Appendix B
ADS 506 Final Project Code

```
library(astsa)
library(dplyr)
library(lubridate)
library(zoo)
library(forecast)
library(ggplot2)
library(reshape)
library(TSstudio)
library(h2o)
library(plotly)
```

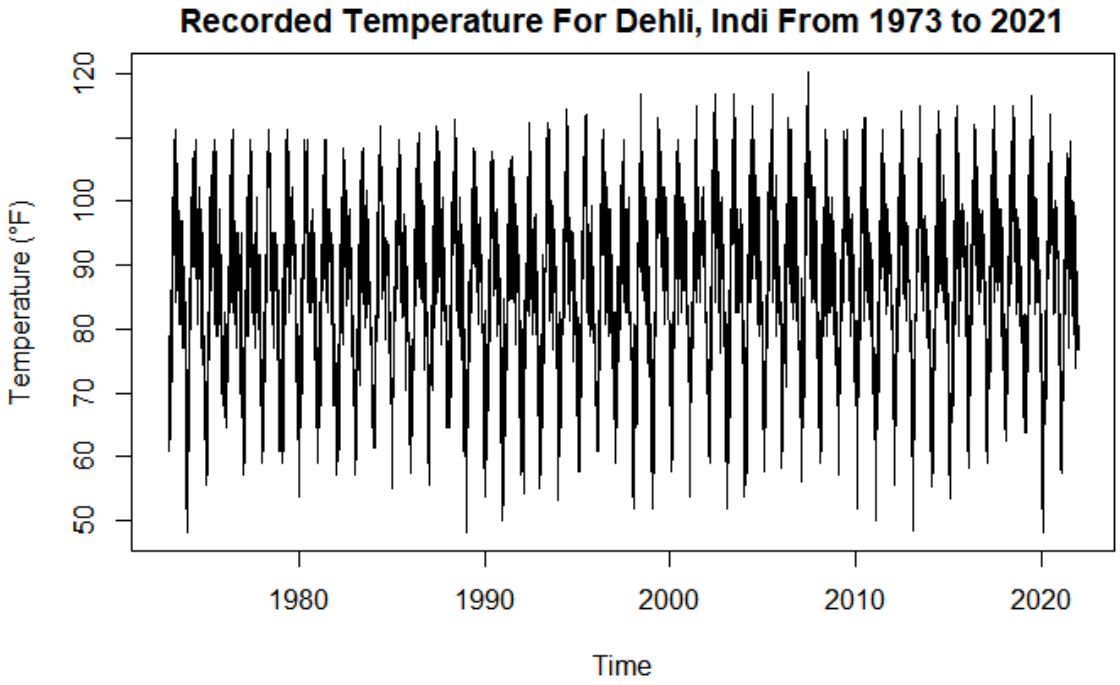
```
# Import Delhi_Weather_Data.csv
data = read.csv("F:/School/ADS/506/Final Project/dehli_weather_info.csv")
```

```
# Checking the percentage of NA values
length(which(is.na(data$tempmin))) / dim(data)[1]
length(which(is.na(data$tempmax))) / dim(data)[1]
length(which(is.na(data$temp))) / dim(data)[1]
length(which(is.na(data$humidity))) / dim(data)[1]
length(which(is.na(data$pressure))) / dim(data)[1]
length(which(is.na(data$windspeed))) / dim(data)[1]
```

```
[1] 0.0003918057
[1] 0.0003918057
[1] 0.0003918057
[1] 0.0003918057
[1] 0.0007276391
[1] 0.0003918057
```

```
# Replace NA's
# The missing values will be replaced with the average during a certain week. For
example, if a "temp" value is missing,
# the data from 3 days prior and 3 days after will be added together and then
divided by 6 to find the average "temp".

NA_replace <- function(df) {
  for(j in 1:ncol(df)){
    for(i in 1:nrow(df)){
      if(is.na(df[i,j]) == TRUE && i > 3){
        avg <- sum(df[(i-3):(i+3),j], na.rm = TRUE) / 6
        df[i,j] <- avg
      }
    }
  }
  return(df)
}
```

