Food Kiosk EDA

Manjusha Motamarry

2024-03-16

```
library(readr)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(modelr)
```

```
#Loading datasets
sold <- read.csv("/Users/mgmanjusha/Documents/NEU/Sem-1/IDMP/Cafe+-+Sell+Meta+Data.csv",
                 na = "NULL")

transaction <- read_csv("/Users/mgmanjusha/Documents/NEU/Sem-1/IDMP/Cafe+-+Transaction+-+Store.csv", na
```

```
## Rows: 5404 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): CALENDAR_DATE
## dbl (4): PRICE, QUANTITY, SELL_ID, SELL_CATEGORY
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
dates <- read_csv("/Users/mgmanjusha/Documents/NEU/Sem-1/IDMP/Cafe+-+DateInfo.csv",
                  na = "NULL")
```

```
## Rows: 1349 Columns: 7
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr (2): CALENDAR_DATE, HOLIDAY
## dbl (5): YEAR, IS_WEEKEND, IS_SCHOOLBREAK, AVERAGE_TEMPERATURE, IS_OUTDOOR
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

SOLD Dataset

```
#SOLD
print(sold)
```

```
##    SELL_ID SELL_CATEGORY ITEM_ID ITEM_NAME
## 1     1070             0    7821    BURGER
## 2     3055             0    3052    COFFEE
## 3     3067             0    5030      COKE
## 4     3028             0    6249  LEMONADE
## 5     2051             2    7821    BURGER
## 6     2051             2    5030      COKE
## 7     2052             2    7821    BURGER
## 8     2052             2    6249  LEMONADE
## 9     2053             2    7821    BURGER
## 10    2053             2    5030      COKE
## 11    2053             2    3052    COFFEE
```

```
summary(sold)
```

```
##     SELL_ID     SELL_CATEGORY      ITEM_ID      ITEM_NAME
##  Min.   :1070   Min.   :0.000   Min.   :3052   Length:11
##  1st Qu.:2052   1st Qu.:0.000   1st Qu.:5030   Class :character
##  Median :2053   Median :2.000   Median :6249   Mode  :character
##  Mean   :2235   Mean   :1.273   Mean   :5907
##  3rd Qu.:2540   3rd Qu.:2.000   3rd Qu.:7821
##  Max.   :3067   Max.   :2.000   Max.   :7821
```

```
#Checking for null values
any(is.na(sold))
```

```
## [1] FALSE
```

Information about Sold df: SELL_ID: a categorical variable, identifier of the combination of items that is contained in the product.

SELL_CATEGORY: “0” identifies single products; the category “2” identifies the combo ones.

ITEM_ID: a categorical variable, identifier of the item that is contained in the product 1-to-1 relation with item_name.

ITEM_NAME: a categorical variable, identifying the name of the item

TRANSACTION Dataset

```
#TRANSACTION

transaction <- transaction |>
  mutate(
    # Convert CALENDAR_DATE to a date object, add 7 years, and then format it back to a character strin
```

```
    CALENDAR_DATE = as.character(as.Date(CALENDAR_DATE, format = "%m/%d/%y") %m+% years(7))
  )

head(transaction)
```

```
## # A tibble: 6 x 5
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY
##   <chr>         <dbl>    <dbl>   <dbl>         <dbl>
## 1 2019-01-01     15.5       46    1070             0
## 2 2019-01-01     12.7       22    2051             2
## 3 2019-01-01     12.8       18    2052             2
## 4 2019-01-01     12.6       30    2053             2
## 5 2019-01-02     15.5       70    1070             0
## 6 2019-01-02     12.7       22    2051             2
```

```
summary(transaction)
```

```
##  CALENDAR_DATE          PRICE          QUANTITY         SELL_ID
##  Length:5404        Min.   :10.12   Min.   :  8.00   Min.   :1070
##  Class :character   1st Qu.:11.53   1st Qu.: 24.00   1st Qu.:1806
##  Mode  :character   Median :12.64   Median : 36.00   Median :2052
##                     Mean   :12.87   Mean   : 44.34   Mean   :1806
##                     3rd Qu.:13.56   3rd Qu.: 60.00   3rd Qu.:2052
##                     Max.   :16.50   Max.   :124.00   Max.   :2053
##  SELL_CATEGORY
##  Min.   :0.0
##  1st Qu.:1.5
##  Median :2.0
##  Mean   :1.5
##  3rd Qu.:2.0
##  Max.   :2.0
```

```
#Checking for null values
any(is.na(transaction))
```

```
## [1] FALSE
```

Information about Transaction df: Important: It's supposed the PRICE for that product in that day will not vary.

In details: CALENDAR_DATE: a date/time variable, having the time always set to 00:00 AM.

PRICE: a numeric variable, associated with the price of the product identified by the SELL_ID.

QUANTITY: a numeric variable, associated with the quantity of the product sold, identified by the SELL_ID.

SELL_ID: a categorical variable, identifier of the product sold.

SELL_CATEGORY: a categorical variable, category of the product sold.

DATES Dataset

```
#DATE INFO

dates <- dates %>%
  mutate(
    # Add 7 years to the YEAR column
    YEAR = YEAR + 7,
```

```
    CALENDAR_DATE = as.character(as.Date(CALENDAR_DATE, format = "%m/%d/%y") %m+% years(7))
  )

dates
```

```
## # A tibble: 1,349 x 7
##    CALENDAR_DATE  YEAR HOLIDAY  IS_WEEKEND IS_SCHOOLBREAK AVERAGE_TEMPERATURE
##    <chr>         <dbl> <chr>         <dbl>          <dbl>               <dbl>
##  1 2019-01-01     2019 New Year          1              0                24.8
##  2 2019-01-02     2019 New Year          0              0                24.8
##  3 2019-01-03     2019 New Year          0              0                32
##  4 2019-01-04     2019 <NA>              0              0                32
##  5 2019-01-05     2019 <NA>              0              0                24.8
##  6 2019-01-06     2019 <NA>              0              0                23
##  7 2019-01-07     2019 <NA>              1              0                26.6
##  8 2019-01-08     2019 <NA>              1              0                26.6
##  9 2019-01-09     2019 <NA>              0              0                23
## 10 2019-01-10     2019 <NA>              0              0                24.8
## # i 1,339 more rows
## # i 1 more variable: IS_OUTDOOR <dbl>
```

```
summary(dates)
```

```
##  CALENDAR_DATE           YEAR        HOLIDAY            IS_WEEKEND
##  Length:1349        Min.   :2019   Length:1349        Min.   :0.0000
##  Class :character   1st Qu.:2019   Class :character   1st Qu.:0.0000
##  Mode  :character   Median :2020   Mode  :character   Median :0.0000
##                     Mean   :2020                      Mean   :0.2854
##                     3rd Qu.:2021                      3rd Qu.:1.0000
##                     Max.   :2022                      Max.   :1.0000
##  IS_SCHOOLBREAK   AVERAGE_TEMPERATURE   IS_OUTDOOR
##  Min.   :0.0000   Min.   :14.00       Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:35.60       1st Qu.:1.0000
##  Median :0.0000   Median :60.80       Median :1.0000
##  Mean   :0.2046   Mean   :56.33       Mean   :0.8621
##  3rd Qu.:0.0000   3rd Qu.:75.20       3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :87.80       Max.   :1.0000
```

```
#Checking for null values
any(is.na(dates))
```

```
## [1] TRUE
```

Since NA values exist in Holiday, we have to deal with them

```
dates[!complete.cases(dates), ]
```

```
## # A tibble: 1,244 x 7
##    CALENDAR_DATE  YEAR HOLIDAY IS_WEEKEND IS_SCHOOLBREAK AVERAGE_TEMPERATURE
##    <chr>         <dbl> <chr>        <dbl>          <dbl>               <dbl>
##  1 2019-01-04     2019 <NA>             0              0                32
##  2 2019-01-05     2019 <NA>             0              0                24.8
##  3 2019-01-06     2019 <NA>             0              0                23
##  4 2019-01-07     2019 <NA>             1              0                26.6
##  5 2019-01-08     2019 <NA>             1              0                26.6
```

```
##  6 2019-01-09     2019 <NA>             0              0                23
##  7 2019-01-10     2019 <NA>             0              0                24.8
##  8 2019-01-11     2019 <NA>             0              0                21.2
##  9 2019-01-12     2019 <NA>             0              0                24.8
## 10 2019-01-13     2019 <NA>             0              0                26.6
## # i 1,234 more rows
## # i 1 more variable: IS_OUTDOOR <dbl>
```

```
#replacing na values with "NO holiday"
dates <- dates %>%
  mutate(HOLIDAY = replace_na(HOLIDAY, "No Holiday"))
dates
```

```
## # A tibble: 1,349 x 7
##    CALENDAR_DATE  YEAR HOLIDAY    IS_WEEKEND IS_SCHOOLBREAK AVERAGE_TEMPERATURE
##    <chr>         <dbl> <chr>           <dbl>          <dbl>               <dbl>
##  1 2019-01-01     2019 New Year            1              0                24.8
##  2 2019-01-02     2019 New Year            0              0                24.8
##  3 2019-01-03     2019 New Year            0              0                32
##  4 2019-01-04     2019 No Holiday          0              0                32
##  5 2019-01-05     2019 No Holiday          0              0                24.8
##  6 2019-01-06     2019 No Holiday          0              0                23
##  7 2019-01-07     2019 No Holiday          1              0                26.6
##  8 2019-01-08     2019 No Holiday          1              0                26.6
##  9 2019-01-09     2019 No Holiday          0              0                23
## 10 2019-01-10     2019 No Holiday          0              0                24.8
## # i 1,339 more rows
## # i 1 more variable: IS_OUTDOOR <dbl>
```

```
summary(dates)
```

```
##  CALENDAR_DATE           YEAR        HOLIDAY            IS_WEEKEND
##  Length:1349        Min.   :2019   Length:1349        Min.   :0.0000
##  Class :character   1st Qu.:2019   Class :character   1st Qu.:0.0000
##  Mode  :character   Median :2020   Mode  :character   Median :0.0000
##                     Mean   :2020                      Mean   :0.2854
##                     3rd Qu.:2021                      3rd Qu.:1.0000
##                     Max.   :2022                      Max.   :1.0000
##  IS_SCHOOLBREAK   AVERAGE_TEMPERATURE   IS_OUTDOOR
##  Min.   :0.0000   Min.   :14.00       Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:35.60       1st Qu.:1.0000
##  Median :0.0000   Median :60.80       Median :1.0000
##  Mean   :0.2046   Mean   :56.33       Mean   :0.8621
##  3rd Qu.:0.0000   3rd Qu.:75.20       3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :87.80       Max.   :1.0000
```

```
#checking for null data again
any(is.na(dates))
```

```
## [1] FALSE
```

```
sold_wide <- sold %>%
  mutate(is_filled = 1) %>%
  pivot_wider(names_from = ITEM_NAME, values_from = is_filled, values_fill = list(is_filled = 0)) %>%
  group_by(SELL_ID, SELL_CATEGORY) %>%
  summarise(
    BURGER = sum(BURGER, na.rm = TRUE),
```

```
    COFFEE = sum(COFFEE, na.rm = TRUE),
    COKE = sum(COKE, na.rm = TRUE),
    LEMONADE = sum(LEMONADE, na.rm = TRUE),
    .groups = 'drop'  # Automatically ungroup after summarise
  ) %>%
  dplyr::select(-SELL_CATEGORY)


head(sold_wide)
```

```
## # A tibble: 6 x 5
##   SELL_ID BURGER COFFEE  COKE LEMONADE
##     <int>  <dbl>  <dbl> <dbl>    <dbl>
## 1    1070      1      0     0        0
## 2    2051      1      0     1        0
## 3    2052      1      0     0        1
## 4    2053      1      1     1        0
## 5    3028      0      0     0        1
## 6    3055      0      1     0        0
```

```
## SOLD + TRANSACTION => MERGED_DATA
merged_data <- inner_join(transaction, sold_wide, by = "SELL_ID")

# View the result
head(merged_data)
```

```
## # A tibble: 6 x 9
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE  COKE
##   <chr>         <dbl>    <dbl>   <dbl>         <dbl>  <dbl>  <dbl> <dbl>
## 1 2019-01-01     15.5       46    1070             0      1      0     0
## 2 2019-01-01     12.7       22    2051             2      1      0     1
## 3 2019-01-01     12.8       18    2052             2      1      0     0
## 4 2019-01-01     12.6       30    2053             2      1      1     1
## 5 2019-01-02     15.5       70    1070             0      1      0     0
## 6 2019-01-02     12.7       22    2051             2      1      0     1
## # i 1 more variable: LEMONADE <dbl>
```

Here, I inner joined sold and transaction on sell_id. This is crucial for a comprehensive analysis where we might need to study transaction details alongside the sales data. By selecting to remove SELL_ID from 'sold' and SELL_CATEGORY from 'transaction', it simplifies the dataset by eliminating irrelevant information for the subsequent analysis steps.

```
kiosk_data <- inner_join(merged_data, dates, by = "CALENDAR_DATE") |>
            dplyr::select(CALENDAR_DATE, PRICE, QUANTITY, SELL_ID, SELL_CATEGORY, BURGER, COFFEE, COKE,
```

```
## Warning in inner_join(merged_data, dates, by = "CALENDAR_DATE"): Detected an unexpected many-to-many
## i Row 1697 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
kiosk_data
```

```
## # A tibble: 5,420 x 15
##    CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE  COKE
##    <chr>         <dbl>    <dbl>   <dbl>         <dbl>  <dbl>  <dbl> <dbl>
##  1 2019-01-01     15.5       46    1070             0      1      0     0
```

```
##  2 2019-01-01     12.7       22    2051              2       1       0      1
##  3 2019-01-01     12.8       18    2052              2       1       0      0
##  4 2019-01-01     12.6       30    2053              2       1       1      1
##  5 2019-01-02     15.5       70    1070              0       1       0      0
##  6 2019-01-02     12.7       22    2051              2       1       0      1
##  7 2019-01-02     12.8       16    2052              2       1       0      0
##  8 2019-01-02     12.6       34    2053              2       1       1      1
##  9 2019-01-03     15.5       62    1070              0       1       0      0
## 10 2019-01-03     12.7       26    2051              2       1       0      1
## # i 5,410 more rows
## # i 7 more variables: LEMONADE <dbl>, YEAR <dbl>, HOLIDAY <chr>,
## #   IS_WEEKEND <dbl>, IS_SCHOOLBREAK <dbl>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <dbl>
```

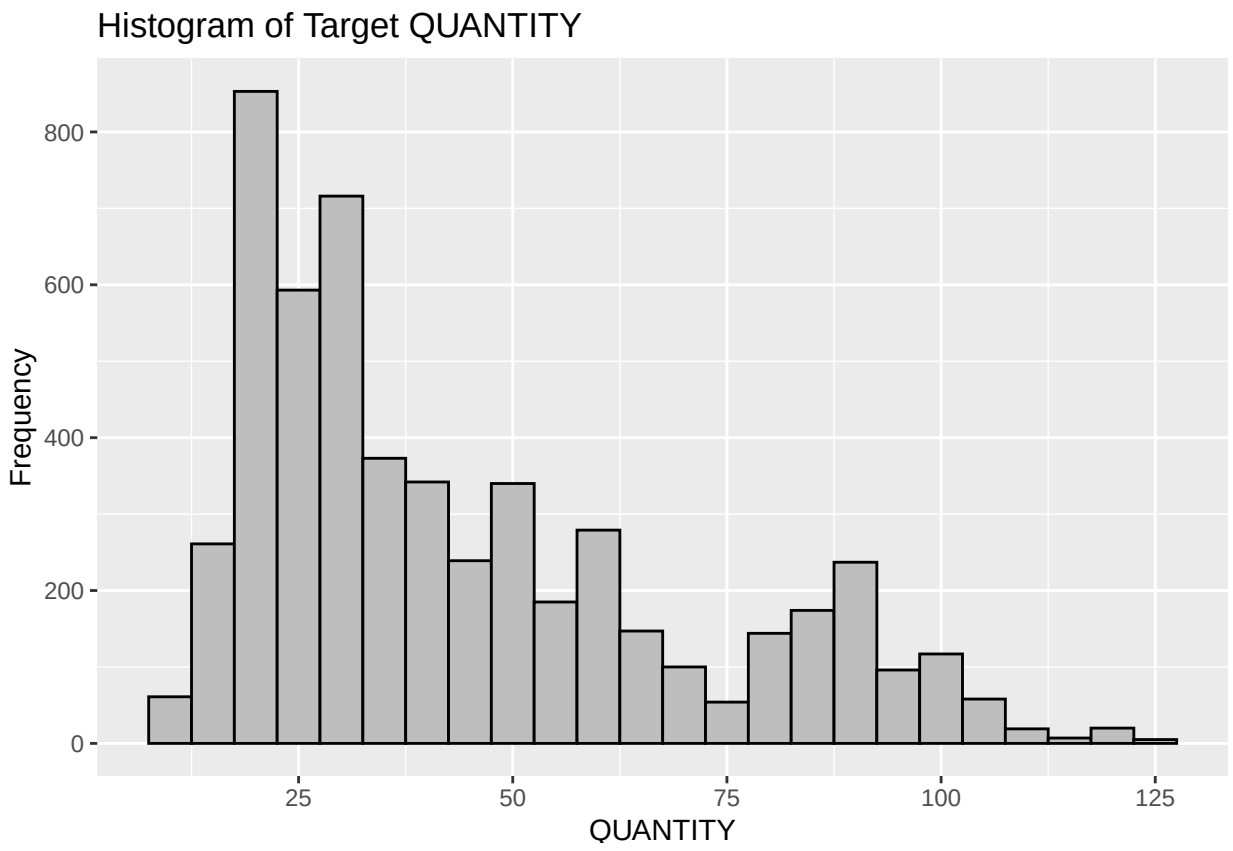
```
#csv
#write_csv(kiosk_data,"Kiosk.csv")
```

Univariate Analysis

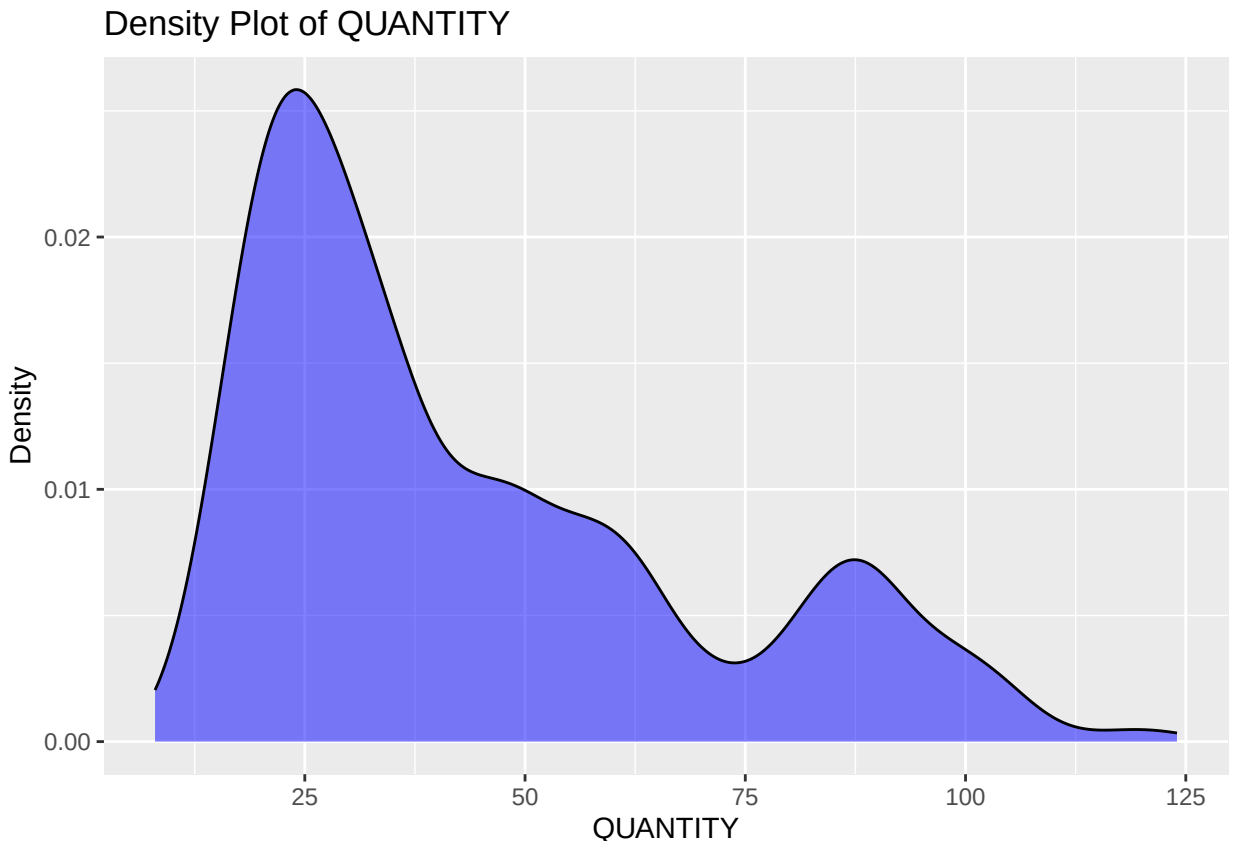
Numerical Variables

QUANTITY

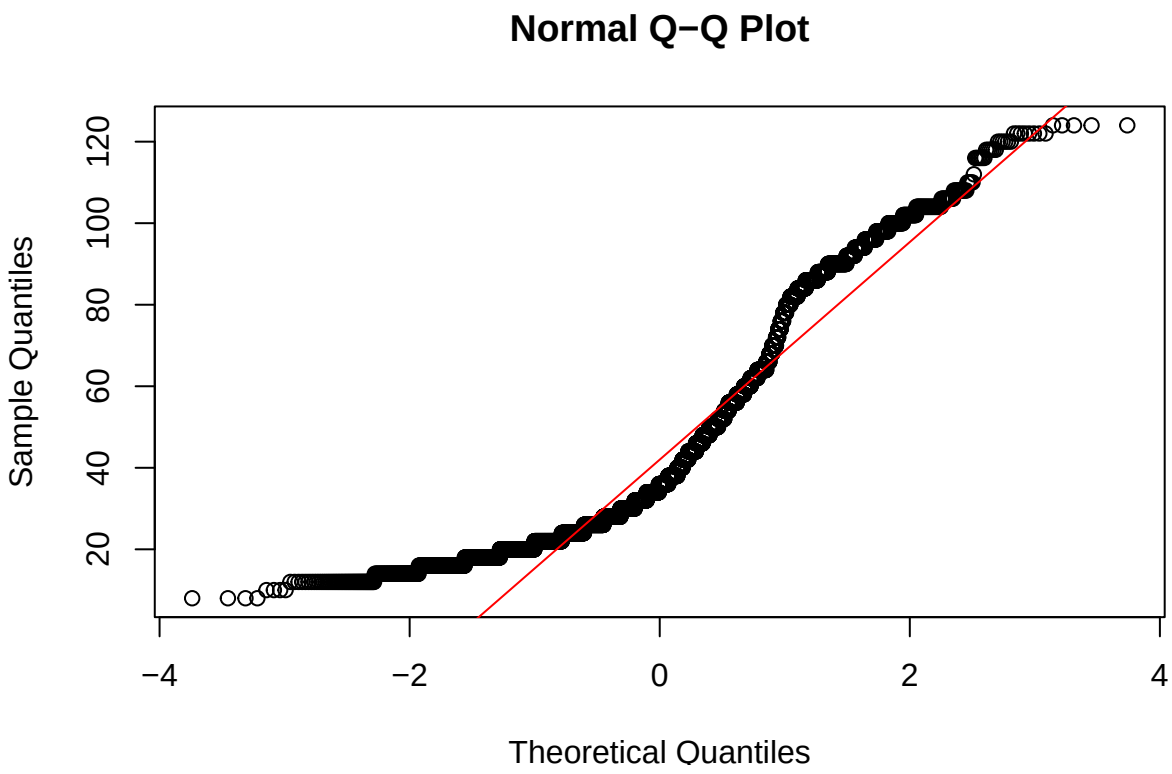
```
# Histogram of QUANTITY
ggplot(kiosk_data, aes(x = QUANTITY)) +
  geom_histogram(binwidth = 5, fill = "grey", color = "black") +
  labs(title = "Histogram of Target QUANTITY", x = "QUANTITY", y = "Frequency")
```



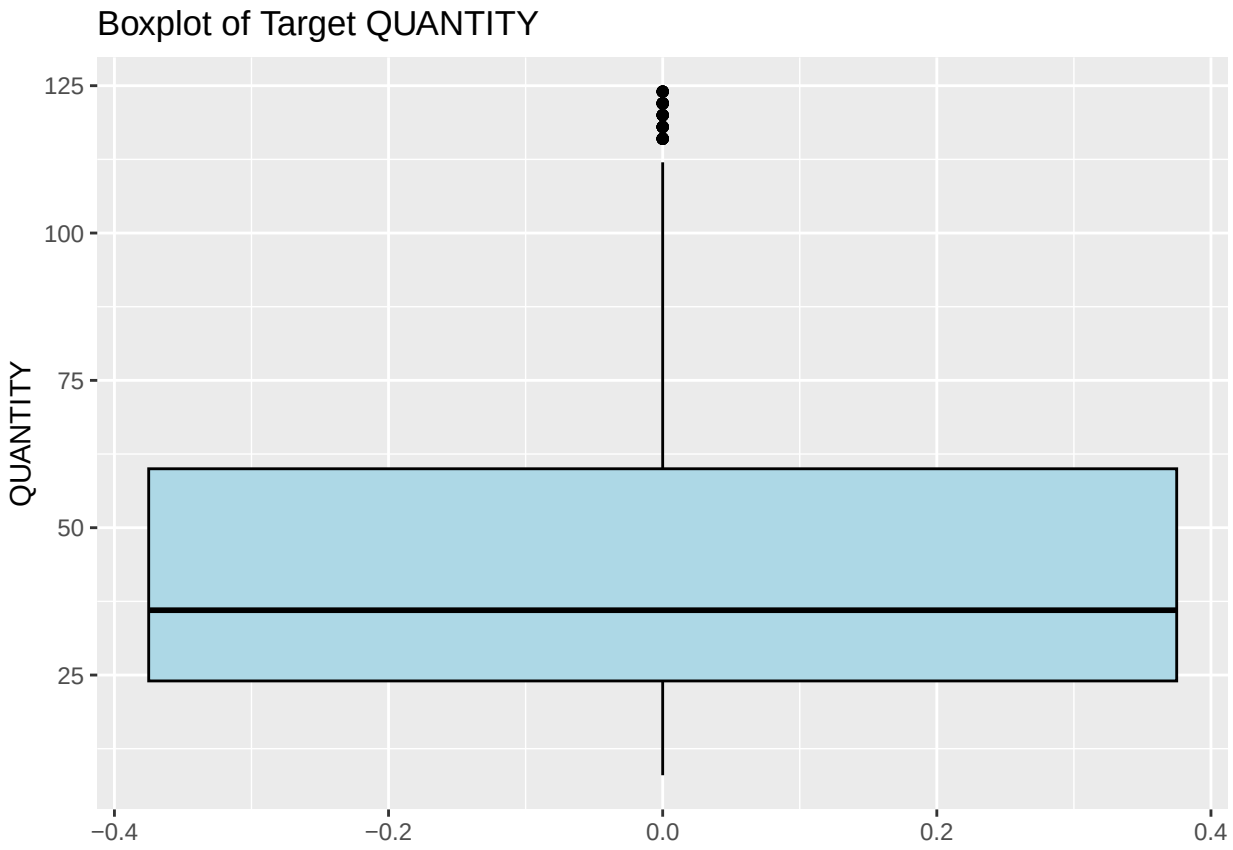
```r
# Density Plot of QUANTITY
ggplot(kiosk_data, aes(x = QUANTITY)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of QUANTITY", x = "QUANTITY", y = "Density")
```




