main

2024-06-10

Importing dataset and libraries

#EDA ##Data Cleaning

###Checking and Removing Duplicates

```
# 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"
```

As we can see there aren't any duplicates in our data set, so we can proceed to find which columns have missing values

###Fixing Features of the Dataset

```
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"
```

```
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"
```

If we go throughout the column `Item_Fat_Content` we can cearly see that there are some inconsistencies, in fact 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 edited 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 to chek if the process ended successfully

###Checking for Missing Values

```r
# Crea una copia del dataframe convertendo tutte le colonne in stringhe
df_copy <- lapply(df, as.character)

# Creiamo una funzione per contare "NA", "" o "0" in ogni colonna della copia
count_na_empty_or_zero_strings <- function(column) {
  # Sostituiamo i veri NA con stringhe vuote per una conta coerente
  column[is.na(column)] <- ""
  sum(column == "" | column == "0", na.rm = TRUE)
}

# Applicare la funzione a tutte le colonne della copia e stampare i risultati
na_empty_or_zero_counts <- sapply(df_copy, count_na_empty_or_zero_strings)

# Stampa il numero di "NA", "" o "0" per ogni colonna della copia
print(na_empty_or_zero_counts)
```

```
##           Item_Identifier                Item_Weight          Item_Fat_Content
##                         0                       1463                         0
##           Item_Visibility                 Item_Type                  Item_MRP
##                       526                         0                         0
##         Outlet_Identifier Outlet_Establishment_Year               Outlet_Size
##                         0                         0                      2410
##      Outlet_Location_Type               Outlet_Type         Item_Outlet_Sales
##                         0                         0                         0
```

We see that there are 3 columns with missing values: -Item_Weight -Item_Visibility -Outlet_Size

```r
#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"
```

```r
#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 %"
```

```r
# 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 %"
```

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

To address the significant number of missing values were noted in the `Outlet_Size` column, we first calculated the percentage of missing entries to understand the scope of the issue.

Subsequently, we explored potential relationships between `Outlet_Type` and `Outlet_Size` by creating a cross-tabulation of these variables. This analysis revealed distinct patterns: all entries categorized as "Grocery Store" were consistently labeled as "Small", while "Supermarket Type2" and "Supermarket Type3" were uniformly classified as "Medium".

Considering this insight, we proceeded to change missing `Outlet_Size` values based on `Outlet_Type`:

For "Grocery Store", missing sizes were filled with "Small". For "Supermarket Type2" and "Supermarket Type3", missing sizes were filled with "Medium". These imputations reduced the percentage of missing values from 28% to 21%, decreasing the total number of missing entries from 2410 to 1855.

Regarding "Supermarket Type1" we do not have enough informations or clear patterns, so we had to fill the remaining missing entries with a placeholder value of "NA", ensuring no data point within "Outlet_Size" remained unaddressed.

```
# 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
```

We converted as factor the columns `Item_Fat_Content` and we also want to convert the column `Outlet_Size` giving respectively: 0 -> "Low Fat" 1 -> "Regular"

1 -> Small 2 -> Medium 3 -> Large

```r
# Calcolare la media del peso per ogni categoria di prodotto
average_weight_per_type <- aggregate(Item_Weight ~ Item_Type, data = df, mean, na.rm = TRUE)

# Funzione per riempire i pesi mancanti
fill_missing_weights <- function(item_id, item_type) {
  # Controlla se esiste un valore non NA per lo stesso Item_Identifier
  if (any(!is.na(df$Item_Weight[df$Item_Identifier == item_id]))) {
    return(df$Item_Weight[df$Item_Identifier == item_id & !is.na(df$Item_Weight)][1])
  } else {
    # Altrimenti usa la media del Item_Type corrispondente
    return(average_weight_per_type$Item_Weight[average_weight_per_type$Item_Type == item_type])
  }
}

# Applicare la funzione ai valori NA in Item_Weight
df$Item_Weight[is.na(df$Item_Weight)] <- mapply(fill_missing_weights, df$Item_Identifier[is.na(df$Item_

# Visualizzare il dataframe aggiornato
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
```

Here we handled missing values for `Item_Weight`. The first idea was to use the general mean of `Item_Weight`, but since we have also the category of each product, we can compute the mean of every product category. At the end, we checked for every record with missing `Item_Weight` and we will do a double check: 1. If there were another item with the same id and missing weight, they will have for sure the same weight. 2. We will put in item_weight, the mean of the weights of the products of the same category.

```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"
```

```r
# 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
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
```

```r
# 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

# 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")]
```

```r
# 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       5.985                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)
```

Finally we fixed the missing values for `Item_Visibility`. To do it we firstly count the number of missing values for each of the 10 shops and their visibility to understand how much visibility we are missing from each store. Our first idea was to subtract each visibility percentage from 100% and then equally divide the remaining visibility among the number of missing visibility product. In doing that we obtained biased and not consistent data because:

1. The filled visibility in each product was too high compared to the non-zeros values
2. After adding again all the visibility the result was 100% for each shop, but could also be true that in some shops some products are out of stock so the final visibility could be less than 100%

For this reason we decided to change our strategy and to fill the missing values we firstly find the missing `Item_Visibility`, then through the `Item_ID` we find all the shops that have for that product a real value, then we check the `Outlet_Size` of each shop and compare to the one of the market with missing value if it's the same we compute the mean throu all the visibility that match these criteria.

We did this because for example if the missing visibility is in a Medium shop, we cannot consider the same product visibility in a Small shop, because it's obviously higher, so we take into accounts a visibility to compute the mean only if the `Outlet_Size` is the same.

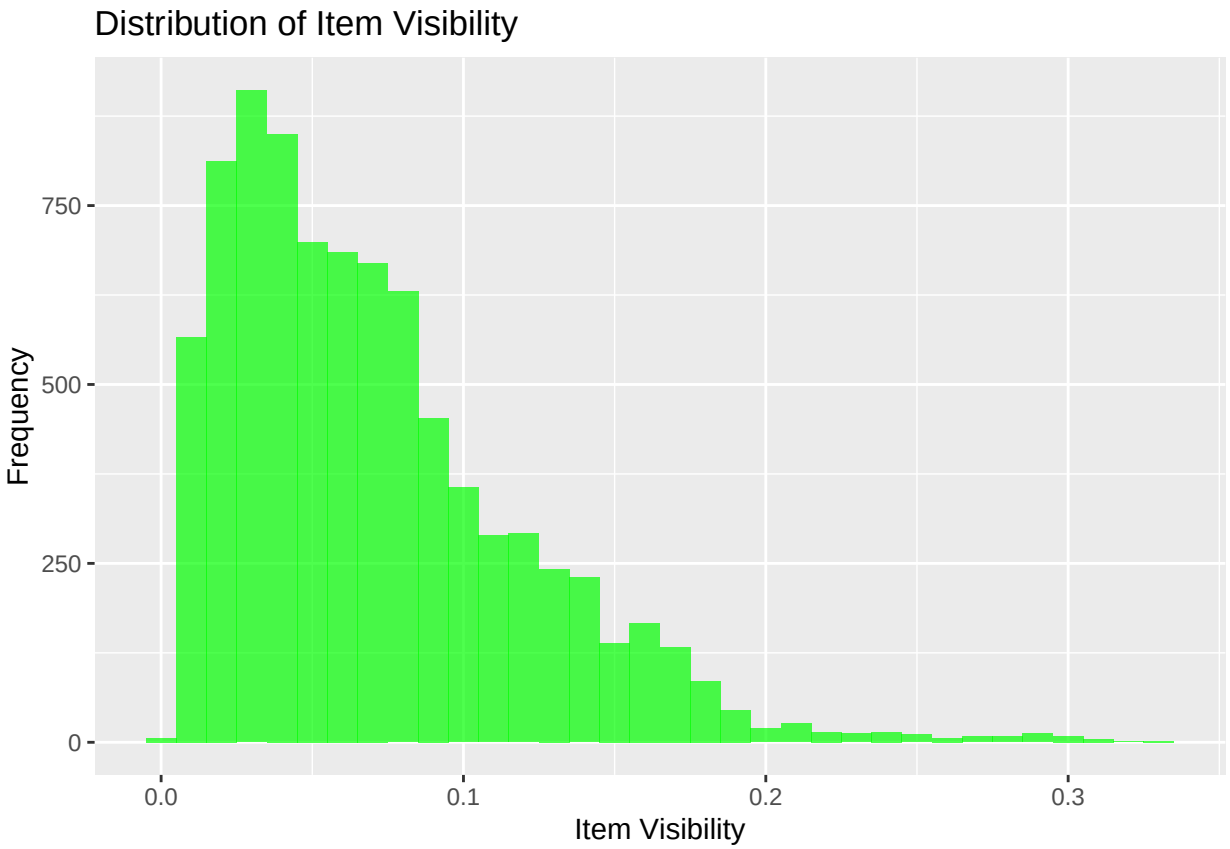
```r
#finding out all the uniques


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

print(item_fat_content_uniques)
```

```
##  [1] "Baking Goods"          "Breads"                "Breakfast"
##  [4] "Canned"                "Dairy"                 "Frozen Foods"
##  [7] "Fruits and Vegetables" "Hard Drinks"           "Health and Hygiene"
## [10] "Household"             "Meat"                  "Others"
## [13] "Seafood"               "Snack Foods"           "Soft Drinks"
## [16] "Starchy Foods"
```

```r
df <- df %>%
  mutate(Item_Type_Reduced = case_when(
    Item_Type %in% c("Baking Goods", "Breads", "Breakfast", "Dairy", "Meat", "Seafood") ~ "Food Basics",
    Item_Type %in% c("Canned", "Frozen Foods", "Snack Foods", "Starchy Foods") ~ "Processed Foods",
    Item_Type %in% c("Hard Drinks", "Soft Drinks") ~ "Beverages",
    Item_Type %in% c("Health and Hygiene", "Household") ~ "Non-Food Items",
    Item_Type == "Fruits and Vegetables" ~ "Produce",
    Item_Type == "Others" ~ "Others",
    TRUE ~ "Others"  # Catch-all for any undefined categories
  ))

#rename the column item_type_reduced into item_type

#drop old item_Type column

df <- df %>%
  select(-Item_Type)


# Renaming the column using dplyr
df <- df %>%
  rename(Item_Type = Item_Type_Reduced)
```

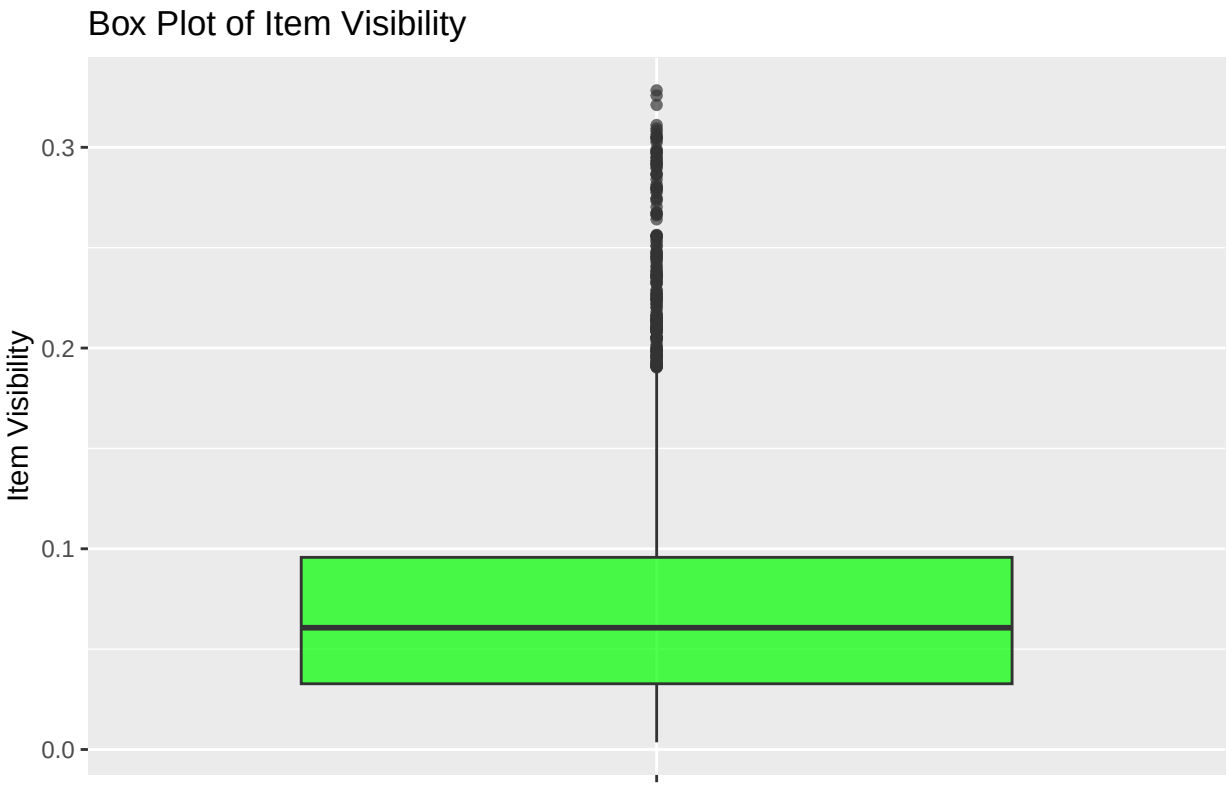
```r
# Histogram and Box Plot for Item_Visibility
p3 <- ggplot(df, aes(x = Item_Visibility)) +
  geom_histogram(binwidth = 0.01, fill = 'green', alpha = 0.7) +
  ggtitle("Distribution of Item Visibility") +
  xlab("Item Visibility") +
  ylab("Frequency")

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()`).
```



Finally we fixed the missing values for `Item_Visibility`. To do it we firstly count the number of missing values for each of the 10 shops and their visibility to understand how much visibility we are missing from each store. Our first idea was to subtract each visibility percentage from 100% and then equally divide the remaining visibility among the number of missing visibility product. In doing that we obtained biased and not consistent data because:

1. The filled visibility in each product was too high compared to the non-zeros values
2. After adding again all the visibility the result was 100% for each shop, but could also be true that in some shops some products are out of stock so the final visibility could be less than 100%

For this reason we decided to change our strategy and to fill the missing values we firstly find the missing `Item_Visibility`, then through the `Item_ID` we find all the shops that have for that product a real value, then we check the `Outlet_Size` of each shop and compare to the one of the market with missing value if it's the same we compute the mean throu all the visibility that match these criteria.

We did this because for example if the missing visibility is in a Medium shop, we cannot consider the same product visibility in a Small shop, because it's obviously higher, so we take into accounts a visibility to compute the mean only if the `Outlet_Size` is the same.

```
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)
```

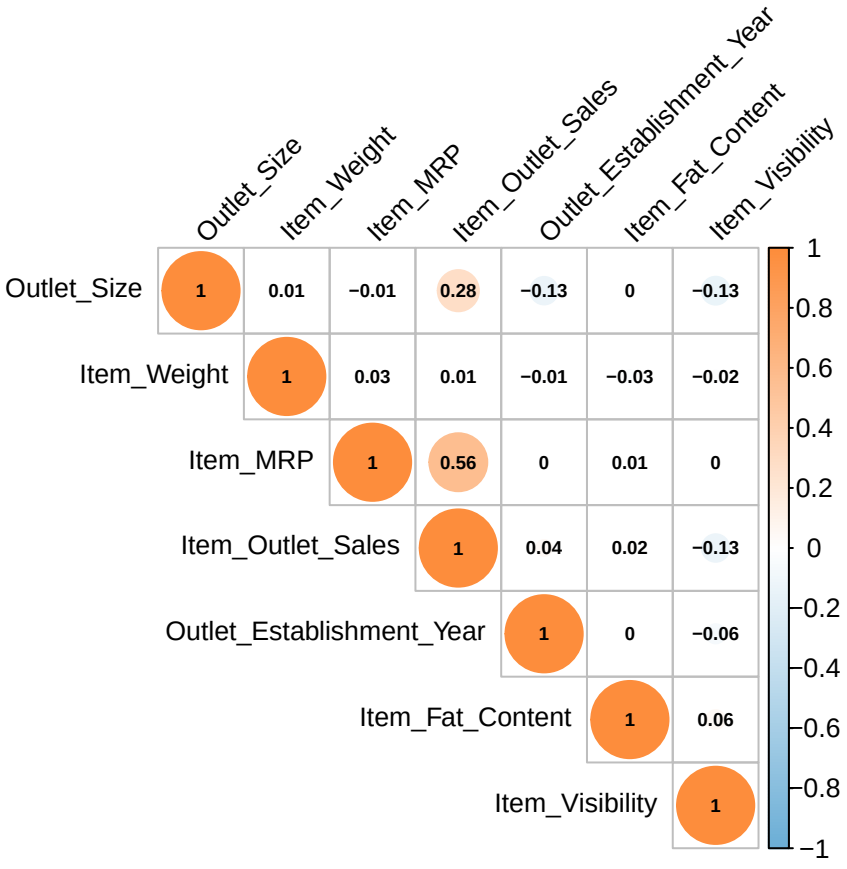
```
##    Outlet_Identifier Total_Item_Visibility
```

```
## 1              OUT010              58.63381
## 2              OUT013              59.62270
## 3              OUT017              56.83465
## 4              OUT018              60.79797
## 5              OUT019              59.73351
## 6              OUT027              58.64375
## 7              OUT035              61.42696
## 8              OUT045              56.18078
## 9              OUT046              61.26425
## 10             OUT049              59.94402
```

