Group 4 – Project (Customer Segmentation)

```
customers_df<-read.csv("Wholesale Customers data.csv")

library(dplyr)
```

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

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

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

```
library(ggplot2)
library(writexl)
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 -- ## v forcats 1.0.0      v stringr
                         1.5.1
## v lubridate 1.9.3          v tibble          3.2.1
## v purrr           1.0.2          v tidyr           1.3.1
## v readr            2.1.5

## --· Conflicts ---------------------------------------------- tidyverse_conflicts() --·
## x dplyr::filter() masks stats::filter() ## x dplyr::lag()
                              masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

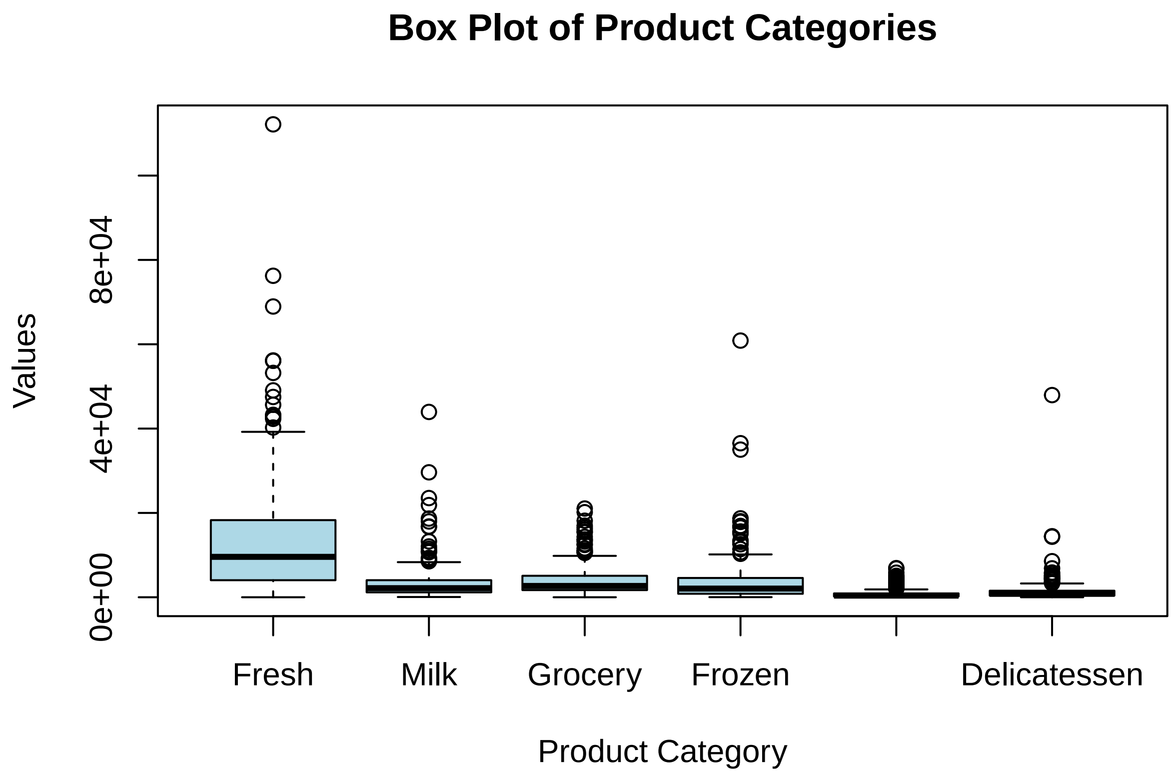
```
library(cluster)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

