Olist Association and Clustering

Lydia Savatsky | Sourabh Koul | Kalpavalli Kiran Phaniharam | Yujun Wang | Yiyuan Zhang

```
library(tidyverse)
library(plyr)
library(dplyr)
library(arules)
library(fastDummies)
library(stats)
library(ggpubr)
library(factoextra)
library(cluster)
library(arulesViz)
library(RColorBrewer)

setwd("G:/My Drive/Business Analytics in R/Homeworks/Homework2")

customers <- read.csv('olist_customers_dataset.csv')
items <- read.csv('olist_order_items_dataset.csv')
orders <- read.csv('olist_orders_dataset.csv')
products <- read.csv('olist_products_dataset.csv')
translation <- read.csv('product_category_name_translation.csv')
```

Pre-processing the the Data

To clean the data, we combined the data sets items, products, and product category.

```
category_association <- left_join(items,
                                  products,
                                  by=c("product_id"="product_id"))
category_association <- left_join(category_association,
                                  translation,
                                  by=c("product_category_name"="ï..product_category_name"))


category_association_final = subset(category_association,
                                    product_category_name_english!="NA")

category_association_final <- category_association_final %>%
  mutate(product_category_name_english = as.factor(product_category_name_english)) %>%
  select(-c('product_category_name'))
```

1. Are there any interesting relationships among different product categories?

To answer this question, we performed Association Rule Mining.

We first created a data frame called transactionData so that we have a column for all product categories purchased per order. Having this column will allow us to create association rules on this column.

```
order_cat <- category_association_final %>% select('order_id','product_category_name_english')
dist_order_cat <- distinct(order_cat)
transactionData <- ddply(dist_order_cat,c("order_id"),
                         function(df1)paste(df1$product_category_name_english,
                                            collapse = ","))
```

We tried to understand how many transactions have more than one product category to give us an idea on how much min support is need to be input in the apriori algorithm. We viewed these data frame separately.

```
order_items <- data.frame(table(dist_order_cat$order_id))
mult_items <- order_items[order_items$Freq > 1,]
```

We wrote the transaction data out as a csv in order to read the data back in as transaction data in the basket format.

```
tr <- transactionData %>% select('V1')
names(tr) = 'items'
write.csv(tr,"market_basket_transactions.csv", quote = FALSE, row.names = FALSE)
transaction <- read.transactions('market_basket_transactions.csv',
                         format = 'basket', sep=',')
```

The following allows us to understand which (unique) items are there, in all shopping baskets

