ADS 506 Final Project

2022-12-05

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Including Plots

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

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

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

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

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

```
library(ggplot2)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
```

```
## v tibble  3.1.8     v stringr 1.4.1
## v readr   2.1.3     v forcats 0.5.2
## v purrr   0.3.5
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::lift()   masks caret::lift()
```

```
library(tidyr)
library(lubridate)
```

```
## Loading required package: timechange
##
## Attaching package: 'lubridate'
##
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(forecast)
```

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

```
library(tseries)
library(zoo)
```

```
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
# Importing the dataset
walmart_df <- read.csv("walmart-sales-dataset-of-45stores.csv")
head(walmart_df)
```

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

```
dim(walmart_df)
```

```
## [1] 6435    8
```

```
#The dataset contains 6435 rows and 11 columns
#Checking null vales
colSums(is.na(walmart_df))
```

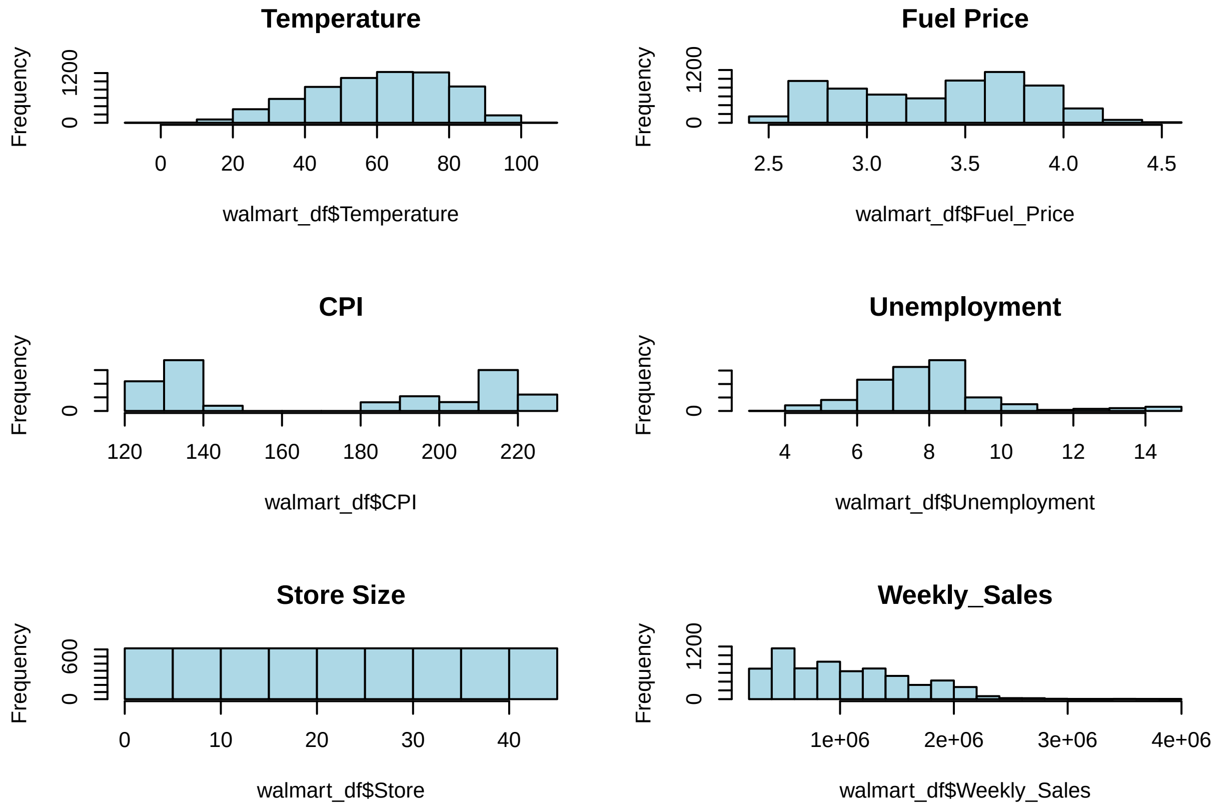
```
##        Store         Date Weekly_Sales Holiday_Flag  Temperature   Fuel_Price
##            0            0            0            0            0            0
##          CPI Unemployment
##            0            0
```

```
#The dataset contains no missing values
```

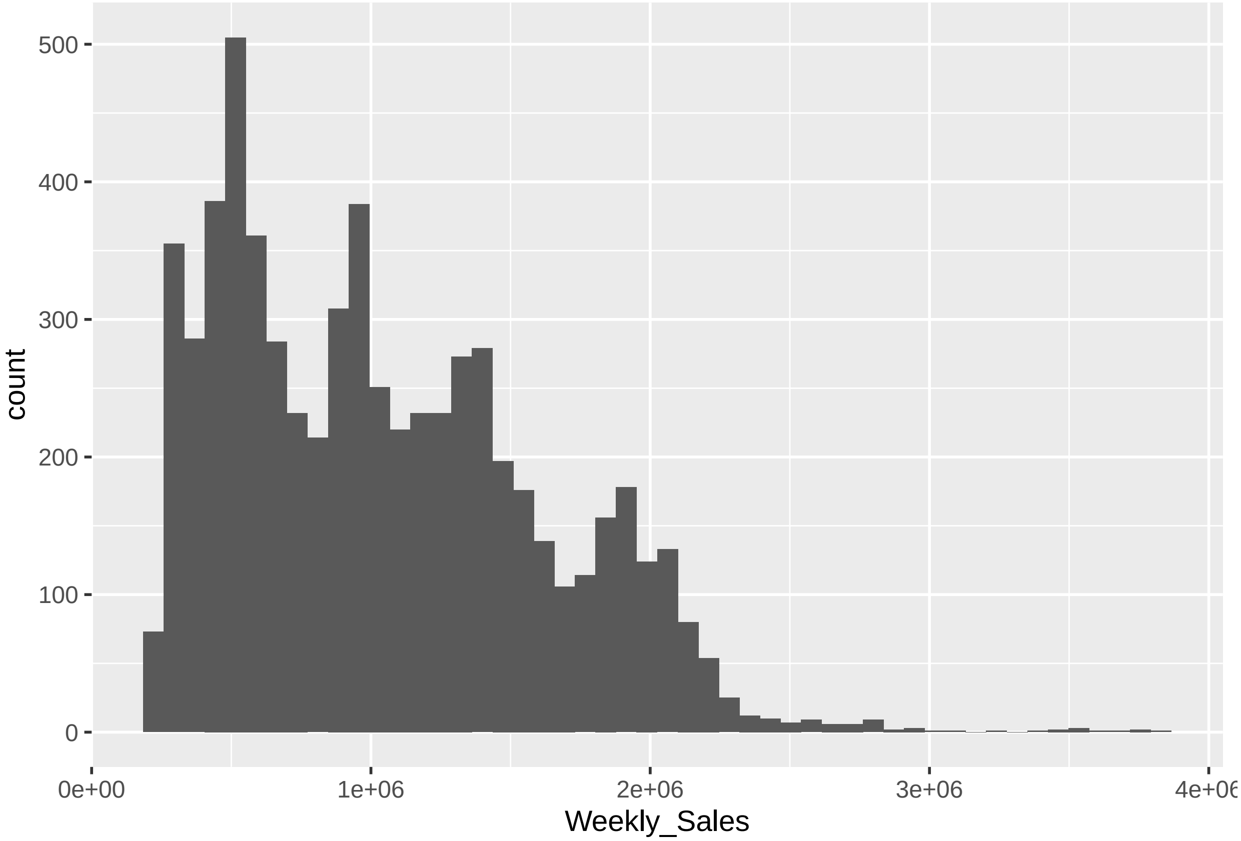
```
summary(walmart_df)
```

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

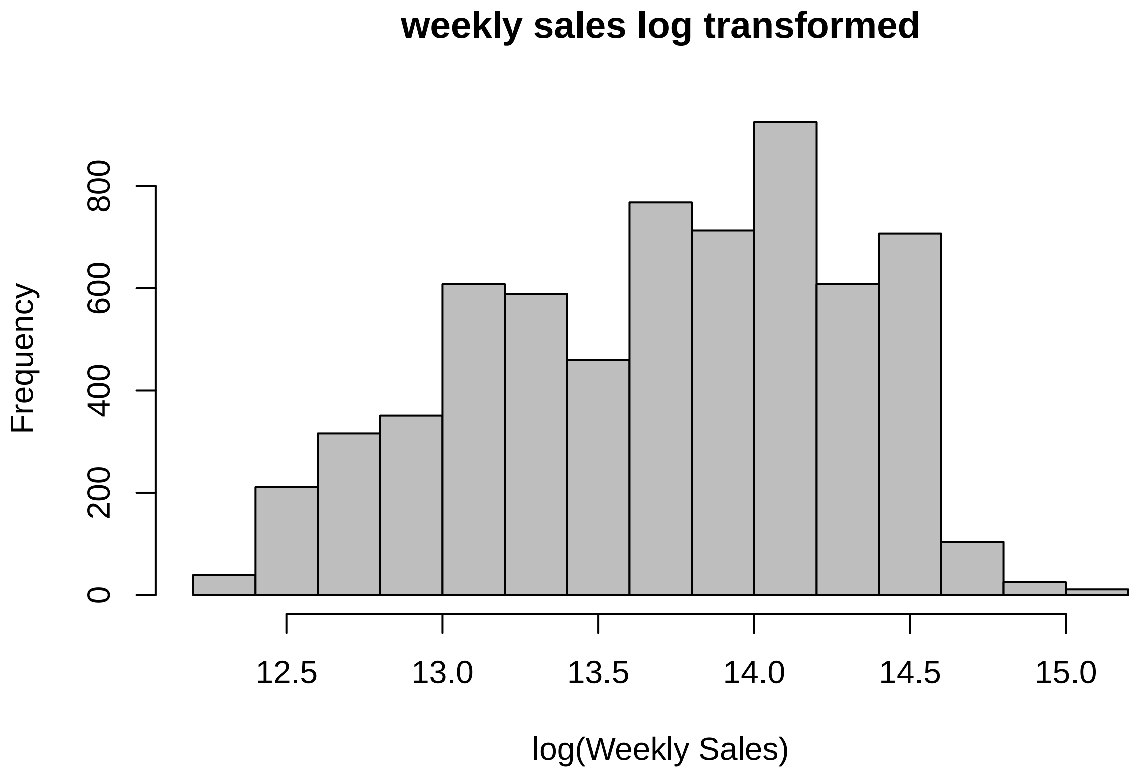
```
par(mfrow=c(3,2))
hist(walmart_df$Temperature, col = 'light blue', main = "Temperature")
hist(walmart_df$Fuel_Price, col = 'light blue', main = "Fuel Price")
hist(walmart_df$CPI, col = 'light blue', main = "CPI")
hist(walmart_df$Unemployment, col = 'light blue', main = "Unemployment")
hist(walmart_df$Store, col = 'light blue', main = "Store Size")
hist(walmart_df$Weekly_Sales, col = 'light blue', main = "Weekly_Sales")
```



```r
ggplot(data = walmart_df,aes(x=Weekly_Sales)) + geom_histogram(bins=50)
```



