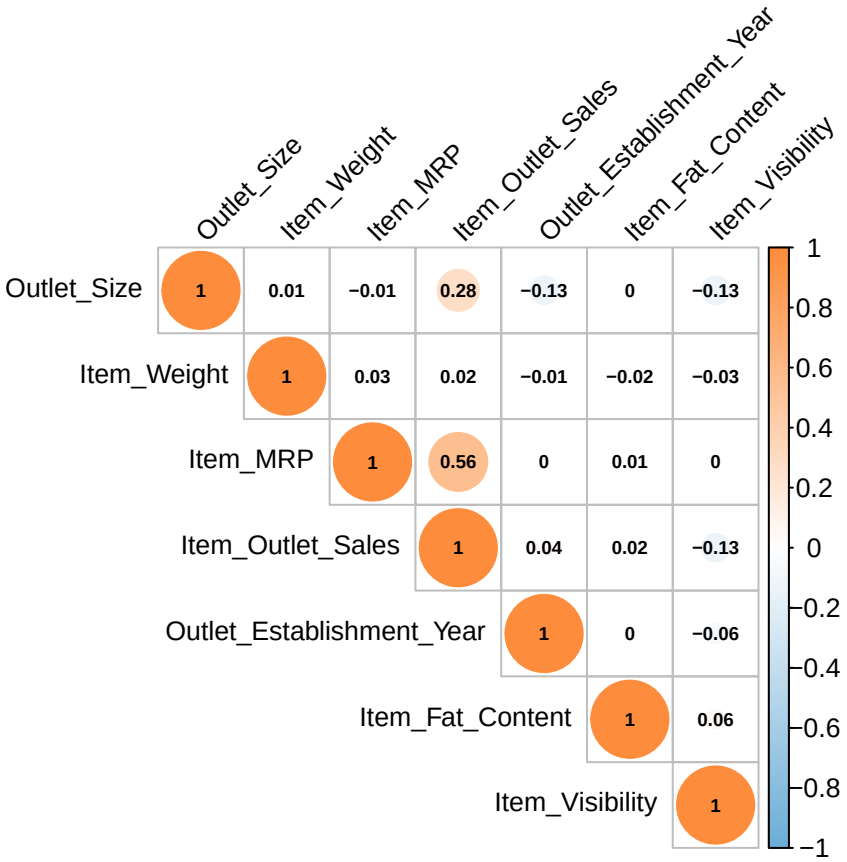
```
## [1] "Entirely duplicated rows:"
```

```
print(entire_row_duplicate_entries)
```

```
##  [1] Item_Identifier           Item_Weight
##  [3] Item_Fat_Content          Item_Visibility
##  [5] Item_Type                 Item_MRP
##  [7] Outlet_Identifier         Outlet_Establishment_Year
##  [9] Outlet_Size               Outlet_Location_Type
## [11] Outlet_Type               Item_Outlet_Sales
## <0 rows> (or 0-length row.names)
```

Now we will start to find out some correlations between variables.

```
# Calculate Spearman's rank correlation matrix again if not already calculated
cor_matrix <- cor(df %>% select(where(is.numeric)),
                  method = "spearman",
                  use = "pairwise.complete.obs")

# Visualize the correlation matrix with coefficients inside the circles
corrplot(cor_matrix, method = "circle", type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45, tl.cex = 0.8, cl.cex = 0.8,
         col = colorRampPalette(c("#6BAED6", "#FFFFFF", "#FD8D3C"))(200),
         addCoef.col = "black",  # Sets color of the coefficients to black (choose based on your color
         number.cex = 0.6)  # Adjust coefficient text size appropriately
```

