Nicehair Technical Report

BI Experts: Pawel Gach, Bu Bu Zhang, Aleksandra Kusz

2024-04-30

Introduction

In the rapidly evolving world of e-commerce, leveraging data for actionable insights has become essential for maintaining a competitive edge. Nicehair, a leading retailer in the beauty industry, possesses a vast amount of customer data detailing transactions and purchasing behaviors. However, the potential of this data has yet to be fully harnessed, resulting in missed opportunities for targeted marketing, precision recommendations, and ultimately, increased profitability.

Problem Statement

The core challenge lies in unlocking the value within Nicehair's data, particularly in identifying purchase patterns, product relationships, and cross-selling opportunities. A data-driven solution can address this challenge with:

1. **Customer Segmentation:** Grouping customers based on purchasing behaviors, allowing for tailored marketing strategies and product recommendations.
2. **Product Recommendations:** Applying rule-seeking models to uncover relationships between different products, thereby creating opportunities for upselling and cross-selling.

Data

To solve this problem, we utilize a comprehensive dataset comprising 344,056 transactions spanning from January 1, 2024, to March 21, 2024. This dataset includes details about each transaction together with Google Analytics 4 data, more specifically:

- **Transaction details:** transaction_date, user_id, transaction_id, transaction_line_number, product_id and product_name associated with each transaction.
- **Customer information:** Including demographic and tracking data like country, brand, source/medium, campaign ID, default_channel_group, city, browser, device, operating system and revenue.

GitHub Repository

For the full, working codebase, extracted rules, visualisations, as well as the un-knitted R markdown file, please refer to our **GitHub repository**.

Methodology

An overview of the analytical approaches used, including segmentation, predictive modeling, and association rules mining.

The external libraries used are as follows:

```r
library(tidyr) # For data manipulation
library(dplyr) # For data manipulation
library(forcats) # For factorization
library(cluster) # For k-means clustering
library(fastDummies) # For dummy encoding
library(arules) # For association rules mining
library(arulesViz) # For visualisation of ASM
```

Handling NA values

After initialising the dataset within R, we proceed with initial analysis to understand its structure and key statistics.

```r
# Function to calculate proportions of NAs and empty strings in a column
proportion_missing <- function(x) {
  na_prop <- mean(is.na(x))  # Proportion of NAs
  empty_prop <- mean(x == "")  # Proportion of empty strings
  return(c(na_prop, empty_prop))
}

# Applying this function to each column
missing_summary <- t(apply(data, 2, proportion_missing))

rownames(missing_summary) <- colnames(data)
colnames(missing_summary) <- c("NA_Proportion", "Empty_String_Proportion")

print(missing_summary)
```

```
##                         NA_Proportion Empty_String_Proportion
## X                           0.0000000               0.0000000
## transaction_date            0.0000000               0.0000000
## user_id                     0.0000000               0.0000000
## transaction_id              0.0000000               0.0000000
## transaction_line_number     0.0000000               0.0000000
## product_id                  0.0000000               0.0000000
## product_name                0.0000000               0.0000000
## country                     0.0000000               0.0000000
## brand                       0.0000000               0.0000000
## source_medium               0.0000000               0.1739407
## campaign_id                 0.0000000               0.1739407
## default_channel_group       0.0000000               0.1739407
## city                        0.0000000               0.1739407
## browser                     0.0000000               0.1739407
## device                      0.0000000               0.1739407
## operating_system            0.0000000               0.1739407
## revenue                     0.1739407                      NA
```

An immediately noticed problem is that a significant portion (~17%) of the data contains missing values in the Google Analytics columns related to customer information - revenue, source/medium and so on. Since it is impossible to find clusters without identyfing variables, we decided to delete the rows with missing data.

```
data <- data[!is.na(data$revenue),]
```

Furthermoe, the `source_medium` column contains information about the source and medium of each transaction, separated by a "/". We split this column into two:

```
data <- data %>%
  separate(col = source_medium, into = c("source", "medium"), sep = "/")
```

To handle remaining missing values in the `medium` column, we replace them with "(not set)":

```
data$medium[is.na(data$medium)] <- "(not set)"
```

Handling "(not set)" strings

Several columns contain entries labeled "(not set)" instead of actual values or NA. We assess the prevalence of these entries as follows:

```
for (i in 1:ncol(data)) {
  if(i == 1) {
    cat(sprintf(
      "%-3s %-25s %-15s %-5s\n",
      "ID", "Column Name", "Not Set Count", "Proportion"
    ))
  }
  if (is.character(data[, i])) {
    not_set_count <- sum(data[, i] == "(not set)")
    formatted_output <- sprintf(
      "%-3s %-25s %-15s %-5s",
      i,
      colnames(data)[i],
      not_set_count,
      round(not_set_count / nrow(data), 2)
    )
    cat(formatted_output, "\n")
  }
}
```

```
## ID  Column Name               Not Set Count   Proportion
## 2   transaction_date          0               0
## 3   user_id                   0               0
## 4   transaction_id            0               0
## 7   product_name              0               0
## 8   country                   0               0
## 9   brand                     0               0
## 10  source                    2691            0.01
## 11  medium                    2691            0.01
## 12  campaign_id               128304          0.45
## 13  default_channel_group     0               0
## 14   city                      5130            0.02
## 15   browser                   0               0
## 16   device                    252621          0.89
## 17   operating_system          0               0
```

Notably, campaign_id has 0.45% of entries as (not set), however this is not a problem as not every sale has to be tied to a campaign

However, we observe that the `device` column has a high incompleteness rate of 89% and contains redundant information, since the `operating_system` column contains similiar information. Therefore, we remove it from the dataset:

```r
data <- subset(data, select = -device)
```

Feature Engineering

To simplify the analysis, we convert the long alphanumeric identifiers from the `user_id` and `transaction_id` columns into numeric IDs, while retaining the original identifiers for later use.

```r
# Mapping user IDs
user_mapping <- data.frame(
  user_original = levels(factor(data$user_id)),
  user_numeric = seq_along(levels(factor(data$user_id))
))

# Mapping transaction IDs
transaction_mapping <- data.frame(
  transaction_original = levels(factor(data$transaction_id)),
  transaction_numeric = seq_along(levels(factor(data$transaction_id))
))

# Converting to numeric
data$user_id <- as.numeric(factor(data$user_id))
data$transaction_id <- as.numeric(factor(data$transaction_id))
```

After cleaning, the `country` column only contains a single variable (`DK`), making it redundant.

```r
data <- subset(data, select = -country)
```

To reduce repetition, we retain only rows with the highest `transaction_line_number` per transaction ID. This approach retains the useful information about the user's basket size and deletes redundant rows.

```r
data <- data %>%
  group_by(transaction_id) %>%
  filter(transaction_line_number == max(transaction_line_number)) %>%
  ungroup()
```

The Google Analytics columns contain categorical data that needs to be converted to factors to be used in analysis.

```r
cols <- c(
  "brand", "source", "medium", "campaign_id",
  "default_channel_group", "city", "browser", "operating_system"
)

for (col in cols) {
  data[[col]] <- factor(data[[col]])
}
```

To reduce dimensionality before dummy encoding, we select the top 5 factor levels from each columns and lump the rest.

```r
data_lump <- data %>%
  mutate_if(is.factor, ~ fct_lump(., n = 5))

data_dummies <- dummy_cols(data_lump, select_columns = cols)

# Removing the original factors
data_dummies <- data_dummies %>% select_if(~ !is.factor(.))
```

Finally, we remove columns irrelevant to the upcoming analysis. The date is irrelevant due to the low collection period of the data, while product columns range in the thousands, which makes lumping them a large loss of information.

```r
data_dummies <- subset(data_dummies, select = -c(transaction_date, product_id, product_name))
```

Data Reduction

To prepare the dataset for customer segmentation, we first aggregate and summarize unique information for each user.

```r
data_users <- data_dummies %>%
  group_by(user_id) %>%
  summarise(
    revenue = sum(revenue),
    num_transactions = length(transaction_id),
    total_products = sum(transaction_line_number),
    across(5:52, ~ as.integer(any(.x == 1)), .names = "{.col}_flag")
  )
```

This step consolidates the dataset to a unique record for each user, including:

- **Revenue:** The total revenue generated by each user.
- **Number of Transactions:** The count of transactions per user.
- **Total Products:** The total number of products bought by each user.
- **Dummy Columns:** Flags indicating the presence of specific factors.

