Walmart Sales Prediction in R

Code ▾

In this notebook, we will conduct an exploratory data analysis and linear regression with R using the Walmart sales data set from this Kaggle link (https://www.kaggle.com/datasets/yasserh/walmart-dataset?select=Walmart.csv). For that let's load the important libraries for data analysis. Here we will use *pacman* package for managing add on packages. If the packages already exist, it will load them, otherwise it will download and load the packages.

Hide

```
#use require() or library() to load the base packages
require(pacman) # gives a confirmation message
library(pacman) # load the package, but no confirmation message
```

Hide

```
# We can load all these packages at at time which are commonly used
pacman::p_load(pacman, dplyr, GGally, ggplot2, ggthemes,
  ggvis, httr, lubridate, plotly, rio, rmarkdown, shiny,
  stringr, tidyr)
# you can install the packages independently via " install.packages("package_name")
```

Now let's read in the Walmart dataset and conduct some exploratory data analysis and visualizations. We will utilize the *import* function from rio library to import files like csv, xlsx, txt, etc. Other wise we need to use specific functions like read.csv, read.table, etc.

Hide

```
data1 <- import('Walmart.csv') # specify the path location

# Alternatively we could also use the read.csv(filepath, header = True) option
#data1 = read.csv('Walmart.csv', header = TRUE)
```

Hide

```
# disaplay the first 20 entries of the data
head(data1,20)
```

	Store <int>	Date <chr>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>	Unemployment <dbl>
1	1	05-02-2010	1643691	0	42.31	2.572	211.0964	8.106
2	1	12-02-2010	1641957	1	38.51	2.548	211.2422	8.106
3	1	19-02-2010	1611968	0	39.93	2.514	211.2891	8.106
4	1	26-02-2010	1409728	0	46.63	2.561	211.3196	8.106
5	1	05-03-2010	1554807	0	46.50	2.625	211.3501	8.106
6	1	12-03-2010	1439542	0	57.79	2.667	211.3806	8.106
7	1	19-03-2010	1472516	0	54.58	2.720	211.2156	8.106
8	1	26-03-2010	1404430	0	51.45	2.732	211.0180	8.106
9	1	02-04-2010	1594968	0	62.27	2.719	210.8204	7.808
10	1	09-04-2010	1545419	0	65.86	2.770	210.6229	7.808

1-10 of 20 rows Previous **1** 2 Next

Hide

```
# dimension of the dataset
dim(data1)
```

```
[1] 6435    8
```

The dataset has 6435 rows and 8 columns which correspond to the following attribute

- 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
- Holiday Events
 Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
 Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
 Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
 Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

````
```r
summary(data1)
```
````

````
```
     Store        Date            Weekly_Sales
 Min.   : 1   Length:6435        Min.   : 209986
 1st Qu.:12   Class :character   1st Qu.: 553350
 Median :23   Mode  :character   Median : 960746
 Mean   :23                      Mean   :1046965
 3rd Qu.:34                      3rd Qu.:1420159
 Max.   :45                      Max.   :3818686
  Holiday_Flag      Temperature       Fuel_Price
 Min.   :0.00000   Min.   : -2.06   Min.   :2.472
 1st Qu.:0.00000   1st Qu.: 47.46   1st Qu.:2.933
 Median :0.00000   Median : 62.67   Median :3.445
 Mean   :0.06993   Mean   : 60.66   Mean   :3.359
 3rd Qu.:0.00000   3rd Qu.: 74.94   3rd Qu.:3.735
 Max.   :1.00000   Max.   :100.14   Max.   :4.468
      CPI         Unemployment
 Min.   :126.1   Min.   : 3.879
 1st Qu.:131.7   1st Qu.: 6.891
 Median :182.6   Median : 7.874
 Mean   :171.6   Mean   : 7.999
 3rd Qu.:212.7   3rd Qu.: 8.622
 Max.   :227.2   Max.   :14.313
```
````

Since it it is little bit cluttered, let’s take a look at the weekly sales column.

Hide

```
summary(data1$Weekly_Sales)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 209986  553350  960746 1046965 1420159 3818686
```

Hide

```
# Let's get the unique store values
unique(data1$Store)
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
[17] 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
[33] 33 34 35 36 37 38 39 40 41 42 43 44 45
```

So there are 45 Walmart stores in this data set. We need to aggregate the data by store number and add the weekly sales to see if certain stores have more sales compared to others. In order to do this, we will utilize the *group_by* function from dplyr library. Let's group the data by store number and store the sum of weekly sales into another data frame, gdf.

'%>%' is used to pipe different functions in R.

Hide

```
gdf <- data1 %>% group_by(data1$Store) %>%
       summarise(Total_sales = sum(Weekly_Sales))
gdf
```

data1$Store <int>	Total_sales <dbl>
1	222402809
2	275382441
3	57586735
4	299543953
5	45475689
6	223756131
7	81598275
8	129951181
9	77789219
10	271617714

1-10 of 45 rows   Previous **1** 2 3 4 5 Next

Hide

```
# plot the sales as a function of store number
plot(gdf, col = 'blue', type = 'h', pch = 19, main = "Total Sales", xlab = "Store Number", ylab= "Sales")
```

As we can see, some of the stores have higher cumulative sales compared to others and this could be a regional factor as well. Now let’s see how the sales change as a function of date for a single store, e.g. store 1. For this we will use the *plot_ly* tool in the plotly library.

Hide

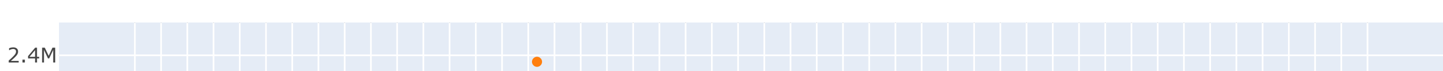
```
#plot the sales as a function of the date as well
fig <- plot_ly(data1, type = 'scatter', mode = 'markers')%>%
  add_trace(x = data1$Date[data1$Store == 1], y = data1$Weekly_Sales[data1$Store == 1])%>%
  layout(showlegend = F)
fig <- fig %>%
  layout(
         xaxis = list(zerolinecolor = '#ffff',
                      zerolinewidth = 2,
                      gridcolor = 'ffff'),
         yaxis = list(zerolinecolor = '#ffff',
                      zerolinewidth = 2,
                      gridcolor = 'ffff'),
         plot_bgcolor='#e5ecf6', width = 900)
```

