Olist Data Analysis Project

Ahmed Nasser

2024-03-20

Introduction:

As a senior data analyst working at a fast-growing e-commerce platform, Olist, based in Brazil. -The company operates a marketplace connecting sellers and buyers across various product categories-. My task is to analyze the available dataset to derive actionable insights to improve business operations and customer experience, by first answering the business objectives.
In this analysis case study I'll use the six steps of the data analysis process, which is (Ask, Prepare, Process, Analyze, Share, and Act).

Business Objectives:

1. Exploring the data.
2. Find the customer segmentation.
3. Create Churn Analysis.
4. Design a Predictive Model.
5. Visualize the findings and reporting it.

Ask:

In this step we will ask the important questions -*and answering them in the upcoming steps*- to meet the business objectives.

- **Exploring the data:**
  1. What is the distribution of orders over time, and trends in order volume and order statuses?
  2. What the most popular product categories and sellers on the platform?
  3. What is the customer demographics, such as location and purchasing behavior?
- **Find the customer segmentation:**
  1. Create segment customers based on their purchasing behavior, geographic location, or other relevant factors.
  2. Explore different segmentation techniques tailored to the characteristics of the Brazilian e-commerce market.
- **Create Churn Analysis:**
  1. Analyze customer churn rate over time, identifying factors influencing churn and proposing strategies to retain customers.
  2. Examine the impact of product quality, delivery speed, and customer service on customer satisfaction and retention.
- **Design a Predictive Model:**

1. Build predictive models to forecast sales, customer demand, or product popularity.
2. Evaluate the performance of different forecasting algorithms and assess their suitability for the Brazilian e-commerce market.

- **Visualize the findings and reporting it:**

1. Create visualizations and dashboards to present key insights and trends.
2. Prepare a comprehensive report summarizing my analysis and recommendations.
3. Present my findings to stakeholders, highlighting actionable recommendations to drive business growth and improve customer satisfaction.

Prepare:

- Now we will download the Brazilian E-commerce Dataset from Olist from https://drive.google.com/file/d/1pHkWkE4lveePWOU9Ornn8uNeI7RjQ1BF/view and start exploring.

Process:

- In this case study I'll be using **R** to analyze the data.

1. Unzip the csv files in the same folder.
2. Open RStudio and start new session.
3. Create .RMD file and start our script.
4. Read csv files into multiple dataframes.

```
library(tidyverse)
orders <- read_csv('olist_orders_dataset.csv')
customers <- read_csv('olist_customers_dataset.csv')
products <- read_csv('olist_products_dataset.csv')
sellers <- read_csv('olist_sellers_dataset.csv')
payments <- read_csv('olist_order_payments_dataset.csv')
reviews <- read_csv('olist_order_reviews_dataset.csv')
geolocation <- read_csv('olist_geolocation_dataset.csv')
items <- read_csv('olist_order_items_dataset.csv')
category <- read_csv('product_category_name_translation.csv')
```

5. Use the **skimr** library to summarize the data and search for any inconsistency or null values in the important datasets (orders, customers, products, sellers, payments, and reviews).

```
# Load necessary libraries
library(dplyr)
library(lubridate)
library(cluster)
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 4.3.3
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(survival)
library(skimr)
```

```r
#check the datasets
skim(orders)
```

Table 1: Data summary

Name	orders
Number of rows	99441
Number of columns	8
Column type frequency:	
character	3
POSIXct	5
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
order_id	0	1	32	32	0	99441	0
customer_id	0	1	32	32	0	99441	0
order_status	0	1	7	11	0	8	0

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
order_purchase_timestamp	0	1.00	2016-09-04 21:15:19	2018-10-17 17:30:18	2018-01-18 23:04:36	98875
order_approved_at	160	1.00	2016-09-15 12:16:38	2018-09-03 17:40:06	2018-01-19 11:36:13	90733
order_delivered_carrier_date	1783	0.98	2016-10-08 10:34:01	2018-09-11 19:48:28	2018-01-24 16:10:58	81018
order_delivered_customer_date	2965	0.97	2016-10-11 13:46:32	2018-10-17 13:22:46	2018-02-02 19:28:10	95664
order_estimated_delivery_date	0	1.00	2016-09-30 00:00:00	2018-11-12 00:00:00	2018-02-15 00:00:00	459

```r
skim(customers)
```

Table 4: Data summary

Name	customers
Number of rows	99441
Number of columns	5

Column type frequency:	
character	5
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
customer_id	0	1	32	32	0	99441	0
customer_unique_id	0	1	32	32	0	96096	0
customer_zip_code_prefix	0	1	5	5	0	14994	0
customer_city	0	1	3	32	0	4119	0
customer_state	0	1	2	2	0	27	0

```r
skim(products)
```

Table 6: Data summary

Name	products
Number of rows	32951
Number of columns	9
Column type frequency:	
character	2
numeric	7
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
product_id	0	1.00	32	32	0	32951	0
product_category_name	610	0.98	3	46	0	73	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
product_name_lenght	610	0.98	48.48	10.25	5	42	51	57	76	
product_description_lenght	610	0.98	771.50	635.12	4	339	595	972	3992	
product_photos_qty	610	0.98	2.19	1.74	1	1	1	3	20	
product_weight_g	2	1.00	2276.47	4282.04	0	300	700	1900	40425	
product_length_cm	2	1.00	30.82	16.91	7	18	25	38	105	
product_height_cm	2	1.00	16.94	13.64	2	8	13	21	105	
product_width_cm	2	1.00	23.20	12.08	6	15	20	30	118	

```r
skim(sellers)
```

Table 9: Data summary

Name	sellers
Number of rows	3095
Number of columns	4
Column type frequency:	
character	4
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
seller_id	0	1	32	32	0	3095	0
seller_zip_code_prefix	0	1	5	5	0	2246	0
seller_city	0	1	2	40	0	611	0
seller_state	0	1	2	2	0	23	0

```r
skim(payments)
```

Table 11: Data summary

Name	payments
Number of rows	103886
Number of columns	5
Column type frequency:	
character	2
numeric	3
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
order_id	0	1	32	32	0	99440	0
payment_type	0	1	6	11	0	5	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
payment_sequential	0	1	1.09	0.71	1	1.00	1	1.00	29.00	
payment_installments	0	1	2.85	2.69	0	1.00	1	4.00	24.00	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
payment_value	0	1	154.10	217.49	0	56.79	100	171.84	13664.08	

```
skim(reviews)
```

Table 14: Data summary

Name	reviews
Number of rows	99224
Number of columns	7
Column type frequency:	
character	4
numeric	1
POSIXct	2
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
review_id	0	1.00	32	32	0	98410	0
order_id	0	1.00	32	32	0	98673	0
review_comment_title	87658	0.12	1	26	0	4178	0
review_comment_message	58256	0.41	1	208	0	35743	18

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
review_score	0	1	4.09	1.35	1	4	5	5	5	

Variable type: POSIXct

skim_variable	n_missing	complete_rate	min	max	median	n_unique
review_creation_date	0	1	2016-10-02 00:00:00	2018-08-31 00:00:00	2018-02-02 00:00:00	636
review_answer_timestamp	0	1	2016-10-07 18:32:28	2018-10-29 12:27:35	2018-02-04 22:41:47	98248

6. Now we will create a dataframe list with the datasets that has the same *Primary_key* or *Foreign_key* (orders, payments, reviews, items) to join the them together using *reduce* function in *dfx* dataframe.

```
df_list <- list(orders,payments,reviews,items)
dfx <- df_list %>% reduce(full_join, by='order_id')
```

7. Join the rest of the datasets (products, customers, sellers) to the *dfx* and save it in *df_all* dataframe.

```r
dfx <- full_join(dfx, products)
```

```
## Joining with `by = join_by(product_id)`
```

```r
dfx <- full_join(dfx, customers)
```

```
## Joining with `by = join_by(customer_id)`
```

```r
df_all <- full_join(dfx, sellers)
```

```
## Joining with `by = join_by(seller_id)`
```

```r
# Create Mode function
Mode <- function(x) {
  unique_values <- unique(x)
  freq_table <- tabulate(match(x, unique_values))
  modes <- unique_values[freq_table == max(freq_table)]
  return(modes)
}
```

8. Now we need to review the collective dataframe's column names using *colnames()* function and the first few rows using *head()* function.

```r
colnames(df_all)
```

```
##  [1] "order_id"                      "customer_id"
##  [3] "order_status"                  "order_purchase_timestamp"
##  [5] "order_approved_at"             "order_delivered_carrier_date"
##  [7] "order_delivered_customer_date" "order_estimated_delivery_date"
##  [9] "payment_sequential"            "payment_type"
## [11] "payment_installments"          "payment_value"
## [13] "review_id"                     "review_score"
## [15] "review_comment_title"          "review_comment_message"
## [17] "review_creation_date"          "review_answer_timestamp"
## [19] "order_item_id"                 "product_id"
## [21] "seller_id"                     "shipping_limit_date"
## [23] "price"                         "freight_value"
## [25] "product_category_name"         "product_name_lenght"
## [27] "product_description_lenght"    "product_photos_qty"
## [29] "product_weight_g"              "product_length_cm"
## [31] "product_height_cm"             "product_width_cm"
## [33] "customer_unique_id"            "customer_zip_code_prefix"
## [35] "customer_city"                 "customer_state"
## [37] "seller_zip_code_prefix"        "seller_city"
## [39] "seller_state"
```

```r
head(df_all)
```

```
## # A tibble: 6 x 39
##   order_id   customer_id order_status order_purchase_times~1 order_approved_at
##   <chr>      <chr>       <chr>        <dttm>                 <dttm>
## 1 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 2 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 3 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 4 53cdb2fc8~ b0830fb474~ delivered    2018-07-24 20:41:37    2018-07-26 03:24:27
## 5 47770eb91~ 41ce2a54c0~ delivered    2018-08-08 08:38:49    2018-08-08 08:55:23
## 6 949d5b44d~ f88197465e~ delivered    2017-11-18 19:28:06    2017-11-18 19:45:59
## # i abbreviated name: 1: order_purchase_timestamp
## # i 34 more variables: order_delivered_carrier_date <dttm>,
## #   order_delivered_customer_date <dttm>, order_estimated_delivery_date <dttm>,
## #   payment_sequential <dbl>, payment_type <chr>, payment_installments <dbl>,
## #   payment_value <dbl>, review_id <chr>, review_score <dbl>,
## #   review_comment_title <chr>, review_comment_message <chr>,
## #   review_creation_date <dttm>, review_answer_timestamp <dttm>, ...
```