```r
# Q-Q Plot of QUANTITY
qqnorm(kiosk_data$QUANTITY)
qqline(kiosk_data$QUANTITY, col = "red")
```

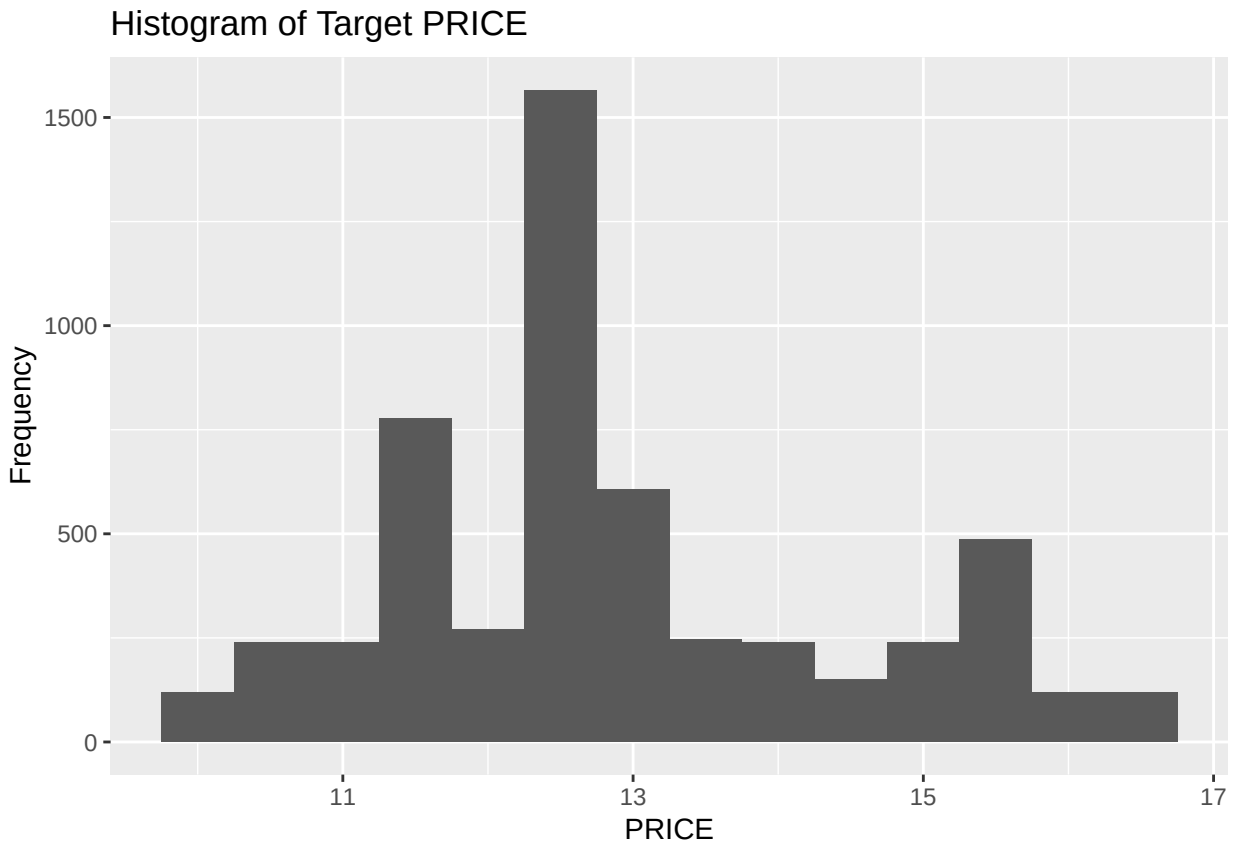



```
# Boxplot of QUANTITY
ggplot(kiosk_data, aes(y = QUANTITY)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Boxplot of Target QUANTITY", y = "QUANTITY")
```

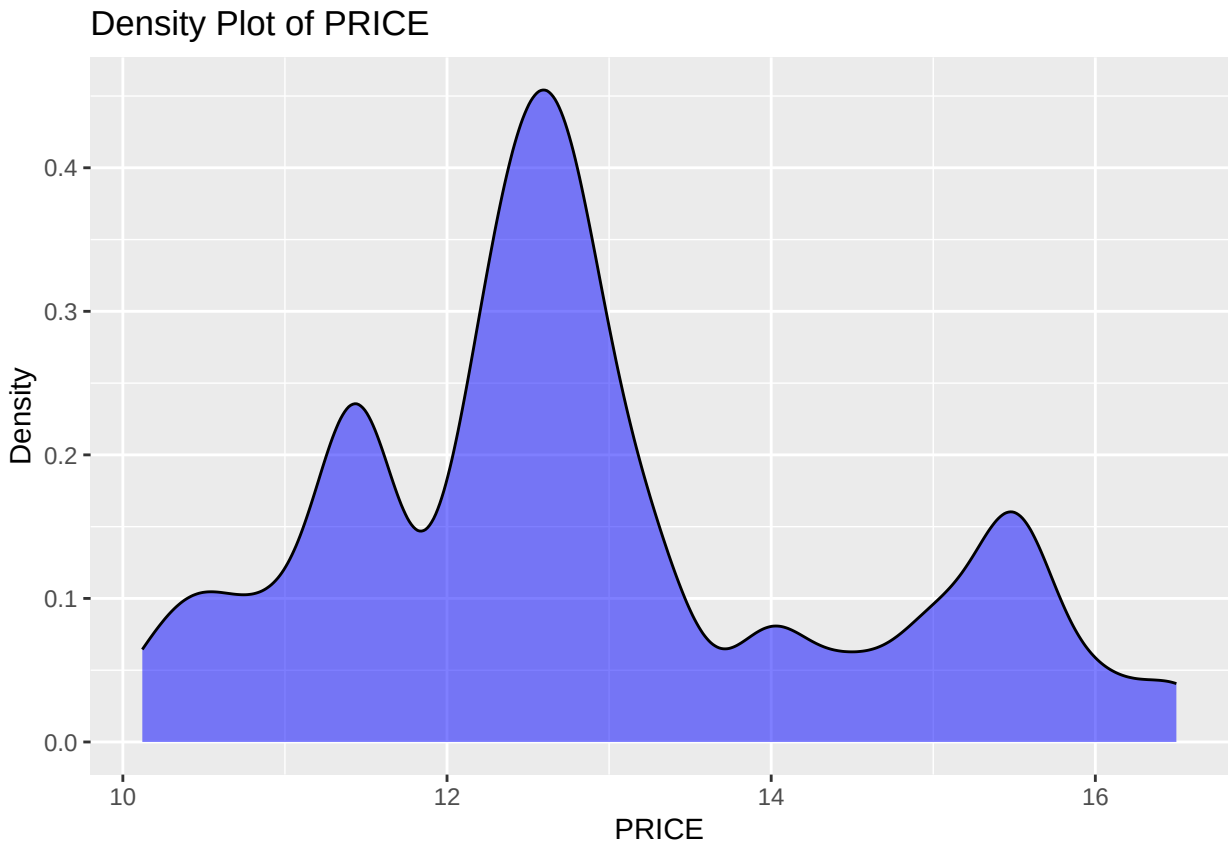


PRICE

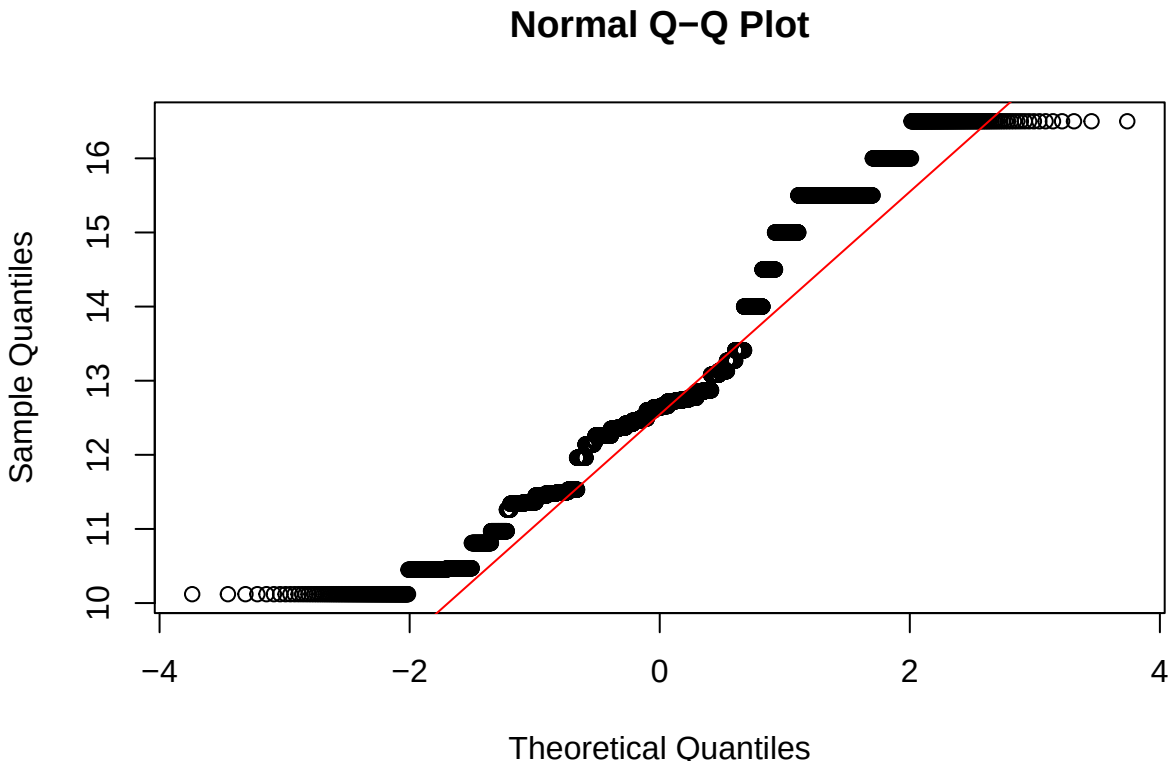
```
# Histogram of QUANTITY
ggplot(kiosk_data, aes(x = PRICE)) +
  geom_histogram(binwidth = 0.5) +
  labs(title = "Histogram of Target PRICE", x = "PRICE", y = "Frequency")
```



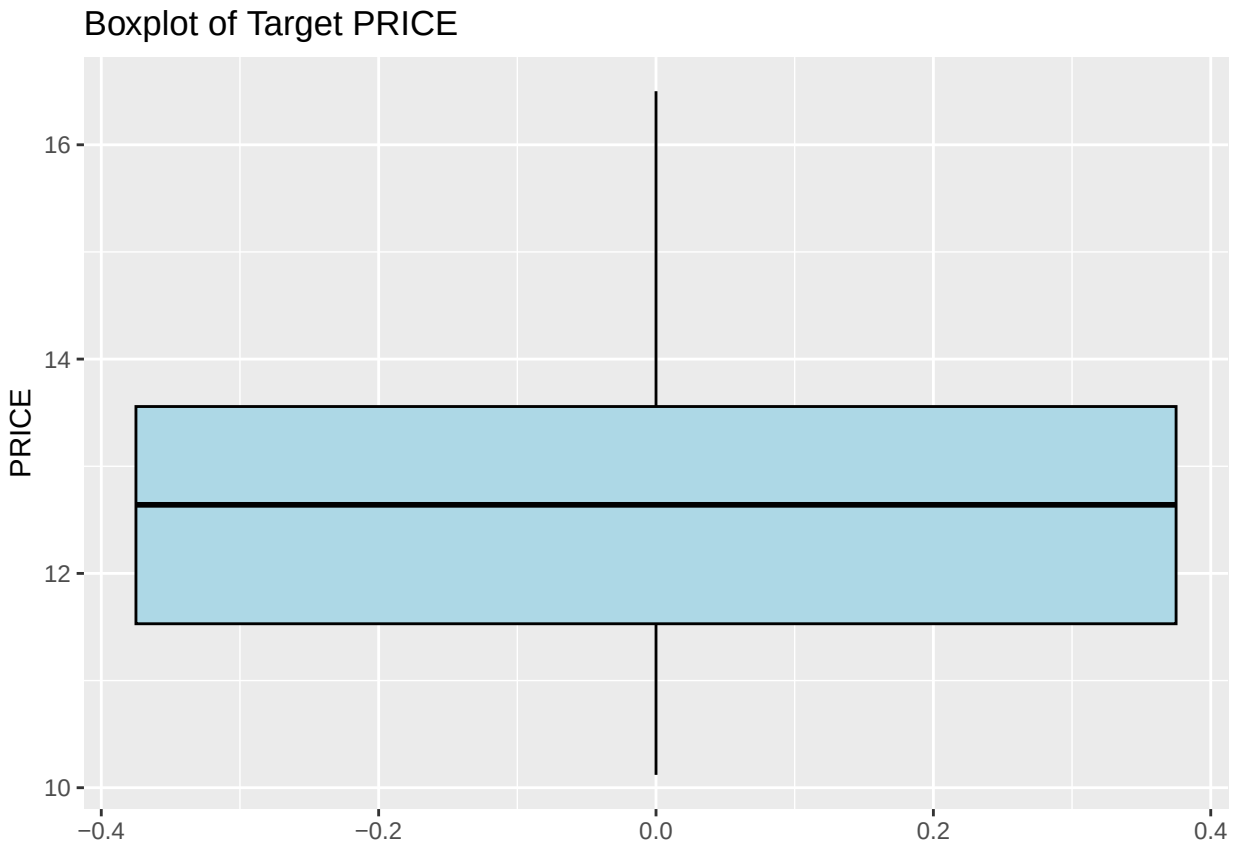

```
# Density Plot of QUANTITY
ggplot(kiosk_data, aes(x = PRICE)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of PRICE", x = "PRICE", y = "Density")
```