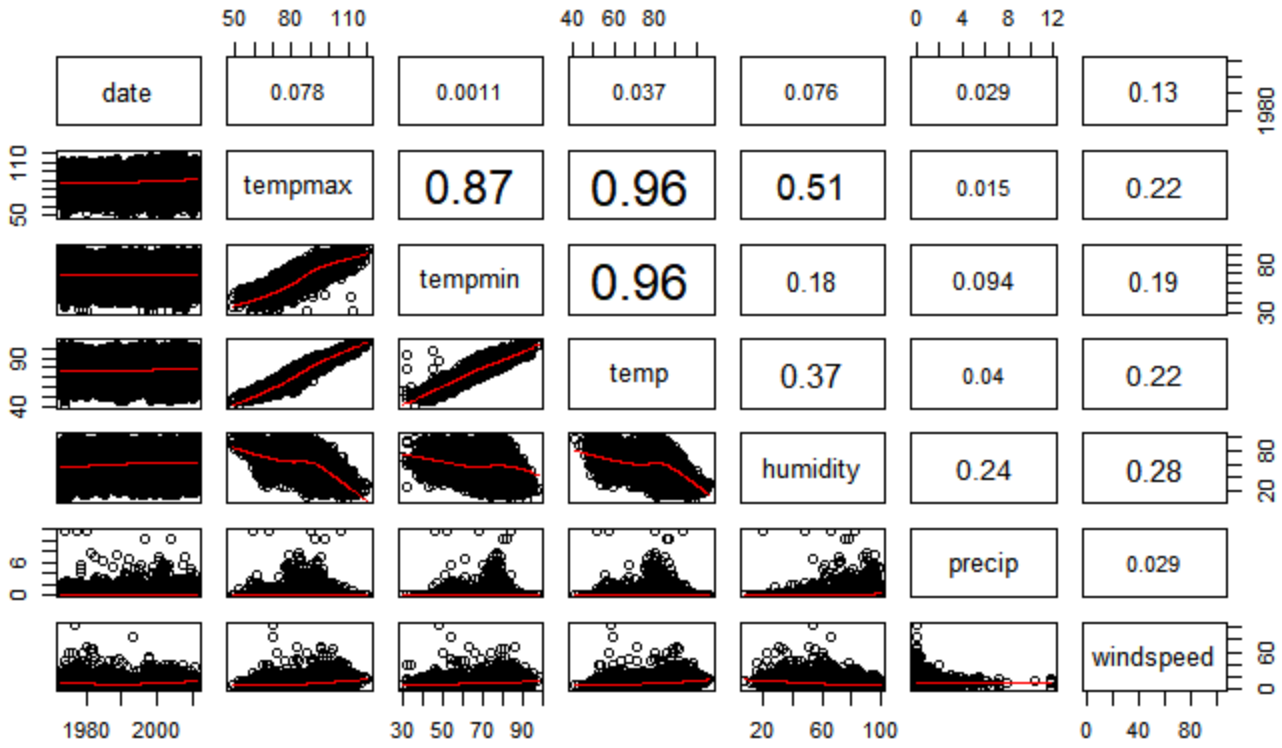
```
data = NA_replace(data)

colnames(data)[2] = 'date'

data$date = strptime(data$date, "%m/%d/%Y")
data$date = format(data$date, "%Y-%m-%d")
data$date = as.Date(data$date)


data$tempmax = ts(data = data$tempmax,
                  frequency = 365,
                  start = c(1973, yday(head(data$date,1))))

plot(data$tempmax,
     main = 'Recorded Temperature For Dehli, Indi From 1973 to 2021',
     ylab = 'Temperature (°F)',
     xlab = 'Time')
```




```
# for use later in ML model
train = subset(data, data$year <= '2010')
test = subset(data, data$year > '2010' & data$year <= '2021' )
```

```
# Checking the percentage of NA values
length(which(is.na(data$tempmin))) / dim(data)[1]
length(which(is.na(data$tempmax))) / dim(data)[1]
length(which(is.na(data$temp))) / dim(data)[1]
length(which(is.na(data$humidity))) / dim(data)[1]
length(which(is.na(data$pressure))) / dim(data)[1]
length(which(is.na(data$windspeed))) / dim(data)[1]
```

```r
# **Correlation between each feature**


  panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) {
    usr <- par("usr")
    on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    Cor <- abs(cor(x, y))
    txt <- paste0(prefix, format(c(Cor, 0.123456789), digits = digits)[1])
    if(missing(cex.cor)) {
        cex.cor <- 0.4 / strwidth(txt)
    }
    text(0.5, 0.5, txt,
         cex = 1 + cex.cor * Cor) # This will show correlation value and change the
size based on the value
}

# Plot correlation
pairs(train[, c(2:8)],
      upper.panel = panel.cor,
      lower.panel = panel.smooth)
```

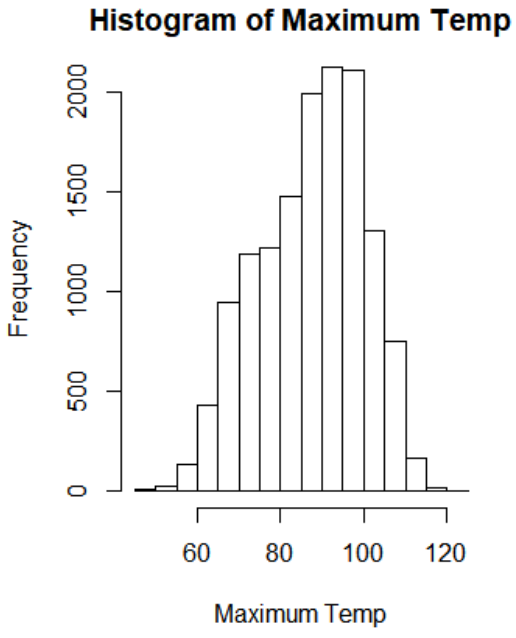


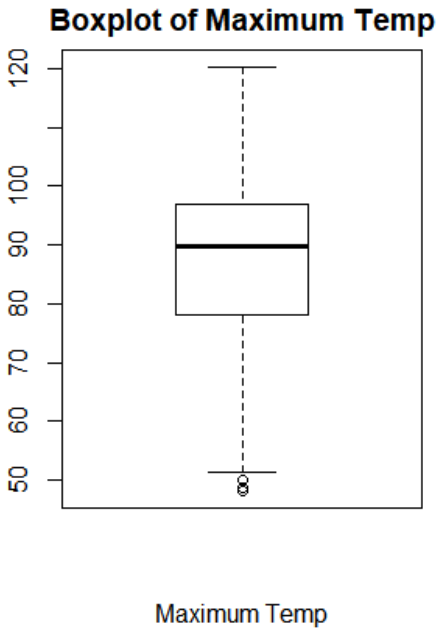
There are strong positive correlation between temp columns (min, max, mean) and pressure. The next big correlation is between humidity and temp. Precipitation and windspeed has slightly higher correlation than neutral.