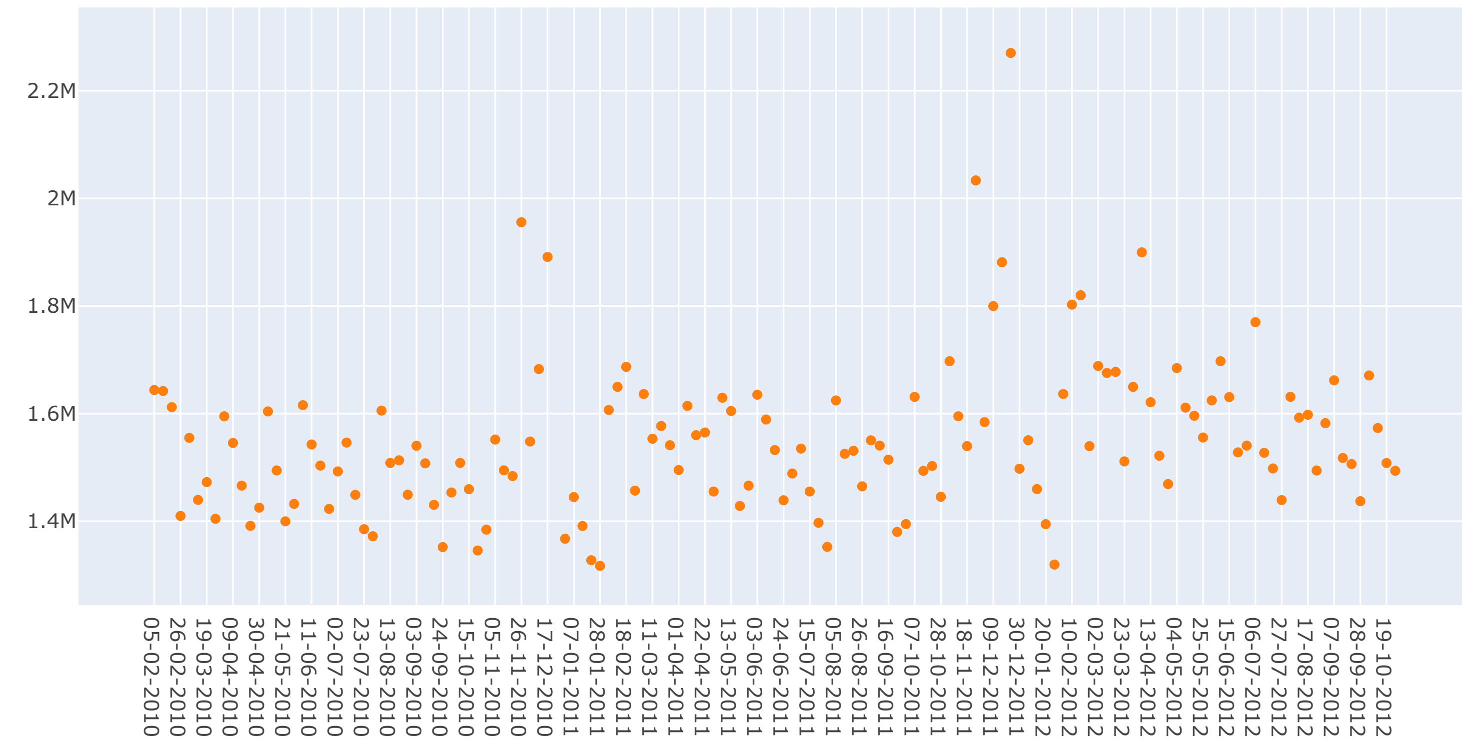
```
Warning: Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

Hide

```
fig
```

```
Warning: Can't display both discrete & non-discrete data on same axisWarning: Can't display both discrete & no
n-discrete data on same axis
```





Interesting there is a spike in the weeky sales during the time between Thanksgiving and Christmas in 2010 and 2011. For that we will group the data by date. Let's plot the same for all stores here.

Hide

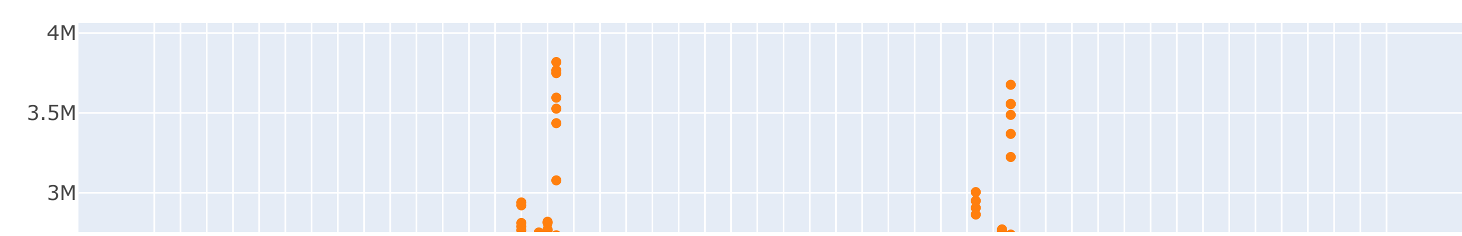
```
#plot the sales as a function of the date as well
fig <- plot_ly(data1, type = 'scatter', mode = 'markers')%>%
  add_trace(x = data1$Date, y = data1$Weekly_Sales)%>%
  layout(showlegend = F)
fig <- fig %>%
  layout(
         xaxis = list(zerolinecolor = '#ffff',
                      zerolinewidth = 2,
                      gridcolor = 'ffff'),
         yaxis = list(zerolinecolor = '#ffff',
                      zerolinewidth = 2,
                      gridcolor = 'ffff'),
         plot_bgcolor='#e5ecf6', width = 900)
```

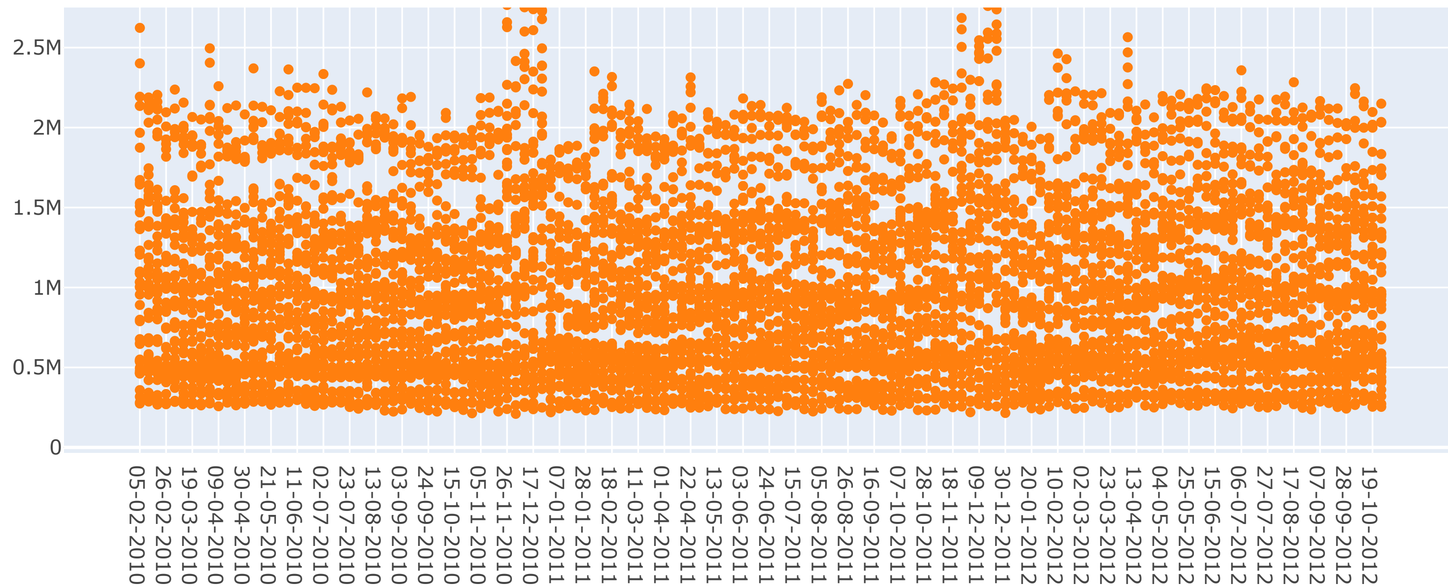
```
Warning: Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

Hide

```
fig
```

```
Warning: Can't display both discrete & non-discrete data on same axisWarning: Can't display both discrete & no
n-discrete data on same axis
```