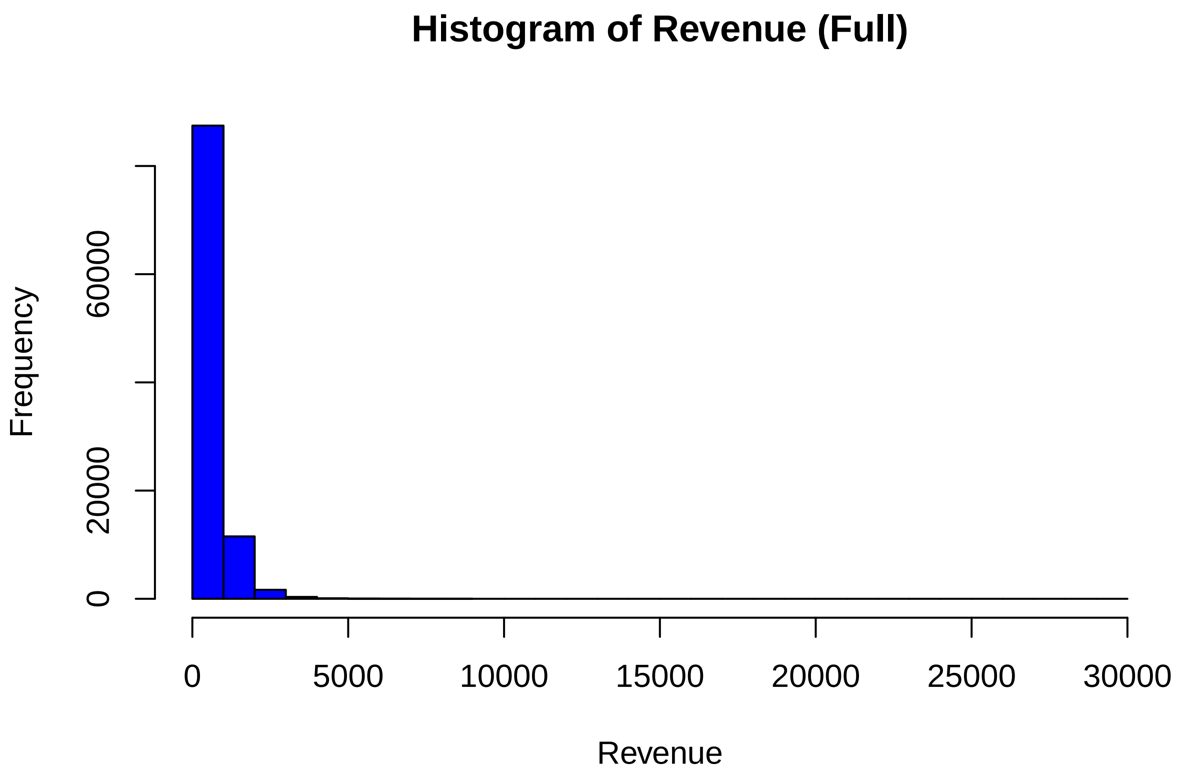
As user_id is now a unique identifier, we proceed without it

```
data_ready <- data_users[, -1]
```

Distribution of Revenue

To understand the distribution of revenue per user, we create a histogram:

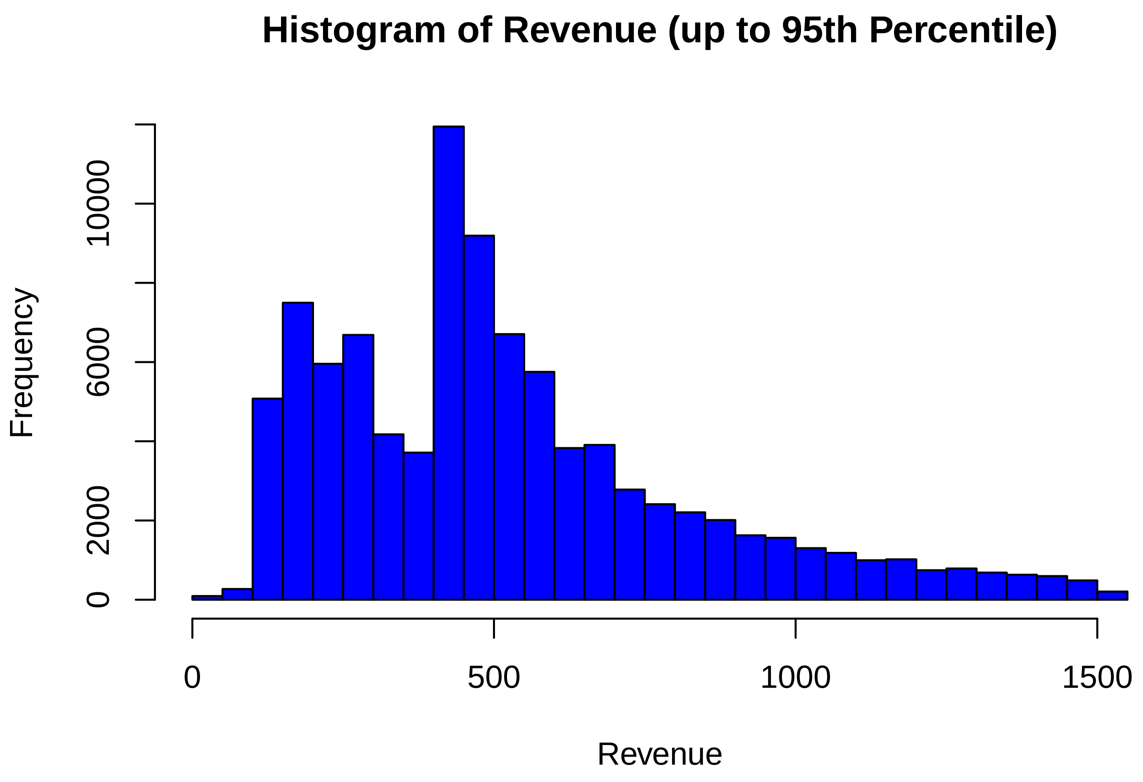
```
hist(
  data_ready$revenue,
  main = "Histogram of Revenue (Full)",
  xlab = "Revenue",
  col = "blue",
  breaks = 30
)
```



This reveals the skewness of the revenue distribution, indicating that most users contribute lower amounts with few extreme spenders.

We address the skewness by limiting the revenue distribution to the 95th percentile:

```
limits <- quantile(data_ready$revenue, probs = c(0, 0.95))
filtered_revenue <-
  data_ready$revenue[data_ready$revenue > limits[1] &
                     data_ready$revenue < limits[2]]
hist(
  filtered_revenue,
  main = "Histogram of Revenue (up to 95th Percentile)",
  xlab = "Revenue",
  col = "blue",
  breaks = 30
)
```



This provides a clearer picture of the revenue distribution by removing outliers. The immediate source of concern that can be observed from the histogram is that revenue is, overall, normally distributed. This may lead to it being very hard or impossible to cluster, since it may have no natural or reproducible clusters.