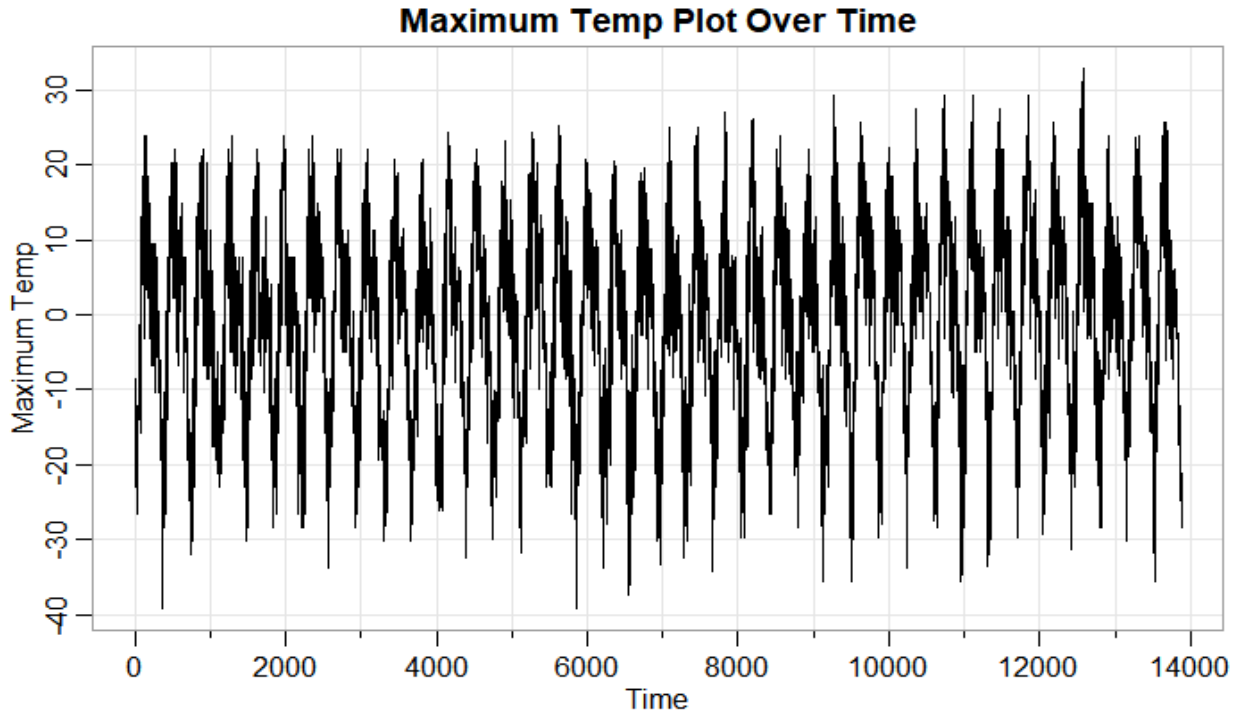
```r
visualize_feature = function(x, name) {

  # Show Histogram of feature before subtracting mean value

  par(mfrow=c(1,2))
  hist(x, main=paste('Histogram of', name), xlab=name)
  boxplot(x, main=paste('Boxplot of', name), xlab=name)

  x = x - mean(x)
  summary(x)

  par(mfrow=c(1,1))
  tsplot(x, main=paste(name, 'Plot Over Time'), ylab=name)

  acf2(x)

  best_param = auto.arima(x)
  print(best_param)

  x
}
```

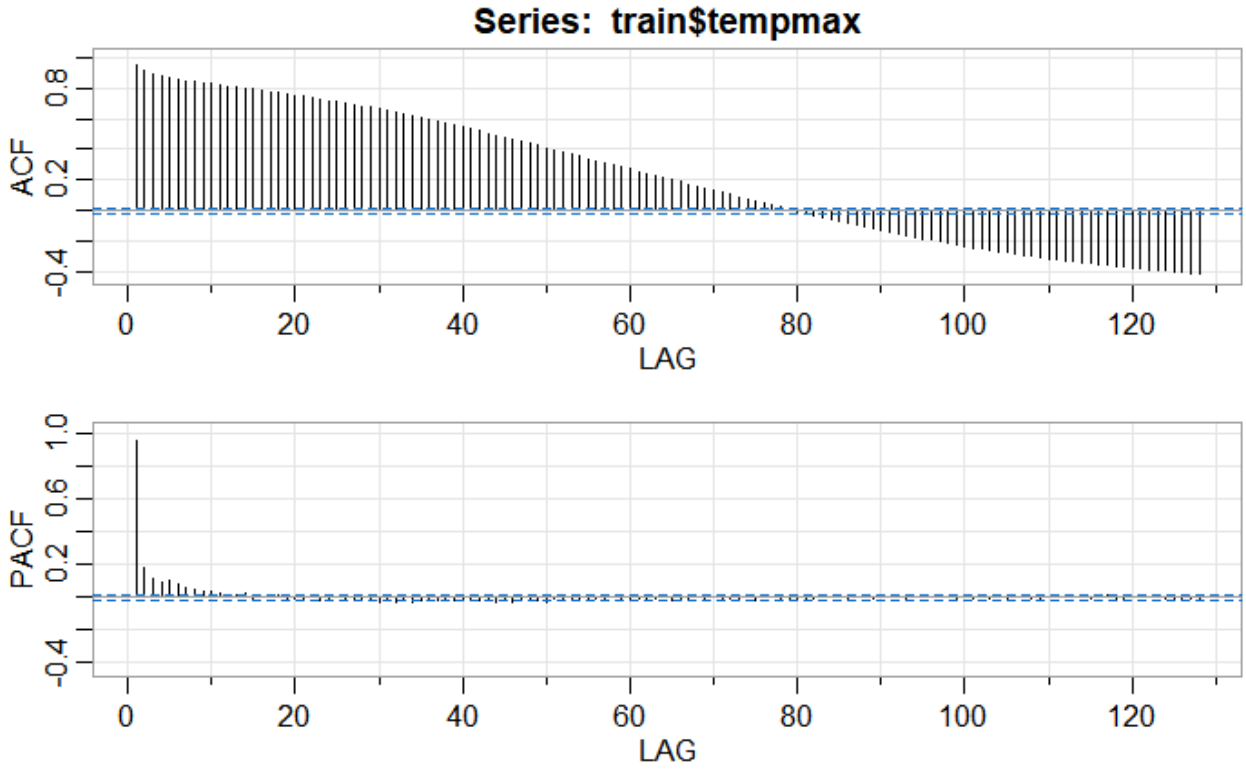
```r
# Maximum temperature

tempmax = visualize_feature(train$tempmax, 'Maximum Temp')
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-39.117	-9.217	2.283	0.000	9.483	32.883



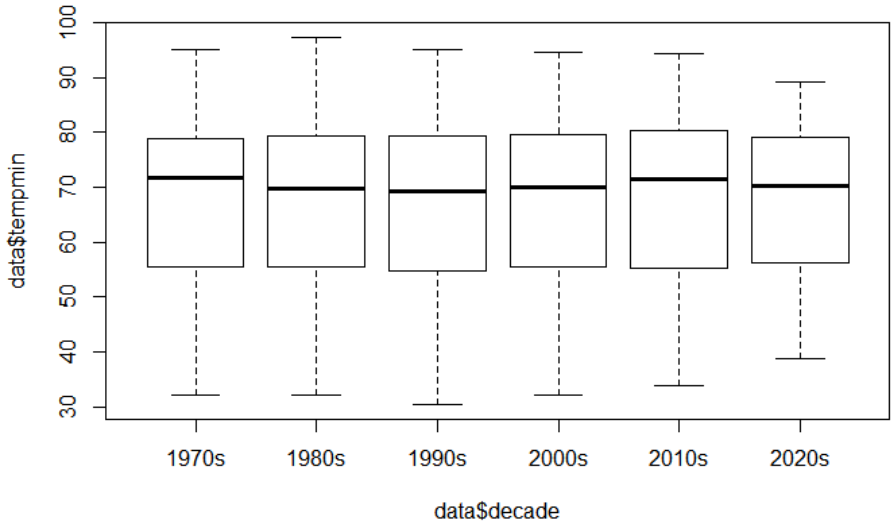


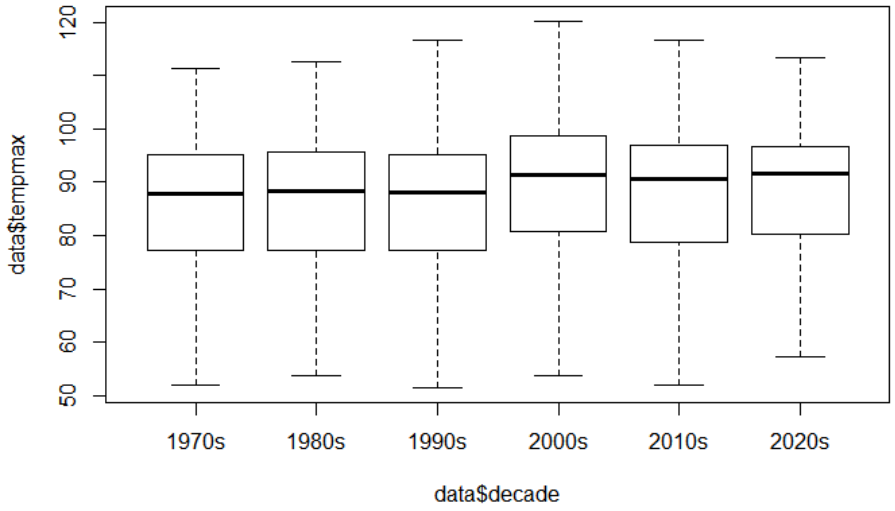


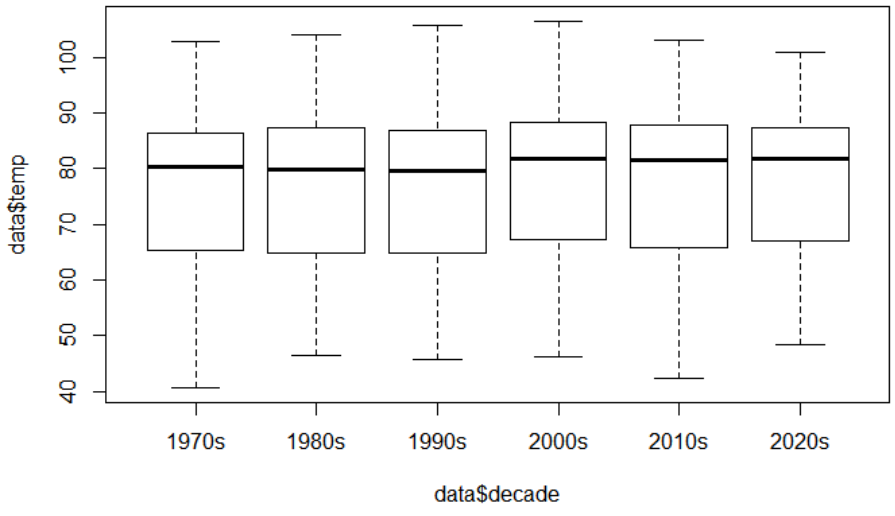