```
# Q-Q Plot of QUANTITY
qqnorm(kiosk_data$PRICE)
qqline(kiosk_data$PRICE, col = "red")
```



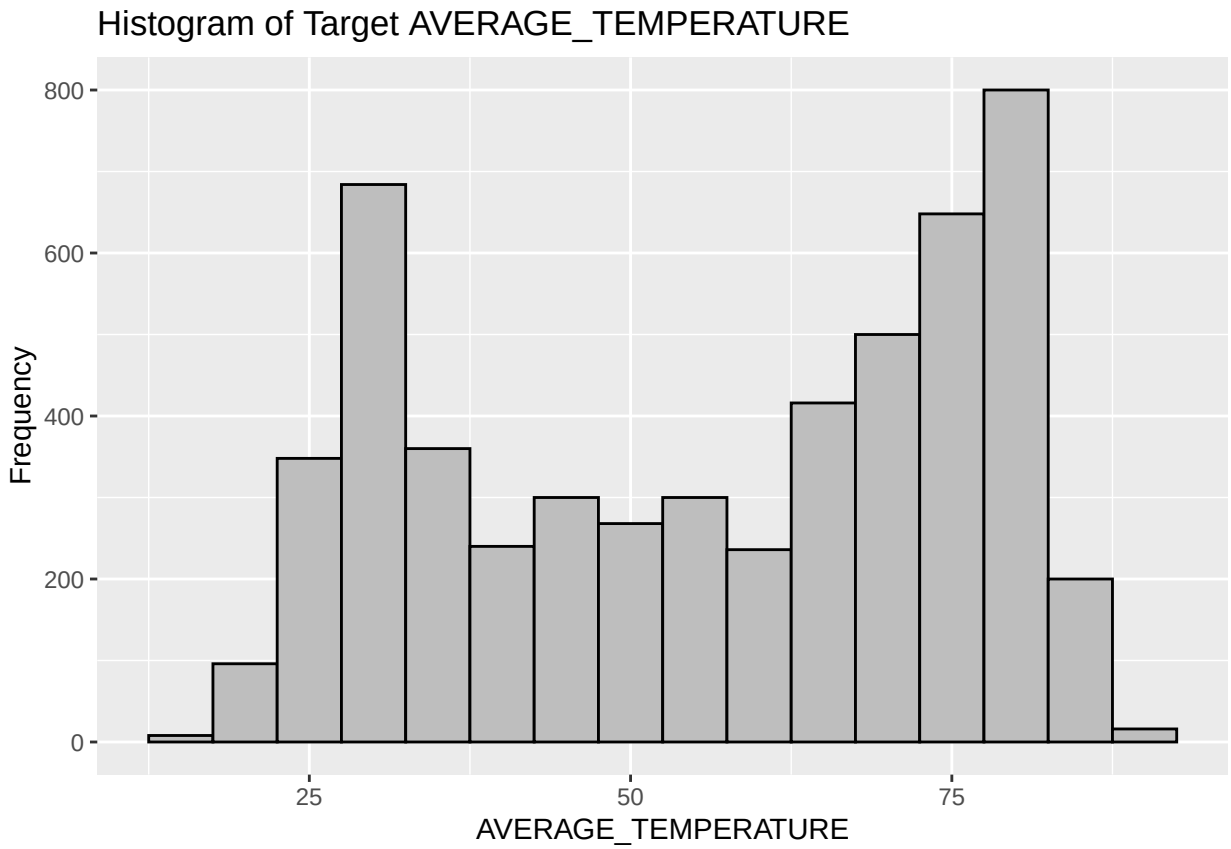
```r
# Boxplot of QUANTITY
ggplot(kiosk_data, aes(y = PRICE)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Boxplot of Target PRICE", y = "PRICE")
```



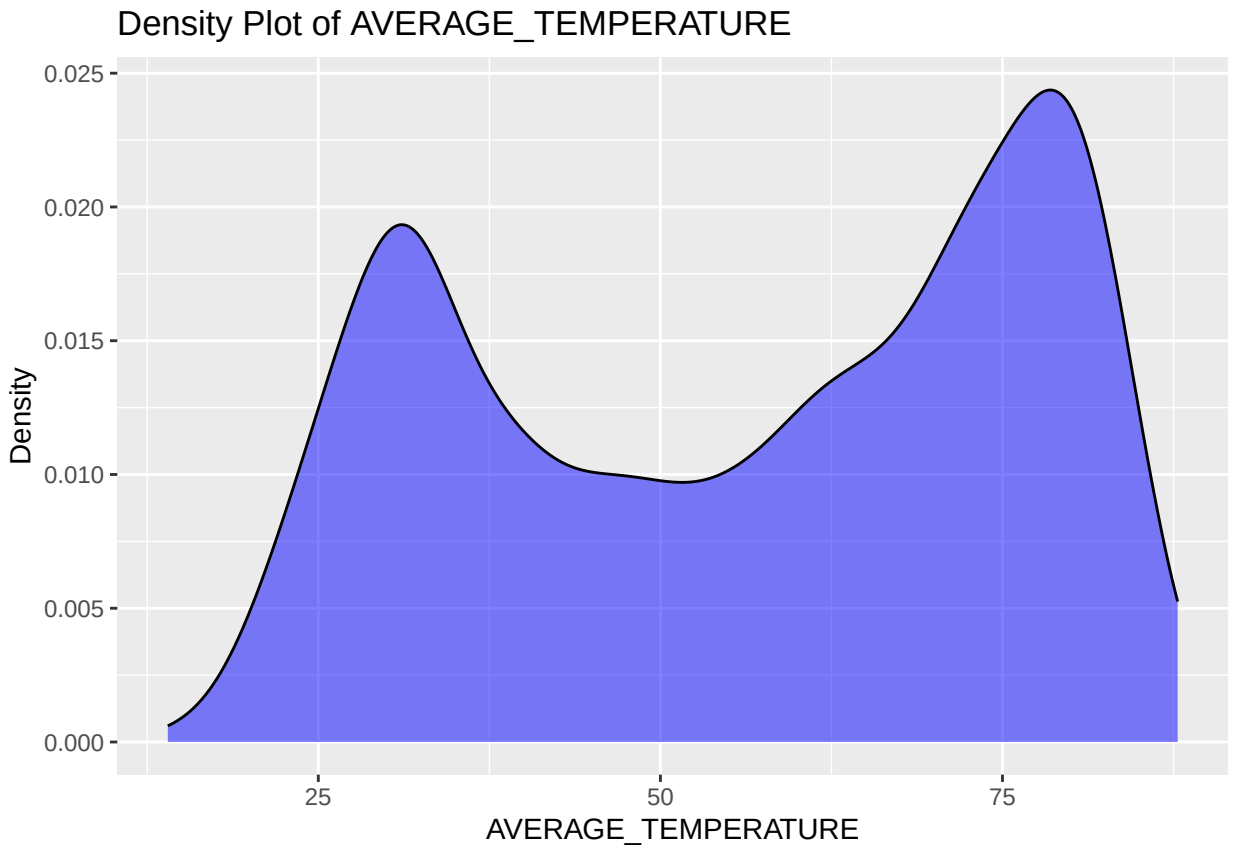
The PRICE variable seems to be bimodal too and just a little bit right-skewed. The boxplot shows no outliers.

AVERAGE TEMPERATURE

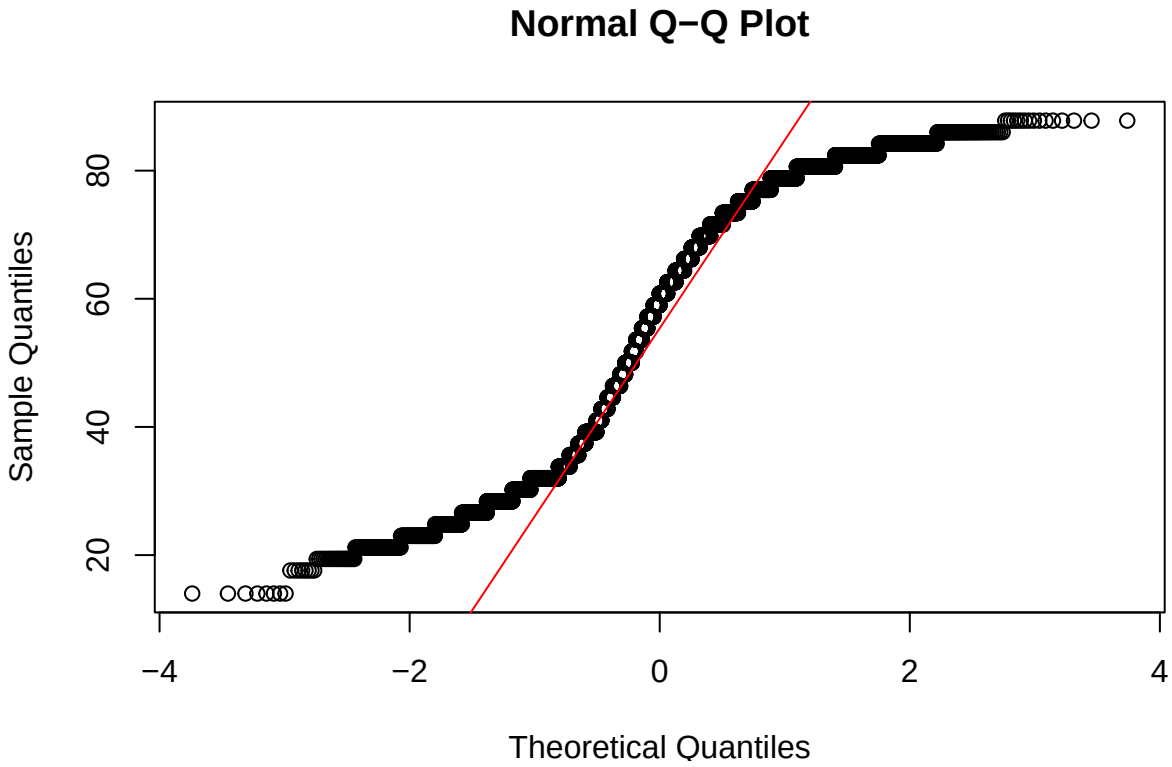
```r
# Histogram of AVERAGE_TEMPERATURE
ggplot(kiosk_data, aes(x = AVERAGE_TEMPERATURE)) +
  geom_histogram(binwidth = 5, fill = "grey", color = "black") +
  labs(title = "Histogram of Target AVERAGE_TEMPERATURE", x = "AVERAGE_TEMPERATURE", y = "Frequency")
```



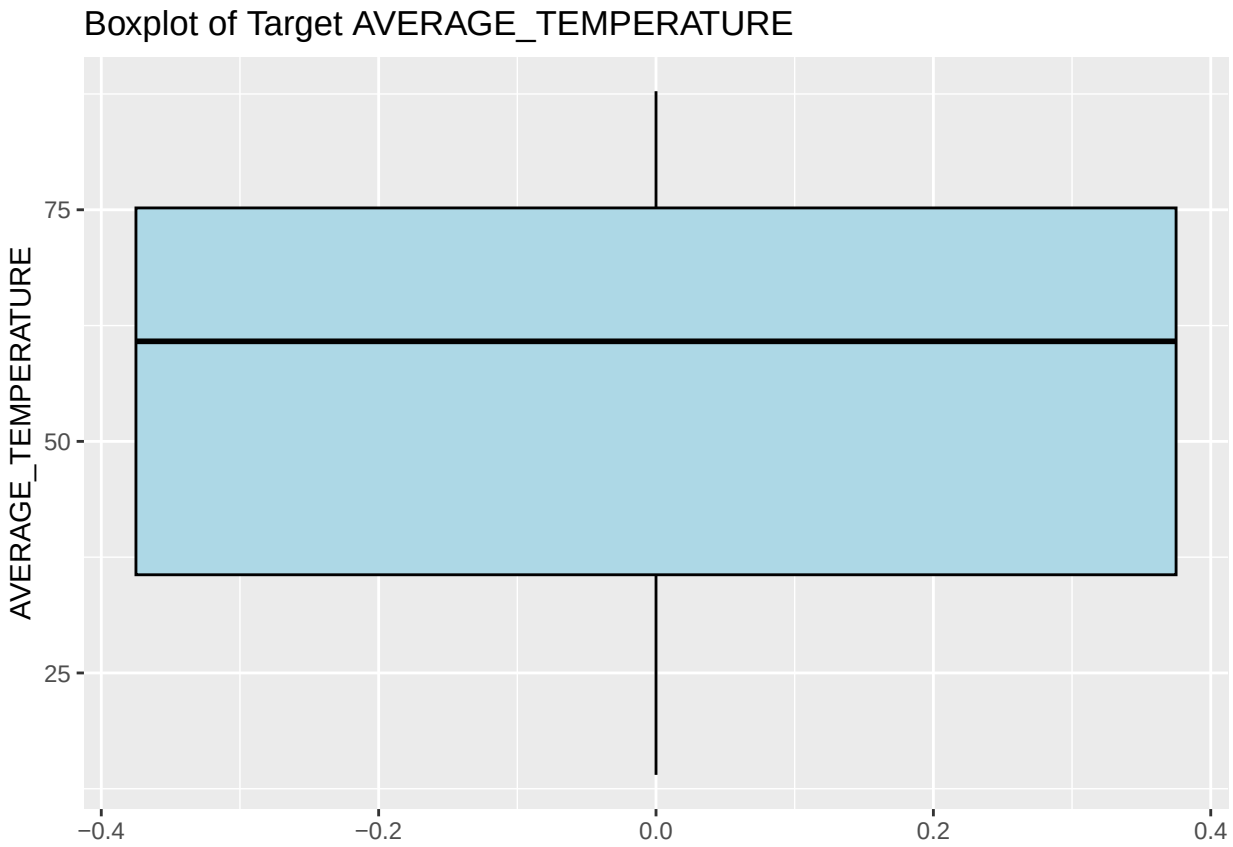

```
# Density Plot of AVERAGE_TEMPERATURE
ggplot(kiosk_data, aes(x = AVERAGE_TEMPERATURE)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(title = "Density Plot of AVERAGE_TEMPERATURE", x = "AVERAGE_TEMPERATURE", y = "Density")
```




```
# Q-Q Plot of AVERAGE_TEMPERATURE
qqnorm(kiosk_data$AVERAGE_TEMPERATURE)
qqline(kiosk_data$AVERAGE_TEMPERATURE, col = "red")
```



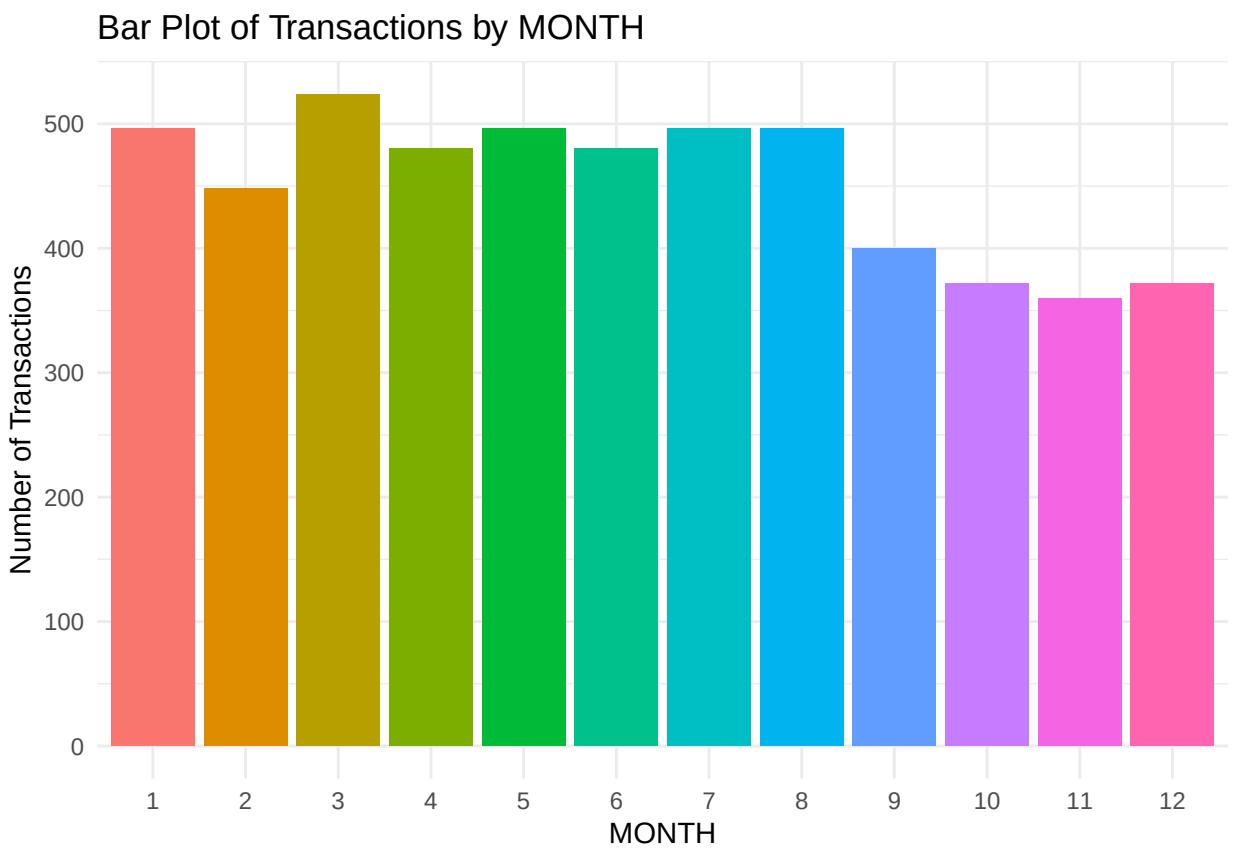
```r
# Boxplot of AVERAGE_TEMPERATURE
ggplot(kiosk_data, aes(y = AVERAGE_TEMPERATURE)) +
  geom_boxplot(fill = "lightblue", color = "black") +
  labs(title = "Boxplot of Target AVERAGE_TEMPERATURE", y = "AVERAGE_TEMPERATURE")
```



The AVERAGE_TEMPERATURE is bimodal too. This time the distribution is a little bit left-skewed.

Categorical variables

```r
kiosk_data %>%
  mutate(MONTH = month(CALENDAR_DATE)) %>%
ggplot(aes(x = as.factor(MONTH), fill = as.factor(MONTH))) +
  geom_bar() +
  labs(title = "Bar Plot of Transactions by MONTH", x = "MONTH", y = "Number of Transactions") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme_minimal() +
  theme(legend.position = "none")
```



Multi-variate Analysis

Numerical variables

1. PRICE
2. TEMPERATURE
3. AVERAGE_TEMPERATURE
4. YEAR

Categorical variables

1. SELL_ID
2. SELL_CATEGORY
3. BURGER
4. COFFEE
5. LEMONADE
6. COKE
7. CALENDAR_DATE
8. IS_WEEKEND
9. IS_OUTDOOR
10. IS_SCHOOLBREAK
11. CALENDAR_DATE

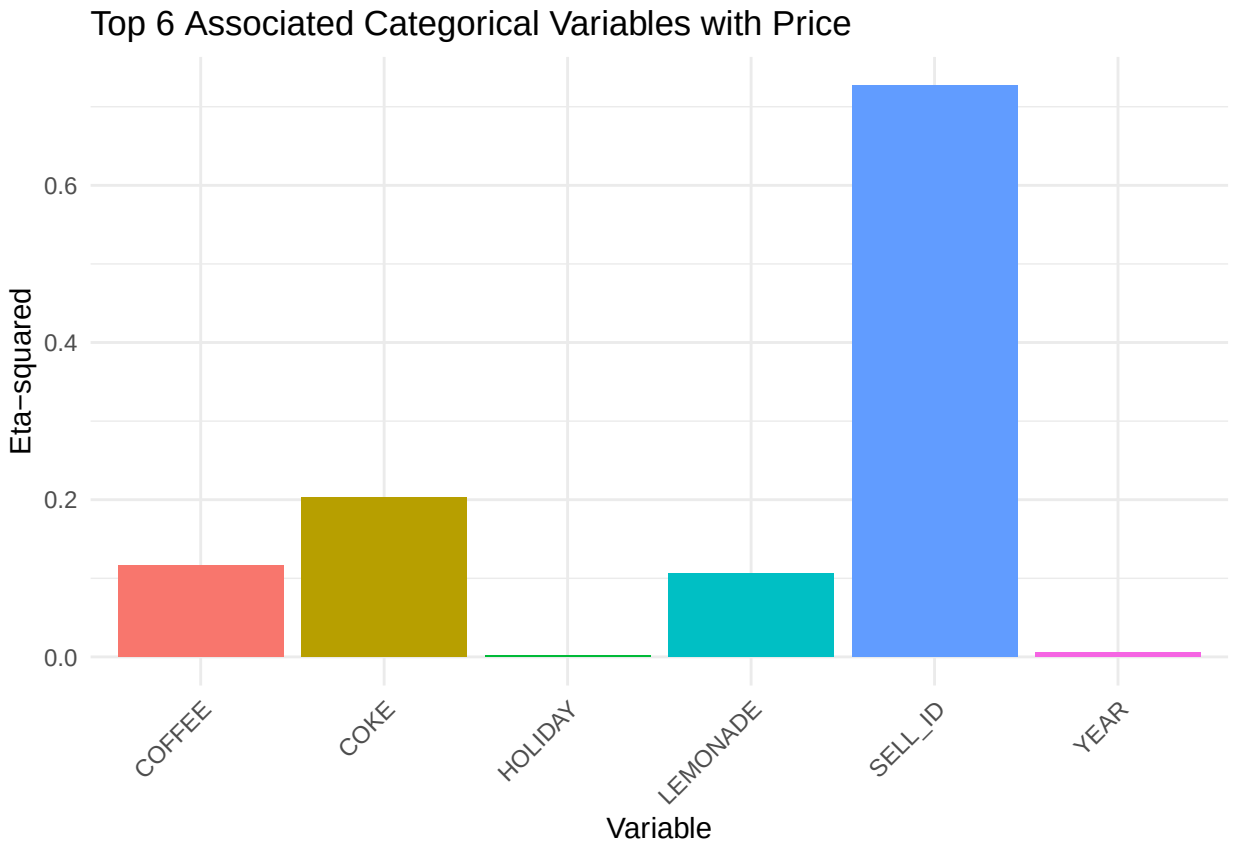
```
numerical_r_squared <- sapply(kiosk_data[, sapply(kiosk_data, is.numeric)], function(x) {
  summary(lm(kiosk_data$QUANTITY ~ x, data = kiosk_data))$r.squared
})
```

