Walmart commerce analysis and ARIMA forecasting

Brian Chen

Data from https://www.kaggle.com/yasserh/walmart-dataset.

Packages to be used.

```
library(tidyverse)
library(janitor)
library(lubridate)
library(scales)

options(scipen = 999)
```

Import data and do some light cleaning.

```
walmart = read.csv("walmart.csv")
head(walmart)
```

```
##   Store       Date Weekly_Sales
## 1     1 05-02-2010      1643691
## 2     1 12-02-2010      1641957
## 3     1 19-02-2010      1611968
## 4     1 26-02-2010      1409728
## 5     1 05-03-2010      1554807
## 6     1 12-03-2010      1439542
##   Holiday_Flag Temperature
## 1            0       42.31
## 2            1       38.51
## 3            0       39.93
## 4            0       46.63
## 5            0       46.50
## 6            0       57.79
##   Fuel_Price      CPI Unemployment
## 1      2.572 211.0964        8.106
## 2      2.548 211.2422        8.106
## 3      2.514 211.2891        8.106
## 4      2.561 211.3196        8.106
## 5      2.625 211.3501        8.106
## 6      2.667 211.3806        8.106
```

```
walmart$Date = dmy(walmart$Date)
walmart = clean_names(walmart)

head(walmart)
```

```
##   store       date weekly_sales
## 1     1 2010-02-05      1643691
## 2     1 2010-02-12      1641957
## 3     1 2010-02-19      1611968
## 4     1 2010-02-26      1409728
## 5     1 2010-03-05      1554807
## 6     1 2010-03-12      1439542
##   holiday_flag temperature
## 1            0       42.31
## 2            1       38.51
## 3            0       39.93
## 4            0       46.63
## 5            0       46.50
## 6            0       57.79
##   fuel_price      cpi unemployment
## 1      2.572 211.0964        8.106
## 2      2.548 211.2422        8.106
## 3      2.514 211.2891        8.106
## 4      2.561 211.3196        8.106
## 5      2.625 211.3501        8.106
## 6      2.667 211.3806        8.106
```

How many total weeks are being examined for each store?

```
walmart %>% count(store)
```

```
##    store   n
## 1      1 143
## 2      2 143
## 3      3 143
## 4      4 143
## 5      5 143
## 6      6 143
## 7      7 143
## 8      8 143
## 9      9 143
## 10    10 143
## 11    11 143
## 12    12 143
## 13    13 143
## 14    14 143
## 15    15 143
## 16    16 143
## 17    17 143
## 18    18 143
## 19    19 143
## 20    20 143
## 21    21 143
## 22    22 143
## 23    23 143
## 24    24 143
## 25    25 143
## 26    26 143
```

```
## 27    27 143
## 28    28 143
## 29    29 143
## 30    30 143
## 31    31 143
## 32    32 143
## 33    33 143
## 34    34 143
## 35    35 143
## 36    36 143
## 37    37 143
## 38    38 143
## 39    39 143
## 40    40 143
## 41    41 143
## 42    42 143
## 43    43 143
## 44    44 143
## 45    45 143
```

Take a look at monthly sales for all years combined.

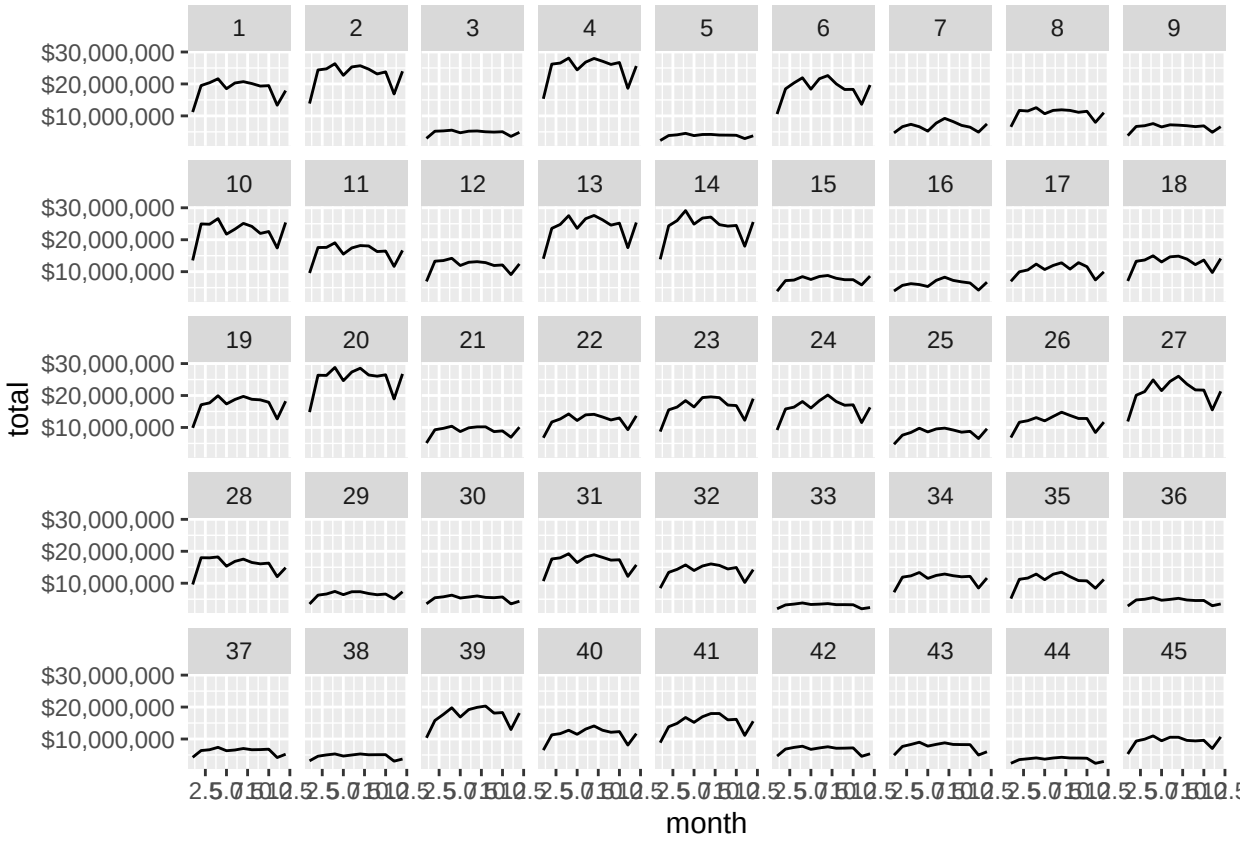
```
monthly_sales = walmart %>%
  group_by(store, month(date)) %>%
  summarize(total = sum(weekly_sales)) %>%
  rename(month = `month(date)`)
```

