Milestone 1

Yesh Onipede

2024-04-17

Setting up data paths

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

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)
```

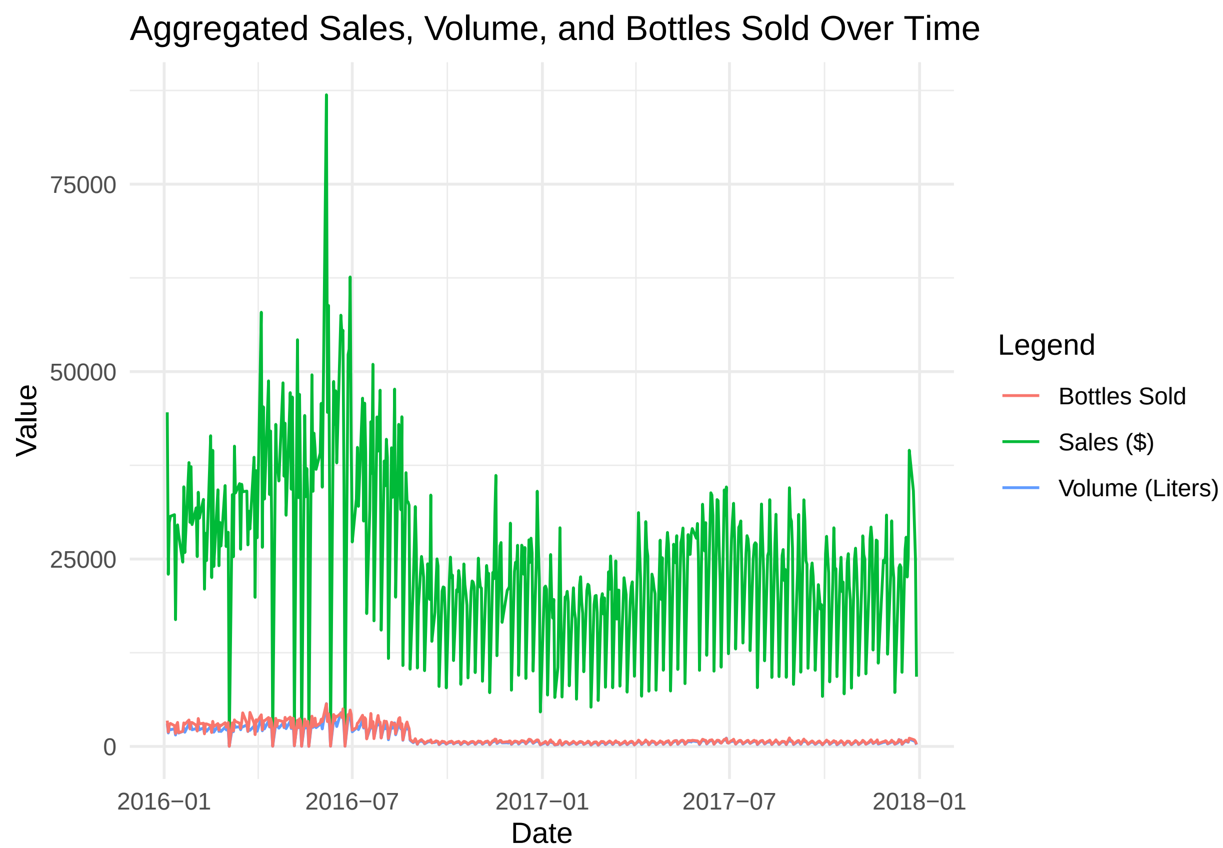
Creating an aggregated table and plotting

```
# 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, month) %>%
  summarize(total_sale_dollars = sum(sale.dollars),
            total_sale_volume = sum(sale.volume),
            total_sale_bottles = sum(sale.bottles))
```

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

```
  #mutate(year_quarter = paste0(as.character(year), quarter))

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

```
## # A tibble: 24 x 5
## # Groups:   year [2]
##     year month total_sale_dollars total_sale_volume total_sale_bottles
##    <dbl> <chr>              <dbl>             <dbl>              <int>
##  1  2016 Apr              628990.            44507.              53304
##  2  2016 Aug              730100.            48754.              56523
##  3  2016 Dec              485263.            12387.              14577
```

```
## 4 2016 Feb 505842. 39008. 47078
## 5 2016 Jan 482181. 36250. 44069
## 6 2016 Jul 679458. 47153. 55461
## 7 2016 Jun 896498. 64111. 73628
## 8 2016 Mar 590910. 44994. 58071
## 9 2016 May 721247. 50966. 59892
## 10 2016 Nov 429182. 11196. 13093
## # i 14 more rows
```