9. I noticed that some rows are duplicated due to the payment_type, so we'll remove the columns responsible for the duplication, and remove NA values.

```r
df_all_unique <- df_all %>% group_by(customer_id) %>%
  subset(select = -c(payment_sequential, payment_type, payment_installments, order_item_id, payment_valu
  mutate(n_payment_value = sum(price)+sum(freight_value)) %>%
  unique() %>%
  ungroup()
```

```r
df_all_unique <- left_join(df_all_unique, category)
```

```
## Joining with `by = join_by(product_category_name)`
```

```r
colnames(df_all_unique)
```

```
##  [1] "order_id"                      "customer_id"
##  [3] "order_status"                  "order_purchase_timestamp"
##  [5] "order_approved_at"             "order_delivered_carrier_date"
##  [7] "order_delivered_customer_date" "order_estimated_delivery_date"
##  [9] "review_id"                     "review_score"
## [11] "review_comment_title"          "review_comment_message"
## [13] "review_creation_date"          "review_answer_timestamp"
## [15] "product_id"                    "seller_id"
## [17] "shipping_limit_date"           "price"
## [19] "freight_value"                 "product_category_name"
## [21] "product_name_lenght"           "product_description_lenght"
## [23] "product_photos_qty"            "product_weight_g"
## [25] "product_length_cm"             "product_height_cm"
## [27] "product_width_cm"              "customer_unique_id"
## [29] "customer_zip_code_prefix"      "customer_city"
## [31] "customer_state"                "seller_zip_code_prefix"
## [33] "seller_city"                   "seller_state"
## [35] "n_payment_value"               "product_category_name_english"
```

```r
head(df_all_unique)
```

```
## # A tibble: 6 x 36
##   order_id   customer_id order_status order_purchase_times~1 order_approved_at
##   <chr>      <chr>       <chr>        <dttm>                 <dttm>
## 1 e481f51cb~ 9ef432eb62~ delivered    2017-10-02 10:56:33    2017-10-02 11:07:15
## 2 53cdb2fc8~ b0830fb474~ delivered    2018-07-24 20:41:37    2018-07-26 03:24:27
## 3 47770eb91~ 41ce2a54c0~ delivered    2018-08-08 08:38:49    2018-08-08 08:55:23
## 4 949d5b44d~ f88197465e~ delivered    2017-11-18 19:28:06    2017-11-18 19:45:59
## 5 ad21c59c0~ 8ab97904e6~ delivered    2018-02-13 21:18:39    2018-02-13 22:20:29
## 6 a4591c265~ 503740e9ca~ delivered    2017-07-09 21:57:05    2017-07-09 22:10:13
## # i abbreviated name: 1: order_purchase_timestamp
## # i 31 more variables: order_delivered_carrier_date <dttm>,
## #   order_delivered_customer_date <dttm>, order_estimated_delivery_date <dttm>,
## #   review_id <chr>, review_score <dbl>, review_comment_title <chr>,
## #   review_comment_message <chr>, review_creation_date <dttm>,
## #   review_answer_timestamp <dttm>, product_id <chr>, seller_id <chr>,
## #   shipping_limit_date <dttm>, price <dbl>, freight_value <dbl>, ...
```

- Check the last few months to see if there are any missing data:

```r
df_all_unique %>% count(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(date))
```

```
## # A tibble: 25 x 2
##    date    count_of_orders
##    <chr>             <int>
##  1 2018-10               4
##  2 2018-09              16
##  3 2018-08            6790
##  4 2018-07            6533
##  5 2018-06            6417
##  6 2018-05            7147
##  7 2018-04            7247
##  8 2018-03            7483
##  9 2018-02            6999
## 10 2018-01            7613
## # i 15 more rows
```

- Apparently there's an issue in the last two months, so we will drop them:

```r
df_all_unique <- df_all_unique %>% filter(as.Date(order_purchase_timestamp)<'2018-9-1')
```

After reviewing and cleaning, the data is consistent and descriptive.

Analyze & Visualize:

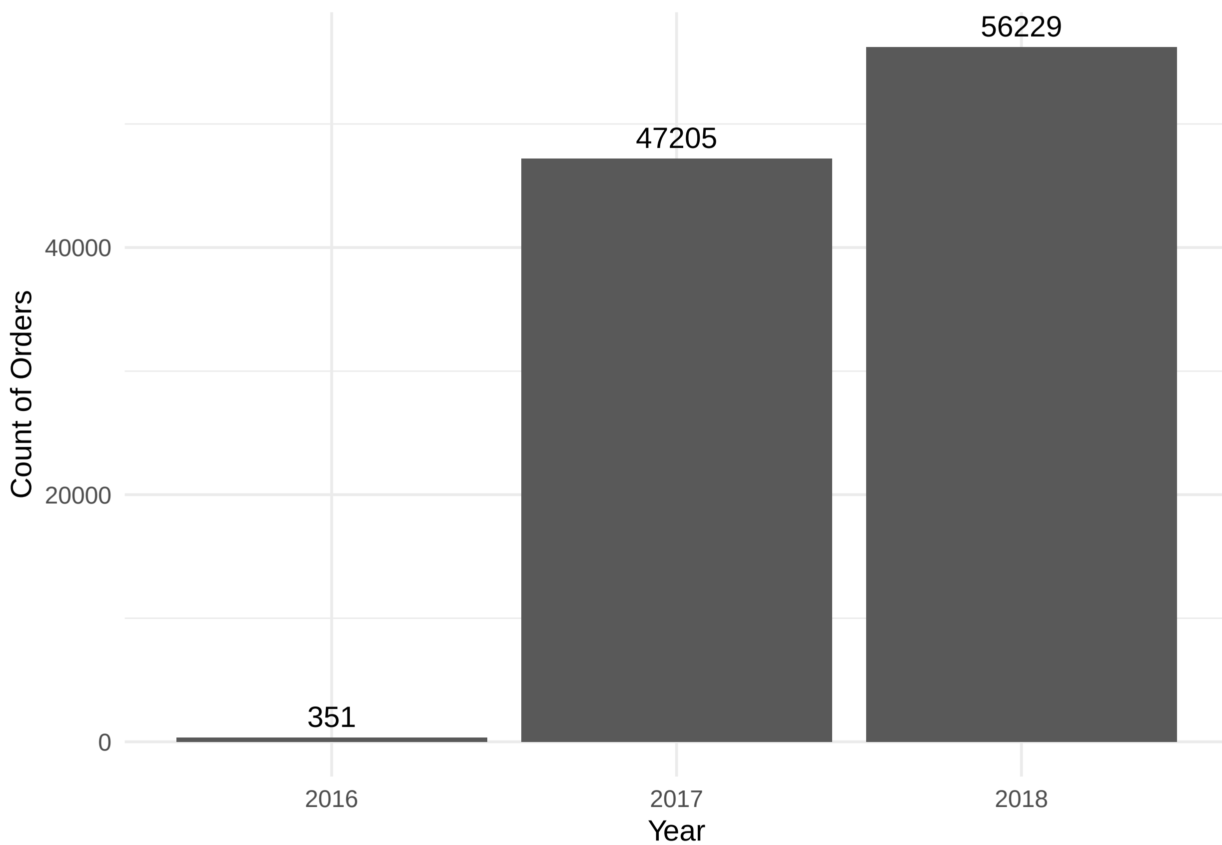
- Now it's time to analyze the data and answer the business task.
    1. What is the distribution of orders over time, and trends in order volume and order statuses?

– Highest year (ranked)

```
df_all_unique %>% count(date=format(order_purchase_timestamp,'%Y')) %>%
  rename(count_of_orders=n) %>%
  mutate(rank=round(rank(-count_of_orders),0)) %>%
  arrange(count_of_orders)
```

```
## # A tibble: 3 x 3
##   date  count_of_orders  rank
##   <chr>           <int> <dbl>
## 1 2016              351     3
## 2 2017            47205     2
## 3 2018            56229     1
```

```
ggplot(df_all_unique %>%
         count(date = format(order_purchase_timestamp, '%Y')) %>%
         rename(count_of_orders = n) %>%
         mutate(rank = round(rank(-count_of_orders), 0)) %>%
         arrange(count_of_orders)) +
  geom_bar(aes(x = date, y = count_of_orders), stat = "identity") +
  labs(x = "Year", y = "Count of Orders") +
  theme_minimal() +
  geom_text(aes(x = date, y = count_of_orders, label = count_of_orders), vjust = -0.5)
```



```r
summary(df_all_unique %>% count(date = format(order_purchase_timestamp, '%Y')) %>% rename(count_of_orde
```

```
##      date           count_of_orders
##  Length:3           Min.   :  351  
##  Class :character   1st Qu.:23778  
##  Mode  :character   Median :47205  
##                     Mean   :34595  
##                     3rd Qu.:51717  
##                     Max.   :56229
```

```r
df_all_unique %>% count(date = format(order_purchase_timestamp, '%Y-%m')) %>%
  summarise(lowest_month = min(n), avg_orders_monthly = mean(n), median = median(n), highest_month = ma
```

```
## # A tibble: 1 x 4
##   lowest_month avg_orders_monthly median highest_month
##          <int>              <dbl>  <int>         <int>
## 1            1              4512.   4550          7934
```

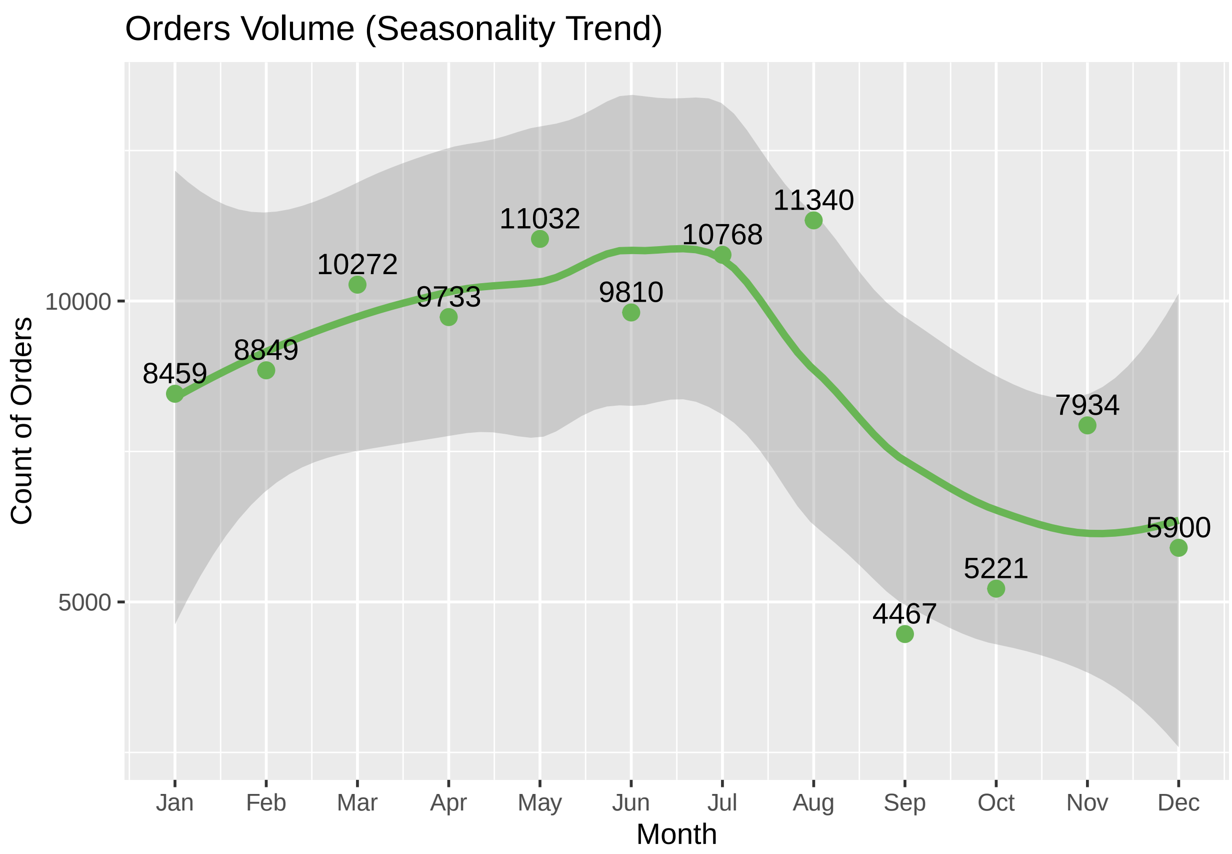
```r
df_all_unique %>% drop_na(n_payment_value) %>%
  group_by(date = format(order_purchase_timestamp, '%Y-%m')) %>%
  summarise(total_value = sum(n_payment_value)) %>%
  summarise(lowest_month = min(total_value), avg_orders_monthly = mean(total_value), median = median(to
```

```
## # A tibble: 1 x 4
##   lowest_month avg_orders_monthly  median highest_month
##          <dbl>              <dbl>   <dbl>         <dbl>
## 1         19.6            777396. 817059.      1348281.
```

