Milestone 1

Yesh Onipede

2024-04-11

Setting up data paths

```
setwd('/Users/yeshimonipede/Desktop/BC_Spring2024')
merged_data <- read.csv("Gin_LiquorSales_Education.csv")
```

Loading packages

```
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(ggplot2)
library(tidyr)
library(dplyr)
```

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

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

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

```
library(cowplot)
```

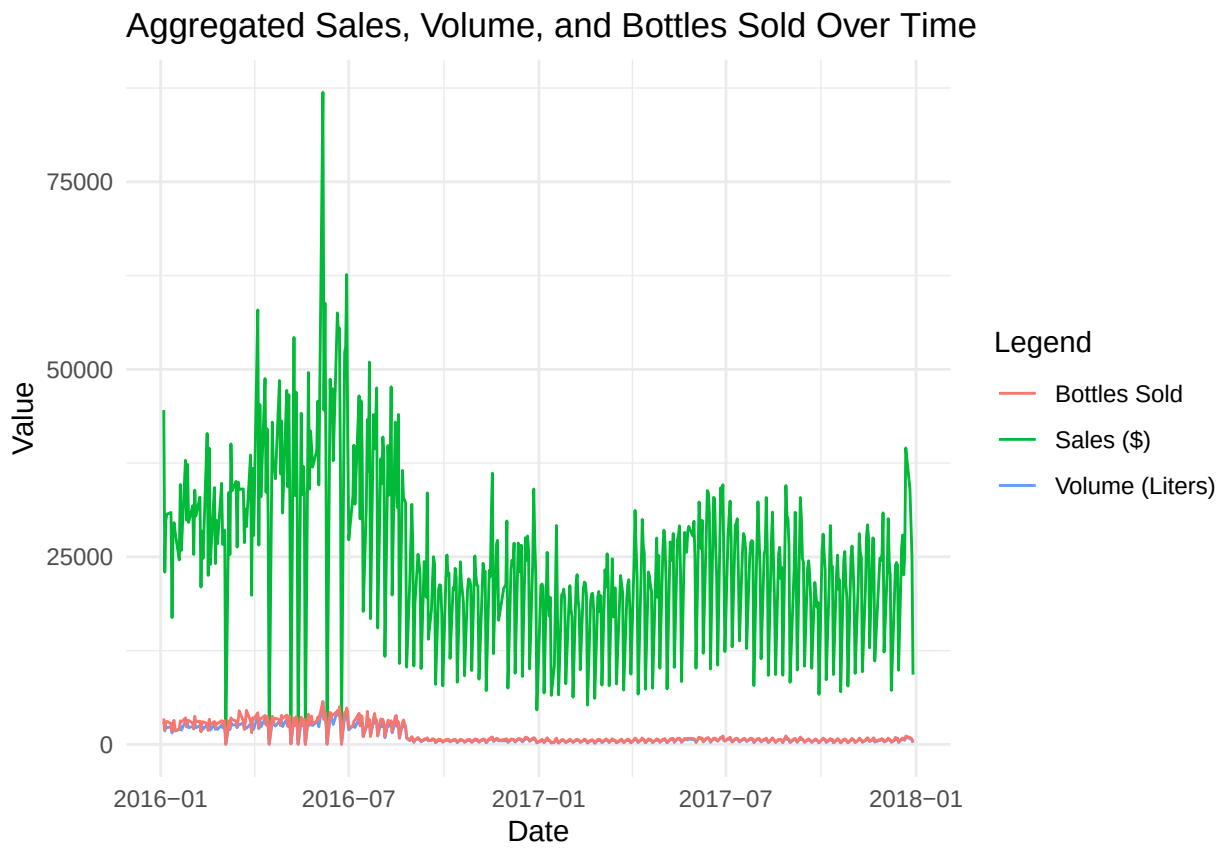
Creating an aggregated table and plotting

```
# Convert the date column to Date format
merged_data$date <- as.Date(merged_data$date)

# Group the data by 'date and summarize to calculate the sum of each variable for each date
aggregated_data <- merged_data %>%
  group_by(date) %>%
```

```r
  summarize(total_sale_dollars = sum(sale.dollars),
            total_sale_volume = sum(sale.volume),
            total_sale_bottles = sum(sale.bottles))

# Print the aggregated data frame
print(aggregated_data)
```

```
## # A tibble: 492 x 4
##    date       total_sale_dollars total_sale_volume total_sale_bottles
##    <date>                  <dbl>             <dbl>               <int>
##  1 2016-01-04             44565.             3035.                3446
##  2 2016-01-05             22968.             1797.                1923
##  3 2016-01-06             29866.             2385.                2877
##  4 2016-01-07             30697.             2215.                3072
##  5 2016-01-11             30916.             2359.                2772
##  6 2016-01-12             16911.             1520.                1730
##  7 2016-01-13             28690.             2193.                2869
##  8 2016-01-14             29557.             2105.                3202
##  9 2016-01-15             28258.             1782.                1841
## 10 2016-01-19             24588.             1999.                2008
## # i 482 more rows
```

```r
# Plotting
ggplot(aggregated_data, aes(x = date)) +
  geom_line(aes(y = total_sale_dollars, color = "Sales ($)")) +
  geom_line(aes(y = total_sale_volume, color = "Volume (Liters)")) +
  geom_line(aes(y = total_sale_bottles, color = "Bottles Sold")) +
  labs(x = "Date", y = "Value", color = "Legend") +
  ggtitle("Aggregated Sales, Volume, and Bottles Sold Over Time") +
  theme_minimal()
```




```
# Extract Year from the date column
merged_data$year <- (lubridate::year(merged_data$date))

# Creating an aggregated dataset based on year and quarter
aggregated_data <- merged_data %>%
  group_by(year, quarter) %>%
  summarize(total_sale_dollars = sum(sale.dollars),
            total_sale_volume = sum(sale.volume),
            total_sale_bottles = sum(sale.bottles)) %>%
  mutate(year_quarter = paste0(as.character(year), quarter))
```

