MSDS 6306: Project 2 - Predicting and Forecasting Bike Sharing Data

Swee K Chew, Mallory Hightower

08/16/2018

Introduction

Bike Sharing, a system for bike rentals, has become increasingly popular in cities across the United States. Bike sharing enables people to locate a bike, rent the bike for transportation to their destination, and then leave the bike at the destination. This aspect of finding a nearby bike and leaving it when you don't need it anymore is central to the Bike Sharing model because it makes the bikes extremely convenient for the user. Compared to alternative rideshare options, such as Uber or Lyft, Bike Sharing is less expensive and in large, high traffic cities, bikes often prove to be a faster form of transportation. Bike Sharing is also an environmentally friendly form of transportation and lends itself well to leisurely activities, such as a Sunday bike ride with friends.

Because the Bike Sharing companies must place a certain number of bikes around cities to remain extremely conventinet to the user, a big question for these companies centers on the demand for the bikes and how many bikes should be placed around the city at any given time. The UCI Machine Learning Repository has Bike Sharing data that will be used in this analysis to help answer the question. Next question to ask is 'Could the bike demand be forecasted using the historical data?'. To answer this question, we will use the raw bike trip data from the source's databank.

In the first part of the project, we will examine the Bike Sharing data and build a random forest model to predict the number of bikes that will be rented at any given time and therefore, estimate bike demand. The model will also tell us which predictors are important in predecting the bike counts.

Then in the second part of the project, a time series forecasting will be performed using the monthly data from 2011 to 2016 to forecast 2017 bike rentals. Three forecasting methods will be applied to fit the models and criteria such as AIC, RMSE and residuals will be used to assess which method performs the best.

Data Description

The Data was obtained from the UCI Machine Learning Repository[1]. The dataset contains the daily bike rental information from metro DC's bikeshare system named Capital Bikeshare. The data is collected in 2010 and 2011, and it consists of 731 observations and 16 variables.

Part 1 - Random Forest

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(caret)
```

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

[1] https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset

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

```
##
## Attaching package: 'ggplot2'
```

```
## The following object is masked from 'package:randomForest':
##
##     margin
```

Data Preparation

```
bike_data <- read.csv('bike_day_data.csv')
head(bike_data)
```

```
##   instant     dteday season yr mnth holiday weekday workingday weathersit
## 1       1 2011-01-01      1  0    1       0       6          0          2
## 2       2 2011-01-02      1  0    1       0       0          0          2
## 3       3 2011-01-03      1  0    1       0       1          1          1
## 4       4 2011-01-04      1  0    1       0       2          1          1
## 5       5 2011-01-05      1  0    1       0       3          1          1
## 6       6 2011-01-06      1  0    1       0       4          1          1
##       temp    atemp      hum windspeed casual registered  cnt
## 1 0.344167 0.363625 0.805833 0.1604460    331        654  985
## 2 0.363478 0.353739 0.696087 0.2485390    131        670  801
## 3 0.196364 0.189405 0.437273 0.2483090    120       1229 1349
## 4 0.200000 0.212122 0.590435 0.1602960    108       1454 1562
## 5 0.226957 0.229270 0.436957 0.1869000     82       1518 1600
## 6 0.204348 0.233209 0.518261 0.0895652     88       1518 1606
```

The variable *instant* is removed since it's a record index which is the count of observation. *dteday* is also removed since it represents the rental date and year and month variables are already present in the dataset. *casual* and *registered*, which sum up to *cnt* are also discarded since they are highly correlated with the response variable.

```
data <- bike_data[ ,c(-1,-2,-14,-15)]
sum(is.na(data))
```

```
## [1] 0
```

The cleaned data is then split into training and test data sets using a random sampling method, and subsets are created to assign features and response variable labels in order to fit a random forest model.

```
sample <- sample.int(n = nrow(data), size = floor(.9*nrow(data)), replace = F)
train_data <- data[sample, ]
test_data  <- data[-sample, ]

train_X <- subset(train_data, select=-which(names(data) == "cnt"))
train_y <- subset(train_data, select=which(names(data) == "cnt"))
test_X <- subset(test_data, select=-which(names(data) == "cnt"))
test_y <- subset(test_data, select=which(names(data) == "cnt"))
dim(train_X)
```

```
## [1] 657  11
```

```
dim(train_y)
```

```
## [1] 657   1
```

```
dim(test_X)
```

```
## [1] 74 11
```

```
dim(test_y)
```

```
## [1] 74  1
```

To find out how much variance is explained using Random Forest Regressor, 5-fold cross validation is performed on the training data. The output below indicates that ~88% variation is explained by the model.