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

```
head(monthly_sales)
```

```
## # A tibble: 6 x 3
## # Groups:   store [1]
##   store month     total
##   <int> <dbl>     <dbl>
## 1     1     1 11203741.
## 2     1     2 19505307.
## 3     1     3 20380667.
## 4     1     4 21623140.
## 5     1     5 18505333.
## 6     1     6 20299636.
```

```
monthly_sales %>%
  ggplot(aes(x=month,y=total,group=store))+
  geom_line()+
  facet_wrap(~store, ncol=9)+
  scale_y_continuous(labels=scales::dollar_format())
```

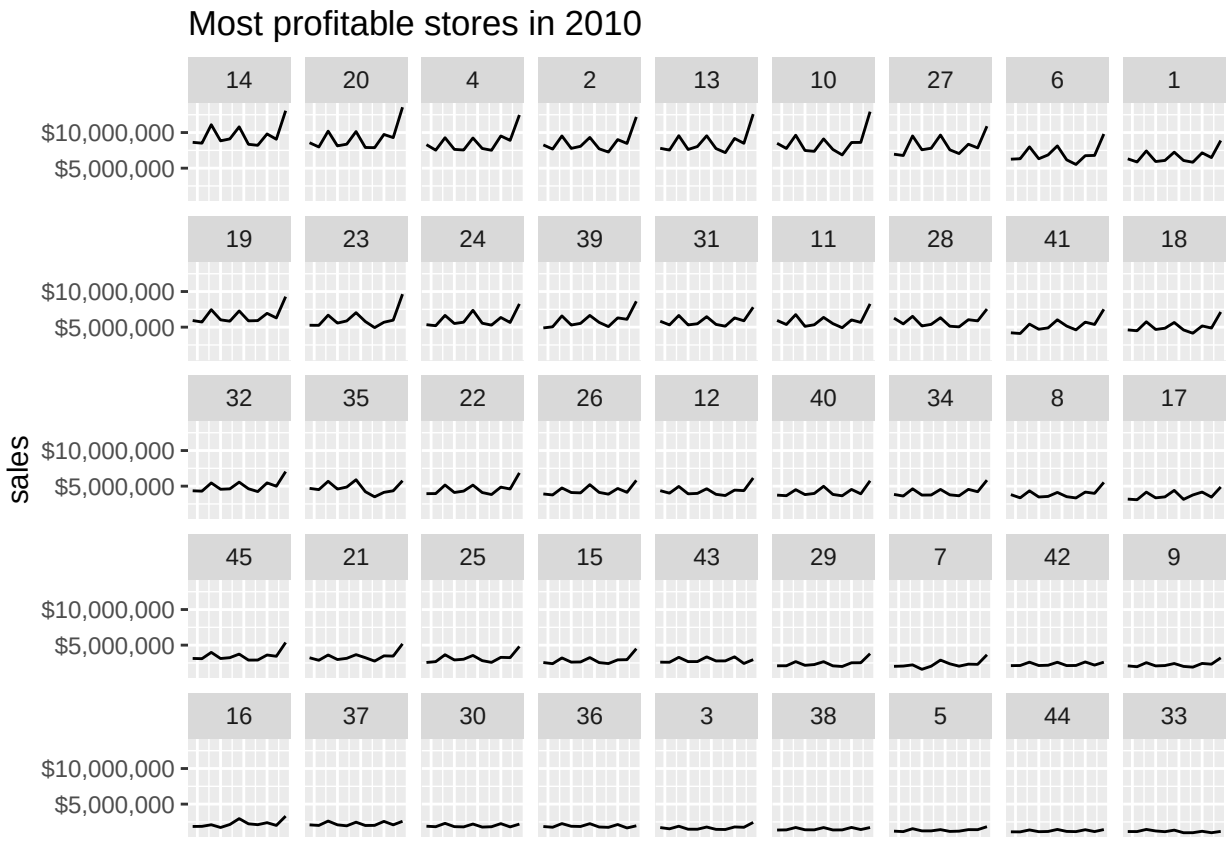


Create function to show the most/least profitable store per year.

```
years = walmart %>%
  group_by(year(date)) %>%
  group_split()

yearly_sales = function(year) {
  year %>%
  group_by(store, month(date)) %>%
  summarize(sales = sum(weekly_sales)) %>%
  ungroup() %>%
  group_by(store) %>%
  mutate(total = sum(sales)) %>%
  ungroup() %>%
  rename(month = `month(date)`) %>%
  arrange(desc(total)) %>%
  ggplot(aes(x=month, y=sales, group=store))+
    geom_line()+
    facet_wrap(~reorder(store, -total), ncol=9)+
    scale_y_continuous(labels=scales::dollar_format())+
    theme(
      axis.text.x = element_blank(),
      axis.ticks.x = element_blank(),
      axis.title.x = element_blank()
    )+
    ggtitle(paste0("Most profitable stores in ",
                   year$date[1] %>% str_extract("(201.)")))
```