```
Series: y_train
ARIMA(5,0,1)(0,1,0)[365]

Coefficients:
         ar1      ar2      ar3      ar4     ar5      ma1
      1.3094  -0.3890  -0.0142  -0.0156  0.0307  -0.6460
s.e.  0.0984   0.0668   0.0151   0.0143  0.0112   0.0983

sigma^2 estimated as 26.72:  log likelihood=-41373.25
AIC=82760.49   AICc=82760.5   BIC=82813.08

Training set error measures:
                     ME     RMSE      MAE        MPE     MAPE     MASE          ACF1
Training set 0.01179757 5.099908 3.767222 -0.1716369 4.444871 0.660315 0.0002324551
```

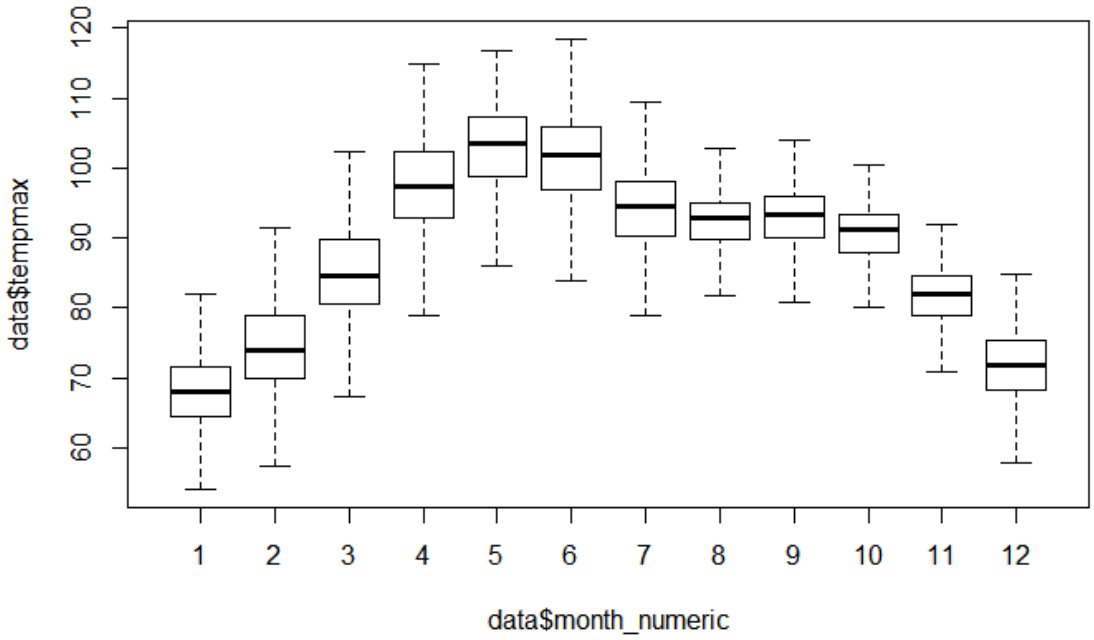
```r
boxplot(data$tempmin ~ data$decade, outline = FALSE)

