Grocery Sales Forecast

Christine Vu[1], Dave Friesen[2]

12/12/2022

Get Data

Reference: [https://www.kaggle.com/competitions/store-sales-time-series-forecasting/data?select=transactions.csv (https://www.kaggle.com/competitions/store-sales-time-series-forecasting/data?select=transactions.csv)]

```
#dir <- "/Users/christinevu/Downloads/"
dir <- "../data/"

# Load dataset(s)
setwd(dir)
sales_df <- read.csv("train.csv", header = TRUE)
sales_test_df <- read.csv("test.csv", header = TRUE)
stores_df <- read.csv("stores.csv", header = TRUE)
oil_df <- read.csv("oil.csv", header = TRUE)
events_df <- read.csv("holidays_events.csv", header = TRUE)

# Data validation and understanding, including structure, content, and statistical characteristics covered below
```

Explore & Visualize Series

Univariate Analysis and Preliminary Pre-Processing

```
# e.g., statistical characteristics (including distribution, skewness, outliers)
#   +[optionally] review sample observations
univariate(sales_df); head(sales_df, 3); #str(sales_df)
```

```
Summary Univariate Analysis for (sales_df) (3,000,888 observations)
            Type         NA% Blank%    Unique Freq       Min       Max       Mean    Median Outlier< >Outlier Skewness nZV ACF1
id          integer                  3000888                3000887            1500444    No       No               N   1.0
date        character                   1684    1                                                                    N   1.0
store_nbr   integer                       54             1        54                 28    No      Yes               N   1.0
family      character                     33                                                                         N   0.8
sales       numeric             31%  379610                124717.0     357.8      11.0    No      Yes          7.4  N
onpromotion integer             79%     362                     741                        No       No         11.2  N
```

```
  id       date store_nbr     family sales onpromotion
1  0 2013-01-01         1 AUTOMOTIVE     0           0
2  1 2013-01-01         1  BABY CARE     0           0
3  2 2013-01-01         1     BEAUTY     0           0
```

```
univariate(sales_test_df); head(sales_test_df, 3); #str(sales_test_df)
```

```
Summary Univariate Analysis for (sales_test_df) (28,512 observations)
            Type         NA% Blank%    Unique Freq       Min       Max       Mean    Median Outlier< >Outlier Skewness nZV ACF1
id          integer                    28512          3000888   3029399            3015144    No       No               N   1.0
date        character                     16    1                                                                    N   1.0
store_nbr   integer                       54             1        54                 28    No      Yes               N   1.0
family      character                     33                                                                         N   0.8
onpromotion integer             55%     212                     646                        No      Yes          8.5  N
```

```
       id       date store_nbr     family onpromotion
1 3000888 2017-08-16         1 AUTOMOTIVE           0
2 3000889 2017-08-16         1  BABY CARE           0
3 3000890 2017-08-16         1     BEAUTY           2
```

```
univariate(stores_df); head(stores_df, 3); #str(stores_df)
```

```
Summary Univariate Analysis for (stores_df) (54 observations)
          Type         NA% Blank%    Unique Freq       Min       Max       Mean    Median Outlier< >Outlier Skewness nZV ACF1
store_nbr integer                       54              1        54                 28    No       No              N   0.9
city      character                     22                                                                      N   0.3
state     character                     16                                                                      N   0.4
type      character                      5                                                                      N   0.6
cluster   integer                       17              1        17                  8    No      Yes             N   0.2
```

```
  store_nbr  city     state type cluster
1         1 Quito Pichincha    D      13
2         2 Quito Pichincha    D      13
3         3 Quito Pichincha    D       8
```

```
univariate(oil_df); head(oil_df, 3); #str(oil_df)
```

```
Summary Univariate Analysis for (oil_df) (1,218 observations)
           Type         NA% Blank%    Unique Freq       Min       Max      Mean    Median Outlier< >Outlier Skewness nZV ACF1
date       character                    1218    1                                                                    N   1.0
dcoilwtico numeric       3               998           26.2     110.6      67.7      53.2    No      Yes         0.3  N   0.9
```

```
        date dcoilwtico
1 2013-01-01         NA
2 2013-01-02         93
3 2013-01-03         93
```

```
univariate(events_df); head(events_df, 3); #str(events_df)
```

```
Summary Univariate Analysis for (events_df) (350 observations)
            Type         NA% Blank%    Unique Freq       Min       Max      Mean    Median Outlier< >Outlier Skewness nZV ACF1
date        character                    312    7                                                                    N   1.0
type        character                      6                                                                        N   0.3
locale      character                      3                                                                        N   0.3
locale_name character                     24                                                                        N
description character                    103                                                                       N   0.2
transferred character                      2                                                                       Y
```

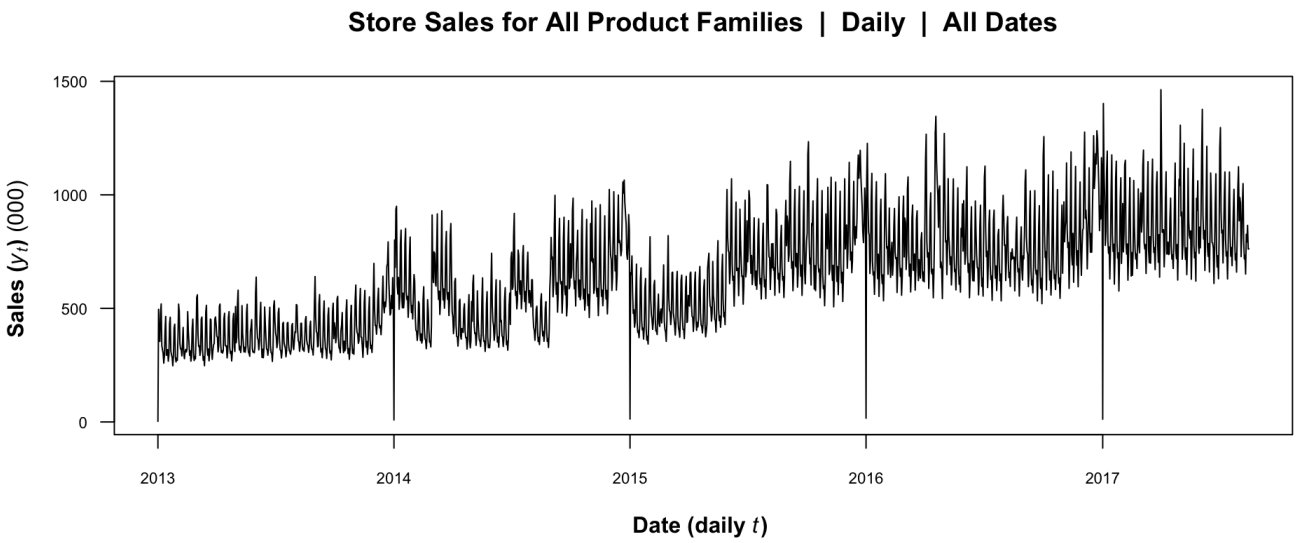
```
        date    type   locale locale_name                  description transferred
1 2012-03-02 Holiday    Local       Manta            Fundacion de Manta       False
2 2012-04-01 Holiday Regional    Cotopaxi Provincializacion de Cotopaxi       False
3 2012-04-12 Holiday    Local      Cuenca           Fundacion de Cuenca       False
```

```
# The following code was key for preliminary EDA, with individual results ultimately summarized
#   through the above univariate() calls

# Data summary statistics
#summary(sales_df)
#summary(sales_test_df)
#summary(stores_df)
#summary(oil_df)
#summary(events_df)

# Data types
#lapply(sales_df, class)
#lapply(sales_test_df, class)
#lapply(stores_df, class)
#lapply(oil_df, class)
#lapply(events_df, class)

# Row and column counts
#nrow(sales_df)
#row(sales_test_df)
#ncol(sales_df)
#ncol(sales_test_df)

# Missing values
#sum(is.na(sales_df))
#sum(is.na(sales_test_df))
#sum(is.na(stores_df))
#sum(is.na(oil_df))
#sum(is.na(events_df))
```

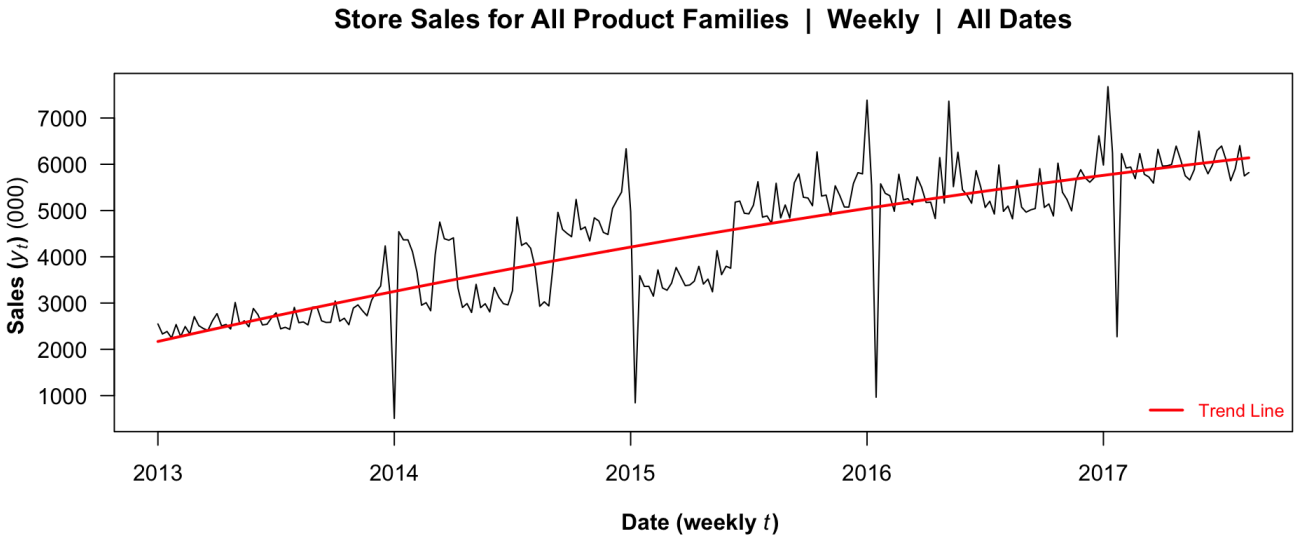
```
# Convert string mm/dd/yyyy to Date values and confirm sort
sales_df <- (sales_df %>%
               mutate(date = as.Date(date, format = "%Y-%m-%d"),) %>%
               arrange(date))
```

Series Visualization - All Product Families

```
# Aggregate base dataframe by date (i.e., sum all product lines) and initlally plot series at
#   provided time granularity
sales_agg_df <- as.data.frame(sales_df %>%
                                group_by(date) %>%
                                summarize(sales = sum(sales / 1000.0)))
plot(x = sales_agg_df$date, y = sales_agg_df$sales, type = "l",
     main = "Store Sales for All Product Families  |  Daily  |  All Dates",
     xlab = TeX(r"(\textbf{Date (daily \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7)
```



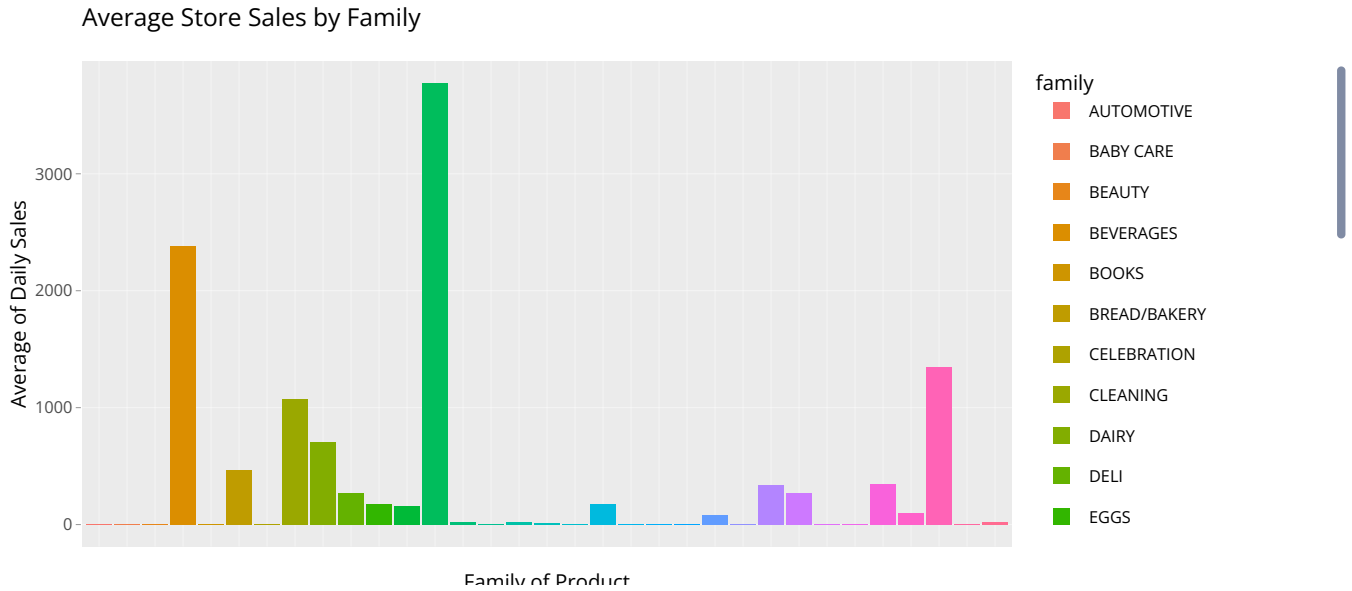

```
# Aggregate base dataframe from daily to weekly for all product families
sales_wk_df <- as.data.frame(sales_df %>%
                               mutate(year = year(date), week = week(date)) %>%
                               group_by(year, week) %>%
                               summarize(sales = sum(sales / 1000.0)))

# Create overall time series
sales_begin_year <- head(sales_wk_df$year, 1)
sales_begin_week <- head(sales_wk_df$week, 1)
sales_end_year <- tail(sales_wk_df$year, 1)
sales_end_week <- tail(sales_wk_df$week, 1)
sales_ts <- ts(sales_wk_df$sales,
               start = c(sales_begin_year, sales_begin_week),
               end = c(sales_end_year, sales_end_week), freq = 52)

# Plot overall time series with trend line
plot(sales_ts, type = "l",
     main = "Store Sales for All Product Families  |  Weekly  |  All Dates",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7)

sales_lm <- tslm(sales_ts ~ trend + I(trend^2))
lines(sales_lm$fitted, lwd = 2, lty = 1, col = "red")

legend("bottomright",
       legend = c("Trend Line"),
       col = c("red"),
       lwd = 2, lty = 1.2, cex = 0.8,
       box.lty = 0, bg = NULL, text.col = "red")
```




```
# Proportion by family
fam_prop <- sales_df %>%
              group_by(family) %>%
              summarise(Mean_Sales = round(mean(sales, na.rm = TRUE), 2),
                        Median_Sales = round(median(sales, na.rm = TRUE), 2),
                        Maximum_Sales = max(sales),
                        IQR_Sales = IQR(sales),
                        Total_Sales = sum(sales))
head(fam_prop, 3)
```

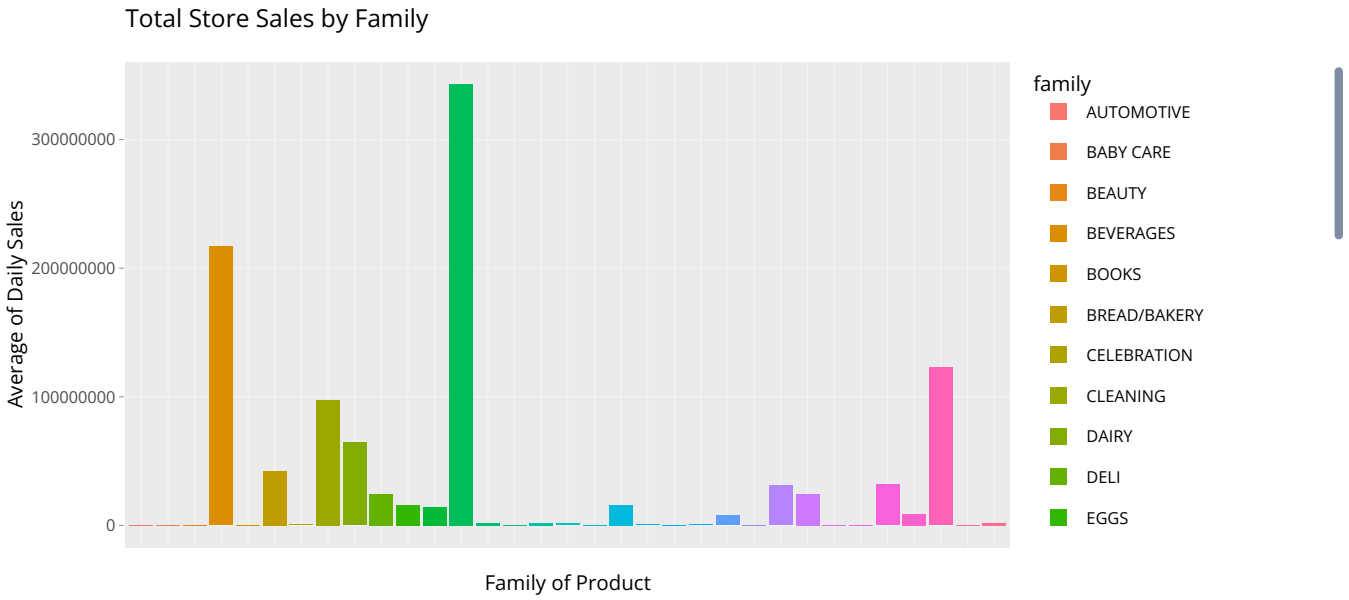
```
# A tibble: 3 × 6
  family     Mean_Sales Median_Sales Maximum_Sales IQR_Sales Total_Sales
  <chr>           <dbl>        <dbl>         <dbl>     <dbl>       <dbl>
1 AUTOMOTIVE       6.1             5           255         6      554822
2 BABY CARE        0.11            0           116         0       10051
3 BEAUTY           3.72            2           136         5      337893
```

```
# Box plot: Mean of store sales by family
fam_plot <- ggplot(fam_prop) +
            geom_col(aes(x = family, y = Mean_Sales, fill = family)) +
            theme(axis.ticks.x = element_blank(),
                  axis.text.x = element_blank()) +
            labs(title = "Average Store Sales by Family",
                 x = "Family of Product",
                 y = "Average of Daily Sales",
                 legend = "Family of Product Sold",
                 bty = "l")
ggplotly(fam_plot)
```