##Multivariate Analysis

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
```

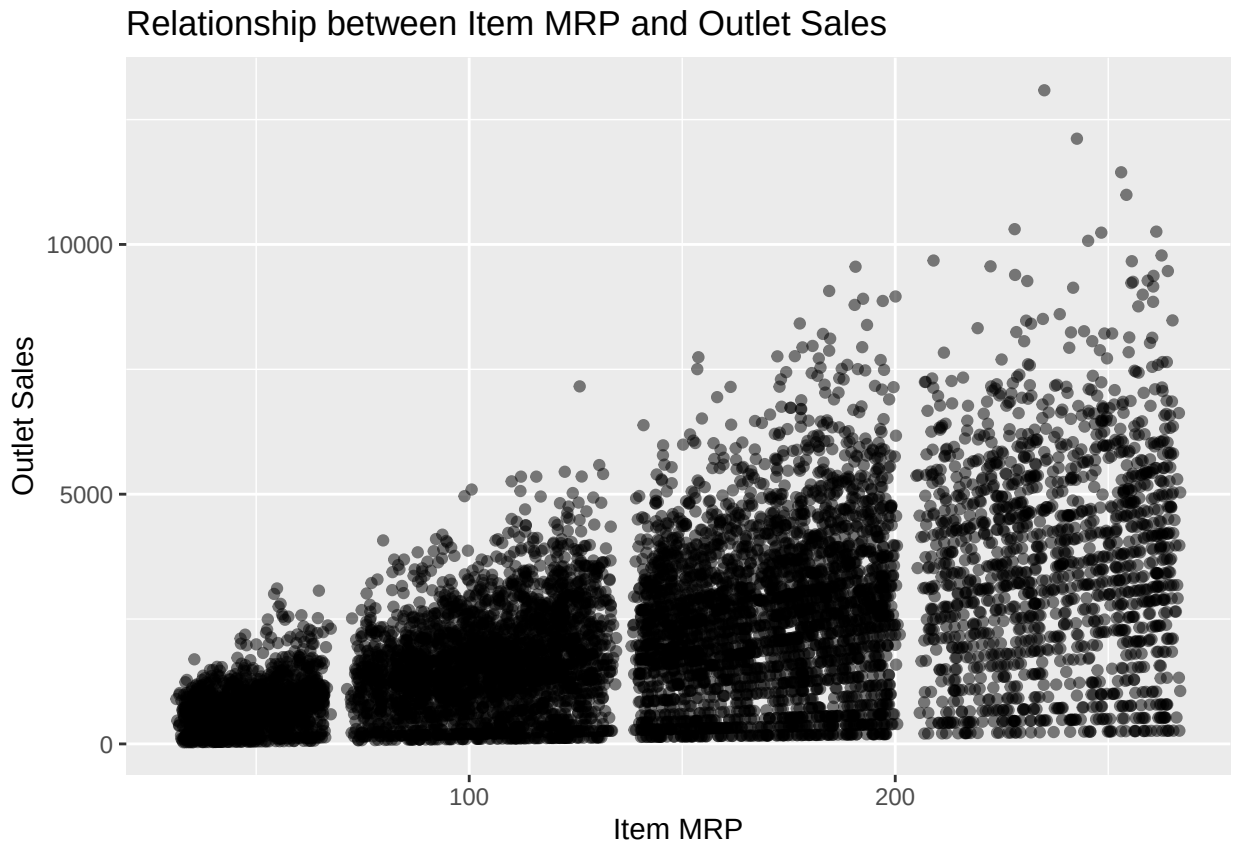


DOBBIAMO DROPPARE LE COLONNE REMINAINING VISIBILITY EZERO ITEM VISIBILITY COUNT

We can see that there are big positive relation between:

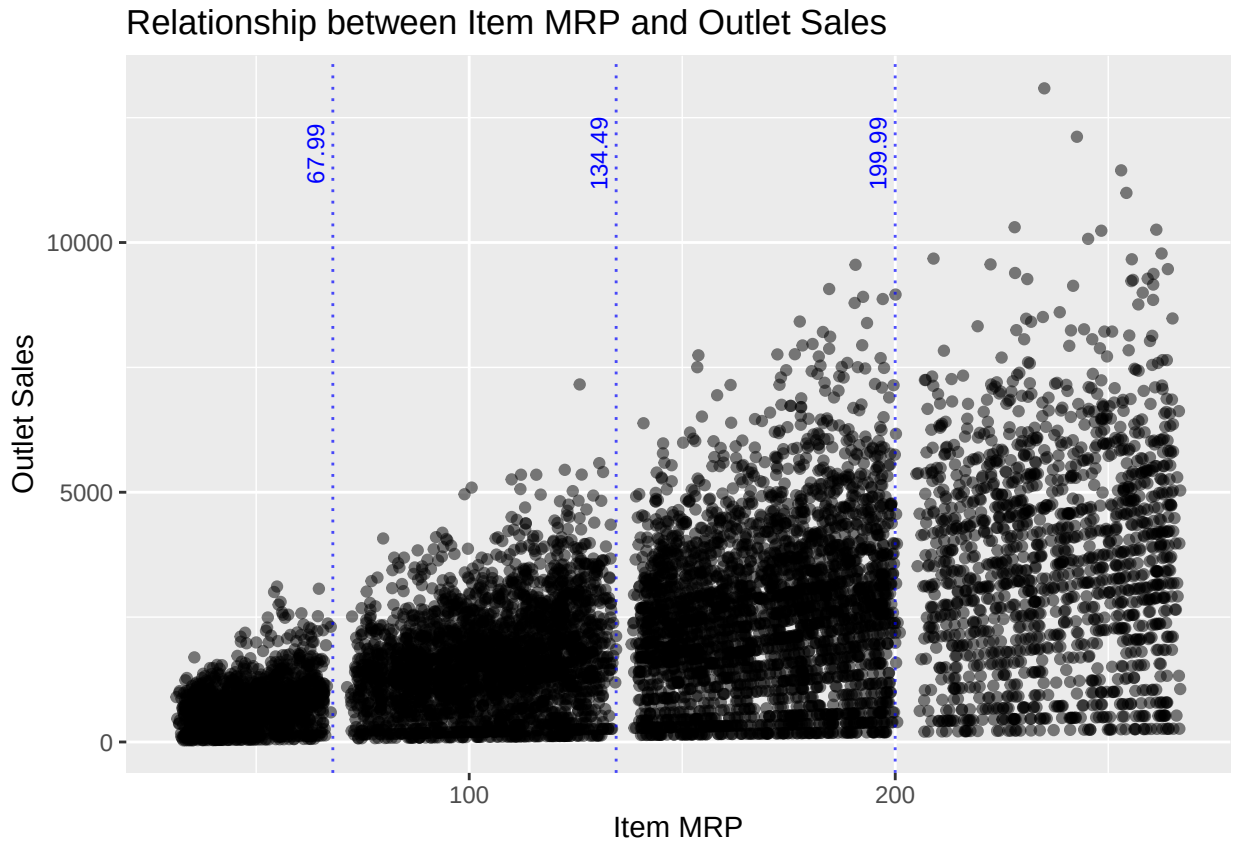
outlet_size-item visibility item_outlet_sales and outlet_size item_mrp and item_outlet_sales

```
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")
```




```
# 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)
```

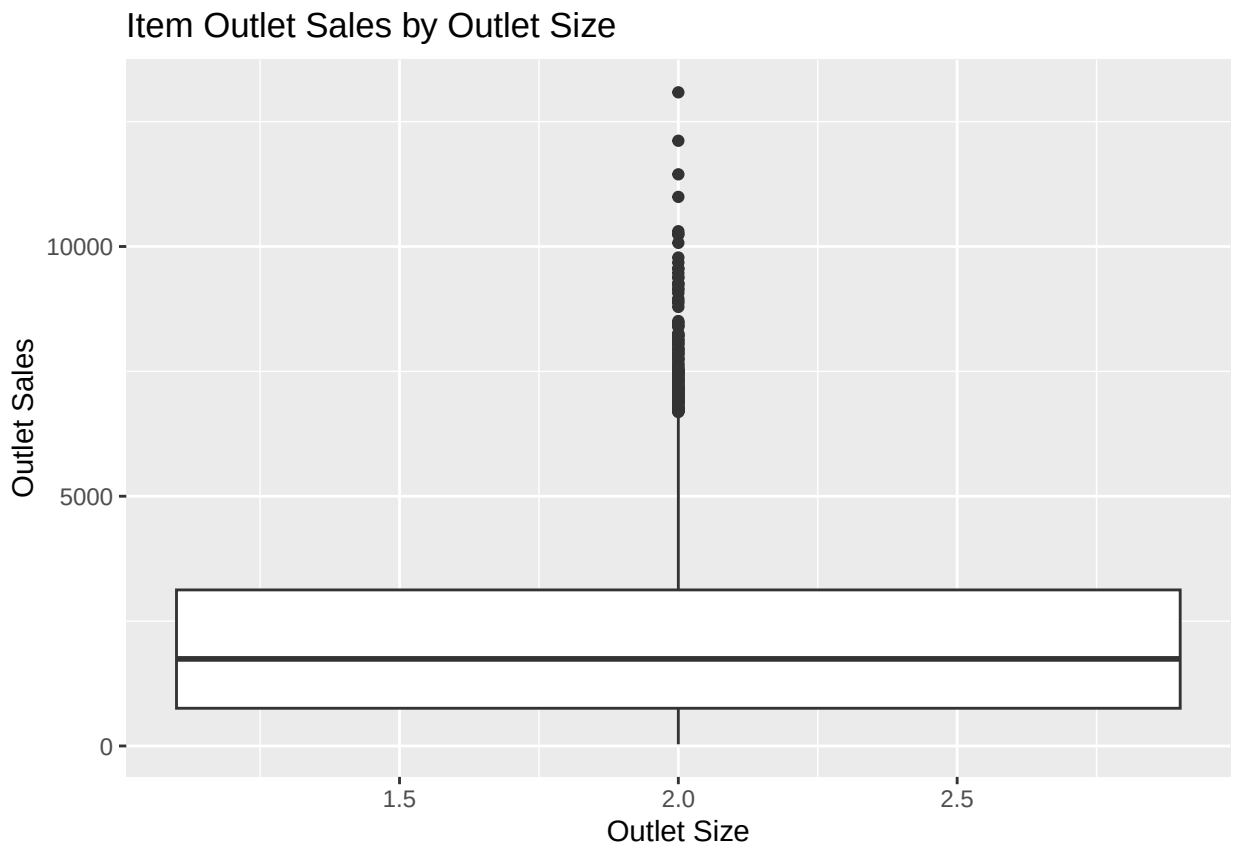



```
ggplot(df, aes(x = Outlet_Size, y = Item_Outlet_Sales)) +
  geom_boxplot(aes(fill = Outlet_Size)) +
  labs(title = "Item Outlet Sales by Outlet Size",
       x = "Outlet Size", y = "Outlet Sales")
```

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

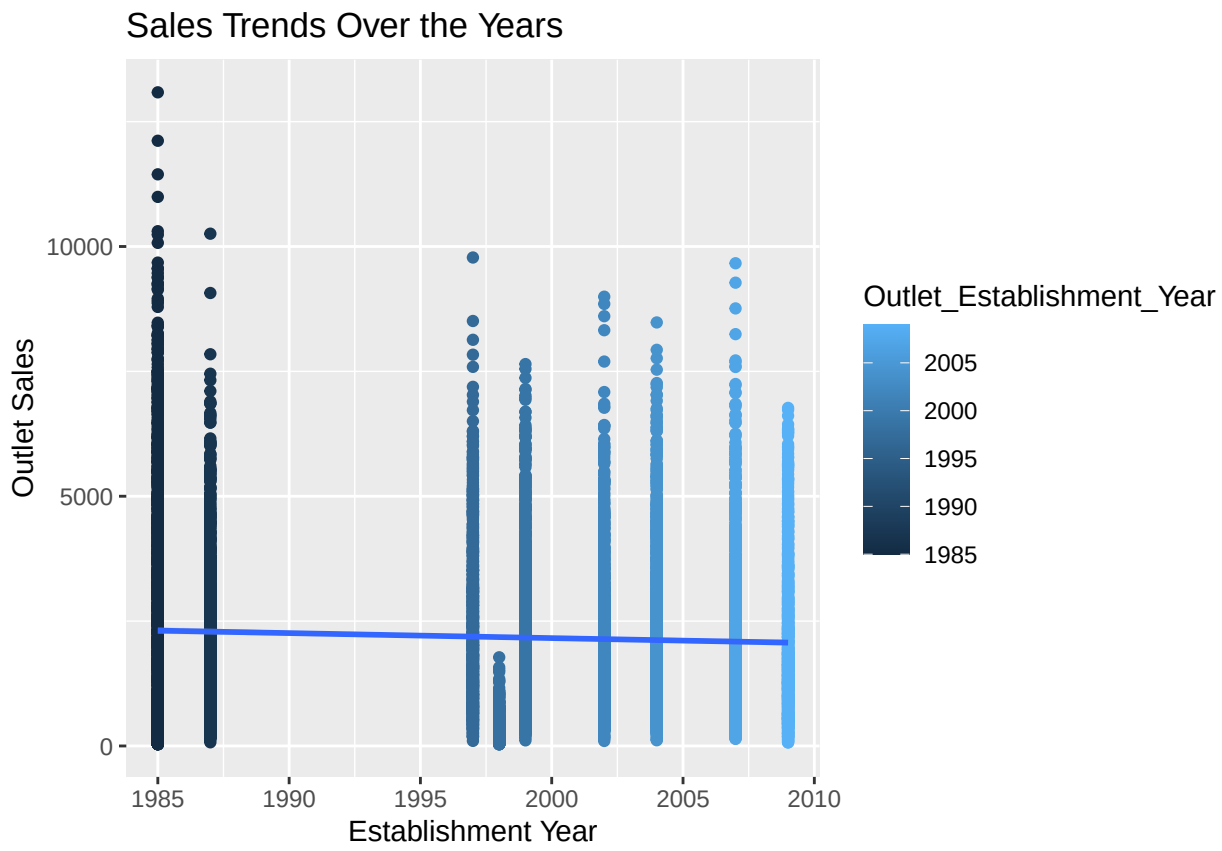
```
## Warning: Removed 1855 rows containing missing values or values outside the scale range
## (`stat_boxplot()`).
```

```
## 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?
```

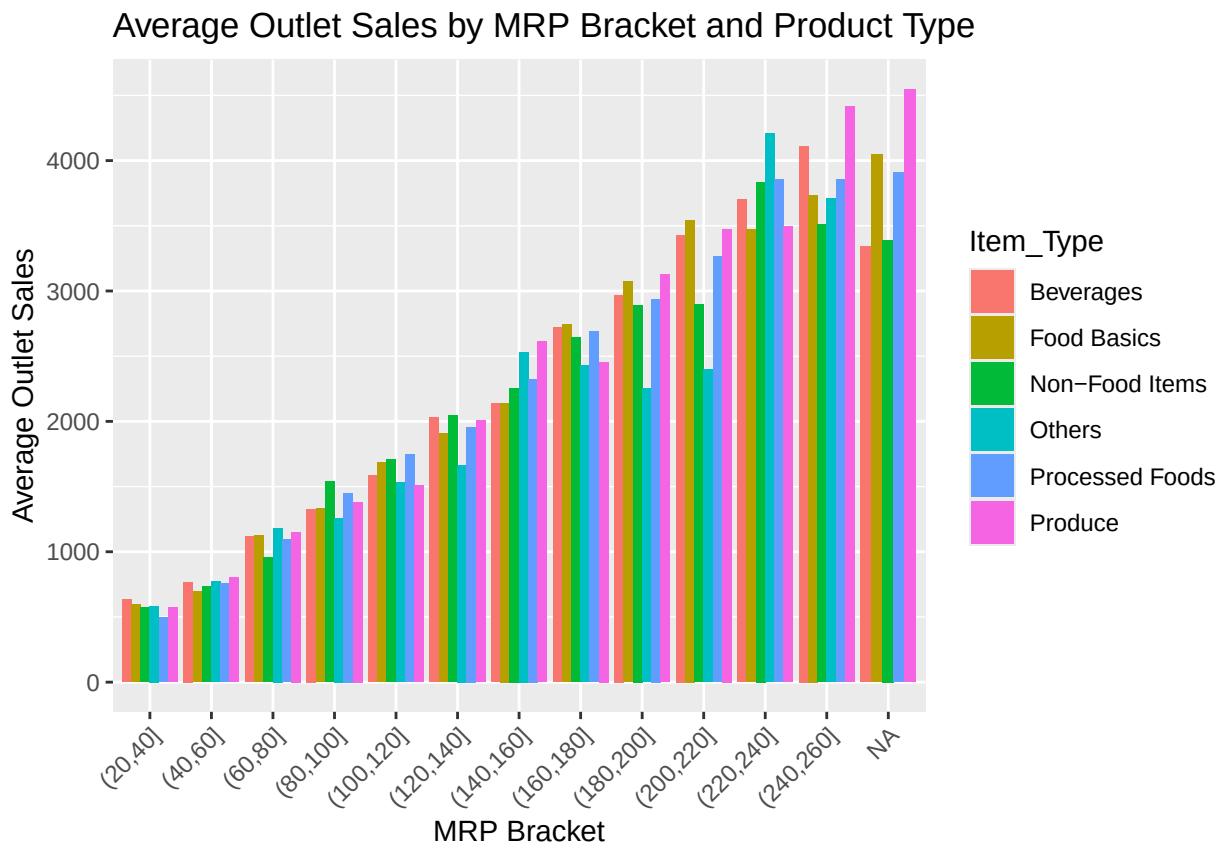



```
ggplot(df, aes(x = Outlet_Establishment_Year, y = Item_Outlet_Sales)) +
  geom_point(aes(color = Outlet_Establishment_Year)) +
  geom_smooth(method = "lm") +
  labs(title = "Sales Trends Over the Years",
       x = "Establishment Year", y = "Outlet Sales")
```

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



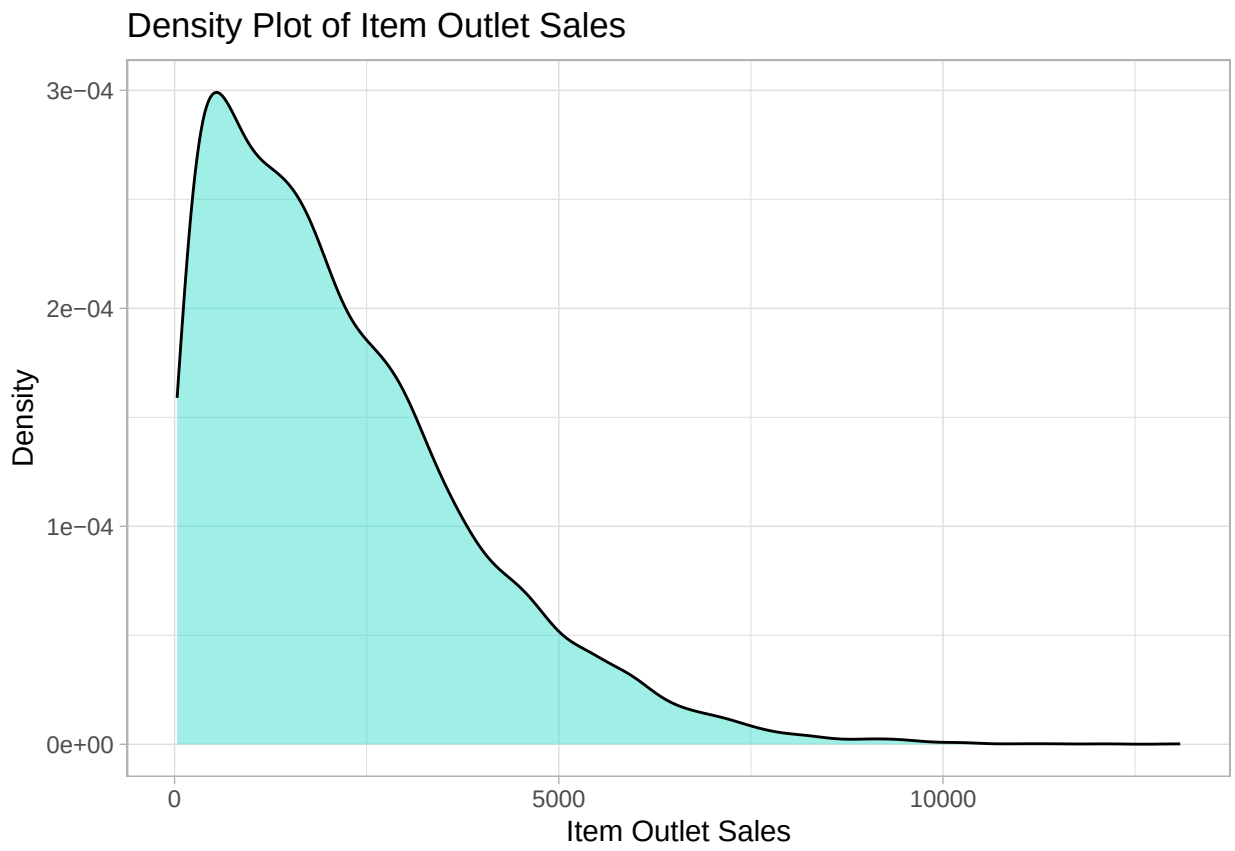

```
library(dplyr)

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()
```




```
#ciao2
```

```
# Ensure 'Item_Visibility' and 'Outlet_Sales' are treated correctly
df$Item_Visibility <- as.numeric(as.character(df$Item_Visibility))
df$Item_Outlet_Sales <- as.numeric(as.character(df$Item_Outlet_Sales))

# Convert categorical variables to factors
df$Item_Type <- as.factor(df$Item_Type)
df$Outlet_Type <- as.factor(df$Outlet_Type)

# Segmented Regression by Item Type
item_type_models <- df %>%
  group_by(Item_Type) %>%
  do(model = lm(Item_Outlet_Sales ~ Item_Visibility, data = .))

