# Final Project

Emmanuel Agbeli

2025-12-07

**Exploratory Analysis of Custom Datasets.**

Dataset source:

The goal of the project is understand the consumer behaviour of different products living in Turkey. The dataset has different information of these customers. The dataset was fetch from kaggle dataset.

```
# read the dataset.
dataCommerce <- read.csv("ecommerce_customer_behavior_dataset_v2.csv")
```

- Generate new features from the whole dataset where we aggregate the numeric informations. We sample 1000 individuals information where the code snippet shows the implementation pipeline of these datasets.

```
# extract unique customer ids.
df_customer <- dataCommerce %>%
  group_by(Customer_ID) %>%
  summarise(
    first_date = min(Date),
    last_date = max(Date),
    total_spend = sum(Total_Amount),
    total_quantity = sum(Quantity),
    avg_unit_price = mean(Unit_Price),
    total_discount = sum(Discount_Amount),
    age = first(Age),  # assuming age is constant per customer
    n_orders = n()
  )

# convert into actual date
df_customer <- df_customer %>%
  mutate(
    first_date = as.Date(first_date),
    last_date = as.Date(last_date)
  )

df_customer <- df_customer %>%
  mutate(
    # Days between first and last purchase
    customer_lifespan = as.numeric(last_date - first_date),

    # How recent is the customer (from dataset end date)
    recency = as.numeric(max(last_date) - last_date),
```

```
    # Which month they started
    start_month = month(first_date),

    end_month = month(last_date)
  )

# sample information from the aggregated data.
sampled_data <- df_customer %>%
  sample_frac(500 / nrow(df_customer))

# view the first six rows
head(sampled_data)
```

```
## # A tibble: 6 x 13
##   Customer_ID first_date last_date  total_spend total_quantity avg_unit_price
##   <chr>       <date>     <date>           <dbl>          <int>          <dbl>
## 1 CUST_04969  2023-07-12 2023-07-14        289.              8           45.6
## 2 CUST_02707  2023-01-15 2023-12-18       8741.             11          649.
## 3 CUST_01528  2024-01-05 2024-01-05       1033.              5          232.
## 4 CUST_02779  2023-04-01 2023-11-17       6914.              7          730.
## 5 CUST_01567  2023-08-01 2024-03-08        534.              6           95.0
## 6 CUST_02594  2023-12-17 2023-12-17        360.              1          360.
## # i 7 more variables: total_discount <dbl>, age <int>, n_orders <int>,
## #   customer_lifespan <dbl>, recency <dbl>, start_month <dbl>, end_month <dbl>
```

This is an Ecommence dataset showing the behavior of customers with 18 variables and 17049 rows representing the number of records. In this project we explore the behavior of consumers. Due to the large nature of the dataset we heavily rely on visual analysis instead numeric representation since this gives a better understanding the datasets.

```
# column ames
cat("Column names: \n")
```

```
## Column names:
```

```
print(colnames(sampled_data))
```

```
##  [1] "Customer_ID"       "first_date"        "last_date"
##  [4] "total_spend"       "total_quantity"    "avg_unit_price"
##  [7] "total_discount"    "age"               "n_orders"
## [10] "customer_lifespan" "recency"           "start_month"
## [13] "end_month"
```

```
cat("\n data structure \n")
```

```
##
##  data structure
```

```r
print(str(sampled_data))
```

```
## tibble [500 x 13] (S3: tbl_df/tbl/data.frame)
##  $ Customer_ID      : chr [1:500] "CUST_04969" "CUST_02707" "CUST_01528" "CUST_02779" ...
##  $ first_date       : Date[1:500], format: "2023-07-12" "2023-01-15" ...
##  $ last_date        : Date[1:500], format: "2023-07-14" "2023-12-18" ...
##  $ total_spend      : num [1:500] 289 8741 1033 6914 534 ...
##  $ total_quantity   : int [1:500] 8 11 5 7 6 1 7 15 12 11 ...
##  $ avg_unit_price   : num [1:500] 45.6 648.8 231.8 730.4 95 ...
##  $ total_discount   : num [1:500] 75.38 230.06 126.63 0 3.91 ...
##  $ age              : int [1:500] 31 36 53 34 35 35 18 31 18 48 ...
##  $ n_orders         : int [1:500] 2 4 1 2 3 1 2 5 5 3 ...
##  $ customer_lifespan: num [1:500] 2 337 0 230 220 0 43 254 261 298 ...
##  $ recency          : num [1:500] 255 98 80 129 17 99 240 167 52 105 ...
##  $ start_month      : num [1:500] 7 1 1 4 8 12 6 1 5 2 ...
##  $ end_month        : num [1:500] 7 12 1 11 3 12 7 10 2 12 ...
## NULL
```

### Preprocessing of Datasets

In this case, we will preprocess the datasets where we treat relevant columns of interest and also work with relevant columns since we cannot work with all the 18 variables.

- Summary Statistics

This is summary statistics of the various columns where we have to convert the character columns into category. We have to regenerate new information from the dataframe and also extract unique customer IDs where we aggregate the consumer behaviours based on relevant features.

```r
summary(sampled_data)
```

```
##  Customer_ID          first_date           last_date
##  Length:500         Min.   :2023-01-01   Min.   :2023-01-02
##  Class :character   1st Qu.:2023-02-10   1st Qu.:2023-09-20
##  Mode  :character   Median :2023-04-09   Median :2023-12-23
##                     Mean   :2023-05-14   Mean   :2023-11-24
##                     3rd Qu.:2023-08-04   3rd Qu.:2024-02-18
##                     Max.   :2024-03-25   Max.   :2024-03-25
##   total_spend       total_quantity  avg_unit_price    total_discount
##  Min.   :   25.78   Min.   : 1.00   Min.   :  15.12   Min.   :   0.0
##  1st Qu.:  868.30   1st Qu.: 5.00   1st Qu.: 149.66   1st Qu.:   0.0
##  Median : 2452.34   Median : 9.00   Median : 287.70   Median :  67.3
##  Mean   : 4534.33   Mean   :10.55   Mean   : 427.00   Mean   : 266.4
##  3rd Qu.: 6252.15   3rd Qu.:14.25   3rd Qu.: 564.48   3rd Qu.: 283.3
##  Max.   :28871.17   Max.   :39.00   Max.   :4582.58   Max.   :5158.9
##       age           n_orders      customer_lifespan    recency
##  Min.   :18.00   Min.   : 1.000   Min.   :  0.00    Min.   :  0.0
##  1st Qu.:27.00   1st Qu.: 2.000   1st Qu.: 43.75    1st Qu.: 36.0
##  Median :35.00   Median : 3.000   Median :202.00    Median : 93.0
##  Mean   :35.19   Mean   : 3.506   Mean   :194.27    Mean   :121.4
##  3rd Qu.:43.00   3rd Qu.: 5.000   3rd Qu.:327.50    3rd Qu.:186.2
```

```
##  Max.   :75.00   Max.   :10.000   Max.   :442.00    Max.   :448.0
##   start_month      end_month
##  Min.   : 1.00   Min.   : 1.000
##  1st Qu.: 2.00   1st Qu.: 2.000
##  Median : 3.00   Median : 4.000
##  Mean   : 4.35   Mean   : 5.798
##  3rd Qu.: 7.00   3rd Qu.:10.000
##  Max.   :12.00   Max.   :12.000
```

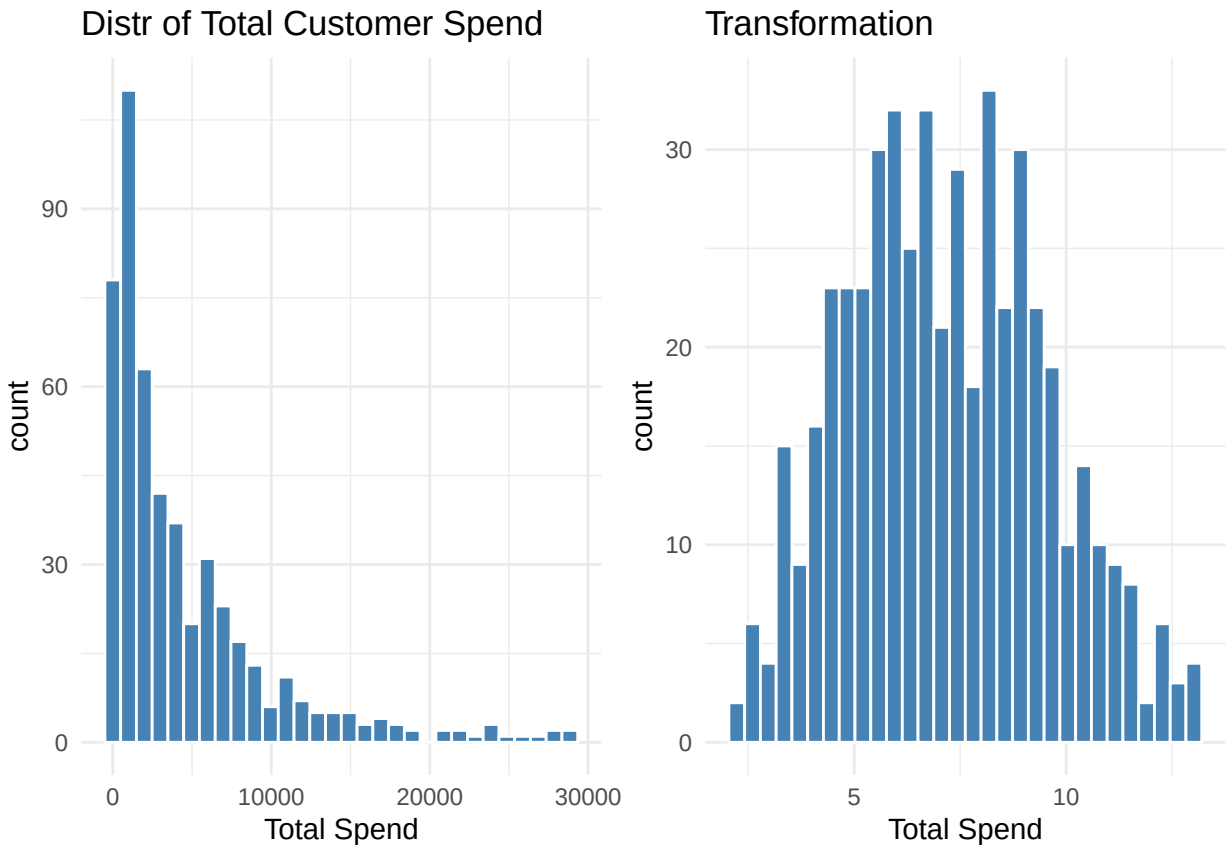
### Exploratory Analysis

In the first we would like to explore the univariate analysis of the continuous information in the dataframe. This is to understand the symmetric nature of these features and also transformation required.

- Distribution of the total spending of customers

```
# Total spend
sp1 <- ggplot(sampled_data, aes(x = total_spend)) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  labs(title = "Distr of Total Customer Spend", x = "Total Spend") +
  theme_minimal()

sp2 <- ggplot(sampled_data, aes(x = total_spend^(1/4))) +
  geom_histogram(bins = 30, fill = "steelblue", color = "white") +
  labs(title = "Transformation", x = "Total Spend") +
  theme_minimal()
grid.arrange(sp1, sp2, ncol = 2)
```




- Hinkley measure to assess the symmetric nature of the total spending of customers. We observed that distribution of spending is right-skewed and after we had to transform the features using fourth-root to make it symmetric as showed in the graph above.