If we look at the holiday events,

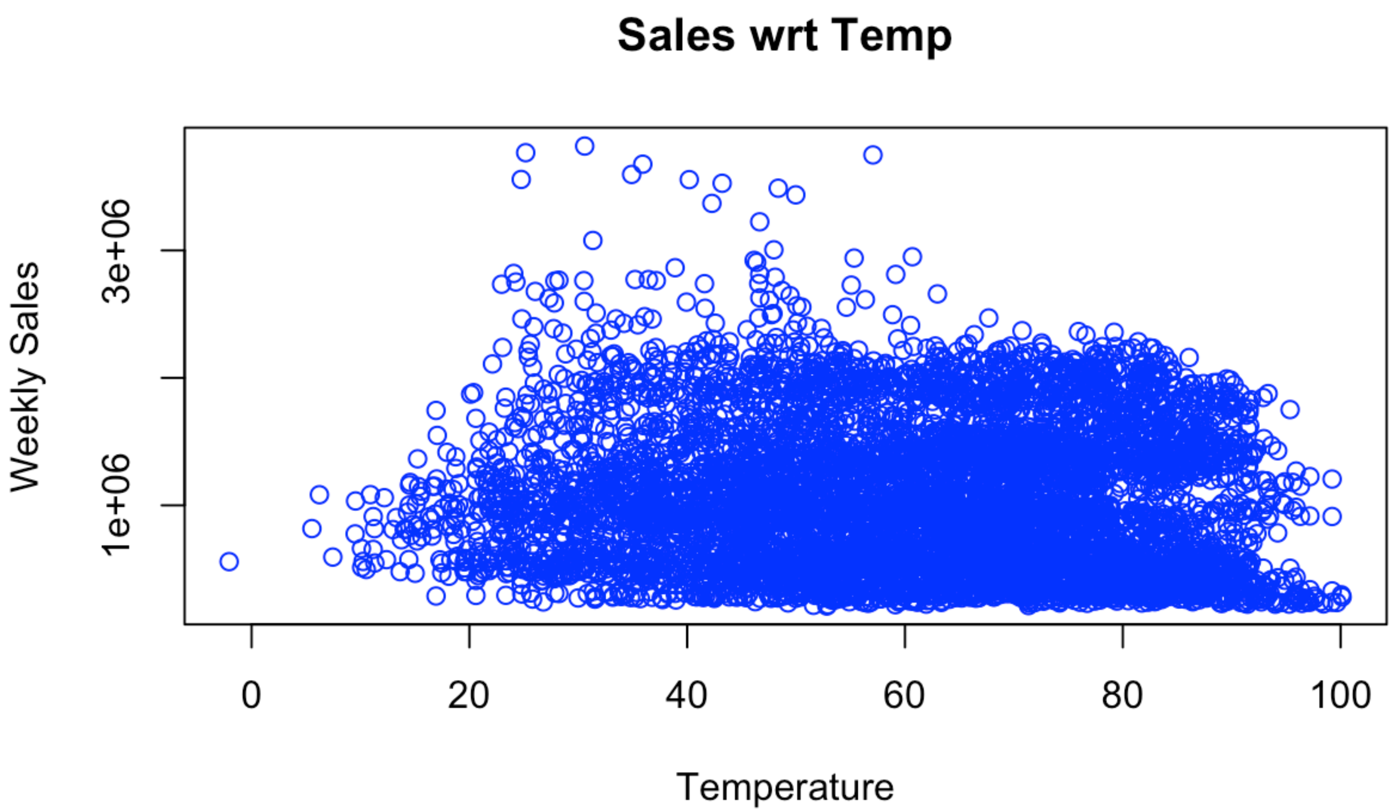
- Super Bowl: 12-Feb-10, 11-Feb-11, 10-Feb-12, 8-Feb-13
- Labour Day: 10-Sep-10, 9-Sep-11, 7-Sep-12, 6-Sep-13
- Thanksgiving: 26-Nov-10, 25-Nov-11, 23-Nov-12, 29-Nov-13
- Christmas: 31-Dec-10, 30-Dec-11, 28-Dec-12, 27-Dec-13

We can clearly see an increase in Sales during the holiday season and it always reaches a peak during the time between Thanksgiving and Christmas.

Let’s make a scatter plot of Weekly sales and temperature.

Hide

```
plot( data1$Temperature, data1$Weekly_Sales, col = 'blue', main = 'Sales wrt Temp', ylab = "Weekly Sales", xla
b = "Temperature")
```



The Weekly sales and temperature seems to be not correlate with each other. Let’ make do some more plotting in subplots format to look for correlations using the plotly library

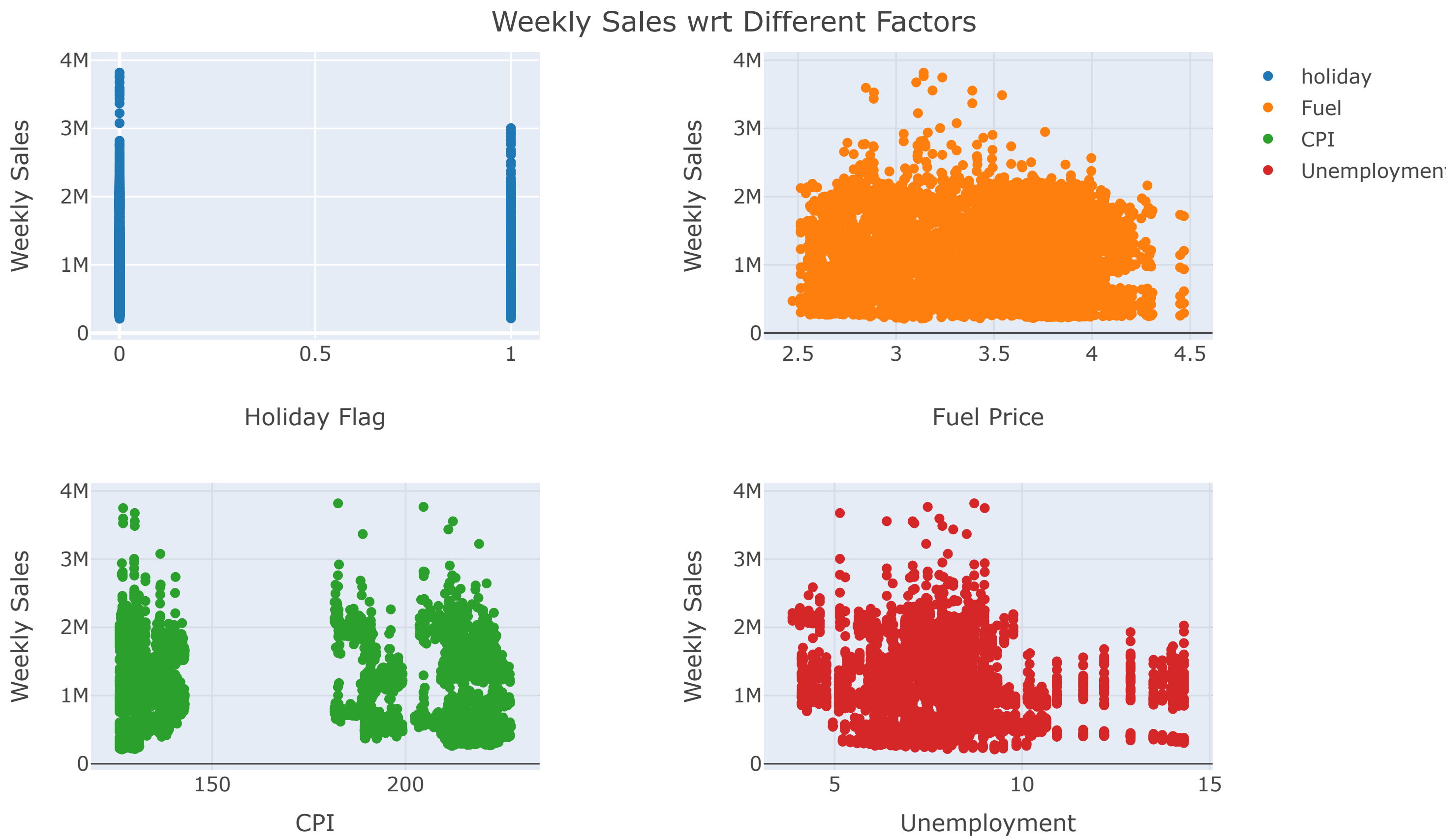
Hide

```
#Initialize figures
fig1 <- plot_ly(x = data1$Holiday_Flag, y = data1$Weekly_Sales, type = 'scatter', name = 'holiday', mode = 'ma
rkers') %>%
  layout(xaxis = list(title = 'Holiday Flag'), yaxis = list(title = 'Weekly Sales'))

fig2 <- plot_ly(x = data1$Fuel_Price, y = data1$Weekly_Sales, type = 'scatter', name = 'Fuel', mode = 'markers
') %>%
  layout(xaxis = list(title = 'Fuel Price'), yaxis = list(title = 'Weekly Sales'))

fig3 <- plot_ly(x = data1$CPI, y = data1$Weekly_Sales, type = 'scatter', name = 'CPI',  mode = 'markers') %>%
  layout(xaxis = list(title = 'CPI'), yaxis = list(title = 'Weekly Sales'))

fig4 <- plot_ly(x = data1$Unemployment, y = data1$Weekly_Sales, type = 'scatter', name = 'Unemployment', mode
= 'markers') %>%
  layout(xaxis = list(title = 'Unemployment'), yaxis = list(title = 'Weekly Sales'))

#creating subplot
fig <- subplot(fig1, fig2, fig3, fig4, nrows = 2, titleY = TRUE, titleX = TRUE, margin = 0.1 )
fig <- fig %>%layout(title = 'Weekly Sales wrt Different Factors',
                     plot_bgcolor='#e5ecf6',
         xaxis = list(
           zerolinecolor = '#ffff',
           zerolinewidth = 2,
           gridcolor = 'ffff'),
         yaxis = list(
           zerolinecolor = '#ffff',
           zerolinewidth = 2,
           gridcolor = 'ffff'), autosize = F, width = 900, height = 500)
```

```
Warning: Specifying width/height in layout() is now deprecated.
Please specify in ggplotly() or plot_ly()
```