Principal Component Analysis

To prepare the dataset for further analysis, the only numeric variable, `revenue`, is scaled. This will let us observe its variance distribution on the same level as the dummy-encoded variables later down the line.

```
data_ready$revenue <- scale(data_ready$revenue)
```

We then reduce the dataset's dimensionality using Principal Component Analysis.

```
data_pca <- prcomp(data_ready, scale. = T)
```

This provides a calculation of the variance explained by each principal component. To limit the dimensionality of the data, we retain components that contribute to 95% of the variance - 29 in total.

```r
cumulative_variance <- summary(data_pca)$importance[3, ]
num_components_to_keep <- max(which(cumulative_variance <= 0.95))
data_reduced <- data_pca$x[, 1:num_components_to_keep]
```

Customer Segmentation

Since the dataset contains both scaled numeric data and dummy-encoded variables, as well as due to the fact that the dataset's dimensionality exceeds the limits of R's vectors, we decide to proceed with non-hierarchical k-means clustering using the `cluster` library

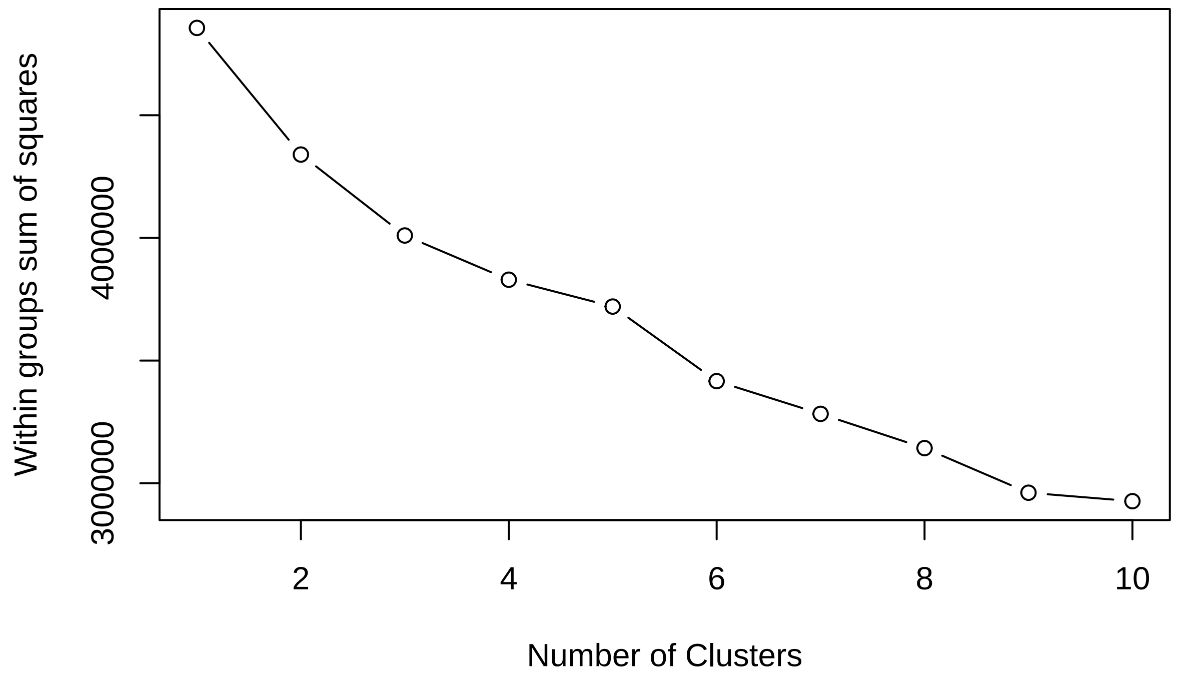
Finding the Optimal Number of Clusters

To determine the appropriate number of clusters for customer segmentation, we apply the k-means algorithm and examine the within-cluster sum of squares (WSS) for different values of k:

```r
set.seed(123) # For reproducibility

# WSS values for 1-10 clusters
wss <- sapply(1:10, function(k) {
  kmeans(data_reduced, k, nstart = 10)$tot.withinss
})

# Plotting WSS
plot(1:10,
     wss,
     type = "b",
     xlab = "Number of Clusters",
     ylab = "Within groups sum of squares")
```



The resultant elbow plot is a bit vague due to the nature of the dataset, but we deduce that it shows a decrease in WSS until 4 clusters, suggesting that 4 is an appropriate number for k.

K-means Clustering

We perform k-means clustering with 4 clusters and 25 random starts. We then attach the resultant cluster vector to the pre-PCA dataset.

```
set.seed(123)
km_clust <- kmeans(data_reduced, centers = 4, nstart = 25)
data_ready$clust <- as.factor(km_clust$cluster)
```

We can observe the distribution of data across the clusters before delving deeper into the analysis.

```
table(data_ready$clust)
```

```
##
##     1     2     3     4
## 21658 54003 15850  9657
```

The clusters seem to be slightly imbalanced in terms of size, but that is not necessarily a problem in of itself.

Inter-cluster Variance

To identify variables with the highest variance between clusters, we calculate and sort their variance:

```
cluster_means <- data_ready %>%
  group_by(clust) %>%
  summarise(across(where(is.numeric), mean, na.rm = TRUE), .groups = 'drop')

inter_cluster_variance <- cluster_means %>%
  summarise(across(where(is.numeric), var, na.rm = TRUE)) %>%
  pivot_longer(cols = everything(), names_to = "variable", values_to = "variance")

high_inter_cluster_variance <- inter_cluster_variance %>%
  arrange(desc(variance))
```

We then view the variables with the highest inter-cluster variance.

```
print(high_inter_cluster_variance, n = 20)
```

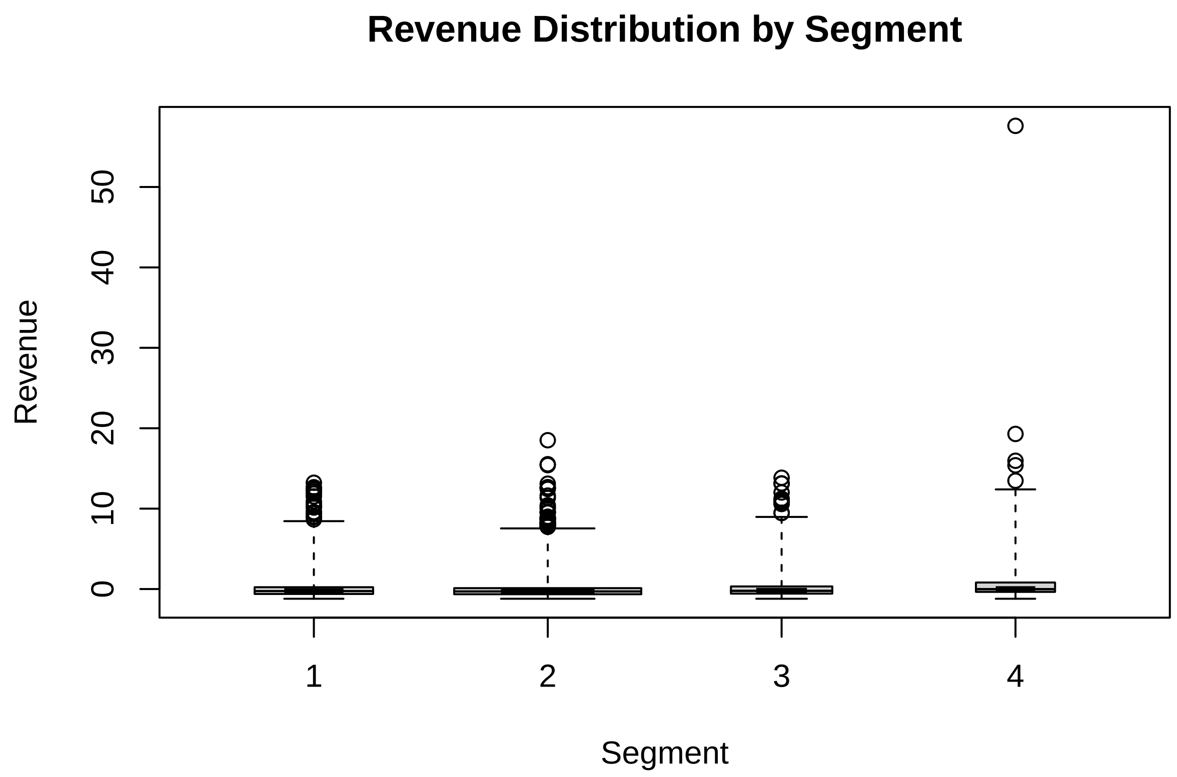
```
## # A tibble: 51 x 2
##    variable                                 variance
##    <chr>                                       <dbl>
##  1 total_products                             0.258
##  2 medium_ email_flag                         0.250
##  3 default_channel_group_Email_flag           0.249
##  4 source_(direct) _flag                      0.245
##  5 medium_ (none)_flag                        0.245
##  6 default_channel_group_Direct_flag          0.244
##  7 campaign_id_(not set)_flag                 0.240
##  8 source_Other_flag                          0.216
##  9 medium_ cpc_flag                           0.191
## 10 source_google _flag                        0.174
## 11 default_channel_group_Cross-network_flag   0.107
## 12 default_channel_group_Other_flag           0.0806
## 13 campaign_id_20537313264_flag               0.0519
## 14 revenue                                    0.0483
## 15 default_channel_group_Unassigned_flag      0.0422
## 16 medium_ organic_flag                       0.0257
## 17 medium_Other_flag                          0.0254
## 18 default_channel_group_Paid Search_flag     0.0175
## 19 num_transactions                           0.0146
## 20 medium_ partner_flag                       0.0120
## # i 31 more rows
```