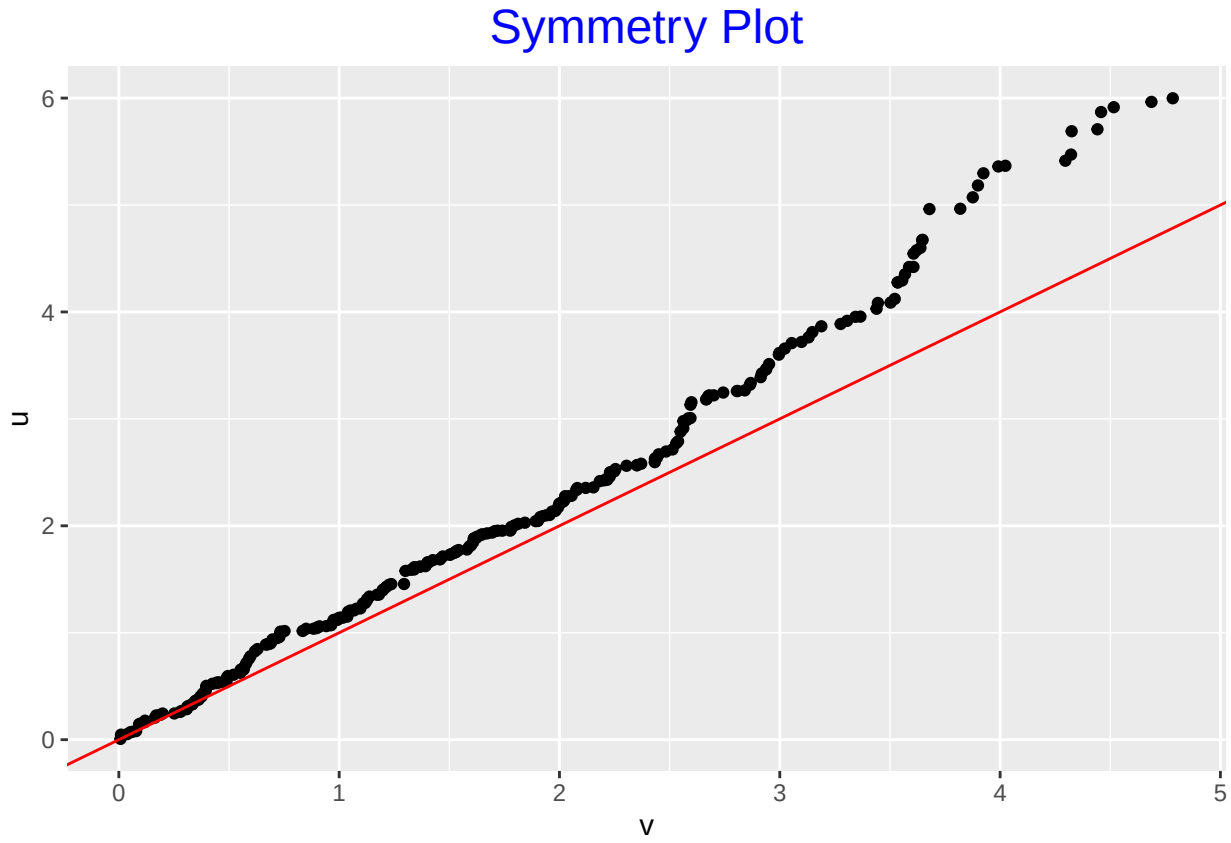
```
hinkley(sampled_data$total_spend)
```

```
##         h
## 0.3867116
```

```
hinkley(sampled_data$total_spend^(1/4))
```

```
##          h
## 0.04642867
```

```
# symmetric plot
symplot(sampled_data$total_spend^(1/4))
```




- five number analysis where we assess the total spending less than median **2815.81** and also for the transformed data. Check for outlier in the total spending

```
fivenum(sampled_data$total_spend)
```

```
## [1]    25.780   867.810  2452.335  6278.335 28871.170
```

```
fivenum(sampled_data$total_spend^(1/4))
```

```
## [1]  2.253309  5.427578  7.037107  8.901399 13.035152
```

```
cat("stem and leaf plot of total spending")
```

```
## stem and leaf plot of total spending
```

```
aplpack::stem.leaf(sampled_data$total_spend)
```

```
## 1 | 2: represents 1200
##  leaf unit: 100
##             n: 500
##    140     0 | 00000000000001111111111111111111222222222222223333333333333333344444444444444455555555
##    226     1 | 00000000000111111122222222222233333333333444444455555666666667777777888888899999999999
```

```
##   (48)    2 | 00000000011122223333334445555566677778888899999
##   226     3 | 000001112222222334455566678899999
##   194     4 | 00011222222223334444444445566667788999
##   156     5 | 001122555566667777899999
##   131     6 | 0001123334444555556677789999
##   103     7 | 0013334555777779
##    87     8 | 000123334555677789
##    69     9 | 02367
##    64    10 | 01116799
##    56    11 | 00012225589
##    45    12 | 11388
##    40    13 | 03368
##    35    14 | 12356
##    30    15 | 0235
## HI: 16381.39 16383.44 16486.29 16828.5 17233.78 17235.12 17998.07 18191.45 18333.07 18795.72 18811.4
```

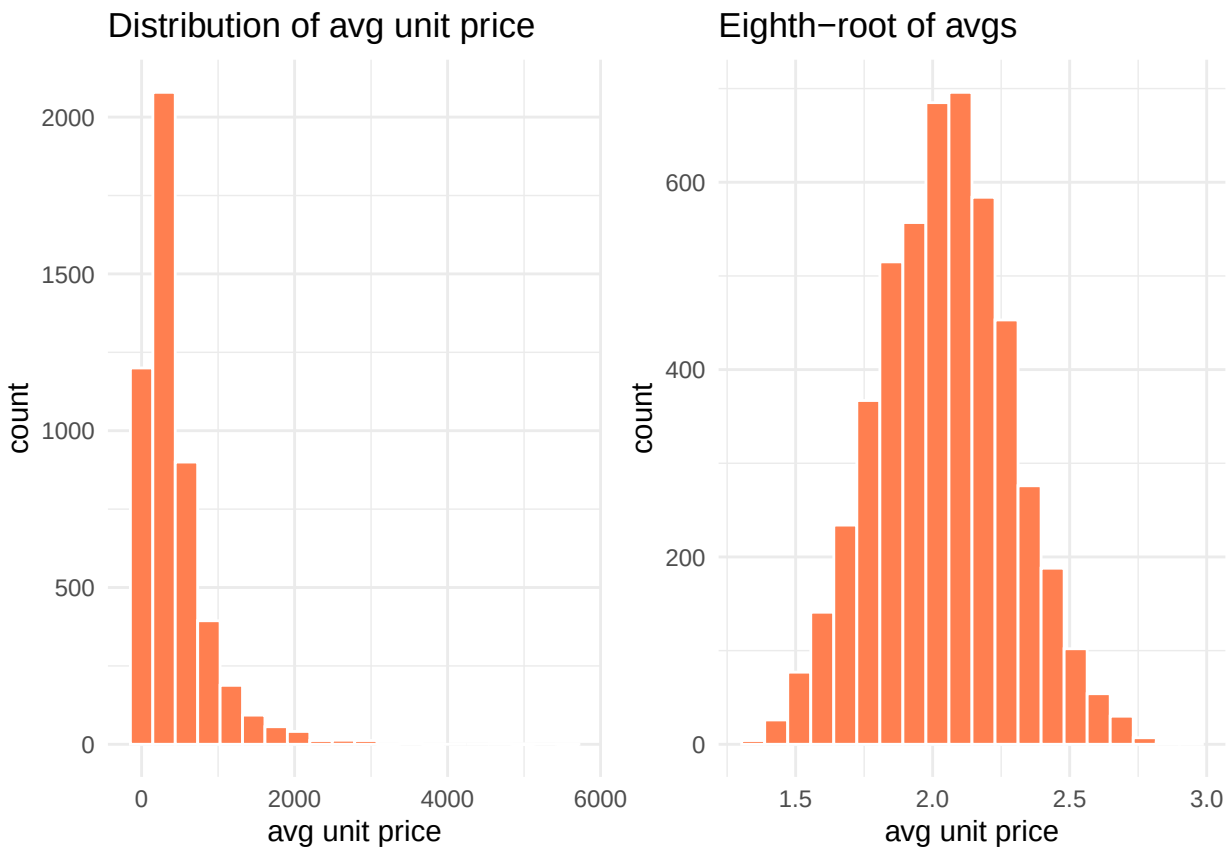
- Average Unit price of items purchased by customers.

From the representation of the distribution, we observed that the distribution is right-skewed one and needs to be reexpressed using the eighth-root into a symmetric distribution.

```
# Age
age1 <- ggplot(df_customer, aes(x = avg_unit_price)) +
  geom_histogram(bins = 20, fill = "coral", color = "white") +
  labs(title = "Distribution of avg unit price", x = "avg unit price") +
  theme_minimal()

age2 <- ggplot(df_customer, aes(x = avg_unit_price^(1/8))) +
  geom_histogram(bins = 20, fill = "coral", color = "white") +
  labs(title = "Eighth-root of avgs", x = "avg unit price") +
  theme_minimal()

grid.arrange(age1, age2, ncol = 2)
```




This confirms the how transformation has impacted the symmetric nature of the distribution of the unit price of items purchased by the customers.

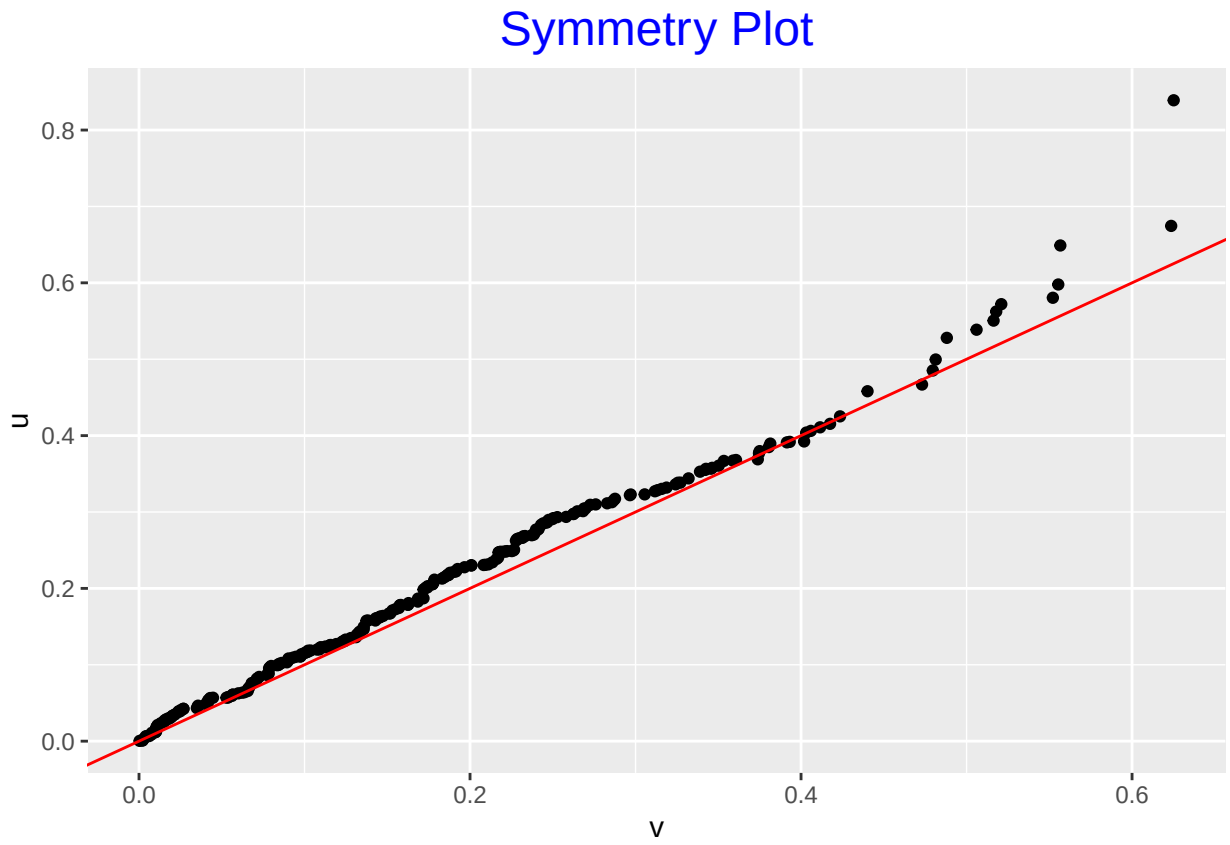
```
hinkley(sampled_data$avg_unit_price)
```

```
##         h
## 0.3358147
```

```
hinkley(sampled_data$avg_unit_price^(1/8))
```

```
##          h
## 0.02769427
```

```
symplot(sampled_data$avg_unit_price^(1/8))
```