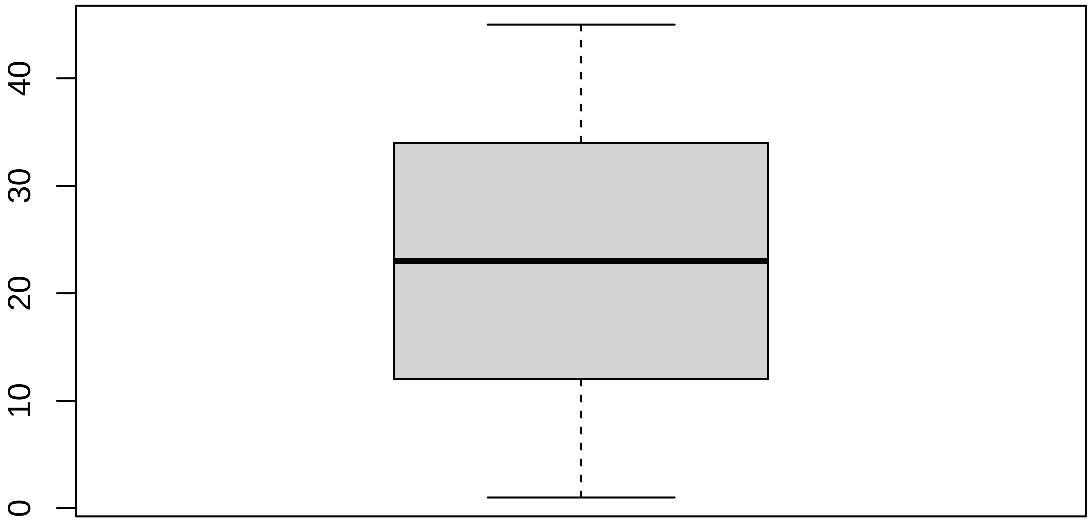
```r
hist(log(walmart_df$Weekly_Sales), col = 'gray',
     main = "weekly sales log transformed",
     xlab ='log(Weekly Sales)')
```



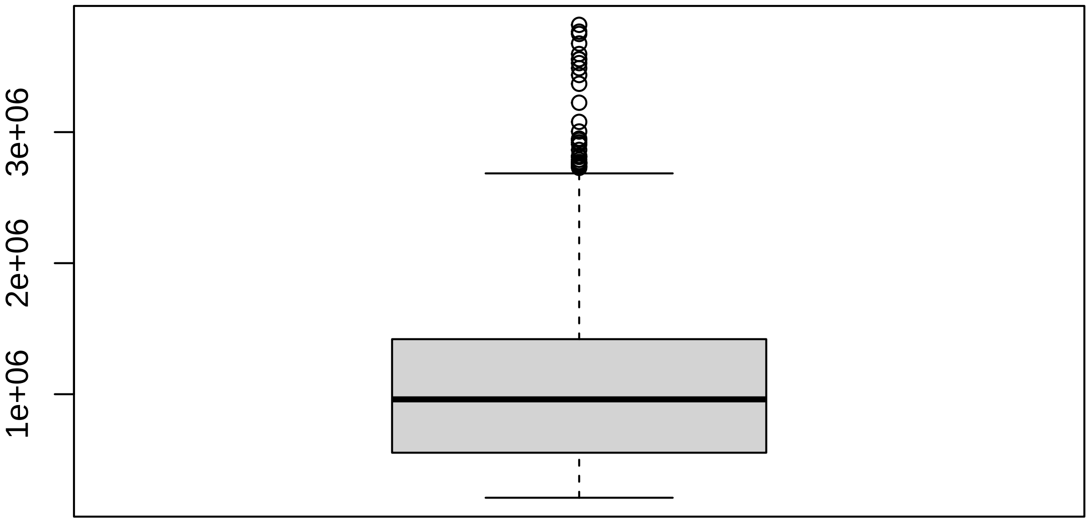
```
num_cols <- unlist(lapply(walmart_df, is.numeric))         # Identify numeric columns
data_numeric <- walmart_df[, num_cols]
head(data_numeric)
```

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

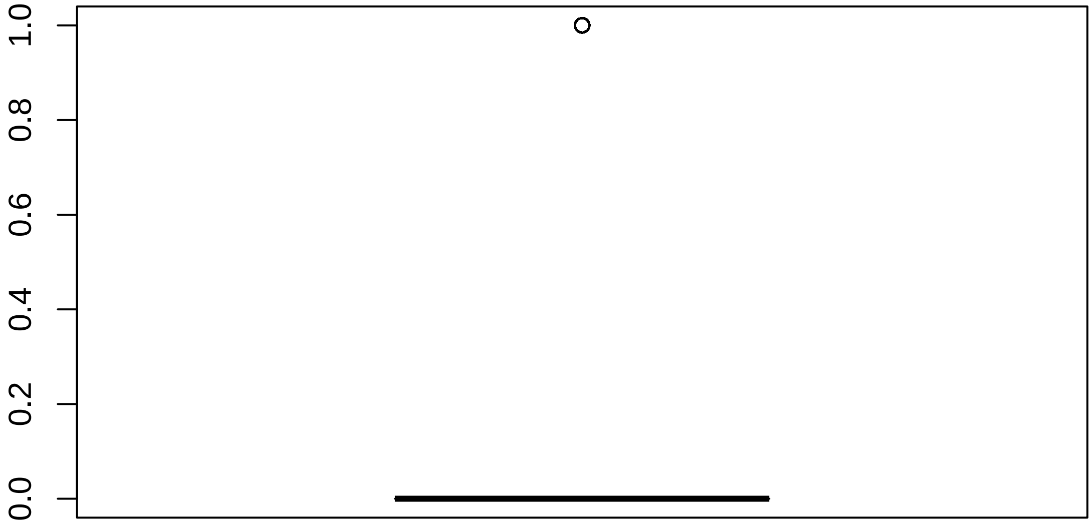
```
boxplot(data_numeric$Store)
```



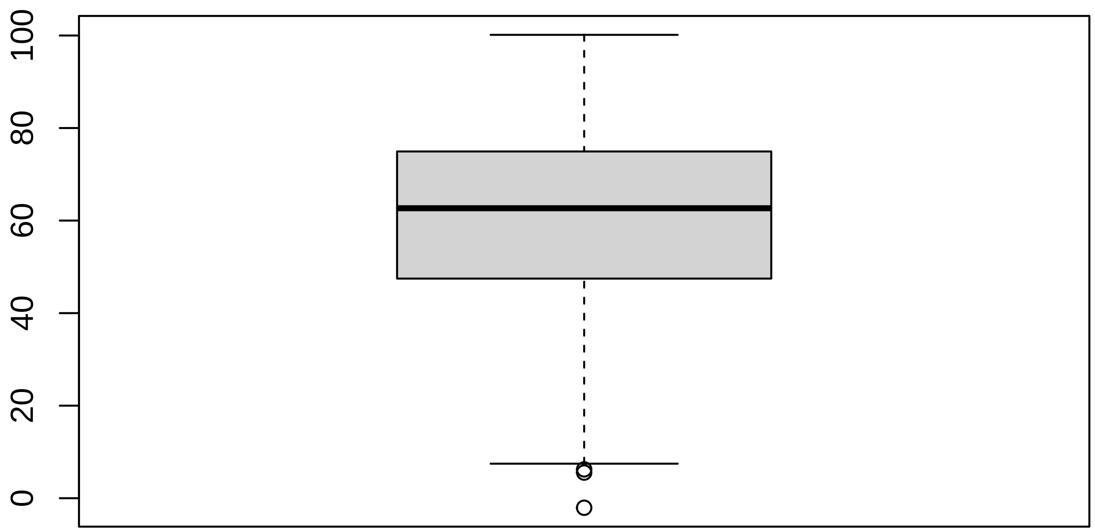
```
boxplot(data_numeric$Weekly_Sales)
```



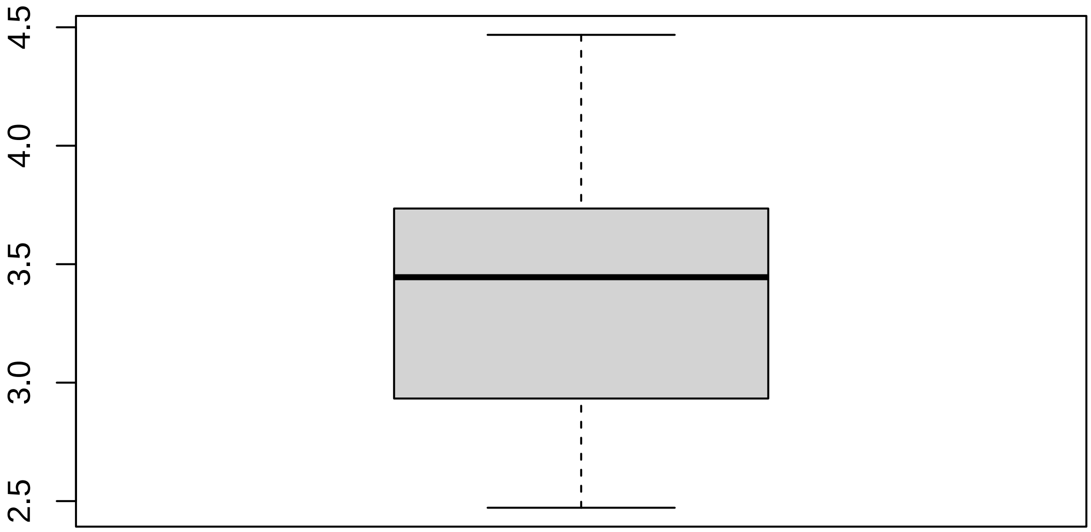
```
boxplot(data_numeric$Holiday_Flag)
```



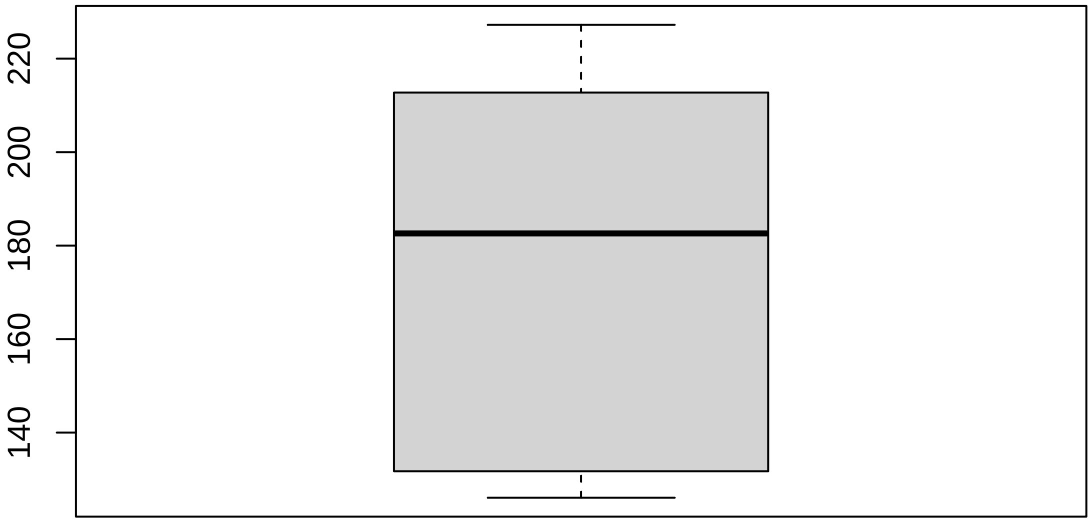
```
boxplot(data_numeric$Temperature)
```



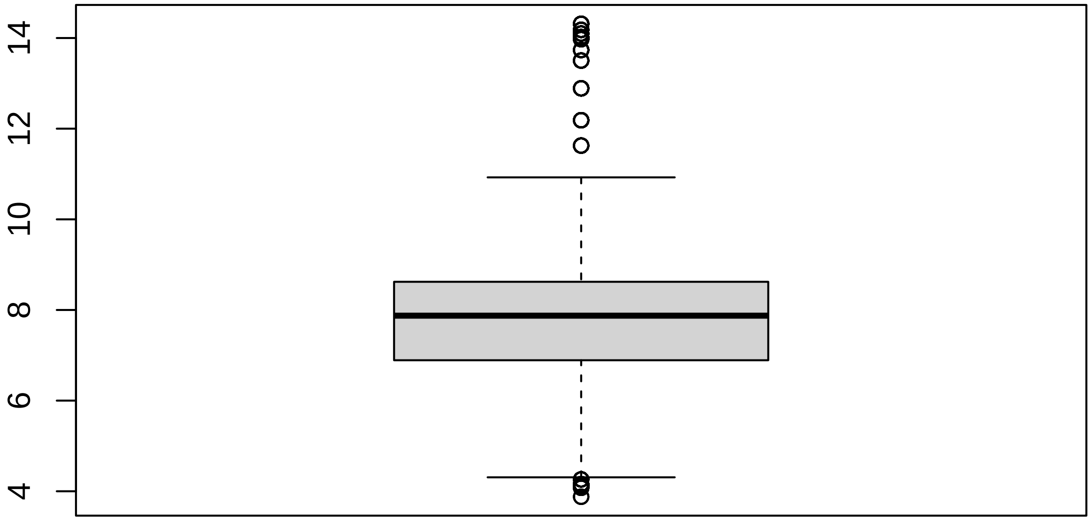
```
boxplot(data_numeric$Fuel_Price)
```



```
boxplot(data_numeric$CPI)
```



```
boxplot(data_numeric$Unemployment)
```