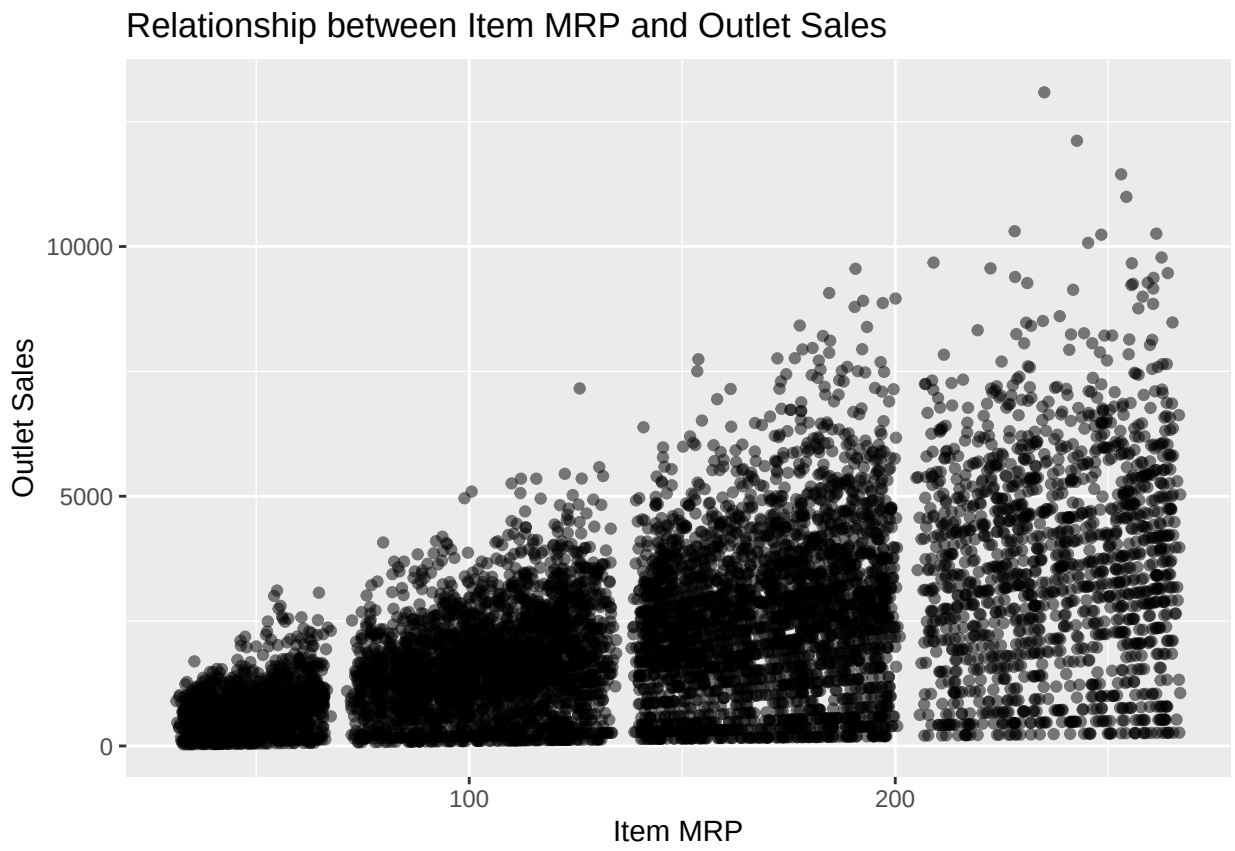


DOBBIAMO DROPPARE LE COLONNE REMINAINING VISIBILITY EZERO ITEM VISIBILITY COUNT

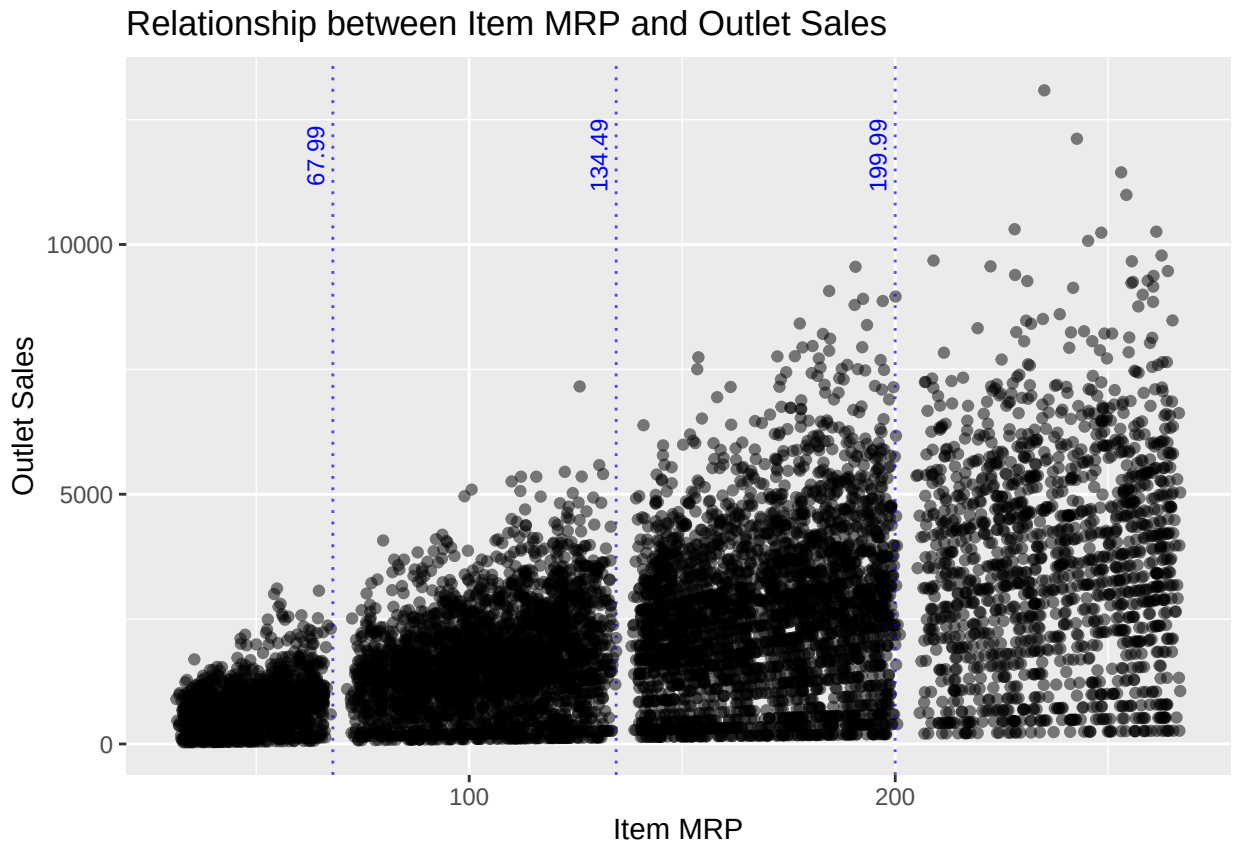
We can see that there are big positive relation between:

outlet_size-item visibility item_outlet_sales and outlet_size item_mrp and item_outlet_sales

```
library(ggplot2)
ggplot(df, aes(x = Item_MRP, y = Item_Outlet_Sales)) +
  geom_point(alpha = 0.5) +
  labs(title = "Relationship between Item MRP and Outlet Sales",
       x = "Item MRP", y = "Outlet Sales")
```




```r
# Load necessary libraries
library(ggplot2)

# Specific price points to add to the plot
specific_price_points <- c(67.99, 134.49, 199.99)

# Plot the relationship between Item_MRP and Item_Outlet_Sales
p <- ggplot(df, aes(x = Item_MRP, y = Item_Outlet_Sales)) +
  geom_point(alpha = 0.5) +
  labs(title = "Relationship between Item MRP and Outlet Sales",
       x = "Item MRP", y = "Outlet Sales")

# Add vertical lines for specific price points
for (price in specific_price_points) {
  p <- p + geom_vline(xintercept = price, linetype = "dotted", color = "blue", alpha = 0.7)
  p <- p + annotate("text", x = price, y = max(df$Item_Outlet_Sales) * 0.9, label = price, color = "blu
}

# Print the plot
print(p)
```

