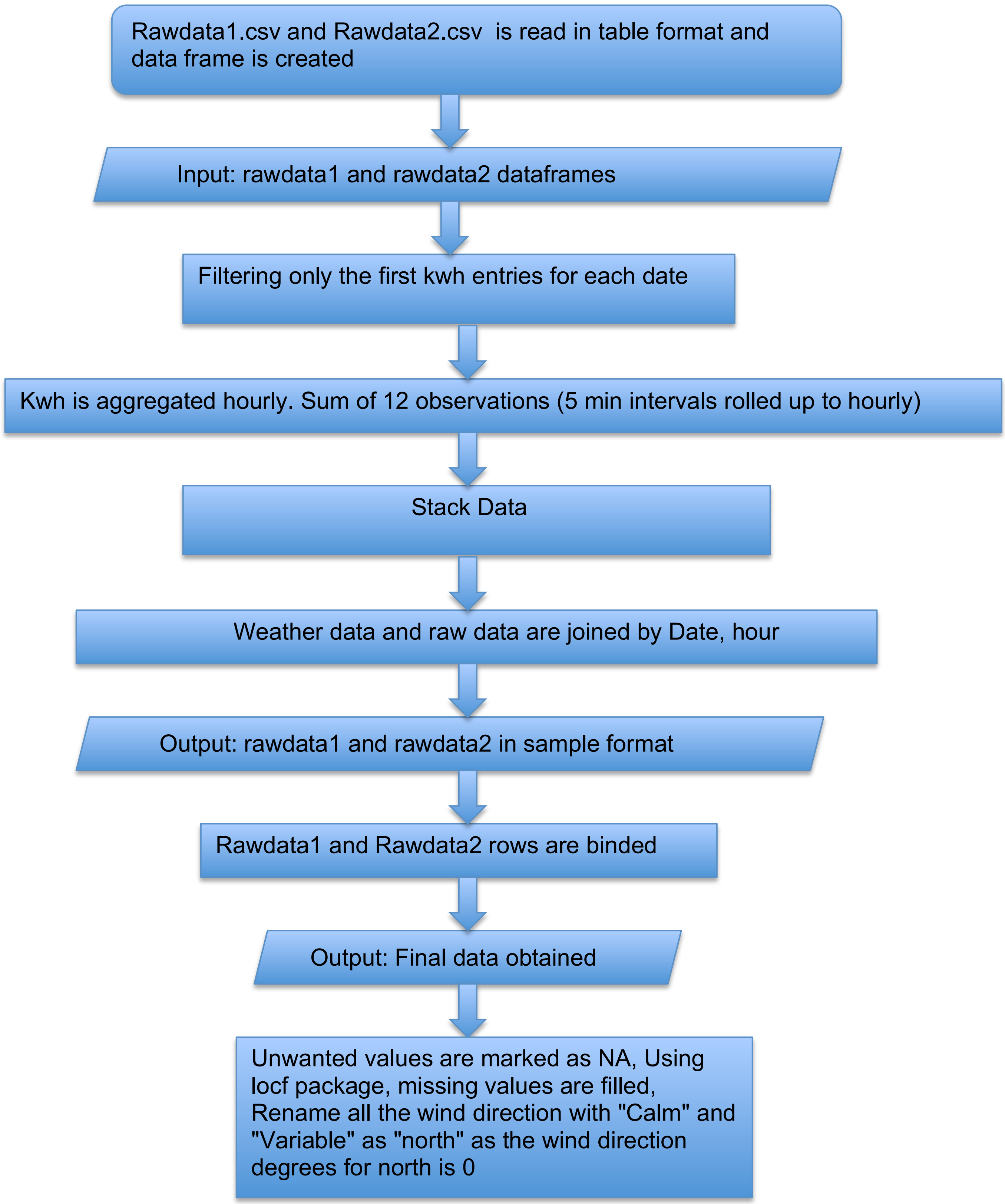
Part 1.1.1 and 1.1.2: Data wrangling and cleansing

1) Dplyr and tidyr package is used.



```r
rawData1 <- filter(rawData1, grepl('kWh', Channel))
j<-0
last <- ncol(rawData1)
i<-5

#Adding hourly data of each date i.e. Kwh is aggregated hourly. Sum of 12 observations (5 min intervals rol
library(lazyeval)
f = function(df, n, new_col_name) {
  mutate_call = lazyeval::interp(~ rowSums(df[, c(n:(n+11))]) )
  rawData1 %>% mutate_(.dots = setNames(list(mutate_call), new_col_name))
}

while(i <= (last-11)){
  newname <- j
  rawData1 <- f(df = rawData1, n = i,new_col_name = newname)
  i <- i+12
  j<-j+1
}
rawData1<-rawData1[-(5:last)]
rawData1<-rawData1[-(3:4)]

#stack data
rawData1 <- rawData1 %>% gather(hour, kWh, -(1:2))
```

```r
#stack data
rawData1 <- rawData1 %>% gather(hour, kWh, -(1:2))

#Convert to date format
rawData1 <- rawData1 %>% mutate(Date = as.Date(rawData1$Date , "%m/%d/%Y"))

#ordering the data with respect to date
rawData1 <- arrange(rawData1, Date)

#creating year, month, day, day of week, weekday columns
#month 1-12 => Jan-Dec - Derived from dates
#day 1-31 - Derived from dates
#Dayof Week 0-6 -Sun-Sat - Derived from dates
#Weekday 1- Yes 0- No - Derived from dates
rawData1 <- rawData1 %>% mutate(year = as.numeric(format(rawData1$Date, format = "%Y")))
rawData1 <- rawData1 %>% mutate(month = as.numeric(format(rawData1$Date, format = "%m")))
rawData1 <- rawData1 %>% mutate(day = as.numeric(format(rawData1$Date, format = "%d")))
rawData1 <- rawData1 %>% mutate(dayOfWeek = as.numeric(format(rawData1$Date, format = "%w")))
rawData1 <- rawData1 %>% mutate(Weekday = ifelse(dayOfWeek %in% 1:5 , 1, 0))
rawData1$hour <- as.numeric(rawData1$hour)
distinct_df = rawData1 %>% distinct(Date)
XX <- NULL
g<-1

#Extracting weather data using weatherdata package
while(g<= nrow(distinct_df)){
XX_temp <- getDetailedWeather(station_id = "KBOS", distinct_df[g, ] , opt_custom_columns = T, custom_columns = c(2:8, 12:13))
XX_temp <- filter(XX_temp, grepl('.*:54:00', Time ))
XX_temp$Wind_SpeedMPH <- as.factor(XX_temp$Wind_SpeedMPH)
XX_temp$Humidity <- as.factor(XX_temp$Humidity)
XX <- bind_rows(XX, XX_temp)
g <- g+1
}
```

```r
#sepearte column into date and time
XX <- separate(XX, Time, into = c("Date","Time"), sep=" ", remove = TRUE)

#creating hour column:
XX <- separate(XX, Time, into = c("hour"), sep=":", remove = FALSE)
XX <- transform(XX, hour = as.numeric(hour))
XX$Date <- as.Date(XX$Date, "%Y-%m-%d")

#Joining rawdata and weatherdata by date and then hour
fulldata <- full_join(rawData1, XX, by = c("Date", "hour"))
library(chron)

#Converting to time format
fulldata$Time <- chron::times(fulldata$Time)

#creating peakhour column: 7AM-7PM - 1 ; 7PM-7AM - 0
fulldata <- mutate(fulldata, Peakhour = ifelse(hour %in% 7:18 , 1, 0))
fulldata$Time <- NULL
write.csv(XX, "rawdata1_clean.csv")
write.csv(fulldata, "fulldata.csv")
```