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

```
# Print the updated aggregated data frame
print(aggregated_data)
```

```
## # A tibble: 8 x 6
## # Groups:   year [2]
##    year quarter total_sale_dollars total_sale_volume total_sale_bottles
##   <dbl> <chr>                <dbl>             <dbl>              <int>
## 1  2016 Q1                1578933.          120252.             149218
## 2  2016 Q2                2246735.          159584.             186824
## 3  2016 Q3                1845867.          107837.             125716
## 4  2016 Q4                1329442.           33930.              39604
```

```
## 5  2017 Q1               1130376.            28310.              33653
## 6  2017 Q2               1521828.            37704.              42584
## 7  2017 Q3               1474235.            35618.              40683
## 8  2017 Q4               1378807.            34235.              39167
## # i 1 more variable: year_quarter <chr>
```

Creating time series objects

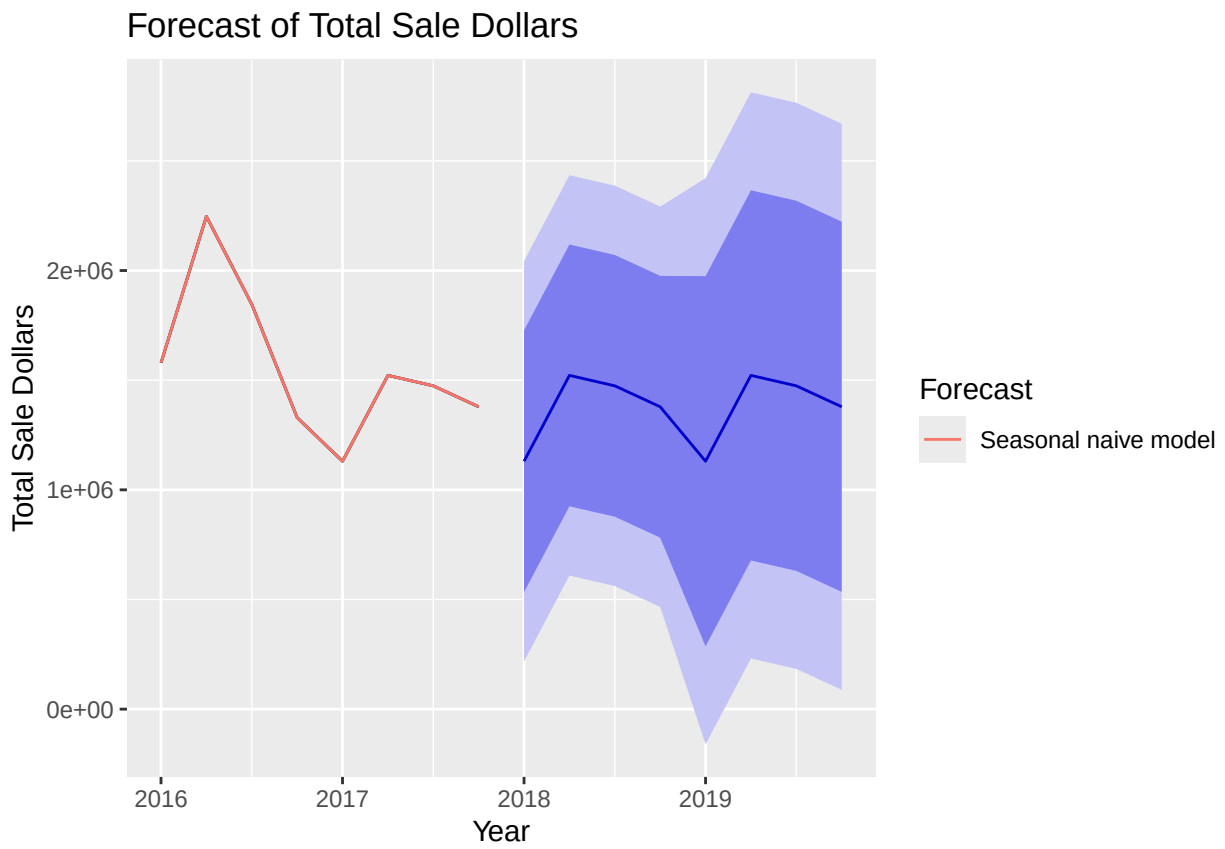
```
# Create a date column using the year and quarter components
aggregated_data$date <- as.Date(paste0(aggregated_data$year, "-", aggregated_data$quarter), format = "%

# Convert to quarterly time series for sales (dollars)
ts_data_dollars <- ts(aggregated_data$total_sale_dollars, frequency = 4, start = c(min(aggregated_data$