Hide

```
fig
```



As we can see the weekly sales is not directly correlated with the holiday flag, fuel price and CPI. The weekly sales goes down as the unemployment rates go up.

From our primary exploratory data analysis, what we can understand is that the Weekly sales mainly depend on the holiday time and the geographical location/store number in this data set. Also, the Sales are better during lower unemployment index.

Cleaning the data

Let' see if the data has any missing values or Nan values before modeling the data. We will use the *filter* function to filter missing/Nan values and use the *mutate* to replace the bad values.

Hide

```
data1 %>%
  summarise(count = sum(is.na(data1)))
```

count
<int>
0

1 row

This data was taken from Kaggle and does not contain any NA/Nan values. But we could introduce some Nan values and clean the data set.

Hide

```
data1[5,5] <- NA
data1[9,5] <- NaN
head(data1, 10)
```

	Store <int>	Date <chr>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>	Unemployment <dbl>
1	1	05-02-2010	1643691	0	42.31	2.572	211.0964	8.106
2	1	12-02-2010	1641957	1	38.51	2.548	211.2422	8.106
3	1	19-02-2010	1611968	0	39.93	2.514	211.2891	8.106
4	1	26-02-2010	1409728	0	46.63	2.561	211.3196	8.106
5	1	05-03-2010	1554807	0	*NA*	2.625	211.3501	8.106
6	1	12-03-2010	1439542	0	57.79	2.667	211.3806	8.106
7	1	19-03-2010	1472516	0	54.58	2.720	211.2156	8.106
8	1	26-03-2010	1404430	0	51.45	2.732	211.0180	8.106
9	1	02-04-2010	1594968	0	NaN	2.719	210.8204	7.808
10	1	09-04-2010	1545419	0	65.86	2.770	210.6229	7.808

1-10 of 10 rows

Now let's try again for NA/NaN values. *is.na* would check for both NA and NaN values while *is.nan* will only check for NaN values.

Hide

```
data1 %>%
  summarise(count = sum(is.na(data1)))
```

count <int>
2

1 row

Hide

```
#is.nan requires a list of data
data1 %>%
  summarise(count = sum(is.nan(data1$Temperature)))
```

count <int>
1

1 row

Let's replace the NA/NaNs with the median values in the data set.

Hide

```
data1 <- data1 %>%
        mutate(Temperature = replace(Temperature, is.na(Temperature), median(Temperature, na.rm = TRUE)))
head(data1, 10)
```

Store <int>	Date <chr>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>	Unemployment <dbl>

1	1	05-02-2010	1643691	0	42.31	2.572	211.0964	8.106
2	1	12-02-2010	1641957	1	38.51	2.548	211.2422	8.106
3	1	19-02-2010	1611968	0	39.93	2.514	211.2891	8.106
4	1	26-02-2010	1409728	0	46.63	2.561	211.3196	8.106
5	1	05-03-2010	1554807	0	62.68	2.625	211.3501	8.106
6	1	12-03-2010	1439542	0	57.79	2.667	211.3806	8.106
7	1	19-03-2010	1472516	0	54.58	2.720	211.2156	8.106
8	1	26-03-2010	1404430	0	51.45	2.732	211.0180	8.106
9	1	02-04-2010	1594968	0	62.68	2.719	210.8204	7.808
10	1	09-04-2010	1545419	0	65.86	2.770	210.6229	7.808

1-10 of 10 rows

Preprocessing

Before modeling the data, we need to convert the dates into a more meaning full numbers. In our case, rather than converting days into some numbers, we need it as a cyclic variable going from 1-365 as our sales are a function of different time of an year, especially the holiday time. Let’s write a function to do that.

Hide

```
#defining a function to convert the dates into day in a year
date_to_number <- function(dates){
  num_date <- c()
  #print(length(num_date))
  for (i in seq(1:length(dates))){
      date <-  dates[i]
      d <- strtoi(substr(date, 1, 2), 10) # getting the string values and converting to integers, using base 1
0 here.
      m <- strtoi(substr(date, 4, 5), 10)
      y <- strtoi(substr(date, 7, 10), 10)

      num_date <- append(num_date, m*30 + d)

      #cat(i, date, num_date[[i]], "\n")
  }
  return (num_date)
}

new_dates <- date_to_number(data1$Date)
#print(new_dates)

# add the new date numbers to the dataframe
data1 <- data1 %>%
         mutate(date_number = new_dates)
head(data1,6)
```