# Viewing summaries for each Item Type
item_type_summaries <- lapply(item_type_models$model, summary)

# Print the summaries for review
print(item_type_summaries)
```

```
## [[1]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
```

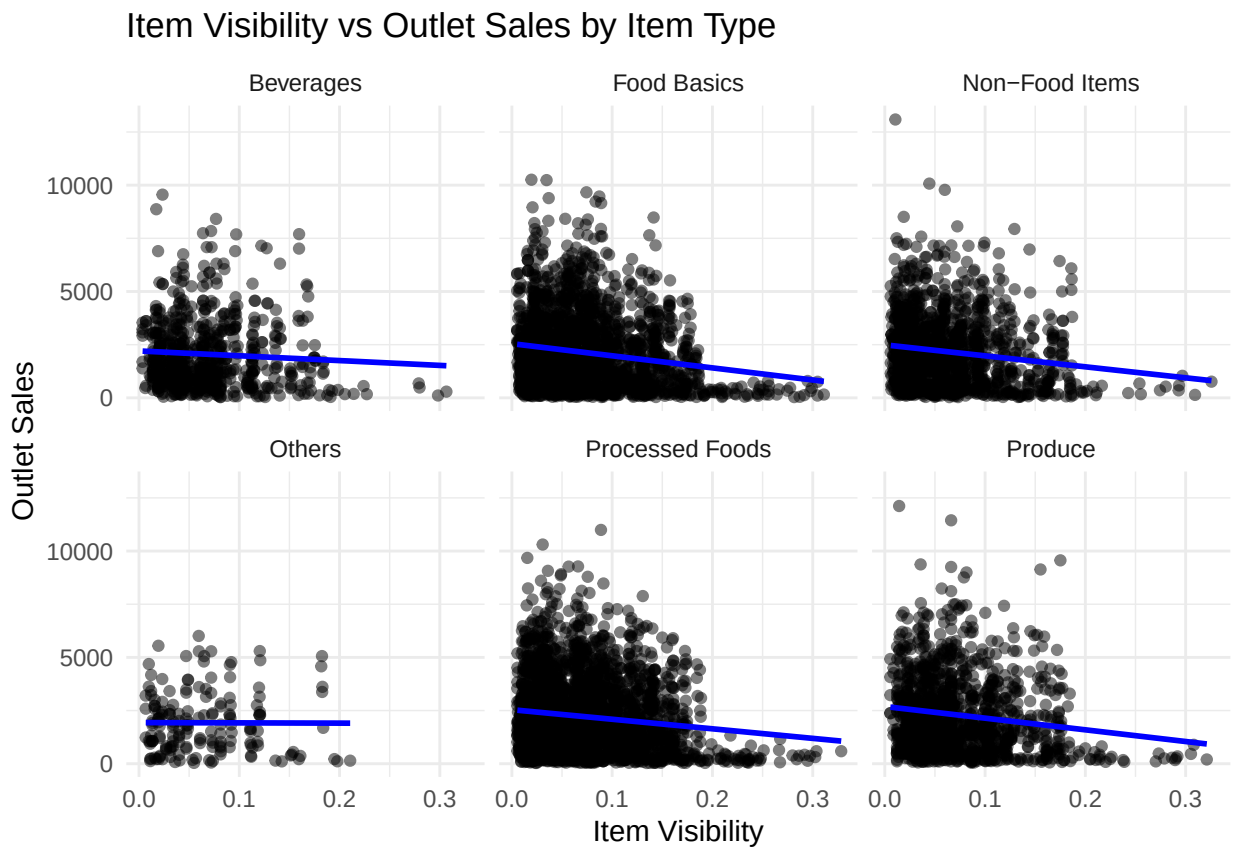
```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2097.3 -1246.0  -453.1   879.8  7402.9
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2204.5      115.2  19.133   <2e-16 ***
## Item_Visibility  -2264.4     1360.1  -1.665   0.0964 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1658 on 642 degrees of freedom
##   (15 observations deleted due to missingness)
## Multiple R-squared:  0.004299,   Adjusted R-squared:  0.002748
## F-statistic: 2.772 on 1 and 642 DF,  p-value: 0.09642
##
##
## [[2]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2395.6 -1287.0  -452.1   891.2  7886.7
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2546.84      65.35  38.973  < 2e-16 ***
## Item_Visibility -5693.86     731.38  -7.785 1.07e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1706 on 2147 degrees of freedom
##   (31 observations deleted due to missingness)
## Multiple R-squared:  0.02745,    Adjusted R-squared:  0.027
## F-statistic: 60.61 on 1 and 2147 DF,  p-value: 1.075e-14
##
##
## [[3]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2295.9 -1278.5  -304.9   899.3 10649.6
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2491.9       72.0  34.611  < 2e-16 ***
## Item_Visibility  -5167.1      913.9  -5.654  1.9e-08 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1633 on 1408 degrees of freedom
##   (20 observations deleted due to missingness)
## Multiple R-squared:  0.0222, Adjusted R-squared:  0.0215
## F-statistic: 31.96 on 1 and 1408 DF,  p-value: 1.899e-08
##
##
## [[4]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1886.8 -1135.1  -219.4   798.5  4082.7
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1933.7      186.7  10.357   <2e-16 ***
## Item_Visibility   -127.1     2334.9  -0.054    0.957
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1440 on 166 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  1.784e-05,  Adjusted R-squared:  -0.006006
## F-statistic: 0.002961 on 1 and 166 DF,  p-value: 0.9567
##
##
## [[5]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2404.2 -1293.2  -383.7   903.5  8852.6
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2540.31      56.18  45.217  < 2e-16 ***
## Item_Visibility -4493.95     652.36  -6.889 6.91e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1684 on 2818 degrees of freedom
##   (33 observations deleted due to missingness)
## Multiple R-squared:  0.01656,    Adjusted R-squared:  0.01621
## F-statistic: 47.46 on 1 and 2818 DF,  p-value: 6.913e-12
##
##
## [[6]]
##
```

```
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2512.9 -1341.4  -438.9   945.5  9504.5
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2691.51      91.64   29.37  < 2e-16 ***
## Item_Visibility -5488.80    1031.76   -5.32 1.24e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1782 on 1215 degrees of freedom
##   (15 observations deleted due to missingness)
## Multiple R-squared:  0.02276,    Adjusted R-squared:  0.02196
## F-statistic:  28.3 on 1 and 1215 DF,  p-value: 1.236e-07
```

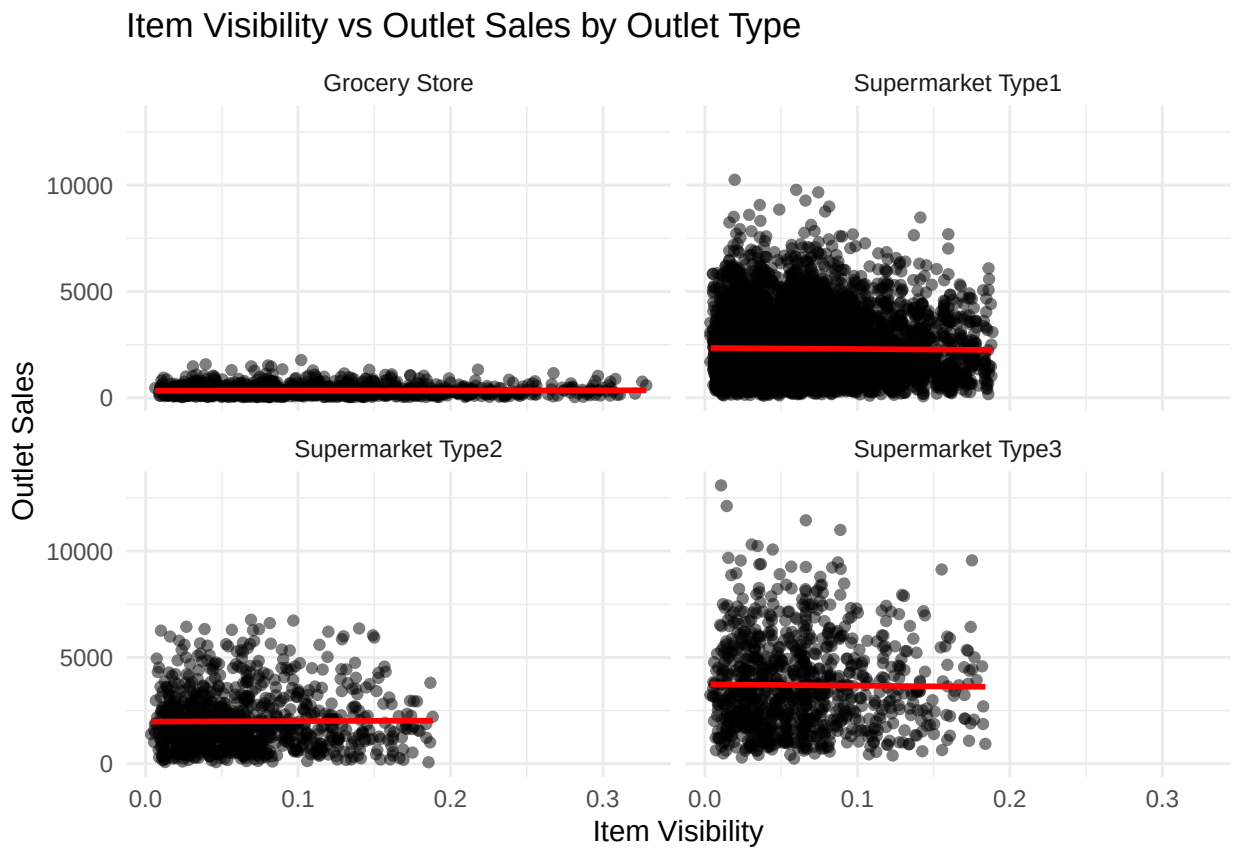
```
# Creating scatter plots segmented by Item_Type
p_item_type <- ggplot(df, aes(x = Item_Visibility, y = Item_Outlet_Sales)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", color = "blue", se = FALSE) +
  facet_wrap(~ Item_Type) +
  labs(title = "Item Visibility vs Outlet Sales by Item Type",
       x = "Item Visibility", y = "Outlet Sales") +
  theme_minimal()

# Creating scatter plots segmented by Outlet_Type
p_outlet_type <- ggplot(df, aes(x = Item_Visibility, y = Item_Outlet_Sales)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", color = "red", se = FALSE) +
  facet_wrap(~ Outlet_Type) +
  labs(title = "Item Visibility vs Outlet Sales by Outlet Type",
       x = "Item Visibility", y = "Outlet Sales") +
  theme_minimal()

# Print the plots
print(p_item_type)
```

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

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

## Warning: Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
print(p_outlet_type)
```

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

## Warning: Removed 115 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```r
# Fit a linear model
model <- lm(Item_Outlet_Sales ~ Item_Visibility + Item_Type + Outlet_Type + Item_Fat_Content + Item_MRP

# Summary of the model to understand influences
summary(model)
```

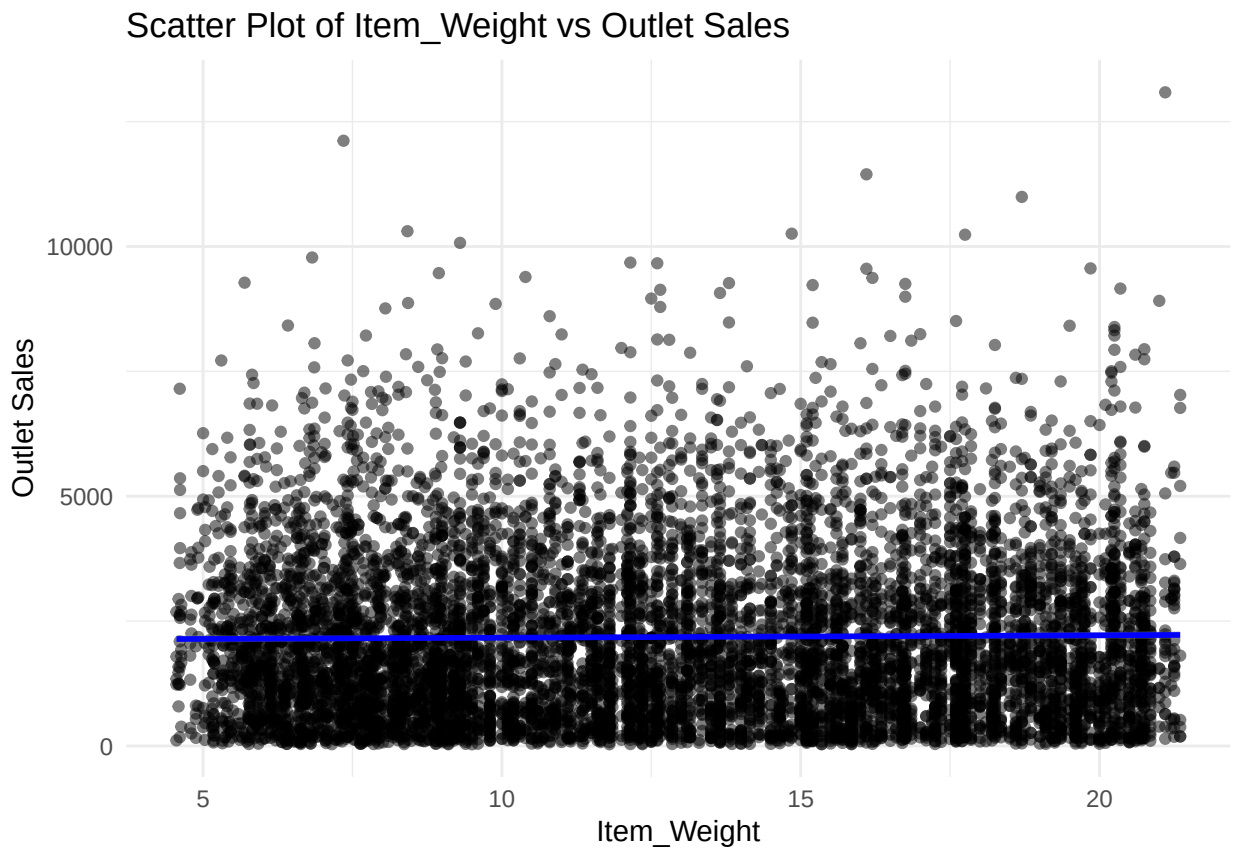
```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility + Item_Type +
##     Outlet_Type + Item_Fat_Content + Item_MRP, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4301.7  -673.2   -87.0   568.2  7938.0
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   -1868.8702    73.8755 -25.298   <2e-16 ***
## Item_Visibility                -231.8978   263.9352  -0.879    0.380
## Item_TypeFood Basics             -1.3790    51.7702  -0.027    0.979
## Item_TypeNon-Food Items         -22.8586    53.8105  -0.425    0.671
## Item_TypeOthers                 -20.8778    97.8053  -0.213    0.831
## Item_TypeProcessed Foods         -3.6506    50.2542  -0.073    0.942
## Item_TypeProduce                 34.0321    56.0621   0.607    0.544
## Outlet_TypeSupermarket Type1   1948.9614    39.3191  49.568   <2e-16 ***
## Outlet_TypeSupermarket Type2   1623.4238    51.7824  31.351   <2e-16 ***
```

```
## Outlet_TypeSupermarket Type3  3350.4226    51.8547  64.612   <2e-16 ***
## Item_Fat_Content               40.7263    28.2093   1.444    0.149
## Item_MRP                       15.5408     0.1979  78.548   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1128 on 8396 degrees of freedom
##   (115 observations deleted due to missingness)
## Multiple R-squared:  0.5641, Adjusted R-squared:  0.5636
## F-statistic: 987.9 on 11 and 8396 DF,  p-value: < 2.2e-16
```

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

# Convert categorical variables to factor if not already
df$Item_Type <- as.factor(df$Item_Type)
df$Outlet_Type <- as.factor(df$Outlet_Type)
df$Item_Fat_Content <- as.factor(df$Item_Fat_Content)
df$Outlet_Size <- as.factor(df$Outlet_Size)
df$Outlet_Location_Type <- as.factor(df$Outlet_Location_Type)

# Continuous Variables
continuous_vars <- c("Item_Weight", "Item_Visibility", "Item_MRP")

# Categorical Variables
categorical_vars <- c("Item_Type", "Outlet_Type", "Item_Fat_Content", "Outlet_Size", "Outlet_Location_Ty

# Plotting Distribution of Sales for Continuous Variables
for(var in continuous_vars) {
  p <- ggplot(df, aes_string(x = var, y = "Item_Outlet_Sales")) +
    geom_point(alpha = 0.5) +
    geom_smooth(method = "lm", color = "blue") +
    labs(title = paste("Scatter Plot of", var, "vs Outlet Sales"),
         x = var, y = "Outlet Sales") +
    theme_minimal()
  print(p)
}
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

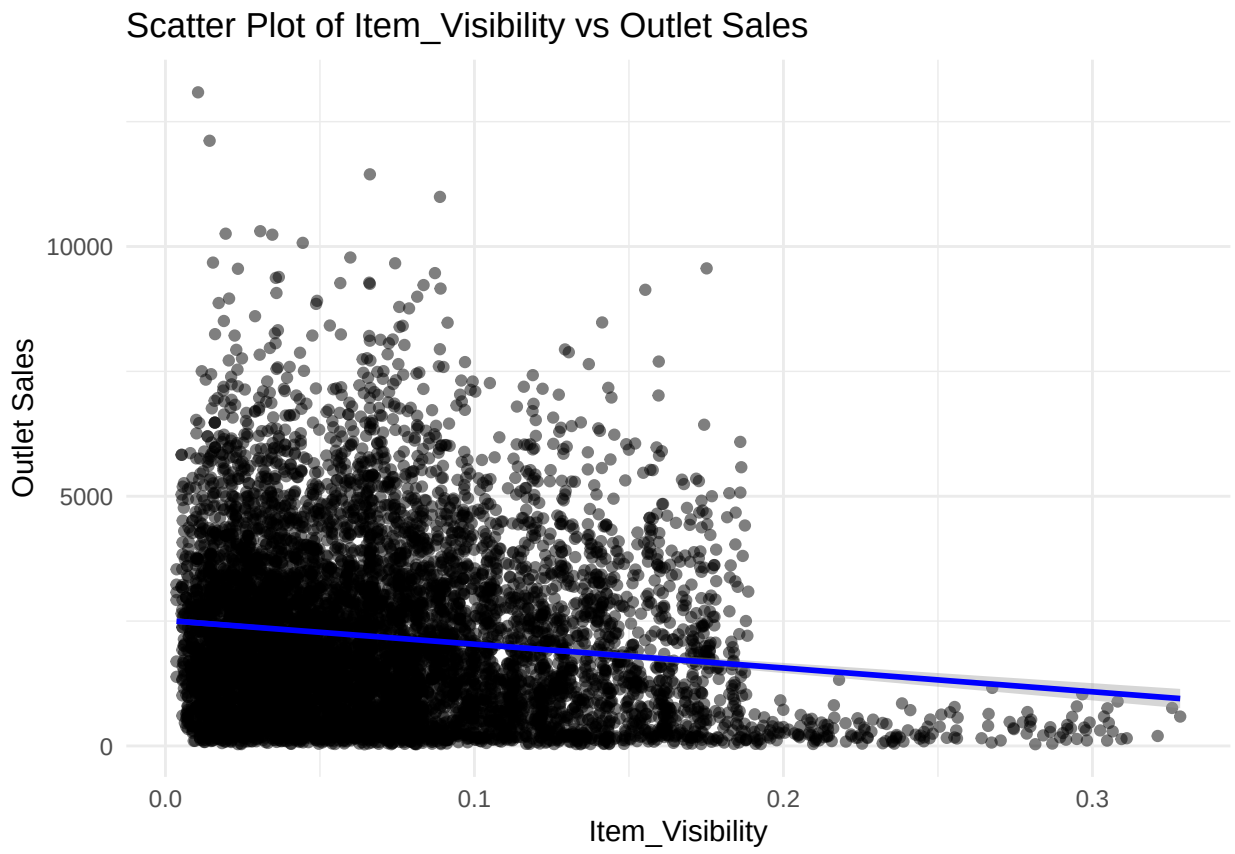
```
## `geom_smooth()` using formula = 'y ~ x'
```



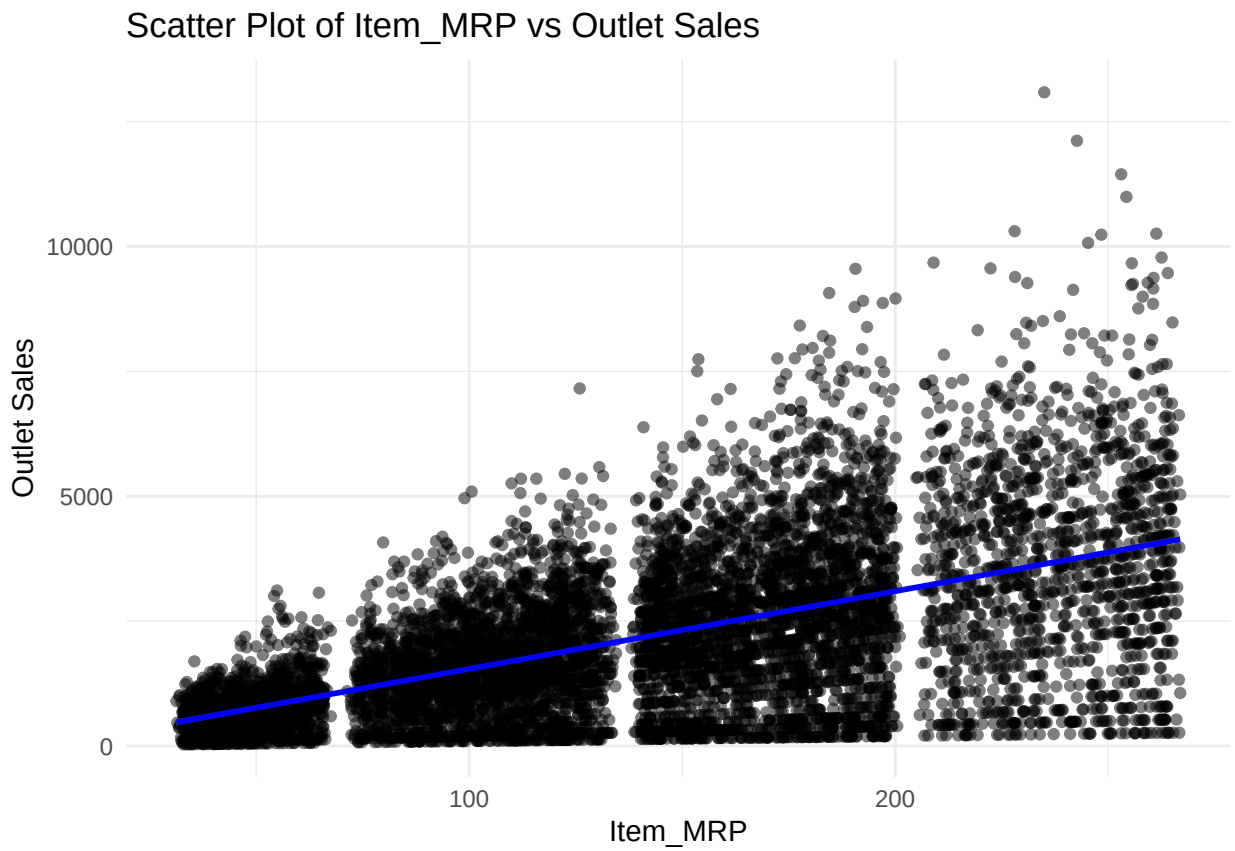