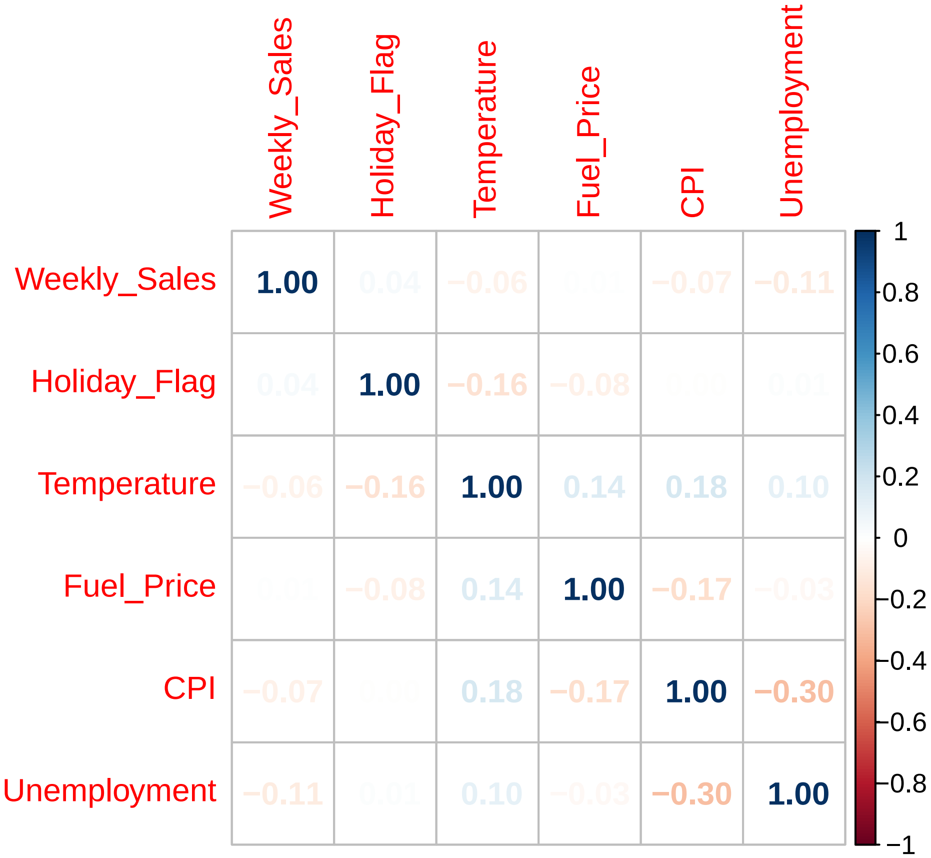


# Correlation Matrix

```
corr = cor(walmart_df[, c(3:8)])
corr
```

```
##              Weekly_Sales Holiday_Flag Temperature   Fuel_Price          CPI
## Weekly_Sales  1.000000000  0.036890968 -0.06381001  0.009463786 -0.072634162
## Holiday_Flag  0.036890968  1.000000000 -0.15509133 -0.078346518 -0.002162091
## Temperature  -0.063810013 -0.155091329  1.00000000  0.144981806  0.176887676
## Fuel_Price    0.009463786 -0.078346518  0.14498181  1.000000000 -0.170641795
## CPI          -0.072634162 -0.002162091  0.17688768 -0.170641795  1.000000000
## Unemployment -0.106176090  0.010960284  0.10115786 -0.034683745 -0.302020064
##              Unemployment
## Weekly_Sales  -0.10617609
## Holiday_Flag   0.01096028
## Temperature    0.10115786
## Fuel_Price    -0.03468374
## CPI           -0.30202006
## Unemployment   1.00000000
```

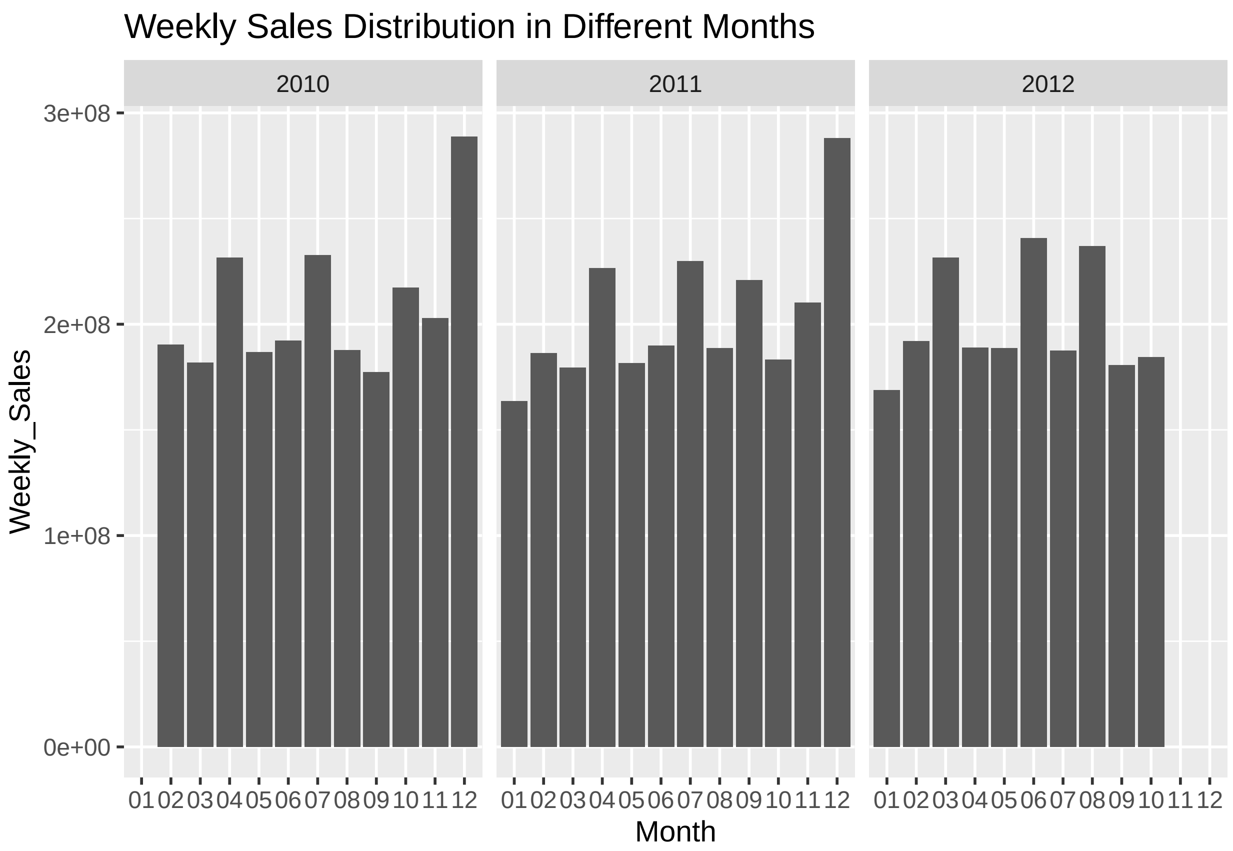
```
corrplot(corr, method = 'number')
```



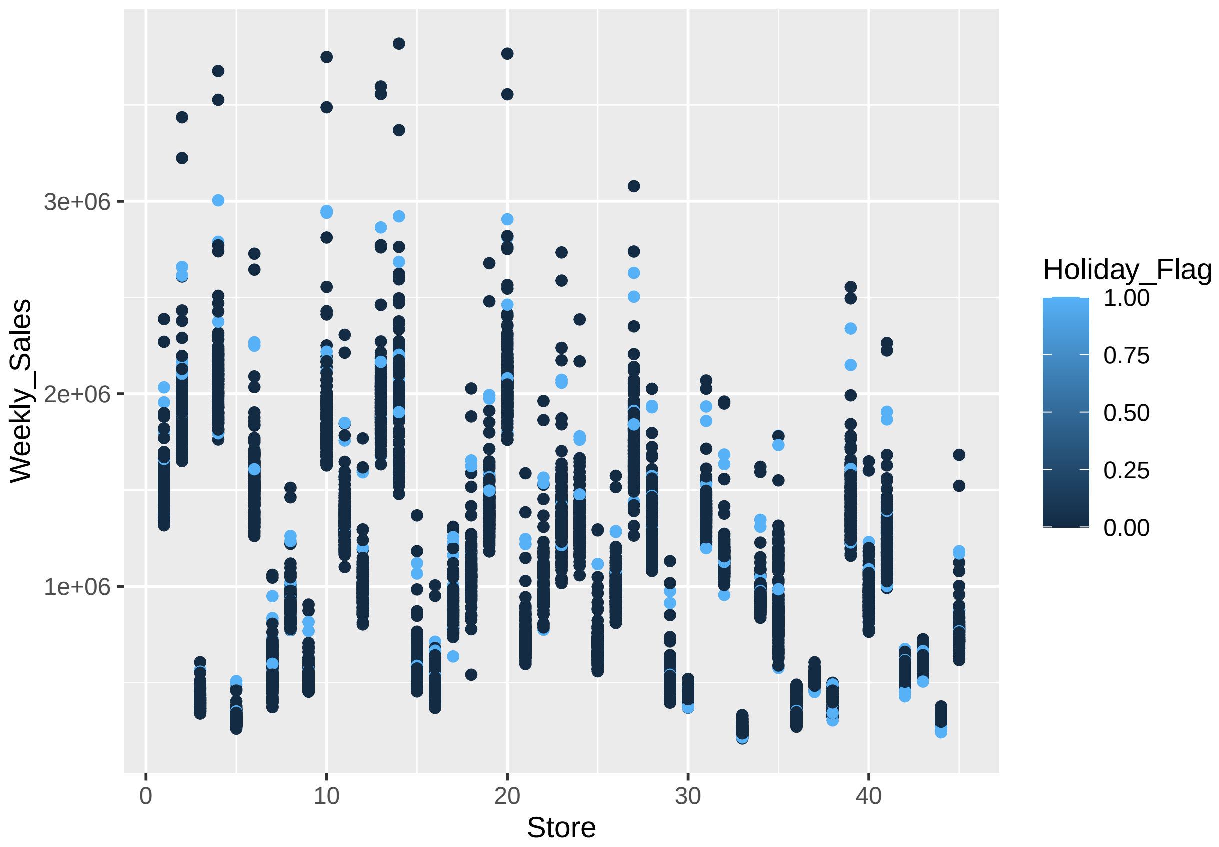
There is a moderate positive correlation between sales, fuel price and holidays, and negative correlation between sales, unemployment, CPI, and temperature.