Creating time series objects

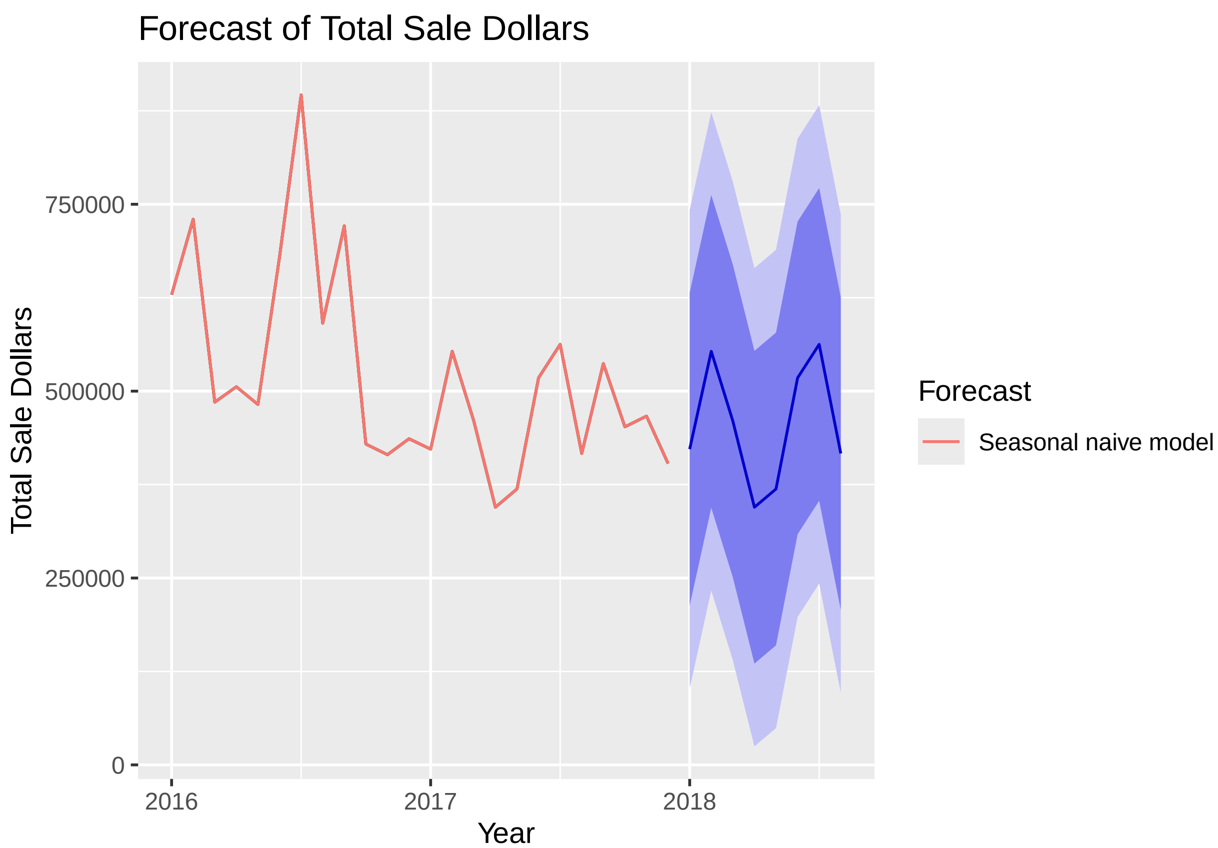
```r
# Create a date column using the year and quarter components
aggregated_data$date <- as.Date(paste0(aggregated_data$year, "-", aggregated_data$month), format = "%Y-%

# Convert to quarterly time series for sales (dollars)
ts_data_dollars <- ts(aggregated_data$total_sale_dollars, frequency = 12, start = c(min(aggregated_data$

# Convert to quarterly time series for sales (bottles)
ts_data_bottles <- ts(aggregated_data$total_sale_bottles, frequency = 12, start = c(min(aggregated_data$

# Convert to quarterly time series for sales (volume)
ts_data_volume <- ts(aggregated_data$total_sale_volume, frequency = 12, start = c(min(aggregated_data$ye
```

Forecasting with an snaive model (dollars)

```r
# 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
## Jan 2018       422360.9 213187.4 631534.5 102457.61 742264.3
## Feb 2018       553255.0 344081.5 762428.5 233351.67 873158.3
## Mar 2018       459940.1 250766.5 669113.6 140036.72 779843.4
## Apr 2018       344671.2 135497.6 553844.7  24767.84 664574.5
## May 2018       369111.9 159938.4 578285.5  49208.58 689015.2
## Jun 2018       517776.5 308603.0 726950.1 197873.21 837679.9
## Jul 2018       562564.6 353391.1 771738.2 242661.31 882468.0
## Aug 2018       416593.1 207419.6 625766.6  96689.77 736496.4
```

```r
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"))
```



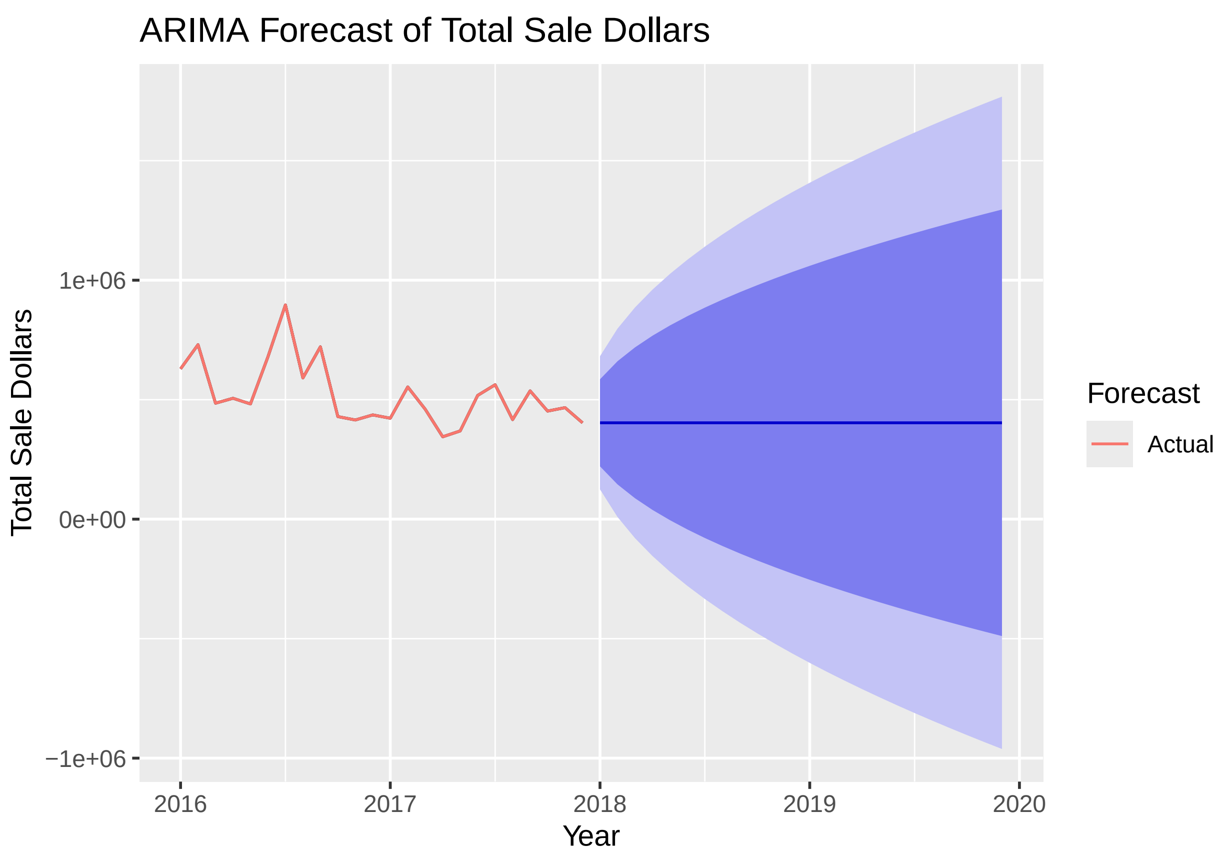
#ARIMA forecast but I am not sure why this is not working yet since I am getting the same point forecast for every upcoming quarter