```
## `geom_smooth()` using formula = 'y ~ x'
```

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

```
## Warning: Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



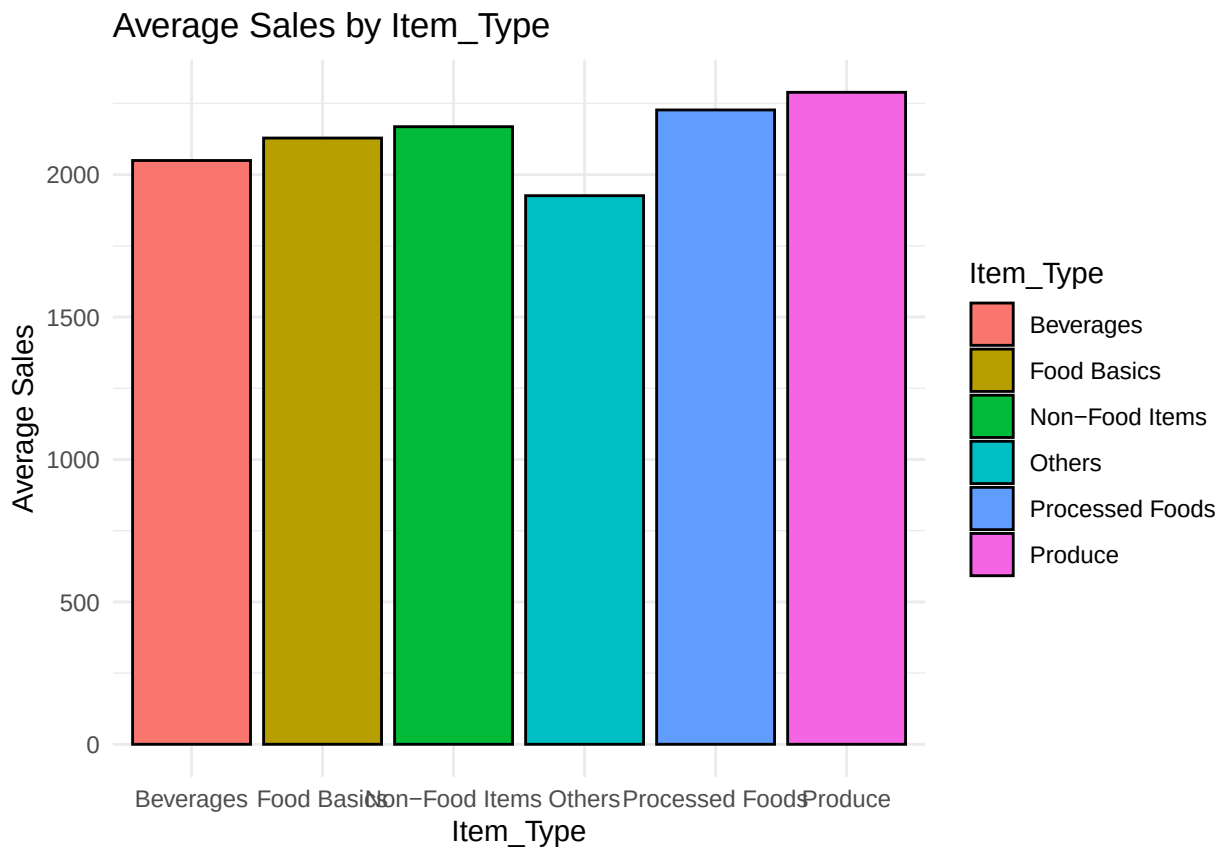

```
## `geom_smooth()` using formula = 'y ~ x'
```



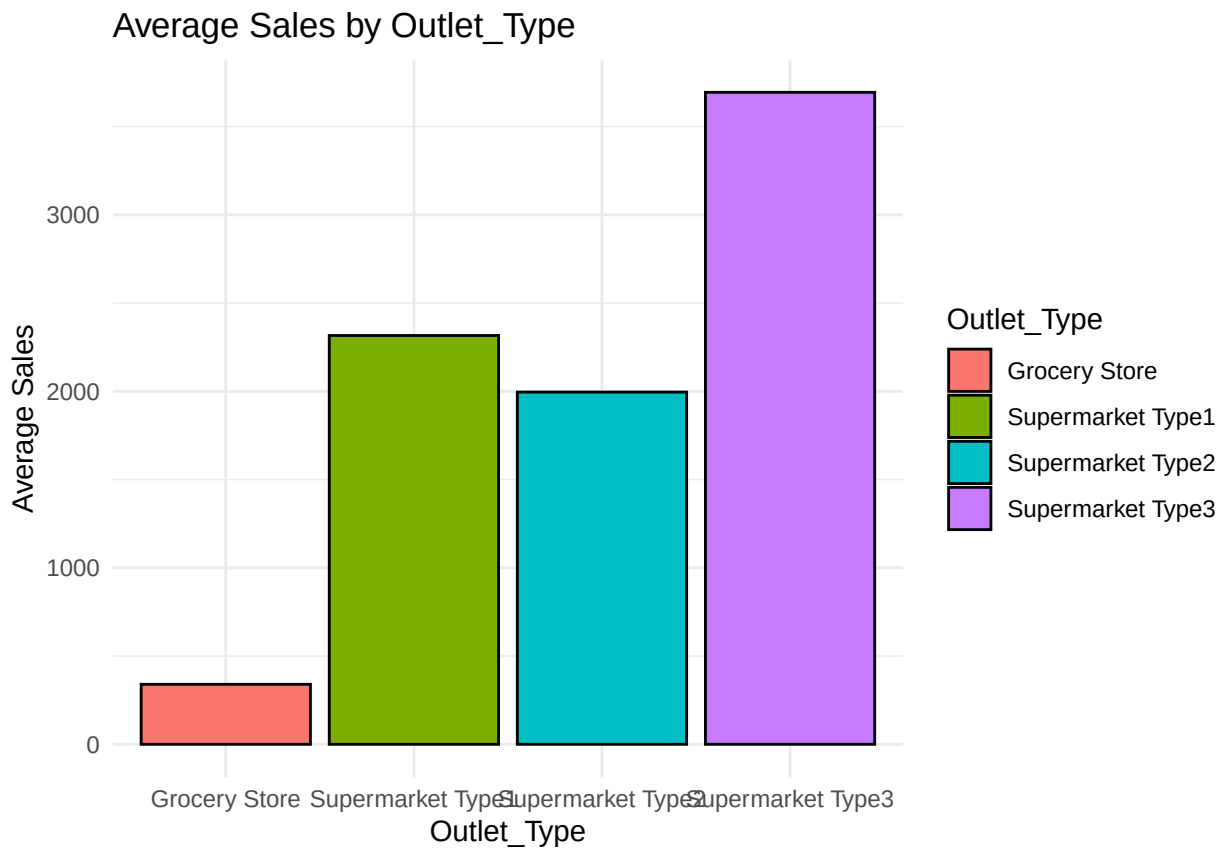

```
# Plotting Distribution of Sales for Categorical Variables
for(var in categorical_vars) {
  p <- df %>%
    group_by_(.dots = var) %>%
    summarise(Average_Sales = mean(Item_Outlet_Sales)) %>%
    ggplot(aes_string(x = var, y = "Average_Sales", fill = var)) +
    geom_bar(stat = "identity", color = "black") +
    labs(title = paste("Average Sales by", var),
         x = var, y = "Average Sales") +
    theme_minimal()
  print(p)
}
```

```
## Warning: ‘group_by_()‘ was deprecated in dplyr 0.7.0.
## i Please use ‘group_by()‘ instead.
## i See vignette('programming') for more help
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

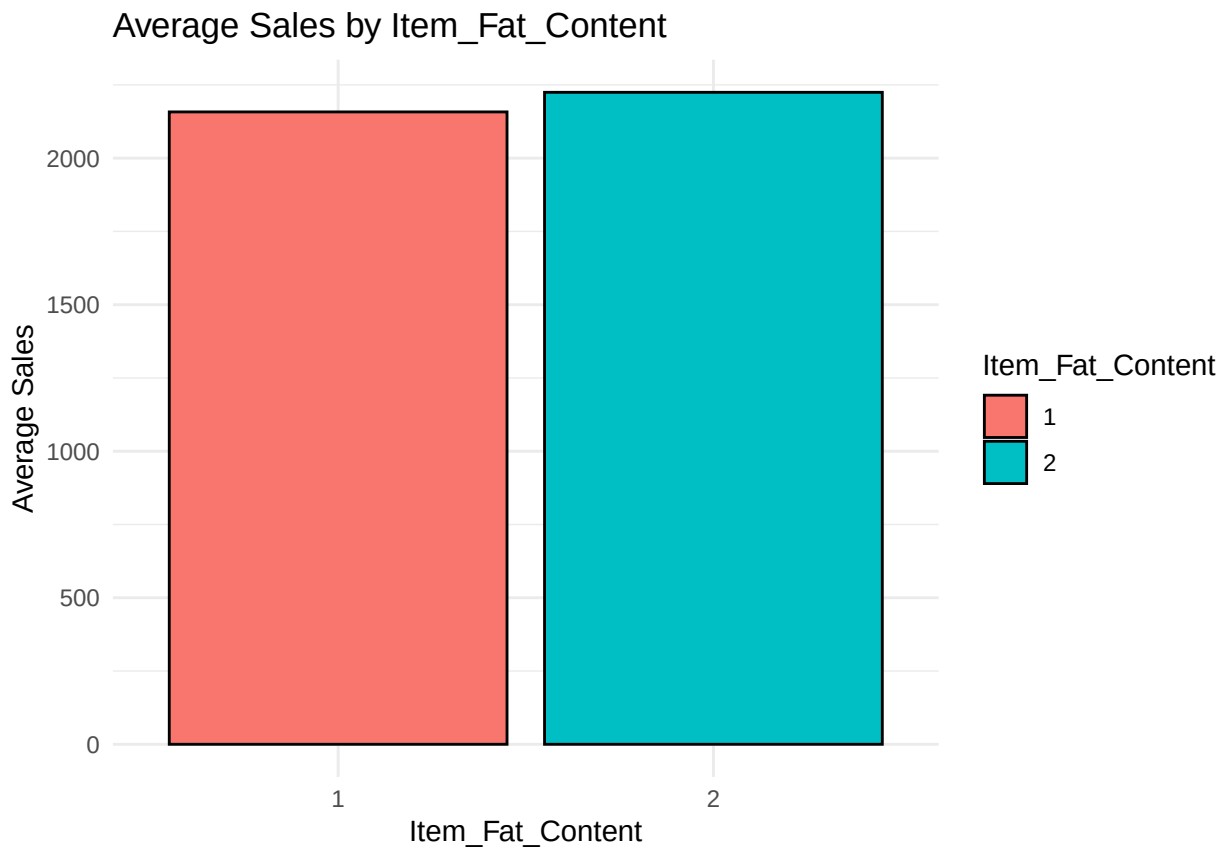
```
## Warning: ‘group_by_()‘ was deprecated in dplyr 0.7.0.
## i Please use ‘group_by()‘ instead.
## i See vignette('programming') for more help
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



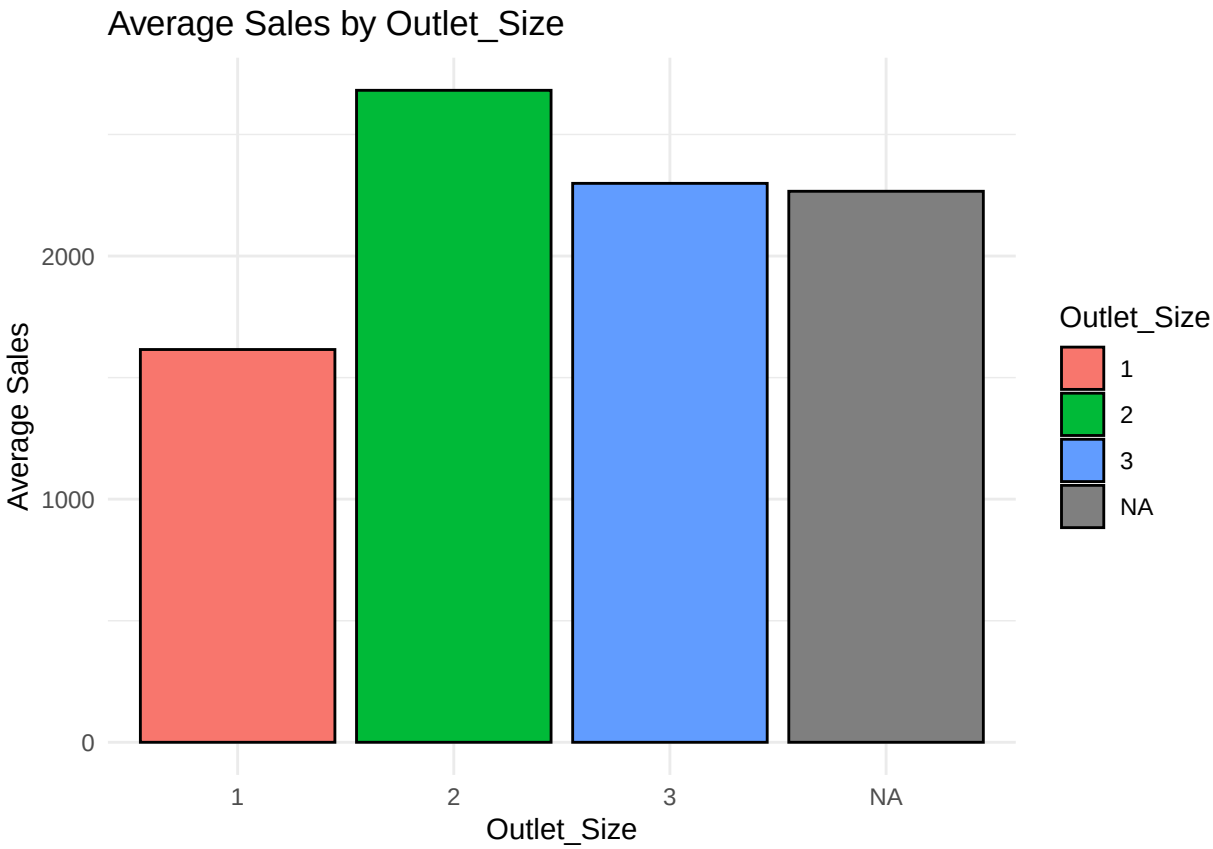

```
## Warning: ‘group_by_()‘ was deprecated in dplyr 0.7.0.
## i Please use ‘group_by()‘ instead.
## i See vignette(’programming’) for more help
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

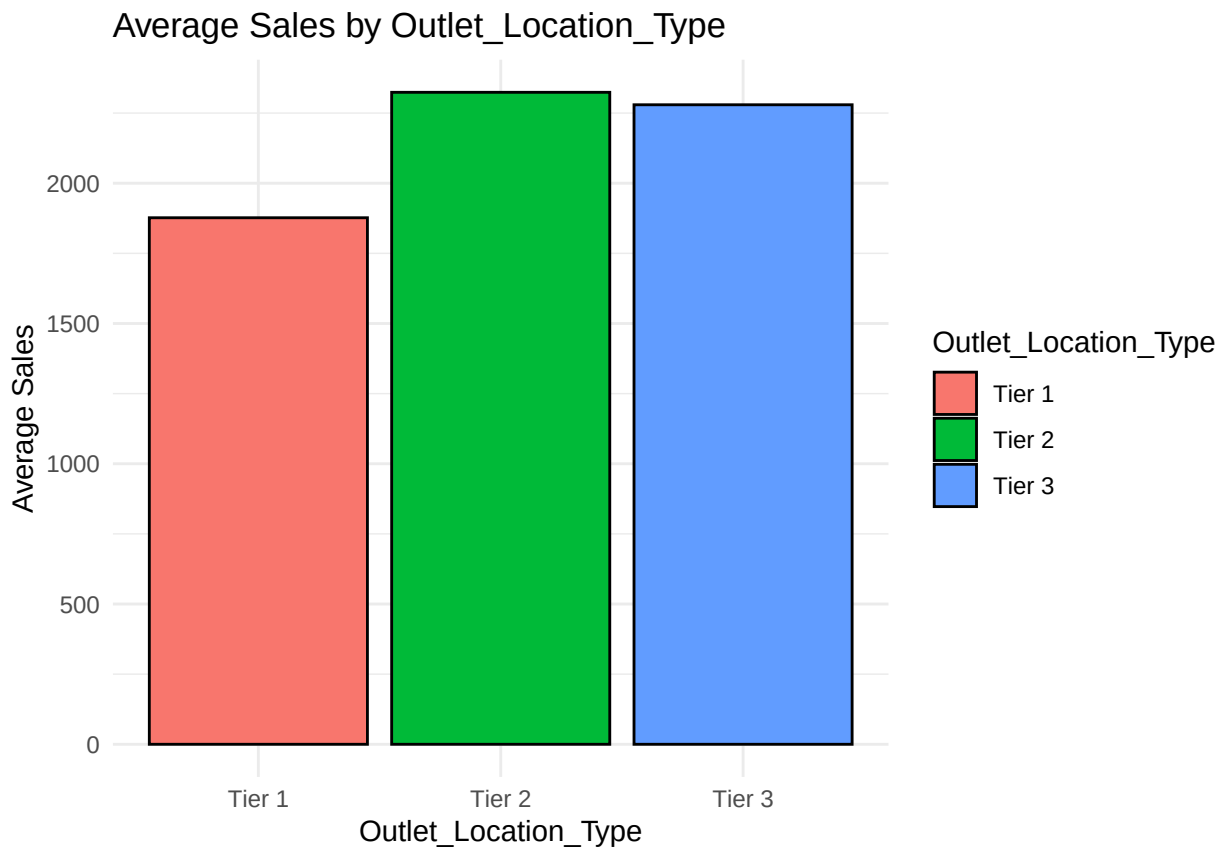