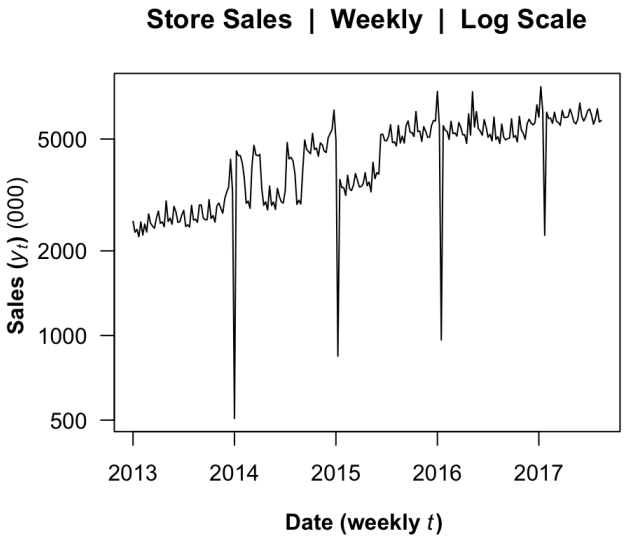
Family of Product

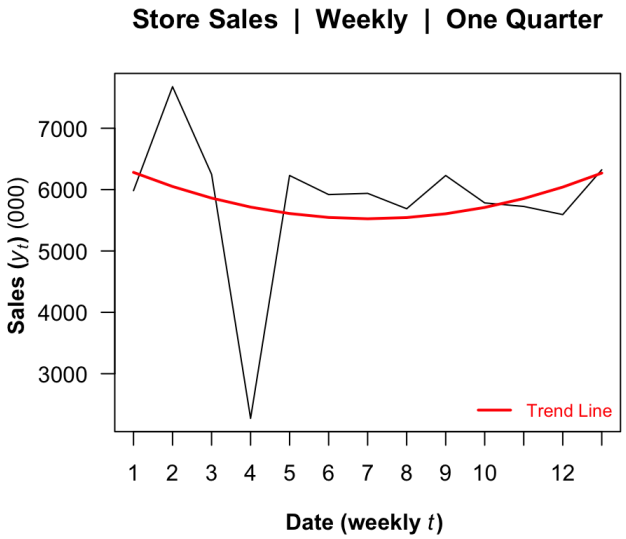
```
# Box plot: Total store sales by family
fam_plot2 <- ggplot(fam_prop) +
             geom_col(aes(x = family, y = Total_Sales, fill = family)) +
             theme(axis.ticks.x = element_blank(),
                   axis.text.x = element_blank()) +
             labs(title = "Total Store Sales by Family",
                  x = "Family of Product",
                  y = "Average of Daily Sales",
                  legend = "Family of Product Sold",
                  bty = "l")
ggplotly(fam_plot2)
```




```
# Extract day, month, year, etc. from sales_df
sales_df$day <- sales_df$date %>% day()
sales_df$month <- sales_df$date %>% month()
sales_df$month_lab <- sales_df$date %>% month(label = TRUE)
sales_df$year <- sales_df$date %>% year()
sales_df$week <- sales_df$date %>% week()
sales_df$week_day <- sales_df$date %>%
  wday(week_start = getOption("lubridate.week.start", 1), label = TRUE)
```

```
opar = par()
par(mfrow = c(1, 2))

# Plot overall time series w/log scale
plot(sales_ts, type = "l",
     main = "Store Sales  |  Weekly  |  Log Scale",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7,
     log = "y")

# Plot zoomed time series with trend line
sales_zoom_ts <- window(sales_ts, start = c(sales_end_year, 1), end = c(sales_end_year, 13))
plot(sales_zoom_ts, type = "l",
     main = "Store Sales  |  Weekly  |  One Quarter",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     xaxt = "n",
     las = 1, cex = 0.7)

sales_zoom_lm <- tslm(sales_zoom_ts ~ trend + I(trend^2))
lines(sales_zoom_lm$fitted, lwd = 2, lty = 1, col = "red")

axis(1, at = as.numeric(time(sales_zoom_ts)), labels = seq(sales_zoom_ts))

legend("bottomright",
       legend = "Trend Line",
       col = "red",
       lwd = 2, lty = 1.2, cex = 0.8,
       box.lty = 0, bg = NULL, text.col = "red")
```



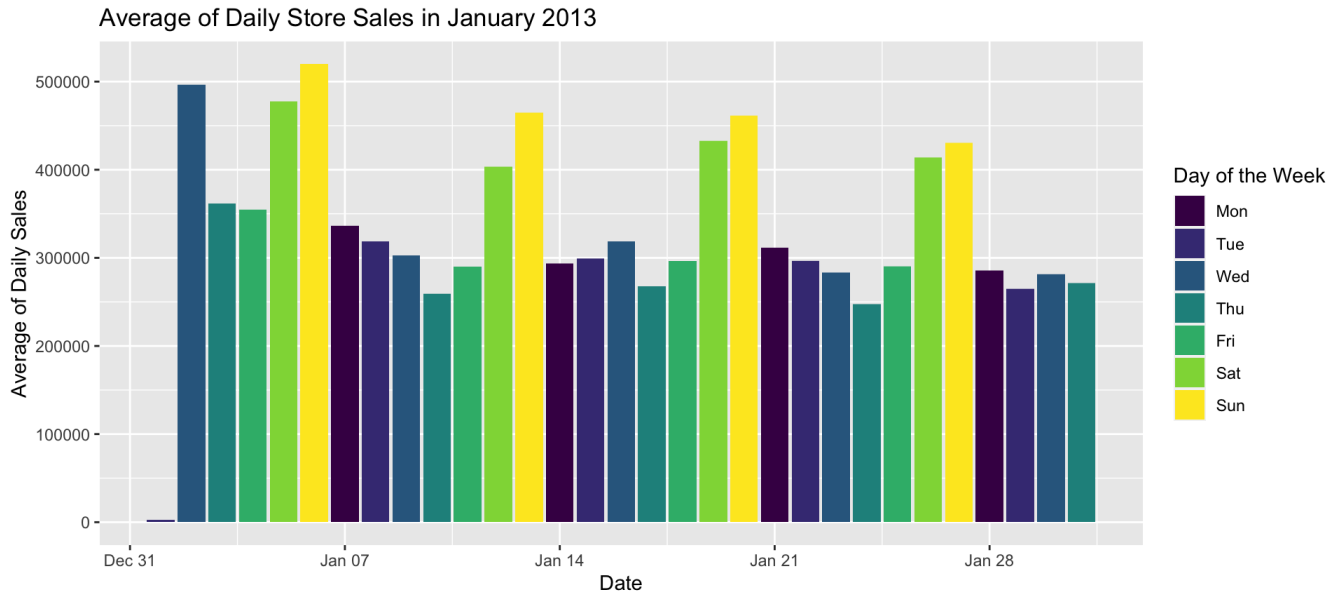



```
par(opar)
```

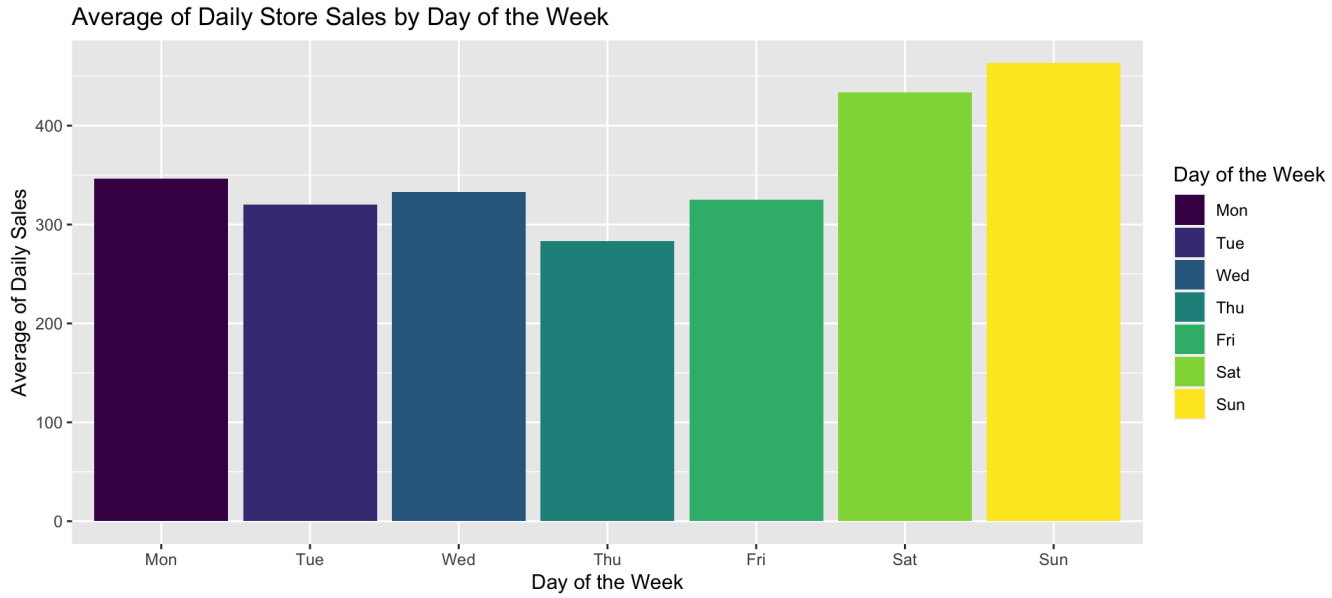
```
# Proportion by sales
sales_prop <- sales_df %>%
                group_by(store_nbr) %>%
                summarise(Mean_Sales = mean(sales),
                          Median_Sales = median(sales),
                          Maximum_Sales = max(sales),
                          IQR_Sales = IQR(sales),
                          IQR_to_avg_sales = IQR(sales) / mean(sales),
                          Total_Sales = sum(sales))
head(sales_prop, 3)
```

```
# A tibble: 3 × 7
  store_nbr Mean_Sales Median_Sales Maximum_Sales IQR_Sales IQR_to_avg_sales Total_Sales
      <int>      <dbl>        <dbl>         <dbl>     <dbl>            <dbl>       <dbl>
1         1       255.         19            9065       165            0.648   14145013.
2         2       388.         23.2        124717       289            0.745   21557389.
3         3       908.         64           21858       582            0.641   50481910.
```

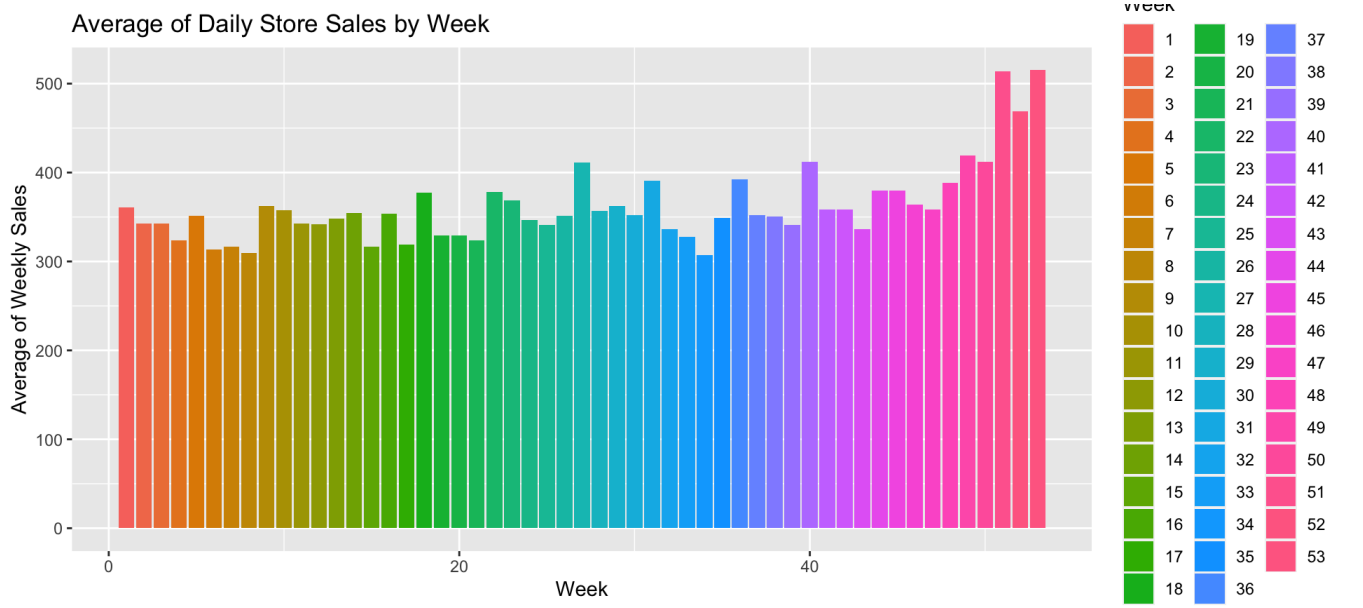
```
# Bar plot: Average of store sales by day of the week in January 2013
sales_plot <- sales_df %>%
                group_by(date) %>%
                  summarise(avg_sales = mean(sales), wday = week_day, .groups = "keep") %>%
                    filter(date <= '2013-01-31') %>%
                      ggplot() +
                      geom_col(aes(x = date, y = avg_sales, fill = wday)) +
                      labs(title = "Average of Daily Store Sales in January 2013",
                           x = "Date",
                           y = "Average of Daily Sales",
                           fill = "Day of the Week",
                           bty = "l")
sales_plot
```



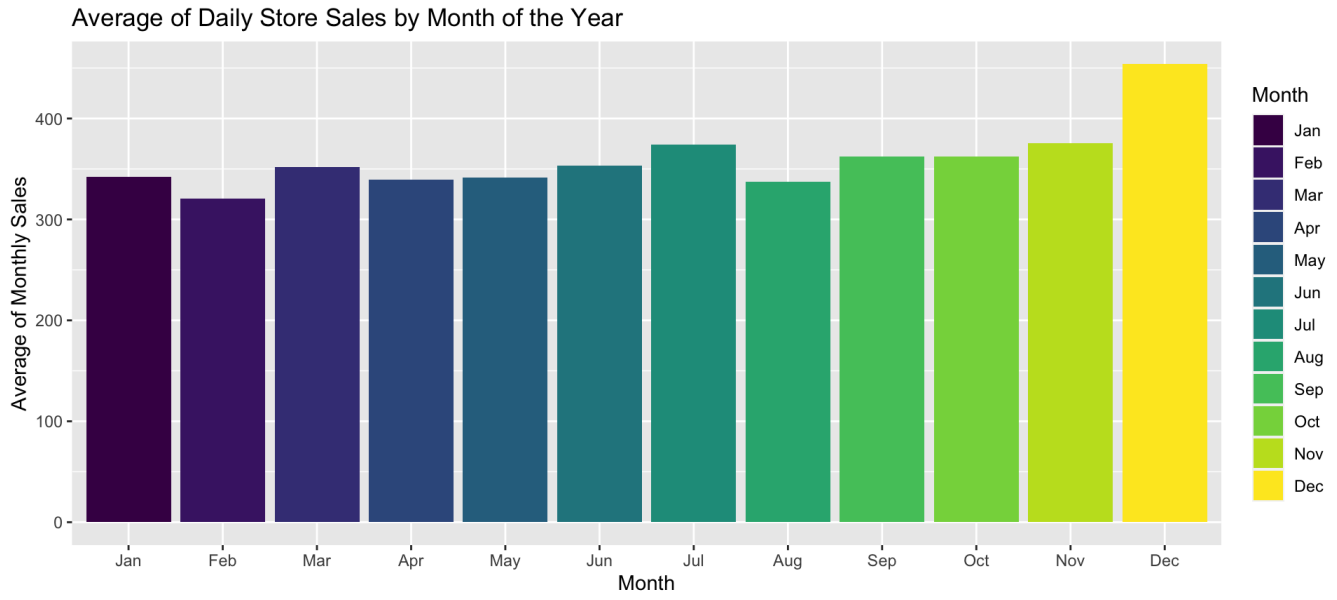

```
# Bar plot: Average of store sales by day of the week
sales_plot1 <- sales_df %>%
                group_by(week_day) %>%
                  summarise(avg_sales = mean(sales)) %>%
                    ggplot() +
                    geom_col(aes(x = week_day,
                                 y = avg_sales,
                                 fill = week_day)) +
                    labs(title = "Average of Daily Store Sales by Day of the Week",
                         x = "Day of the Week",
                         y = "Average of Daily Sales",
                         fill = "Day of the Week",
                         bty = "l")
sales_plot1
```



