Case 1: Energy Forecasting

Part 1: Algorithm Implementation

1. The raw data you have been provided is real data collected in 2014 at the Jeremiah Burke school. See **RawData.csv.** You will need to clean the data and modify the format in the format given to you. (**sample format.csv)**

Answer

1. This code is used to filter out Powerfactor and KVARh in Units Column. So that we get 365 rows.

```
data<- read.csv(file.choose(),header = T)

df <- data.frame(data)

data_rawdata<-df[df$Units!= 'Power Factor' & df$Units!= 'kVARh', ]

attach(data_rawdata)

View(data_rawdata)
```

	Account	Date	Channel	Units	X0.05	X0.10	X0.15	X0.20	X0.25	X0.30
1	26435791004	1/1/2014	605105283 1 kWh	kWh	7.440	6.960	6.510000	6.480	6.600	6.720
4	26435791004	1/2/2014	605105283 1 kWh	kWh	7.290	7.335	7.215000	7.260	7.065	7.395
7	26435791004	1/3/2014	605105283 1 kWh	kWh	7.155	7.320	7.320000	7.185	7.320	7.215
10	26435791004	1/4/2014	605105283 1 kWh	kWh	6.990	6.975	6.810000	6.810	7.080	6.915
13	26435791004	1/5/2014	605105283 1 kWh	kWh	16.800	16.890	15.420000	15.675	15.015	15.195
16	26435791004	1/6/2014	605105283 1 kWh	kWh	6.630	6.450	6.735000	6.765	6.495	6.540
19	26435791004	1/7/2014	605105283 1 kWh	kWh	7.575	7.170	7.230000	7.320	7.335	7.560
22	26435791004	1/8/2014	605105283 1 kWh	kWh	7.215	7.125	7.500000	7.665	7.515	7.035
25	26435791004	1/9/2014	605105283 1 kWh	kWh	7.620	7.530	7.620000	7.605	7.590	7.665
28	26435791004	1/10/2014	605105283 1 kWh	kWh	6.810	6.840	6.915000	7.530	7.545	7.020
31	26435791004	1/11/2014	605105283 1 kWh	kWh	6.645	6.795	6.705000	7.155	7.695	7.245

Showing 1 to 12 of 365 entries

2. This code is used to aggregate hourly sum of 12 observation

```
aggregate_12<-sapply(seq(5,292,by=12),function(i) rowSums(data_rawdata[,i:(i+11)]))

View(aggregate_12)

nrow(aggregate_12)

data_rawdata_dframe<-cbind(data_rawdata[1:4],aggregate_12)

View(data_rawdata_dframe)
```

81.285	81.705	81.510	83.115	83.835	85.530	151.695	153.765	151.080	151.050	151.350	151.080
87.465	86.700	88.140	93.525	213.870	212.460	254.370	251.640	257.970	257.625	258.225	255.705
86.970	87.615	87.195	88.320	87.195	93.795	258.480	260.235	254.595	250.530	247.500	245.100
82.920	83.670	82.965	83.775	114.810	211.350	215.205	204.525	200.130	197.445	192.630	200.265
190.470	188.070	190.350	188.460	189.375	189.240	196.155	191.220	187.740	185.730	184.035	182.655
81.480	81.270	80.820	80.850	140.445	177.705	225.180	242.685	255.495	267.195	270.945	266.235
88.005	87.060	87.780	88.800	88.605	92.580	251.445	284.655	291.870	300.525	304.800	308.430
88.500	88.230	88.395	89.460	88.395	93.255	260.685	283.020	289.110	294.555	301.620	300.960
91.260	90.705	91.080	91.410	91.245	97.380	261.795	285.120	295.665	298.500	293.970	295.905
84.780	83.820	83.520	85.080	85.620	88.425	265.965	276.015	298.800	305.775	311.895	306.930
82.500	82.275	81.345	82.155	81.615	81.870	162.465	149.295	184.890	185.790	177.000	173.685

3. This code is used to transpose the data using melt function (8760 entries)

install.packages("reshape")

library(reshape)

data_rawdata_meltdata <- melt(data_rawdata_dframe, id=c("Account","Date","Channel","Units")

View(data_rawdata_meltdata)

	Account	Date	Channel	Units	variable	value
1	26435791004	1/1/2014	605105283 1 kWh	kWh	1	81.285
2	26435791004	1/2/2014	605105283 1 kWh	kWh	1	87.465
3	26435791004	1/3/2014	605105283 1 kWh	kWh	1	86.970
4	26435791004	1/4/2014	605105283 1 kWh	kWh	1	82.920
5	26435791004	1/5/2014	605105283 1 kWh	kWh	1	190.470
6	26435791004	1/6/2014	605105283 1 kWh	kWh	1	81.480
7	26435791004	1/7/2014	605105283 1 kWh	kWh	1	88.005
8	26435791004	1/8/2014	605105283 1 kWh	kWh	1	88.500
9	26435791004	1/9/2014	605105283 1 kWh	kWh	1	91.260
10	26435791004	1/10/2014	605105283 1 kWh	kWh	1	84.780
11	26435791004	1/11/2014	605105283 1 kWh	kWh	1	82.500
12	26435791004	1/12/2014	605105283 1 kWh	kWh	1	83.265

ing 1 to 12 of 8,760 entries

4. This code is used to order all the detail according to respective dates

install.packages("doBy")

library(doBy)

ordereddata <- data_rawdata_meltdata[order(data_rawdata_meltdata$Date),]

View(ordereddata)

attach(ordereddata)

	Account	Date	Channel	Units	variable	value
1	26435791004	1/1/2014	605105283 1 kWh	kWh	1	81.285
366	26435791004	1/1/2014	605105283 1 kWh	kWh	2	81.705
731	26435791004	1/1/2014	605105283 1 kWh	kWh	3	81.510
1096	26435791004	1/1/2014	605105283 1 kWh	kWh	4	83.115
1461	26435791004	1/1/2014	605105283 1 kWh	kWh	5	83.835
1826	26435791004	1/1/2014	605105283 1 kWh	kWh	6	85.530
2191	26435791004	1/1/2014	605105283 1 kWh	kWh	7	151.695
2556	26435791004	1/1/2014	605105283 1 kWh	kWh	8	153.765
2921	26435791004	1/1/2014	605105283 1 kWh	kWh	9	151.080
3286	26435791004	1/1/2014	605105283 1 kWh	kWh	10	151.050
3651	26435791004	1/1/2014	605105283 1 kWh	kWh	11	151.350
4016	26435791004	1/1/2014	605105283 1 kWh	kWh	12	151.080

Showing 1 to 12 of 8,760 entries

5. This code is used to display month , date, year and weekday

```
install.packages("lubridate")
library(lubridate)
ordereddata_year<-year(as.Date(ordereddata$Date, format="%m/%d/%Y"))
View(ordereddata_year)
ordereddata_month<-month(as.Date(ordereddata$Date, format="%m/%d/%Y"))
View(ordereddata_month)
ordereddata_day<-day(as.Date(ordereddata$Date, format="%m/%d/%Y"))
View(ordereddata_day)
ordereddata_weekday <-(wday(as.Date(ordereddata$Date, format="%m/%d/%Y")))-1


##dd<-as.Date(newdata$Date)
      ordereddata1<-
cbind(ordereddata[1:6],ordereddata_year,ordereddata_month,ordereddata_day,ordereddata_weekday)
View(ordereddata1)
```

s	variable	value	ordereddata_year	ordereddata_month	ordereddata_day	ordereddata_weekday
	1	81.285	2014	1	1	3
	2	81.705	2014	1	1	3
	3	81.510	2014	1	1	3
	4	83.115	2014	1	1	3
	5	83.835	2014	1	1	3
	6	85.530	2014	1	1	3
	7	151.695	2014	1	1	3
	8	153.765	2014	1	1	3
	9	151.080	2014	1	1	3
	10	151.050	2014	1	1	3
	11	151.350	2014	1	1	3

6. This code is used to display weekend ,peakhours and hour

```
ordereddata_weekend<-NA

ordereddata2<-cbind(ordereddata1,ordereddata_weekend)

ordereddata_weekend<- ifelse( ordereddata1$ordereddata_weekday == 0 |
    ordereddata1$ordereddata_weekday == 6 , 0 , 1)

##warnings()

ordereddata2<-cbind(ordereddata1,ordereddata_weekend)

View(ordereddata2)


ordereddata_peakhours <-NA

ordereddata3<-cbind(ordereddata1,ordereddata_weekend,ordereddata_peakhours)

ordereddata_peakhours <- ifelse( as.numeric(ordereddata1$variable) > 6 &
    as.numeric(ordereddata1$variable) < 20 , 0 , 1)

##Dates <- as.Date(newdata$Date, "%m/%d/%Y")

FinalDate <- as.Date(ordereddata$Date, format="%m/%d/%Y")

Hour<-as.numeric(ordereddata3$variable)-1

View(Hour)

ordereddata3<-cbind(ordereddata1,ordereddata_weekend,ordereddata_peakhours,FinalDate,Hour)

View(ordereddata3)
```

reddata_day	ordereddata_weekday	ordereddata_weekend	ordereddata_peakhours	FinalDate	Hour
1	3	1	1	2014-01-01	0
1	3	1	1	2014-01-01	1
1	3	1	1	2014-01-01	2
1	3	1	1	2014-01-01	3
1	3	1	1	2014-01-01	4
1	3	1	1	2014-01-01	5
1	3	1	0	2014-01-01	6
1	3	1	0	2014-01-01	7
1	3	1	0	2014-01-01	8
1	3	1	0	2014-01-01	9
1	3	1	0	2014-01-01	10

7. This code is to get data from weatherData..

   install.packages("weatherData")

   library(weatherData)

   weather_data<-getWeatherForDate("KBOS","2014-1-1","2014-12-31", opt_detailed = TRUE,opt_all_columns=TRUE)

   View(weather_data)

2014-01-01 00:54:00	12:54 AM	23.0	5.0	46	30.20	
2014-01-01 01:54:00	1:54 AM	21.9	3.9	46	30.23	
2014-01-01 02:54:00	2:54 AM	21.9	3.9	46	30.25	
2014-01-01 03:54:00	3:54 AM	21.9	3.0	44	30.27	
2014-01-01 04:54:00	4:54 AM	21.0	3.0	46	30.29	
2014-01-01 05:54:00	5:54 AM	21.0	3.0	46	30.30	
2014-01-01 06:54:00	6:54 AM	21.0	3.9	48	30.32	
2014-01-01 07:54:00	7:54 AM	19.9	5.0	52	30.35	
2014-01-01 08:54:00	8:54 AM	23.0	6.1	48	30.37	
2014-01-01 09:54:00	9:54 AM	24.1	5.0	44	30.39	
2014-01-01 10:54:00	10:54 AM	26.1	5.0	41	30.35	

8. This code is to aggregate temperature from weatherData which has 8687 rows and hours

   weather_data_aggregate<-aggregate(list(temperature = weather_data$TemperatureF), list(Time = cut(weather_data$Time, "1 hour")), mean)

   View(weather_data_aggregate)

   weather_hour <- format(as.POSIXct(weather_data_aggregate$Time),"%H")

   Hour<- as.numeric(weather_hour)

   View(Hour)

finalweatherdata = cbind(weather_data_aggregate,Hour)

View(finalweatherdata)

	Time	temperature
1	2014-01-01 00:00:00	23.000000
2	2014-01-01 01:00:00	21.900000
3	2014-01-01 02:00:00	21.900000
4	2014-01-01 03:00:00	21.900000
5	2014-01-01 04:00:00	21.000000
6	2014-01-01 05:00:00	21.000000
7	2014-01-01 06:00:00	21.000000
8	2014-01-01 07:00:00	19.900000
9	2014-01-01 08:00:00	23.000000
10	2014-01-01 09:00:00	24.100000
11	2014-01-01 10:00:00	26.100000
12	2014-01-01 11:00:00	26.100000

Showing 1 to 12 of 8,687 entries

	Time	temperature	Hour
1	2014-01-01 00:00:00	23.000000	0
2	2014-01-01 01:00:00	21.900000	1
3	2014-01-01 02:00:00	21.900000	2
4	2014-01-01 03:00:00	21.900000	3
5	2014-01-01 04:00:00	21.000000	4
6	2014-01-01 05:00:00	21.000000	5
7	2014-01-01 06:00:00	21.000000	6
8	2014-01-01 07:00:00	19.900000	7
9	2014-01-01 08:00:00	23.000000	8
10	2014-01-01 09:00:00	24.100000	9
11	2014-01-01 10:00:00	26.100000	10
12	2014-01-01 11:00:00	26.100000	11

9. This code is to get Date from Day

final_rawdata<-data.frame(ordereddata3)

View(final_rawdata)

FinalDate <- as.Date(finalweatherdata$Time)

final_weatherdata<-cbind(finalweatherdata,FinalDate)

colnames(final_weatherdata)<-c("Time","Temperature","Hour","FinalDate")

View(final_weatherdata)

	Time	Temperature	Hour	FinalDate
1	2014-01-01 00:00:00	23.000000	0	2014-01-01
2	2014-01-01 01:00:00	21.900000	1	2014-01-01
3	2014-01-01 02:00:00	21.900000	2	2014-01-01
4	2014-01-01 03:00:00	21.900000	3	2014-01-01
5	2014-01-01 04:00:00	21.000000	4	2014-01-01
6	2014-01-01 05:00:00	21.000000	5	2014-01-01
7	2014-01-01 06:00:00	21.000000	6	2014-01-01
8	2014-01-01 07:00:00	19.900000	7	2014-01-01
9	2014-01-01 08:00:00	23.000000	8	2014-01-01
10	2014-01-01 09:00:00	24.100000	9	2014-01-01
11	2014-01-01 10:00:00	26.100000	10	2014-01-01
12	2014-01-01 11:00:00	26.100000	11	2014-01-01

