Optimizing Retail Inventory: A Risk-Based Allocation Framework

Zachary Tang

2026-02-18

Contents

Initialisation of Notebook

Loading Packages

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.1     v stringr   1.6.0
## v ggplot2   4.0.0     v tibble    3.3.0
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.2.0
## -- 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 errors
```

```
library(readxl)
library(ggplot2)
library(dplyr)
library(stringr)
library(lubridate)
library(slider)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

Loading Data

```r
#Loading Transaction Data
QVI_transaction_data <- read_xlsx("C:/Users/Lee Meng/Downloads/QVI_transaction_data.xlsx")

head(QVI_transaction_data)
```

```
## # 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
```

Executive Summary

This project addresses the operational problem of determining optimal inventory levels under uncertain demand.

The objective is to balance: - Service reliability (minimising stockouts) - Capital efficiency (avoiding excess inventory)

A statistical demand model was developed using historical weekly data.
Safety stock was calibrated using volatility estimates and validated via historical backtesting.

The results demonstrate a clear trade-off between service level and capital allocation, allowing decision-makers to select an inventory buffer consistent with their risk appetite.

1. Business Context

Inventory decisions directly affect:

- Customer satisfaction
- Revenue stability
- Working capital allocation
- Operational risk exposure

Holding insufficient stock increases service failure risk.
Holding excessive stock reduces capital efficiency.

The core question becomes:

> What level of safety buffer appropriately balances uncertainty and capital usage?

Cleaning Data

```r
# Imported from Excel, So Changing date origin
QVI_transaction_data$DATE <- as.Date(QVI_transaction_data$DATE, origin = "1899-12-30")

# Checking for any missing values
na.fail(QVI_transaction_data)
```

```
## # A tibble: 264,836 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 Chees~        3
##  3 2019-05-20         1           1343    383       61 Smiths Crinkle ~        2
##  4 2018-08-17         2           2373    974       69 Smiths Chip Thi~        5
##  5 2018-08-18         2           2426   1038      108 Kettle Tortilla~        3
##  6 2019-05-19         4           4074   2982       57 Old El Paso Sal~        1
##  7 2019-05-16         4           4149   3333       16 Smiths Crinkle ~        1
##  8 2019-05-16         4           4196   3539       24 Grain Waves    ~        1
##  9 2018-08-20         5           5026   4525       42 Doritos Corn Ch~        1
## 10 2018-08-18         7           7150   6900       52 Grain Waves Sou~        2
## # i 264,826 more rows
## # i 1 more variable: TOT_SALES <dbl>
```

```r
# Checking for logically consistent data types
sapply(QVI_transaction_data, class)
```

```
##           DATE      STORE_NBR LYLTY_CARD_NBR         TXN_ID       PROD_NBR
##         "Date"      "numeric"      "numeric"      "numeric"      "numeric"
##      PROD_NAME       PROD_QTY      TOT_SALES
##    "character"      "numeric"      "numeric"
```

```r
# Checking for duplicate entries
count(QVI_transaction_data) - count(distinct(QVI_transaction_data))
```

```
##   n
## 1 1
```

```r
QVI_transaction_data %>%
  group_by(DATE, STORE_NBR, LYLTY_CARD_NBR, PROD_NBR, PROD_QTY) %>%
  summarise(count = n()) %>%
  filter(count > 1)
```

```
## `summarise()` has grouped output by 'DATE', 'STORE_NBR', 'LYLTY_CARD_NBR',
## 'PROD_NBR'. You can override using the `.groups` argument.
```

```
## # A tibble: 1 x 6
## # Groups:   DATE, STORE_NBR, LYLTY_CARD_NBR, PROD_NBR [1]
##   DATE       STORE_NBR LYLTY_CARD_NBR PROD_NBR PROD_QTY count
##   <date>         <dbl>          <dbl>    <dbl>    <dbl> <int>
## 1 2018-10-01       107         107024       45        2     2
```

```r
# Removing duplicate
QVI_transaction_data <- distinct(QVI_transaction_data)

# Viewing Summary Statistics
lapply(QVI_transaction_data, summary)
```

```
## $DATE
##         Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
## "2018-07-01" "2018-09-30" "2018-12-30" "2018-12-30" "2019-03-31" "2019-06-30"
##
## $STORE_NBR
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.0    70.0   130.0   135.1   203.0   272.0
##
## $LYLTY_CARD_NBR
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1000   70021  130358  135550  203095 2373711
##
## $TXN_ID
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1   67601  135138  135158  202702 2415841
##
## $PROD_NBR
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00   28.00   56.00   56.58   85.00  114.00
##
## $PROD_NAME
##    Length     Class      Mode
##    264835 character character
##
## $PROD_QTY
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   1.907   2.000 200.000
##
## $TOT_SALES
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.500   5.400   7.400   7.304   9.200 650.000
```

```r
QVI_transaction_data %>%
  arrange(desc(PROD_QTY))
```

```
## # A tibble: 264,835 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
##  3 2018-08-17         2           2373    974       69 Smiths Chip Thi~        5
##  4 2018-08-20         8           8294   8221      114 Kettle Sensatio~        5
##  5 2019-05-16        74          74336  73182       84 GrnWves Plus Bt~        5
##  6 2018-08-19        96          96203  96025        7 Smiths Crinkle ~        5
##  7 2018-08-16        97          97159  97271       35 Woolworths Mild~        5
##  8 2019-05-20       130         130108 134125        2 Cobs Popd Sour ~        5
```

```
##  9 2018-08-19       160         160136 160968       82 Smith Crinkle C~        5
## 10 2018-08-20       186         186401 188768        1 Smiths Crinkle ~        5
## # i 264,825 more rows
## # i 1 more variable: TOT_SALES <dbl>
```

```
# Cases of highest product quantity bought and total sales align. Maximum product quantity and total sa


QVI_transaction_data <- QVI_transaction_data %>%
  filter(PROD_QTY < max(PROD_QTY))

# Removing the non-chip related products (salsa / dips)
Chips_products <- QVI_transaction_data %>%
  filter(str_detect(PROD_NAME, regex("salsa|dip", ignore_case = TRUE), negate = TRUE))
```

Exploratory Data Analysis

Cumulative Product Performance

An initial examination of product-level performance indicates that revenue is primarily volume-driven. Products generating the highest total revenue are consistently those with the largest transaction counts and highest quantities sold.

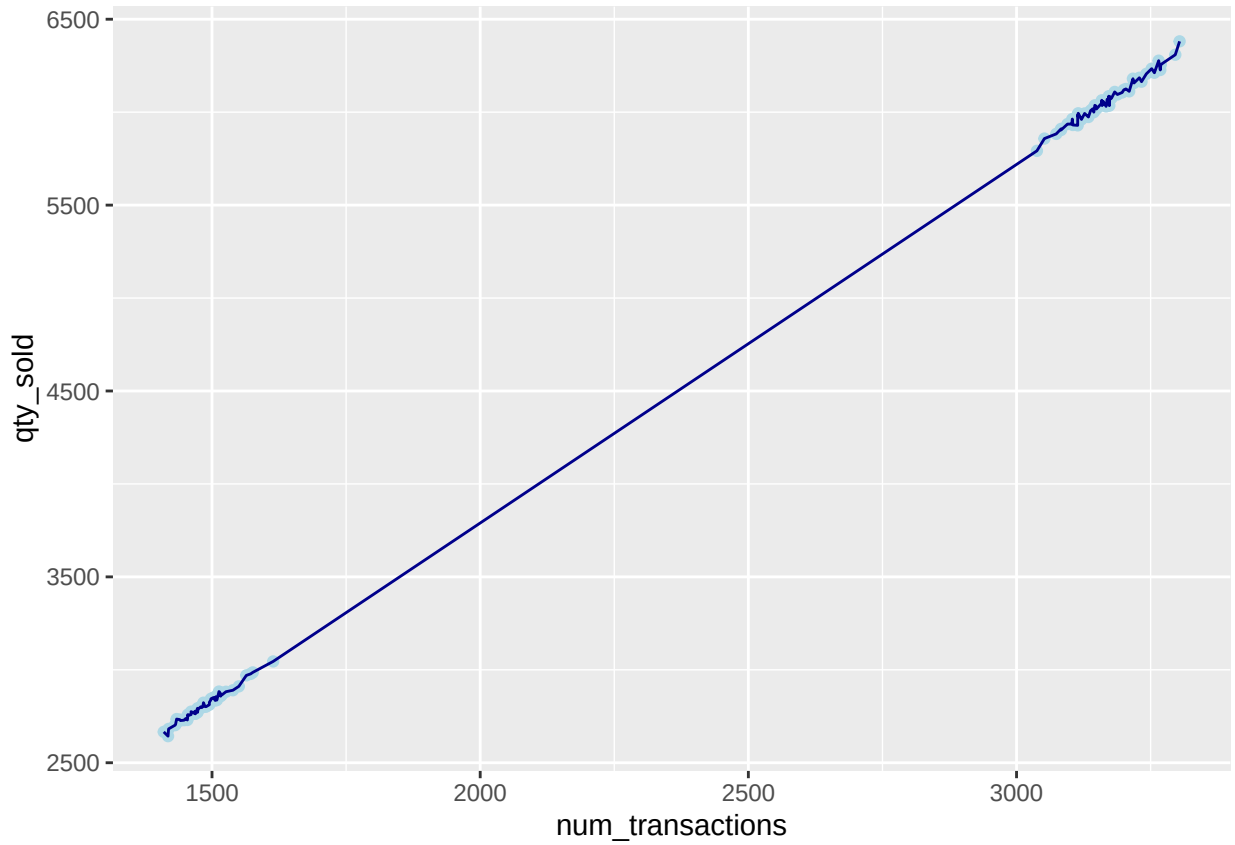
This relationship suggests that revenue performance is largely explained by sales intensity rather than premium pricing alone. In other words, frequent purchase and high turnover are the dominant contributors to total revenue generation.