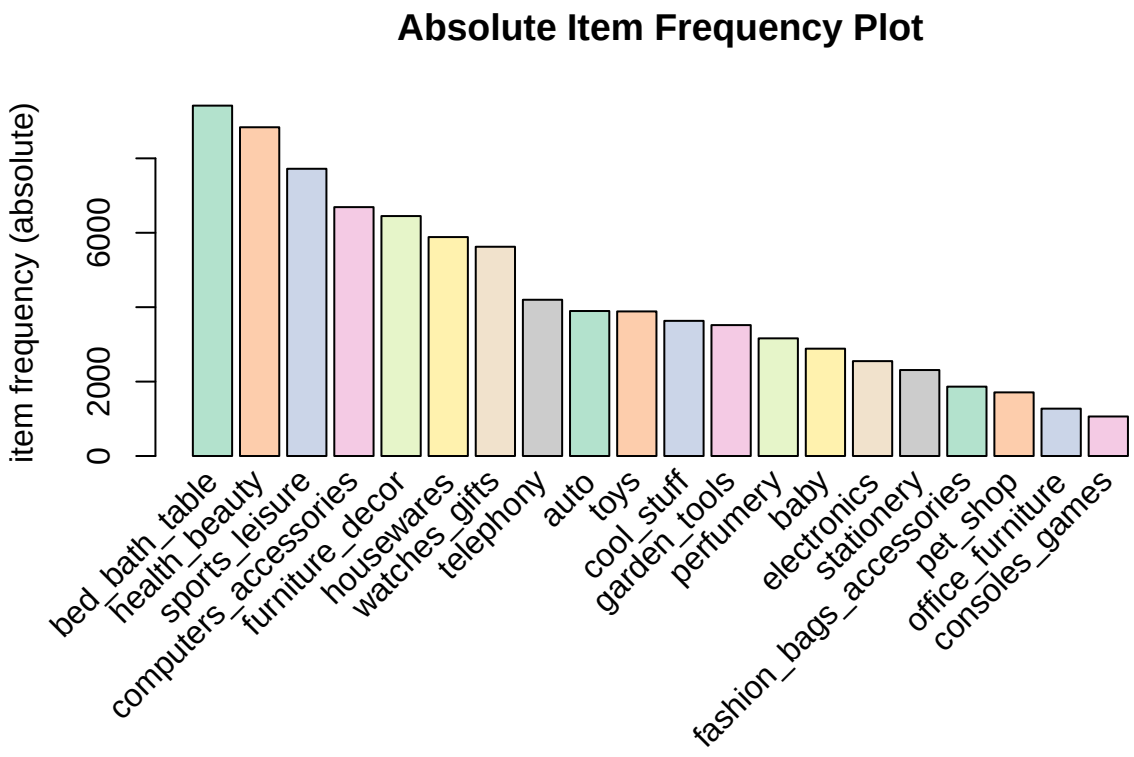
```
itemInfo(transaction)
```

```
##                                    labels
## 1               agro_industry_and_commerce
## 2                         air_conditioning
## 3                                      art
## 4                     arts_and_craftmanship
## 5                                    audio
## 6                                     auto
## 7                                     baby
## 8                             bed_bath_table
## 9                    books_general_interest
## 10                           books_imported
## 11                          books_technical
## 12                        cds_dvds_musicals
## 13                       christmas_supplies
## 14                               cine_photo
## 15                                computers
## 16                    computers_accessories
## 17                           consoles_games
## 18         construction_tools_construction
## 19               construction_tools_lights
## 20               construction_tools_safety
## 21                               cool_stuff
## 22                costruction_tools_garden
## 23                  costruction_tools_tools
```

```
## 24                      diapers_and_hygiene
## 25                                   drinks
## 26                              dvds_blu_ray
## 27                               electronics
## 28                    fashio_female_clothing
## 29                  fashion_bags_accessories
## 30                 fashion_childrens_clothes
## 31                      fashion_male_clothing
## 32                              fashion_shoes
## 33                              fashion_sport
## 34                   fashion_underwear_beach
## 35                             fixed_telephony
## 36                                    flowers
## 37                                       food
## 38                                 food_drink
## 39                          furniture_bedroom
## 40                            furniture_decor
## 41                       furniture_living_room
## 42         furniture_mattress_and_upholstery
## 43                               garden_tools
## 44                             health_beauty
## 45                           home_appliances
## 46                         home_appliances_2
## 47                            home_comfort_2
## 48                              home_confort
## 49                         home_construction
## 50                                housewares
## 51            industry_commerce_and_business
## 52                                     items
## 53 kitchen_dining_laundry_garden_furniture
## 54                                la_cuisine
## 55                       luggage_accessories
## 56                              market_place
## 57                                     music
## 58                       musical_instruments
## 59                          office_furniture
## 60                            party_supplies
## 61                                 perfumery
## 62                                  pet_shop
## 63                     security_and_services
## 64                    signaling_and_security
## 65                          small_appliances
## 66   small_appliances_home_oven_and_coffee
## 67                             sports_leisure
## 68                                 stationery
## 69                    tablets_printing_image
## 70                                 telephony
## 71                                      toys
## 72                               watches_gifts
```

To visualize top frequently purchased product categories, we created the following graph.

```
itemFrequencyPlot(transaction,
                  topN=20,
                  type="absolute",
                  col=brewer.pal(8,'Pastel2'),
                  main="Absolute Item Frequency Plot")
```



Creating association rules

We understood that roughly 0.8% of orders have more than one product category attached to it. Hence we will run the apriori algotrithms with low support. After various iterations, we will be able to generate rules with min support as 0.0001 and confidence 0.001

```
association.rules <- apriori(transaction, parameter = list(supp=0.0001, conf=0.001,minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##       0.001    0.1    1 none FALSE            TRUE       5   1e-04      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
```

```
## Absolute minimum support count: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[72 item(s), 97257 transaction(s)] done [0.01s].
## sorting and recoding items ... [69 item(s)] done [0.00s].
## creating transaction tree ... done [0.02s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [34 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

We were able to generate 34 rules using the above parameters

```
association.rules
```

```
## set of 34 rules
```

We created a subset of top 20 rules sorted on the basis of confidence

```
top20subRules <- head(association.rules, n = 20, by = "confidence")
inspect(top20subRules)
```