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

# Forecast for the next 8 quarters
forecast_result_arima <- forecast(arima_model, h = 24)

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

```
##          Point Forecast       Lo 80     Hi 80      Lo 95     Hi 95
## Jan 2018       403203.9  221040.660  585367.1  124609.27  681798.5
## Feb 2018       403203.9  145586.175  660821.6    9211.59  797196.2
## Mar 2018       403203.9   87687.913  718719.9  -79336.16  885744.0
## Apr 2018       403203.9   38877.420  767530.4 -153985.37  960393.2
## May 2018       403203.9   -4125.488  810533.3 -219752.64 1026160.4
## Jun 2018       403203.9  -43003.088  849410.9 -279210.80 1085618.6
## Jul 2018       403203.9  -78754.731  885162.5 -333888.22 1140296.0
## Aug 2018       403203.9 -112031.550  918439.3 -384780.72 1191188.5
## Sep 2018       403203.9 -143285.820  949693.6 -432580.00 1238987.8
## Oct 2018       403203.9 -172846.845  979254.6 -477789.69 1284197.5
## Nov 2018       403203.9 -200963.218 1007371.0 -520789.97 1327197.8
## Dec 2018       403203.9 -227828.074 1034235.9 -561876.22 1368284.0
## Jan 2019       403203.9 -253595.003 1060002.8 -601283.34 1407691.1
## Feb 2019       403203.9 -278388.533 1084796.3 -639201.77 1445609.6
```

```
## Mar 2019       403203.9 -302311.295 1108719.1 -675788.48 1482196.3
## Apr 2019       403203.9 -325449.061 1131856.9 -711174.64 1517582.4
## May 2019       403203.9 -347874.380 1154282.2 -745471.20 1551879.0
## Jun 2019       403203.9 -369649.274 1176057.1 -778773.03 1585180.8
## Jul 2019       403203.9 -390827.255 1197235.1 -811161.96 1617569.8
## Aug 2019       403203.9 -411454.876 1217862.7 -842709.18 1649117.0
## Sep 2019       403203.9 -431572.937 1237980.7 -873477.10 1679884.9
## Oct 2019       403203.9 -451217.432 1257625.2 -903520.76 1709928.6
## Nov 2019       403203.9 -470420.309 1276828.1 -932889.03 1739296.8
## Dec 2019       403203.9 -489210.076 1295617.9 -961625.50 1768033.3
```

```
# 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"))
```