- Highest month of year (ranked)

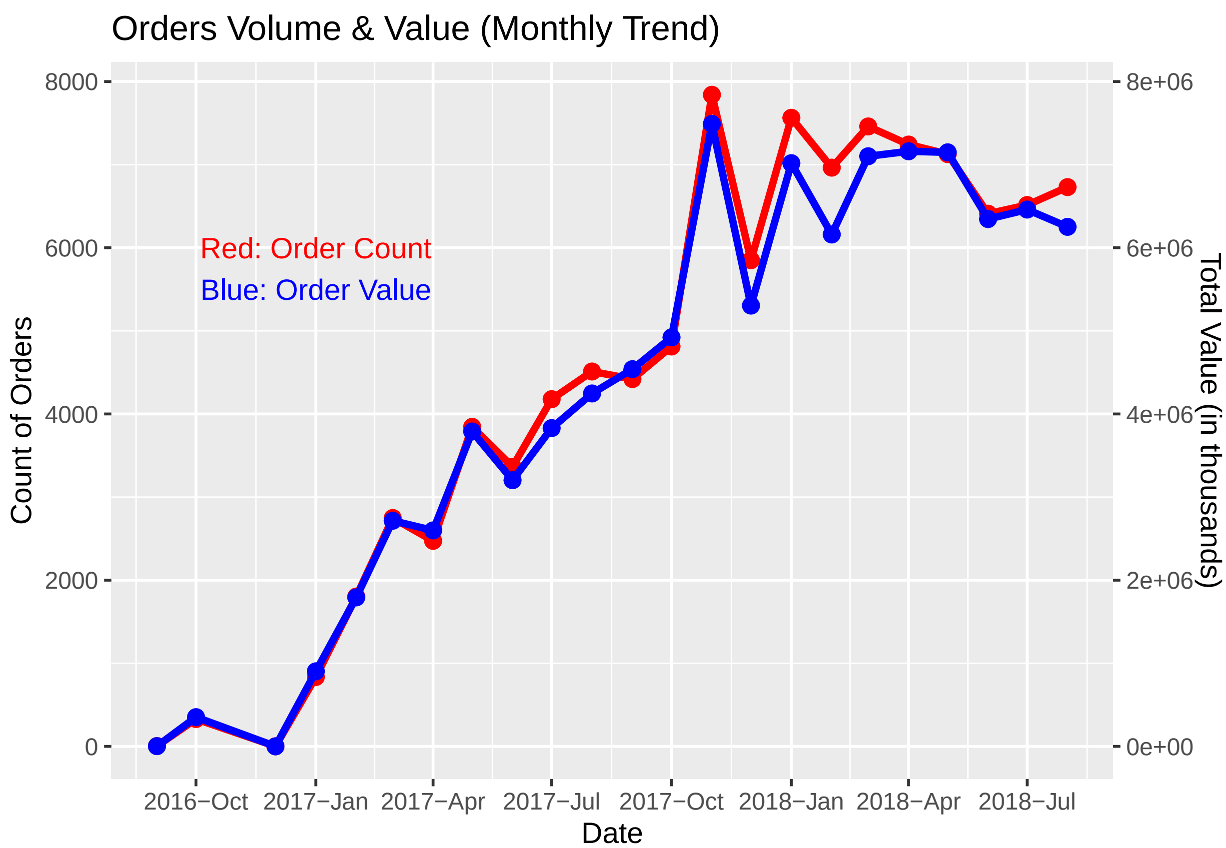
```r
df_all_unique %>%
    count(date = format(order_purchase_timestamp, '%m')) %>%
    rename(count_of_orders = n) %>%
    ggplot(aes(x = as.numeric(date), y = count_of_orders, group = 1)) +
    geom_smooth(linewidth = 1.3, colour = "#69b555") +
    geom_point(size = 2.5, colour = "#69b555") +
    scale_x_continuous(breaks = 1:12, labels = month.abb) +
    labs(title = 'Orders Volume (Seasonality Trend)', x = 'Month', y = 'Count of Orders')+
    geom_text(aes(label = count_of_orders), vjust = -0.5)
```

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



- Orders volume (monthly trend):

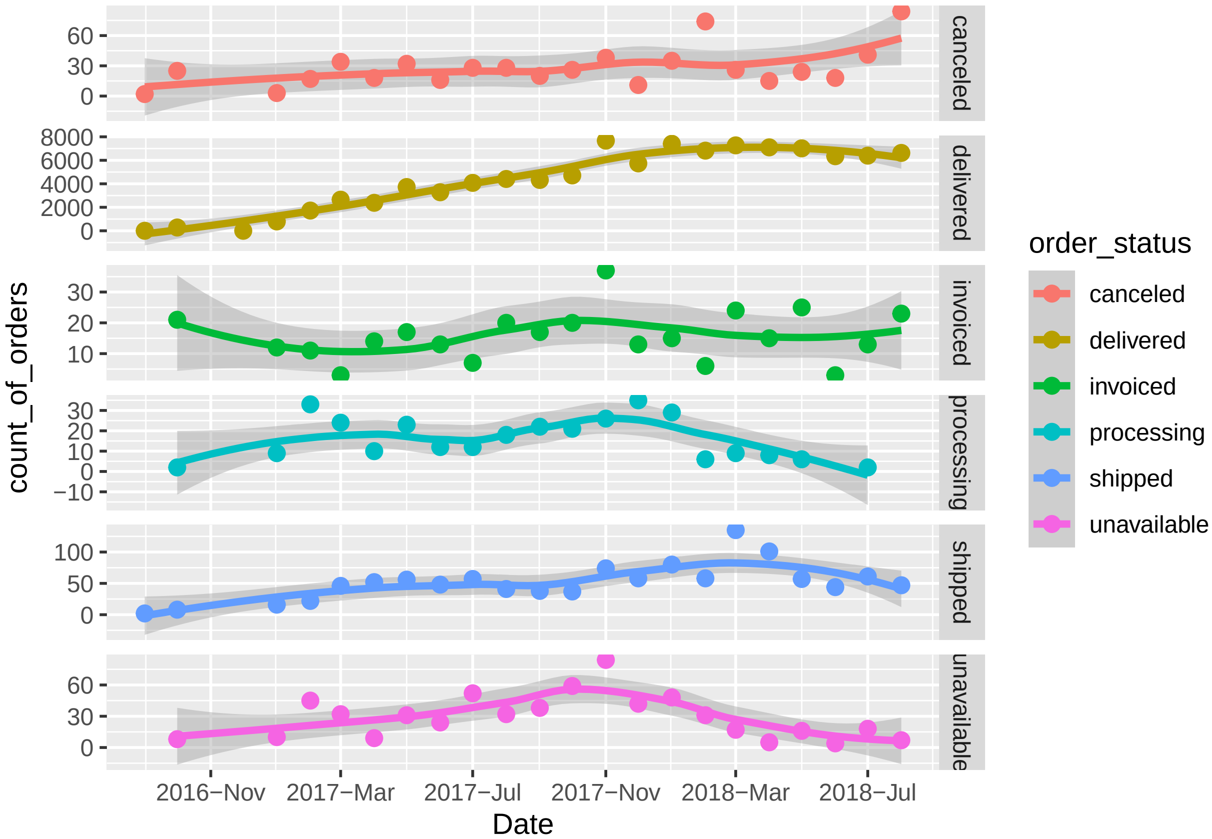
```
df_all_unique %>%
  drop_na(n_payment_value) %>%
  group_by(date = format(order_purchase_timestamp, '%Y-%m')) %>%
  summarise(total_value = sum(n_payment_value), count_of_orders = n()) %>%
  ggplot() +
  geom_line(aes(x = ym(date), y = count_of_orders), linewidth = 1.3, colour = "red") +
  geom_point(aes(x = ym(date), y = count_of_orders), size = 2.5, colour = "red") +
  geom_line(aes(x = ym(date), y = total_value / 180), linewidth = 1.3, colour = "blue") +
  geom_point(aes(x = ym(date), y = total_value / 180), size = 2.5, colour = "blue") +
  scale_x_date(date_breaks = '3 months', date_labels = '%Y-%b') +
  scale_y_continuous(name = "Count of Orders", sec.axis = sec_axis(~ .* 1000, name = "Total Value (in tl
  labs(title = 'Orders Volume & Value (Monthly Trend)', x = 'Date') +
  annotate("text", x = as.Date("2017-01-01"), y = 6000, label = "Red: Order Count", color = "red", size
  annotate("text", x = as.Date("2017-01-01"), y = 5500, label = "Blue: Order Value", color = "blue", si:
```



- number of order status per month:

```
df_all_unique %>% count(order_status,date=format(order_purchase_timestamp,'%Y-%m')) %>%
  rename(count_of_orders=n) %>%
  filter(order_status!="approved",order_status!="created") %>%
  ggplot(aes(ym(date),count_of_orders, colour=order_status, group=order_status)) +
  geom_smooth(linewidth = 1.3) +
  geom_point(size=2.5) +
  scale_x_date(date_breaks = '4 months',date_labels='%Y-%b') +
  labs(x='Date') +
  facet_grid(order_status~., scales = "free")
```

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



2. What the most popular product categories and sellers on the platform?

- Most ordered product categories:

```
df_all_unique %>% full_join(category) %>%
  count(product_category_name_english) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 72 x 2
##    product_category_name_english count_of_orders
##    <chr>                                   <int>
##  1 bed_bath_table                          10295
##  2 health_beauty                            9078
##  3 sports_leisure                           7914
##  4 computers_accessories                    6944
##  5 furniture_decor                          6844
##  6 housewares                               6042
##  7 watches_gifts                            5809
##  8 telephony                                4292
##  9 auto                                     4012
## 10 toys                                     3987
## # i 62 more rows
```

- Highest sellers: (Note: the sellers are shown by the id, as we don't have their names for data privacy & security reasons)

```r
df_all_unique %>% count(seller_id) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 3,096 x 2
##    seller_id                        count_of_orders
##    <chr>                                      <int>
##  1 6560211a19b47992c3666cc44a7e94c0            1988
##  2 4a3ca9315b744ce9f8e9374361493884            1907
##  3 cc419e0650a3c5ba77189a1882b7556a            1763
##  4 1f50f920176fa81dab994f9023523100            1480
##  5 da8622b14eb17ae2831f4ac5b9dab84a            1459
##  6 955fee9216a65b617aa5c0531780ce60            1292
##  7 7a67c85e85bb2ce8582c35f2203ad736            1170
##  8 ea8482cd71df3c1969d7b9473ff13abc            1170
##  9 4869f7a5dfa277a7dca6462dcf3b52b2            1143
## 10 3d871de0142ce09b7081e2b9d1733cb1            1126
## # i 3,086 more rows
```

3. What is the customer demographics, such as location and purchasing behavior?

- Highest cities in purchasing behavior:

```r
df_all_unique %>% count(customer_city) %>%
  rename(count_of_orders=n) %>%
  arrange(desc(count_of_orders))
```

```
## # A tibble: 4,119 x 2
##    customer_city         count_of_orders
##    <chr>                           <int>
##  1 sao paulo                       16320
##  2 rio de janeiro                   7160
##  3 belo horizonte                   2902
##  4 brasilia                         2239
##  5 curitiba                         1573
##  6 campinas                         1509
##  7 porto alegre                     1452
##  8 salvador                         1307
##  9 guarulhos                        1230
## 10 sao bernardo do campo             973
## # i 4,109 more rows
```