```
##      lhs                             rhs                support      confidence
## [1]  {home_confort}                => {bed_bath_table}   0.0004421276 0.108312343
## [2]  {construction_tools_lights} => {furniture_decor}  0.0001131024 0.045081967
## [3]  {home_construction}          => {furniture_decor}  0.0001336665 0.026530612
## [4]  {furniture_decor}            => {bed_bath_table}   0.0007197425 0.010854396
## [5]  {bed_bath_table}             => {furniture_decor}  0.0007197425 0.007433365
## [6]  {baby}                       => {cool_stuff}       0.0002056407 0.006932409
## [7]  {baby}                       => {toys}             0.0001953587 0.006585789
## [8]  {baby}                       => {bed_bath_table}   0.0001747946 0.005892548
## [9]  {cool_stuff}                 => {baby}             0.0002056407 0.005506608
## [10] {toys}                       => {baby}             0.0001953587 0.004889346
## [11] {garden_tools}               => {furniture_decor}  0.0001747946 0.004832291
## [12] {bed_bath_table}             => {home_confort}     0.0004421276 0.004566210
## [13] {baby}                       => {furniture_decor}  0.0001233844 0.004159445
## [14] {housewares}                 => {furniture_decor}  0.0002467689 0.004078858
## [15] {perfumery}                  => {health_beauty}    0.0001233844 0.003795066
## [16] {furniture_decor}            => {housewares}       0.0002467689 0.003721507
## [17] {housewares}                 => {bed_bath_table}   0.0002056407 0.003399048
## [18] {garden_tools}               => {housewares}       0.0001131024 0.003126777
## [19] {cool_stuff}                 => {bed_bath_table}   0.0001028204 0.002753304
## [20] {furniture_decor}            => {garden_tools}     0.0001747946 0.002636068
##      coverage    lift       count
## [1]  0.004081968 1.11862945 43
## [2]  0.002508817 0.67987857 11
## [3]  0.005038198 0.40010665 13
## [4]  0.066308852 0.11210216 70
## [5]  0.096825935 0.11210216 70
## [6]  0.029663675 0.18563472 20
## [7]  0.029663675 0.16482605 19
## [8]  0.029663675 0.06085712 17
## [9]  0.037344356 0.18563472 20
## [10] 0.039955993 0.16482605 19
```

```
## [11] 0.036172204 0.07287551 17
## [12] 0.096825935 1.11862945 43
## [13] 0.029663675 0.06272836 12
## [14] 0.060499501 0.06151302 24
## [15] 0.032511799 0.04177193 12
## [16] 0.066308852 0.06151302 24
## [17] 0.060499501 0.03510473 20
## [18] 0.036172204 0.05168268 11
## [19] 0.037344356 0.02843560 10
## [20] 0.066308852 0.07287551 17
```

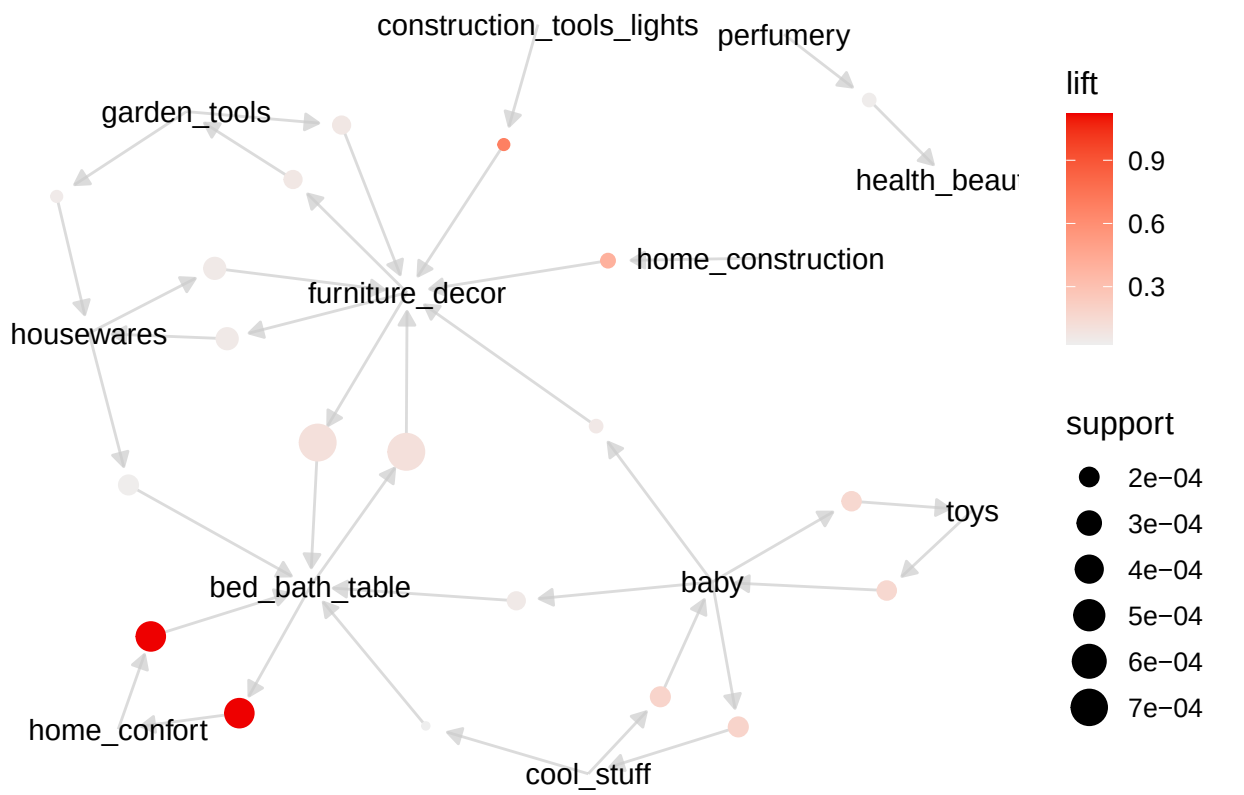
Visualization of Assocation Rules

We wanted to visualize the rules on the basis of lift and confidence.
So we used this link:
https://datascienceplus.com/a-gentle-introduction-on-market-basket-analysis%E2%80%8A-%E2%80%8Aassociation-rules/
as reference to come up with an different visualization.

The color represents the lift value and size represents the support.

Based on the graph : 1. The rules connecting home_comfort and bed_bath_table has higher lift 2. The rules connecting bed bath table and furniture_decor have low lift and higher support.

