Tutorial #5

The data-driven approach that we are studying can be divided into three sections: Data Acquisition, Analysis, and Modelling. Specialization in each of these sections is trending, as each of these sections is developing at a rapid rate. A Data Engineer must acquire data required to solve the problem. The data engineer must collect and manage locally stored data or create data pipelines to retrieve data from the servers. In any case, data engineers should provide easy access to relevant data. Good knowledge of various database management systems and system automation is essential for a data engineer to fulfill his/her duties.

A Data Analyst is responsible for extracting useful information from the data. By using statistical tools and the use of visualization techniques, a data analyst reveals patterns and relationships between various attributes in a dataset. The plots and graphs that are created should be appropriate and helpful to show the most relevant information and patterns. A data analyst should also have a good understanding of statistical methods to test various hypotheses.

Data modeling is developing mathematical equations to explain the relationship between two or more attributes of a dataset. The latest techniques used for this are called machine learning, which has been the primary subject of the previous tutorials. A data scientist is responsible for creating these data models. A data engineer provides the relevant data which is used by a data analyst to find patterns and relationship between data attributes which is used by a data scientist to create data models to solve a problem.

To forecast the load of residential buildings, we will require electricity demand data of the households. The collection of this data is the job of a data engineer. This paper provides the details of the process followed by EIG to collect PRECON dataset.

List the steps followed to collect PRECON dataset

The clean data for 42 household's electricity consumption can be downloaded from the link above. The locally stored data can then be imported into the R environment using *read.csv()* function. Another way of accessing this data is by directly pasting the download link in the function, as shown below.

The summary of the data shows that the data timeline is from 2018-07-15 till 2019-05-31. This is not a full year. We will skip **House_1** from our analysis, for now, the data is incomplete for this household.

```
## Read data House_1 =
read.csv("http://web.lums.edu.pk/~eig/precon_files/House1.csv")
## Covert Date_Time column to posixct House_1$Date_Time =
as.POSIXct(House_1$Date_Time)
## Summary of the data summary(House_1)
```

```
##        Date_Time                       Usage_kW            AC_DR_kW
## Min.  :2018-07-15 00:00:00       Min.      :0.0000  Min.      :0.000000 ## 1st Qu.:2018-
10-03 05:59:45   1st Qu.:0.2980    1st Qu.:0.000100 ## Median :2018-12-22 11:59:30
          Median :0.6417   Median :0.000300 ## Mean          :2018-12-22 11:59:30
          Mean     :0.7752  Mean     :0.004244 ## 3rd Qu.:2019-03-12 17:59:15   3rd
Qu.:1.0555         3rd Qu.:0.000600 ## Max. :2019-05-31 23:59:00       Max.      :5.2049
          Max.      :2.157400
##          UPS_kW                LR_kW               Kitchen_kW            AC_Dr_kW
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.00000 ## 1st Qu.:0.0570 1st Qu.:0.0114
1st Qu.:0.0657 1st Qu.:0.00130 ## Median :0.1130 Median :0.0413 Median :0.1140 Median
:0.00270
## Mean        :0.1252       Mean      :0.1147       Mean      :0.1310       Mean      :0.01749
## 3rd Qu.:0.1758           3rd Qu.:0.2096         3rd Qu.:0.1786         3rd Qu.:0.00520
```

```
## Max.   :0.6742   Max.   :1.4398   Max.   :2.1057   Max.   :1.71760
##      AC_BR_kW
## Min.  :0.00000 ## 1st Qu.:0.00010 ## Median :0.00020 ## Mean :0.03934 ## 3rd Qu.:0.00050 ## Max. :1.93430
```

Let's move on to the next house in our metadata list i.e., **House 2**. The data for this house is complete, as we can see in the summary below. There are a total of 7 columns in the dataset. The first column is the *Date_Time*, which shows that the data is a time series. At EIG, most of the data that we deal with is time-series data, as variation in energy consumption over time is the most important thing that creates most of the problems in the field of energy. If the energy consumption and generation had been constant over time, there would have been no need for forecasting and no uncertainty in power system planning.

The second column is *Usage_kW*, which is typical for every house in the dataset. This column represents the total electricity consumption of the household. The values are in kW, which is a unit of power. On average, 411 W of electricity is consumed in the household.

How much watts are there in 1 horse-power?

What is the wattage of the refrigerator at your home? There should be a nameplate at the back. Send a picture of it

What is the relationship between Energy & Power?

The unit of Energy commonly used in the field of Energy is kWh. How is kWh equal to J (joules)?

How much Energy will your refrigerator consume in 2 hours? How much will it cost?

```
## Read data House_2 =
read.csv("http://web.lums.edu.pk/~eig/precon_files/House2.csv")
## Covert Date_Time column to posixct House_2$Date_Time =
as.POSIXct(House_2$Date_Time)
## Summary of the data summary(House_2)
```

```
##      Date_Time                    Usage_kW          AC_LR_kW
## Min.  :2018-06-01 00:00:00   Min.   :0.0000   Min.   :0.0000000 ## 1st Qu.:2018-08-31 05:59:45   1st Qu.:0.1420   1st Qu.:0.0000000 ## Median :2018-11-30 11:59:30   Median :0.3126   Median :0.0000000 ## Mean   :2018-11-30 11:59:30   Mean   :0.4111   Mean   :0.0002841 ## 3rd Qu.:2019-03-01 17:59:15 3rd Qu.:0.5650   3rd Qu.:0.0001000 ## Max.   :2019-05-31 23:59:00   Max.   :4.5157   Max.   :0.8022000
##      AC_DR_kW          AC_BR_kW          Kitchen_kW          UPS_kW
## Min.  :0.000000  Min.  :0.00000  Min.  :0.00000  Min.  :0.00000  ## 1st Qu.:0.000000  1st Qu.:0.00000  1st Qu.:0.00010  1st Qu.:0.02000 ## Median :0.000000  Median :0.00010  Median :0.00020  Median :0.04390 ## Mean   :0.001666  Mean   :0.07398  Mean   :0.01384  Mean   :0.07906 ## 3rd Qu.:0.000100  3rd Qu.:0.00020  3rd Qu.:0.00040  3rd Qu.:0.13970
## Max.   :1.289000  Max.   :1.85010  Max.   :2.06970  Max.   :0.53300
```