boxplot(data$tempmax ~ data$decade, outline = FALSE)

boxplot(data$temp ~ data$decade, outline = FALSE)
```

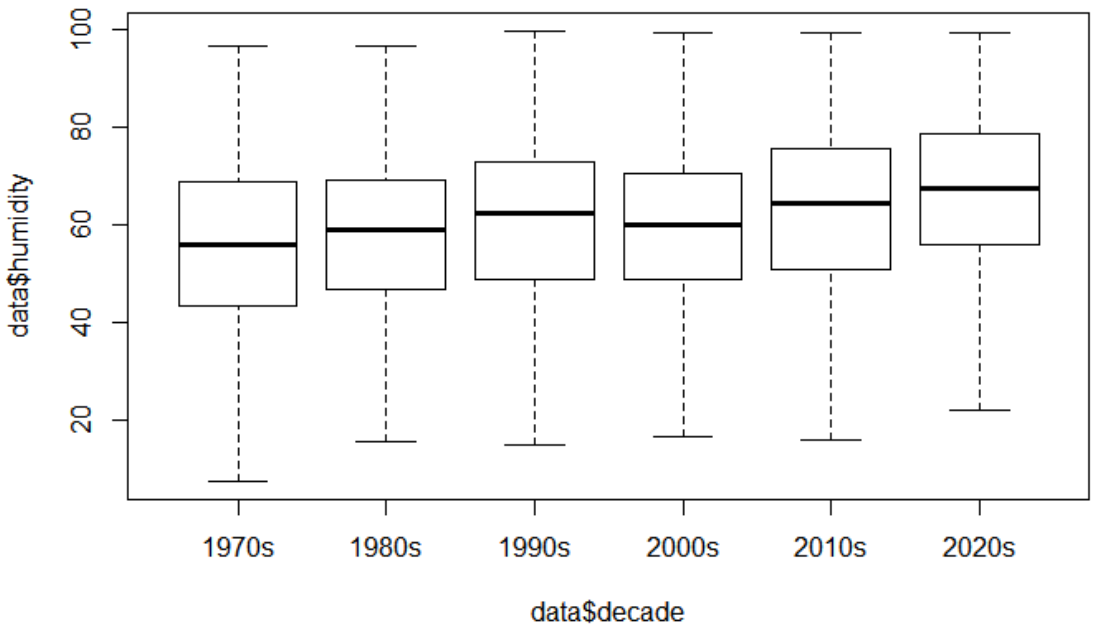


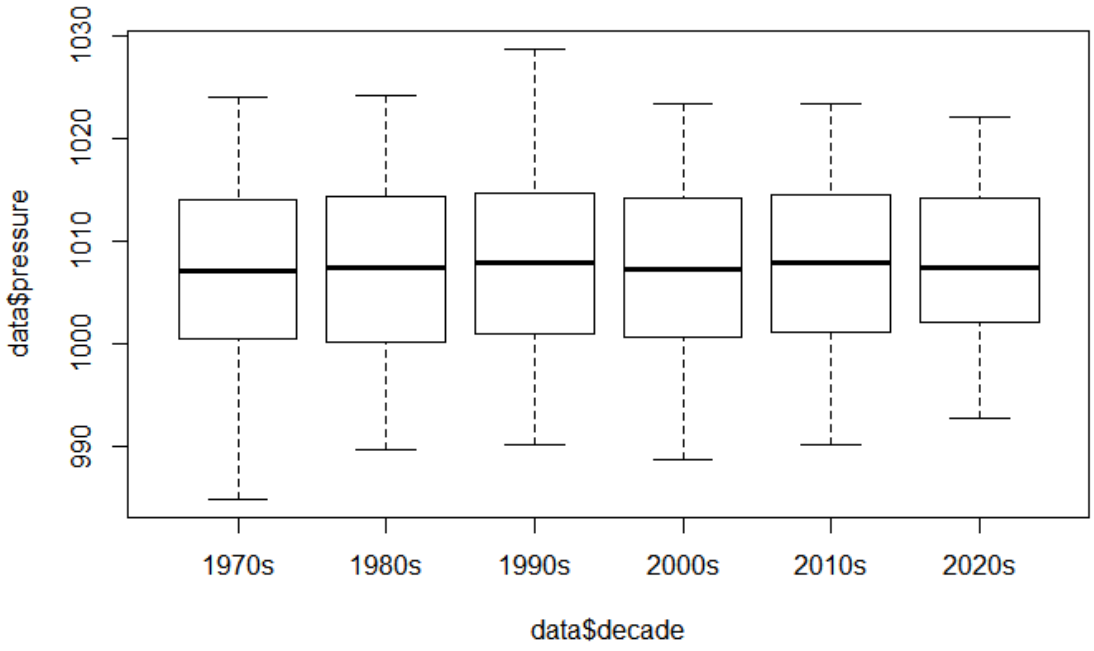


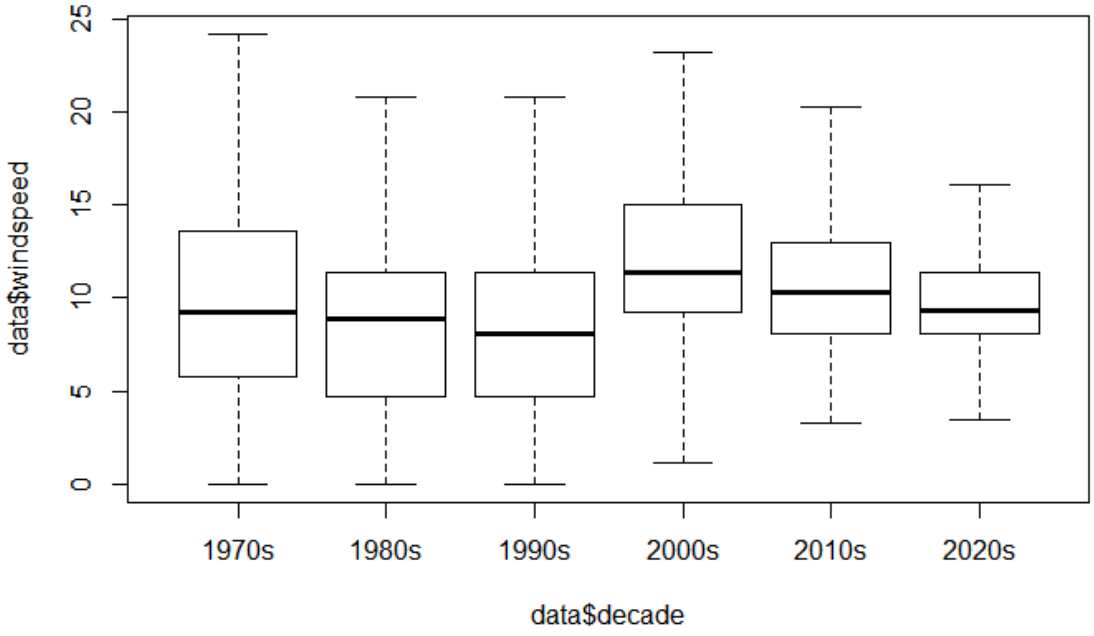