```
plot(top20subRules, method = "graph")
```



2. Based on their shopping behaviors and location information, what are some distinct and meaningful customer groups?

We performed clustering to understand customer behavior and location. We decided to cluster on three four features:
1) Average Price per order for each customer
2) Average number of items in each order
3) Popularity of product category
4) Region the customer lives including: North, Northeast, Midwest, Southeast, South. The following website provides a map of the states in each region. https://www.researchgate.net/figure/Map-of-Brazil-showing-the-states-belonging-to-each-region-Acronyms-for-each-state_fig1_341914696

The following code creates a data frame for 1) average price per order and 2) average number of items per order for each customer.

```
order_level_features <- category_association %>%
  group_by(order_id) %>%
  dplyr::summarise(total_value = sum(price),
                   items = n())

order_all = inner_join(orders, order_level_features, by = "order_id")

cust_data_agg <- order_all %>%
  group_by(customer_id) %>%
  dplyr::summarise(avg_order_price = mean(total_value),
                   avg_items_per_order = mean(items))
```

The following code creates a data frame defining the region per customer. We created dummy columns for the regions in order to easily perform K-means clustering on a categorical variable.

We downloaded data from the following website to create a data frame of the regions by state.
https://brazil-help.com/brazilian_states.htm

```
state_region <- read.csv('state_region.csv')
cust_states <- customers %>% select('customer_id', 'customer_state')
join_state_region <- left_join(cust_states, state_region,
                               by = c('customer_state' = 'State'))

# make regions dummy columns
cust_regions_dummy <- dummy_cols(join_state_region, select_columns = 'Region')

# filter out unnecessary columns to get final customer regions
cust_regions <- cust_regions_dummy %>% select(-c('customer_state', 'Region'))
```

We then joined the region data frame with the average price and average items per order data frame. This data frame has three of our features needed for clustering.

```
### Joining all region and order_level_features
cust_data_final = inner_join(cust_data_agg,cust_regions,by='customer_id')
```

The following code creates a popularity index for each product category. Popularity index is a measure of a customers' inclination towards buying a popular category.

Here, we define popularity as the fraction of the total transactions containing a category among all orders. This is also known as the support of every category. In other words, the more frequent a category shows up on customers' orders, the more "popular" this category is.

Considering the shopping behavior for every customer's order, we calculate the average popularity of all items as this customer's popularity index.

This feature will help in the supply chain management decision. For instance,

- For regions where people have higher popularity index, Olist can stock more popular categories than those lower popularity index areas.
- For lower popularity index regions, Olist can recommend more niches for customers in this area.

```
### Adding pop_index

# 1. collect correct data sets
cluster_customer <- left_join(orders,
                              category_association,
                              by=c("order_id"="order_id"))
cluster_customer <- left_join(customers,
                              cluster_customer,
                              by=c("customer_id"="customer_id"))

# 2. add a column to calculate category freq
category_freq_withna <- group_by(cluster_customer,order_id)

# drop the NA values in product categories
category_freq_original <- subset(category_freq_withna,product_category_name_english!="NA")
f <- table(category_freq_original$product_category_name_english)
category_freq <- left_join(category_freq_original,as.data.frame(f),
                           by=c("product_category_name_english"="Var1"))

# Calculate the frequency of each product
# 111023 is the total number of transactions
category_freq$freq_pcg <- category_freq$Freq/111023
cluster<-category_freq


# 3. add a column of popularity index of each customer
# order_id = 00143d0f86d6fbd9f9b38ab440ac16f5
popular_index <- aggregate(cluster$freq_pcg,
                           by=list(type=cluster$customer_id),
                           mean)
cluster_final <- left_join(cluster,popular_index,
                           by=c('customer_id'='type'))

names(popular_index) <- c('customer_id','pop_index')
```

The following code joins all four of our features (average price per order, average number of items per order, region, and popularity index) into one data frame.