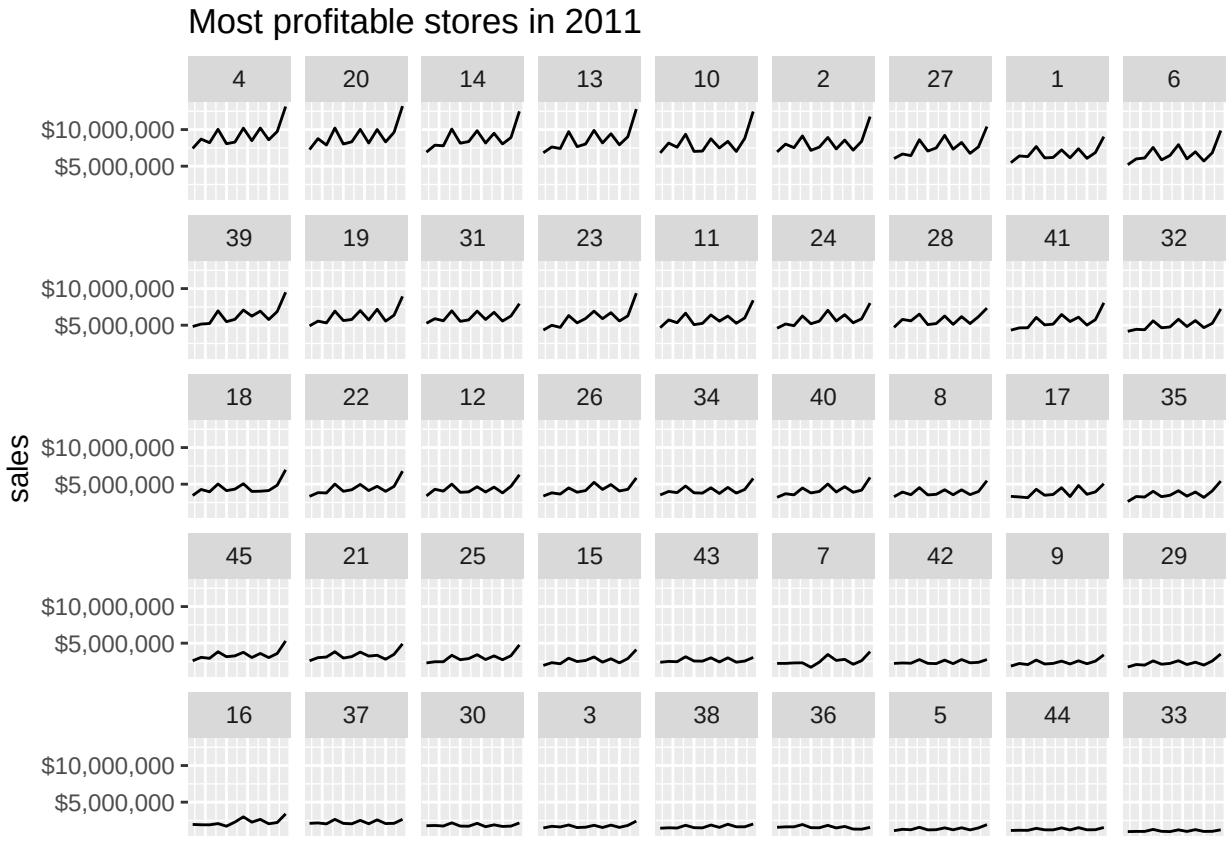
```
}

yearly_sales(years[[1]]) #Caveat: data is missing January
```

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



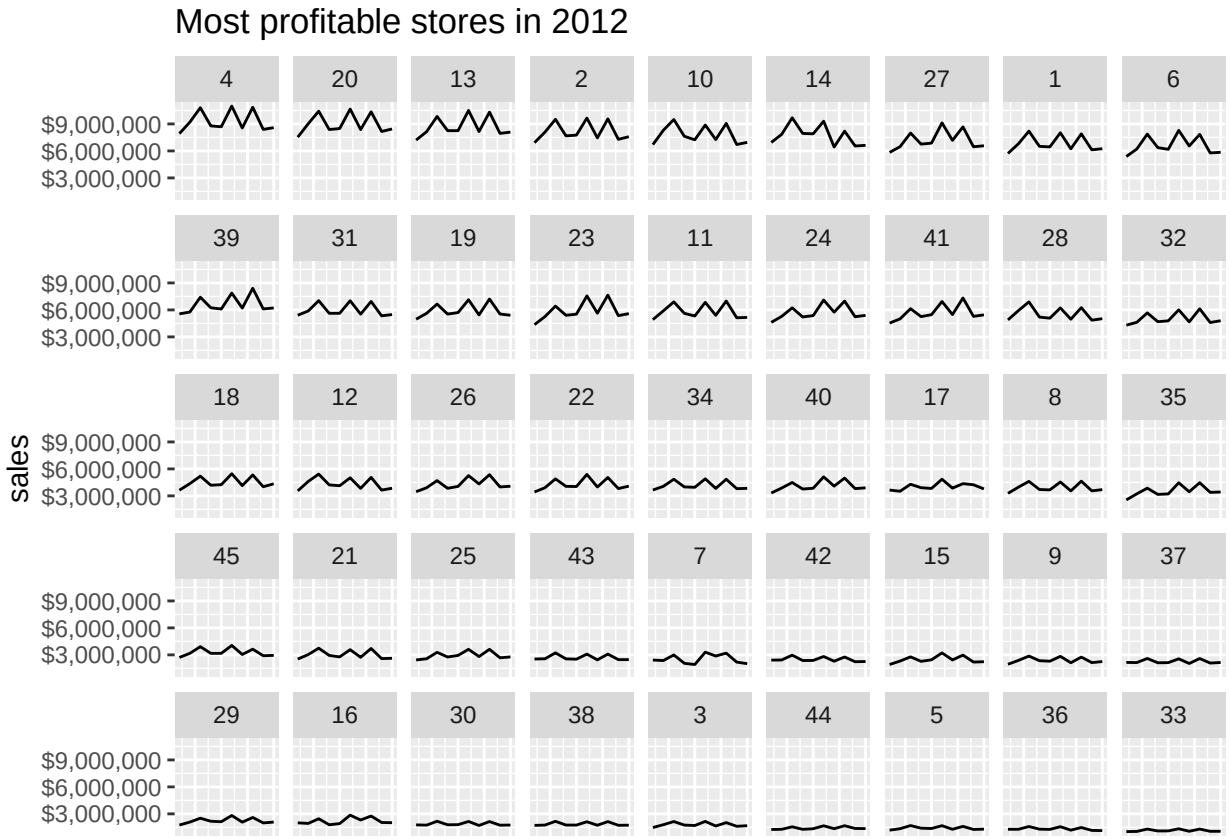

```
yearly_sales(years[[2]])
```

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




```
yearly_sales(years[[3]]) #Caveat: data is missing November and December
```