```
control <- trainControl(method="repeatedcv", number=5, repeats=1)
cv_model <- train(cnt ~ ., data=train_data, method="rf", tuneGrid=data.frame(mtry=5),
            trControl=control)
print(cv_model)
```

```
## Random Forest
##
## 657 samples
##  11 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 525, 525, 525, 527, 526
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   673.1419  0.880957  465.4341
##
## Tuning parameter 'mtry' was held constant at a value of 5
```

Fitting the Random Forest Model

We fit a random forest model using all the samples from the training dataset and predict the response variable, *cnt*, on the test dataset to examine how well the model performs. The predicted and the actual values are plotted for comparsion.

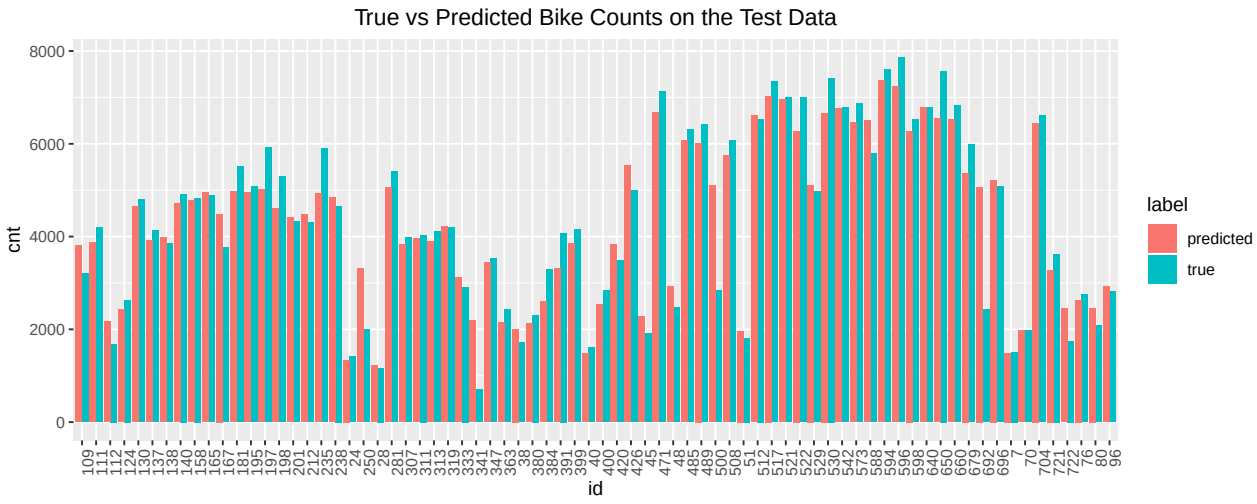
```
rf.model <- randomForest(cnt ~ . , data = train_data)
rf.model
```

```
##
## Call:
##  randomForest(formula = cnt ~ ., data = train_data)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 446274.4
##                     % Var explained: 88.11
```

```
y_hat <- predict(rf.model, test_X)
results <- data.frame(cnt=y_hat, label='predicted', id=row.names(test_X))
results <- rbind(results, data.frame(cnt=test_y, label='true', id=row.names(test_X)))
```

```
library(ggplot2)
ggplot(data=results, aes(x=id, y=cnt, fill=label)) +
ggtitle("True vs Predicted Bike Counts on the Test Data") +
```

```
geom_bar(stat="identity", position=position_dodge()) +
theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
theme(plot.title = element_text(hjust = 0.5))
```

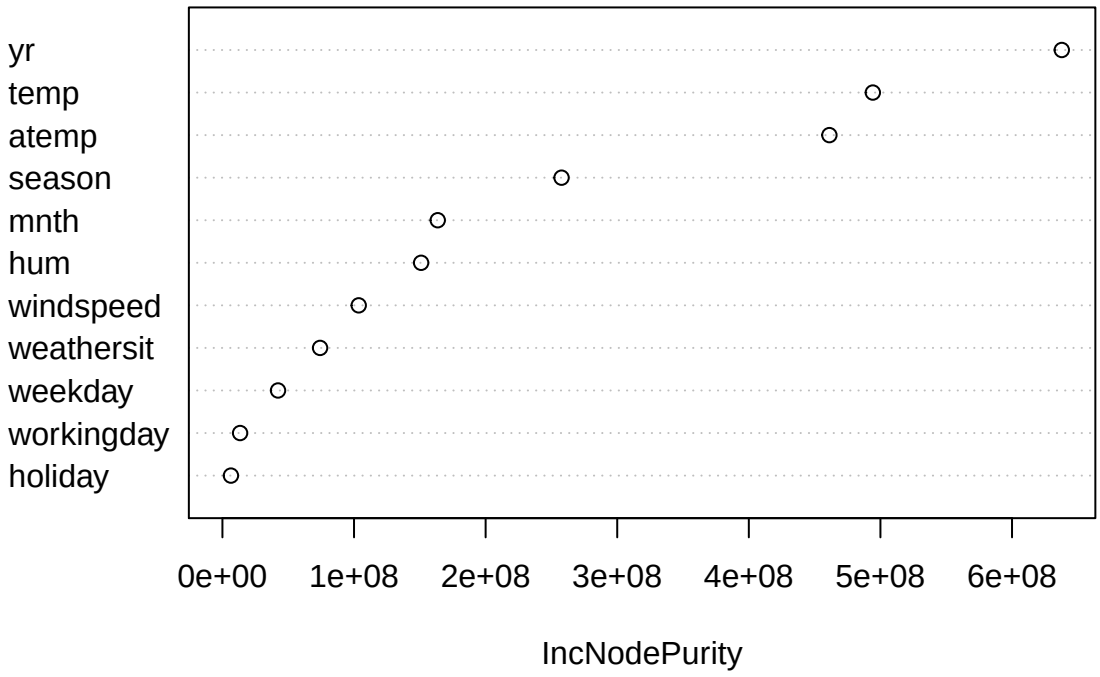


Finally, the variable importance plot is used to evaluate the importance of certain variables when predicting *cnt*. The plot below shows that *yr* (year), *atemp* (normalized feeling temperature), and *temp* (normalized temperature) are highly important variables in predicting *cnt* (count of total rental bikes).