10. This code is used to merge two table based on 2 common columns. Hour and Temperature. But it has 80 rows NA

    mergeddata<-merge(x = final_rawdata,final_weatherdata,by=c("FinalDate","Hour"),all.x=TRUE

    View(mergeddata)

	Account	Date	Channel	Units	variable	value	ordereddata_year	ordereddata_
1	26435791004	1/1/2014	605105283 1 kWh	kWh	1	81.285	2014	
366	26435791004	1/1/2014	605105283 1 kWh	kWh	2	81.705	2014	
731	26435791004	1/1/2014	605105283 1 kWh	kWh	3	81.510	2014	
1096	26435791004	1/1/2014	605105283 1 kWh	kWh	4	83.115	2014	
1461	26435791004	1/1/2014	605105283 1 kWh	kWh	5	83.835	2014	
1826	26435791004	1/1/2014	605105283 1 kWh	kWh	6	85.530	2014	
2191	26435791004	1/1/2014	605105283 1 kWh	kWh	7	151.695	2014	
2556	26435791004	1/1/2014	605105283 1 kWh	kWh	8	153.765	2014	
2921	26435791004	1/1/2014	605105283 1 kWh	kWh	9	151.080	2014	
3286	26435791004	1/1/2014	605105283 1 kWh	kWh	10	151.050	2014	
3651	26435791004	1/1/2014	605105283 1 kWh	kWh	11	151.350	2014	

Showing 1 to 12 of 8,760 entries

11. This code is used to replace N/A with the mean of the previous column

    mergeddata[is.na(mergeddata)] <- 22

    final<-cbind(mergeddata[2:14],mergeddata$Temperature)

    View(final)

	Hour	Account	Date	Channel	Units	variable	value	ordereddata_year	ordered
1	0	26435791004	1/1/2014	605105283 1 kWh	kWh	1	81.285	2014	
2	1	26435791004	1/1/2014	605105283 1 kWh	kWh	2	81.705	2014	
3	2	26435791004	1/1/2014	605105283 1 kWh	kWh	3	81.510	2014	
4	3	26435791004	1/1/2014	605105283 1 kWh	kWh	4	83.115	2014	
5	4	26435791004	1/1/2014	605105283 1 kWh	kWh	5	83.835	2014	
6	5	26435791004	1/1/2014	605105283 1 kWh	kWh	6	85.530	2014	
7	6	26435791004	1/1/2014	605105283 1 kWh	kWh	7	151.695	2014	
8	7	26435791004	1/1/2014	605105283 1 kWh	kWh	8	153.765	2014	
9	8	26435791004	1/1/2014	605105283 1 kWh	kWh	9	151.080	2014	
10	9	26435791004	1/1/2014	605105283 1 kWh	kWh	10	151.050	2014	
11	10	26435791004	1/1/2014	605105283 1 kWh	kWh	11	151.350	2014	

Finally writing it to csv file

```
sampleformat<-cbind(final[2:3],KWh=final$value,month=final$ordereddata_month,day =
final$ordereddata_day,year= final$ordereddata_year,hour= final$Hour,DayofWeek =
final$ordereddata_weekday, WeekDay = final$ordereddata_weekend, Peakhour =
final$ordereddata_peakhours,Temperature = mergeddata$Temperature)

View(sampleformat)

write.csv(mergeddata,file="sampleformat_Final.csv")
```

PART 1 -> Q.2

Multiple-Linear Regression

- Implementing a multiple linear regression using the cleansed data as the input (sample format.csv). Review the different concepts we discussed in class (feature selection, feature transformation etc.) and create the best possible implementation. Evaluate the performance metrics and create a model that produces the best possible Regression coefficients you can.

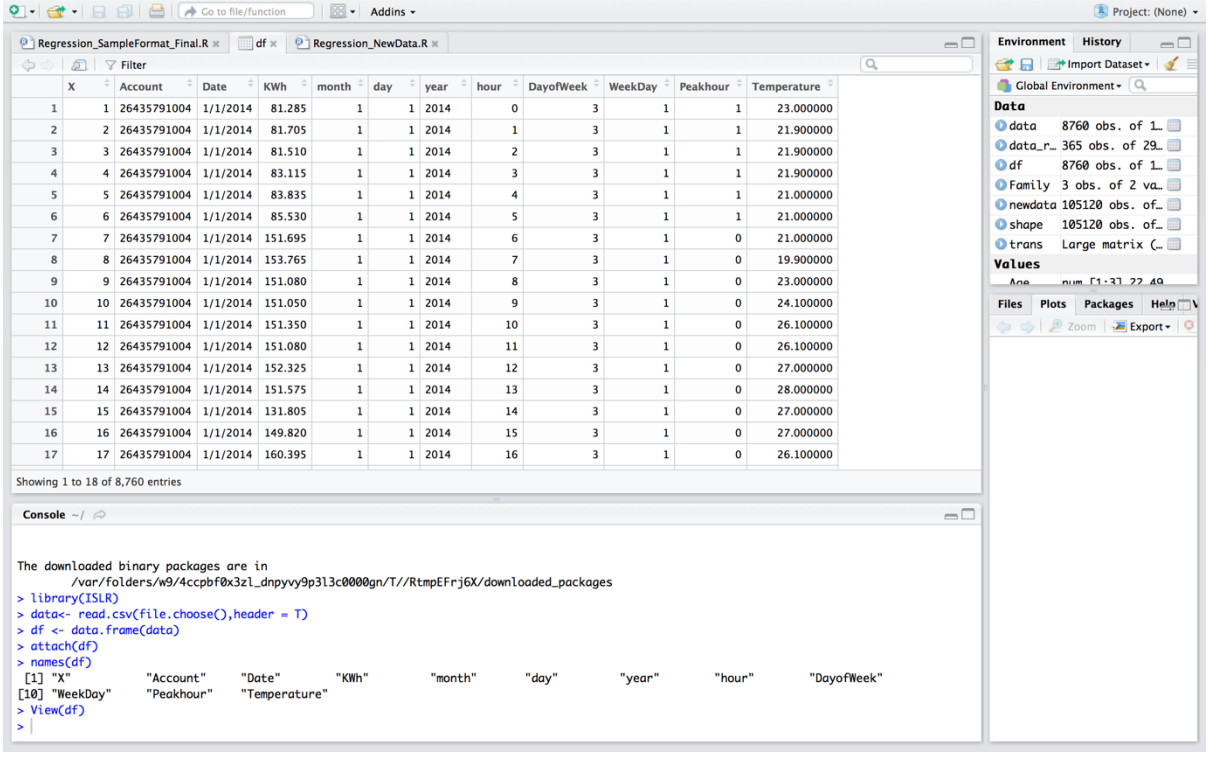
- A script that can take any input file in the format sample format.csv and outputs the regression coefficients into a text file. The script also computes the RMS error, MAPE and MAE for your model and output to a text file.

Answer:

Data used : sampleformat_Final.csv

1) We include the 2 libraries namely MASS and ISLR for building the regression model and then we read the cleansed csv file named **sampleformat_Final.csv.** We view the file using the View() function and then we also display the names of the variables in the data file.

```
library(MASS)
install.packages("ISLR")
library(ISLR)
data<- read.csv(file.choose(),header = T)
df <- data.frame(data)
View(df)
```



	X	Account	Date	KWh	month	day	year	hour	DayofWeek	WeekDay	Peakhour	Temperature
1	1	26435791004	1/1/2014	81.285	1	1	2014	0	3	1	1	23.000000
2	2	26435791004	1/1/2014	81.705	1	1	2014	1	3	1	1	21.900000
3	3	26435791004	1/1/2014	81.510	1	1	2014	2	3	1	1	21.900000
4	4	26435791004	1/1/2014	83.115	1	1	2014	3	3	1	1	21.900000
5	5	26435791004	1/1/2014	83.835	1	1	2014	4	3	1	1	21.000000
6	6	26435791004	1/1/2014	85.530	1	1	2014	5	3	1	1	21.000000
7	7	26435791004	1/1/2014	151.695	1	1	2014	6	3	1	0	21.000000
8	8	26435791004	1/1/2014	153.765	1	1	2014	7	3	1	0	19.900000
9	9	26435791004	1/1/2014	151.080	1	1	2014	8	3	1	0	23.000000
10	10	26435791004	1/1/2014	151.050	1	1	2014	9	3	1	0	24.100000
11	11	26435791004	1/1/2014	151.350	1	1	2014	10	3	1	0	26.100000
12	12	26435791004	1/1/2014	151.080	1	1	2014	11	3	1	0	26.100000
13	13	26435791004	1/1/2014	152.325	1	1	2014	12	3	1	0	27.000000
14	14	26435791004	1/1/2014	151.575	1	1	2014	13	3	1	0	28.000000
15	15	26435791004	1/1/2014	131.805	1	1	2014	14	3	1	0	27.000000
16	16	26435791004	1/1/2014	149.820	1	1	2014	15	3	1	0	27.000000
17	17	26435791004	1/1/2014	160.395	1	1	2014	16	3	1	0	26.100000

2) Next, we take out the 75% data to divide it into training and test data named as train and test.

```
#75% of the sample size
smp_size <- floor(0.75 * nrow(df))

#Set the seed to make your partition reproductible
set.seed(123)
train_ind <- sample(seq_len(nrow(df)), size = smp_size)

#Split the data into training and testing
```

train <- df[train_ind,]

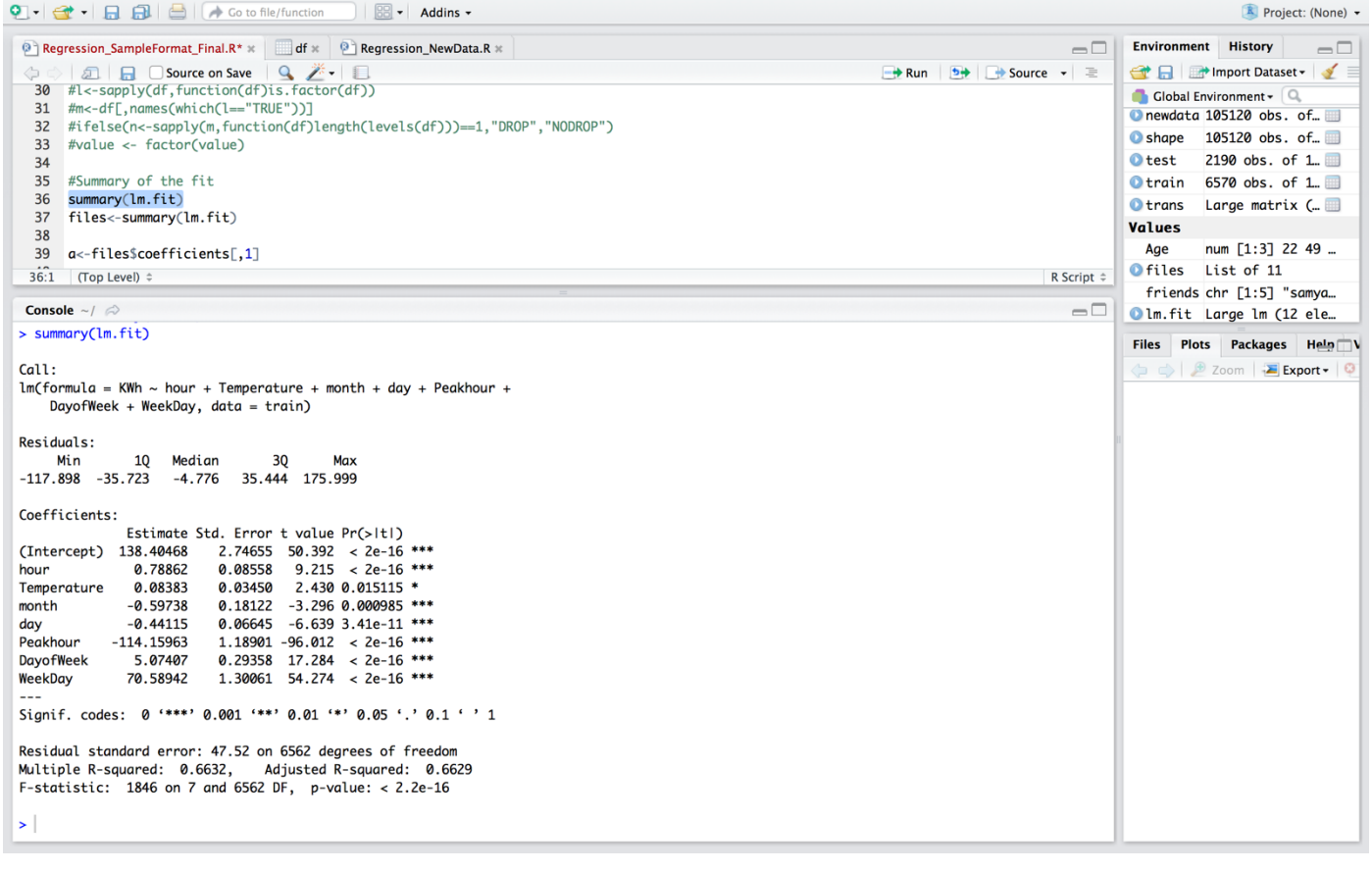
test <- df[-train_ind,]