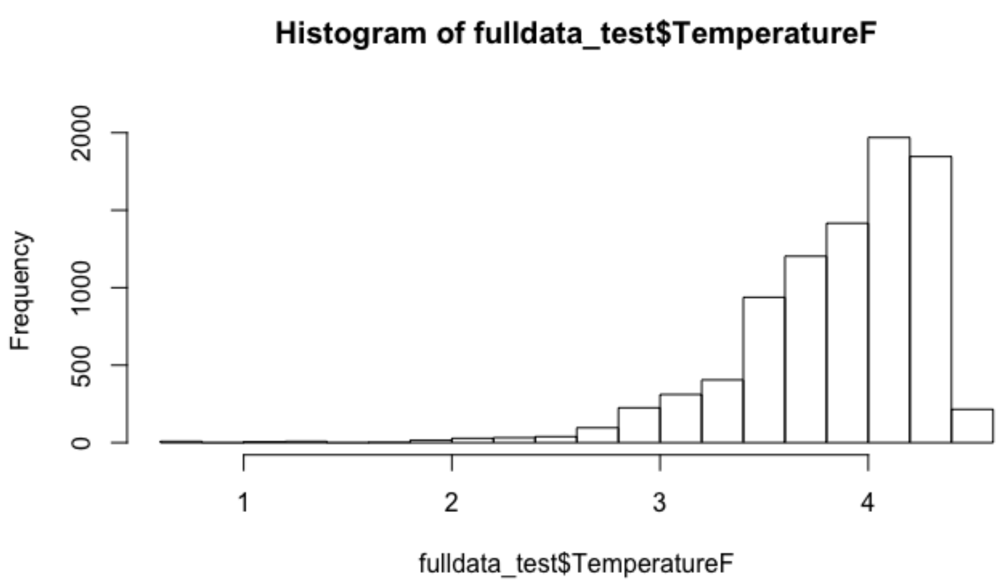
Part 1.2: Multiple-Linear Regression

Feature Transformation:

log transformation on Temperature to make the growth approximately exponential.
This makes the data less skewed.

WindDirDegrees column: Introduced 2 variables, sin(WindDirDegrees) and cos(WindDirDegrees). This will yield less residual than using only WindDirDegrees

Histogram of log(TemperatureF):

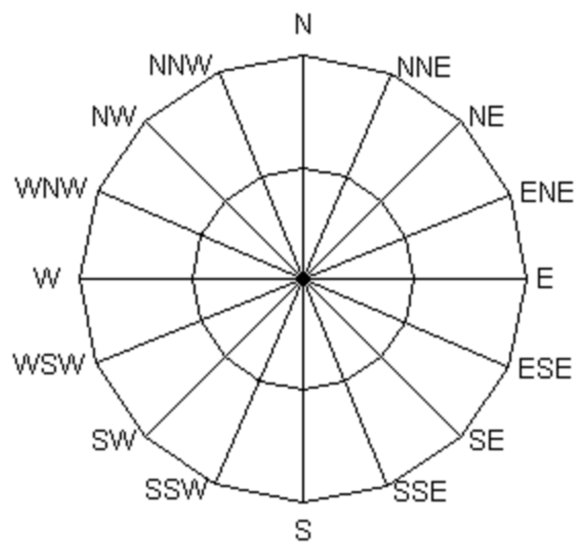


Feature Selection:

- Date, Account, Year are not selected as they are single values
- Wind_Direction is not selected as it is related to WindDirDegrees