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



Checking overall performance of stores over the three years, check the ranking over each year, and check the best/worst performing stores over that period.

```
rankings = walmart %>%
  group_by(store, year(date)) %>%
  summarize(total = sum(weekly_sales)) %>%
  ungroup() %>%
  rename(year = `year(date)`) %>%
  arrange(year, desc(total)) %>%
  mutate(score = rep(1:45,3))  #can assign score so easily b/c already arranged
```

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

```
rankings %>%
  select(store, year, score) %>%
  pivot_wider(names_from = year, values_from = score)
```

```
## # A tibble: 45 x 4
##    store `2010` `2011` `2012`
##    <int>  <int>  <int>  <int>
##  1    14      1      3      6
##  2    20      2      2      2
##  3     4      3      1      1
##  4     2      4      6      4
##  5    13      5      4      3
```

```
##  6    10      6      5      5
##  7    27      7      7      7
##  8     6      8      9      9
##  9     1      9      8      8
## 10    19     10     11     12
## # ... with 35 more rows
```

```
rankings %>%
  group_by(store) %>%
  summarize(ranking = sum(score)) %>%
  arrange(ranking) #the lower the score, the better performing
```

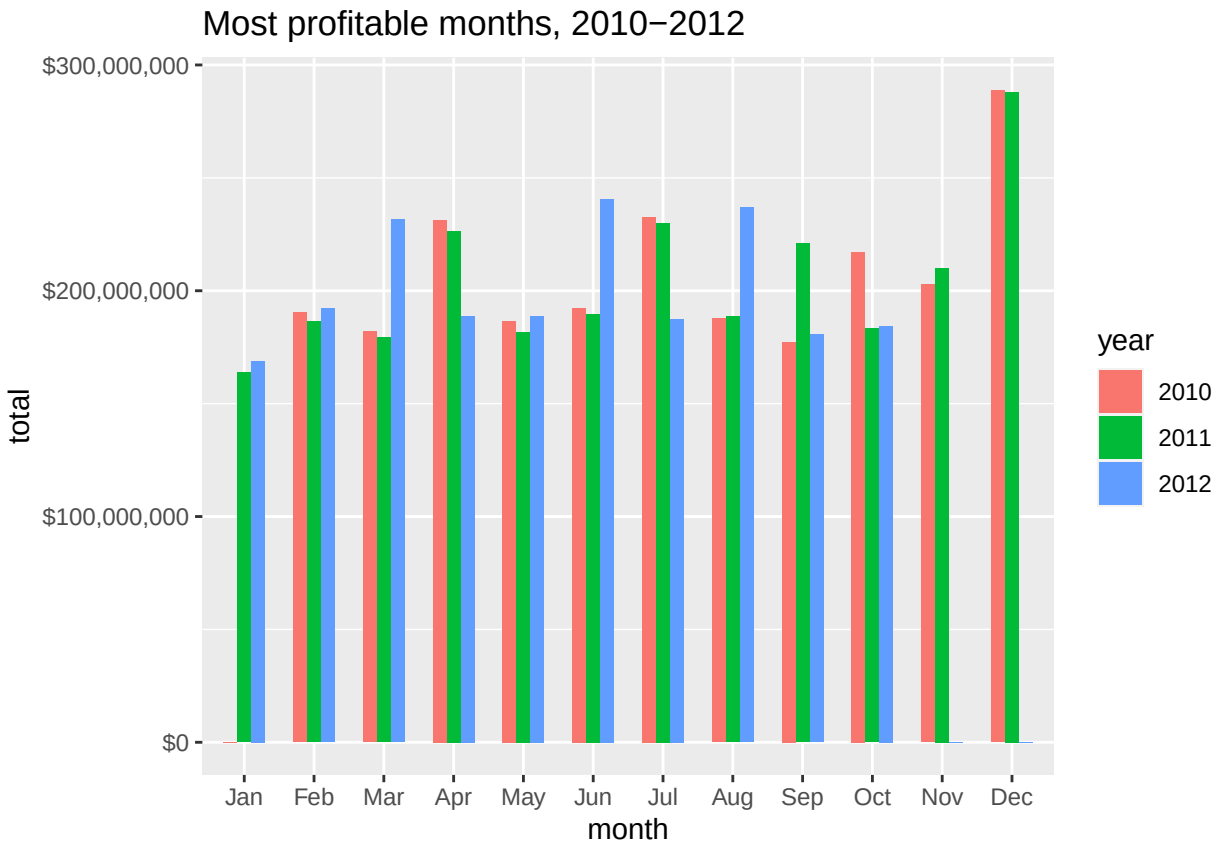
```
## # A tibble: 45 x 2
##    store ranking
##    <int>   <int>
##  1     4       5
##  2    20       6
##  3    14      10
##  4    13      12
##  5     2      14
##  6    10      16
##  7    27      21
##  8     1      25
##  9     6      26
## 10    19      33
## # ... with 35 more rows
```

Checking the most profitable months.

```
#most profitable months?

year_sales = walmart %>%
  group_by(year(date), month(date)) %>%
  summarize(total = sum(weekly_sales)) %>%
  rename(month = `month(date)`,
         year = `year(date)`) %>%
  pivot_wider(names_from = year, values_from=total) %>% #doing this to use replace 0s
  arrange(month) %>%
  replace(is.na(.), 0) %>%
  pivot_longer(!month, names_to="year", values_to="total")
```