3) Here, we write the function for finding the fit for the building of linear regression model. We use lm.fit() function for finding the best fit for our data. The column KWh is dependent on Temperature, hour, month,day,Peakhour,DayofWeek and weekday. We use train data obtained in step 2.

```
#Fit a linear regression model

lm.fit = lm(KWh ~ hour+Temperature+month+day+Peakhour+DayofWeek+WeekDay, data = train)
```

```
#Summary of the fit

files<-summary(lm.fit)
```

The Multiple R-squared value is 0.6632



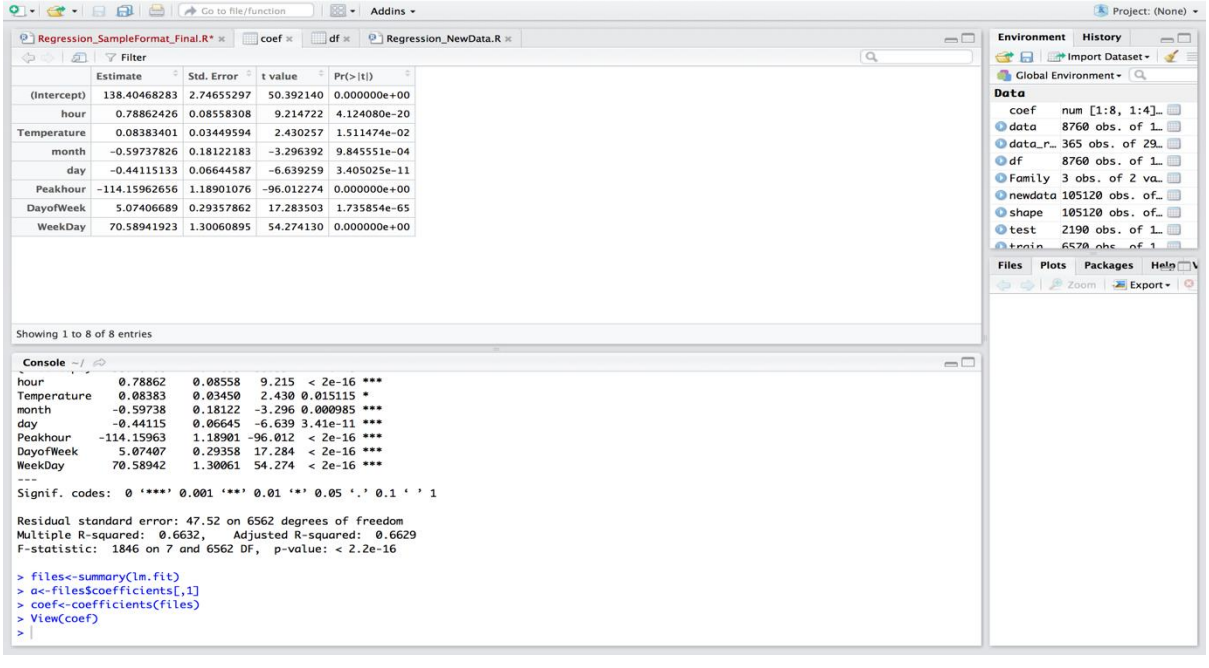
4) The summary is inserted into the files variables and then we format it to get the first reading. Finally, we viewed the coefficients. This is then written in a csv file named **Coefficients_one(SampleData).csv**

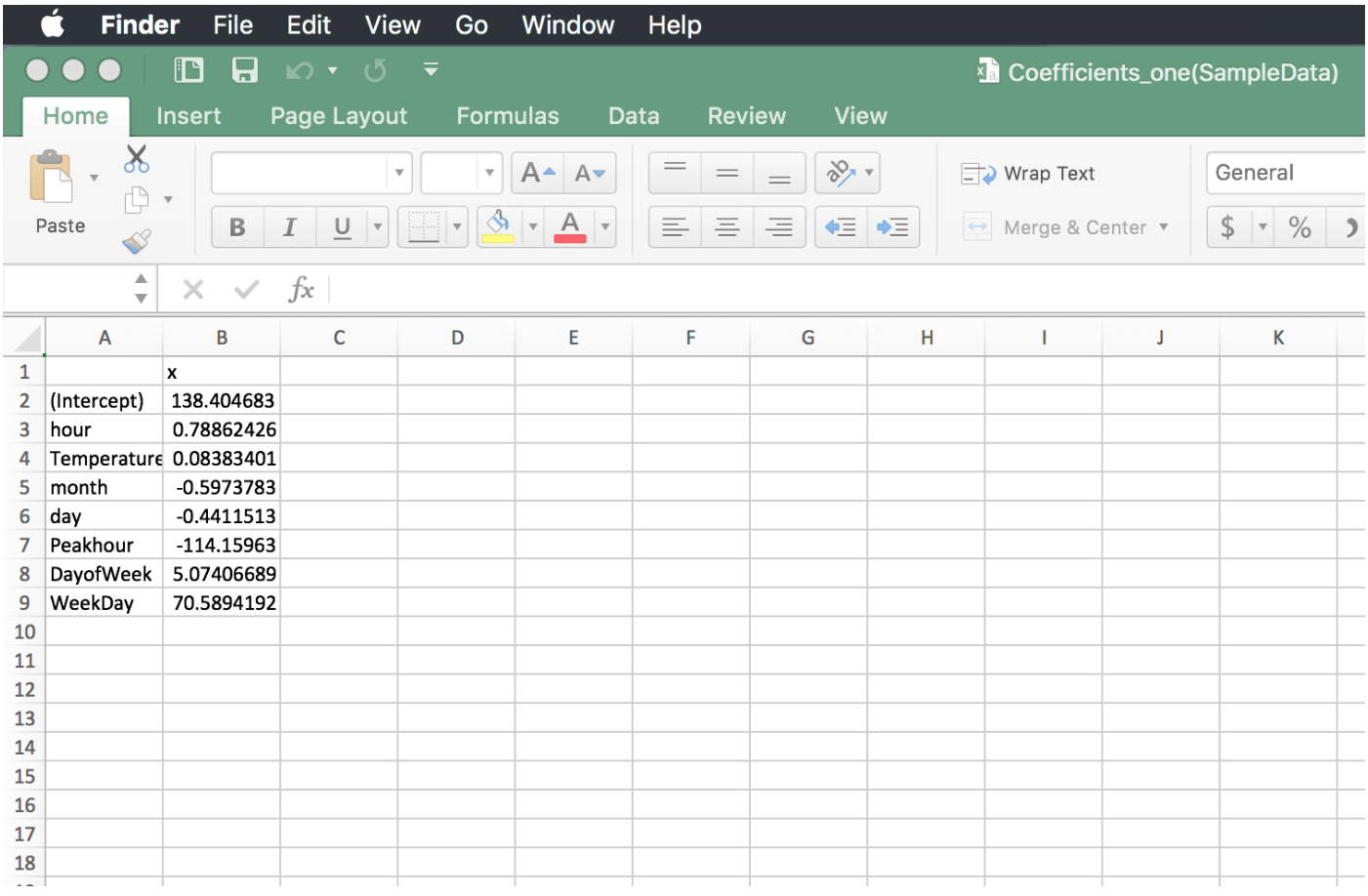
a<-files$coefficients[,1]

coef<-coefficients(files)

View(coef)

write.csv(a, file="Coefficients_one(SampleData).csv")

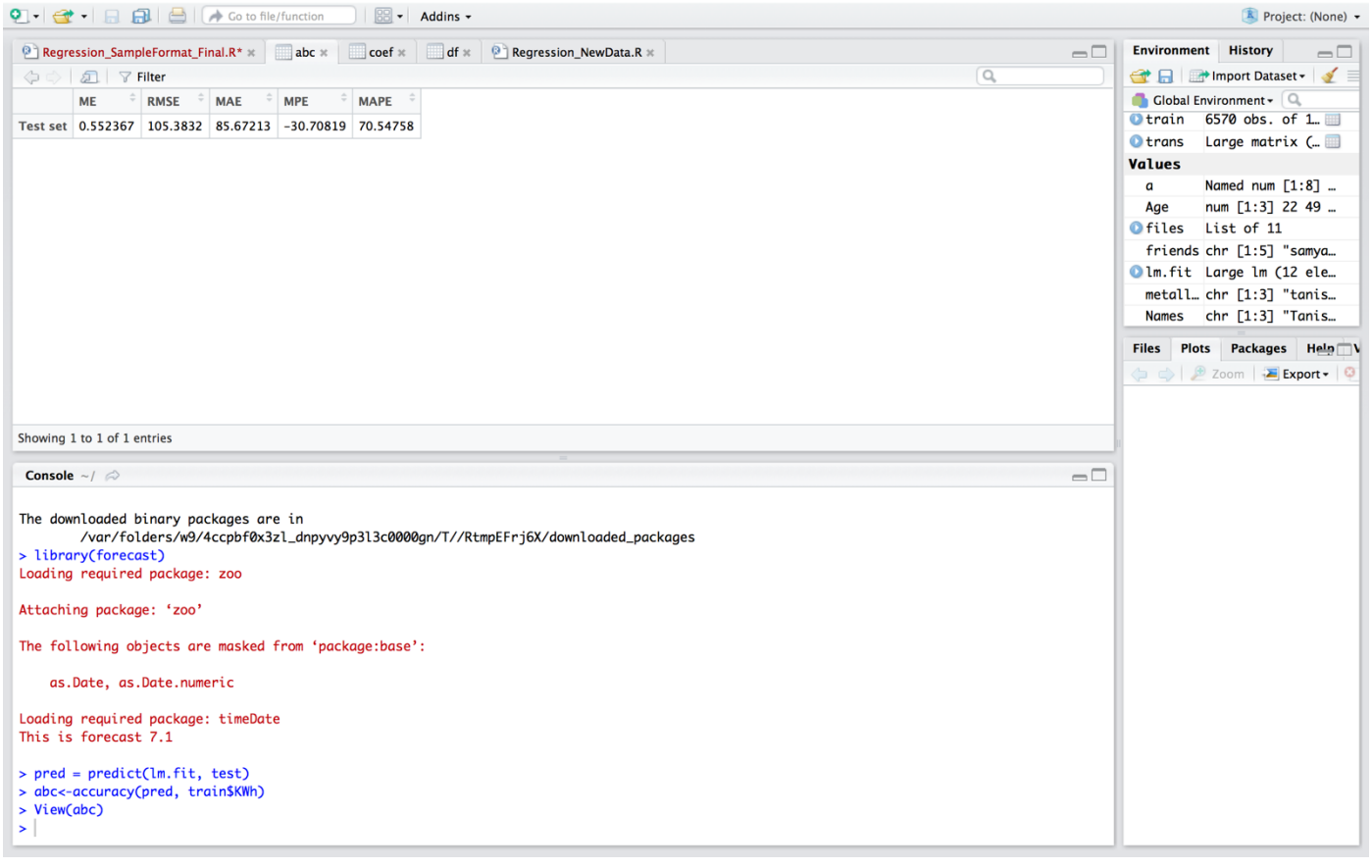




	A	B
1		x
2	(Intercept)	138.404683
3	hour	0.78862426
4	Temperature	0.08383401
5	month	-0.5973783
6	day	-0.4411513
7	Peakhour	-114.15963
8	DayofWeek	5.07406689
9	WeekDay	70.5894192

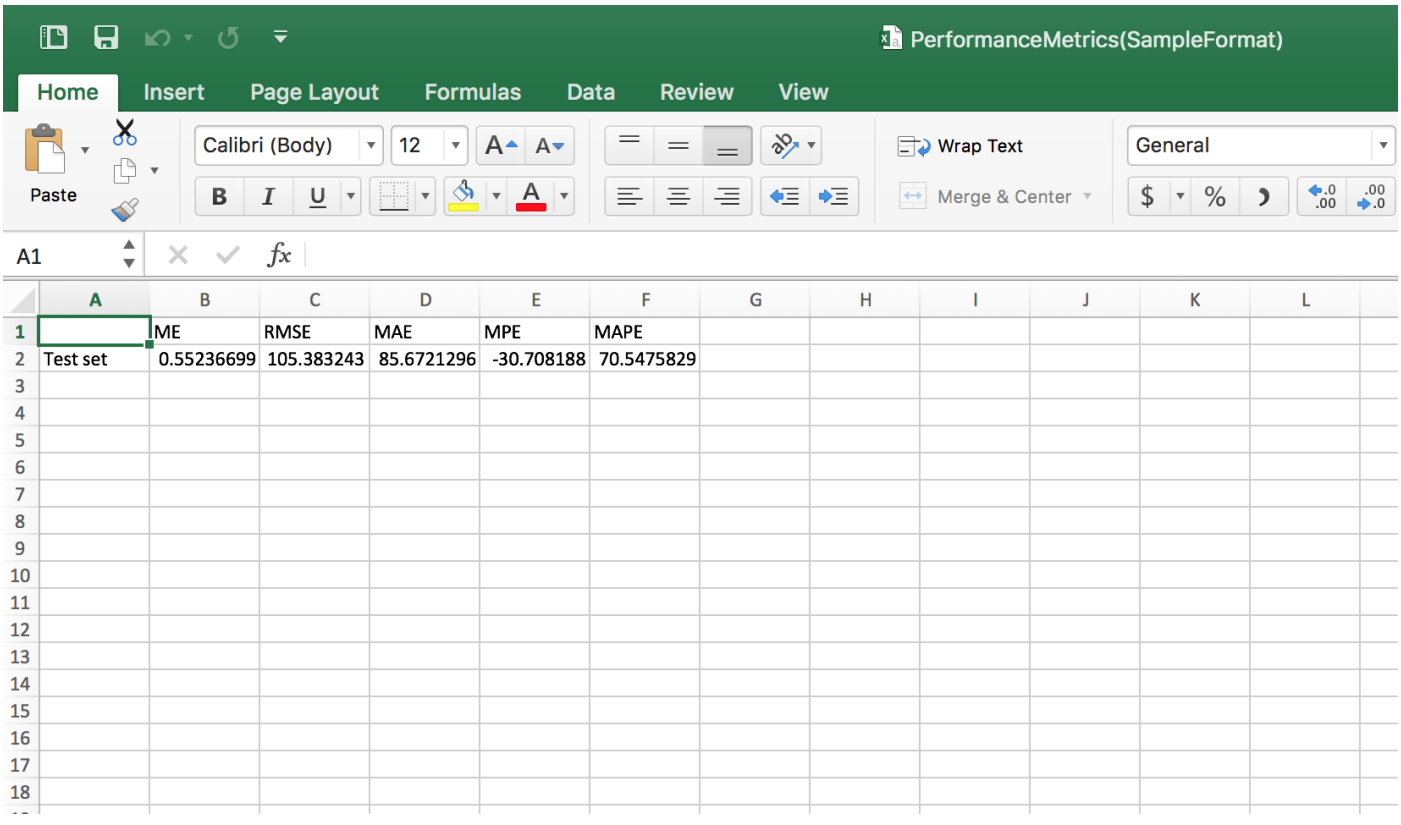
5) Next, we predict the accuracy and measure the Performance metrics by using the forecast library and the predict() function. Later, we view the performance metrics using View() function.

```
#Measures of predictive accuracy

install.packages("forecast")

library(forecast)

pred = predict(lm.fit, test)

abc<-accuracy(pred, train$KWh)

View(abc)
```