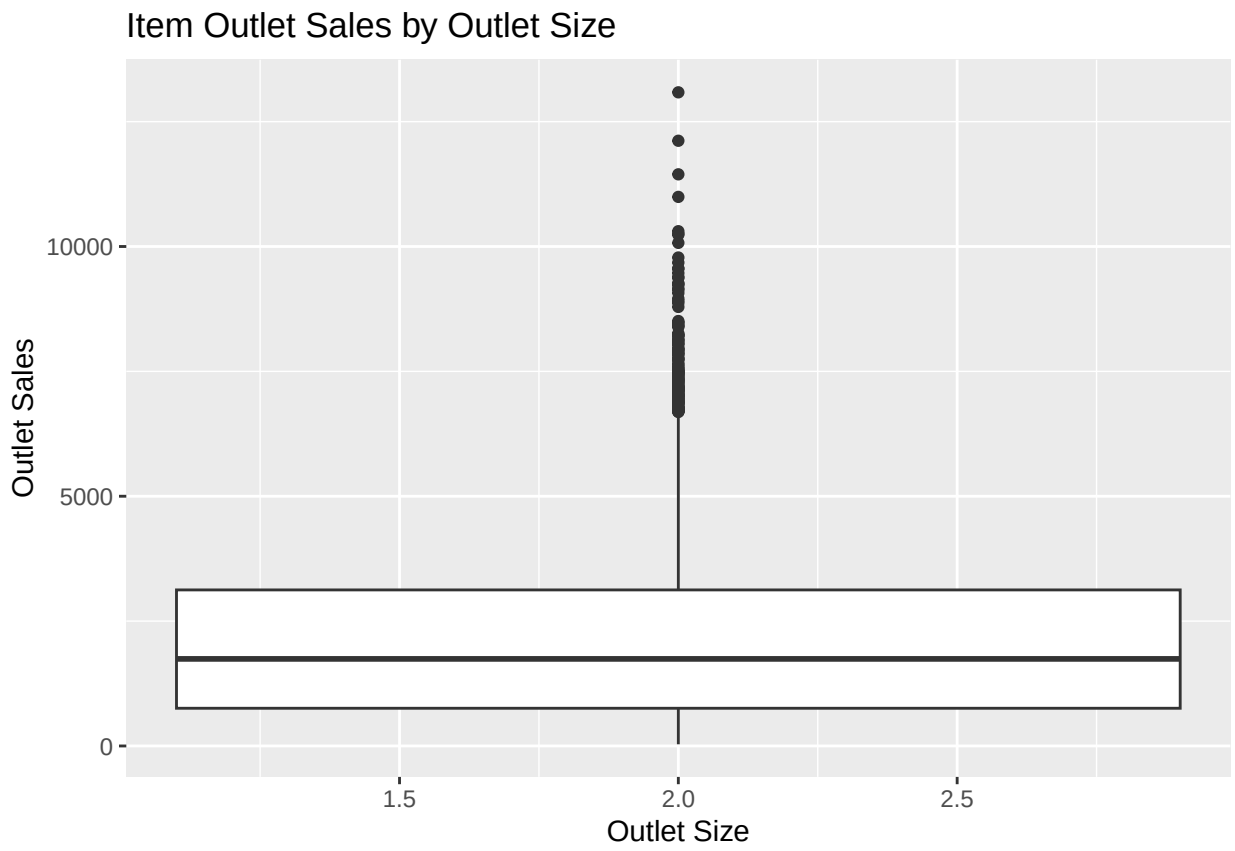



```
ggplot(df, aes(x = Outlet_Size, y = Item_Outlet_Sales)) +
  geom_boxplot(aes(fill = Outlet_Size)) +
  labs(title = "Item Outlet Sales by Outlet Size",
       x = "Outlet Size", y = "Outlet Sales")
```

```
## Warning: Continuous x aesthetic
## i did you forget `aes(group = ...)`?
```