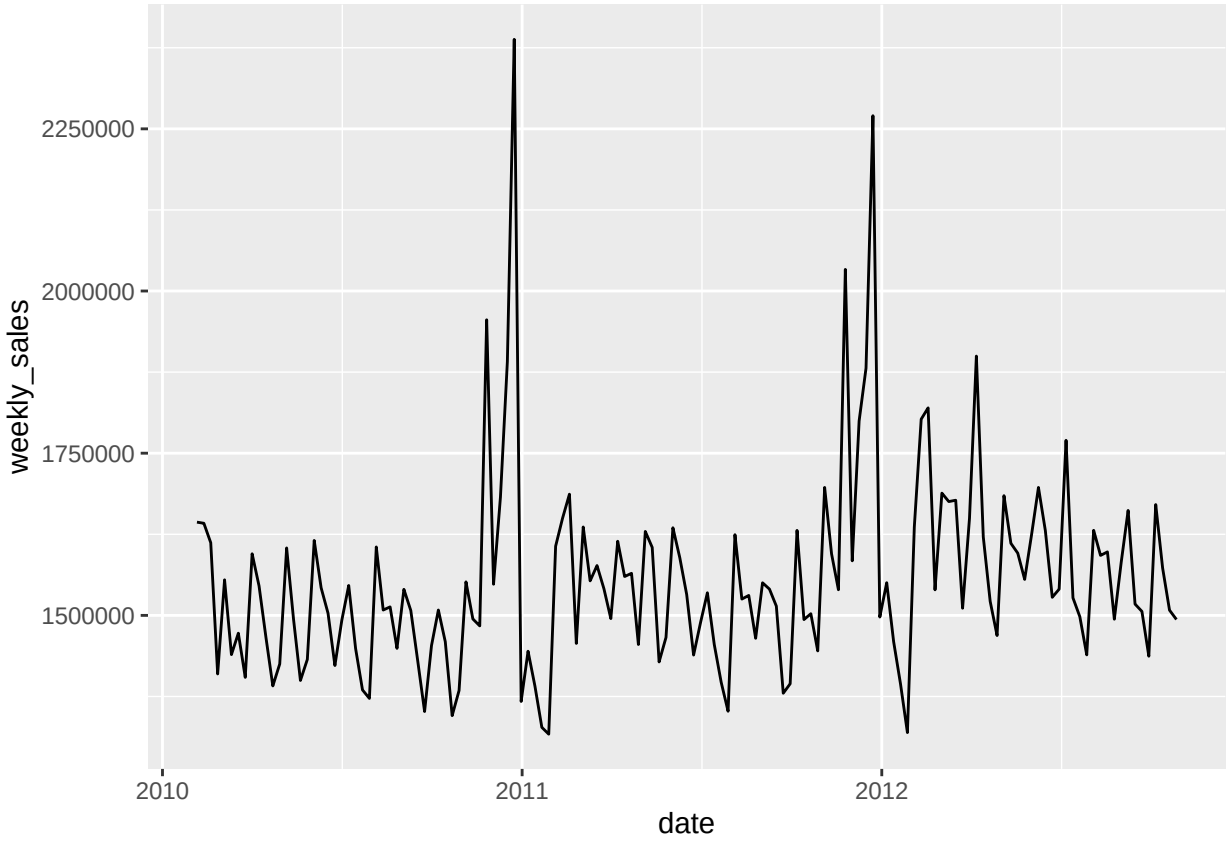
```
## `summarise()` has grouped output by 'year(date)'. You can override using the `.groups` argument.
```

```
ggplot(year_sales, aes(x=month,y=total,fill=year))+
  geom_bar(position="dodge", width=0.6, stat="identity") +
  scale_x_discrete(limits = month.abb) +
  scale_y_continuous(labels=scales::dollar_format()) +
  ggtitle(label="Most profitable months, 2010-2012")
```

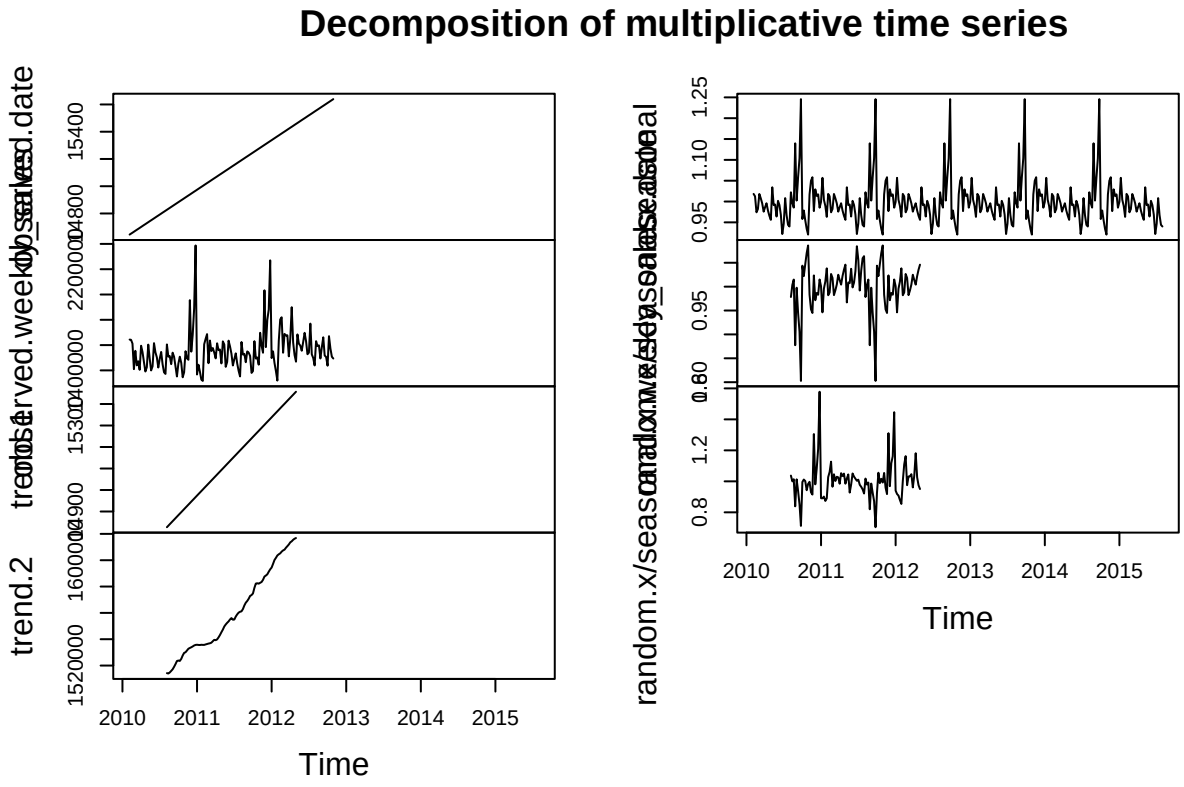


Build ARIMA model for store 1

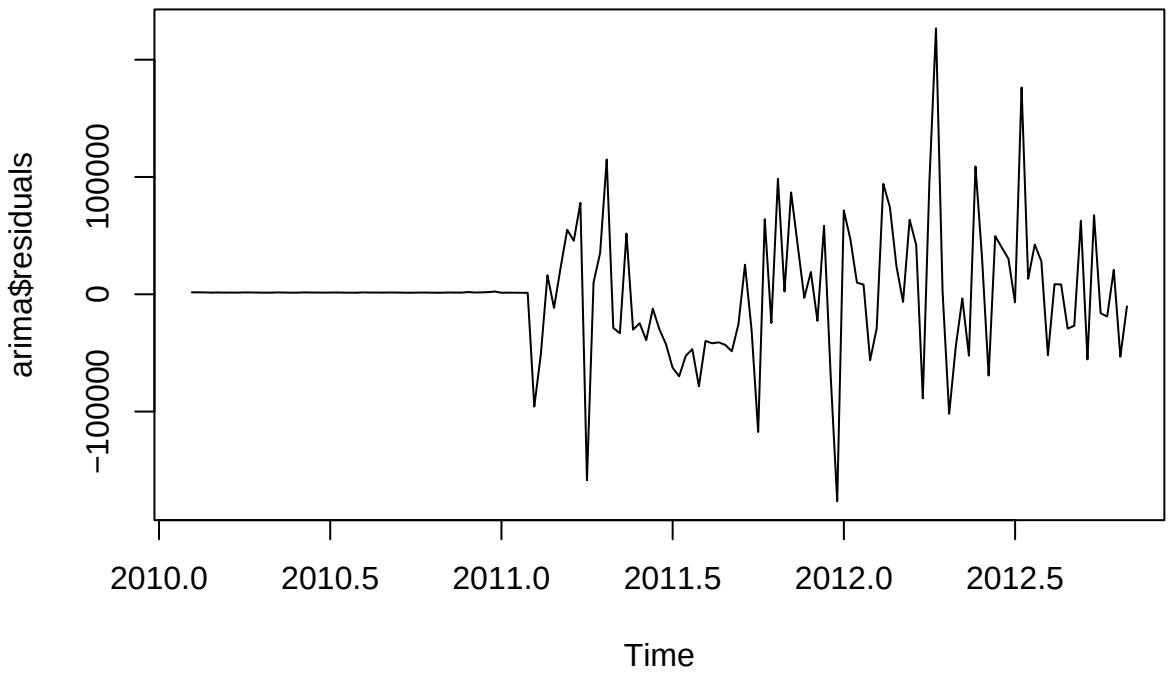
```
#Store 1 ARIMA

store1 = walmart %>% filter(store == 1) %>% select(date, weekly_sales)
ggplot(store1, aes(x=date, y=weekly_sales)) +
  geom_line()
```



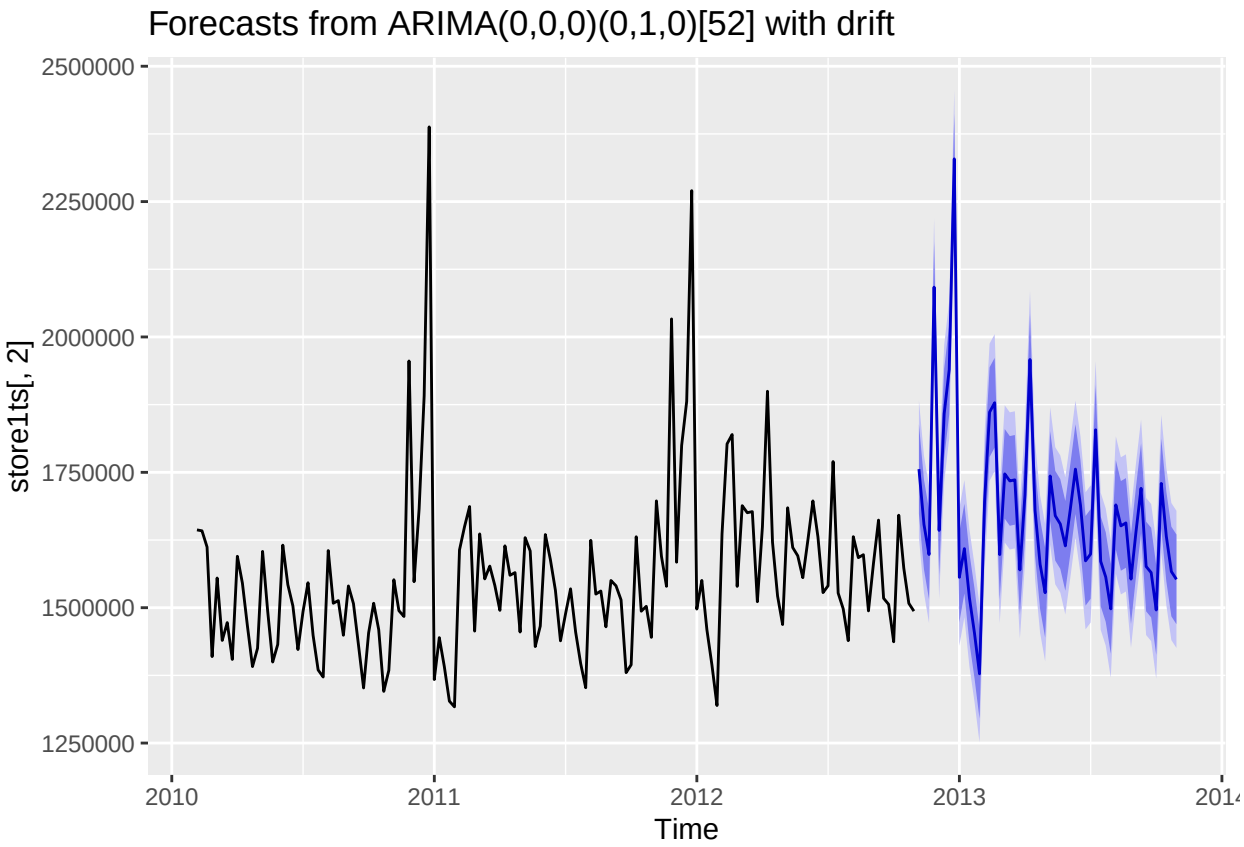

```
store1ts = ts(store1, frequency=52, start=c(decimal_date(ymd("2010-02-05"))))
store1d = decompose(store1ts, "multiplicative")
plot(store1d)
```




```
library(forecast)

arima = auto.arima(store1ts[,2])
plot.ts(arima$residuals)
```




```
forecast = forecast(arima, h=52)
autoplot(forecast)
```



Using the Ljung-Box test to check accuracy.

```
Box.test(arima$resid, lag=5, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  arima$resid
## X-squared = 5.0543, df = 5,
## p-value = 0.4093
```

```
Box.test(arima$resid, lag=10, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  arima$resid
## X-squared = 9.6349, df = 10,
## p-value = 0.4731
```

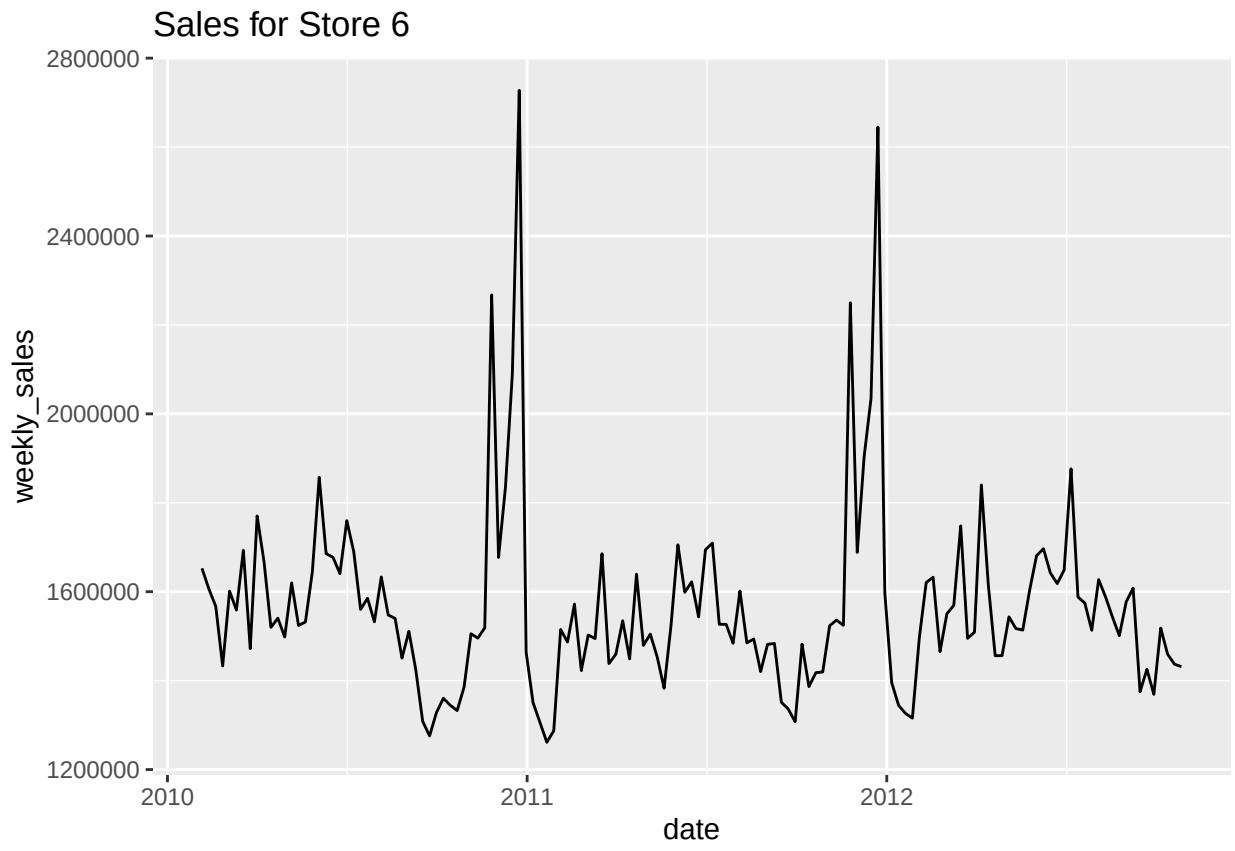
```
Box.test(arima$resid, lag=15, type="Ljung-Box")
```

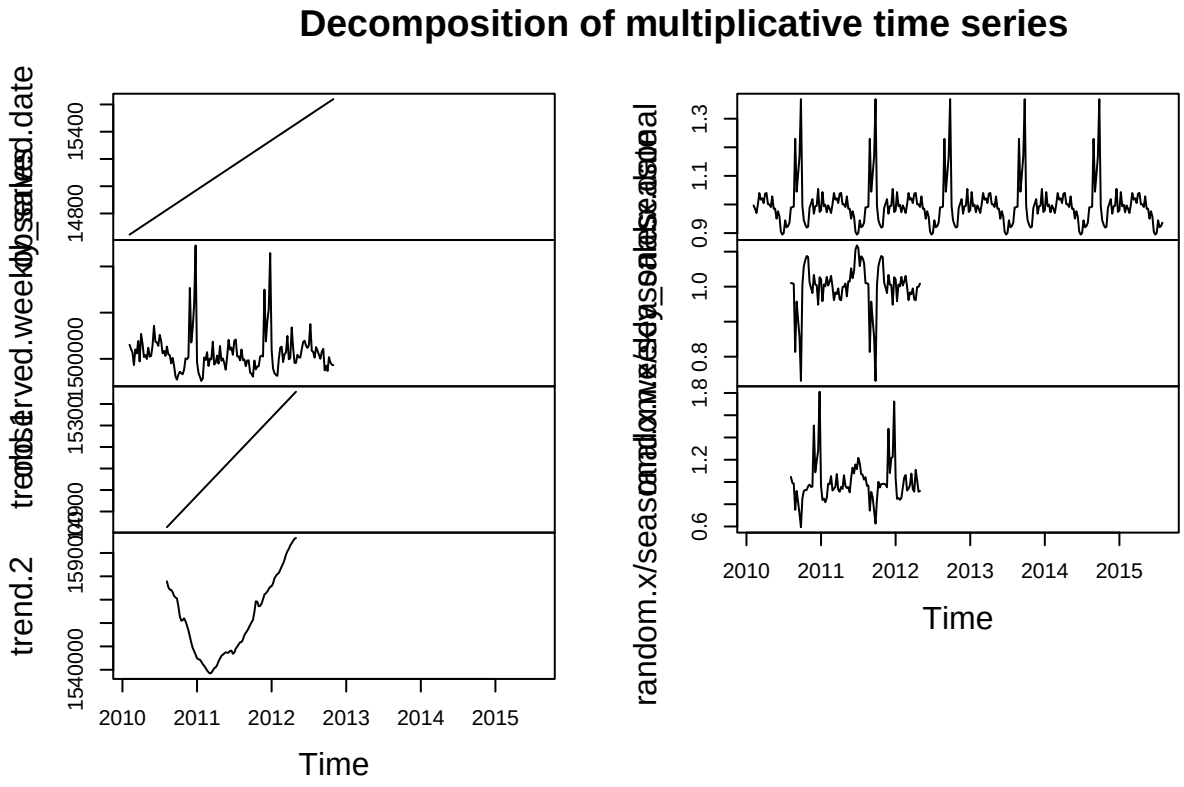
```
##
##  Box-Ljung test
```

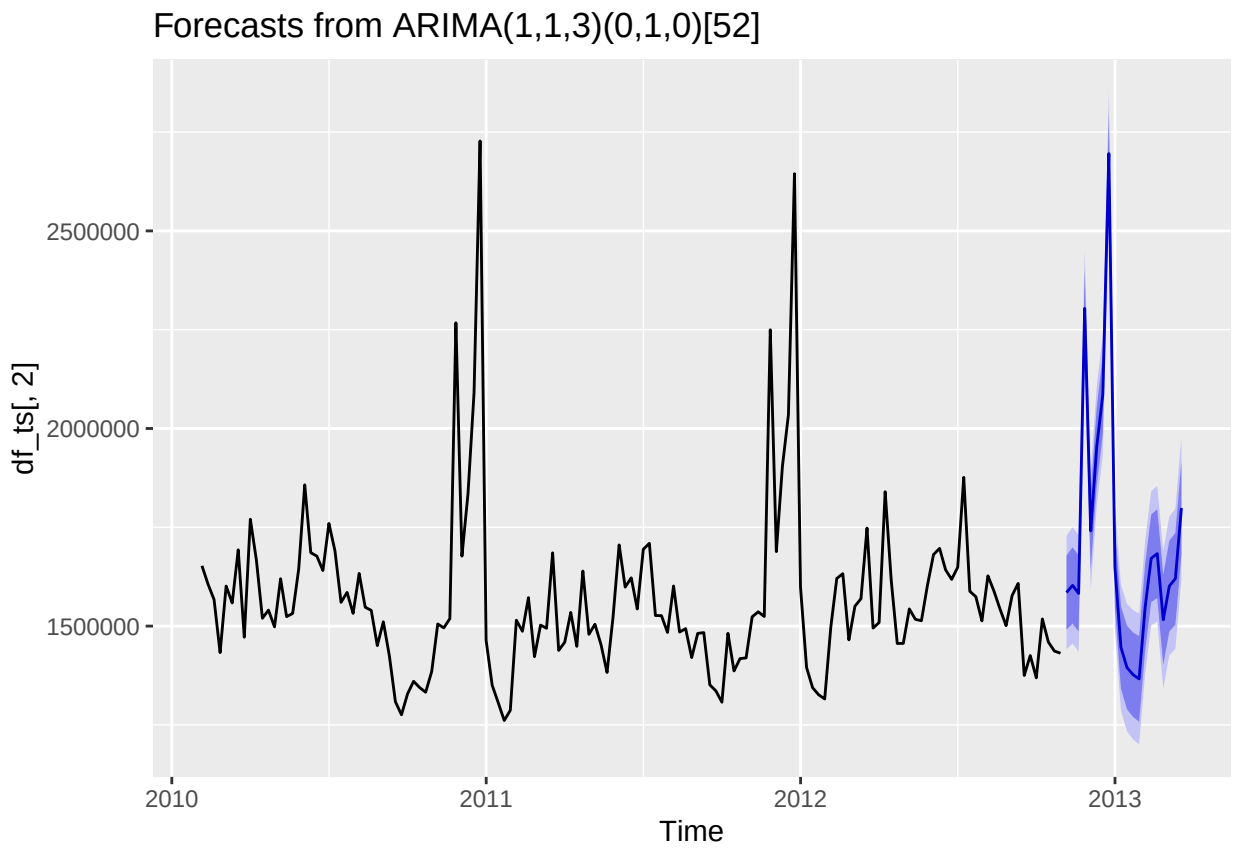
```
##
## data:  arima$resid
## X-squared = 22.434, df = 15,
## p-value = 0.09693
```

Writing a reusable ARIMA function.

```
store_arima = function(storeNum, weeks_forecasted) {
  df = walmart %>%
    filter(store == storeNum) %>%
    select(date, weekly_sales)

  print(ggplot(df, aes(x=date, y=weekly_sales)) +
      geom_line() +
      ggtitle(label=paste0("Sales for Store ", storeNum)))

  df_ts = ts(df, frequency=52, start=c(decimal_date(ymd("2010-02-05"))))

  df_d = decompose(df_ts, "multiplicative") %>% plot()

  forecast(auto.arima(df_ts[,2]), h=weeks_forecasted) %>% autoplot()
}

store_arima(6,20) #store_arima(which store, forecast how many weeks)
```




Decomposition of multiplicative time series
trend.2
random.x/seasonal.weekly_seasonal.date
Time
2010
2011
2012
2013
2014
2015


Forecasts from ARIMA(1,1,3)(0,1,0)[52]
df_ts[, 2]
2500000
2000000
1500000
2010
2011
2012
2013
Time