Weekly sales distribution in differnt month

```
walmart_df$Date <- as.Date(walmart_df$Date, format = "%d-%m-%Y")
walmart_df[,"Year"] <- format(walmart_df[,"Date"],"%Y")
walmart_df[,"Month"] <- format(walmart_df[,"Date"],"%m")
ggplot(walmart_df, aes(x = Month,y = Weekly_Sales )) + geom_col() + facet_wrap(~Year) + ggtitle("Weekly
```



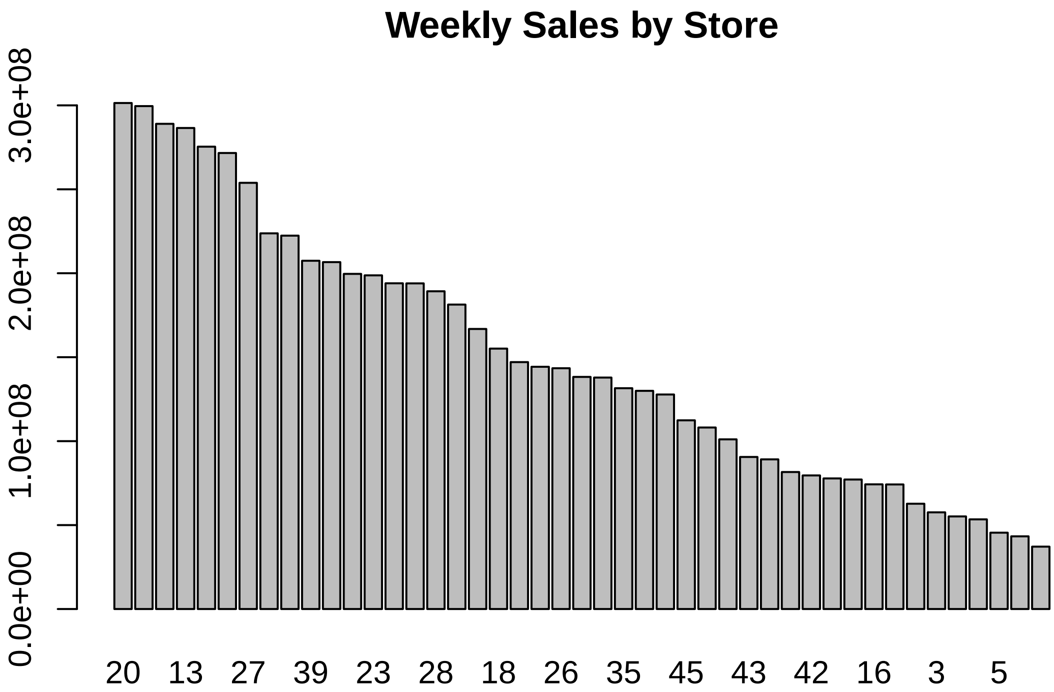
```
ggplot(walmart_df, aes(x=Store, y=Weekly_Sales, color=Holiday_Flag)) +
  geom_point()
```



```
sales_store <- aggregate(Weekly_Sales ~ Store, data = walmart_df, sum)
sales_store <- arrange(sales_store, desc(Weekly_Sales))
sales_store
```

```
##    Store Weekly_Sales
## 1     20    301397792
## 2      4    299543953
## 3     14    288999911
## 4     13    286517704
## 5      2    275382441
## 6     10    271617714
## 7     27    253855917
## 8      6    223756131
## 9      1    222402809
## 10    39    207445542
## 11    19    206634862
## 12    31    199613906
## 13    23    198750618
## 14    24    194016021
## 15    11    193962787
## 16    28    189263681
## 17    41    181341935
## 18    32    166819246
## 19    18    155114734
## 20    22    147075649
## 21    12    144287230
## 22    26    143416394
## 23    34    138249763
## 24    40    137870310
## 25    35    131520672
## 26     8    129951181
## 27    17    127782139
## 28    45    112395341
## 29    21    108117879
## 30    25    101061179
## 31    43     90565435
## 32    15     89133684
## 33     7     81598275
## 34    42     79565752
## 35     9     77789219
## 36    29     77141554
## 37    16     74252425
## 38    37     74202740
## 39    30     62716885
## 40     3     57586735
## 41    38     55159626
## 42    36     53412215
## 43     5     45475689
## 44    44     43293088
## 45    33     37160222
```

```
sales_store$Store <- as.character(sales_store$Store)
sales_store$Store <- factor(sales_store$Store, levels=unique(sales_store$Store))
colnames(sales_store) <- c("Store","Weekly_Sales")
barplot(sales_store$Weekly_Sales,names=sales_store$Store, main="Weekly Sales by Store")
```



```r
#Creating Holidays Data dataframe
Holiday_date <- c("12-02-2010", "11-02-2011", "10-02-2012", "08-02-2013","10-09-2010", "09-09-2011", "0
holidays <-c(rep("Super Bowl", 4),
           rep("Labour Day", 4),
           rep("Thanksgiving", 4),
           rep("Christmas", 4))
holidays
```

```
##  [1] "Super Bowl"   "Super Bowl"   "Super Bowl"   "Super Bowl"   "Labour Day"
##  [6] "Labour Day"   "Labour Day"   "Labour Day"   "Thanksgiving" "Thanksgiving"
## [11] "Thanksgiving" "Thanksgiving" "Christmas"    "Christmas"    "Christmas"
## [16] "Christmas"
```

```r
Holidays_Data <- data.frame(holidays,Holiday_date)
Holidays_Data
```

```
##        holidays Holiday_date
## 1    Super Bowl   12-02-2010
## 2    Super Bowl   11-02-2011
## 3    Super Bowl   10-02-2012
## 4    Super Bowl   08-02-2013
## 5    Labour Day   10-09-2010
## 6    Labour Day   09-09-2011
## 7    Labour Day   07-09-2012
## 8    Labour Day   06-09-2013
## 9  Thanksgiving   26-11-2010
## 10 Thanksgiving   25-11-2011
## 11 Thanksgiving   23-11-2012
## 12 Thanksgiving  29- 11-2013
## 13    Christmas   31-12-2010
## 14    Christmas   30-12-2011
## 15    Christmas   28-12-2012
## 16    Christmas   27-12-2013
```

```
#merging both dataframes
walmart_df.2<-merge(walmart_df,Holidays_Data, by.x= "Date", by.y="Holiday_date", all.x = TRUE)
head(walmart_df.2)
```

```
##         Date Store Weekly_Sales Holiday_Flag Temperature Fuel_Price      CPI
## 1 2010-02-05     1    1643690.9            0       42.31      2.572 211.0964
## 2 2010-02-05     2    2136989.5            0       40.19      2.572 210.7526
## 3 2010-02-05     3     461622.2            0       45.71      2.572 214.4249
## 4 2010-02-05     4    2135143.9            0       43.76      2.598 126.4421
## 5 2010-02-05     5     317173.1            0       39.70      2.572 211.6540
## 6 2010-02-05     6    1652635.1            0       40.43      2.572 212.6224
##   Unemployment Year Month holidays
## 1        8.106 2010    02     <NA>
## 2        8.324 2010    02     <NA>
## 3        7.368 2010    02     <NA>
## 4        8.623 2010    02     <NA>
## 5        6.566 2010    02     <NA>
## 6        7.259 2010    02     <NA>
```

```
#Replacing null values in Event with No_Holiday
walmart_df.2$holidays = as.character(walmart_df.2$holidays)
walmart_df.2$holidays[is.na(walmart_df.2$holidays)]= "No_Holiday"
head(walmart_df.2)
```

```
##         Date Store Weekly_Sales Holiday_Flag Temperature Fuel_Price      CPI
## 1 2010-02-05     1    1643690.9            0       42.31      2.572 211.0964
## 2 2010-02-05     2    2136989.5            0       40.19      2.572 210.7526
## 3 2010-02-05     3     461622.2            0       45.71      2.572 214.4249
## 4 2010-02-05     4    2135143.9            0       43.76      2.598 126.4421
## 5 2010-02-05     5     317173.1            0       39.70      2.572 211.6540
## 6 2010-02-05     6    1652635.1            0       40.43      2.572 212.6224
##   Unemployment Year Month   holidays
## 1        8.106 2010    02 No_Holiday
## 2        8.324 2010    02 No_Holiday
## 3        7.368 2010    02 No_Holiday
## 4        8.623 2010    02 No_Holiday
## 5        6.566 2010    02 No_Holiday
## 6        7.259 2010    02 No_Holiday
```

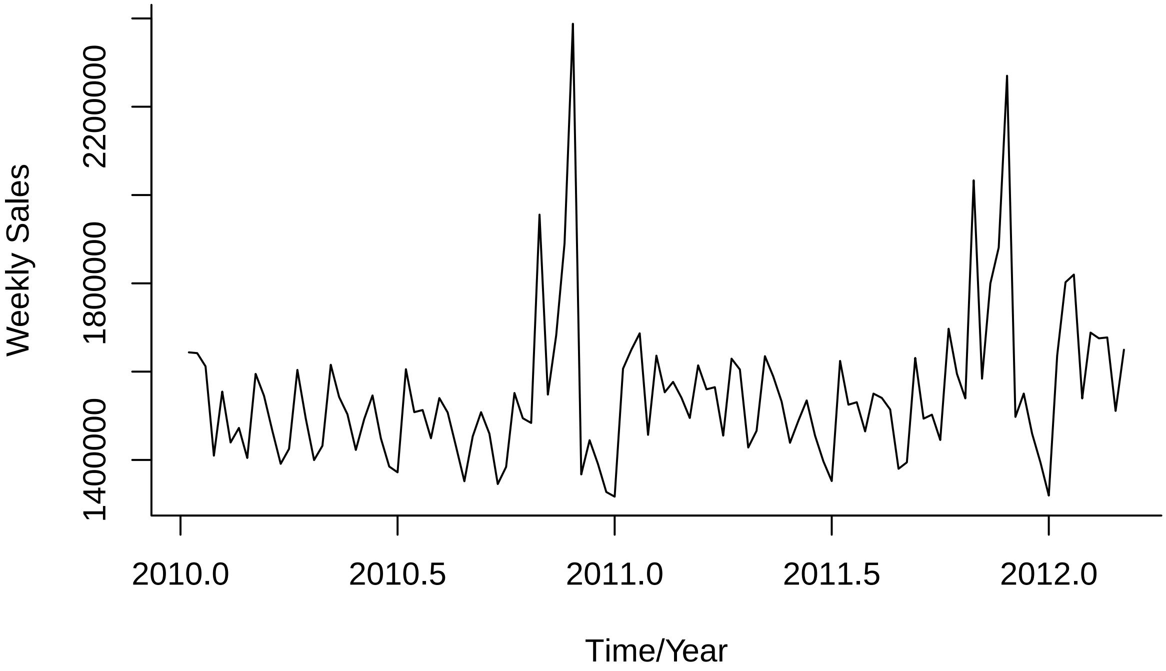
```
Holiday_Sales<-aggregate(Weekly_Sales ~ holidays, data = walmart_df.2, mean)
Holiday_Sales
```

```
##     holidays Weekly_Sales
## 1 No_Holiday      1046965
```

We have the most sales in the Thanksgiving and super bowl each year.

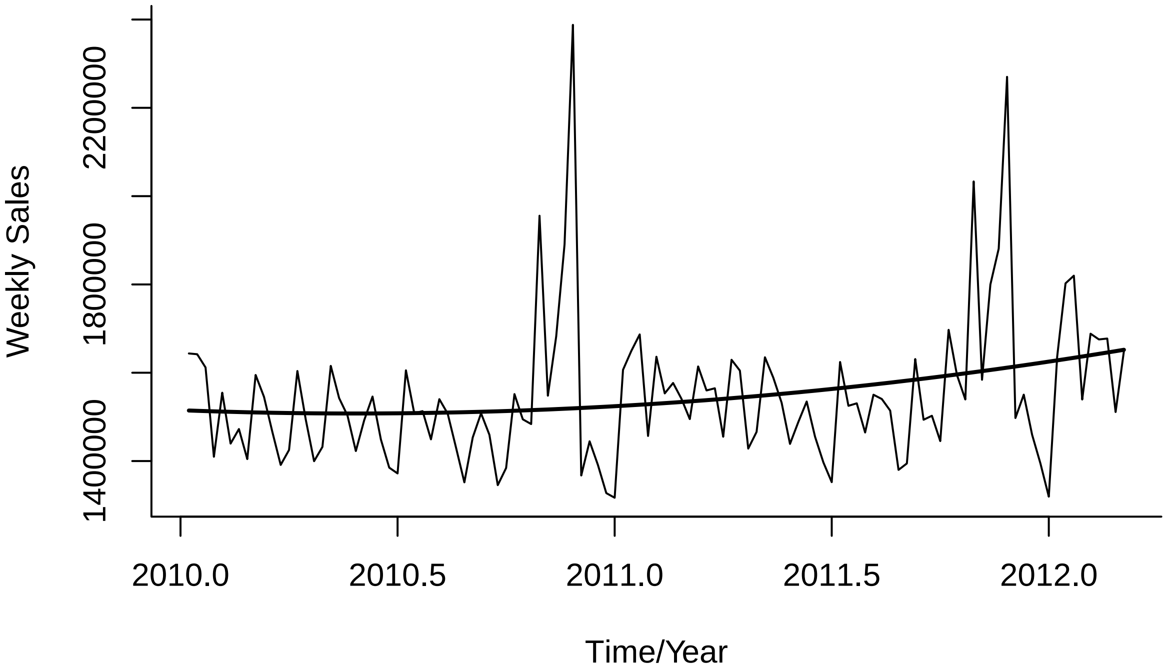
```
# plotting timeseries
walmart.ts  = ts(walmart_df$Weekly_Sales, frequency = 52,
                 start = c(2010,2), end = c(2012,10))
```

```
plot(walmart.ts, xlab="Time/Year",
     ylab="Weekly Sales", bty='l')
```