# Convert to quarterly time series for sales (bottles)
ts_data_bottles <- ts(aggregated_data$total_sale_bottles, frequency = 4, start = c(min(aggregated_data$

# Convert to quarterly time series for sales (volume)
ts_data_volume <- ts(aggregated_data$total_sale_volume, frequency = 4, start = c(min(aggregated_data$yea
```

Forecasting with an snaive model (dollars)

```
# Fit the snaive model
snaive_model_dollars <- snaive(ts_data_dollars)

# Forecast for the next 4 quarters
forecast_result_dollars<- forecast(snaive_model_dollars, h = 8)

# Print the forecast
print(forecast_result_dollars)
```

```
##         Point Forecast    Lo 80   Hi 80      Lo 95   Hi 95
## 2018 Q1        1130376 533649.3 1727103  217761.20 2042991
## 2018 Q2        1521828 925101.4 2118555  609213.28 2434443
## 2018 Q3        1474235 877508.6 2070962  561620.46 2386850
## 2018 Q4        1378807 782080.6 1975534  466192.51 2291422
## 2019 Q1        1130376 286477.0 1974275 -160256.30 2421009
## 2019 Q2        1521828 677929.0 2365727  231195.78 2812461
## 2019 Q3        1474235 630336.2 2318135  183602.96 2764868
## 2019 Q4        1378807 534908.3 2222707   88175.01 2669440
```

```
autoplot(forecast_result_dollars) +
  autolayer(ts_data_dollars, series = "Seasonal naive model") +
  xlab("Year") +
  ylab("Total Sale Dollars") +
  ggtitle("Forecast of Total Sale Dollars") +
  guides(colour = guide_legend(title = "Forecast"))
```