The products differ quite significantly in multiple of the dummy-encoded columns. However, the top products column, despite being the highest in terms of variance, is still quite small when it's units are taken into account.

Revenue Distribution by Segment

The most important variable in customer segmentation is revenue, as it allows the company to manage their marketing efforts most effectively. To compare revenue across clusters, we create a boxplot.

```r
boxplot(
  data_ready$revenue ~ data_ready$clust,
  range = 10,
  varwidth = TRUE,
  notch = TRUE,
  main = "Revenue Distribution by Segment",
  xlab = "Segment",
  ylab = "Revenue"
)
```



Unfortunately this reveals that revenue distribution shows insignificant differences between clusters.

ANOVA for Validity

To ensure variables are valid predictors, we perform an ANOVA test on all the variables. The output can be seen in Appendix 2, but overall over 90% of the variables are significant, which supports their relevance in clustering.

```r
lapply(data_ready[, 1:51], function(x) summary(aov(x ~ clust, data = data_ready)))
```

Variable Distribution

To explore the distribution of top variables aside from numeric ones, we print their proportions. The full output is available in Appendix 3, but generally speaking, the clusters differ in columns related to source, medium and default channel.

```r
for (i in 1:10) {
  var_name <- high_inter_cluster_variance$variable[i]

  # Check if the column is numeric
  if (!is.numeric(data_ready[[var_name]])) {
    cat(var_name, "\n")

    count_table <- table(data_ready[[var_name]], data_ready$clust)
    proportion_table <- prop.table(count_table, margin = 2)
    print(proportion_table)

    cat("--------------------------------------\n\n")
  }
}
```

Product Recommendation

To enhance product recommendations and cross-selling opportunities, we explore associations between products based on customer transactions.

First, we reload and clean the dataset once more to start fresh. The only necessary cleaning step for product recommendation is simplying transaction IDs.

```
data$transaction_id <- as.numeric(factor(data$transaction_id))
```

Second, we create a list of transactions, grouping products by `transaction_id`:

```
trans_list <- data %>%
  group_by(transaction_id) %>%
  summarise(items = list(product_name), .groups = 'drop') %>%
  pull(items)

trans_list <- lapply(trans_list, unlist)
transactions <- as(trans_list, "transactions")
```

Association Rules Mining

We use the Apriori algorithm to mine association rules from the transaction data. It should be noted that the support is set to a very small amount due to the very high amount of products in the dataset - the products are simply spread to thin, so when support is set to a higher amount, no rules will be generated. Also notice that the threshold for confidence is set to a high percentage of 0.8, guaranteeing the algorithm will only consider the best rules under that metric.

```
rules <- apriori(transactions,
          parameter = list(
            supp = 0.00001, # Low support due to many products
            conf = 0.8,
            target = "rules"
          ))

summary(rules)
```

```
## set of 12988 rules
##
## rule length distribution (lhs + rhs):sizes
##    2    3    4    5    6    7    8
##  267 5427 3983 2144  935  216   16
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   3.000   4.000   3.905   5.000   8.000
##
## summary of quality measures:
##     support            confidence        coverage              lift
##  Min.   :1.442e-05   Min.   :0.8000   Min.   :1.442e-05   Min.   :   19.94
##  1st Qu.:1.442e-05   1st Qu.:1.0000   1st Qu.:1.442e-05   1st Qu.:  208.95
##  Median :1.442e-05   Median :1.0000   Median :1.442e-05   Median :  636.44
##  Mean   :1.819e-05   Mean   :0.9916   Mean   :1.876e-05   Mean   : 2585.12
##  3rd Qu.:1.442e-05   3rd Qu.:1.0000   3rd Qu.:1.442e-05   3rd Qu.: 2167.88
##  Max.   :2.314e-03   Max.   :1.0000   Max.   :2.753e-03   Max.   :69372.00
##      count
##  Min.   :  2.000
##  1st Qu.:  2.000
##  Median :  2.000
##  Mean   :  2.524
##  3rd Qu.:  2.000
##  Max.   :321.000
##
## mining info:
##          data ntransactions support confidence
##  transactions        138744   1e-05        0.8
##                                                                                   call
##  apriori(data = transactions, parameter = list(supp = 1e-05, conf = 0.8, target = "rules"))
```

Even with the high confidence threshold, the algorithm discovers almost 13 thousand rules. To analyze them, we sort them by lift and inspect the best ones.

```r
top_rules <- head(sort(rules, by = "lift"), 1000)

for (i in 1:10) {
  lhs <- labels(lhs(top_rules[i]))
  rhs <- labels(rhs(top_rules[i]))
  con <- quality(top_rules[i])$confidence
  lif <- quality(top_rules[i])$lift

  cat("Rule", i, ":\n")
  cat("LHS:", lhs, "\n")
  cat("RHS:", rhs, "\n")
  cat("Confidence:", con, "\n")
  cat("Lift:", lif, "\n")
  cat("--------------------------------------\n")
}
```

```
## Rule 1 :
## LHS: {Wella Elements Instant Detangling Conditioner 200 ml}
## RHS: {Wella Elements Mild Shampoo 250 ml}
## Confidence: 1
## Lift: 69372
## --------------------------------------
## Rule 2 :
## LHS: {Wella Elements Mild Shampoo 250 ml}
## RHS: {Wella Elements Instant Detangling Conditioner 200 ml}
## Confidence: 1
## Lift: 69372
## --------------------------------------
## Rule 3 :
## LHS: {Goldwell Dualsenses Color Revive Color Giving Shampoo 250 ml - Cool Red}
## RHS: {Goldwell Dualsenses Color Revive Color Giving Conditioner 200 ml - Cool Red }
## Confidence: 1
## Lift: 69372
## --------------------------------------
## Rule 4 :
```

```
## LHS: {Goldwell Dualsenses Color Revive Color Giving Conditioner 200 ml - Cool Red }
## RHS: {Goldwell Dualsenses Color Revive Color Giving Shampoo 250 ml - Cool Red}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
## Rule 5 :
## LHS: {amika: Mirrorball Conditioner 500 ml}
## RHS: {amika: Mirrorball Shampoo 500 ml}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
## Rule 6 :
## LHS: {amika: Mirrorball Shampoo 500 ml}
## RHS: {amika: Mirrorball Conditioner 500 ml}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
## Rule 7 :
## LHS: {Dolce & Gabbana Devotion EDP 50 ml}
## RHS: {Dolce & Gabbana Devotion Tote Bag (GWP)}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
## Rule 8 :
## LHS: {Dolce & Gabbana Devotion Tote Bag (GWP)}
## RHS: {Dolce & Gabbana Devotion EDP 50 ml}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
## Rule 9 :
## LHS: {By Stær THIT Hairtie - Yellow}
## RHS: {By Stær BRAIDED Hairtie Slim - Light Yellow Glitter}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
## Rule 10 :
## LHS: {By Stær BRAIDED Hairtie Slim - Light Yellow Glitter}
## RHS: {By Stær THIT Hairtie - Yellow}
## Confidence: 1
## Lift: 69372
## ----------------------------------------
```