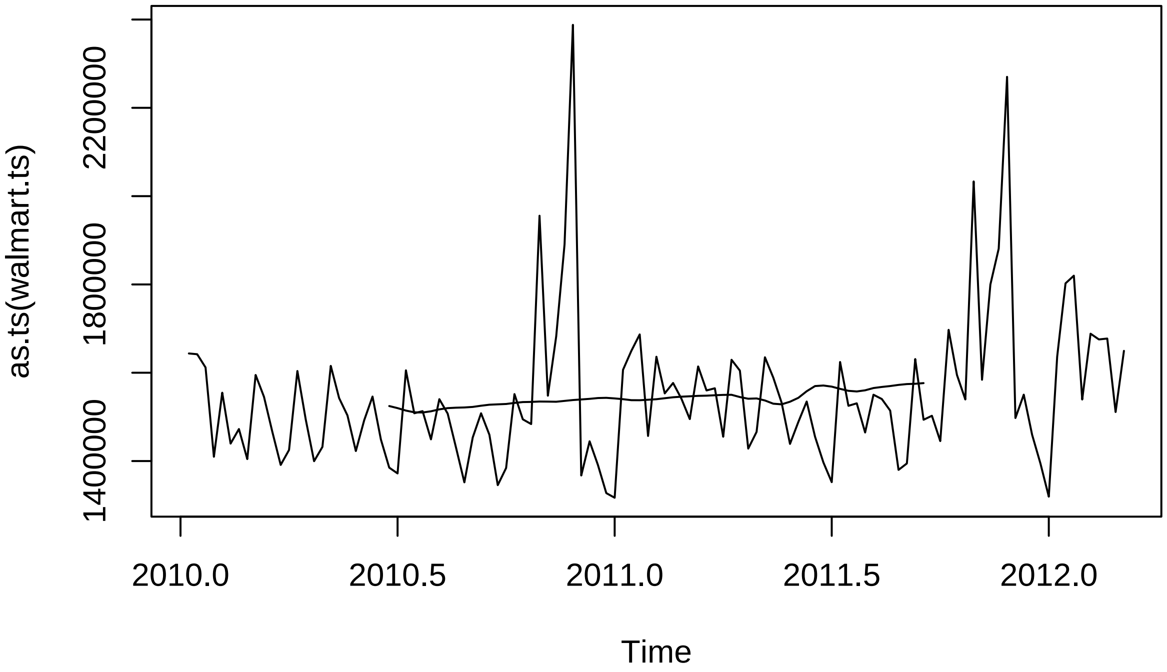
The time series start in February 2010, and ends in October 2012, and has frequency of 52 weeks per year.

```
walmart.lm <- tslm(walmart.ts ~ trend + I(trend^2))
par(mfrow = c(1,1))
plot(walmart.ts, xlab="Time/Year", ylab="Weekly Sales", bty="l")
lines(walmart.lm$fitted, lwd=2)
```

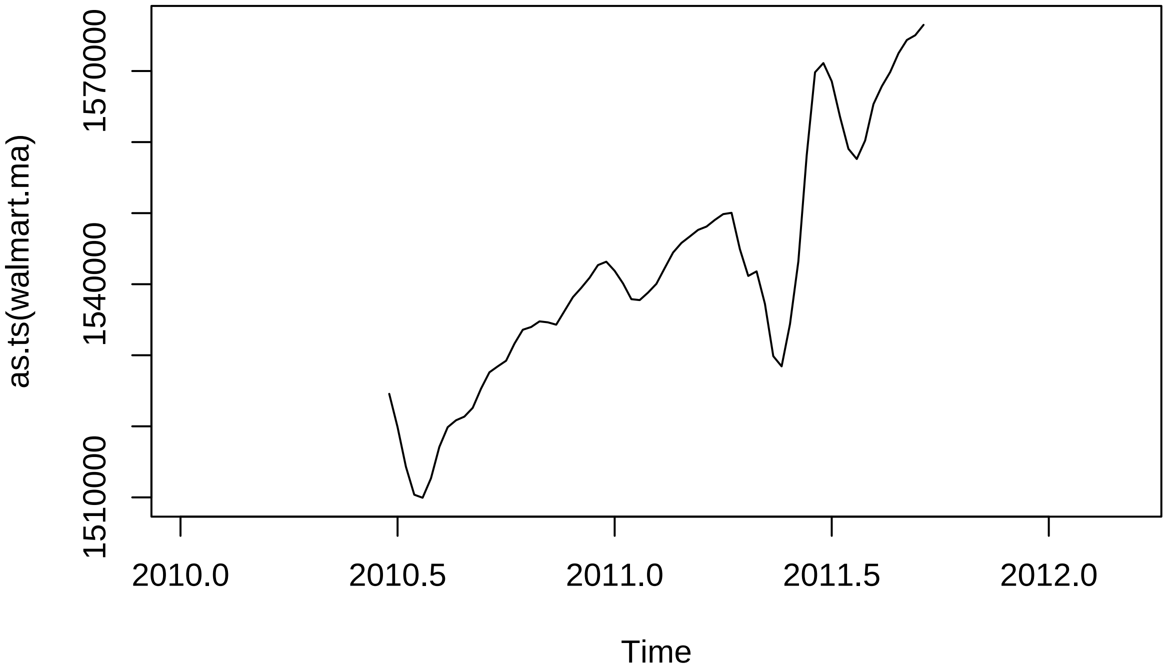


It is helpful to divide a time series into a systematic component and a non-systematic part in order to select appropriate forecasting techniques, which the componenets are level, trend, seasonality and noise. From the figure above we can understand that the weekly sales for walmart organization has constant trend with additive seasonality.

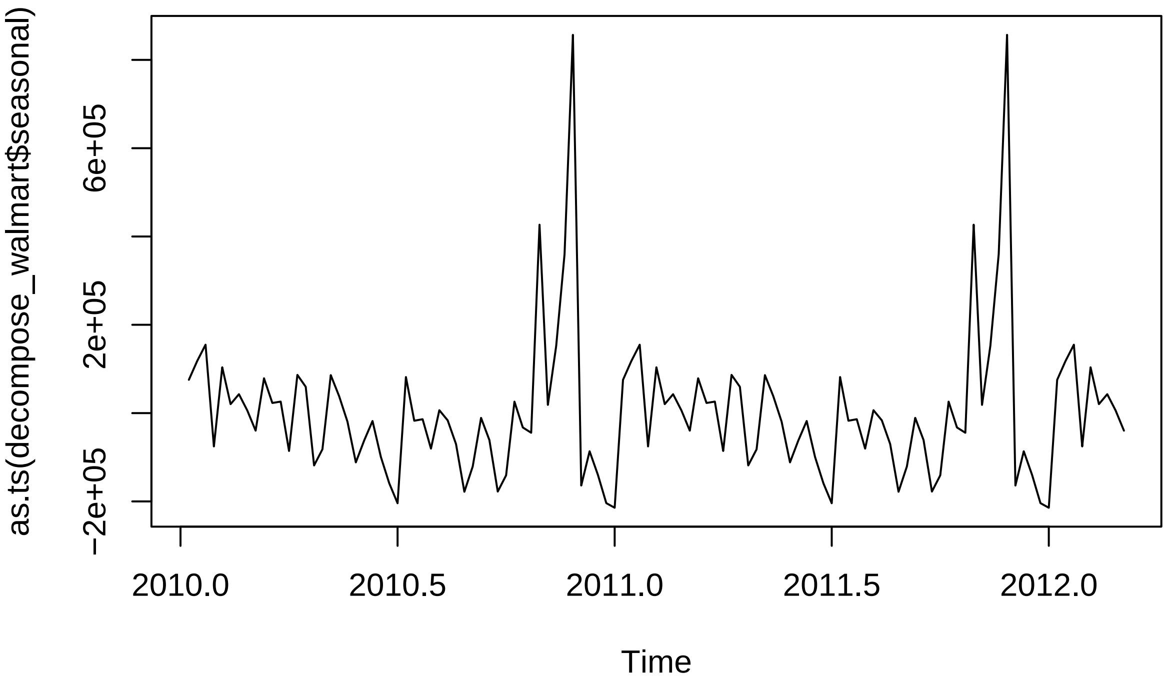
```
walmart.ma = ma(walmart.ts, order = 48, centre = T) # As it is recorded yearly, there are 48 data point
plot(as.ts(walmart.ts))
lines(walmart.ma)
```