The rest of the columns shows the electricity cosnumption of air conditioners, kitchen and UPS of the household. We are not interested in those columns, so removing those columns from the dataset.

```
## Removing unwanted columns
```

```
House_2 = House_2[,-c(3:7)]

## structure of the dataset str(House_2)
```

```
## 'data.frame':                525600 obs. of 2 variables:
## $ Date_Time: POSIXct, format: "2018-06-01 00:00:00" "2018-06-01 00:01:00" ...
## $ Usage_kW : num 0.802 0.801 0.797 0.802 0.81 ...
```

The dataset contains 525600 rows. The data is at minute interval. As a data engineer, it is our duty to make sure that there is data for each Date and Time value in the time period defined. The year 2019 was not a leap year, so there should be 8760 hours from 2018-06-01 00:00:00 PKT till 2019-05-31 23:59:00.

```
## nrow() gets the number of rows in the dataframe
nrow(House_2)/60
```

```
## [1] 8760
```

However, there is still some doubt that there might be duplicate values for some time instant in the dataset and missing for others. We have to make sure that each time instant has a single row and no time instant is missing. To check this we will create a time series of our own for which we are sure that all rows are correct and then compare it to the dataset to check the integrity of the dataset.

```
## This dataframe contains all minutes between the specified time
All_minutes = data.frame(Date_Time = seq.POSIXt(from = as.POSIXct("2018-06-01 00:00:00 PKT"
                                                                  to = as.POSIXct("2019-05-31 23:59:00 PKT" by =
                                                                  "min"))

## Merge the two dataframes. Read its documentation ?merge.

House_2 = merge(House_2, All_minutes, by = "Date_Time", all.y = TRUE) summary(House_2)
```

```
##        Date_Time                       Usage_kW
## Min.          :2018-06-01 00:00:00   Min.        :0.0000
## 1st Qu.:2018-08-31 05:59:45          1st Qu.:0.1420
## Median :2018-11-30 11:59:30          Median :0.3126
## Mean          :2018-11-30 11:59:30   Mean        :0.4111
## 3rd Qu.:2019-03-01 17:59:15          3rd Qu.:0.5650
## Max.          :2019-05-31 23:59:00   Max.        :4.5157
```

The summary above shows that no rows have NA in **Usage_kW** column, which proves that the dataset has all Date and Time instances in the specified time.

Explain the merge() function used above

The summary above also shows that **Usage_kW** has no value less than 0, which means that there was no generation at this household. This makes sense as this household does not have any source of generation. A data engineer should make sure the data is clean and does not have any outliers. Fortunately this data is clean and does not have any outliers.

Another thing to consider is the data type of each column of the dataset. When reading the data, the *read.csv()* function tries to specify the most appropriate data type automatically. However, it can go worng sometimes.

The code below shows that both columns have the right data type. *POSIXCT* is the data/time object of R language and *Usage_kW* is *numeric* as expected.

Explain the command below

```
# Explain this sapply(House_2, class)
```

```
## $Date_Time
## [1] "POSIXct" "POSIXt"
##
## $Usage_kW
## [1] "numeric"
```

The next thing we need to do is convert the data from minute-interval to hour-interval data. The *tapply()* function below can be used to change the granularity of the data from a high interval to low interval.

Explain the code below

```
## Explain this
# tapply(House_2$Usage_kW,(row(as.matrix(House_2$Usage_kW))-1)%/%60, mean)

## Use the tapply() function to create hourly data and replace it in House_2 dataframe
House_2 = data.frame( Date_Time = seq.POSIXt(from = as.POSIXct("2018-06-01 PKT"),
                                             to = as.POSIXct("2019-05-31 23:00:00 PKT" by =
                      "hour"), Usage_kW = c(tapply(House_2$Usage_kW,
                                                   (row(as.matrix(House_2$Usage_kW))-1)%/%60
                      )

summary(House_2)
```

),

,

mean))

```
##       Date_Time                      Usage_kW
## Min.          :2018-06-01 00:00:00   Min.       :0.0000
## 1st Qu.:2018-08-31 05:45:00          1st Qu.:0.1657
## Median :2018-11-30 11:30:00          Median :0.3245
## Mean          :2018-11-30 11:30:00   Mean       :0.4111
## 3rd Qu.:2019-03-01 17:15:00          3rd Qu.:0.5568
## Max.          :2019-05-31 23:00:00   Max.       :3.4504
```

The summary of the hourly data shows that the average of *Usage_kW* is not changed. But the maximum value has decreased from 4.515 kW to 3.45 kW. *Why is it so?*

The job of a data engineer ends here, by collecting data from the households, cleaning the data and processing it to be useful for data analyst and scientist.

Create a script to import data of the 9 houses (Ignore House 1) specified in the metadata file. Process the data and make sure it is ready to be used for further processes. I would recommend that you download the data and import it from local directory as reading data from server as I did above might take a lot of time.
And reading from server also requires a stable internet connection.

Email me your detailed report.