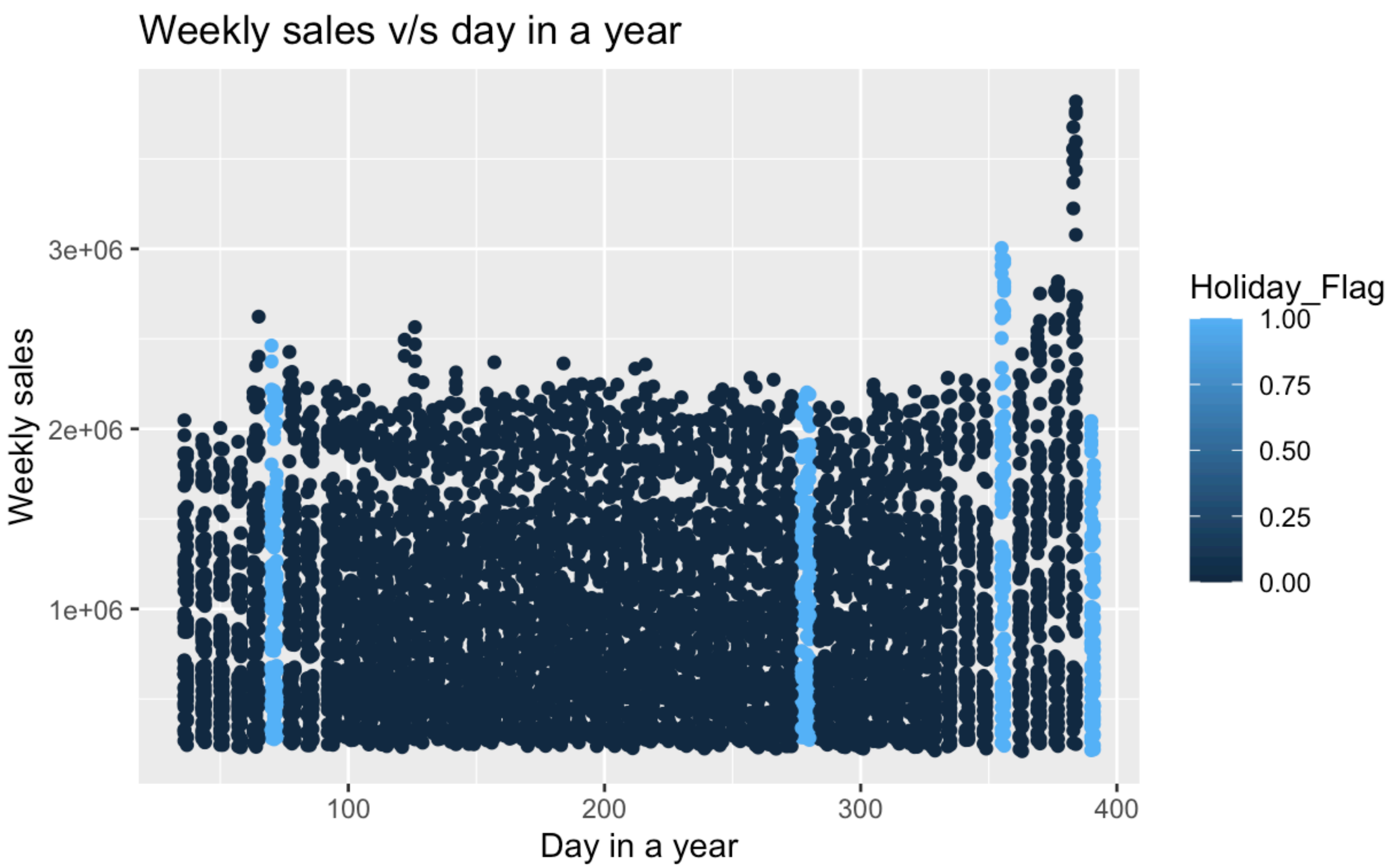
	Store <int>	Date <chr>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>	Unemployment <dbl>
1	1	05-02-2010	1643691	0	42.31	2.572	211.0964	8.106
2	1	12-02-2010	1641957	1	38.51	2.548	211.2422	8.106
3	1	19-02-2010	1611968	0	39.93	2.514	211.2891	8.106
4	1	26-02-2010	1409728	0	46.63	2.561	211.3196	8.106

5	1	05-03-2010	1554807	0	62.68	2.625	211.3501	8.106
6	1	12-03-2010	1439542	0	57.79	2.667	211.3806	8.106

6 rows | 1-9 of 9 columns

Hide

```
# Now let's make a plot using ggplot to plot the sales as a fuction of the new date numbers we created
ggplot(data = data1, mapping = aes(y = Weekly_Sales, x = date_number, color = Holiday_Flag)) + geom_point() +
labs(title = "Weekly sales v/s day in a year", x = "Day in a year", y = "Weekly sales")
```



One interesting thing to note here is that, some of the high sales time between after Thanksgiving and before Christmas has been marked as not a holiday flag which might affect the modeling of the data.

Correlation calculation

Let's build a correlation matrix first using the Pearson correlation coefficient.

Hide

```
data_new <- data1[-2] # removing the dates column
head(data_new)
```

	Store <int>	Weekly_Sales <dbl>	Holiday_Flag <int>	Temperature <dbl>	Fuel_Price <dbl>	CPI <dbl>	Unemployment <dbl>	date_number <dbl>
1	1	1643691	0	42.31	2.572	211.0964	8.106	65
2	1	1641957	1	38.51	2.548	211.2422	8.106	72
3	1	1611968	0	39.93	2.514	211.2891	8.106	79
4	1	1409728	0	46.63	2.561	211.3196	8.106	86

5	1	1554807	0	62.68	2.625	211.3501	8.106	95
6	1	1439542	0	57.79	2.667	211.3806	8.106	102

6 rows

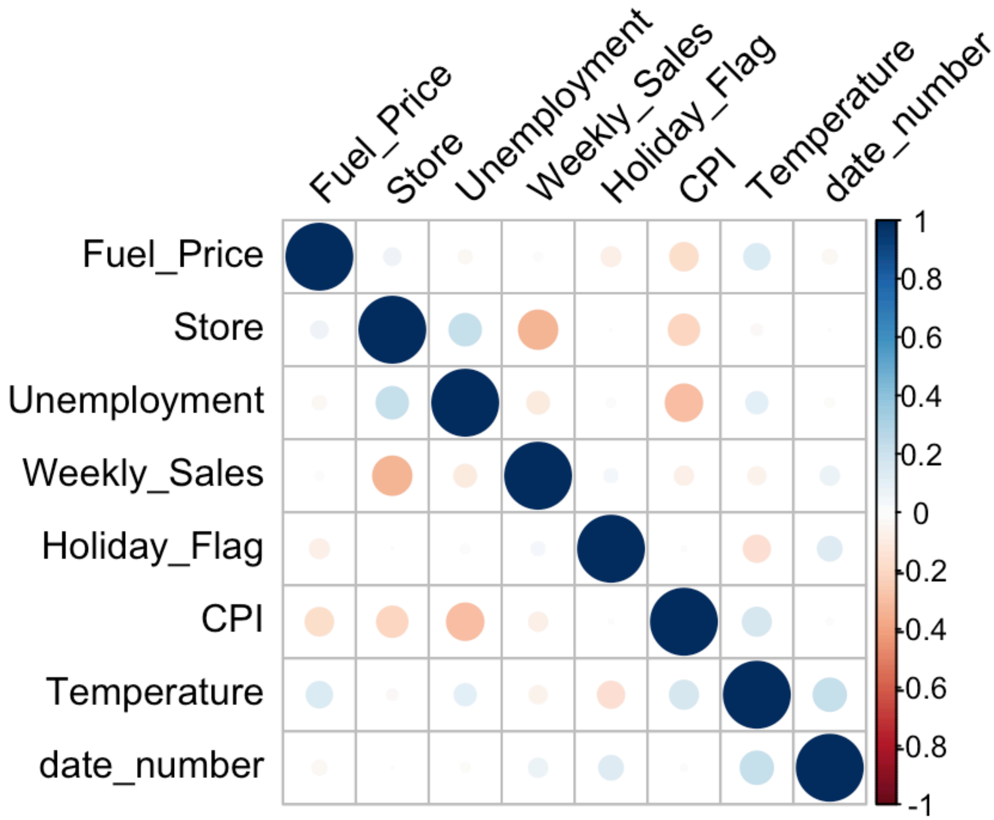
Hide

```
#use the cor function to get the correlation of features in the data frame
res = cor(data_new)
round(res,2)
```

```
             Store Weekly_Sales Holiday_Flag
Store         1.00        -0.34         0.00
Weekly_Sales -0.34         1.00         0.04
Holiday_Flag  0.00         0.04         1.00
Temperature  -0.02        -0.06        -0.16
Fuel_Price    0.06         0.01        -0.08
CPI          -0.21        -0.07         0.00
Unemployment  0.22        -0.11         0.01
date_number   0.00         0.07         0.13
             Temperature Fuel_Price   CPI
Store              -0.02       0.06 -0.21
Weekly_Sales       -0.06       0.01 -0.07
Holiday_Flag       -0.16      -0.08  0.00
Temperature         1.00       0.14  0.18
Fuel_Price          0.14       1.00 -0.17
CPI                 0.18      -0.17  1.00
Unemployment        0.10      -0.03 -0.30
date_number         0.24      -0.04  0.01
             Unemployment date_number
Store                0.22        0.00
Weekly_Sales        -0.11        0.07
Holiday_Flag         0.01        0.13
Temperature          0.10        0.24
Fuel_Price          -0.03       -0.04
CPI                 -0.30        0.01
Unemployment         1.00       -0.01
date_number         -0.01        1.00
```