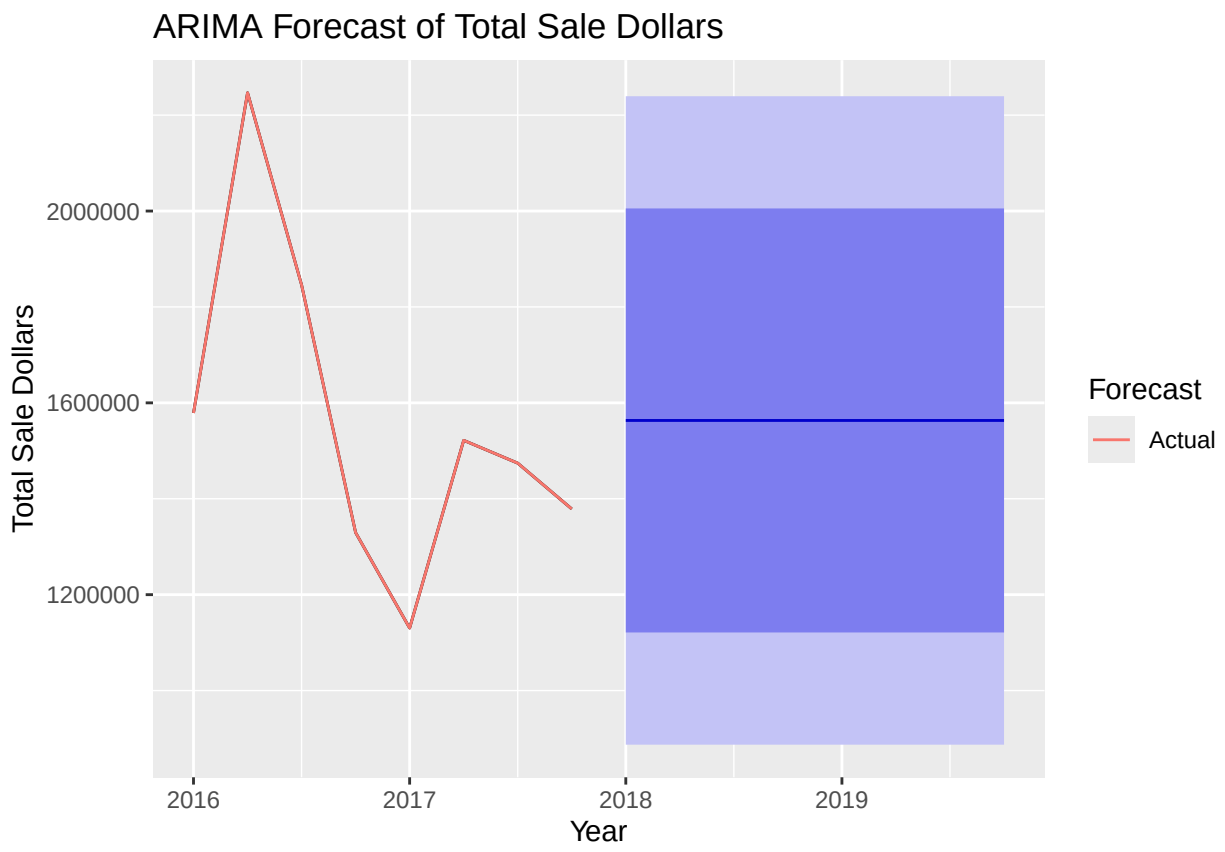
#ARIMA forecast but I am not sure why this is not working yet since I am getting the same point forecast for every upcoming quarter

```
# Fit the ARIMA model
arima_model <- auto.arima(ts_data_dollars)

# Forecast for the next 8 quarters
forecast_result_arima <- forecast(arima_model, h = 8)

# Print the ARIMA forecast
print(forecast_result_arima)
```

```
##          Point Forecast   Lo 80   Hi 80    Lo 95   Hi 95
## 2018 Q1        1563278 1121196 2005360 887172.2 2239384
## 2018 Q2        1563278 1121196 2005360 887172.2 2239384
## 2018 Q3        1563278 1121196 2005360 887172.2 2239384
## 2018 Q4        1563278 1121196 2005360 887172.2 2239384
## 2019 Q1        1563278 1121196 2005360 887172.2 2239384
## 2019 Q2        1563278 1121196 2005360 887172.2 2239384
## 2019 Q3        1563278 1121196 2005360 887172.2 2239384
## 2019 Q4        1563278 1121196 2005360 887172.2 2239384
```

```
# Plot the ARIMA forecast
autoplot(forecast_result_arima) +
  autolayer(ts_data_dollars, series = "Actual") +
  xlab("Year") +
```

```
  ylab("Total Sale Dollars") +
  ggtitle("ARIMA Forecast of Total Sale Dollars") +
  guides(colour = guide_legend(title = "Forecast"))
```