We went further to compute for the letter values of the raw data and transformed datasets as showed in the table below.

```
lval(sampled_data$avg_unit_price)
```

```
##   depth        lo        hi      mids   spreads
## M 250.5 287.70083  287.7008  287.7008    0.0000
## H 125.5 148.96900  564.6180  356.7935  415.6490
## E  63.0 100.58000  839.6800  470.1300  739.1000
## D  32.0  64.38333 1054.7725  559.5779  990.3892
## C  16.5  46.25250 1266.8770  656.5648 1220.6245
## B   8.5  28.23500 1927.1517  977.6933 1898.9167
## A   4.5  22.48000 2211.1383 1116.8092 2188.6583
## Z   2.5  18.69000 2752.5825 1385.6362 2733.8925
## Y   1.0  15.12000 4582.5800 2298.8500 4567.4600
```

```
lval(sampled_data$avg_unit_price^(1/8))
```

```
##   depth       lo       hi      mids   spreads
## M 250.5 2.029400 2.029400 2.029400 0.0000000
## H 125.5 1.869110 2.207849 2.038480 0.3387386
## E  63.0 1.779565 2.320140 2.049853 0.5405748
## D  32.0 1.683049 2.387233 2.035141 0.7041845
## C  16.5 1.614867 2.442533 2.028700 0.8276666
## B   8.5 1.518207 2.573987 2.046097 1.0557800
```

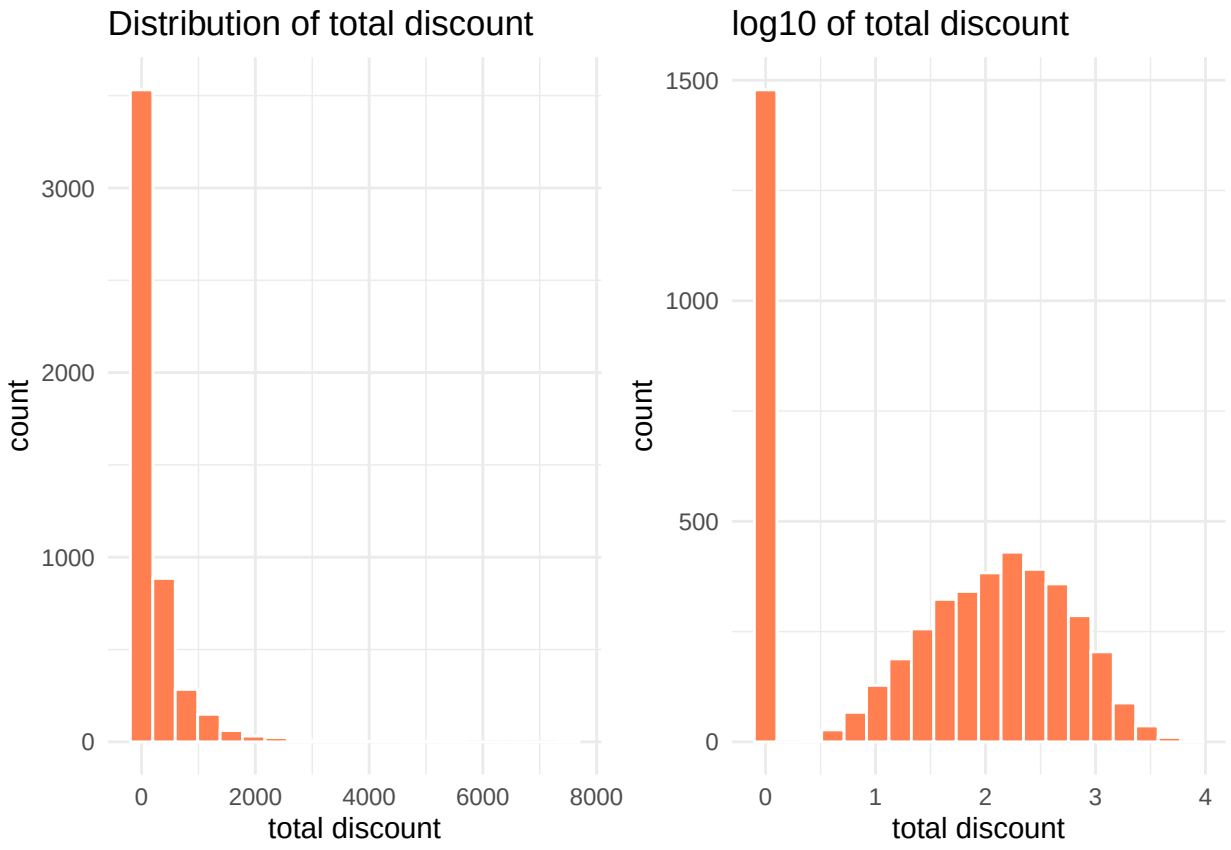
```
## A   4.5 1.475614 2.618546 2.047080 1.1429327
## Z   2.5 1.439240 2.691120 2.065180 1.2518802
## Y   1.0 1.404249 2.868394 2.136321 1.4641452
```

- We also consider the distribution of the total discount offered to customer we observed that the distribution is right-skewed one, so we have to readjust to make it look symmetric in nature. Though after transformation the graph looks a little bit symmetric with an extreme value.

```
# Age
age1 <- ggplot(df_customer, aes(x = total_discount)) +
  geom_histogram(bins = 20, fill = "coral", color = "white") +
  labs(title = "Distribution of total discount", x = "total discount") +
  theme_minimal()

age2 <- ggplot(df_customer, aes(x = log10(total_discount +1))) +
  geom_histogram(bins = 20, fill = "coral", color = "white") +
  labs(title = "log10 of total discount", x = "total discount") +
  theme_minimal()

grid.arrange(age1, age2, ncol = 2)
```




```
hinkley(sampled_data$total_discount)
```

```
##         h
## 0.7027749
```

```
hinkley(log10(sampled_data$total_discount + 1))
```

```
##          h
## -0.1247846
```

The tables shows the measure of various letter values from the previous analysis.

```
lval(sampled_data$total_discount)
```

```
##   depth     lo       hi      mids  spreads
## M 250.5 67.295   67.295   67.2950    0.000
## H 125.5  0.000  285.310  142.6550  285.310
## E  63.0  0.000  624.810  312.4050  624.810
## D  32.0  0.000 1103.750  551.8750 1103.750
## C  16.5  0.000 1483.705  741.8525 1483.705
## B   8.5  0.000 2264.695 1132.3475 2264.695
## A   4.5  0.000 2789.565 1394.7825 2789.565
## Z   2.5  0.000 3115.300 1557.6500 3115.300
## Y   1.0  0.000 5158.850 2579.4250 5158.850
```

```
lval(log10(sampled_data$total_discount + 1))
```

```
##   depth       lo       hi     mids  spreads
## M 250.5 1.834389 1.834389 1.834389 0.000000
## H 125.5 0.000000 2.456792 1.228396 2.456792
## E  63.0 0.000000 2.796442 1.398221 2.796442
## D  32.0 0.000000 3.043264 1.521632 3.043264
## C  16.5 0.000000 3.171638 1.585819 3.171638
## B   8.5 0.000000 3.355185 1.677592 3.355185
## A   4.5 0.000000 3.445664 1.722832 3.445664
## Z   2.5 0.000000 3.492985 1.746493 3.492985
## Y   1.0 0.000000 3.712637 1.856319 3.712637
```

- Bivariate Analysis (Batch Analysis)

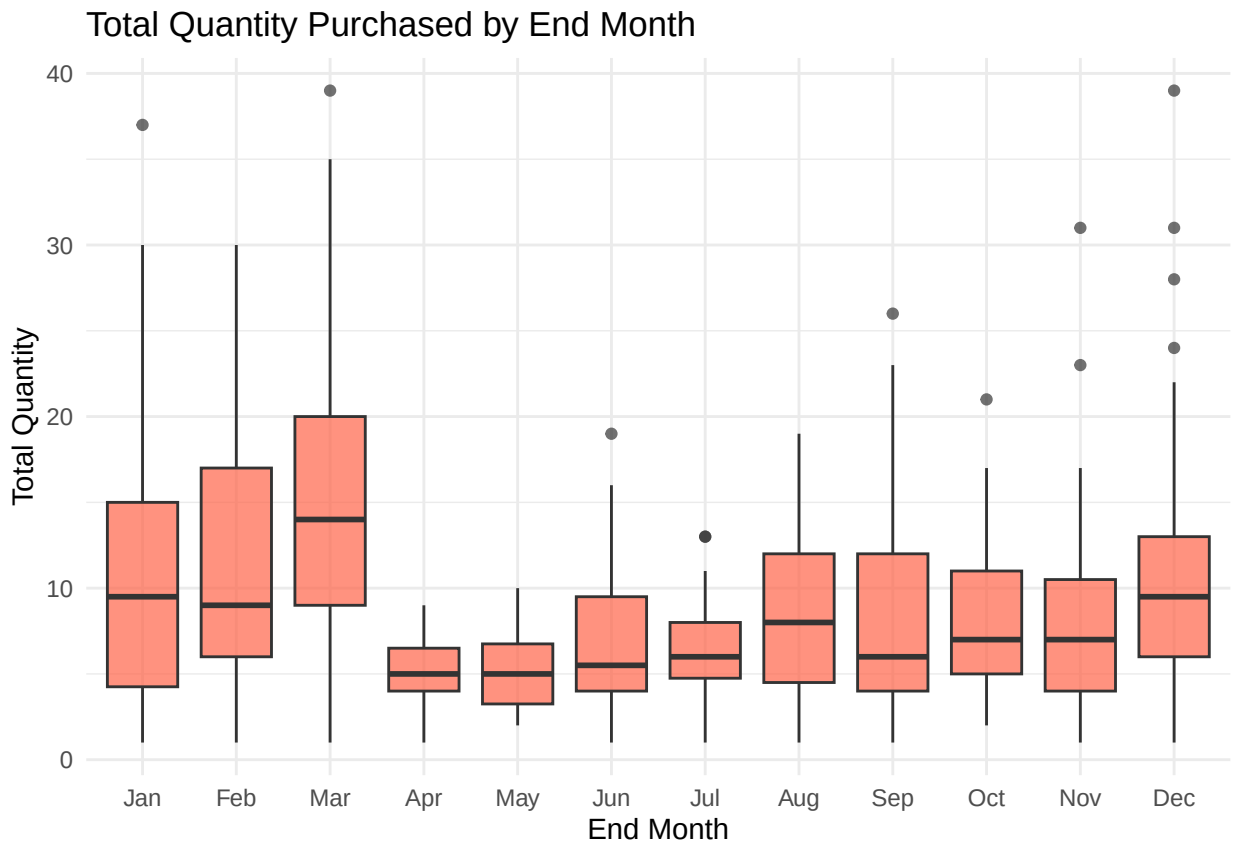
Next we consider a bivariate analysis of the features previously analysed considering the end month of the customer in other words the last purchased made based on months.