```
importance(rf.model)
```

```
##            IncNodePurity
## season          257682810
## yr              637811835
## mnth            163602541
## holiday           6457885
## weekday          42248625
## workingday       13419165
## weathersit       74192004
## temp            494228294
## atemp           461216947
## hum             150984980
## windspeed       103555461
```

```
varImpPlot(rf.model,type=2,main="Variable Importance")
```



Conclusion

The resulting random forest model can be very helpful for bike companies to predict the number of bikes that should be placed around the city. The suppliers then can meet the demand of their consumers and thus lead a more profitable and efficient business.

Part 2 - Time Series Forecasting

From the variable importance plot in the previous section, *yr* (year) seems to be the most important variable in predicting rental bike counts. Thus, we would like to find out whether time alone can be used to predict and forecast the bike demand. The following three time series forecasting methods will be used to predict 2017 bike counts:

- Holt-Winters' Additive Model
- Holt-Winters' Multiplicative Model
- Holt-Winters' Damped Multiplicative Model

The raw individual bike trip data files are obtained from the data source[2] to perform forecasting analyses. The data are available from 2011 to 2017 in quarterly data files.

Other research questions are as follow:

- Can time alone be used to predict and forecast the bike rental counts in 2017?
- How close are the predictions to the actual bike rental counts?
- Which forecasting method performs the best?

[2]https://s3.amazonaws.com/capitalbikeshare-data/index.html

Data Preparation

For 2011, the trip data are all in one file and for 2012-2017, the data are available in 4 quarterly files. The quarterly files are extracted separately and combined into a yearly data file.

```
# Extracting 2011 data from the source data
bike_2011 <- read.csv("2011-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")

# Extracting 2012-2017 data from the source data
bike_2012Q1 <- read.csv("2012Q1-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2012Q2 <- read.csv("2012Q2-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2012Q3 <- read.csv("2012Q3-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2012Q4 <- read.csv("2012Q4-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2012 <- rbind(bike_2012Q1, bike_2012Q2, bike_2012Q3, bike_2012Q4)

bike_2013Q1 <- read.csv("2013Q1-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2013Q2 <- read.csv("2013Q2-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2013Q3 <- read.csv("2013Q3-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2013Q4 <- read.csv("2013Q4-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2013 <- rbind(bike_2013Q1, bike_2013Q2, bike_2013Q3, bike_2013Q4)

bike_2014Q1 <- read.csv("2014Q1-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2014Q2 <- read.csv("2014Q2-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2014Q3 <- read.csv("2014Q3-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2014Q4 <- read.csv("2014Q4-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2014 <- rbind(bike_2014Q1, bike_2014Q2, bike_2014Q3, bike_2014Q4)

bike_2015Q1 <- read.csv("2015Q1-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2015Q2 <- read.csv("2015Q2-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2015Q3 <- read.csv("2015Q3-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2015Q4 <- read.csv("2015Q4-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2015 <- rbind(bike_2015Q1, bike_2015Q2, bike_2015Q3, bike_2015Q4)

bike_2016Q1 <- read.csv("2016Q1-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2016Q2 <- read.csv("2016Q2-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2016Q3 <- read.csv("2016Q3-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2016Q4 <- read.csv("2016Q4-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2016 <- rbind(bike_2016Q1, bike_2016Q2, bike_2016Q3, bike_2016Q4)

bike_2017Q1 <- read.csv("2017Q1-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2017Q2 <- read.csv("2017Q2-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2017Q3 <- read.csv("2017Q3-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2017Q4 <- read.csv("2017Q4-capitalbikeshare-tripdata.csv", header = TRUE, sep = ",")
bike_2017 <- rbind(bike_2017Q1, bike_2017Q2, bike_2017Q3, bike_2017Q4)
head(bike_2017)
```

```
##   Duration          Start.date            End.date Start.station.number
## 1      221 2017-01-01 00:00:41 2017-01-01 00:04:23                31634
## 2     1676 2017-01-01 00:06:53 2017-01-01 00:34:49                31258
## 3     1356 2017-01-01 00:07:10 2017-01-01 00:29:47                31289
## 4     1327 2017-01-01 00:07:22 2017-01-01 00:29:30                31289
## 5     1636 2017-01-01 00:07:36 2017-01-01 00:34:52                31258
## 6     1603 2017-01-01 00:08:11 2017-01-01 00:34:55                31258
##                              Start.station End.station.number
## 1                         3rd & Tingey St SE              31208
```

```
## 2                            Lincoln Memorial             31270
## 3 Henry Bacon Dr & Lincoln Memorial Circle NW             31222
## 4 Henry Bacon Dr & Lincoln Memorial Circle NW             31222
## 5                            Lincoln Memorial             31270
## 6                            Lincoln Memorial             31270
##                End.station Bike.number Member.type
## 1  M St & New Jersey Ave SE      W00869      Member
## 2             8th & D St NW      W00894      Casual
## 3 New York Ave & 15th St NW      W21945      Casual
## 4 New York Ave & 15th St NW      W20012      Casual
## 5             8th & D St NW      W22786      Casual
## 6             8th & D St NW      W20890      Casual
```

Next, the monthly bike rental counts are obtained from the yearly bike trip data sets and then binded into a overtime data for further analyses.