6) Finally, we write the R-script content in csv file using write.csv() function. The file name is PerformanceMetrics(SampleFormat).csv

```
write.csv(abc, file="PerformanceMetrics(SampleFormat).csv")
```



	A	B	C	D	E	F
1		ME	RMSE	MAE	MPE	MAPE
2	Test set	0.55236699	105.383243	85.6721296	-30.708188	70.5475829

PART 1 -> Q3

Forecast

a. Convert the format to the format in forecastInput.csv

Answer:

1. This code is used to load the data from csv file and insert the ouput of csv in the data frame

   ```
   data<- read.csv(file.choose(),header = T)
   df <- data.frame(data)
   ```

2. Lubridate package and library is used for parsing the date into different formats. For ex: month out of date

   ```
   install.packages("lubridate")
   library(lubridate)
   ```

3. This code is used to parse the date into month, year, date and weekday as per date. It also depicts how to bind the columns using cbind() function.

   ```
   forecast_Year<-year(as.Date(df$Day, format="%m/%d/%Y"))
   forecast_Month<-month(as.Date(df$Day, format="%m/%d/%Y"))
   forecast_Day<-day(as.Date(df$Day, format="%m/%d/%Y"))
   forecastdata<-cbind(df[1],forecast_Month,forecast_Day,forecast_Year,df[2])
   View(forecastdata)

   ordereddata_weekday <-(wday(as.Date(forecastdata$Day, format="%m/%d/%Y")))-1
   ```

ordereddata1<-cbind(forecastdata,ordereddata_weekday)
View(ordereddata1)

_final | forecastnewdata.R* | forecast.R* | forecastdata | ordereddata1 | df

Filter

	Date	forecast_Month	forecast_Day	forecast_Year	Hour	ordereddata_weekday
1	20160613	6	13	2016	17	1
2	20160613	6	13	2016	18	1
3	20160613	6	13	2016	19	1
4	20160613	6	13	2016	20	1
5	20160613	6	13	2016	21	1
6	20160613	6	13	2016	22	1
7	20160613	6	13	2016	23	1
8	20160614	6	14	2016	0	2
9	20160614	6	14	2016	1	2
10	20160614	6	14	2016	2	2
11	20160614	6	14	2016	3	2
12	20160614	6	14	2016	4	2
13	20160614	6	14	2016	5	2

4. This code is used to provide information whether the specific day of the week is weekend or not. If weekday is 0 or 6, then it's a weekend (0) else it is a weekday.

   ordereddata_weekend<-NA
   ordereddata2<-cbind(ordereddata1,ordereddata_weekend)
   ordereddata_weekend<- ifelse(ordereddata1$ordereddata_weekday == 0 | ordereddata1$ordereddata_weekday == 6 , 0 , 1)
   ordereddata2<-cbind(ordereddata1,ordereddata_weekend)
   View(ordereddata2)

Untitled1 | df_final | forecastnewdata.R* | forecast.R* | ordereddata2

Filter

Month	forecast_Day	forecast_Year	Hour	ordereddata_weekday	ordereddata_weekend
	17	2016	15	5	1
	17	2016	16	5	1
	17	2016	17	5	1
	17	2016	18	5	1
	17	2016	19	5	1
	17	2016	20	5	1
	17	2016	21	5	1
	17	2016	22	5	1
	17	2016	23	5	1
	18	2016	0	6	0
	18	2016	1	6	0
	18	2016	2	6	0
	18	2016	3	6	0

5. This code is used to order to provide output whether the hour is peak hour or not.Condition used in this code is if 7 A.M- 7 P.M = 1 else 7 P.M – 7 A.M = 0

```
ordereddata_peakhours <-NA
finalforecastdata<-cbind(ordereddata1,ordereddata_weekend,ordereddata_peakhours)
ordereddata_peakhours <- ifelse( as.numeric(ordereddata1$Hour) > 6 &  as.numeric(ordereddata1$Hour) < 20 , 1 , 0)
finalforecastdata<-cbind(ordereddata1,ordereddata_weekend,ordereddata_peakhours,df[3])
```

Untitled1 | df_final | forecastnewdata.R* | forecast.R* | finalforecastdata

Filter

	Date	month	Day	Year	Hour	Day of Week	WeekDay	Peakhour	Temperature
1	20160613	6	13	2016	17	1	1	1	71
2	20160613	6	13	2016	18	1	1	1	70
3	20160613	6	13	2016	19	1	1	1	70
4	20160613	6	13	2016	20	1	1	0	69
5	20160613	6	13	2016	21	1	1	0	67
6	20160613	6	13	2016	22	1	1	0	65
7	20160613	6	13	2016	23	1	1	0	63
8	20160614	6	14	2016	0	2	1	0	61
9	20160614	6	14	2016	1	2	1	0	60
10	20160614	6	14	2016	2	2	1	0	59
11	20160614	6	14	2016	3	2	1	0	58
12	20160614	6	14	2016	4	2	1	0	57

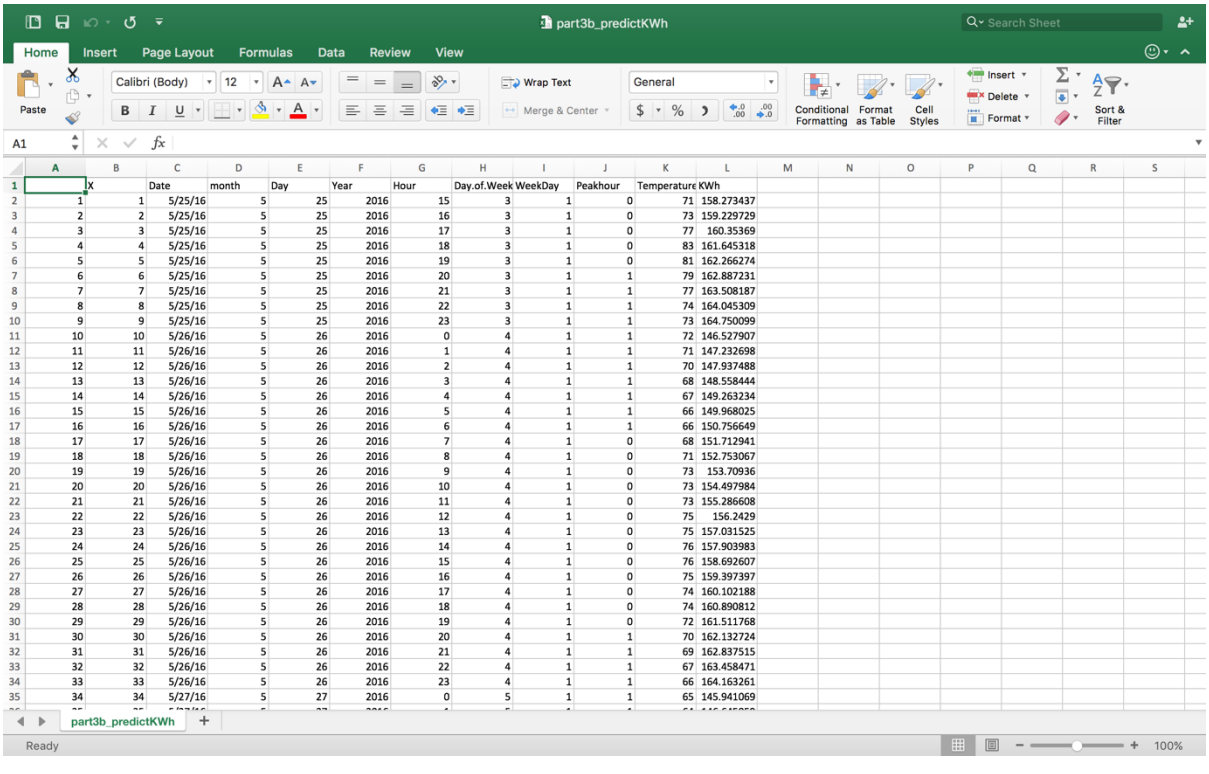
6. This function i.e colnames() is used to provide column names of the dataframe.

   colnames(finalforecastdata)<-c("Date","month","Day","Year", "Hour","Day of Week","WeekDay","Peakhour","Temperature")
   View(finalforecastdata)

	Date	month	Day	Year	Hour	Day of Week	WeekDay	Peakhour	Temperature

PART 1 -> Q3

Use the regression inputs generated in step 2 and the forecastInput.csv to predict the power usage in KWH for each hour.



	X	Date	month	Day	Year	Hour	Day.of.Week	WeekDay	Peakhour	Temperature	KWh
1	1	5/25/16	5	25	2016	15	3	1	0	71	158.273437
2	2	5/25/16	5	25	2016	16	3	1	0	73	159.229729
3	3	5/25/16	5	25	2016	17	3	1	0	77	160.35369
4	4	5/25/16	5	25	2016	18	3	1	0	83	161.645318
5	5	5/25/16	5	25	2016	19	3	1	0	81	162.266274
6	6	5/25/16	5	25	2016	20	3	1	1	79	162.887231
7	7	5/25/16	5	25	2016	21	3	1	1	77	163.508187
8	8	5/25/16	5	25	2016	22	3	1	1	74	164.045309
9	9	5/25/16	5	25	2016	23	3	1	1	73	164.750099
10	10	5/26/16	5	26	2016	0	4	1	1	72	146.527907
11	11	5/26/16	5	26	2016	1	4	1	1	71	147.232698
12	12	5/26/16	5	26	2016	2	4	1	1	70	147.937488
13	13	5/26/16	5	26	2016	3	4	1	1	68	148.558444
14	14	5/26/16	5	26	2016	4	4	1	1	67	149.263234
15	15	5/26/16	5	26	2016	5	4	1	1	66	149.968025
16	16	5/26/16	5	26	2016	6	4	1	1	66	150.756649
17	17	5/26/16	5	26	2016	7	4	1	0	68	151.712941
18	18	5/26/16	5	26	2016	8	4	1	0	71	152.753067
19	19	5/26/16	5	26	2016	9	4	1	0	73	153.70936
20	20	5/26/16	5	26	2016	10	4	1	0	73	154.497984
21	21	5/26/16	5	26	2016	11	4	1	0	73	155.286608
22	22	5/26/16	5	26	2016	12	4	1	0	75	156.2429
23	23	5/26/16	5	26	2016	13	4	1	0	75	157.031525
24	24	5/26/16	5	26	2016	14	4	1	0	76	157.903983
25	25	5/26/16	5	26	2016	15	4	1	0	76	158.692607
26	26	5/26/16	5	26	2016	16	4	1	0	75	159.397397
27	27	5/26/16	5	26	2016	17	4	1	0	74	160.102188
28	28	5/26/16	5	26	2016	18	4	1	0	74	160.890812
29	29	5/26/16	5	26	2016	19	4	1	0	72	161.511768
30	30	5/26/16	5	26	2016	20	4	1	1	70	162.132724
31	31	5/26/16	5	26	2016	21	4	1	1	69	162.837515
32	32	5/26/16	5	26	2016	22	4	1	1	67	163.458471
33	33	5/26/16	5	26	2016	23	4	1	1	66	164.163261
34	34	5/27/16	5	27	2016	0	5	1	1	65	145.941069

PART 2 -> Q1

A raw data file from another school (similar to the raw data.csv file). Your code should take this file and the station code (for example KBOS) and run steps 1 & 2 above and generate the regression coefficients and performance metrics for this new file.