- The total amount spend over the end month?

We investigated the distribution of total spending over the months using boxplot to indicate the spread over the end month of purchase by customers. We can see that there were more purchase extreme spending made on march compared to the rest.

```
sampled_data <- sampled_data %>%
  mutate(
    start_month = month(first_date, label = TRUE, abbr = TRUE),  #Feb,Mar...
    end_month = month(last_date, label = TRUE, abbr = TRUE)
  )
```

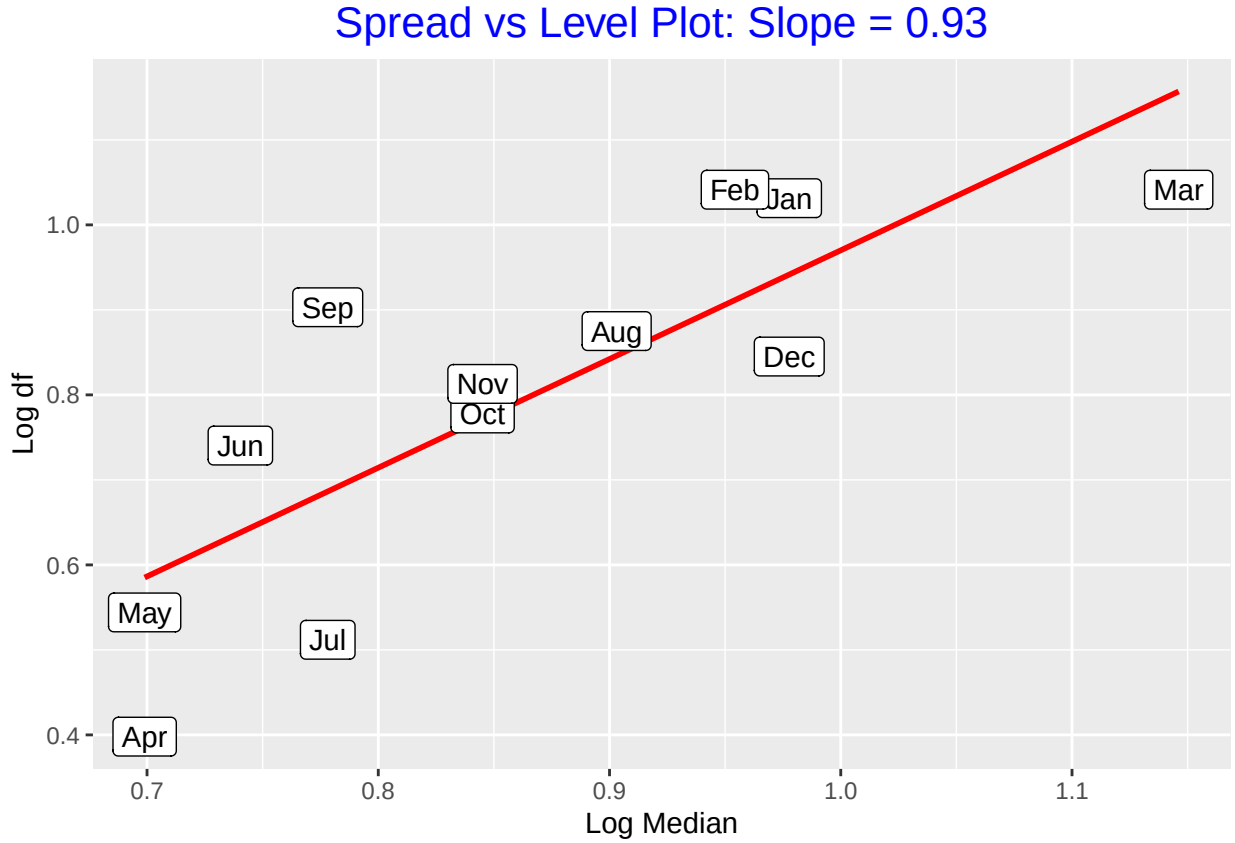
```r
ggplot(sampled_data, aes(x = factor(end_month), y = total_quantity)) +
  geom_boxplot(fill = "tomato", alpha = 0.7) +
  labs(
    title = "Total Quantity Purchased by End Month",
    x = "End Month",
    y = "Total Quantity"
  ) +
  theme_minimal()
```




```r
spread_level_plot(sampled_data, total_quantity, end_month)
```

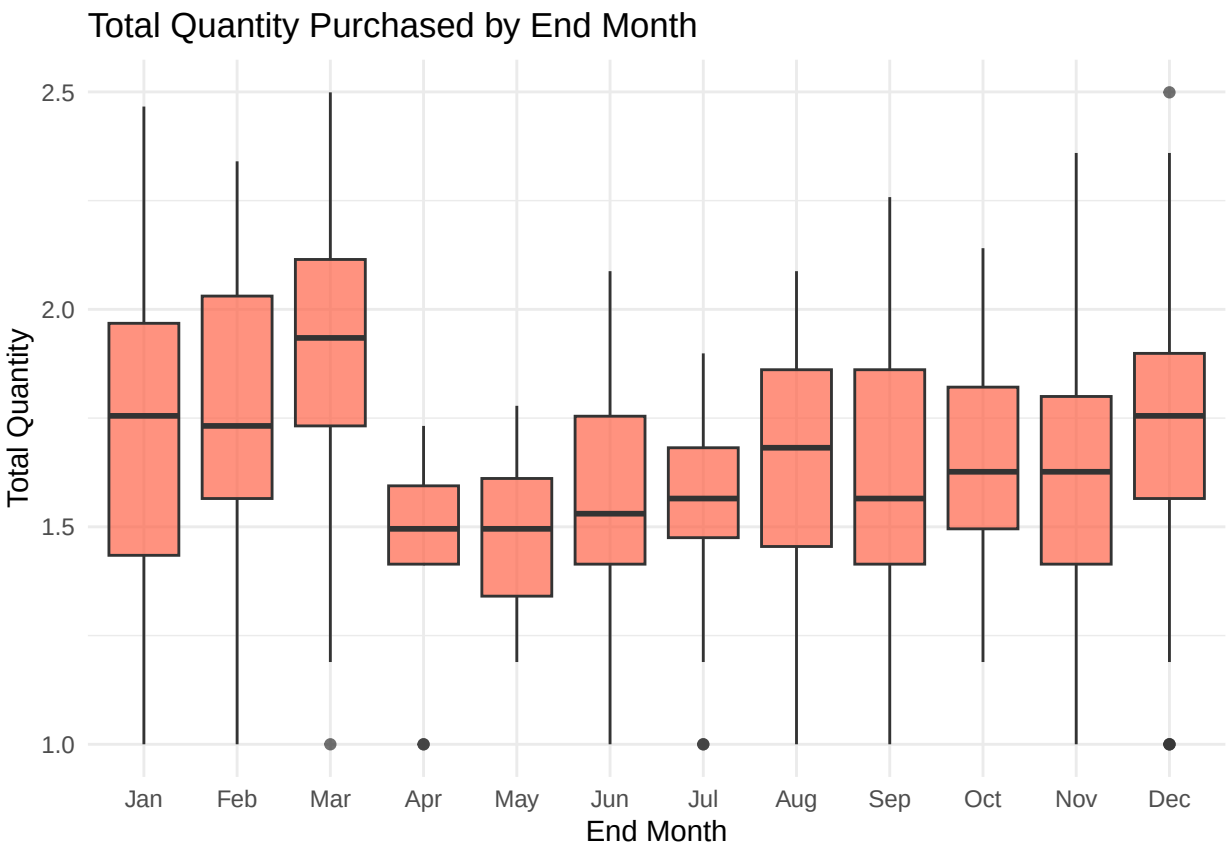
```
## `geom_smooth()` using formula = 'y ~ x'

## Warning: The following aesthetics were dropped during statistical transformation: label.
## 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?
```




```
## # A tibble: 12 x 5
##    end_month     M    df log.M log.df
##    <ord>     <dbl> <dbl> <dbl>  <dbl>
##  1 Jan         9.5 10.8  0.978  1.03
##  2 Feb         9   11    0.954  1.04
##  3 Mar        14   11    1.15   1.04
##  4 Apr         5    2.5  0.699  0.398
##  5 May         5    3.5  0.699  0.544
##  6 Jun         5.5  5.5  0.740  0.740
##  7 Jul         6    3.25 0.778  0.512
##  8 Aug         8    7.5  0.903  0.875
##  9 Sep         6    8    0.778  0.903
## 10 Oct         7    6    0.845  0.778
## 11 Nov         7    6.5  0.845  0.813
## 12 Dec         9.5  7    0.978  0.845
```

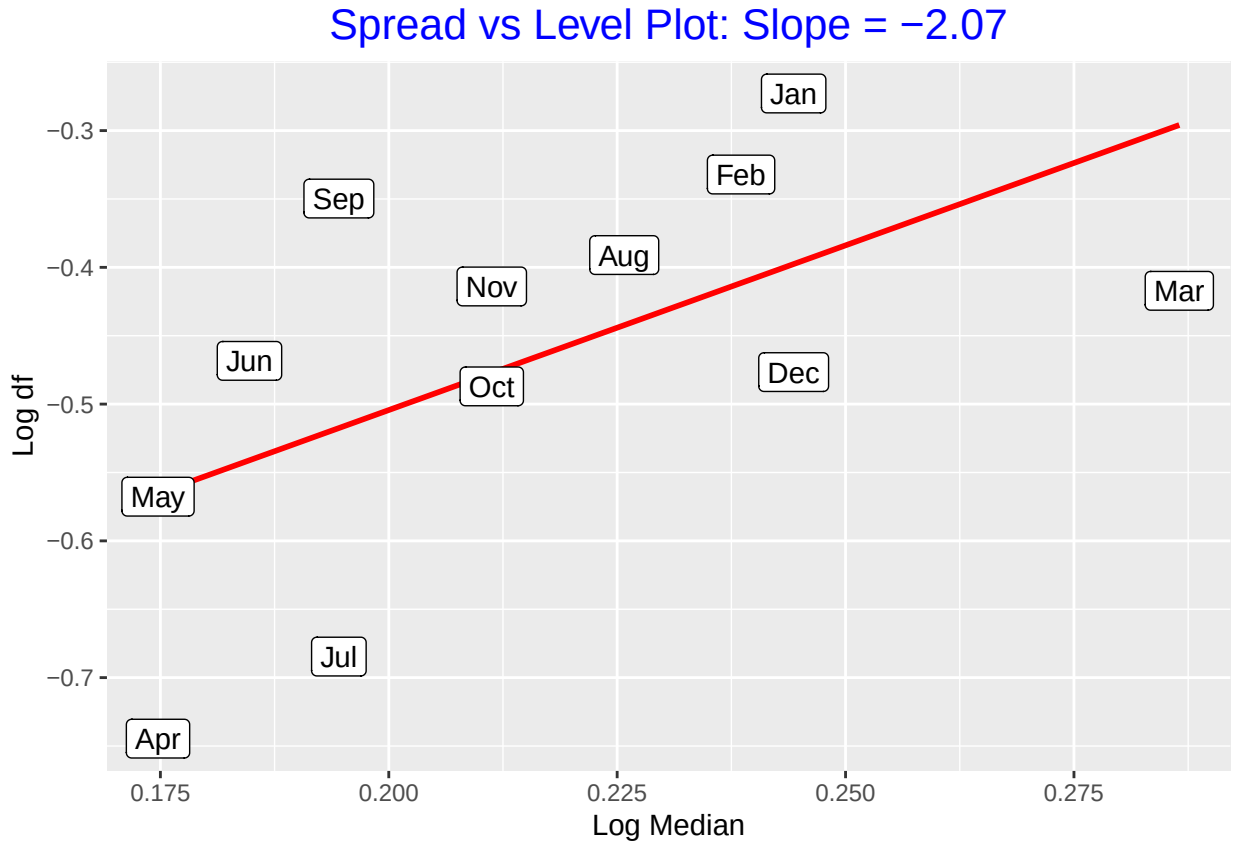
```r
# transformed features
ggplot(sampled_data, aes(x = factor(end_month), y = total_quantity^(1/4))) +
  geom_boxplot(fill = "tomato", alpha = 0.7) +
  labs(
    title = "Total Quantity Purchased by End Month",
    x = "End Month",
    y = "Total Quantity"
  ) +
  theme_minimal()
```




```
spread_level_plot(sampled_data, total_quantity^(1/4), end_month)
```