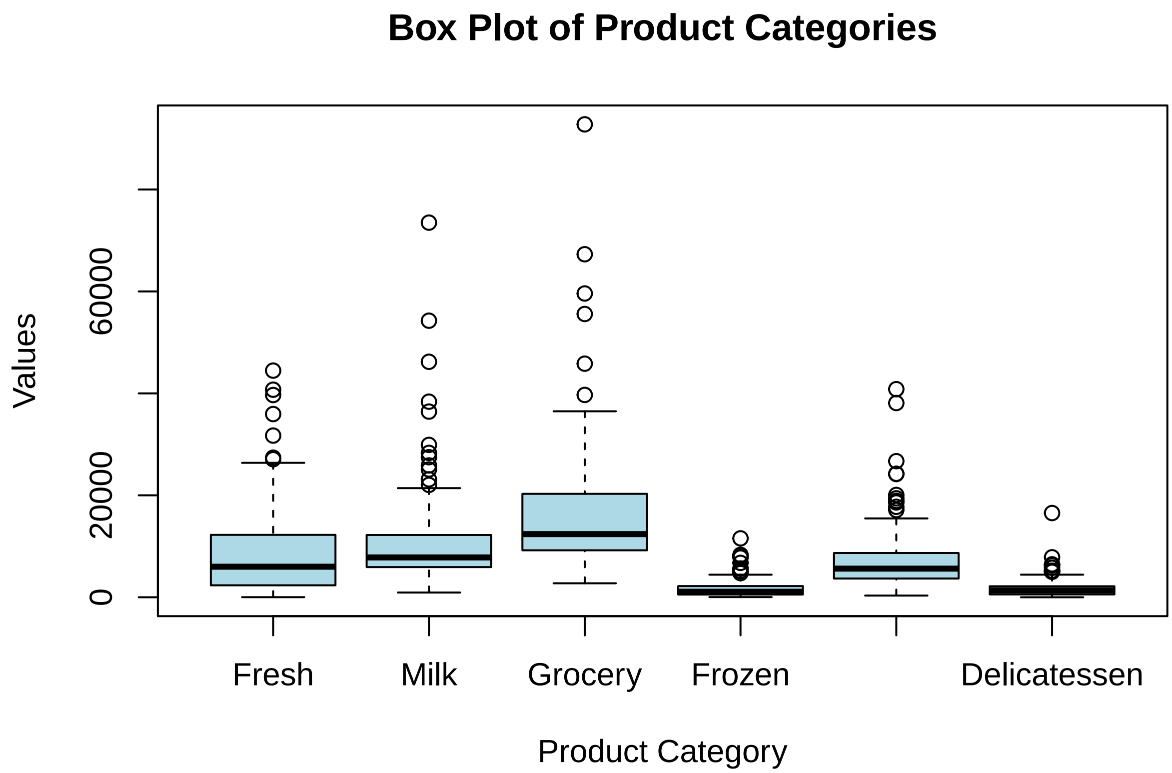
```
library(gridExtra)
```

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

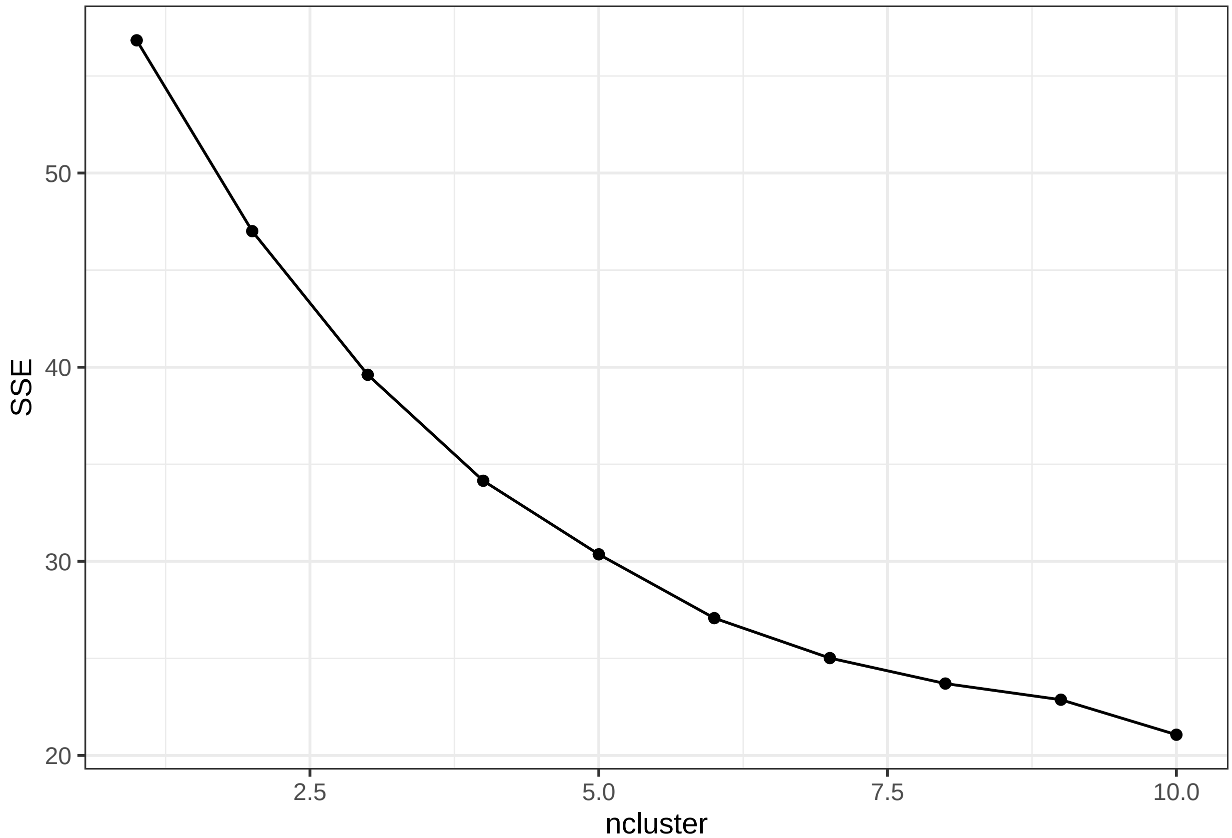
```r
# Function to filter outliers based on IQR
remove_outliers <- function(df) {
  for (col in colnames(df)) {
    Q1 <- quantile(df[[col]], 0.25)
    Q3 <- quantile(df[[col]], 0.75)
    IQR <- Q3 - Q1

    # Define lower and upper bounds for outliers
    lower_bound <- Q1 - 1.5 * IQR
    upper_bound <- Q3 + 1.5 * IQR

    # Filter out the outliers
    df <- df[df[[col]] >= lower_bound & df[[col]] <= upper_bound, ]
  }
  return(df)
}

customers_1_rawdf<- customers_df %>% filter(Channel==1)
customers_2_rawdf<- customers_df %>% filter(Channel==2)

#checking for outliers with box plot

# Select only the columns we need
customer1_data <- customers_1_rawdf[, c("Fresh", "Milk", "Grocery", "Frozen",
                                        "Detergents_Paper", "Delicatessen")]
customer2_data <- customers_2_rawdf[, c("Fresh", "Milk", "Grocery", "Frozen",
                                        "Detergents_Paper", "Delicatessen")]

#Channel 1
# Convert the dataframe to long format for ggplot


boxplot(customers_1_rawdf$Fresh, customers_1_rawdf$Milk, customers_1_rawdf$Grocery,
        customers_1_rawdf$Frozen, customers_1_rawdf$Detergents_Paper,
        customers_1_rawdf$Delicatessen,
        names = c("Fresh", "Milk", "Grocery", "Frozen", "Detergents_Paper", "Delicatessen"),
        main = "Box Plot of Product Categories",
        xlab = "Product Category", ylab = "Values", col = "lightblue")
```



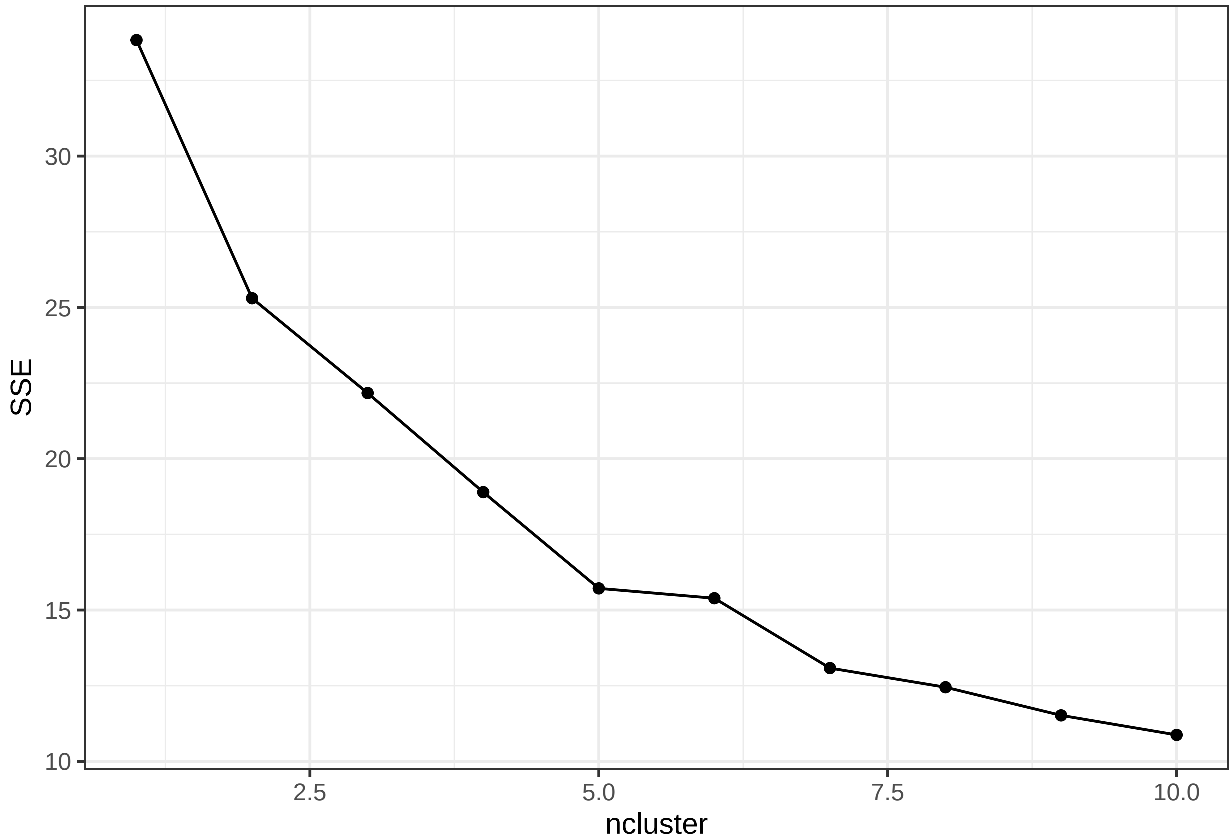

```r
#Channel 2
# Convert the dataframe to long format for ggplot
boxplot(customers_2_rawdf$Fresh, customers_2_rawdf$Milk, customers_2_rawdf$Grocery,
        customers_2_rawdf$Frozen, customers_2_rawdf$Detergents_Paper,
        customers_2_rawdf$Delicatessen,
        names = c("Fresh", "Milk", "Grocery", "Frozen", "Detergents_Paper", "Delicatessen"),
        main = "Box Plot of Product Categories",
        xlab = "Product Category", ylab = "Values", col = "lightblue")
```




```r
# Apply the function to remove outliers
customers_1_df <- remove_outliers(customers_1_rawdf)
customers_2_df <- remove_outliers(customers_2_rawdf)

customers_1_wootliers_df<-customers_1_df
customers_2_wootliers_df<-customers_2_df


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

customers_1_norm_df = customers_1_df %>% mutate_at(c(3:8), normalize)
customers_2_norm_df = customers_2_df %>% mutate_at(c(3:8), normalize)

#sse for channel 1
SSE_curve = c()
for (n in 1:10) {
  # fill in
  kcluster = kmeans(customers_1_norm_df[,3:8], centers = n)
  SSE_curve[n] = kcluster$tot.withinss
}
plot_data = data.frame(ncluster = 1:10, SSE = SSE_curve)
ggplot(plot_data, aes(x = ncluster, y = SSE)) +
  geom_line() + geom_point() +
  theme_bw()
```




```
#sse for channel 2

SSE_curve = c()
for (n in 1:10) {
  # fill in
  kcluster = kmeans(customers_2_norm_df[,3:8], centers = n)
  SSE_curve[n] = kcluster$tot.withinss
}
plot_data = data.frame(ncluster = 1:10, SSE = SSE_curve)
ggplot(plot_data, aes(x = ncluster, y = SSE)) +
  geom_line() + geom_point() +
  theme_bw()
```




```r
# k cluster for center 5 channel 1
kcluster = kmeans(customers_1_norm_df[,3:8], centers = 5)
kcluster$centers
```

```
##       Fresh      Milk   Grocery    Frozen Detergents_Paper Delicatessen
## 1 0.5435876 0.2274126 0.2914163 0.2673853        0.2002865    0.5223861
## 2 0.2445067 0.3180849 0.3785355 0.1968291        0.6616611    0.2599196
## 3 0.1799970 0.1521339 0.2087176 0.1874952        0.1513777    0.1966866
## 4 0.3197886 0.3597746 0.3538528 0.8156375        0.2572043    0.3972915
## 5 0.1589994 0.5644444 0.4520962 0.1629418        0.1918103    0.4238905
```

```r
kclusters <-as.data.frame(kcluster$cluster)
colnames(kclusters) <- "cluster"
customers_1_df$cluster_5<- kclusters$cluster
kcluster$centers
```

```
##       Fresh      Milk   Grocery    Frozen Detergents_Paper Delicatessen
## 1 0.5435876 0.2274126 0.2914163 0.2673853        0.2002865    0.5223861
## 2 0.2445067 0.3180849 0.3785355 0.1968291        0.6616611    0.2599196
## 3 0.1799970 0.1521339 0.2087176 0.1874952        0.1513777    0.1966866
## 4 0.3197886 0.3597746 0.3538528 0.8156375        0.2572043    0.3972915
## 5 0.1589994 0.5644444 0.4520962 0.1629418        0.1918103    0.4238905
```

```r
# Function to denormalize values channel 1
denormalize <- function(normalized_value, original_min, original_max) {
  return(normalized_value * (original_max - original_min) + original_min)
}

# Calculate min and max values of original data
original_mins <- apply(customers_1_df[, c("Fresh", "Milk", "Grocery",
                                          "Detergents_Paper","Delicatessen", "Frozen")], 2, min)
original_maxs <- apply(customers_1_df[, c("Fresh", "Milk", "Grocery",
                                          "Detergents_Paper", "Delicatessen", "Frozen")], 2, max)

# Denormalize cluster centroids
denormalize_centroids <- function(centroids, original_mins, original_maxs) {
  denormalized_centroids <- centroids
  for (col in names(centroids)) {
    if (col %in% names(original_mins)) {
      denormalized_centroids[[col]] <- denormalize(centroids[[col]],
                                                   original_mins[[col]], original_maxs[[col]])
    }
  }
  return(denormalized_centroids)
}

# Get normalized centroids
normalized_centroids <- aggregate(customers_1_norm_df[, c("Fresh", "Milk", "Grocery",
                                                          "Detergents_Paper", "Delicatessen",
                                                          "Frozen")],
                                  by = list(Cluster = kcluster$cluster),
                                  FUN = mean)

# Denormalize centroids
denormalized_centroids <- denormalize_centroids(normalized_centroids, original_mins, original_maxs)

# Print denormalized centroids
print("Denormalized Cluster Centroids:")
```

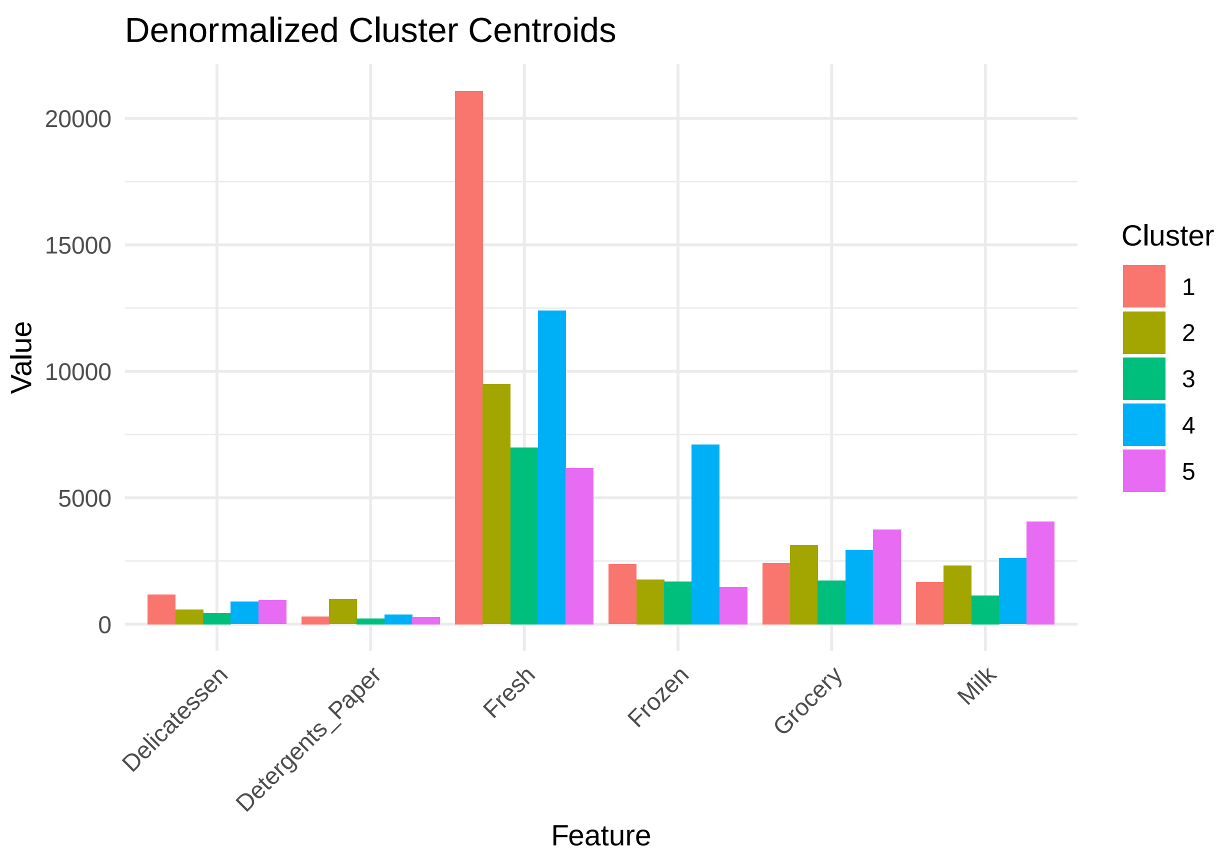
```
## [1] "Denormalized Cluster Centroids:"
```

```r
print(denormalized_centroids)
```

```
##   Cluster     Fresh     Milk  Grocery Detergents_Paper Delicatessen   Frozen
## 1       1 21088.763 1668.947 2415.053         301.0263    1167.9211 2372.000
## 2       2  9487.414 2312.448 3136.138         987.5517     582.6207 1763.241
## 3       3  6985.083 1134.694 1730.556         228.2500     441.6111 1682.708
## 4       4 12407.600 2608.320 2931.840         385.7200     888.9600 7102.320
## 5       5  6170.586 4060.862 3745.000         288.4138     948.2759 1470.862
```

```r
# Visualize the denormalized centroids
ggplot(tidyr::pivot_longer(denormalized_centroids, cols = -Cluster, names_to = "Feature",
                           values_to = "Value"),
       aes(x = Feature, y = Value, fill = as.factor(Cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
```

```r
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Denormalized Cluster Centroids",
       x = "Feature",
       y = "Value",
       fill = "Cluster")
```




```r
# k cluster for center 4 channel 2
kcluster = kmeans(customers_2_norm_df[,3:8], centers = 4)
kcluster$centers
```

```
##       Fresh      Milk   Grocery    Frozen Detergents_Paper Delicatessen
## 1 0.2212132 0.5216589 0.6123059 0.4966760        0.6784217    0.3632251
## 2 0.1460549 0.3325824 0.2670536 0.2585482        0.3527394    0.6630160
## 3 0.5338033 0.2457337 0.1599348 0.4070340        0.2326483    0.3369857
## 4 0.1059451 0.3499693 0.3673288 0.1232082        0.4283439    0.1475843
```

```r
kclusters <-as.data.frame(kcluster$cluster)
colnames(kclusters) <- "cluster"
customers_2_df$cluster_4<- kclusters$cluster
kcluster$centers
```

```
##       Fresh      Milk   Grocery    Frozen Detergents_Paper Delicatessen
## 1 0.2212132 0.5216589 0.6123059 0.4966760        0.6784217    0.3632251
```

```
## 2 0.1460549 0.3325824 0.2670536 0.2585482        0.3527394    0.6630160
## 3 0.5338033 0.2457337 0.1599348 0.4070340        0.2326483    0.3369857
## 4 0.1059451 0.3499693 0.3673288 0.1232082        0.4283439    0.1475843
```

```
# Function to denormalize values channel 2
denormalize <- function(normalized_value, original_min, original_max) {
  return(normalized_value * (original_max - original_min) + original_min)
}

# Calculate min and max values of original data
original_mins <- apply(customers_2_df[, c("Fresh", "Milk", "Grocery", "Detergents_Paper",
                                          "Delicatessen", "Frozen")], 2, min)
original_maxs <- apply(customers_2_df[, c("Fresh", "Milk", "Grocery", "Detergents_Paper",
                                          "Delicatessen", "Frozen")], 2, max)

# Denormalize cluster centroids
denormalize_centroids <- function(centroids, original_mins, original_maxs) {
  denormalized_centroids <- centroids
  for (col in names(centroids)) {
    if (col %in% names(original_mins)) {
      denormalized_centroids[[col]] <- denormalize(centroids[[col]], original_mins[[col]],
                                                    original_maxs[[col]])
    }
  }
  return(denormalized_centroids)
}

# Get normalized centroids
normalized_centroids <- aggregate(customers_2_norm_df[, c("Fresh", "Milk", "Grocery",
                                                          "Detergents_Paper", "Delicatessen",
                                                          "Frozen")],
                                  by = list(Cluster = kcluster$cluster),
                                  FUN = mean)

# Denormalize centroids
denormalized_centroids <- denormalize_centroids(normalized_centroids, original_mins, original_maxs)

# Print denormalized centroids
print("Denormalized Cluster Centroids:")
```

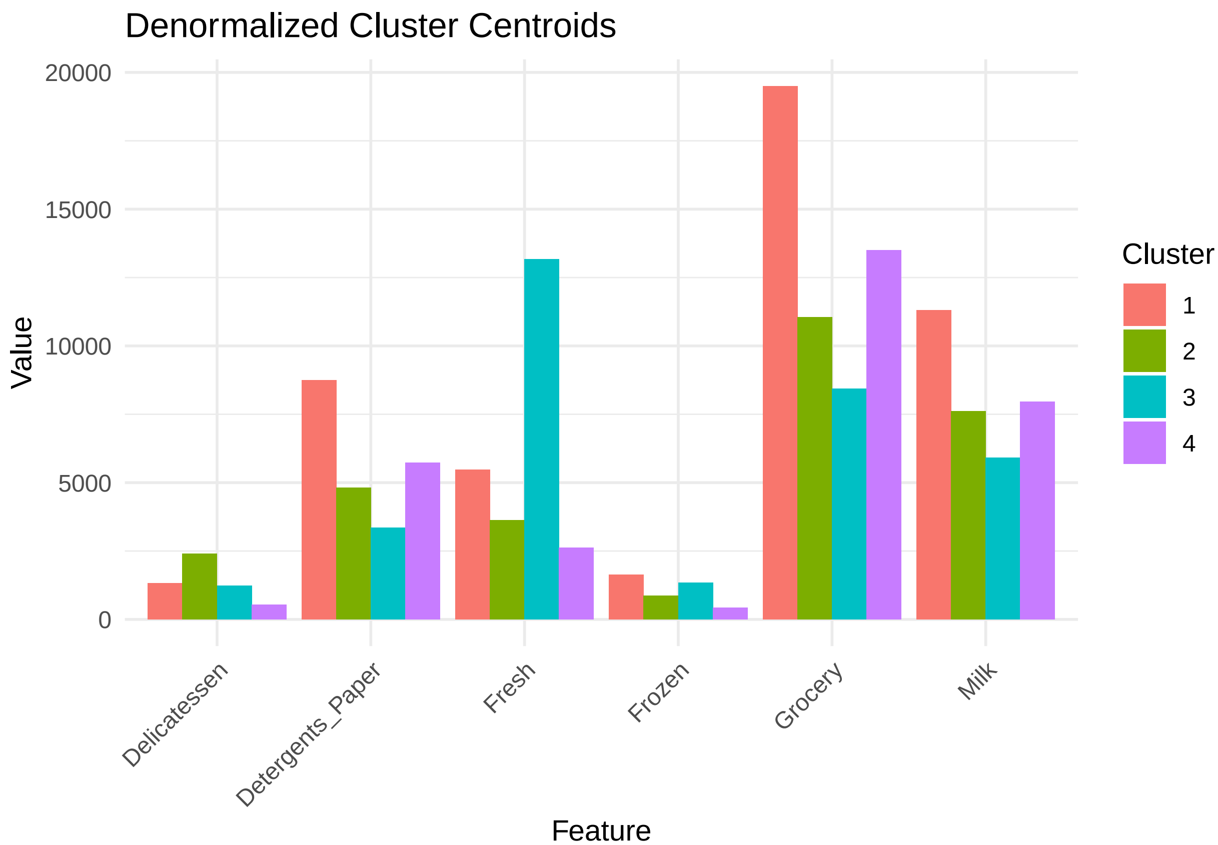
```
## [1] "Denormalized Cluster Centroids:"
```

```
print(denormalized_centroids)
```

```
##   Cluster     Fresh      Milk   Grocery Detergents_Paper Delicatessen    Frozen
## 1       1  5471.480 11312.520 19501.840         8750.440    1322.9600 1631.8000
## 2       2  3620.333  7619.667 11055.933         4813.267    2412.4000  865.2667
## 3       3 13170.576  5923.424  8435.485         3361.485    1227.6061 1343.2424
## 4       4  2632.429  7959.250 13508.964         5727.250     539.3214  429.6071
```

```
# Visualize the denormalized centroids
ggplot(tidyr::pivot_longer(denormalized_centroids, cols = -Cluster, names_to = "Feature",
```

```r
                              values_to = "Value"),
     aes(x = Feature, y = Value, fill = as.factor(Cluster))) +
  geom_bar(stat = "identity", position = "dodge") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Denormalized Cluster Centroids",
       x = "Feature",
       y = "Value",
       fill = "Cluster")
```



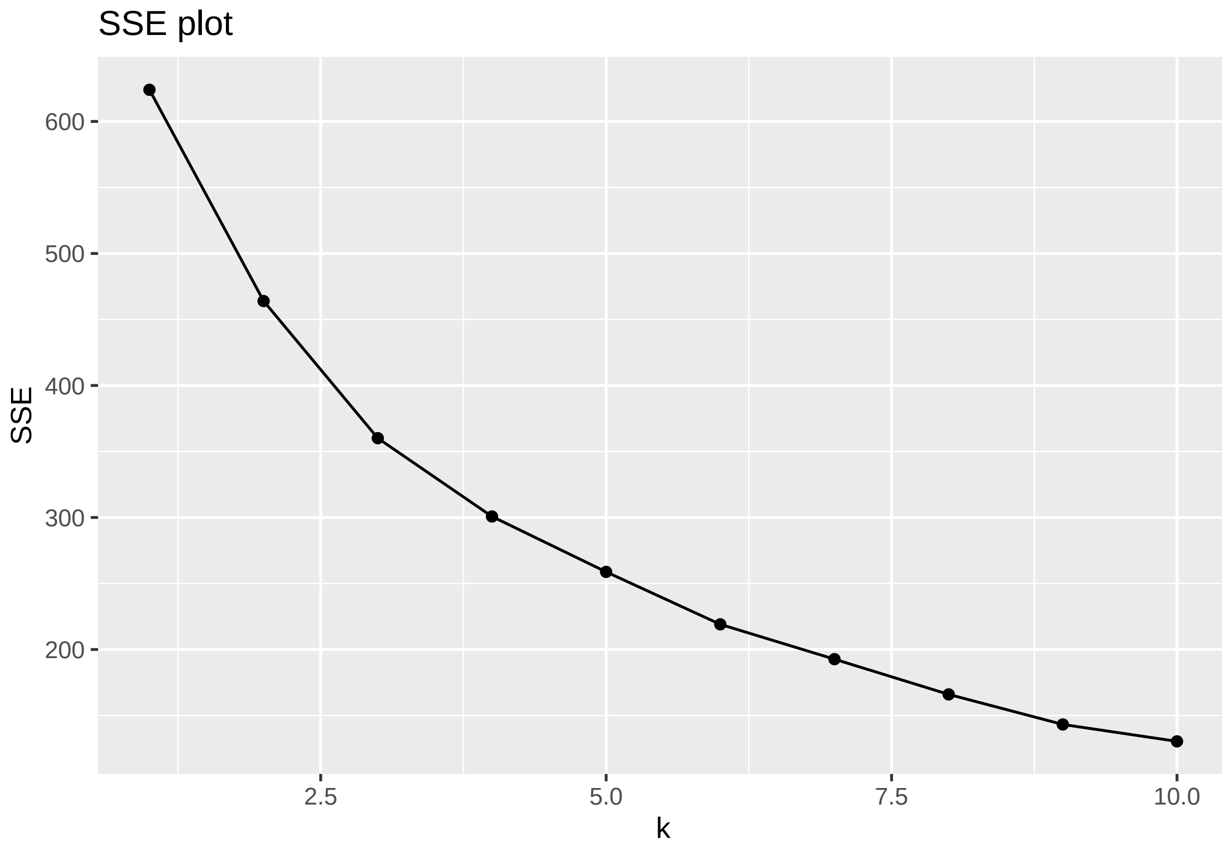

```r
#save channel 1 and channel 2 solution
write_xlsx(customers_1_df,"/Users/justinvarghese/Library/CloudStorage/GoogleDrive-just4dec@gmail.com/My
write_xlsx(customers_2_df,"/Users/justinvarghese/Library/CloudStorage/GoogleDrive-just4dec@gmail.com/My


#outlier analysis for channel 1

customers_1_outlier_df <- setdiff(customers_1_rawdf,customers_1_wootliers_df )
customers_2_outlier_df <- setdiff(customers_2_rawdf,customers_2_wootliers_df)


#import data
data <- customers_1_outlier_df
data$Channel <- factor(data$Channel,
                       levels = c(1, 2),
```

```r
                          labels = c("Horeca", "Retail"))
data$Region <- factor(data$Region,
                         levels = c(1, 2, 3),
                         labels = c("Lisbon", "Oporto", "Other"))

# 2. EDA
# correlation matrix
cor_matrix <- cor(data[, 3:8])
print(cor_matrix)
```

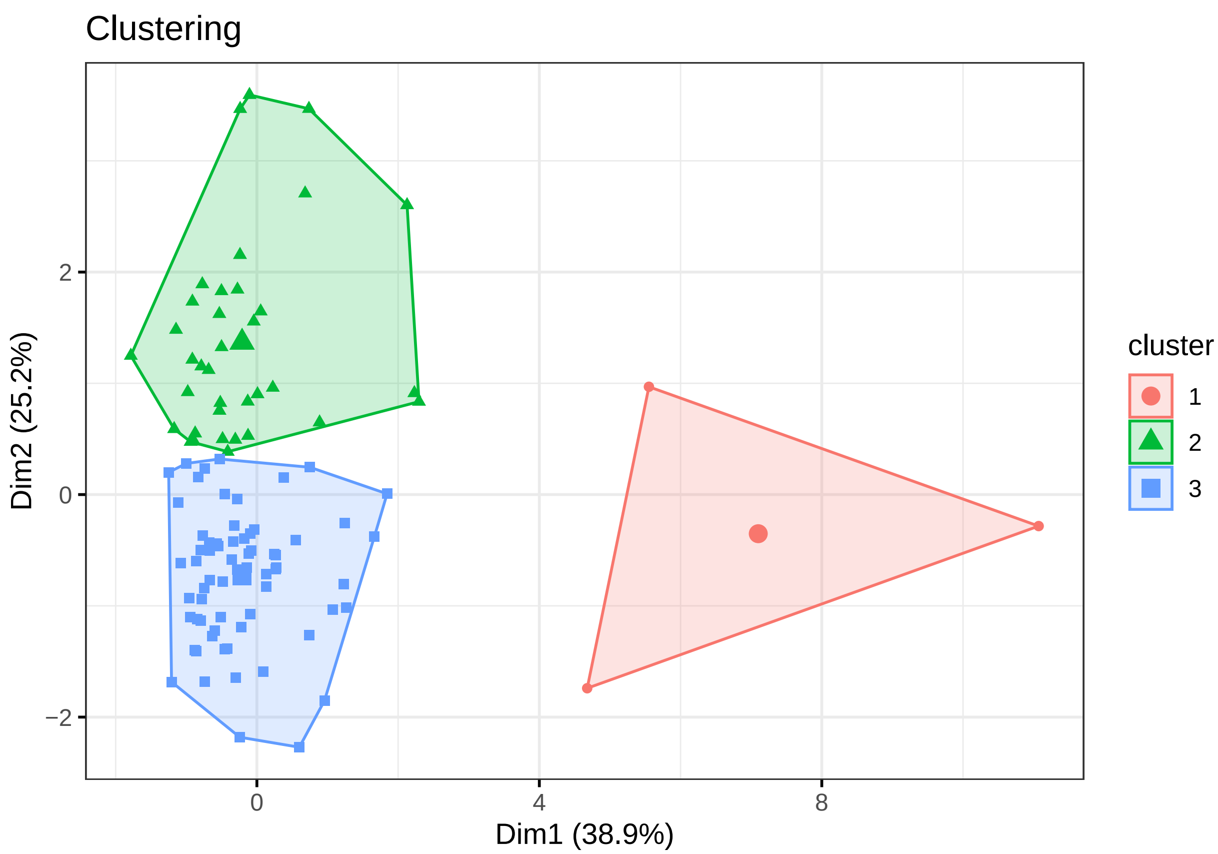
```
##                        Fresh       Milk    Grocery     Frozen Detergents_Paper
## Fresh             1.00000000 0.16616621 0.09422933  0.2979889      -0.20020448
## Milk              0.16616621 1.00000000 0.50176760  0.3496359       0.03241055
## Grocery           0.09422933 0.50176760 1.00000000  0.1279663       0.39906921
## Frozen            0.29798895 0.34963594 0.12796626  1.0000000      -0.24104057
## Detergents_Paper -0.20020448 0.03241055 0.39906921 -0.2410406       1.00000000
## Delicatessen      0.21226766 0.61488226 0.39042247  0.3916349      -0.08321266
##                  Delicatessen
## Fresh              0.21226766
## Milk               0.61488226
## Grocery            0.39042247
## Frozen             0.39163492
## Detergents_Paper  -0.08321266
## Delicatessen       1.00000000
```

```r
# 3. cluster analysis

# normalize data
cluster_data <- data[, 3:8]
scaled_data <- scale(cluster_data)

# elbow
set.seed(123)
wss <- sapply(1:10, function(k){
  kmeans(scaled_data, k, nstart=25)$tot.withinss
})

# elbow plot
p1 <- ggplot(data.frame(k=1:10, wss=wss), aes(x=k, y=wss)) +
  geom_line() +
  geom_point() +
  labs(title="SSE plot",
       x="k",
       y="SSE")

# K-means clustering
km_result <- kmeans(scaled_data, centers = 3, nstart = 25)
data$Cluster <- as.factor(km_result$cluster)

# visualization clustering result
p2 <- fviz_cluster(km_result, data = scaled_data,
                   geom = "point",
                   ellipse.type = "convex",
```

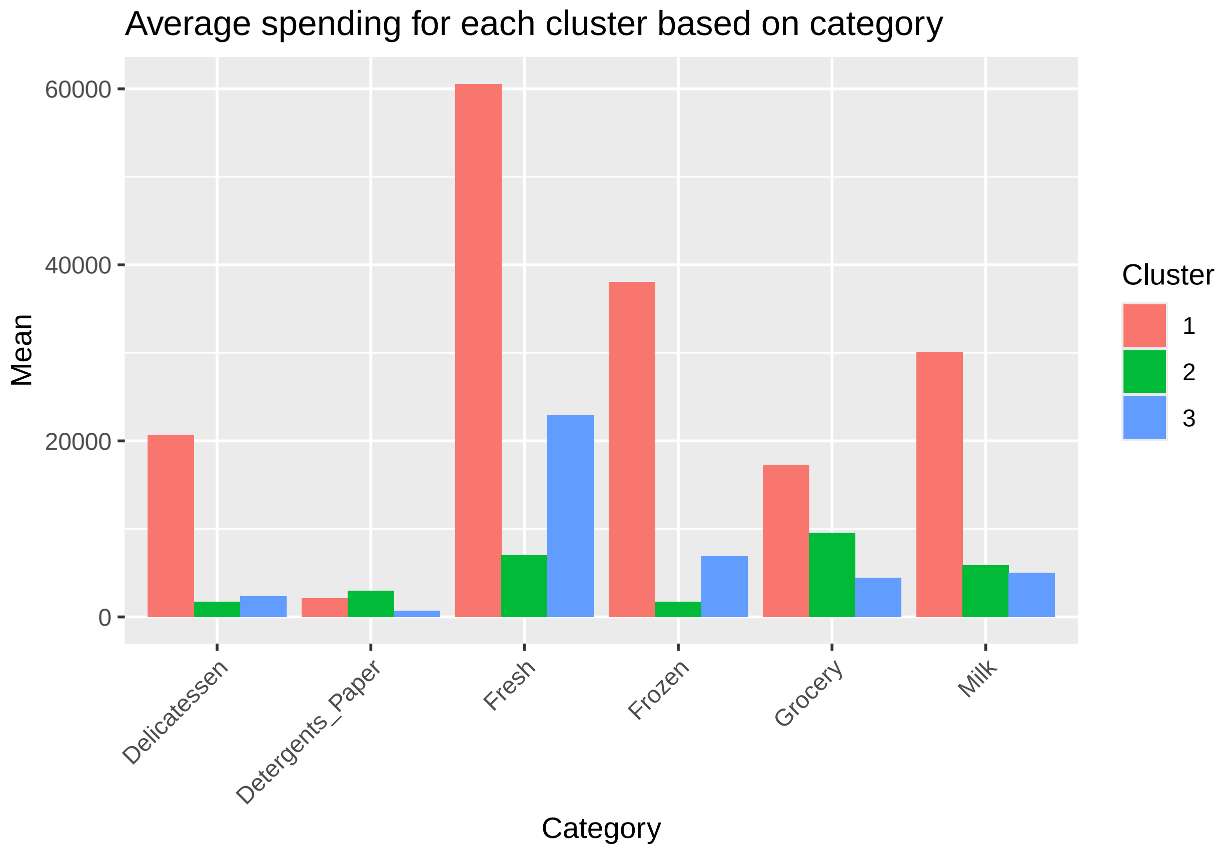
```r
                   ggtheme = theme_bw()) +
  labs(title="Clustering")

# cluster means
cluster_means <- aggregate(cluster_data,
                          by = list(Cluster = data$Cluster),
                          mean)

# Average spending for each cluster based on category
cluster_means_long <- cluster_means %>%
  gather(key = "Category", value = "Mean", -Cluster)

p3 <- ggplot(cluster_means_long,
             aes(x = Category, y = Mean, fill = Cluster)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Average spending for each cluster based on category")

# channel/region distribution
channel_distribution <- table(data$Channel, data$Cluster)
region_distribution <- table(data$Region, data$Cluster)

# Clusters' distribution among different channels and regions
p4_prop <- ggplot(data, aes(x = Channel, fill = Cluster)) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of clusters among different channels", y = "proportion")

p5_prop <- ggplot(data, aes(x = Region, fill = Cluster)) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of clusters among different regions ", y = "proportion")

p4_abs <- ggplot(data, aes(x = Channel, fill = Cluster)) +
  geom_bar(position = "stack") +
  labs(title = "Amount of clusters among different channels (amount)", y = "amount")

p5_abs <- ggplot(data, aes(x = Region, fill = Cluster)) +
  geom_bar(position = "stack") +
  labs(title = "Amount of clusters among different regions (amount)", y = "amount")

# Outputs
p1
```



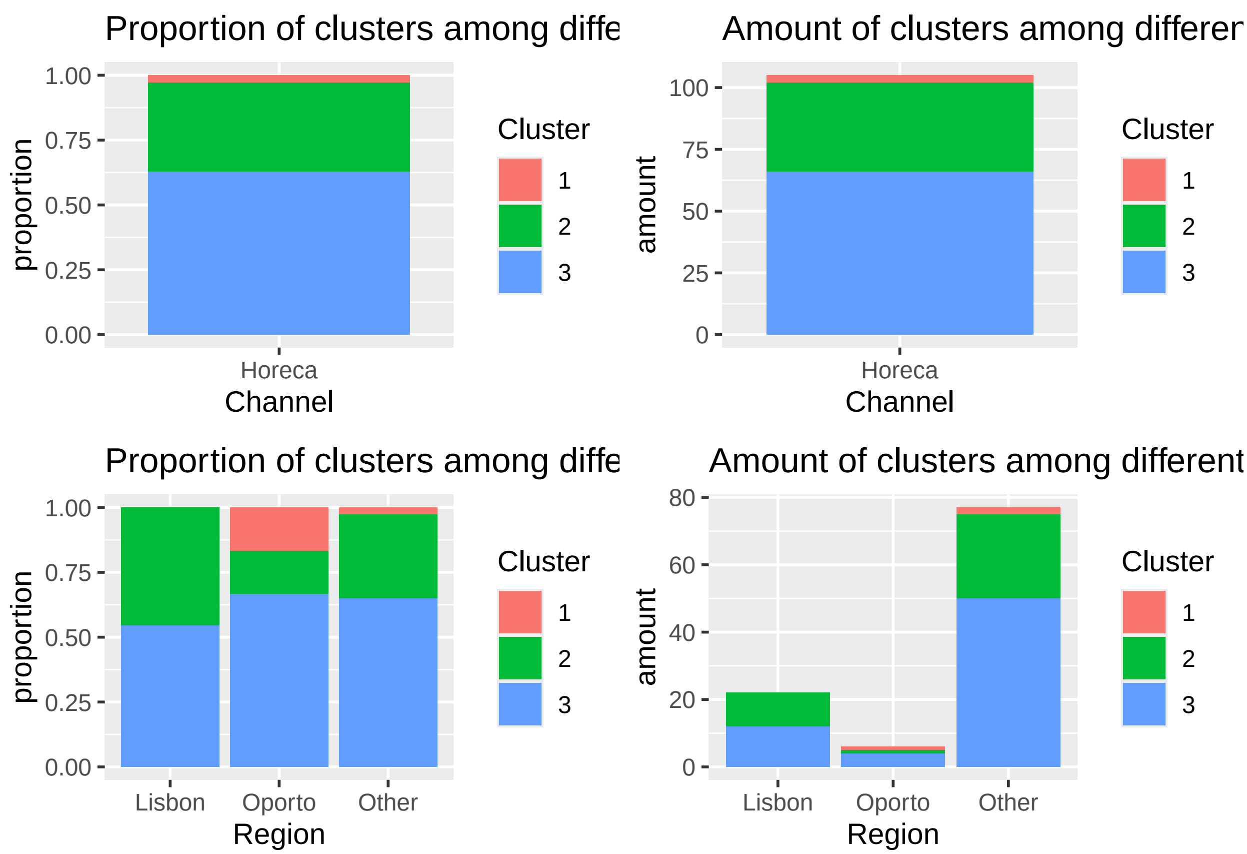

```
p2
```




```
p3
```




```
grid.arrange(p4_prop, p4_abs, p5_prop, p5_abs, ncol = 2)
```




```r
print(channel_distribution)
```

```
##
##           1  2  3
##   Horeca  3 36 66
##   Retail  0  0  0
```

```r
print(region_distribution)
```

```
##
##           1  2  3
##   Lisbon  0 10 12
##   Oporto  1  1  4
##   Other   2 25 50
```

```r
print(cluster_means)
```

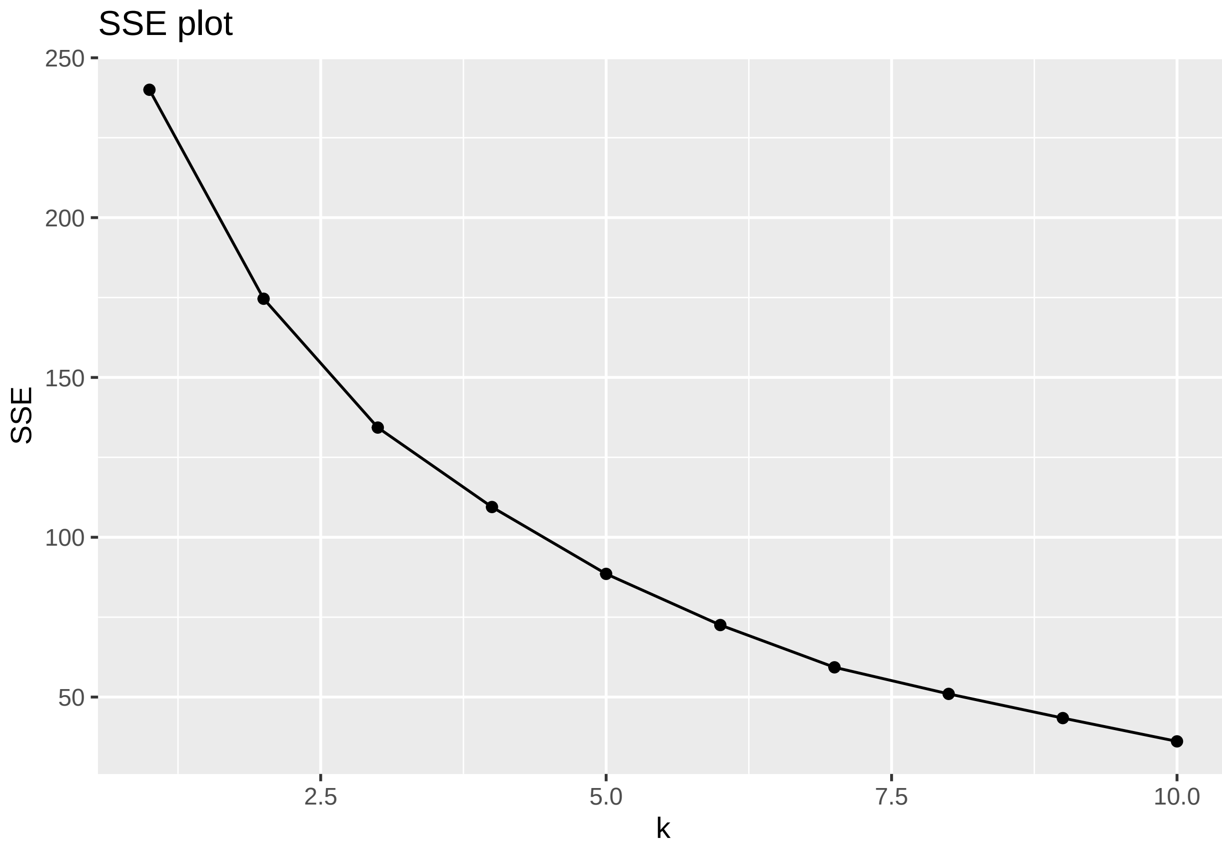
```
##   Cluster     Fresh      Milk   Grocery    Frozen Detergents_Paper Delicatessen
## 1       1 60571.667 30120.333 17314.667 38049.333        2153.0000    20700.667
## 2       2  7025.306  5877.500  9572.806  1754.806        2983.1667     1745.111
## 3       3 22916.970  5022.894  4468.652  6924.545         715.3788     2336.848
```

```r
#outlier analysis for channel 2


#import data
data <- customers_2_outlier_df
data$Channel <- factor(data$Channel,
                       levels = c(1, 2),
                       labels = c("Horeca", "Retail"))
data$Region <- factor(data$Region,
                      levels = c(1, 2, 3),
                      labels = c("Lisbon", "Oporto", "Other"))

# 2. EDA
# correlation matrix
cor_matrix <- cor(data[, 3:8])
print(cor_matrix)
```

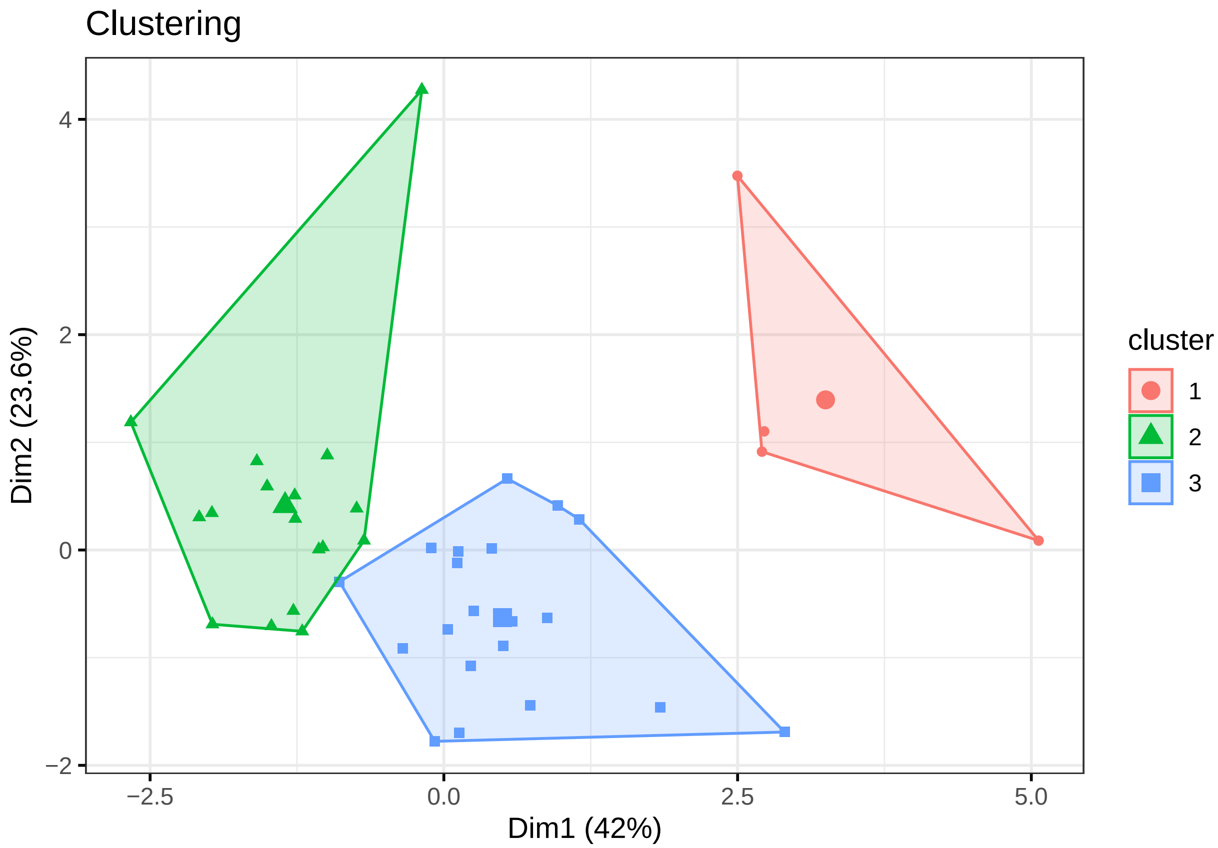
```
##                        Fresh        Milk     Grocery      Frozen
## Fresh             1.00000000  0.20276827  0.01802480  0.08799693
## Milk              0.20276827  1.00000000  0.59397583 -0.07773864
## Grocery           0.01802480  0.59397583  1.00000000 -0.25980755
## Frozen            0.08799693 -0.07773864 -0.25980755  1.00000000
## Detergents_Paper -0.08249167  0.56434057  0.95110043 -0.30273687
## Delicatessen      0.17818232  0.23401003 -0.01860488  0.14803779
##                  Detergents_Paper Delicatessen
## Fresh                 -0.08249167   0.17818232
## Milk                   0.56434057   0.23401003
## Grocery                0.95110043  -0.01860488
## Frozen                -0.30273687   0.14803779
## Detergents_Paper       1.00000000  -0.15501950
## Delicatessen          -0.15501950   1.00000000
```

```r
# 3. cluster analysis

# normalize data
cluster_data <- data[, 3:8]
scaled_data <- scale(cluster_data)

# elbow
set.seed(123)
wss <- sapply(1:10, function(k){
  kmeans(scaled_data, k, nstart=25)$tot.withinss
})

# elbow plot
p1 <- ggplot(data.frame(k=1:10, wss=wss), aes(x=k, y=wss)) +
  geom_line() +
  geom_point() +
  labs(title="SSE plot",
       x="k",
       y="SSE")

# K-means clustering
```

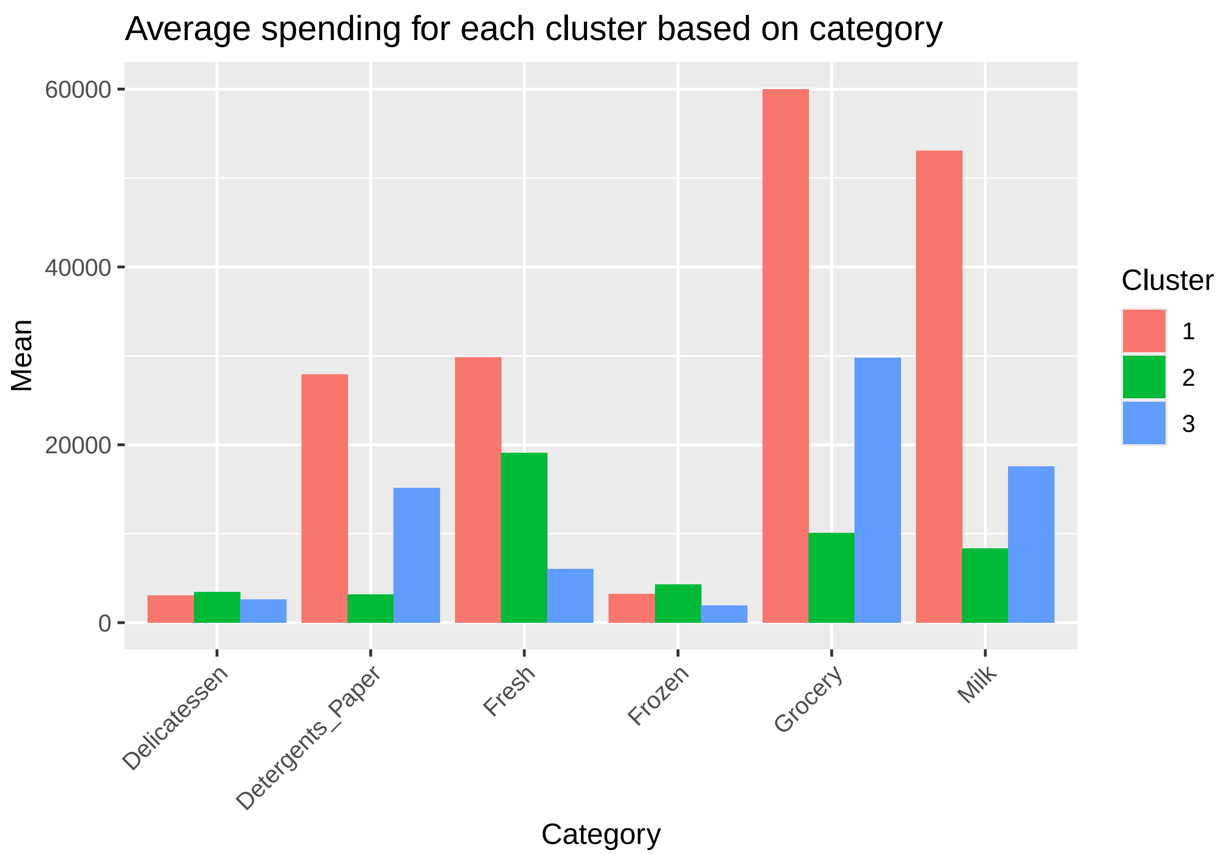
```r
km_result <- kmeans(scaled_data, centers = 3, nstart = 25)
data$Cluster <- as.factor(km_result$cluster)

# visualization clustering result
p2 <- fviz_cluster(km_result, data = scaled_data,
                   geom = "point",
                   ellipse.type = "convex",
                   ggtheme = theme_bw()) +
  labs(title="Clustering")

# cluster means
cluster_means <- aggregate(cluster_data,
                           by = list(Cluster = data$Cluster),
                           mean)

# Average spending for each cluster based on category
cluster_means_long <- cluster_means %>%
  gather(key = "Category", value = "Mean", -Cluster)

p3 <- ggplot(cluster_means_long,
             aes(x = Category, y = Mean, fill = Cluster)) +
  geom_bar(stat = "identity", position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Average spending for each cluster based on category")

# channel/region distribution
channel_distribution <- table(data$Channel, data$Cluster)
region_distribution <- table(data$Region, data$Cluster)

# Clusters' distribution among different channels and regions
p4_prop <- ggplot(data, aes(x = Channel, fill = Cluster)) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of clusters among different channels", y = "proportion")

p5_prop <- ggplot(data, aes(x = Region, fill = Cluster)) +
  geom_bar(position = "fill") +
  labs(title = "Proportion of clusters among different regions ", y = "proportion")

p4_abs <- ggplot(data, aes(x = Channel, fill = Cluster)) +
  geom_bar(position = "stack") +
  labs(title = "Amount of clusters among different channels (amount)", y = "amount")

p5_abs <- ggplot(data, aes(x = Region, fill = Cluster)) +
  geom_bar(position = "stack") +
  labs(title = "Amount of clusters among different regions (amount)", y = "amount")

# Outputs
p1
```



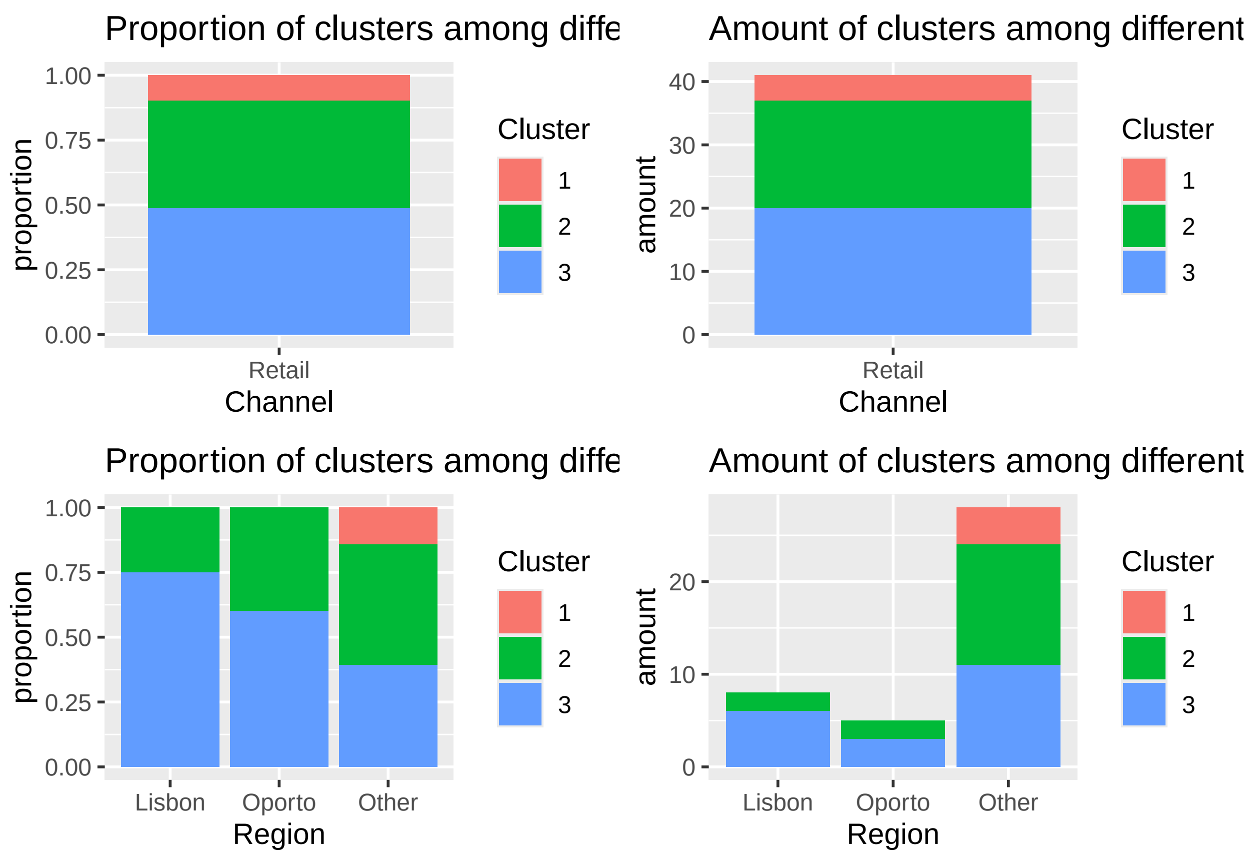

```
p2
```




```
p3
```




```r
grid.arrange(p4_prop, p4_abs, p5_prop, p5_abs, ncol = 2)
```




```r
print(channel_distribution)
```

```
##
##           1  2  3
##   Horeca  0  0  0
##   Retail  4 17 20
```

```r
print(region_distribution)
```

```
##
##           1  2  3
##   Lisbon  0  2  6
##   Oporto  0  2  3
##   Other   4 13 11
```

```r
print(cluster_means)
```

```
##   Cluster    Fresh      Milk  Grocery   Frozen Detergents_Paper Delicatessen
## 1       1 29862.50 53080.750 60015.75 3262.250        27942.250     3082.250
## 2       2 19122.82  8370.824 10098.82 4293.529         3205.824     3475.235
## 3       3  6022.35 17583.800 29804.75 1925.100        15187.500     2635.400
```