```
bike_2011$date <- as.Date(bike_2011$Start.date)
bike_2011$month <- strftime(bike_2011$date, "%m")
bike_cnt_2011 <- data.frame(table(bike_2011$month))
colnames(bike_cnt_2011) <- c("month", "count")
bike_cnt_2011$date <- c("2011-01-01","2011-02-01","2011-03-01","2011-04-01","2011-05-01",
                        "2011-06-01","2011-07-01","2011-08-01","2011-09-01","2011-10-01",
                        "2011-11-01","2011-12-01")

bike_2012$date <- as.Date(bike_2012$Start.date)
bike_2012$month <- strftime(bike_2012$date, "%m")
bike_cnt_2012 <- data.frame(table(bike_2012$month))
colnames(bike_cnt_2012) <- c("month", "count")
bike_cnt_2012$date <- c("2012-01-01","2012-02-01","2012-03-01","2012-04-01","2012-05-01",
                        "2012-06-01","2012-07-01","2012-08-01","2012-09-01","2012-10-01",
                        "2012-11-01","2012-12-01")

bike_2013$date <- as.Date(bike_2013$Start.date)
bike_2013$month <- strftime(bike_2013$date, "%m")
bike_cnt_2013 <- data.frame(table(bike_2013$month))
colnames(bike_cnt_2013) <- c("month", "count")
bike_cnt_2013$date <- c("2013-01-01","2013-02-01","2013-03-01","2013-04-01","2013-05-01",
                        "2013-06-01","2013-07-01","2013-08-01","2013-09-01","2013-10-01",
                        "2013-11-01","2013-12-01")

bike_2014$date <- as.Date(bike_2014$Start.date)
bike_2014$month <- strftime(bike_2014$date, "%m")
bike_cnt_2014 <- data.frame(table(bike_2014$month))
colnames(bike_cnt_2014) <- c("month", "count")
bike_cnt_2014$date <- c("2014-01-01","2014-02-01","2014-03-01","2014-04-01","2014-05-01",
                        "2014-06-01","2014-07-01","2014-08-01","2014-09-01","2014-10-01",
                        "2014-11-01","2014-12-01")

bike_2015$date <- as.Date(bike_2015$Start.date)
bike_2015$month <- strftime(bike_2015$date, "%m")
bike_cnt_2015 <- data.frame(table(bike_2015$month))
colnames(bike_cnt_2015) <- c("month", "count")
bike_cnt_2015$date <- c("2015-01-01","2015-02-01","2015-03-01","2015-04-01","2015-05-01",
                        "2015-06-01","2015-07-01","2015-08-01","2015-09-01","2015-10-01",
                        "2015-11-01","2015-12-01")
```

```
bike_2016$date <- as.Date(bike_2016$Start.date)
bike_2016$month <- strftime(bike_2016$date, "%m")
bike_cnt_2016 <- data.frame(table(bike_2016$month))
colnames(bike_cnt_2016) <- c("month", "count")
bike_cnt_2016$date <- c("2016-01-01","2016-02-01","2016-03-01","2016-04-01","2016-05-01",
                        "2016-06-01","2016-07-01","2016-08-01","2016-09-01","2016-10-01",
                        "2016-11-01","2016-12-01")

bike_2017$date <- as.Date(bike_2017$Start.date)
bike_2017$month <- strftime(bike_2017$date, "%m")
bike_cnt_2017 <- data.frame(table(bike_2017$month))
colnames(bike_cnt_2017) <- c("month", "count")
bike_cnt_2017$date <- c("2017-01-01","2017-02-01","2017-03-01","2017-04-01","2017-05-01",
                        "2017-06-01","2017-07-01","2017-08-01","2017-09-01","2017-10-01",
                        "2017-11-01","2017-12-01")

bike_cnt <- rbind(bike_cnt_2011, bike_cnt_2012, bike_cnt_2013, bike_cnt_2014,
                  bike_cnt_2015, bike_cnt_2016, bike_cnt_2017)
head(bike_cnt,12)
```

```
##    month  count       date
## 1     01  37503 2011-01-01
## 2     02  47558 2011-02-01
## 3     03  63195 2011-03-01
## 4     04  93100 2011-04-01
## 5     05 133785 2011-05-01
## 6     06 141743 2011-06-01
## 7     07 139578 2011-07-01
## 8     08 135020 2011-08-01
## 9     09 125901 2011-09-01
## 10    10 122048 2011-10-01
## 11    11 101024 2011-11-01
## 12    12  86312 2011-12-01
```

Time Series Forecasting

A time series generally has three components - trend, seasonal and random activity that is not explained by the trend or the seasonal value. In an additive model, the seasonal and random fluctuations are roughly constant in size over time. In a multiplicative time series, the three components multiple together to make the time series. The size of the seasonal and random fluctuations increase with the level of the time series. Looking at the initial time series plot, the data suggested that it's a multiplicative time series model with seasonal component (seasonality). The rental counts drop at the beginning and at the end of each year.