```
## Warning: ‘group_by_()‘ was deprecated in dplyr 0.7.0.
## i Please use ‘group_by()‘ instead.
## i See vignette(’programming’) for more help
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



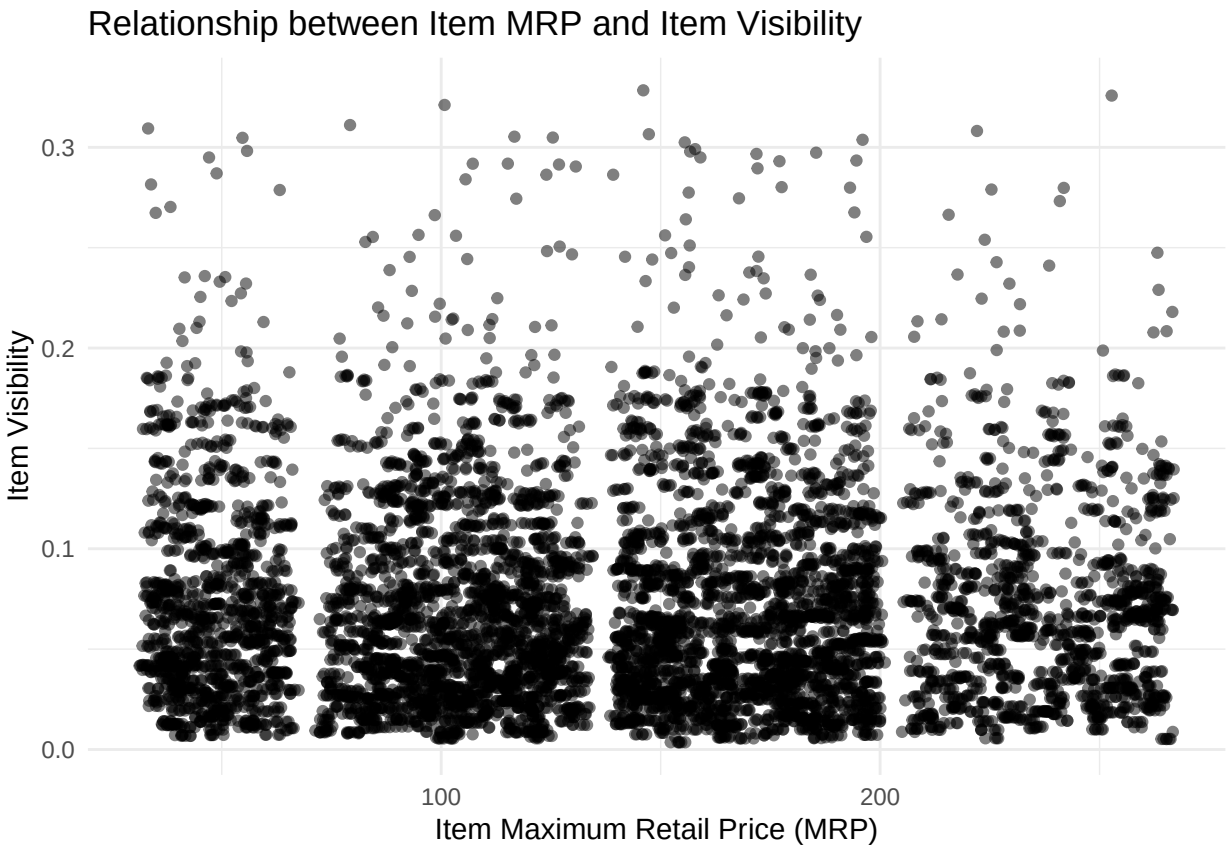

```
## Warning: ‘group_by_()‘ was deprecated in dplyr 0.7.0.
## i Please use ‘group_by()‘ instead.
## i See vignette(’programming’) for more help
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```


Average Sales by Outlet_Size
Average Sales
2000
1000
0
1
2
3
NA
Outlet_Size
Outlet_Size
1
2
3
NA

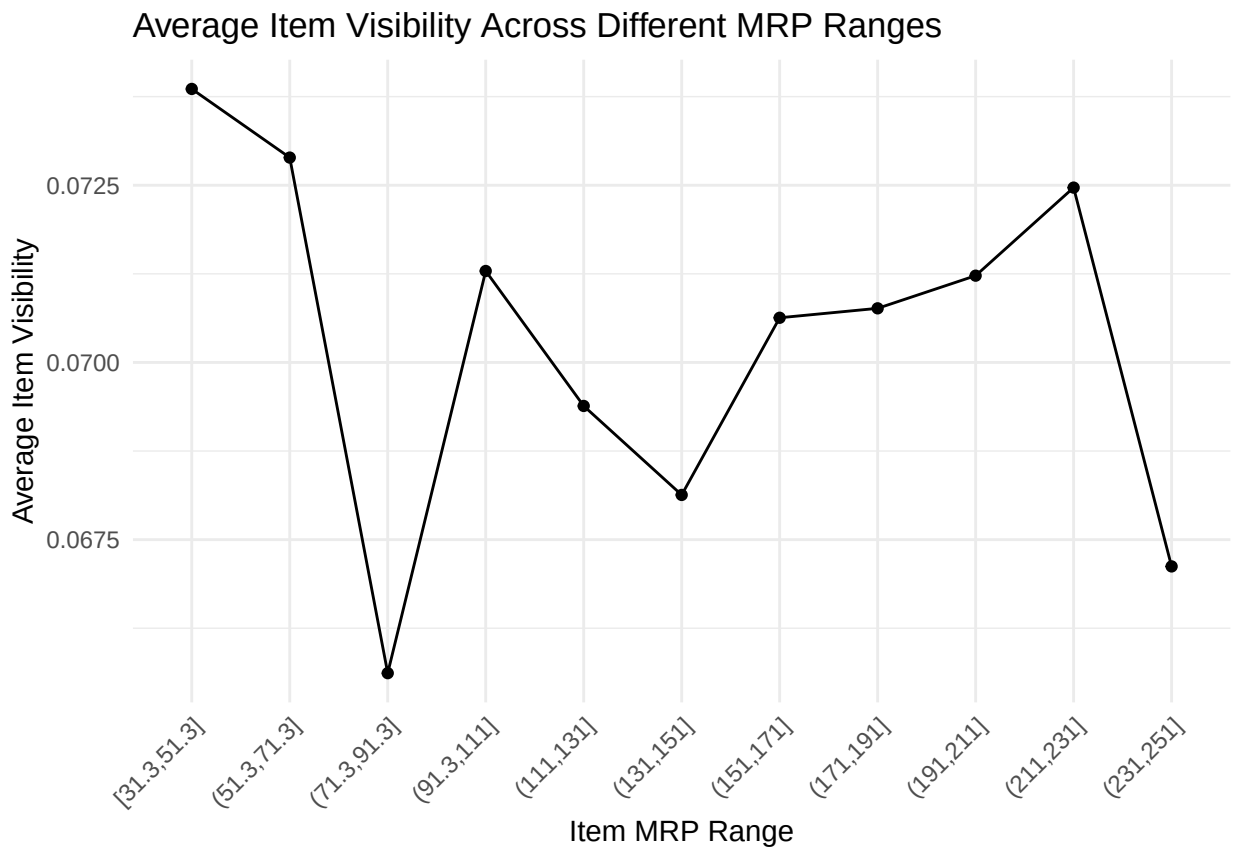



```
# Load ggplot2
library(ggplot2)


# Create a scatter plot
ggplot(df, aes(x = Item_MRP, y = Item_Visibility)) +
  geom_point(alpha = 0.5) +  # Use semi-transparent points to handle overplotting
  labs(x = "Item Maximum Retail Price (MRP)",
       y = "Item Visibility",
       title = "Relationship between Item MRP and Item Visibility") +
  theme_minimal()  # Clean minimalistic theme
```

```
## Warning: Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```r
# Create bins for Item MRP using a reasonable interval
df$MRP_Bin <- cut(df$Item_MRP, breaks=seq(from=min(df$Item_MRP), to=max(df$Item_MRP), by=20), include.l

# Calculate average visibility per MRP bin
average_visibility_per_mrp <- aggregate(Item_Visibility ~ MRP_Bin, data = df, mean)

# Create a line graph
ggplot(average_visibility_per_mrp, aes(x = MRP_Bin, y = Item_Visibility, group=1)) +
  geom_line() +  # Adds a line graph
  geom_point() +  # Adds points to each average point
  labs(x = "Item MRP Range", y = "Average Item Visibility",
       title = "Average Item Visibility Across Different MRP Ranges") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))  # Improve x-axis label readability
```



gli item che fanno vedere di può sono quelli che vendono a di meno quindi quelli di cui vogliono sbarazzarsi.

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




# Segmented Regression by Outlet Type
outlet_type_models <- df %>%
  group_by(Outlet_Type) %>%
  do(model = lm(Item_Outlet_Sales ~ Item_Visibility, data = .))

# Viewing summaries for each Outlet Type
outlet_type_summaries <- lapply(outlet_type_models$model, summary)

# Print the summaries for review
print(outlet_type_summaries)
```

```
## [[1]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
```

```
##    Min     1Q Median     3Q    Max
## -315.9 -185.5  -82.7  119.4 1436.3
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       332.59      14.53  22.884   <2e-16 ***
## Item_Visibility    66.22     111.44   0.594    0.553
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 260.9 on 1081 degrees of freedom
## Multiple R-squared:  0.0003265,  Adjusted R-squared:  -0.0005983
## F-statistic: 0.353 on 1 and 1081 DF,  p-value: 0.5525
##
##
## [[2]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2229.2 -1161.7  -317.7   817.1  7919.3
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2347.26      37.59  62.440   <2e-16 ***
## Item_Visibility  -509.99     484.61  -1.052    0.293
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1514 on 5460 degrees of freedom
##   (115 observations deleted due to missingness)
## Multiple R-squared:  0.0002028,  Adjusted R-squared:  1.968e-05
## F-statistic: 1.107 on 1 and 5460 DF,  p-value: 0.2927
##
##
## [[3]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1953.2 -1017.5  -346.2   711.6  4772.2
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      1980.78      83.49   23.73   <2e-16 ***
## Item_Visibility   224.60    1071.52    0.21    0.834
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1377 on 926 degrees of freedom
```

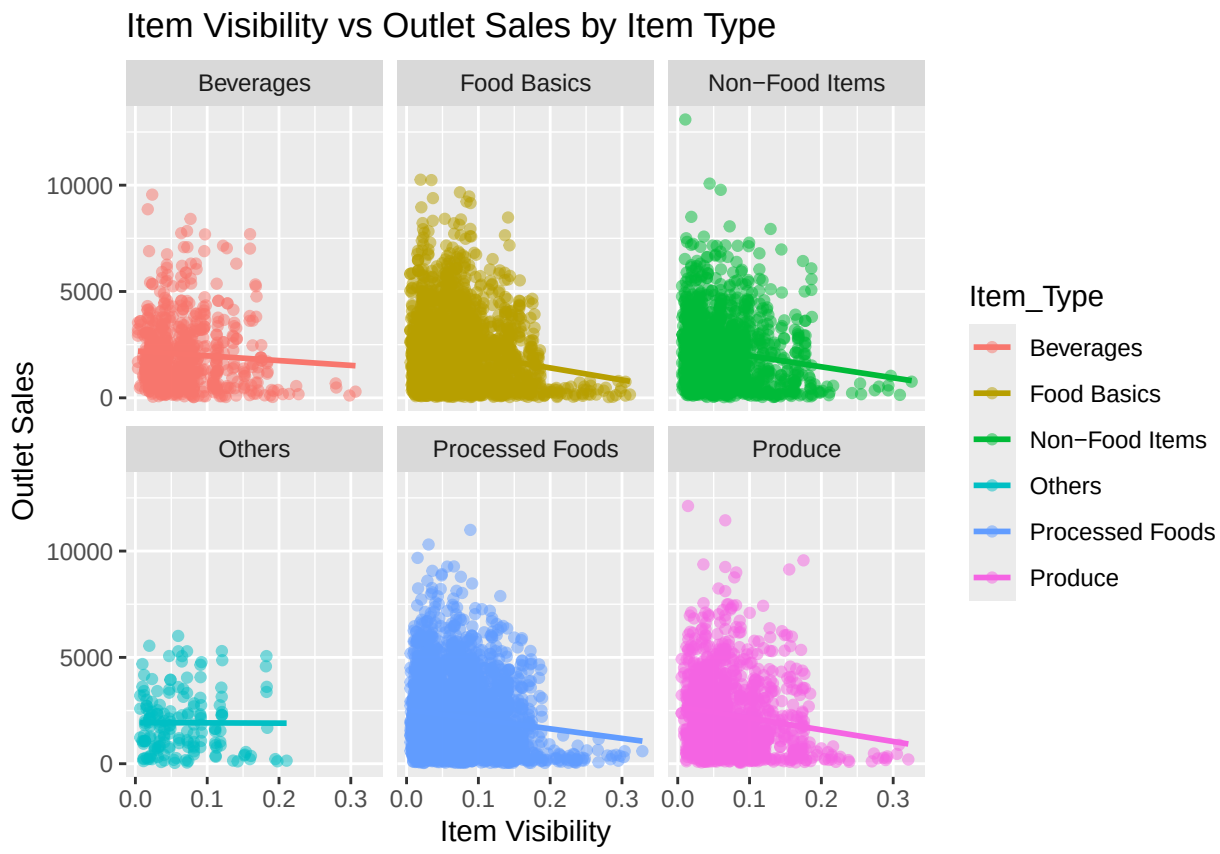
```
## Multiple R-squared:  4.744e-05,  Adjusted R-squared:  -0.001032
## F-statistic: 0.04393 on 1 and 926 DF,  p-value: 0.834
##
##
## [[4]]
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = .)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3455.1 -1654.9  -342.8  1274.6  9360.0
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3733.7      128.7  29.003   <2e-16 ***
## Item_Visibility   -632.0     1726.5  -0.366    0.714
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2129 on 933 degrees of freedom
## Multiple R-squared:  0.0001436,  Adjusted R-squared:  -0.0009281
## F-statistic: 0.134 on 1 and 933 DF,  p-value: 0.7144
```

```
# Optional: Plotting the regression lines for each type on a scatter plot
# Plotting for Item Type
ggplot(df, aes(x = Item_Visibility, y = Item_Outlet_Sales, color = Item_Type)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~ Item_Type) +
  labs(title = "Item Visibility vs Outlet Sales by Item Type", x = "Item Visibility", y = "Outlet Sales
```

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

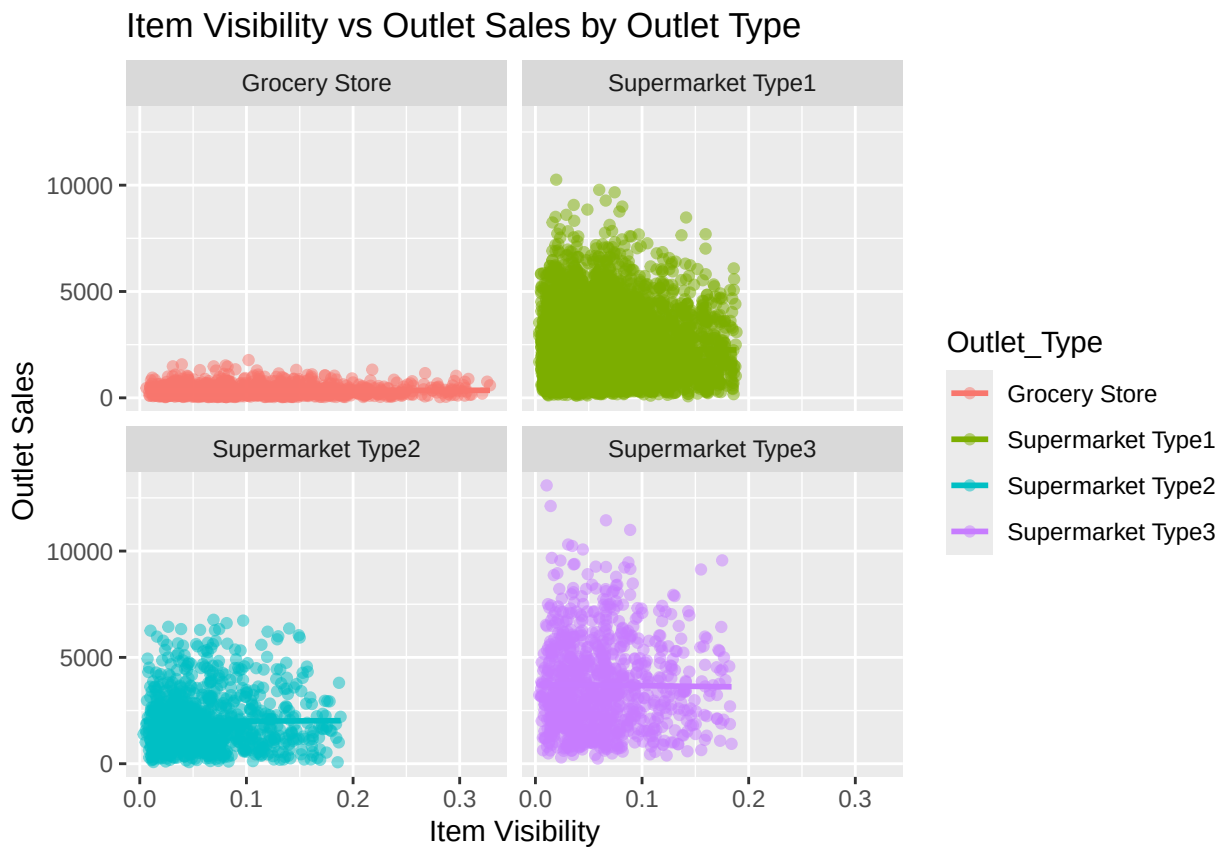
```
## Warning: Removed 115 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
# Plotting for Outlet Type
ggplot(df, aes(x = Item_Visibility, y = Item_Outlet_Sales, color = Outlet_Type)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~ Outlet_Type) +
  labs(title = "Item Visibility vs Outlet Sales by Outlet Type", x = "Item Visibility", y = "Outlet Sal
```

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

## Warning: Removed 115 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

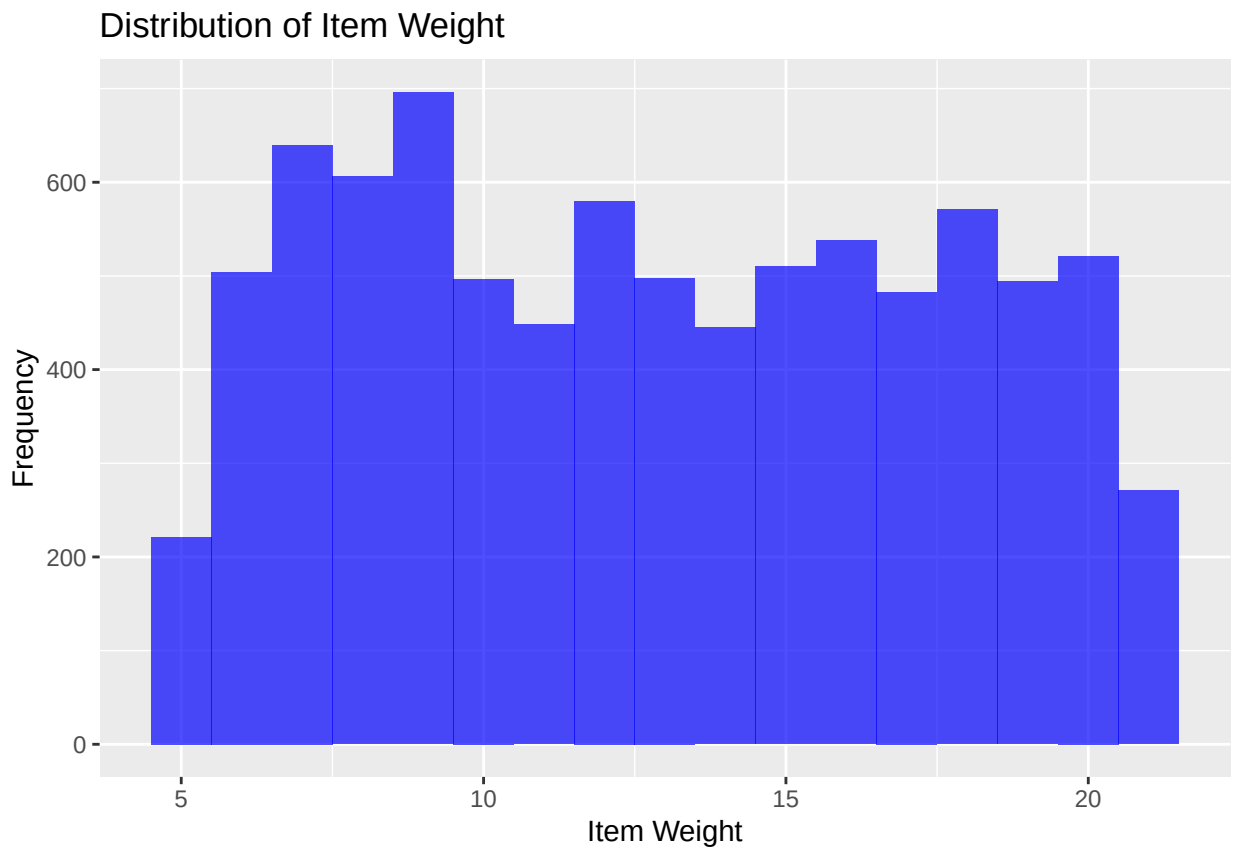


Nei grocery store l'item visibility non influisce sul numero di vendite perchè gia sai cosa vai a prendere