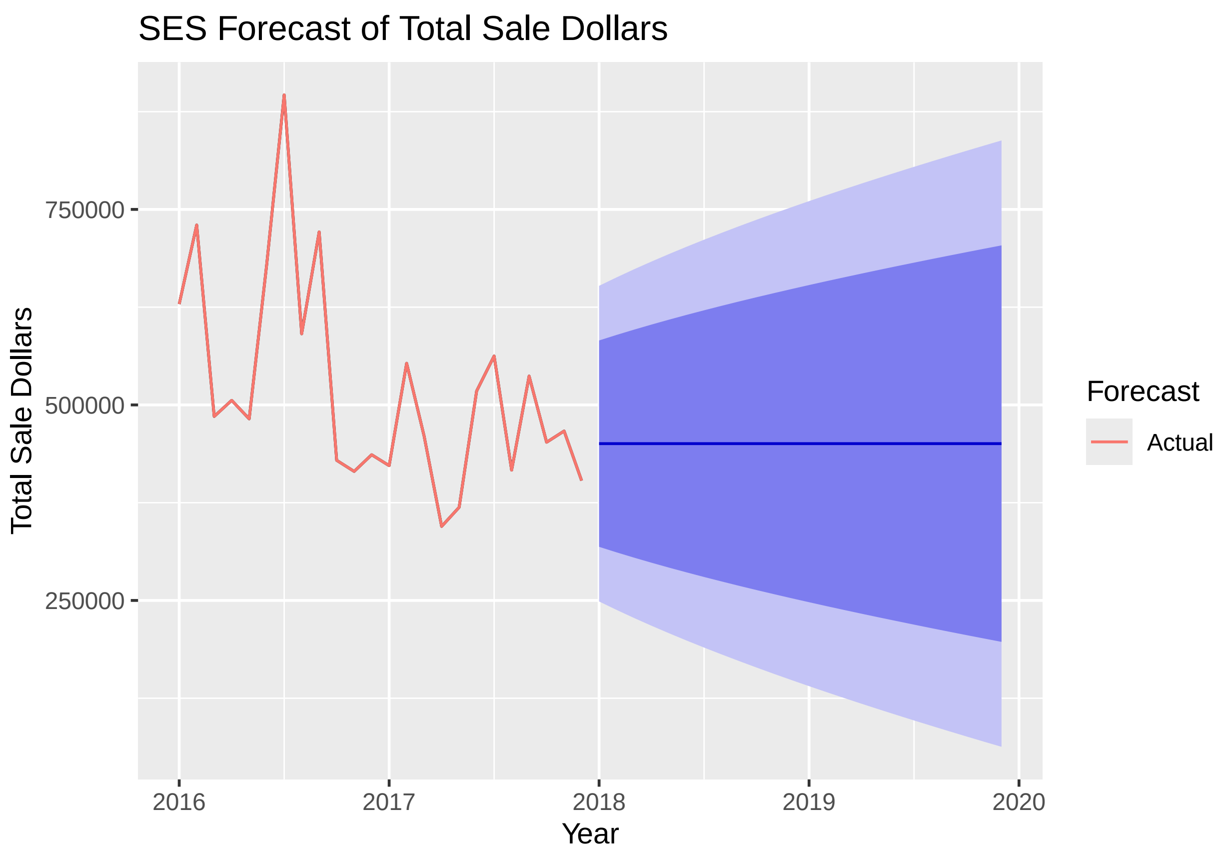



```
# Fit SES model
ses_model <- ets(ts_data_dollars)

# Forecast for the next 8 quarters
forecast_result_ses <- forecast(ses_model, h = 24)
```

```r
# Print the SES forecast
print(forecast_result_ses)
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2018       450492.3 318554.9 582429.6 248711.45 652273.1
## Feb 2018       450492.3 311480.9 589503.6 237892.75 663091.8
## Mar 2018       450492.3 304714.0 596270.5 227543.59 673440.9
## Apr 2018       450492.3 298212.9 602771.6 217601.09 683383.4
## May 2018       450492.3 291944.9 609039.7 208014.92 692969.6
## Jun 2018       450492.3 285883.0 615101.5 198744.13 702240.4
## Jul 2018       450492.3 280005.2 620979.3 189754.80 711229.7
## Aug 2018       450492.3 274292.8 626691.7 181018.51 719966.0
## Sep 2018       450492.3 268730.2 632254.3 172511.15 728473.4
## Oct 2018       450492.3 263303.7 637680.8 164212.07 736772.4
## Nov 2018       450492.3 258001.7 642982.8 156103.42 744881.1
## Dec 2018       450492.3 252814.1 648170.4 148169.65 752814.9
## Jan 2019       450492.3 247731.9 653252.6 140397.11 760587.4
## Feb 2019       450492.3 242747.3 658237.2 132773.74 768210.8
## Mar 2019       450492.3 237853.2 663131.3 125288.83 775695.7
## Apr 2019       450492.3 233043.3 667941.2 117932.81 783051.7
## May 2019       450492.3 228312.2 672672.4 110697.11 790287.4
## Jun 2019       450492.3 223654.6 677329.9 103573.99 797410.5
## Jul 2019       450492.3 219066.1 681918.4  96556.45 804428.1
## Aug 2019       450492.3 214542.4 686442.1  89638.11 811346.4
## Sep 2019       450492.3 210079.8 690904.7  82813.20 818171.3
## Oct 2019       450492.3 205674.9 695309.6  76076.39 824908.1
## Nov 2019       450492.3 201324.3 699660.2  69422.82 831561.7
## Dec 2019       450492.3 197025.3 703959.2  62848.01 838136.5
```

```r
# Plot the SES forecast
autoplot(forecast_result_ses) +
  autolayer(ts_data_dollars, series = "Actual") +
  xlab("Year") +
  ylab("Total Sale Dollars") +
  ggtitle("SES Forecast of Total Sale Dollars") +
  guides(colour = guide_legend(title = "Forecast"))
```