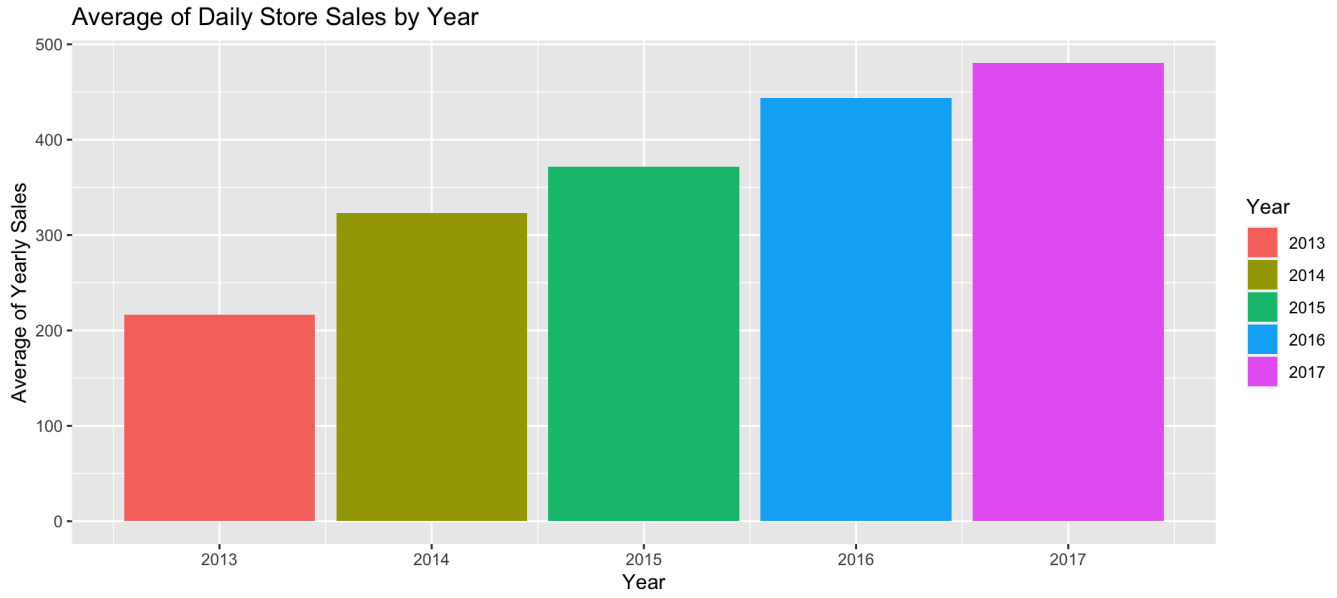
```
# Bar plot: Average of store sales by week
sales_plot2 <- sales_df %>%
                group_by(week) %>%
                  summarise(avg_sales = mean(sales)) %>%
                    ggplot() +
                    geom_col(aes(x = week,
                                 y = avg_sales,
                                 fill = as.factor(week))) +
                    labs(title = "Average of Daily Store Sales by Week",
                         x = "Week",
                         y = "Average of Weekly Sales",
                         fill = "Week",
                         bty = "l")
sales_plot2
```



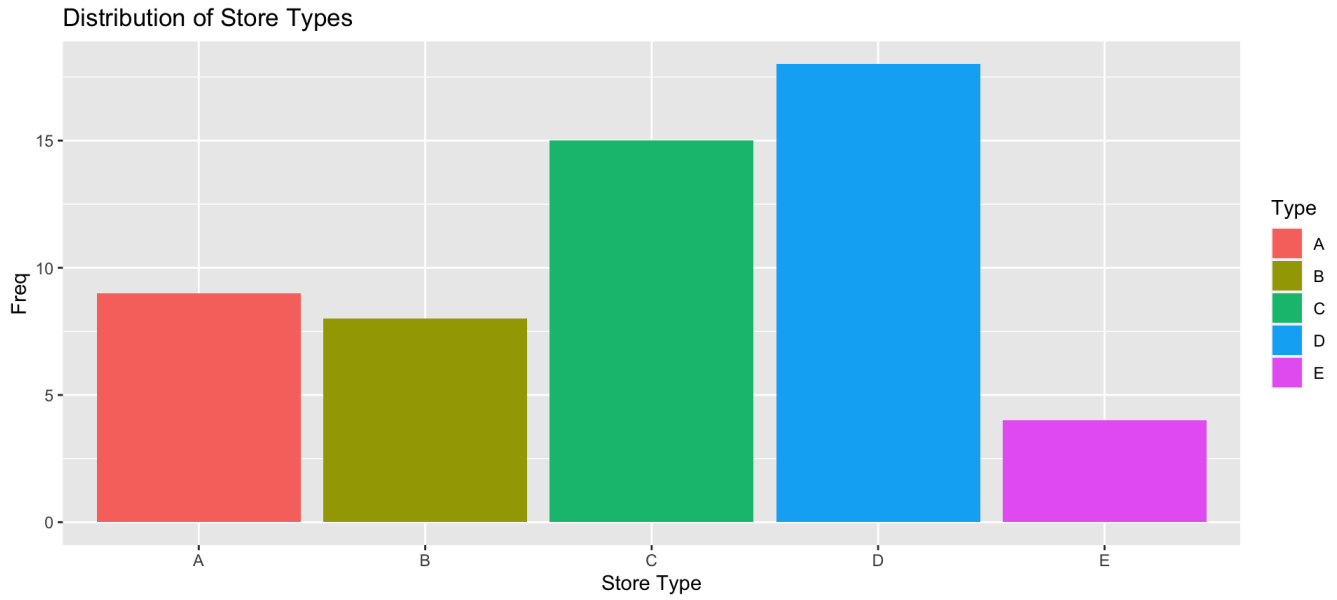

```
# Bar plot: Average of store sales by month of the year
sales_plot3 <- sales_df %>%
                group_by(month_lab) %>%
                  summarise(avg_sales = mean(sales)) %>%
                    ggplot() +
                    geom_col(aes(x = month_lab,
                                 y = avg_sales,
                                 fill = month_lab)) +
                    labs(title = "Average of Daily Store Sales by Month of the Year",
                         x = "Month",
                         y = "Average of Monthly Sales",
                         fill = "Month",
                         bty = "l")
sales_plot3
```



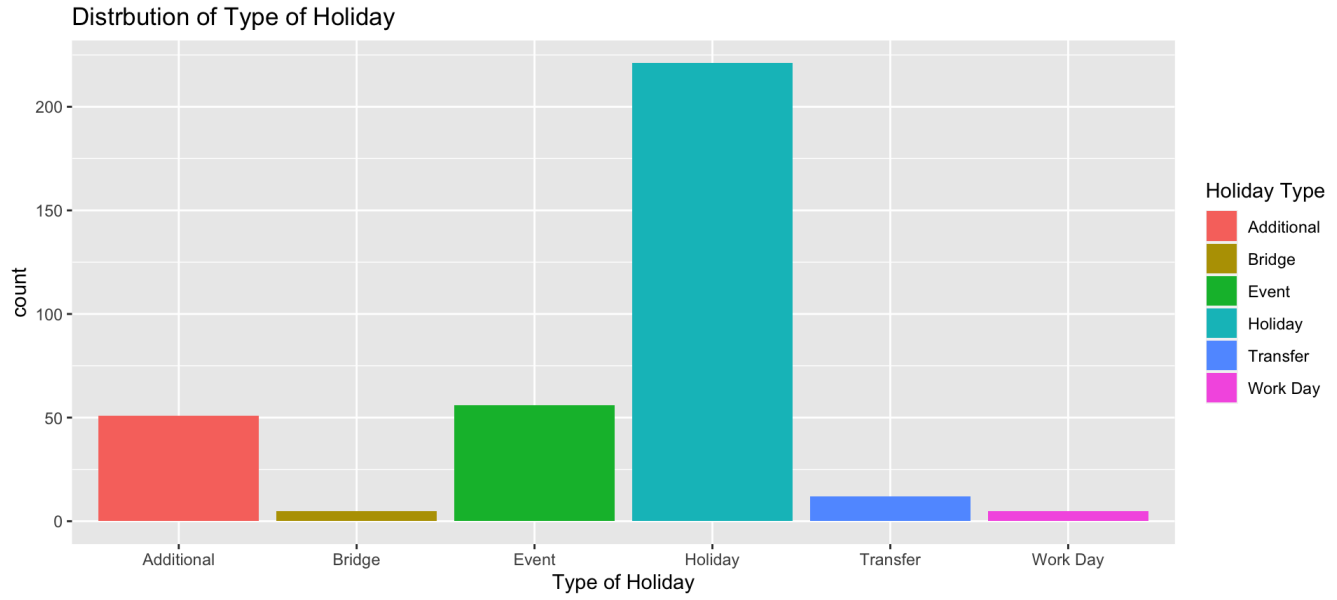

```
# Bar plot: Average of store sales by year
sales_plot4 <- sales_df %>%
                group_by(year) %>%
                  summarise(avg_sales = mean(sales)) %>%
                    ggplot() +
                    geom_col(aes(x = year,
                                 y = avg_sales,
                                 fill = as.factor(year))) +
                    labs(title = "Average of Daily Store Sales by Year",
                         x = "Year",
                         y = "Average of Yearly Sales",
                         fill = "Year",
                         bty = "l")
sales_plot4
```



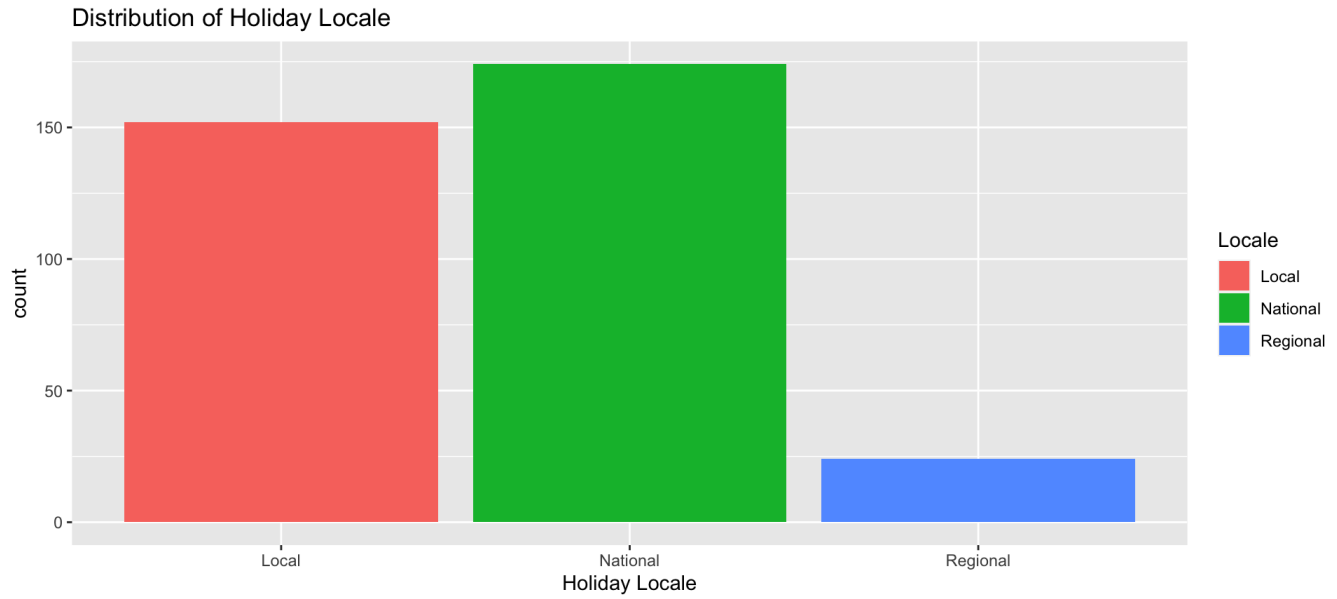

```
# Bar plot: Distribution of store types
store_plot <- stores_df$type %>%
                table() %>%
                  as.data.frame() %>%
                    ggplot() +
                    geom_col(aes(y = Freq, x = ., fill = as.factor(.))) +
                    labs(title = "Distribution of Store Types",
                         x = "Store Type",
                         fill = "Type",
                         bty = "l")
store_plot
```



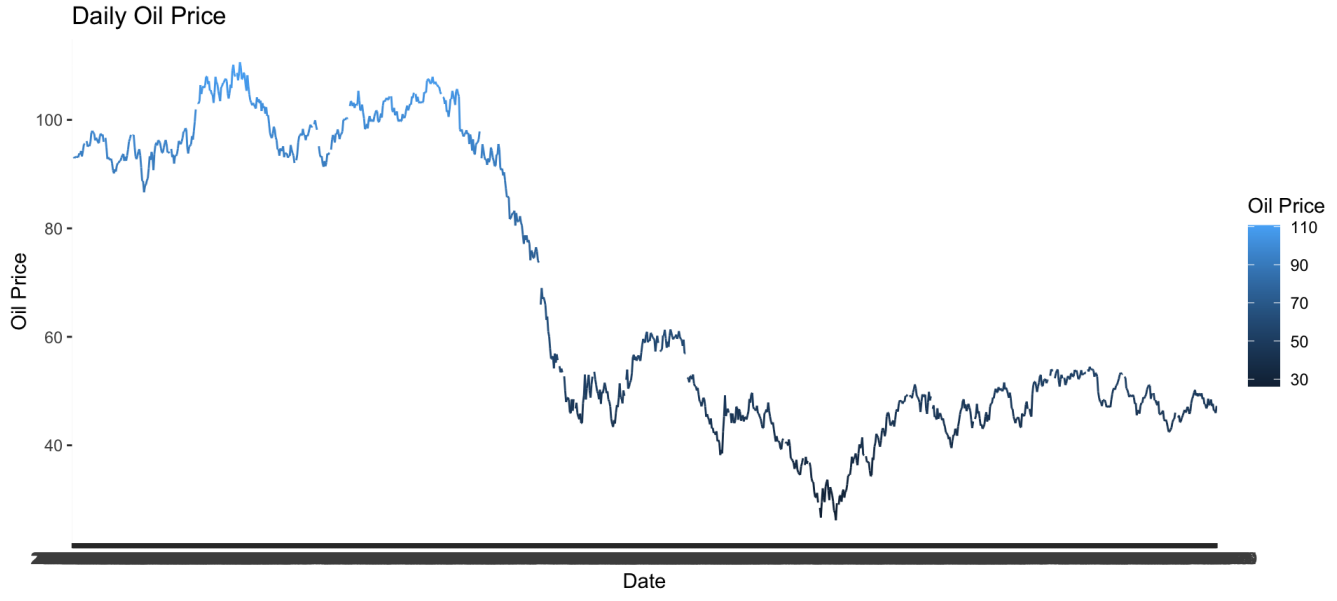

```
# Bar plot: Distribution of holiday type
events_plot1 <- events_df %>%
                  ggplot() +
                  geom_bar(aes(x = type, fill = type)) +
                  labs(title = "Distrbution of Type of Holiday",
                       x = "Type of Holiday",
                       fill = "Holiday Type",
                       bty = "l")
events_plot1
```




```
# Bar plot: Distribution of holiday locale
events_plot2 <- events_df %>%
                  ggplot() +
                  geom_bar(aes(x = locale, fill = locale)) +
                  labs(title = "Distribution of Holiday Locale",
                       x = "Holiday Locale",
                       fill = "Locale",
                       bty = "l")
events_plot2
```

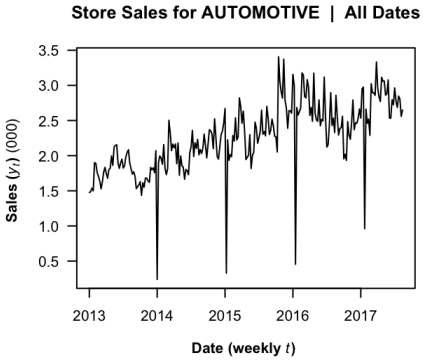


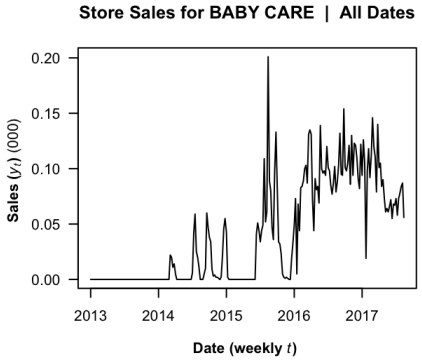

```
# Line chart: Daily oil price
oil_plot <- oil_df %>%
              ggplot(aes(x = date, y = dcoilwtico, color = dcoilwtico, group = 1)) +
              geom_line(na.rm = TRUE) +
              labs(title = "Daily Oil Price",
                   x = "Date",
                   y = "Oil Price",
                   color = "Oil Price",
                   bty = "l")
oil_plot
```

