~~~~
---
title: "Final Dissertation code "
output: html_document
date: "2025-08-12"
---



```{r}
library(tidyverse)
library(lubridate)
library(tsibble)
library(feasts)
library(slider)
library(GGally)
library(anomalize)
library(vip)
library(dbscan)
theme_set(theme_minimal(base_size = 12))

holiday_dates <- as.Date(c(
  "2010-11-26","2010-12-24",
  "2011-11-25","2011-12-23",
  "2012-11-23","2012-12-21"
))

```
## 01 Load & Clean
```{r }
~~~~

~~~
train    <- read_csv("train.csv", show_col_types = FALSE)
features <- read_csv("features.csv", show_col_types = FALSE)
stores   <- read_csv("stores.csv",  show_col_types = FALSE)

merged_data <- train %>%
  mutate(Date = as.Date(Date)) %>%
  left_join(features %>% mutate(Date = as.Date(Date)),
          by = c("Store","Date")) %>%
  left_join(stores, by = "Store") %>%
  select(-IsHoliday.y) %>%
  rename(IsHoliday = IsHoliday.x)



```



```{r}
missing_tbl <- merged_data %>%
  summarise(
    across(
      .cols = everything(),
      .fns  = list(
        count   = ~ sum(is.na(.x)),
        percent = ~ mean(is.na(.x)) * 100
      ),
      .names = "{.col}_{.fn}"
    )
~~~

~~~~
 ) %>%
 # ————— reshape —————
 pivot_longer(
   cols      = everything(),
   names_to   = c("variable", "metric"),
   names_sep  = "_(?=[^_]+$)",   # split on *last* underscore
   values_to  = "value"
 ) %>%
 pivot_wider(names_from = metric, values_from = value) %>%
 arrange(desc(percent))

print(missing_tbl)

```


#Handle Missing Values


```{r}
# Remove records with missing sales
merged_data <- merged_data %>% filter(!is.na(Weekly_Sales))

# Median imputation for continuous vars
merged_data <- merged_data %>%
 mutate(across(starts_with("MarkDown"), ~ replace_na(., 0)))

```
~~~~

~~~
```{r}
colSums(is.na(merged_data))


```
#Date & Key Features
```{r}
merged_data <- merged_data %>%
  mutate(
    year     = year(Date),
    quarter  = quarter(Date),
    week     = isoweek(Date),
    month    = month(Date, label = TRUE),
    wday_lbl = wday(Date, label = TRUE),
    ym       = yearmonth(Date)
  )



```




```{r}


str(merged_data)
```


# Sales Trends
#total weekly line plot
```{r}
~~~

~~~~
total_sales <- merged_data %>%
  group_by(Date) %>%
  summarise(Total_Weekly_Sales = sum(Weekly_Sales))


ggplot(total_sales, aes(x = Date, y = Total_Weekly_Sales)) +
  geom_line(color = "#2c3e50") +
  labs(title = "Total Weekly Sales Over Time",
       x = "Date", y = "Sales") +
  scale_y_continuous(labels = scales::dollar) +
  theme_minimal()
```
#Weekly sales by store type


```{r}
merged_data %>%
  group_by(Type) %>%
  summarise(Average_Sales = mean(Weekly_Sales)) %>%
  ggplot(aes(x = Type, y = Average_Sales, fill = Type)) +
  geom_col() +
  labs(title = "Average Weekly Sales by Store Type", x = "Store Type", y = "Average Sales") +
  scale_y_continuous(labels = scales::dollar) +
  theme_minimal()



```
#correlation Hetamap
```{r}
~~~~

~~~
library(reshape2)

numeric_vars <- merged_data %>%
  select_if(is.numeric) %>%
  drop_na()

cor_matrix <- cor(numeric_vars)

melted_cormat <- melt(cor_matrix)

ggplot(melted_cormat, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient2(low = "red", high = "blue", mid = "white",
                       midpoint = 0, limit = c(-1, 1)) +
  theme_minimal() +
  labs(title = "Correlation Heatmap")+
 theme(
  axis.text.x = element_text(angle = 45, hjust = 1)
)

```

```{r}
# Correlations as numbers (no plots)
~~~

```r
library(tidyverse)

library(Hmisc)     # rcorr() gives r, p, and n in one shot

library(knitr)     # nice table printing


# 1) Select numeric columns

num_df <- merged_data %>%

  dplyr::select(where(is.numeric))    # same as select_if(is.numeric) but newer


# 2) Correlation matrix with p-values (pairwise complete obs)

rc <- Hmisc::rcorr(as.matrix(num_df), type = "pearson")  # use type="spearman" if you
prefer

cor_mat <- rc$r   # correlations

p_mat   <- rc$P   # p-values

n_mat   <- rc$n   # pair counts


# 3) Print the full correlation matrix (rounded)

print(round(cor_mat, 3))


# 4) Tidy, de-duplicated pair list, sorted by |correlation|

cor_pairs <- cor_mat %>%

  as.data.frame() %>%

  rownames_to_column("var1") %>%

  pivot_longer(-var1, names_to = "var2", values_to = "corr") %>%

  # keep each pair once and drop self-correlations

  filter(var1 < var2) %>%

  mutate(

    p = map2_dbl(var1, var2, ~ p_mat[.x, .y]),
```

~~~
   n = map2_int(var1, var2, ~ n_mat[.x, .y])
  ) %>%
  arrange(desc(abs(corr)))

# 5) Show the top N strongest correlations
cor_pairs %>%
  mutate(corr = round(corr, 3),
         p    = signif(p, 3)) %>%
  slice_head(n = 30) %>%
  kable(caption = "Top 30 absolute correlations (Pearson)")