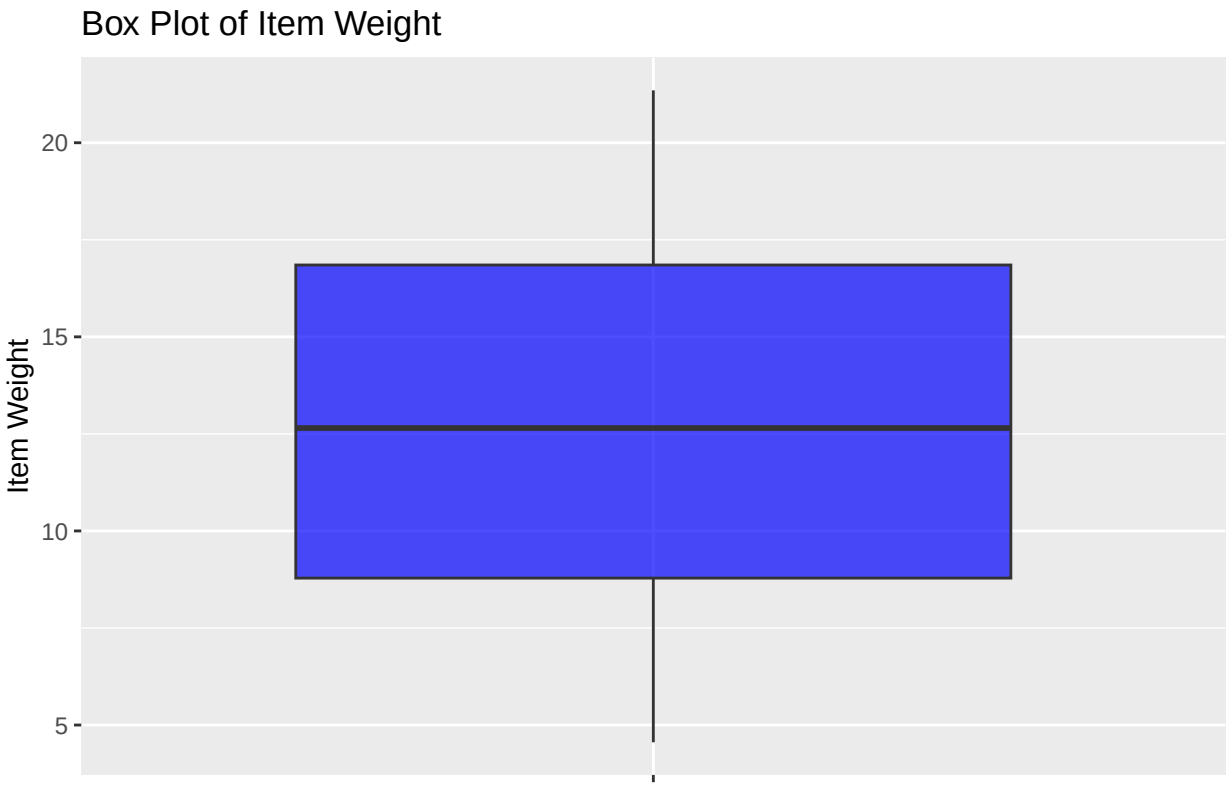
###Checking for Outliers

```
# 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")

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")
```

```r
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)
```

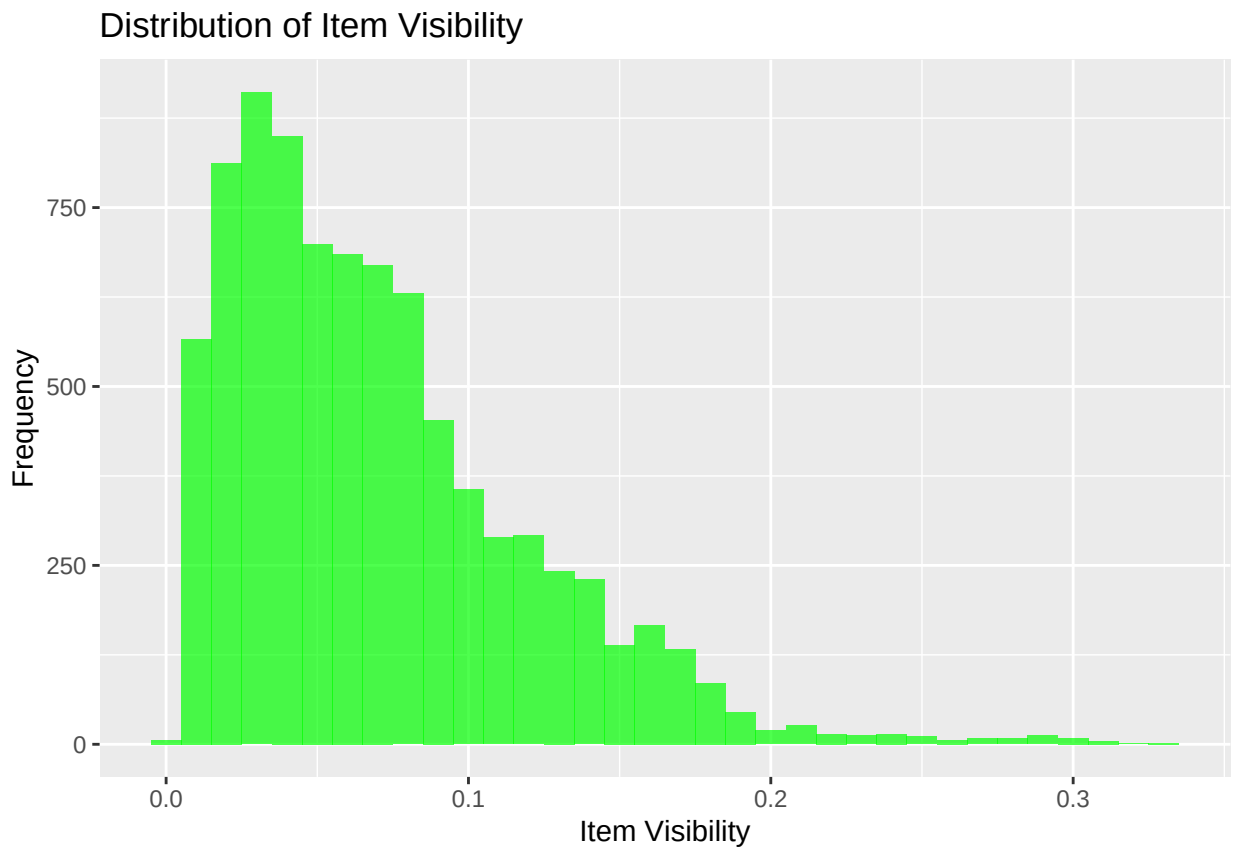

Distribution of Item Weight
Frequency
600
400
200
0
5
10
15
20
Item Weight

```
print(p2)
```



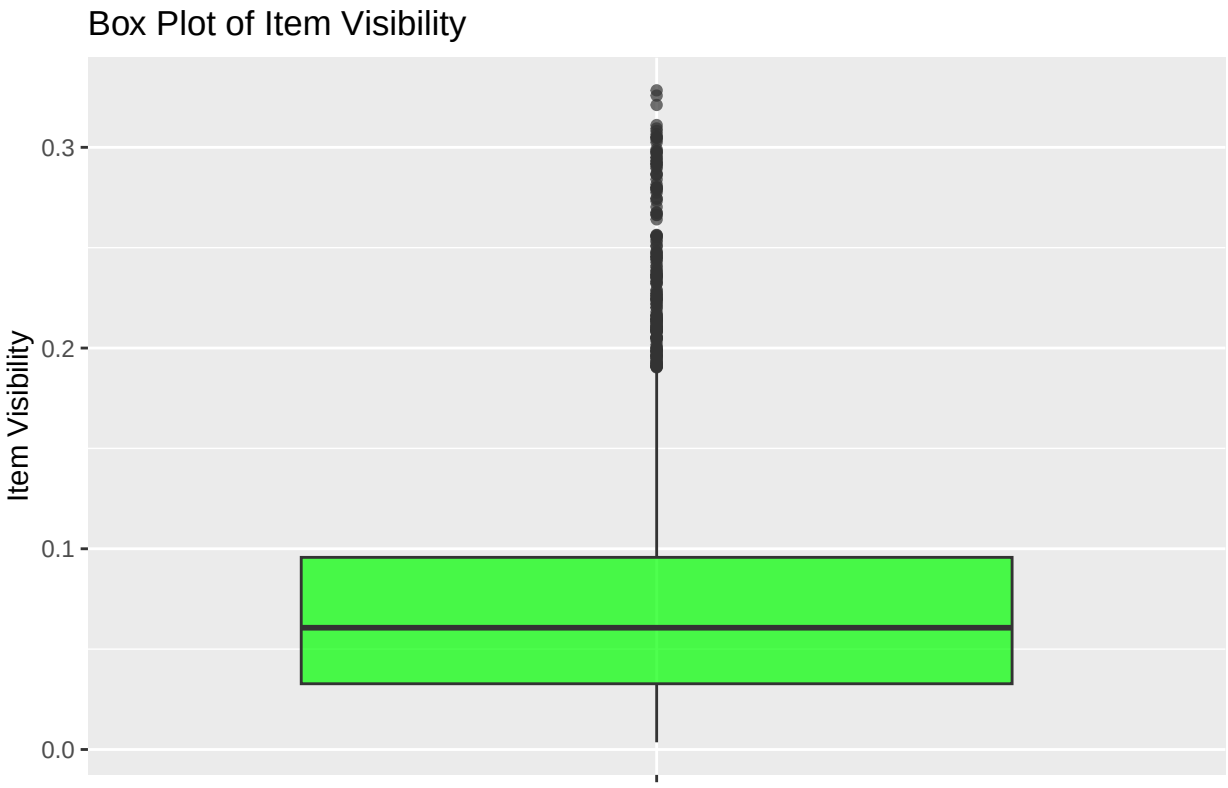

```
print(p3)
```

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

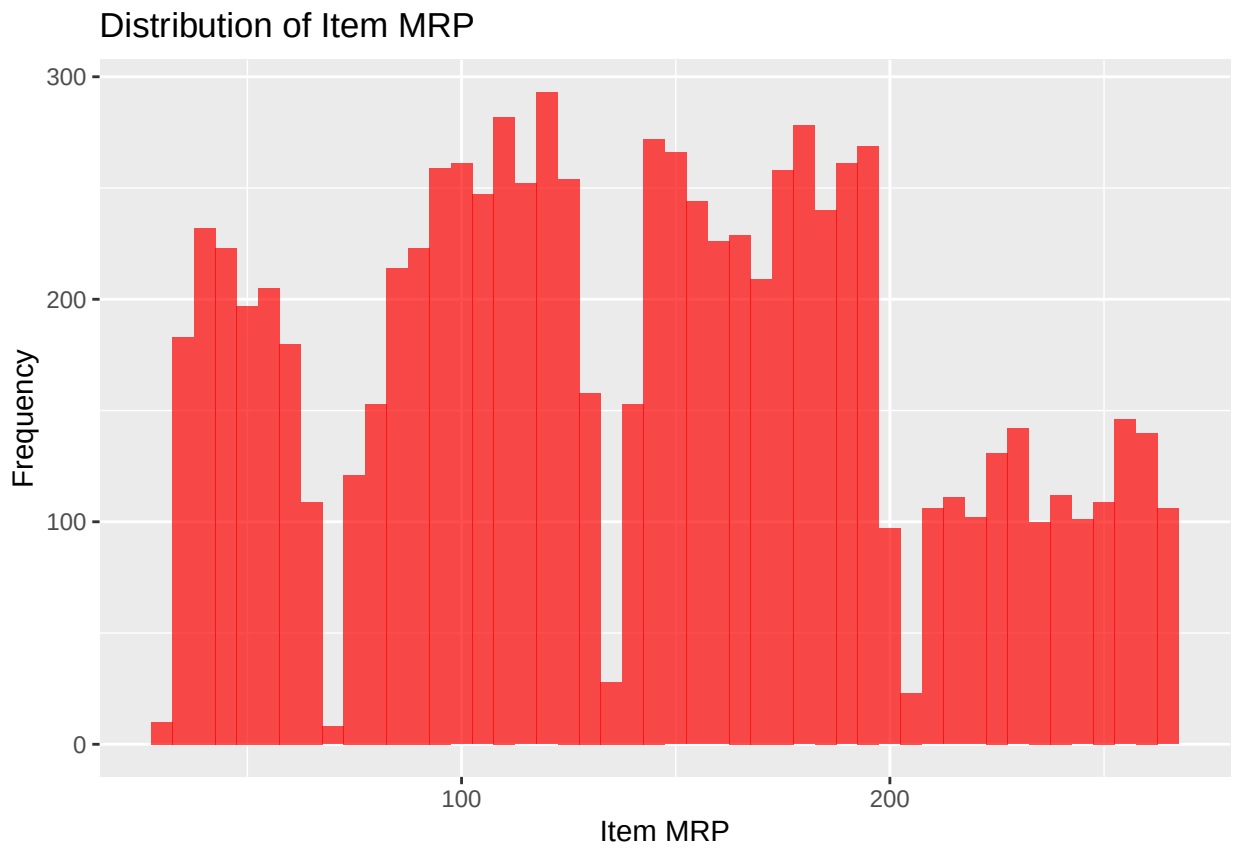



```
print(p4)
```

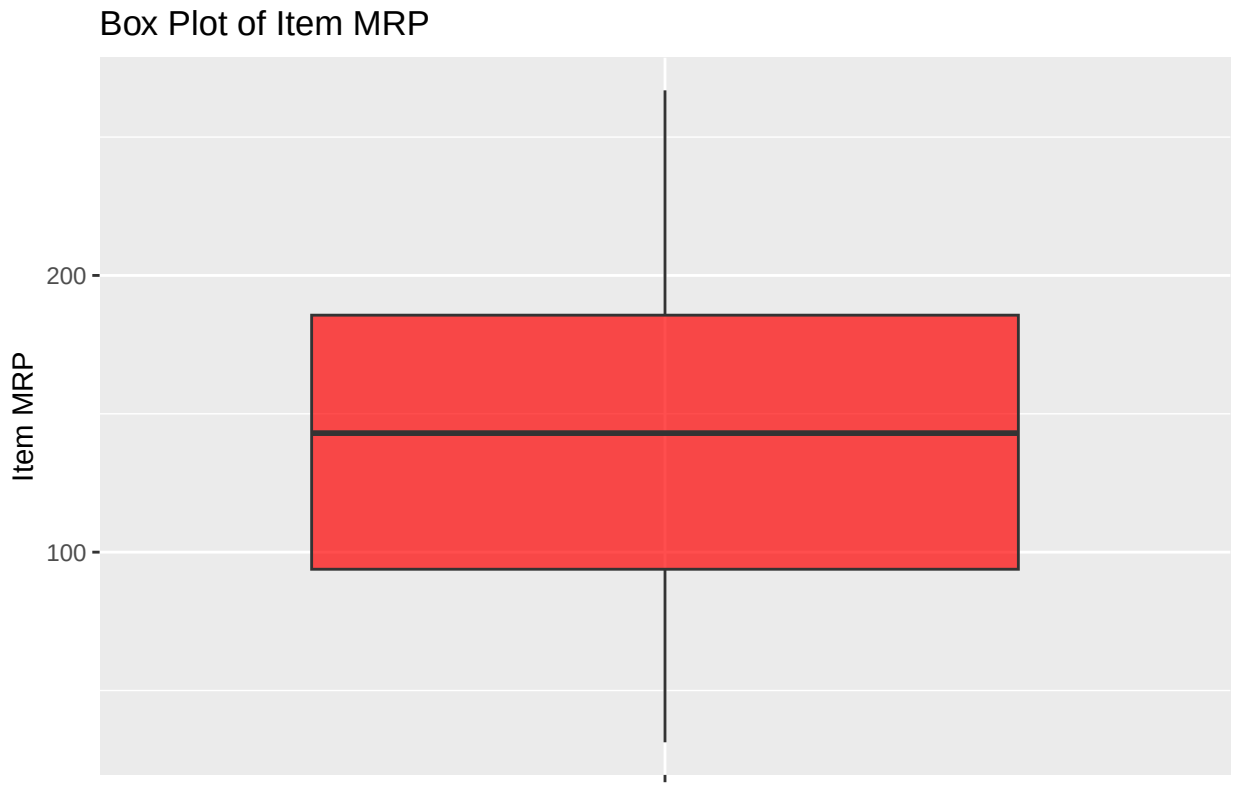
```
## Warning: Removed 115 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



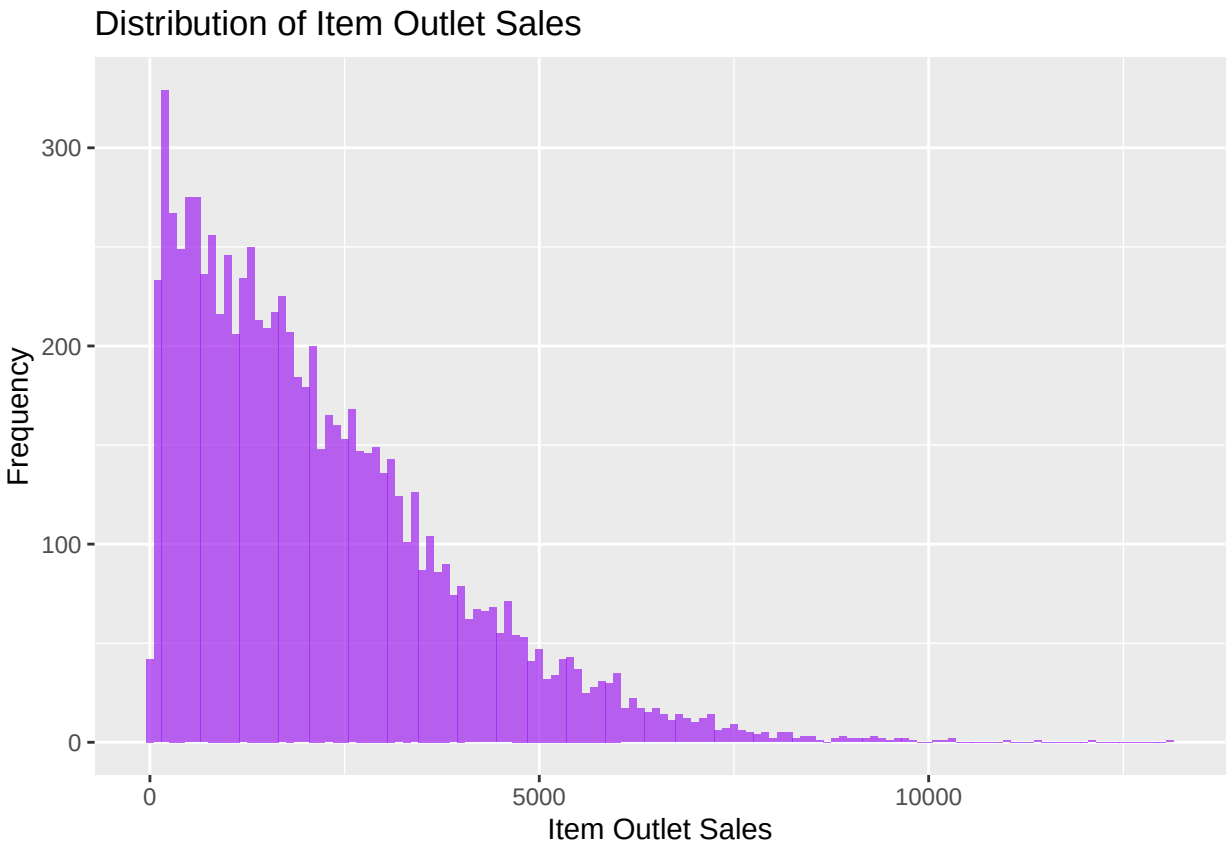