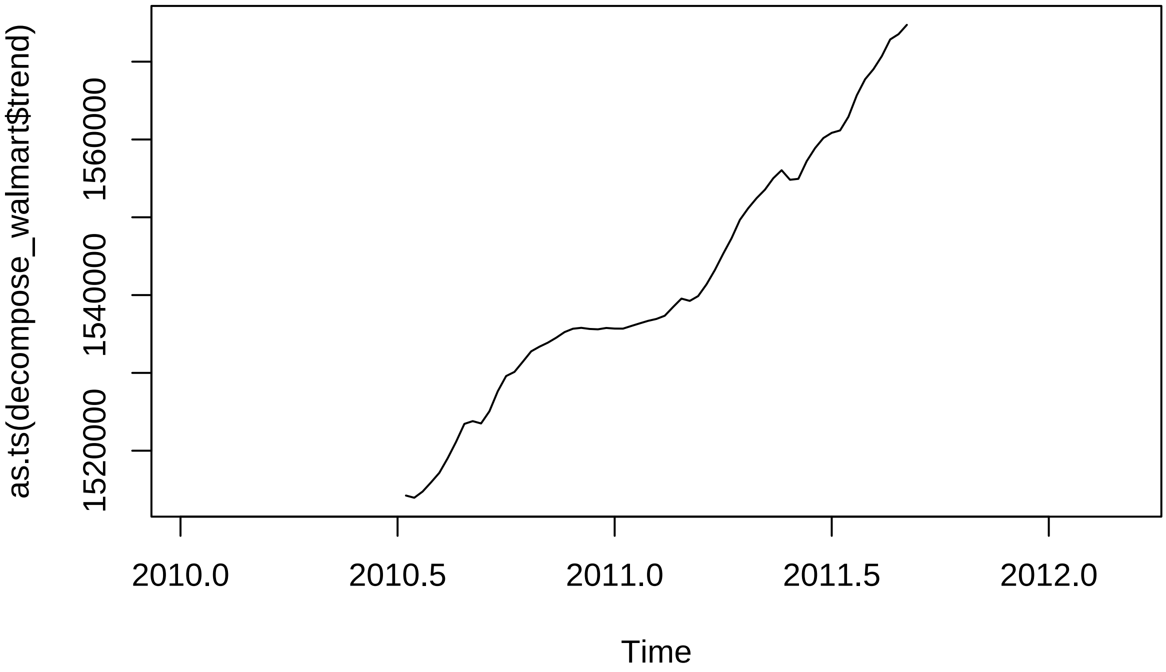
```
plot(as.ts(walmart.ma))
```



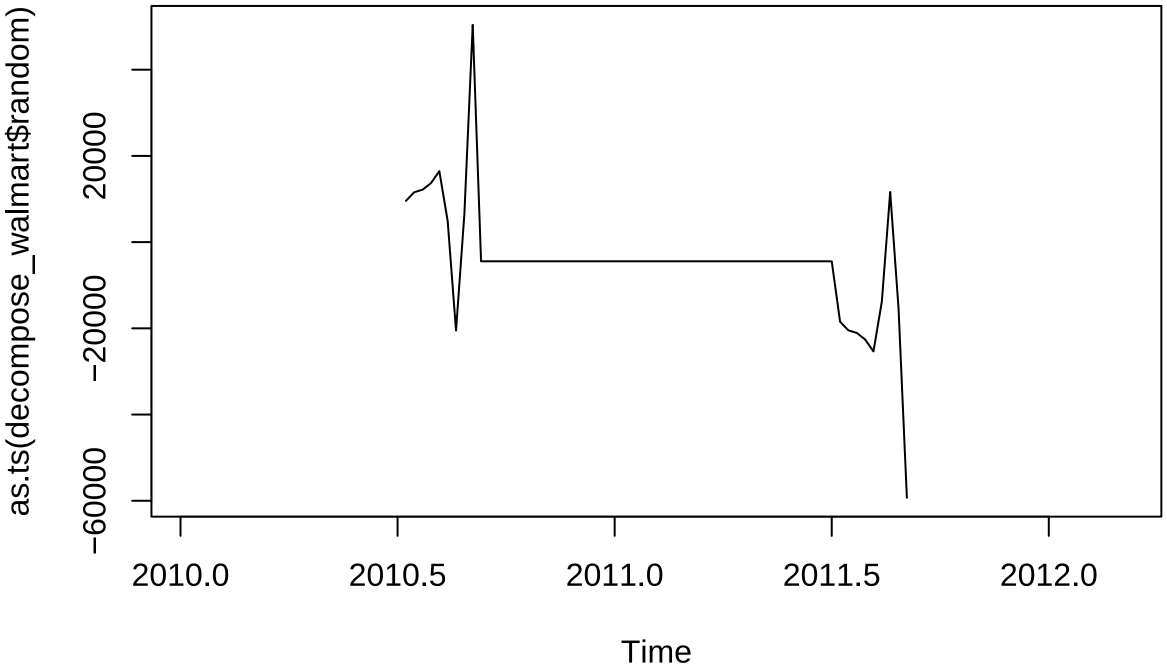
```
decompose_walmart = decompose(walmart.ts, "additive")

plot(as.ts(decompose_walmart$seasonal))
```



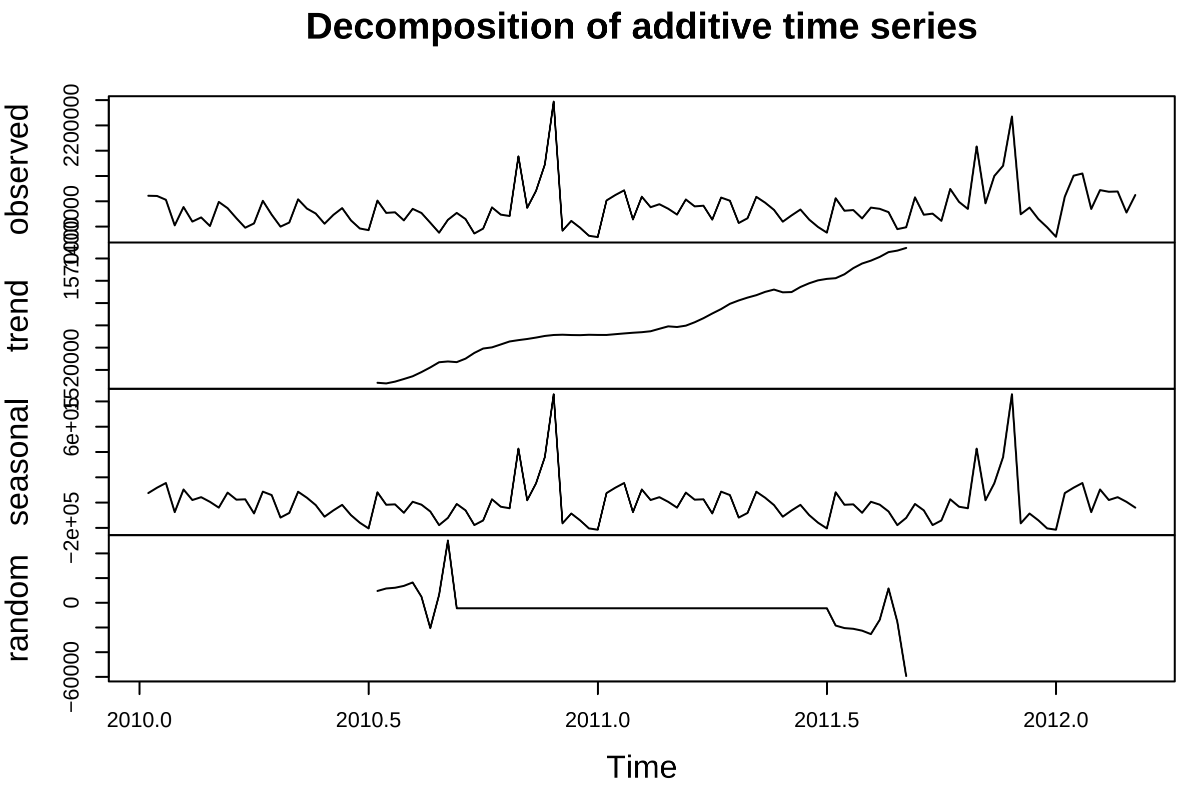
```
plot(as.ts(decompose_walmart$trend))
```



```
plot(as.ts(decompose_walmart$random))
```



```
plot(decompose_walmart)
```

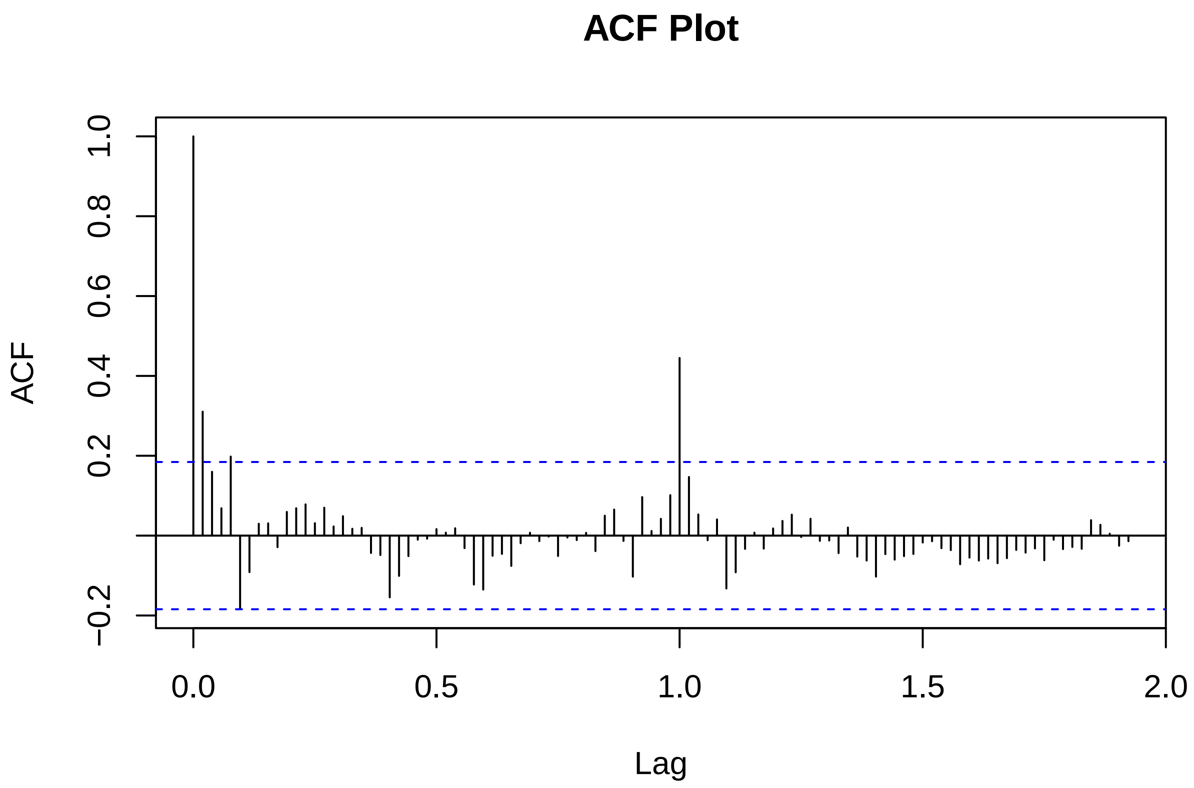


Modeling Time Series

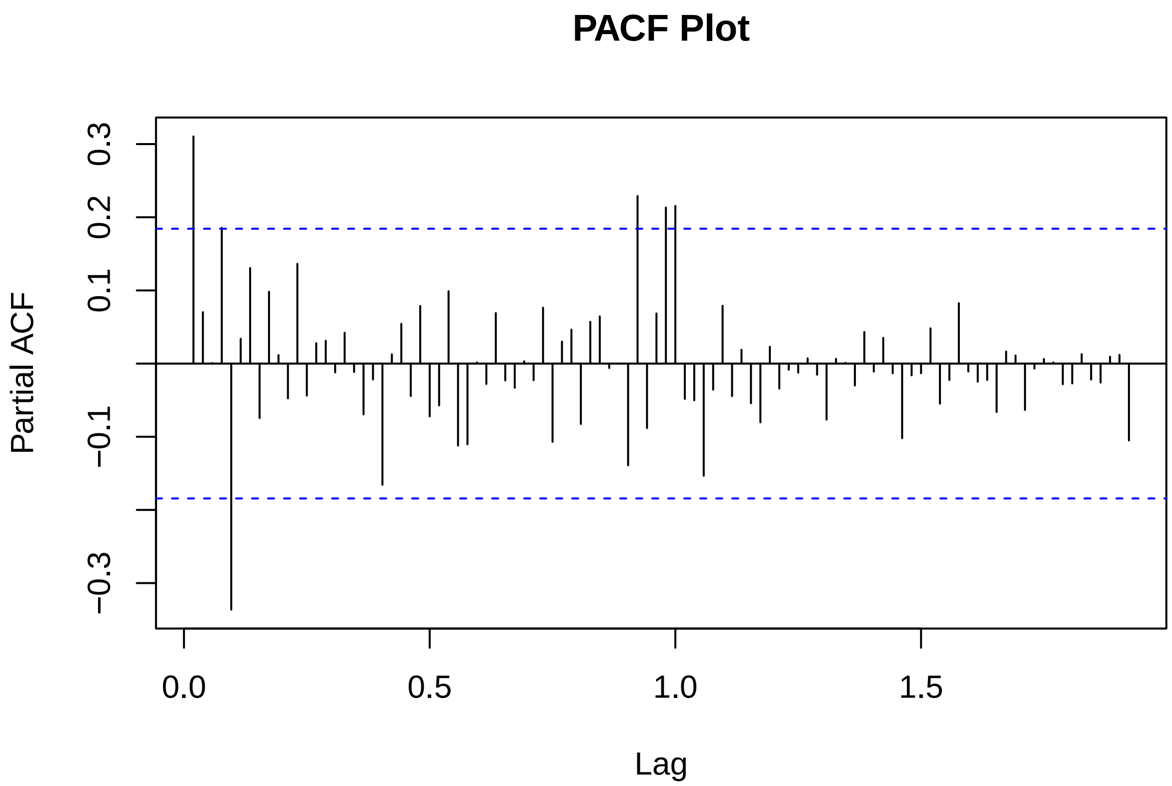
Standard statistical models presuppose that observations are independent. This premise is false for time series. The reliance that only the past up to time t allows us to anticipate what will happen at time t+k is what we wish to model in time series. We refer to this type of dependence—where each observation is connected to itself at a prior time—as autocorrelation. If autocorrelation exists, the dependent variable must be appropriately delayed as predictive variables in the model.