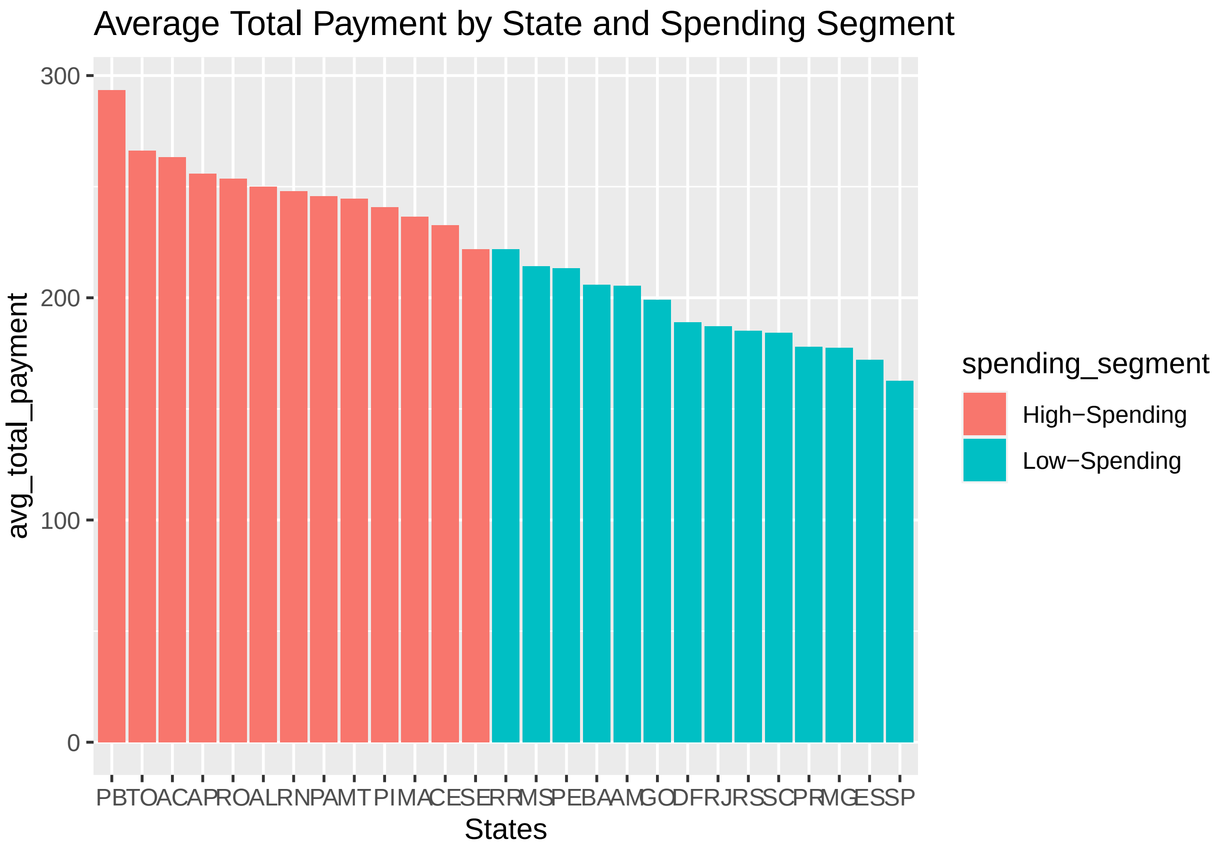
- **Find the customer segmentation:**
    1. Segment customers based on their purchasing behavior, geographic location, or other relevant factors.
    - Create a purchase_behavior df:

```r
df_all_unique %>% drop_na(n_payment_value) %>% group_by(Most_Frequent_State=customer_state, Most_Freque
                                                        Most_Common_Month=as.numeric(format.Date(order_
  reframe(Total_Purchases = n(), avg_payment=round(mean(n_payment_value)), total_payment=sum(n_payment_
  arrange(desc(Total_Purchases))
```

```
## # A tibble: 57,748 x 8
##    Most_Frequent_State Most_Frequent_City Most_Frequent_Category
##    <chr>               <chr>              <chr>
##  1 SP                  sao paulo          health_beauty
##  2 SP                  sao paulo          health_beauty
##  3 SP                  sao paulo          health_beauty
##  4 SP                  sao paulo          bed_bath_table
##  5 SP                  sao paulo          bed_bath_table
##  6 SP                  sao paulo          bed_bath_table
##  7 SP                  sao paulo          bed_bath_table
##  8 SP                  sao paulo          housewares
##  9 SP                  sao paulo          sports_leisure
## 10 SP                  sao paulo          bed_bath_table
## # i 57,738 more rows
## # i 5 more variables: Most_Common_Year <dbl>, Most_Common_Month <dbl>,
## #   Total_Purchases <int>, avg_payment <dbl>, total_payment <dbl>
```

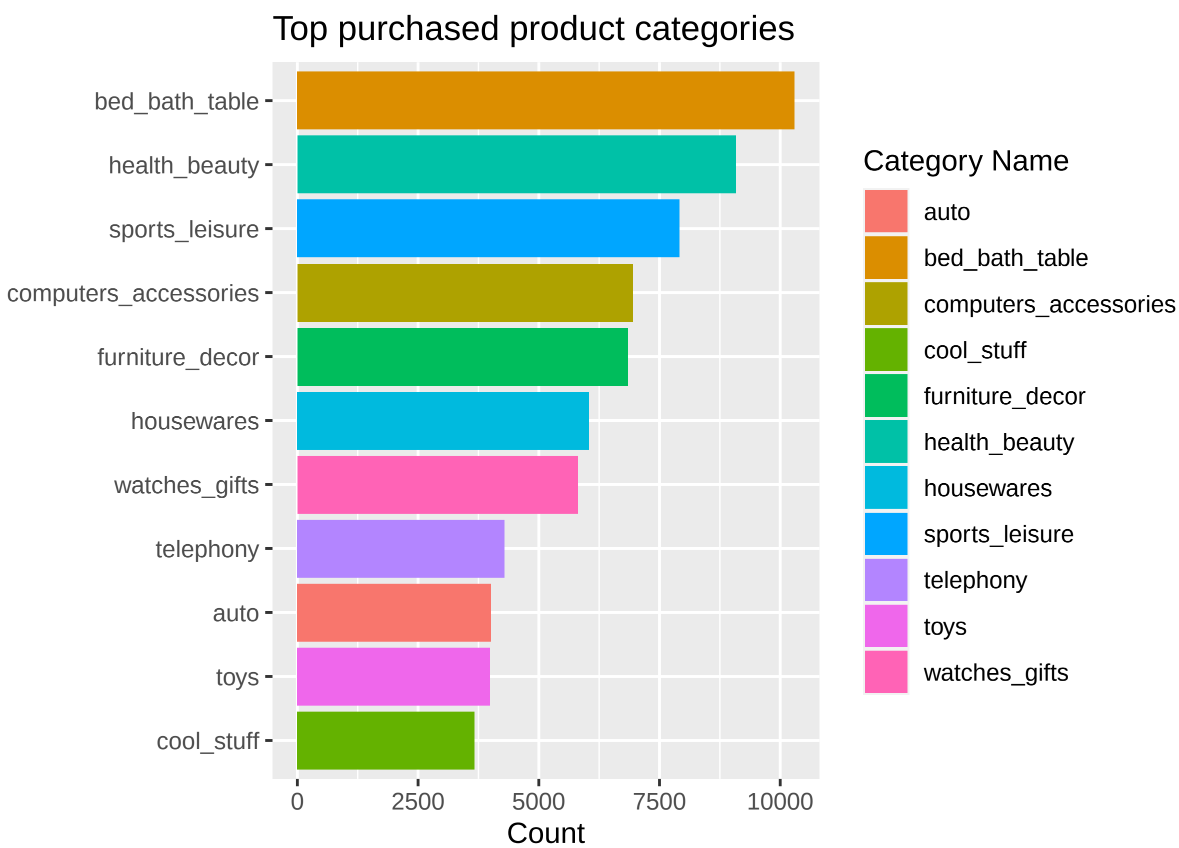
```
purchase_behavior <- df_all_unique %>% drop_na(n_payment_value,review_score) %>% group_by(customer_uniq
                                                        Most_Common_Month=as.numeric(format.Date(order_p
  reframe(Total_Purchases = n(), avg_payment=round(mean(n_payment_value)), total_payment=sum(n_payment_v
  arrange(desc(Total_Purchases))
```

```
# Group customers by geographic location and calculate summary statistics
grouped_by_state <- purchase_behavior %>%
  group_by(Most_Frequent_State) %>%
  summarize(
    avg_total_payment = mean(total_payment, na.rm = TRUE),
    avg_total_purchases = mean(Total_Purchases, na.rm = TRUE),
    avg_review_score = mean(avg_review_score, na.rm = TRUE))

# Group customers by most frequent category and calculate summary statistics
grouped_by_category <- purchase_behavior %>%
  group_by(Most_Frequent_Category) %>%
  summarize(
    avg_total_payment = mean(total_payment, na.rm = TRUE),
    avg_total_purchases = mean(Total_Purchases, na.rm = TRUE),
    avg_review_score = mean(avg_review_score, na.rm = TRUE))

# Create customer segments based on the summary statistics
grouped_by_state <- grouped_by_state %>%
  mutate(spending_segment = if_else(avg_total_payment > median(avg_total_payment), "High-Spending", "Lo

# Visualize and analyze the segments
ggplot(grouped_by_state, aes(x = reorder(Most_Frequent_State,desc(avg_total_payment)), y = avg_total_pa
  geom_bar(stat = "identity") +
  labs(title = "Average Total Payment by State and Spending Segment", x= "States")
```



- Top purchased product categories:

```
df_all_unique %>% full_join(category) %>%
  mutate(product_category_name_english=fct_lump_n(product_category_name_english, 11)) %>%
  drop_na(product_category_name_english) %>%
  group_by(product_category_name_english) %>%
  summarize(Count=n()) %>%
  filter(product_category_name_english != 'Other') %>%
  ggplot() +
    geom_bar(aes(x=Count,y=reorder(product_category_name_english,Count),fill=product_category_name_engli
    labs(title = 'Top purchased product categories',y='',fill='Category Name' )
```



2. Explore different segmentation techniques tailored to the characteristics of the Brazilian e-commerce market.

- The percent of customers whose bought multiple products:

```
resultx <- df_all_unique %>% count(customer_unique_id) %>%
  summarise(total_cust_count = n())

df_all_unique %>% group_by(customer_unique_id) %>%
  summarise(num_of_prod = n(),count=n_distinct(customer_unique_id)) %>%
  filter(num_of_prod > 1) %>%
  reframe(num_retained_cust = sum(count),total_cust = resultx$total_cust_count,
          retained_cust_perc = round(num_retained_cust/total_cust*100,2))
```

```
## # A tibble: 1 x 3
##   num_retained_cust total_cust retained_cust_perc
##               <int>      <int>              <dbl>
## 1              5941      96090               6.18
```

- Highest states in purchasing:

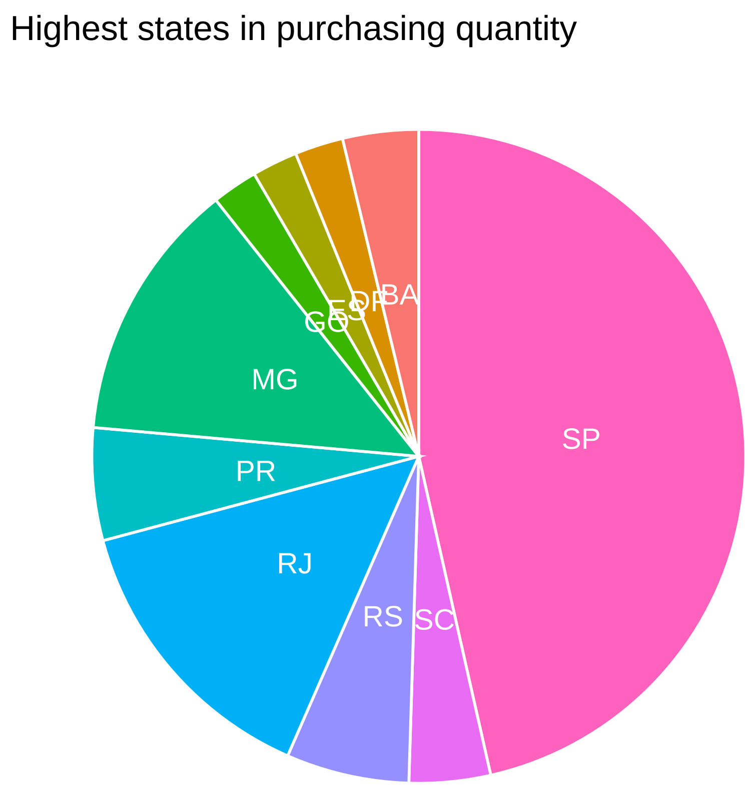
```
df_all_unique %>%
  mutate(customer_state=fct_lump_n(customer_state, 10)) %>%
  drop_na(customer_state) %>%
```

```r
  group_by(customer_state) %>%
  summarize(Count=n()) %>%
  filter(customer_state != 'Other') %>%
  arrange(desc(customer_state))
```

```
## # A tibble: 10 x 2
##    customer_state Count
##    <fct>          <int>
##  1 SP             43653
##  2 SC              3775
##  3 RS              5717
##  4 RJ             13415
##  5 PR              5223
##  6 MG             12150
##  7 GO              2126
##  8 ES              2118
##  9 DF              2248
## 10 BA              3522
```

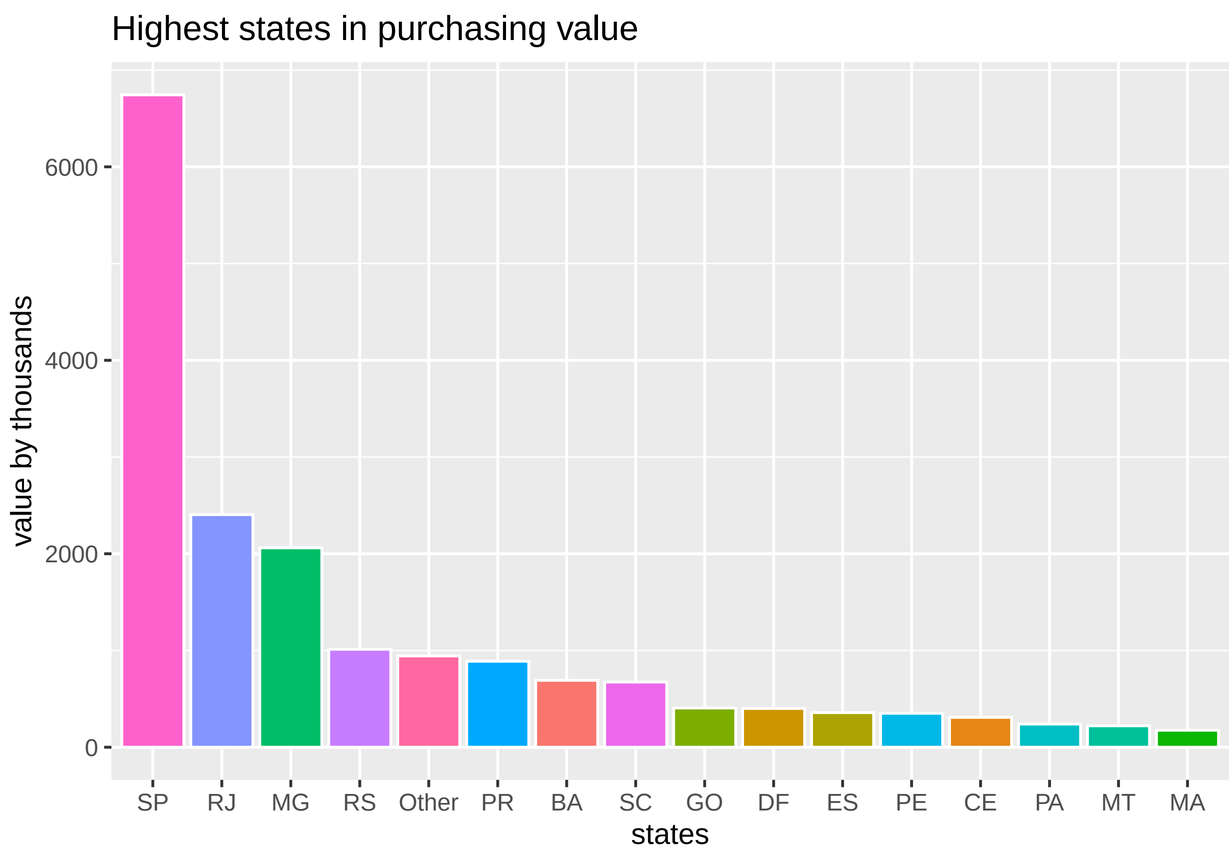
- Highest states in purchasing pie chart:

```r
df_all_unique %>%
  mutate(customer_state=fct_lump_n(customer_state, 10)) %>%
  drop_na(customer_state) %>%
  group_by(customer_state) %>%
  summarize(Count=n()) %>%
  filter(customer_state != 'Other') %>%
  arrange(desc(customer_state)) %>%
  mutate(prop = Count / sum(Count) *100) %>%
  mutate(ypos = cumsum(prop)- 0.5*prop ) %>%
  ggplot(aes(x="", y=prop, fill=customer_state)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  theme_void() +
  theme(legend.position="none") +
  geom_text(aes(y = ypos, label = customer_state), color = "white", size=4) +
  labs(title = 'Highest states in purchasing quantity')
```



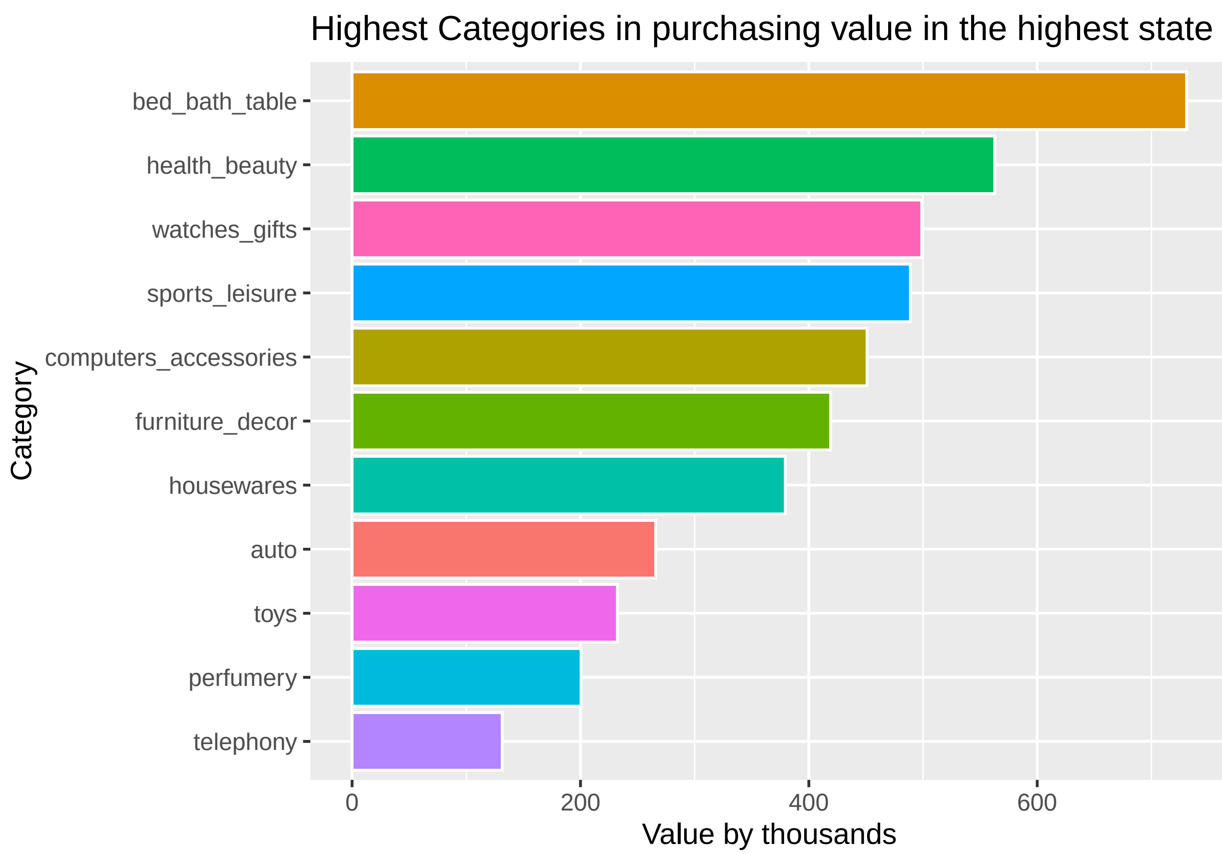
- Highest states in purchasing value:

```
df_all_unique %>% drop_na(n_payment_value) %>%
  mutate(customer_state=fct_lump_n(customer_state, 15)) %>%
  drop_na(customer_state) %>%
  group_by(customer_state) %>%
  summarize(total_value_k=sum(n_payment_value)/1000) %>%
  ggplot(aes(reorder(customer_state,desc(total_value_k)), total_value_k, fill=customer_state)) +
  geom_bar(stat="identity", color="white") +
  labs(title = 'Highest states in purchasing value', x='states', y='value by thousands')+
  theme(legend.position = "none")  # Remove the legend
```



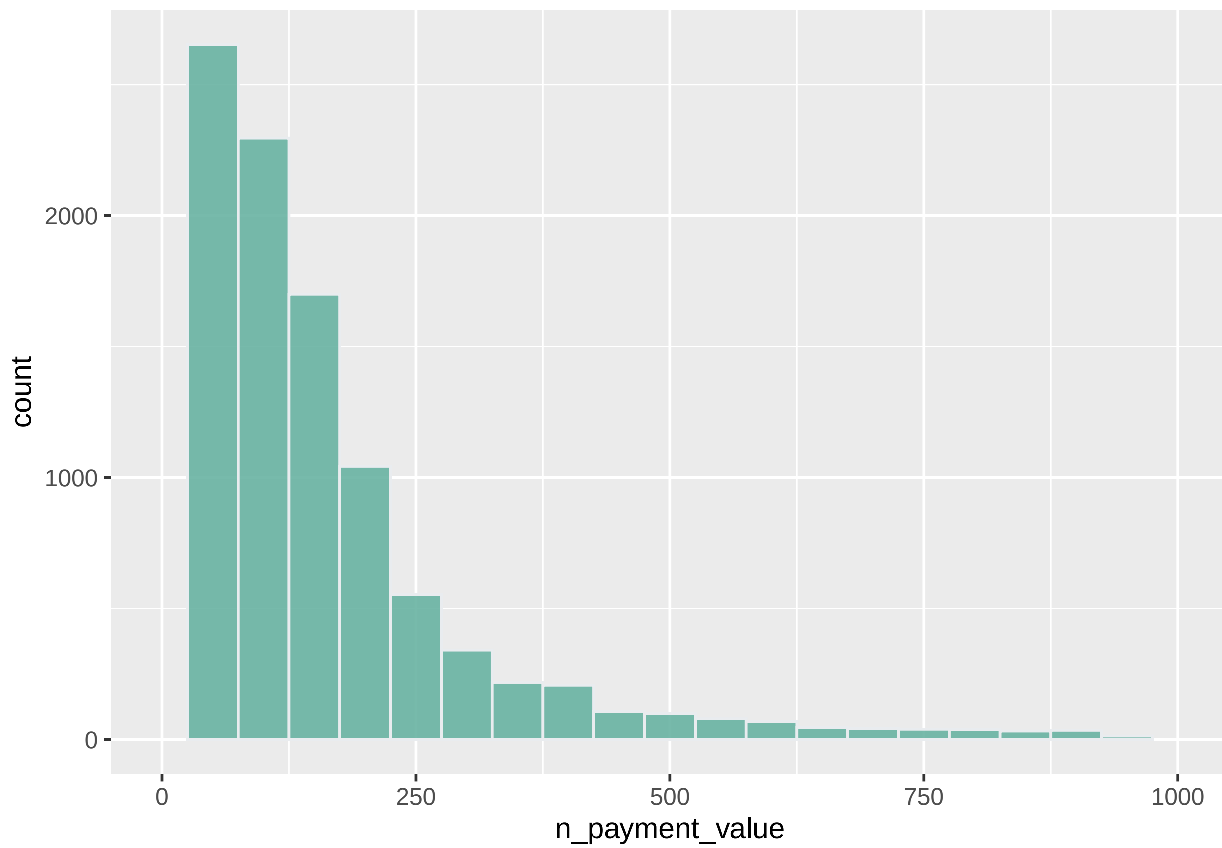
- Highest category in purchasing value in SP state:

```
df_all_unique %>% drop_na(n_payment_value) %>%
  mutate(customer_state=fct_lump_n(customer_state, 1)) %>%
  drop_na(customer_state) %>%
  group_by(customer_state) %>% filter(customer_state != "Other") %>%
  mutate(product_category_name_english=fct_lump_n(product_category_name_english, 11)) %>%
  group_by(product_category_name_english) %>% filter(product_category_name_english != "Other") %>%
  summarize(total_value_k=sum(n_payment_value)/1000) %>%
  ggplot(aes(y=reorder(product_category_name_english,total_value_k), total_value_k, fill=product_catego
  geom_bar(stat="identity", color="white") +
  labs(title = 'Highest Categories in purchasing value in the highest state', y='Category', x='Value by
    theme(legend.position = "none")  # Remove the legend
```