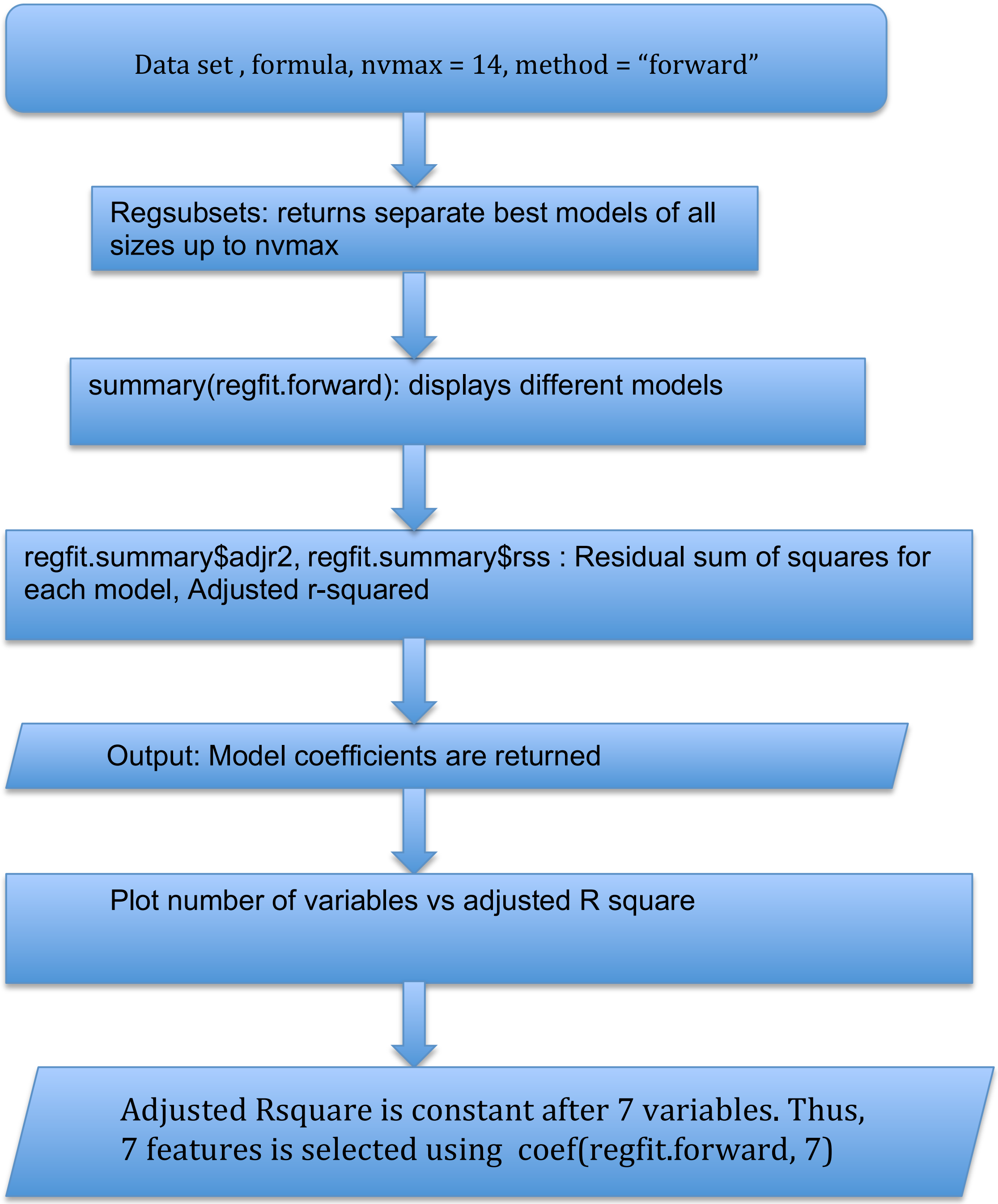
Wind Direction and Degrees

Cardinal Direction	Degree Direction
N	348.75 - 11.25
NNE	11.25 - 33.75
NE	33.75 - 56.25
ENE	56.25 - 78.75
E	78.75 - 101.25
ESE	101.25 - 123.75
SE	123.75 - 146.25
SSE	146.25 - 168.75
S	168.75 - 191.25
SSW	191.25 - 213.75
SW	213.75 - 236.25
WSW	236.25 - 258.75
W	258.75 - 281.25
WNW	281.25 - 303.75
NW	303.75 - 326.25
NNW	326.25 - 348.75



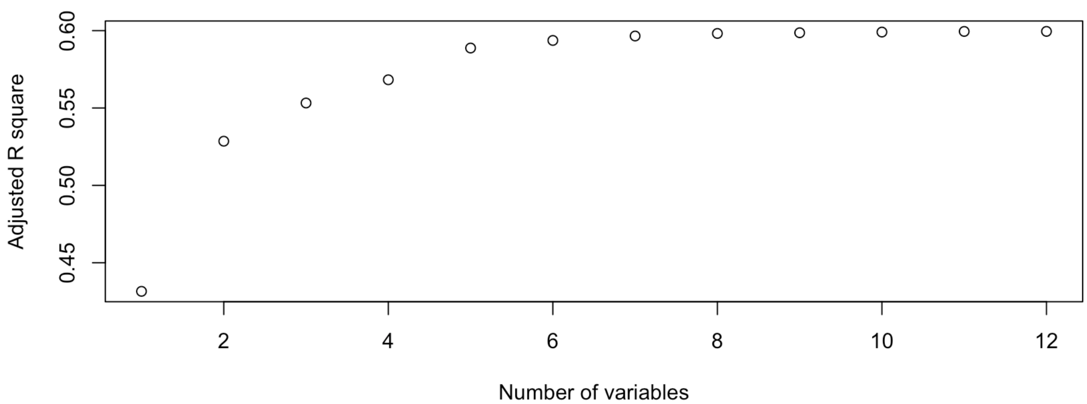
- WindDirDegrees is not selected as it is split into cos(WindDirDegrees) and sin(WindDirDegrees). Hence, WindDirecSin and WindDireccos is selected
- Leaps package used for variable selection: Here subset regression is performed wherein number of subsets of each sizes is reported. Models are plotted and ordered by the selection statistic.
- Stepwise selection (forward, backward, both) is done using the stepAIC() function from the MASS package. stepAIC() performs stepwise model selection by exact AIC.

Forward Regression:

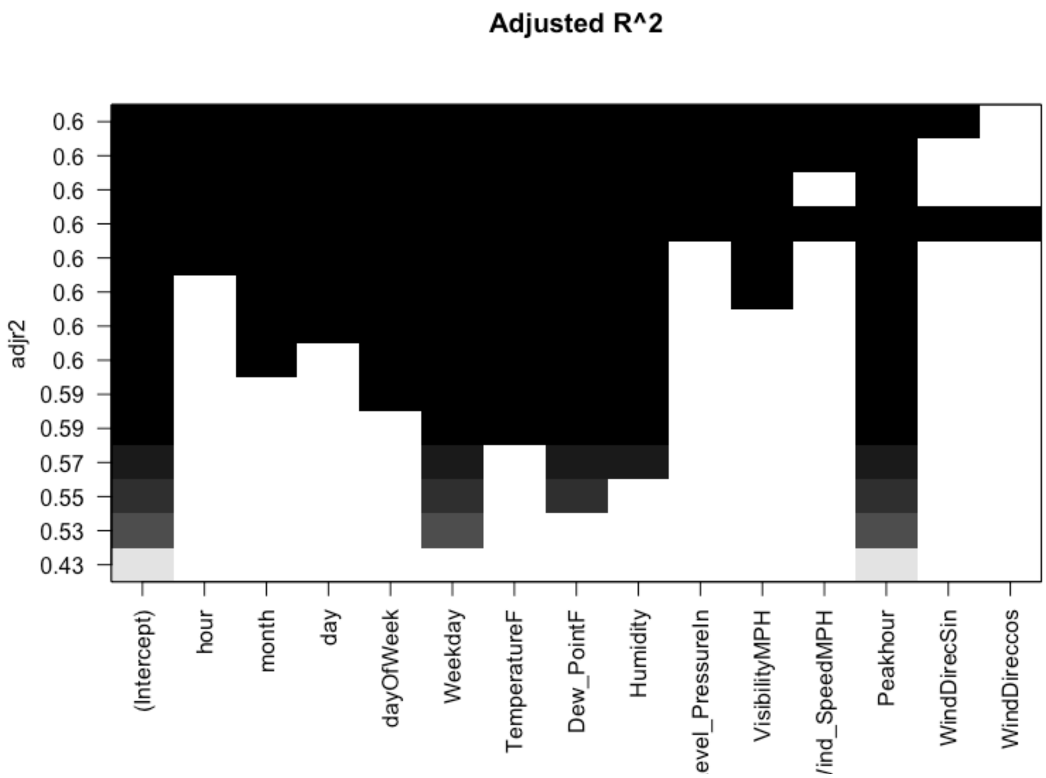


```r
#Forward regression
regfit.forward <- regsubsets(kWh ~ .-Date - Account - year -Wind_Direction -Conditions -WindDirDegrees ,
                             data = fulldata_test, nvmax = 14, method = "forward")
regfit.summary <- summary(regfit.forward)
names(regfit.summary)
regfit.summary$adjr2
regfit.summary$rss
coef(regfit.forward, 7)

x = 1:14
y = regfit.summary$adjr2
plot(x,y, xlab = "Number of variables", ylab = "Adjusted R square")
```