# 6) (Optional) Keep only "meaningful" correlations, e.g., |r| ≥ 0.3 and p < 0.05
meaningful_corrs <- cor_pairs %>%
  filter(abs(corr) >= 0.30, p < 0.05)




```
#Distributions of weekly_sales
#histogram full range & 99-pct zoom(Distribution of Weekly Sales)
```{r}
merged_data %>%
  filter(Weekly_Sales <= 100000) %>%
  ggplot(aes(x = Weekly_Sales)) +
  geom_histogram(bins = 100, fill = "#3498db", color = "white") +
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +
  labs(
~~~

```
  title = "Distribution of Weekly Sales (up to $100K)",
  x = "Weekly Sales", y = "Count"
 ) +
 theme_minimal()


```
```

#Weekly sales by store type

```
```{r}


library(tidyverse)
library(lubridate)
library(scales)

# 1 — WEEKLY totals by Date × Type × Year ——————————————————
weekly_type_year <- merged_data %>%
 mutate(Year = factor(year(Date))) %>%      # factor → legend / colour
 group_by(Date, Type, Year) %>%
 summarise(weekly_sales = sum(Weekly_Sales), .groups = "drop")

# 2 — Vibrant colour palette for the three years ——————————————
year_pal <- c("2010" = "#FF5733",  # bright orange-red
       "2011" = "#33C1FF",  # electric cyan
       "2012" = "#F820FF")  # neon magenta

# 3 — Plot: raw weekly spikes, coloured by Year, facets by Type ———
```

```
ggplot(weekly_type_year,
       aes(x = Date, y = weekly_sales,
           colour = Year, group = Year)) +
  geom_line(linewidth = 0.5) +               # thin = spikes visible
  facet_wrap(~ Type, ncol = 1, scales = "free_y") +
  scale_colour_manual(
    values = year_pal, name = "Year",
    guide  = guide_legend(override.aes = list(linewidth = 2))
  ) +
  scale_y_continuous(labels = dollar_format(scale = 1)) +
  labs(
    title = "Weekly Sales Spikes by Store Type (2010–2012)",
    x = NULL, y = "Weekly Sales ($)"
  ) +
  theme_minimal(base_size = 13) +
  theme(
    axis.text.x     = element_text(angle = 45, hjust = 1),
    strip.text      = element_text(size = 12, face = "bold"),
    legend.position = "right"
  )
```

` ` `

#Seasonality Patterns

Q. Does total sales differ between months, Are there seasonal differences by time period?

```{r}
library(dplyr);  library(lubridate);  library(ggplot2)
library(scales); library(knitr);     library(effectsize)

wmt_fy <- merged_data %>%
 mutate(
  fy  = if_else(month(Date) >= 2,  year(Date), year(Date) - 1L),    # fiscal year
  fm  = if_else(month(Date) >= 2,  month(Date) - 1L, 12L),         # fiscal month 1-12
  woy = isoweek(Date)                                  # ISO week (1-52/53)
 ) %>%
 filter(fy %in% 2010:2011)                             # keep two complete FYs


```

#Monthly profile (Feb → Jan pooled over FY-10 & FY-11)

```{r}
# — totals per fiscal month (pooled) ---------------------------------
month_tbl <- wmt_fy %>%
 group_by(fm) %>%
 summarise(total_sales = sum(Weekly_Sales), .groups = "drop")

kable(month_tbl,
   col.names = c("Fiscal Month (1 = Feb)", "Total Sales ($)"),
   format.args = list(big.mark = ",", scientific = FALSE))
```

~~~
# — weekly chain totals for ANOVA  -----------------------------------
wk_tbl <- wmt_fy %>%
  group_by(Date, fm) %>%
  summarise(chain_sales = sum(Weekly_Sales), .groups = "drop")

aov_m  <- aov(chain_sales ~ factor(fm), data = wk_tbl)
p_val  <- summary(aov_m)[[1]][["Pr(>F)"]][1]
F_stat <- summary(aov_m)[[1]][["F value"]][1]
eta2   <- effectsize::eta_squared(aov_m)$Eta2[1]

# gaps
hi <- month_tbl %>% slice_max(total_sales, n = 1)
lo <- month_tbl %>% slice_min(total_sales, n = 1)
gap_abs <- hi$total_sales - lo$total_sales
gap_pct <- 100 * gap_abs / hi$total_sales



```




```{r}
library(dplyr)
library(lubridate)
library(ggplot2)
library(scales)

# assumes `wmt_fy` already has columns:
#   fy  = fiscal year (2010 or 2011)
~~~

```
#   fm  = fiscal month (1 = Feb … 12 = Jan)


# ── monthly totals by fiscal year ──────────────────────────────────

stack_tbl <- wmt_fy %>%

 group_by(fy, fm) %>%                       # two FYs × 12 months

 summarise(total_sales = sum(Weekly_Sales), .groups = "drop") %>%

 mutate(fm_lab = factor(fm, levels = 1:12,

                labels = c("Feb","Mar","Apr","May","Jun",

                      "Jul","Aug","Sep","Oct","Nov","Dec","Jan")),

     fy_lab = factor(fy))                   # ensure fy is factor


# ── stacked bar plot -------------------------------------------------

ggplot(stack_tbl, aes(fm_lab, total_sales, fill = fy_lab)) +

 geom_col(position = "stack", colour = "white") +

 scale_fill_brewer(palette = "Set1", name = "Fiscal Year") +

 scale_y_continuous(labels = dollar) +

 labs(title = "Stacked fiscal-month sales (FY-2010 & FY-2011)",

    x = "Fiscal month (Feb = 1 … Jan = 12)",

    y = "Total sales ($)") +

 theme_minimal(base_size = 11)
```

#Week-of-Year profile (ISO weeks 1–52 pooled)
`