Answer:

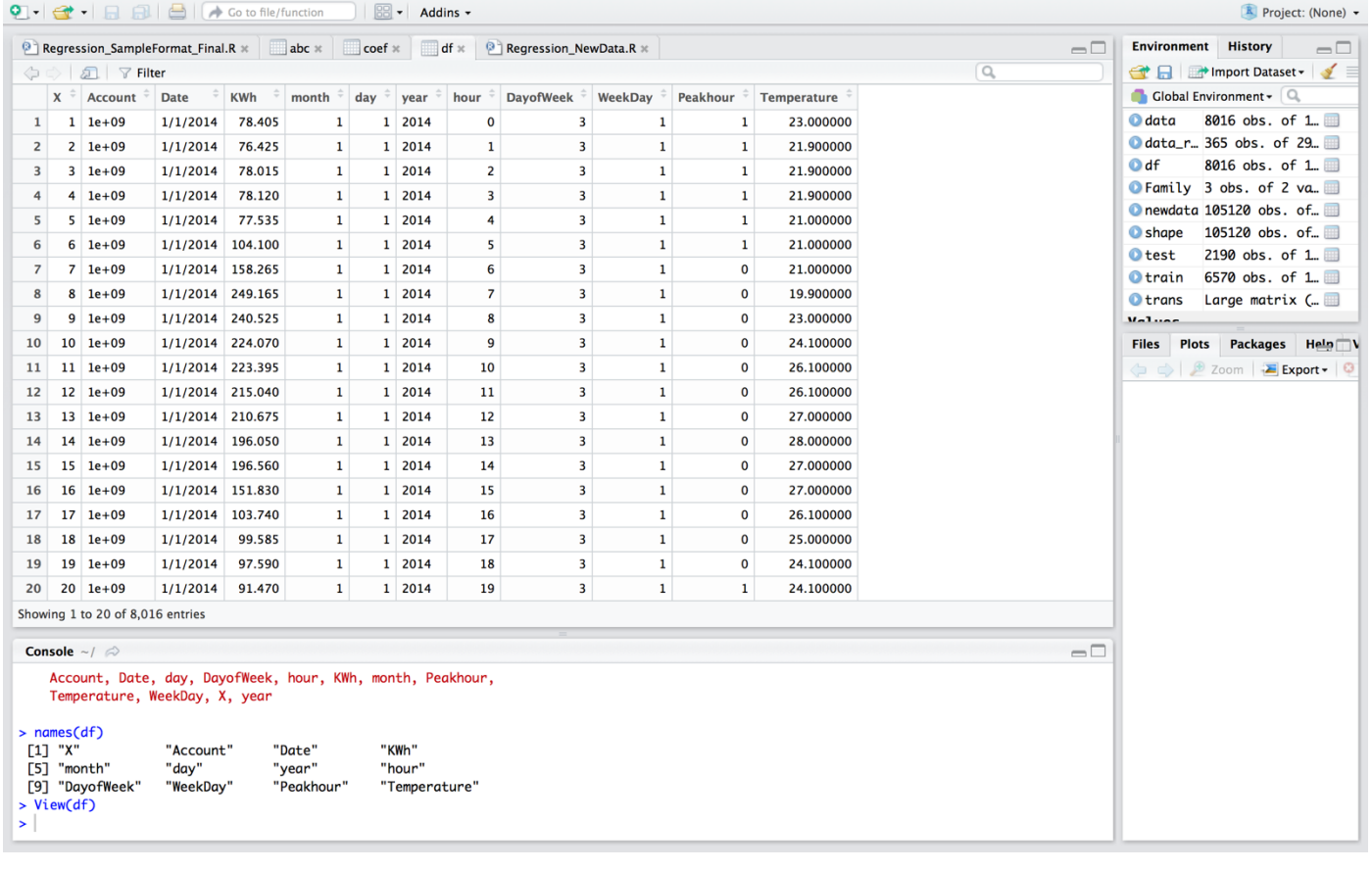
Data Wrangling and cleansing for newdata(given on 15th june)

To perform this follow all the steps of Part 1 -> Q1

Multiple Linear Regression for newdata(given on 15th june)

1) We include the 2 libraries namely MASS and ISLR for building the regression model and then we read the cleansed csv file named **sampleformat_NewData.csv.** We view the file using the View() function and then we also display the names of the variables in the data file.

```
library(MASS)

install.packages("ISLR")

library(ISLR)

data<- read.csv(file.choose(),header = T)

df <- data.frame(data)

View(df)
```



	X	Account	Date	KWh	month	day	year	hour	DayofWeek	WeekDay	Peakhour	Temperature
1	1	1e+09	1/1/2014	78.405	1	1	2014	0	3	1	1	23.000000
2	2	1e+09	1/1/2014	76.425	1	1	2014	1	3	1	1	21.900000
3	3	1e+09	1/1/2014	78.015	1	1	2014	2	3	1	1	21.900000
4	4	1e+09	1/1/2014	78.120	1	1	2014	3	3	1	1	21.900000
5	5	1e+09	1/1/2014	77.535	1	1	2014	4	3	1	1	21.000000
6	6	1e+09	1/1/2014	104.100	1	1	2014	5	3	1	1	21.000000
7	7	1e+09	1/1/2014	158.265	1	1	2014	6	3	1	0	21.000000
8	8	1e+09	1/1/2014	249.165	1	1	2014	7	3	1	0	19.900000
9	9	1e+09	1/1/2014	240.525	1	1	2014	8	3	1	0	23.000000
10	10	1e+09	1/1/2014	224.070	1	1	2014	9	3	1	0	24.100000
11	11	1e+09	1/1/2014	223.395	1	1	2014	10	3	1	0	26.100000
12	12	1e+09	1/1/2014	215.040	1	1	2014	11	3	1	0	26.100000
13	13	1e+09	1/1/2014	210.675	1	1	2014	12	3	1	0	27.000000
14	14	1e+09	1/1/2014	196.050	1	1	2014	13	3	1	0	28.000000
15	15	1e+09	1/1/2014	196.560	1	1	2014	14	3	1	0	27.000000
16	16	1e+09	1/1/2014	151.830	1	1	2014	15	3	1	0	27.000000
17	17	1e+09	1/1/2014	103.740	1	1	2014	16	3	1	0	26.100000
18	18	1e+09	1/1/2014	99.585	1	1	2014	17	3	1	0	25.000000
19	19	1e+09	1/1/2014	97.590	1	1	2014	18	3	1	0	24.100000
20	20	1e+09	1/1/2014	91.470	1	1	2014	19	3	1	1	24.100000

2) Next, we take out the 75% data to divide it into training and test data named as train and test.

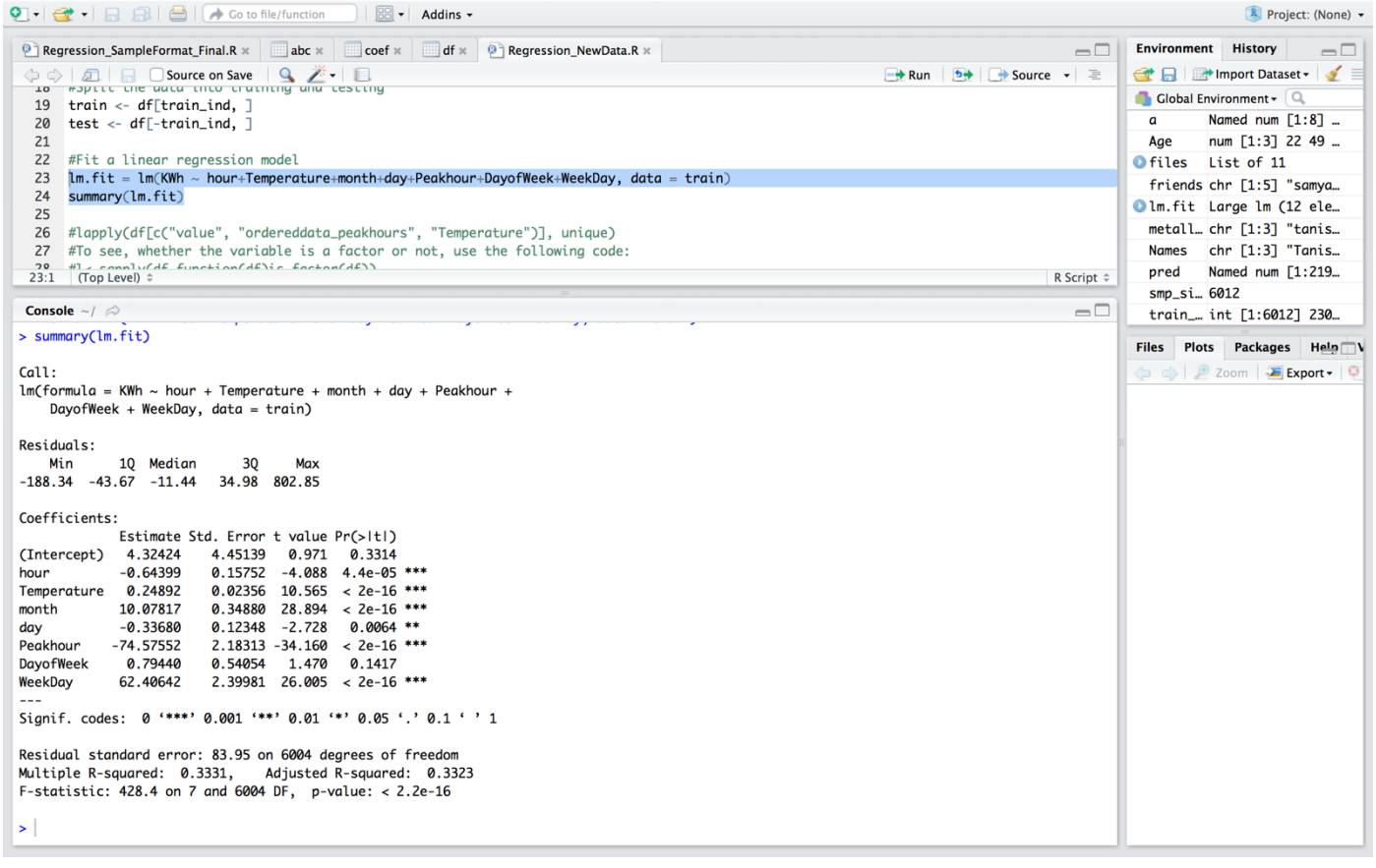
```
#75% of the sample size

smp_size <- floor(0.75 * nrow(df))
```

```
#Set the seed to make your partition reproductible

set.seed(123)

train_ind <- sample(seq_len(nrow(df)), size = smp_size)


#Split the data into training and testing

train <- df[train_ind, ]

test <- df[-train_ind, ]
```

3) Here, we write the function for finding the fit for the building of linear regression model. We use lm.fit() function for finding the best fit for our data. The column KWh is dependent on Temperature, hour, month,day,Peakhour,DayofWeek and weekday. We use train data obtained in step 2.

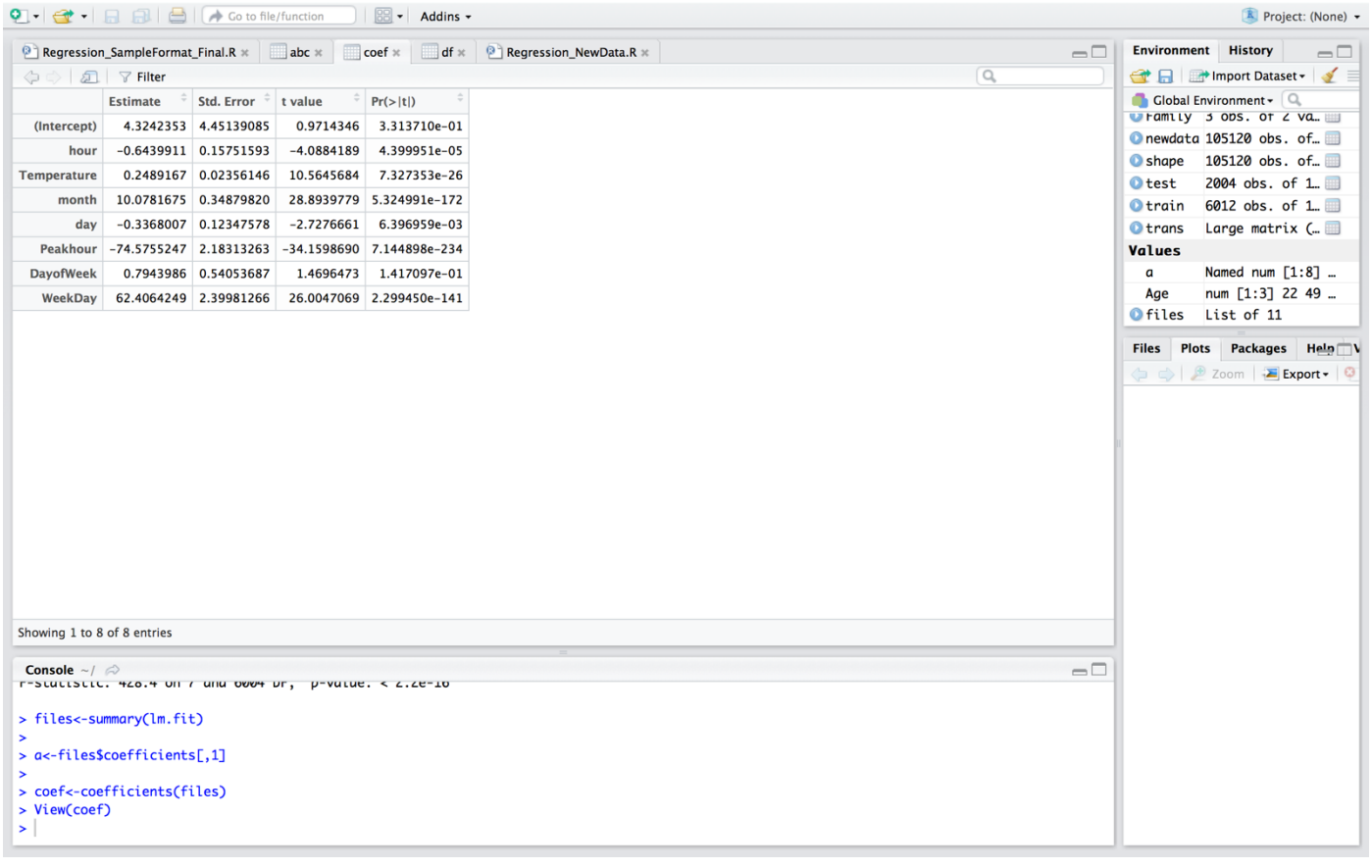
```
#Fit a linear regression model

lm.fit = lm(KWh ~ hour+Temperature+month+day+Peakhour+DayofWeek+WeekDay, data = train)


#Summary of the fit

files<-summary(lm.fit)
```