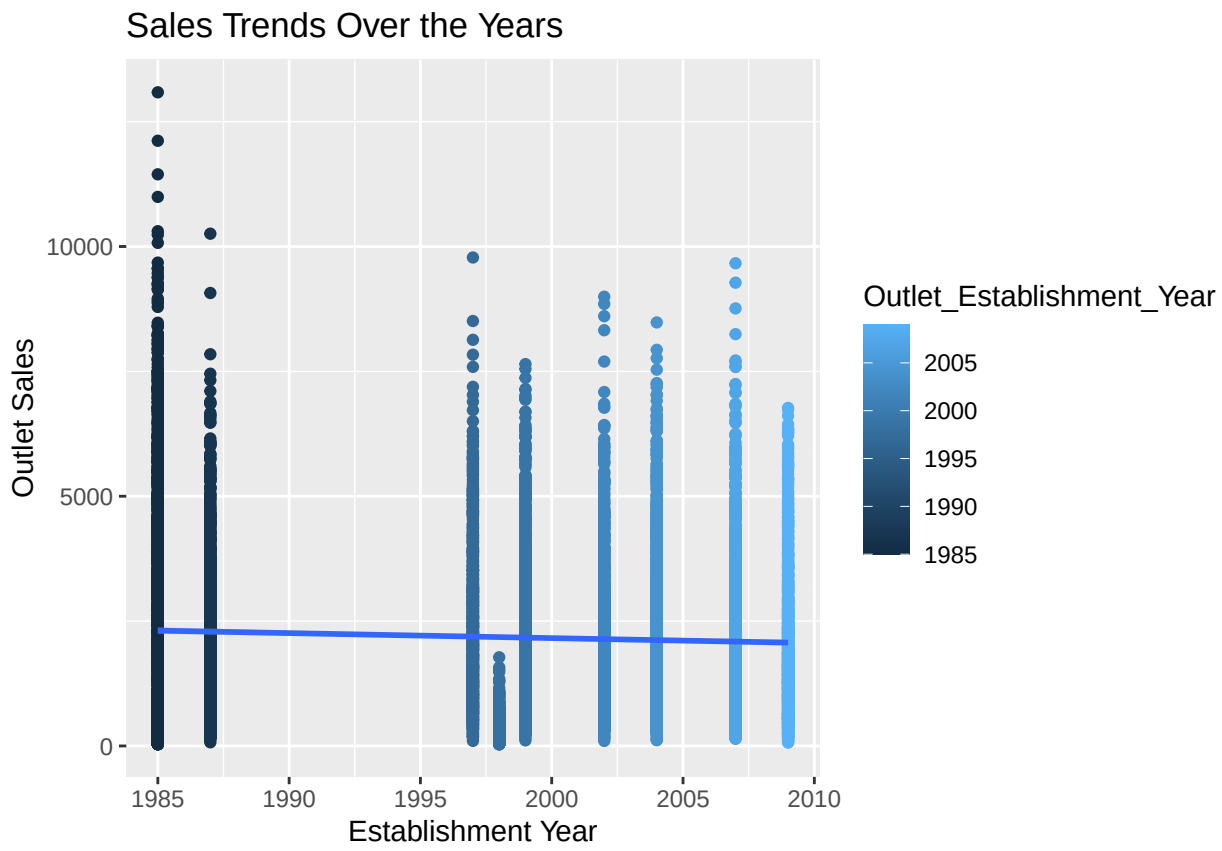
```
## Warning: Removed 1855 rows containing missing values or values outside the scale range
## (`stat_boxplot()`).
