Quantium Virtual Internship - Retail Strategy and Analytics - Task 1

David Singh

Solution for Task 1

This document provides a comprehensive solution for Task 1 of the Quantium Virtual Internship. The analysis involves loading and cleaning retail transaction and customer data, performing exploratory data analysis, and deriving insights into customer segments based on purchasing behavior.

Step 1: Loading Required Libraries and Datasets

We begin by loading the necessary R libraries and datasets. Ensuring these packages are installed prior to running the script.

```r
# Load required libraries
library(data.table)
library(ggplot2)
library(ggmosaic)
library(readr)
library(readxl)
library(janitor)
library(dplyr)
library(lubridate)
```

Setting the file path to our current working directory and loading the transaction and customer datasets. Adjusting the path as needed for our system (Windows 10 pro here).

```r
filePath <- "C:/Users/david/Desktop/QT1/"

# Load Datasets
transactionData <- read_excel(paste0(filePath, "QVI_transaction_data.xlsx"))
customerData <- fread(paste0(filePath, "QVI_purchase_behaviour.csv"))
```

Step 2: Exploratory Data Analysis - Transaction Data

Examine the Transaction Data

Let's inspect the structure and initial rows of the transaction data to understand its format and content.

```r
str(transactionData)
```

```
## tibble [264,836 x 8] (S3: tbl_df/tbl/data.frame)
##  $ DATE          : num [1:264836] 43390 43599 43605 43329 43330 ...
##  $ STORE_NBR     : num [1:264836] 1 1 1 2 2 4 4 4 5 7 ...
##  $ LYLTY_CARD_NBR: num [1:264836] 1000 1307 1343 2373 2426 ...
##  $ TXN_ID        : num [1:264836] 1 348 383 974 1038 ...
##  $ PROD_NBR      : num [1:264836] 5 66 61 69 108 57 16 24 42 52 ...
##  $ PROD_NAME     : chr [1:264836] "Natural Chip        Compny SeaSalt175g" "CCs Nacho Cheese    175g
##  $ PROD_QTY      : num [1:264836] 2 3 2 5 3 1 1 1 1 2 ...
##  $ TOT_SALES     : num [1:264836] 6 6.3 2.9 15 13.8 5.1 5.7 3.6 3.9 7.2 ...
```

```r
head(transactionData)
```

```
## # A tibble: 6 x 8
##    DATE STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME    PROD_QTY TOT_SALES
##   <dbl>     <dbl>          <dbl>  <dbl>    <dbl> <chr>           <dbl>     <dbl>
## 1 43390         1           1000      1        5 Natural Chi~        2       6
## 2 43599         1           1307    348       66 CCs Nacho C~        3       6.3
## 3 43605         1           1343    383       61 Smiths Crin~        2       2.9
## 4 43329         2           2373    974       69 Smiths Chip~        5      15
## 5 43330         2           2426   1038      108 Kettle Tort~        3      13.8
## 6 43604         4           4074   2982       57 Old El Paso~        1       5.1
```

The `DATE` column is in an integer format, which we need to convert to a proper date format for analysis.

Convert "DATE" to Date Format

Excel integer dates start from December 30, 1899. We use this origin to convert the `date` column.

```r
transactionData$DATE <- as.Date(transactionData$DATE, origin = "1899-12-30")
head(transactionData)
```

```
## # A tibble: 6 x 8
##   DATE       STORE_NBR LYLTY_CARD_NBR TXN_ID PROD_NBR PROD_NAME         PROD_QTY
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                <dbl>
## 1 2018-10-17         1           1000      1        5 Natural Chip    ~        2
## 2 2019-05-14         1           1307    348       66 CCs Nacho Cheese~        3
## 3 2019-05-20         1           1343    383       61 Smiths Crinkle C~        2
## 4 2018-08-17         2           2373    974       69 Smiths Chip Thin~        5
## 5 2018-08-18         2           2426   1038      108 Kettle Tortilla ~        3
## 6 2019-05-19         4           4074   2982       57 Old El Paso Sals~        1
## # i 1 more variable: TOT_SALES <dbl>
```

Cleaning & Standardizing names of transactionData & customerData attributes

```r
library(janitor)
transactionData <- clean_names(transactionData)
customerData <- clean_names(customerData)
names(transactionData)
```

```
## [1] "date"           "store_nbr"      "lylty_card_nbr" "txn_id"
## [5] "prod_nbr"       "prod_name"      "prod_qty"       "tot_sales"
```

```r
names(customerData)
```

```
## [1] "lylty_card_nbr"   "lifestage"        "premium_customer"
```

Summary of prod_name

Next, we summarize the `prod_name` column to explore the range of products.

```r
# Top 10 most frequent product names
head(sort(table(transactionData$prod_name), decreasing = TRUE), 10)
```

```
##
##   Kettle Mozzarella   Basil & Pesto 175g
##                                     3304
## Kettle Tortilla ChpsHny&Jlpno Chili 150g
##                                     3296
## Cobs Popd Swt/Chlli &Sr/Cream Chips 110g
##                                     3269
##   Tyrrells Crisps     Ched & Chives 165g
##                                     3268
##           Cobs Popd Sea Salt  Chips 110g
##                                     3265
##             Kettle 135g Swt Pot Sea Salt
##                                     3257
##            Tostitos Splash Of  Lime 175g
##                                     3252
## Infuzions Thai SweetChili PotatoMix 110g
##                                     3242
##   Smiths Crnkle Chip  Orgnl Big Bag 380g
##                                     3233
##     Thins Potato Chips  Hot & Spicy 175g
##                                     3229
```

```r
# We can use "table(transactionData$prod_name)" to show all values.
```

This shows various potato chip products, but we need to ensure no non-chip items (e.g., salsa) are included.

Further Examining prod_name

As we are only interested in words that will tell us if the product is chips or not, let's remove all words with digits and special characters such as '&' from our set of product words. We can do this using `grepl()`

To confirm we're analyzing only chips, we break down `prod_name` into individual words and analyze their frequency, excluding numbers and punctuation.

```r
# Step 1: Convert prod_name to character
prod_names <- as.character(unique(transactionData$prod_name))

# Step 2: Split into individual words
productWords <- data.table(words = unlist(strsplit(prod_names, " ")))

# Step 3: Remove numbers and punctuation
productWords <- productWords[!grepl("[0-9]|[[:punct:]]", words)]

# Step 4: Count word frequency
word_freq <- productWords[, .N, by = words][order(-N)]

# Step 5: View results
print(word_freq)
```

```
##          words     N
##         <char> <int>
##   1:             234
##   2:     Chips    21
##   3:    Smiths    16
##   4:   Crinkle    14
##   5:       Cut    14
##  ---
## 165:       Rst     1
## 166:      Pork     1
## 167:     Belly     1
## 168:        Pc     1
## 169: Bolognese     1
```

Words like "Salsa" appear, indicating non-chip products that need removal.

Remove Salsa Products

We filter out salsa products by identifying them in `prod_name` and excluding them.

```r
library(dplyr)
transactionData <- transactionData %>%
  filter(!grepl("salsa", tolower(prod_name)))
```

Initial Summary

We check for nulls and outliers using a summary of the cleaned dataset.

```r
summary(transactionData)
```

```
##       date              store_nbr     lylty_card_nbr        txn_id
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :    1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:   70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135183
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135531   Mean   : 135131
```

```
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203084   3rd Qu.: 202654
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     prod_nbr       prod_name            prod_qty         tot_sales
##  Min.   :  1.00   Length:246742      Min.   :  1.000   Min.   :  1.700
##  1st Qu.: 26.00   Class :character   1st Qu.:  2.000   1st Qu.:  5.800
##  Median : 53.00   Mode  :character   Median :  2.000   Median :  7.400
##  Mean   : 56.35                      Mean   :  1.908   Mean   :  7.321
##  3rd Qu.: 87.00                      3rd Qu.:  2.000   3rd Qu.:  8.800
##  Max.   :114.00                      Max.   :200.000   Max.   :650.000
```

No nulls are present, but `prod_qty` shows a maximum of 200, suggesting a potential outlier.

Filter Outliers

Investigating transactions with `prod_qty` of 200 reveals they belong to one customer (loyalty card 226000), likely a commercial buyer rather than a typical retail customer.

```r
library(dplyr)
outlier_transactions <- filter(transactionData, prod_qty == 200)
print(outlier_transactions)
```

```
## # A tibble: 2 x 8
##   date       store_nbr lylty_card_nbr txn_id prod_nbr prod_name          prod_qty
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-08-19       226         226000 226201        4 Dorito Corn Chp ~      200
## 2 2019-05-20       226         226000 226210        4 Dorito Corn Chp ~      200
## # i 1 more variable: tot_sales <dbl>
```

```r
customer_transactions <- filter(transactionData,lylty_card_nbr == 226000)
print(customer_transactions)
```

```
## # A tibble: 2 x 8
##   date       store_nbr lylty_card_nbr txn_id prod_nbr prod_name          prod_qty
##   <date>         <dbl>          <dbl>  <dbl>    <dbl> <chr>                 <dbl>
## 1 2018-08-19       226         226000 226201        4 Dorito Corn Chp ~      200
## 2 2019-05-20       226         226000 226210        4 Dorito Corn Chp ~      200
## # i 1 more variable: tot_sales <dbl>
```

Only these two transactions exist, suggesting a commercial buyer.

```r
transactionData <- filter(transactionData,lylty_card_nbr != 226000)
summary(transactionData)
```

```
##       date              store_nbr     lylty_card_nbr        txn_id
##  Min.   :2018-07-01   Min.   :  1.0   Min.   :   1000   Min.   :      1
##  1st Qu.:2018-09-30   1st Qu.: 70.0   1st Qu.:  70015   1st Qu.:  67569
##  Median :2018-12-30   Median :130.0   Median : 130367   Median : 135182
##  Mean   :2018-12-30   Mean   :135.1   Mean   : 135530   Mean   : 135130
##  3rd Qu.:2019-03-31   3rd Qu.:203.0   3rd Qu.: 203083   3rd Qu.: 202652
##  Max.   :2019-06-30   Max.   :272.0   Max.   :2373711   Max.   :2415841
##     prod_nbr       prod_name            prod_qty        tot_sales
```

```
##  Min.   :  1.00   Length:246740      Min.   :1.000   Min.   : 1.700
##  1st Qu.: 26.00   Class :character   1st Qu.:2.000   1st Qu.: 5.800
##  Median : 53.00   Mode  :character   Median :2.000   Median : 7.400
##  Mean   : 56.35                      Mean   :1.906   Mean   : 7.316
##  3rd Qu.: 87.00                      3rd Qu.:2.000   3rd Qu.: 8.800
##  Max.   :114.00                      Max.   :5.000   Max.   :29.500
```

After removal, the maximum `prod_qty` is now 5,and tot_sales max is 29.5 which is more reasonable.

Step 3: Transaction Trends Over Time

We analyze transactions by date to check for missing data or patterns.

```r
# Converting tibble to a data.table
library(data.table)
setDT(transactionData)
```

```r
# Count transactions by date
transactions_by_day <- transactionData[, .(count = .N), by = date]

# Checking result
str(transactions_by_day)
```

```
## Classes 'data.table' and 'data.frame':   364 obs. of  2 variables:
##  $ date : Date, format: "2018-10-17" "2019-05-14" ...
##  $ count: int  682 705 707 663 683 664 644 652 626 632 ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```r
# Checking unique dates
nrow(transactions_by_day) # Should show 364, not 365
```

```
## [1] 364
```

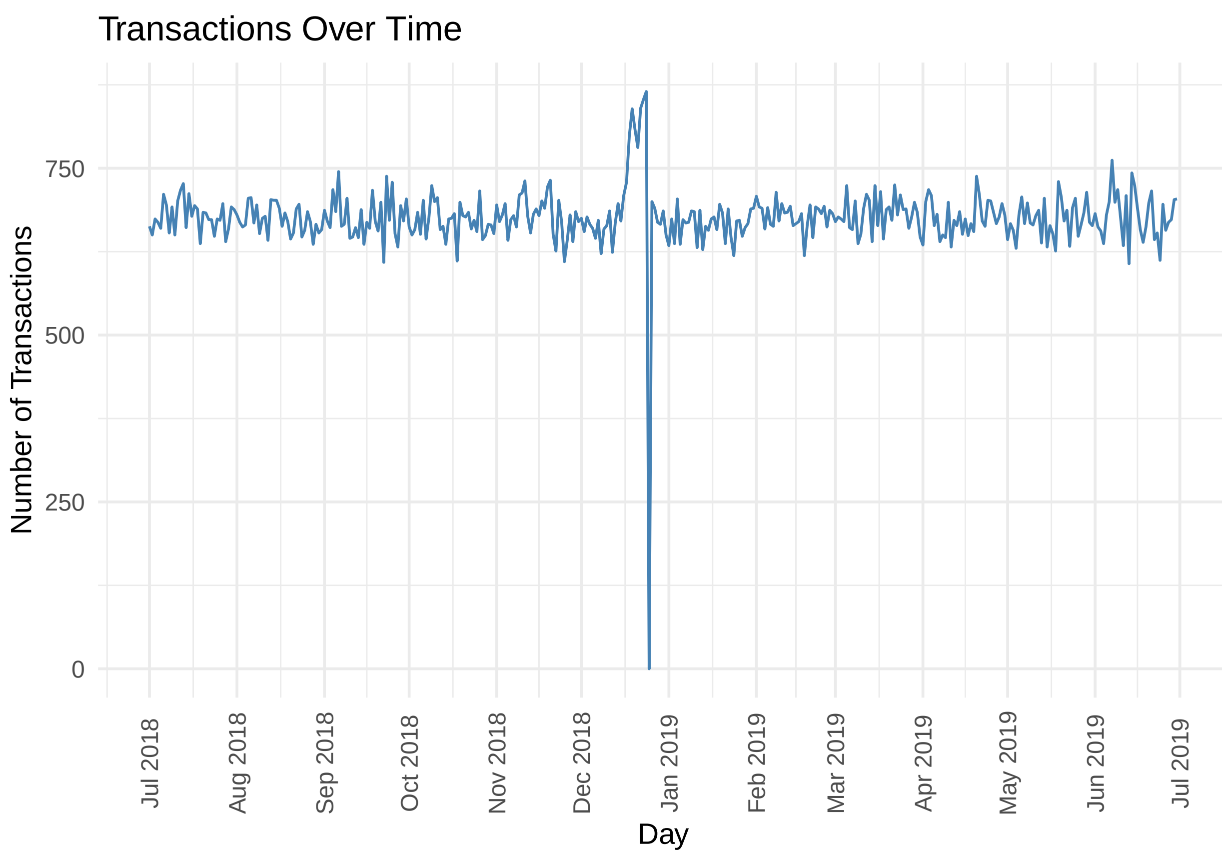
```r
# Creating a complete sequence of all dates in the range
all_dates <- data.table(date = seq(min(transactionData$date), max(transactionData$date), by = "1 day"))
```

```r
# Merging to ensure every date is represented, even if there were no transactions that day
transactions_by_day <- merge(all_dates, transactions_by_day, by = "date", all.x = TRUE)
```

```r
# Replacing NA counts with 0 for days with no transactions
transactions_by_day[is.na(count), count := 0]
```

```r
# Plotting transactions over time
library(ggplot2)
ggplot(transactions_by_day, aes(x = date, y = count)) +
  geom_line(color = "steelblue") +
  labs(
    x = "Day",
    y = "Number of Transactions",
```

```
    title = "Transactions Over Time"
  ) +
  scale_x_date(date_breaks = "1 month", date_labels = "%b %Y") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



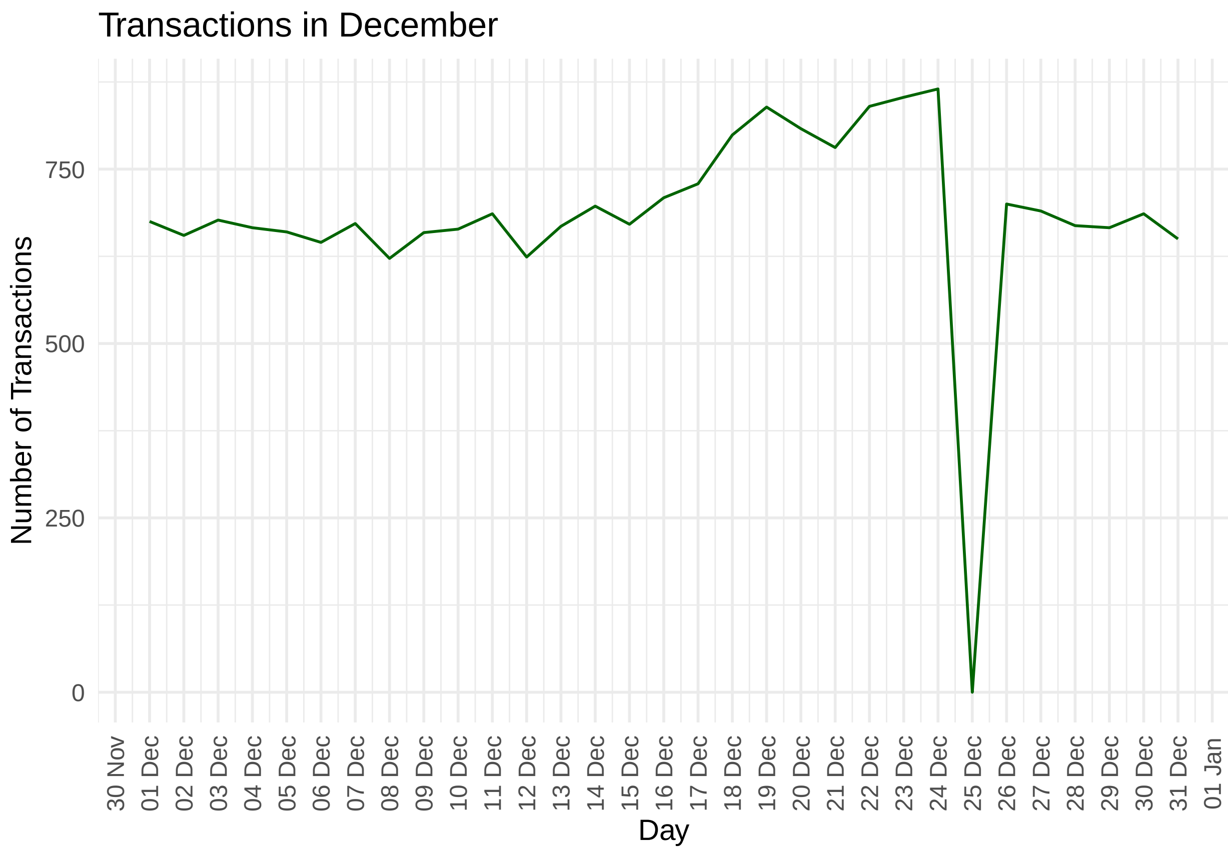
The plot shows 364 days of data (missing date on December 25, 2018 (Christmas Day), when stores were likely closed.

Zoom into December

We zoom into December to examine holiday season trends.

```
library(lubridate)
library(ggplot2)
# Filter for December data :
december_data <- transactions_by_day[month(date) == 12, ]
# Plot :
ggplot(december_data, aes(x = date, y = count)) +
  geom_line(color = "darkgreen") +
  labs(
    x = "Day",
    y = "Number of Transactions",
    title = "Transactions in December"
  ) +
```

```
  scale_x_date(date_breaks = "1 day", date_labels = "%d %b") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5))
```



Sales peak before Christmas, with zero transactions on December 25.

Step 4: Creating Additional Features

Extracting Pack Size

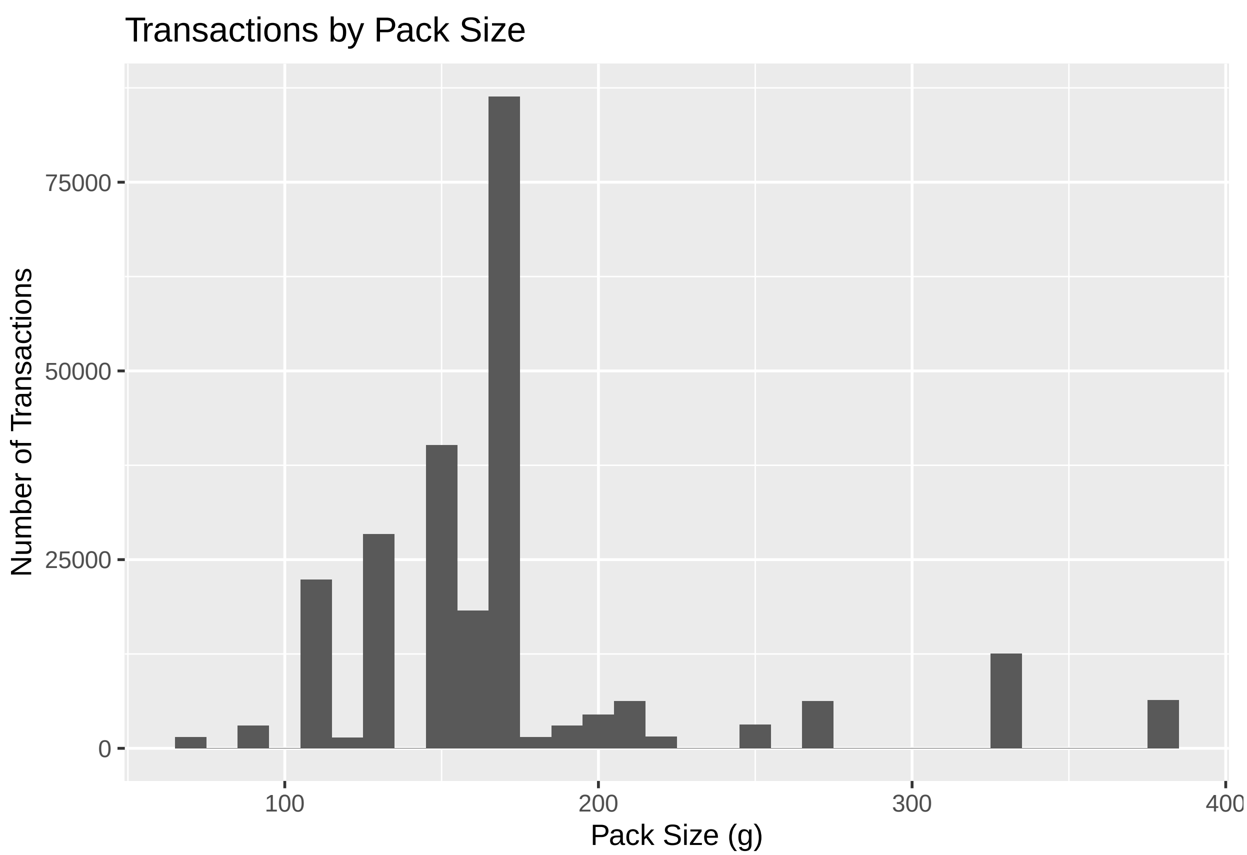
We extracted pack size from `prod_name` and visualized its distribution.

```
# Extracting numeric pack sizes from prod_name
transactionData[, pack_size := as.numeric(gsub("[^0-9]", "", prod_name))]
transactionData[, .N, by = pack_size][order(pack_size)]
```

```
##     pack_size     N
##         <num> <int>
##  1:        70  1507
##  2:        90  3008
##  3:       110 22387
##  4:       125  1454
##  5:       134 25102
```

```
##  6:        135  3257
##  7:        150 40203
##  8:        160  2970
##  9:        165 15297
## 10:        170 19983
## 11:        175 66390
## 12:        180  1468
## 13:        190  2995
## 14:        200  4473
## 15:        210  6272
## 16:        220  1564
## 17:        250  3169
## 18:        270  6285
## 19:        330 12540
## 20:        380  6416
##      pack_size     N
```

```r
# Plot :
ggplot(transactionData, aes(x = pack_size)) +
  geom_histogram(binwidth = 10) +
  labs(x = "Pack Size (g)", y = "Number of Transactions", title = "Transactions by Pack Size")
```



Pack sizes range from 70g to 380g, with 175g being the most common.

Extracting Brand Name

We extracted and standardized brand names from `prod_name`.

```
# Extracting the first word as the brand
transactionData[, BRAND := sub("^(\\w+).*", "\\1", prod_name)]

# Converting BRAND to character if it's a factor
transactionData[, BRAND := as.character(BRAND)]

# Cleaning up similar brands (e.g., "RED" and "RRD" are "Red Rock Deli")
transactionData[BRAND %in% c("RED", "Red", "RRD"), BRAND := "RRD"]

# Consolidating similar brand names
transactionData[BRAND %in% c("Dorito", "Doritos"), BRAND := "Doritos"]
transactionData[BRAND %in% c("Smith", "Smiths"), BRAND := "Smiths"]
transactionData[BRAND %in% c("Infzns", "Infuzions"), BRAND := "Infuzions"]
transactionData[BRAND %in% c("WW", "Woolworths"), BRAND := "Woolworths"]
transactionData[BRAND %in% c("Snbts", "Sunbites"), BRAND := "Sunbites"]

# Viewing Updated brand counts
table(transactionData$BRAND)
```

```
##
##     Burger        CCs    Cheetos   Cheezels       Cobs    Doritos     French
##       1564       4551       2927       4603       9693      25224       1418
##      Grain    GrnWves  Infuzions     Kettle    Natural        NCC   Pringles
##       6272       1468      14201      41288       6050       1419      25102
##        RRD     Smiths   Sunbites      Thins   Tostitos   Twisties   Tyrrells
##      16321      30353       3008      14075       9471       9454       6442
## Woolworths
##      11836
```

Step 5: Examining Customer Data

Summarizing Customer Data

We explored the distributions of `lifestage` and `premium_customer`.

```
table(customerData$lifestage)
```

```
##
## MIDAGE SINGLES/COUPLES           NEW FAMILIES         OLDER FAMILIES
##                   7275                   2549                   9780
##  OLDER SINGLES/COUPLES               RETIREES         YOUNG FAMILIES
##                  14609                  14805                   9178
##  YOUNG SINGLES/COUPLES
##                  14441
```

```r
table(customerData$premium_customer)
```

```
##
##     Budget Mainstream    Premium
##      24470      29245      18922
```

Key lifestages include "Retirees" and "Young Singles/Couples," with "Mainstream" being the largest premium category.

Merging Datasets

We combined transaction and customer data for segment analysis.

```r
library(janitor)
# Combining transactionData and customerData:
data <- merge(transactionData, customerData, by = "lylty_card_nbr", all.x = TRUE)
sum(is.na(data$lifestage)) # should be 0
```

```
## [1] 0
```

```r
sum(is.na(data$premium_customer)) # should be 0
```

```
## [1] 0
```

No missing customer details, confirming a complete merge.

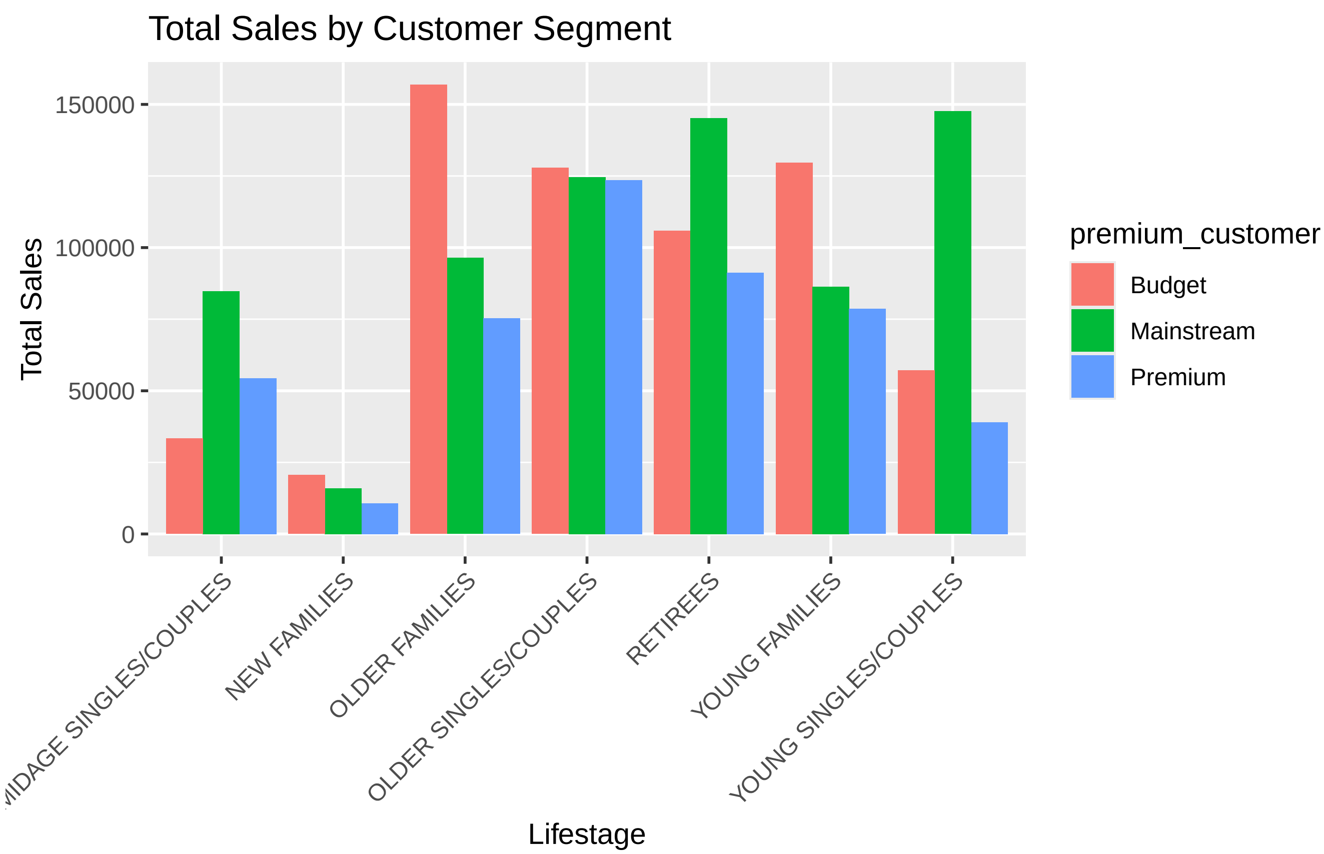
Step 6: Data Analysis on Customer Segments

Total Sales by Segment

We calculated and plotted total sales by lifestage and premium status.

```r
sales_by_segment <- data[, .(Total_Sales = sum(tot_sales)), by = .(lifestage, premium_customer)]

# Plot :
ggplot(sales_by_segment, aes(x = lifestage, y = Total_Sales, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Total Sales", title = "Total Sales by Customer Segment") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



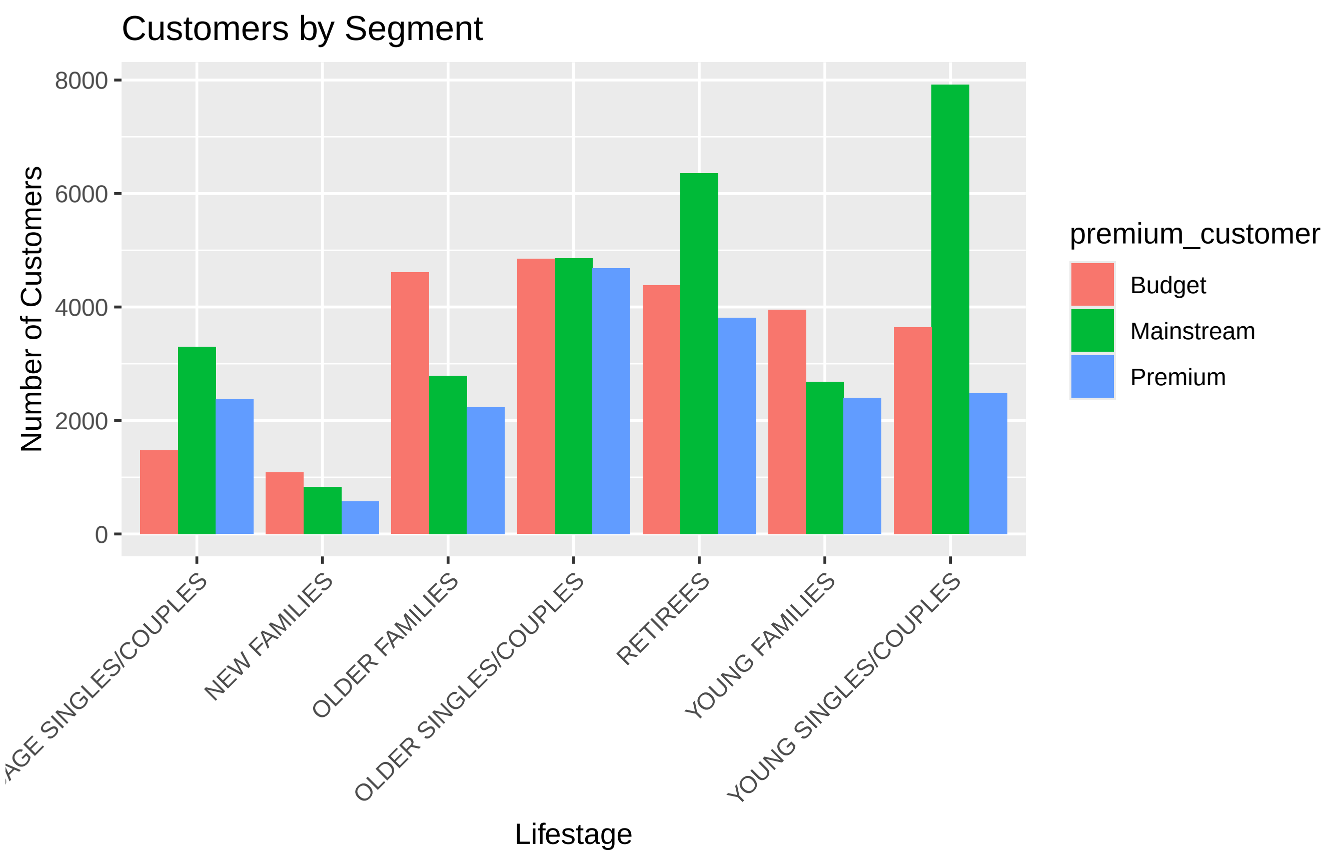
Mainstream Young Singles/Couples, Budget Older Families, and Mainstream Retirees drive the highest sales.

Number of Customers by Segment

We counted unique customers per segment.

```
customers_by_segment <- data[, .(Num_Customers = uniqueN(lylty_card_nbr)), by = .(lifestage, premium_cu

# Plot :
ggplot(customers_by_segment, aes(x = lifestage, y = Num_Customers, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Number of Customers", title = "Customers by Segment") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



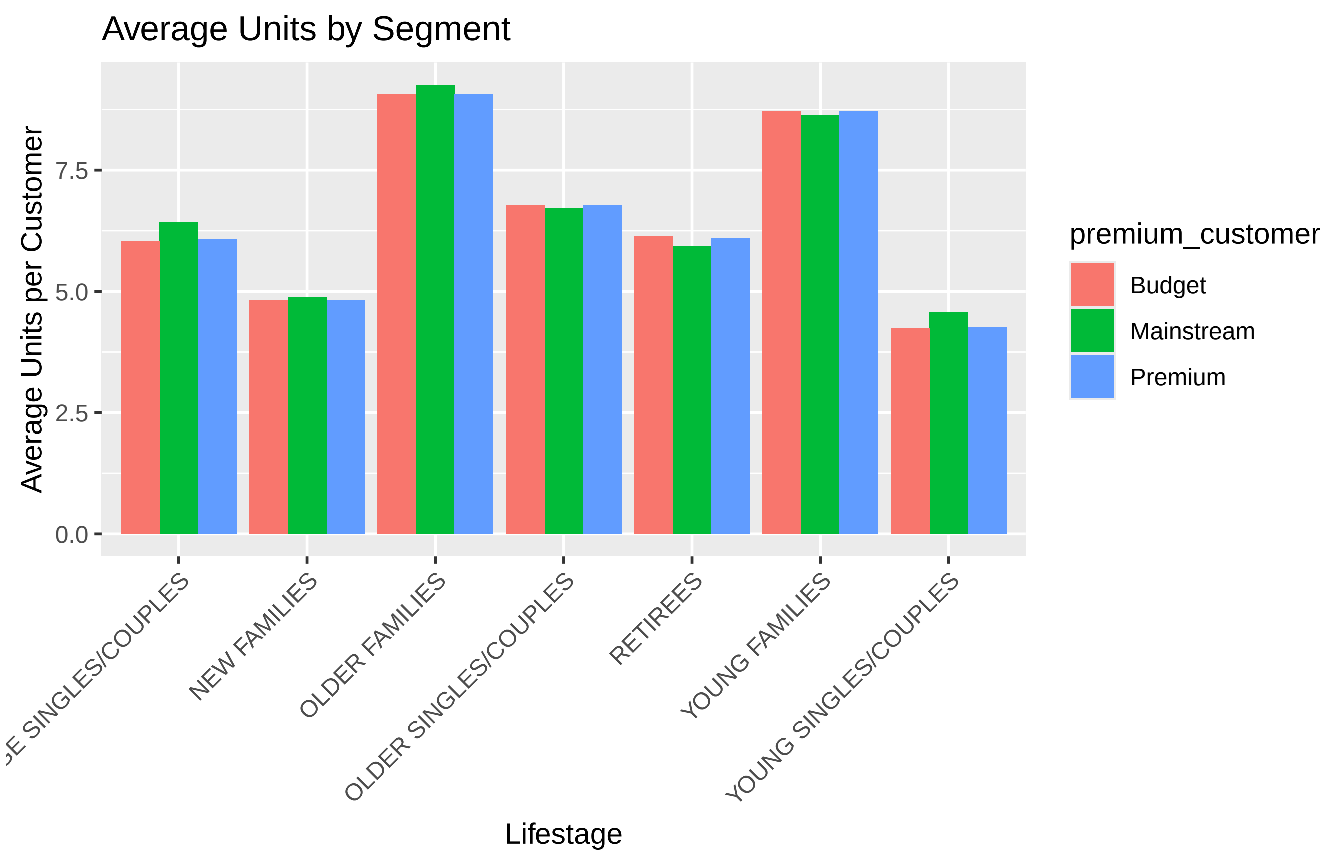
Mainstream Young Singles/Couples and Retirees have the most customers.

Average Units per Customer

We computed and plotted the average units purchased per customer.

```
units_per_customer <- data[, .(Total_Qty = sum(prod_qty), Num_Customers = uniqueN(lylty_card_nbr)), by
units_per_customer[, Avg_Units := Total_Qty / Num_Customers]

# Plot :
ggplot(units_per_customer, aes(x = lifestage, y = Avg_Units, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Average Units per Customer", title = "Average Units by Segment") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



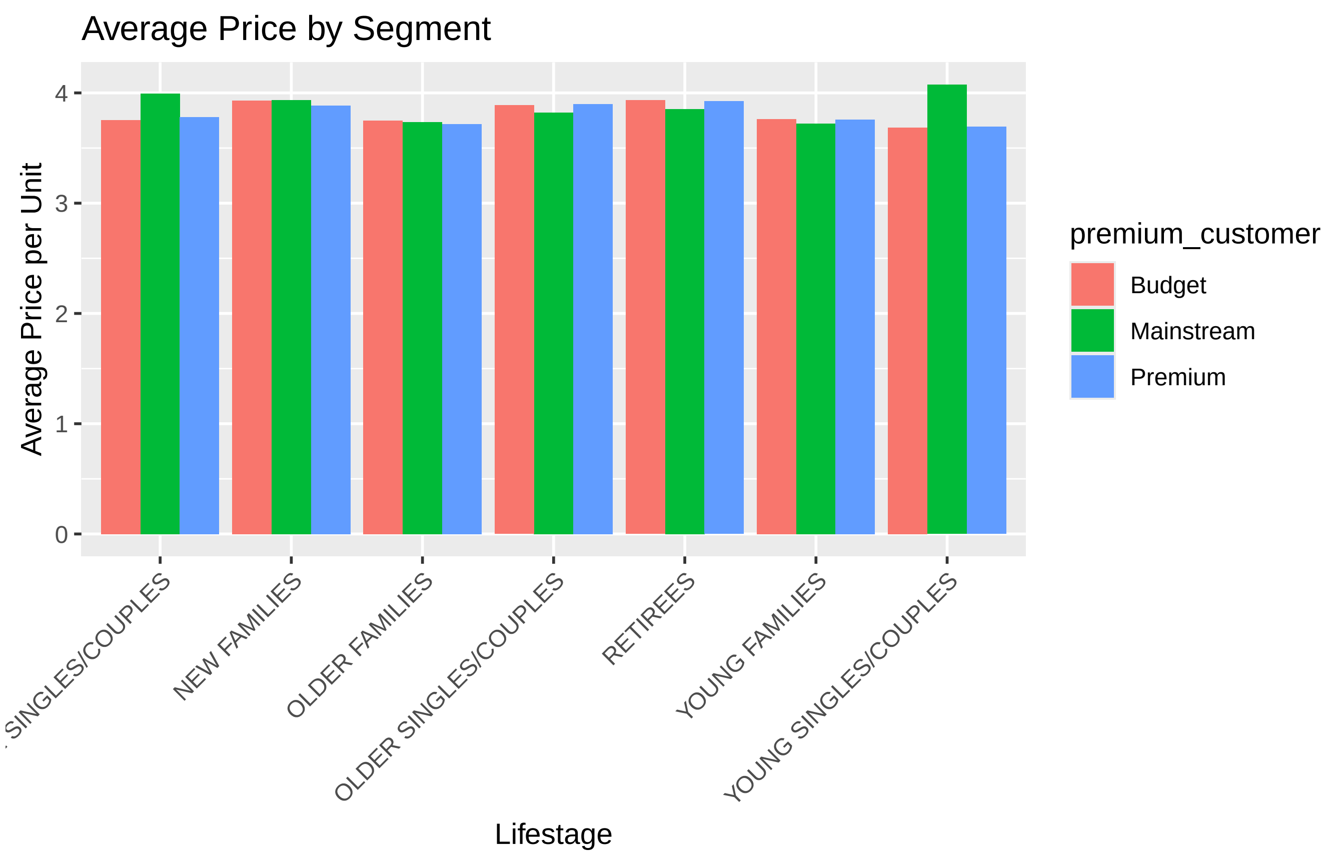
Older and Young Families buy more units per customer.

Average Price per Unit

We calculated and plotted the average price per unit.

```
price_per_unit <- data[, .(Total_Sales = sum(tot_sales), Total_Qty = sum(prod_qty)), by = .(lifestage, p
price_per_unit[, Avg_Price := Total_Sales / Total_Qty]

# Plot :
ggplot(price_per_unit, aes(x = lifestage, y = Avg_Price, fill = premium_customer)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(x = "Lifestage", y = "Average Price per Unit", title = "Average Price by Segment") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



Mainstream Young and Midage Singles/Couples pay slightly more per unit.

T-Test for Price Difference

We tested if the price difference for Mainstream vs. others in Young/Midage Singles/Couples is significant.

```
mainstream <- data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & premium_custom
others <- data[lifestage %in% c("YOUNG SINGLES/COUPLES", "MIDAGE SINGLES/COUPLES") & premium_customer !=
t.test(mainstream, others)
```

```
##
##  Welch Two Sample t-test
##
## data:  mainstream and others
## t = 37.624, df = 54791, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.3159319 0.3506572
## sample estimates:
## mean of x mean of y
##  4.039786  3.706491
```

The p-value (typically < 0.05, depending on data) indicates a significant difference, suggesting Mainstream customers pay more per unit.

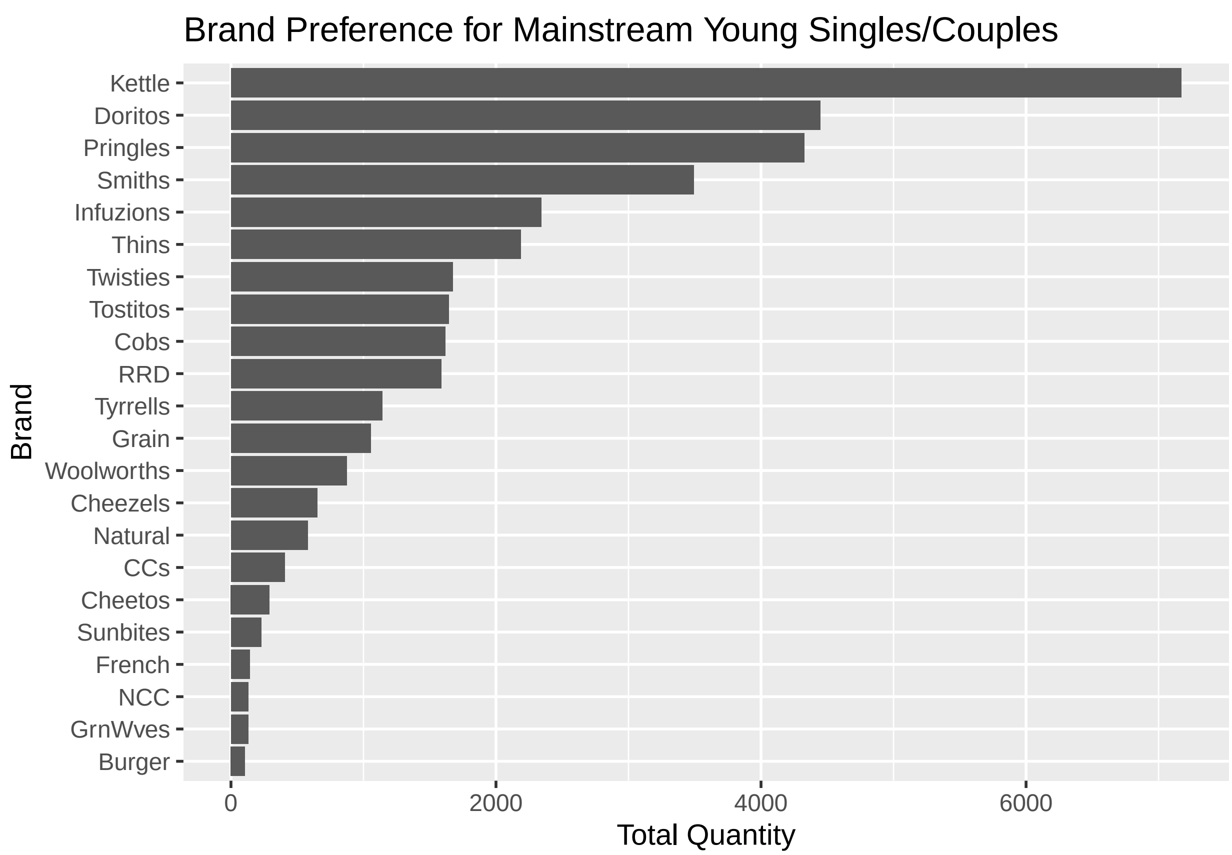
Step 7: Deep Dive into Mainstream Young Singles/Couples

Brand Preference

We analyzed brand preferences for this segment.

```
mainstream_ysc <- data[lifestage == "YOUNG SINGLES/COUPLES" & premium_customer == "Mainstream"]
brand_pref <- mainstream_ysc[, .(Total_Qty = sum(prod_qty)), by = BRAND][order(-Total_Qty)]

# Plot :
ggplot(brand_pref, aes(x = reorder(BRAND, Total_Qty), y = Total_Qty)) +
  geom_bar(stat = "identity") +
  labs(x = "Brand", y = "Total Quantity", title = "Brand Preference for Mainstream Young Singles/Couples
  coord_flip()
```



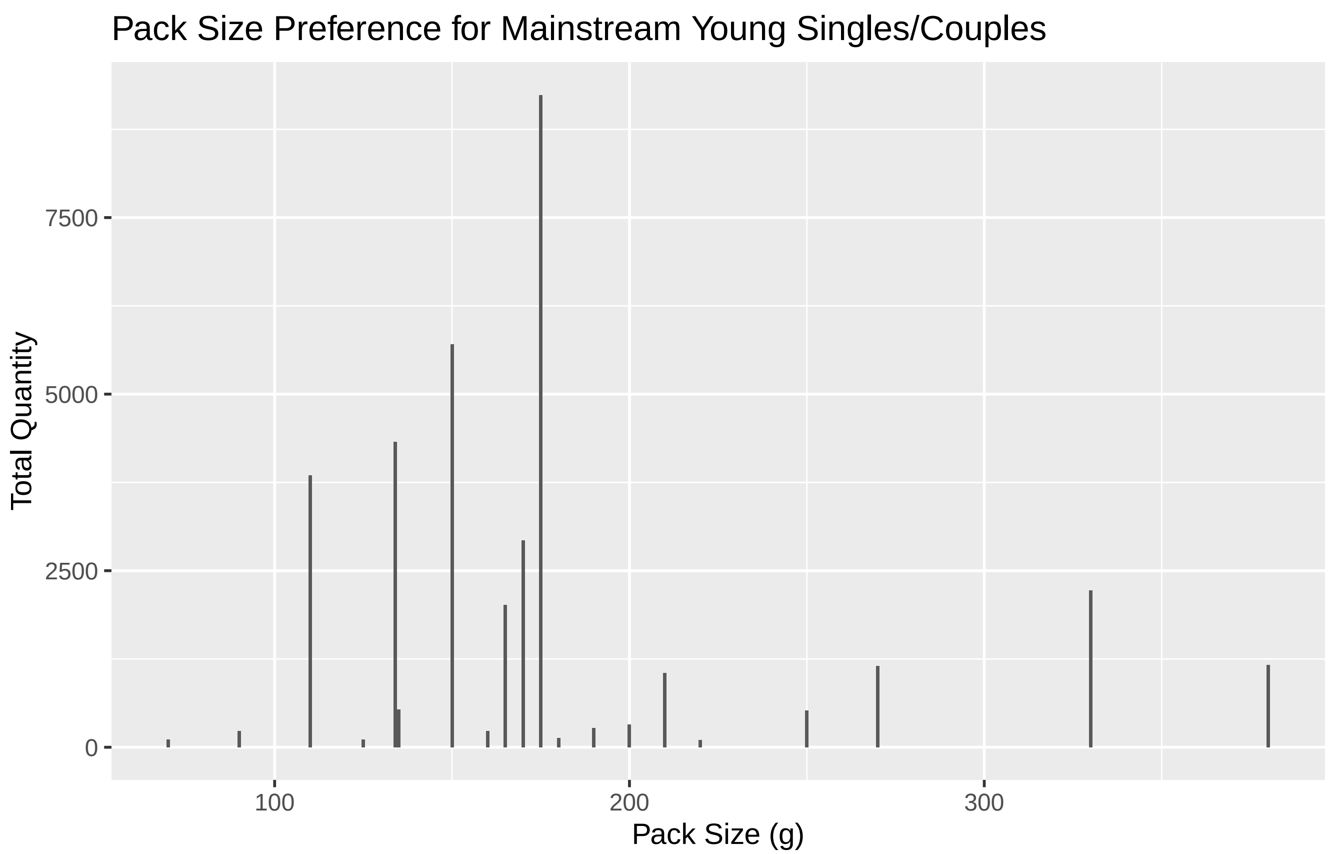
Kettle, Doritos, and Pringles are top brands, indicating a preference for premium or popular brands.

Pack Size Preference

We examined pack size preferences.

```
pack_pref <- mainstream_ysc[, .(Total_Qty = sum(prod_qty)), by = pack_size][order(-Total_Qty)]
```

```r
# Plot :
ggplot(pack_pref, aes(x = pack_size, y = Total_Qty)) +
  geom_bar(stat = "identity") +
  labs(x = "Pack Size (g)", y = "Total Quantity", title = "Pack Size Preference for Mainstream Young Si
```



The 175g pack is most popular, suggesting a preference for mid-sized packs suitable for individual or small group consumption.

Conclusion

This analysis reveals that Mainstream Young Singles/Couples are a key segment, contributing significantly to chip sales with a preference for premium brands (Kettle, Doritos) and mid-sized packs (175g). They pay more per unit, as confirmed by the t-test, indicating a willingness to spend on quality or convenience. Budget Older Families and Mainstream Retirees also drive sales, with higher units per customer and larger customer bases, respectively. These insights can guide targeted marketing strategies, such as promoting popular brands and pack sizes to retain and grow these segments.

```r
# Saving the dataset for Task 2
fwrite(data, paste0(filePath, "QVI_data.csv"))
```