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

## Warning: The following aesthetics were dropped during statistical transformation: label.
## 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?
```



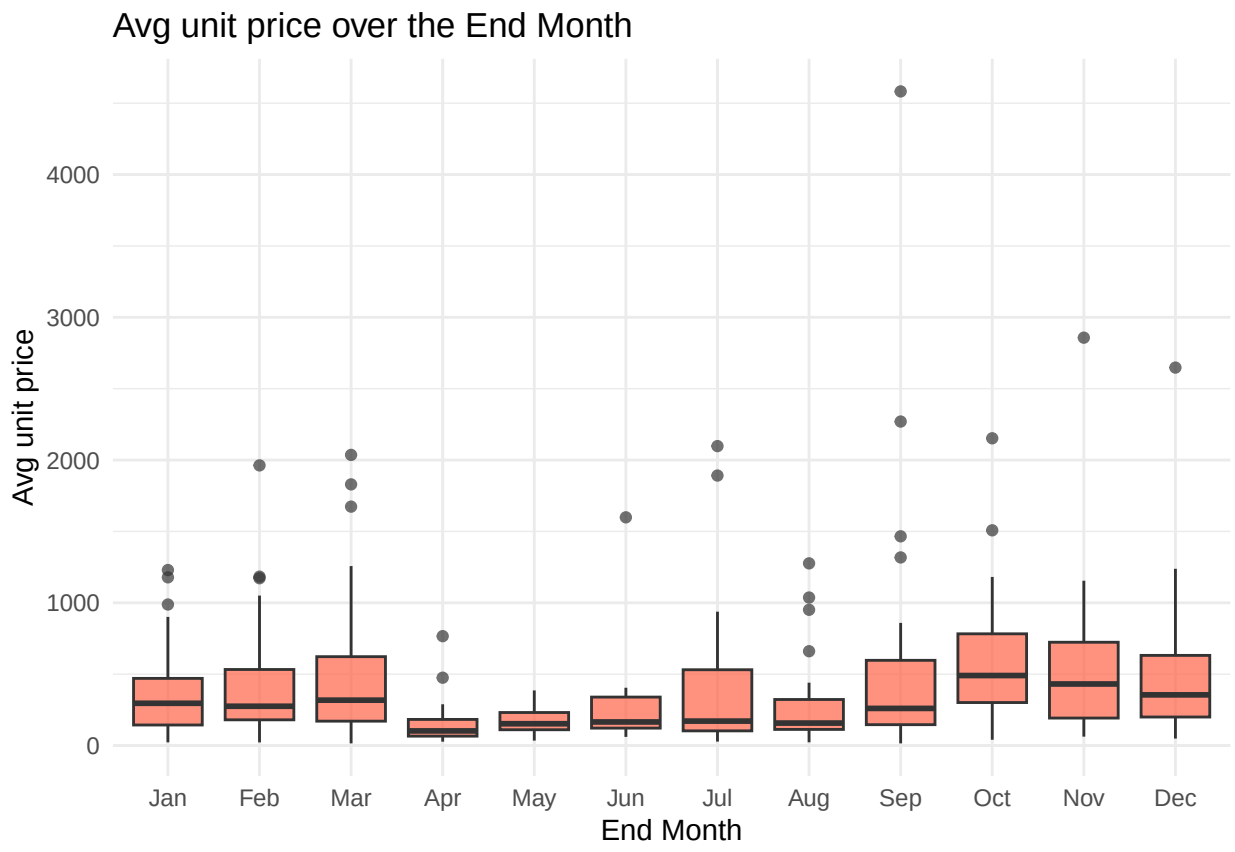

```
## # A tibble: 12 x 5
##    end_month     M    df log.M log.df
##    <ord>     <dbl> <dbl> <dbl>  <dbl>
##  1 Jan        1.76 0.533 0.244 -0.273
##  2 Feb        1.73 0.465 0.239 -0.332
##  3 Mar        1.93 0.383 0.287 -0.417
##  4 Apr        1.50 0.180 0.175 -0.745
##  5 May        1.50 0.271 0.175 -0.568
##  6 Jun        1.53 0.340 0.185 -0.468
##  7 Jul        1.57 0.207 0.195 -0.685
##  8 Aug        1.68 0.406 0.226 -0.391
##  9 Sep        1.57 0.447 0.195 -0.350
## 10 Oct        1.63 0.326 0.211 -0.487
## 11 Nov        1.63 0.386 0.211 -0.414
## 12 Dec        1.76 0.334 0.244 -0.477
```

```
#spread_level_plot(sampled_data, log(total_quantity,0.09), end_month)
```

- There is an uneven distribution of avg unit price over the end month of purchase by customers as indicated in the plot. There were a lot outliers in some months as showed in the figure below.

```
ggplot(sampled_data, aes(x = factor(end_month), y = avg_unit_price)) +
  geom_boxplot(fill = "tomato", alpha = 0.7) +
  labs(
```

```
    title = "Avg unit price over the End Month",
    x = "End Month",
    y = "Avg unit price"
  ) +
  theme_minimal()
```

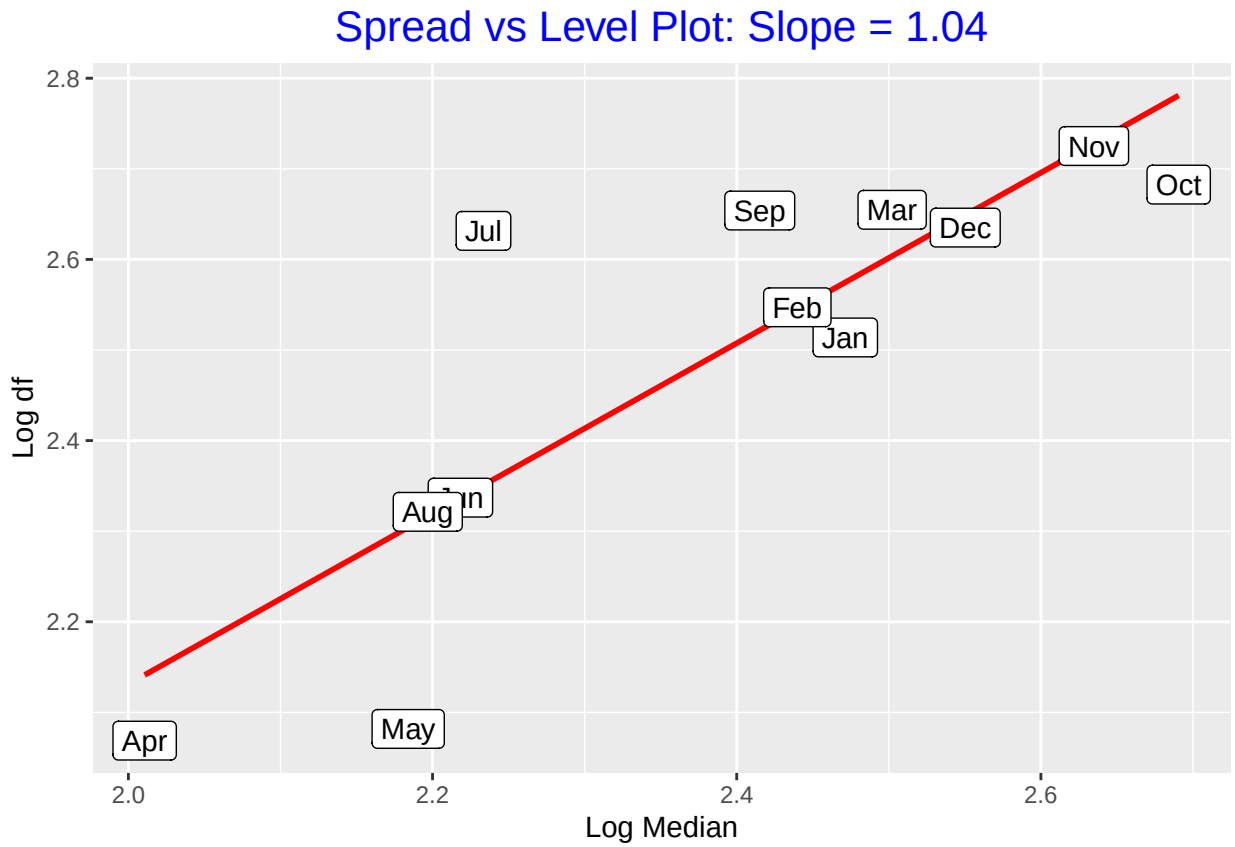



- There is relationship between the spread and level avg unit price of items purchased by customers.