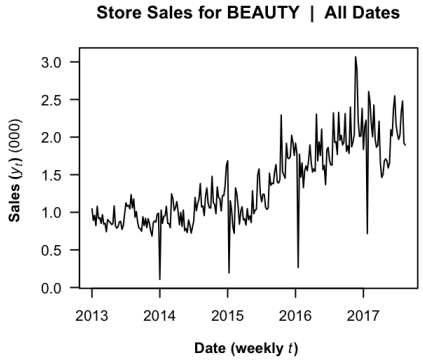


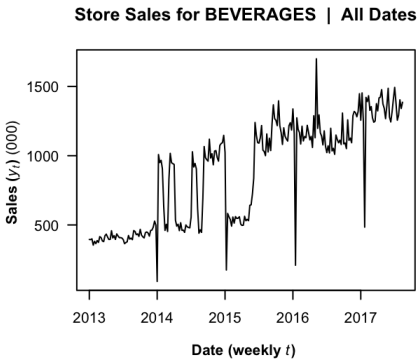
Series Visualization - Individual Product Families

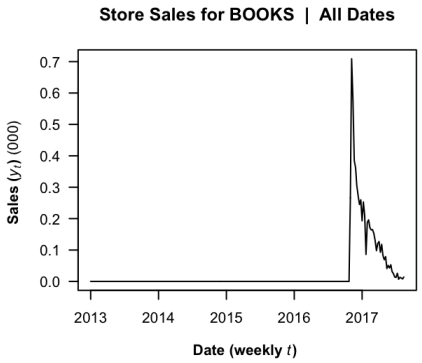
```r
# Aggregate base dataframe from daily to weekly by product family
sales_wk_df <- as.data.frame(sales_df %>%
                               mutate(year = year(date), week = week(date)) %>%
                               group_by(family, year, week) %>%
                               summarize(sales = sum(sales / 1000.0)))

opar = par()
par(mfrow = c(1, 3))

for (f in unique(sales_wk_df$family)) {
  # Subset data by product family and create time series
  df <- filter(sales_wk_df, family == f)
  df_ts <- ts(df$sales,
              start = c(sales_begin_year, sales_begin_week),
              end = c(sales_end_year, sales_end_week), freq = 52)

  # Plot time series
  plot(df_ts, type = "l",
       main = paste("Store Sales for ", f, "  |  All Dates", sep = ""),
       height = 0.8,
       xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
       las = 1, cex.axis = 0.7)
}
```

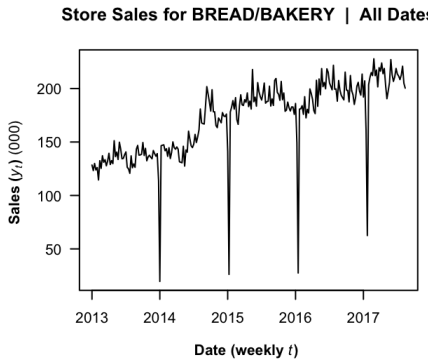


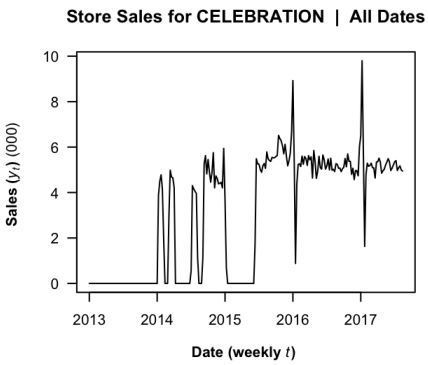


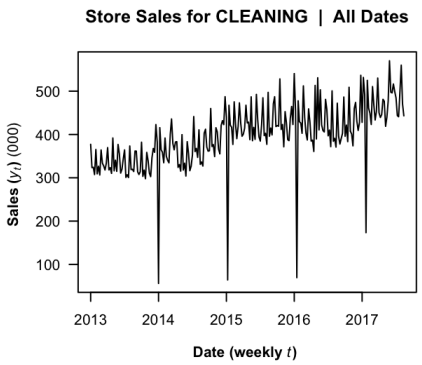


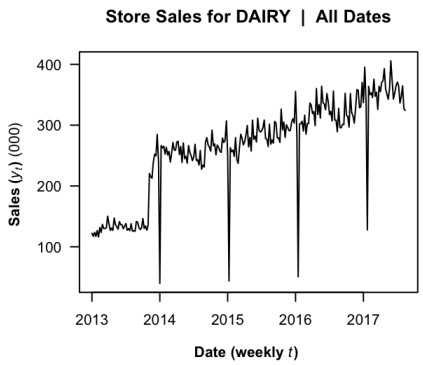


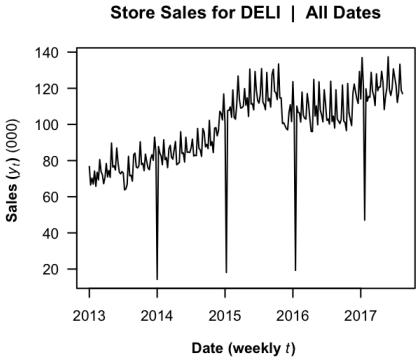


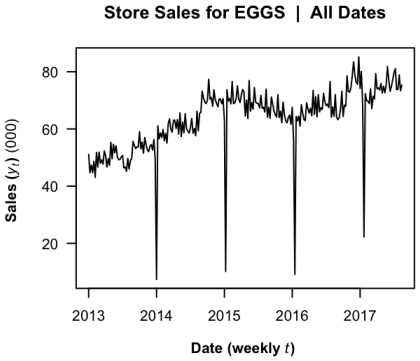


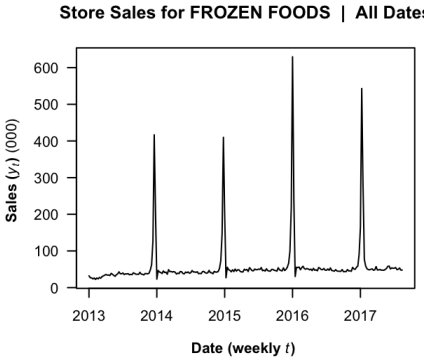


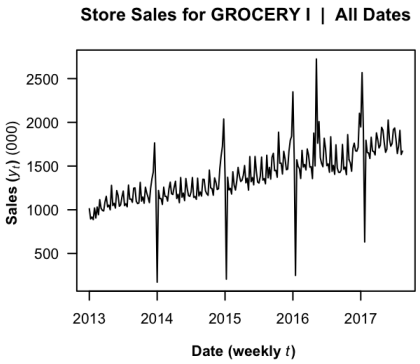


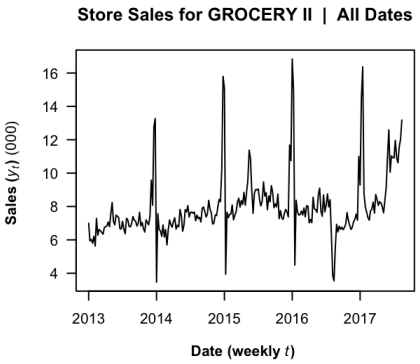


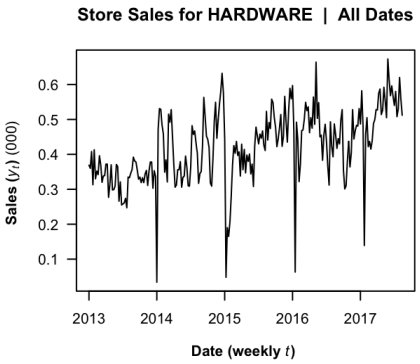

Store Sales for DELI | All Dates
Sales (y_t) (000)
Date (weekly t)

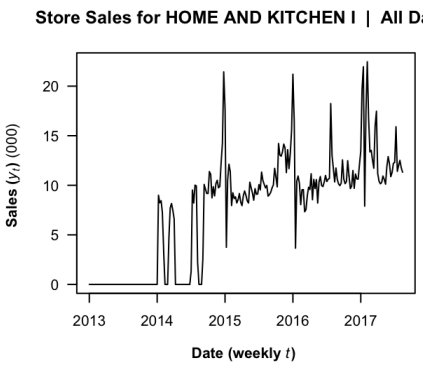

Store Sales for EGGS | All Dates
Sales (y_t) (000)
Date (weekly t)

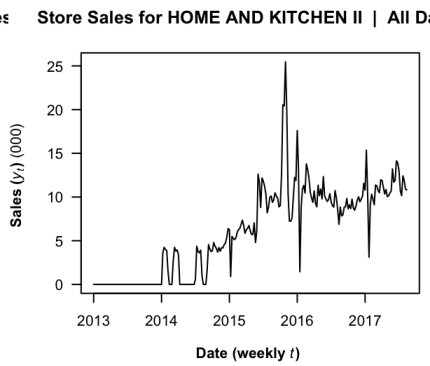

Store Sales for FROZEN FOODS | All Dates
Sales (y_t) (000)
Date (weekly t)

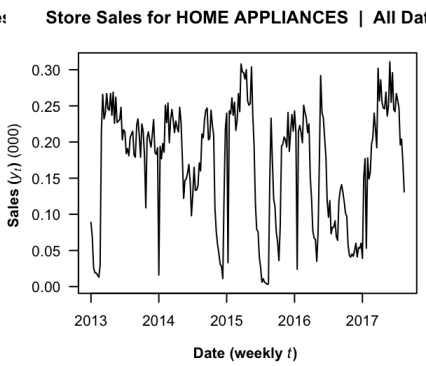

Store Sales for GROCERY I | All Dates
Sales (y_t) (000)
Date (weekly t)

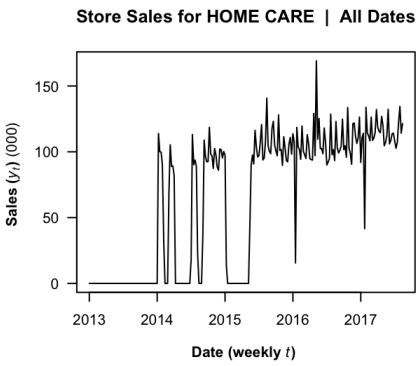

Store Sales for GROCERY II | All Dates
Sales (y_t) (000)
Date (weekly t)

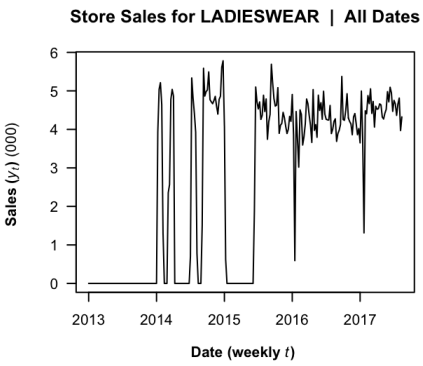

Store Sales for HARDWARE | All Dates
Sales (y_t) (000)
Date (weekly t)

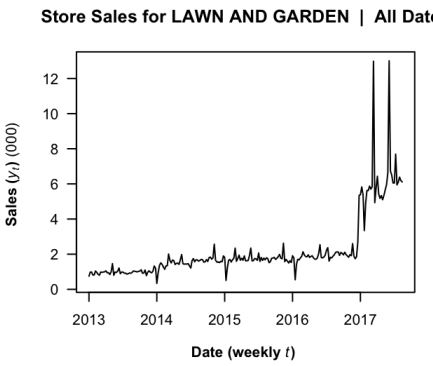

Store Sales for HOME AND KITCHEN I | All Dates
Sales (y_t) (000)
Date (weekly t)

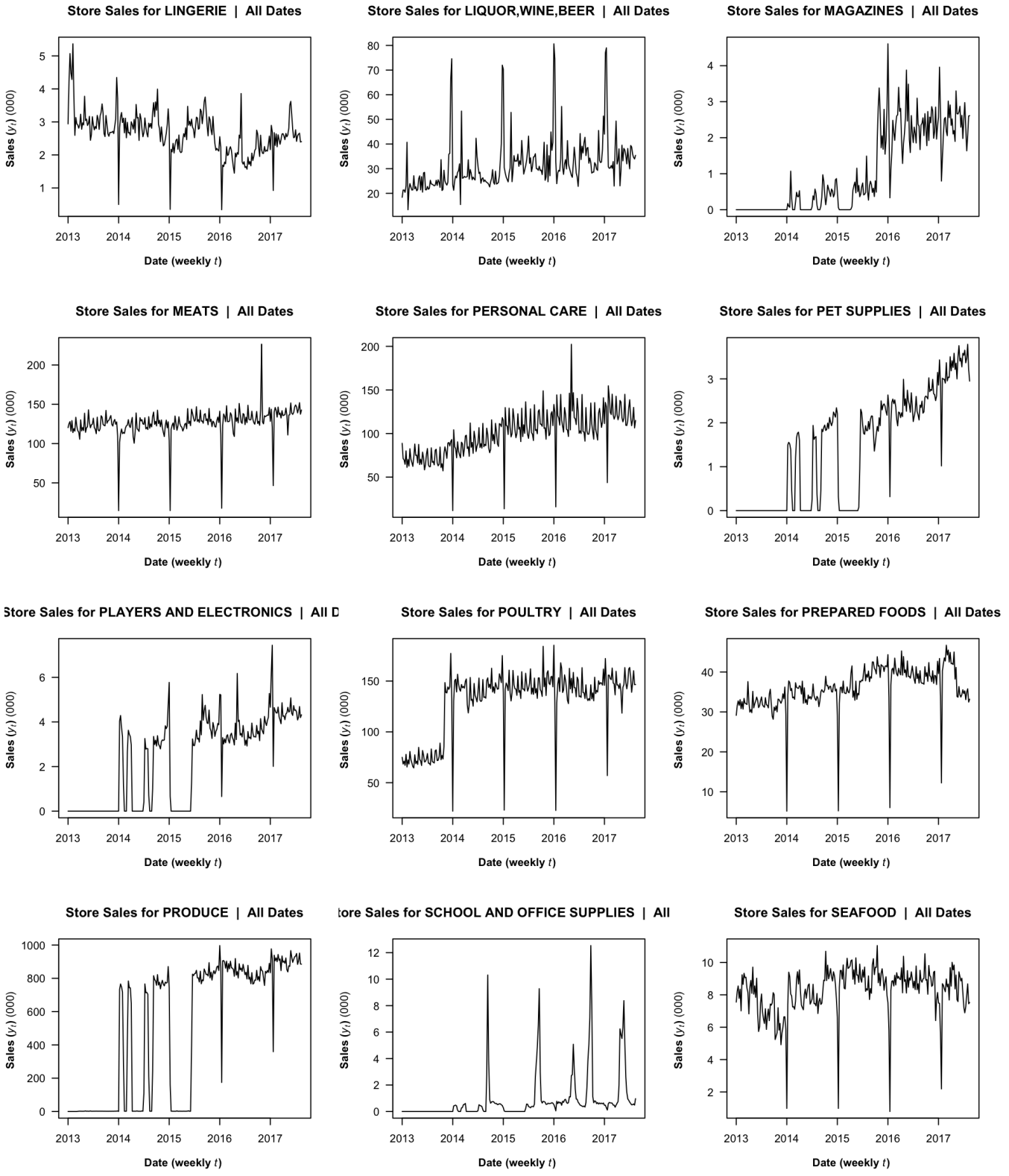

Store Sales for HOME AND KITCHEN II | All Dates
Sales (y_t) (000)
Date (weekly t)


Store Sales for HOME APPLIANCES | All Dates
Sales (y_t) (000)
Date (weekly t)


Store Sales for HOME CARE | All Dates
Sales (y_t) (000)
Date (weekly t)


Store Sales for LADIESWEAR | All Dates
Sales (y_t) (000)
Date (weekly t)


Store Sales for LAWN AND GARDEN | All Dates
Sales (y_t) (000)
Date (weekly t)




```
par(opar)
```

Pre-Process Data

```
# Placeholder code to emove rows with NAs
#df %>%
#  na.omit()

# Placeholder code to remove rows with NAs in specific column
#df %>%
#  filter(!is.na(column_name))

# Placeholder code to reemove duplicates
#df %>%
#  distinct()

# Placeholder code to remove rows by index position
#df %>%
#  filter(!row_number() %in% c(1, 2, 4))

# Remove rows not considered "core"
keep_families <- c("BEAUTY", "BEVERAGES", "BREAD/BAKERY", "CLEANING", "DAIRY",
                   "DELI", "EGGS", "FROZEN FOODS", "GROCERY", "GROCERY II", "HARDWARE",
                   "LIQUOR,WINE,BEER", "MEATS", "PERSONAL CARE", "PET SUPPLIES", "POULTRY",
                   "PREPARED FOODS", "PRODUCE", "SEAFOOD")
remove_families <- c("AUTOMOTIVE", "BABY CARE", "BOOKS", "CELEBRATION", "HOME APPLIANCES",
                     "HOME AND KITCHEN I", "HOME AND KITCHEN II", "HOME CARE", "LADIESWEAR",
                     "LAWN AND GARDEN", "LINGERIE", "MAGAZINES", "PET SUPPLIES",
                     "PLAYERS AND ELECTRONICS", "SCHOOL AND OFFICE SUPPLIES")
sales_core_df <- (sales_df %>%
                    filter(family %in% keep_families))
```

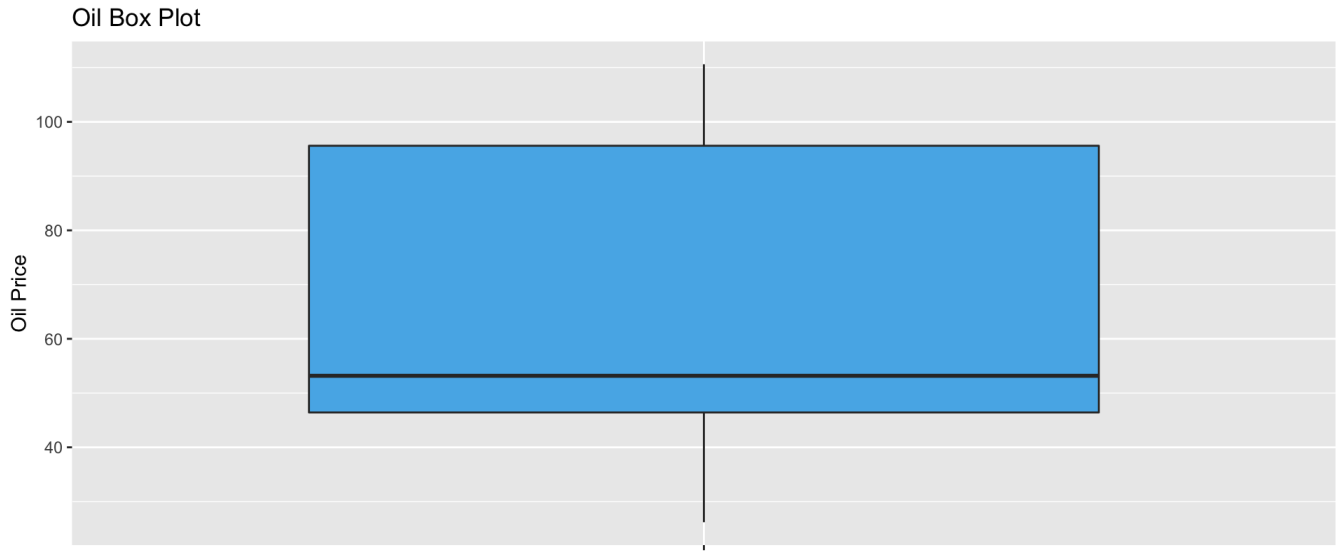
```
# Mean of missing oil_df data
mean(is.na(oil_df))
```

```
[1] 0.02
```

```
# Option: Drop missing records since they only account for 2% of data
# na.omit(oil_df)
# oil_df2: New 'oil_df' data frame without missing values
# Foward and backward fill for missing values
oil_df2 <- na.locf(oil_df, fromLast = TRUE)
# Confirm oil_df2 has no more missing values
sum(is.na(oil_df2))
```

```
[1] 0
```

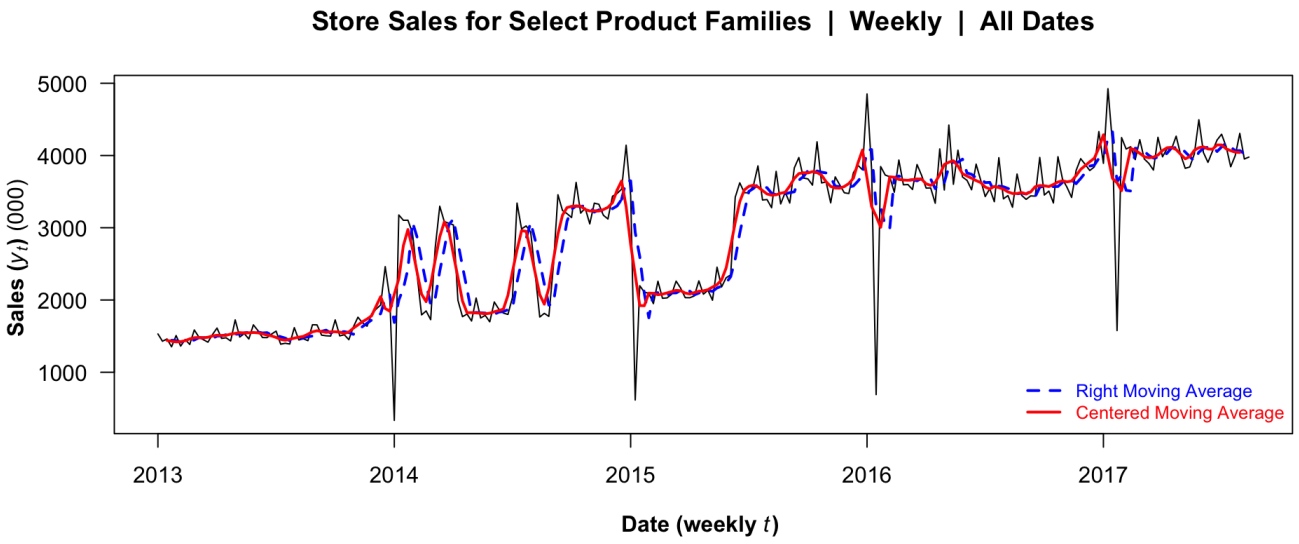
```
# Box plot: oil_df2
oil_outliers <- ggplot(oil_df2) +
                aes(x = "", y = dcoilwtico) +
                geom_boxplot(fill = "#56B4E9") +
                labs(title = "Oil Box Plot",
                     x = "",
                     y = "Oil Price",
                     bty = "l")
oil_outliers
```




```
# Convert string mm/dd/yyyy to Date values and confirm sort
oil_df <- (oil_df %>%
             mutate(date = as.Date(date, format = "%Y-%m-%d"),) %>%
             arrange(date))

# Add oil prices to sales dataset for potential use as external predictor
sales_core_df <- left_join(sales_core_df, oil_df, by = c("date"))
```

Series Visualization - Select ("Core") Product Families

```
# Re-aggregate base dataframe from daily to weekly for all product families
sales_wk_df <- as.data.frame(sales_core_df %>%
                               mutate(year = year(date), week = week(date)) %>%
                               group_by(year, week) %>%
                               summarize(sales = sum(sales / 1000.0), dcoilwtico = mean(dcoilwtico, na.rm = TRUE)))

# Re-create overall time series
sales_begin_year <- head(sales_wk_df$year, 1)
sales_begin_week <- head(sales_wk_df$week, 1)
sales_end_year <- tail(sales_wk_df$year, 1)
sales_end_week <- tail(sales_wk_df$week, 1)
sales_ts <- ts(sales_wk_df$sales,
               start = c(sales_begin_year, sales_begin_week),
               end = c(sales_end_year, sales_end_week), freq = 52)

# Plot overall time series with moving averages
plot(sales_ts, type = "l",
     main = "Store Sales for Select Product Families  |  Weekly  |  All Dates",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7)

sales_l_ma <- rollmean(sales_ts, k = 4, align = "right")
sales_c_ma <- ma(sales_ts, order = 4)
lines(sales_l_ma, lwd = 2, lty = 2, col = "blue")
lines(sales_c_ma, lwd = 2, lty = 1, col = "red")

legend("bottomright",
       legend = c("Right Moving Average", "Centered Moving Average"),
       col = c("blue", "red"),
       lwd = 2, lty = c(2, 1), cex = 0.8,
       box.lty = 0, bg = NULL, text.col = c("blue", "red"))
```



Partition Series

```
# Use one year (52 weeks) as validation period (representative set of quarters, seasons)
sales_n_valid <- 52
sales_n_train <- length(sales_ts) - sales_n_valid

# Split data into training and validation periods
sales_train_ts <- window(sales_ts, start = c(sales_begin_year, 1), end = c(sales_begin_year, sales_n_train))
sales_valid_ts <- window(sales_ts, start = c(sales_begin_year, sales_n_train + 1), end = c(sales_begin_year, sales_n_train +
sales_n_valid))

# Set x axis values based on ts ranges
x_train <- sales_begin_year
x_valid <- sales_begin_year + ((sales_n_train + 1) / 52)
x_future <- x_valid + ((sales_n_valid + 1) / 52)
```

Apply Forecasting Method(s)

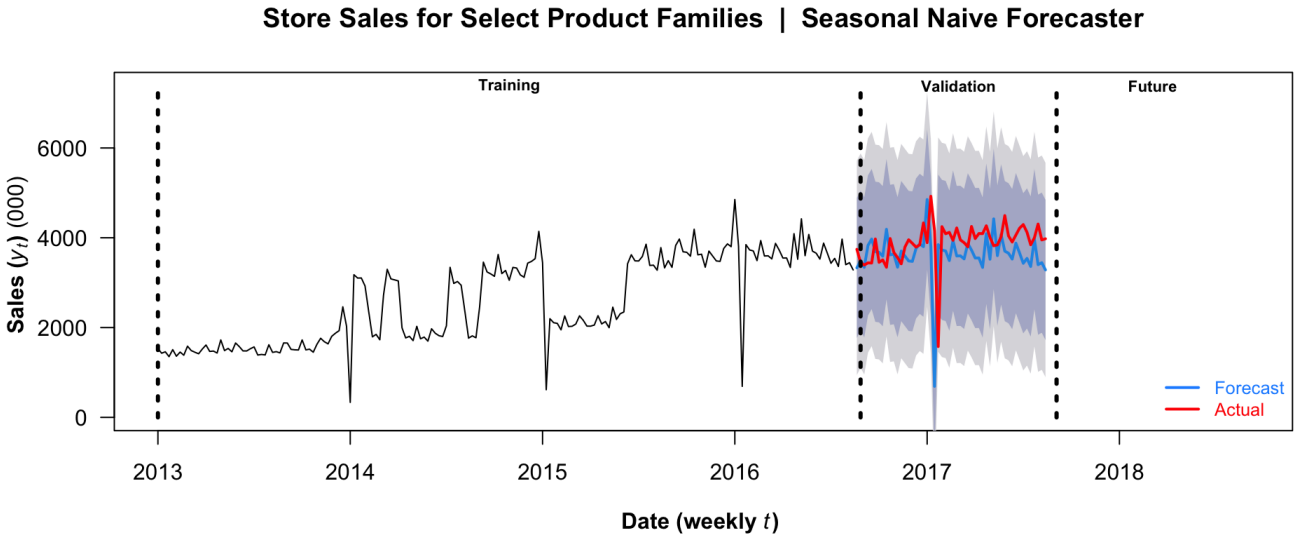
```
# Initialize dataframe for tracking results across model approaches
results_df <- data.frame()
```

Naive Forecast Baseline

```
# Fit seasonal naive model
sales_snaive_pred <- snaive(sales_train_ts, h = sales_n_valid)

# Summarize seasonal naive forecast results
accuracy(sales_snaive_pred, sales_valid_ts)
```

```
                ME RMSE MAE MPE MAPE MASE ACF1 Theil's U
Training set 784 1217 996  17   38  1.0  0.5        NA
Test set     261  752 507   4   14  0.5 -0.2       0.7
```

```
results_df <- rbind(results_df, data.frame(Model = "Naive (baseline)",
                                           Set = c("Train", "Test"),
                                           accuracy(sales_snaive_pred, sales_valid_ts)))

# Plot seasonal naive forecaster
ymax <- max(sales_ts) * 1.5
xmax <- x_future + 1

plot(sales_snaive_pred, type = "l",
     main = "Store Sales for Select Product Families  |  Seasonal Naive Forecaster",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7,
     xlim = c(x_train, xmax), ylim = c(0, ymax))

# Add additional lines
lines(sales_valid_ts, lwd = 2, col = "red")

# Add plot defintions
lines(c(x_train, x_train), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_valid, x_valid), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_future, x_future), c(0, ymax), lwd = 3, lty = 3)

text(x_train + ((x_valid - x_train) * 0.5), ymax, "Training", font = 2, cex = 0.7)
text(x_valid + ((x_future - x_valid) * 0.5), ymax, "Validation", font = 2, cex = 0.7)
text(x_future + 0.5, ymax, "Future", font = 2, cex = 0.7)

legend("bottomright",
       legend = c("Forecast", "Actual"),
       col = c("dodgerblue", "red"),
       lwd = 2, lty = 1, cex = 0.8,
       box.lty = 0, bg = NULL, text.col = c("dodgerblue", "red"))
```

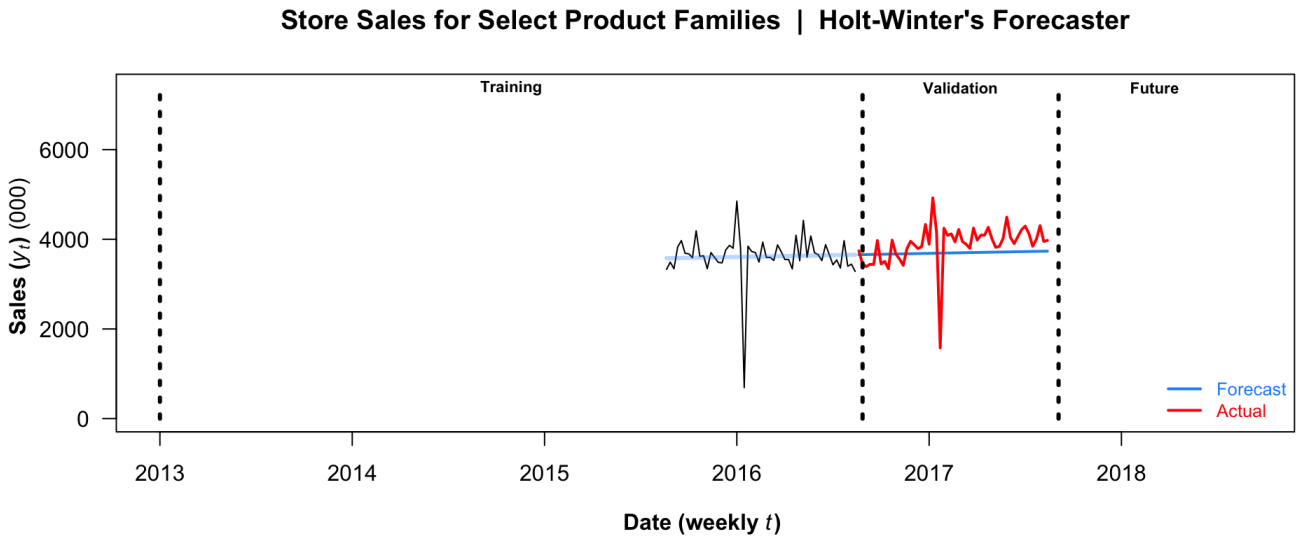


Data-Driven: Smoothing Methods

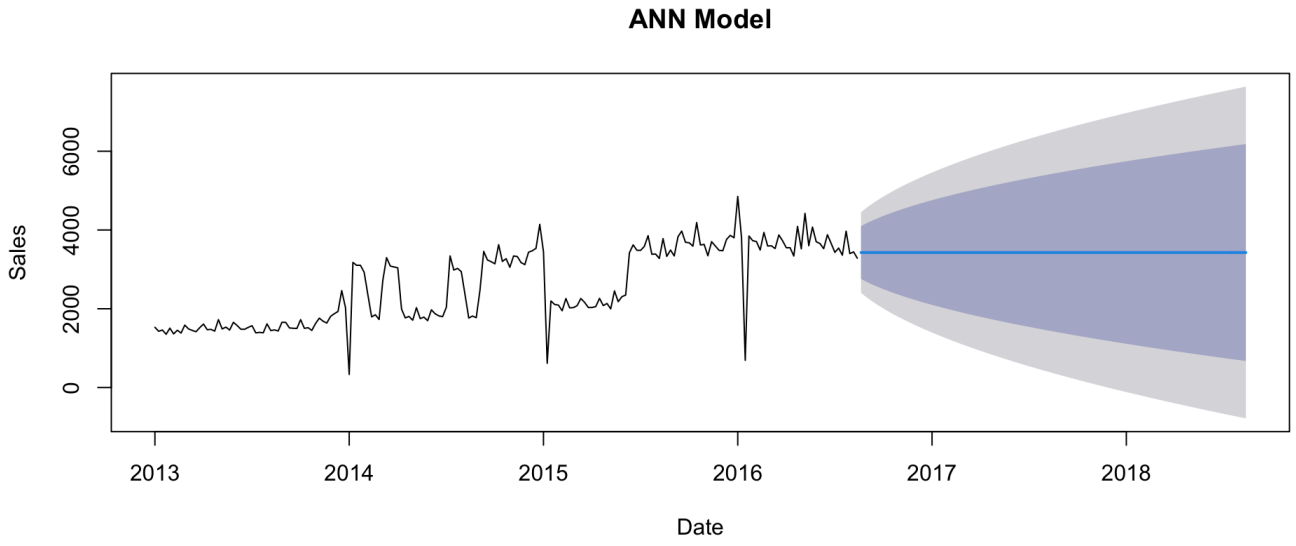
```
# Fit Holt-Winter's model with (A)dditive error, (A)dditive trend, and (A)dditive seasonality
#   for last year of training series
sales_hw <- ets(window(sales_train_ts, start = c(sales_begin_year, sales_n_train + 1 - sales_n_valid),
                       end = c(sales_begin_year, sales_n_train + 1)),
                model = "AAA")

# Forecast (predict) for validation period
sales_hw_pred <- forecast(sales_hw, h = length(sales_valid_ts), level = 0)

# Summarize forecast (validation) performance
accuracy(sales_hw_pred, sales_valid_ts)
```

```
              ME RMSE MAE MPE MAPE MASE  ACF1 Theil's U
Training set  14  501 260  -7   13  NaN -0.07        NA
Test set     195  478 353   3   10  NaN  0.05       0.5
```

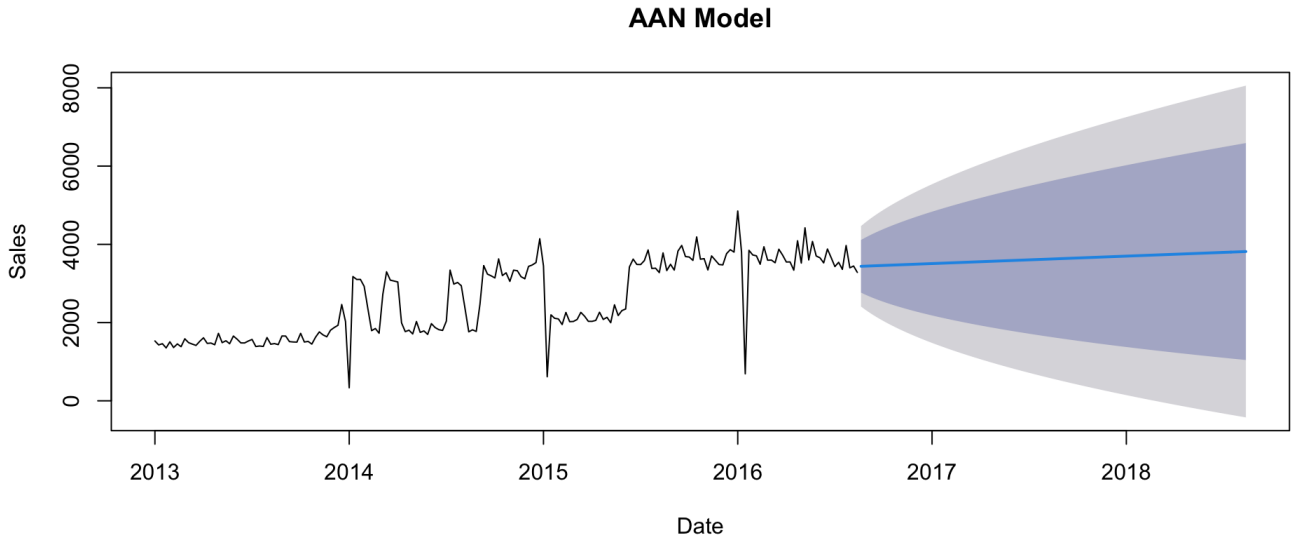
```
results_df <- rbind(results_df, data.frame(Model = "Holt-Winter's Smoothing",
                                           Set = c("Train", "Test"),
                                           accuracy(sales_hw_pred, sales_valid_ts)))

# Plot Holt-Winter's model
plot(sales_hw_pred, type = "l",
     main = "Store Sales for Select Product Families  |  Holt-Winter's Forecaster",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7,
     xlim = c(x_train, xmax), ylim = c(0, ymax))

# Add additional lines
lines(sales_hw$fitted, lwd = 3, col = adjustcolor("dodgerblue", alpha.f = 0.3))
lines(sales_valid_ts, lwd = 2, col = "red")

# Add plot defintions
lines(c(x_train, x_train), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_valid, x_valid), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_future, x_future), c(0, ymax), lwd = 3, lty = 3)

text(x_train + ((x_valid - x_train) * 0.5), ymax, "Training", font = 2, cex = 0.7)
text(x_valid + ((x_future - x_valid) * 0.5), ymax, "Validation", font = 2, cex = 0.7)
text(x_future + 0.5, ymax, "Future", font = 2, cex = 0.7)

legend("bottomright",
       legend = c("Forecast", "Actual"),
       col = c("dodgerblue", "red"),
       lwd = 2, lty = 1, cex = 0.8,
       box.lty = 0, bg = NULL, text.col = c("dodgerblue", "red"))
```



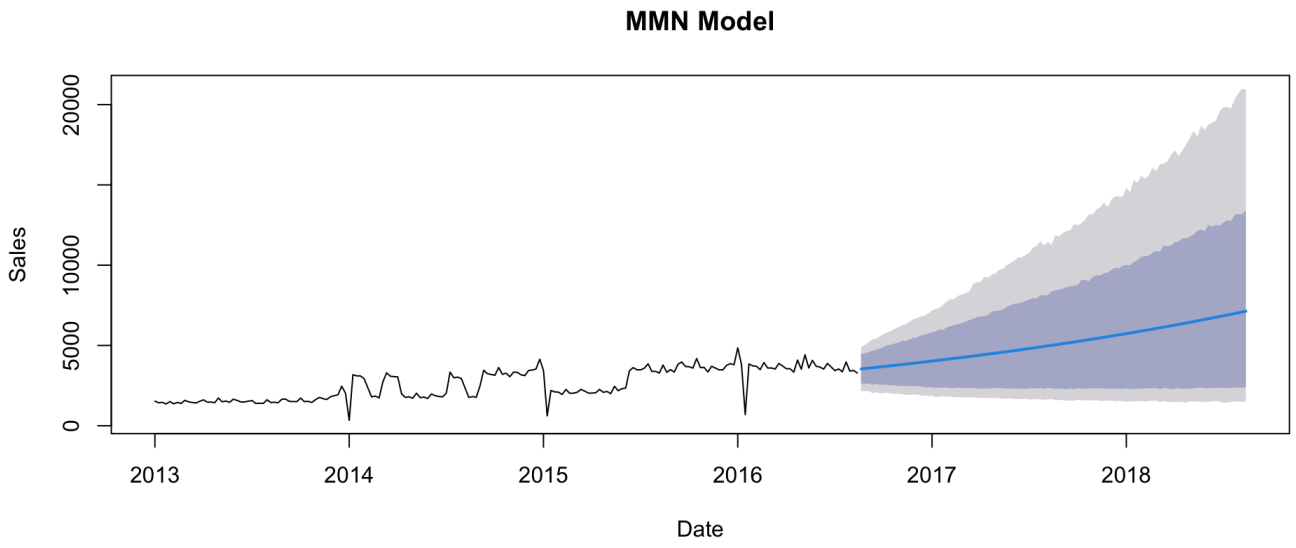

```
# Exponential smoothing models:
# ANN: (A)dditive error, (N)o trend, and (N)o seasonality
# AAN: (A)dditive error, (A)dditive trend, and (N)o seasonality
# MMN: (M)ultiplicative error, (M)ultiplicative trend, and (N)o seasonality
# MMdN: Multiplicative damped trend (Md)

sales.ets.ANN <- ets(sales_train_ts, model = "ANN")
sales.ets.AAN <- ets(sales_train_ts, model = "AAN")
sales.ets.MMN <- ets(sales_train_ts, model = "MMN", damped = FALSE)
sales.ets.MMdN <- ets(sales_train_ts, model = "MMN", damped = TRUE)

sales.ets.ANN.pred <- forecast(sales.ets.ANN)
sales.ets.AAN.pred <- forecast(sales.ets.AAN)
sales.ets.MMN.pred <- forecast(sales.ets.MMN)
sales.ets.MMdN.pred <- forecast(sales.ets.MMdN)

# Plot exponential smoothing models
plot(sales.ets.ANN.pred, main = "ANN Model", xlab = "Date", ylab = "Sales")
```



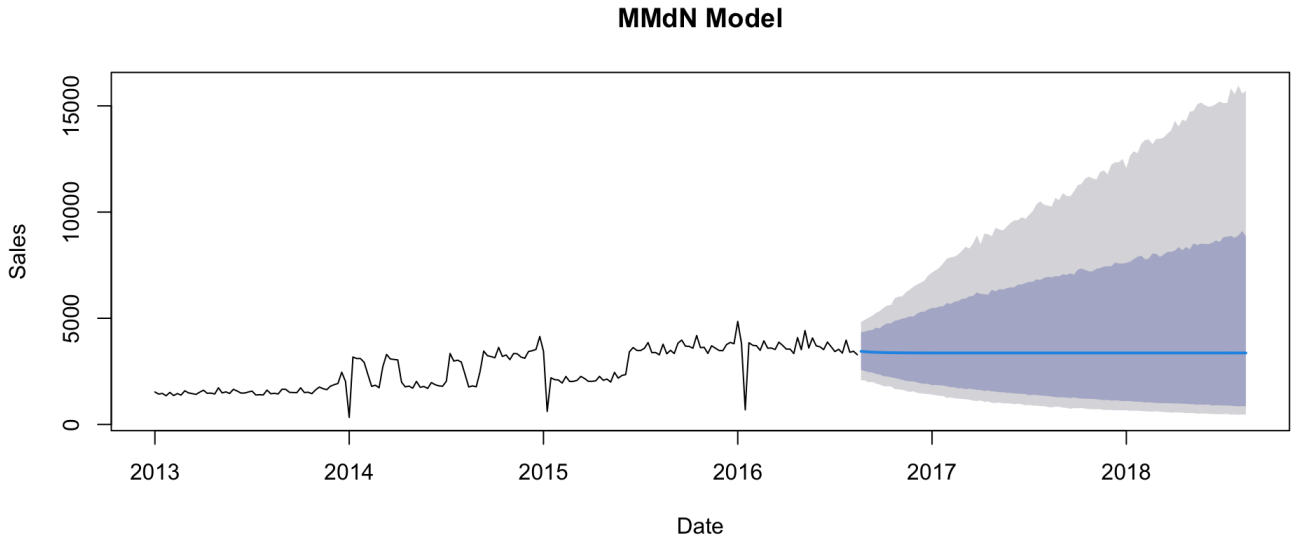

```
plot(sales.ets.AAN.pred, main = "AAN Model", xlab = "Date", ylab = "Sales")
```




```
plot(sales.ets.MMN.pred, main = "MMN Model", xlab = "Date", ylab = "Sales")
```




```
plot(sales.ets.MMdN.pred, main = "MMdN Model", xlab = "Date", ylab = "Sales")
```




```
# Summarize forecast (validation) performance
accuracy(sales.ets.ANN.pred, sales_valid_ts)
```

```
              ME RMSE MAE MPE MAPE MASE ACF1 Theil's U
Training set  26  519 295  -7   17  0.3 0.12        NA
Test set     464  643 540  10   15  0.5 0.09       0.7
```

```
accuracy(sales.ets.AAN.pred, sales_valid_ts)
```

```
              ME RMSE MAE MPE MAPE MASE  ACF1 Theil's U
Training set  18  519 296  -7   17  0.3 0.125        NA
Test set     359  557 445   7   12  0.4 0.002       0.6
```

```
accuracy(sales.ets.MMN.pred, sales_valid_ts)
```

```
               ME RMSE MAE MPE MAPE MASE ACF1 Theil's U
Training set  -22  523 310  -9   18  0.3  0.2        NA
Test set     -339  589 423 -11   13  0.4  0.2       0.6
```

```
accuracy(sales.ets.MMdN.pred, sales_valid_ts)
```

```
             ME RMSE MAE MPE MAPE MASE ACF1 Theil's U
Training set  13  528 306  -7   18  0.3  0.2        NA
Test set     518  686 589  11   16  0.6  0.1       0.7
```

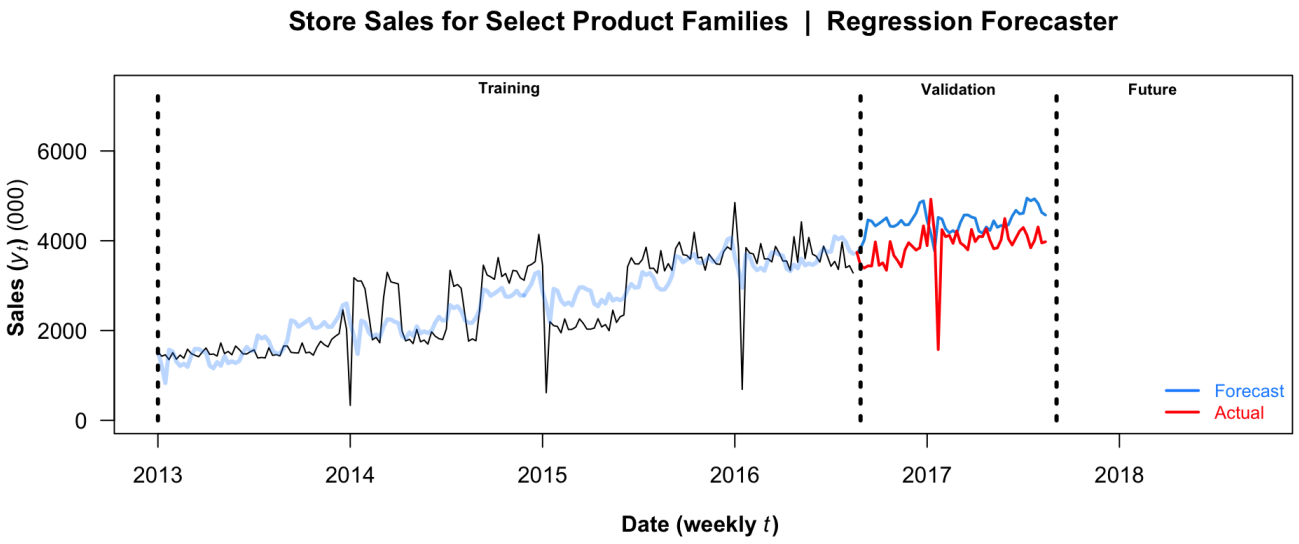
```
results_df <- rbind(results_df, data.frame(Model = "Exponential Smoothing (AAN)",
                                           Set = c("Train", "Test"),
                                           accuracy(sales.ets.ANN.pred, sales_valid_ts)))
results_df <- rbind(results_df, data.frame(Model = "Exponential Smoothing (AAN)",
                                           Set = c("Train", "Test"),
                                           accuracy(sales.ets.AAN.pred, sales_valid_ts)))
results_df <- rbind(results_df, data.frame(Model = "Exponential Smoothing (MMN)",
                                           Set = c("Train", "Test"),
                                           accuracy(sales.ets.MMN.pred, sales_valid_ts)))
results_df <- rbind(results_df, data.frame(Model = "Exponential Smoothing (MMdN)",
                                           Set = c("Train", "Test"),
                                           accuracy(sales.ets.MMdN.pred, sales_valid_ts)))
```

Model-Based: Regression

```
# Fit basic regression model with trend and seasonality
sales_lm <- tslm(sales_train_ts ~ trend + I(trend ^ 2) + season)

# Forecast (predict) for validation period
sales_lm_pred <- forecast(sales_lm, h = sales_n_valid, level = 0)

# Summarize performance metrics
accuracy(sales_lm_pred, sales_valid_ts)
```

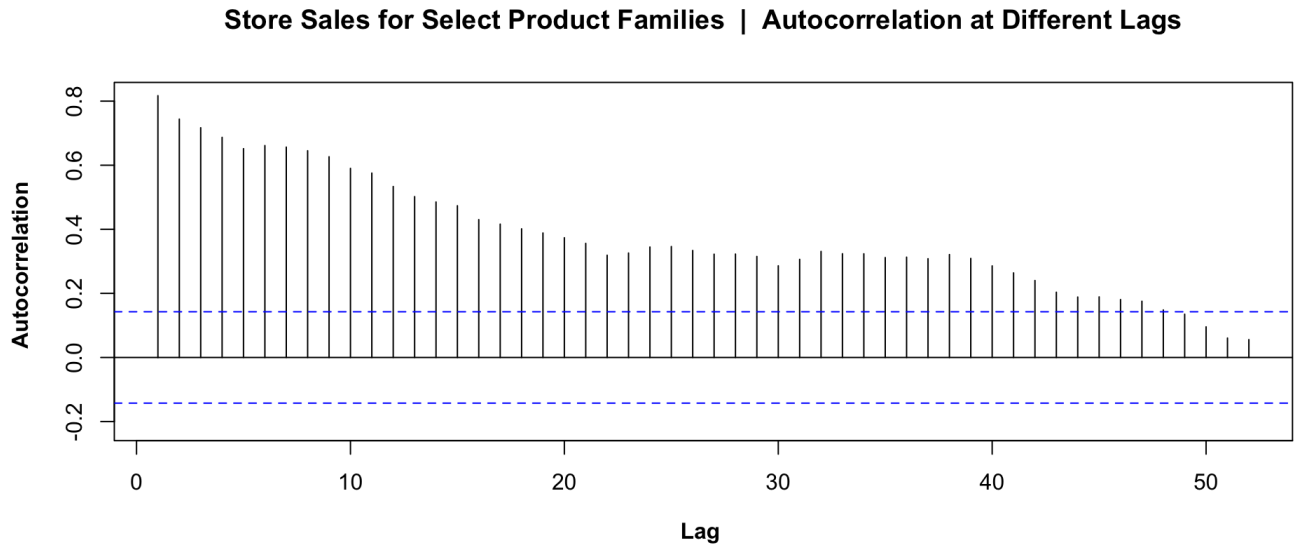
```
                            ME RMSE MAE MPE MAPE MASE ACF1 Theil's U
Training set   -0.000000000000005  525 393  -8   21  0.4 0.47        NA
Test set     -529.415459820942715  735 583 -16   17  0.6 0.06       0.7
```

```
results_df <- rbind(results_df, data.frame(Model = "Simple Regression",
                                           Set = c("Train", "Test"),
                                           accuracy(sales_lm_pred, sales_valid_ts)))

# Plot simple regression model
plot(sales_lm_pred, type = "l",
     main = "Store Sales for Select Product Families  |  Regression Forecaster",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7,
     xlim = c(x_train, xmax), ylim = c(0, ymax))

# Add additional lines
lines(sales_lm$fitted, lwd = 3, col = adjustcolor("dodgerblue", alpha.f = 0.3))
lines(sales_valid_ts, lwd = 2, col = "red")

# Add plot defintions
lines(c(x_train, x_train), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_valid, x_valid), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_future, x_future), c(0, ymax), lwd = 3, lty = 3)

text(x_train + ((x_valid - x_train) * 0.5), ymax, "Training", font = 2, cex = 0.7)
text(x_valid + ((x_future - x_valid) * 0.5), ymax, "Validation", font = 2, cex = 0.7)
text(x_future + 0.5, ymax, "Future", font = 2, cex = 0.7)

legend("bottomright",
       legend = c("Forecast", "Actual"),
       col = c("dodgerblue", "red"),
       lwd = 2, lty = 1, cex = 0.8,
       box.lty = 0, bg = NULL, text.col = c("dodgerblue", "red"))
```



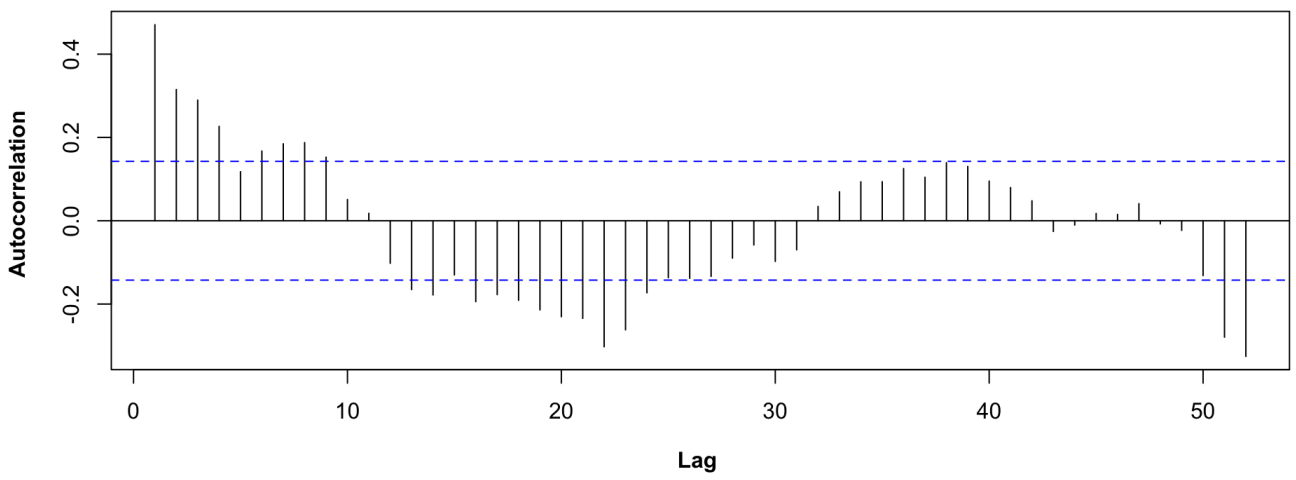
Model-Based: Regression w/Autocorrelation

```
# Compute and plot autocorrelation
Acf(sales_train_ts, lag.max = 52, type = "correlation", plot = TRUE,
    main = "Store Sales for Select Product Families  |  Autocorrelation at Different Lags",
    xlab = TeX(r"(\textbf{Lag} )"), ylab = TeX(r"(\textbf{Autocorrelation} )"),
    xaxt = "n",
    level = 95)
```



Model-Based: Regression w/Autocorrelation - AR(1)

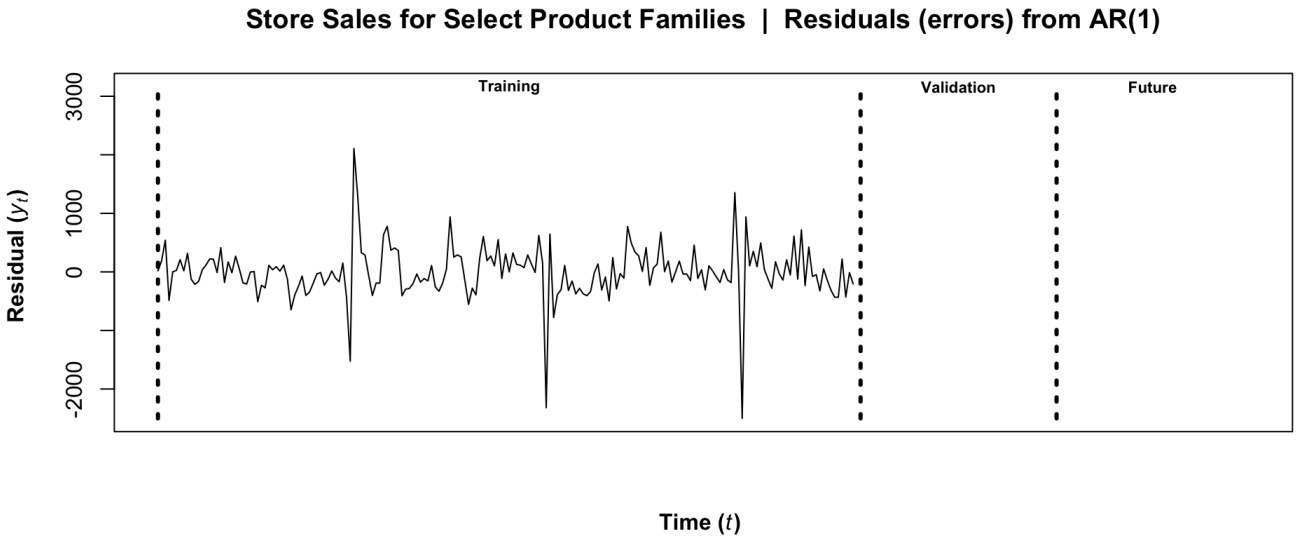
```
# Compute and plot autocorrelation for regression residuals
Acf(sales_lm$residuals, lag.max = 52, type = "correlation", plot = TRUE,
    main = "Regression Residuals  |  Autocorrelation at Different Lags",
    xlab = TeX(r"(\textbf{Lag} )"), ylab = TeX(r"(\textbf{Autocorrelation} )"),
    xaxt = "n",
    level = 95)
```



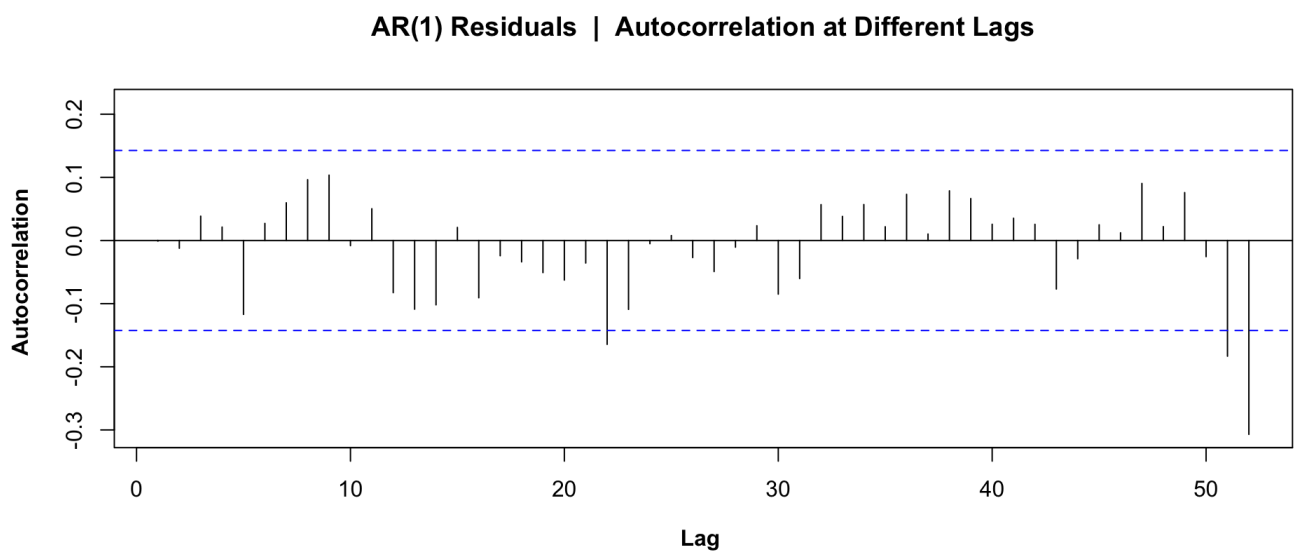

```
# Fit AR(1) second-layer model to regression residuals and forecast (predict) vs. validation;
#   use AR of 1, MA of 0, and integration (differencing) of 0
sales_res_arima <- Arima(sales_lm$residuals, order = c(1, 0, 2))
sales_res_pred <- forecast(sales_res_arima, h = sales_n_valid)

# Summarize forecast (validation) performance
accuracy(sales_res_pred, sales_valid_ts)
```

```
               ME RMSE  MAE MPE MAPE MASE   ACF1 Theil's U
Training set   -1  455  288 278  328  0.4 -0.001        NA
Test set     3926 3949 3926 101  101  5.1  0.015         4
```

```
results_df <- rbind(results_df, data.frame(Model = "AR(1)",
                                           Set = c("Train", "Test"),
                                           accuracy(sales_res_pred, sales_valid_ts)))

rymin <- min(sales_res_pred$residuals)
rymax <- max(sales_res_pred$residuals) * 1.5

# Plot residuals
plot(sales_res_pred$residuals, type = "l",
     main = "Store Sales for Select Product Families  |  Residuals (errors) from AR(1)",
     xlab = TeX(r"(\textbf{Time (\textit{$t$})} )"), ylab = TeX(r"(\textbf{Residual (\textit{$y_t$})} )"),
     xaxt = "n",
     xlim = c(x_train, xmax), ylim = c(rymin, rymax))

# Add plot defintions
lines(c(x_train, x_train), c(rymin, rymax), lwd = 3, lty = 3)
lines(c(x_valid, x_valid), c(rymin, rymax), lwd = 3, lty = 3)
lines(c(x_future, x_future), c(rymin, rymax), lwd = 3, lty = 3)

text(x_train + ((x_valid - x_train) * 0.5), rymax, "Training", font = 2, cex = 0.7)
text(x_valid + ((x_future - x_valid) * 0.5), rymax, "Validation", font = 2, cex = 0.7)
text(x_future + 0.5, rymax, "Future", font = 2, cex = 0.7)
```




```
# Compute and plot autocorrelation for residual of residuals
Acf(sales_res_arima$residuals, lag.max = 52, type = "correlation", plot = TRUE,
    main = "AR(1) Residuals  |  Autocorrelation at Different Lags",
    xlab = TeX(r"(\textbf{Lag} )"), ylab = TeX(r"(\textbf{Autocorrelation} )"),
    xaxt = "n",
    level = 95)
```

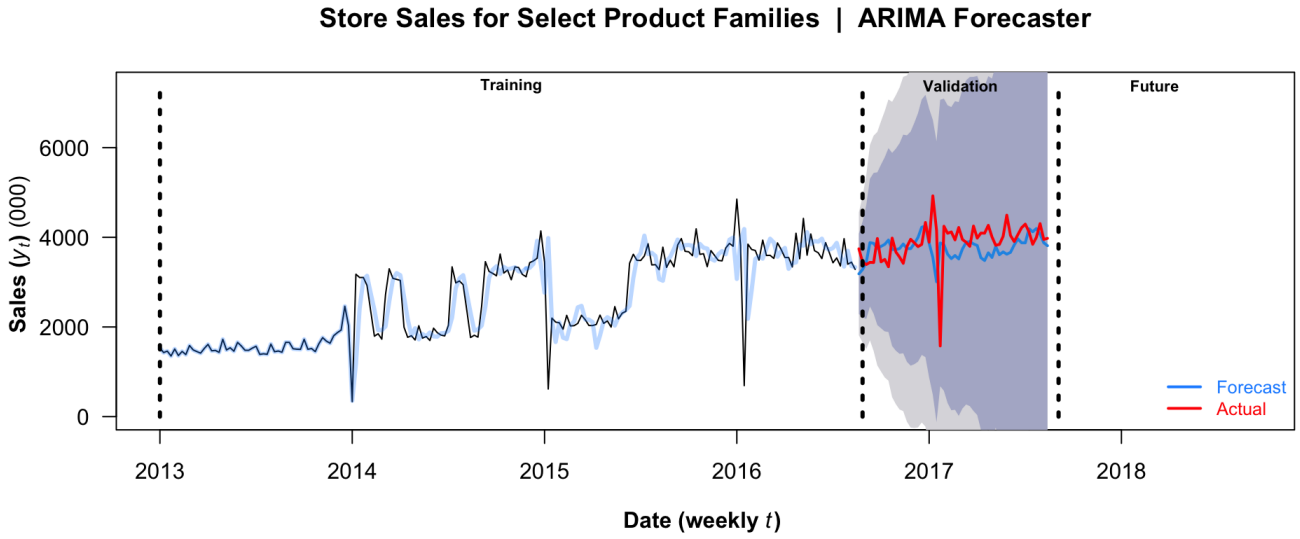


Model-Based: Regression w/Autocorrelation - ARIMA

```
# Fit ARIMA model
sales_arima <- Arima(sales_train_ts, order = c(1, 1, 0), seasonal = c(0, 1, 2))
sales_arima_pred <- forecast(sales_arima, h = length(sales_valid_ts))

# Summarize forecast (validation) performance
accuracy(sales_arima_pred, sales_valid_ts)
```

```
              ME RMSE MAE  MPE MAPE MASE  ACF1 Theil's U
Training set   9  513 249 -4.8   13  0.2 -0.07        NA
Test set     125  534 371  0.9   11  0.4  0.06       0.5
```

```
results_df <- rbind(results_df, data.frame(Model = "ARIMA",
                                           Set = c("Train", "Test"),
                                           accuracy(sales_arima_pred, sales_valid_ts)))

# Plot model
plot(sales_arima_pred, type = "l",
     main = "Store Sales for Select Product Families  |  ARIMA Forecaster",
     xlab = TeX(r"(\textbf{Date (weekly \textit{$t$})} )"), ylab = TeX(r"(\textbf{Sales (\textit{$y_t$})} (000) )"),
     las = 1, cex.axis = 0.7,
     xlim = c(x_train, xmax), ylim = c(0, ymax))

# Add additional lines
lines(sales_arima$fitted, lwd = 3, col = adjustcolor("dodgerblue", alpha.f = 0.3))
lines(sales_valid_ts, lwd = 2, col = "red")

# Add plot defintions
lines(c(x_train, x_train), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_valid, x_valid), c(0, ymax), lwd = 3, lty = 3)
lines(c(x_future, x_future), c(0, ymax), lwd = 3, lty = 3)

text(x_train + ((x_valid - x_train) * 0.5), ymax, "Training", font = 2, cex = 0.7)
text(x_valid + ((x_future - x_valid) * 0.5), ymax, "Validation", font = 2, cex = 0.7)
text(x_future + 0.5, ymax, "Future", font = 2, cex = 0.7)

legend("bottomright",
       legend = c("Forecast", "Actual"),
       col = c("dodgerblue", "red"),
       lwd = 2, lty = 1, cex = 0.8,
       box.lty = 0, bg = NULL, text.col = c("dodgerblue", "red"))
```



Model-Based: Regression w/Autocorrelation - ARIMA + External Predictor(r)

```
# Setup train and validation
train_span <- sales_n_train / 52
valid_span <- sales_n_valid / 52
total_span <- train_span + valid_span
train_end <- ((sales_begin_year + trunc(train_span)) * 100) + ((train_span - floor(train_span)) * 52)
valid_end <- ((sales_begin_year + trunc(total_span)) * 100) + ((total_span - floor(total_span)) * 52)
sales_train_df <- as.data.frame(sales_wk_df %>%
                                filter(((year * 100) + week) <= train_end))
sales_valid_df <- as.data.frame(sales_wk_df %>%
                                filter(((year * 100) + week) > train_end))

# Identify outcome (target) - sales
sales_train_y <- ts(sales_train_df$sales, freq = 52)
sales_valid_y <- ts(sales_valid_df$sales, freq = 52)

# Identify external predictor(s) - oil price
sales_X <- c("dcoilwtico")
sales_train_X <- as.matrix(sales_train_df[ , sales_X])
sales_valid_X <- as.matrix(sales_valid_df[ , sales_X])

# Fit using auto.arima for sense of best component configuration, and summarize results both without and with external
#   predictor(s)
sales_aam <- auto.arima(sales_train_y)
sales_aamX <- auto.arima(sales_train_y, xreg = sales_train_X)
summary(sales_aam)
```

```
Series: sales_train_y
ARIMA(0,1,2)(0,0,1)[52]

Coefficients:
        ma1    ma2  sma1
      -0.42  -0.26  -0.3
s.e.   0.07   0.07   0.1

sigma^2 = 246511:  log likelihood = -1457
AIC=2923   AICc=2923   BIC=2936

Training set error measures:
             ME RMSE MAE MPE MAPE MASE  ACF1
Training set 42  491 300  -5   16  0.3 0.004
```

```
summary(sales_aamX)
```

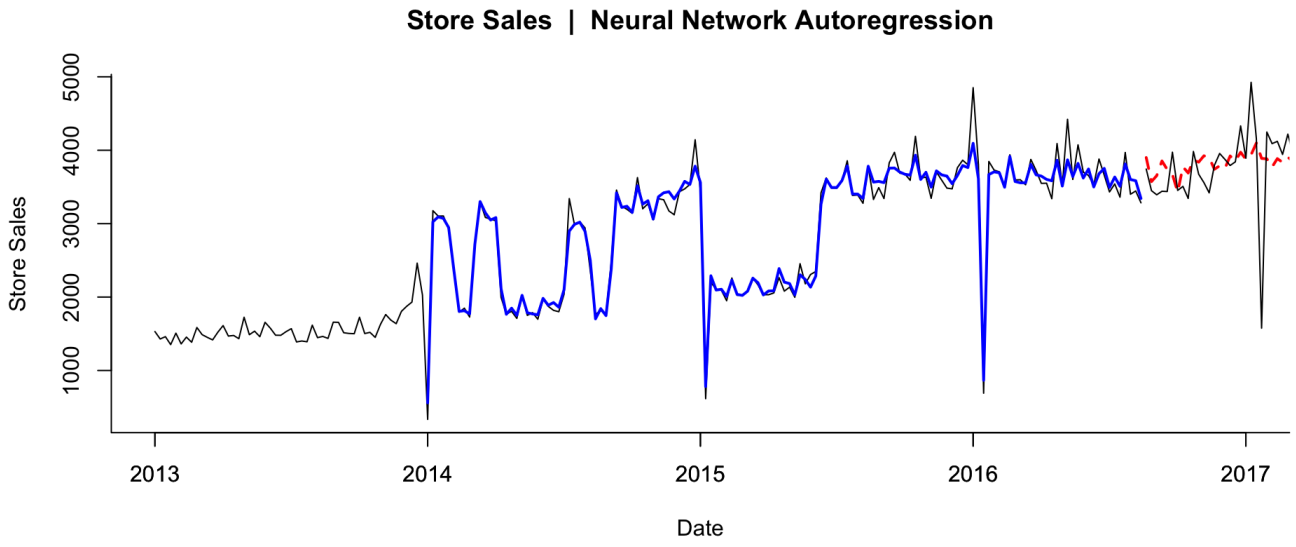
```
Series: sales_train_y
Regression with ARIMA(0,1,2)(0,0,1)[52] errors

Coefficients:
        ma1    ma2  sma1  drift  xreg
      -0.43  -0.28  -0.3     13     4
s.e.   0.07   0.07   0.1      9     9

sigma^2 = 245991:  log likelihood = -1456
AIC=2925   AICc=2925   BIC=2944

Training set error measures:
             ME RMSE MAE MPE MAPE MASE ACF1
Training set -1  488 307  -7   17  0.3 0.02
```

Model-Based: Neural Network Autoregression

```
#
set.seed(201)
nValid <- 36

# Fit Neural Network Autoregression model
sales_train.nnetar <- nnetar(sales_train_ts, repeats = 20, p = 10, P = 1, size = 7)
summary(sales_train.nnetar$model[[1]])
```

```
a 11-7-1 network with 92 weights
options were - linear output units
  b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1
  -0.37   -6.05   -7.91    0.86    4.15   -0.35    1.92    0.71   -5.29   -3.03   -2.17    3.24
  b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i9->h2 i10->h2 i11->h2
  13.33   -3.38  -21.20   -4.17   -1.53   -2.54    4.19    9.57   10.12   10.17    7.36   -1.57
  b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i9->h3 i10->h3 i11->h3
  -0.21   11.24    2.32   -3.20    2.28   -3.54    0.24    0.08    4.56   -0.56   -6.32   -2.36
  b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i9->h4 i10->h4 i11->h4
   2.66    0.57    5.96   -0.88  -10.23    3.75    2.45   -1.69   12.19  -10.80    2.77   -8.55
  b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5  i7->h5  i8->h5  i9->h5 i10->h5 i11->h5
   0.04   -1.37   -5.18   -1.31   -4.65   -0.83   -3.93    5.06   -4.12    8.94   -4.56   -4.62
  b->h6  i1->h6  i2->h6  i3->h6  i4->h6  i5->h6  i6->h6  i7->h6  i8->h6  i9->h6 i10->h6 i11->h6
  -3.60    0.92    6.03    5.79   -5.23   -9.16    2.98   -6.32   -5.35  -13.22   -1.61    6.01
  b->h7  i1->h7  i2->h7  i3->h7  i4->h7  i5->h7  i6->h7  i7->h7  i8->h7  i9->h7 i10->h7 i11->h7
  -5.77  -10.02   -3.86   -0.69    5.24    5.62   -0.47    0.81   -4.12    5.79   -2.77   -0.50
  b->o  h1->o  h2->o  h3->o  h4->o  h5->o  h6->o  h7->o
 -3.17  -2.78   3.04   1.53  -0.36  -0.46   2.88   2.64
```

```
sales_train.nnetar.pred <- forecast(sales_train.nnetar, h = nValid)

# Plot model
plot(sales_train_ts, main = "Store Sales  |  Neural Network Autoregression",
     xlab = "Date", ylab = "Store Sales", xlim = c(2013, 2017),
     bty = "l", lty = 1)
axis(1, at = seq(2013, 2017, 1), labels = format(seq(2013, 2017, 1)))

# Add additional lines
lines(sales_train.nnetar.pred$fitted, lwd = 2, col = "blue")
lines(sales_train.nnetar.pred$mean, lwd = 2, col = "red", lty = 2)
lines(sales_valid_ts)
```




```
# Summarize performance
accuracy(sales_train.nnetar.pred, sales_valid_ts)
```

```
                ME RMSE MAE MPE MAPE MASE  ACF1 Theil's U
Training set  0.3  135  85  -1    3 0.09 0.060        NA
Test set     -6.7  485 294  -3   10 0.30 0.005       0.4
```

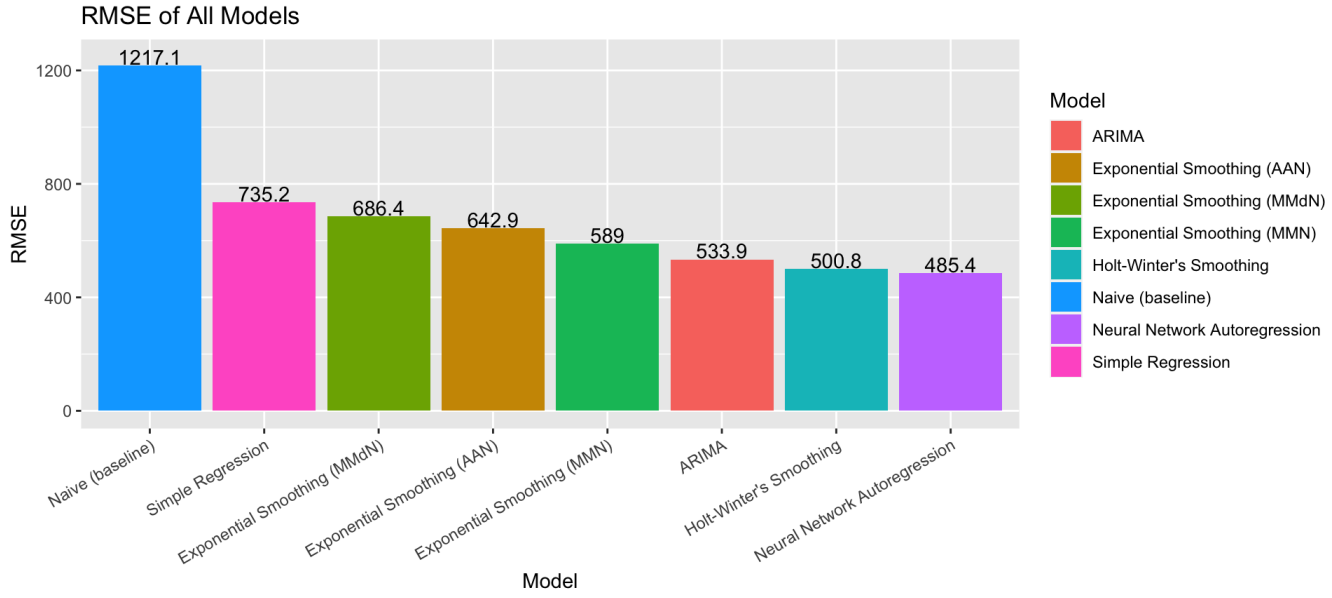
```
results_df <- rbind(results_df, data.frame(Model = "Neural Network Autoregression",
                                           Set = c("Train", "Test"),
                                           accuracy(sales_train.nnetar.pred, sales_valid_ts)))
```

Evaluate & Compare Performance

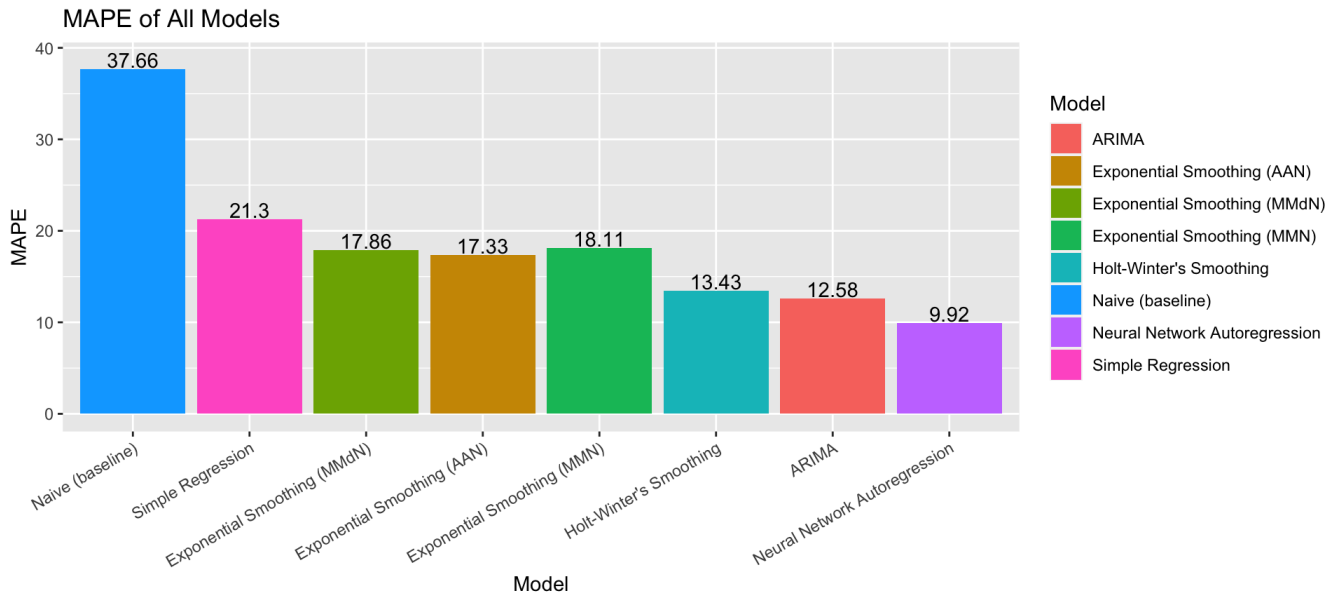
```
# Show table of summary results
print(results_df, row.names = FALSE)
```

Model	Set	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil.s.U
Naive (baseline)	Train	783.614908859564025	1217	996	17.1	38	1.00	0.477	NA
Naive (baseline)	Test	261.373668023741175	752	507	4.3	14	0.51	-0.237	0.7
Holt-Winter's Smoothing	Train	14.310911044518965	501	260	-6.7	13	NaN	-0.069	NA
Holt-Winter's Smoothing	Test	195.100466155941945	478	353	2.9	10	NaN	0.049	0.5
Exponential Smoothing (AAN)	Train	26.387628754067201	519	295	-6.8	17	0.30	0.121	NA
Exponential Smoothing (AAN)	Test	463.951556368900299	643	540	9.9	15	0.54	0.085	0.7
Exponential Smoothing (AAN)	Train	18.013021668575746	519	296	-7.2	17	0.30	0.125	NA
Exponential Smoothing (AAN)	Test	359.483123328557554	557	445	7.2	12	0.45	0.002	0.6
Exponential Smoothing (MMN)	Train	-22.164336330586867	523	310	-8.9	18	0.31	0.192	NA
Exponential Smoothing (MMN)	Test	-339.196046696425185	589	423	-10.6	13	0.43	0.187	0.6
Exponential Smoothing (MMdN)	Train	12.691237758458620	528	306	-7.4	18	0.31	0.209	NA
Exponential Smoothing (MMdN)	Test	518.098516790985059	686	589	11.3	16	0.59	0.106	0.7
Simple Regression	Train	-0.000000000000005	525	393	-8.2	21	0.39	0.471	NA
Simple Regression	Test	-529.415459820942715	735	583	-16.2	17	0.59	0.056	0.7
AR(1)	Train	-1.225461539994329	455	288	278.2	328	0.37	-0.001	NA
AR(1)	Test	3925.739961464764747	3949	3926	100.9	101	5.11	0.015	4.0
ARIMA	Train	8.853827849217511	513	249	-4.8	13	0.25	-0.074	NA
ARIMA	Test	124.533557948679359	534	371	0.9	11	0.37	0.059	0.5
Neural Network Autoregression	Train	0.284419670987839	135	85	-1.0	3	0.09	0.060	NA
Neural Network Autoregression	Test	-6.727065818912592	485	294	-3.2	10	0.30	0.005	0.4

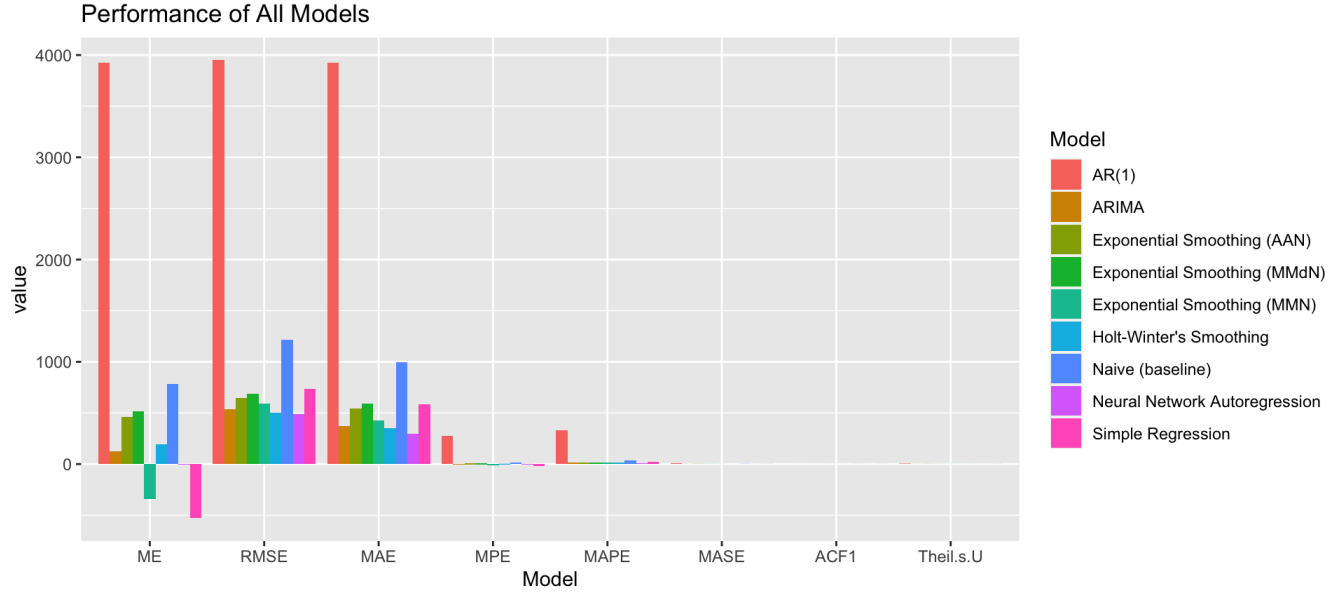
```
# Bar plot: RMSE of all models (Except AR(1))
results_df_reshape1 <- reshape2::melt(results_df, c("Model", "RMSE")) %>%
                       filter(Model != "AR(1)")
results_df_reshape1 %>%
  ggplot() +
  geom_bar(aes(x = reorder(Model, -RMSE), y = RMSE, fill = Model),
           stat = "identity",
           position = "dodge") +
  geom_text(data = results_df_reshape1 %>%
              group_by(Model) %>%
              summarise(RMSE = max(RMSE),
                        label_RMSE = round(sum(RMSE, na.rm = TRUE), 1)),
            aes(x = Model, y = RMSE, label = label_RMSE),
            nudge_y = 30) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1, vjust = 1)) +
  labs(title = "RMSE of All Models",
       x = "Model",
       y = "RMSE",
       bty = "l")
```



```
# Bar plot: MAPE of all models (Except AR(1))
results_df_reshape2 <- reshape2::melt(results_df, c("Model", "MAPE")) %>%
                       filter(Model != "AR(1)")
results_df_reshape2 %>%
  ggplot() +
  geom_bar(aes(x = reorder(Model, -MAPE), y = MAPE, fill = Model),
           stat = "identity",
           position = "dodge") +
  geom_text(data = results_df_reshape2 %>%
              group_by(Model) %>%
              summarise(MAPE = max(MAPE),
                        label_MAPE = round(sum(MAPE, na.rm = TRUE), 2)),
            aes(x = Model, y = MAPE, label = label_MAPE), nudge_y = 1) +
  theme(axis.text.x = element_text(angle = 30, hjust = 1, vjust = 1)) +
  labs(title = "MAPE of All Models",
       x = "Model",
       y = "MAPE",
       bty = "l")
```




```
# Bar plot: Performance of all models
results_df_reshape = reshape2::melt(results_df, c("Model", "Set"))
ggplot(results_df_reshape) +
  geom_bar(aes(x = variable, y = value, fill = Model),
           stat = "identity",
           position = "dodge") +
  labs(title = "Performance of All Models",
       x = "Model",
       bty = "l")
```



1. University of San Diego, cvu@sandiego.edu (mailto:cvu@sandiego.edu)↩
2. University of San Diego, dfriesen@sandiego.edu (mailto:dfriesen@sandiego.edu)↩