```
### Joining popindex with other data sets
cust_data_final_v = inner_join(cust_data_final,popular_index,
```

```
                                  by='customer_id')
cust_data_final_v$pop_index = as.numeric(cust_data_final_v$pop_index)
```

We will now perform clustering on our four features to group customers into clusters.

First, we will normalize our data

```
normalize = function(x){
  return ((x - min(x))/(max(x) - min(x)))}

data_normalized <- cust_data_final_v %>%
  mutate_at(c(2,3,9), normalize)

data_ <- data_normalized %>% select('avg_order_price', 'avg_items_per_order', 'pop_index')
```

Next, we will perform K-means clustering on our data. We chose to create four clusters because of our SSE curve, showed in the next line of code.

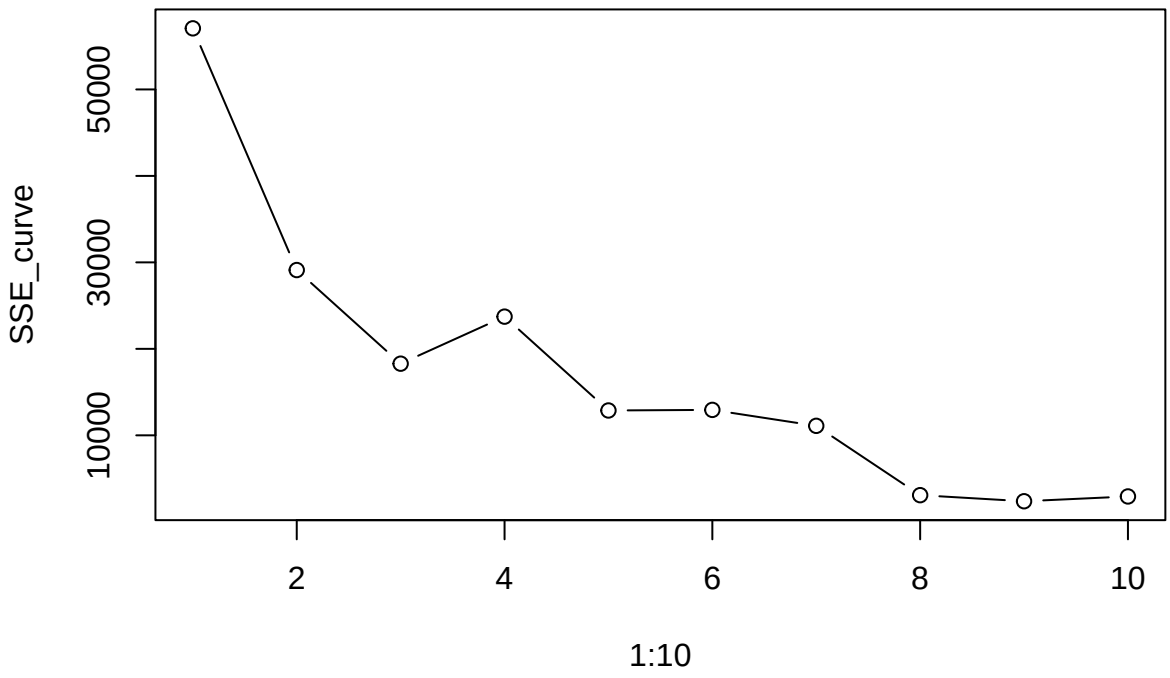
```
kcluster_4 = kmeans(data_normalized[,2:9], centers = 4)

kcluster_4$centers
```

```
##   avg_order_price avg_items_per_order Region_Center West Region_North
## 1      0.00996476         0.005825826           0.000000   0.00000000
## 2      0.01124419         0.006932105           0.185031   0.05917451
## 3      0.01077590         0.008231539           0.000000   0.00000000
## 4      0.00877516         0.008551429           0.000000   0.00000000
##   Region_Northeast Region_South Region_Southeast pop_index
## 1        0.0000000    0.0000000                1 0.2426949
## 2        0.3022981    0.4534964                0 0.5101597
## 3        0.0000000    0.0000000                1 0.6277458
## 4        0.0000000    0.0000000                1 0.8660954
```

Evaluating our cluster solutions:
Below we can see a plot of the sum of squared errors values for each number of clusters, 1-10. We can see that the sum of squared errors is relatively low at 4 clusters. We want a low SSE because this means the clusters are more cohesive. Therefore, we believe that choosing 4 clusters is the best choice.

```
SSE_curve <- c()
for (n in 1:10) {
kcluster = kmeans(data_normalized[,2:9], n)
sse = kcluster$tot.withinss
SSE_curve[n] = sse}
# plot SSE against number of clusters
plot(1:10, SSE_curve, type = "b")
```



Silhouette Value:
We tried running the silhouette value using the following code, but our computer crashed. So, we decided to only use the SSE value in determining the number of clusters to use.
We used this site to understand how to find a silhouette value after clustering using k-means.
https://uc-r.github.io/kmeans_clustering

```
# function to compute average silhouette for k clusters
# avg_sil <- function(k) {
#   km.res <- kmeans(data_normalized[2:9], centers = k, nstart = 25)
#   ss <- silhouette(km.res$cluster, dist(data_normalized[2:9]))
#   mean(ss[, 3])
# }
#
# # Compute and plot within-sum-of-squares for k = 2 to k = 10
# k.values <- 2:10
#
# # extract avg silhouette for 2-15 clusters
# avg_sil_values <- map_dbl(k.values, avg_sil)
#
# plot(k.values, avg_sil_values,
#        type = "b", pch = 19, frame = FALSE,
#        xlab = "Number of clusters K",
#        ylab = "Average Silhouettes")
```

Understanding Our Clusters
We added a column labeled clusters to our non-normalized data frame. We also added a column to name

the clusters based on the customer behaviors.

- Cluster 1 is named 'Niche' because the consumers belonging to this group generally tend to buy products that are not that popular i.e. products having a lower popularity index. The average item value is higher for this group.
- Cluster 2 is named 'Consumables' because Consumers tend to buy products that are cheap and highly rated on the popularity index.
- Cluster 3 is named 'Neutral' because consumers in this group are unresponsive to price changes and the popularity index. They are middle of the road customers.
- Cluster 4 is named 'Luxury' because consumers in this group tend to buy more expensive products and are unresponsive to the popularity index.

```
# create column for cluster 1-4
data_clusters <- cust_data_final_v %>%
  as_tibble() %>%
  mutate(cluster = kcluster_4$cluster)

# create new column labeling each cluster with new name
data_clusters <- data_clusters %>%
  mutate(cluster_name = case_when(cluster == 1 ~ 'Niche',
                                  cluster == 2 ~ 'Consumables',
                                  cluster == 3 ~ 'Neutral',
                                  cluster == 4 ~ 'Luxury'))
```

Count the number of customers in each cluster. We will use this in our recommendations because it provides valuable insight into which clusters to target and how to target them.

```
table(data_clusters$cluster_name)
```

```
##
## Consumables      Luxury     Neutral       Niche
##       30503       22429       12689       31635
```

Create data frame for only numeric features based on cluster

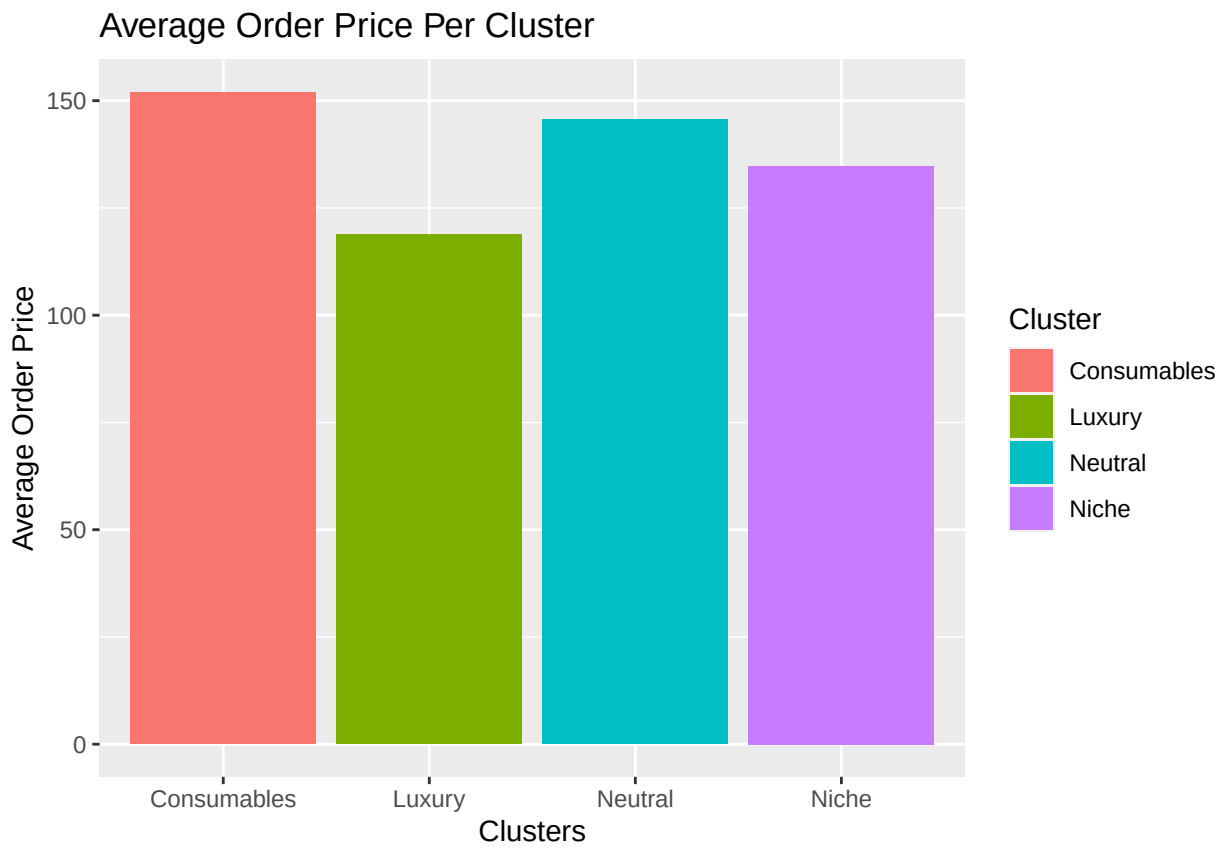
```
num_clusters <- data_clusters %>% select('avg_order_price',
                                         'avg_items_per_order',
                                         'pop_index', 'cluster_name')
```

Group by clusters and find average of price, number of items per order, and popularity index for each cluster. This will give us an idea of the center of the price, number of items per order, and popularity index for each cluster of customers.

```
clusters_grouped <- num_clusters %>%
  group_by(cluster_name) %>%
  dplyr::summarise(avg_order_price_cluster = mean(avg_order_price),
                   avg_items_per_order_cluster = mean(avg_items_per_order),
                   avg_pop_index_cluster = mean(pop_index))
```

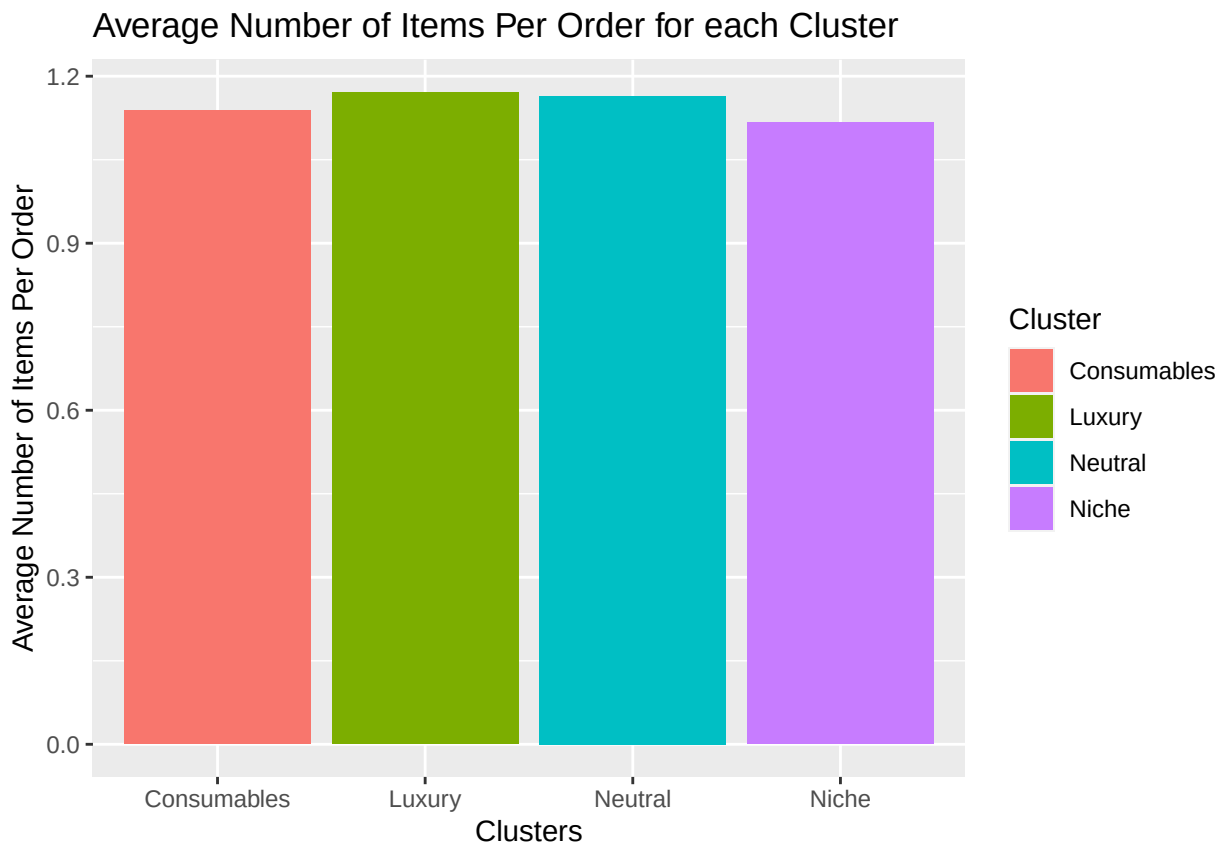
Average price for each cluster. This can also be described as the center price for each cluster.