```{r}
```

```
woy_tbl <- wmt_fy %>%
  group_by(woy) %>%
  summarise(total_sales = sum(Weekly_Sales), .groups = "drop")

kable(woy_tbl,
      col.names = c("ISO Week", "Total Sales ($)"),
      format.args = list(big.mark = ",", scientific = FALSE))

top3 <- woy_tbl %>% slice_max(total_sales, n = 3)
bot3 <- woy_tbl %>% slice_min(total_sales, n = 3)

w_gap_abs <- max(top3$total_sales) - min(bot3$total_sales)
w_gap_pct <- 100 * w_gap_abs / max(top3$total_sales)


```

```{r}


# ── weekly totals pooled over FY-10 & 11 ──────────────────────────────
woy_tot <- wmt_fy %>%
  group_by(woy) %>%                 # iso week number (1-52)
  summarise(total_sales = sum(Weekly_Sales), .groups = "drop")

# ── line plot ------------------------------------------------------
ggplot(woy_tot, aes(woy, total_sales)) +
  geom_line(colour = "#2980b9", linewidth = 0.8) +
  geom_point(size = 1.5, colour = "#2980b9") +
  scale_x_continuous(breaks = seq(1, 52, by = 4)) +
  scale_y_continuous(labels = dollar) +
  labs(title = "ISO week-of-year sales (FY-2010 & FY-2011 pooled)",
      x = "ISO week number",
      y = "Total sales ($)") +
  theme_minimal(base_size = 11)
```

STORE FISCAL ANALYSIS

Which stores are performing best/worst and How do stores compare systematically?

```{r}
library(dplyr)
library(lubridate)
library(tidyr)
library(ggplot2)
library(viridis)
library(scales)
library(knitr)

# ── 0 · fiscal-year helper ------------------------------------------
merged_fy <- merged_data %>%
 mutate(
```

```r
    fy = if_else(month(Date) >= 2, year(Date), year(Date) - 1L), # FY key
    fy = factor(fy, levels = 2010:2012, labels = c("FY2010","FY2011","FY2012"))
  )

# ── 1·1  Year-by-store aggregates -----------------------------------
store_year_tbl <- merged_fy %>%
  group_by(Store, fy) %>%
  summarise(year_sales = sum(Weekly_Sales, na.rm = TRUE), .groups = "drop")

store_wide <- store_year_tbl %>%
  pivot_wider(names_from  = fy,
              values_from = year_sales,
              names_prefix = "Sales_") %>%
  mutate(Sales_Total = Sales_FY2010 + Sales_FY2011 + Sales_FY2012)

# ── 1·2  Rank & category flags ---------------------------------------
ranked_tbl <- store_wide %>%
  arrange(desc(Sales_Total)) %>%
  mutate(
    Rank_Overall = row_number(),
    Rank_FY2012  = rank(-Sales_FY2012, ties.method = "min"),
    Category     = case_when(
      Rank_Overall <= 3          ~ "Top 3",
      Rank_Overall > n() - 3     ~ "Bottom 3",
      TRUE                       ~ "Middle"
    )
  )
```

```
kable(ranked_tbl,
   col.names = c("Store","Sales FY10","Sales FY11","Sales FY12*",
            "Sales Total","Rank Overall","Rank FY12","Category"),
   format.args = list(big.mark = ",", scientific = FALSE),
   caption = "*FY-2012 covers Feb → Oct only")

# ── 2 · heat-map --------------------------------------------------------
heat_df <- ranked_tbl %>%
 select(Store, Sales_FY2010, Sales_FY2011, Sales_FY2012, Sales_Total, Category) %>%
 pivot_longer(starts_with("Sales_"),
        names_to  = "Year",
        values_to = "Sales") %>%
 mutate(Store = factor(Store, levels = ranked_tbl$Store))  # best → worst

ggplot(heat_df, aes(Year, Store, fill = Sales)) +
 geom_tile(colour = "white", linewidth = 0.3) +
 scale_fill_viridis(option = "C", labels = dollar,
            name = "Sales $") +
 # red outline for Top / Bottom stores
 geom_tile(data = heat_df %>% filter(Category != "Middle"),
       colour = "red", linewidth = 0.8, fill = NA) +
 labs(title = "Store performance heat-map by fiscal year (FY-2010 – FY-2012*)",
    subtitle = "*FY-2012 bar shorter because Nov–Jan withheld",
    x = NULL, y = "Store (ranked best → worst)") +
 theme_minimal(base_size = 10) +
 theme(axis.text.y = element_text(size = 6))

```
```
```

#DEPARTMENT

#Department-Level Performance by Fiscal Year

#Q. What is the sales distribution across departments and how stable is this distribution

over time?