```
print(p5)
```


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

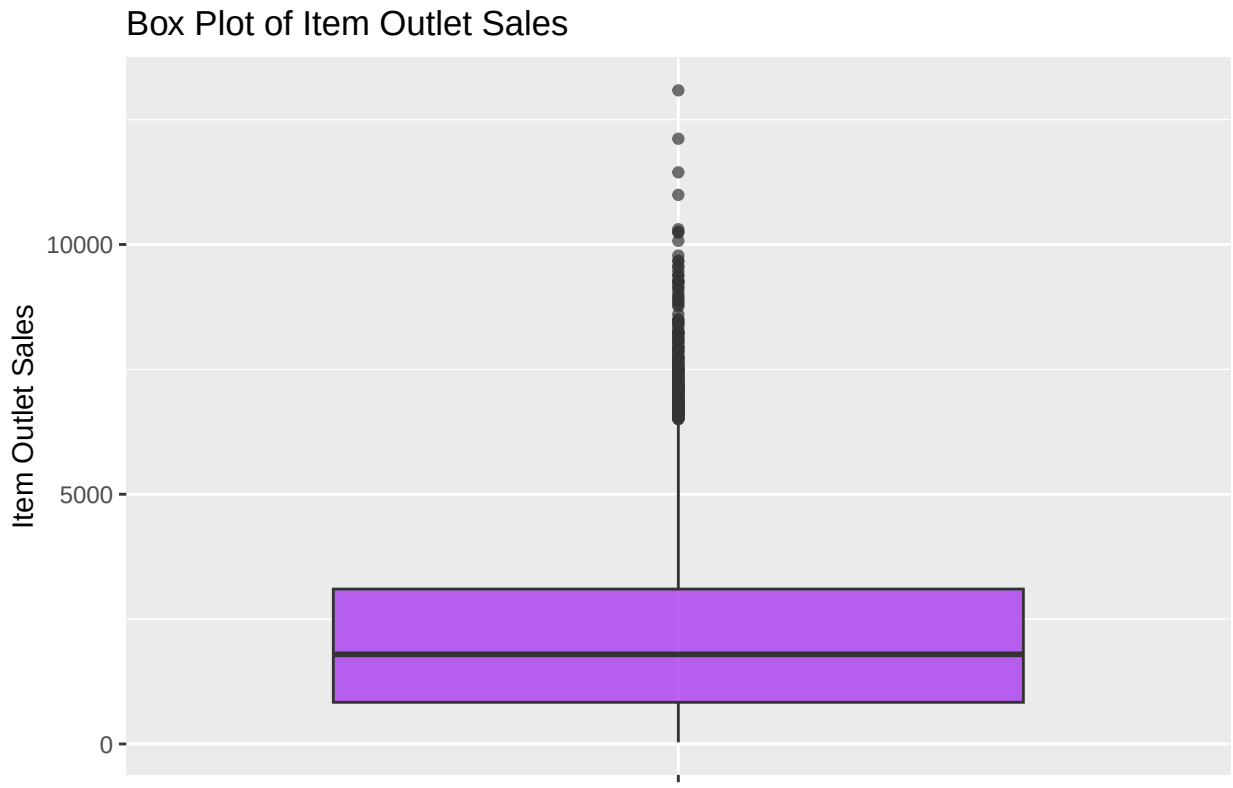
```
print(p6)
```



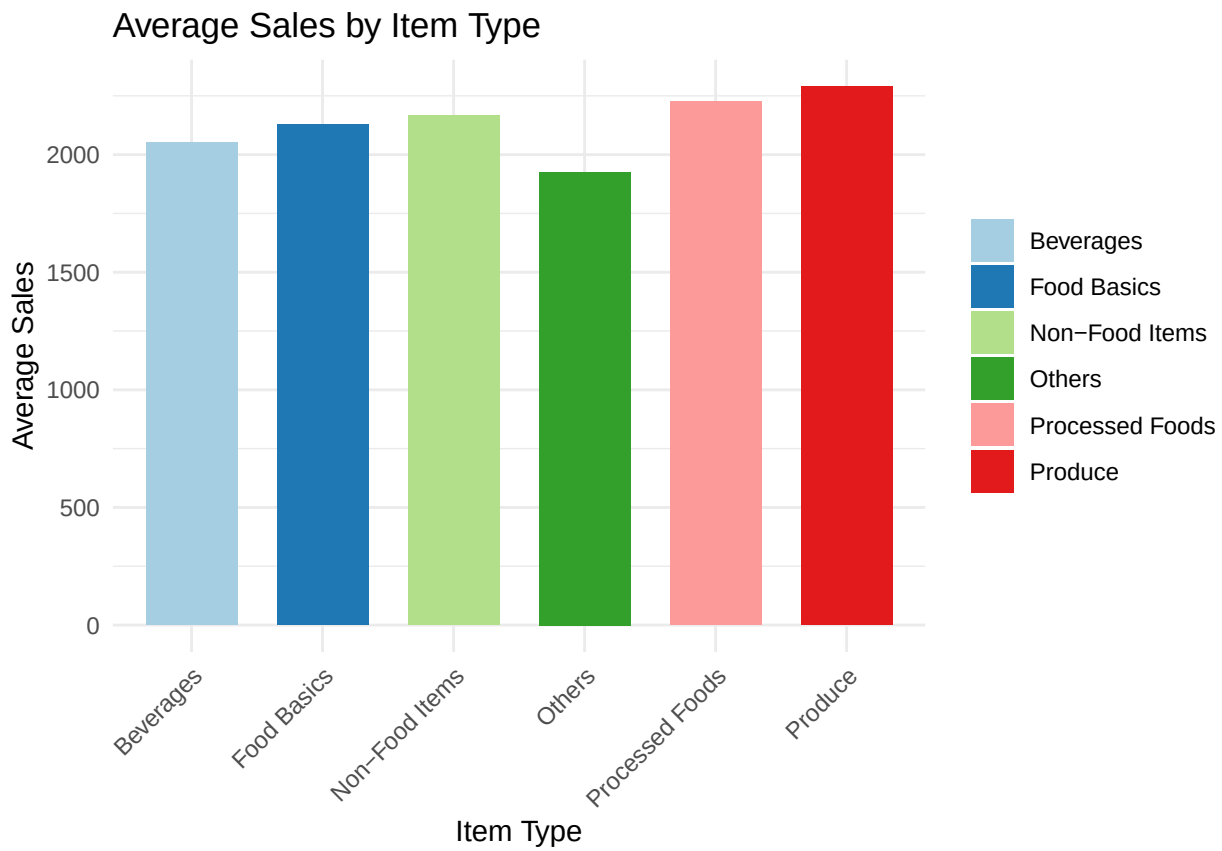

```
print(p7)
```


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

```
print(p8)
```




```r
# Esegui ANOVA
avg_sales_by_type <- df %>%
  group_by(Item_Type) %>%
  summarise(Average_Sales = mean(Item_Outlet_Sales, na.rm = TRUE))

# Creating a bar graph of average sales by item type
ggplot(avg_sales_by_type, aes(x = Item_Type, y = Average_Sales, fill = Item_Type)) +
  geom_bar(stat = "identity", width = 0.7) +  # Using identity to use the heights of the bars to repres
  labs(title = "Average Sales by Item Type", x = "Item Type", y = "Average Sales") +
  theme_minimal() +  # Clean minimalistic theme
  theme(axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x labels for better visibility
        legend.title = element_blank()) +  # Remove legend title if not needed
  scale_fill_brewer(palette = "Paired")  # Optional: Use a color palette that is visually appealing
```



There are not effective evidences of difference in item_sales based on item_type

Lower Dimensional Models

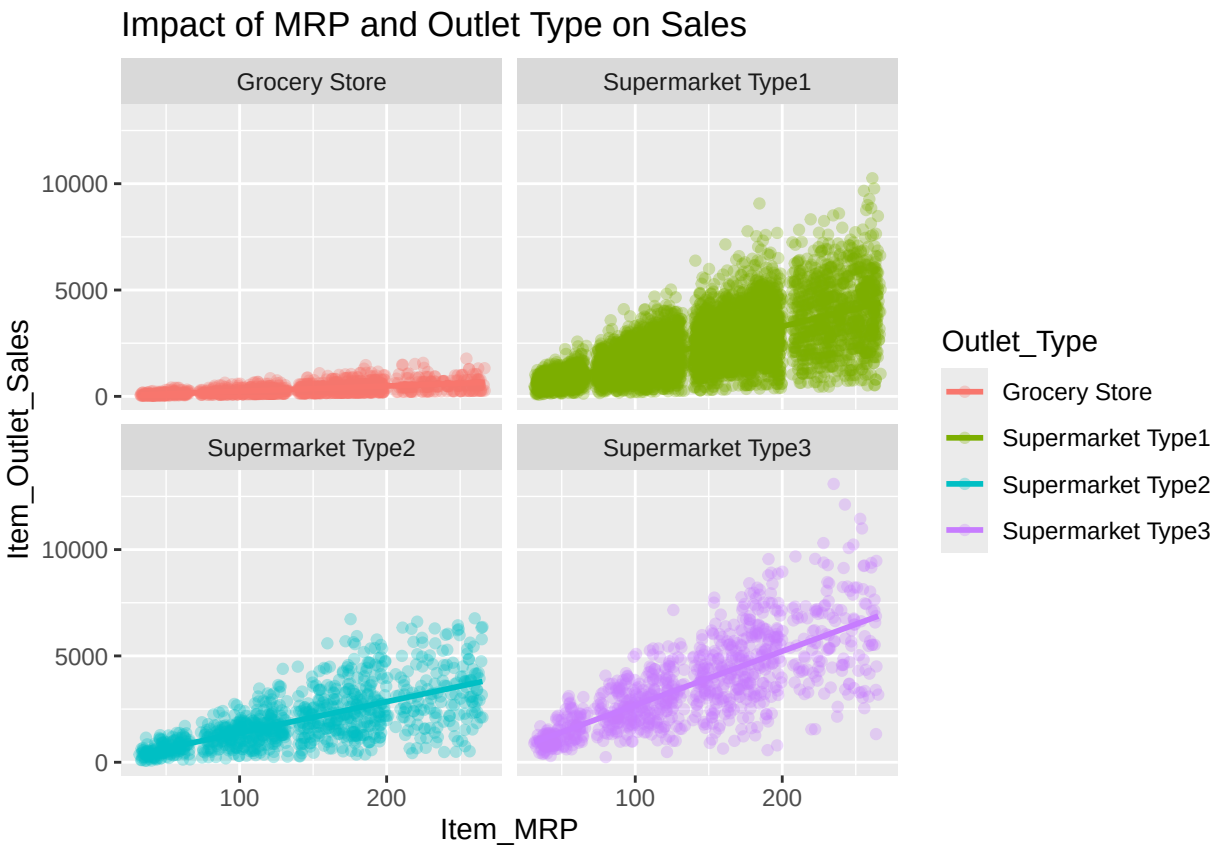
```
model_mrp_outlet <- lm(Item_Outlet_Sales ~ Item_MRP * Outlet_Type, data = df)
summary(model_mrp_outlet)
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_MRP * Outlet_Type, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5512.1  -532.6   -55.0   455.0  6979.8
##
## Coefficients:
##                                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                               -7.0777    80.9316  -0.087    0.930
## Item_MRP                                   2.4727     0.5280   4.683 2.87e-06 ***
## Outlet_TypeSupermarket Type1             -20.9350    88.3997  -0.237    0.813
## Outlet_TypeSupermarket Type2             -68.6300   118.5809  -0.579    0.563
## Outlet_TypeSupermarket Type3             157.2138   118.6379   1.325    0.185
## Item_MRP:Outlet_TypeSupermarket Type1     14.1276     0.5761  24.523  < 2e-16 ***
## Item_MRP:Outlet_TypeSupermarket Type2     12.1463     0.7690  15.796  < 2e-16 ***
```

```
## Item_MRP:Outlet_TypeSupermarket Type3  22.8768     0.7752  29.512  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1072 on 8515 degrees of freedom
## Multiple R-squared:  0.6054, Adjusted R-squared:  0.6051
## F-statistic:  1866 on 7 and 8515 DF,  p-value: < 2.2e-16
```

```r
# Plotting this relationship
ggplot(df, aes(x = Item_MRP, y = Item_Outlet_Sales, color = Outlet_Type)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~Outlet_Type) +
  labs(title = "Impact of MRP and Outlet Type on Sales")
```

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



Visualization Interpretation

The scatter plots across different outlet types (Grocery Store, Supermarket Type1, Type2, and Type3) show how Item_MRP influences sales:

Grocery Store: The sales seem mostly flat regardless of the MRP, suggesting that price changes do not significantly influence sales in grocery stores. This could indicate price sensitivity or a limited range of products where price does not play a major role.

Supermarket Type1: There appears to be a moderate positive relationship between MRP and sales. This indicates that as the price of an item increases, sales also tend to increase, possibly due to a perception of higher quality or the availability of a wider range of products.

Supermarket Type2: Similar to Type1, but the relationship looks slightly weaker. This might be due to different customer demographics or store locations.

Supermarket Type3: Shows a strong positive relationship between MRP and sales, suggesting that customers at Type3 supermarkets are less price-sensitive and possibly more driven by product quality or brand.

Model Summary Interpretation The regression output provides the following insights:

Intercept: The intercept is not meaningful on its own without context as it represents sales when MRP is zero and not belonging to any specific outlet type, which is not a practical scenario.

Item_MRP: There's a significant positive coefficient for MRP, indicating that overall, an increase in MRP tends to lead to an increase in sales across the dataset.

Outlet_Type Coefficients:

Supermarket Type1: The interaction term Item_MRP:Outlet_TypeSupermarket Type1 is significant and positive, which supports the visual interpretation that higher MRP increases sales more in this supermarket type compared to the baseline (Grocery Store). Supermarket Type2: Similarly, the positive coefficient for the interaction with MRP suggests increased sales with higher MRP, but the effect is smaller than in Type1. Supermarket Type3: Exhibits the strongest positive interaction effect with MRP, indicating that sales in these stores are most positively influenced by higher MRP. Model Fit:

The Multiple R-squared value is 0.6054, meaning that about 60.54% of the variability in sales is explained by the model, which is reasonably good given the complexity of retail sales dynamics. The F-statistic is highly significant ($p < 2.2e\text{-}16$), indicating that the model is statistically significant and that the model fits the data better than a model with no predictors. Conclusions This analysis suggests that MRP and outlet type are significant predictors of sales. Higher MRP generally leads to higher sales, especially in supermarket types where customers might be less sensitive to price increases. Grocery stores show minimal changes in sales with changes in MRP, indicating a different customer base or purchase behavior.

These insights can help BigMart tailor its pricing strategies according to the type of outlet, potentially focusing on premium pricing strategies in supermarket types where the customer base responds positively to higher-priced items.

```r
# Adjusting the model to focus only on Item Visibility
model_visibility <- lm(Item_Outlet_Sales ~ Item_Visibility, data = df)
summary(model_visibility)
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2374.9 -1299.2  -404.0   899.1 10623.0
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2514.29      32.28   77.88   <2e-16 ***
## Item_Visibility -4766.33     375.59  -12.69   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

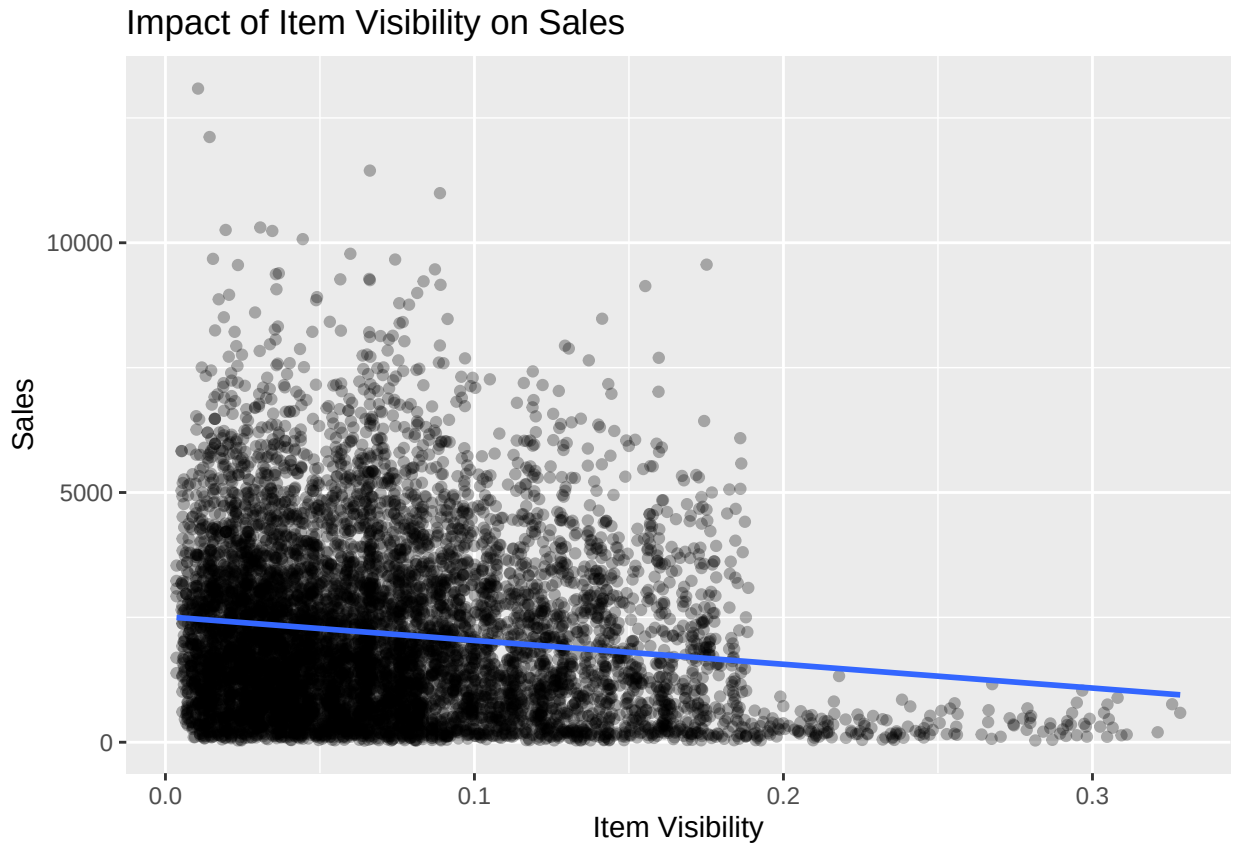
```
## Residual standard error: 1692 on 8406 degrees of freedom
##   (115 observations deleted due to missingness)
## Multiple R-squared:  0.0188, Adjusted R-squared:  0.01868
## F-statistic:   161 on 1 and 8406 DF,  p-value: < 2.2e-16
```

```r
# Plotting the relationship between Item Visibility and Sales
ggplot(df, aes(x = Item_Visibility, y = Item_Outlet_Sales)) +
  geom_point(alpha = 0.3) +  # Point opacity reduced for better visualization of density
  geom_smooth(method = "lm", se = FALSE) +  # Linear model smoothing without confidence intervals
  labs(title = "Impact of Item Visibility on Sales",  # Updated title
       x = "Item Visibility",  # X-axis label
       y = "Sales")  # Y-axis label
```