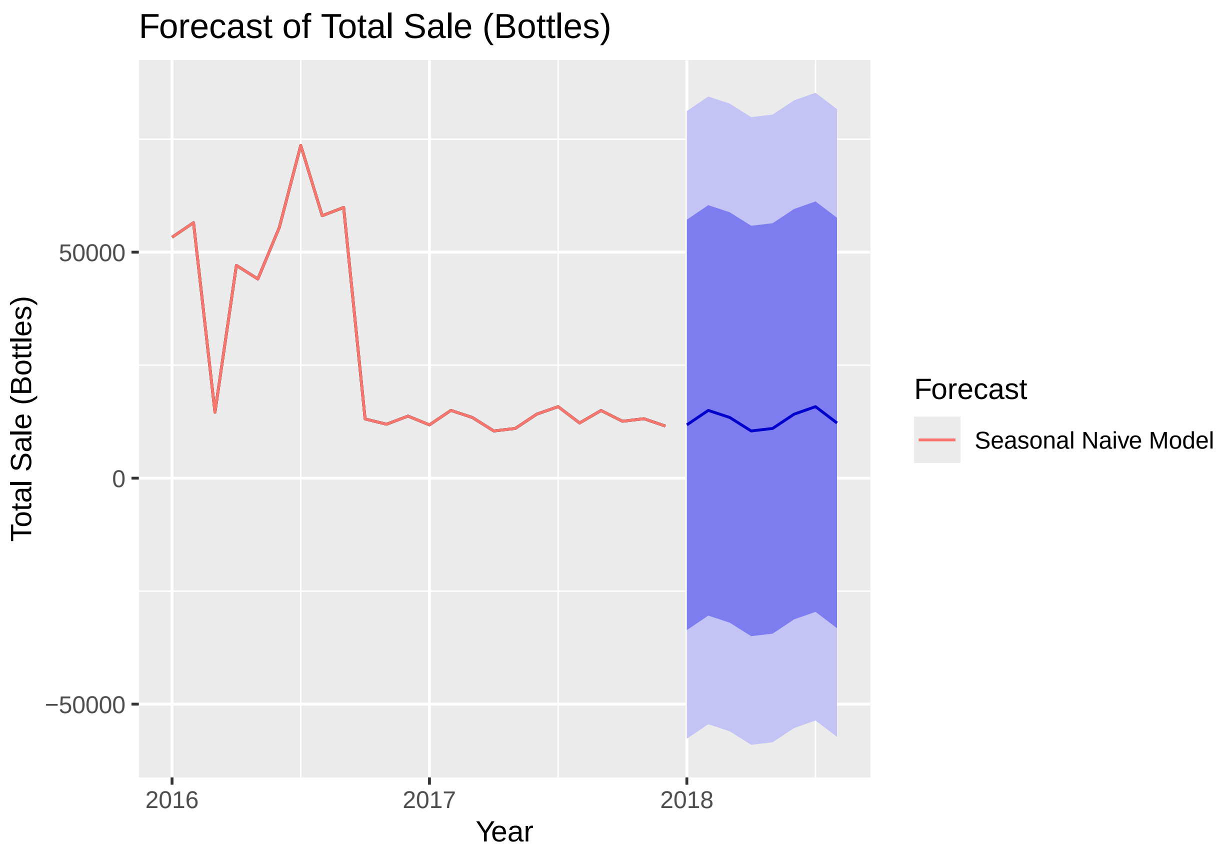
Forecasting with an snaive model (bottles)

```
# 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
## Jan 2018          11792 -33608.34 57192.34 -57641.82 81225.82
## Feb 2018          14997 -30403.34 60397.34 -54436.82 84430.82
## Mar 2018          13427 -31973.34 58827.34 -56006.82 82860.82
## Apr 2018          10441 -34959.34 55841.34 -58992.82 79874.82
## May 2018          11002 -34398.34 56402.34 -58431.82 80435.82
## Jun 2018          14157 -31243.34 59557.34 -55276.82 83590.82
## Jul 2018          15820 -29580.34 61220.34 -53613.82 85253.82
## Aug 2018          12210 -33190.34 57610.34 -57223.82 81643.82
```

```
autoplot(forecast_result_bottles) +
  autolayer(ts_data_bottles, series = "Seasonal Naive Model") +
  xlab("Year") +
  ylab("Total Sale (Bottles)") +
```

```r
  ggtitle("Forecast of Total Sale (Bottles)") +
  guides(colour = guide_legend(title = "Forecast"))
```