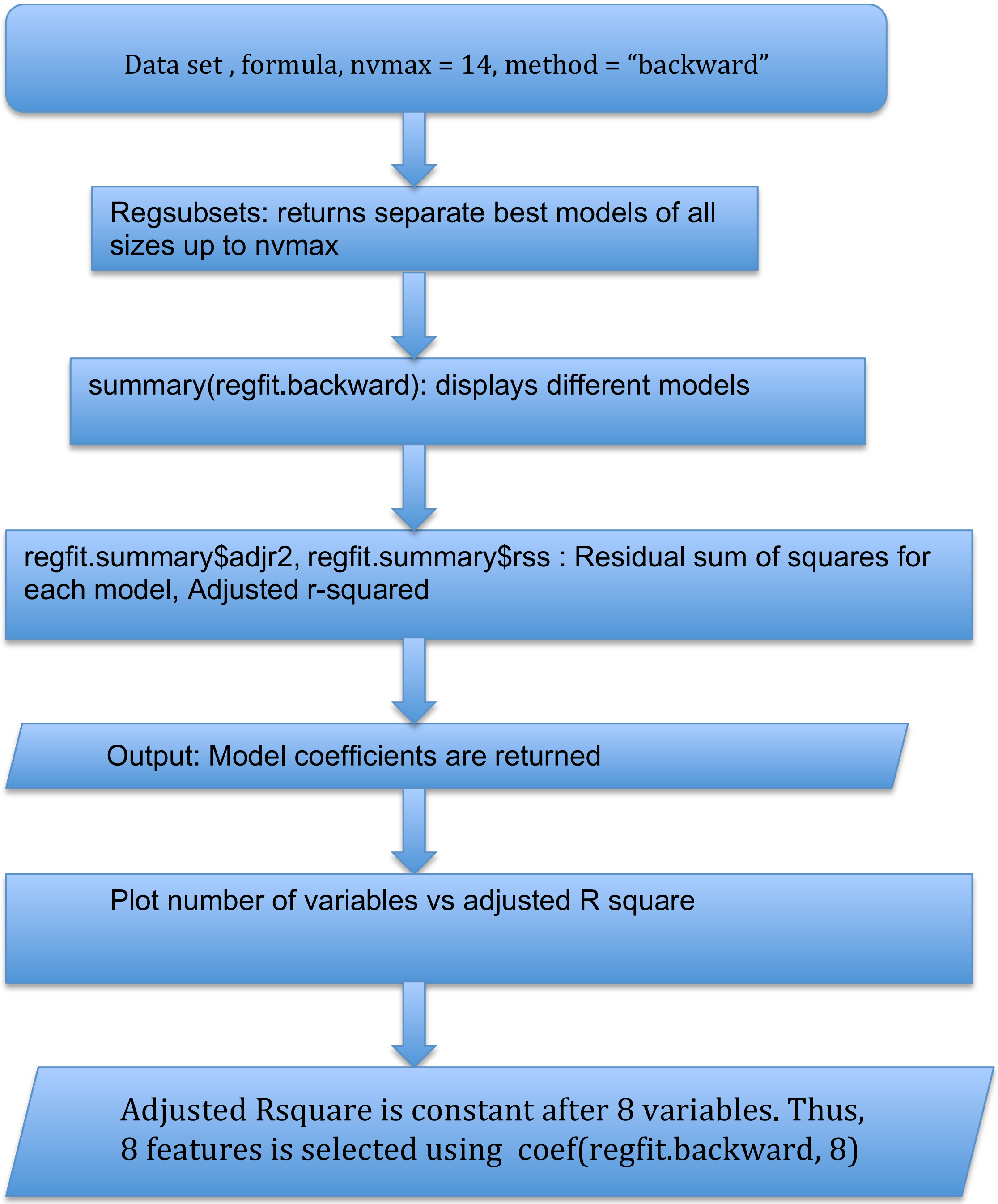
```
> regfit.summary$adjr2
 [1] 0.4315115 0.5285401 0.5532266 0.5682366 0.5887473 0.5936602 0.5965230 0.5981792
 [9] 0.5986297 0.5990690 0.5994766 0.5995052
> regfit.summary$rss
 [1] 55973572 46414808 43979417 42497010 40473587 39985521 39699272 39531802 39482965
[10] 39435243 39390657 39383343
> coef(regfit.forward, 7)
 (Intercept)        month    dayOfWeek      Weekday TemperatureF   Dew_PointF     Humidity
  406.695067    -1.836586     3.743180    72.433312   -94.404741     3.691028    -1.678642
    Peakhour
  132.256952
```



```
plot(regfit.forward, scale = "adjr2", main = "Adjusted R^2")
```

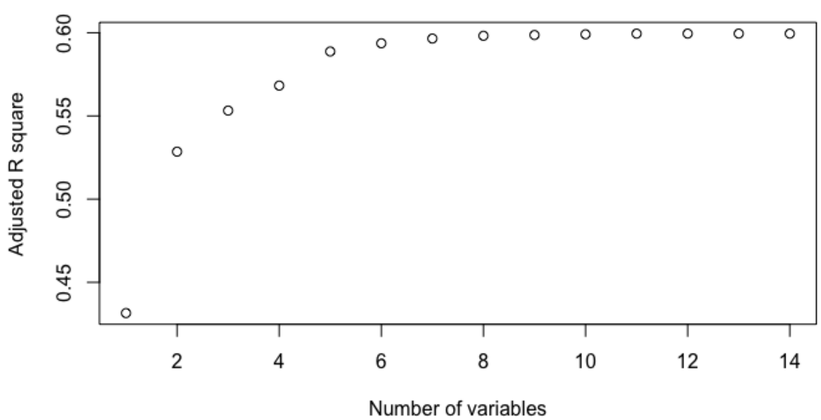


Backward Regression:

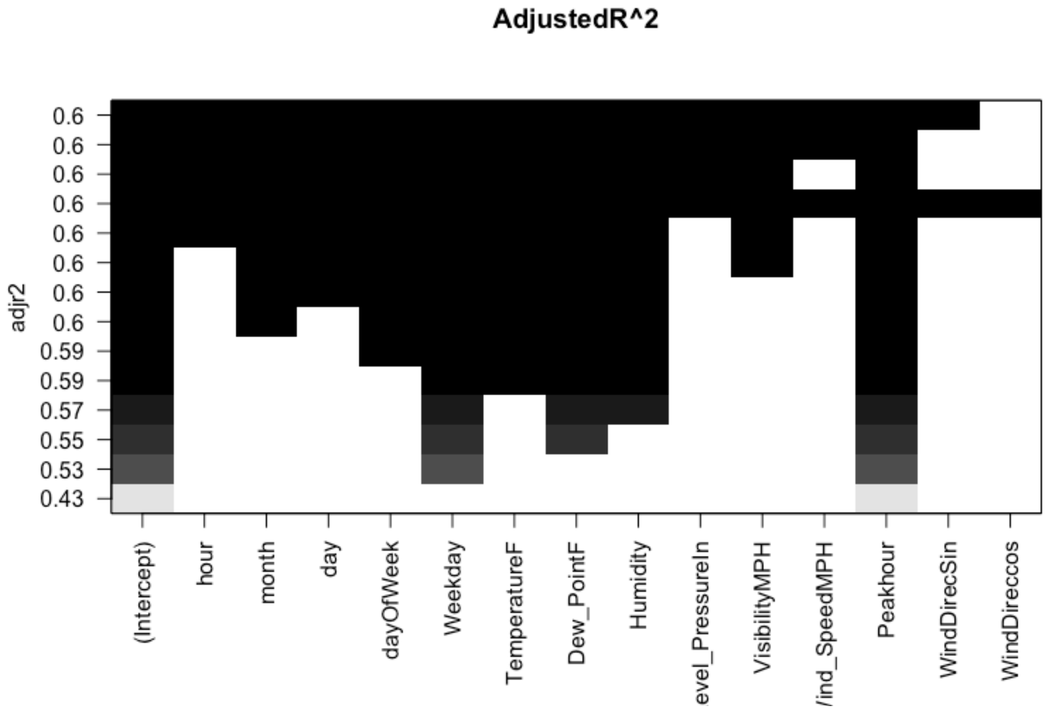


```
#Backward regression
regfit.forward <- regsubsets(kWh ~ .-Date - Account - year -Wind_Direction -Conditions -WindDirDegrees ,
                             data = fulldata_test, nvmax = 14, method = "backward")
regfit.summary <- summary(regfit.forward)
names(regfit.summary)
regfit.summary$adjr2
regfit.summary$rss
coef(regfit.forward, 7)

x = 1:14
y = regfit.summary$adjr2
plot(x,y, xlab = "Number of variables", ylab = "Adjusted R square")
```