```
library(fpp2)
```

```
## Loading required package: forecast
## Loading required package: fma
## Loading required package: expsmooth
```

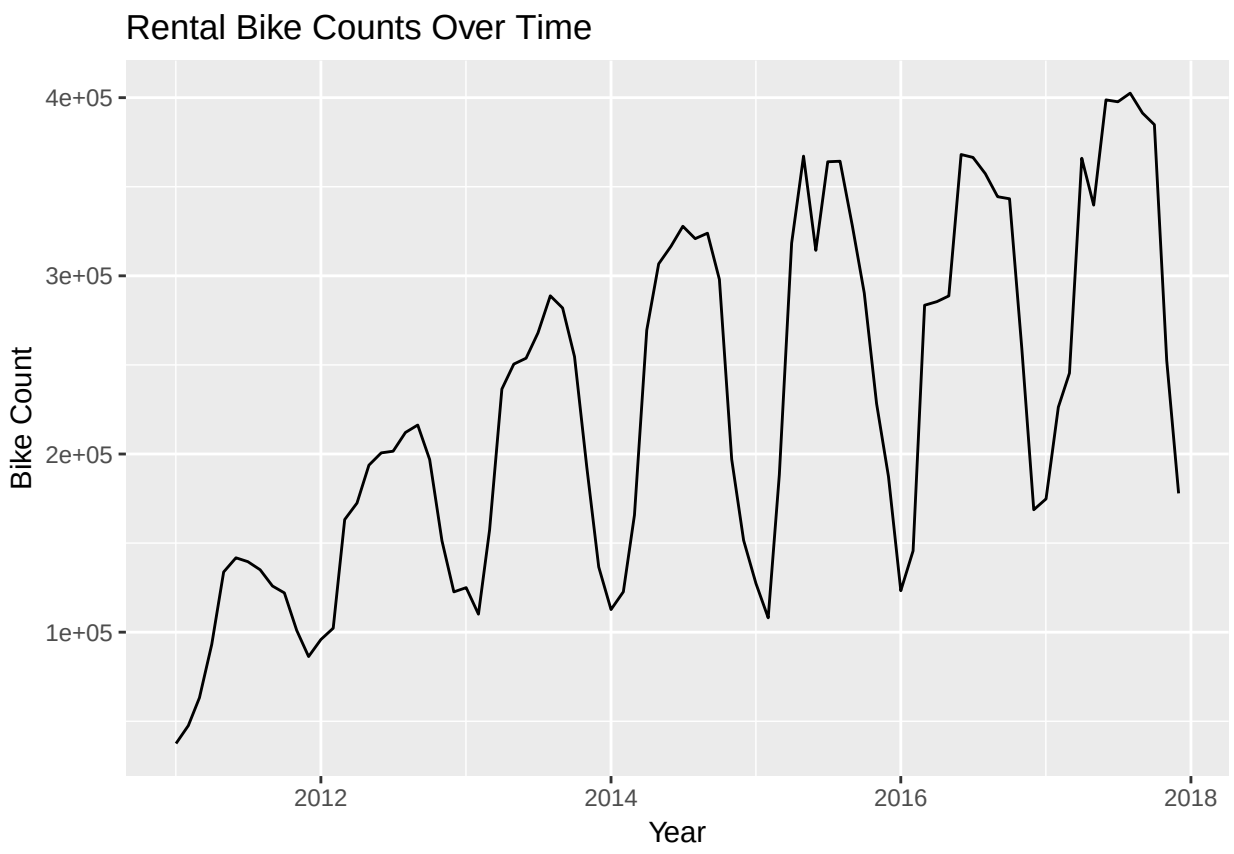
```
library(xts)
```

```
## Loading required package: zoo
##
```

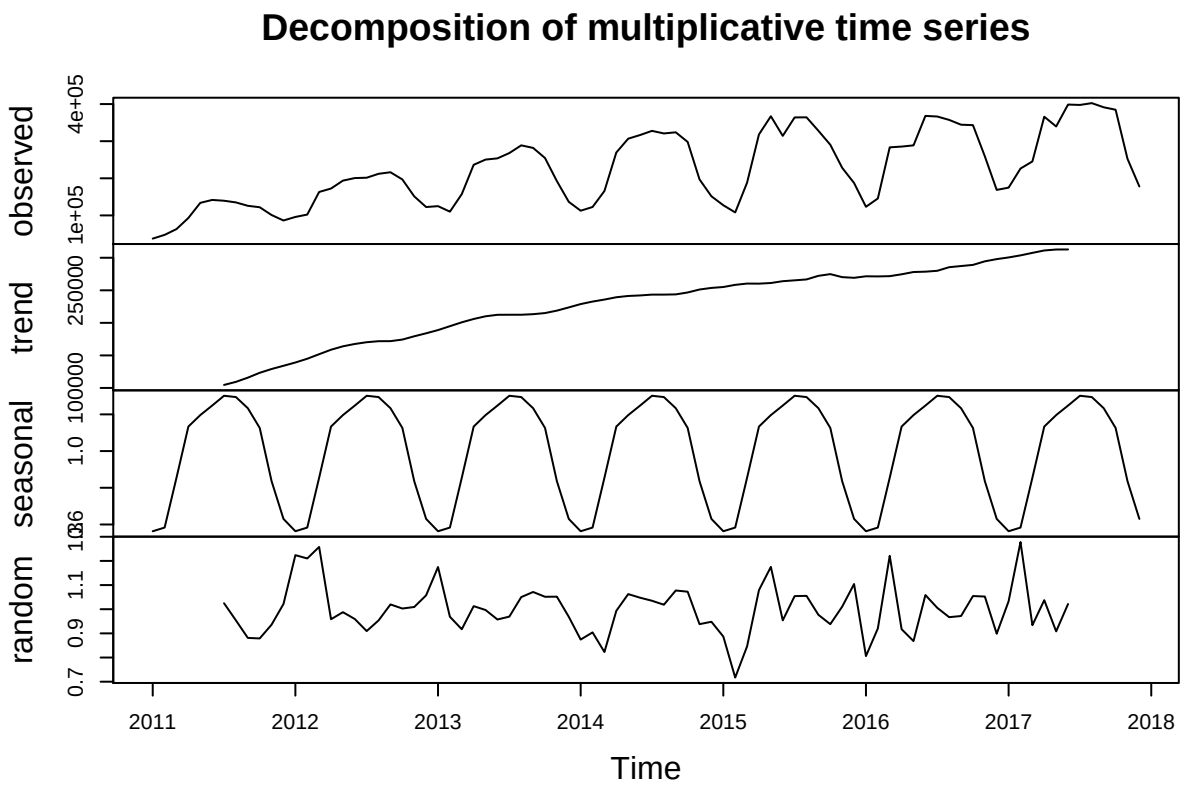
```
## Attaching package: 'zoo'
```

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