```
spread_level_plot(sampled_data,  avg_unit_price, end_month)
```

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

## Warning: The following aesthetics were dropped during statistical transformation: label.
## 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?
```

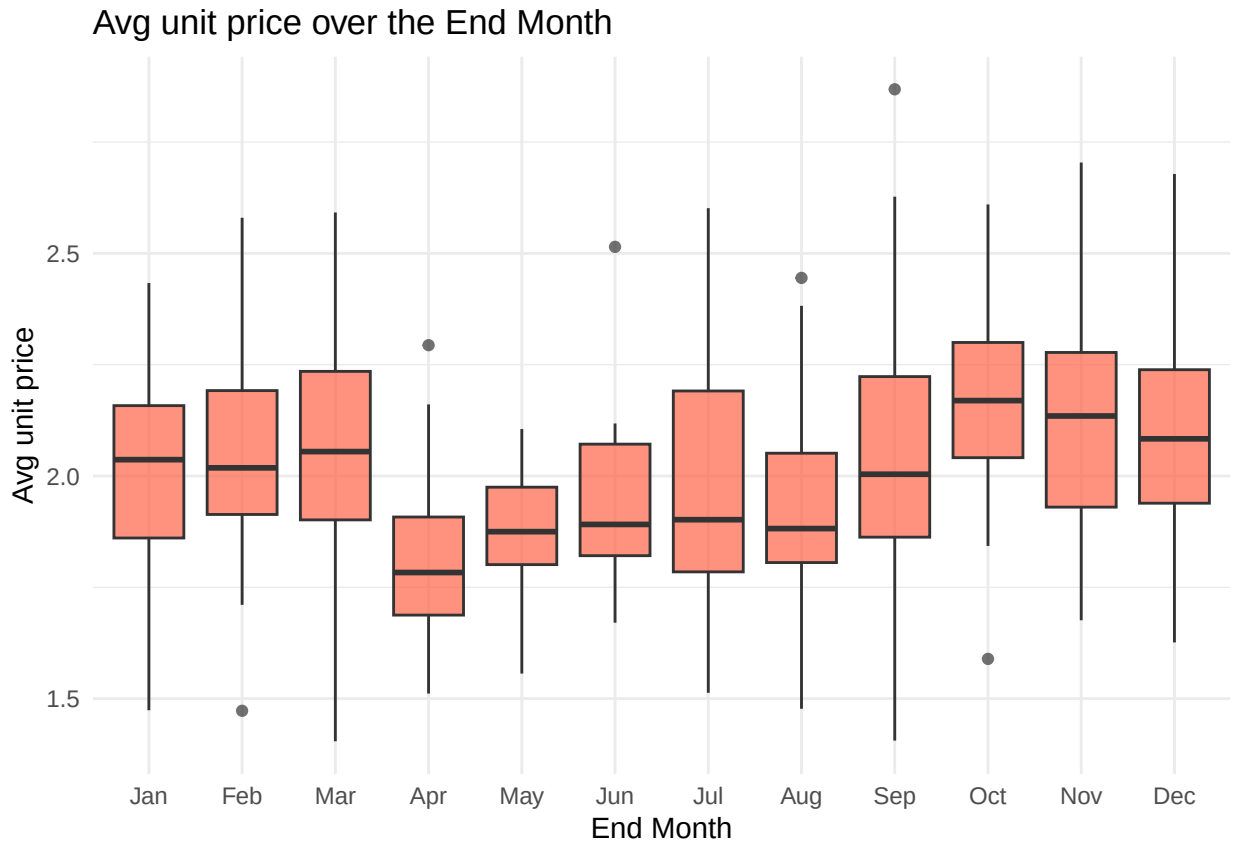



```
## # A tibble: 12 x 5
##    end_month     M    df log.M log.df
##    <ord>     <dbl> <dbl> <dbl>  <dbl>
##  1 Jan        296.  326.  2.47   2.51
##  2 Feb        275.  352.  2.44   2.55
##  3 Mar        318.  452.  2.50   2.65
##  4 Apr        102.  117.  2.01   2.07
##  5 May        153.  121.  2.18   2.08
##  6 Jun        165.  217.  2.22   2.34
##  7 Jul        171.  428.  2.23   2.63
##  8 Aug        157.  210.  2.20   2.32
##  9 Sep        260.  451.  2.42   2.65
## 10 Oct        490.  481.  2.69   2.68
## 11 Nov        431.  531.  2.63   2.73
## 12 Dec        355.  432.  2.55   2.64
```

_ Reexpression of the raw datasets for avg unit price into eighth-root and we could clearly analyse the spread of the data over the months.

```
ggplot(sampled_data, aes(x = factor(end_month), y = avg_unit_price^(1/8))) +
  geom_boxplot(fill = "tomato", alpha = 0.7) +
  labs(
    title = "Avg unit price over the End Month",
    x = "End Month",
    y = "Avg unit price"
```

```
  ) +
  theme_minimal()
```

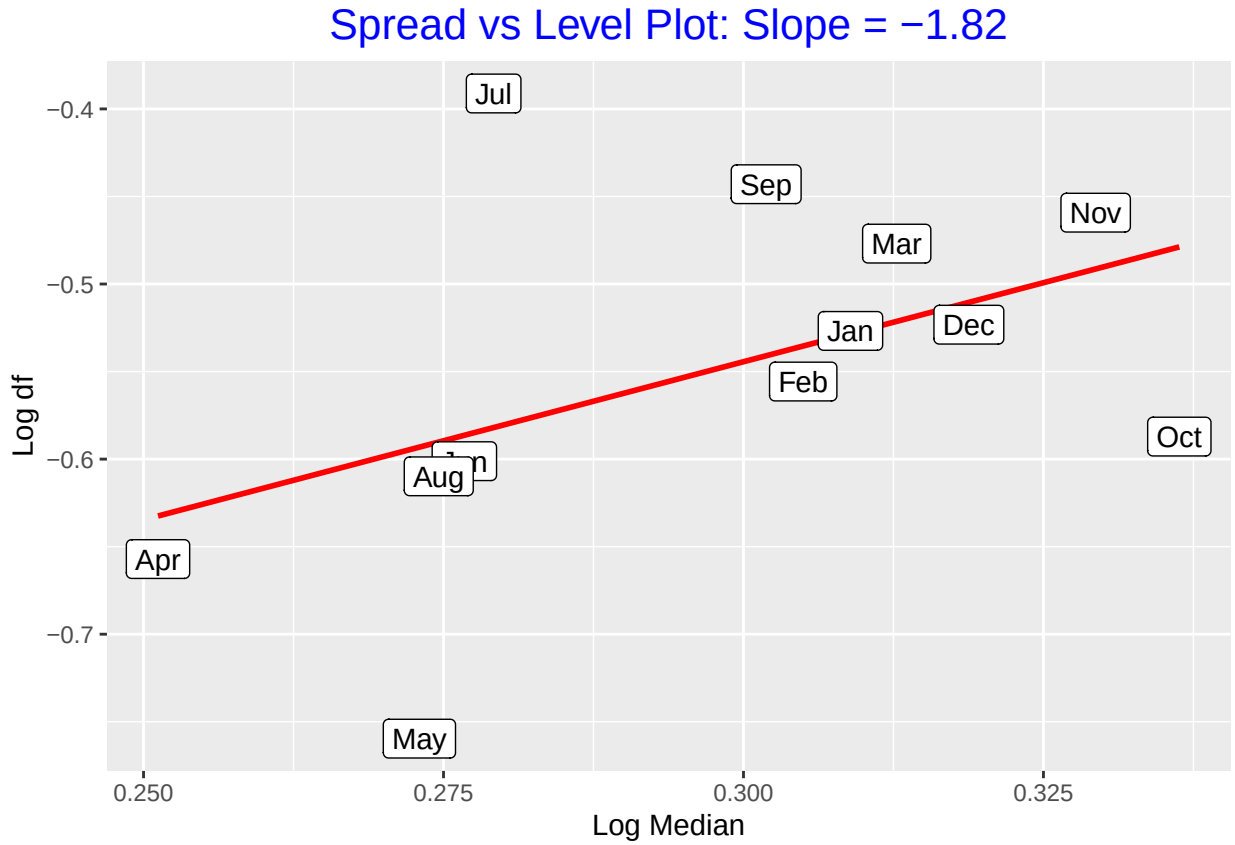



- Check for spread and level plot of the avg unit price of items purchased by customers. We could see there is an independence between spread and level of eighth-root of the avg unit price of the items.

```
spread_level_plot(sampled_data,  avg_unit_price^(1/8), end_month)
```

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

```
## Warning: The following aesthetics were dropped during statistical transformation: label.
## 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?
```




```
## # A tibble: 12 x 5
##    end_month     M    df log.M log.df
##    <ord>     <dbl> <dbl> <dbl>  <dbl>
##  1 Jan        2.04 0.297 0.309 -0.527
##  2 Feb        2.02 0.278 0.305 -0.556
##  3 Mar        2.05 0.333 0.313 -0.477
##  4 Apr        1.78 0.220 0.251 -0.657
##  5 May        1.87 0.174 0.273 -0.760
##  6 Jun        1.89 0.251 0.277 -0.601
##  7 Jul        1.90 0.406 0.279 -0.391
##  8 Aug        1.88 0.246 0.275 -0.610
##  9 Sep        2.00 0.361 0.302 -0.443
## 10 Oct        2.17 0.259 0.336 -0.587
## 11 Nov        2.13 0.347 0.329 -0.459
## 12 Dec        2.08 0.300 0.319 -0.523
```

From our conclusion, we observed that it is expediates to reexpressed the dataset from the raw form into another representation for the purpose of symmetric measures and also avoid the problem of skewness in the data.

**Fitting and plotting** In this next phase of our analysis, we will consider the total spending of customers to unit price of items. We want to investigate if the total spending of the customers are affected by the unit price of items. Since we have alread investigated the distribution these features we will used the reexpressed features of these information to do our fitting.

We will perform comparative analysis of different fitting methodologies as explained in the class.

- linear fitting:

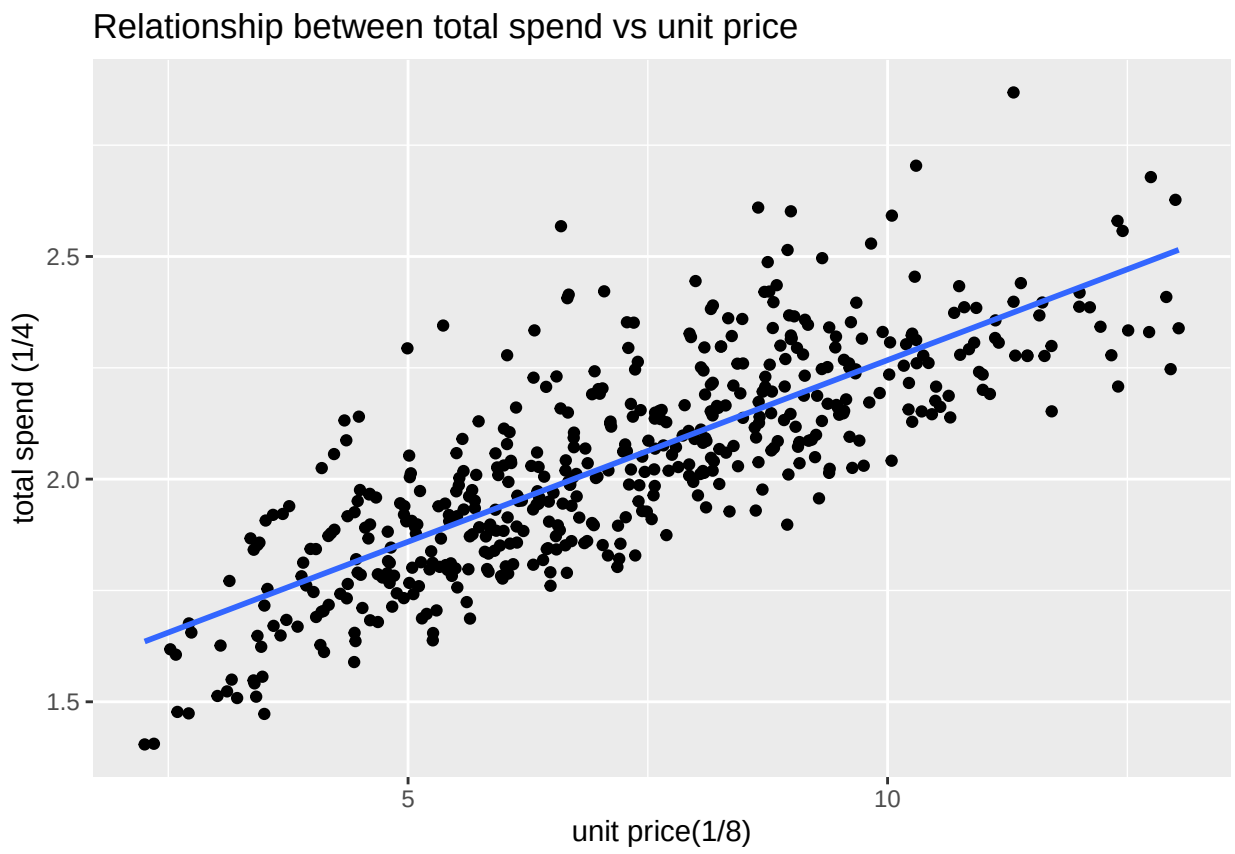
We will perform linear fitting of the data.

```
# summary of the fitting
forthroot = sampled_data$total_spend^(1/4)
eighthroot = sampled_data$avg_unit_price^(1/8)
lm(forthroot ~ eighthroot)
```

```
##
## Call:
## lm(formula = forthroot ~ eighthroot)
##
## Coefficients:
## (Intercept)   eighthroot
##      -8.716        7.806
```