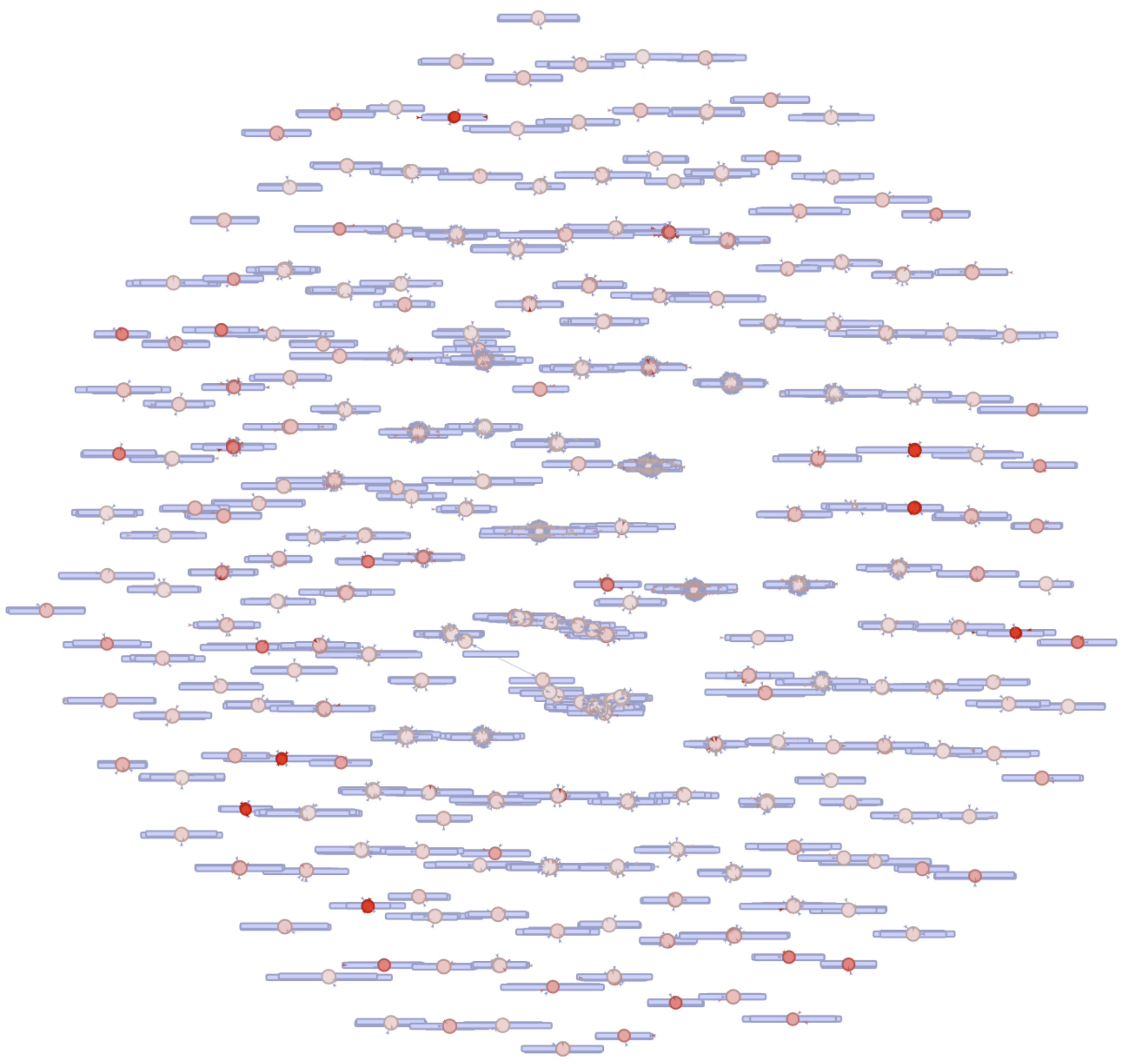
The full list of all discovered rules in a CSV format, together with a visualisation of the top rules from the `arulesViz` library in HTML format is available as part of the GitHub repository. Please note that the visualisation requires significant processing power to display properly.

```
# Code used to export the rules into csv

rules_df <- as(rules, "data.frame")
write.csv(rules_df, "apriori_product_rules.csv", row.names = FALSE)

# Code used to visualise the rules
plot(top_rules,
     method = "graph",
     engine = "htmlwidget",
     max = 1000)
```

For simplicity, we're also including a non-interactable visualisation of the top rules.



As can be seen from the visualization, the top 1000 rules are mostly isolated - they are usually based on pairs or trios of products, with larger clusters of more complicated relations being rarer. This, however, doesn't impact the usefulness of the rules.

Brand-specific Product Recommendation

Since the dataset is, unfortunately, not categorised in any way, the rules generated by association rules mining can only apply to individual products. To increase the real-life potential usability of the model, we also explore associations between brands listed in the dataset.

```r
data$brand <- factor(data$brand)

trans_list_b <- data %>%
  group_by(transaction_id) %>%
  summarise(items = list(brand), .groups = 'drop') %>%
  pull(items)

trans_list_b <- lapply(trans_list_b, unlist)
transactions_b <- as(trans_list_b, "transactions")
```

```
## Warning in asMethod(object): removing duplicated items in transactions
```

We apply the Apriori algorithm once again with similiar parameters to discover brand-level associations.

```r
rules_b <- apriori(transactions_b,
          parameter = list(
            supp = 0.00001,
            conf = 0.8,
            target = "rules"
          ))
```