```r
boxplot(data$tempmax ~ data$month_numeric, outline = FALSE)

boxplot(data$humidity ~ data$decade, outline = FALSE)

boxplot(data$pressure ~ data$decade, outline = FALSE)

boxplot(data$windspeed ~ data$decade, outline = FALSE)
```

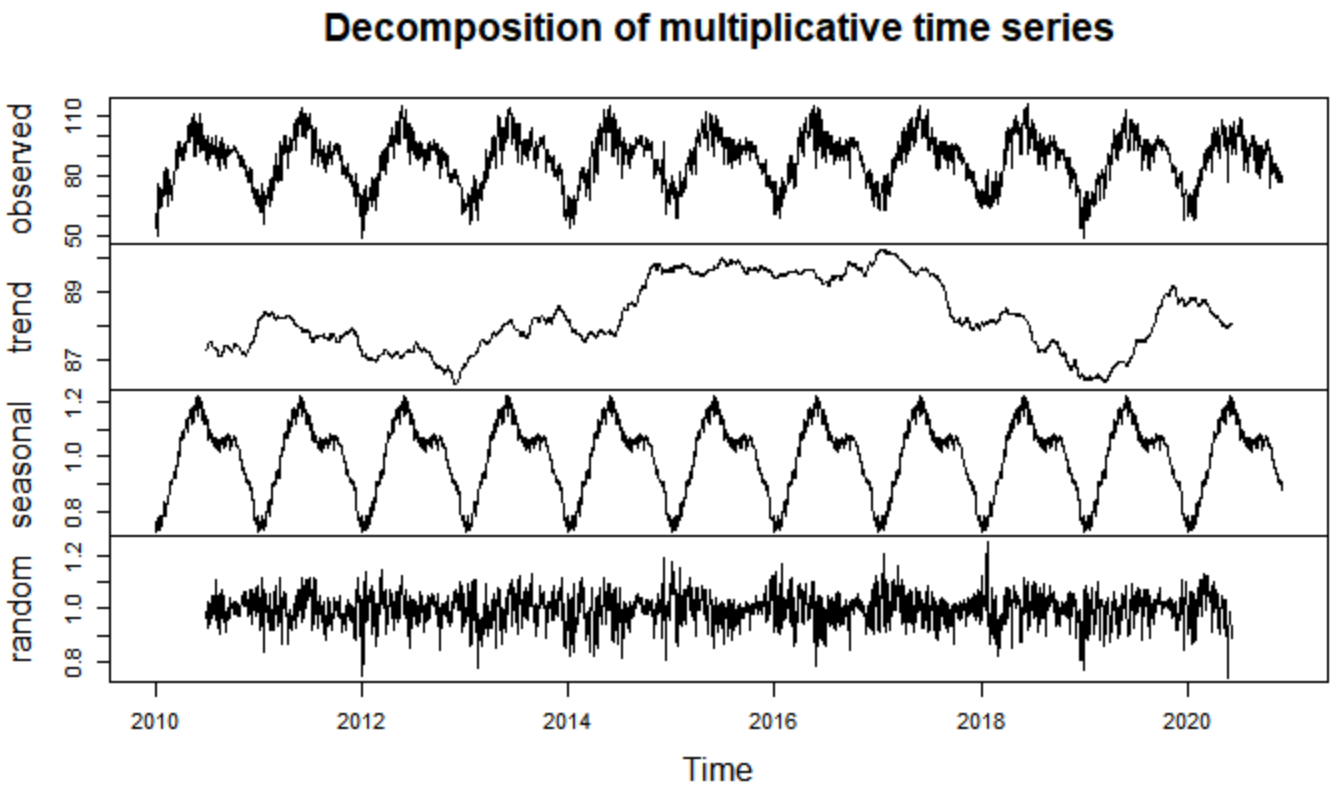





data$pressure
990
1000
1010
1020
1030
1970s
1980s
1990s
2000s
2010s
2020s
data$decade


data$windspeed
0
5
10
15
20
25
1970s
1980s
1990s
2000s
2010s
2020s
data$decade

```r
tempmax_ts = ts(data = test$tempmax,
                frequency = 365,
                start = c(2010, yday(head(test$date,1))))