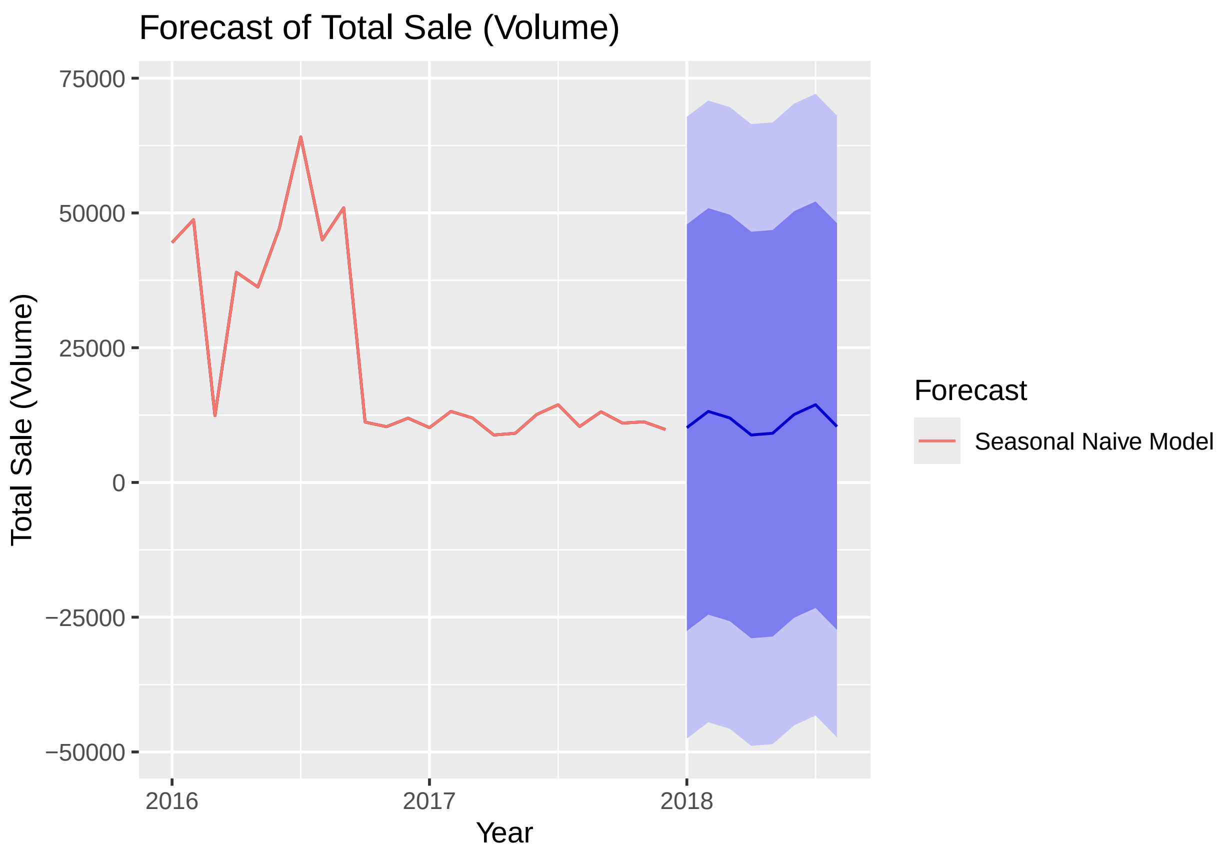


Forecasting with an snaive (volume)

```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
## Jan 2018       10165.51 -27545.90 47876.92 -47509.11 67840.13
## Feb 2018       13180.35 -24531.06 50891.76 -44494.27 70854.97
## Mar 2018       11975.66 -25735.75 49687.07 -45698.96 69650.28
## Apr 2018        8809.29 -28902.12 46520.70 -48865.33 66483.91
## May 2018        9123.97 -28587.44 46835.38 -48550.65 66798.59
## Jun 2018       12612.89 -25098.52 50324.30 -45061.73 70287.51
## Jul 2018       14424.66 -23286.75 52136.07 -43249.96 72099.28
## Aug 2018       10376.57 -27334.84 48087.98 -47298.05 68051.19
```

```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, month, 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, month))
```

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

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

```
## # A tibble: 2,343 x 7
## # Groups:   year, month [24]
##     year month county    total_sale_dollars total_sale_volume total_sale_bottles
##    <dbl> <chr> <chr>                  <dbl>             <dbl>              <int>
##  1  2016 Apr   Adair                   36.5               4                    4
##  2  2016 Apr   Adams                  121.               14.2                 15
##  3  2016 Apr   Allamakee              694.               52.2                 58
##  4  2016 Apr   Appanoose             1189.               88.5                 74
##  5  2016 Apr   Audubon                139.               16.9                 12
##  6  2016 Apr   Benton                 777.               73.1                 67
##  7  2016 Apr   Black Ha~            36713.             2878.                5447
##  8  2016 Apr   Boone                 1741.              185.                 146
##  9  2016 Apr   Bremer                4542.              319.                 260
## 10  2016 Apr   Buchanan              1617.              123                  131
## # i 2,333 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$mon

# Convert to quarterly time series for sales (dollars)
ts_data_dollars_adair <- ts(aggregated_data_adair$total_sale_dollars, frequency = 12, start = c(min(agg

# Convert to quarterly time series for sales (bottles)
ts_data_bottles_adair <- ts(aggregated_data_adair$total_sale_bottles, frequency = 12, start = c(min(agg

# Convert to quarterly time series for sales (volume)
ts_data_volume_adair <- ts(aggregated_data_adair$total_sale_volume, frequency = 12, start = c(min(aggre
```

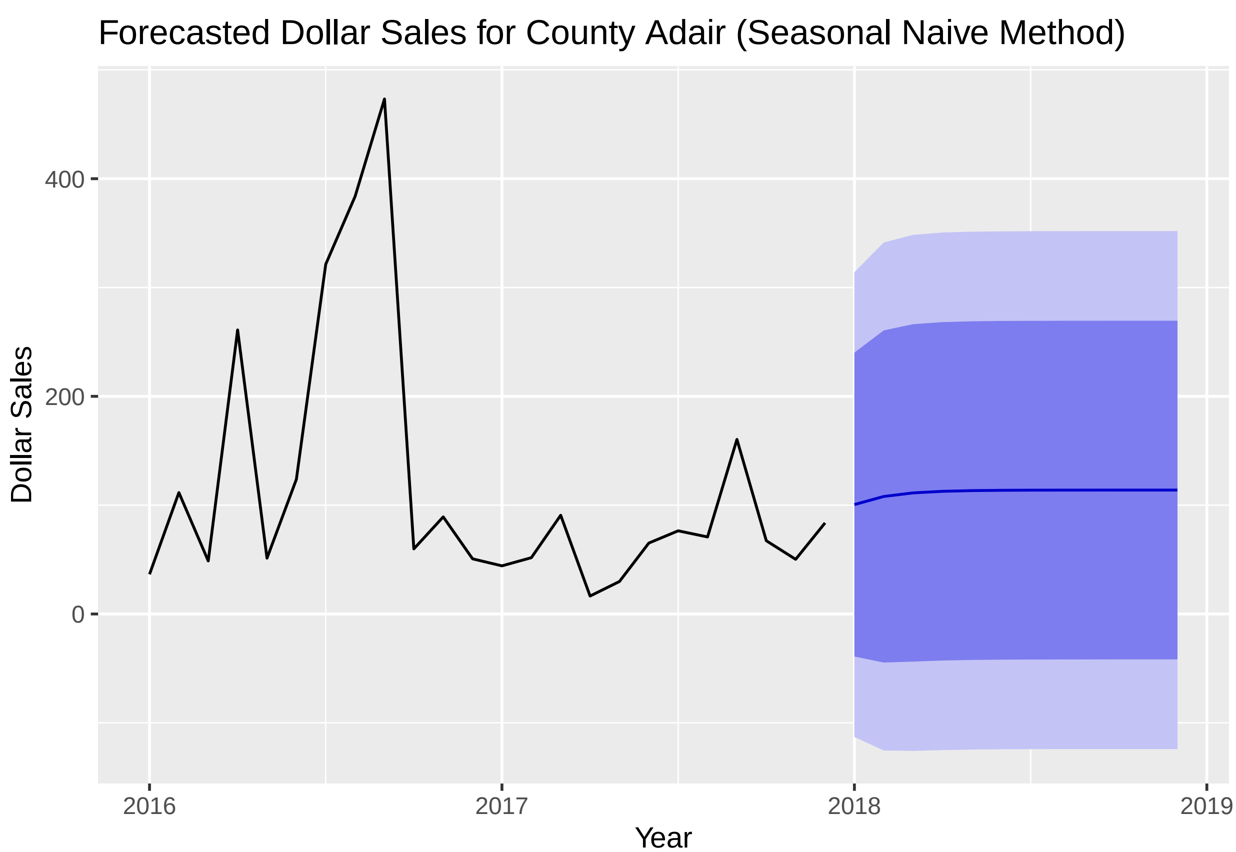
Forecasating for dollar sales for adair

```r
arima_dollars_adair <- auto.arima(ts_data_dollars_adair)

forecast_adair<-forecast(arima_dollars_adair,h=12)
# Print the forecasted values
print(forecast_adair)
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2018       100.5446 -39.07245 240.1616 -112.9812 314.0704
## Feb 2018       107.9899 -44.62766 260.6075 -125.4185 341.3983
## Mar 2018       111.2769 -43.74765 266.3013 -125.8127 348.3664
## Apr 2018       112.7280 -42.76132 268.2172 -125.0724 350.5283
## May 2018       113.3686 -42.21111 268.9483 -124.5700 351.3072
## Jun 2018       113.6514 -41.94590 269.2487 -124.3141 351.6170
```

```
## Jul 2018       113.7763 -41.82448 269.3770 -124.1945 351.7471
## Aug 2018       113.8314 -41.77002 269.4328 -124.1404 351.8032
## Sep 2018       113.8557 -41.74582 269.4573 -124.1163 351.8278
## Oct 2018       113.8665 -41.73510 269.4680 -124.1056 351.8385
## Nov 2018       113.8712 -41.73036 269.4728 -124.1009 351.8433
## Dec 2018       113.8733 -41.72827 269.4749 -124.0988 351.8454
```