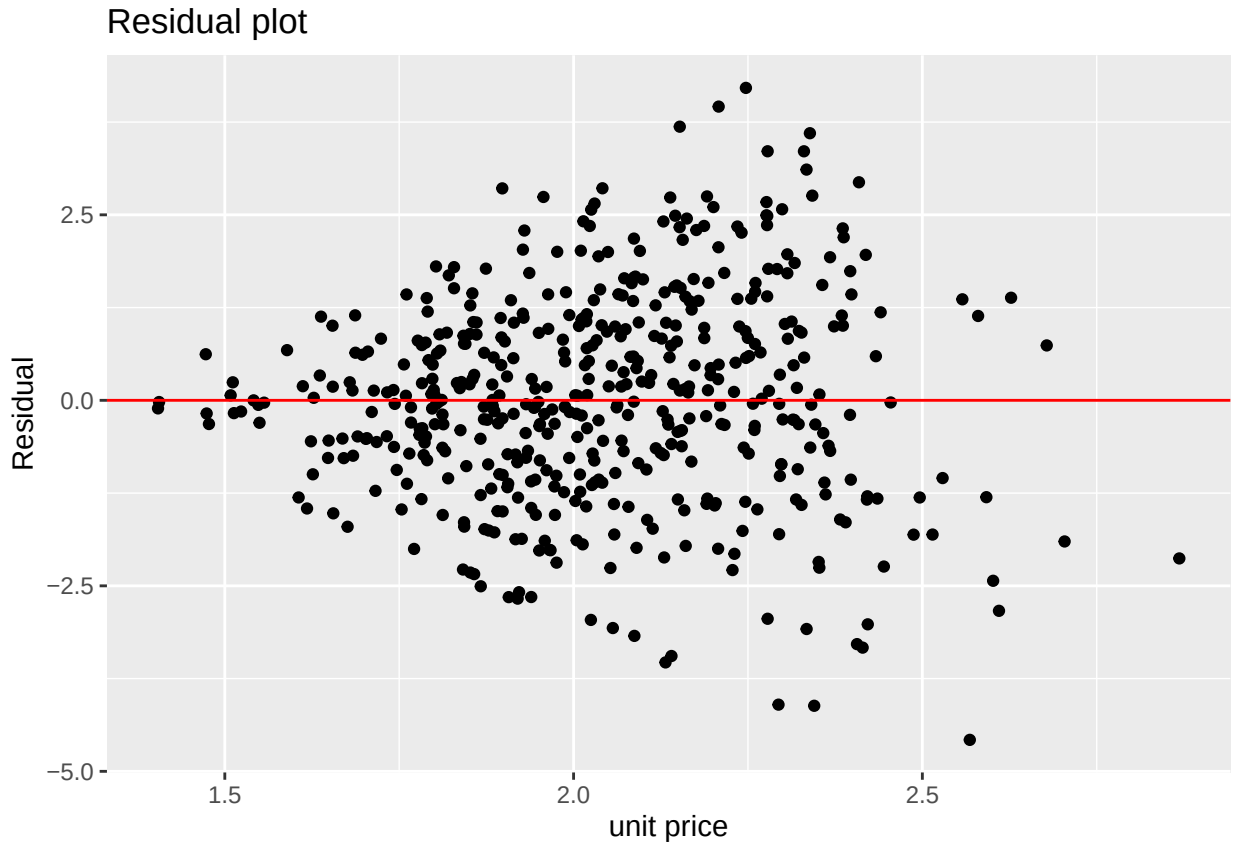
```
ggplot(sampled_data,
       aes(total_spend^(1/4),avg_unit_price^(1/8))) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  xlab("unit price(1/8)") + ylab("total spend (1/4)") +
  ggtitle("Relationship between total spend vs unit price")
```

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




- Residual plot of the fitted model. The residuals are non-constant based on the graph

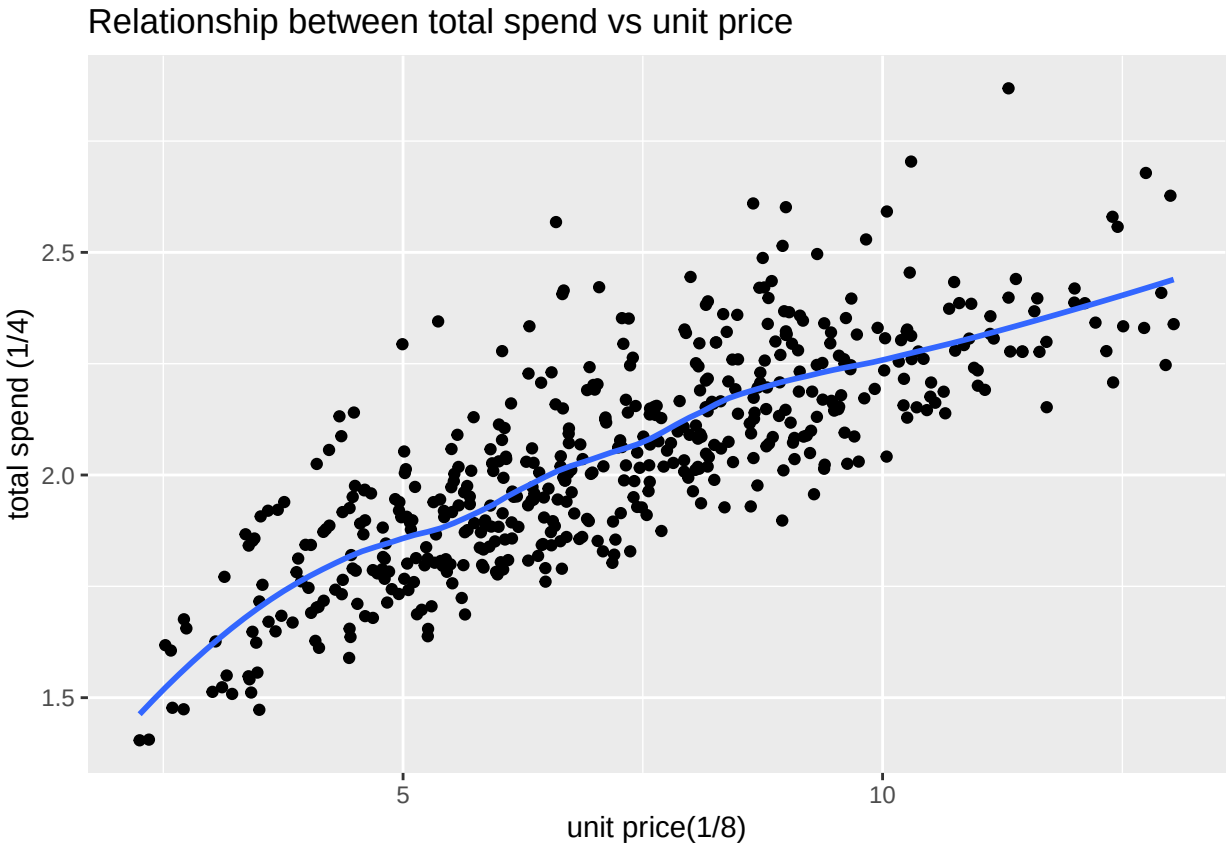
```
sampled_data %>%
  mutate(FIT = 7.569 *
    (avg_unit_price^(1/8)) + -8.267,
    Residual =  total_spend^(1/4) - FIT) %>%
  ggplot(aes(avg_unit_price^(1/8), Residual)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  xlab("unit price") +
  ggtitle("Residual plot")
```




- Nonlinear fit was made

```
ggplot(sampled_data,
       aes(total_spend^(1/4),avg_unit_price^(1/8))) +
  geom_point() +
  geom_smooth(method = "loess",span =0.5, se = FALSE) +
  xlab("unit price(1/8)") + ylab("total spend (1/4)") +
  ggtitle("Relationship between total spend vs unit price")
```