```
#Initial time series plot
bike_cnt_ts <- bike_cnt[, c(3,2)]
bike_cnt_ts$date <- as.Date(bike_cnt_ts$date)
bike_cnt_xts <- xts(bike_cnt_ts[,-1], order.by=bike_cnt_ts$date)
autoplot(bike_cnt_xts) + xlab("Year") + ylab("Bike Count") + ggtitle("Rental Bike Counts Over Time")
```

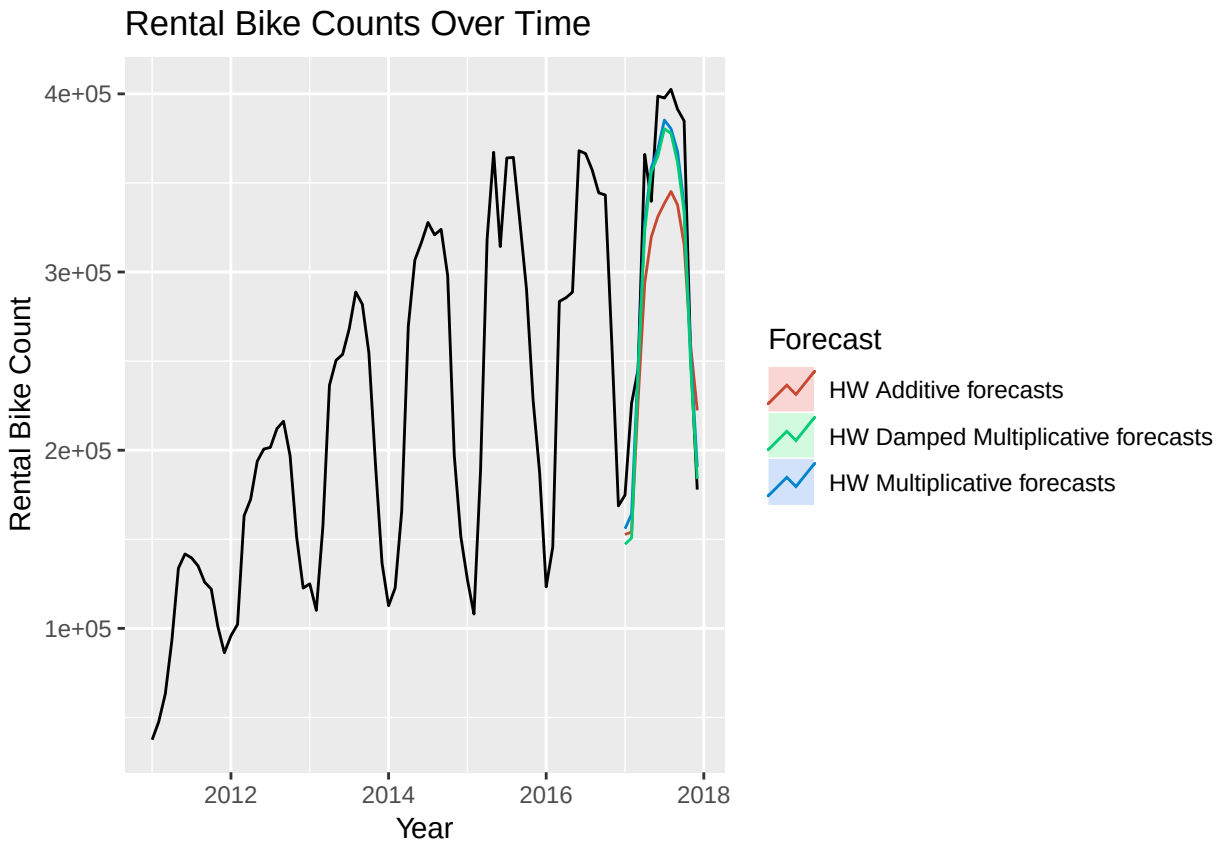