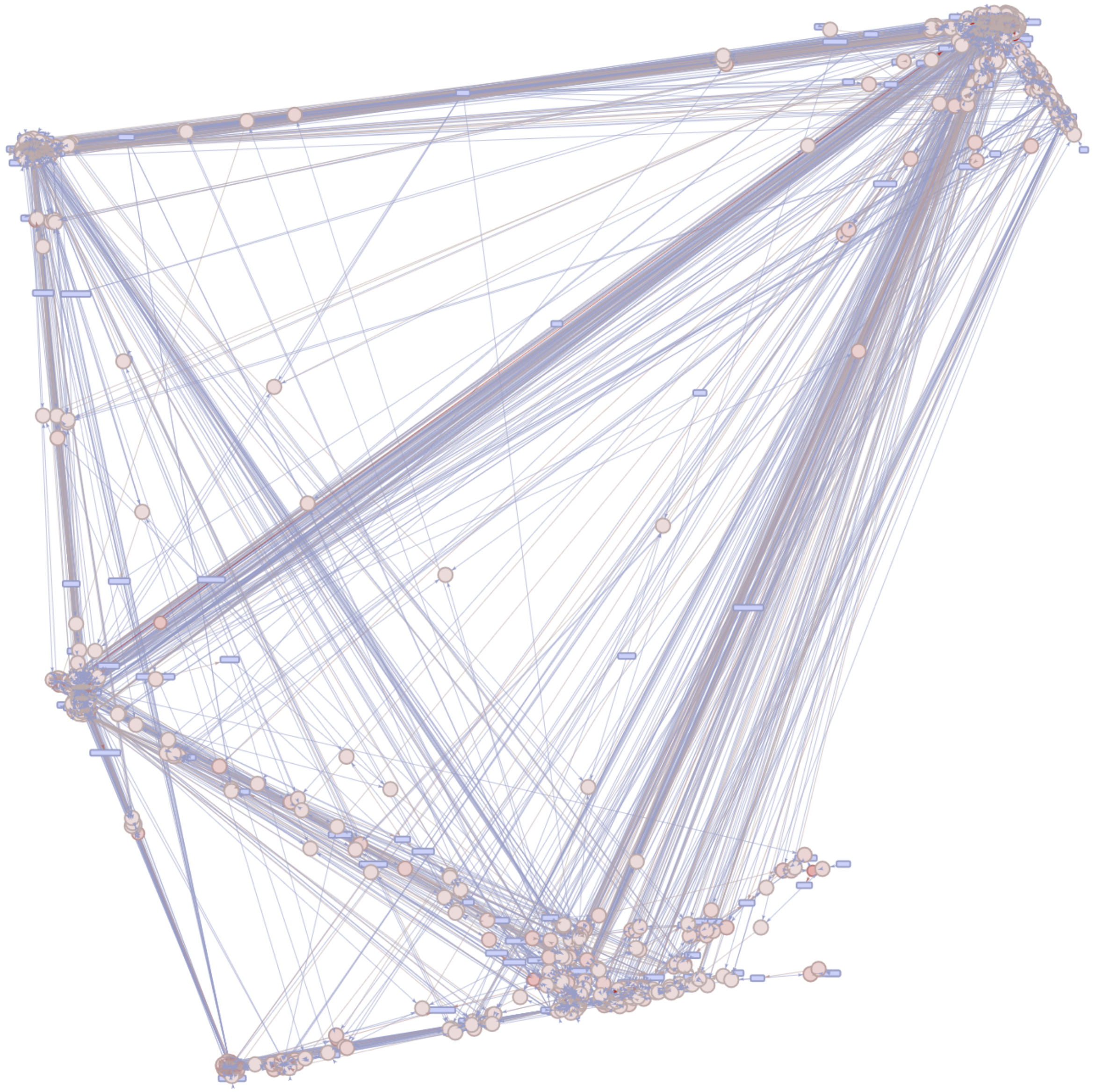
```r
top_rules_b <- head(sort(rules_b, by = "lift"), 1000)

for (i in 1:10) {
  lhs <- labels(lhs(top_rules_b[i]))
  rhs <- labels(rhs(top_rules_b[i]))
  con <- quality(top_rules_b[i])$confidence
  lif <- quality(top_rules_b[i])$lift

  cat("Rule", i, ":\n")
  cat("LHS:", lhs, "\n")
  cat("RHS:", rhs, "\n")
  cat("Confidence:", con, "\n")
  cat("Lift:", lif, "\n")
  cat("--------------------------------------\n")
}
```

```
## Rule 1 :
## LHS: {Brushworks,MAC Cosmetics,Maybelline,The Ordinary}
## RHS: {Beautyblender}
## Confidence: 1
## Lift: 2434.105
## --------------------------------------
## Rule 2 :
## LHS: {Comme Deux ,Redken}
## RHS: {Scandinavian Biolabs}
## Confidence: 1
## Lift: 2312.4
## --------------------------------------
## Rule 3 :
## LHS: {American Crew,ESEA,Makeup Revolution}
## RHS: {King C. Gillette}
## Confidence: 1
## Lift: 2202.286
## --------------------------------------
## Rule 4 :
## LHS: {Anua,CeraVe,Pyunkang Yul}
## RHS: {Tanrevel}
## Confidence: 1
## Lift: 2040.353
## --------------------------------------
## Rule 5 :
## LHS: {Anua,Lumene,Meraki}
## RHS: {Narciso Rodriguez}
## Confidence: 1
## Lift: 1185.846
## --------------------------------------
## Rule 6 :
## LHS: {b.tan,Gordon}
## RHS: {Tree Hut}
## Confidence: 1
## Lift: 937.4595
## --------------------------------------
## Rule 7 :
## LHS: {Jimmy Choo,Nilens Jord}
## RHS: {Murad}
## Confidence: 1
## Lift: 872.6038
## --------------------------------------
## Rule 8 :
## LHS: {L'Oreal Paris,Lancôme,Tweezerman}
## RHS: {Grande Cosmetics}
## Confidence: 1
## Lift: 825.8571
## --------------------------------------
## Rule 9 :
## LHS: {Batiste,Neccin,Pyunkang Yul}
## RHS: {L'Occitane}
## Confidence: 1
## Lift: 825.8571
## --------------------------------------
## Rule 10 :
## LHS: {Batiste,Beauty of Joseon,Neccin,Pyunkang Yul}
## RHS: {L'Occitane}
## Confidence: 1
## Lift: 825.8571
## --------------------------------------
```

The rules and visualisation for the brand-based application of apriori are also available on GitHub. Below is another non-interactable visualisation.



In stark contrast to the product rules, the brand rules form many more complicated relationships. This is likely due to the lower amount of unique brands included in the dataset. This also doesn't impact the usefulness of the discovered rules, but it is still a notable difference.

Conclusions

The analysis of Nicehair's data yields valuable insights into both customer segmentation and product recommendation strategies.

Customer Segmentation

The k-means clustering of the dataset reveals that, while there are differences between clusters for certain variables, revenue and total products purchased remain consistent across all clusters. This finding suggests that Nicehair's current marketing strategies attract customers with similar purchasing power and behavior patterns.

Business Implications

1. **Marketing Budget Reallocation:** The consistency in revenue and total purchases indicates that some high-cost marketing channels may not provide a significantly better return on investment. Therefore, Nicehair can consider reallocating the budget from more expensive channels to explore alternative strategies or to strengthen existing ones that yield similar results at a lower cost.
2. **Tailored Campaigns:** The variables that differentiate clusters (such as channels, sources, or customer characteristics) can be leveraged to design targeted marketing campaigns. This can help Nicehair refine its strategies to maximize effectiveness, even within its current customer base, possibly achieving a higher repurchase rate.

Product Recommendation

The association rules mining generates two extensive rule sets, which can be applied in various ways to enhance business operations:

1. **Bundling Products:** The discovered associations can be used to create product bundles, allowing Nicehair to offer complementary or related items together. This not only increases sales volume per transaction (and increases average basket size) but also provides added value to customers.
2. **Upselling Opportunities:** The rules can also be used to recommend related products when an item is added to a customer's basket. This can lead to increased average order value and contribute to revenue growth.

Next Steps

To further optimize the product recommendation model, Nicehair can consider:

1. Adding product categories and sub-categories can refine the association rules, making them more relevant and actionable.
2. By exploring additional metrics or even simple testing, Nicehair can refine the discovered rules, improving their accuracy and applicability.

Final Thoughts

The insights gained from this analysis pave the way for data-driven strategies that enhance both marketing effectiveness and product recommendation systems. By leveraging these findings, Nicehair can optimize its operations, enhance customer experience, and drive further growth.