Autocorrelation

```
par(mfrow = c(1,1))
acf = acf(walmart.ts, main='ACF Plot', lag.max=100)
```



```
pacf = pacf(walmart.ts, main='PACF Plot', lag.max=100)
```



```
#Augmented Dickey-Fuller(ADF) Test
print(adf.test(walmart.ts))
```

```
## Warning in adf.test(walmart.ts): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  walmart.ts
## Dickey-Fuller = -5.457, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

Data partitioning

```
walmart_df$Date[1]
```

```
## [1] "2010-02-05"
```

```
walmart_df$Date[6435]
```

```
## [1] "2012-10-26"
```

Partition the data into training and validation periods, so that years 2010 - October 2011 are the training period and the rest of the data is for validation

#traindf <- window(walmart.ts, start = c(2010,5), end = c(2011,10)) #testdf <- window(walmart.ts, start = c(2011,11)) #testdf

```
nValid <- 52
nTrain <- length(walmart.ts) - nValid
train.ts_df <- window(walmart.ts, start = c(2010, 5), end = c(2010, nTrain))
train.ts_df
```

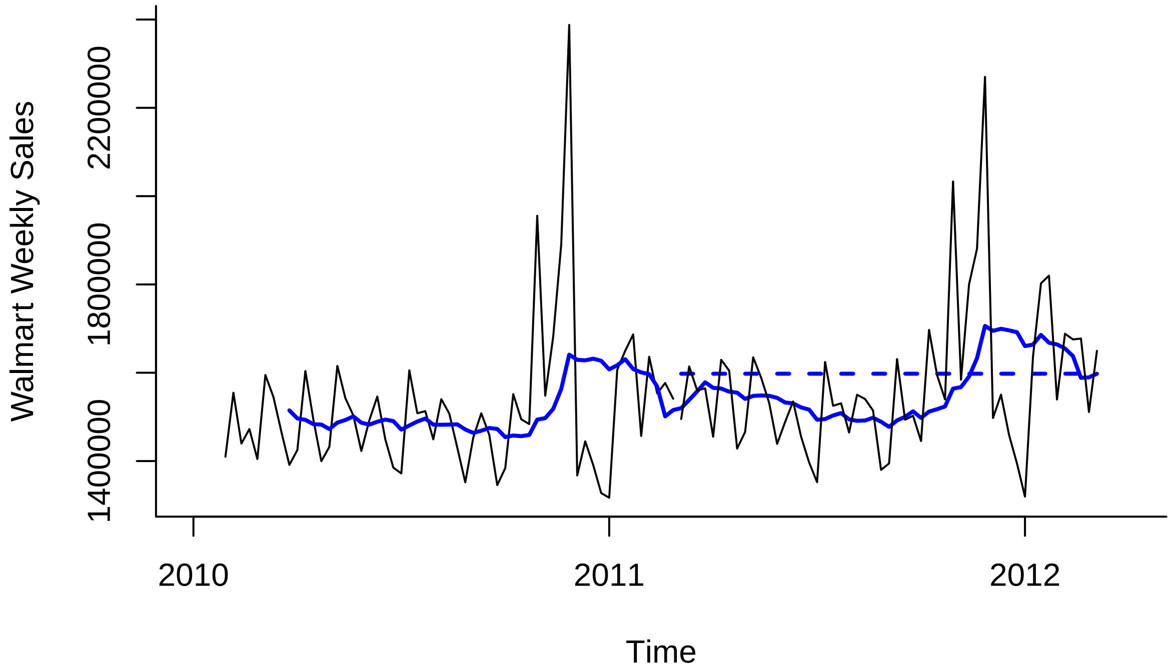
```
## Time Series:
## Start = c(2010, 5)
## End = c(2011, 9)
## Frequency = 52
##  [1] 1409728 1554807 1439542 1472516 1404430 1594968 1545419 1466058 1391256
## [10] 1425101 1603955 1494252 1399662 1432070 1615525 1542561 1503284 1422712
## [19] 1492418 1546074 1448939 1385065 1371987 1605492 1508238 1513080 1449143
## [28] 1540164 1507461 1430379 1351791 1453330 1508240 1459409 1345454 1384209
## [37] 1551659 1494479 1483784 1955624 1548034 1682614 1891035 2387950 1367320
## [46] 1444732 1391014 1327405 1316899 1606630 1649615 1686843 1456800 1636263
## [55] 1553192 1576818 1541102
```

```
valid.ts_df <- window(walmart.ts, start = c(2010, nTrain + 1), end = c(2012,10))
valid.ts_df
```

```
## Time Series:
## Start = c(2011, 10)
## End = c(2012, 10)
## Frequency = 52
##  [1] 1495065 1614259 1559889 1564820 1455091 1629391 1604776 1428218 1466047
## [10] 1635078 1588948 1532115 1438830 1488538 1534850 1455120 1396927 1352220
## [19] 1624384 1525147 1530761 1464693 1550229 1540471 1514260 1380020 1394562
## [28] 1630990 1493526 1502563 1445249 1697230 1594939 1539484 2033321 1584084
## [37] 1799682 1881177 2270189 1497463 1550370 1459601 1394394 1319326 1636340
## [46] 1802477 1819870 1539388 1688421 1675431 1677473 1511068 1649605
```

Moving Average

```
ma.walmart <- rollmean(walmart.ts, k = 12, align = "right")
walmart.last.ma <- tail(ma.walmart, 1)
ma.trailing.pred <- ts(rep(walmart.last.ma, nValid), start = c(2010, nTrain + 1), end = c(2012,10), fre
plot(train.ts_df, ylab = "Walmart Weekly Sales", xlab = "Time", bty = "l", xaxt = "n", xlim = c(2010,20
axis(1, at = seq(2010, 2012, 1), labels = format(seq(2010, 2012, 1)))
lines(ma.walmart, lwd = 2, col = "blue")
lines(ma.trailing.pred, lwd = 2, col = "blue", lty = 2)
lines(valid.ts_df)
```

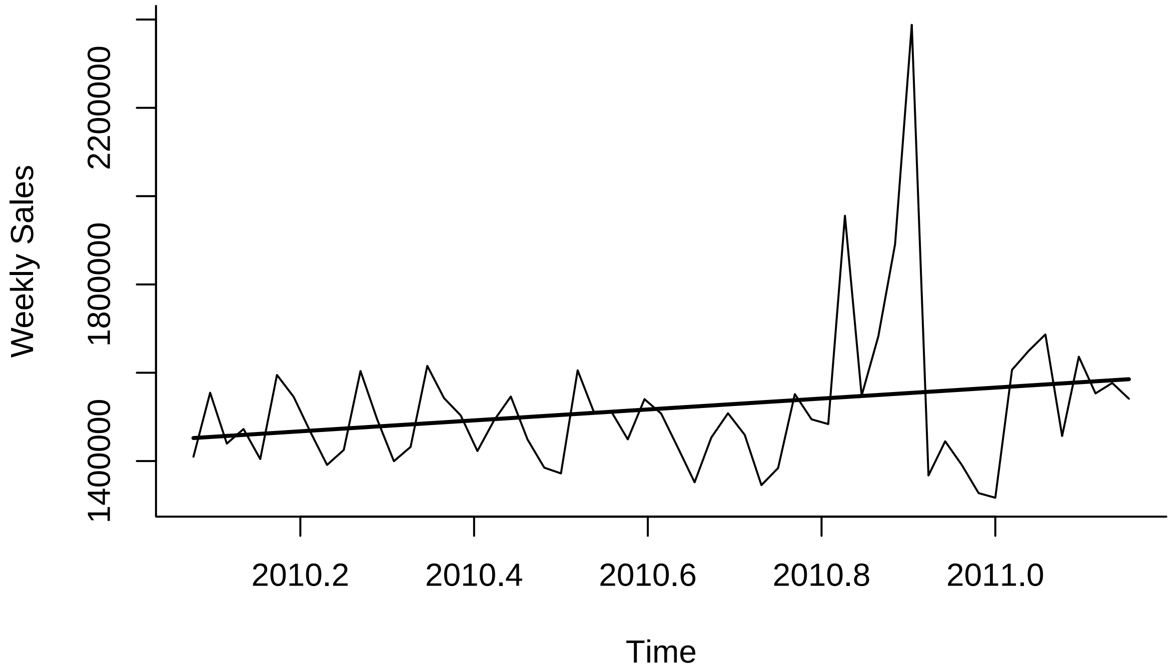


## Regression

```
# series linear model
train.lm <- tslm(train.ts_df ~ trend)
summary(train.lm)
```

```
##
## Call:
## tslm(formula = train.ts_df ~ trend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -249432  -72554  -17344   48741  833515
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1449748      44250  32.762   <2e-16 ***
## trend           2379       1327   1.793   0.0785 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 164800 on 55 degrees of freedom
## Multiple R-squared:  0.05521,    Adjusted R-squared:  0.03803
## F-statistic: 3.214 on 1 and 55 DF,  p-value: 0.07852
```

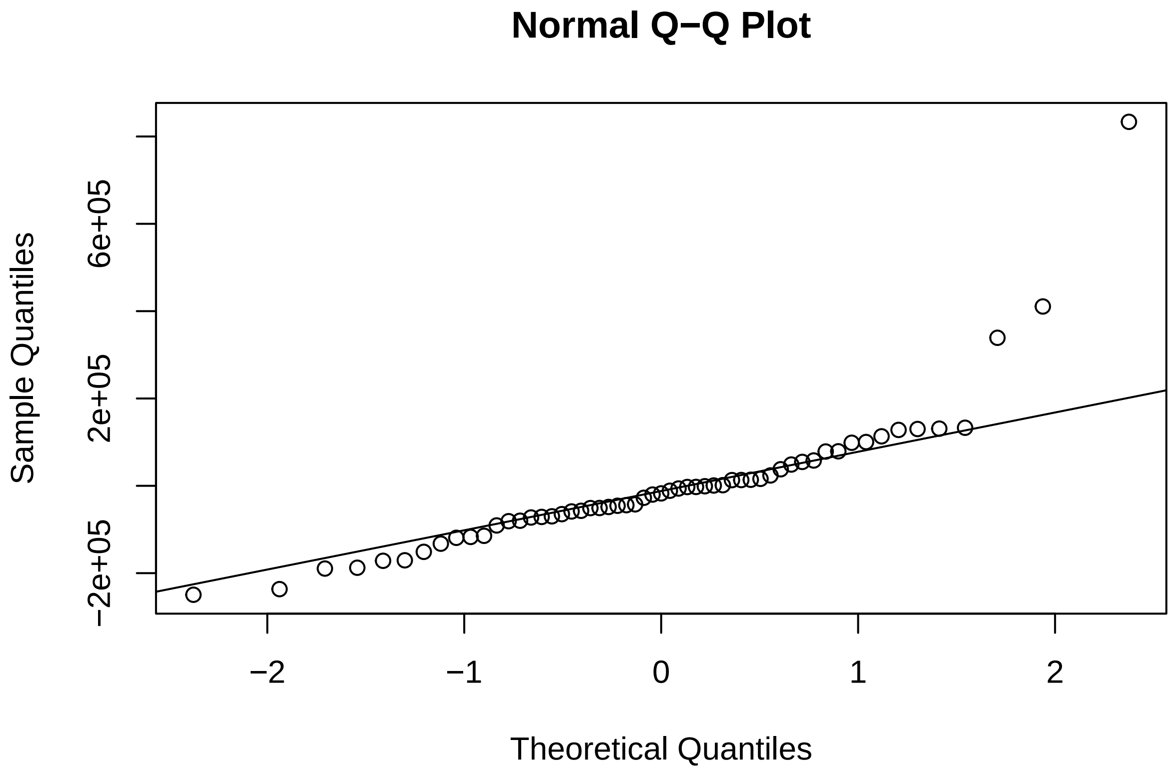
```
plot(train.ts_df, ylab="Weekly Sales", bty="l")
lines(train.lm$fitted.values, lwd=2)
```