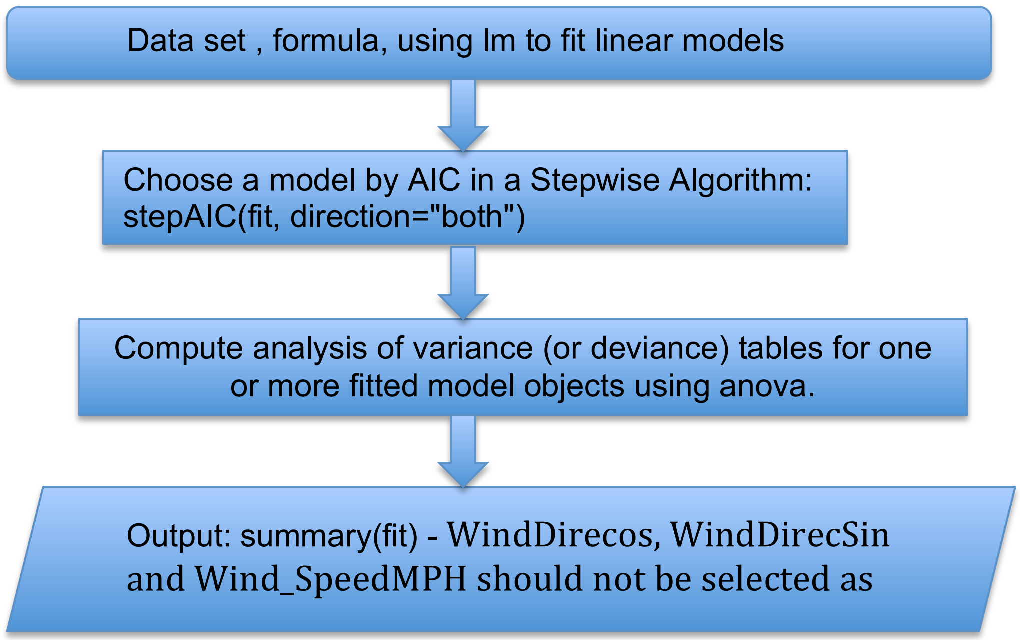
```
> regfit.summary$adjr2
 [1] 0.4315115 0.5285401 0.5532266 0.5682366 0.5887473 0.5936602 0.5965230
 [8] 0.5981792 0.5986297 0.5990690 0.5994766 0.5995052 0.5995059 0.5994674
> regfit.summary$rss
 [1] 55973572 46414808 43979417 42497010 40473587 39985521 39699272
 [8] 39531802 39482965 39435243 39390657 39383343 39378773 39378055
> coef(regfit.forward, 7)
 (Intercept)        month     dayOfWeek      Weekday TemperatureF
  406.695067    -1.836586      3.743180    72.433312   -94.404741
  Dew_PointF     Humidity      Peakhour
    3.691028    -1.678642    132.256952
```



```
plot(regfit.backward, scale = "adjr2", main = "AdjustedR^2")
```



Stepwise Regression:

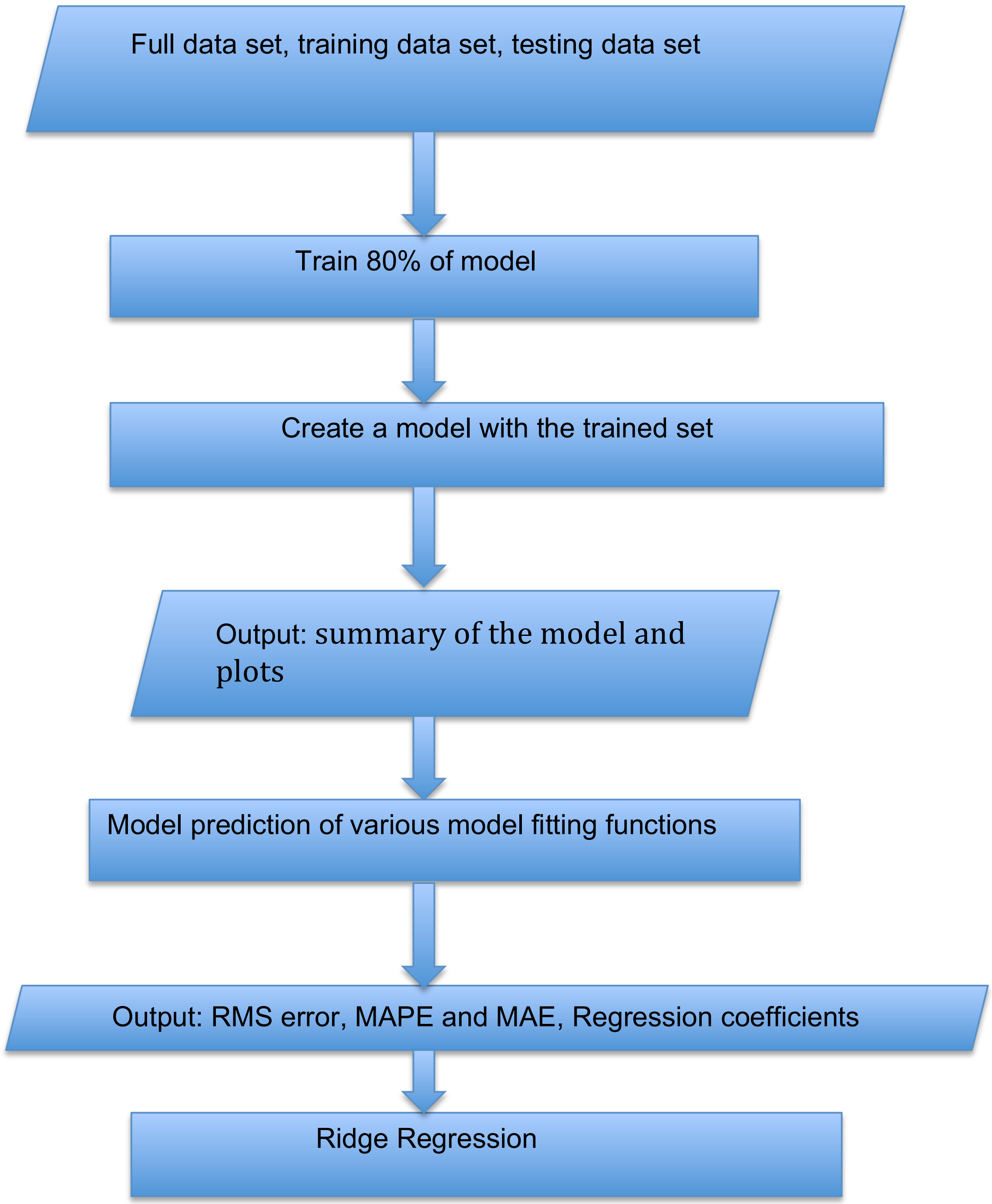



```
#Stepwise regression
library(MASS)
fit <- lm(kWh~.-Date - Account - year -Wind_Direction -Conditions -WindDirDegrees,data=fulldata_test)
step <- stepAIC(fit, direction="both")
step$anova # display results
summary(fit)
```

```
Stepwise Model Path
Analysis of Deviance Table

Initial Model:
kWh ~ (Account + Date + hour + year + month + day + dayOfWeek +
    Weekday + TemperatureF + Dew_PointF + Humidity + Sea_Level_PressureIn +
    VisibilityMPH + Wind_Direction + Wind_SpeedMPH + Conditions +
    WindDirDegrees + Peakhour + WindDirecSin + WindDireccos) -
    Date - Account - year - Wind_Direction - Conditions - WindDirDegrees

Final Model:
kWh ~ hour + month + day + dayOfWeek + Weekday + TemperatureF +
    Dew_PointF + Humidity + Sea_Level_PressureIn + VisibilityMPH +
    Peakhour


             Step Df  Deviance Resid. Df Resid. Dev      AIC
1                                   8747   39378055 73723.15
2   - WindDireccos  1  718.4911      8748   39378773 73721.31
3   - WindDirecSin  1 4570.3619      8749   39383343 73720.33
4 - Wind_SpeedMPH  1 7313.5936      8750   39390657 73719.95
```

```
> summary(fit)

Call:
lm(formula = kWh ~ . - Date - Account - year - Wind_Direction -
    Conditions - WindDirDegrees, data = fulldata_test)

Residuals:
     Min       1Q   Median       3Q      Max
-257.153  -44.831   -3.685   37.249  278.848

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           84.70554  103.25424   0.820 0.412034
hour                  -0.35147    0.10644  -3.302 0.000964 ***
month                 -1.86378    0.23423  -7.957 1.98e-15 ***
day                   -0.47364    0.08199  -5.777 7.86e-09 ***
dayOfWeek              3.68256    0.36123  10.194  < 2e-16 ***
Weekday               72.10207    1.60149  45.022  < 2e-16 ***
TemperatureF         -90.32268    5.02834 -17.963  < 2e-16 ***
Dew_PointF             3.68537    0.12824  28.738  < 2e-16 ***
Humidity              -1.78157    0.07295 -24.423  < 2e-16 ***
Sea_Level_PressureIn  11.23561    3.30949   3.395 0.000689 ***
VisibilityMPH         -1.49955    0.38080  -3.938 8.28e-05 ***
Wind_SpeedMPH          0.21005    0.16123   1.303 0.192681
Peakhour             131.11302    1.56085  84.001  < 2e-16 ***
WindDirecSin           1.01203    1.03417   0.979 0.327810
WindDireccos           0.40148    1.00496   0.399 0.689537
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 67.1 on 8747 degrees of freedom
Multiple R-squared:  0.6001,	Adjusted R-squared:  0.5995
F-statistic: 937.6 on 14 and 8747 DF,  p-value: < 2.2e-16

>
```

Model:


Full data set, training data set, testing data set
Train 80% of model
Create a model with the trained set
Output: summary of the model and plots
Model prediction of various model fitting functions
Output: RMS error, MAPE and MAE, Regression coefficients
Ridge Regression

Training 80% model:

```
#Model
smp_size <- floor(0.80 * nrow(fulldata_test))
set.seed(123)
train_ind <- sample(seq_len(nrow(fulldata_test)), size = smp_size)
train <- fulldata_test[train_ind,]
test <- fulldata_test[-train_ind,]
```

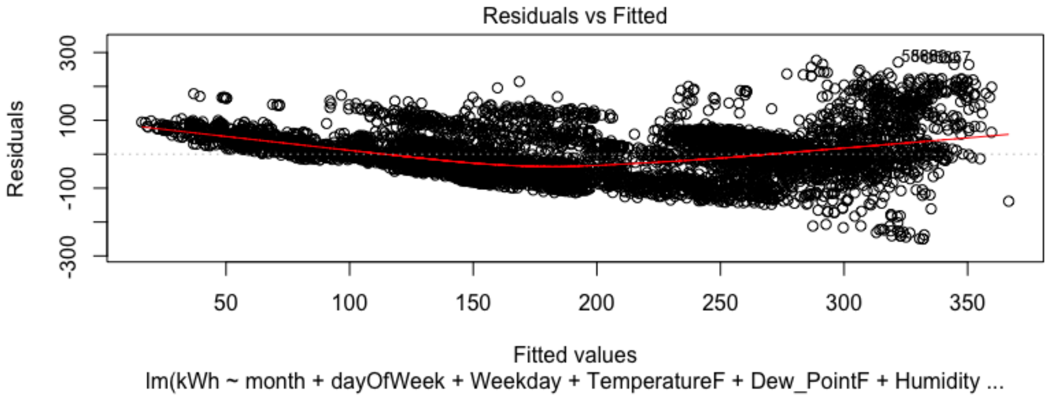
Creating a model:

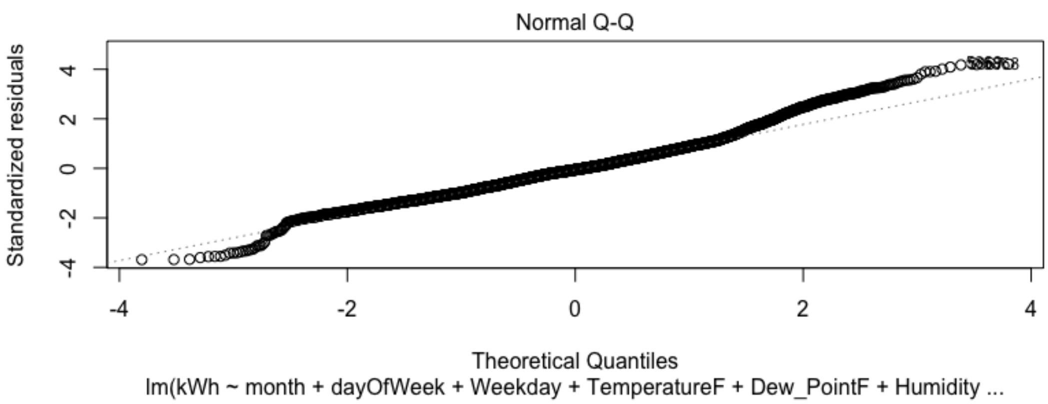
```
model1<- lm(kWh ~ month+dayOfWeek+Weekday+TemperatureF+Dew_PointF+Humidity+Peakhour , data = train)
summary(model1)
plot(model1)
ppp <- predict(model1, test)
accuracy(ppp, train$kWh)
```

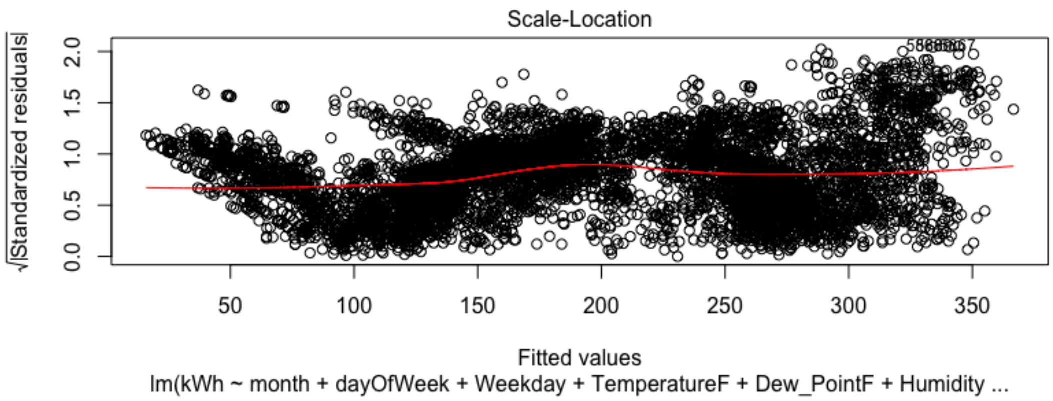
Below is the summary of model:

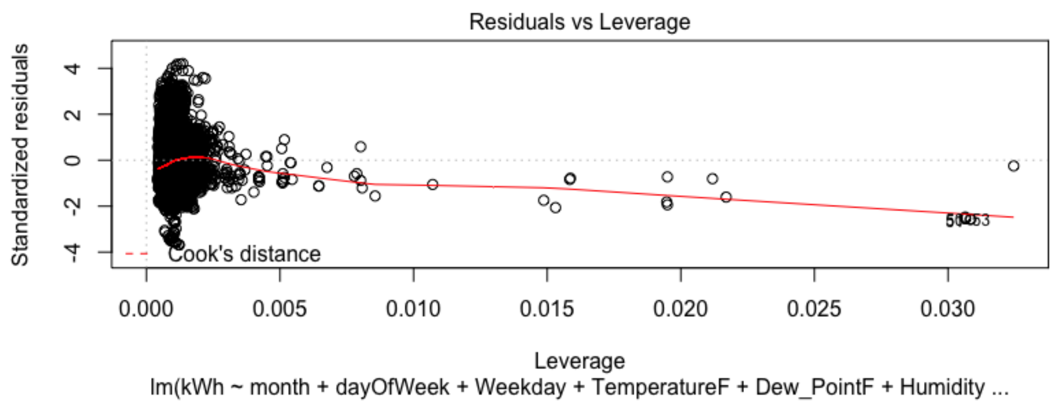
```
Call:
lm(formula = kWh ~ month + dayOfWeek + Weekday + TemperatureF +
    Dew_PointF + Humidity + Peakhour, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-250.002  -45.982   -3.701   37.781  285.092

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 408.48166   20.15592  20.266  < 2e-16 ***
month        -1.91171    0.25960  -7.364 1.99e-13 ***
dayOfWeek     3.87388    0.40646   9.531  < 2e-16 ***
Weekday      72.63454    1.79959  40.362  < 2e-16 ***
TemperatureF -94.52354   5.69998 -16.583  < 2e-16 ***
Dew_PointF     3.66942   0.14456  25.383  < 2e-16 ***
Humidity      -1.67368   0.07161 -23.374  < 2e-16 ***
Peakhour     131.23579   1.69935  77.227  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 67.81 on 7001 degrees of freedom
Multiple R-squared:  0.5906,    Adjusted R-squared:  0.5902
F-statistic:  1443 on 7 and 7001 DF,  p-value: < 2.2e-16
```

Plotting the model:









RMS error, MAPE and MAE:

```
> accuracy(ppp, train$kWh)
                ME     RMSE      MAE       MPE     MAPE
Test set -0.885018 134.1416 105.0416 -30.96672 66.65354
>
```

RegressionOutput.csv: using broom package

```
#Regression
library(devtools)
install_github("dgrtwo/broom")
library(broom)
tidy_lmfit <- tidy(summary(model1))
tidy_lmfit <- tidy_lmfit[-c(3:5)]
tidy_lmfit <- rbind(c("Account",fulldata_test$Account[1]), tidy_lmfit)
tidy_lmfit[2,1] = "constant"
write.table(tidy_lmfit, file = "RegressionOutputs.csv", sep = "," , row.names = FALSE, col.names = FALSE)
```

A	B	C
Account	26908650026	
constant	408.4816557	
month	-1.911711096	
dayOfWeek	3.873878815	
Weekday	72.63453752	
Temperature	-94.52353846	
Dew_PointF	3.669424598	
Humidity	-1.673684406	
Peakhour	131.235788	

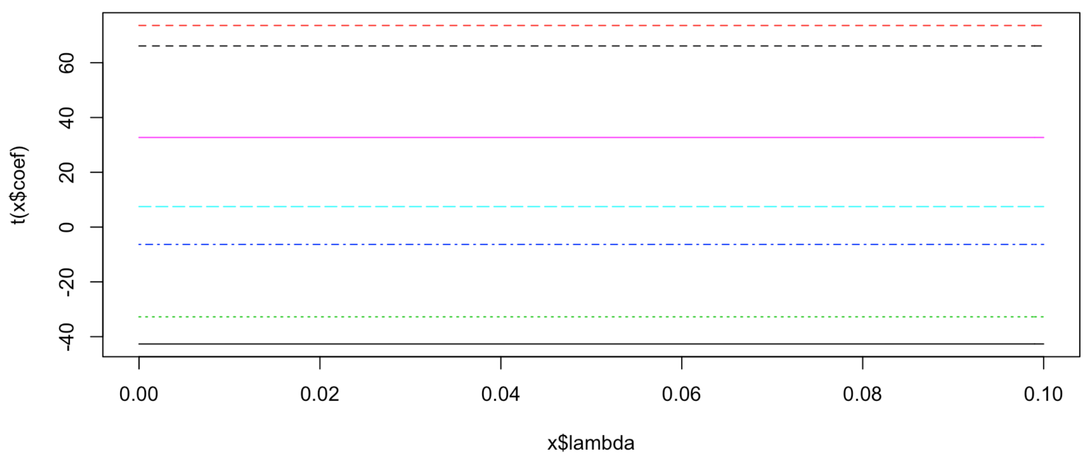
PerformanceMetrics.csv: using broom package

```
#Performance Metrics
tidy_p <- tidy(accuracy(ppp, train$kWh))
tidy_p <- tidy_p[-c(1:2,5)]
tidy_p <- tidyr::gather(tidy_p)
tidy_p <- rbind(c("Account",fulldata_test$Account[1]), tidy_p)
write.table(tidy_p, file = "PerformanceMetrics.csv", sep = "," , row.names = FALSE, col.names = FALSE)
```