```r
top_numerical <- sort(numerical_r_squared, decreasing = TRUE)[1:2]


eta_squared <- function(model) {
  model_sum <- summary(model)
  ss_total <- sum(model_sum[[1]]$SumSq)
  ss_model <- model_sum[[1]]$SumSq[1]
  return(ss_model / ss_total)
}

numerical_var <- "PRICE"

categorical_vars <- c("SELL_ID", "BURGER", "COFFEE", "COKE", "LEMONADE",
                      "YEAR", "HOLIDAY", "IS_WEEKEND", "IS_OUTDOOR")


for(cat_var in categorical_vars) {
  kiosk_data[[cat_var]] <- as.factor(kiosk_data[[cat_var]])
}


categorical_eta_squared <- list()

for(cat_var in categorical_vars) {
  # Ensure the variable is a factor and drop unused levels
  kiosk_data[[cat_var]] <- droplevels(as.factor(kiosk_data[[cat_var]]))

  # Skip the variable if it has less than two levels
  if (length(levels(kiosk_data[[cat_var]])) < 2) {
    next
  }

  formula <- as.formula(paste(numerical_var, cat_var, sep = " ~ "))
  categorical_eta_squared[[cat_var]] <- aov(formula, data = kiosk_data)
}


eta_squared <- sapply(categorical_eta_squared, function(model) {
  # Extract the Sum of Squares for the Model (Effect) and Residuals (Error)
  anova_table <- summary(model)
  ss_model <- anova_table[[1]]$"Sum Sq"[1]  # Sum of Squares for the effect
  ss_total <- sum(anova_table[[1]]$"Sum Sq")  # Total Sum of Squares
  eta_sq <- ss_model / ss_total
  eta_sq
})

top_eta_squared <- sort(eta_squared, decreasing = TRUE)[1:6]

head(top_eta_squared)
```

```
##     SELL_ID        COKE      COFFEE    LEMONADE        YEAR     HOLIDAY
## 0.726951612 0.202979165 0.116917677 0.105770278 0.006245940 0.001671734
```

The associations between categorical and numerical variables are computed using the eta-squared metric.

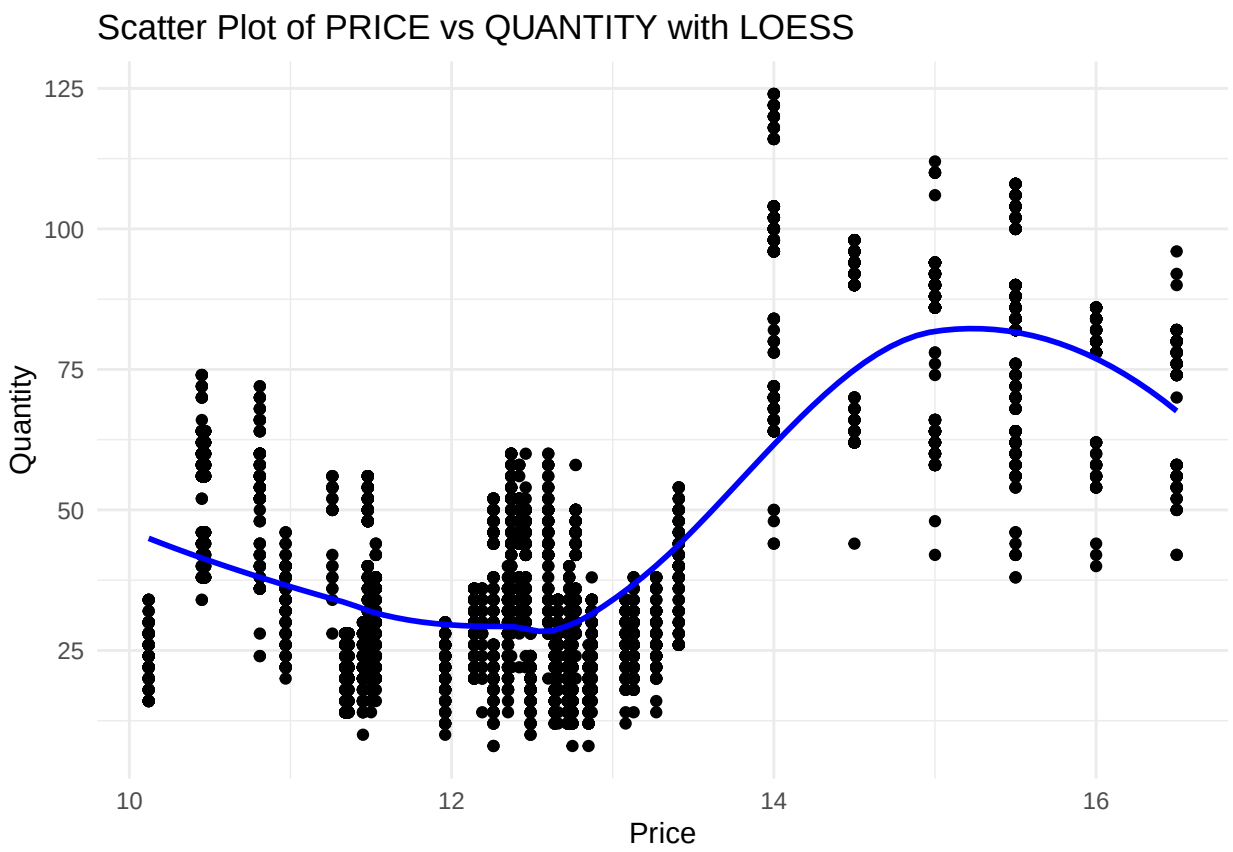
```
df_for_plot <- data.frame(
  Variable = names(top_eta_squared),
  Eta_squared = top_eta_squared
)

ggplot(df_for_plot, aes(x = Variable, y = Eta_squared, fill = Variable)) +
  geom_bar(stat = "identity") +
  labs(title = "Top 6 Associated Categorical Variables with Price", x = "Variable", y = "Eta-squared") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(legend.position = "none")
```



PRICE VS QUANTITY

```
ggplot(kiosk_data, aes(x = PRICE, y = QUANTITY)) +
  geom_point() +  # Plot the raw data as points
  geom_smooth(method = "loess", se = FALSE, color = "blue") +  # Add LOESS line
  labs(title = "Scatter Plot of PRICE vs QUANTITY with LOESS",
       x = "Price",
       y = "Quantity") +
  theme_minimal()
```

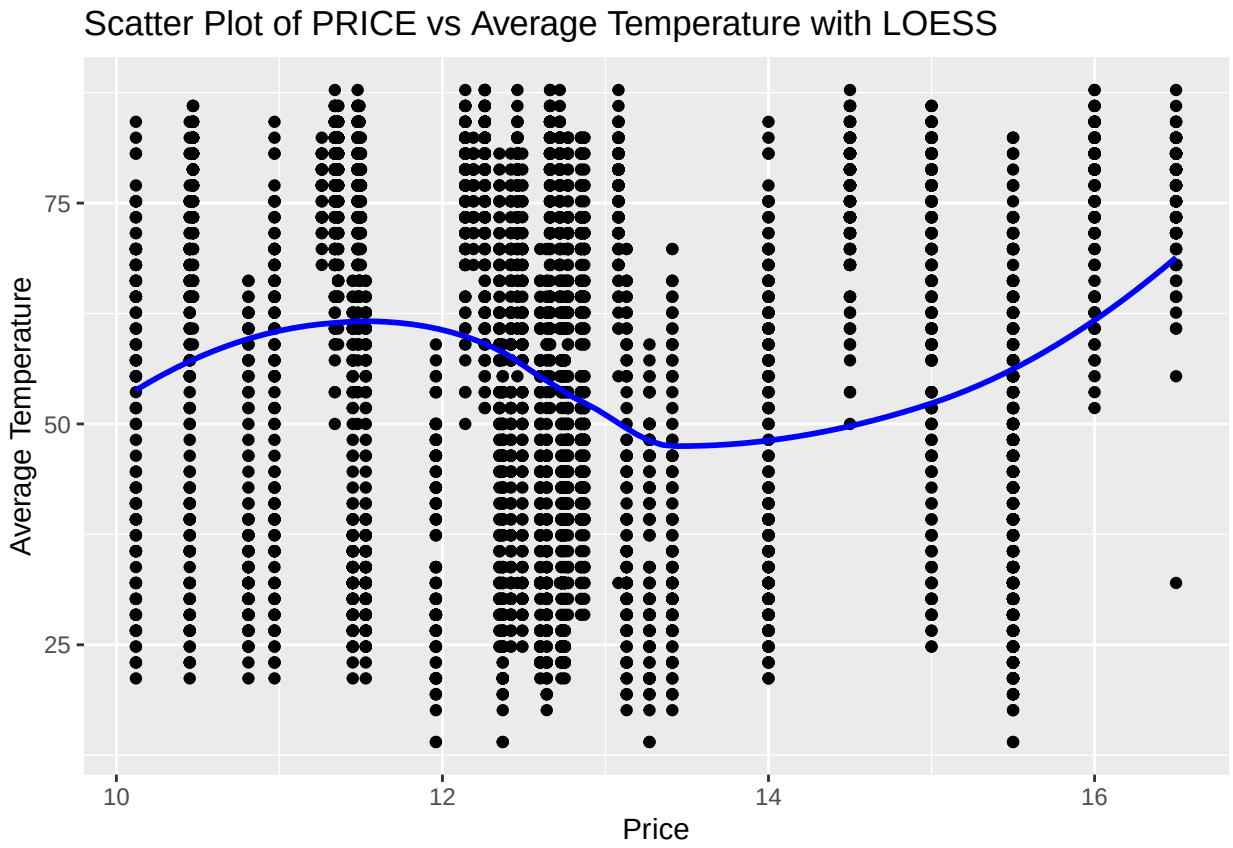
```
## `geom_smooth()` using formula = 'y ~ x'
```



PRICE VS AVERAGE_TEMPERATURE

```
ggplot(kiosk_data, aes(x = PRICE, y = AVERAGE_TEMPERATURE)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue")+
  labs(title = "Scatter Plot of PRICE vs Average Temperature with LOESS",x = "Price",y = "Average Tempe
```

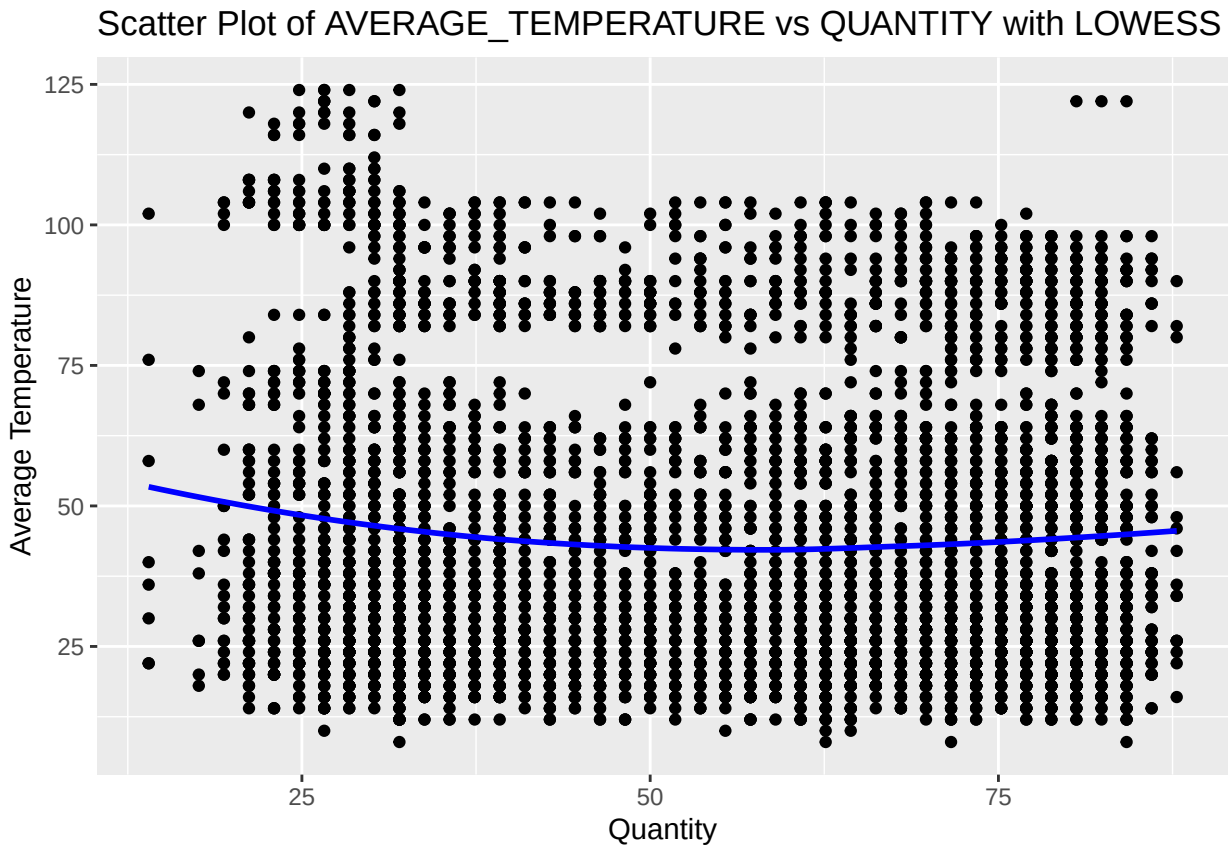
```
## `geom_smooth()` using formula = 'y ~ x'
```



QUANTITY VS AVERAGE_TEMPERATURE

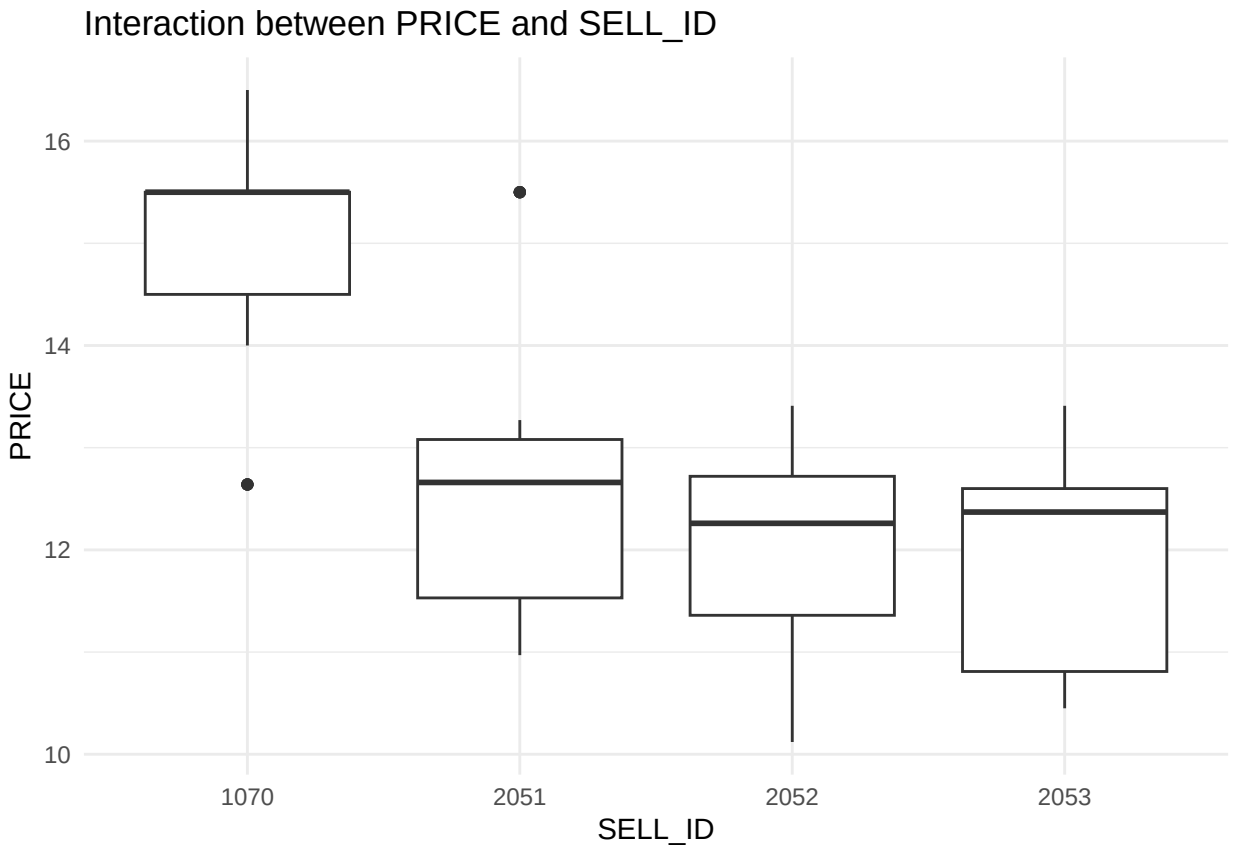
```
ggplot(kiosk_data, aes(x = AVERAGE_TEMPERATURE, y = QUANTITY)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE, color = "blue")+
  labs(title = "Scatter Plot of AVERAGE_TEMPERATURE vs QUANTITY with LOWESS",x = "Quantity",y = "Averag
```

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



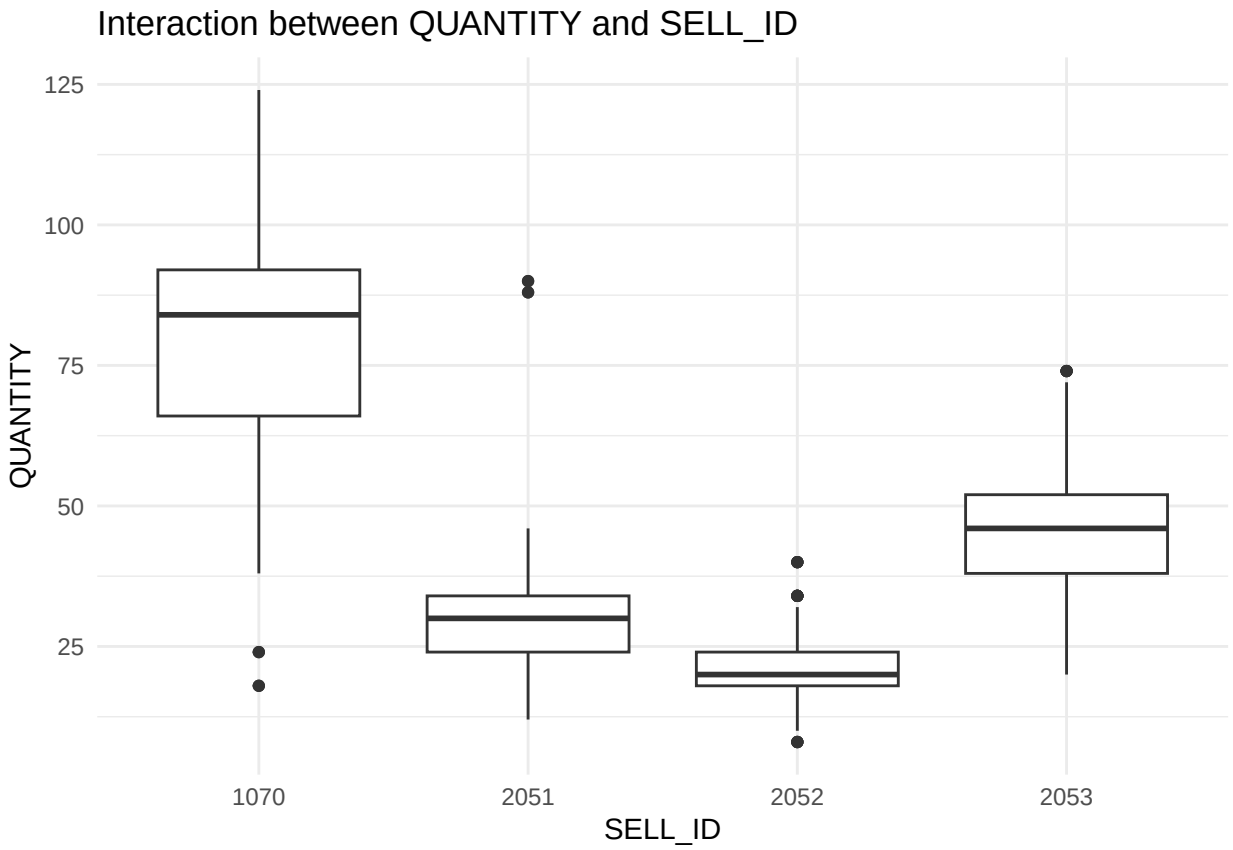
SELL_ID VS PRICE