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



i.e. Forecasating for a specific county

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

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

# Convert to quarterly time series for sales (dollars)
ts_data_dollars_adams<- ts(aggregated_data_adams$total_sale_dollars, frequency = 12, start = c(min(aggre

# Convert to quarterly time series for sales (bottles)
ts_data_bottles_adams <- ts(aggregated_data_adams$total_sale_bottles, frequency = 12, start = c(min(agg
```

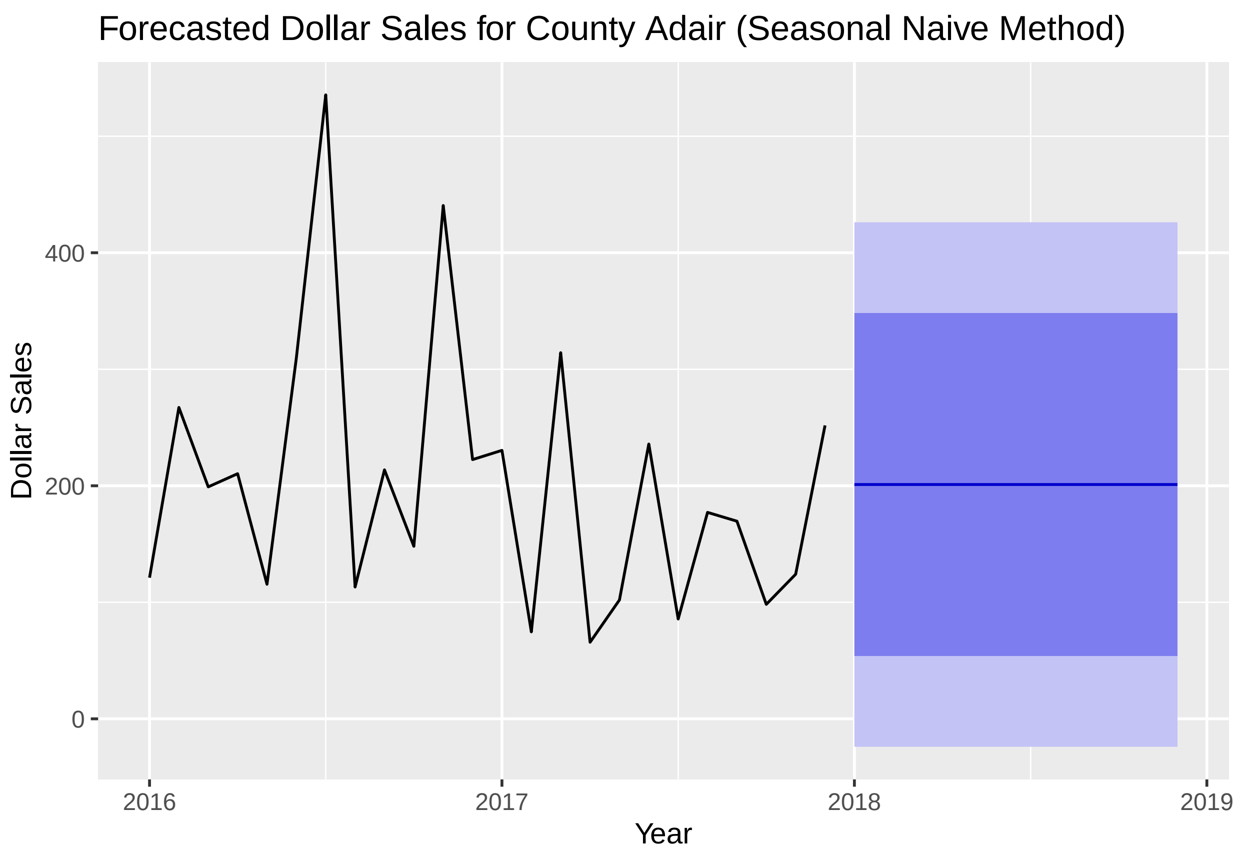
```r
# Convert to quarterly time series for sales (volume)
ts_data_volume_adams <- ts(aggregated_data_adams$total_sale_volume, frequency = 12, start = c(min(aggreg
```

Forecasating for dollar sales for adair