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

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

```
## Warning: Removed 115 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



Dobbiamo scriveree interprertazione di questo grafico

#Feature Engineering

```r
df <- df %>%
  mutate(
    # Convert Establishment Year to Age
    Outlet_Age = as.numeric(format(Sys.Date(), "%Y")) - Outlet_Establishment_Year,

    # Interaction between MRP and Outlet Type
    MRP_x_OutletType = Item_MRP * as.numeric(as.factor(Outlet_Type)),


    # Simplifying Item Type
    Item_Type_Simplified = case_when(
      grepl("Foods", Item_Type) ~ "Foods",
      grepl("Drinks", Item_Type) ~ "Drinks",
      TRUE ~ "Non-Consumables"
    )
  )

# Ensure that categorical variables are in the correct format
df$Outlet_Type <- as.factor(df$Outlet_Type)
df$Item_Fat_Content <- as.factor(df$Item_Fat_Content)
df$Item_Type_Simplified <- as.factor(df$Item_Type_Simplified)
df$Outlet_Identifier <- as.factor(df$Outlet_Identifier)

df <- df %>%
  select(-c(Outlet_Age, MRP_x_OutletType))
```

trasformare da anno di nascita ad età

trasformare in percentuale item visibility.

Modelli di previsione su:

-item rmp -outlet size(?) o type -Percentage of visibility -Outlet id

poi una volta fatto questo, scegliamo quali variabili sono più influenti e le buttiamo nel modello finale

e poi ragioniamo su come fare il modello finale

Feature Selection with Stepwise Regression

#“‘{r feature selection with stepwise regression} #library(MASS)

Set up the full model with all predictors

Remove rows with any missing values in the dataset

#df_clean <- na.omit(df)

Check the dimensions to see how many rows are left

#dim(df_clean) #full_model <- lm(Item_Outlet_Sales ~ ., data = df_clean)

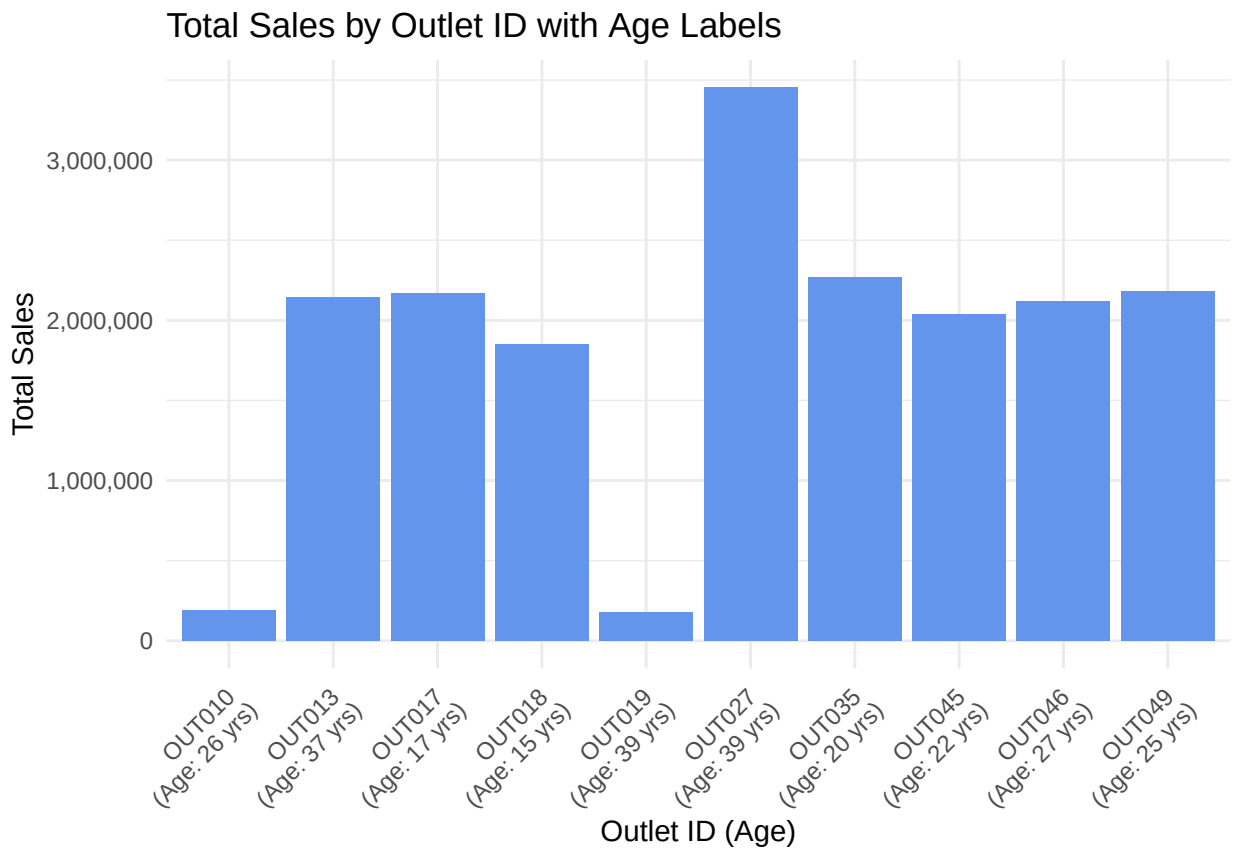
Perform stepwise regression based on AIC

#stepwise_model <- stepAIC(full_model, direction = "both") #summary(stepwise_model) #final_formula <- formula(stepwise_model) #"'

```
# Add a new column 'Outlet_Age' to the dataframe
df <- df %>%
  mutate(Outlet_Age = as.numeric(format(Sys.Date(), "%Y")) - Outlet_Establishment_Year)

# Display the age of each store
# This creates a summary table with Outlet_ID and its corresponding Outlet_Age
store_ages <- df %>%
  select(Outlet_Identifier, Outlet_Age) %>%  # Select the necessary columns
  distinct() %>%  # Remove duplicate rows to ensure each store is listed once
  arrange(Outlet_Identifier)  # Optional: Sort by Outlet Identifier for easier reading

# Print the resulting table to see each store's age
print(store_ages)
```

```
##    Outlet_Identifier Outlet_Age
## 1             OUT010         26
## 2             OUT013         37
## 3             OUT017         17
## 4             OUT018         15
## 5             OUT019         39
## 6             OUT027         39
## 7             OUT035         20
## 8             OUT045         22
## 9             OUT046         27
## 10            OUT049         25
```

```
# Aggregate total sales by Outlet_ID
sales_summary <- df %>%
  group_by(Outlet_Identifier) %>%
  summarise(Total_Sales = sum(Item_Outlet_Sales)) %>%
  ungroup()
# Merge sales summary with store ages
sales_with_age <- merge(sales_summary, store_ages, by = "Outlet_Identifier")
# Load the ggplot2 library
library(ggplot2)

# Plot total sales by Outlet_ID with Outlet_Age in the x-axis labels
ggplot(sales_with_age, aes(x = Outlet_Identifier, y = Total_Sales)) +
  geom_bar(stat = "identity", fill = "cornflowerblue") +
  scale_y_continuous(labels = scales::comma) +
  labs(title = "Total Sales by Outlet ID with Age Labels",
       x = "Outlet ID (Age)",
       y = "Total Sales") +
  scale_x_discrete(labels = paste(sales_with_age$Outlet_Identifier, "\n(Age:", sales_with_age$Outlet_Ag
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1))  # Adjust text to make it readabl
```



####counting number of sales for store perche mi sembra strano

```
# Sum the total sales for each outlet, grouped by Outlet Identifier and include Outlet Age
sales_summary <- df %>%
  group_by(Outlet_Identifier, Outlet_Age) %>%
  summarise(Total_Sales = sum(Item_Outlet_Sales), .groups = 'drop') %>%
  arrange(desc(Total_Sales))

# Print the summary table with total sales and outlet age
print(sales_summary)
```

```
## # A tibble: 10 x 3
##    Outlet_Identifier Outlet_Age Total_Sales
##    <fct>                  <dbl>       <dbl>
##  1 OUT027                    39    3453926.
##  2 OUT035                    20    2268123.
##  3 OUT049                    25    2183970.
##  4 OUT017                    17    2167465.
##  5 OUT013                    37    2142664.
##  6 OUT046                    27    2118395.
##  7 OUT045                    22    2036725.
##  8 OUT018                    15    1851823.
##  9 OUT010                    26     188340.
## 10 OUT019                    39     179694.
```

Modelli di previsione su:

-item rmp -outlet size(?) o type -Percentage of visibility -Outlet id

Ora dobbiamo creare dei lower-level model per vedere quali variabili utilizzare nel nostro dataset finale, intuitivamente abbiamo pensato che item rmp, outlet size, item visibility e outlet_id possano essere dei buoni indicatori.

Per fare ciò, proviamo a predicare la nostra variabile target Item_Sales con queste 4 tramite GBM, possiamo usare gbm perchè avendo poche va

Non possiamo usare outlet_ID perchè:

non possiamo trasformarlo in numerico in quanto i valori non seguono un ordin -I record unici sono solamente 10 quindi il modello avrebbe errori

Possiaom però sostituire Outlet_ID con un'altra variabile ad esempio "Outlet_Age", per vedere se il numero di anni da cui è aperto un negozio inlfuisce su quanto venda un determinato prodotto.

prima di tutto però dobbiamo dividere in test set e train set il nostro modello, useremo 80% in train ed il 20% in test

Poiche non si possono eseguire lm se delle variabili hanno missing values, andremo a rimuovere i missing value/NA rimasti fino ad ora in modo tale da skippare le righe contenenti.

```
library(caret)

# Impostazione del seed per la riproducibilità
set.seed(123)

# Divisione del dataset in set di training e test
index <- createDataPartition(df$Item_Outlet_Sales, p = 0.8, list = FALSE)
train_data <- df[index, ]
test_data <- df[-index, ]
```

Ora andremo ad eseguire i 4 modelli sul train set

```
# Carico le librerie necessarie, se non già caricato
if (!require("ggplot2")) install.packages("ggplot2")
library(ggplot2)

# Assumo che il tuo dataframe si chiami train_data

# Modello per Item_MRP
train_data_MRP <- na.omit(train_data[, c("Item_MRP", "Item_Outlet_Sales")])  # Omette NA solo nelle col
model_Item_MRP <- lm(Item_Outlet_Sales ~ Item_MRP, data = train_data_MRP)

# Modello per Outlet_Size
train_data_Size <- na.omit(train_data[, c("Outlet_Size", "Item_Outlet_Sales")])
model_Outlet_Size <- lm(Item_Outlet_Sales ~ Outlet_Size, data = train_data_Size)

# Modello per Item_Visibility
train_data_Visibility <- na.omit(train_data[, c("Item_Visibility", "Item_Outlet_Sales")])
model_Item_Visibility <- lm(Item_Outlet_Sales ~ Item_Visibility, data = train_data_Visibility)

# Modello per Outlet_Age
train_data_Age <- na.omit(train_data[, c("Outlet_Age", "Item_Outlet_Sales")])
model_Outlet_Age <- lm(Item_Outlet_Sales ~ Outlet_Age, data = train_data_Age)
```

```r
# Stampa i riepiloghi dei modelli
cat("\nRiepilogo del modello per Item MRP:\n")
```

```
##
## Riepilogo del modello per Item MRP:
```

```r
print(summary(model_Item_MRP))
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_MRP, data = train_data_MRP)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3893.2  -773.4   -55.8   709.5  9427.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -36.5803    42.0658   -0.87    0.385
## Item_MRP     15.7289     0.2728   57.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1406 on 6818 degrees of freedom
## Multiple R-squared:  0.3277, Adjusted R-squared:  0.3276
## F-statistic:  3324 on 1 and 6818 DF,  p-value: < 2.2e-16
```

```r
cat("\nRiepilogo del modello per Outlet Size:\n")
```

```
##
## Riepilogo del modello per Outlet Size:
```

```r
print(summary(model_Outlet_Size))
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Outlet_Size, data = train_data_Size)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2599.7 -1307.3  -436.4   907.8 10418.0
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1610.90      35.14   45.84   <2e-16 ***
## Outlet_Size2  1058.06      50.09   21.12   <2e-16 ***
## Outlet_Size3   715.70      71.26   10.04   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1694 on 5323 degrees of freedom
## Multiple R-squared:  0.0787, Adjusted R-squared:  0.07836
## F-statistic: 227.4 on 2 and 5323 DF,  p-value: < 2.2e-16
```

```r
cat("\nRiepilogo del modello per Item Visibility:\n")
```

```
##
## Riepilogo del modello per Item Visibility:
```

```r
print(summary(model_Item_Visibility))
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility, data = train_data_Visibility)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2376.4 -1293.9  -412.6   890.6 10616.1
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2522.4       36.1   69.86   <2e-16 ***
## Item_Visibility  -4883.8      419.3  -11.65   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1699 on 6725 degrees of freedom
## Multiple R-squared:  0.01977,    Adjusted R-squared:  0.01963
## F-statistic: 135.7 on 1 and 6725 DF,  p-value: < 2.2e-16
```

```r
cat("\nRiepilogo del modello per Outlet Age:\n")
```

```
##
## Riepilogo del modello per Outlet Age:
```

```r
print(summary(model_Outlet_Age))
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Outlet_Age, data = train_data_Age)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2289.1 -1332.3  -368.9   905.3 10763.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1888.660     67.838  27.841  < 2e-16 ***
## Outlet_Age    11.155      2.464   4.527 6.07e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1712 on 6818 degrees of freedom
## Multiple R-squared:  0.002997,   Adjusted R-squared:  0.002851
## F-statistic:  20.5 on 1 and 6818 DF,  p-value: 6.074e-06
```

```r
# Assumo che il tuo dataframe si chiami test_data

# Modello per Item_MRP sul test set
test_data_MRP <- na.omit(test_data[, c("Item_MRP", "Item_Outlet_Sales")])
predictions_MRP <- predict(model_Item_MRP, newdata = test_data_MRP)

# Modello per Outlet_Size sul test set
test_data_Size <- na.omit(test_data[, c("Outlet_Size", "Item_Outlet_Sales")])
predictions_Size <- predict(model_Outlet_Size, newdata = test_data_Size)

# Modello per Item_Visibility sul test set
test_data_Visibility <- na.omit(test_data[, c("Item_Visibility", "Item_Outlet_Sales")])
predictions_Visibility <- predict(model_Item_Visibility, newdata = test_data_Visibility)

# Modello per Outlet_Age sul test set
test_data_Age <- na.omit(test_data[, c("Outlet_Age", "Item_Outlet_Sales")])
predictions_Age <- predict(model_Outlet_Age, newdata = test_data_Age)

# Calcolo delle metriche di performance (RMSE e R-squared) per ogni modello
library(Metrics)

# RMSE
rmse_MRP <- rmse(test_data_MRP$Item_Outlet_Sales, predictions_MRP)
rmse_Size <- rmse(test_data_Size$Item_Outlet_Sales, predictions_Size)
rmse_Visibility <- rmse(test_data_Visibility$Item_Outlet_Sales, predictions_Visibility)
rmse_Age <- rmse(test_data_Age$Item_Outlet_Sales, predictions_Age)

# R-squared
r2_MRP <- summary(lm(Item_Outlet_Sales ~ predictions_MRP, data = test_data_MRP))$r.squared
r2_Size <- summary(lm(Item_Outlet_Sales ~ predictions_Size, data = test_data_Size))$r.squared
r2_Visibility <- summary(lm(Item_Outlet_Sales ~ predictions_Visibility, data = test_data_Visibility))$r
r2_Age <- summary(lm(Item_Outlet_Sales ~ predictions_Age, data = test_data_Age))$r.squared

# Stampa dei risultati
print("Performance Metrics for Each Variable on the Test Set:")
```

```
## [1] "Performance Metrics for Each Variable on the Test Set:"
```

```r
print(data.frame(
  Variable = c("Item_MRP", "Outlet_Size", "Item_Visibility", "Outlet_Age"),
  RMSE = c(rmse_MRP, rmse_Size, rmse_Visibility, rmse_Age),
  R_Squared = c(r2_MRP, r2_Size, r2_Visibility, r2_Age)
))
```

```
##          Variable     RMSE    R_Squared
## 1        Item_MRP 1402.674 0.2996302881
## 2     Outlet_Size 1648.759 0.0861579380
## 3 Item_Visibility 1663.209 0.0149598009
## 4      Outlet_Age 1674.936 0.0006534232
```

scrivi il chunk in cui vengono comparate le performance del test e del train.

- Item_MRP è la variabile più predittiva tra quelle considerate, spiegando circa il 30% della varianza
- Outlet_Size ha una capacità predittiva molto bassa ma potrebbe ancora fornire qualche informazione
- Item_Visibility e Outlet_Age hanno una capacità predittiva trascurabile e probabilmente non aggiung

Selezione del miglior modello con la stepwise regression

```
if (!require(caret)) install.packages("caret")
if (!require(MASS)) install.packages("MASS")
library(caret)
library(MASS)

# Identificazione e rimozione delle variabili categoriche
train_data <- train_data[, sapply(train_data, is.numeric)]
test_data <- test_data[, sapply(test_data, is.numeric)]
test_data <- na.omit(test_data)  # Rimuove tutte le righe con NA
train_data <- na.omit(train_data)
# Modello iniziale con le variabili numeriche rimanenti
fit_initial <- lm(Item_Outlet_Sales ~ ., data = train_data)

# Selezione delle variabili usando il metodo backward
step_model <- stepAIC(fit_initial, direction = "backward", trace = FALSE)

# Stampa del modello finale dopo la selezione backward
print(summary(step_model))
```

```
##
## Call:
## lm(formula = Item_Outlet_Sales ~ Item_Visibility + Item_MRP +
##     Outlet_Establishment_Year, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4178.5  -779.0   -56.1   685.6  8967.7
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 28175.1365  4009.2576   7.028 2.31e-12 ***
## Item_Visibility             -4849.8105   342.4139 -14.164  < 2e-16 ***
## Item_MRP                       15.6788     0.2703  57.995  < 2e-16 ***
## Outlet_Establishment_Year     -13.9492     2.0061  -6.954 3.90e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1384 on 6723 degrees of freedom
## Multiple R-squared:  0.3495, Adjusted R-squared:  0.3493
## F-statistic:  1204 on 3 and 6723 DF,  p-value: < 2.2e-16
```

```
# Previsione sul test set utilizzando il modello selezionato
predictions <- predict(step_model, newdata = test_data)

# Calcolo delle metriche di performance sul test set
actuals <- test_data$Item_Outlet_Sales
rmse_value <- sqrt(mean((predictions - actuals)^2))
```

```r
r_squared <- summary(lm(actuals ~ predictions))$r.squared

# Stampa delle metriche di performance
cat("Performance Metrics on Test Set:\n")
```

```
## Performance Metrics on Test Set:
```

```r
cat("RMSE:", rmse_value, "\n")
```

```
## RMSE: 1384.88
```

```r
cat("R-squared:", r_squared, "\n")
```

```
## R-squared: 0.3177471
```

```r
# Puoi anche desiderare di vedere come il modello si comporta sul set di training per confronto
train_predictions <- predict(step_model, newdata = train_data)
train_actuals <- train_data$Item_Outlet_Sales
train_rmse <- sqrt(mean((train_predictions - train_actuals)^2))
train_r_squared <- summary(lm(train_actuals ~ train_predictions))$r.squared

# Stampa delle metriche di performance sul train set
cat("Performance Metrics on Train Set:\n")
```

```
## Performance Metrics on Train Set:
```

```r
cat("RMSE:", train_rmse, "\n")
```

```
## RMSE: 1383.417
```

```r
cat("R-squared:", train_r_squared, "\n")
```

```
## R-squared: 0.3495447
```

Il modello migliore include Item Visibility, Item_MRP e Outlet_Establishment_Year ed il modello ha un punteggio relativamente basso, per migliorarlo, procediamo con la creazione di nuove features.

Ad esempio andiamo a creare le variabili: