Walmart Store Sales Prediction

Dongjie Cheng

11/19/2020

Contents

1 Introduction

This project is based on a dataset of Walmart weekly sales originally posted on **kaggle.com**. Its link is: https://www.kaggle.com/aditya6196/retail-analysis-with-walmart-data. Since Kaggle.com requires user logon information, I transferred the data file to my Github site. As a result, my program will download the data file automatically from my Github rather than Kaggle.com.

The dataset, **Walmart_Store_sales.csv**, contains *6435* samples of weekly in-store sales for *45* Walmart stores in *143* consecutive weeks. It covers the time period from February 5, 2010 to November, 1, 2012. Moreover, the author also proposed a series of task requirements to the competitors in a description file and the file was also uploaded to my Github.

In this paper, I will present my study as a pretended competitor by following the original requirements. But, I also have to add extra nonlinear data analysis to meet the **HarvardX** course requirement, because the author only asked using linear regression algorithm. Another issue is that the original description did not mention the **unit** of the sales values. To be simple, I assume it is **US Dollar**. Finally, the following table lists the tasks from the author's assignment (linear regression requirement is removed):

Table 1: Assignment of Walmart weekly sales prediction

Basic statistics tasks	Statistical Model
1. Store with maximum Sales	1. prediction models to forecast demand for Store 1: hypothesize if CPI, unemployment, and fuel price have any impact on sales
2. Store with maximum standard deviation and its coefficient of mean	
3. Store(s) with good quarterly growth in Q3, 2012	
4. Holidays with higher sales than the mean sales in non-holidays for all stores	
5. Monthly and semester view of sales in units and insights	

I will address this two parts assignment sequentially. First, I run **basic statistics analysis** in the section of **Data Manipulation and Exploratory Analysis**. Then, **Statistical Model** in **Modeling Data Analysis** section.

2 Data Manipulation and Exploratory Analysis

The data file was provided in csv format having a size of *355 KB*. It contains *6435* records matching exactly *45* stores by *143* weeks. After download, I create a data frame, **dat**, in R to hold all the samples. Its summary is shown below:

```
# New location on GitHub
url<- "https://raw.githubusercontent.com/dongjiecheng/walmart_Store_Sales/main/Walmart_Store_sales.csv"

#read in data
dat<-read_csv(url)
#summary
glimpse(dat)
```

```
Rows: 6,435
```

```
Columns: 8
$ Store        <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...
$ Date         <chr> "5/2/2010", "12/2/2010", "19-02-2010", "26-02-2010", "...
$ Weekly_Sales <dbl> 1643691, 1641957, 1611968, 1409728, 1554807, 1439542, ...
$ Holiday_Flag <dbl> 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ Temperature  <dbl> 42.31, 38.51, 39.93, 46.63, 46.50, 57.79, 54.58, 51.45...
$ Fuel_Price   <dbl> 2.572, 2.548, 2.514, 2.561, 2.625, 2.667, 2.720, 2.732...
$ CPI          <dbl> 211.10, 211.24, 211.29, 211.32, 211.35, 211.38, 211.22...
$ Unemployment <dbl> 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 8.106, 8.106...
```

Easily we can find there are *8* features. The author has provided their names and explanation listed in the following table. The **Weekly_Sales** represents our outcome to be predicted and other *7* features would be our input for prediction. In this section, I will run **Data Cleaning and Manipulation** to have the data ready, then run **Exploratory Analysis** to complete the **Basic statistics tasks** listed in the top table, Table 1.

Table 2: Explanation of the features

Name	Explanation
Store	the store number
Date	the week of sales
Weekly_Sales	sales for the given store
Holiday_Flag	whether the week is a special holiday week 1 – Holiday week 0 – Non-holiday week
Temperature	Temperature on the day of sale
Fuel_Price	Cost of fuel in the region
CPI	Prevailing consumer price index
Unemployment	Prevailing unemployment rate

2.1 Data Cleaning and Manipulation

The following code shows the dataset has no empty cells and the frequency histogram of the **Weekly_Sales** in *log* scale also shows the weekly sales have a reasonable distribution with the peak about 1×10^6. Therefore, there are no errorneous values. However, looking at the **Date** column, we can spot two different formats as presented by the code below: either in "**dd/mm/yy**" or "**dd-mm-yy**" — we must unify the formats.

```
#check NA
sapply(dat, function(x)
  sum(is.na(x)))
```

```
      Store         Date Weekly_Sales Holiday_Flag  Temperature   Fuel_Price
          0            0            0            0            0            0
        CPI Unemployment
          0            0
```

```
# check Date formats
unique(dat$Date)
```

```
  [1] "5/2/2010"   "12/2/2010"  "19-02-2010" "26-02-2010" "5/3/2010"
  [6] "12/3/2010"  "19-03-2010" "26-03-2010" "2/4/2010"   "9/4/2010"
 [11] "16-04-2010" "23-04-2010" "30-04-2010" "7/5/2010"   "14-05-2010"
```

```
 [16] "21-05-2010" "28-05-2010" "4/6/2010"   "11/6/2010"  "18-06-2010"
 [21] "25-06-2010" "2/7/2010"   "9/7/2010"   "16-07-2010" "23-07-2010"
 [26] "30-07-2010" "6/8/2010"   "13-08-2010" "20-08-2010" "27-08-2010"
 [31] "3/9/2010"   "10/9/2010"  "17-09-2010" "24-09-2010" "1/10/2010"
 [36] "8/10/2010"  "15-10-2010" "22-10-2010" "29-10-2010" "5/11/2010"
 [41] "12/11/2010" "19-11-2010" "26-11-2010" "3/12/2010"  "10/12/2010"
 [46] "17-12-2010" "24-12-2010" "31-12-2010" "7/1/2011"   "14-01-2011"
 [51] "21-01-2011" "28-01-2011" "4/2/2011"   "11/2/2011"  "18-02-2011"
 [56] "25-02-2011" "4/3/2011"   "11/3/2011"  "18-03-2011" "25-03-2011"
 [61] "1/4/2011"   "8/4/2011"   "15-04-2011" "22-04-2011" "29-04-2011"
 [66] "6/5/2011"   "13-05-2011" "20-05-2011" "27-05-2011" "3/6/2011"
 [71] "10/6/2011"  "17-06-2011" "24-06-2011" "1/7/2011"   "8/7/2011"
 [76] "15-07-2011" "22-07-2011" "29-07-2011" "5/8/2011"   "12/8/2011"
 [81] "19-08-2011" "26-08-2011" "2/9/2011"   "9/9/2011"   "16-09-2011"
 [86] "23-09-2011" "30-09-2011" "7/10/2011"  "14-10-2011" "21-10-2011"
 [91] "28-10-2011" "4/11/2011"  "11/11/2011" "18-11-2011" "25-11-2011"
 [96] "2/12/2011"  "9/12/2011"  "16-12-2011" "23-12-2011" "30-12-2011"
[101] "6/1/2012"   "13-01-2012" "20-01-2012" "27-01-2012" "3/2/2012"
[106] "10/2/2012"  "17-02-2012" "24-02-2012" "2/3/2012"   "9/3/2012"
[111] "16-03-2012" "23-03-2012" "30-03-2012" "6/4/2012"   "13-04-2012"
[116] "20-04-2012" "27-04-2012" "4/5/2012"   "11/5/2012"  "18-05-2012"
[121] "25-05-2012" "1/6/2012"   "8/6/2012"   "15-06-2012" "22-06-2012"
[126] "29-06-2012" "6/7/2012"   "13-07-2012" "20-07-2012" "27-07-2012"
[131] "3/8/2012"   "10/8/2012"  "17-08-2012" "24-08-2012" "31-08-2012"
[136] "7/9/2012"   "14-09-2012" "21-09-2012" "28-09-2012" "5/10/2012"
[141] "12/10/2012" "19-10-2012" "26-10-2012"
```

```
# sales histogram plots
dat%>%group_by(Weekly_Sales)%>%
  ggplot(aes(Weekly_Sales))+geom_histogram(fill="blue",color="red")+scale_x_log10()+
  scale_fill_brewer(palette = "Spectral")+guides(color = "none")+
  labs(title = "Histogram of Weekly_Sales", y = "count", x = "weekly sales($)",
       caption="Walmart Stores Sales")
```

Using the following code, I convert all dates into "**dd-mm-yy**" format. Then, create a column **Day_class** based on **Holiday_Flag** to reflect the holiday names. A new dataset is then created, named **ndat** and the weeks with special and normal days are listed in the table next. The sum of the numbers of the days equals *143*. Now, we have a dataset ready to proceed for analysis.

```
# Date format conversion and differentiate normal day and holiday
ndat<-dat%>%mutate(Date=dmy(dat$Date),nDate=as.numeric(Date))%>%mutate(
  Holiday_Flag=factor(Holiday_Flag),
  Day_class=ifelse(Holiday_Flag==1,"Holiday","Normalday")
)
# setup holidays
index<-as.character(ndat$Date)%in%c("2010-02-12","2011-02-11","2012-02-10","2013-02-08")
ndat$Day_class[index]<-"Super_Bowl"
index<-as.character(ndat$Date)%in%c("2010-09-10","2011-09-09","2012-09-07","2013-09-06")
ndat$Day_class[index]<-"Labor_Day"
index<-as.character(ndat$Date)%in%c("2010-11-26","2011-11-25","2012-11-23","2013-11-29")
ndat$Day_class[index]<-"Thanksgiving"
index<-as.character(ndat$Date)%in%c("2010-12-31","2011-12-30","2012-12-28","2013-12-27")
ndat$Day_class[index]<-"Christmas"
```

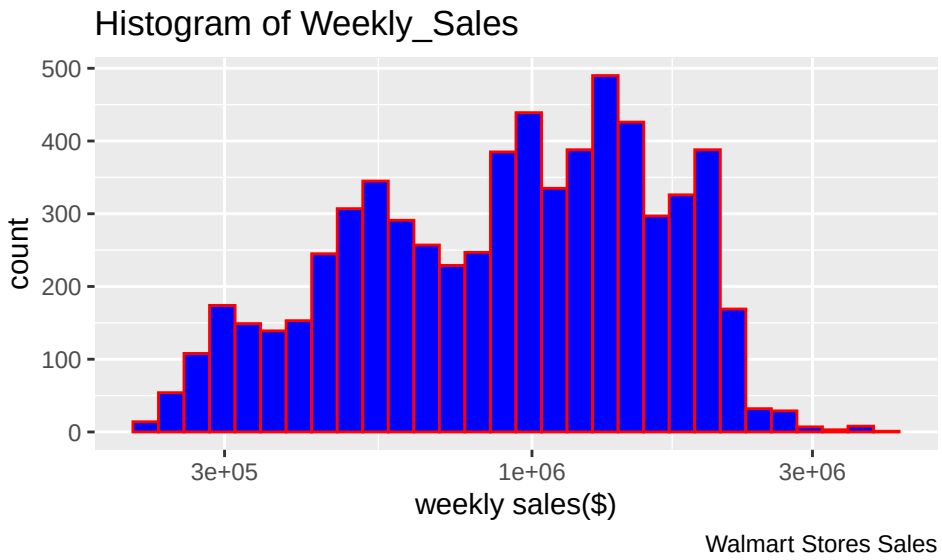

Figure 1: Frequency of weekly sales

Table 3: Date counts

Day_class	n
Christmas	2
Labor_Day	3
Normalday	133
Super_Bowl	3
Thanksgiving	2

```
# unique dates count
subset(ndat, !duplicated(subset(ndat, select=Date)))%>%count(Day_class)%>%
  knitr::kable(caption = "Date counts")%>%
  column_spec(1, width = "3cm")%>%column_spec(2, width = "3cm")%>%
  kable_styling(latex_options = c("striped", "center"))
```

2.2 Exploratory Analysis

Because the weekly sales are ordered by date, let us investigate the date information first. The code below shows there are *143* records uniformly for all store and the recording day is **Friday** for each week. The map next further shows the dates for all stores with holidays marked. It proves the correctness of our date conversion. Also, we know that the start date is 2010-02-05 and ending date is 2012-10-26. Therefore, we only have full year data for *2011*, but not *2010* and *2012*.

```
# check days
ndat%>%group_by(Store)%>%summarise(sum=n())%>%.$sum
```

```
 [1] 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143
[20] 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143 143
[39] 143 143 143 143 143 143 143
```

```
# how many week days
ndat%>%mutate(weekday=weekdays(Date))%>%summarize(week_days=unique(weekday))
```

```
# A tibble: 1 x 1
  week_days
  <chr>
1 Friday
```

```
# check date distribution
ndat%>%group_by(Date)%>%ggplot(aes(Date,Store,color=Day_class))+geom_point(size=0.01)+
  labs(title = "Map of dates", y = "store", x = "dates", caption="Walmart Stores Sales")
```

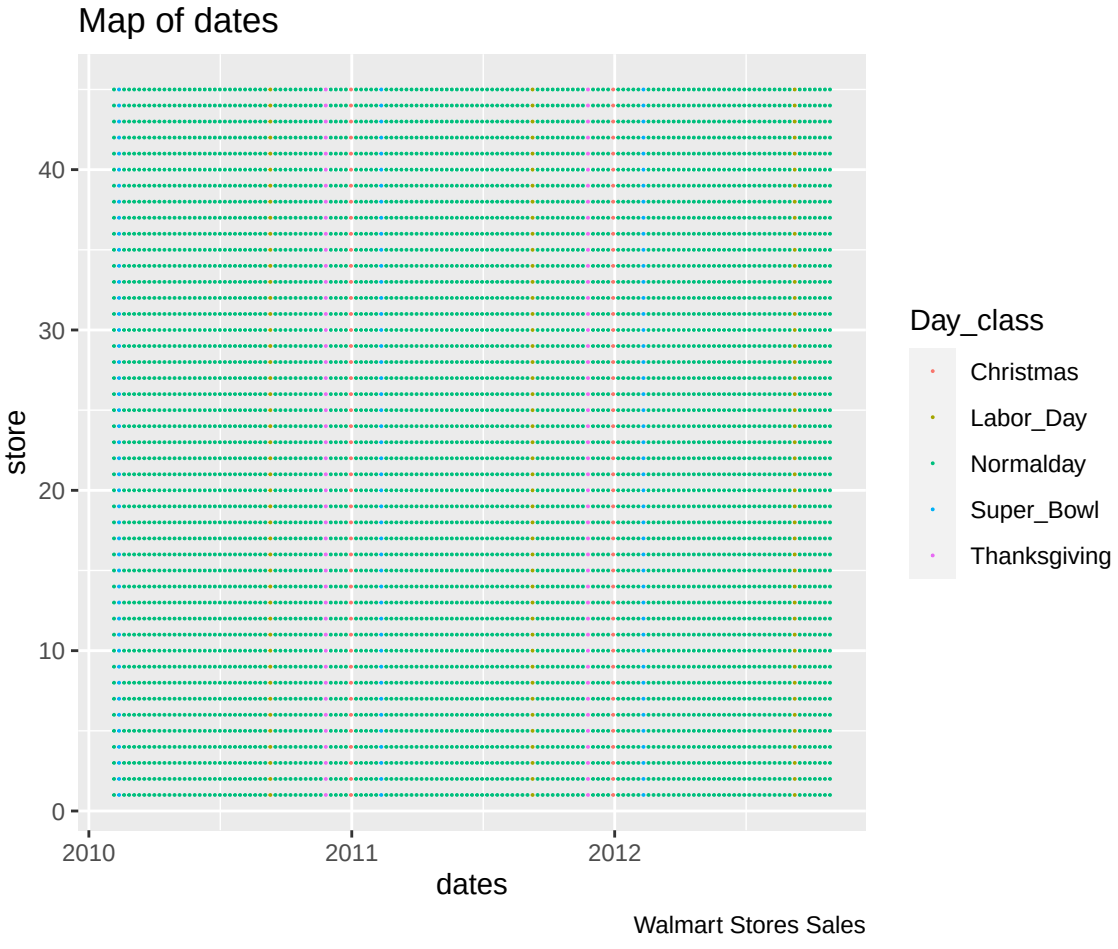

Figure 2: Date distribution of stores

Next, let us try to answer the questions one by one assigned by **Basic statistics tasks** above.

2.2.1 Task 1 and 2: Store with maximum sales and store with maximum standard deviation and its coefficient of mean

The figure below shows the **total store sales** with respect to the store number in *143* weeks. The sale values look random. Nevertheless, we can easily identify that Store *20* has the maximum accumulated sales.

```
# store total sales plot
ndat%>%group_by(Store)%>%summarize(whole_sale=sum(Weekly_Sales))%>%
  ggplot() +
  geom_line(aes(Store,whole_sale),color="blue")+
  geom_point(aes(Store,whole_sale), color="blue")+
  labs(title = "Plot of total store sales", y = "sales", x = "store number",
       caption="Walmart Stores Sales")
```

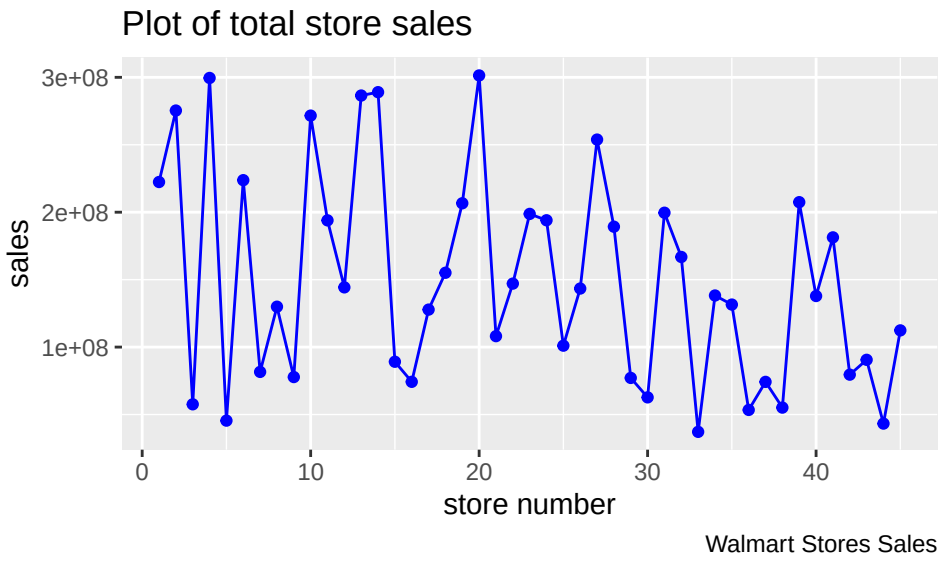

Figure 3: Total store sales for all 45 stores

The next figure shows the **mean store sales** with respect to the store number with error bars. Same to the total sales, the **mean store sales** look random and store *20* of course is the store with the highest **mean store sales**. The error bars are much small compared to the sales and smaller the sales, smaller error bars. That means that the sales of each store are well predictable with little uncertainty. The figure following shows the standard deviations and the standard errors for each store. Obviously Store *14* has the maximum standard deviation.

```
# store sales mean errorbar plot
ndat%>%group_by(Store)%>%summarize(sale_mean=mean(Weekly_Sales),
                                   std=sd(Weekly_Sales),n=n())%>%
  mutate(low=sale_mean-std/sqrt(n),high=sale_mean+std/sqrt(n))%>%
  ggplot(aes(Store,sale_mean))+
  geom_point(color="red",size=0.8)+
  geom_errorbar(aes(ymin=low, ymax=high), color="blue", width=.9,
                position=position_dodge(.9))+
  labs(title = "Bar plot of store sales", y = "store sales mean",
       x = "store number", caption="Walmart Stores Sales")
```

```
# store sales standard deviation and standard errors line plots
ndat%>%group_by(Store)%>%summarize(sale_mean=mean(Weekly_Sales),std=sd(Weekly_Sales),n=n())%>%
  mutate(ste=std/sqrt(n))%>%
  ggplot() +
  geom_line(aes(Store,std,color="std"))+geom_point(aes(Store,std,color="std"))+
```

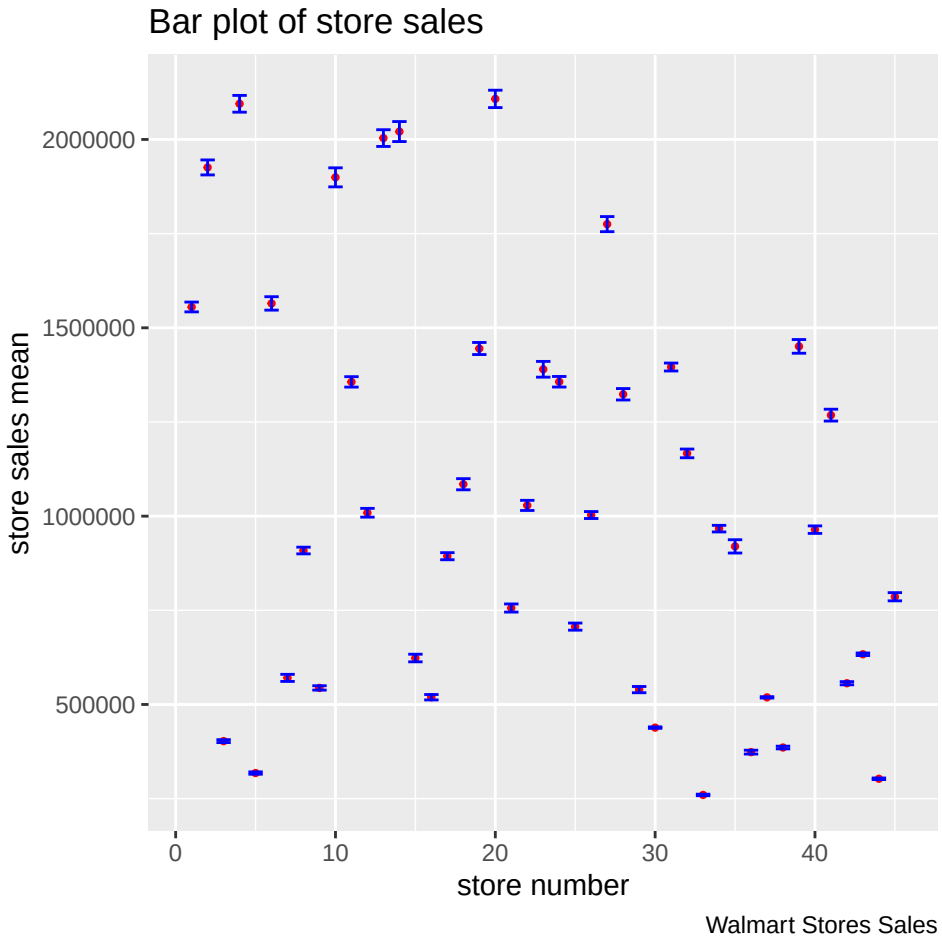

Figure 4: Mean sales with error bar of all stores

```
  geom_line(aes(Store,ste,color="ste"))+geom_point(aes(Store,ste,color="ste"))+
  scale_color_discrete(name="Errors",labels=c("Standard deviation","standard error"))+
  labs(title = "Plot of standard deviation and standard error",
       y = "error", x = "store number", caption="Walmart Stores Sales")
```

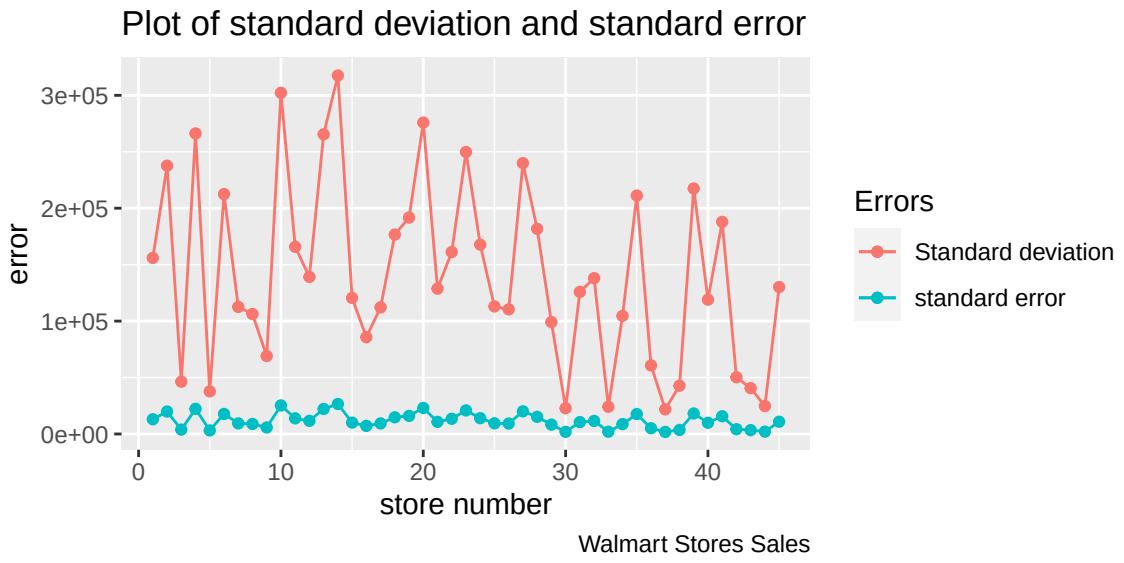

Figure 5: Standard deviation and standard error

Further investigation below can also prove the conclusion above. Store *20* has the maximum **total store sales** and the maximum **mean store sales**, which are *$301397792* and *$2107677* respectively. Meanwhile, Store *14* has the **mean store sales** of *$2020978* and the maximum **standard deviation** of *$317570*.

```
# store with maximum total sales
mTotal_store<-ndat%>%group_by(Store)%>%summarize(whole_sale=sum(Weekly_Sales),
                                                 sale_mean=mean(Weekly_Sales))%>%
  summarize(max_whole_sale=max(whole_sale),
            store_max_whole_sale=Store[which.max(whole_sale)])
mTotal_store
```

```
# A tibble: 1 x 2
  max_whole_sale store_max_whole_sale
           <dbl>                <dbl>
1     301397792.                   20
```

```
# store with maximum mean sales
mMean_store<-ndat%>%group_by(Store)%>%summarize(sale_mean=mean(Weekly_Sales),
                                                std=sd(Weekly_Sales))%>%
  summarize(max_mean=(max(sale_mean)),store_max_mean=Store[which.max(sale_mean)])
mMean_store
```

```
# A tibble: 1 x 2
  max_mean store_max_mean
     <dbl>          <dbl>
1 2107677.             20
```

```
# store with maximum standard deviation and its mean
mStd_store<-ndat%>%group_by(Store)%>%summarize(sale_mean=mean(Weekly_Sales),
                                               std=sd(Weekly_Sales))%>%
  summarize(max_standard_deviation=(max(std)),
            store_max_standard_deviation=which.max(std),
            store_mean=sale_mean[which.max(std)] )
mStd_store
```

```
# A tibble: 1 x 3
  max_standard_deviation store_max_standard_deviation store_mean
                   <dbl>                         <int>      <dbl>
1                317570.                            14   2020978.
```

Now let us compute the **variation coefficient** required by Task *2*. Its definition is:

$$c = \frac{\sigma}{\mu}$$

where c denotes the coefficient, μ denotes the **mean** and σ denotes the **standard deviation**. So that, the coefficient for Store *14* can be derived by:

```
coeff<-mStd_store$max_standard_deviation/mStd_store$store_mean
coeff
```

```
[1] 0.15714
```

2.2.2 Task 3: Store(s) with good quarterly growth in Q3, 2012

Regarding to the growth issue of Q3, 2012, let us assume the store with **good quarterly growth** is that who has **positive** quarterly sale growth. First, I read the sales data of Q2 and Q3 into a new data frame, ***q2_3_sales*** below. Second, the code next counting week numbers for the quarters can prove that there are *13* weeks for each quarter. Therefore it is fair to compare the two quarterly sales directly. The figure further down shows the bar plot of the total sales for all store in Q2 and Q3. Intuitively, we can find that the majority stores have worse sales in Q3, 2012.

```
# Q3 2012 growth
# all Q 2 & 3 2012 sales
q2_3_sales_all<-ndat%>%filter(Date>"2012-03-31"&Date<"2012-10-01")%>%
  mutate(quarter=ifelse(Date>"2012-06-30",3,2))

# count weeks
q2_3_sales_all%>%filter(Store==1)%>%count(quarter)%>%
  knitr::kable(caption="Week numbers of Q2 and Q3")%>%
  column_spec(1, width = "3cm")%>%column_spec(2, width = "3cm")%>%
  kable_styling(latex_options = c("striped","hold_position"))
```

Table 4: Week numbers of Q2 and Q3

quarter	n
2	13
3	13

```
#  compute quarterly sales for Q 2 & 3
q2_3_sales<-q2_3_sales_all%>% group_by(Store,quarter)%>%
  summarize(quarter_sales=sum(Weekly_Sales))%>%
  mutate(Store=as.factor(Store),quarter=as.factor(quarter))

# Q 2 & 3 2012 quarterly sales plot
q2_3_sales%>%group_by(quarter)%>%ggplot(aes(Store,quarter_sales,fill=quarter))+
  geom_col(position = "dodge")+
 scale_fill_manual(values = alpha(c("blue", "red"), .9))+
labs(title = "Plot of store sales, quater 2&3, 2012", y = "sales",
     x = "store number", caption="Walmart Stores Sales")
```

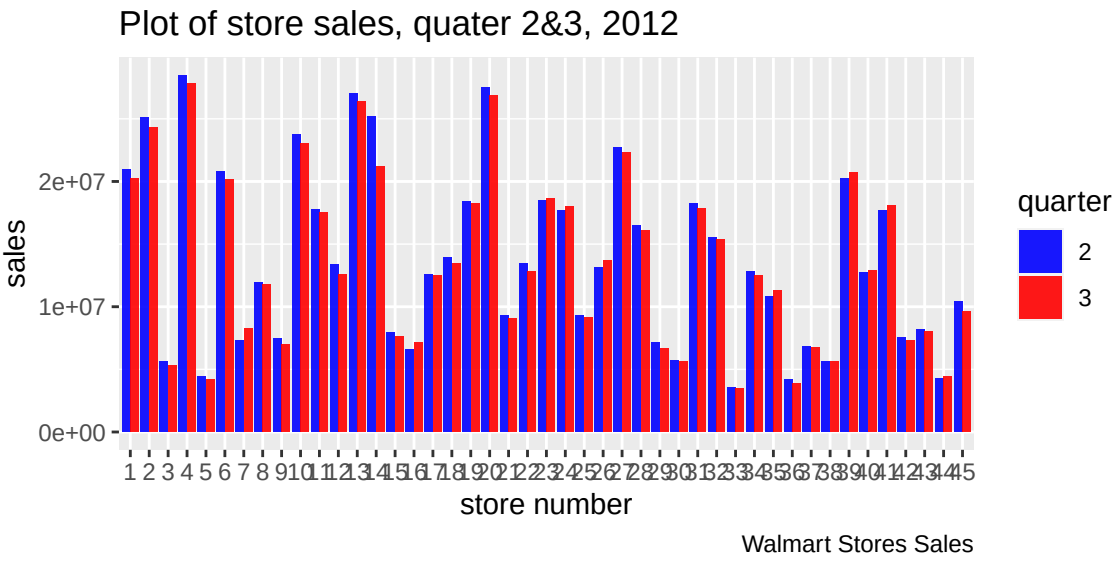

Figure 6: Standard deviation and standard error

Next, let us compute the Q3 growth of **growth** in **Dollar** values and relative **percentage** in the following chunk of code. The next two side by side figures show the **store sale growth** in terms of total sales and percentage. The green dash lines marks the **zero growth** point. Clearly, the most stores' **sale growth** is negative in Q3, 2012. However, we can observe that there are ***10*** stores have positive growth. Store ***7*** is the best performer. It has achieved a ***13.3%*** quarterly increase. In one word, 22.222 percent stores have a good quarter in Q3, 2012.

```
# Q3 growth compute & plot

# Q3 absolute growth in dollars
q3_abs_grow<-q2_3_sales%>%group_by(Store)%>%
  summarize(quarter_grow=(quarter_sales-lag(quarter_sales)))%>%
  filter(!is.na(quarter_grow))

# Q3 relative growth in percentage
q3_grow<-q2_3_sales%>%filter(quarter==2)%>%left_join(q3_abs_grow)%>%
  mutate(percent=quarter_grow/quarter_sales*100,Store=as.numeric(Store))

# Q3 abs growth plot
q3_grow%>%ggplot(aes(x=Store)) +
```

```
  geom_line(aes(y=quarter_grow), size=1, color="red") +geom_point(aes(y=quarter_grow))+
  scale_color_discrete(name="Growth",labels=c("Absolute value","Percentage"))+
  geom_hline(yintercept=0,size=1., color="green",linetype="dashed")+
  labs(title = "Plot of store sales growth, quater 3, 2012", x="store number",
       y="sales growth ($)", caption="Walmart Stores Sales")

# Q3 relative growth plot
q3_grow%>%ggplot(aes(x=Store)) +
  geom_line(aes(y=percent), size=1, color="blue") +geom_point(aes(y=percent))+
    geom_hline(yintercept=0,size=1., color="green",linetype="dashed")+
  labs(title = "Plot of store sales growth, quater 3, 2012", x="store number",
       y="sales growth (percentage)", caption="Walmart Stores Sales")
```

Figure 7: Store growth of Q3, 2012

The store names with **good growth** are listed in the table created by the code below ordered descendingly by the **percentage increment**. Please note that Stores *7* and *16* are the top two stores with more than *5%* growth.

```
# table of good performance stores
good_quart_store<-q3_grow%>%subset(quarter_grow>0)%>%
  select(Store,quarter_grow, percent)%>%arrange(desc(percent))
good_quart_store%>% knitr::kable(caption = "Stores with good growth")%>%
  column_spec(1, width = "3cm")%>%column_spec(2, width = "3cm")%>%
  column_spec(3, width = "3cm")%>%
  kable_styling(latex_options = c("striped","hold_position"))
```

Table 5: Stores with good growth

Store	quarter_grow	percent
7	971928	13.3308
16	557206	8.4884
35	484108	4.4666
26	520356	3.9555
39	500988	2.4784
41	433901	2.4570
44	104845	2.4346
24	292159	1.6521
40	145458	1.1428
23	152606	0.8254

2.2.3 Task 4: Holidays with higher sales than the mean sales in non-holidays for all stores

First, let us split the dataset **ndat** into two datasets: **holiday_sale** and **non_holiday sale**. Then, compute the holiday mean sales and the non-holiday mean sale for each store in the following chunk of code. The figure after shows the box plot of the holidays' weekly sales overlaid by their means and the mean sale for all stores. From the figure and the print-out numbers , we can derive that *3* holidays have higher mean sales than the non-holidays' mean sale except **Christmas**, though the **Labor day's** is very tight. Of them, the **Thanksgiving** sale is the highest, probably because of **Black Friday**. Moreover, it is very interesting to note that the **uncertainty** increases along with the sales. As a reference, the box plot of the non-holiday store sales is also present next. It again shows a similar pattern: the stores with smaller sales has smaller uncertainty.

```
# holiday store sale
holiday_sale<-ndat%>%filter(Holiday_Flag==1)

# holiday store mean sale for individual holiday
holiday_mean<-ndat%>%filter(Holiday_Flag==1)%>%group_by(Day_class)%>%
  summarize(mean_sale=mean(Weekly_Sales ))
holiday_mean%>% knitr::kable(caption = "Holiday mean sales")%>%
  column_spec(1, width = "3cm")%>%column_spec(2, width = "3cm")%>%
  kable_styling(latex_options = c("striped","hold_position"))
```

Table 6: Holiday mean sales

Day_class	mean_sale
Christmas	960833
Labor_Day	1042427
Super_Bowl	1079128
Thanksgiving	1471273

```
# non-holiday store sales
non_holiday_sale<-ndat%>%filter(Holiday_Flag==0)%>%group_by(Store)
# non-holiday mean store sales
non_holiday_mean<-ndat%>%filter(Holiday_Flag==0)%>%ungroup()%>%
  summarize(nonholiday_mean_sale=mean(Weekly_Sales ))
non_holiday_mean
```

```
# A tibble: 1 x 1
  nonholiday_mean_sale
                 <dbl>
1             1041256.
```

```
# store sales each holiday compared to non-holiday sales, plot
holiday_sale%>%group_by(Day_class)%>%mutate(mean_sale=mean(Weekly_Sales))%>%
  ggplot(aes(Day_class,Weekly_Sales ))+geom_boxplot()+
  geom_line(aes(Day_class,mean_sale,group=1),color="blue")+
  geom_point(aes(Day_class,mean_sale,group=1),color="blue")+
  geom_text(aes(4,1471273,label="holiday mean", vjust=-0.5),color="blue")+
  geom_hline(yintercept=1041256,size=1., color="red",linetype="dashed")+
  geom_text(aes(4,1041256,label="non-holiday mean", vjust=-0.5),color="red")+
     labs(title = "Plot of store holiday sales", x="store number",
          y="weekly sales($)", caption="Walmart Stores Sales")
```

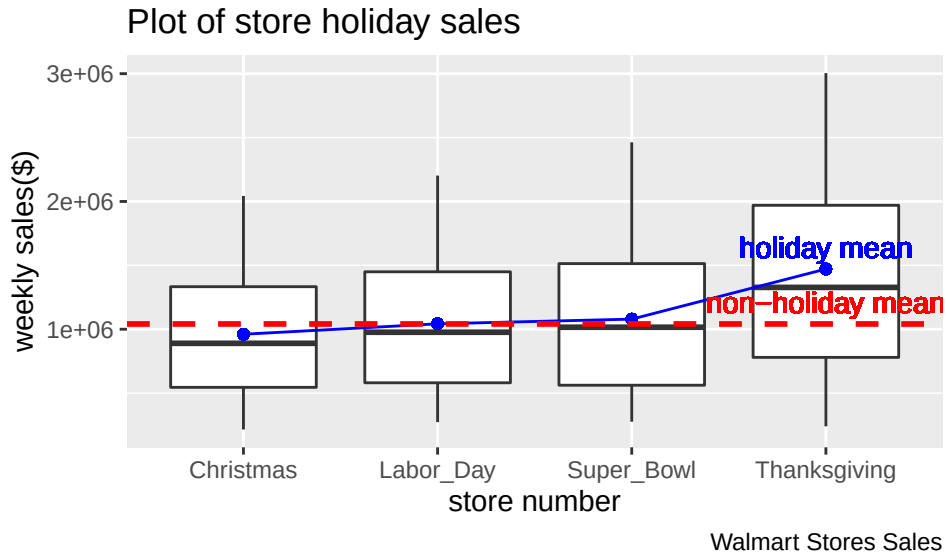

Figure 8: Store holiday sales vs. non-holiday mean sale

```
# non-holiday sales for each store, plot
non_holiday_sale%>%mutate(Store=as.factor(Store))%>%ggplot(aes(Store,Weekly_Sales ))+
  geom_boxplot()+
labs(title = "Plot of store non-holiday sales", x="store number", y="sales($)",
     caption="Walmart Stores Sales")
```

Figure 9: Store non-holiday sales

2.2.4 Task 5: Monthly and semester view of sales in units and insights

First, I can not determine the term of **Semester sale** from the references. Therefore, let us just assume the **Semester sale** means **quarterly sale**. The code below creates a new data frame **ndat1** with additional necessary variables for the analysis: **month**, **quarter** and **year**. As shown in the two side-side figures next, the numbers of the weeks for each month or quarter are not uniform in our dataset. Moreover, there are *3* missing months: **January** for *2010*, and **November** and **December** for *2012*. Thus, it does not make sense to compare the total sales for each month or quarter. Rather, let us focus on the mean and median values in the coming graphic analyses.

```
# create new variables: month, year, quarter
ndat1<-ndat%>%mutate(month=month(Date),year=year(Date))

#set up quarters
ndat1<-ndat1%>%mutate(quarter=1)
# quarter 2
index<-ndat1$month>=4 & ndat1$month<=6
ndat1$quarter[index]<-2
# quarter 3
index<-ndat1$month>=7 & ndat1$month<=9
ndat1$quarter[index]<-3
# quarter 4
index<-ndat1$month>=10 & ndat1$month<=12
ndat1$quarter[index]<-4

# make month, year, quarter factors
ndat1<-ndat1%>%mutate(month=as.factor(month),year=as.factor(year),quarter=as.factor(quarter))

# weeks per month plot
ndat1%>%filter(Store==1)%>%ggplot(aes(month,fill=year))+geom_bar(position = "dodge")+
labs(title = "Plot of count of weeks per month", x="month", y="week numbers",
     caption="Walmart Stores Sales")
```

```
# weeks per quarter plot
ndat1%>%filter(Store==1)%>%ggplot(aes(quarter,fill=year))+geom_bar(position = "dodge")+
labs(title = "Plot of count of weeks per quarter", x="quarter", y="week numbers",
     caption="Walmart Stores Sales")
```

Figure 10: Week numbers of each month and each quarter

Considering the monthly sales, below are the box plot of the monthly sales and the bar plot of the monthly mean sales for all stores. Although, there are missing data, we can still find a general pattern. Basically, the stores have worst sales in **January**. After that, the sales grow gradually up to the **middle** year. Since then, the sales slide down slowly to the **start** of the holiday season. Interestingly, **February** is an outlier. It is the **fourth** best month of each year. The sales jump in this month and are much better than that of the adjacent months. The best sales however come from **November** and **December**, although we are missing the data for *2012*. Year-wise, in *2011* the most stores seem have a bad year. Most months, the stores have smaller or close sales than the previous year, except **October** and **November**. In year *2012*, the sales bounced back and are better than year *2010*. Also, please note that the high-end outliers in **November** and **December**, showing the fluctuation in holidays.

```
# monthly sale box plot
ndat1%>%group_by(year,month)%>%
  ggplot(aes(month,Weekly_Sales,color=year))+geom_boxplot()+
  labs(title = "Plot of monthly sales", x="month", y="sale($)", caption="Walmart Stores Sales")

# monthly mean sale bar plot
ndat1%>%group_by(year,month)%>%summarise(mean_sale_month=mean(Weekly_Sales))%>%
  ggplot(aes(month,mean_sale_month,fill=year))+geom_col(position = "dodge")+
labs(title = "Plot of monthly mean sales", x="month", y="sale($)", caption="Walmart Stores Sales")
```

For the quarterly sales, below are the box plot of the quarterly sales and the bar plot of the quarterly mean sales of the all stores. As mentioned from above, we miss the sales of **January**, year *2010* and sales of **November** and **December**, 2012. Therefore, we have to exclude the corresponding data of quarter *1*, *2010* and quarter *4*, *2012*. Now looking at the figures below, we can derive a similar pattern as the monthly data does. Quarter *1* is the worst and Quarter *2* becomes better, then, Quarter *3* drops again. Finally, Quarter *4* receives a big jump and it is the best quarter for each year.

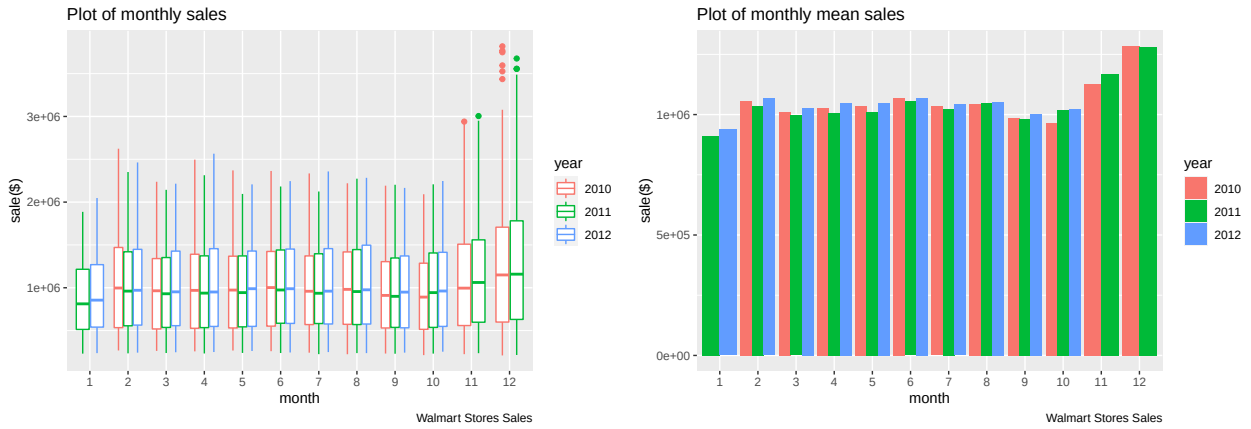

Figure 11: Store monthly sales and mean sales for all stores

```
# quarterly sale box plot
ndat1%>%group_by(year,quarter)%>%
  ggplot(aes(quarter,Weekly_Sales,color=year))+geom_boxplot()+
  labs(title = "Plot of quaterly sales", x="month", y="sale($)", caption="Walmart Stores Sales")

# quarter mean sale bar plot
ndat1%>%group_by(year,quarter)%>%summarise(mean_sale_quarter=mean(Weekly_Sales))%>%
  ggplot(aes(quarter,mean_sale_quarter,fill=year))+geom_col(position = "dodge")+
  labs(title = "Plot of quaterly mean sales", x="month", y="sale($)", caption="Walmart Stores Sales")
```

Figure 12: Store quaterly sales and mean sales for all stores

Summarizing the study of the *5* tasks by **Basic statistics tasks**, I put all my answers in the following table.

Table 7: Answers to basic statistics tasks

Basic statistics tasks	Answers
1. Store with maximum Sales	Store *20* has the maximum sales of *$301397792*.
2. Store with maximum standard deviation and its coefficient of mean	Store *14* has the maximum standard deviation of $317570 and its coefficient of variation is *0.157*.
3. Store(s) with good quarterly growth in Q3, 2012	*10* stores have positive growth and the top two are Stores *7* and *16*.
4. Holidays with higher sales than the mean sales in non-holidays for all stores	Thanksgiving, Super Bowl and Labor Day.
5. Monthly and semester view of sales in units and insights	Quart 1 and January are the worst quarter and month respectively, however February is the fourth highest month in each year. Quart 2 is better thank Quarter 1 and Quarter 3. Quarter 4 is always the best.

3 Modeling Data Analysis

Let us revisit the **Statistical model** assignment: ***"prediction models to forecast demand for Store 1: hypothesize if CPI, unemployment, and fuel price have any impact on sale"***.

Per requirement, I am only going to work on the weekly sales data of Store *1* in this section. For this store, we know, the dataset will have *143* weekly samples.

First, let us have a insight look at Store 1 data. The following two side-by side figures show the monthly and quarterly sales of the store. Unlike what we saw above all stores' case, Store 1 has positive growth from year to year. The Q4 decrease of *2012* is only due to missing data.

```
# Store 1 monthly sale box plot
ndat1%>%group_by(year,month)%>%filter(Store==1)%>%
  ggplot(aes(month,Weekly_Sales,color=year))+geom_boxplot()+
  labs(title = "Plot of  Store 1 monthly sales", x="month", y="sale($)",
       caption="Walmart Stores Sales")

# Store 1 quarterly sale box plot
ndat1%>%group_by(year,quarter)%>%filter(Store==1)%>%
  ggplot(aes(quarter,Weekly_Sales,color=year))+geom_boxplot()+
  labs(title = "Plot of  Store 1 quaterly sales", x="month", y="sale($)",
       caption="Walmart Stores Sales")
```

Figure 13: Quaterly weekly sales and mean sales for Store 1

Since our objective is to examine the impact of **CPI**, **Unemployment** and **Fuel_Price** to the weekly sales, let us also have quick look at their behaviors. The next *3* figures shows the corresponding quarterly variation respectively. The **CPI** demonstrates a distinct pattern: it grows year by year. On the other hand, the **Unemployment** and **Fuel_Price** have different type of patterns. The **Unemployment rate** stays high in first two years with no clear boundary, and then, it drops in the third year. In contrast, the **Fuel_Price** stays low in first year, and then, jumps in second year. But, comparing the second and third year. there is now significant difference.

```
# quarterly CPI box plot
ndat1%>%group_by(year,quarter)%>%filter(Store==1)%>%
  ggplot(aes(quarter,CPI,color=year))+geom_boxplot()+
  labs(title = "Plot of quaterly CPI", x="month", y="CPI", caption="Walmart Stores Sales")
```

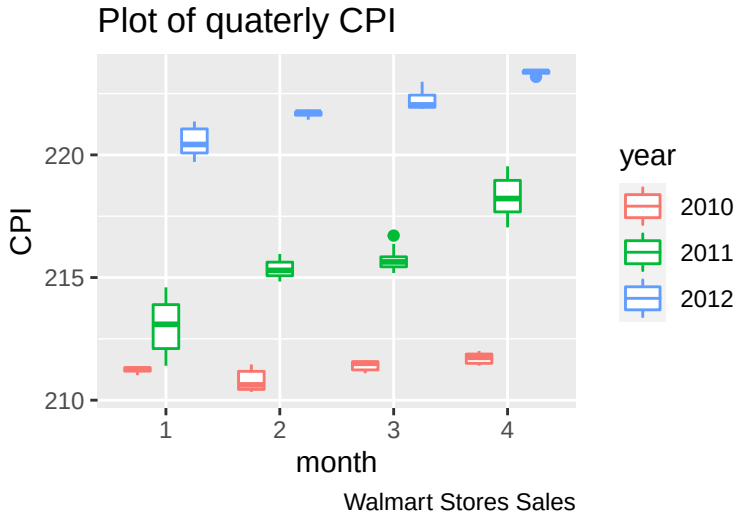

Figure 14: Count of CPI

```
# quarterly Unemployment box plot
ndat1%>%group_by(year,quarter)%>%filter(Store==1)%>%
  ggplot(aes(quarter,Unemployment,color=year))+geom_boxplot()+
  labs(title = "Plot of quaterly Unemployment", x="month", y="Unemployment",
       caption="Walmart Stores Sales")
```

Figure 15: Count of Unemployment

```
# quarterly Fuel_Price box plot
ndat1%>%group_by(year,quarter)%>%filter(Store==1)%>%
  ggplot(aes(quarter,Fuel_Price,color=year))+geom_boxplot()+
  labs(title = "Plot of quaterly Fuel_Price", x="month", y="Fuel_Price",
       caption="Walmart Stores Sales")
```

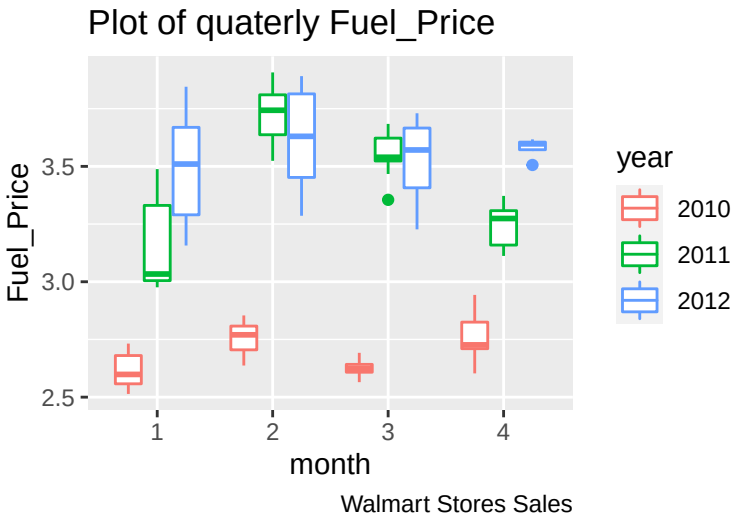

Figure 16: Count of Fuel Price

3.1 Data Analysis

First of all, for simplicity, I create a new dataset called ***ndat2*** by choosing only the necessary parameters for data analysis in the chunk of code below.

```
  # create data only for data analysis
ndat2<-ndat1%>%filter(Store==1)%>%select(Weekly_Sales,Fuel_Price,CPI,Unemployment)
```

Then, let us partition ***ndat2*** to two parts: the training set, namely ***train***, and the validation set, ***test*** statistically half to half. Here train set has *71* samples and test set has *72*. After this, we can start our modeling study.

```
# partition
set.seed(1, sample.kind = "Rounding")    # if using R 3.6 or later
test_index <- createDataPartition(ndat2$Weekly_Sales, times = 1, p = 0.5, list = FALSE)
test <- ndat2[test_index,]
train <- ndat2[-test_index,]
list(n_train=nrow(train), n_test=nrow(test))
```

```
$n_train
[1] 71

$n_test
[1] 72
```

3.1.1 Linear Models

First, let us start with a series of linear models. We can setup the models based on the following equation:

$$Y_{c,u,f} = \mu + b_c + b_u + b_f + \epsilon_{c,u,f}$$

where $Y_{c,u,f}$ denotes the weekly sales; ϵ denotes the independent errors; μ is the *mean* of all *Weekly_Sales* of the dataset; b_c, b_u and b_f represent the bias for specific *CPI*, *Unemployment* and *Fuel Price* respectively. I test the equation step by step by starting from single variable model, μ.

3.1.1.1 Model *1*: μ only

Let us consider the simplest model only using μ, then we run the code below by saving the **RMSE**. The figure next shows the data mismatch between the test set and the prediction ordered by their vector index (x-axis). Since the it is a single value, μ, the prediction just shows a horizontal line (blue).

```
# model 1: mu only
mu <- mean(train$Weekly_Sales)
y_hat <- mu
rmse_1 <- RMSE(y_hat,test$Weekly_Sales)
rmse_1
```

```
[1] 188532
```

```
rmse_results <- data_frame(method = "mu Only",model=1, RMSE = rmse_1)
rmse_results%>% knitr::kable()%>%kable_styling(latex_options = c("striped","hold_position"))
```

method	model	RMSE
mu Only	1	188532

```
# data mismatch
plot(test$Weekly_Sales,col="red")
abline(h=y_hat,col="blue",lwd=2)
legend(1, 2300000, legend=c("Input", "Predicted"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```

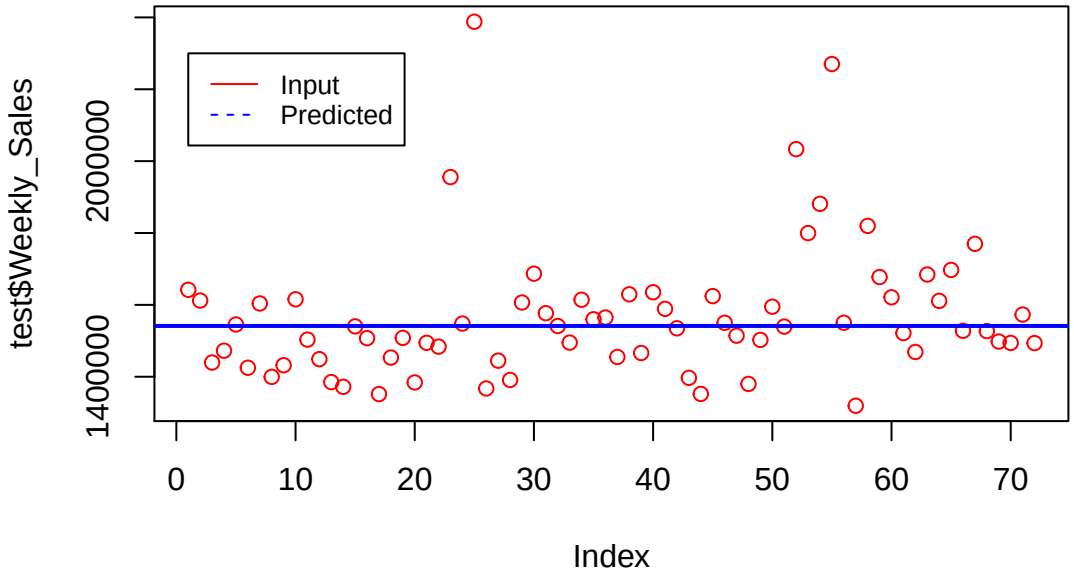

Figure 17: Model 1 data mismatch

3.1.1.2 Model *2*: μ + CPI

Let us add in *CPI* factor while leaving μ unchanged. In this paper, I will use the linear regression module ***lm*** from **caret** package, since it can provide much useful regression information. In the following figure, the prediction line starts showing some features other than a straight as Model 1, while the **RMSE** values also has decreased. Therefore, Model *2* better fits than Model *1* does.

```
# model 2: mu + CPI
model="lm"
fit<-train( Weekly_Sales~CPI, method = model, data = train)
y_hat<-predict(fit, test)
rmse_2 <- RMSE(y_hat, test$Weekly_Sales)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "mu + CPI",model=2,
                                     RMSE = rmse_2))
rmse_results%>% knitr::kable()%>%kable_styling(latex_options = c("striped","hold_position"))
```

method	model	RMSE
mu Only	1	188532
mu + CPI	2	185839

```
# data mismatch
plot(test$Weekly_Sales,col="red")
lines(y_hat,col="blue",lwd=2)
points(y_hat,col="blue",cex = .5)
legend(1, 2300000, legend=c("Input", "Predicted"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```

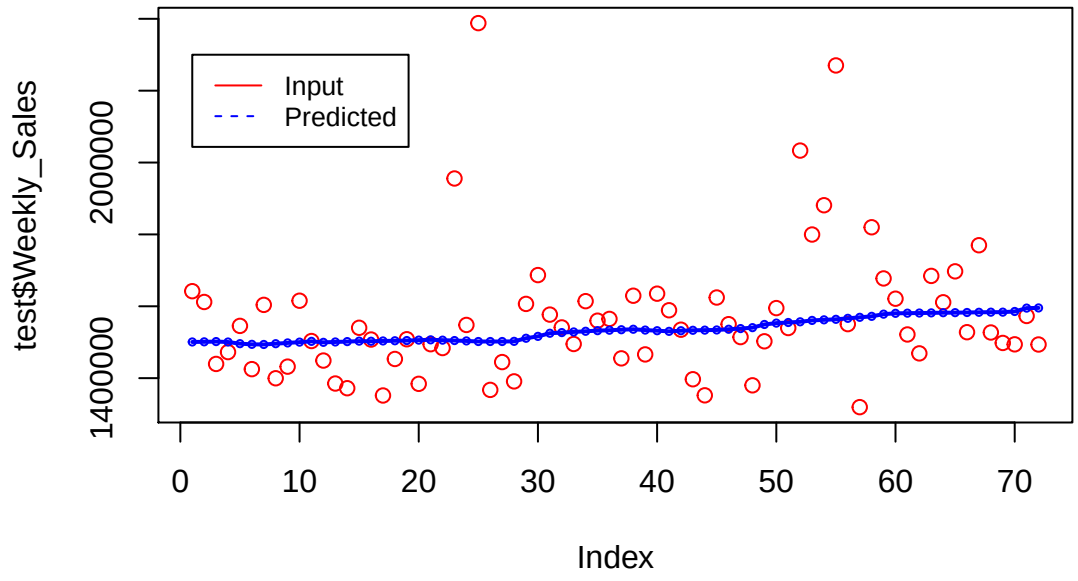

Figure 18: Model 2 data mismatch

3.1.1.3 Model *3*: μ + CPI + Unemployment

Adding in *Unemployment* factor, re-run the linear regression, the result is shown below. Unfortunately, the **RMSE** here does not reduce. On the contrary, it becomes higher. Also, the prediction line does not change much as shown in the figure next.

```
## model 3: mu + CPI + Unemployment
model="lm"
fit<-train( Weekly_Sales~CPI+Unemployment, method = model, data = train)
y_hat<-predict(fit, test)
rmse_3 <- RMSE(y_hat, test$Weekly_Sales)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "mu + CPI + Unemployment",model=3,
```

```
                                    RMSE = rmse_3))
rmse_results%>% knitr::kable()%>%kable_styling(latex_options = c("striped","hold_position"))
```

method	model	RMSE
mu Only	1	188532
mu + CPI	2	185839
mu + CPI + Unemployment	3	186542

```
# data mismatch
plot(test$Weekly_Sales,col="red")
lines(y_hat,col="blue",lwd=2)
points(y_hat,col="blue",cex = .5)
legend(1, 2300000, legend=c("Input", "Predicted"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```

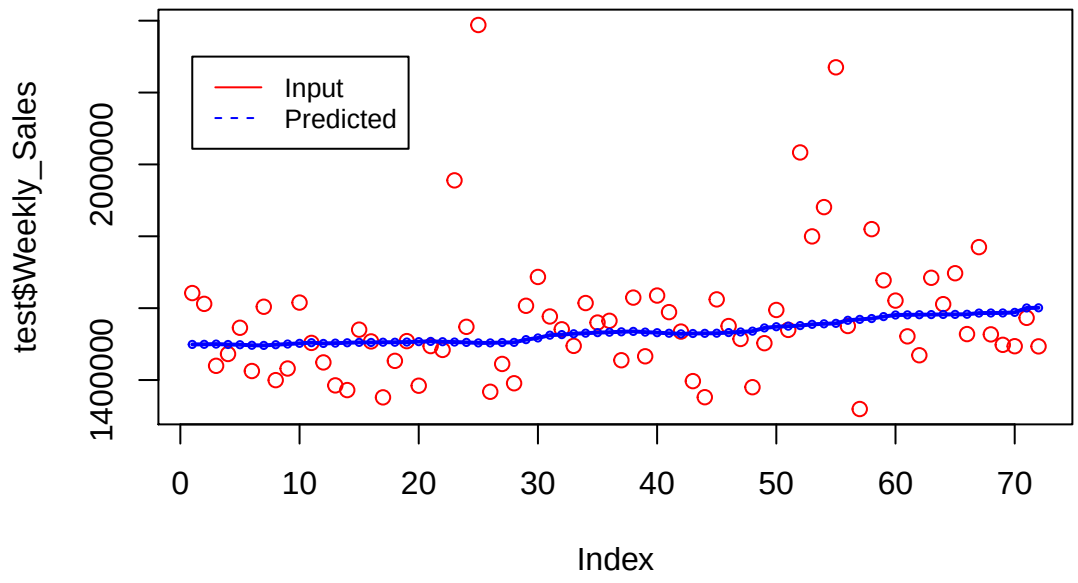

Figure 19: Model 3 data mismatch

3.1.1.4 Model 4: μ + CPI + Unemployment + Fuel Price

Finally, let us consider the *Fuel Price* factor. As shown in the following table, the **RMSE** increases further, while the prediction line in the mismatch figure does not change much either.

```
# model 4: mu + CPI + Unemployment + Fuel_Price
model="lm"
fit<-train( Weekly_Sales~CPI+Unemployment+Fuel_Price, method = model, data = train)
```

```
y_hat<-predict(fit, test)
rmse_4 <- RMSE(y_hat, test$Weekly_Sales)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "mu + CPI + Unemployment + Fuel_Price",model=4,
                                     RMSE = rmse_4))
rmse_results%>% knitr::kable()%>%kable_styling(latex_options = c("striped","hold_position"),
                                               position = "center")
```

method	model	RMSE
mu Only	1	188532
mu + CPI	2	185839
mu + CPI + Unemployment	3	186542
mu + CPI + Unemployment + Fuel_Price	4	187236

```
# data mismatch
plot(test$Weekly_Sales,col="red")
lines(y_hat,col="blue",lwd=2)
points(y_hat,col="blue",cex = .5)
legend(1, 2300000, legend=c("Input", "Predicted"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```

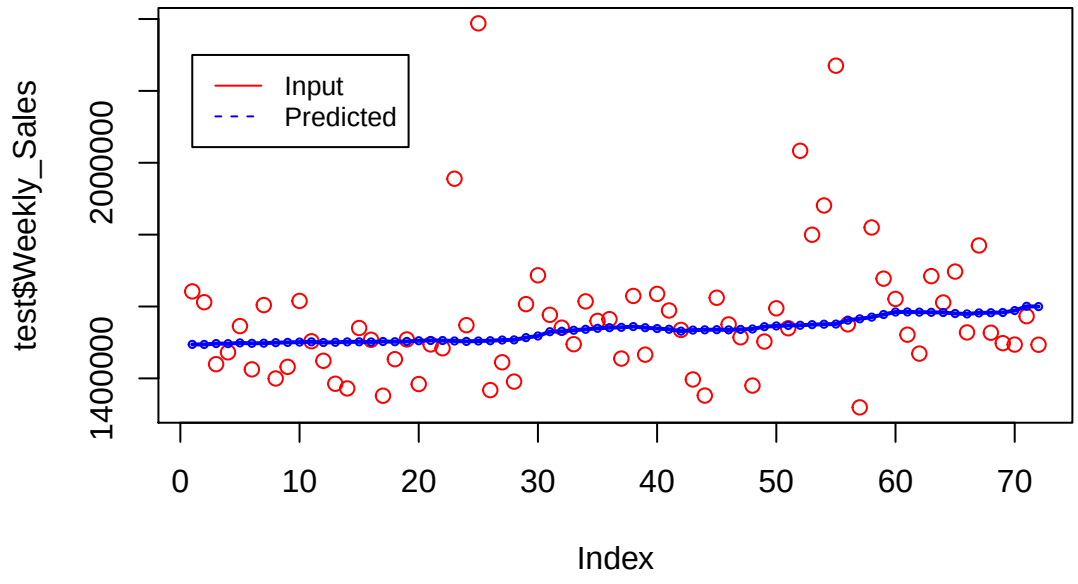

Figure 20: Model 4 data mismatch

As a summary of the linear model tests, the **RMSE** increases with increase of the complexity of the models, such that the linear regression seems cannot properly model our data. Nevertheless, the **importance** function may give us a hint. As shown below, the importance function shows that *CPI* is the dominant

factor here. *Unemployment's* is **0** while *Fuel_Price's* is ~**5**. The latter two factors have extremely limited contribution to the solution. Therefore, we may need look for alternatives algorithms.

```
varImp(fit)
```

```
lm variable importance

             Overall
CPI           100.00
Unemployment    5.42
Fuel_Price      0.00
```

3.2 Nonlinear Models

Since linear modeling has the weakness, I introduce two nonlinear models here. They are ***K-Nearest Neighbors*** and ***Random Forests*** methods, plus cross validation.

3.2.0.1 Model *5*: K-Nearest Neighbors

K-Nearest Neighbors method is well known in data analysis. It can be used on both categorical and continuous variables while our data belongs to continuous category. As shown below, I input the three independent variables, **CPI**, **Unemployment** and **Fuel_Price**, simultaneously. In Regard to the **cross validation**, I tune the fold number k. Furthermore, since we have totally *71* records in the train set, I choose the maximum k as *71*.

Now let us look at the result. As shown in the table below, the **RMSE** has decreased this time. The fitting curve shows the optimal k as *27*. Also the prediction line shows more features than all the linear models before.

```
#model 5: knn:  mu + CPI + Unemployment + Fuel_Price
set.seed(2, sample.kind = "Rounding")
model="knn"
train_control <- trainControl(method="cv", number=5, p=0.9)
fit<-train( Weekly_Sales~CPI+Unemployment+Fuel_Price, method = model, data = train,
            tuneGrid = data.frame(k = seq(1, 71, 2)),trControl=train_control)

y_hat<-predict(fit, test)
rmse_5 <- RMSE(y_hat, test$Weekly_Sales)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "knn: mu + CPI + Unemployment + Fuel_Price",
                                     model=5, RMSE = rmse_5))
rmse_results%>% knitr::kable()%>%kable_styling(latex_options = c("striped","hold_position"),
                                               position = "center")
```

method	model	RMSE
mu Only	1	188532
mu + CPI	2	185839
mu + CPI + Unemployment	3	186542
mu + CPI + Unemployment + Fuel_Price	4	187236
knn: mu + CPI + Unemployment + Fuel_Price	5	183818

```
#minimum k
fit$results$k[which.min(fit$results$RMSE)]
```

```
[1] 27
```

```
ggplot(fit)
```

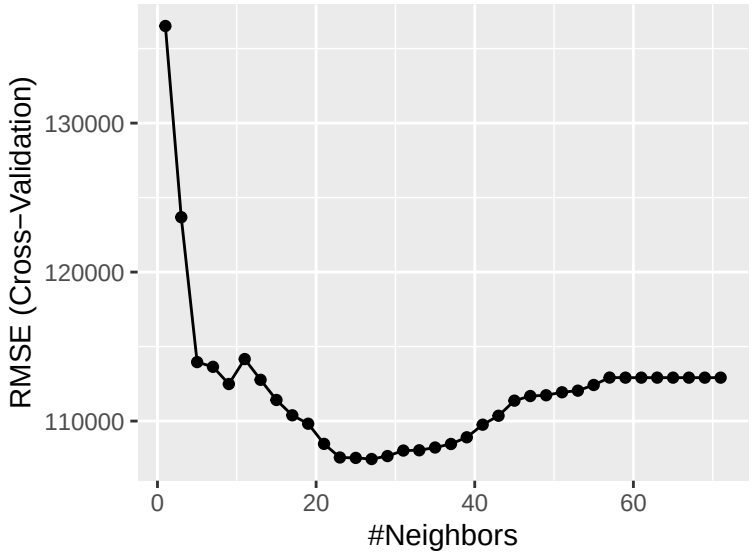

Figure 21: knn cross validation curve

```
plot(test$Weekly_Sales,col="red")
lines(y_hat,col="blue",lwd=2)
points(y_hat,col="blue",cex = .5)
legend(1, 2300000, legend=c("Input", "Predicted"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```

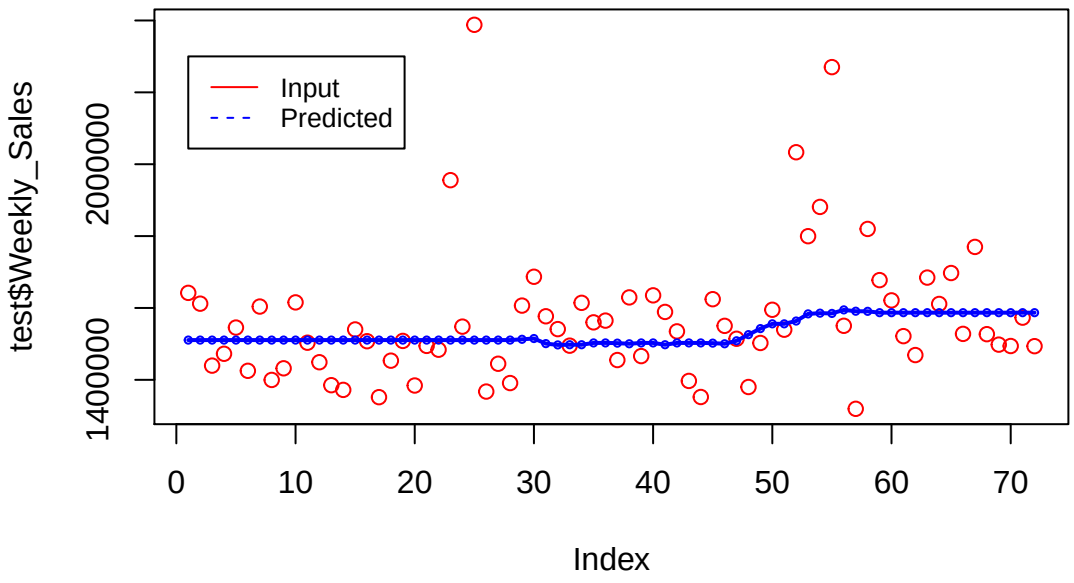

Figure 22: Model 5 data mismatch

3.2.0.2 Model *6*: Random Forests

Let us try another nonlinear method, ***Random Forests***. It also works for categorical and continuous variables. Same as Model 5, the input also contains all the three independent variables together. In regards to **cross validation**, I tune parameter, *mtry*, ranging from *1* to *7*.

As shown in the table below, the **RMSE** has been reduced further than that of Model 5. The fitting curve also shows the optimal *mtry* as *1*. Looking at the mismatch plot below, the prediction has much more features than all the models above. The result from the importance function below also provides further insight. It shows that *CPI* is still the most dominant factor, but the weight of the *Fuel_Price* goes up dramatically from *0* to *87.5* here. The *Unemployment's* reduces to *0*. Resultantly, the **Random Forests**, thus, has produced the best fitting model in this study.

```
set.seed(3, sample.kind = "Rounding")
model="rf"
control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")
fit<-train( Weekly_Sales~CPI+Unemployment+Fuel_Price, method = model,
            data = train, tuneGrid = data.frame(mtry = seq(1,70,10)),
            ntree=100, trControl=control)
y_hat<-predict(fit, test)
rmse_6 <- RMSE(y_hat, test$Weekly_Sales)
rmse_results <- bind_rows(rmse_results,
                          data_frame(method = "rf: mu + CPI + Unemployment + Fuel_Price",
                                     model=6, RMSE = rmse_6))

rmse_results%>% knitr::kable()%>%
  kable_styling(latex_options = c("striped","hold_position"),position = "center")
```

method	model	RMSE
mu Only	1	188532
mu + CPI	2	185839
mu + CPI + Unemployment	3	186542
mu + CPI + Unemployment + Fuel_Price	4	187236
knn: mu + CPI + Unemployment + Fuel_Price	5	183818
rf: mu + CPI + Unemployment + Fuel_Price	6	181146

```
#minimum mtry
fit$results$mtry[which.min(fit$results$RMSE)]
```

```
[1] 1
```

```
plot(fit)
```

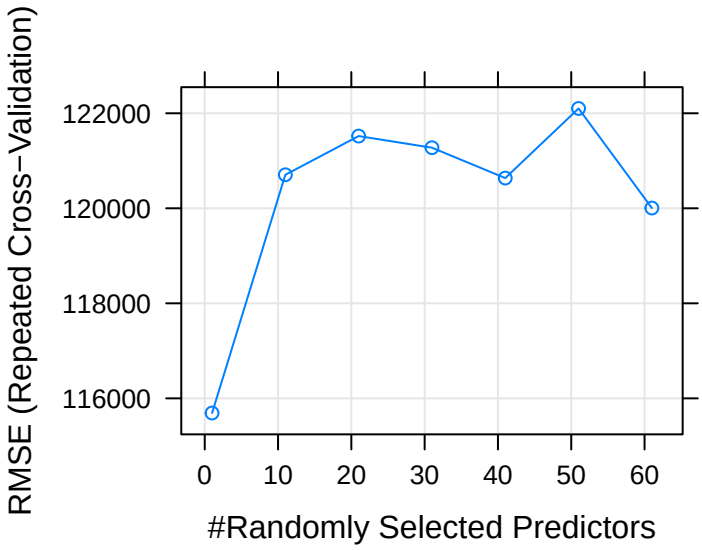

Figure 23: random forests cross validation curve

```
varImp(fit)
```

```
rf variable importance

             Overall
CPI            100.0
Fuel_Price      87.5
Unemployment     0.0
```

```
# data mismatch
plot(test$Weekly_Sales,col="red")
lines(y_hat,col="blue",lwd=2)
points(y_hat,col="blue",cex = .5)
legend(1, 2300000, legend=c("Input", "Predicted"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```

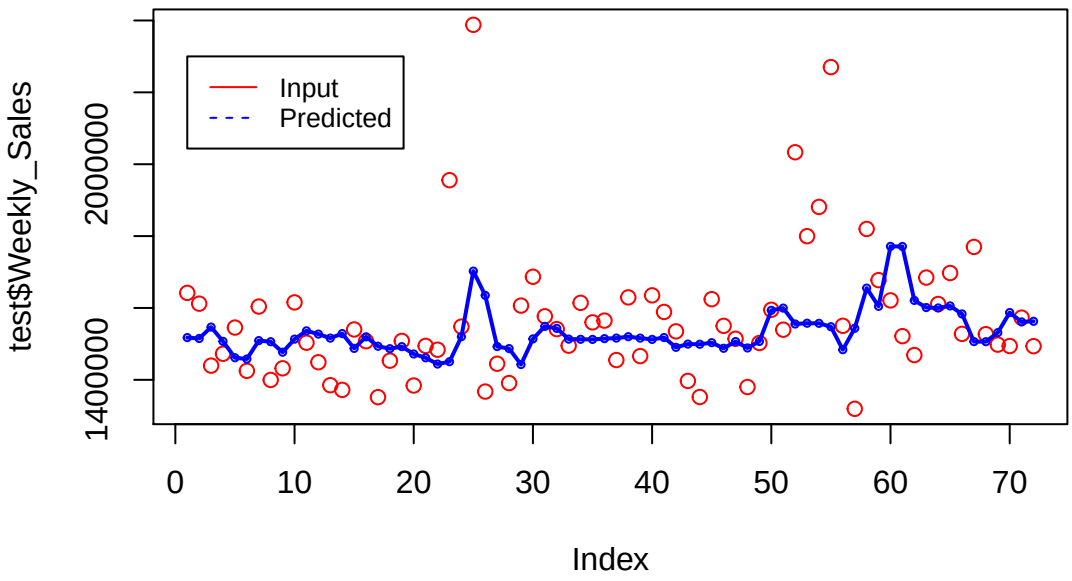

Figure 24: Model 6 data mismatch

For comparison purpose, I plot the **RMSE** of all ***6*** models in a bar graph below. All methods and corresponding models are labeled and legended. The **RMSEs** of the nonlinear models are far smaller than the linear's. Therefore, there is probably a nonlinear relationship between the **Walmart Weekly sales**, and the input parameters: **CPI**, **Fuel_Price** and **Unemployment**.

```
## prepare for final plot
rmse_results$method <- reorder(rmse_results$method, rmse_results$model)

############### final results plot
rmse_results %>%
  ggplot(aes(model, RMSE, fill = method)) +
  geom_bar(stat = 'identity') +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  coord_cartesian(ylim = c(180000, 190000)) +
  guides(fill = guide_legend(reverse = TRUE)) +
  labs(title = "6 moldels' RMSE for weekly store sales " , x = "model", caption = "")
```

4 Conclusion

In conclusion, in this project, I have completed two types of tasks initially assigned by the author on kaggle.com. First, the **Basic statistics tasks** tasks were successfully accomplished in section **2.2** with clear conclusions.

Second, I have tried ***six*** prediction models in section **3** using both **linear** and **nonlinear** approaches to address **Statistical Model** task. The linear model did not work well here. With increased predictors,

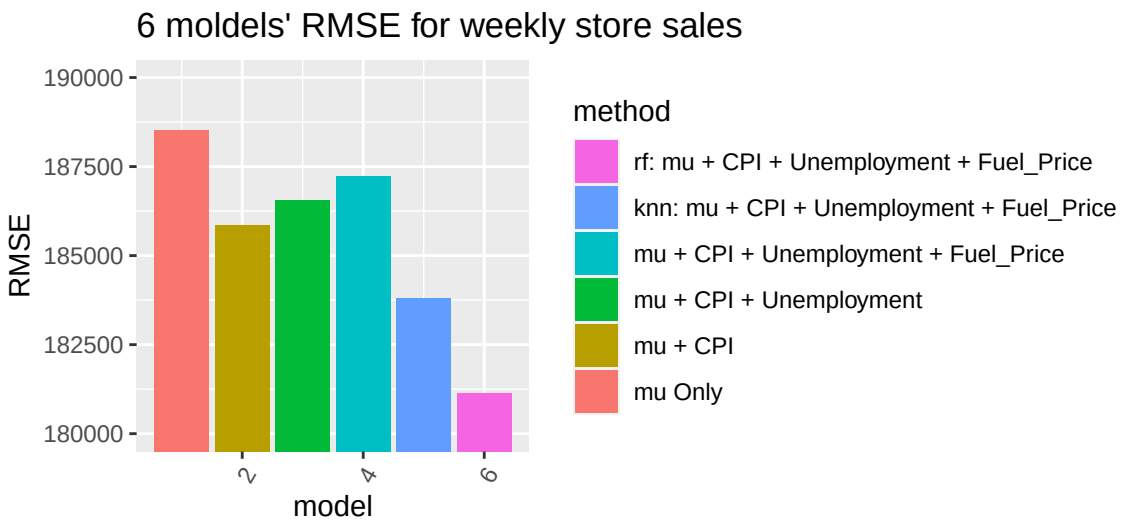

Figure 25: RMSE of six models

the **RMSE** did not decrease accordingly. Rather it has been increasing. As alternatives, I tried two more nonlinear models, **K-Nearest Neighbors** and **Random Forests**, with **cross validation**. It turns out that the nonlinear Models can produce much better fits. Their **RMSEs** are smaller than the linear models. Resultantly, **Random Forests** becomes our best method by a **RMSE** of 1.811×10^5.

Moreover, I also find that the most important factor is **CPI**. The dominance of the **Fuel_Price** increased after adopting nonlinear method. However, the importance of the **Unemployment** is *0* or extremely low for all methods. Thus, we may conclude that **Unemployment** has little impact to the **weekly sales**, whereas **CPI** and **Fuel_Price** do, weighted by their dominance.

Furthermore, the data mismatch plot of our best model, Model 6, still shows that there are big mismatch between the data and prediction. Therefore, there must be a big room to improve our modeling methods and more study is necessary. For instance, we maybe introduce more input factors, like, Holiday, and try more advanced algorithms, etc.

Finally, this store sales dataset provides lot information and we only modeled the data from Store 1. The other stores' data may give us more insights about the in-store sales. In the future, we may look into them also.

5 Reference

- https://courses.edx.org/courses/course-v1:HarvardX+PH125.8x+2T2020/course/
- https://www.kaggle.com/aditya6196/retail-analysis-with-walmart-data ‘