# Show Classical Seasonal Decomposition by Moving Averages
decomposition = decompose(tempmax_ts, type = 'multiplicative')

plot(decomposition)
```

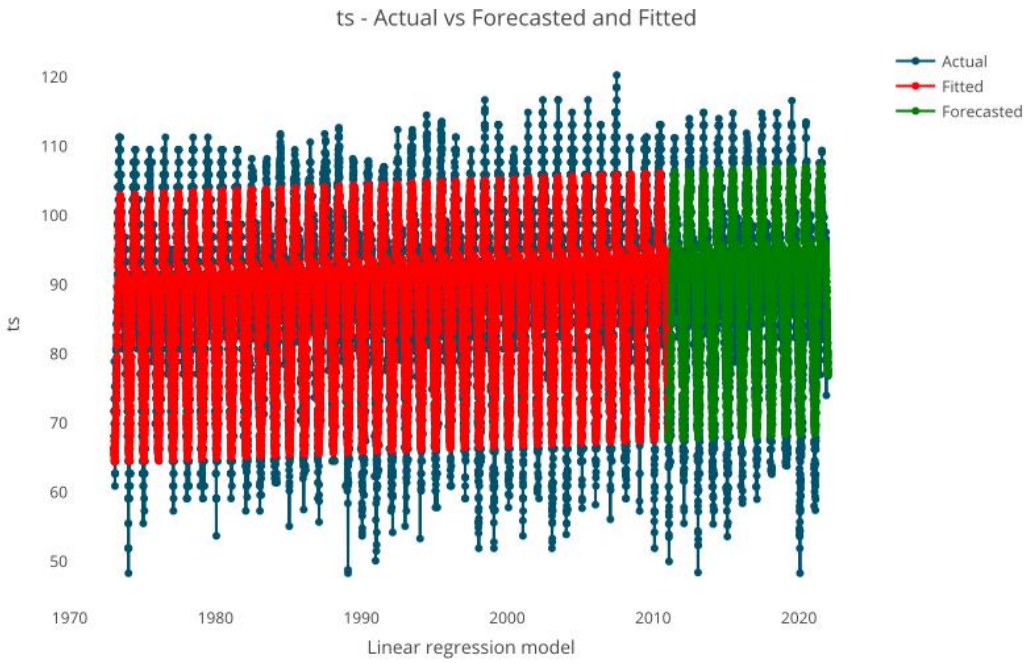


```r
# Moldeing Setup
h = nrow(test)

ts = na.locf(data$tempmax)
ts_par = ts_split(ts, sample.out = h)
y_train = ts_par$train
y_test = ts_par$test

# Training Model
md_tslm = tslm(y_train ~ season + trend)
hw_model = HoltWinters(y_train, seasonal = 'multiplicative')
arima_model = auto.arima(y_train)
```

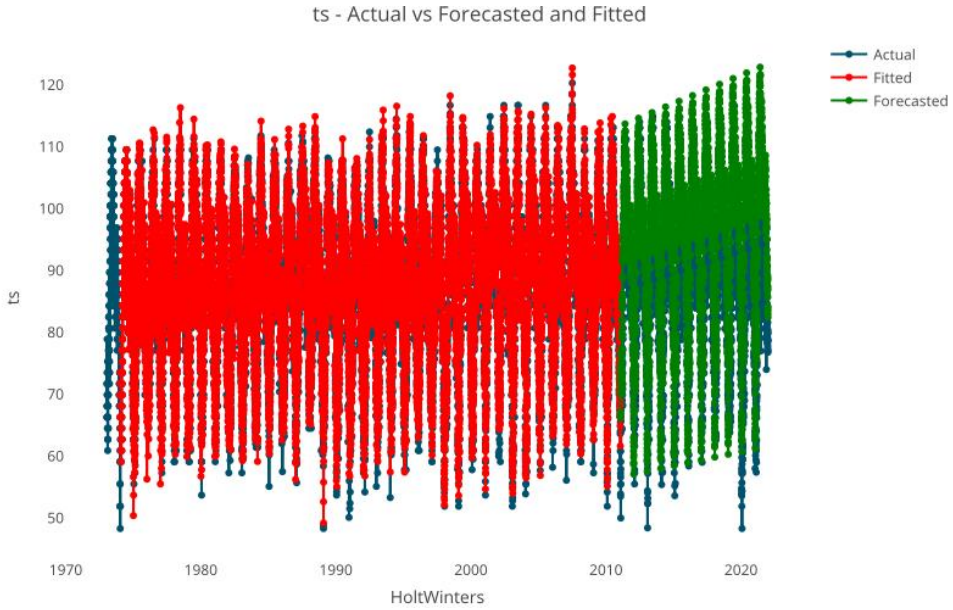
```r
# Forcast Portion
fc_tslm = forecast(md_tslm, h = h)
fc_hw = forecast(hw_model, h = h)
fc_arima = forecast(arima_model, h = h)
```

```r
# Plot
test_forecast(actual = ts,forecast.obj = fc_tslm, test = y_test)
test_forecast(actual = ts,forecast.obj = fc_hw,   test = y_test)
test_forecast(actual = ts,forecast.obj = fc_arima,test = y_test)
```

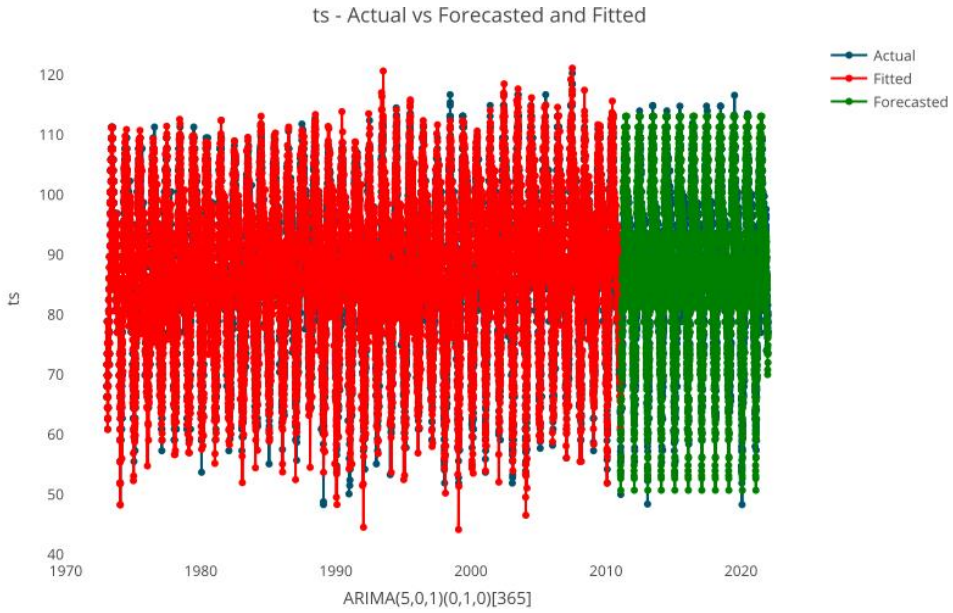
Plot 1
Linear Model



Plot 2
Holt Winters



Plot 3
SARIMA (5 , 0 , 1 , 0 , 1 , 0)