Account	26908650026
RMSE	134.1415962
MAE	105.0416332
MAPE	66.6535353

Ridge Regression:

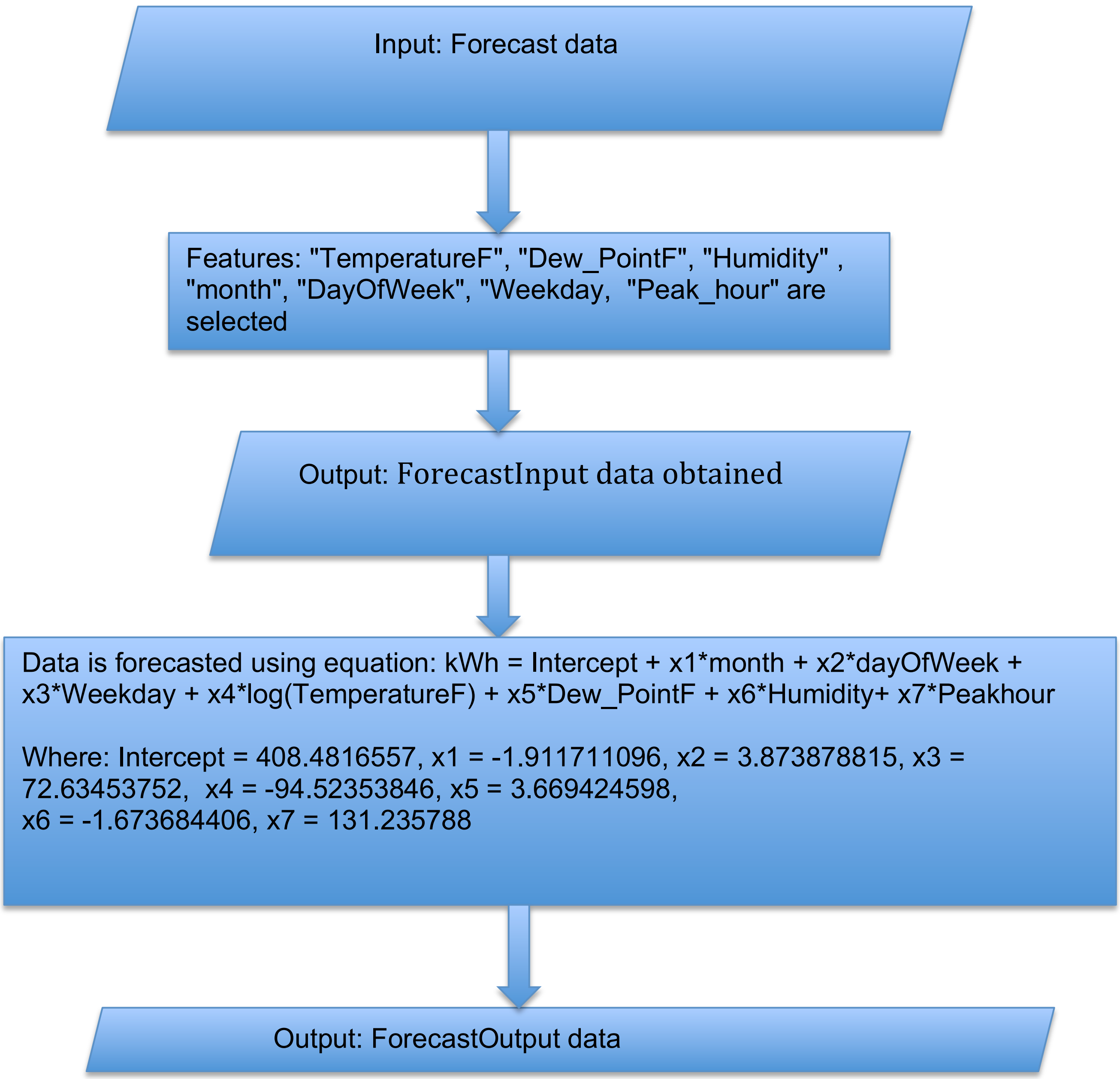
```
#Ridge
lm.ridge(kWh ~ TemperatureF+Dew_PointF+ Humidity+month+dayOfWeek+Weekday+Peakhour,fulldata_test )
plot(lm.ridge(kWh ~ TemperatureF+Dew_PointF+ Humidity+month+dayOfWeek+Weekday+Peakhour ,fulldata_test,lambda = seq(0,0.1,0.001)))
```



There were large number of variables, high collinearity and hence adopted ridge regression.

Lasso will not necessarily yield good results in presence of high collinearity as the performance of the lasso will be dominated by ridge regression. Lasso selects only one variable among a group of predictors with high pairwise correlations.

Part1.3 Forecast



```
ForeCastData <- read.csv("forecastData.csv")
ForeCastData <- separate(ForeCastData, Time, into = c("Date", "Time"), sep=" ")
library(chron)
ForeCastData$Time <- times(ForeCastData$Time)
ForeCastData <- mutate(ForeCastData, Date = as.Date(ForeCastData$Date , "%Y-%m-%d"))
ForeCastData <- separate(ForeCastData, Time, into = c("hour"), sep=":", remove = FALSE)
ForeCastData <- transform(ForeCastData, hour = as.numeric(hour))
ForeCastData <- dplyr::select(ForeCastData,TemperatureF,Dew_PointF, Humidity, hour, Date)
hghghgj <- ForeCastData
ForeCastData <- ForeCastData %>% group_by(Date, hour) %>% summarise(TemperatureF = mean(TemperatureF),
                Dew_PointF = mean(Dew_PointF), Humidity = mean(Humidity))
ForeCastData$month <- as.numeric(format(ForeCastData$Date, format = "%m"))
ForeCastData$DayOfWeek <- as.numeric(format(ForeCastData$Date, format = "%w"))
ForeCastData <- mutate(ForeCastData, Weekday = ifelse(DayOfWeek %in% 1:5 , 1, 0))
ForeCastData <- mutate(ForeCastData, Peak_hour = ifelse(hour %in% 7:18 , 1, 0))
#ForeCastData <- ForeCastData[-c(1:2)]
write.csv(ForeCastData, "Forecast_Input.csv")


ForecastDataOutput <- ForeCastData
tidy_lmfit$estimate <- as.numeric(tidy_lmfit$estimate)
ForecastDataOutput$kWh <- tidy_lmfit[2,2] + (tidy_lmfit[3,2]*ForeCastData$month) +
  (tidy_lmfit[4,2]*ForeCastData$DayOfWeek)+ (tidy_lmfit[5,2]*ForeCastData$Weekday) +
  (tidy_lmfit[6,2]*log(ForeCastData$TemperatureF)) + (tidy_lmfit[7,2]*ForeCastData$Dew_PointF) +
  (tidy_lmfit[8,2]*ForeCastData$Humidity)+ (tidy_lmfit[9,2]*ForeCastData$Peak_hour)

write.csv(ForecastDataOutput, sprintf("ForecastData_Output_Account_%s.csv",tidy_lmfit[1,2]))
```

github link: https://github.com/pagarwal123/Assignment2_Team12