```
ggplot(kiosk_data, aes(x = factor(SELL_ID), y = PRICE)) +
  geom_boxplot() +
  labs(x = "SELL_ID", y = "PRICE", title = "Interaction between PRICE and SELL_ID") +
  theme_minimal()
```



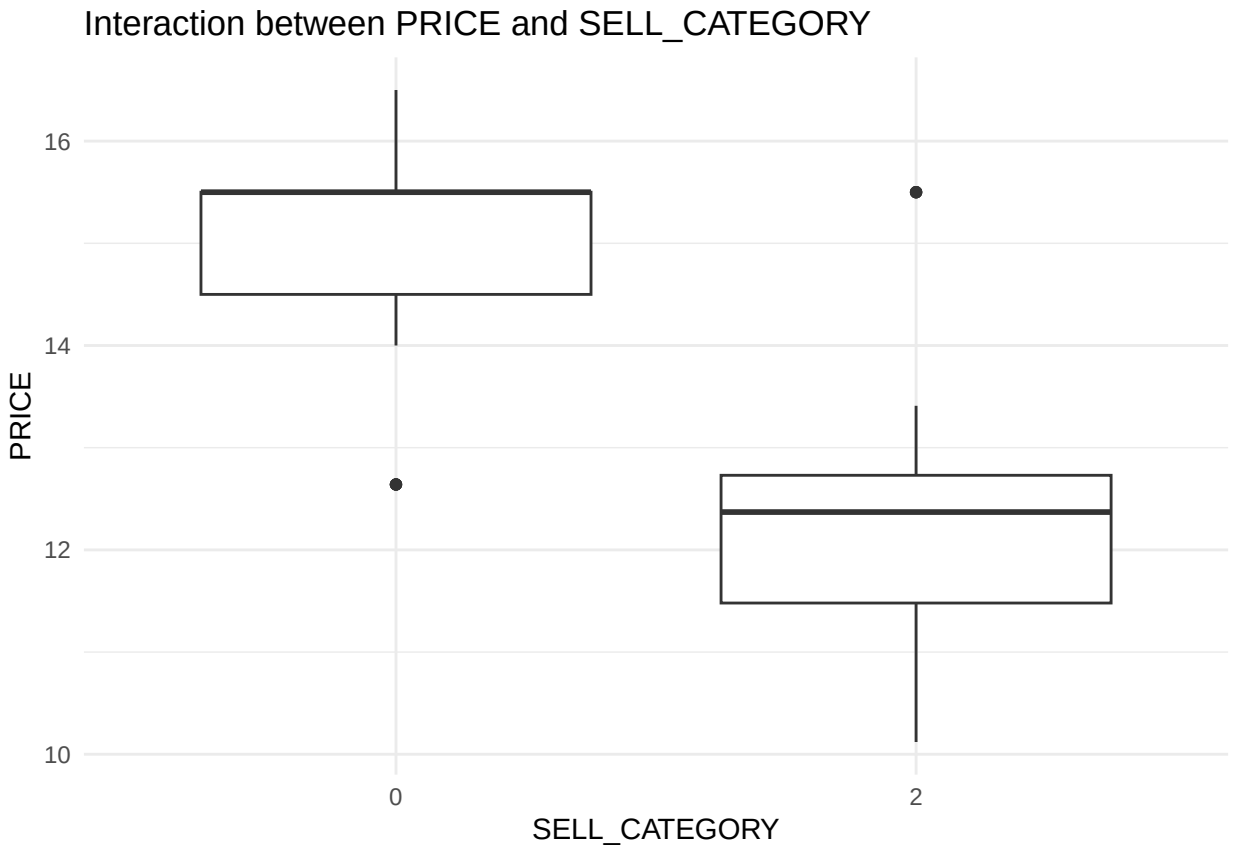
SELL_ID VS QUANTITY

```
ggplot(kiosk_data, aes(x = factor(SELL_ID), y = QUANTITY)) +
  geom_boxplot() +
  labs(x = "SELL_ID", y = "QUANTITY", title = "Interaction between QUANTITY and SELL_ID") +
  theme_minimal()
```



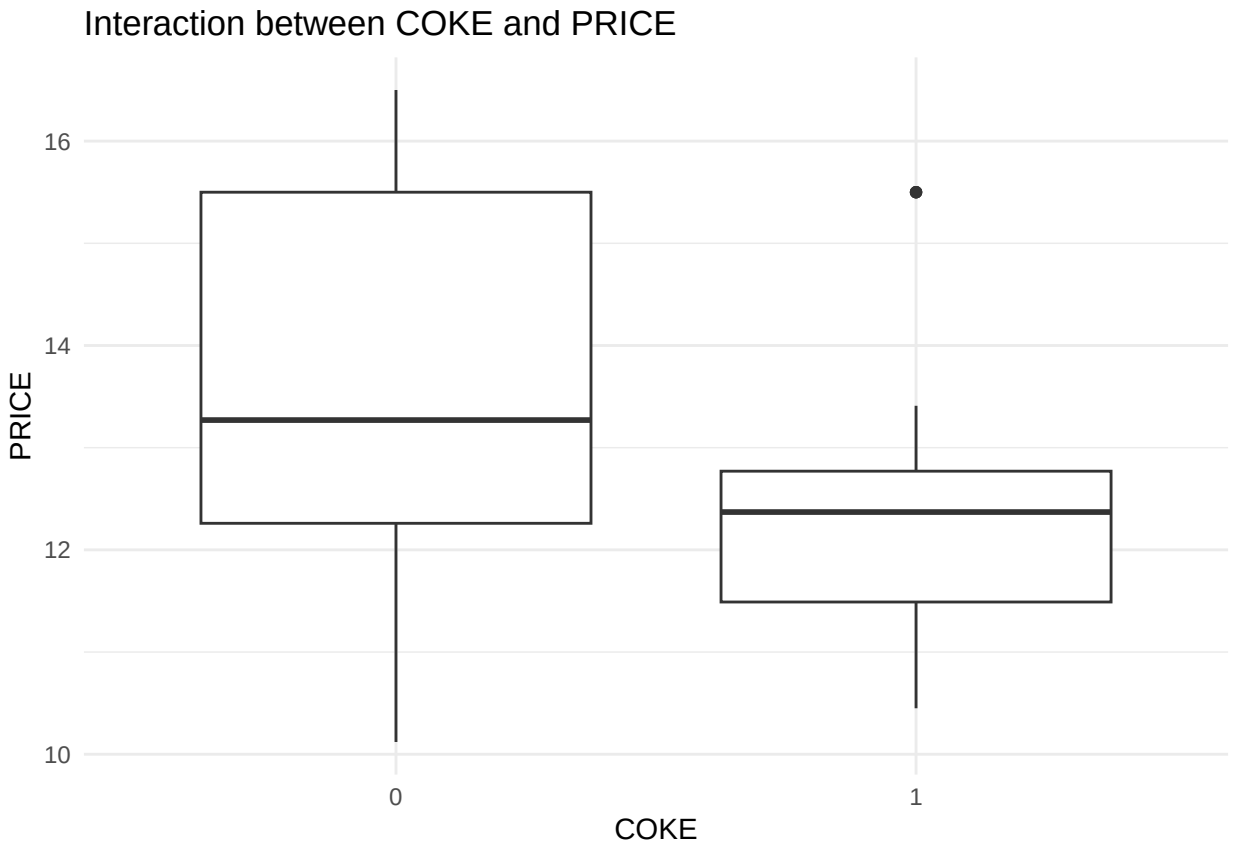
SELL_CATEGORY VS PRICE

```
ggplot(kiosk_data, aes(x = factor(SELL_CATEGORY), y = PRICE)) +
  geom_boxplot() +
  labs(x = "SELL_CATEGORY", y = "PRICE", title = "Interaction between PRICE and SELL_CATEGORY") +
  theme_minimal()
```



COKE VS PRICE

```
ggplot(kiosk_data, aes(x = factor(COKE), y = PRICE)) +
  geom_boxplot() +
  labs(x = "COKE", y = "PRICE", title = "Interaction between COKE and PRICE") +
  theme_minimal()
```



Outlier treatment

```
cat_cols <- c("COFFEE", "COKE", "LEMONADE", "HOLIDAY", "IS_WEEKEND", "IS_SCHOOLBREAK", "IS_OUTDOOR")
df1 <- kiosk_data %>% mutate_at(cat_cols, factor)

df1 %>%
  filter(SELL_ID == "1070" &
         PRICE < 14)
```

```
## # A tibble: 4 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     12.6       24 1070                0 1      0      0
## 2 2020-03-01     12.6       24 1070                0 1      0      0
## 3 2020-03-01     12.6       18 1070                0 1      0      0
## 4 2020-03-01     12.6       18 1070                0 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <fct>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

```
df1 %>%
  filter(SELL_ID == "1070",
           CALENDAR_DATE >= as.Date("2020-02-27"),
           CALENDAR_DATE <= as.Date("2020-03-02"))
```

```
## # A tibble: 11 x 15
##    CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
```

```
##    <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
##  1 2020-02-27     15.5       90 1070                0 1      0      0
##  2 2020-02-28     15.5       84 1070                0 1      0      0
##  3 2020-03-01     15.5       90 1070                0 1      0      0
##  4 2020-03-01     15.5       90 1070                0 1      0      0
##  5 2020-03-01     12.6       24 1070                0 1      0      0
##  6 2020-03-01     12.6       24 1070                0 1      0      0
##  7 2020-03-01     15.5       90 1070                0 1      0      0
##  8 2020-03-01     15.5       90 1070                0 1      0      0
##  9 2020-03-01     12.6       18 1070                0 1      0      0
## 10 2020-03-01     12.6       18 1070                0 1      0      0
## 11 2020-03-02     15.5       58 1070                0 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <fct>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

There are more than one row for the date 2020-03-01, so there've been a mistake in registering aggregated transactions. Looking at the prices of the neighboring lines, the price of $15.5 seems the right one, the right quantity seems to be 90 (value present into the 2020-03-01 rows) and the average temperature is 32, due to the same reason. So, let's get the "true/cleaned" row

```
cleaned_row_20200301_1070 <- df1 %>%
  filter(SELL_ID == "1070" &
         CALENDAR_DATE == "2020-03-01" &
         PRICE == 15.5 &
         AVERAGE_TEMPERATURE == 32) %>%
  distinct()

print(cleaned_row_20200301_1070)
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     15.5       90 1070                0 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <fct>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

Now let's filter out all the "old" 2020-03-01 rows and let's add the cleaned row into the data frame

```
kiosk_data <- rbind(
  kiosk_data %>%
    filter( !(SELL_ID == "1070" &
                CALENDAR_DATE == "2020-03-01") ),
  cleaned_row_20200301_1070 )
```

Let's check that now there is only one row for the key (CALENDAR_DATE="2020-03-01", SELL_ID="1070")

```
kiosk_data %>%
  filter(SELL_ID == "1070" &
           CALENDAR_DATE == "2020-03-01")
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     15.5       90 1070                0 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

Now let's check if there are other cases of multiple aggregate transaction rows for the same date

and the same SELL_ID

```
kiosk_data %>%
  group_by(CALENDAR_DATE, SELL_ID) %>%
  summarise(n = n()) %>%
  filter(n > 1)
```

```
## `summarise()` has grouped output by 'CALENDAR_DATE'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 3 x 3
## # Groups:   CALENDAR_DATE [1]
##   CALENDAR_DATE SELL_ID     n
##   <chr>         <fct>   <int>
## 1 2020-03-01    2051        8
## 2 2020-03-01    2052        8
## 3 2020-03-01    2053        8
```

Wow! It seems there had been some issues on 2020-03-01 for each SELL_ID.

Let's keep only the rows having SELL_ID = "2051" and PRICE > 15

```
kiosk_data %>%
  filter(SELL_ID == "2051" &
           PRICE > 15)
```

```
## # A tibble: 4 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     15.5       90 2051                2 1      0      1
```

```
## 2 2020-03-01     15.5       90 2051                2 1      0      1
## 3 2020-03-01     15.5       88 2051                2 1      0      1
## 4 2020-03-01     15.5       88 2051                2 1      0      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

let's check what's happen to the transactions for the day before and the day

after the 2020-03-01 for SELL_ID="2051"

```
kiosk_data %>%
  filter(SELL_ID == "2051" &
         CALENDAR_DATE > as.Date("2020-02-26") &
         CALENDAR_DATE <= as.Date("2020-03-03"))
```

```
## # A tibble: 12 x 15
##    CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##    <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
##  1 2020-02-27     13.1       28 2051                2 1      0      1
##  2 2020-02-28     13.1       28 2051                2 1      0      1
##  3 2020-03-01     15.5       90 2051                2 1      0      1
##  4 2020-03-01     15.5       90 2051                2 1      0      1
##  5 2020-03-01     12.6       22 2051                2 1      0      1
##  6 2020-03-01     12.6       22 2051                2 1      0      1
##  7 2020-03-01     15.5       88 2051                2 1      0      1
##  8 2020-03-01     15.5       88 2051                2 1      0      1
##  9 2020-03-01     12.6       16 2051                2 1      0      1
## 10 2020-03-01     12.6       16 2051                2 1      0      1
## 11 2020-03-02     13.1       18 2051                2 1      0      1
## 12 2020-03-03     13.1       22 2051                2 1      0      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

```
cleaned_row_20200301_2051 <- kiosk_data %>%
  filter(SELL_ID == "2051" &
           CALENDAR_DATE == "2020-03-01" &
           PRICE < 15 &
           QUANTITY == 22 &
           AVERAGE_TEMPERATURE == 32) %>%
  distinct()

print(cleaned_row_20200301_2051)
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     12.6       22 2051                2 1      0      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

Now let's filter out all the "old" 2020-03-01 rows for SELL_ID = "2051" and let's add the cleaned row

into the data frame

```
kiosk_data <- rbind(
  kiosk_data %>%
    filter( !(SELL_ID == "2051" &
                CALENDAR_DATE == "2020-03-01") ),
  cleaned_row_20200301_2051 )
```

Let's check that now there is only one row for the key (CALENDAR_DATE="2020-03-01", SELL_ID="2051")

```
kiosk_data %>%
  filter(SELL_ID == "2051" &
           CALENDAR_DATE == "2020-03-01")
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     12.6       22 2051                2 1      0      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

Let's check what's happen to the transactions for the day before and the day

after the 2020-03-01 for SELL_ID="2052"

```
kiosk_data %>%
  filter(SELL_ID == "2052" &
           CALENDAR_DATE > as.Date("2020-02-26") &
           CALENDAR_DATE <= as.Date("2020-03-03"))
```

```
## # A tibble: 12 x 15
##    CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##    <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
##  1 2020-02-27     12.6       22 2052                2 1      0      0
##  2 2020-02-28     12.6       22 2052                2 1      0      0
##  3 2020-03-01     13.1       30 2052                2 1      0      0
##  4 2020-03-01     13.1       30 2052                2 1      0      0
##  5 2020-03-01     13.4       40 2052                2 1      0      0
##  6 2020-03-01     13.4       40 2052                2 1      0      0
##  7 2020-03-01     13.1       26 2052                2 1      0      0
##  8 2020-03-01     13.1       26 2052                2 1      0      0
##  9 2020-03-01     13.4       40 2052                2 1      0      0
## 10 2020-03-01     13.4       40 2052                2 1      0      0
## 11 2020-03-02     12.6       20 2052                2 1      0      0
```

```
## 12 2020-03-03     12.6       20 2052                2 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

```
cleaned_row_20200301_2052 <- kiosk_data %>%
  filter(SELL_ID == "2052" &
           CALENDAR_DATE == "2020-03-01" &
           QUANTITY == 26 &
           AVERAGE_TEMPERATURE == 32) %>%
  distinct()

print(cleaned_row_20200301_2052)
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     13.1       26 2052                2 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

```
kiosk_data <- rbind(
  kiosk_data %>%
    filter( !(SELL_ID == "2052" &
                CALENDAR_DATE == "2020-03-01") ),
  cleaned_row_20200301_2052 )
```

Let's check that now there is only one row for the key (CALENDAR_DATE="2 03-01", SELL_ID="2052")

```
kiosk_data %>%
  filter(SELL_ID == "2052" &
           CALENDAR_DATE == "2020-03-01")
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     13.1       26 2052                2 1      0      0
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

Let's check what's happen to the transactions for the day before and the day

after the 2020-03-01 for SELL_ID="2053"

```
kiosk_data %>%
  filter(SELL_ID == "2053" &
           CALENDAR_DATE > as.Date("2020-02-26") &
           CALENDAR_DATE <= as.Date("2020-03-03"))
```

```
## # A tibble: 12 x 15
##    CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##    <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
##  1 2020-02-27     13.4       42 2053                2 1      1      1
##  2 2020-02-28     13.4       44 2053                2 1      1      1
##  3 2020-03-01     13.1       26 2053                2 1      1      1
##  4 2020-03-01     13.1       26 2053                2 1      1      1
##  5 2020-03-01     13.4       40 2053                2 1      1      1
##  6 2020-03-01     13.4       40 2053                2 1      1      1
##  7 2020-03-01     13.1       24 2053                2 1      1      1
##  8 2020-03-01     13.1       24 2053                2 1      1      1
##  9 2020-03-01     13.4       40 2053                2 1      1      1
## 10 2020-03-01     13.4       40 2053                2 1      1      1
## 11 2020-03-02     13.4       26 2053                2 1      1      1
## 12 2020-03-03     13.4       34 2053                2 1      1      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

```
cleaned_row_20200301_2053 <- kiosk_data %>%
  filter(SELL_ID == "2053" &
           CALENDAR_DATE == "2020-03-01" &
           QUANTITY == 40 &
           AVERAGE_TEMPERATURE == 32) %>%
  distinct()

print(cleaned_row_20200301_2053)
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     13.4       40 2053                2 1      1      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

```
kiosk_data <- rbind(
  kiosk_data %>%
    filter( !(SELL_ID == "2053" &
                CALENDAR_DATE == "2020-03-01") ),
  cleaned_row_20200301_2053 )
```

Let's check that now there is only one row for the key (CALENDAR_DATE="2 03-01", SELL_ID="2053")

```
kiosk_data %>%
  filter(SELL_ID == "2053" &
           CALENDAR_DATE == "2020-03-01")
```

```
## # A tibble: 1 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     13.4       40 2053                2 1      1      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
```

```
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

Now we expect to have one row for each SELL_ID in the date 2020-03-01

```
kiosk_data %>%
  filter( CALENDAR_DATE == "2020-03-01" )
```

```
## # A tibble: 4 x 15
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <chr>         <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2020-03-01     15.5       90 1070                0 1      0      0
## 2 2020-03-01     12.6       22 2051                2 1      0      1
## 3 2020-03-01     13.1       26 2052                2 1      0      0
## 4 2020-03-01     13.4       40 2053                2 1      1      1
## # i 7 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>
```

We were lucky to find a data issue just checking for outliers. Removing the wrong transaction rows we have also removed all the extreme outliers showed before. So, we have just killed multiple birds with one stone!

Variable transformation

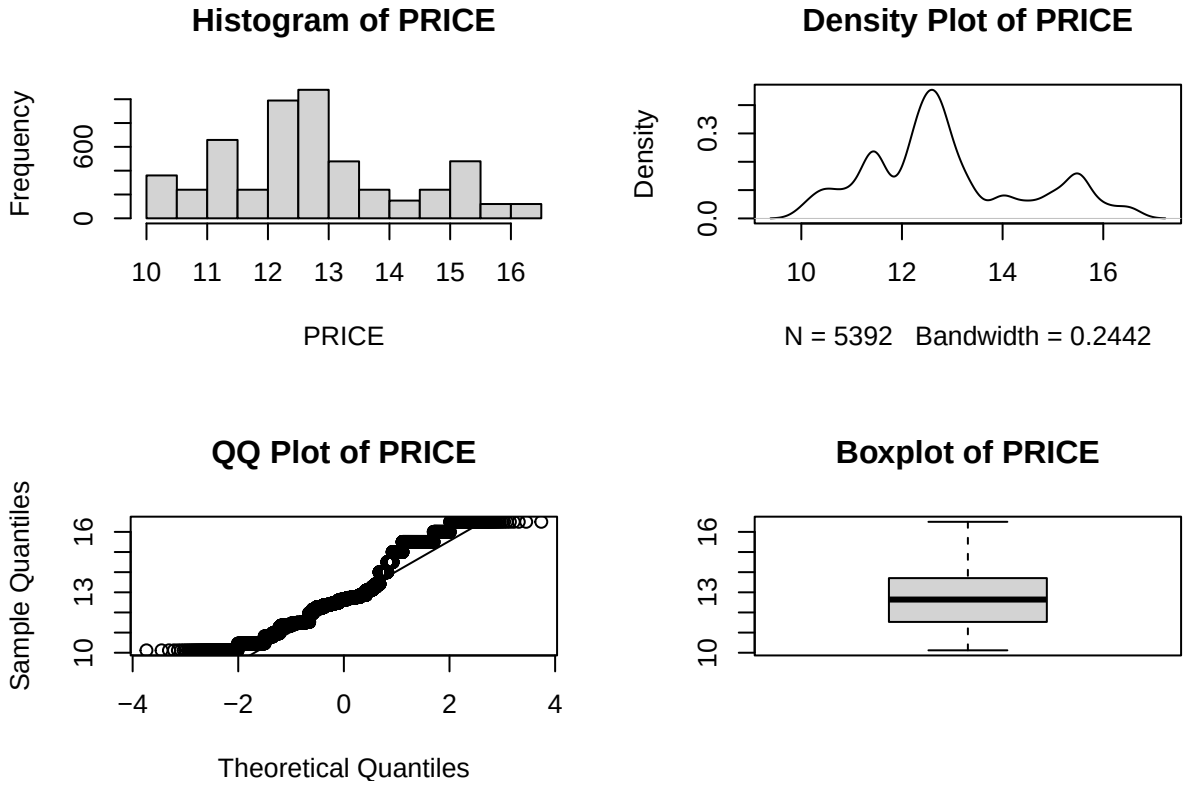
Sometimes outliers in a variable distribution aren't wrong measures, they may be inherent in the nature of the variable itself, especially if it's skewed. Cutting away these values from a distribution can remove useful information for a successful predictive modeling. So, it's preferable to transform the variable to mitigate the skewness, trying to make its distribution much similar to a normal one. Reducing non-normality often reduces non-linearity as well and even if the transformed distribution is not exactly normal, it will be usually symmetric.

When a distribution is right-skewed, a log transformation is often used. But what if it's left-skewed? Fortunately there is a generic way to transform non-normal distribution: the Box-Cox transformation. It'll be applied to PRICE and QUANTITY only, because AVERAGE_TEMPERATURE is already symmetric (a Box-Cox transformation upon it doesn't change its distribution, since it cannot be transformed in a normal one).