```r
# Checking Performance Of Each Model
accuracy(fc_tslm, y_test)
accuracy(fc_hw, y_test)
accuracy(fc_arima, y_test)
```

```
                      ME     RMSE      MAE        MPE     MAPE      MASE      ACF1 Theil's U
Training set -1.001005e-16 5.244222 4.051048 -0.3883931 4.777445 0.7100637 0.7199728        NA
Test set     -1.352905e+00 5.420191 4.124090 -2.0379698 4.967317 0.7228663 0.7220514  1.468115
```

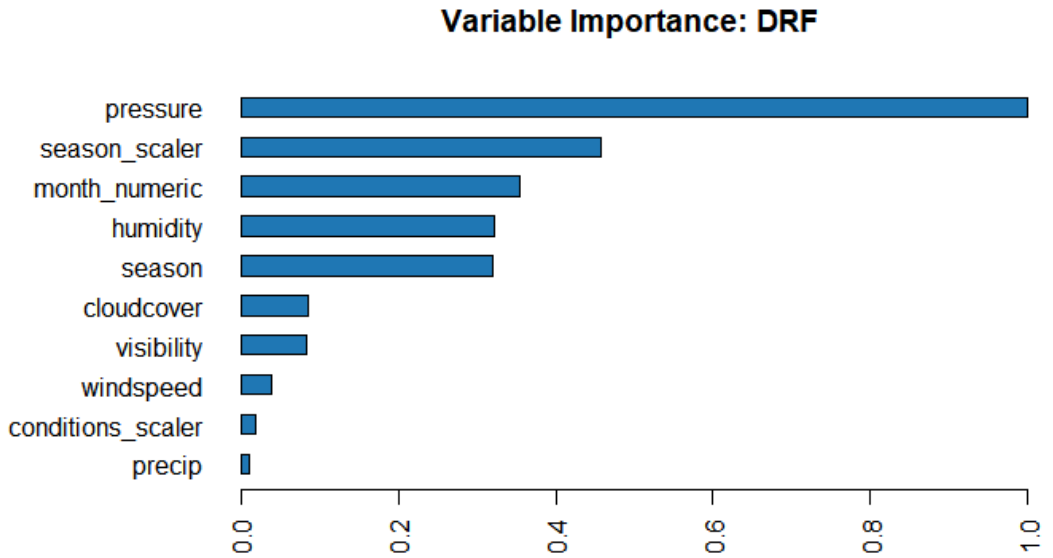
```
                    ME      RMSE      MAE        MPE     MAPE      MASE      ACF1 Theil's U
Training set -0.0276418  4.350542 3.245102 -0.1937894 3.849248 0.5687983 0.1625329        NA
Test set     -7.1987998 10.282641 8.248469 -8.5142413 9.768979 1.4457835 0.8363367  2.684651
```

```
                     ME      RMSE      MAE        MPE     MAPE     MASE          ACF1 Theil's U
Training set  0.01179757  5.099908 3.767222 -0.1716369 4.444871 0.660315 0.0002324551        NA
Test set     -0.34436945 10.171569 8.291481 -0.9856161 9.875336 1.453323 0.8580748692  2.710774
```

```
h2o.init(max_mem_size = "16G")


train_h = as.h2o(train)
test_h <- as.h2o(test)


x <-
c('humidity','precip','windspeed','pressure','visibility','cloudcover','conditions_s
caler','season','season_scaler', 'month_numeric')
y <- "tempmax"

rf_md <- h2o.randomForest(training_frame = train_h,
                          nfolds = 5,
                          x = x,
                          y = y,
                          ntrees = 100,
                          stopping_rounds = 10,
                          stopping_metric = "RMSE",
                          score_each_iteration = TRUE,
                          stopping_tolerance = 0.0001,
                          seed = 1234)

h2o.varimp_plot(rf_md)
tree_score <- rf_md@model$scoring_history$training_rmse
plot_ly(x = seq_along(tree_score), y = tree_score,
        type = "scatter", mode = "line") %>%
  layout(title = "The Trained Model Score History",
         yaxis = list(title = "RMSE"),
         xaxis = list(title = "Num. of Trees"))

x = c('month_numeric','year','season_scaler')
y = "tempmax"
```

```
rf_md = h2o.randomForest(training_frame = train_h,
                         nfolds = 5,
                         x = x,
                         y = y,
                         ntrees = 100,
                         stopping_rounds = 10,
                         stopping_metric = "RMSE",
                         score_each_iteration = TRUE,
                         stopping_tolerance = 0.0001,
                         seed = 1234)

test_h$pred_rf = h2o.predict(rf_md, test_h)

test_1 <- as.data.frame(test_h)
mape_rf <- mean(abs(test_1$tempmax - test_1$pred_rf) / test_1$tempmax)
mape_rf
```

```r
gbm_md <- h2o.gbm(
  training_frame = train_h,
  nfolds = 5,
  x = x,
  y = y,
  max_depth = 20,
  distribution = "gaussian",
  ntrees = 500,
  learn_rate = 0.1,
  score_each_iteration = TRUE
)

test_h$pred_gbm <- h2o.predict(gbm_md, test_h)
test_1 <- as.data.frame(test_h)
mape_gbm <- mean(abs(test_1$tempmax - test_1$pred_gbm) / test_1$tempmax)
mape_gbm

mape_hw <- mean(abs(test_1$tempmax - fc_hw$mean) / test_1$tempmax)
mape_hw
```



```
plot_ly(data = test_1) %>%
  add_lines(x = ~ test_1$date, y = ~ test_1$tempmax,
  name = "Actual",color = I("gray")) %>%
  add_lines(x = ~ test_1$date, y = ~ test_1$pred_rf,
  name = "Random Forest", color = I("red")) %>%
  add_lines(x = ~ test_1$date, y = ~ test_1$pred_gbm,
  name = "Gradient Boosting Machine",color = I("blue")) %>%
  layout(title = "Max Temperature (°F) - Actual vs. Prediction (Gradient Boosting
  Machine & Random Forest)", yaxis = list(title = "Temperature (°F)"), xaxis =
  list(title = "Month"))
```