Hide

```
# Let's import the corrplot library for the visualization of the correlation
library(corrplot)
corrplot(res, type = "full", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



In this correlogram, the radius of the circle represent the correlation strength and the colors represents the positive/negative correlation. As we can see, for the weekly sales has some correlation with the store number and it weakly/not correlated with the rest of features.

Principal component analysis

Before modeling of the data, let's do principal component analysis (PCA) of the data for visualization and understand the correlation within the data set.

Hide

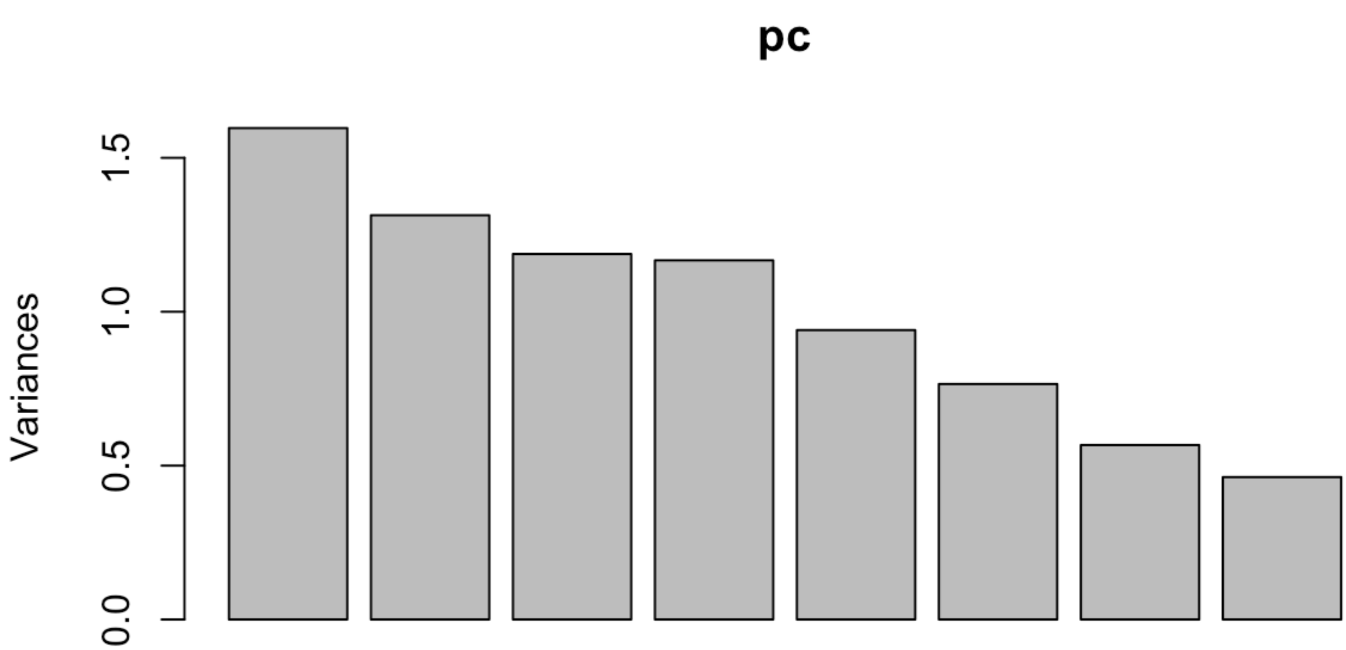
```
# using prcomp function for PCA
pc <- prcomp(data_new,
        center = TRUE,  # Centers means to 0 (optional)
        scale = TRUE)   # Sets unit variance (helpful)

# Get summary stats
summary(pc)
```

```
Importance of components:
                          PC1    PC2    PC3    PC4
Standard deviation     1.2636 1.1461 1.0898 1.0803
Proportion of Variance 0.1996 0.1642 0.1484 0.1459
Cumulative Proportion  0.1996 0.3638 0.5122 0.6581
                          PC5     PC6     PC7
Standard deviation     0.9697 0.87475 0.75293
Proportion of Variance 0.1176 0.09565 0.07086
Cumulative Proportion  0.7756 0.87130 0.94216
                           PC8
Standard deviation     0.68023
Proportion of Variance 0.05784
Cumulative Proportion  1.00000
```

As you can see the variance is mostly spread out and the data is not much correlated.

Hide

```
#Screeplot for number of components
plot(pc)
```



Hide

```
# Get standard deviations and rotation
pc
```

```
Standard deviations (1, .., p=8):
[1] 1.2636174 1.1460754 1.0897825 1.0802661 0.9697365
[6] 0.8747479 0.7529284 0.6802261