```
var_distribution <- function(data, var_name){
  par(mfrow=c(2,2))
  if(length(data) >= 5000){
    sampled_data = data[sample(1:length(data), 5000, replace=FALSE)]
    normtest <- shapiro.test(sampled_data)
  } else{
    normtest <- shapiro.test(data)
  }


  p.value <- round(normtest$p.value,4)
  if (p.value < 0.05) {
    h0 <- 'rejected.'
    color <- 'red'
  } else {
```

```
    h0 <- 'accepted.'
    color <- 'blue'
  }
  hist(data, xlab = var_name, main = paste('Histogram of', var_name))

  d <- density(data)
  plot(d, main = paste('Density Plot of', var_name))
  qqnorm(data, main = paste('QQ Plot of', var_name))
  qqline(data)
  boxplot(data, main = paste('Boxplot of', var_name))
  mtext(paste('Normality test of', var_name, h0, '( p-value=', p.value, ')'),
        side = 3, line = -1, outer = TRUE, col=color)
  par(mfrow=c(1,1))
}

boxcox_transf <- function(data){
  require(MASS)
  box <- boxcox( data ~ 1,                    # Transform data as a single vector
                 lambda = seq(-6,6,0.1),      # Try values from -6 to 6 by 0.1
                 plotit = FALSE )

  cox <- data.frame( box$x, box$y )           # Create a data frame with the results
  cox2 <- cox[with(cox, order(-cox$box.y)),] # Order the new data frame by decreasing y

  lambda <- cox2[1, "box.x"]                  # Extract that lambda

  list( "data"=(data ^ lambda - 1)/lambda,
        "lambda"=lambda )                     # Return a list containing the transformed
                                              # data and lambda value
}


var_distribution(kiosk_data[['PRICE']], 'PRICE')
```

Normality test of PRICE rejected. (p−value= 0)




```
t_price <- boxcox_transf(kiosk_data[['PRICE']])
```

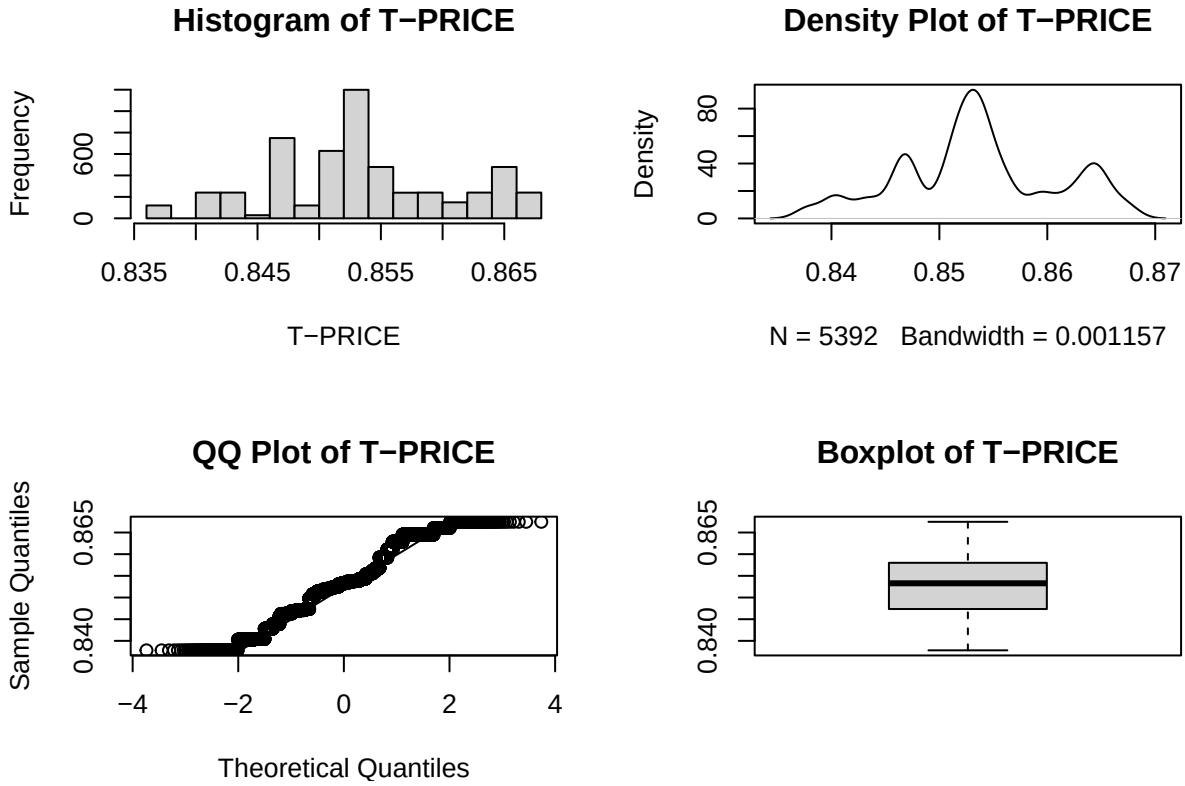
```
## Loading required package: MASS
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
var_distribution(t_price$data, 'T-PRICE')
```

Normality test of T−PRICE rejected. (p−value= 0)

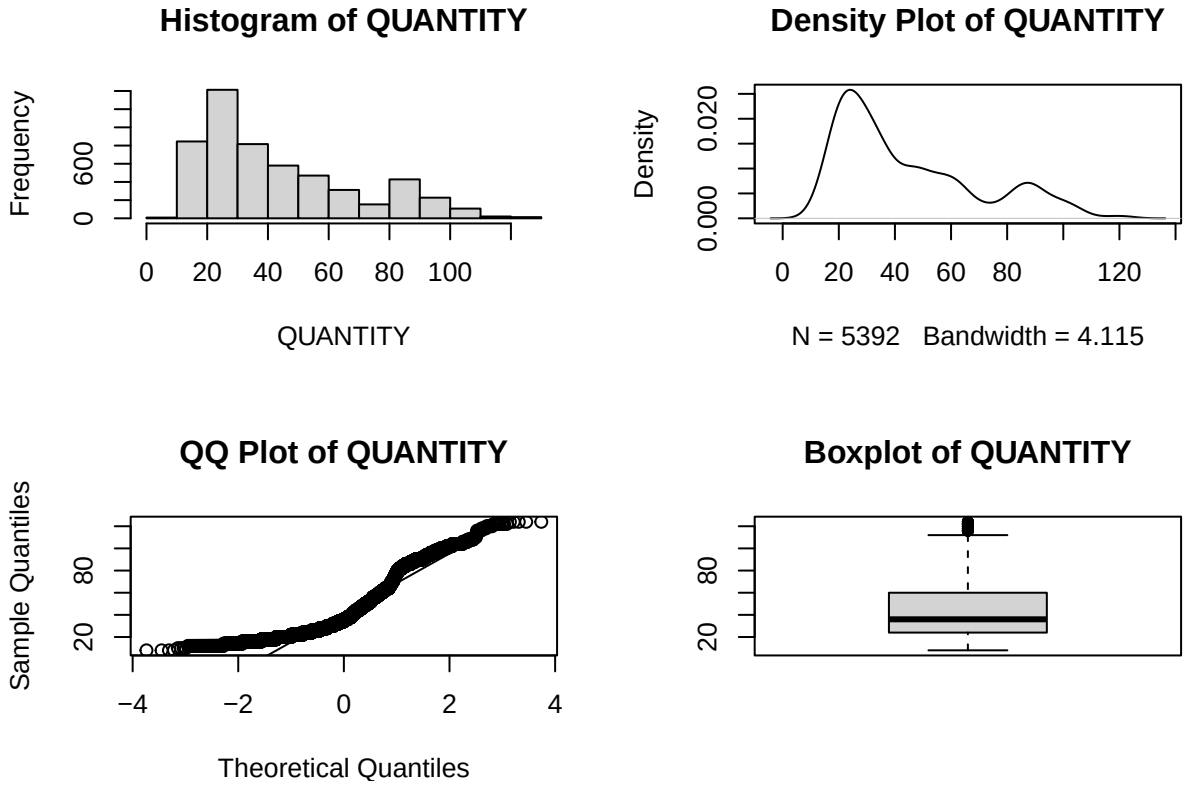



```
print(t_price$lambda)
```

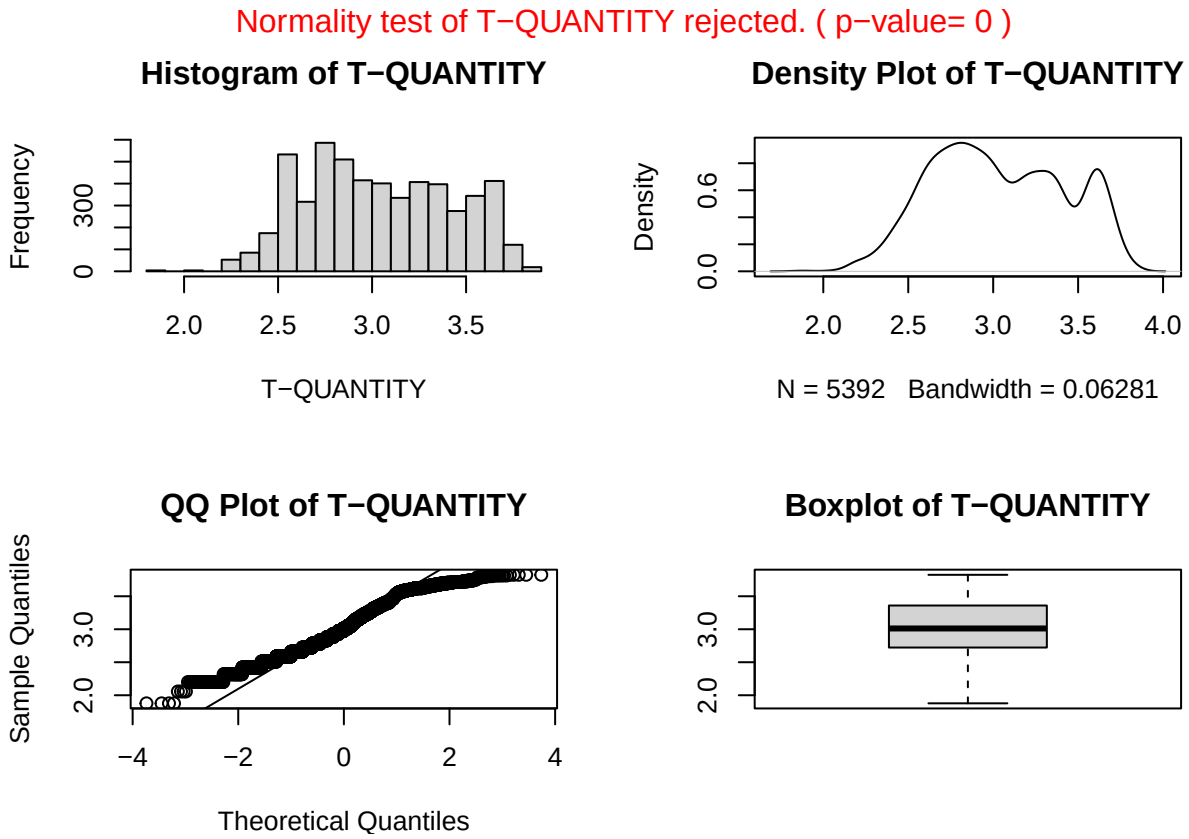
```
## [1] -1.1
```

```
var_distribution(kiosk_data[['QUANTITY']], 'QUANTITY')
```

Normality test of QUANTITY rejected. (p−value= 0)




```
t_quantity <- boxcox_transf(kiosk_data[['QUANTITY']])
var_distribution(t_quantity$data, 'T-QUANTITY')
```




```
print(t_quantity$lambda)
```

```
## [1] -0.1
```

The var_distribution function gives us four plots.

The boxcox_transf function returns a list containing the transformed data and the value of lambda calculated by the Box-Cox transformation.

As you can see, the transformed density plot shows a more symmetric curve and the transformed Q-Q plot is closer to the normal line. The lambda value used to transform the PRICE variable is -1.1.

The lambda value used to transform the QUANTITY variable is -0.1. In this case, as you can see from the T-QUANTITY boxplot, the outliers have disappeared without removing them thanks to the transformation.

Variable creation

New Date/Time variables

It's also useful to extract a new variable representing the counter of days since the first day we can find in the data set. In general, date/time variable are cyclical (e.g. number of month goes from 1 to 12; number of day goes from 1 to 31). A machine learning algorithm doesn't take into account the cyclicity of variables. So, a counter of days that represents the passing of the time is a good variable to add, because it can help the algorithm to catch any sales growth since the beginning of the activity.

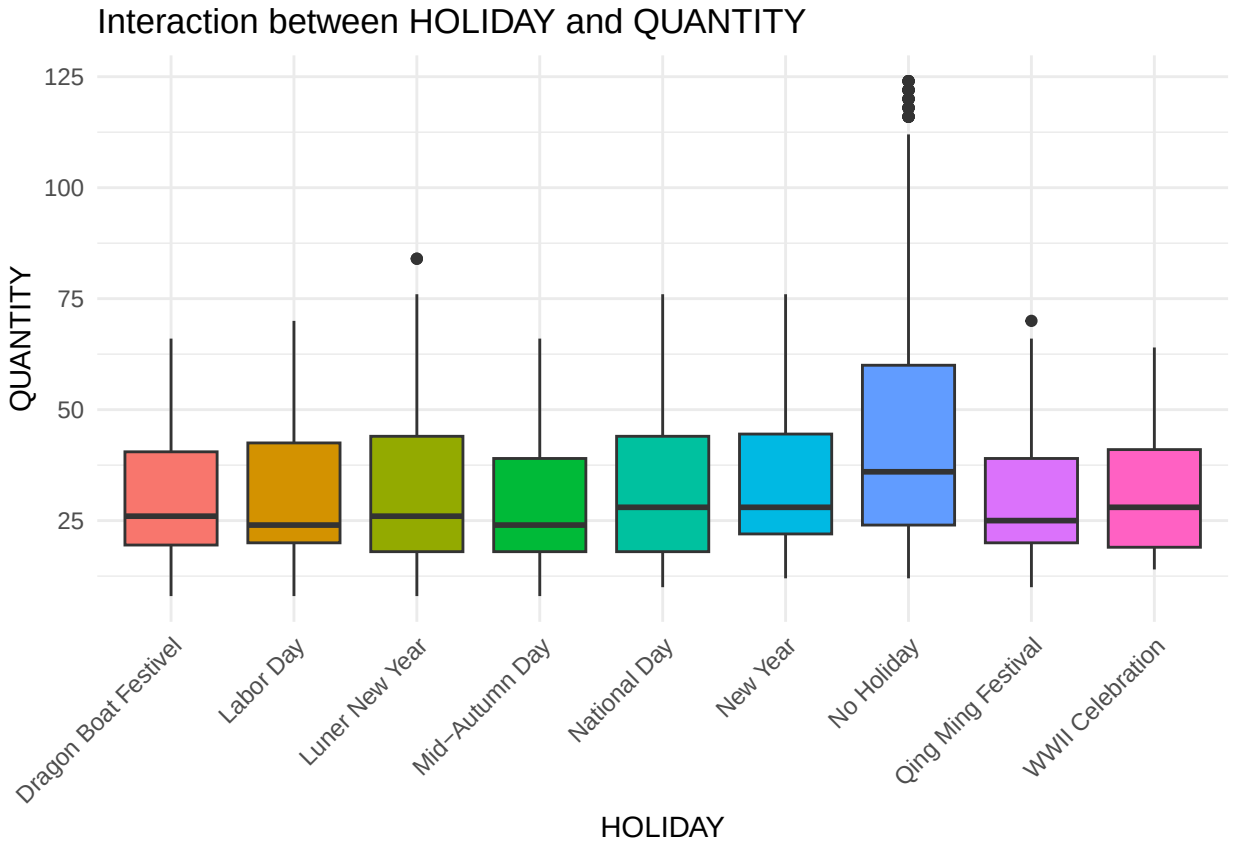
```
library(lubridate)

# Ensure CALENDAR_DATE is a Date object
kiosk_data$CALENDAR_DATE <- as.Date(kiosk_data$CALENDAR_DATE)
```

```r
# MONTH and DAY variables will be extracted from CALENDAR_DATE
# A new variable counting the days from the beginning in the data frame will be added

min_date <- min( kiosk_data$CALENDAR_DATE )

kiosk_data <- kiosk_data %>%
        mutate( DAYS_FROM_BEGINNING = as.integer(CALENDAR_DATE - min_date) ) %>%
        mutate( MONTH = month(CALENDAR_DATE) ) %>%
        mutate( DAY = day(CALENDAR_DATE) ) %>%
        mutate( WDAY = wday(CALENDAR_DATE) )

head(kiosk_data)
```

```
## # A tibble: 6 x 19
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <date>        <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2019-01-01     15.5       46 1070                0 1      0      0
## 2 2019-01-01     12.7       22 2051                2 1      0      1
## 3 2019-01-01     12.8       18 2052                2 1      0      0
## 4 2019-01-01     12.6       30 2053                2 1      1      1
## 5 2019-01-02     15.5       70 1070                0 1      0      0
## 6 2019-01-02     12.7       22 2051                2 1      0      1
## # i 11 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>, DAYS_FROM_BEGINNING <int>, MONTH <dbl>, DAY <int>,
## #   WDAY <dbl>
```

HOLIDAY VS QUANTITY

```r
ggplot(kiosk_data, aes(x = HOLIDAY, y = QUANTITY, fill = HOLIDAY)) +
  geom_boxplot() +
  labs(x = "HOLIDAY", y = "QUANTITY", title = "Interaction between HOLIDAY and QUANTITY") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme(legend.position = "none")
```



It seems that in business days (HOLIDAY = "No Holiday") the quantities sold are significantly greater then during the holidays. So, a new dummy variable called IS_HOLIDAY (1 if the transactions happen in holidays; 0 otherwise) might help machine learning algorithms to perform in a better way.

```
library(dplyr)

kiosk_data <- kiosk_data %>%
        mutate( IS_HOLIDAY = as.factor(ifelse(HOLIDAY == "No Holiday", 0, 1)) )

head(kiosk_data)
```

```
## # A tibble: 6 x 20
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
##   <date>        <dbl>    <dbl> <fct>           <dbl> <fct>  <fct>  <fct>
## 1 2019-01-01     15.5       46 1070                0 1      0      0
## 2 2019-01-01     12.7       22 2051                2 1      0      1
## 3 2019-01-01     12.8       18 2052                2 1      0      0
## 4 2019-01-01     12.6       30 2053                2 1      1      1
## 5 2019-01-02     15.5       70 1070                0 1      0      0
## 6 2019-01-02     12.7       22 2051                2 1      0      1
## # i 12 more variables: LEMONADE <fct>, YEAR <fct>, HOLIDAY <fct>,
## #   IS_WEEKEND <fct>, IS_SCHOOLBREAK <chr>, AVERAGE_TEMPERATURE <dbl>,
## #   IS_OUTDOOR <fct>, DAYS_FROM_BEGINNING <int>, MONTH <dbl>, DAY <int>,
## #   WDAY <dbl>, IS_HOLIDAY <fct>
```

There are also some factor columns that have to be converted to one-hot encoded variables (as you can see, we already have some of them in our data set: COFFEE, LEMONADE, ...). One-hot encoded variables (or dummy variables) are numeric variables, since they represent a characteristic that exists (1) or not exists (0).

In our analysis they're often transformed in categorical variables to facilitate the graphical representations. In R one-hot encoded variables can be achieved thanks to the model.matix function.