```
ggplot(clusters_grouped, aes(x=cluster_name,
                             y = avg_order_price_cluster,
                             fill=as.factor(cluster_name))) +
  geom_bar(stat = 'identity') +
  labs(title = 'Average Order Price Per Cluster',
       x = 'Clusters',
       y = "Average Order Price")+
  guides(fill=guide_legend(title="Cluster"))
```



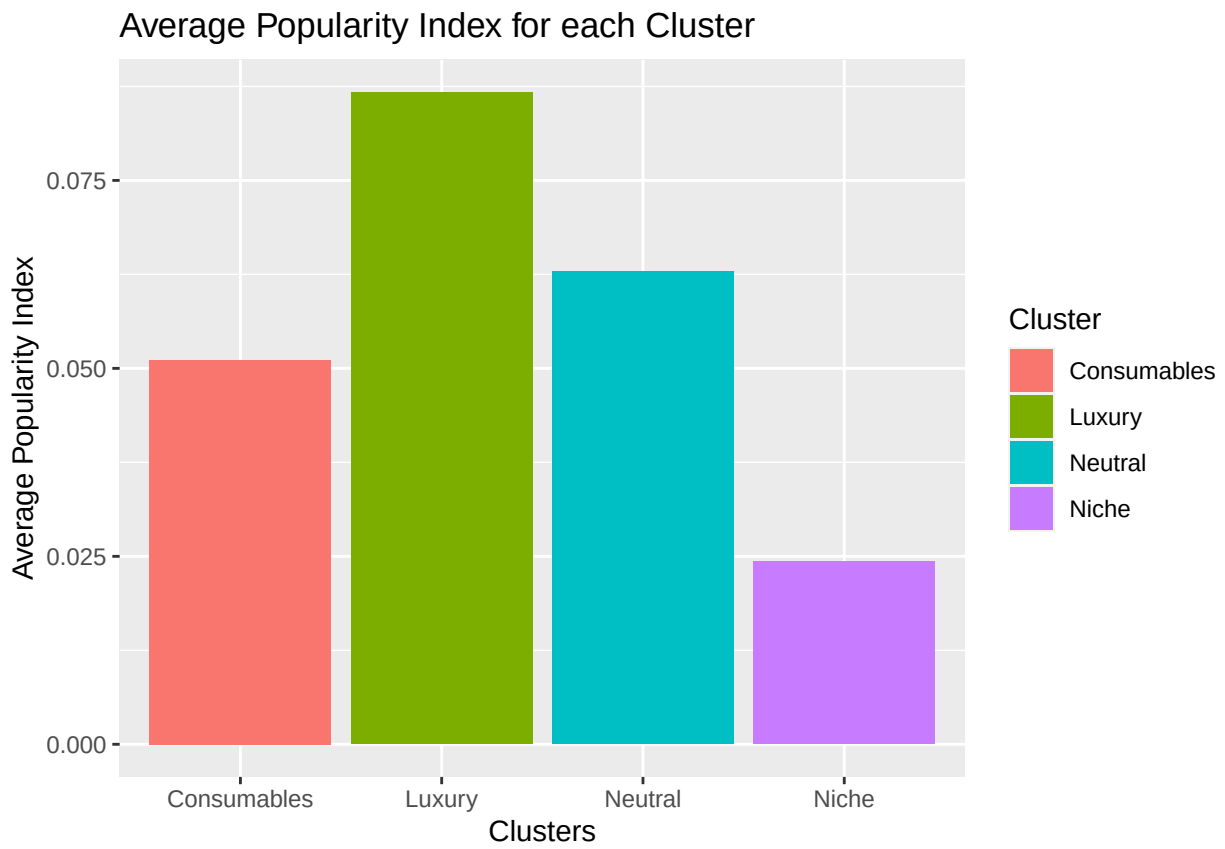
Average number of items per order in each cluster.

```
ggplot(clusters_grouped, aes(x=cluster_name,
                             y = avg_items_per_order_cluster,
                             fill=as.factor(cluster_name))) +
  geom_bar(stat = 'identity') +
  labs(title = 'Average Number of Items Per Order for each Cluster',
       x = 'Clusters',
       y = "Average Number of Items Per Order")+
  guides(fill=guide_legend(title="Cluster"))
```



Average popularity index in each cluster.

```
ggplot(clusters_grouped, aes(x=cluster_name,
                             y = avg_pop_index_cluster,
                             fill=as.factor(cluster_name))) +
  geom_bar(stat = 'identity')+
  labs(title = 'Average Popularity Index for each Cluster',
       x = 'Clusters',
       y = "Average Popularity Index")+
  guides(fill=guide_legend(title="Cluster"))
```



Based on all of the information we found above, we will provide insights and reccomendations to Olist in our managerial document.