- Histogram with the purchasing values:

```
df_all_unique %>% drop_na() %>%
  ggplot() +
    geom_histogram(aes(n_payment_value),binwidth=50, fill="#69b3a2",color="#e9ecef", alpha=0.9)+
    xlim(0, 1000)
```



- **Create Churn Analysis:**
  1. Analyze customer churn rate over time, identifying factors influencing churn and proposing strategies to retain customers.

  – create a Churn Analysis:

```
# Add df_all_unique to data
data <- data.frame(
  customer_unique_id = df_all_unique$customer_unique_id,
  order_purchase_timestamp = as.Date(df_all_unique$order_purchase_timestamp)
)

# Convert OrderDate to month
data$OrderMonth <- format(data$order_purchase_timestamp, "%Y-%m")

# Calculate churn rate
churn_rate <- function(data) {
  result <- data.frame()
  for (i in 2:length(unique(data$OrderMonth))) {
    prev_month <- unique(data$OrderMonth)[i - 1]
    current_month <- unique(data$OrderMonth)[i]

    # Active customers in previous month
    active_customers_prev_month <- unique(data[data$OrderMonth == prev_month, "customer_unique_id"])
```

```r
    # Active customers in current month
    active_customers_current_month <- unique(data[data$OrderMonth == current_month, "customer_unique_id

    # Churned customers
    churned_customers <- setdiff(active_customers_prev_month, active_customers_current_month)

    # Retained customers
    retained_customers <- intersect(active_customers_prev_month, active_customers_current_month)

    # Churn rate calculation
    churn_rate <- length(churned_customers) / length(active_customers_prev_month) * 100

    # Append result
    result <- rbind(result, data.frame(Month = current_month, ChurnRate = churn_rate, Num_Churned_Custom
  }
  result <- arrange(result, Month) %>%
    mutate(Retention = 100 - ChurnRate) %>%
    select(Month, ChurnRate, Retention,Num_Churned_Customers,Num_Retained_Customers)
  return(result)
}

# Calculate churn rate
churnMonthByMonth <- churn_rate(data)
head(churnMonthByMonth)
```

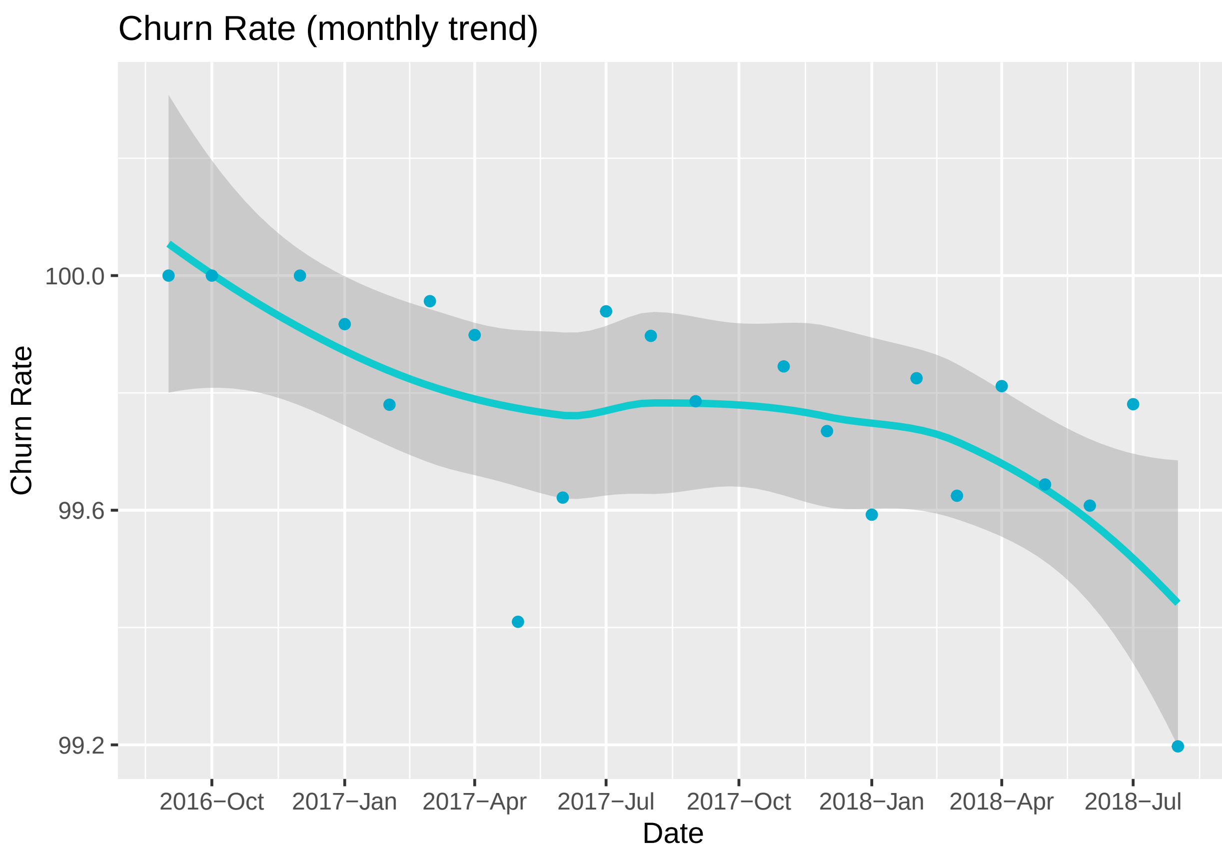
```
##     Month ChurnRate  Retention Num_Churned_Customers Num_Retained_Customers
## 1 2016-09 100.00000 0.00000000                   321                      0
## 2 2016-10 100.00000 0.00000000                  1755                      0
## 3 2016-12 100.00000 0.00000000                     4                      0
## 4 2017-01  99.91724 0.08275862                  3622                      3
## 5 2017-02  99.77987 0.22012579                  3173                      7
## 6 2017-03  99.95641 0.04359198                  6879                      3
```

```r
churnMonthByMonth %>% summarise(avg_ChurnRate=mean(ChurnRate),avg_Retention=mean(Retention))
```

```
##   avg_ChurnRate avg_Retention
## 1      99.76679     0.2332122
```

```r
ggplot(churnMonthByMonth) +
  geom_smooth(aes(ym(Month), ChurnRate),group=T,linewidth = 1.3,colour="#10cacd")+
  geom_point(aes(ym(Month), ChurnRate),group=T,colour="#00aacd")+
  scale_x_date(date_breaks = '3 months',date_labels='%Y-%b') +
  labs(title = 'Churn Rate (monthly trend)',x='Date',y='Churn Rate')
```

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



- find the purchasing behavior for the Churned Customers.

```
data2 <- data.frame(
  customer_unique_id = df_all_unique$customer_unique_id,
  order_purchase_timestamp = as.Date(df_all_unique$order_purchase_timestamp))

# Convert OrderDate to month
data2$OrderMonth <- format(data2$order_purchase_timestamp, "%Y-%m")

# Calculate churn and retention customers
churn_and_retention_cust <- function(data2) {
  churned_cust <- data.frame()
  retained_cust <- data.frame()

  for (i in 2:length(unique(data2$OrderMonth))) {
    prev_month <- unique(data2$OrderMonth)[i - 1]
    current_month <- unique(data2$OrderMonth)[i]

    # Active customers in previous month
    active_customers_prev_month <- unique(data2[data2$OrderMonth == prev_month, "customer_unique_id"])

    # Active customers in current month
    active_customers_current_month <- unique(data2[data2$OrderMonth == current_month, "customer_unique_

    # Churned customers
```

```r
    churned_customers <- setdiff(active_customers_prev_month, active_customers_current_month)

    # Retained customers
    retained_customers <- intersect(active_customers_prev_month, active_customers_current_month)

    # Append churned customers
    churned_cust <- rbind(churned_cust, data.frame(Churned_Customers = churned_customers)) %>% unique()

    # Append retained customers
    retained_cust <- rbind(retained_cust, data.frame(Retained_Customers = retained_customers)) %>% uniq
  }

  return(list(Churned_Customers = churned_cust, Retained_Customers = retained_cust))
}

# Calculate churned and retained customers
churned_and_retained <- churn_and_retention_cust(data2)
churned_cust <- churned_and_retained$Churned_Customers %>% unique()
retained_cust <- churned_and_retained$Retained_Customers %>% unique()
churned_cust <- data.frame(Churned_Customers = churned_cust[!(churned_cust$Churned_Customers %in% retai
```

```r
# Join with the purchasing behavior dataset
churned_cust <- inner_join(churned_cust, purchase_behavior, by = c("Churned_Customers" = "customer_uniq

Mode(churned_cust$Most_Frequent_Category)
```

```
## [1] "bed_bath_table"
```

```r
Mode(churned_cust$Most_Frequent_State)
```

```
## [1] "SP"
```

```r
Mode(churned_cust$Most_Common_Month)
```

```
## [1] 8
```