```
bike_ts <- ts(bike_cnt_ts[,-1], frequency=12, start=c(2011,1), end=c(2017,12))
bike_components <- decompose(bike_ts, "multiplicative")
plot(bike_components)
```

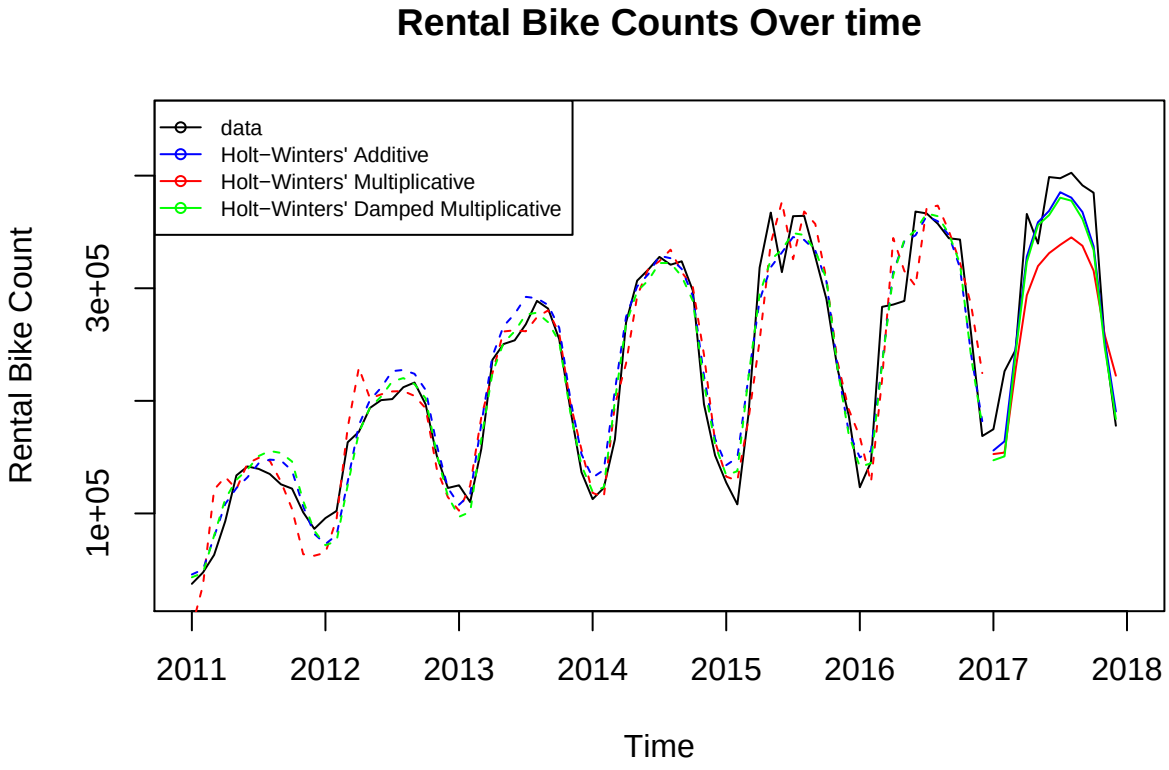



```
# Fitting three forecasting models
bike_timeseries <- ts(bike_cnt_ts[,-1], frequency=12, start=c(2011,1), end=c(2016,12))
fit1 <- hw(bike_timeseries, seasonal = "additive", h=12)
fit2 <- hw(bike_timeseries, seasonal = "multiplicative", h=12)
fit3 <- hw(bike_timeseries, seasonal = "multiplicative", damped = TRUE, h=12)
```

```
autoplot(bike_ts) +
autolayer(fit1, series="HW Additive forecasts", PI=FALSE) +
autolayer(fit2, series="HW Multiplicative forecasts",PI=FALSE) +
autolayer(fit3, series="HW Damped Multiplicative forecasts",PI=FALSE) +
xlab("Year") + ylab("Rental Bike Count") +  ggtitle("Rental Bike Counts Over Time") +
guides(colour=guide_legend(title="Forecast"))
```




```r
plot(bike_ts, ylab = "Rental Bike Count", main = "Rental Bike Counts Over time",
     xlim = c(2011,2018), ylim = c(30000,450000))
lines(fitted(fit1), col = "red", lty = 2)
lines(fitted(fit2), col = "blue", lty = 2)
lines(fitted(fit3), col = "green", lty = 2)
lines(fit1$mean, col = "red")
lines(fit2$mean, col = "blue")
lines(fit3$mean, col = "green")
legend("topleft", lty = 1, col = c(1,"blue","red","green"), c("data", "Holt-Winters' Additive",
      "Holt-Winters' Multiplicative", "Holt-Winters' Damped Multiplicative"), pch=21, cex=0.7)
```




```
fit1$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2017 152731.1 154049.2 228409.2 293746.3 319668.0 331127.4 338540.4
##           Aug      Sep      Oct      Nov      Dec
## 2017 345186.8 337691.1 315579.1 258492.2 222345.0
```

```
fit2$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2017 155930.2 164069.7 247741.9 328000.8 358618.5 368727.3 385282.2
##           Aug      Sep      Oct      Nov      Dec
## 2017 380457.6 367668.9 337253.0 253904.2 190561.1
```

```
fit3$mean
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 2017 147220.5 150719.2 242469.9 323305.5 356022.8 364925.8 380425.8
##           Aug      Sep      Oct      Nov      Dec
## 2017 377755.7 361504.3 333880.4 247520.3 183929.2
```