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill.
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

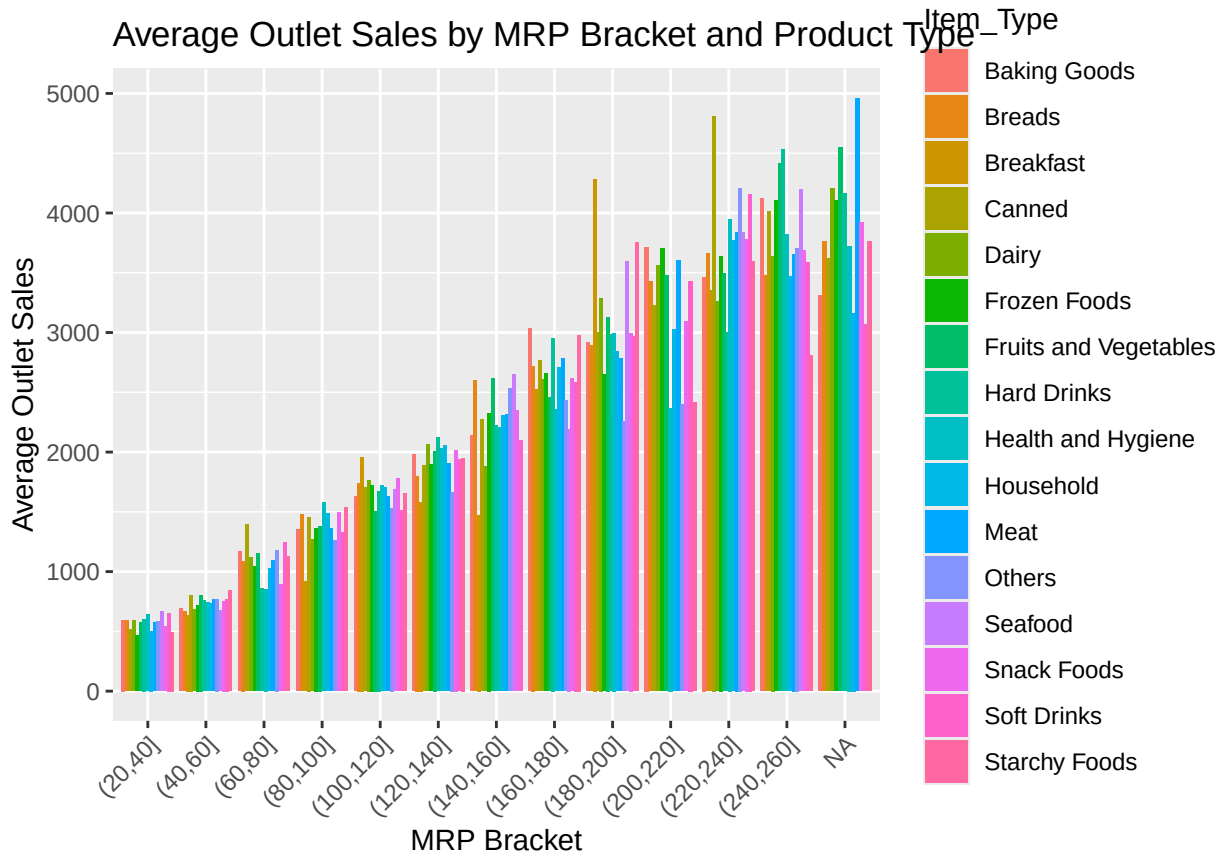