Rotation (n x k) = (8 x 8):
                    PC1         PC2        PC3
Store        -0.58302714  0.06282082  0.1876872
Weekly_Sales  0.37061204 -0.24412301 -0.5807988
Holiday_Flag  0.04756473 -0.31022505 -0.1891669
Temperature   0.01940246  0.75361202 -0.1550988
Fuel_Price   -0.16554419  0.21550230 -0.3006047
CPI           0.47216154  0.30498145  0.4688994
Unemployment -0.51150062  0.02703337 -0.2094692
date_number   0.09007179  0.36345716 -0.4620599
                   PC4         PC5         PC6
Store         0.1645521 -0.23624741 -0.32652943
Weekly_Sales -0.1815134  0.21885436 -0.05564072
Holiday_Flag  0.5898542 -0.45713176  0.51900724
Temperature   0.0101879  0.13179102  0.28400046
Fuel_Price   -0.5389580 -0.61367045  0.23289369
CPI           0.1664354 -0.04469587  0.22873143
Unemployment  0.1523670  0.52771281  0.44046373
date_number   0.5005512 -0.11349736 -0.48958352
                    PC7          PC8
Store         0.65674624 -0.008894007
Weekly_Sales  0.61394405 -0.069684476
Holiday_Flag  0.07426895  0.184206619
Temperature   0.18263804  0.525500962
Fuel_Price   -0.05610621 -0.333670674
CPI           0.30464448 -0.537920392
Unemployment -0.03216957 -0.443868529
date_number  -0.23641808 -0.295411396
```

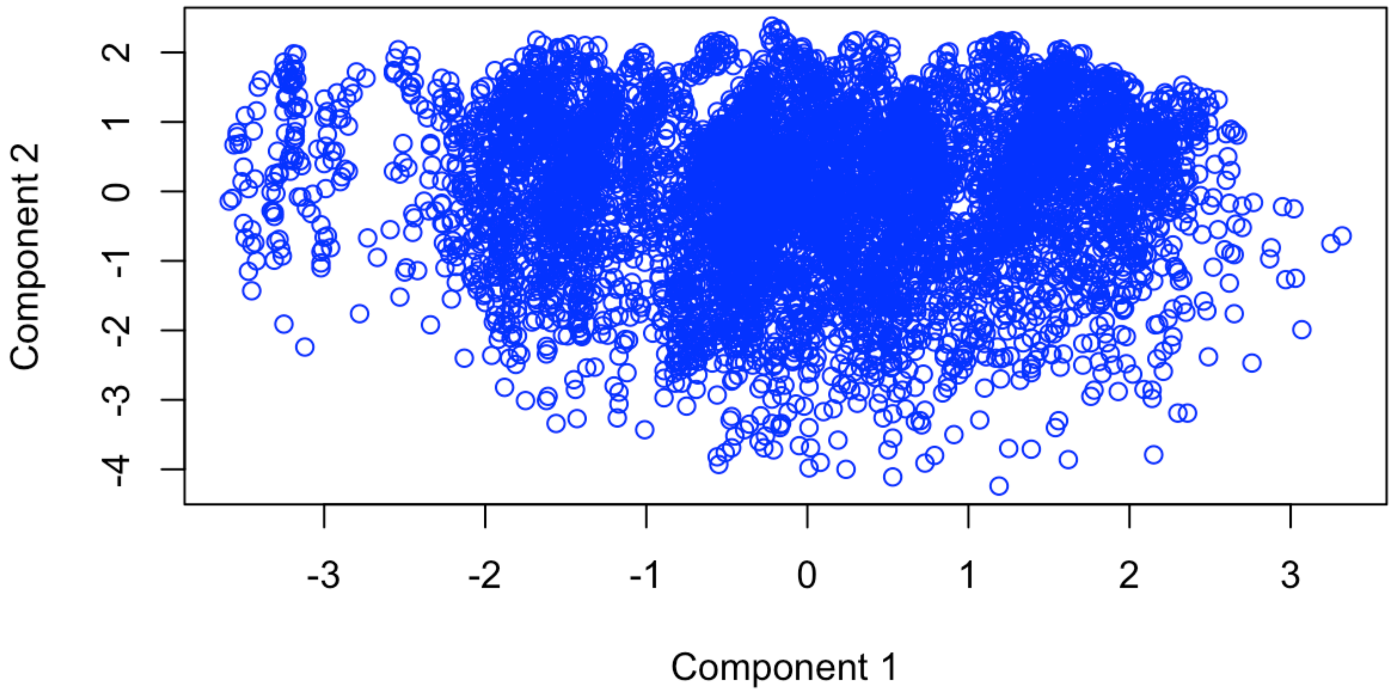
Hide

```
# See how cases load on PCs
pre <- predict(pc) %>% round(2)
dim(pre)
```

```
[1] 6435    8
```

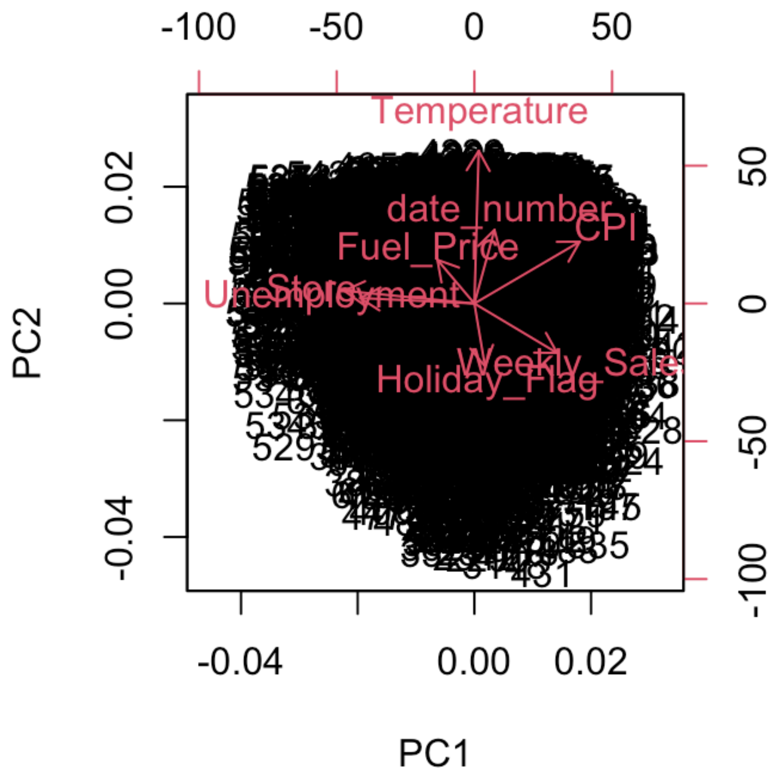
Hide

```
#plotting the first 2 components
plot(pre[,1], pre[,2], xlab = "Component 1", ylab = "Component 2", col="blue")
```



Hide

```
# Biplot of first two components
biplot(pc)
```



As you can see, there the first 2 principal components only explains only 36% of the data and they don't have any linear correlation as well. Here in the biplot, length of vectors denote how much it has contributed to the component and cos(angle between vectors) is proportional to the correlation between them. As you can see, the weekly sales and holiday flag are correlated.

Multivariate linear regression

Now let's do the multivariate modeling of the data. For that let's define the x and y data.

Hide

```
# Let's shuffle the dataset before splitting
data_shuff <- data1[sample(1:nrow(data1)),]

# define x and y values
x = data_shuff[c(-2, -3)]
x <- as.matrix(x)
y <- data_shuff$Weekly_Sales

# let's split the data into test, validatation and test datasets with 70:20:10 ratio
# In total the dataset has 6435 rows
xtrain <- x[1:4504,]
ytrain <- y[1:4504]

xval <- x[4505:5792, ]
yval <- y[4505:5792]

xtest <- x[5793:6435,]
ytest <- y[5793:6435]

head(xval,10)
```

```
     Store Holiday_Flag Temperature Fuel_Price
1158     9            0       68.58      2.835
844      6            0       81.57      3.311
4586    33            0       68.43      3.004
4400    31            0       57.16      3.669
2051    15            0       30.53      3.351
3567    25            0       68.55      3.867
2762    20            0       24.27      3.109
1327    10            0       71.04      3.009
243      2            1       44.57      3.129
1876    14            0       69.27      2.899
          CPI Unemployment date_number
1158 213.8485        6.384         157
844  223.5425        5.668         230
4586 126.6019        9.849         129
4400 220.6974        7.057          99
2051 132.8823        7.771          37
3567 215.0878        7.280         271
2762 204.6877        7.484         370
1327 126.4913        9.003         335
243  219.1773        7.441         390
1876 182.0464        8.899         178
```

Hide

```
# Now let's use a linear model on the test dataset first
reg_test <- lm(ytrain ~ xtrain)

reg_test # print the coefficients only
```

```

Call:
lm(formula = ytrain ~ xtrain)

Coefficients:
       (Intercept)         xtrainStore
         1897418.2            -15635.5
xtrainHoliday_Flag   xtrainTemperature
           59484.0             -1473.6
  xtrainFuel_Price           xtrainCPI
           22003.6             -2398.4
xtrainUnemployment   xtraindate_number
          -21394.5               467.7
```

Hide

```
summary(reg_test)  # Inferential tests
```

```
Call:
lm(formula = ytrain ~ xtrain)

Residuals:
     Min       1Q   Median       3Q      Max 
-1019784  -381002   -50500   375769  2621283 

Coefficients:
                    Estimate Std. Error t value
(Intercept)        1897418.2    92771.9  20.453
xtrainStore         -15635.5      626.7 -24.949
xtrainHoliday_Flag   59484.0    32125.2   1.852
xtrainTemperature    -1473.6      463.8  -3.177
xtrainFuel_Price     22003.6    17607.4   1.250
xtrainCPI            -2398.4      221.6 -10.823
xtrainUnemployment  -21394.5     4618.2  -4.633
xtraindate_number      467.7       84.3   5.548
                   Pr(>|t|)    
(Intercept)         < 2e-16 ***
xtrainStore         < 2e-16 ***
xtrainHoliday_Flag   0.0641 .  
xtrainTemperature    0.0015 ** 
xtrainFuel_Price     0.2115    
xtrainCPI           < 2e-16 ***
xtrainUnemployment 3.71e-06 ***
xtraindate_number  3.05e-08 ***
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 521100 on 4496 degrees of freedom
Multiple R-squared:  0.1501,	Adjusted R-squared:  0.1488 
F-statistic: 113.4 on 7 and 4496 DF,  p-value: < 2.2e-16
```

Let’ look at the actual weekly sales and predicted weekly sales from the training data

Hide

```
pred_ytrain <- predict(reg_test, newdata = as.data.frame(xtrain))

for (i in seq(1:30)){
  str <- sprintf("Actual : %f, predicted :%f \n", ytrain[i], pred_ytrain[i])
  cat(str)
}
```

```
Actual : 465108.520000, predicted :774242.166427
Actual : 826155.950000, predicted :913077.714832
Actual : 2036231.390000, predicted :1470608.032416
Actual : 1931668.640000, predicted :994497.462273
Actual : 1368318.170000, predicted :1387375.413981
Actual : 1227118.750000, predicted :996458.041920
Actual : 2119163.010000, predicted :1022618.700012
Actual : 1532114.860000, predicted :1246604.049566
Actual : 921612.530000, predicted :1177310.521001
Actual : 413042.120000, predicted :1219310.863526
Actual : 1552934.640000, predicted :1186973.702320
Actual : 504760.570000, predicted :857505.456329
Actual : 324801.130000, predicted :708011.076116
Actual : 757330.950000, predicted :604570.921285
Actual : 1462941.030000, predicted :1036702.121735
Actual : 808030.150000, predicted :899068.480968
Actual : 438760.620000, predicted :768973.639183
Actual : 1794962.640000, predicted :1300800.071943
Actual : 1584083.950000, predicted :1355952.404693
Actual : 1015737.610000, predicted :899961.756977
Actual : 375629.510000, predicted :1250521.901853
Actual : 441683.740000, predicted :819556.618904
Actual : 491115.860000, predicted :788535.638231
Actual : 975500.870000, predicted :931461.253843
Actual : 1033719.500000, predicted :905150.040844
Actual : 1686842.780000, predicted :1223643.618865
Actual : 1373270.060000, predicted :1165419.895546
Actual : 549967.890000, predicted :1121636.926376
Actual : 1182733.000000, predicted :926194.424002
Actual : 1460234.310000, predicted :714664.859139
```

The R statistics should be close to 1 and in our case we are getting 0.14. Also the residual error is really high. Maybe our simple multivariate regression model is not good enough for the prediction purpose here which is also evident after looking at the first 30 actual weekly sales and predicted sales. Feature engineering could have been done if some of the features exhibited some non-linear relationship with the Weekly sales.

Polynomial regression

Let’s try the polynomial regression and see how the model performs in the prediction task.

Hide

```
library(tidyverse)
library(caret)

#Build the polynomial model with degree 3
pmod <- lm(ytrain ~ poly(xtrain, 5, raw = TRUE))

# Model summary
# coef(summary(pmod))
```

Hide

```
# Make predictions
ypred_poly_train <- predict(pmod, poly(xtrain, 5, raw = TRUE))

# Model performance
poly_train_metrics = data.frame(
                    RMSE = RMSE(ypred_poly_train, ytrain),
                     R2 = R2(ypred_poly_train, ytrain))

print(poly_train_metrics)
```

RMSE <dbl>	R2 <dbl>
278411.4	0.7569364

1 row

The RMSE of the polynomial regression has gone down and the R2 value increased significantly compared to the linear regression.

Random Forest

Now let's try the widely used random forest algorithm for this regression problem.

Hide

```
library(randomForest)

# Fitting Random Forest to the train dataset
set.seed(120)  # Setting seed
rf = randomForest(x = xtrain, y = ytrain, ntree = 300, samp_size=2000)

rf
```

```


Call:
 randomForest(x = xtrain, y = ytrain, ntree = 300, samp_size = 2000) 
               Type of random forest: regression
                     Number of trees: 300
No. of variables tried at each split: 2

          Mean of squared residuals: 25773225461
                    % Var explained: 91.92
```

Hide

```
print("Predicting the RF train metrics")
```

```
[1] "Predicting the RF train metrics"
```

Hide

```
# Predicting the Test set results
ypred_rf_train = predict(rf, newdata = xtrain)

# Model performance
rf_train_metrics = data.frame(
                    RMSE = RMSE(ypred_rf_train, ytrain),
                     R2 = R2(ypred_rf_train, ytrain))

print(rf_train_metrics)
```

RMSE <dbl>	R2 <dbl>
82533.37	0.9833679

1 row

The RMSE has gone down and R2 for random forest has reached upto 98% which is really good score. Let's see how it generalize it on the validation data.

Hide

```
print("Predicting the RF validation metrics")
```

```
[1] "Predicting the RF validation metrics"
```

Hide

```
# Predicting the Test set results
ypred_rf_val = predict(rf, newdata = xval)

# Model performance
rf_val_metrics = data.frame(
                    RMSE = RMSE(ypred_rf_val, yval),
                     R2 = R2(ypred_rf_val, yval))

print(rf_val_metrics)
```

RMSE <dbl>	R2 <dbl>
165757.2	0.920209

1 row

Hide

```
print("Predicting the RF test  metrics")
```

```
[1] "Predicting the RF test  metrics"
```

Hide

```
# Predicting the Test set results
ypred_rf_test = predict(rf, newdata = xtest)

# Model performance
rf_test_metrics = data.frame(
                    RMSE = RMSE(ypred_rf_test, ytest),
                     R2 = R2(ypred_rf_test, ytest))

print(rf_test_metrics)
```

RMSE <dbl>	R2 <dbl>
155735.9	0.937997

1 row

The RMSE on validation and test datasets are little higher than the training data. The R2 score is around 93 % for both the validation and test data suggesting that the model is generalizing well on unseen data.

Conclusion

From the exploratory analysis, I have found that the Weekly sales Weekly sales is highly correlted with the holiday time and the geographical location/store number in this data set. Also, the Sales are relatively higher during lower unemployment index. For modeling the sales prediction, the random forest model is doing a good job with the predictions on the unseen data.

Hide

```
# How to clear packages
#p_unload(dplyr, tidyr, stringr) # Clear specific packages
p_unload(all)  # Easier: clears all add-ons
```

```
The following packages have been unloaded:
randomForest, caret, lattice, forcats, purrr, readr, tibble, tidyverse, corrplot, tidyr, stringr, shiny, rmar
kdown, rio, plotly, lubridate, httr, ggvis, ggthemes, GGally, ggplot2, dplyr, pacman
```

Hide

```
#detach("package:datasets", unload = TRUE)  # For base packages

# Clear console
#cat("\014")  # ctrl+L
```