Appendix 1: PCA Output

```
## Importance of components:
##                          PC1     PC2     PC3     PC4     PC5    PC6     PC7
## Standard deviation     2.369 1.89143 1.76949 1.70021 1.54247 1.4792 1.44013
## Proportion of Variance 0.110 0.07015 0.06139 0.05668 0.04665 0.0429 0.04067
## Cumulative Proportion  0.110 0.18016 0.24156 0.29824 0.34489 0.3878 0.42846
##                            PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     1.35008 1.32574 1.30848 1.27480 1.17592 1.08520 1.07667
## Proportion of Variance 0.03574 0.03446 0.03357 0.03187 0.02711 0.02309 0.02273
## Cumulative Proportion  0.46420 0.49866 0.53223 0.56409 0.59121 0.61430 0.63703
##                           PC15    PC16    PC17    PC18    PC19   PC20   PC21
## Standard deviation     1.06195 1.05947 1.03606 1.03032 1.02644 1.0226 1.0151
## Proportion of Variance 0.02211 0.02201 0.02105 0.02081 0.02066 0.0205 0.0202
## Cumulative Proportion  0.65914 0.68115 0.70220 0.72301 0.74367 0.7642 0.7844
##                           PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     1.01291 1.01141 1.00759 1.00686 1.00252 0.99892 0.98733
## Proportion of Variance 0.02012 0.02006 0.01991 0.01988 0.01971 0.01957 0.01911
## Cumulative Proportion  0.80450 0.82455 0.84446 0.86434 0.88404 0.90361 0.92272
##                           PC29    PC30    PC31    PC32    PC33    PC34    PC35
## Standard deviation     0.96924 0.75806 0.74672 0.67428 0.60131 0.44341 0.42006
## Proportion of Variance 0.01842 0.01127 0.01093 0.00891 0.00709 0.00386 0.00346
## Cumulative Proportion  0.94114 0.95241 0.96335 0.97226 0.97935 0.98320 0.98666
##                           PC36    PC37   PC38    PC39    PC40    PC41    PC42
## Standard deviation     0.35429 0.32423 0.2770 0.26465 0.26370 0.22727 0.21197
## Proportion of Variance 0.00246 0.00206 0.0015 0.00137 0.00136 0.00101 0.00088
## Cumulative Proportion  0.98913 0.99119 0.9927 0.99406 0.99543 0.99644 0.99732
##                           PC43    PC44    PC45    PC46    PC47    PC48    PC49
## Standard deviation     0.19357 0.16898 0.14590 0.12055 0.11052 0.09323 0.09144
## Proportion of Variance 0.00073 0.00056 0.00042 0.00028 0.00024 0.00017 0.00016
## Cumulative Proportion  0.99806 0.99862 0.99903 0.99932 0.99956 0.99973 0.99989
##                           PC50     PC51
## Standard deviation     0.07378 0.006052
## Proportion of Variance 0.00011 0.000000
## Cumulative Proportion  1.00000 1.000000
```

Appendix 2: ANOVA of Cluster Variables

```
## $revenue
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3   2247   749.1   766.1 <2e-16 ***
## Residuals   101164  98920     1.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $num_transactions
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    707  235.70    1257 <2e-16 ***
## Residuals   101164  18965    0.19
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $total_products
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3  12626    4209   726.3 <2e-16 ***
## Residuals   101164 586219       6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $brand_Biotherm_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3      5  1.7124   52.64 <2e-16 ***
## Residuals   101164   3291  0.0325
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $brand_Clinique_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    5.4  1.7847   62.05 <2e-16 ***
## Residuals   101164 2909.4  0.0288
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $brand_ESEA_flag
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## clust            3      4   1.331   24.21 1.18e-15 ***
## Residuals   101164   5562   0.055
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $brand_Kérastase_flag
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## clust            3      2  0.7453    15.8 2.86e-10 ***
## Residuals   101164   4771  0.0472
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $brand_Redken_flag
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## clust            3      2  0.6678    17.1 4.26e-11 ***
## Residuals   101164   3951  0.0391
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $brand_Other_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3      1  0.3064   2.092 0.0989 .
## Residuals   101164  14812  0.1464
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`source_(direct) _flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3  13272    4424 1394664 <2e-16 ***
## Residuals   101164    321       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`source_google _flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3  15322    5107   62493 <2e-16 ***
## Residuals   101164   8268       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`source_onskeskyen _flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    786  262.02    6923 <2e-16 ***
## Residuals   101164   3829    0.04
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`source_pa _flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    219   73.01    3218 <2e-16 ***
## Residuals   101164   2295    0.02
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`source_pricerunner _flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3  178.3   59.43    2847 <2e-16 ***
## Residuals   101164 2111.4    0.02
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $source_Other_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3   8389  2796.4   51232 <2e-16 ***
## Residuals   101164   5522     0.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`medium_ (none)_flag`
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3  13272    4424 1390306 <2e-16 ***
## Residuals    101164    322       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`medium_ cpc_flag`
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3  18516    6172  108828 <2e-16 ***
## Residuals    101164   5737       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`medium_ email_flag`
##                  Df Sum Sq Mean Sq  F value Pr(>F)
## clust             3   8728    2909 22647788 <2e-16 ***
## Residuals    101164     13       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`medium_ organic_flag`
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3   1793   597.6   11592 <2e-16 ***
## Residuals    101164   5216     0.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`medium_ partner_flag`
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3    838  279.30    7196 <2e-16 ***
## Residuals    101164   3927    0.04
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $medium_Other_flag
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3   1814   604.6   11422 <2e-16 ***
## Residuals    101164   5355     0.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`campaign_id_(not set)_flag`
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3  23882    7961  656668 <2e-16 ***
## Residuals    101164   1226       0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $campaign_id_20537313264_flag
##                  Df Sum Sq Mean Sq F value Pr(>F)
## clust             3   5190  1730.1   11271 <2e-16 ***
## Residuals    101164  15529     0.2
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## $campaign_id_20542877450_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3    182   60.58    1204 <2e-16 ***
## Residuals   101164   5090    0.05                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $campaign_id_20546864332_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3    288   95.84    1548 <2e-16 ***
## Residuals   101164   6265    0.06                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $campaign_id_770579953_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3    480  159.93    1994 <2e-16 ***
## Residuals   101164   8113    0.08                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $campaign_id_Other_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3    480  160.07    1790 <2e-16 ***
## Residuals   101164   9045    0.09                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`default_channel_group_Cross-network_flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3  10776    3592   26634 <2e-16 ***
## Residuals   101164  13644       0                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $default_channel_group_Direct_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3  13224    4408  944213 <2e-16 ***
## Residuals   101164    472       0                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $default_channel_group_Email_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3   8678    2893 1560549 <2e-16 ***
## Residuals   101164    188       0                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`default_channel_group_Paid Search_flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)    
## clust            3   1783   594.2    4691 <2e-16 ***
## Residuals   101164  12815     0.1                   
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $default_channel_group_Unassigned_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3   2975   991.6   17285 <2e-16 ***
## Residuals   101164   5803     0.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $default_channel_group_Other_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3   5615  1871.8   30244 <2e-16 ***
## Residuals   101164   6261     0.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $city_Aarhus_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3      0 0.15627   2.461 0.0607 .
## Residuals   101164   6425 0.06351
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $city_Copenhagen_flag
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## clust            3      4  1.2287   6.271 0.000299 ***
## Residuals   101164  19821  0.1959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $city_Esbjerg_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3      1  0.3340   15.77  3e-10 ***
## Residuals   101164   2142  0.0212
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $city_Odense_flag
##                 Df Sum Sq  Mean Sq F value Pr(>F)
## clust            3      0 0.004525   0.176  0.913
## Residuals   101164   2605 0.025752
##
## $city_Roskilde_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    0.1 0.02583   1.236  0.295
## Residuals   101164 2113.8 0.02090
##
## $city_Other_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3     30  10.122   43.35 <2e-16 ***
## Residuals   101164  23620   0.233
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## $browser_Chrome_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    266   88.69   485.6 <2e-16 ***
## Residuals   101164  18477    0.18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $browser_Edge_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3     18   6.150   124.6 <2e-16 ***
## Residuals   101164   4992   0.049
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $browser_Firefox_flag
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## clust            3    0.1  0.0415   5.611 0.000765 ***
## Residuals   101164  748.3  0.0074
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $browser_Safari_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    220   73.19   326.1 <2e-16 ***
## Residuals   101164  22703    0.22
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`browser_Samsung Internet_flag`
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3     48  15.916   365.9 <2e-16 ***
## Residuals   101164   4400   0.043
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $browser_Other_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3   26.6   8.853   805.1 <2e-16 ***
## Residuals   101164 1112.3   0.011
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $operating_system_Android_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3     11   3.795   33.09 <2e-16 ***
## Residuals   101164  11603   0.115
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $`operating_system_Chrome OS_flag`
##                 Df Sum Sq  Mean Sq F value   Pr(>F)
## clust            3   0.05 0.015377   7.795 3.36e-05 ***
## Residuals   101164 199.56 0.001973
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $operating_system_iOS_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3    132   44.11     182 <2e-16 ***
## Residuals   101164  24519    0.24
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $operating_system_Macintosh_flag
##                 Df Sum Sq Mean Sq F value   Pr(>F)
## clust            3      4  1.3101   10.05 1.28e-06 ***
## Residuals   101164  13184  0.1303
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $operating_system_Windows_flag
##                 Df Sum Sq Mean Sq F value Pr(>F)
## clust            3     84  27.861   221.1 <2e-16 ***
## Residuals   101164  12746   0.126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $operating_system_Other_flag
##                 Df Sum Sq  Mean Sq F value Pr(>F)
## clust            3    0.0 0.001244   0.824  0.481
## Residuals   101164  152.8 0.001510
```