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



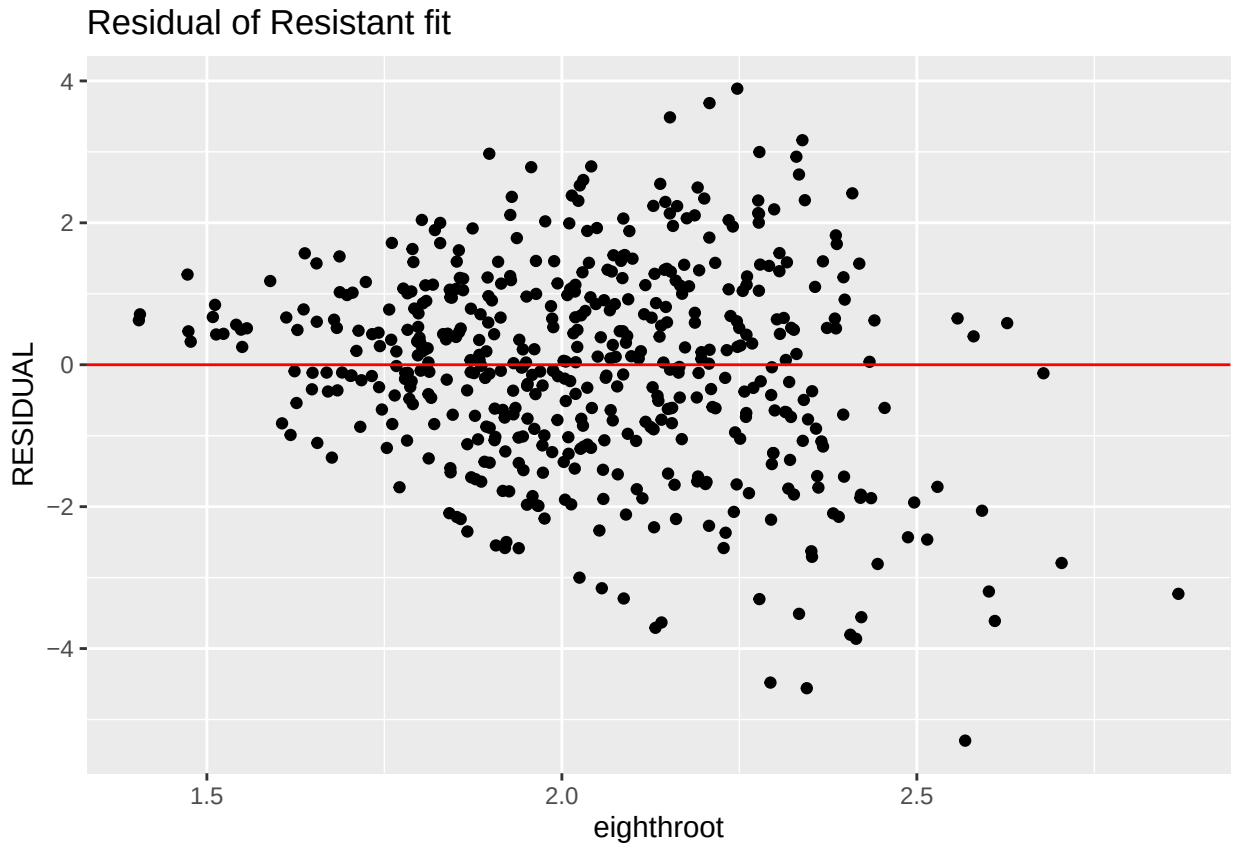

- The resistant fit shows a slight representation compared to the least-squared fit.

```
sampled_data <- sampled_data %>%
  mutate(
    forthroot = total_spend^(1/4),
    eighthroot = avg_unit_price^(1/8)
  )
myfit <- rline(forthroot ~ eighthroot,data.frame(sampled_data))
sampled_data<-
  mutate(sampled_data,
         FIT = myfit$a + myfit$b * (eighthroot - myfit$xC),
         RESIDUAL = forthroot - FIT)
select(sampled_data, FIT, RESIDUAL)
```

```
## # A tibble: 500 x 2
##      FIT RESIDUAL
##    <dbl>    <dbl>
##  1  3.46   0.666
##  2  9.06   0.613
##  3  6.66  -0.996
##  4  9.35  -0.234
##  5  4.83  -0.0181
##  6  7.65  -3.29
##  7  6.93  -0.511
##  8  8.38   1.00
```

```
##  9  5.65   1.05
## 10  9.49  -0.0355
## # i 490 more rows
```

```
ggplot(sampled_data, aes(eighthroot, RESIDUAL)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red") +
  ggtitle("Residual of Resistant fit")
```




```
myfit
```

```
## $a
## [1] 7.141076
##
## $b
## [1] 8.821492
##
## $xC
## [1] 2.0294
##
## $half.slope.ratio
## [1] 0.7810958
##
## $residual
##   [1]  0.66579979  0.61276996 -0.99580391 -0.23365006 -0.01812616 -3.29440830
```

```
##   [7] -0.51128033  1.00008107  1.04973362 -0.03551316  0.27071399  2.06393740
##  [13]  0.52027771 -1.12034406  1.49493471  0.90951012  0.71179724 -2.18396082
##  [19]  1.31225285  1.02233991 -0.18590449 -0.73614654  0.65021860  2.31012410
##  [25]  1.88517190 -1.24495588  0.29917429  0.42312094 -1.32067870  0.69634110
##  [31] -1.65223161  0.98145439 -3.00083370 -1.78430073 -2.27021380 -1.48046866
##  [37] -1.05182463 -3.63222743  0.53307302 -1.17187417  0.43022869 -1.01094537
##  [43] -1.02141220 -2.29097497 -2.36782005 -0.60881127  0.03229991 -0.46261062
##  [49] -0.82389292 -1.87542794 -3.86227335 -1.38500011 -0.90086487  0.67263236
##  [55] -4.47976857 -0.85945081 -0.87223094  0.18876434  0.92231376 -1.68658507
##  [61] -1.90184250 -1.61304848  2.52734202 -0.10970168  0.96700218 -1.81036454
##  [67]  2.29453991  2.60366181  1.44311321  0.47303090  1.28126354  0.51365838
##  [73]  0.13237757  0.40574590  1.03300699 -1.56945088 -1.75415340  1.45063119
##  [79] -0.36159481  0.43577912  0.03097062  0.43150602  2.68083341  0.05899266
##  [85]  0.85430615  0.35141930 -0.82592654 -0.32430532  2.06032635 -0.50826517
##  [91]  0.04481121  2.41551447  0.98204810  1.08067997  0.79154407 -0.05895346
##  [97]  1.39285415  0.24416776 -1.97281845 -0.47980764 -0.63125365  3.48644406
## [103]  0.82589154  1.31802251 -0.43835085 -2.16774905 -1.82940991 -0.95330475
## [109] -0.87675471  1.71447058  0.27905032  0.03127919  0.39631180 -0.53964814
## [115] -0.06976177 -0.18393063 -2.05737212 -0.26993055 -1.88286702 -0.19983248
## [121]  1.00650998 -2.17394880  1.61288415 -1.15299877  0.63929859 -0.18513434
## [127]  0.05490473  1.22984653  1.92665884  1.24437118 -0.08484354 -1.07231082
## [133]  1.01565312  0.59775590 -2.58273970 -0.98961495 -0.68040736 -0.15911993
## [139] -1.64786625 -2.49872905  0.63437728 -0.08505153  1.12639422 -0.77537673
## [145]  0.49430924 -0.24347527 -2.14669212  0.66856156  0.13947172 -0.08370190
## [151] -1.85160412 -2.09497152  2.49755301  1.46299901  0.58634711 -1.48622799
## [157] -0.11083022  1.33076406 -2.33646732  0.47234484  0.34847595 -1.06342643
## [163]  3.89059139  1.99257680 -3.14999715 -4.55901278  0.51813843 -2.07363548
## [169] -0.23038867  1.99896734 -3.22899690 -0.90416448  1.04275607  0.43735509
## [175] -1.57774631  0.17751020  1.22398460  0.96063547 -0.60839557  0.47929496
## [181] -0.30579156  1.33790227  0.48970857  2.23563005 -2.14357480  3.16463169
## [187] -0.91056932  2.10518324  1.12607452 -1.10224329 -1.37020703 -0.78513908
## [193]  0.31006101 -0.70495865  1.54584478  1.44703626  0.25080755 -0.09981552
## [199]  0.21855135 -0.73482736 -0.10463308  1.17991408 -1.38063857  1.82311448
## [205] -0.37841332 -0.14288084  1.30282142 -0.34633323 -2.43150974  0.39996404
## [211]  1.11588629  1.23274567 -2.17427557 -0.49622240 -0.83658824  1.06398257
## [217] -0.61460959 -0.71985750  1.88376162 -0.21036881  0.49226788 -3.55763774
## [223] -0.09152376  1.70144626  1.07351957 -1.68121640 -1.53396341 -0.61837982
## [229] -1.18469584  1.06244760 -0.69970681 -0.37734148 -0.32634801  0.28045469
## [235]  0.25093784  0.95084532 -1.89188930  0.54998828  1.14623839  0.43133401
## [241]  2.18965185 -0.67265440  0.38705843 -2.80794433 -1.54488778  0.77949405
## [247]  3.68737620 -1.77838690  1.57036754  2.99755427 -1.04829572 -0.74503367
## [253]  2.31450088 -1.98371866  0.51090766  1.04021884  0.84400319  1.21262169
## [259]  1.24957032 -0.08169801 -1.02680020  0.94482120 -0.41406968 -1.74600747
## [265]  0.08271865 -1.52150292 -1.07392315 -1.73134162 -0.46649840  2.13232541
## [271] -2.58359581 -1.46349735 -0.77032895  0.11480166  1.22078676  0.71046540
## [277] -5.29785484  0.15150524  0.56341284  0.01468421  1.52685736 -1.98948556
## [283]  0.21567901 -0.36129504  0.02186928  2.31914926  0.86815017  0.51870509
## [289] -1.06505157  0.52917789  0.62699942  0.90610292 -0.04072579 -2.11167191
## [295] -2.46401785  0.86491910  1.45358149 -2.35014243  1.09728837  0.77774289
## [301] -1.69158893 -3.51078101  1.35243359  0.91771945  2.00281800 -2.54669423
## [307] -3.70771441  0.76758051 -2.62858848  0.38097382 -1.13716396  1.78493801
## [313] -0.46147053 -0.63975387  0.31895787  0.49276916  0.18740782  1.12198202
## [319]  1.71641616  0.35204684  0.65237904 -0.60745941  0.51201089 -0.23245255
## [325] -1.22191391 -0.63666665  0.23187114 -0.77948756  0.59457378 -0.70304646
```

```
## [331]  0.11208538 -0.61360162  1.43615437 -0.80429209  1.12859052  0.66504645
## [337]  1.40834747  2.12647717  0.21350349 -0.59536045  0.99826277  1.43558348
## [343] -3.61102717  0.60570183 -3.19594157  0.59269768 -1.17269779  0.85504853
## [349] -0.43319841  1.45954117  2.02039906  0.42674573 -0.31617861  1.16607422
## [355]  0.51766779  1.95631204  0.19013753  0.03555625  2.11096063  0.89908904
## [361]  0.20519060 -0.55502258  0.73177207  0.66310431  1.05736473 -0.15372631
## [367]  2.38441090  0.65921338  0.45860992 -1.08086510  0.62432807 -3.80511952
## [373]  2.97369716 -1.34149644 -1.64466918  0.06704136  1.42788124  0.72358134
## [379] -0.13727557 -1.12454775  1.57335509 -1.51648532  0.71135815  0.11990102
## [385]  1.41153586 -0.37326943 -0.64410556 -1.30874437 -1.57487824 -0.03346049
## [391] -0.60942167 -0.83590476  1.63089483  1.06987081  2.23850781  1.46411343
## [397] -1.23148821 -1.24551892 -0.21874481 -0.11141552 -0.62323128  1.18561834
## [403]  0.08908628 -0.88855333 -0.11594410  1.45712108 -1.58665611 -1.15388012
## [409]  1.42483588  0.49135907  0.81437573  0.45164506  0.25148789 -2.58537831
## [415]  1.31377207 -0.19607805  0.18567526  0.26232168  0.21056918 -0.11316076
## [421] -0.34311696 -1.72695330  2.54929237  1.27086286 -0.66461643 -3.30306253
## [427] -0.11774002 -2.70736336  0.32485167 -1.94150313  1.14309998 -1.36730323
## [433]  2.04058195  1.92115827  0.44271790  2.03753641 -0.11811228 -1.40058111
## [439] -0.03110889 -0.36562201 -1.88010891  1.79177584  0.08937112 -2.79421782
## [445] -0.41442570 -0.76199842  0.32856158 -0.01648582 -1.01681273  1.11937490
## [451] -0.12030928  1.89761599 -1.06869849 -1.72101595  0.19465879 -0.15014075
## [457]  0.35771977  2.36612417 -0.16141995 -1.45722153  1.10596282  2.79514562
## [463]  1.54303685 -0.29277671  1.33854330  1.94752756 -0.11468031 -0.29649045
## [469]  2.34247568 -1.83131672 -0.31312441  2.78425343  0.75788380  1.03133487
## [475]  0.39185051  0.04111056  0.47093120  2.93170886  0.65065321  1.53449823
## [481] -0.12598931  0.68686360 -0.15734976  0.06511998 -1.25480501  2.13669219
## [487]  0.79013256 -0.09520813 -2.09236801 -0.87493840 -1.04285176 -0.75961304
## [493] -0.41050408 -1.96823779 -0.31705187 -0.97355638  1.19214158 -0.42699849
## [499]  0.95055112  0.43471459
##
## $spoints.x
## [1] 1.802797 2.029400 2.280054
##
## $spoints.y
## [1] 5.055547 7.314190 9.265662
```

**Smoothing Method**

- Visualization of the smooth