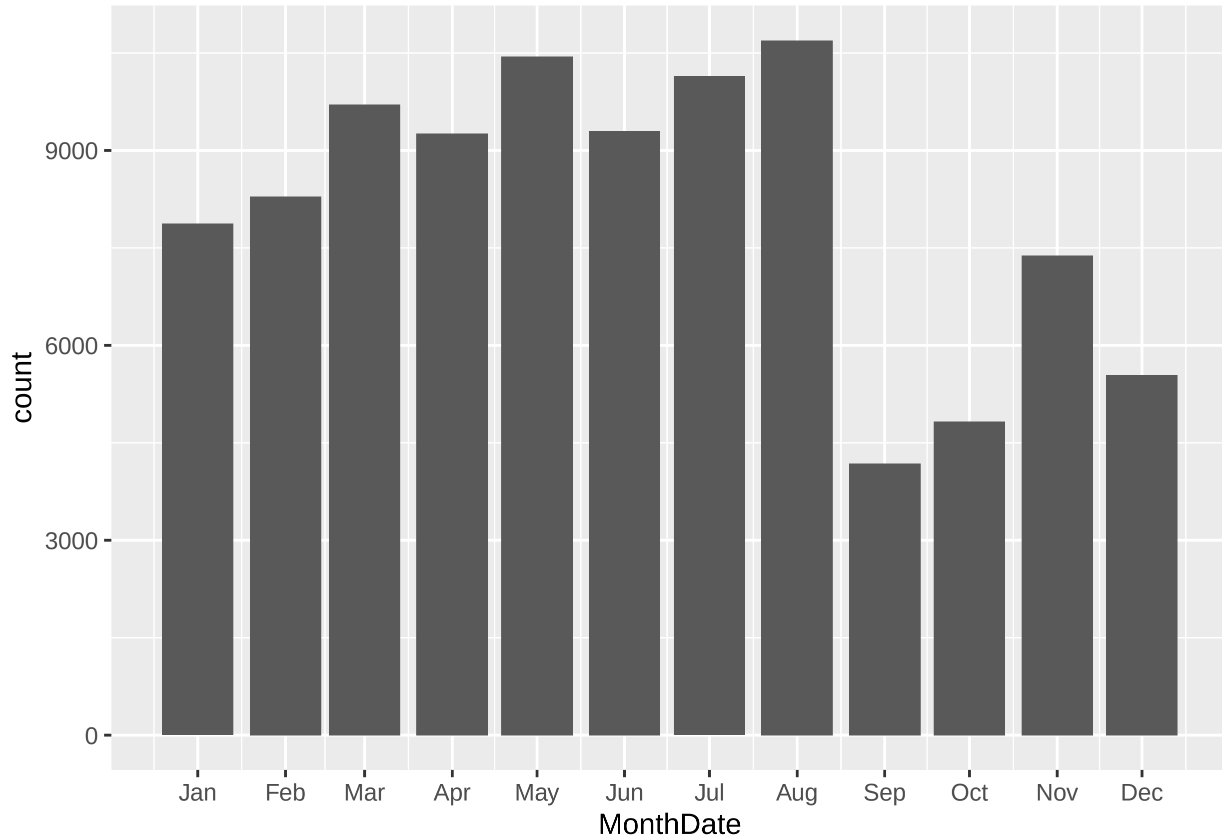
```r
Mode(churned_cust$Most_Common_Year)
```

```
## [1] "2018"
```

```r
Mode(churned_cust$review_score)
```

```
## [1] 5
```

```r
month_num <- match(tolower(churned_cust$Most_Common_Month), tolower(month.name))
churned_cust$MonthDate <- as.Date(paste0("2017-", churned_cust$Most_Common_Month, "-01"))

ggplot(churned_cust) +
  geom_bar(aes(x = MonthDate)) +
  scale_x_date(date_labels = "%b", date_breaks = "1 month")
```




```r
retained_cust <- inner_join(retained_cust, purchase_behavior, by = c("Retained_Customers" = "customer_u

Mode(retained_cust$Most_Frequent_Category)
```

```
## [1] "bed_bath_table"
```

```r
Mode(retained_cust$Most_Frequent_State)
```

```
## [1] "SP"
```

```r
Mode(retained_cust$Most_Common_Month)
```

```
## [1] 7
```

```r
Mode(retained_cust$Most_Common_Year)
```

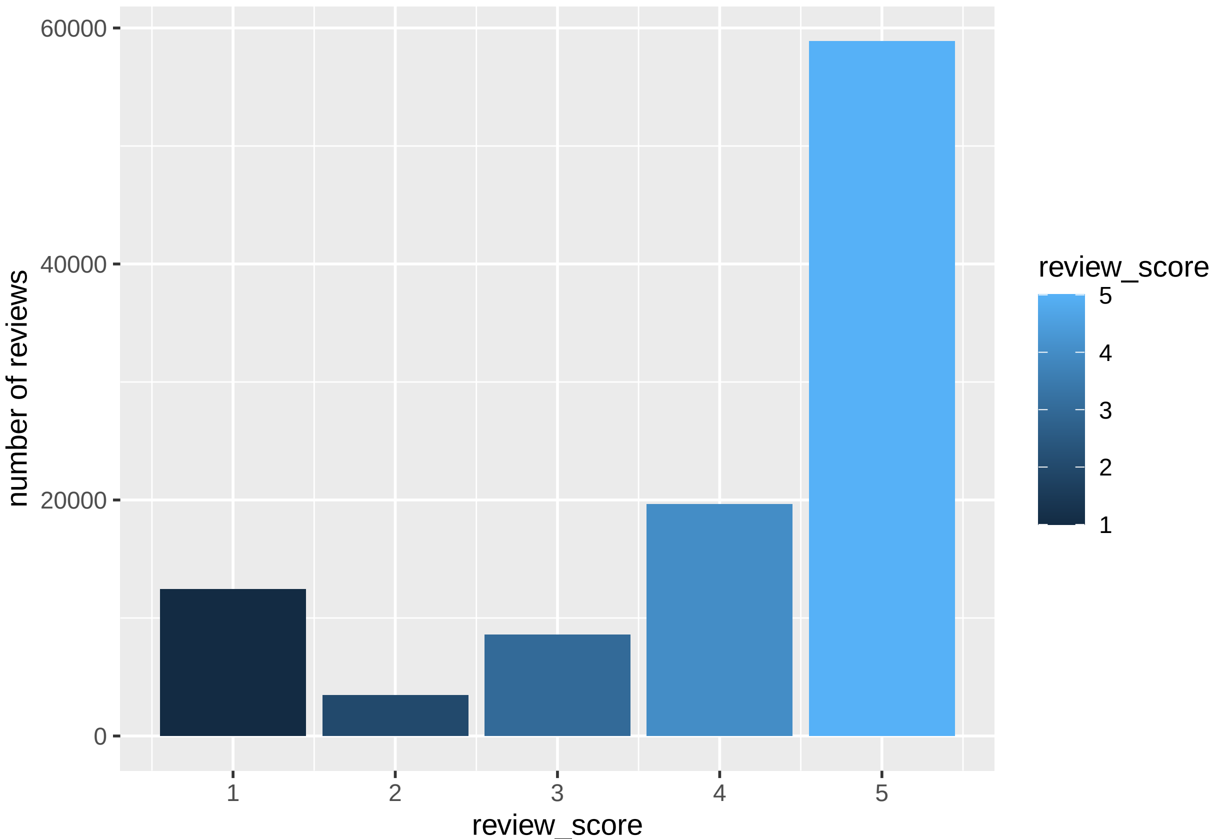
```
## [1] "2018"
```

```r
Mode(retained_cust$review_score)
```

```
## [1] 5
```

- Count of review scores:

```
df_all_unique %>% count(review_score) %>%
  drop_na() %>%
  arrange(desc(n)) %>%
  ggplot() +
  geom_col(aes(review_score,n,fill=review_score)) +
  labs(y='number of reviews')
```



- The most repeated review comment in the scored 1 reviews:

```
df_all_unique %>% group_by(review_comment_message,review_score,customer_state) %>%
  filter(review_score==1, review_comment_message!='NA',order_status=='delivered') %>%
  select(review_comment_title,review_score,review_comment_message) %>%
  count(review_comment_message) %>%
  rename(count_of_review=n) %>%
  arrange(desc(count_of_review))
```

```
## Adding missing grouping variables: `customer_state`

## # A tibble: 7,149 x 4
## # Groups:   review_comment_message, review_score, customer_state [7,149]
##    review_comment_message            review_score customer_state count_of_review
##    <chr>                                    <dbl> <chr>                    <int>
##  1 "Não recebi o produto"                       1 RJ                          13
##  2 "Não recebi"                                 1 SP                           7
```

```
##  3 "Não recebi o produto"                1 SP                      7
##  4 "Ainda quero o produtoq que comp~     1 PR                      6
##  5 "Fiz pedido de 6 produtos e só r~     1 MT                      6
##  6 "Ta faltando um difusor"              1 SP                      6
##  7 "não chegaram todos os produtos\~     1 RS                      6
##  8 "Ainda não recebi o produto"          1 RJ                      5
##  9 "Ainda não recebi o produto"          1 SP                      5
## 10 "Ainda não recebi, mas prazo ate~     1 MA                      5
## # i 7,139 more rows
```

The most frequent review comment is 'I didn't receive the product', although the order status is delivered.

- The AVG lowest sellers in review score:

```r
df_all_unique %>% group_by(seller_id,seller_state) %>%
  reframe(AVG_score=round(mean(review_score)),count_of_reviews=n()) %>%
  arrange(AVG_score,desc(count_of_reviews))
```

```
## # A tibble: 3,096 x 4
##    seller_id                        seller_state AVG_score count_of_reviews
##    <chr>                            <chr>            <dbl>            <int>
##  1 ec2e006556300a79a5a91e4876ab3a56 SP                   1                4
##  2 fc6295add6f51a0936407ead70c1001d SP                   1                4
##  3 4e42581f08e8cfc7c090f930bac4552a SP                   1                3
##  4 61f159ef6da2d441951d2c0efa719362 ES                   1                3
##  5 8d92f3ea807b89465643c219455e7369 SP                   1                3
##  6 90d4125885ab6c86e8820a722be71974 SP                   1                3
##  7 a2e85714b56b1cb6bb24a9a6e6cad36f SP                   1                3
##  8 d65f31d2413268e671989503f6cf9993 SP                   1                3
##  9 f7df46c1e0ec44eed5c6726478da4a17 RS                   1                3
## 10 0aa124728afc1131dff5655f4c6f487b MG                   1                2
## # i 3,086 more rows
```