~~~
```{r}

library(dplyr);  library(lubridate);  library(tidyr)

library(ggplot2); library(scales);    library(knitr)



dept_year_tbl <- merged_data %>%

  mutate(fy = if_else(month(Date) >= 2, year(Date), year(Date) - 1L),

        fy = factor(fy, levels = 2010:2012,

                    labels = c("FY2010","FY2011","FY2012"))) %>%

  group_by(Dept, fy) %>%

  summarise(year_sales = sum(Weekly_Sales), .groups = "drop")


dept_wide <- dept_year_tbl %>%

  pivot_wider(names_from = fy, values_from = year_sales,

            names_prefix = "Sales_") %>%

  mutate(Sales_Total = Sales_FY2010 + Sales_FY2011 + Sales_FY2012) %>%

  arrange(desc(Sales_Total)) %>%

  mutate(Rank_Overall = row_number())


# absolute & % gap between best and worst
~~~

~~~~
abs_gap <- dept_wide$Sales_Total[1] - dept_wide$Sales_Total[nrow(dept_wide)]
pct_gap <- 100 * abs_gap / dept_wide$Sales_Total[1]

kable(dept_wide,
   col.names = c("Dept","Sales FY10","Sales FY11","Sales FY12*",
          "Sales Total","Rank Overall"),
   format.args = list(big.mark = ",", scientific = FALSE),
   caption = paste0("*FY-2012 covers Feb → Oct only  |  Gap best→worst = $",
           comma(abs_gap), " (", round(pct_gap,1), "%)"))


` ` `
#Top-3-department share plot
` ` `{R}
# identify top-3 depts by pooled sales
top3 <- dept_wide$Dept[1:3]

plot_df <- merged_data %>%
 filter(Dept %in% top3) %>%
 mutate(fy = if_else(month(Date) >= 2, year(Date), year(Date)-1L),
     fy = factor(fy, levels = 2010:2012,
            labels = c("FY2010","FY2011","FY2012"))) %>%
 group_by(fy) %>%
 summarise(top3_sales = sum(Weekly_Sales), .groups = "drop") %>%
 # chain totals for denominator
 left_join(
  merged_data %>%
   mutate(fy = if_else(month(Date) >= 2, year(Date), year(Date)-1L),
~~~~

```
        fy = factor(fy, levels = 2010:2012,
                    labels = c("FY2010","FY2011","FY2012"))) %>%
    group_by(fy) %>% summarise(chain_sales = sum(Weekly_Sales), .groups = "drop"),
  by = "fy") %>%
 mutate(pct_share = top3_sales / chain_sales)

ggplot(plot_df, aes(fy, pct_share)) +
 geom_col(fill = "#FF5733") +
  geom_text(
  aes(label = scales::percent(pct_share, accuracy = 1)),
  vjust = -0.3,              # just above each bar
  size  = 4,                 # text size in mm
  fontface = "bold"
 ) +
 scale_y_continuous(labels = percent_format(accuracy = 1),
              limits = c(0, 0.50), expand = c(0,0)) +
 labs(title = "Share of Chain Sales from Top-3 Departments",
    subtitle = "Fiscal-year view (FY-2012 partial Feb–Oct)",
    x = NULL, y = "Percent of Total Sales") +
 theme_minimal(base_size = 11)
```

```{r}

top3 <- dept_wide$Dept[1:3]
```

```
# --- Totals by fiscal year (chain & top-3) ---
chain_by_fy <- dept_year_tbl %>%
  group_by(fy) %>%
  summarise(chain_sales = sum(year_sales), .groups = "drop")

top3_by_fy <- dept_year_tbl %>%
  filter(Dept %in% top3) %>%
  group_by(fy) %>%
  summarise(top3_sales = sum(year_sales), .groups = "drop")

# --- Each non-top-3 department's share of the "rest 80%" (per FY) ---
rest_dept_fy <- dept_year_tbl %>%
  filter(!(Dept %in% top3)) %>%
  left_join(chain_by_fy, by = "fy") %>%
  left_join(top3_by_fy, by = "fy") %>%
  mutate(
    rest_pool = chain_sales - top3_sales,
    share_of_rest_fy  = if_else(rest_pool > 0, year_sales / rest_pool, NA_real_),
    share_of_total_fy = year_sales / chain_sales
  )

# --- Average contributions across FYs ---
rest_avg <- rest_dept_fy %>%
  group_by(Dept) %>%
  summarise(
    pooled_share_of_rest = sum(year_sales, na.rm = TRUE) / sum(rest_pool, na.rm =
TRUE),
    mean_share_of_rest   = mean(share_of_rest_fy, na.rm = TRUE),

    pooled_share_of_total = sum(year_sales, na.rm = TRUE) / sum(chain_sales, na.rm =
TRUE),

    .groups = "drop"

  ) %>%

  arrange(desc(pooled_share_of_rest))



sum_rest_share <- sum(rest_avg$pooled_share_of_rest, na.rm = TRUE)

message("Sum of pooled shares across rest = ", round(sum_rest_share, 3))



rest_tbl <- rest_avg %>%

  mutate(

    `Pooled share of rest` = scales::percent(pooled_share_of_rest, accuracy = 0.1),

    `Mean share of rest (FY avg)` = scales::percent(mean_share_of_rest, accuracy = 0.1),

    `Pooled share of TOTAL` = scales::percent(pooled_share_of_total, accuracy = 0.1)

  ) %>%

  select(Dept, `Pooled share of rest`, `Mean share of rest (FY avg)`, `Pooled share of
TOTAL`)


knitr::kable(rest_tbl %>% slice_head(n = 15),

             caption = "Top 15 contributors among the OTHER ~80% (excl. top-3 depts)")



```
```
```

~~~
```{r}

library(forcats); library(scales); library(ggplot2); library(dplyr)

N <- 15

rest_top <- rest_avg %>%
  arrange(desc(pooled_share_of_rest)) %>%
  mutate(Dept = as.factor(Dept))

rest_topN <- rest_top %>% slice_head(n = N)
others_share <- 1 - sum(rest_topN$pooled_share_of_rest, na.rm = TRUE)

rest_topN_plus <- bind_rows(
  rest_topN,
  tibble(Dept = factor("Other depts"), pooled_share_of_rest = others_share)
) %>%
  mutate(Dept = fct_reorder(Dept, pooled_share_of_rest))

ggplot(rest_topN_plus, aes(x = Dept, y = pooled_share_of_rest)) +
  geom_col(fill = "#2c3e50") +
  coord_flip() +
  geom_text(aes(label = percent(pooled_share_of_rest, accuracy = 0.1)),
            hjust = -0.1, size = 3) +
  scale_y_continuous(labels = percent, expand = expansion(mult = c(0, .15))) +
  labs(
    title = paste0("Which depts make up the OTHER ~80%?  (Top ", N, " + 'Other')"),
    x = "Department (non-top-3 only)",
~~~

```
  y = "Share of the remaining ~80% (pooled across FYs)"
 ) +
 theme_minimal(base_size = 11)
```

```{r}

library(treemapify)

ggplot(rest_top, aes(area = pooled_share_of_rest, fill = pooled_share_of_rest,
           label = Dept)) +
 geom_treemap() +
 geom_treemap_text(colour = "white", place = "centre", grow = TRUE, reflow = TRUE) +
 scale_fill_gradient(low = "#9ecae1", high = "#08519c", labels = percent) +
 labs(
  title = "Treemap: contribution of each non-top-3 department to the OTHER ~80%",
  fill  = "Share of rest"
 ) +
 theme_minimal(base_size = 11) + theme(legend.position = "right")

```

Fiscal-Year Monthly-Totals Analysis

Year over year

#.Are there statistically significant differences in sales performance across fiscal years 2010-2012?