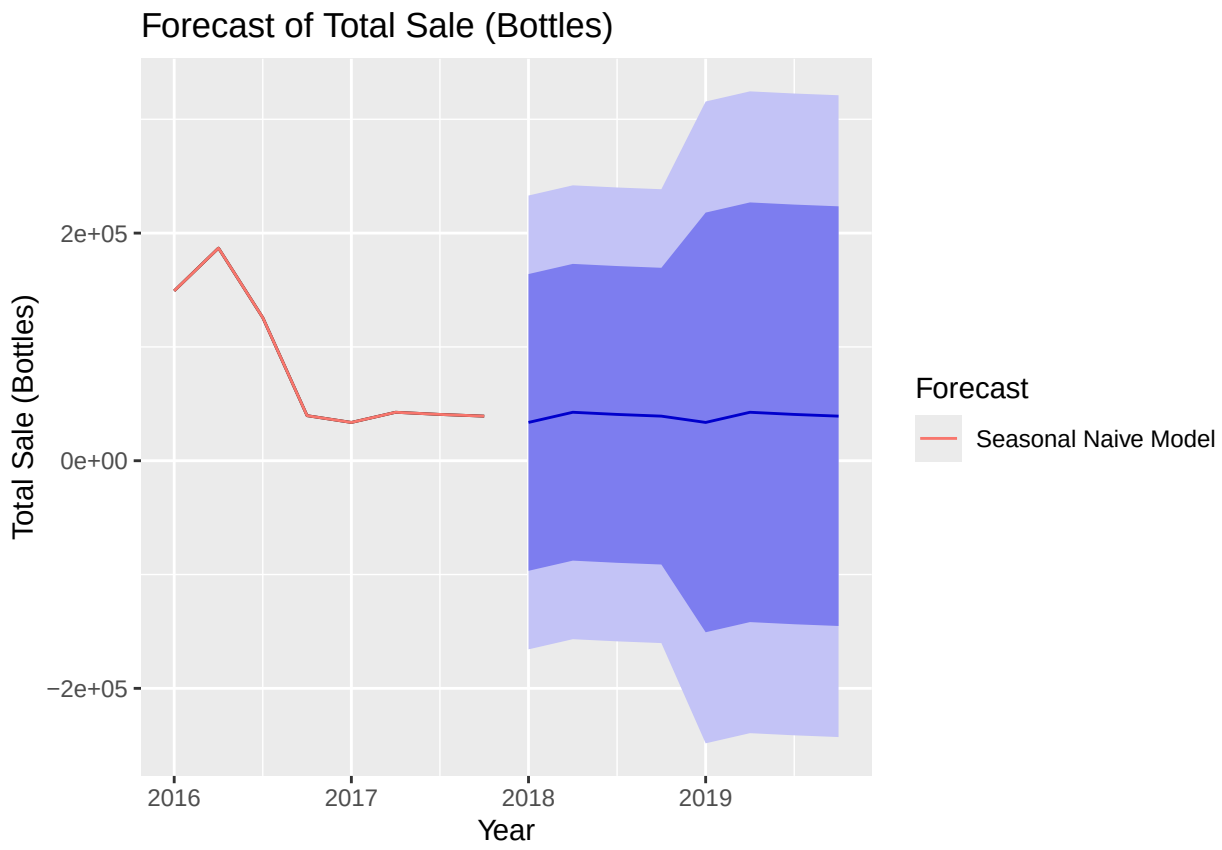


Forecasting with an snaive model (bottles)

```
# Fit the snaive model
snaive_model_bottles <- snaive(ts_data_bottles)

# Forecast for the next 4 quarters
forecast_result_bottles <- forecast(snaive_model_bottles, h = 8)

# Print the forecast
print(forecast_result_bottles)
```

```
##         Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 2018 Q1          33653  -96711.8 164017.8 -165722.8 233028.8
## 2018 Q2          42584  -87780.8 172948.8 -156791.8 241959.8
## 2018 Q3          40683  -89681.8 171047.8 -158692.8 240058.8
## 2018 Q4          39167  -91197.8 169531.8 -160208.8 238542.8
## 2019 Q1          33653 -150710.7 218016.7 -248306.9 315612.9
## 2019 Q2          42584 -141779.7 226947.7 -239375.9 324543.9
## 2019 Q3          40683 -143680.7 225046.7 -241276.9 322642.9
## 2019 Q4          39167 -145196.7 223530.7 -242792.9 321126.9
```

```r
autoplot(forecast_result_bottles) +
  autolayer(ts_data_bottles, series = "Seasonal Naive Model") +
  xlab("Year") +
  ylab("Total Sale (Bottles)") +
  ggtitle("Forecast of Total Sale (Bottles)") +
  guides(colour = guide_legend(title = "Forecast"))
```



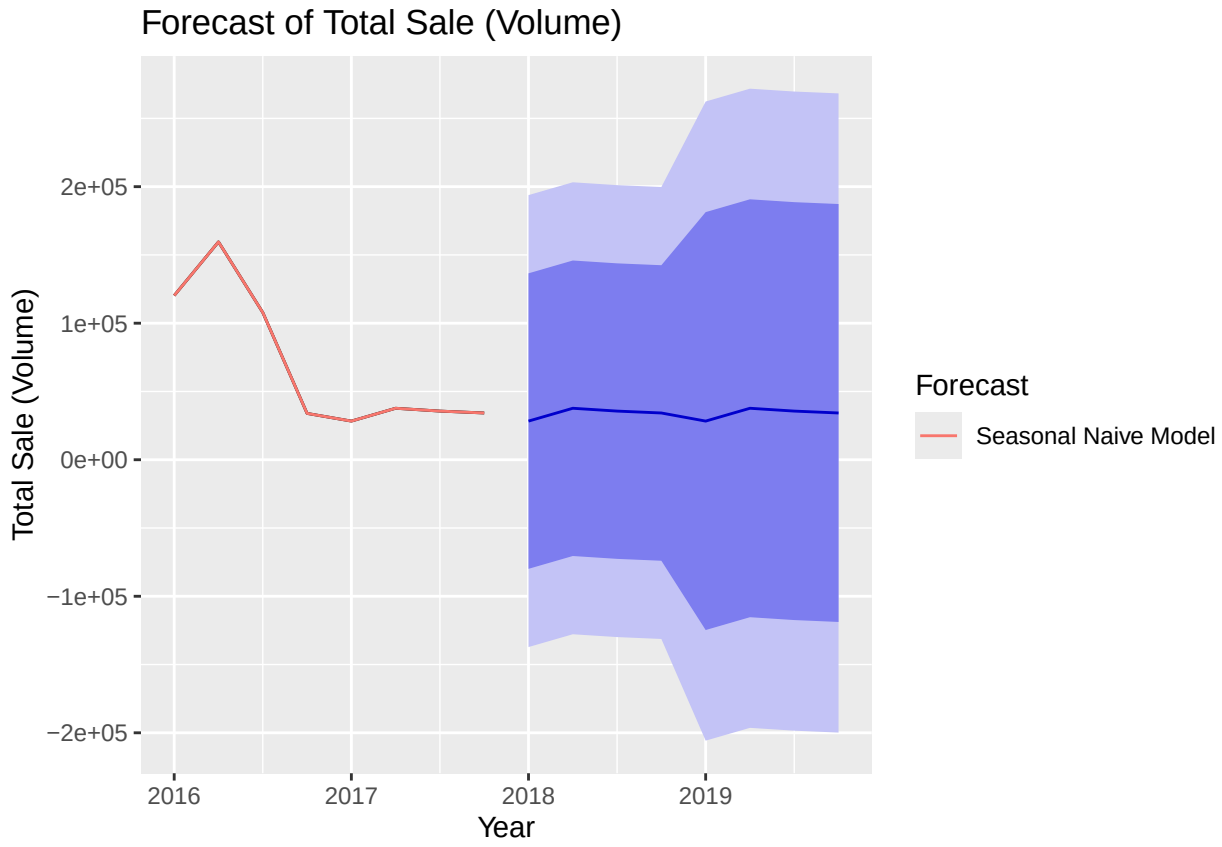
Forecasting with an snaive (volume)

```r
# Fit the snaive model
snaive_model_volume <- snaive(ts_data_volume)

# Forecast for the next 4 quarters
forecast_result_volume <- forecast(snaive_model_volume, h = 8)

# Print the forecast
print(forecast_result_volume)
```

```
##          Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 2018 Q1       28309.83  -79910.96 136530.6 -137199.6 193819.2
## 2018 Q2       37703.65  -70517.14 145924.4 -127805.8 203213.1
## 2018 Q3       35617.73  -72603.06 143838.5 -129891.7 201127.1
## 2018 Q4       34234.52  -73986.27 142455.3 -131274.9 199743.9
## 2019 Q1       28309.83 -124737.47 181357.1 -205755.8 262375.5
## 2019 Q2       37703.65 -115343.65 190751.0 -196362.0 271769.3
```

```
## 2019 Q3       35617.73 -117429.57 188665.0 -198447.9 269683.4
## 2019 Q4       34234.52 -118812.78 187281.8 -199831.1 268300.2
```

```r
autoplot(forecast_result_volume) +
  autolayer(ts_data_volume, series = "Seasonal Naive Model") +
  xlab("Year") +
  ylab("Total Sale (Volume)") +
  ggtitle("Forecast of Total Sale (Volume)") +
  guides(colour = guide_legend(title = "Forecast"))
```



Create an aggregated table by county

```r
# Extract Year from the date column
merged_data$year <- lubridate::year(merged_data$date)

# Creating an aggregated dataset based on year, quarter, and county
aggregated_data_county <- merged_data %>%
  group_by(year, quarter, county) %>%
  summarize(total_sale_dollars = sum(sale.dollars),
            total_sale_volume = sum(sale.volume),
            total_sale_bottles = sum(sale.bottles)) %>%
  mutate(year_quarter = paste0(year, quarter))
```

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

```r
# Print the updated aggregated data frame
print(aggregated_data_county)
```

```
## # A tibble: 788 x 7
## # Groups:   year, quarter [8]
##     year quarter county  total_sale_dollars total_sale_volume total_sale_bottles
##    <dbl> <chr>   <chr>                <dbl>             <dbl>              <int>
##  1  2016 Q1      Adair                 696.              51.8                 65
##  2  2016 Q1      Adams                 439.              38.5                 35
##  3  2016 Q1      Allama~              2517.             202.                  218
##  4  2016 Q1      Appano~              2261.             209                   187
##  5  2016 Q1      Audubon               354.              43.9                 36
##  6  2016 Q1      Benton               1990.             202.                  174
##  7  2016 Q1      Black ~            128633.           10324.                21646
##  8  2016 Q1      Boone                 5882.             473.                 492
##  9  2016 Q1      Bremer                7158.             627.                 532
## 10  2016 Q1      Buchan~               3015.             273.                 240
## # i 778 more rows
## # i 1 more variable: year_quarter <chr>
```

i.e. Forecasating for a specific county

```r
# Filter the aggregated data to include only rows where the county is "Adair"
aggregated_data_adair <- filter(aggregated_data_county, county == "Adair")

# Create a date column using the year and quarter components
aggregated_data_adair$date <- as.Date(paste0(aggregated_data_adair$year, "-", aggregated_data_adair$qua

# Convert to quarterly time series for sales (dollars)
ts_data_dollars_adair <- ts(aggregated_data_adair$total_sale_dollars, frequency = 4, start = c(min(aggr

# Convert to quarterly time series for sales (bottles)
ts_data_bottles_adair <- ts(aggregated_data_adair$total_sale_bottles, frequency = 4, start = c(min(aggr

# Convert to quarterly time series for sales (volume)
ts_data_volume_adair <- ts(aggregated_data_adair$total_sale_volume, frequency = 4, start = c(min(aggrega
```

Forecasating for dollar sales for adair

```r
forecast_dollars_adair <- snaive(ts_data_dollars_adair)

# Print the forecasted values
print(forecast_dollars_adair)
```

```
##         Point Forecast     Lo 80     Hi 80     Lo 95     Hi 95
## 2018 Q1         117.15 -397.4896  631.7896 -669.9233  904.2233
## 2018 Q2         281.10 -233.5396  795.7396 -505.9733 1068.1733
## 2018 Q3         200.58 -314.0596  715.2196 -586.4933  987.6533
## 2018 Q4         208.28 -306.3596  722.9196 -578.7933  995.3533
## 2019 Q1         117.15 -610.6603  844.9603 -995.9397 1230.2397
## 2019 Q2         281.10 -446.7103 1008.9103 -831.9897 1394.1897
## 2019 Q3         200.58 -527.2303  928.3903 -912.5097 1313.6697
## 2019 Q4         208.28 -519.5303  936.0903 -904.8097 1321.3697
```

```r
# Plot the forecasted values
autoplot(forecast_dollars_adair) +
  ggtitle("Forecasted Dollar Sales for County Adair (Seasonal Naive Method)") +
  xlab("Year") +
  ylab("Dollar Sales")
```