```
mean(fit1$residuals)
```

```
## [1] -2178.217
```

```
mean(fit2$residuals)
```

```
## [1] -0.0256198
```

```
mean(fit3$residuals)
```

```
## [1] -0.002797479
```

```
sd(fit1$residuals)
```

```
## [1] 27324.31
```

```
sd(fit2$residuals)
```

```
## [1] 0.1102268
```

```
sd(fit3$residuals)
```

```
## [1] 0.1131761
```

```
fit1$model
```

```
## Holt-Winters' additive method
##
## Call:
##  hw(y = bike_timeseries, h = 12, seasonal = "additive")
##
##   Smoothing parameters:
##     alpha = 0.9344
##     beta  = 1e-04
##     gamma = 1e-04
##
##   Initial states:
##     l = 93624.8819
##     b = 4179.1951
##     s = -75346.61 -35038.76 26206.8 52485.38 64145.77 61657.5
##           58412.08 51114.14 29360.8 -31810.9 -102014.4 -99171.81
##
##   sigma:  30865.92
##
##      AIC     AICc      BIC
## 1812.412 1823.745 1851.115
```

```
fit2$model
```

```
## Holt-Winters' multiplicative method
##
## Call:
##  hw(y = bike_timeseries, h = 12, seasonal = "multiplicative")
##
##   Smoothing parameters:
##     alpha = 0.0734
##     beta  = 0.0096
##     gamma = 1e-04
##
##   Initial states:
##     l = 79371.3497
##     b = 5129.2819
##     s = 0.6321 0.8458 1.1282 1.2351 1.2835 1.3053
##           1.2546 1.2255 1.1257 0.854 0.568 0.5422
##
##   sigma:  0.1275
##
##      AIC     AICc      BIC
## 1781.443 1792.777 1820.147
```

```
fit3$model
```

```
## Damped Holt-Winters' multiplicative method
##
## Call:
##  hw(y = bike_timeseries, h = 12, seasonal = "multiplicative",
##
##  Call:
##      damped = TRUE)
##
##   Smoothing parameters:
##     alpha = 0.0085
##     beta  = 0.0085
##     gamma = 1e-04
##     phi   = 0.976
##
##   Initial states:
##     l = 77568.7133
##     b = 6156.9931
##     s = 0.6274 0.8464 1.1445 1.2424 1.3018 1.3146
##            1.2645 1.2372 1.1269 0.8477 0.5286 0.518
##
##   sigma:  0.1286
##
##      AIC     AICc      BIC
## 1780.114 1793.020 1821.094
```

```
accuracy(fit1)
```

```
##                      ME     RMSE     MAE        MPE     MAPE     MASE
## Training set -2178.217 27221.19 20261.4 -0.9314913 13.09301 0.491255
##                     ACF1
## Training set 0.01710318
```

```
accuracy(fit2)
```

```
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -4705.093 19702.33 15590.55 -3.865657 9.057531 0.3780063
##                    ACF1
## Training set 0.2010208
```

```
accuracy(fit3)
```

```
##                    ME     RMSE     MAE      MPE     MAPE      MASE
## Training set -665.766 17982.32 13349.8 -1.48183 7.711853 0.3236774
##                   ACF1
## Training set 0.176693
```

To select the best time series forecasting model, the following four criteria are used to assess how well the models fit: AIC, RMSE, mean of the residuals, and standard deviation of the residuals. The lower the values are the better the model fits the data. Table 1 below shows the summary criteria for the three forecasting models.

We also check the performance of the three time series models on 2017 data by looking at the RMSE statistics. RMSE is the lowest for the Holt Winters' multiplicative time series model. Since this assessment is only based on 12 data points, it might not be as reliable as the outcome we observe with the overtime (2011-2016) data.

```
actual_2017 <- bike_cnt[c(73:84),2]
RMSE_1 = sqrt(mean((actual_2017-fit1$mean)^2))
RMSE_2 = sqrt(mean((actual_2017-fit2$mean)^2))
```

Method	AIC	RMSE	Mean(Residuals)	SD(Residuals)
Holt-Winters' Additive	1812.412	27221.19	-2178.217	27324.31
Holt-Winters' Multiplicative	1781.443	19702.33	-0.025620	0.11023
Holt-Winters' Damped Multiplicative	1780.114	17982.32	-0.002797	0.11328

Table 1: Summary table of criteria.

```
RMSE_3 = sqrt(mean((actual_2017-fit3$mean)^2))
RMSE_1
```

```
## [1] 52172.81
```

```
RMSE_2
```

```
## [1] 29674.68
```

```
RMSE_3
```

```
## [1] 34371.78
```

Conclusion

Overall, the criteria for the Holt Winters' damped multiplicative time series model are the lowest, thus it performs the best. However, in forcasting 2017 data, the Holt Winters' multiplicative time series model outperform the damped multiplicative model. The better prediction can be obtained by taking into consideration of the population growth, the weather and the number of bikes that are available each year.

In Conclusion, the bike sharing companies can utilize the random forest model and the multiplicative time series models to predict and forecast the count of rental bikes. They can be very useful in making decisions to meet the bike demand and in helping them grow their business.

GitHub Link

https://github.com/sweechew91/MSDS6306_Project2