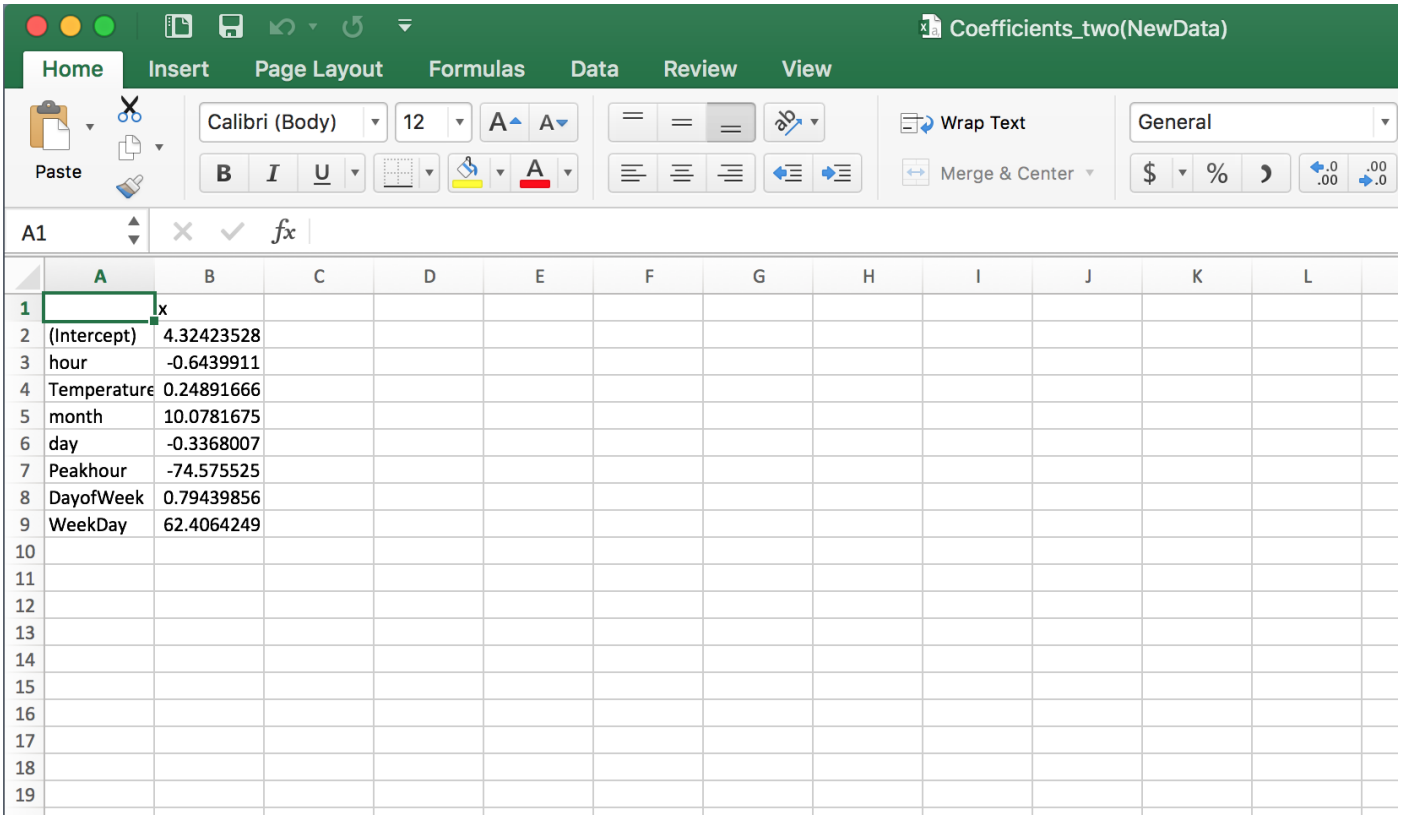
The Multiple R-squared value is 0.3331



4) The summary is inserted into the files variables and then we format it to get the first reading. Finally, we viewed the coefficients. This is then written in a csv file named **Coefficients_two(NewData).csv**

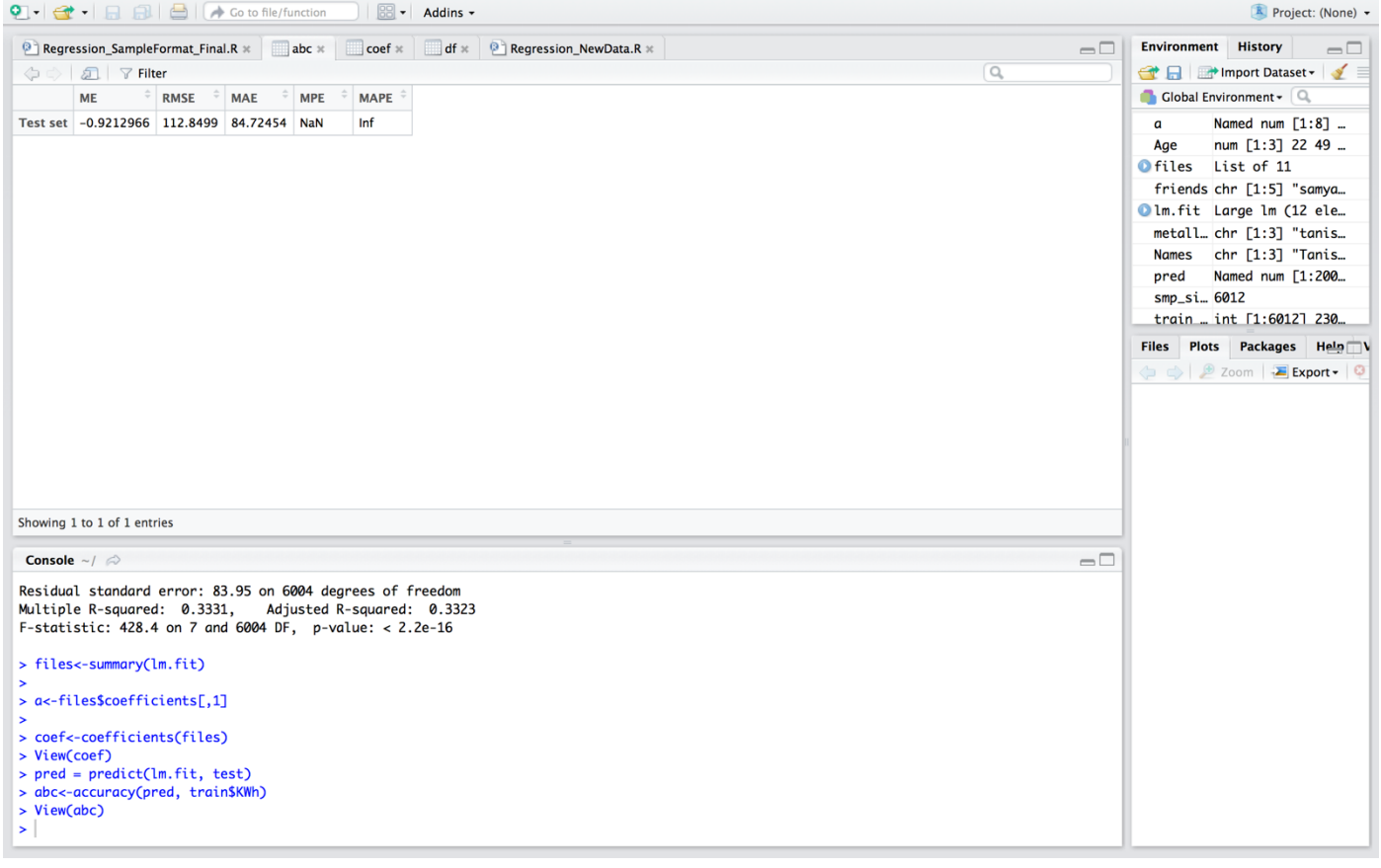
```
a<-files$coefficients[,1]

coef<-coefficients(files)

View(coef)

write.csv(a, file="Coefficients_two(NewData).csv")
```


Regression_SampleFormat_Final.R
abc
coef
df
Regression_NewData.R
Filter
Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.3242353 4.45139085 0.9714346 3.313710e-01
hour -0.6439911 0.15751593 -4.0884189 4.399951e-05
Temperature 0.2489167 0.02356146 10.5645684 7.327353e-26
month 10.0781675 0.34879820 28.8939779 5.324991e-172
day -0.3368007 0.12347578 -2.7276661 6.396959e-03
Peakhour -74.5755247 2.18313263 -34.1598690 7.144898e-234
DayofWeek 0.7943986 0.54053687 1.4696473 1.417097e-01
WeekDay 62.4064249 2.39981266 26.0047069 2.299450e-141
Showing 1 to 8 of 8 entries
Environment
History
Import Dataset
Global Environment
newdata 105120 obs. of...
shape 105120 obs. of...
test 2004 obs. of 1...
train 6012 obs. of 1...
trans Large matrix (...
Values
a Named num [1:8] ...
Age num [1:3] 22 49 ...
files List of 11
Files
Plots
Packages
Zoom
Export
Console
> files<-summary(lm.fit)
> a<-files$coefficients[,1]
> coef<-coefficients(files)
> View(coef)


Coefficients_two(NewData)
Home
Insert
Page Layout
Formulas
Data
Review
View
Paste
Calibri (Body)
12
Wrap Text
General
Merge & Center
A1
x
(Intercept) 4.32423528
hour -0.6439911
Temperature 0.24891666
month 10.0781675
day -0.3368007
Peakhour -74.575525
DayofWeek 0.79439856
WeekDay 62.4064249

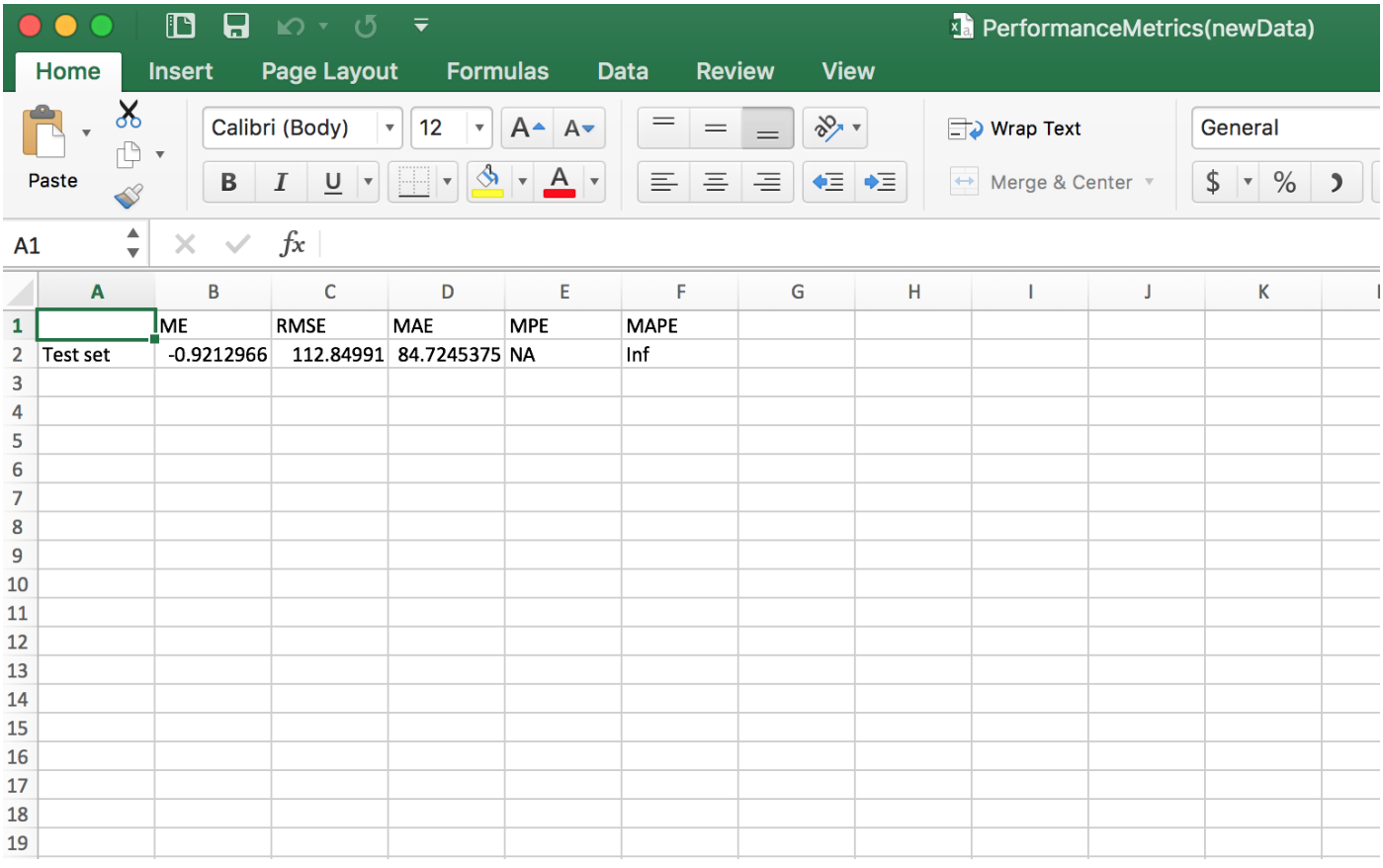
5) Next, we predict the accuracy and measure the Performance metrics by using the forecast library and the predict() function. Later, we view the performance metrics using View() function.

```
#Measures of predictive accuracy

install.packages("forecast")

library(forecast)

pred = predict(lm.fit, test)

abc<-accuracy(pred, train$KWh)

View(abc)
```



6) Finally, we write the R-script content in csv file using write.csv() function. The file name is PerformanceMetrics(SampleFormat).csv

```
write.csv(abc, file="PerformanceMetrics(SampleFormat).csv")
```



	A	B	C	D	E	F
1		ME	RMSE	MAE	MPE	MAPE
2	Test set	-0.9212966	112.84991	84.7245375	NA	Inf

PART 2 -> Q2
Data Wrangling and cleansing of forecast file(data given on 15th june)
2. (i) -> ForecastNewData file cleansed to Forecastnewdatainput file.