Appendix 3: Cluster Variable Distribution

```
## total_products
##
##                 1            2            3            4
##    1  3.135100e-01 3.202970e-01 2.711672e-01 1.775914e-01
##    2  2.462831e-01 2.588560e-01 2.400631e-01 2.152842e-01
##    3  1.686675e-01 1.788604e-01 1.753943e-01 1.869110e-01
##    4  1.055961e-01 1.055312e-01 1.130599e-01 1.362742e-01
##    5  5.997784e-02 5.716349e-02 6.731861e-02 8.729419e-02
##    6  3.642996e-02 3.223895e-02 4.208202e-02 6.057782e-02
##    7  2.188568e-02 1.811010e-02 2.940063e-02 3.789997e-02
##    8  1.509835e-02 1.011055e-02 1.842271e-02 2.702703e-02
##    9  8.818912e-03 6.703331e-03 1.268139e-02 2.039971e-02
##    10 6.187090e-03 3.888673e-03 8.832808e-03 1.315108e-02
##    11 3.878474e-03 2.610966e-03 5.173502e-03 9.009009e-03
##    12 3.647613e-03 1.648057e-03 4.353312e-03 7.455732e-03
##    13 1.616031e-03 1.314742e-03 2.649842e-03 5.384695e-03
##    14 1.431342e-03 6.481121e-04 2.271293e-03 2.899451e-03
##    15 1.385169e-03 3.888673e-04 1.703470e-03 3.520762e-03
##    16 9.234463e-04 4.444198e-04 1.892744e-03 1.863933e-03
##    17 1.061963e-03 4.259023e-04 5.678233e-04 1.346174e-03
##    18 7.387570e-04 1.481399e-04 5.678233e-04 1.242622e-03
##    19 4.617232e-04 9.258745e-05 1.261830e-04 7.248628e-04
##    20 5.078955e-04 1.111049e-04 5.678233e-04 6.213110e-04
##    21 2.770339e-04 7.406996e-05 4.416404e-04 5.177591e-04
##    22 4.155508e-04 7.406996e-05 0.000000e+00 3.106555e-04
##    23 1.385169e-04 7.406996e-05 1.261830e-04 3.106555e-04
##    24 1.385169e-04 3.703498e-05 1.261830e-04 2.071037e-04
##    25 4.617232e-05 3.703498e-05 2.523659e-04 3.106555e-04
##    26 2.308616e-04 1.851749e-05 1.892744e-04 0.000000e+00
##    27 9.234463e-05 1.851749e-05 6.309148e-05 1.035518e-04
##    28 9.234463e-05 1.851749e-05 1.261830e-04 4.142073e-04
##    30 4.617232e-05 1.851749e-05 0.000000e+00 0.000000e+00
##    31 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    32 1.385169e-04 0.000000e+00 0.000000e+00 0.000000e+00
##    33 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    34 4.617232e-05 0.000000e+00 6.309148e-05 0.000000e+00
##    35 0.000000e+00 0.000000e+00 6.309148e-05 1.035518e-04
##    36 0.000000e+00 1.851749e-05 0.000000e+00 0.000000e+00
##    37 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    38 0.000000e+00 0.000000e+00 0.000000e+00 2.071037e-04
##    39 9.234463e-05 0.000000e+00 0.000000e+00 0.000000e+00
##    40 4.617232e-05 1.851749e-05 0.000000e+00 2.071037e-04
##    42 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    45 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    49 4.617232e-05 0.000000e+00 6.309148e-05 0.000000e+00
##    50 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    52 0.000000e+00 0.000000e+00 1.261830e-04 1.035518e-04
##    57 4.617232e-05 0.000000e+00 0.000000e+00 0.000000e+00
##    59 0.000000e+00 0.000000e+00 0.000000e+00 1.035518e-04
##    76 0.000000e+00 0.000000e+00 6.309148e-05 0.000000e+00
## -------------------------------------
##
```

```
## medium_ email_flag
##
##              1            2            3            4
##   0 9.995844e-01 9.999815e-01 1.000000e+00 3.106555e-04
##   1 4.155508e-04 1.851749e-05 0.000000e+00 9.996893e-01
## -------------------------------------
##
## default_channel_group_Email_flag
##
##              1            2            3            4
##   0 0.9992612430 0.9971483066 0.9996845426 0.0013461738
##   1 0.0007387570 0.0028516934 0.0003154574 0.9986538262
## -------------------------------------
##
## source_(direct) _flag
##
##             1           2           3           4
##   0 0.999122726 1.000000000 0.000000000 0.967691830
##   1 0.000877274 0.000000000 1.000000000 0.032308170
## -------------------------------------
##
## medium_ (none)_flag
##
##              1            2            3            4
##   0 0.9990765537 1.0000000000 0.0000000000 0.9676918298
##   1 0.0009234463 0.0000000000 1.0000000000 0.0323081702
## -------------------------------------
##
## default_channel_group_Direct_flag
##
##             1           2           3           4
##   0 0.998568658 0.997481621 0.000000000 0.967277622
##   1 0.001431342 0.002518379 1.000000000 0.032722378
## -------------------------------------
##
## campaign_id_(not set)_flag
##
##             1           2           3           4
##   0 0.049404377 0.996370572 0.000000000 0.001449726
##   1 0.950595623 0.003629428 1.000000000 0.998550274
## -------------------------------------
##
## source_Other_flag
##
##              1            2            3            4
##   0 0.7415273802 0.9790382016 0.9832176656 0.0001035518
##   1 0.2584726198 0.0209617984 0.0167823344 0.9998964482
## -------------------------------------
##
## medium_ cpc_flag
##
##              1            2            3            4
##   0 0.7877458676 0.0008518045 0.9217665615 0.8924096510
##   1 0.2122541324 0.9991481955 0.0782334385 0.1075903490
## -------------------------------------
##
## source_google _flag
##
##              1          2          3          4
##   0 0.61593868 0.01855452 0.91652997 0.88961375
##   1 0.38406132 0.98144548 0.08347003 0.11038625
## -------------------------------------
```