However, notable dispersion exists among top-performing products. Even within high-volume items, revenue varies meaningfully. This variation is attributable to differences in product-specific pricing structures and cost bases, which influence revenue per unit sold.

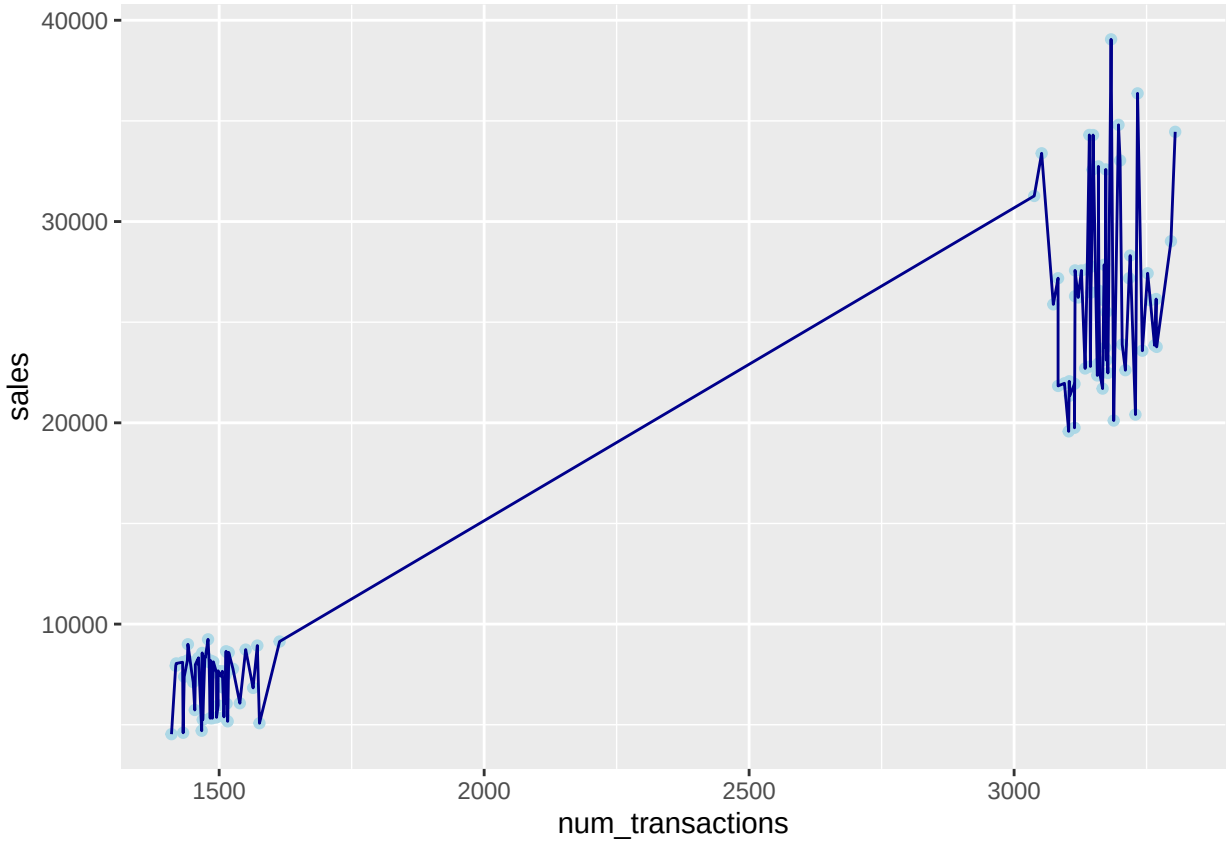
Conclusion:

Revenue concentration is fundamentally a function of transaction frequency and volume, with secondary variation introduced by pricing differentials across products.

```
# ------------------------------------------------------
# Do Transactions Positively Correlate with Quantity ?
# ------------------------------------------------------

transaction_quantity_sales_df <-
  Chips_products %>%
  group_by(PROD_NAME) %>%
  summarise(qty_sold = sum(PROD_QTY),
            num_transactions = n(),
            sales = sum(TOT_SALES))

transaction_quantity_sales_df %>%
  ggplot(mapping = aes(num_transactions, qty_sold)) +
  geom_point(colour = "lightblue") +
  geom_line(colour = "darkblue")
```




```
# ------------------------------------------------------
# Do Transactions Positively Correlate with Sales ?
# ------------------------------------------------------

transaction_quantity_sales_df %>%
  ggplot(mapping = aes(num_transactions, sales)) +
  geom_point(colour = "lightblue") +
  geom_line(colour = "darkblue")
```




```
# ------------------------------------------------------
# What are the Most Popular Product Flavours:
# ------------------------------------------------------

Chips_products <- Chips_products %>%
  mutate(Flavor_Theme = case_when(
    str_detect(PROD_NAME, "(?i)Original|Sea Salt|Lightly Salted|Natural") ~ "Plain & Salted",
    str_detect(PROD_NAME, "(?i)Cheese|Nacho|Cheddar|Camembert|Ricotta|Feta|Mozzarella") ~ "Cheese & Dai
    str_detect(PROD_NAME, "(?i)Sour Cream|Chives|Onion|Herbs") ~ "Sour Cream & Onion",
    str_detect(PROD_NAME, "(?i)Chilli|Chili|Jalapeno|Spicy|Chipotle|Mexicana") ~ "Spicy & Chili",
    str_detect(PROD_NAME, "(?i)Chicken|BBQ|Barbecue|Bacon|Beef|Pork|Snag|Bolognese") ~ "Meat & Savory",
    str_detect(PROD_NAME, "(?i)Vinegar|Vinegr|Lime|Tangy") ~ "Vinegar & Tangy",
    TRUE ~ "Specialty/Other"
  ))

Chips_products %>%
  group_by(Flavor_Theme) %>%
  summarise(sales = sum(TOT_SALES),
            prop = sales / sum(Chips_products$TOT_SALES))%>%
  arrange(desc(sales))
```

```
## # A tibble: 7 x 3
##   Flavor_Theme         sales  prop
##   <chr>                <dbl> <dbl>
## 1 Cheese & Dairy     329055. 0.183
```

```
## 2 Plain & Salted     309692. 0.172
## 3 Meat & Savory      290398. 0.162
## 4 Specialty/Other    270359. 0.150
## 5 Spicy & Chili      226452. 0.126
## 6 Sour Cream & Onion 186954. 0.104
## 7 Vinegar & Tangy    185216. 0.103
```

```
# ------------------------------------------------------
# What are the Most Popular Bag Sizes
# ------------------------------------------------------

Chips_products <- Chips_products %>%
  mutate(
    Weight_g = str_extract(PROD_NAME, "\\d+(?=[gG])"),
    Weight_g = as.numeric(Weight_g),
    Weight_g = as.numeric(Weight_g),
    Weight_Tier = case_when(
    Weight_g <= 175 ~ "Small",
    Weight_g <= 300 ~ "Medium",
    TRUE ~ "Large")
  )

Chips_products %>%
  group_by(Weight_Tier) %>%
  summarise(sales_per_bag_type = sum(TOT_SALES),
            qty_sold_per_bag_type = sum(PROD_QTY),
            ) %>%
  arrange(desc(qty_sold_per_bag_type))
```

```
## # A tibble: 3 x 3
##   Weight_Tier sales_per_bag_type qty_sold_per_bag_type
##   <chr>                    <dbl>                 <dbl>
## 1 Small                1415521.                381503
## 2 Medium                170391.                 49912
## 3 Large                 212214.                 36272
```

Temporal Sales Trends

Daily Sales Volume

Daily sales volume remains broadly stable across the year, fluctuating within a relatively narrow band of approximately 1,200 to 1,400 units per day. This stability indicates a consistent underlying demand for chips products.

Three distinct deviations from this baseline are observed: - A sharp contraction in mid-April, - A second contraction in mid-May, - A pronounced spike at the end of December.

The April and May reductions appear temporary and isolated rather than sustained downturns. The December surge is similarly episodic, likely reflecting heightened seasonal consumption. Outside of these periods, demand reverts to its established baseline range.

Revenue Trends

Given the strong relationship between volume and revenue, revenue patterns closely mirror those of sales volume.

Revenue generally fluctuates between approximately 4,500 and 5,500 per day, maintaining a consistent range throughout the year. The same three temporal deviations—mid-April decline, mid-May decline, and late-December increase—are reflected proportionally in revenue.

This proportional alignment reinforces the earlier conclusion that revenue dynamics are primarily volume-driven rather than price-driven.

Price Segmentation Analysis

To assess whether demand behaviour differs across price tiers, products were segmented into: - The top 25% by price, - The bottom 25% by price, - High-selling products across varying price points.

Across all segments, demand patterns remain highly consistent. Both premium and lower-priced products exhibit: - Similar baseline stability, - Comparable proportional shifts during peak and trough periods, - No evidence of systematic divergence in temporal behaviour.

This suggests that aggregate demand fluctuations are market-wide rather than concentrated within specific pricing strata.

Conclusion:

Temporal sales patterns are consistent across price segments, indicating that demand shocks affect the category broadly rather than selectively.

Sales across flavour themes remain relatively stable over the year. The most popular flavours maintain their ranking position throughout the observation period, even during demand spikes and contractions.

Peak and trough periods scale proportionally across flavour categories rather than materially altering their relative shares. This suggests that fluctuations in total demand reflect overall consumption shifts rather than evolving flavour preferences.

The most dominant flavour themes by volume and revenue are:

- Cheese & Dairy
- Plain & Salted
- Meat & Savoury
- Specialty / Other

These categories consistently account for the majority of total sales.

Bag Size Distribution:

Due to the wide variety of bag sizes, products were grouped into three consolidated categories:

- Small: 175g and under
- Medium: 175g to 300g
- Large: 300g and above

Analysis reveals a highly concentrated distribution:

- Small bags account for over 80% of total sales volume.

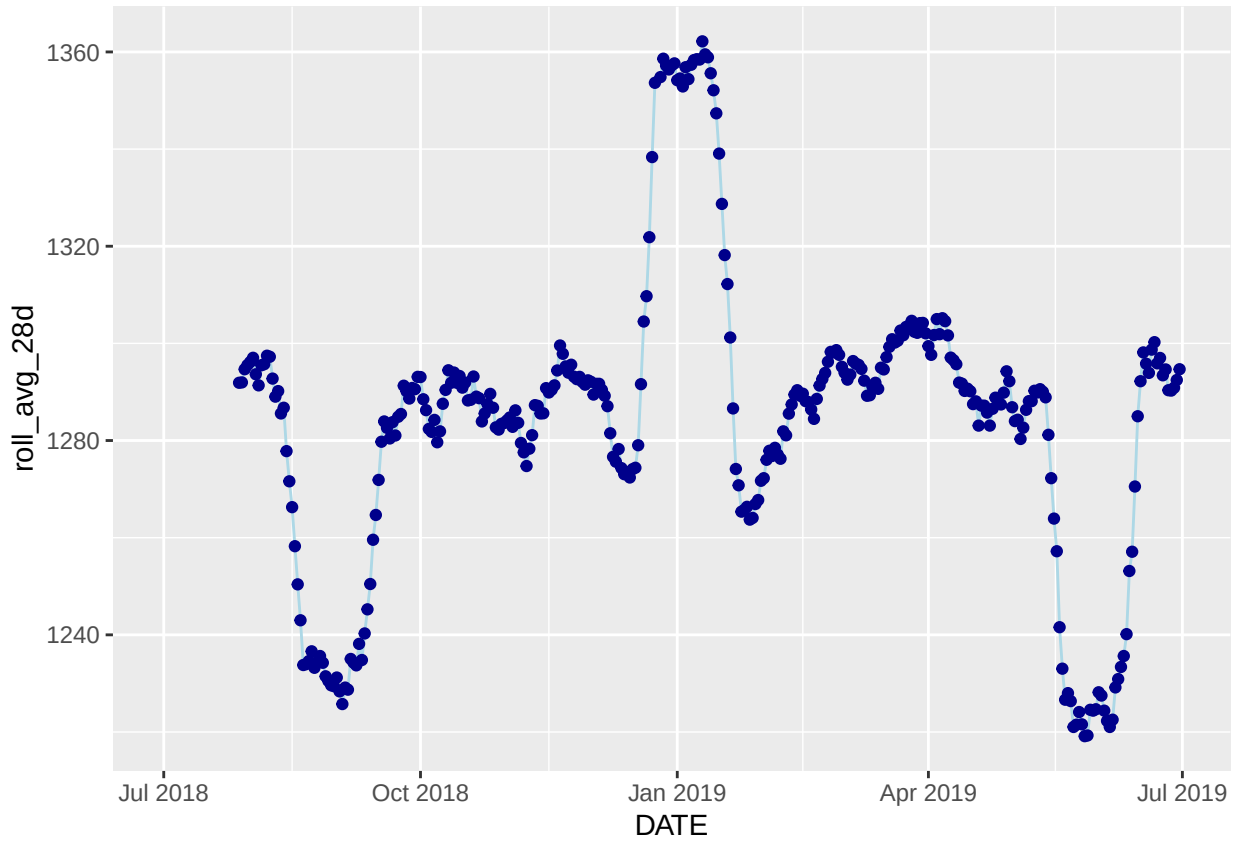
- Medium bags contribute a materially smaller share.
- Large bags represent the smallest proportion of total demand.

Importantly, this proportional structure remains stable throughout the year. Even during periods of elevated or reduced total demand, the dominance of small bags persists.

```
# ------------------------------------------------------
# How Does the Quantity of Chips Sold Vary Over Time ?
# ------------------------------------------------------

Chips_products %>%
  group_by(DATE) %>%
  summarise(qty_sold = sum(PROD_QTY)) %>%
  mutate(roll_avg_28d = slide_dbl(
    .x = qty_sold,
    .f = mean,
    .before = 27,
    .complete = TRUE
  )) %>%
  ggplot(mapping = aes(DATE, roll_avg_28d)) +
  geom_line(colour = "lightblue") +
  geom_point(colour = "darkblue")
```

```
## Warning: Removed 27 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

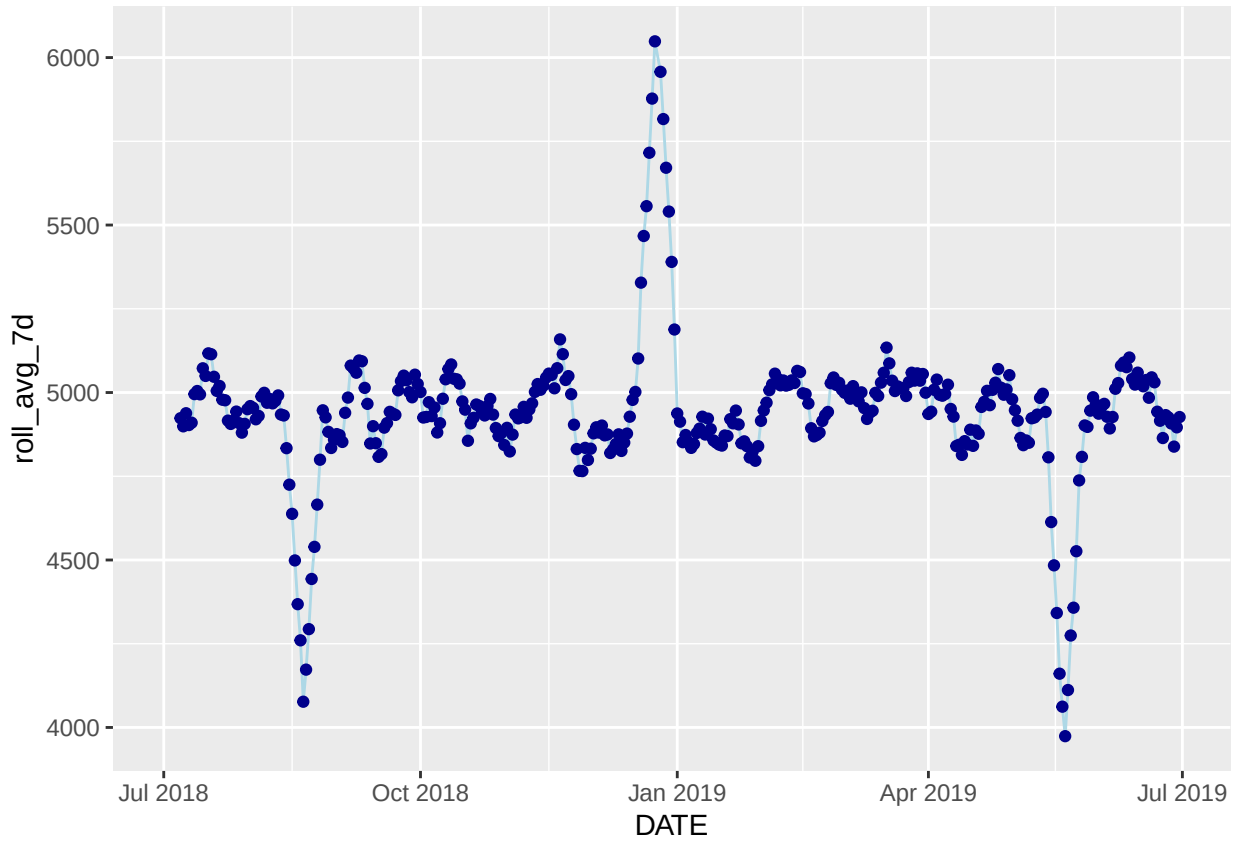
```
## Warning: Removed 27 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



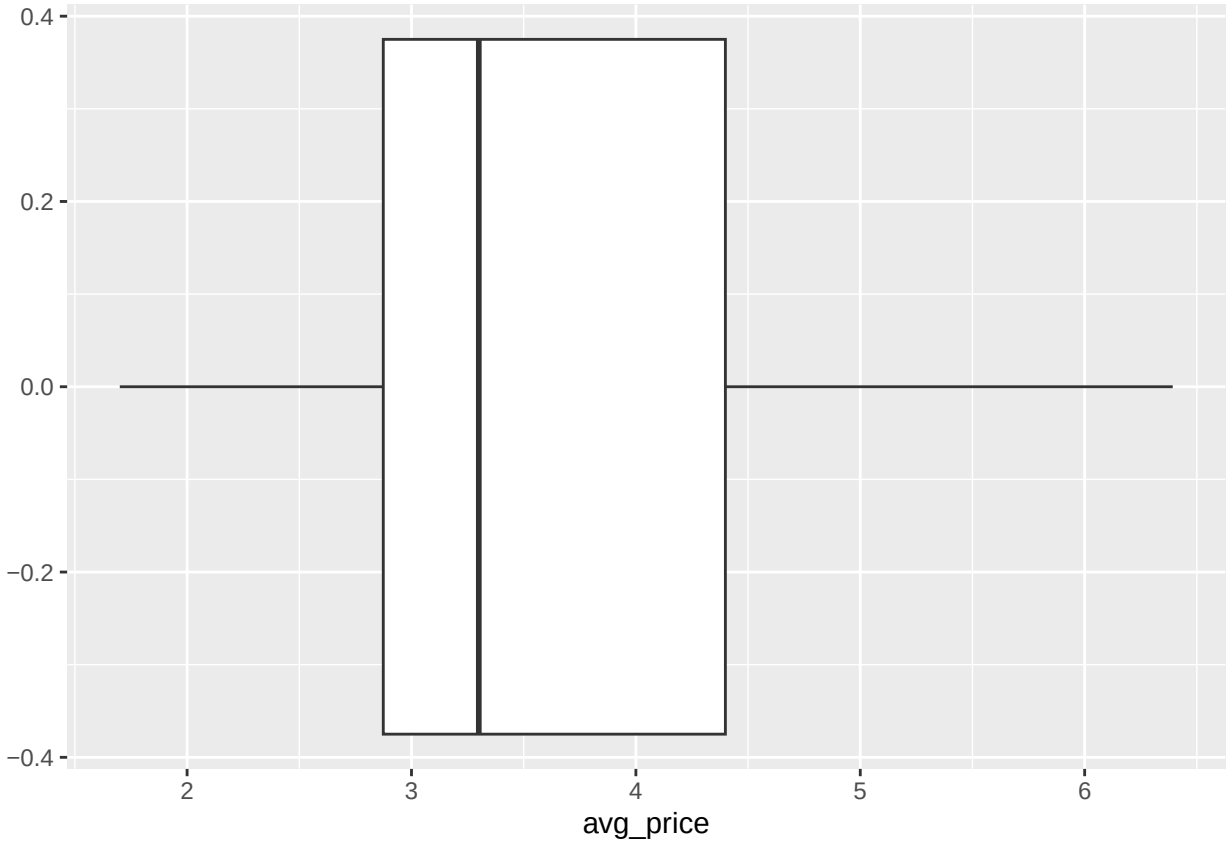

```
# ------------------------------------------------------
# How Do the Sales of Chips Sold Vary Over Time ?
# ------------------------------------------------------

Chips_products %>%
  group_by(DATE) %>%
  summarise(sales = sum(TOT_SALES)) %>%
  mutate(roll_avg_7d = slide_dbl(
    .x = sales,
    .f = mean,
    .before = 6,
    .complete = TRUE
  )) %>%
  ggplot(mapping = aes(DATE, roll_avg_7d)) +
  geom_line(colour = "lightblue") +
  geom_point(colour = "darkblue")
```

```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

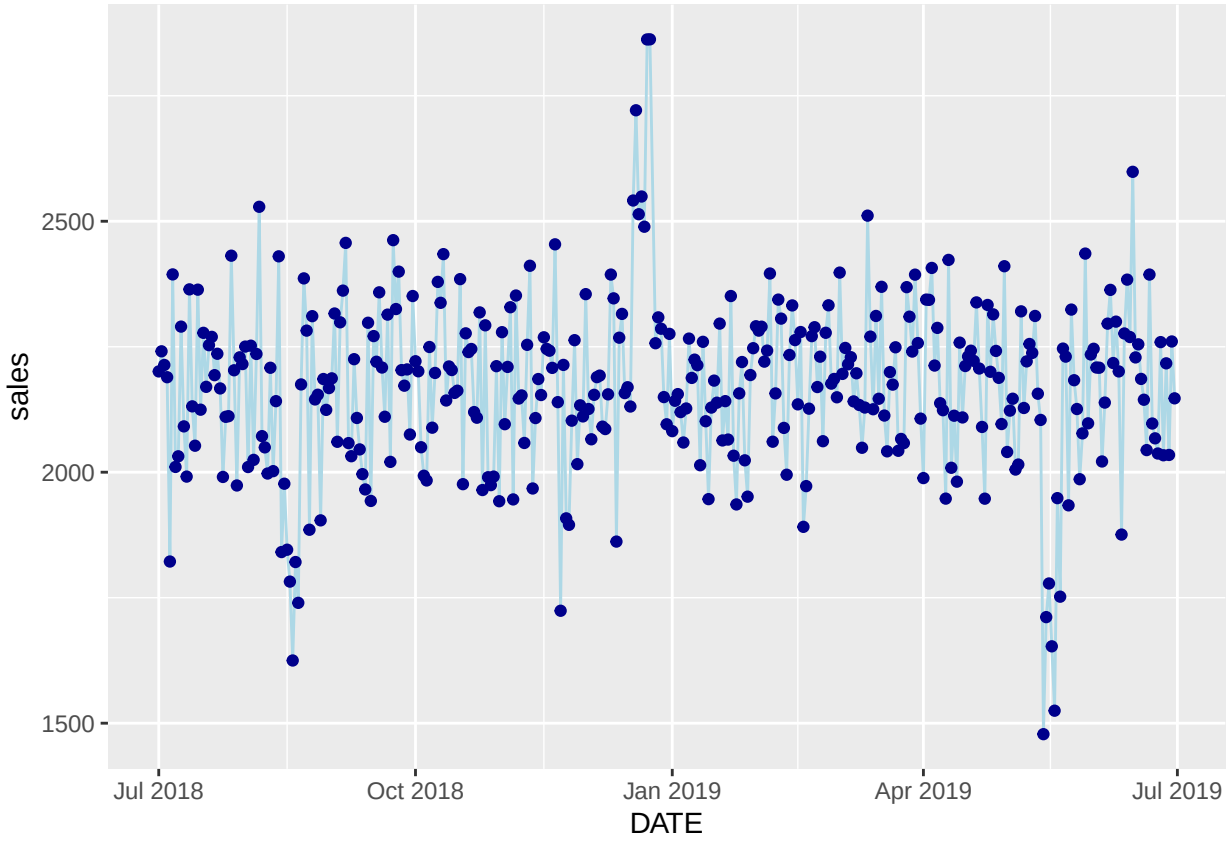
```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



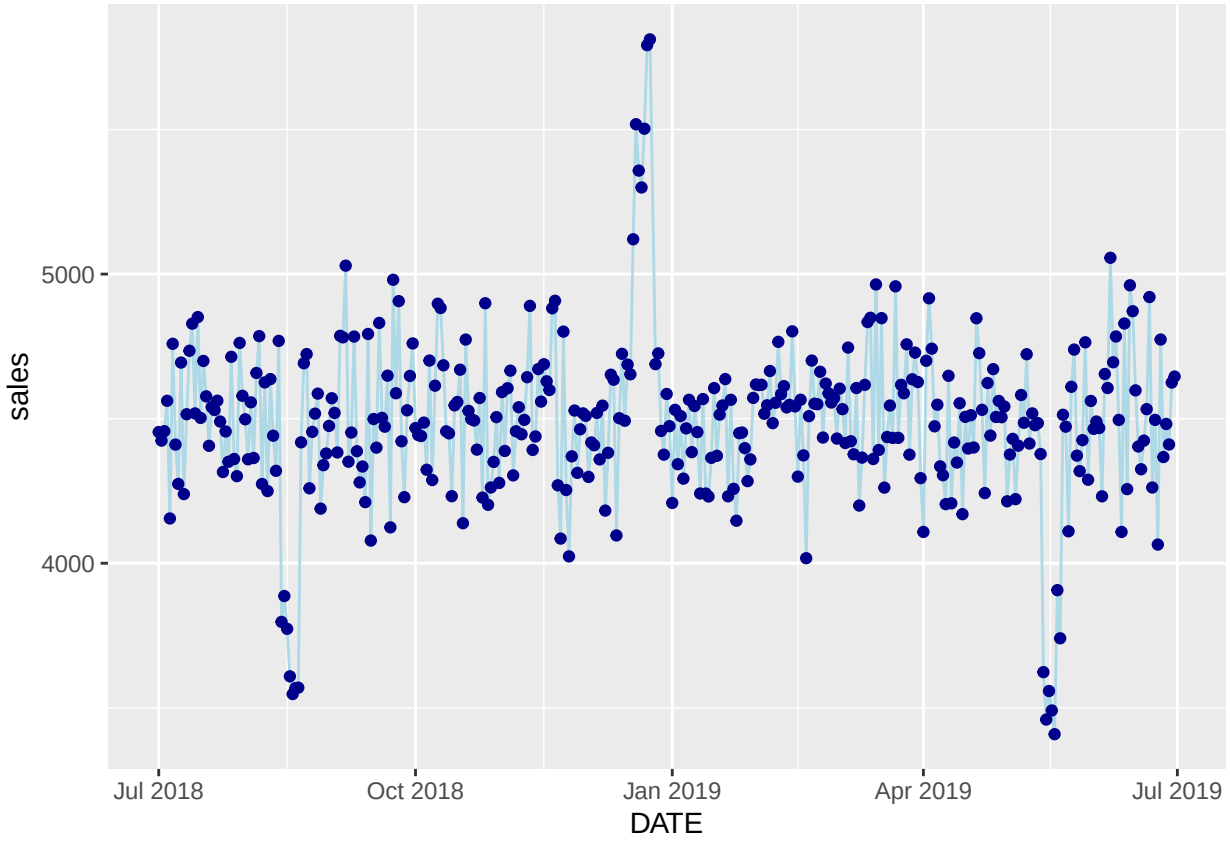

```
# ----------------------------------------------------
# How do the Sales of the Priciest Products Vary Over Time
# ----------------------------------------------------

Chips_products %>%
  group_by(PROD_NAME) %>%
  summarise(avg_price = sum(TOT_SALES) / sum(PROD_QTY)) %>%
  ggplot(mapping = aes(avg_price)) +
  geom_boxplot()
```




```
summary_stats_avg_price <- Chips_products %>%
  group_by(PROD_NAME) %>%
  summarise(avg_price = sum(TOT_SALES) / sum(PROD_QTY)) %>%
  arrange(desc(avg_price)) %>%
  summary(avg_price)

top_25pc <- Chips_products %>% group_by(PROD_NAME) %>%
  summarise(avg_price = sum(TOT_SALES) / sum(PROD_QTY)) %>%
  filter(avg_price >= 4.399)

Chips_products %>%
  filter(PROD_NAME %in% top_25pc$PROD_NAME) %>%
  group_by(DATE) %>%
  summarise(sales = sum(TOT_SALES)) %>%
  mutate(Roll_avg_28d_Sales = slide_dbl(
    .x = sales,
    .f = mean,
    .before = 27,
    .complete = TRUE
  )) %>%
  ggplot(mapping = aes(DATE,sales)) +
  geom_line(colour = "lightblue") +
  geom_point(colour = "darkblue")
```

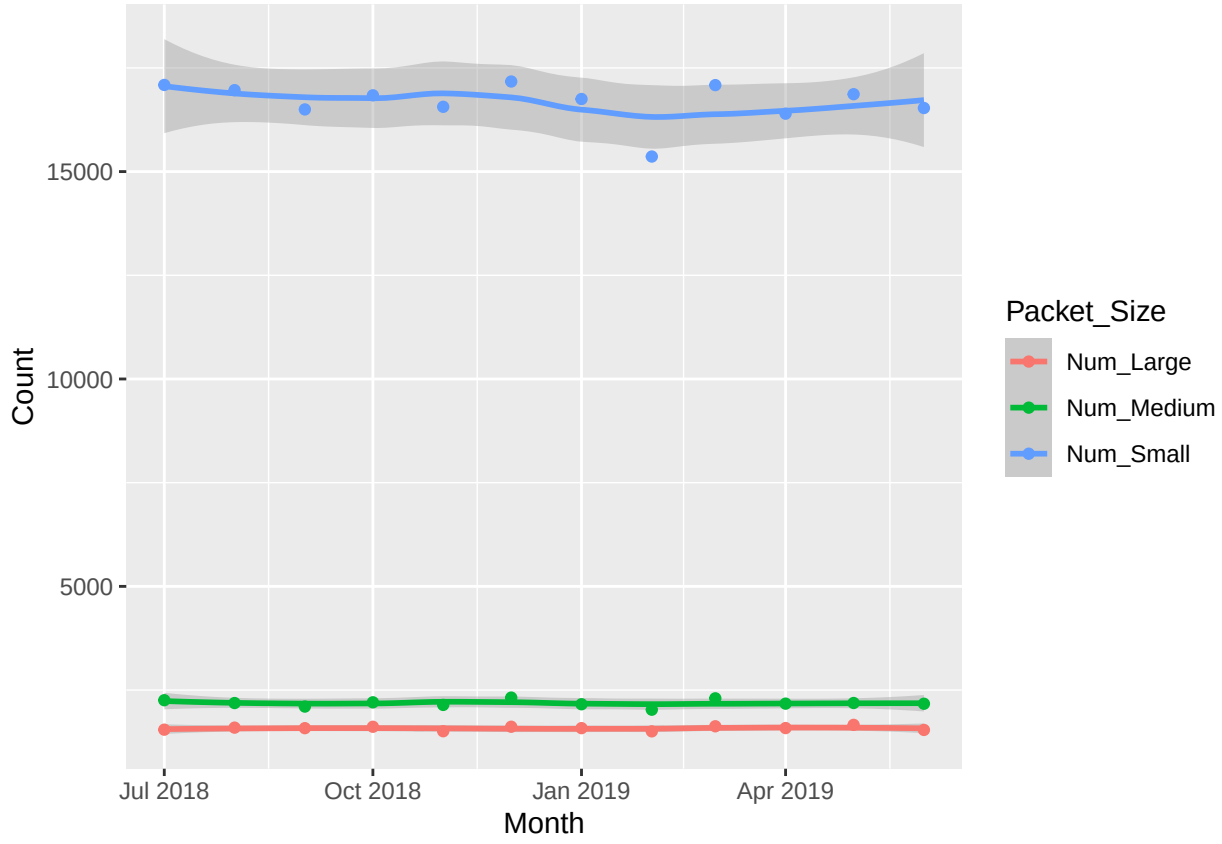



```
# ----------------------------------------------------
# How do the Sales of the Cheapest Products Vary Over Time
# ----------------------------------------------------

bottom_25pc <- Chips_products %>% group_by(PROD_NAME) %>%
  summarise(avg_price = sum(TOT_SALES) / sum(PROD_QTY)) %>%
  filter(avg_price >= 2.874)

Chips_products %>%
  filter(PROD_NAME %in% bottom_25pc$PROD_NAME) %>%
  group_by(DATE) %>%
  summarise(sales = sum(TOT_SALES)) %>%
  mutate(Roll_avg_28d_Sales = slide_dbl(
    .x = sales,
    .f = mean,
    .before = 27,
    .complete = TRUE
  )) %>%
  ggplot(mapping = aes(DATE,sales)) +
  geom_line(colour = "lightblue") +
  geom_point(colour = "darkblue")
```



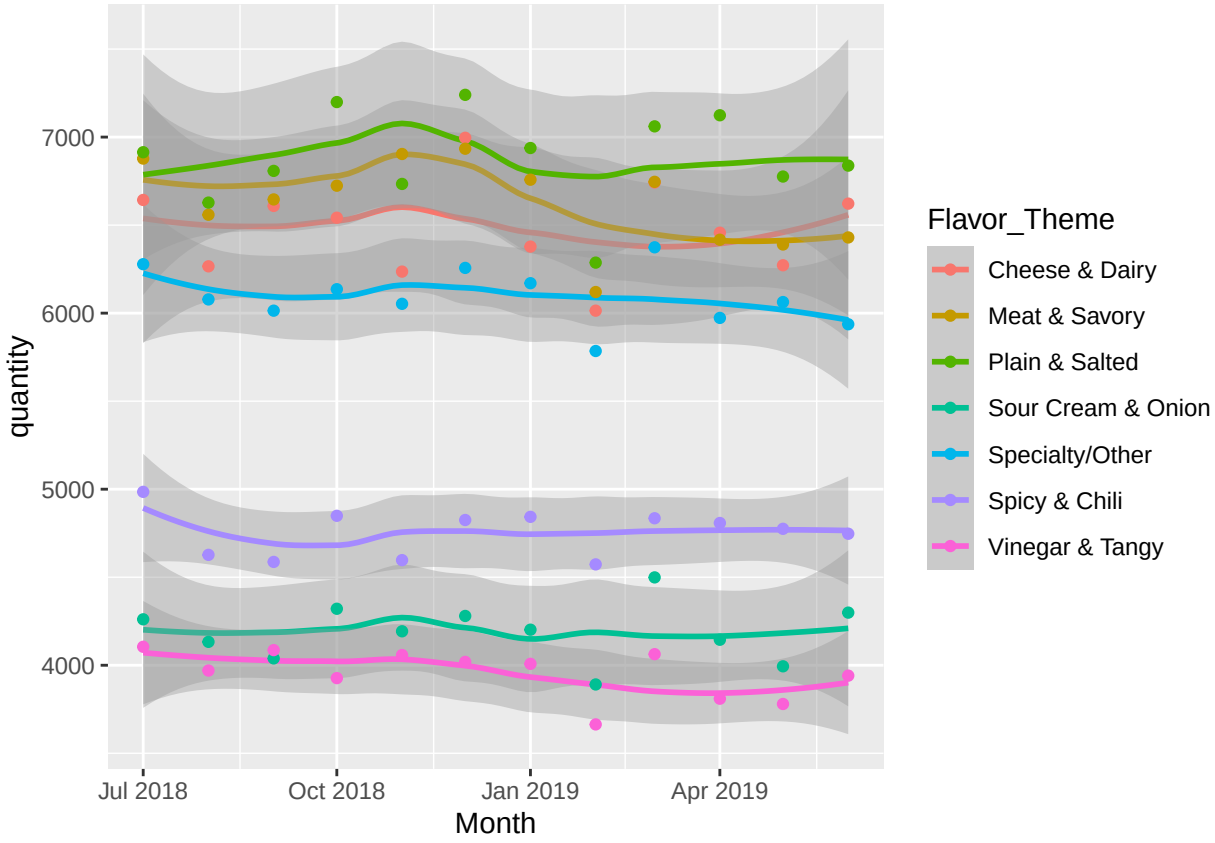

```
# ------------------------------------------------------
# Monthly Sales Trend Over Time of Bag Types
# ------------------------------------------------------

Chips_products %>%
  mutate(Month = floor_date(DATE, unit = "month"),
         Weight_g = str_extract(PROD_NAME, "\\d+(?=[gG])"),
         Weight_g = as.numeric(Weight_g),
         Weight_Tier = case_when(
    Weight_g <= 175 ~ "Small",
    Weight_g <= 300 ~ "Medium",
    TRUE ~ "Large")) %>%
  group_by(Month) %>%
  summarise(Num_Small = sum(Weight_Tier == "Small"),
            Num_Medium = sum(Weight_Tier == "Medium"),
            Num_Large = sum(Weight_Tier == "Large")) %>%
  arrange() %>%
  pivot_longer(
    cols = starts_with("Num_"),
    names_to = "Packet_Size",
    values_to = "Count"
  ) %>%
  ggplot(mapping = aes(Month, Count, colour = Packet_Size)) +
  geom_smooth() +
  geom_point()
```

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




```
# ------------------------------------------------------
# Monthly Sales Trend Over Time of Flavour Themes
# ------------------------------------------------------

Chips_products %>%
  mutate(Month = floor_date(DATE, unit = "month")) %>%
  group_by(Month, Flavor_Theme) %>%
  summarise(quantity = sum(PROD_QTY)) %>%
  ggplot(mapping = aes(Month, quantity, colour = Flavor_Theme)) +
  geom_smooth() +
  geom_point()
```

```
## `summarise()` has grouped output by 'Month'. You can override using the
## `.groups` argument.
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



Inventory Strategy and Policy Recommendations

Two broad approaches can be adopted when determining optimal inventory levels: a static policy, where total inventory remains constant across weeks, and a dynamic policy, where inventory levels are adjusted over time in response to evolving demand conditions.

The selection between these approaches depends fundamentally on whether demand exhibits structural variation (e.g. trend or seasonality) or whether fluctuations are primarily random shocks around a stable mean.

A visual inspection of weekly demand indicates a largely stable trajectory across the year. While isolated deviations are observed (notably during mid-April, mid-May, and late December), these appear to be episodic shocks rather than persistent structural shifts. Importantly:

- A simple linear regression of weekly demand on time shows no statistically significant trend.
- The fitted model is not materially different from an intercept-only specification.
- Residual diagnostics indicate deviations are sporadic rather than systematic.

This evidence suggests that weekly demand is approximately stationary over the observed period. In the absence of sustained trend or strong seasonal patterns and given that only one year of data is available, a static inventory policy is both statistically defensible and operationally simpler.

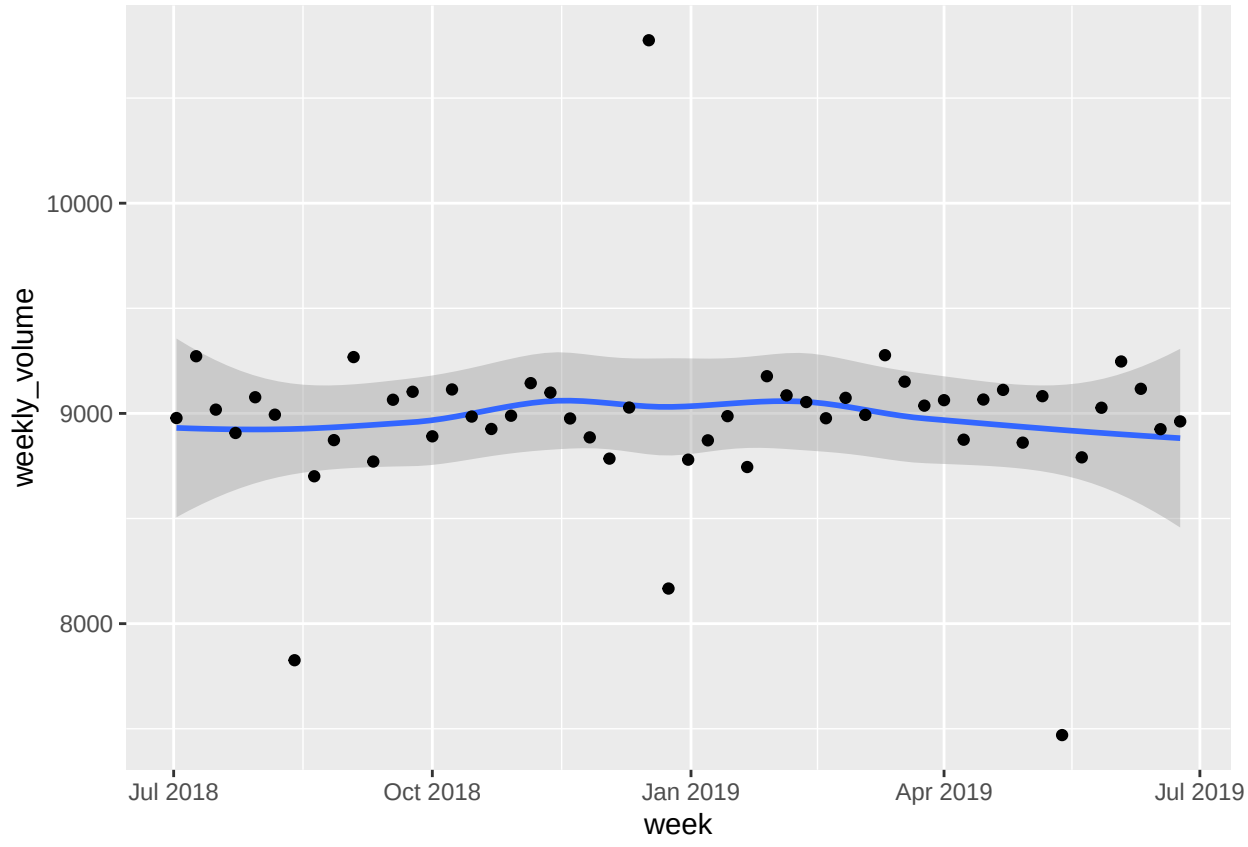
Conclusion:

Weekly demand is largely stable with no significant trend; deviations are isolated shocks.

Static inventory policy is recommended: simple, operationally efficient, and supported by the data. Baseline weekly demand can be set using the mean from historical weekly volumes.

```
data_weekly <- Chips_products %>%
  mutate(week = floor_date(DATE, unit = "week", week_start = 1)) %>%
  group_by(week) %>%
  summarise(weekly_volume = sum(PROD_QTY)) %>%
  filter(weekly_volume > 2000)

Chips_products %>%
  mutate(week = floor_date(DATE, unit = "week", week_start = 1)) %>%
  group_by(week) %>%
  summarise(weekly_volume = sum(PROD_QTY)) %>%
  filter(weekly_volume > 2000) %>%
  ggplot(mapping = aes(x = week, y = weekly_volume)) +
  geom_smooth() +
  geom_point()
```

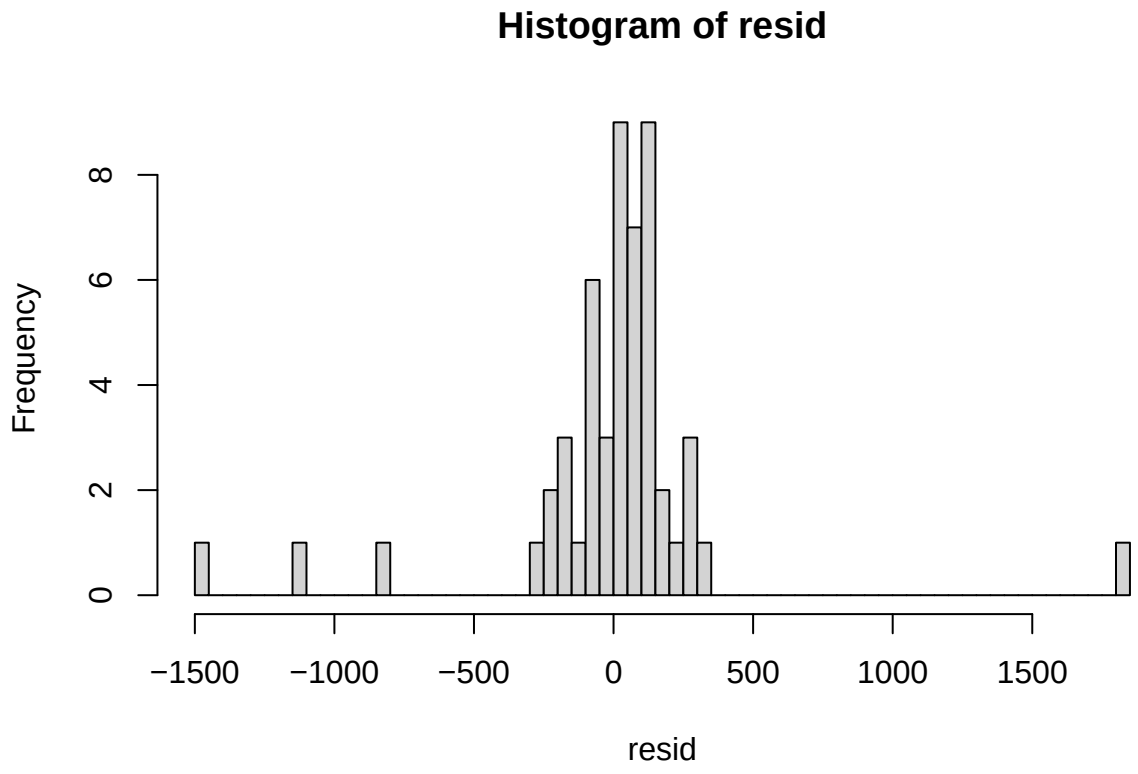
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
model <- lm(weekly_volume ~ week, data = data_weekly)

summary(model) # No statistically significant difference from purely using an intercept
```

```
##
## Call:
## lm(formula = weekly_volume ~ week, data = data_weekly)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1497.54   -86.45    23.15   119.80  1805.10
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 9256.69687 9703.04141   0.954    0.345
## week          -0.01604    0.54229  -0.030    0.977
##
## Residual standard error: 410.8 on 50 degrees of freedom
## Multiple R-squared:  1.749e-05,  Adjusted R-squared:  -0.01998
## F-statistic: 0.0008747 on 1 and 50 DF,  p-value: 0.9765
```

```r
resid <- residuals(model)

hist(resid, breaks = 50)
```



Given the stability of aggregate demand and product mix, baseline weekly demand was estimated using an intercept-only linear model.The intercept serves as the expected weekly demand, this provides a stable central estimate of weekly volume under a static policy framework.

```r
intercept_model <- lm(weekly_volume ~ 1, data = data_weekly)

baseline_demand <- as.numeric(intercept_model$coefficients["(Intercept)"])
```

Stability of Product Mix Over Time

Having established stability at the aggregate level, the next step was to examine whether the composition of demand across bag-sizes and flavour-themes varies meaningfully over time.

Visual inspection of weekly and monthly trends across both dimensions suggests that:

- Relative flavour shares remain broadly consistent throughout the year.
- Bag size distribution (Small, Medium, Large) exhibits stable proportions.
- Peaks and troughs in total demand are reflected proportionally across categories.

Statistical testing further supports this conclusion, as chi-squared tests fail to detect significant changes in category proportions over time. This suggests that fluctuations in total demand appear to be driven by general market demand for chips, rather than shifts in specific product categories.

Therefore, inventory variation does not need to be driven by shifting product mix. Instead, risk management should focus on overall volume variability and cross-category interaction effects.

Conclusion

Individually, flavour and bag-size composition is consistent across the year. Furthermore, the peaks and troughs in total demand affect all categories proportionally.

Inventory adjustments do not need to account for shifting product mix throughout the year, focus on overall volume and key interactions between product categories, such as: flavour and bag size.

```r
# ------------------------------------------------------
# Stocking Arrangements for Bag Sizes
# ------------------------------------------------------

Contingency_Table_Size <- Chips_products %>%
  mutate(week = floor_date(DATE ,unit = "week", week_start = 1)) %>%
  group_by(week) %>%
  summarise(Num_Small = sum(Weight_Tier == "Small"),
            Num_Medium = sum(Weight_Tier == "Medium"),
            Num_Large = sum(Weight_Tier == "Large"))

chisq.test(Contingency_Table_Size[,-1])
```

```
##
##  Pearson's Chi-squared test
##
## data:  Contingency_Table_Size[, -1]
## X-squared = 85.697, df = 104, p-value = 0.9041
```

```r
# ------------------------------------------------------
# Stocking Arrangements for Flavour Themes
# ------------------------------------------------------

Chips_products %>%
  group_by(Weight_Tier) %>%
  summarise(sales = sum(PROD_QTY),
            prop = sales / sum(Chips_products$PROD_QTY))
```

```
## # A tibble: 3 x 3
##   Weight_Tier  sales   prop
##   <chr>        <dbl>  <dbl>
## 1 Large        36272 0.0776
## 2 Medium       49912 0.107
## 3 Small       381503 0.816
```

```r
Contingency_Table_Flavour <- Chips_products %>%
  mutate(week = floor_date(DATE ,unit = "week", week_start = 1)) %>%
  group_by(week) %>%
  summarise(Cheese_Dairy = sum(Flavor_Theme == "Cheese & Dairy"),
            Meat_Savoury = sum(Flavor_Theme == "Meat & Savory"),
            Plain_Salted = sum(Flavor_Theme == "Plain & Salted"),
            SC_Onion = sum(Flavor_Theme == "Sour Cream & Onion"),
            Specialty = sum(Flavor_Theme == "Specialty/Other"),
            Spicy_Chili = sum(Flavor_Theme == "Spicy & Chili"),
            Vinegar_Tangy = sum(Flavor_Theme == "Vinegar & Tangy"))

chisq.test(Contingency_Table_Flavour[,-1])
```

```
##
##  Pearson's Chi-squared test
##
## data:  Contingency_Table_Flavour[, -1]
## X-squared = 326.64, df = 312, p-value = 0.273
```

Although individual proportions are stable over time, a separate analysis reveals that bag size and flavour theme are not independent dimensions.

Specifically:

- Certain flavours are not available in specific bag sizes (Zero counts exist in parts of the size × flavour contingency table).
- Some flavour categories are disproportionately represented within particular size tiers.

This structural dependency implies that inventory allocation cannot treat size and flavour independently. Allocating inventory to sizes first and flavours separately would ignore joint demand structure and introduce mix risk. Therefore, inventory must be managed at the size × flavour cell level.

```r
# ------------------------------------------------------
# Do Bag Sizes and Flavour Themes Exhibit Structural Dependencies ?
# ------------------------------------------------------
```

```r
Chips_products %>%
  group_by(Weight_Tier) %>%
  summarise(Cheese_Dairy = sum(Flavor_Theme == "Cheese & Dairy"),
            Meat_Savoury = sum(Flavor_Theme == "Meat & Savory"),
            Plain_Salted = sum(Flavor_Theme == "Plain & Salted"),
            SC_Onion = sum(Flavor_Theme == "Sour Cream & Onion"),
            Specialty = sum(Flavor_Theme == "Specialty/Other"),
            Spicy_Chili = sum(Flavor_Theme == "Spicy & Chili"),
            Vinegar_Tangy = sum(Flavor_Theme == "Vinegar & Tangy"))
```

```
## # A tibble: 3 x 8
##   Weight_Tier Cheese_Dairy Meat_Savoury Plain_Salted SC_Onion Specialty
##   <chr>              <int>        <int>        <int>    <int>     <int>
## 1 Large               6201            0         3142        0      6416
## 2 Medium              9309         4649         2964     3105      1564
## 3 Small              25187        37006        37273    23295     30432
## # i 2 more variables: Spicy_Chili <int>, Vinegar_Tangy <int>
```

Weekly Inventory Determination:

While baseline demand is stable, residual analysis demonstrates occasional extreme deviations. These shocks introduce stockout risk if inventory is set equal to mean demand.

To mitigate this risk, inventory is allocated proportionally across size × flavour cells and augmented with a volatility buffer:

$$I_{s,f} = \bar{s}_{s,f} \cdot \text{Baseline Demand} + k \cdot \sigma_{s,f}$$

Where:

$$I_{s,f} = \text{Weekly Inventory},\ \bar{s}_{s,f} = \text{mean share},\ \ k = \text{volatility multiplier},\ \ \sigma_{s,f} = \text{standard deviation}$$

This approach preserves proportional allocation while allowing more volatile cells to receive an additional buffer against stockout risk.

```r
# ------------------------------------------------------
# Proportional Stocking Arrangements for Flavour Themes by Bag Size - Weekly
# ------------------------------------------------------

weekly_cell <- Chips_products %>%
  mutate(week = floor_date(DATE, "week", week_start = 1)) %>%
  group_by(week, Weight_Tier, Flavor_Theme) %>%
  summarise(D_cell = sum(PROD_QTY), .groups = "drop")

cell_shares <- weekly_cell %>%
  group_by(Weight_Tier, Flavor_Theme) %>%
  summarise(mean_demand = mean(D_cell), .groups = "drop") %>%
  mutate(mean_share = mean_demand / sum(mean_demand))

cell_inventory <- cell_shares %>%
  mutate(I_cell = mean_share * baseline_demand)

cell_stats <- weekly_cell %>%
  group_by(Weight_Tier, Flavor_Theme) %>%
```

```
  summarise(
    mean_demand = mean(D_cell),
    sd_demand   = sd(D_cell),
    .groups = "drop"
  ) %>%
  mutate(mean_share = mean_demand / sum(mean_demand))

sens_grid <- expand.grid(
  k_val = seq(0, 1.5, by = 0.1)
)

sens_analysis <- sens_grid %>%
  crossing(cell_stats) %>%
  mutate(
    I_cell = mean_share * baseline_demand +
                          k_val * sd_demand
  )
```

A sensitivity analysis was conducted over values of [0,1.5].

For each value:

- Inventory was allocated at the size × flavour level.
- Weekly stockouts were backtested against historical demand.
- Both overall and cell-level stockout rates were computed.

Results indicate:

At = 1, overall stockout rate falls to 5% (our target global stockout rate).

This demonstrates that a modest volatility buffer meaningfully reduces stockout risk while maintaining inventory efficiency.

Conclusion

Occasional spikes in demand introduce stockout risk if inventory equals the baseline mean. Therefore, inventory should be allocated proportionally across flavour × size cells, rather than equally across bag sizes and categories.

In addition, the inclusion of a volatility buffer (k = 1) along size the proportional allocation of product volume, reduces overall stockouts to 5% while minimising storage/transport/upkeep costs associated with larger inventory volumes.

```
backtest <- weekly_cell %>%
  left_join(sens_analysis,
            by = c("Weight_Tier", "Flavor_Theme")) %>%
  mutate(stockout = D_cell > I_cell)
```

```
## Warning in left_join(., sens_analysis, by = c("Weight_Tier", "Flavor_Theme")): Detected an unexpecte
## i Row 1 of `x` matches multiple rows in `y`.
## i Row 1 of `y` matches multiple rows in `x`.
## i If a many-to-many relationship is expected, set `relationship =
##   "many-to-many"` to silence this warning.
```

```
backtest %>%
  group_by(k_val) %>%
  summarise(stockout_rate = mean(stockout))
```

```
## # A tibble: 16 x 2
##    k_val stockout_rate
##    <dbl>         <dbl>
##  1   0          0.502
##  2   0.1        0.425
##  3   0.2        0.346
##  4   0.3        0.286
##  5   0.4        0.230
##  6   0.5        0.173
##  7   0.6        0.134
##  8   0.7        0.101
##  9   0.8        0.0844
## 10   0.9        0.0655
## 11   1          0.0511
## 12   1.1        0.0411
## 13   1.2        0.0355
## 14   1.3        0.0277
## 15   1.4        0.0233
## 16   1.5        0.0200
```

```
backtest %>%
  group_by(k_val, Flavor_Theme, Weight_Tier) %>%
  summarise(stockout_rate = mean(stockout))
```

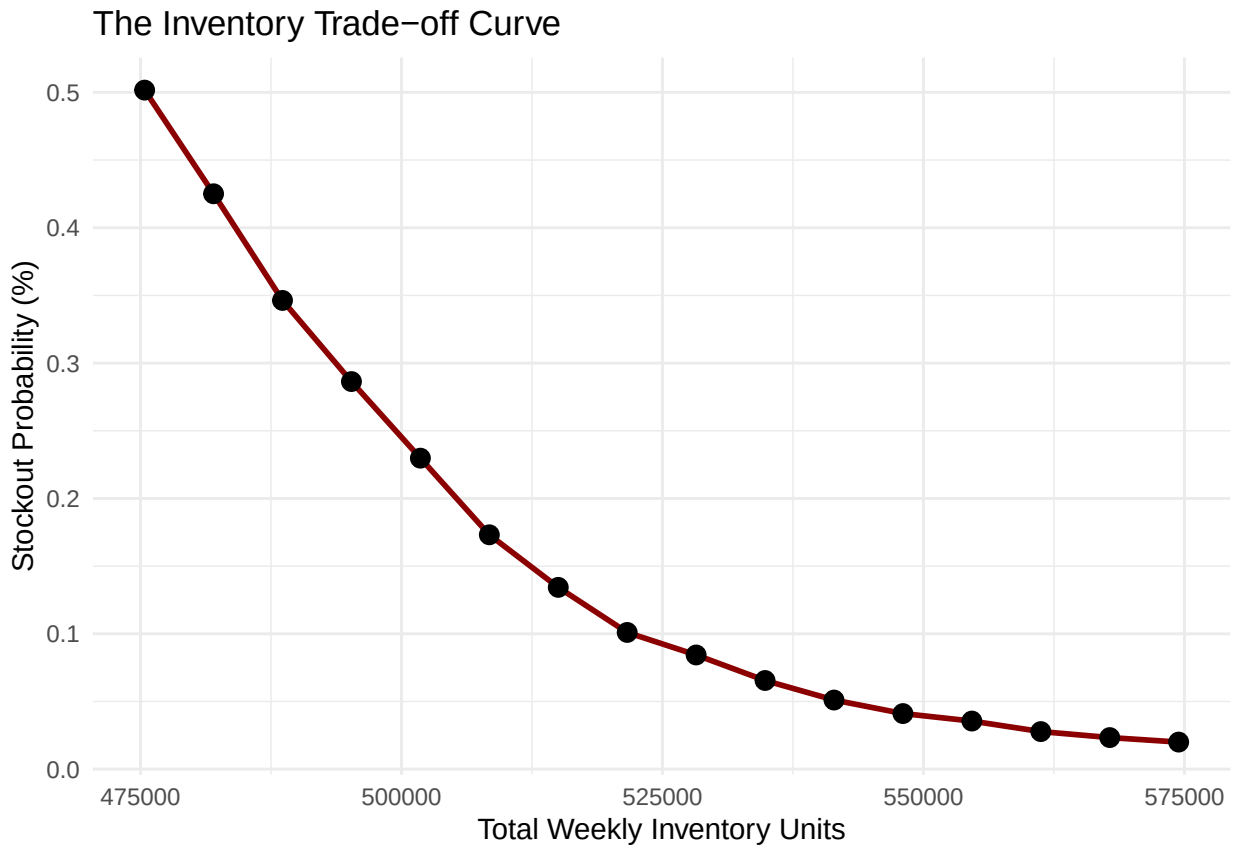
```
## `summarise()` has grouped output by 'k_val', 'Flavor_Theme'. You can override
## using the `.groups` argument.
```

```
## # A tibble: 272 x 4
## # Groups:   k_val, Flavor_Theme [112]
##    k_val Flavor_Theme       Weight_Tier stockout_rate
##    <dbl> <chr>              <chr>               <dbl>
##  1     0 Cheese & Dairy     Large               0.472
##  2     0 Cheese & Dairy     Medium              0.453
##  3     0 Cheese & Dairy     Small               0.453
##  4     0 Meat & Savory      Medium              0.566
##  5     0 Meat & Savory      Small               0.434
##  6     0 Plain & Salted     Large               0.547
##  7     0 Plain & Salted     Medium              0.453
##  8     0 Plain & Salted     Small               0.528
##  9     0 Sour Cream & Onion Medium              0.547
## 10     0 Sour Cream & Onion Small               0.491
## # i 262 more rows
```

```
backtest %>%
  group_by(k_val) %>%
  summarise(stockout_rate = mean(stockout),
            total_inv = sum(I_cell)) %>%
  ggplot(aes(x = total_inv, y = stockout_rate)) +
```

```
  geom_line(size = 1, color = "darkred") +
  geom_point(size = 3) +
  labs(title = "The Inventory Trade-off Curve",
       x = "Total Weekly Inventory Units",
       y = "Stockout Probability (%)") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```
# k = 1 => 5% stock out rate, stockout rates for all products < 10%
# annual inventory of 531216.3, weekly inventory of 10215.7
```

Governance and Model Risk

A statistical model used for operational decision-making introduces model risk.
If assumptions fail or conditions change, inventory decisions may no longer be appropriate.

The following governance considerations directly relate to the inventory optimisation problem:

Periodic Recalibration

The estimates of mean demand $\hat{\mu}$ and volatility $\hat{\sigma}$ are based on historical data.

If demand patterns shift (e.g., growth, decline, product changes), these estimates may become biased.

Recalibration ensures: - Safety stock remains aligned with current demand levels. - Service levels do not deteriorate due to outdated parameter estimates. - Capital allocation remains efficient.

Without recalibration, inventory buffers may become either insufficient (higher stockouts) or excessive (capital inefficiency).

Monitoring Realised Service Levels

Backtesting provides historical validation, but forward monitoring is essential.

Key governance metric: - Realised stockout rate versus target service level.

If realised stockouts exceed tolerance: - The safety multiplier k may be too low. - Volatility may be increasing. - Structural changes may have occurred.

Continuous monitoring ensures the model remains aligned with business risk appetite.

Structural Break Detection

Demand may shift due to: - Pricing changes - New competitors - Supply chain disruptions - Macroeconomic shocks

If such structural breaks occur, the assumption of stationarity fails.

Failing to detect regime changes may result in: - Underestimated volatility - Systematic stockouts - Misallocation of working capital

Formal change detection methods (e.g., rolling parameter estimation) would mitigate this risk.

Documentation of Assumptions

Key assumptions documented in this project:

- Independent weekly demand shocks
- Stable variance
- Approximate normality for safety buffer interpretation
- Symmetric penalty between stockouts and excess inventory

Limitations

The model is intentionally simple and robust; however, several limitations affect interpretation.

Limited Historical Data

The analysis relies on a finite observation window.

Implication: - Volatility estimates may be unstable. - Rare extreme demand events may not be captured. - Tail risk may be underestimated.

This may result in safety buffers that are insufficient under extreme scenarios.

No Explicit Cost Optimisation

The framework evaluates stockout frequency but does not explicitly quantify:

- Lost revenue from stockouts
- Holding cost of excess inventory
- Opportunity cost of capital

Instead, it provides a risk-based trade-off framework.

A cost-minimising approach could yield a more economically optimal safety multiplier.

Assumption of Symmetric Loss

The current formulation implicitly treats:

- One unit of stockout
- One unit of excess inventory

as equally weighted.

In reality: - Stockouts may damage brand reputation. - Excess inventory may be discounted or liquidated. - Costs are unlikely symmetric.

Without explicit cost modelling, the chosen k reflects implicit rather than optimised preferences.

Assumption of Demand Stationarity

The model assumes:

- Constant mean
- Constant variance
- No regime shifts

If demand trends upward or volatility increases over time, the safety stock formula may systematically underestimate required inventory.

No Regime-Switch or External Driver Modelling

The model does not incorporate:

- Promotional effects
- Seasonality indicators
- Macroeconomic variables
- Supply constraints

As a result, the model captures historical distributional behaviour but does not forecast structural shifts.

9. Potential Extensions

The following extensions directly address identified limitations and enhance robustness.

Monte Carlo Simulation

Rather than relying solely on historical realisations, simulated demand paths could:

- Model extreme but plausible scenarios
- Stress-test safety multipliers
- Estimate tail stockout risk

This would better capture rare but high-impact demand shocks.

Cost-Based Optimisation Framework

Introduce explicit cost parameters:

- Cost of stockout
- Cost of holding inventory
- Cost of capital

Optimise k by minimising expected total cost rather than selecting based on service level alone.

This converts the framework into a formal economic optimisation problem.

Scenario Stress Testing

Construct forward-looking stress scenarios:

- Sudden demand surge
- Demand collapse
- Increased volatility regime

Evaluate resulting stockout risk and capital requirement.

Final Conclusions:

Based on the analysis: - Adopt a static inventory framework, as demand exhibits no structural trend or seasonality. - Allocate inventory at the size × flavour level, due to structural dependencies between categories. - Incorporate a volatility-adjusted buffer, calibrated via backtesting. Set the volatility multiplier: 1

This strategy ensures:

- Aggregate risk control
- Protection against mix volatility
- Structural consistency with the product catalogue
- Operational simplicity

In summary, demand variation is primarily random rather than structural. A static, volatility-adjusted proportional allocation at the joint size × flavour level provides a statistically defensible and operationally robust inventory strategy.