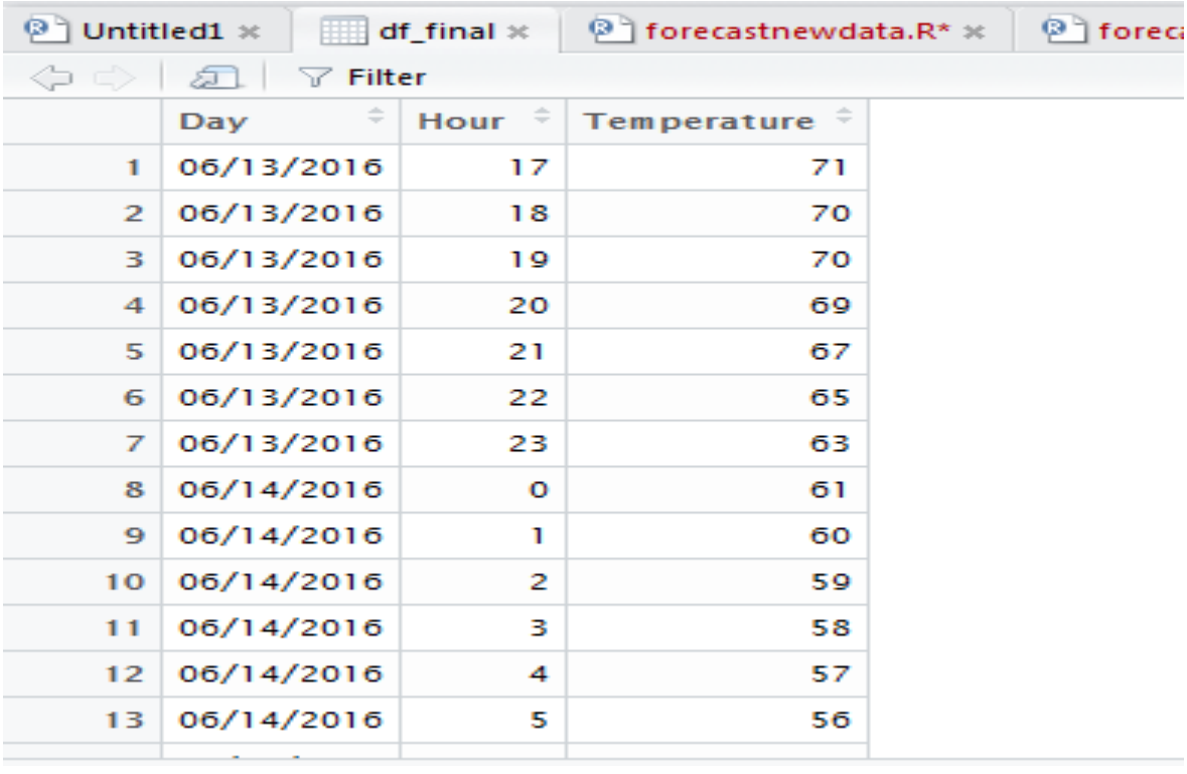
1. Lubridicate package and file is used to split date into month year and date format.
   Dplyr package identifies the most important data manipulation verbs and make them easy to use from R.

```
install.packages("lubridate")
library(lubridate)
install.packages("dplyr")
library(dplyr)
```

2. This code is used to parse the existing date from YYYYMMDD format to MM/DD/YYYY

```
date <-parse_date_time(as.character(df$Date),"%y/%m/%d")
Day<- format(as.Date(date),"%m/%d/%Y")
df_final <- NULL
```

```
df_final<-cbind(Day,df[2:3])
View(df_final)
```



Untitled1 | df_final | forecastnewdata.R* | foreca

Filter

	Day	Hour	Temperature
1	06/13/2016	17	71
2	06/13/2016	18	70
3	06/13/2016	19	70
4	06/13/2016	20	69
5	06/13/2016	21	67
6	06/13/2016	22	65
7	06/13/2016	23	63
8	06/14/2016	0	61
9	06/14/2016	1	60
10	06/14/2016	2	59
11	06/14/2016	3	58
12	06/14/2016	4	57
13	06/14/2016	5	56

3. This code is used to parse the date into month, year, date and weekday as per date. It also depicts how to bind the columns using cbind() function.

```
forecast_Year<-year(as.Date(df_final$Day, format="%m/%d/%Y"))
forecast_Month<-month(as.Date(df_final$Day, format="%m/%d/%Y"))
forecast_Day<-day(as.Date(df_final$Day, format="%m/%d/%Y"))
forecastdata<-cbind(Day,forecast_Month,forecast_Day,forecast_Year,df[2])
View(forecastdata)

ordereddata_weekday <-(wday(as.Date(forecastdata$Day, format="%m/%d/%Y")))-1
ordereddata1<-cbind(forecastdata,ordereddata_weekday)
View(ordereddata1)
```

titled1 | df_final | forecastnewdata.R* | forecastdata | ordereddata1 | forecast

Filter

	Day	forecast_Month	forecast_Day	forecast_Year	Hour	ordereddata_weekday
1	06/13/2016	6	13	2016	17	1
2	06/13/2016	6	13	2016	18	1
3	06/13/2016	6	13	2016	19	1
4	06/13/2016	6	13	2016	20	1
5	06/13/2016	6	13	2016	21	1
6	06/13/2016	6	13	2016	22	1
7	06/13/2016	6	13	2016	23	1
8	06/14/2016	6	14	2016	0	2
9	06/14/2016	6	14	2016	1	2
10	06/14/2016	6	14	2016	2	2
11	06/14/2016	6	14	2016	3	2
12	06/14/2016	6	14	2016	4	2
13	06/14/2016	6	14	2016	5	2

4. This code is used to provide information whether the specific day of the week is weekend or not. If weekday is 0 or 6, then it's a weekend (0) else it is a weekday.

```
ordereddata_weekend<-NA
ordereddata2<-cbind(ordereddata1,ordereddata_weekend)
ordereddata_weekend<- ifelse( ordereddata1$ordereddata_weekday == 0 |
ordereddata1$ordereddata_weekday == 6 , 0 , 1)
##warnings()
ordereddata2<-cbind(ordereddata1,ordereddata_weekend)
View(ordereddata2)
```

titled1 × | df_final × | forecastnewdata.R* × | forecastdata × | ordereddata1 × | forecast

Filter

	Day	forecast_Month	forecast_Day	forecast_Year	Hour	ordereddata_weekday
1	06/13/2016	6	13	2016	17	1
2	06/13/2016	6	13	2016	18	1
3	06/13/2016	6	13	2016	19	1
4	06/13/2016	6	13	2016	20	1
5	06/13/2016	6	13	2016	21	1
6	06/13/2016	6	13	2016	22	1
7	06/13/2016	6	13	2016	23	1
8	06/14/2016	6	14	2016	0	2
9	06/14/2016	6	14	2016	1	2
10	06/14/2016	6	14	2016	2	2
11	06/14/2016	6	14	2016	3	2
12	06/14/2016	6	14	2016	4	2
13	06/14/2016	6	14	2016	5	2

5. This code is used to order to provide output whether the hour is peak hour or not.Condition used in this code is if 7 A.M- 7 P.M = 1 else 7 P.M – 7 A.M = 0

```
ordereddata_peakhours <-NA
finalforecastdata<-cbind(ordereddata1,ordereddata_weekend,ordereddata_peakhours)
ordereddata_peakhours <- ifelse( as.numeric(ordereddata1$Hour) > 6 &  as.numeric(ordereddata1$Hour)
< 20 , 1 , 0)
finalforecastdata<-cbind(ordereddata2,ordereddata_peakhours,df[3])
```

_final × | forecastnewdata.R* × | forecast.R* × | forecastnewdata.R* × | finalforecastdata ×

Filter

	Date	month	Day	Year	Hour	Day of Week	WeekDay	Peakhour	Temperature
1	06/13/2016	6	13	2016	17	1	1	1	71
2	06/13/2016	6	13	2016	18	1	1	1	70
3	06/13/2016	6	13	2016	19	1	1	1	70
4	06/13/2016	6	13	2016	20	1	1	0	69
5	06/13/2016	6	13	2016	21	1	1	0	67
6	06/13/2016	6	13	2016	22	1	1	0	65
7	06/13/2016	6	13	2016	23	1	1	0	63
8	06/14/2016	6	14	2016	0	2	1	0	61
9	06/14/2016	6	14	2016	1	2	1	0	60
10	06/14/2016	6	14	2016	2	2	1	0	59
11	06/14/2016	6	14	2016	3	2	1	0	58
12	06/14/2016	6	14	2016	4	2	1	0	57

6. This function i.e colnames() is used to provide column names of the dataframe.

   colnames(finalforecastdata)<-c("Date","month","Day","Year", "Hour","Day of Week","WeekDay","Peakhour","Temperature")
   View(finalforecastdata)

7. This function is used to write the data to csv file.
   write.csv(finalforecastdata,file="forecastdatanewinput.csv")

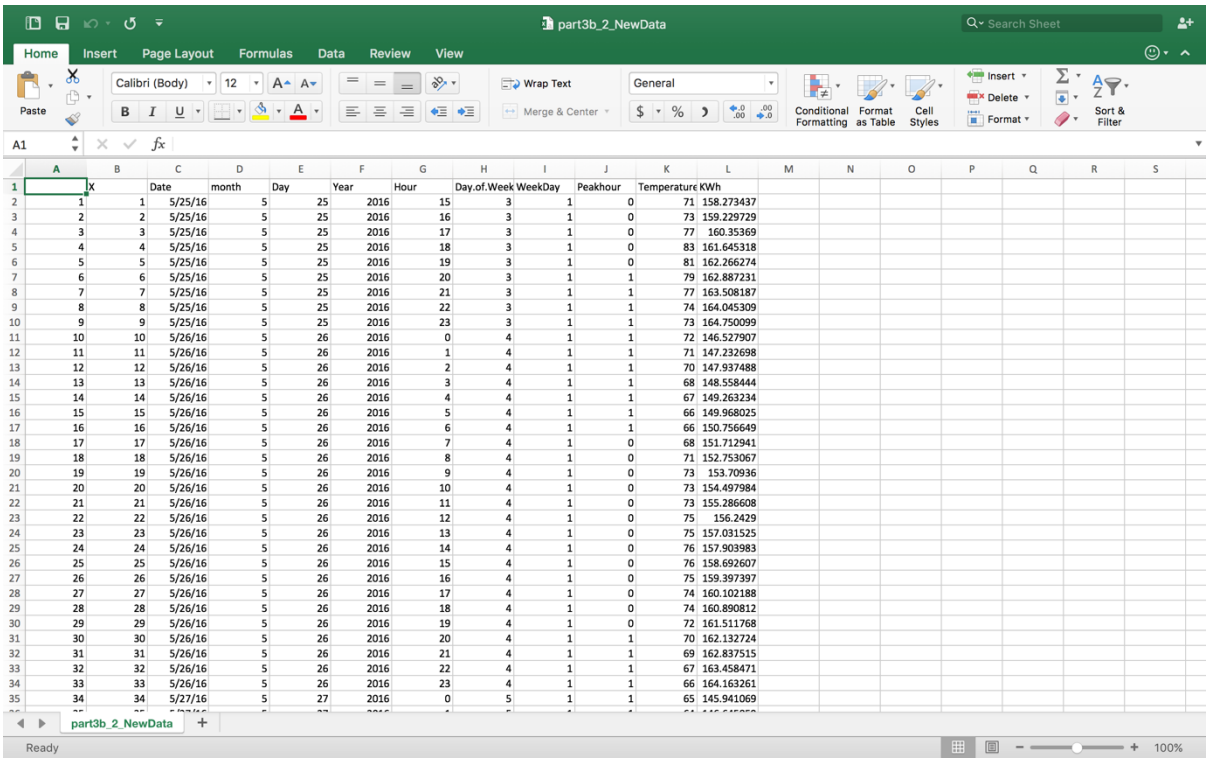
PART 2

2. (ii) -> ForecastNewData2 file cleansed to Forecastnewdata2input file.