~~~
```{r}
library(dplyr);  library(lubridate);  library(tidyr)
library(ggplot2); library(scales);    library(knitr)
library(effectsize);  library(broom)



```
~~~

Aggregate monthly totals by fiscal year (Feb → Oct only)

~~~
```{r}

fy_month_tbl <- merged_data %>%
  # ---- fiscal keys -----------------------------------------------
  mutate(
    fy = if_else(month(Date) >= 2, year(Date), year(Date) - 1L),
    fy = factor(fy, levels = 2010:2012, labels = c("FY2010","FY2011","FY2012")),
    Month = month(Date, label = TRUE, abbr = TRUE)
  ) %>%
  # ---- keep months present in all 3 FYs (Feb–Oct) ------------------
  filter(Month %in% month(2:10, label = TRUE, abbr = TRUE)) %>%
  group_by(fy, Month) %>%
  summarise(total_sales = sum(Weekly_Sales), .groups = "drop")
~~~

~~~
```

```{r}
fy_month_wide <- fy_month_tbl %>%
 pivot_wider(names_from = fy, values_from = total_sales)

kable(fy_month_wide,
    col.names = c("Month","FY-2010","FY-2011","FY-2012*"),
    format.args = list(big.mark = ",", scientific = FALSE),
    caption = "*FY-2012 covers Feb → Oct only")


```

  Grouped-bar plot

```{r}
ggplot(fy_month_tbl, aes(Month, total_sales, fill = fy)) +
 geom_col(position = "dodge") +
 scale_y_continuous(labels = dollar) +
 scale_fill_brewer(palette = "Set2", name = "Fiscal Year") +
 labs(title = "Monthly sales by fiscal year (common window Feb–Oct)",
     x = NULL, y = "Total sales ($)") +
 theme_minimal(base_size = 11)


```
~~~

~~~~
```{r}
library(dplyr);  library(lubridate);  library(broom)
library(effectsize);  library(knitr)

# --- weekly chain totals, Feb–Oct window ----------------------------
weekly_common <- merged_data %>%
  mutate(
    fy = if_else(month(Date) >= 2, year(Date), year(Date)-1L),
    fy = factor(fy, levels = 2010:2012,
                labels = c("FY2010","FY2011","FY2012")),
    Month = month(Date, label = TRUE, abbr = TRUE)
  ) %>%
  filter(Month %in% month(2:10, label = TRUE, abbr = TRUE)) %>%
  group_by(fy, Date) %>%
  summarise(chain_sales = sum(Weekly_Sales), .groups = "drop")

# --- choose parametric vs non-parametric -----------------------------
aov_mod <- aov(chain_sales ~ fy, data = weekly_common)
norm_p  <- shapiro.test(residuals(aov_mod))$p.value

if (norm_p > .05) {
  test_name <- "One-way ANOVA"
  stat_out  <- tidy(aov_mod)[1, c("df","statistic","p.value")]
  eta_tbl   <- effectsize::eta_squared(aov_mod, partial = TRUE)
  term_col  <- intersect(c("Effect","Parameter","Term"), names(eta_tbl))[1]
  eff_sz    <- eta_tbl %>%
          filter(.data[[term_col]] == "fy") %>%
          select(matches("^Eta2")) %>% pull() %>% round(3)
~~~~

~~~~
} else {
  test_name <- "Kruskal–Wallis"
  kw_mod    <- kruskal.test(chain_sales ~ fy, data = weekly_common)
  stat_out  <- tidy(kw_mod)[, c("parameter","statistic","p.value")] %>%
             rename(df = parameter)
  eff_sz    <- effectsize::epsilon_squared(kw_mod)$Epsilon2 %>% round(3)
}



cat("\n***", test_name, "***\n\n")      # bold header
kable(stat_out, digits = 3,
     col.names = c("df", ifelse(test_name=="One-way ANOVA","F","H"), "p"))
cat("\nPartial η$^2$ (fy) =", eff_sz, "\n")



` ` `
#which year is higher/lower
` ` `{r}
weekly_common %>%
  group_by(fy) %>%
  summarise(Mean = mean(chain_sales),
          SD   = sd(chain_sales),
          n    = n()) %>%
  mutate(across(Mean:SD, scales::dollar)) %>%
  knitr::kable()
~~~~

` ` `

#What is the quantitative impact of promotional activities and holiday periods on weekly sales performance?