```
kiosk_data <- cbind( kiosk_data,
       model.matrix( ~ HOLIDAY - 1, data = kiosk_data ))

# We'll keep the HOLDAY variable for a subsequent analysis.
# After that, it can be removed.
# kiosk_data$HOLIDAY <- NULL

kiosk_data <- cbind( kiosk_data,
             model.matrix( ~ SELL_ID - 1, data = kiosk_data ) )

#kiosk_data$SELL_ID <- NULL

head(kiosk_data)
```

```
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
## 1    2019-01-01 15.50       46    1070             0      1      0    0
## 2    2019-01-01 12.73       22    2051             2      1      0    1
## 3    2019-01-01 12.75       18    2052             2      1      0    0
## 4    2019-01-01 12.60       30    2053             2      1      1    1
## 5    2019-01-02 15.50       70    1070             0      1      0    0
## 6    2019-01-02 12.73       22    2051             2      1      0    1
##   LEMONADE YEAR  HOLIDAY IS_WEEKEND IS_SCHOOLBREAK AVERAGE_TEMPERATURE
## 1        0 2019 New Year          1              0                24.8
## 2        0 2019 New Year          1              0                24.8
## 3        1 2019 New Year          1              0                24.8
## 4        0 2019 New Year          1              0                24.8
## 5        0 2019 New Year          0              0                24.8
## 6        0 2019 New Year          0              0                24.8
##   IS_OUTDOOR DAYS_FROM_BEGINNING MONTH DAY WDAY IS_HOLIDAY
## 1          0                   0     1   1    3          1
## 2          0                   0     1   1    3          1
## 3          0                   0     1   1    3          1
## 4          0                   0     1   1    3          1
## 5          0                   1     1   2    4          1
## 6          0                   1     1   2    4          1
##   HOLIDAYDragon Boat Festivel HOLIDAYLabor Day HOLIDAYLuner New Year
## 1                           0                0                     0
## 2                           0                0                     0
## 3                           0                0                     0
## 4                           0                0                     0
## 5                           0                0                     0
## 6                           0                0                     0
##   HOLIDAYMid-Autumn Day HOLIDAYNational Day HOLIDAYNew Year HOLIDAYNo Holiday
## 1                     0                   0               1                 0
## 2                     0                   0               1                 0
## 3                     0                   0               1                 0
## 4                     0                   0               1                 0
## 5                     0                   0               1                 0
## 6                     0                   0               1                 0
##   HOLIDAYQing Ming Festival HOLIDAYWWII Celebration SELL_ID1070 SELL_ID2051
## 1                         0                       0           1           0
## 2                         0                       0           0           1
```

```
## 3                         0                         0           0           0
## 4                         0                         0           0           0
## 5                         0                         0           1           0
## 6                         0                         0           0           1
##   SELL_ID2052 SELL_ID2053
## 1           0           0
## 2           0           0
## 3           1           0
## 4           0           1
## 5           0           0
## 6           0           0
```

New Numerical variables

Having a variable with the number of items in a combo product could be an important feature to determine a price variation. Here it's possible to add the variable NO_ITEMS defined as:

```
kiosk_data$NO_ITEMS <- as.integer(kiosk_data$BURGER) + as.integer(kiosk_data$COFFEE)
+ as.integer(kiosk_data$COKE) + as.integer(kiosk_data$LEMONADE)
```

```
##    [1] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##   [38] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##   [75] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
##  [112] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
##  [149] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##  [186] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##  [223] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
##  [260] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
##  [297] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##  [334] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##  [371] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
##  [408] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
##  [445] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##  [482] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##  [519] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
##  [556] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
##  [593] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##  [630] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##  [667] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
##  [704] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
##  [741] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##  [778] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##  [815] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
##  [852] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
##  [889] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
##  [926] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
##  [963] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1000] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1037] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1074] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [1111] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1148] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1185] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1222] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
```

```
## [1259] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1296] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1333] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1370] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [1407] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1444] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1481] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1518] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [1555] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1592] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1629] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1666] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [1703] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1740] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1777] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1814] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [1851] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [1888] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [1925] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [1962] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [1999] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2036] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2073] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2110] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [2147] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2184] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2221] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2258] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [2295] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2332] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2369] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2406] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [2443] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2480] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2517] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2554] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [2591] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2628] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2665] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2702] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [2739] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2776] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2813] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2850] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [2887] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [2924] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [2961] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [2998] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3035] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3072] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [3109] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [3146] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3183] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3220] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
```

```
## [3257] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [3294] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3331] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3368] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [3405] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [3442] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3479] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3516] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [3553] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [3590] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3627] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3664] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [3701] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [3738] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3775] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3812] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [3849] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [3886] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [3923] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [3960] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [3997] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4034] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4071] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4108] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [4145] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4182] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4219] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4256] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [4293] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4330] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4367] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4404] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [4441] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4478] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4515] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4552] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [4589] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4626] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4663] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4700] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [4737] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4774] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4811] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4848] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [4885] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [4922] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [4959] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [4996] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [5033] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [5070] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
## [5107] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [5144] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [5181] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [5218] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3
```

```
## [5255] 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3
## [5292] 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
## [5329] 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2
## [5366] 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3 2 3 3 3
```

```
head(kiosk_data)
```

```
##   CALENDAR_DATE PRICE QUANTITY SELL_ID SELL_CATEGORY BURGER COFFEE COKE
## 1    2019-01-01 15.50       46    1070             0      1      0    0
## 2    2019-01-01 12.73       22    2051             2      1      0    1
## 3    2019-01-01 12.75       18    2052             2      1      0    0
## 4    2019-01-01 12.60       30    2053             2      1      1    1
## 5    2019-01-02 15.50       70    1070             0      1      0    0
## 6    2019-01-02 12.73       22    2051             2      1      0    1
##   LEMONADE YEAR  HOLIDAY IS_WEEKEND IS_SCHOOLBREAK AVERAGE_TEMPERATURE
## 1        0 2019 New Year          1              0                24.8
## 2        0 2019 New Year          1              0                24.8
## 3        1 2019 New Year          1              0                24.8
## 4        0 2019 New Year          1              0                24.8
## 5        0 2019 New Year          0              0                24.8
## 6        0 2019 New Year          0              0                24.8
##   IS_OUTDOOR DAYS_FROM_BEGINNING MONTH DAY WDAY IS_HOLIDAY
## 1          0                   0     1   1    3          1
## 2          0                   0     1   1    3          1
## 3          0                   0     1   1    3          1
## 4          0                   0     1   1    3          1
## 5          0                   1     1   2    4          1
## 6          0                   1     1   2    4          1
##   HOLIDAYDragon Boat Festivel HOLIDAYLabor Day HOLIDAYLuner New Year
## 1                           0                0                     0
## 2                           0                0                     0
## 3                           0                0                     0
## 4                           0                0                     0
## 5                           0                0                     0
## 6                           0                0                     0
##   HOLIDAYMid-Autumn Day HOLIDAYNational Day HOLIDAYNew Year HOLIDAYNo Holiday
## 1                     0                   0               1                 0
## 2                     0                   0               1                 0
## 3                     0                   0               1                 0
## 4                     0                   0               1                 0
## 5                     0                   0               1                 0
## 6                     0                   0               1                 0
##   HOLIDAYQing Ming Festival HOLIDAYWWII Celebration SELL_ID1070 SELL_ID2051
## 1                         0                       0           1           0
## 2                         0                       0           0           1
## 3                         0                       0           0           0
## 4                         0                       0           0           0
## 5                         0                       0           1           0
## 6                         0                       0           0           1
##   SELL_ID2052 SELL_ID2053 NO_ITEMS
## 1           0           0        2
## 2           0           0        2
## 3           1           0        2
## 4           0           1        3
## 5           0           0        2
```

```
## 6           0           0        2
```

Further analysis after cleaning data

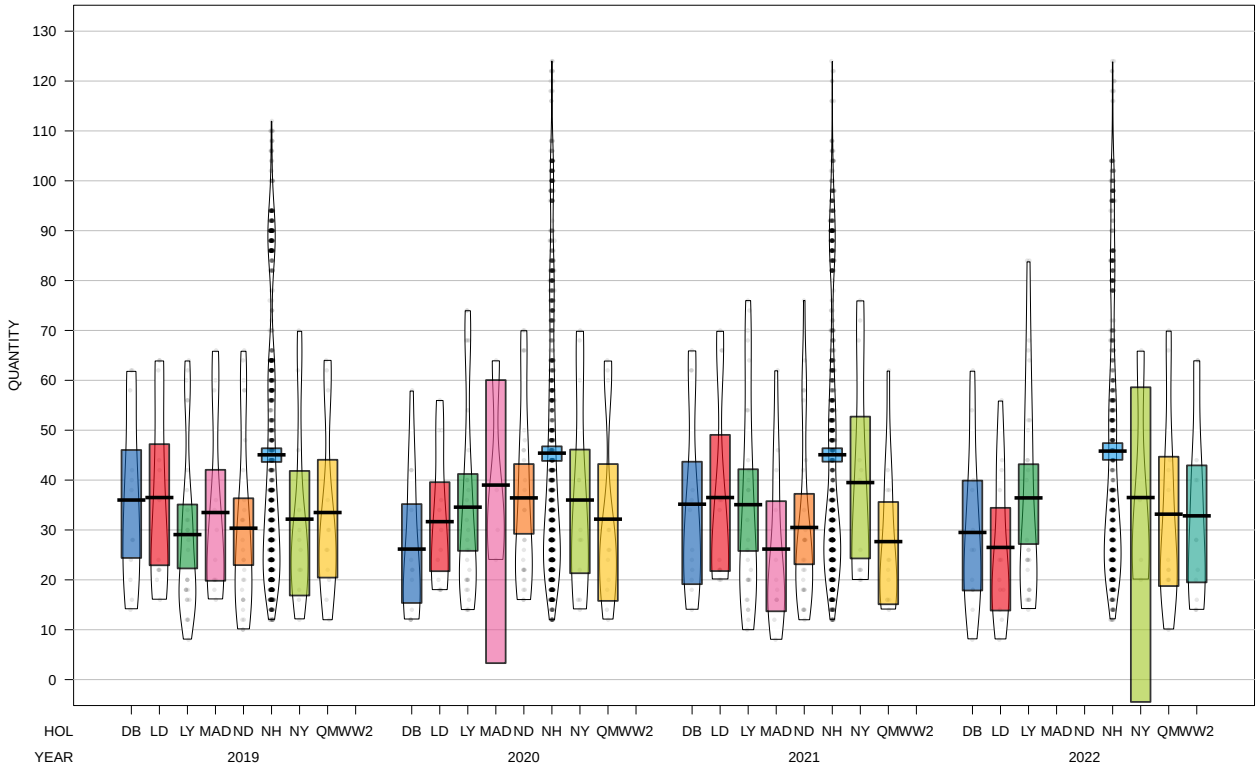
Holidays analysis

Before dropping the HOLIDAY column, as commented in the previous code, it's interesting to check how each holiday is represented in our data set for each year. How to do that? The pirateplot comes to the rescue. Before using this plot, for a better visual representation, we'll add a new variable (HOLIDAY_ABBR) with abbreviated labels for each holiday.

```
library(yarrr)
```

```
## Loading required package: jpeg
## Loading required package: BayesFactor
## Loading required package: coda
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
## ************
## Welcome to BayesFactor 0.9.12-4.7. If you have questions, please contact Richard Morey (richarddmorey
##
## Type BFManual() to open the manual.
## ************
## Loading required package: circlize
## ========================================
## circlize version 0.4.16
## CRAN page: https://cran.r-project.org/package=circlize
## Github page: https://github.com/jokergoo/circlize
## Documentation: https://jokergoo.github.io/circlize_book/book/
##
## If you use it in published research, please cite:
## Gu, Z. circlize implements and enhances circular visualization
##   in R. Bioinformatics 2014.
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(circlize))
## ========================================
## yarrr v0.1.5. Citation info at citation('yarrr'). Package guide at yarrr.guide()
## Email me at Nathaniel.D.Phillips.is@gmail.com
##
## Attaching package: 'yarrr'
## The following object is masked from 'package:ggplot2':
##
##     diamonds
```

```
kiosk_data <- kiosk_data %>%
        mutate(HOL = case_when(
                                .$HOLIDAY == "Luner New Year" ~ "LY",
                                .$HOLIDAY == "No Holiday" ~ "NH",
                                .$HOLIDAY == "Labor Day" ~ "LD",
                                .$HOLIDAY == "Dragon Boat Festivel" ~ "DB",
                                .$HOLIDAY == "Mid-Autumn Day" ~ "MAD",
                                .$HOLIDAY == "New Year" ~ "NY",
                                .$HOLIDAY == "WWII Celebration" ~ "WW2",
                                .$HOLIDAY == "National Day" ~ "ND",
                                .$HOLIDAY == "Qing Ming Festival" ~ "QM"
        ))

par(mfrow=c(1,1))
pirateplot(formula = QUANTITY ~ HOL + YEAR,
           data = kiosk_data, theme = 2)
```



As you can see, a few holidays are missing for some year:

Mid-Autumn Day is missing in 2022 National Day is missing in 2022 WWII Celebration (the end of the Second World War) is missing for years 2019, 2020 and 2021.

It's easy to justify the lack of the first two holidays in the year 2022. The registered transactions we have in the data set have a maximum date of September the 10th for the year 2022. So these holidays fell later than the maximum date.

```
max( kiosk_data$CALENDAR_DATE )
```

```
## [1] "2022-09-10"
```

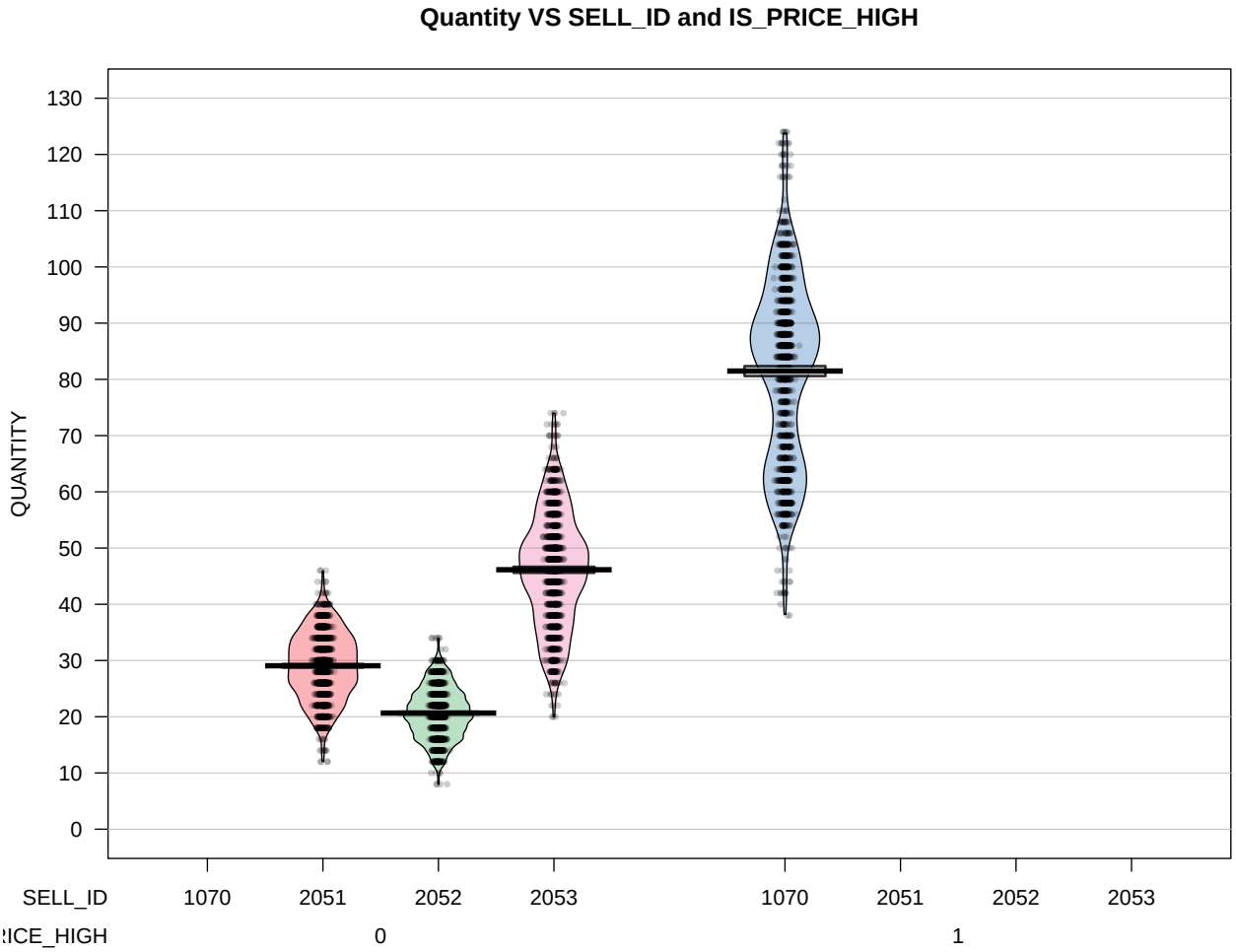
The WWII Celebration is missing in years before the 2022 Since the data set is made by transactions of a Burger Cafè in Microsoft China, it seems China decided to celebrate the end of WWII just since 2022. So,

the reason of lack in years before 2022 is explained.

Impact of SELL_ID to the demand curve

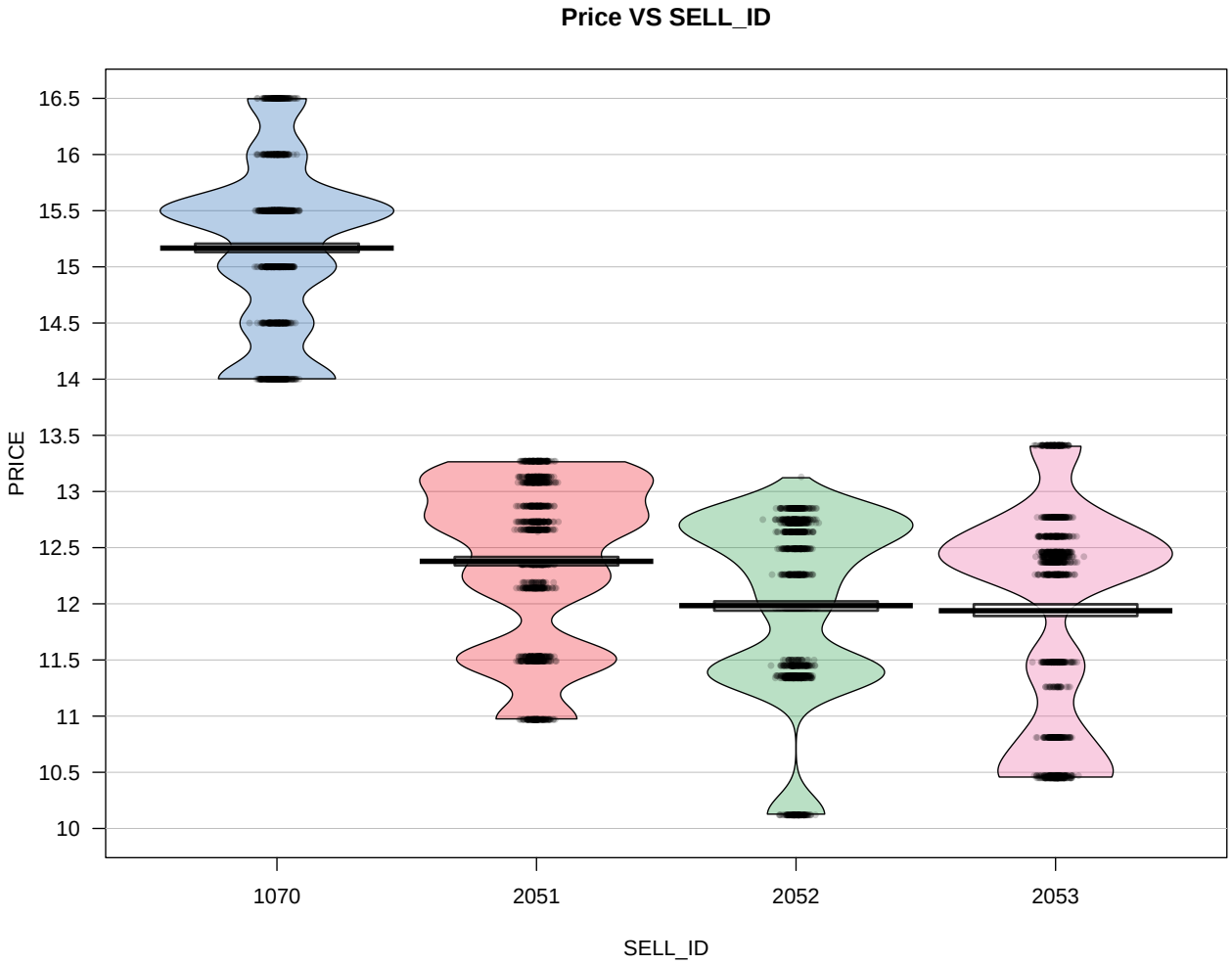
We'd like to know if the burger sales follows a linear demand. We supposed there were another variable that broke what could be a linear behavior between QUANTITY and PRICE. Let's go deeper in the analysis to answer this question.

```
kiosk_data <- kiosk_data %>%
  mutate(IS_PRICE_HIGH = as.factor(ifelse(PRICE >= 14, 1, 0)) )

pirateplot(formula = QUANTITY ~ SELL_ID + IS_PRICE_HIGH,
           data = kiosk_data,
           main = "Quantity VS SELL_ID and IS_PRICE_HIGH")
```



Looking at the plot, the presence of only the SELL_ID 1070 into the IS_PRICE_HIGH area corresponding to prices equal of greater of $14, and the inference bands around the means that aren't overlapping, it seems confirmed that each product identified by SELL_ID is sold at their mean quantity with 9% of confidence and that high prices identify specific SELL_ID.