```
train.lm.pred <- forecast(train.lm, h=nValid, level=0)
accuracy(train.lm.pred, valid.ts_df)
```

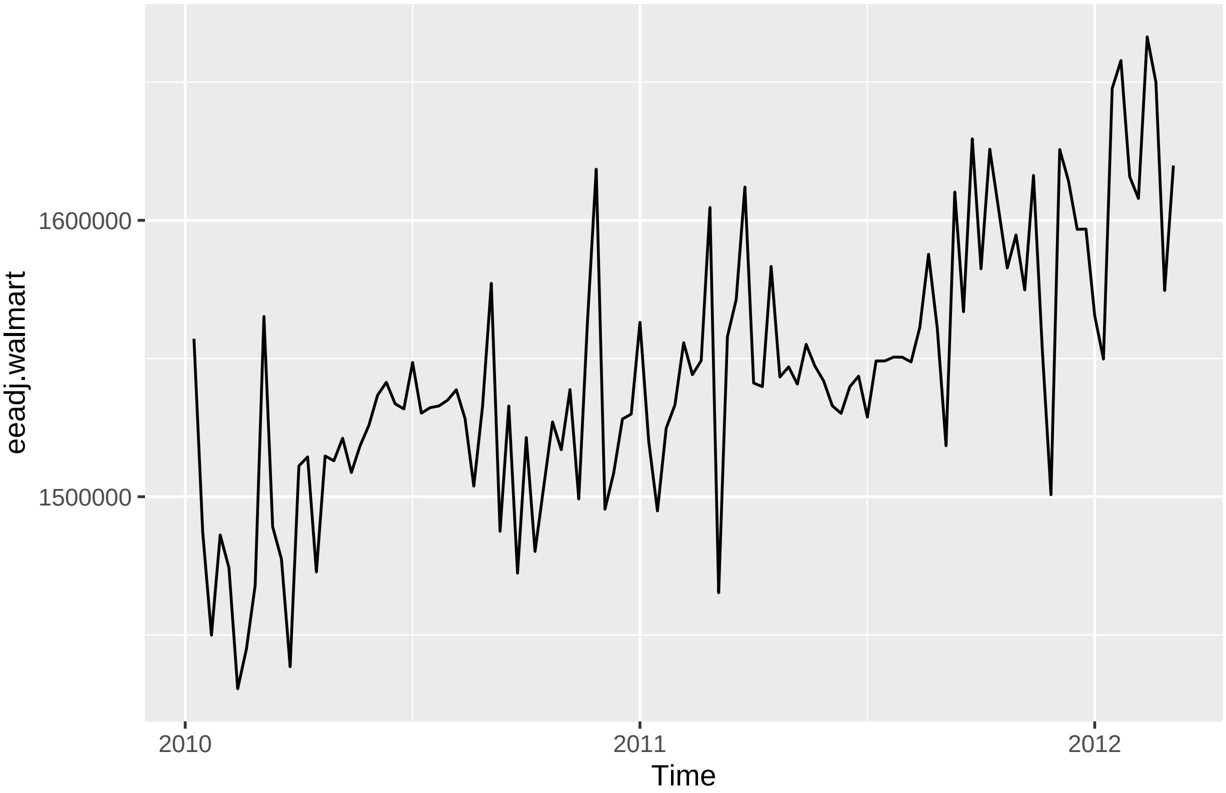
```
##                          ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -1.225300e-11 161930.0 100234.5 -0.9101114 6.285470 1.037293
## Test set     -7.516934e+04 176358.5 138132.5 -5.7385515 8.886532 1.429487
##                   ACF1 Theil's U
## Training set 0.2299189        NA
## Test set     0.2585627  0.996114
```

```
# residuals plot
qqnorm(train.lm$residuals)
qqline(train.lm$residuals)
```

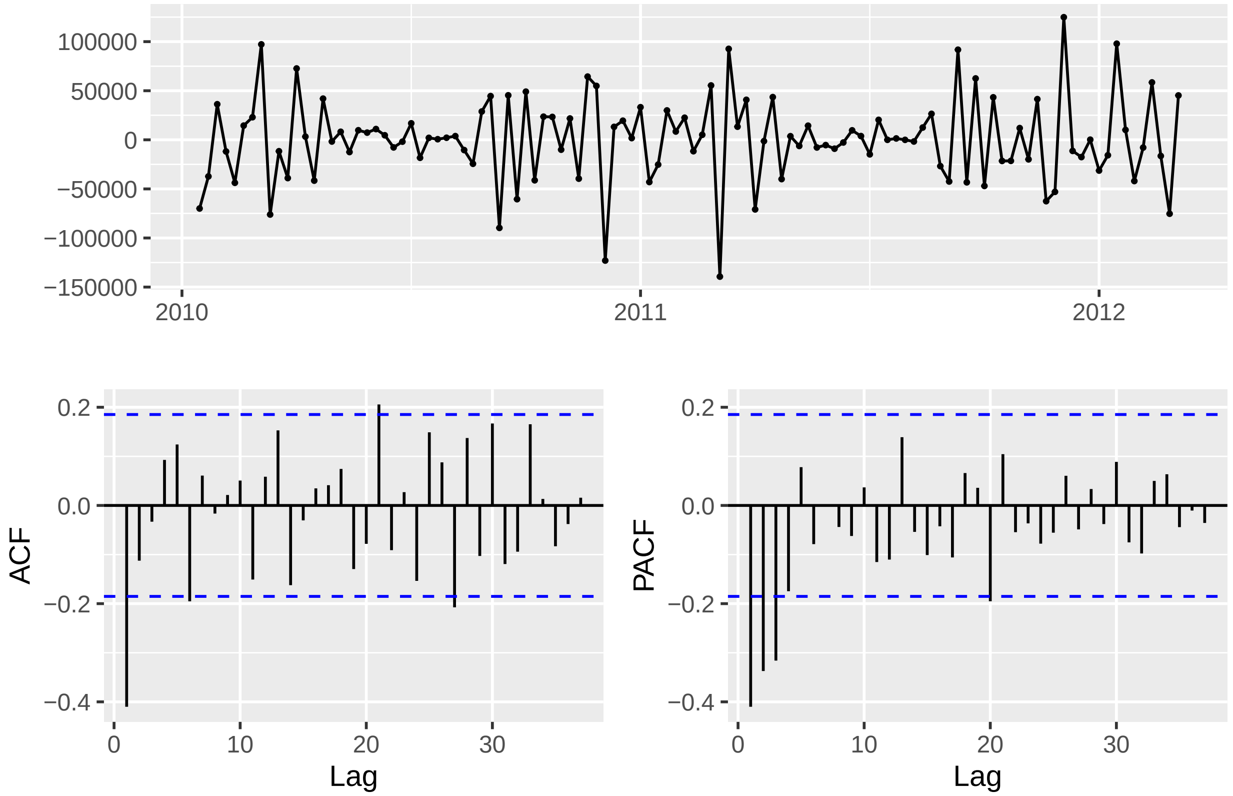


ARIMA Model

```
walmart.ts %>%
  stl(s.window='periodic') %>% seasadj() -> eeadj.walmart
autoplot(eeadj.walmart)
```



```
# Now we take a first difference of the weekly sales to remove trend and seasonality
eeadj.walmart %>%
  diff() %>%
  ggtsdisplay(main="")
```



The AR(5) model is suggested by the PACF in the above figure. An ARIMA is thus a first candidate model (5,1,0)

```
model1 <- Arima(eeadj.walmart, order=c(5,1,0))
model1
```

```
## Series: eeadj.walmart
## ARIMA(5,1,0)
##
## Coefficients:
##           ar1      ar2      ar3      ar4     ar5
##       -0.7745  -0.6860  -0.4979  -0.2046  0.0259
## s.e.   0.0978   0.1241   0.1311   0.1242  0.0992
##
## sigma^2 = 1.148e+09:  log likelihood = -1325.16
## AIC=2662.33   AICc=2663.13   BIC=2678.64
```

#arima.pred <- forecast(model1, h=nValid) #accuracy(arima.pred, valid.ts_df)

```
accuracy(model1)
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 2633.708 32975.27 24279.58 0.1294818 1.570444 0.3720368 0.01979848
```

neural network

```
model.nn <- nnetar(train.ts_df, repeats=20)
summary(model.nn$model[[1]])
```

```
## a 6-4-1 network with 33 weights
## options were - linear output units
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1 i6->h1
##   0.17  -0.05   0.01  -0.28  -0.63  -0.56   0.26
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2
##   0.01  -0.44  -0.34  -0.28  -0.60  -0.47   0.32
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3
##   0.49  -0.84   0.85   0.65   0.75   0.67   0.90
##  b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4
##   0.22  -0.64   0.19   0.68   0.15  -0.43   0.30
##  b->o h1->o h2->o h3->o h4->o
## -1.02 -0.02  0.24  1.26  0.59
```

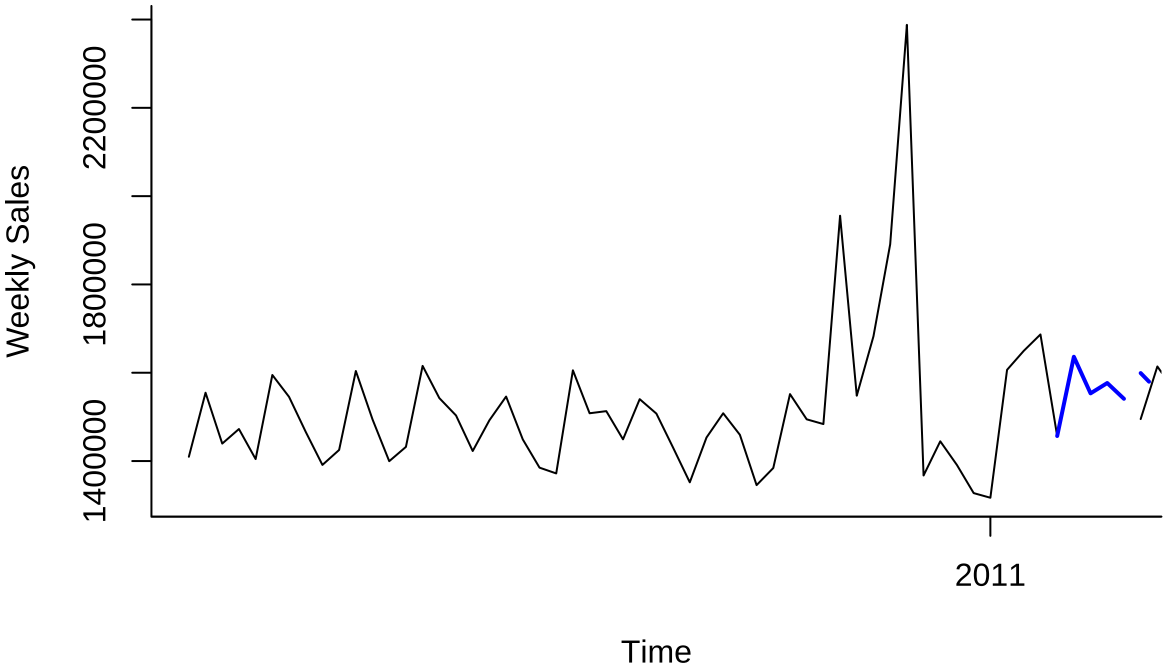
```
model.nn
```

```
## Series: train.ts_df
## Model:  NNAR(5,1,4)[52]
## Call:   nnetar(y = train.ts_df, repeats = 20)
##
## Average of 20 networks, each of which is
## a 6-4-1 network with 33 weights
## options were - linear output units
##
## sigma^2 estimated as 8657
```

```
nn.pred <- forecast(model.nn, h=nValid)
accuracy(nn.pred, valid.ts_df)
```

```
##                        ME         RMSE           MAE          MPE        MAPE
## Training set   34.09613     93.04315     88.68216  0.002211784 0.005726228
## Test set     6293.75626 146870.31223 102219.32634 -0.451415196 6.257839133
##                        MASE       ACF1 Theil's U
## Training set 0.0009177418 -0.3050343        NA
## Test set     1.0578333838  0.3576406 0.7968953
```

```
plot(train.ts_df, bty="l", xaxt="n", lty=1, ylab="Weekly Sales")
axis(1, at = seq(2005, 2015, 1), labels=format(seq(2005,2015,1)))
lines(nn.pred$fitted, lwd=2, col="blue")
lines(nn.pred$mean, lwd=2, col="blue", lty=2)
lines(valid.ts_df)
```


Weekly Sales
1400000
1800000
2200000
2011
Time