#What is the effect of promotions/holidays

Fiscal-Years 2010 & 2011 — Promotion- and Holiday-Week Impact

` ` `{R}

```
library(dplyr);  library(lubridate);   library(ggplot2)
library(scales); library(effsize);     library(knitr)

# — Filter to the two complete fiscal years ----------------------
fy_span <- merged_data %>%
  filter(Date >= as.Date("2010-02-05"),
         Date <= as.Date("2012-01-27"))

# — Promo flag (any Markdown > 0) -------------------------------
fy_span <- fy_span %>%
  mutate(promo_flag = as.integer(if_any(starts_with("MarkDown"), ~ .x > 0)))

# — Weekly aggregation (one row per Friday) ----------------------
weekly_df <- fy_span %>%
  group_by(Date) %>%
  summarise(total_sales = sum(Weekly_Sales, na.rm = TRUE),
        promo_flag  = first(promo_flag),
        holiday     = first(IsHoliday), .groups = "drop")
```

```
` ` `
```

```
` ` `{r}
# ---- non-parametric test & effect size (PROMO) -----------------------
# Was: t.test(...) + Cohen's d
w_promo <- wilcox.test(total_sales ~ promo_flag,
               data = weekly_df,
               exact = FALSE, conf.int = TRUE, alternative = "two.sided")
delta_promo <- effsize::cliff.delta(total_sales ~ promo_flag,
                     data = weekly_df)$estimate

promo_tbl <- tibble(
 Group          = c("No-Promo (0)", "Promo (1)", "Δ (1–0)"),
 Mean_Weekly_Sales  = c(promo_stats$Mean, diff(promo_stats$Mean)),
 SD             = c(promo_stats$SD, NA),
 n              = c(promo_stats$n, NA),
 `95% CI`         = c("", "",
                paste0(scales::dollar(w_promo$conf.int[1]),
                    " → ", scales::dollar(w_promo$conf.int[2]))),
 `p-value`        = c("", "", signif(w_promo$p.value, 3)),
 `Cliff's delta`    = c("", "", round(delta_promo, 2))
)
```

```
knitr::kable(promo_tbl, digits = 0,
          col.names = c("Group","Mean ($)","SD","n",
                    "95 % CI","p-value","Cliff's Δ"))



```

```{r}

# ---- non-parametric test & effect size (HOLIDAY) --------------------
w_hol <- wilcox.test(total_sales ~ holiday,
               data = weekly_df,
               exact = FALSE, conf.int = TRUE, alternative = "two.sided")
delta_hol <- effsize::cliff.delta(total_sales ~ holiday,
                        data = weekly_df)$estimate

hol_tbl <- tibble(
 Group          = c("Non-Holiday (0)", "Holiday (1)", "Δ (1–0)"),
 Mean_Weekly_Sales  = c(hol_stats$Mean, diff(hol_stats$Mean)),
 SD             = c(hol_stats$SD, NA),
 n              = c(hol_stats$n, NA),
 `95% CI`         = c("", "",
                 paste0(scales::dollar(w_hol$conf.int[1]),
                      " → ", scales::dollar(w_hol$conf.int[2]))),
 `p-value`        = c("", "", signif(w_hol$p.value, 3)),
 `Cliff's delta`    = c("", "", round(delta_hol, 2))
)

knitr::kable(hol_tbl, digits = 0,
         col.names = c("Group","Mean ($)","SD","n",
                  "95 % CI","p-value","Cliff's Δ"))

```
```{r}
library(ggplot2); library(scales)

weekly_df <- merged_data %>%
  mutate(promo_flag = as.integer(if_any(starts_with("MarkDown"), ~ .x > 0))) %>%
  group_by(Date) %>%
  summarise(total_sales = sum(Weekly_Sales),
          holiday    = max(IsHoliday),
          promo_flag  = max(promo_flag),
          .groups = "drop")

ggplot(weekly_df, aes(Date, total_sales)) +
  geom_line(colour = "#34495e", linewidth = 0.8) +
  geom_point(data = weekly_df %>% filter(promo_flag==1),
          aes(Date, total_sales), alpha = 0.5, size = 1.2) +
  geom_vline(data = weekly_df %>% filter(holiday==1),
          aes(xintercept = Date), colour = "#e74c3c", linetype = "dashed", linewidth = 0.6) +
  scale_y_continuous(labels = dollar) +
  labs(title = "Chain weekly sales with promo points and holiday markers",
     x = NULL, y = "Weekly sales ($)") +
  theme_minimal(base_size = 11)
```

~~~
```

-------------------------

```{R}
library(tsibble); library(feasts); library(anomalize); library(dplyr)

chain_ts <- merged_data %>%
  group_by(Date) %>%
  summarise(Weekly_Sales = sum(Weekly_Sales), .groups = "drop") %>%
  as_tsibble(index = Date)

# STL decomposition
chain_ts %>%
  model(STL(Weekly_Sales ~ season(window = "periodic") + trend())) %>%
  components() %>%
  autoplot() + ggtitle("STL decomposition (chain weekly)")

```
~~~

~~~
```{r}

library(dplyr)
library(tsibble)
library(feasts)
library(ggplot2)
library(tseries)

# 1) Chain-level weekly totals ---------------------------------------
chain_tbl <- merged_data %>%
  group_by(Date) %>%
  summarise(Weekly_Sales = sum(Weekly_Sales, na.rm = TRUE), .groups = "drop") %>%
  arrange(Date)

# tsibble for ACF/PACF (ggplot style)
chain_tsbl <- chain_tbl %>% as_tsibble(index = Date)



chain_tsbl %>%
  ACF(Weekly_Sales, lag_max = 60) %>%
  autoplot() + ggtitle("ACF (Chain Weekly_Sales)")

chain_tsbl %>%
  PACF(Weekly_Sales, lag_max = 60) %>%
  autoplot() + ggtitle("PACF (Chain Weekly_Sales)")



chain_ts <- ts(chain_tbl$Weekly_Sales, frequency = 52)
~~~

~~~~
adf_raw <- adf.test(chain_ts, alternative = "stationary")
print(adf_raw)


```


-----------------------------------
#Feature engineering & Modeling

```{r}

# -----------------------------
# Final modeling
# -----------------------------

# Load libraries
library(tidyverse)
library(lubridate)
library(slider)
library(forecast)
library(prophet)
library(randomForest)
library(xgboost)
library(purrr)
library(ranger)
library(vip)
~~~~

```
#-------------------------------
# 1) Feature engineering
#-------------------------------
df_chain <- merged_data %>%
 group_by(Date) %>%
 summarise(Weekly_Sales = sum(Weekly_Sales), .groups = "drop") %>%
 arrange(Date) %>%
 mutate(
  lag_1   = lag(Weekly_Sales, 1),
  lag_4   = lag(Weekly_Sales, 4),
  lag_52  = lag(Weekly_Sales, 52),
  roll_mean_4  = slide_dbl(Weekly_Sales, mean, .before = 3,  .complete = TRUE),
  roll_mean_13 = slide_dbl(Weekly_Sales, mean, .before = 12, .complete = TRUE),
  week   = isoweek(Date),
  month  = month(Date),
  year   = year(Date),
  sin52  = sin(2 * pi * week / 52),
  cos52  = cos(2 * pi * week / 52)
 ) %>%
 drop_na()

#-------------------------------
# 2) Monte-Carlo CV setup
#-------------------------------
make_time_mc_cv <- function(data, train_size, test_size, n_reps) {
 n <- nrow(data)
 splits <- vector("list", n_reps)
 set.seed(123)
```

```
  for (i in seq_len(n_reps)) {
    start <- sample(1:(n - train_size - test_size + 1), 1)
    train_idx <- start:(start + train_size - 1)
    test_idx  <- (start + train_size):(start + train_size + test_size - 1)
    splits[[i]] <- list(train = data[train_idx, ], test = data[test_idx, ])
  }
  splits
}
cv_splits <- make_time_mc_cv(df_chain, train_size = 80, test_size = 8, n_reps = 10)

#-------------------------------
# 3) Model fitting function
#    (3 classical + 3 ML models)
#-------------------------------
run_models <- function(split) {
  train <- split$train
  test  <- split$test

  x_vars <- c(
    "lag_1","lag_4","lag_52",
    "roll_mean_4","roll_mean_13",
    "week","month","year","sin52","cos52"
  )

  # ----- Classical -----
  ts_train   <- ts(train$Weekly_Sales, frequency = 52)

  fit_sarima <- auto.arima(ts_train, seasonal = TRUE)
```

```
fc_sarima  <- forecast(fit_sarima, h = nrow(test))
sarima_pred <- as.numeric(fc_sarima$mean)


fit_ets  <- ets(ts_train)
fc_ets   <- forecast(fit_ets, h = nrow(test))
ets_pred <- as.numeric(fc_ets$mean)


fit_snaive <- forecast::snaive(ts_train, h = nrow(test), lag = 52)
snaive_pred <- as.numeric(fit_snaive$mean)
# ----- Machine learning -----
train_prophet <- train %>% dplyr::select(ds = Date, y = Weekly_Sales)
m_prophet <- prophet(
  train_prophet,
  yearly.seasonality = TRUE,
  weekly.seasonality = FALSE,
  daily.seasonality  = FALSE,
  verbose = FALSE
)
future <- test %>% dplyr::select(ds = Date)
fc_prophet   <- predict(m_prophet, future)
prophet_pred <- fc_prophet$yhat



rf_fit <- randomForest(
  Weekly_Sales ~ .,
  data = dplyr::select(train, Weekly_Sales, dplyr::all_of(x_vars)),
  ntree = 200, mtry = 3
)
```

```r
rf_pred <- predict(rf_fit, newdata = dplyr::select(test, dplyr::all_of(x_vars)))


dtrain <- xgb.DMatrix(as.matrix(train[, x_vars]), label = train$Weekly_Sales)
dtest  <- xgb.DMatrix(as.matrix(test[,  x_vars]))
xgb_fit <- xgboost(
  data = dtrain,
  nrounds = 200,
  objective = "reg:squarederror",
  max_depth = 4,
  eta = 0.1,
  subsample = 0.8,
  colsample_bytree = 0.8,
  verbose = 0
)
xgb_pred <- predict(xgb_fit, dtest)


# Return long format
tibble::tibble(
  Date   = test$Date,
  actual = test$Weekly_Sales,
  SARIMA      = sarima_pred,
  ETS         = ets_pred,
  sNaive      = snaive_pred,
  Prophet     = prophet_pred,
  RandomForest = rf_pred,
  XGBoost     = xgb_pred
) %>%
  tidyr::pivot_longer(
    cols = -c(Date, actual),
    names_to = "model",
    values_to = "pred"
  )
}

# Run CV
results <- purrr::map_dfr(cv_splits, run_models, .progress = TRUE)

#-------------------------------
# 4) Evaluate model performance
#-------------------------------
error_metrics <- results %>%
 filter(!is.na(actual), !is.na(pred)) %>%
 group_by(model) %>%
 summarise(
  MAE  = mean(abs(actual - pred)),
  RMSE = sqrt(mean((actual - pred)^2)),
  MAPE = mean(abs(actual - pred) / actual) * 100,
  .groups = 'drop'
 ) %>%
 arrange(MAPE)
print(error_metrics)

#-------------------------------
# 5) Diagnostic plots
#-------------------------------
# Boxplot of absolute errors by model
```

```r
ggplot(results, aes(x = model, y = abs(actual - pred), fill = model)) +
 geom_boxplot() +
 labs(title = "Absolute Error Distribution by Model", x = NULL, y = "Absolute Error") +
 theme_minimal()

# Actual vs. predicted (averaged over folds)
avg_preds <- results %>%
 group_by(Date, model) %>%
 summarise(actual = mean(actual), pred = mean(pred), .groups = "drop")

ggplot(avg_preds, aes(x = Date)) +
 geom_line(aes(y = actual, colour = "Actual"), size = 0.8) +
 geom_line(aes(y = pred, colour = model), linetype = "solid") +
 labs(
  title  = "Actual vs Predicted Weekly Sales (averaged over CV folds)",
  x = "Date", y = "Weekly Sales", colour = "Series"
 ) +
 theme_minimal()

#------------------------------
# 6) Feature importance (full data)
#------------------------------
# Random Forest (ranger) importance
rf_full <- ranger(
 Weekly_Sales ~ .,
 data = dplyr::select(
  df_chain, Weekly_Sales,
  dplyr::all_of(c(
   "lag_1","lag_4","lag_52","roll_mean_4","roll_mean_13",
   "week","month","year","sin52","cos52"
  ))
 ),
 importance = "permutation",
 num.trees = 500,
 seed = 1
)
vip::vip(rf_full, num_features = 10)

# XGBoost importance
dall <- xgb.DMatrix(
 data  = as.matrix(df_chain[, c(
  "lag_1","lag_4","lag_52","roll_mean_4","roll_mean_13",
  "week","month","year","sin52","cos52"
 )]),
 label = df_chain$Weekly_Sales
)
xgb_full <- xgboost(
 data = dall,
 nrounds = 300,
 objective = "reg:squarederror",
 max_depth = 4,
 eta = 0.1,
 subsample = 0.8,
 colsample_bytree = 0.8,
 verbose = 0
)
```

```
xgb_imp <- xgb.importance(model = xgb_full)
ggplot(xgb_imp, aes(x = reorder(Feature, Gain), y = Gain)) +
 geom_col( fill = "#e74c3c") +
 coord_flip() +
 labs(title = "XGBoost – Feature Importance (Gain)",
    x = "Feature", y = "Gain") +
 theme_minimal()



ggplot(avg_preds, aes(x = Date)) +
 geom_line(aes(y = actual, colour = "Actual"), size = 0.8) +
 geom_line(aes(y = pred, colour = "Predicted"), linetype = "dashed") +
 facet_wrap(~ model, scales = "free_y", ncol = 2) +
 labs(
  title = "Test set: Actual vs Predicted Overlay by Model",
  y = "Weekly Sales", colour = ""
 ) +
 theme_minimal()

```

#split by split results

```{r warning=FALSE}
# Apply run_models to each split and add an identifier for the fold
```

~~~
results <- purrr::imap_dfr(
  cv_splits,
  ~ run_models(.x) %>% mutate(split_id = .y),
  .progress = TRUE
)

# Now compute metrics for each split and model
metrics_by_split <- results %>%
  filter(!is.na(actual), !is.na(pred)) %>%
  group_by(split_id, model) %>%
  summarise(
    MAE  = mean(abs(actual - pred)),
    RMSE = sqrt(mean((actual - pred)^2)),
    MAPE = mean(abs(actual - pred) / actual) * 100,
    .groups = 'drop'
  ) %>%
  arrange(split_id, MAPE)

print(metrics_by_split)


```



#not used in dissertation
```{r}
# ---------------------------------------------------------------------
# 7. Classical residual diagnostics (full series)
~~~

```r
# --------------------------------------------------------------------

library(forecast)

# Fit SARIMA, ETS and Prophet on the full chain-level series
ts_full <- ts(df_chain$Weekly_Sales, frequency = 52)

# SARIMA on full series
fit_sarima_full <- auto.arima(ts_full, seasonal = TRUE)
sarima_resid_full <- residuals(fit_sarima_full)

# ETS on full series
fit_ets_full <- ets(ts_full)
ets_resid_full <- residuals(fit_ets_full)

# ---- Prophet with 52-week seasonality + holidays + exogenous ----
df_prophet_full <- df_chain %>%
  transmute(
    ds = Date, y = Weekly_Sales,
    week = isoweek(Date),
    month = month(Date),
    sin52 = sin(2*pi*week/52),
    cos52 = cos(2*pi*week/52)
  )

m_prophet_full <- prophet(
  yearly.seasonality = FALSE,     # we'll add a custom 52-week seasonality
  weekly.seasonality = FALSE,     # not meaningful for weekly data
  daily.seasonality  = FALSE,
  seasonality.mode   = "multiplicative",
  changepoint.prior.scale = 0.2,
  holidays.prior.scale    = 10,
  seasonality.prior.scale = 10,
  verbose = FALSE
)

# custom annual cycle (52 weeks)
m_prophet_full <- add_seasonality(
  m_prophet_full, name = "annual52", period = 52, fourier.order = 10
)

# (optional) add US holidays
m_prophet_full <- add_country_holidays(m_prophet_full, country_name = "US")

# add exogenous regressors (Fourier proxies, calendar)
for (rv in c("sin52","cos52","week","month")) {
  m_prophet_full <- add_regressor(m_prophet_full, rv)
}

m_prophet_full <- fit.prophet(m_prophet_full, df_prophet_full)

p_full <- predict(m_prophet_full, df_prophet_full)
prophet_resid_full <- df_prophet_full$y - p_full$yhat

# Residual diagnostics
forecast::ggAcf(prophet_resid_full) + labs(title = "Prophet Residuals – ACF")
```

```
forecast::ggPacf(prophet_resid_full) + labs(title = "Prophet Residuals – PACF")

Box.test(prophet_resid_full, lag = 24, type = "Ljung-Box")



# Plot residual ACF and PACF for each classical model

# These lines produce separate ACF/PACF plots; you can display them as needed

# SARIMA residual ACF/PACF

ggAcf(sarima_resid_full) + labs(title = "SARIMA Residuals – ACF")

ggPacf(sarima_resid_full) + labs(title = "SARIMA Residuals – PACF")

# Ljung–Box test

Box.test(sarima_resid_full, lag = 24, type = "Ljung-Box")


# ETS residual ACF/PACF

ggAcf(ets_resid_full) + labs(title = "ETS Residuals – ACF")

ggPacf(ets_resid_full) + labs(title = "ETS Residuals – PACF")

Box.test(ets_resid_full, lag = 24, type = "Ljung-Box")


```