```
pirateplot(formula = PRICE ~ SELL_ID ,
           data = kiosk_data,
           main = "Price VS SELL_ID")
```



Mean confidence bars for SELL_IDs 2052 and 2053 are overlapping, so they are hardly distinguishable by price.

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
plot1 <- kiosk_data %>%
  filter(SELL_ID == 1070) %>%
  ggplot(aes(x=PRICE, y=QUANTITY)) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(title="BURGER: PRICE vs QUANTITY",x="Price (US$)", y="Quantity") +
  theme_minimal()

plot2 <- kiosk_data %>%
  filter(SELL_ID == 2051) %>%
  ggplot(aes(x=PRICE, y=QUANTITY)) +
```

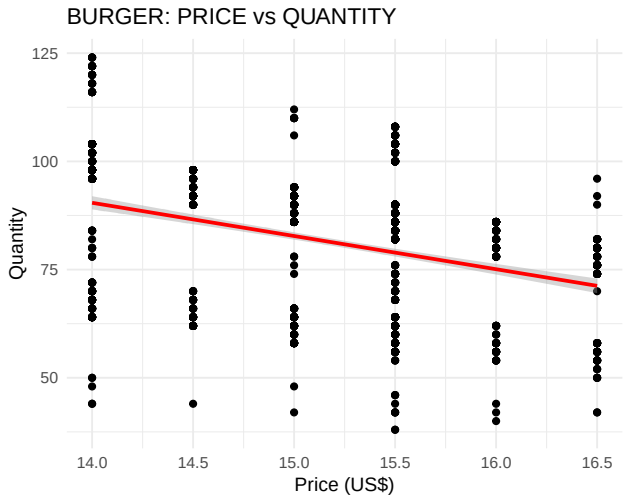
```
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(title="BURGER+COKE: PRICE vs QUANTITY",x="Price (US$)", y="Quantity") +
  theme_minimal()

plot3 <- kiosk_data %>%
  filter(SELL_ID == 2052) %>%
  ggplot(aes(x=PRICE, y=QUANTITY)) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(title="BURGER+LEMONADE: PRICE vs QUANTITY",x="Price (US$)", y="Quantity") +
  theme_minimal()

plot4 <- kiosk_data %>%
  filter(SELL_ID == 2053) %>%
  ggplot(aes(x=PRICE, y=QUANTITY)) +
  geom_point() +
  geom_smooth() +
  geom_smooth(method="lm", color="red") +
  labs(title="BURGER+COKE+COFFEE: PRICE vs QUANTITY",x="Price (US$)", y="Quantity") +
  theme_minimal()

grid.arrange(plot1, plot2, plot3, plot4, ncol=2)
```

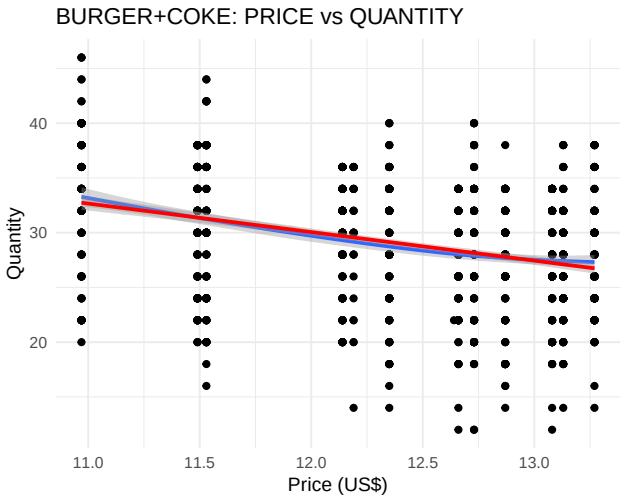
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

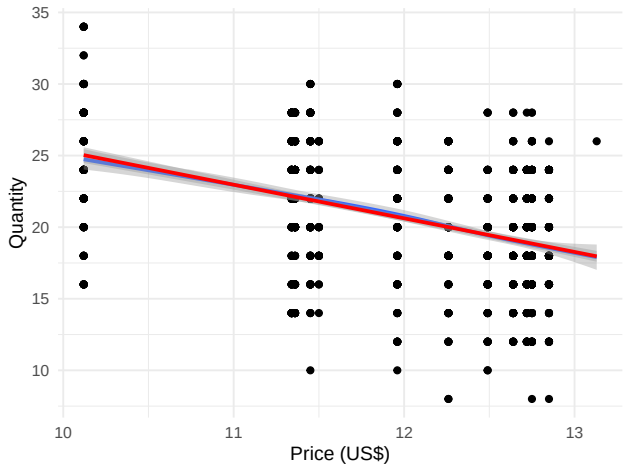
```
## Warning: Failed to fit group -1.
## Caused by error in `smooth.construct.cr.smooth.spec()`:
## ! x has insufficient unique values to support 10 knots: reduce k.
```

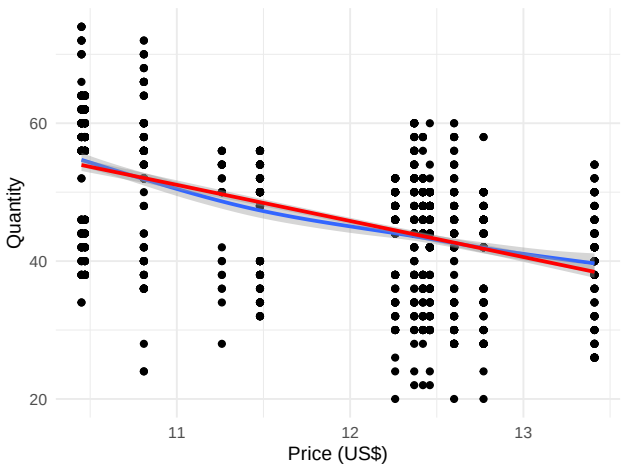
```
## `geom_smooth()` using formula = 'y ~ x'
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```



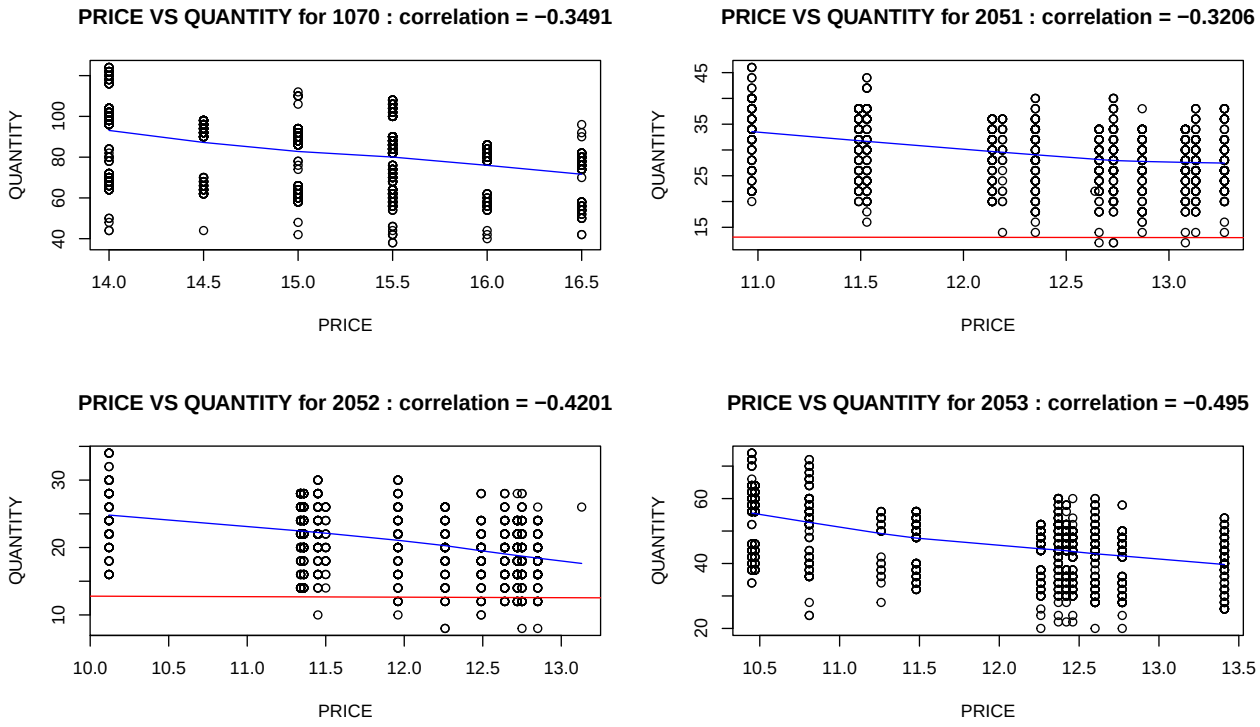







```
scatterplot <- function(dfrm, var1, var2, sell_id){
  correlation <- round(cor(data[[var1]], data[[var2]]),4)
  plot(data[[var1]], data[[var2]], xlab = var1, ylab=var2, main = paste(var1, 'VS', var2, 'for', sell_id
  # regression line (y~x)
  abline(lm(as.formula(paste(var1,'~',var2)), data =data), col='red')
  # lowess line (x,y)
  lines(lowess(data[[var1]], data[[var2]]), col='blue')
}

library(dplyr)

par(mfrow=c(2,2))

data <- kiosk_data %>% filter(SELL_ID == "1070")
scatterplot(data, 'PRICE', 'QUANTITY', '1070')


data <- kiosk_data %>% filter(SELL_ID == "2051")
scatterplot(data, 'PRICE', 'QUANTITY', '2051')


data <- kiosk_data %>% filter(SELL_ID == "2052")
```

```
scatterplot(data, 'PRICE', 'QUANTITY', '2052')


data <- kiosk_data %>% filter(SELL_ID == "2053")
scatterplot(data, 'PRICE', 'QUANTITY', '2053')
```




```
par(mfrow=c(1,1))
```

Exporting cleaned csv file

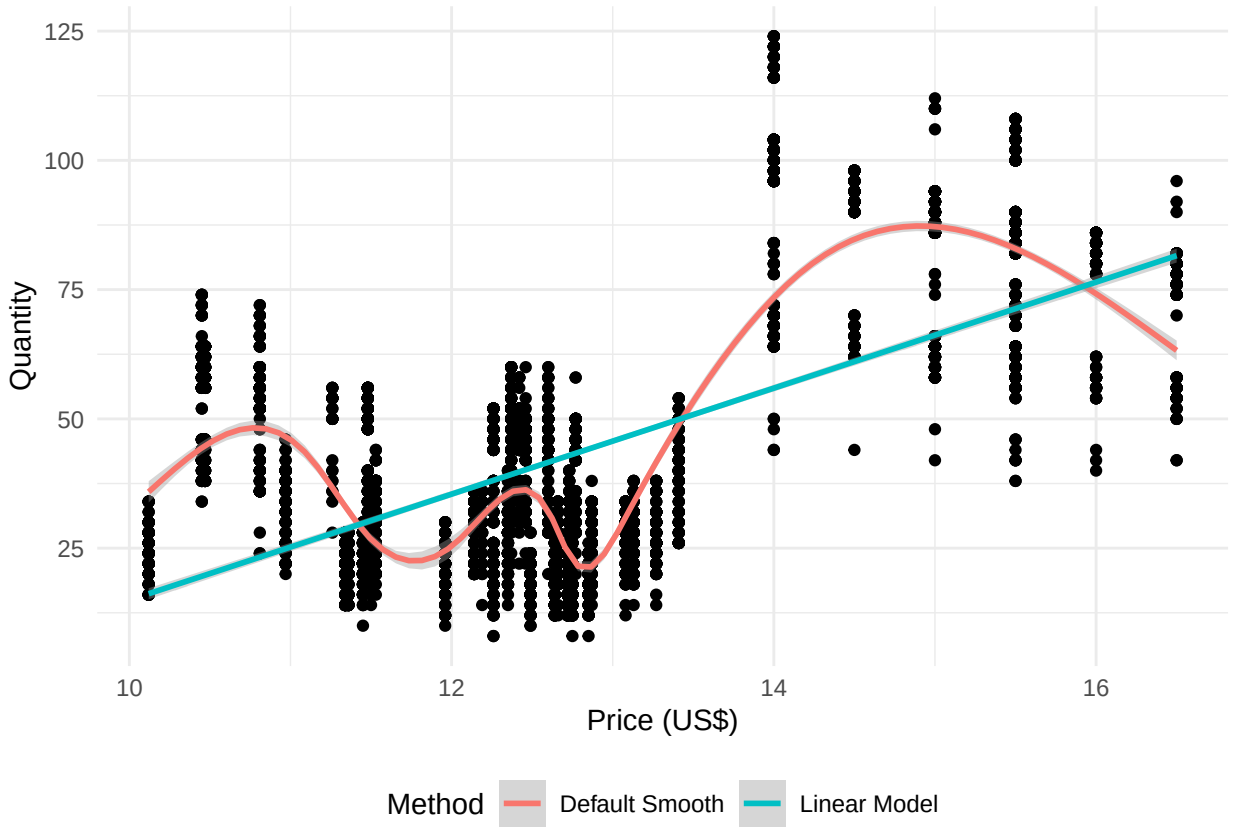
```
#csv
write_csv(kiosk_data,"Food_Kiosk.csv")
```

Building a linear regression model

Plotting a relationship

```
ggplot(kiosk_data, aes(x=PRICE, y=QUANTITY)) +
  geom_point() +
  geom_smooth(aes(color = "Default Smooth")) +
  geom_smooth(method="lm", aes(color = "Linear Model")) +
  labs(x="Price (US$)", y="Quantity",  color = "Method") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using formula = 'y ~ x'
```

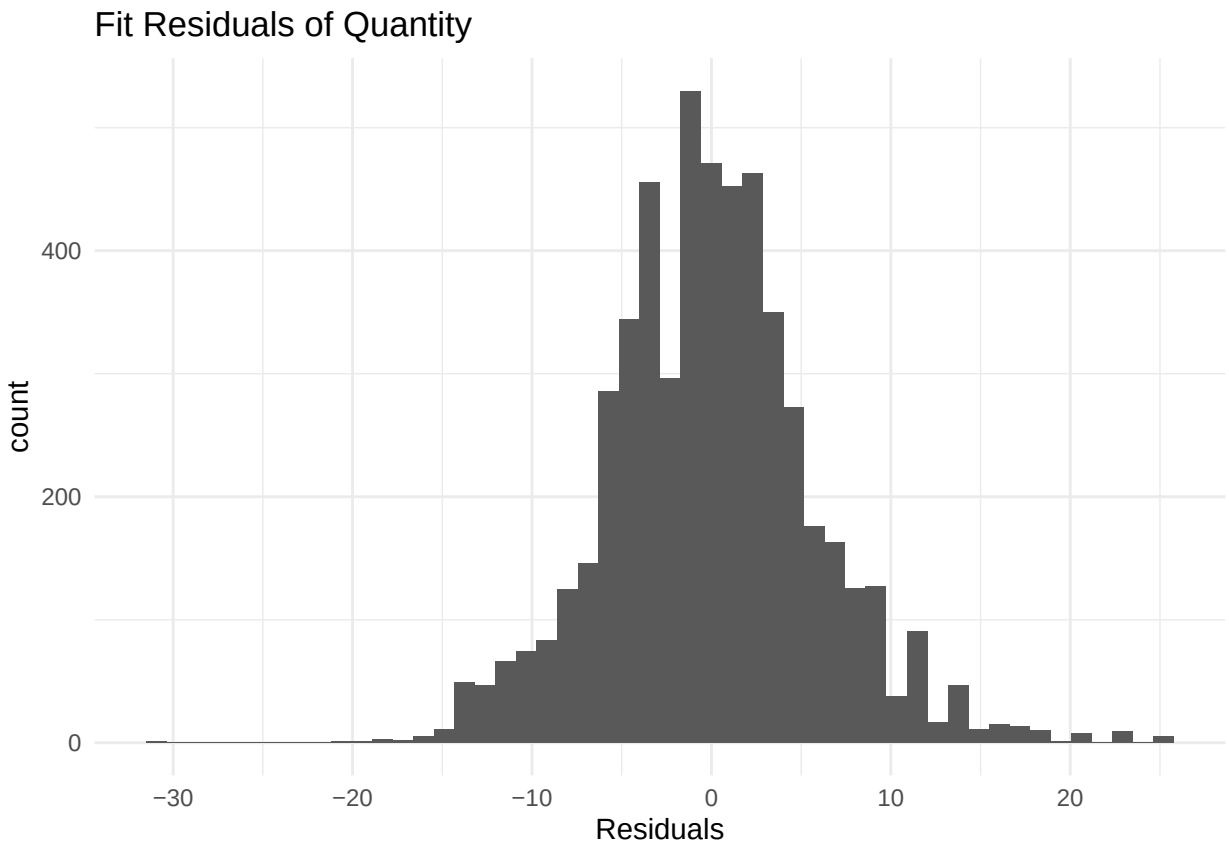



```
# Fit a linear model (lm)
cafe_model <- lm(QUANTITY ~ PRICE + SELL_ID +  HOLIDAY + IS_WEEKEND +
                 + IS_OUTDOOR
                , data = kiosk_data)
summary(cafe_model)
```

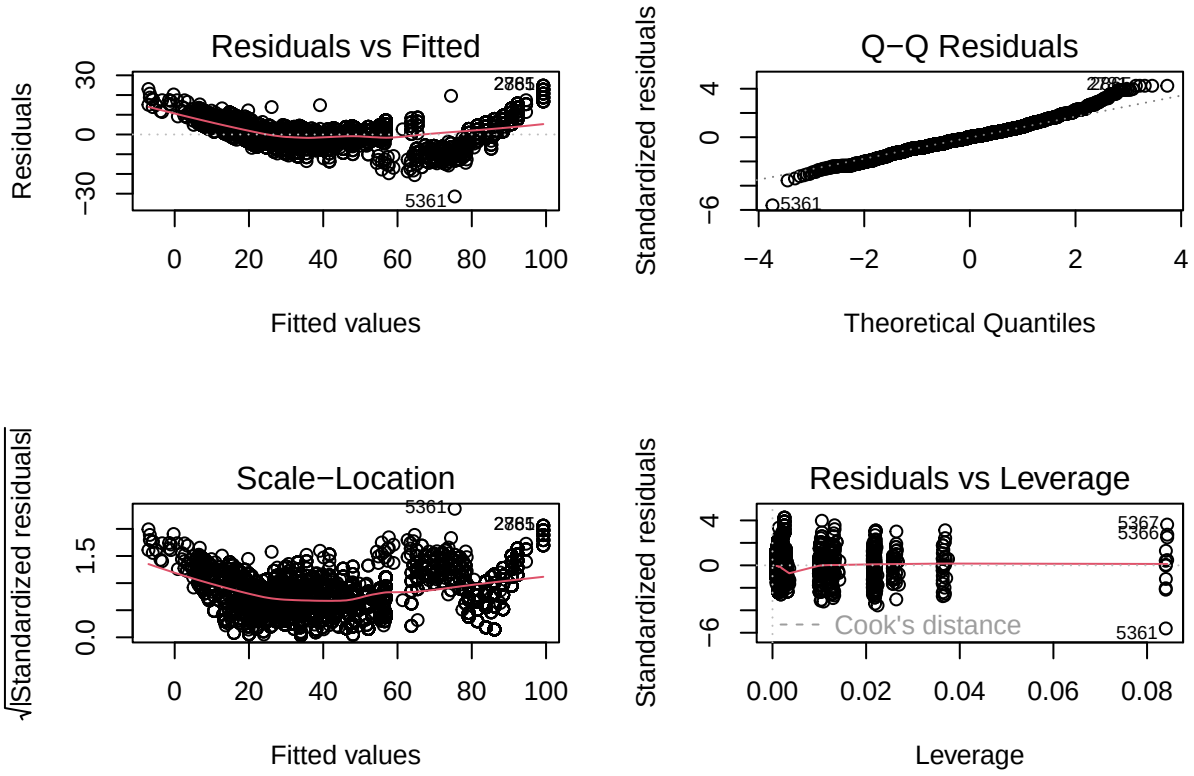
```
##
## Call:
## lm(formula = QUANTITY ~ PRICE + SELL_ID + HOLIDAY + IS_WEEKEND +
##     +IS_OUTDOOR, data = kiosk_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -31.4025  -3.6286  -0.0382   3.2027  24.7459
##
## Coefficients:
##                              Estimate Std. Error  t value Pr(>|t|)
## (Intercept)                 1.511e+02  1.754e+00   86.144   <2e-16 ***
## PRICE                      -4.634e+00  9.884e-02  -46.883   <2e-16 ***
## SELL_ID2051                -6.533e+01  3.558e-01 -183.654   <2e-16 ***
## SELL_ID2052                -7.557e+01  3.868e-01 -195.399   <2e-16 ***
## SELL_ID2053                -5.029e+01  3.904e-01 -128.822   <2e-16 ***
## HOLIDAYLabor Day            4.228e-01  1.192e+00    0.355    0.723
## HOLIDAYLuner New Year      -9.821e-01  1.016e+00   -0.967    0.334
## HOLIDAYMid-Autumn Day       6.436e-01  1.389e+00    0.463    0.643
## HOLIDAYNational Day        -1.492e-01  1.057e+00   -0.141    0.888
## HOLIDAYNew Year            -1.776e+00  1.262e+00   -1.407    0.159
## HOLIDAYNo Holiday           1.303e+01  8.476e-01   15.378   <2e-16 ***
```

```
## HOLIDAYQing Ming Festival  9.101e-01  1.192e+00    0.763    0.445
## HOLIDAYWWII Celebration   -3.907e-04  1.886e+00    0.000    1.000
## IS_WEEKEND1               -1.402e+01  1.763e-01  -79.544   <2e-16 ***
## IS_OUTDOOR1               -8.500e+00  2.376e-01  -35.776   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.84 on 5377 degrees of freedom
## Multiple R-squared:  0.9477, Adjusted R-squared:  0.9475
## F-statistic:  6956 on 14 and 5377 DF,  p-value: < 2.2e-16
```

```
kiosk_data %>%
  add_residuals(cafe_model, "resid") %>%
  ggplot(aes(x=resid)) +
  geom_histogram(bins=50) +
  labs(title = "Fit Residuals of Quantity", x="Residuals") +
  theme_minimal()
```




```
# Diagnostic plots
par(mfrow=c(2,2))
plot(cafe_model)
```



PREDICTING OPTIMAL PRICE

```
# Create a new data frame with the same structure as 'kiosk_data' for the predictors
new_data <- data.frame(
  PRICE = c(9,15),
  SELL_ID = c("2051","2051"),
  HOLIDAY = c("No Holiday","No Holiday"),
  IS_WEEKEND = c("0","1"),
  IS_OUTDOOR = c("1","1")
)

# Use the model to predict quantity
predicted_quantity <- predict(cafe_model, newdata = new_data)

predicted_quantity
```

```
##        1        2
## 48.58850  6.76487
```

```
# The 'predicted_quantity' object now contains the predicted quantities based on your model
```