```
ggplot(df, aes(x = Outlet_Establishment_Year, y = Item_Outlet_Sales)) +
  geom_point(aes(color = Outlet_Establishment_Year)) +
  geom_smooth(method = "lm") +
  labs(title = "Sales Trends Over the Years",
       x = "Establishment Year", y = "Outlet Sales")
```

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



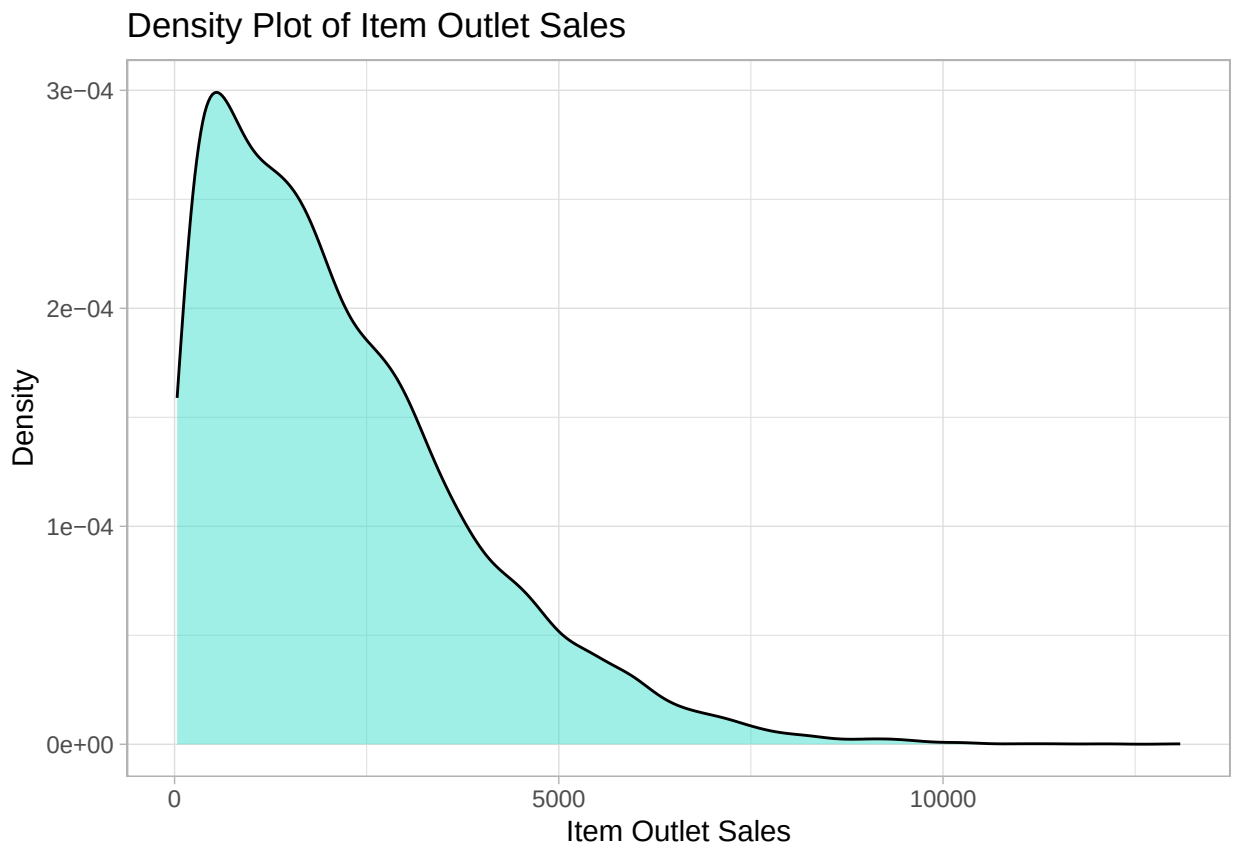

```
library(dplyr)

df_summary <- df %>%
  group_by(Item_Type, MRP_Bracket = cut(Item_MRP, breaks = seq(0, max(Item_MRP), by = 20))) %>%
  summarize(Average_Sales = mean(Item_Outlet_Sales), .groups = 'drop')

ggplot(data = df_summary, aes(x = MRP_Bracket, y = Average_Sales, fill = Item_Type)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  labs(title = "Average Outlet Sales by MRP Bracket and Product Type",
       x = "MRP Bracket",
       y = "Average Outlet Sales") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```
#ciaociao
#namoooo
#prova 3
```

zi qua c'è scritto che più vendi e più alzi il prezzo di base

```
# Create a density plot
ggplot(data = df, aes(x = Item_Outlet_Sales)) +
  geom_density(fill = "turquoise", alpha = 0.5) +  # 'alpha' controls transparency
  labs(title = "Density Plot of Item Outlet Sales",
       x = "Item Outlet Sales",
       y = "Density") +
  theme_light()
```


Density Plot of Item Outlet Sales
3e−04
2e−04
1e−04
0e+00
Density
0
5000
10000
Item Outlet Sales

```
#ciao2
```