```r
arima_dollars_adams <- auto.arima(ts_data_dollars_adams)

forecast_adams<-forecast(arima_dollars_adams,h=12)
# Print the forecasted values
print(forecast_adams)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Feb 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Mar 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Apr 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## May 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Jun 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Jul 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Aug 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Sep 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Oct 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Nov 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
## Dec 2018       201.0808 53.90224 348.2594 -24.0094 426.1711
```

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



Looping through each county and creating a time series of sales in terms of bottles and in terms of volume

```
create_time_series <- function(data, target_county) {
  # Filter the data for the specific county
  county_data <- filter(data, county == target_county)


  # Convert to monthly time series for sales (dollars)
    ts_sales <- ts(county_data$total_sale_dollars, frequency = 12, start = c(2016,1), end = c(2017,12))

  # Convert to monthly time series for volume
    ts_volume <- ts(county_data$total_sale_volume, frequency = 12, start = c(2016,1), end = c(2017,12))

  # Return a list containing the time series objects
  return(list(
    ts_sales = ts_sales,
    ts_volume = ts_volume
  ))
}

# Get unique county names
county_names <- unique(aggregated_data_county$county)

# Create an empty list to store time series data
time_series_list <- list()
```

```r
# Loop through each county and create time series
for (county in county_names) {
  time_series_list[[county]] <- create_time_series(aggregated_data_county, county)
}
```

ARIMA model across all counties and outputting the counties that are predicted to have the largest increase over the next two years

```r
library(forecast)

# Function to fit ARIMA model and forecast liquor sales
fit_arima_forecast <- function(ts_sales) {
  # Fit ARIMA model
  arima_model <- auto.arima(ts_sales)

  # Forecast liquor sales for the next two years
  forecast_values <- forecast(arima_model, h = 24)  # 24 months = 2 years

  # Return the forecasted values
  return(forecast_values)
}

# Initialize a list to store forecasted sales for each county
forecasted_sales <- list()

# Loop through each county
for (county in county_names) {
  # Filter data for the current county
  county_data <- time_series_list[[county]]

  # Fit ARIMA model and forecast liquor sales
  forecast_values <- fit_arima_forecast(county_data$ts_sales)

  # Store the forecasted values for the county
  forecasted_sales[[county]] <- forecast_values
}

# Calculate the increase in liquor sales for each county
increase_sales <- sapply(forecasted_sales, function(forecast_values) {
  # Extract the forecasted values for the next two years
  forecast_sales <- forecast_values$mean

  # Calculate the increase in liquor sales
  increase <- forecast_sales[length(forecast_sales)] - forecast_sales[1]

  return(increase)
})

# Find the counties with the largest forecasted increase in liquor sales
top_counties <- names(increase_sales)[order(increase_sales, decreasing = TRUE)][1:5]

# Print the top counties
print(top_counties)
```

```
## [1] "Black Hawk" "Webster"    "Clinton"    "Harrison"   "Union"
```

```r
# Autoplot the forecasts for the top counties
for (county in top_counties) {
  # Plot the forecast
  autoplot(forecasted_sales[[county]], main = paste("Forecast for", county), ylab = "Sales")
}
```

```r
library(forecast)

# Function to fit seasonal naïve (snaive) model and forecast liquor sales
fit_snaive_forecast <- function(ts_sales) {
  # Fit seasonal naïve (snaive) model
  snaive_model <- snaive(ts_sales)

  # Forecast liquor sales for the next two years
  forecast_values <- forecast(snaive_model, h = 24)  # 24 months = 2 years

  # Return the forecasted values
  return(forecast_values)
}

# Initialize a list to store forecasted sales for each county using snaive model
forecasted_sales_snaive <- list()

# Loop through each county
for (county in county_names) {
  # Filter data for the current county
  county_data <- time_series_list[[county]]

  # Fit snaive model and forecast liquor sales
  forecast_values <- fit_snaive_forecast(county_data$ts_sales)

  # Store the forecasted values for the county
  forecasted_sales_snaive[[county]] <- forecast_values
}

# Calculate the increase in liquor sales for each county using snaive forecasts
increase_sales_snaive <- sapply(forecasted_sales_snaive, function(forecast_values) {
  # Extract the forecasted values for the next two years
  forecast_sales <- forecast_values$mean

  # Calculate the increase in liquor sales
  increase <- forecast_sales[length(forecast_sales)] - forecast_sales[1]

  return(increase)
})

# Find the counties with the largest forecasted increase in liquor sales using snaive forecasts
top_counties_snaive <- names(increase_sales_snaive)[order(increase_sales_snaive, decreasing = TRUE)][1:5

# Print the top counties based on snaive forecasts
print(top_counties_snaive)
```

```
## [1] "Woodbury"    "Winneshiek"  "Clay"        "Cerro Gordo" "Story"
```

```r
# Autoplot the forecasts for the top counties using snaive forecasts
for (county in top_counties_snaive) {
  # Plot the forecast
  autoplot(forecasted_sales_snaive[[county]], main = paste("Seasonal Naïve Forecast for", county), ylab
}
```