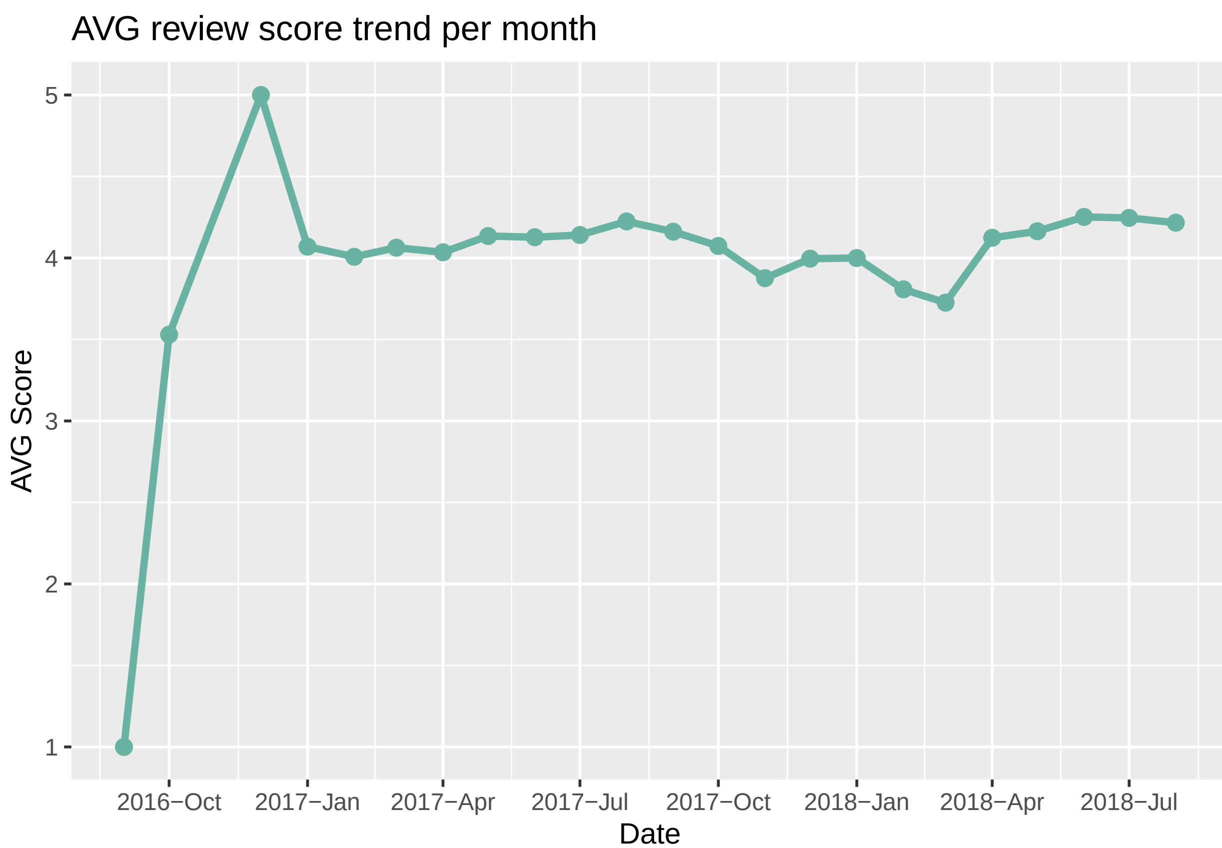
- The AVG lowest cities in review score:

```r
df_all_unique %>% group_by(customer_city) %>%
  drop_na(review_score) %>%
  summarise(AVG_score=round(mean(review_score)),count_of_reviews=n()) %>%
  arrange(AVG_score,desc(count_of_reviews))
```

```
## # A tibble: 4,117 x 3
##    customer_city        AVG_score count_of_reviews
##    <chr>                    <dbl>            <int>
##  1 cafeara                      1                3
##  2 itororo                      1                3
##  3 maioba                       1                3
##  4 mercedes                     1                3
##  5 arembepe                     1                2
##  6 brasilandia de minas         1                2
##  7 buriti                       1                2
##  8 engenheiro beltrao           1                2
##  9 itacoatiara                  1                2
## 10 macaubal                     1                2
## # i 4,107 more rows
```

- AVG review score trend per month:

```
df_all_unique %>% drop_na(review_score) %>% group_by(date=format(order_purchase_timestamp,'%Y-%m')) %>%
  summarise(AVG_score=mean(review_score)) %>%
  ggplot(aes(ym(date),y=AVG_score)) +
  geom_line(linewidth = 1.3,colour="#69b3a2") +
  geom_point(size=2.5,colour="#69b3a2") +
  scale_x_date(date_breaks = '3 months',date_labels='%Y-%b') +
  labs(title = 'AVG review score trend per month', x='Date', y='AVG Score')
```



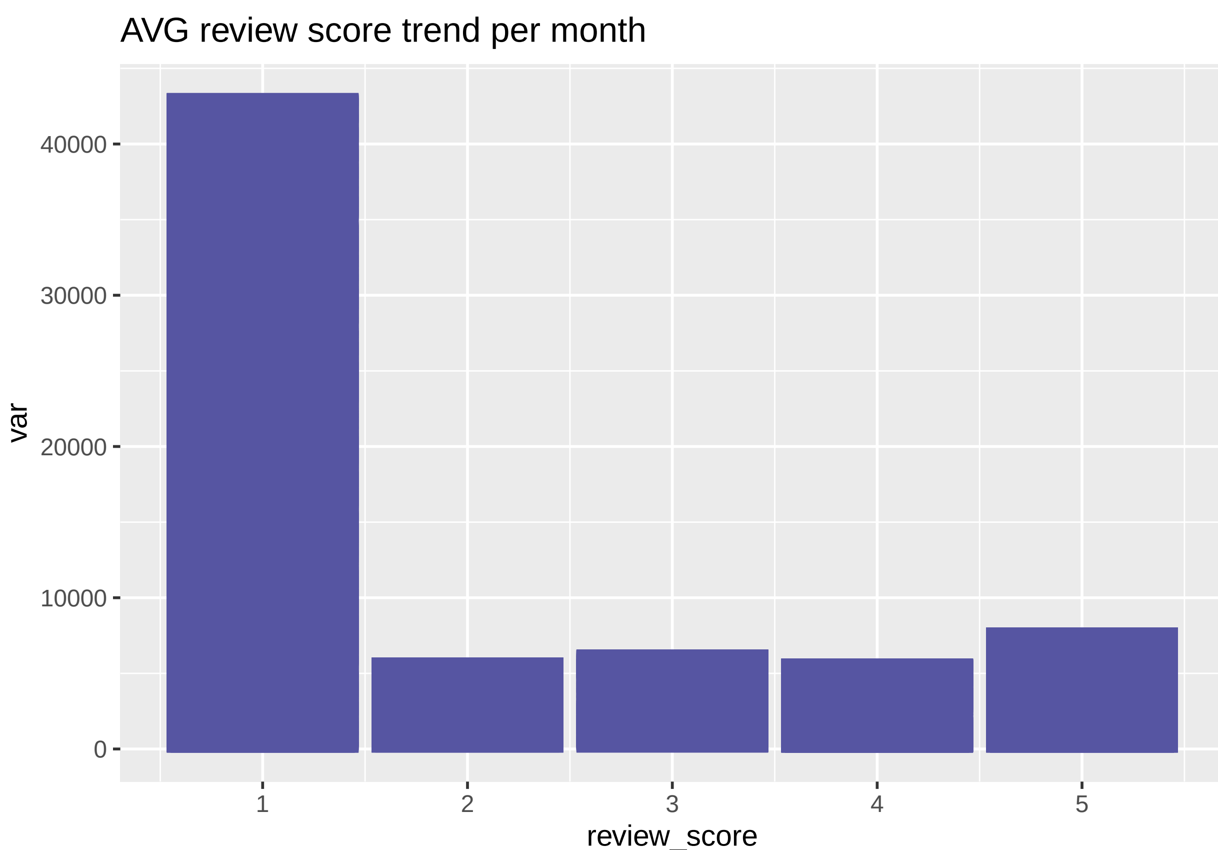
2. Examine the impact of product quality, delivery speed, and customer service on customer satisfaction and retention.

- The impact of the delayed orders on the review score.

```
count_of_1_review <- df_all_unique %>% drop_na(review_score) %>%
  filter(order_status == 'delivered',review_score==1) %>%
  summarise(count_of_1=n())

df_all_unique %>% drop_na(review_score) %>%
  mutate(var=date(order_delivered_customer_date) - date(order_estimated_delivery_date)) %>%
  filter(order_status == 'delivered',var>0, review_score==1) %>%
  summarise(percent_of_delay=n()/count_of_1_review$count_of_1*100)
```

```
## # A tibble: 1 x 1
##   percent_of_delay
##              <dbl>
## 1             33.8
```

```
df_all_unique %>% drop_na(review_score) %>%
  mutate(var=date(order_delivered_customer_date) - date(order_estimated_delivery_date)) %>%
  filter(order_status == 'delivered',var>0) %>%
  ggplot() +
  geom_col(aes(var,x=review_score), linewidth = 1.3,colour="#5655a2") +
  labs(title = 'AVG review score trend per month')
```



- Customer Segmentation using K-Means Clustering based on purchase_behavior

```
set.seed(123)
kmeans_result <- kmeans(purchase_behavior[, c("Total_Purchases", "avg_payment")], centers = 5)
purchase_behavior$Segment <- kmeans_result$cluster
```

- **Design a Predictive Model:**
  1. Build predictive models to forecast sales, customer demand, or product popularity.
  - Building a Predictive Modeling for Sales Forecasting:

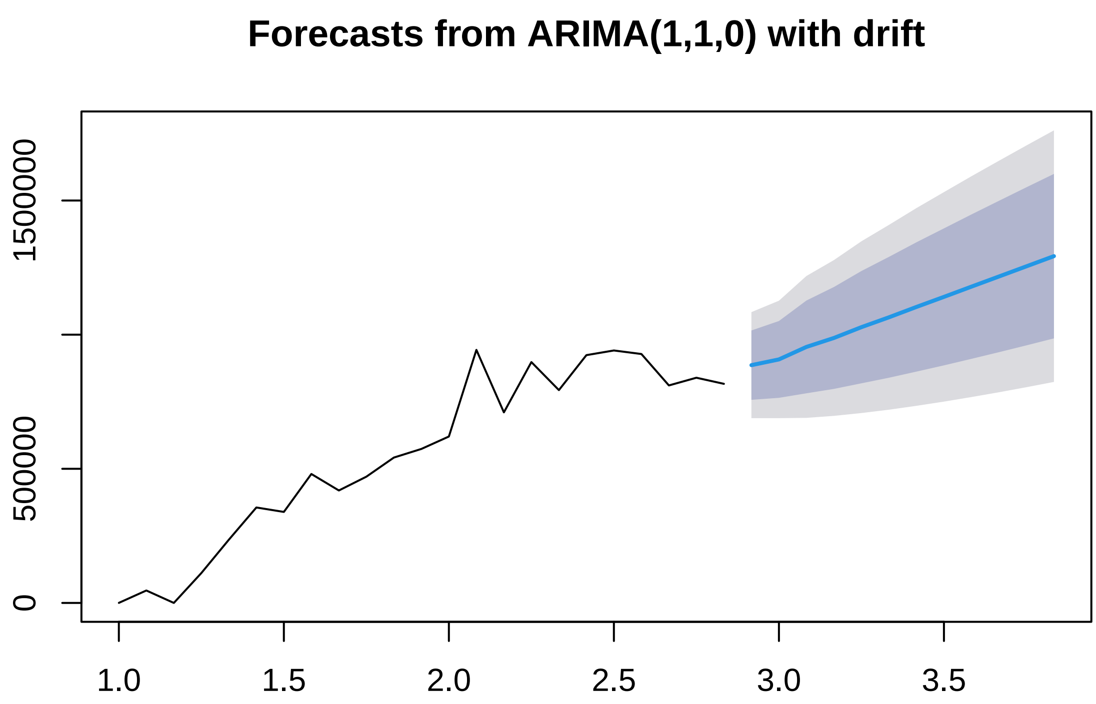
```r
# Aggregate sales data by month
monthly_sales <- df_all_unique %>% drop_na(price) %>%
  group_by(Month = floor_date(order_purchase_timestamp, "month")) %>%
  summarise(Total_Sales = sum(price))
```

- Time Series Forecasting using ARIMA

```r
sales_ts <- ts(monthly_sales$Total_Sales, frequency = 12)
arima_model <- auto.arima(sales_ts)
forecasted_sales <- forecast(arima_model, h = 12) # Forecasting next 12 months
```

- Plotting the forecast

```r
plot(forecasted_sales)
```



- **Visualize the findings and reporting it:**
  - **The findings:**
    * The company is on a growing scale since the beginning.
    * The price & quantity sales seasonality peak is in the fourth quarter of the year.
    * Most selling categories are related to home furniture, beauty products, and sports.
    * The highest states in purchasing value are RJ, MG, and PR.
    * The payments value distribution are between 50 – 200.

- * The company's rating is overall high, but the order delivery delay have a high impact on the low ratings.
- **The Recommendations:**
  * Develop a main dashboard with the most selling categories in the company user's app.
  * Prioritize the low-price items in the search feature in the user's app, to enhance the customer experience.
  * Minimize the order delivery duration, to enhance the customer's reviews score or increase the estimated delivery time to develop honesty with the customers.
  * Cooperation with the SEO team to increase the advertising in the highest states and cities in purchasing, to increase the customer base.
- Link for the repository on GitHub: https://github.com/a7mdNasrr/Olist-Brazilian-E-commerce-Analysis-Project