Repeat the same steps as Part 2 -> 2(i) to cleanse the data.
Note : No need to change the date format as it is already in MM/DD/YYYY format.

PART2 -> Q2

MULTIPLE LINEAR REGRESSION(forecast data given on 15th june)



part3b_2_NewData

Search Sheet

Home Insert Page Layout Formulas Data Review View

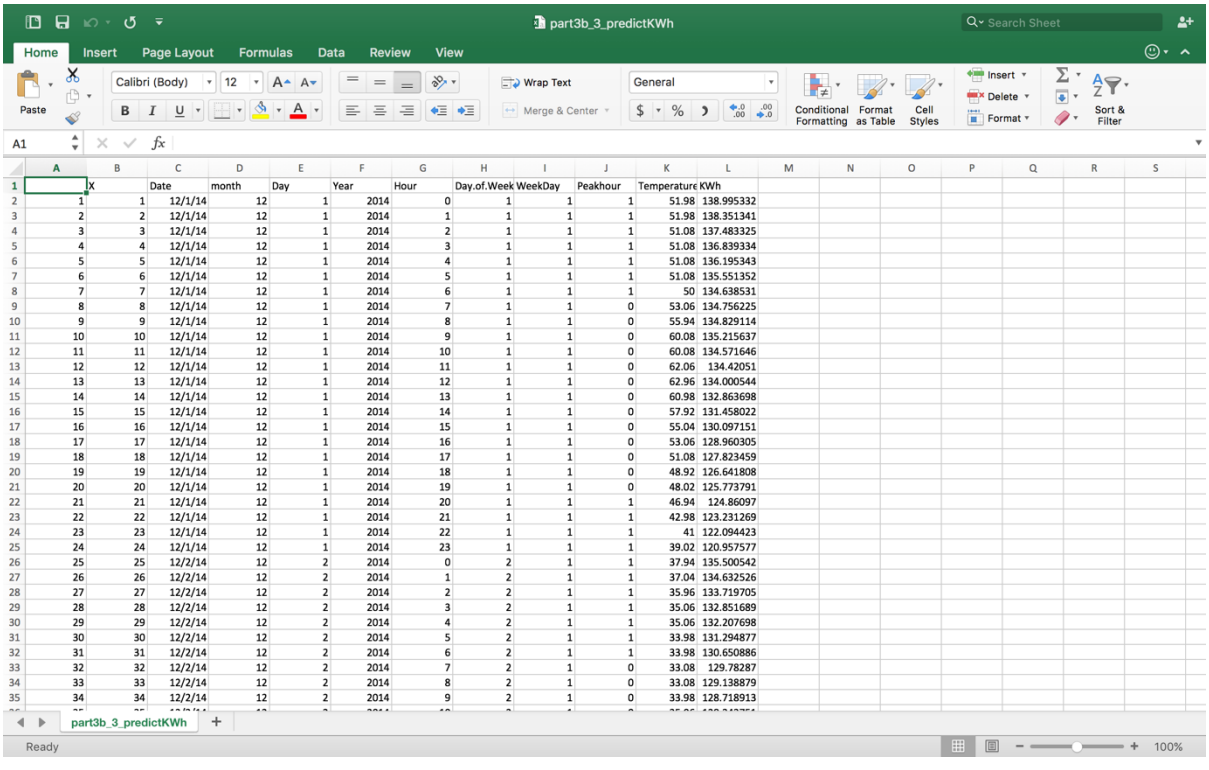
Paste Calibri (Body) 12 Wrap Text General Conditional Formatting Format as Table Cell Styles Insert Delete Format Sort & Filter

A1

	A	B	C	D	E	F	G	H	I	J	K	L
1		X	Date	month	Day	Year	Hour	Day.of.Week	WeekDay	Peakhour	Temperature	KWh
2	1	1	5/25/16	5	25	2016	15	3	1	0	71	158.273437
3	2	2	5/25/16	5	25	2016	16	3	1	0	73	159.229729
4	3	3	5/25/16	5	25	2016	17	3	1	0	77	160.35369
5	4	4	5/25/16	5	25	2016	18	3	1	0	83	161.645318
6	5	5	5/25/16	5	25	2016	19	3	1	0	81	162.266274
7	6	6	5/25/16	5	25	2016	20	3	1	1	79	162.887231
8	7	7	5/25/16	5	25	2016	21	3	1	1	77	163.508187
9	8	8	5/25/16	5	25	2016	22	3	1	1	74	164.045309
10	9	9	5/25/16	5	25	2016	23	3	1	1	73	164.750099
11	10	10	5/26/16	5	26	2016	0	4	1	1	72	146.527907
12	11	11	5/26/16	5	26	2016	1	4	1	1	71	147.232698
13	12	12	5/26/16	5	26	2016	2	4	1	1	70	147.937488
14	13	13	5/26/16	5	26	2016	3	4	1	1	68	148.558444
15	14	14	5/26/16	5	26	2016	4	4	1	1	67	149.263234
16	15	15	5/26/16	5	26	2016	5	4	1	1	66	149.968025
17	16	16	5/26/16	5	26	2016	6	4	1	1	66	150.756649
18	17	17	5/26/16	5	26	2016	7	4	1	0	68	151.712941
19	18	18	5/26/16	5	26	2016	8	4	1	0	71	152.753067
20	19	19	5/26/16	5	26	2016	9	4	1	0	73	153.70936
21	20	20	5/26/16	5	26	2016	10	4	1	0	73	154.497984
22	21	21	5/26/16	5	26	2016	11	4	1	0	73	155.286608
23	22	22	5/26/16	5	26	2016	12	4	1	0	75	156.2429
24	23	23	5/26/16	5	26	2016	13	4	1	0	75	157.031525
25	24	24	5/26/16	5	26	2016	14	4	1	0	76	157.903983
26	25	25	5/26/16	5	26	2016	15	4	1	0	76	158.692607
27	26	26	5/26/16	5	26	2016	16	4	1	0	75	159.397397
28	27	27	5/26/16	5	26	2016	17	4	1	0	74	160.102188
29	28	28	5/26/16	5	26	2016	18	4	1	0	74	160.890812
30	29	29	5/26/16	5	26	2016	19	4	1	0	72	161.511768
31	30	30	5/26/16	5	26	2016	20	4	1	1	70	162.132724
32	31	31	5/26/16	5	26	2016	21	4	1	1	69	162.837515
33	32	32	5/26/16	5	26	2016	22	4	1	1	67	163.458471
34	33	33	5/26/16	5	26	2016	23	4	1	1	66	164.163261
35	34	34	5/27/16	5	27	2016	0	5	1	1	65	145.941069

part3b_2_NewData

Ready 100%



part3b_3_predictKWh

Search Sheet

Home Insert Page Layout Formulas Data Review View

Paste Calibri (Body) 12 Wrap Text General Conditional Formatting Format as Table Cell Styles Insert Delete Format Sort & Filter

A1

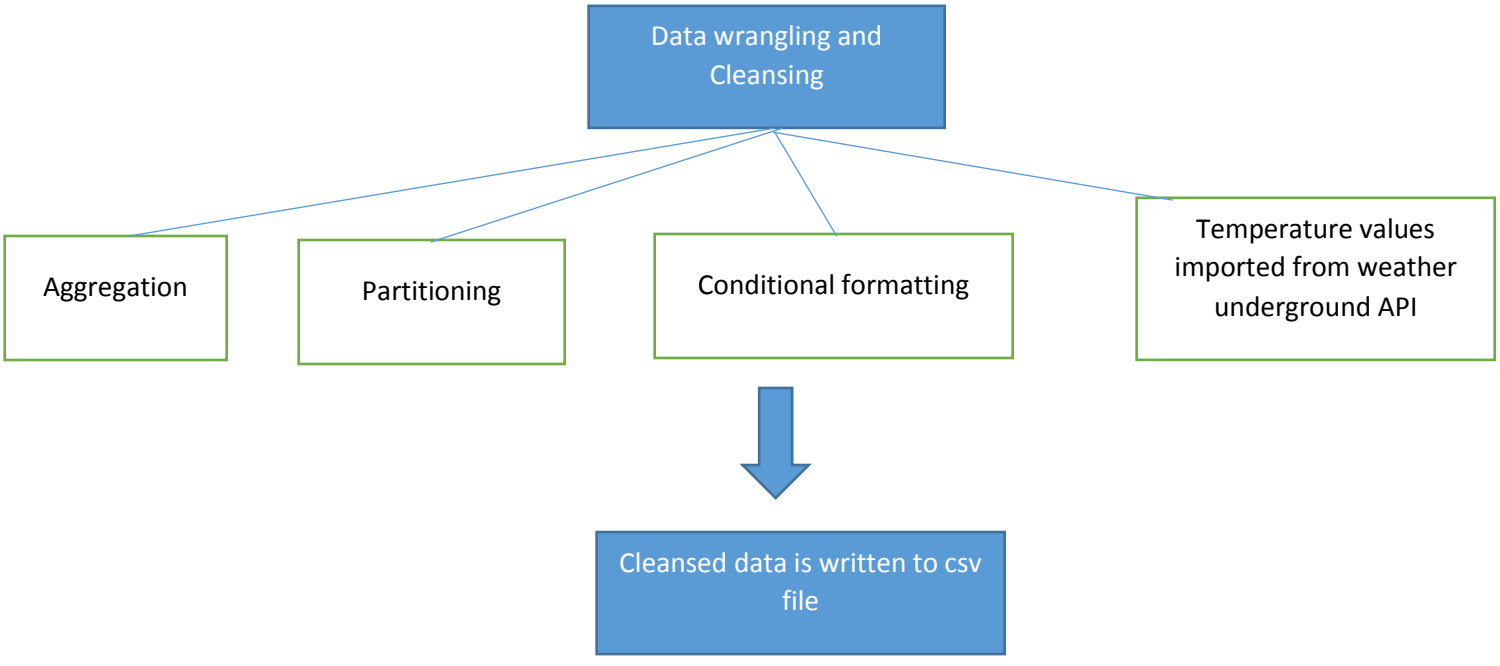
	A	B	C	D	E	F	G	H	I	J	K	L
1		X	Date	month	Day	Year	Hour	Day.of.Week	WeekDay	Peakhour	Temperature	KWh
2	1	1	12/1/14	12	1	2014	0	1	1	1	51.98	138.995332
3	2	2	12/1/14	12	1	2014	1	1	1	1	51.98	138.351341
4	3	3	12/1/14	12	1	2014	2	1	1	1	51.08	137.483325
5	4	4	12/1/14	12	1	2014	3	1	1	1	51.08	136.839334
6	5	5	12/1/14	12	1	2014	4	1	1	1	51.08	136.195343
7	6	6	12/1/14	12	1	2014	5	1	1	1	51.08	135.551352
8	7	7	12/1/14	12	1	2014	6	1	1	1	50	134.638531
9	8	8	12/1/14	12	1	2014	7	1	1	0	53.06	134.756225
10	9	9	12/1/14	12	1	2014	8	1	1	0	55.94	134.829114
11	10	10	12/1/14	12	1	2014	9	1	1	0	60.08	135.215637
12	11	11	12/1/14	12	1	2014	10	1	1	0	60.08	134.571646
13	12	12	12/1/14	12	1	2014	11	1	1	0	62.06	134.42051
14	13	13	12/1/14	12	1	2014	12	1	1	0	62.96	134.000544
15	14	14	12/1/14	12	1	2014	13	1	1	0	60.98	132.863698
16	15	15	12/1/14	12	1	2014	14	1	1	0	57.92	131.458022
17	16	16	12/1/14	12	1	2014	15	1	1	0	55.04	130.097151
18	17	17	12/1/14	12	1	2014	16	1	1	0	53.06	128.960305
19	18	18	12/1/14	12	1	2014	17	1	1	0	51.08	127.823459
20	19	19	12/1/14	12	1	2014	18	1	1	0	48.92	126.641808
21	20	20	12/1/14	12	1	2014	19	1	1	0	48.02	125.773791
22	21	21	12/1/14	12	1	2014	20	1	1	1	46.94	124.86097
23	22	22	12/1/14	12	1	2014	21	1	1	1	42.98	123.231269
24	23	23	12/1/14	12	1	2014	22	1	1	1	41	122.094423
25	24	24	12/1/14	12	1	2014	23	1	1	1	39.02	120.957577
26	25	25	12/2/14	12	2	2014	0	2	1	1	37.94	135.500542
27	26	26	12/2/14	12	2	2014	1	2	1	1	37.04	134.632526
28	27	27	12/2/14	12	2	2014	2	2	1	1	35.96	133.719705
29	28	28	12/2/14	12	2	2014	3	2	1	1	35.06	132.851689
30	29	29	12/2/14	12	2	2014	4	2	1	1	35.06	132.207698
31	30	30	12/2/14	12	2	2014	5	2	1	1	33.98	131.294877
32	31	31	12/2/14	12	2	2014	6	2	1	1	33.98	130.650886
33	32	32	12/2/14	12	2	2014	7	2	1	0	33.08	129.78287
34	33	33	12/2/14	12	2	2014	8	2	1	0	33.08	129.138879
35	34	34	12/2/14	12	2	2014	9	2	1	0	33.98	128.718913

part3b_3_predictKWh

Ready 100%

DATA FLOW

DATA WRANGLING AND CLEANSING



MULTIPLE LINEAR REGRESSION