

**PURPOSE OF PLOT**

* To explore the inter-relationships between sensor readings over time
* To detect abnormal inter-relationships between sensor readings

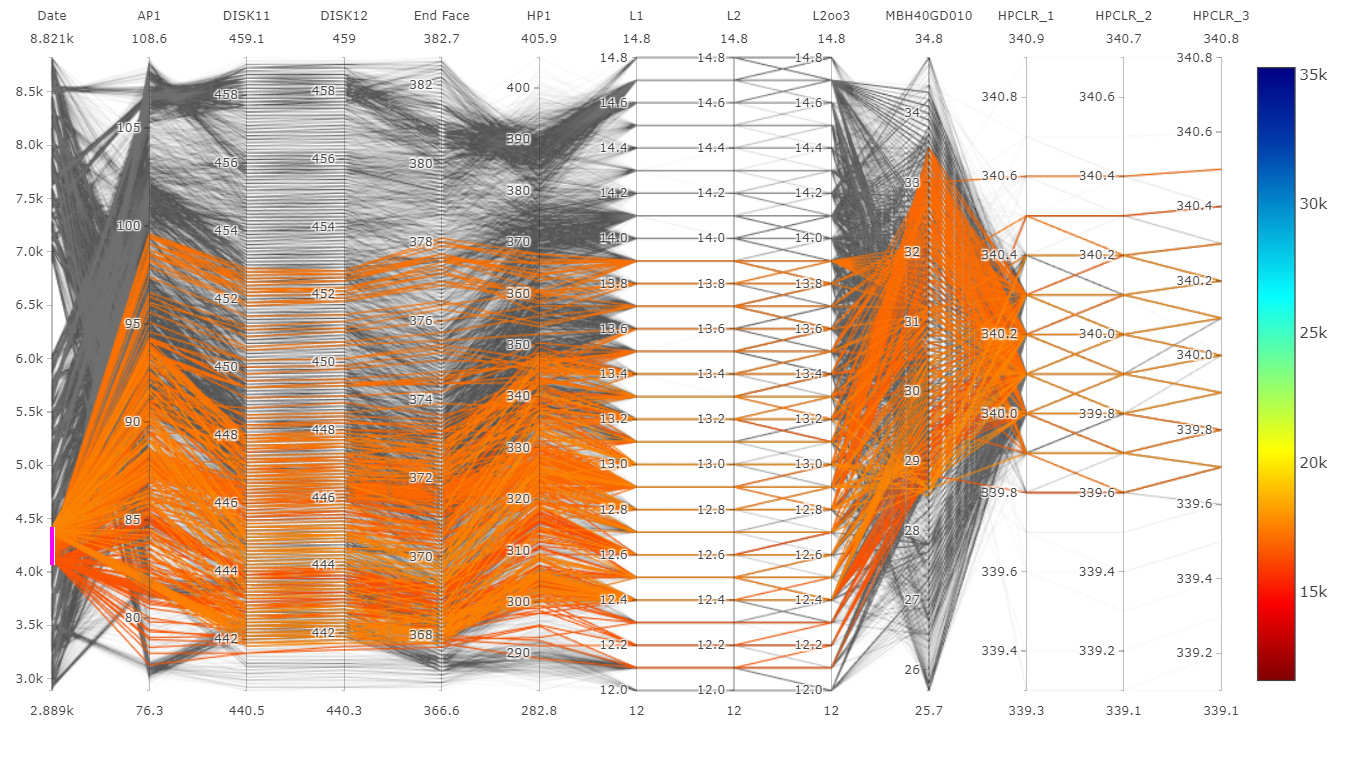
**HOW TO INTERPRET**

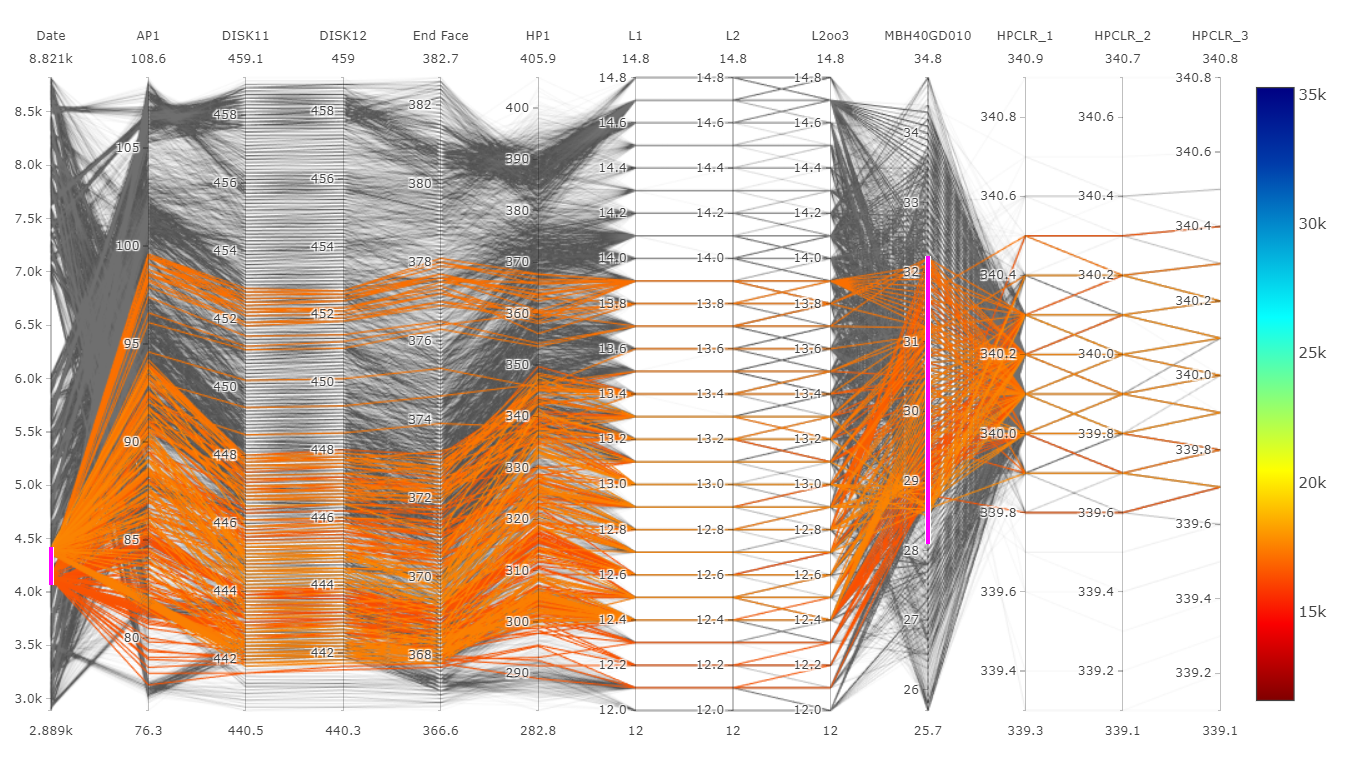
* Each vertical column represents sensor group readings belonging to the Functional System Item selected.
* The scale along each vertical column represents minimal to maximum values of each sensor group readings.
* Each line on the parallel plot represents a unit of the time interval selected. For example, if ‘Every hour’ is selected, each line represents an hour of reading for the entire range of dates selected.
* Key formulas/methods to transform the data for parallel plot:
  + Data points are first aggregated (originally 5 sec intervals) to 5min, hourly and daily intervals using openair package, timeAverage().
  + Sensor readings in the MsureGr\_lvl column are converted into columns using spread() in order to plot them as vertical columns on the parallel plot.
  + Lastly, the data points are grouped by date and FunSys\_item so that each row represents a time interval for a FunSys\_item.

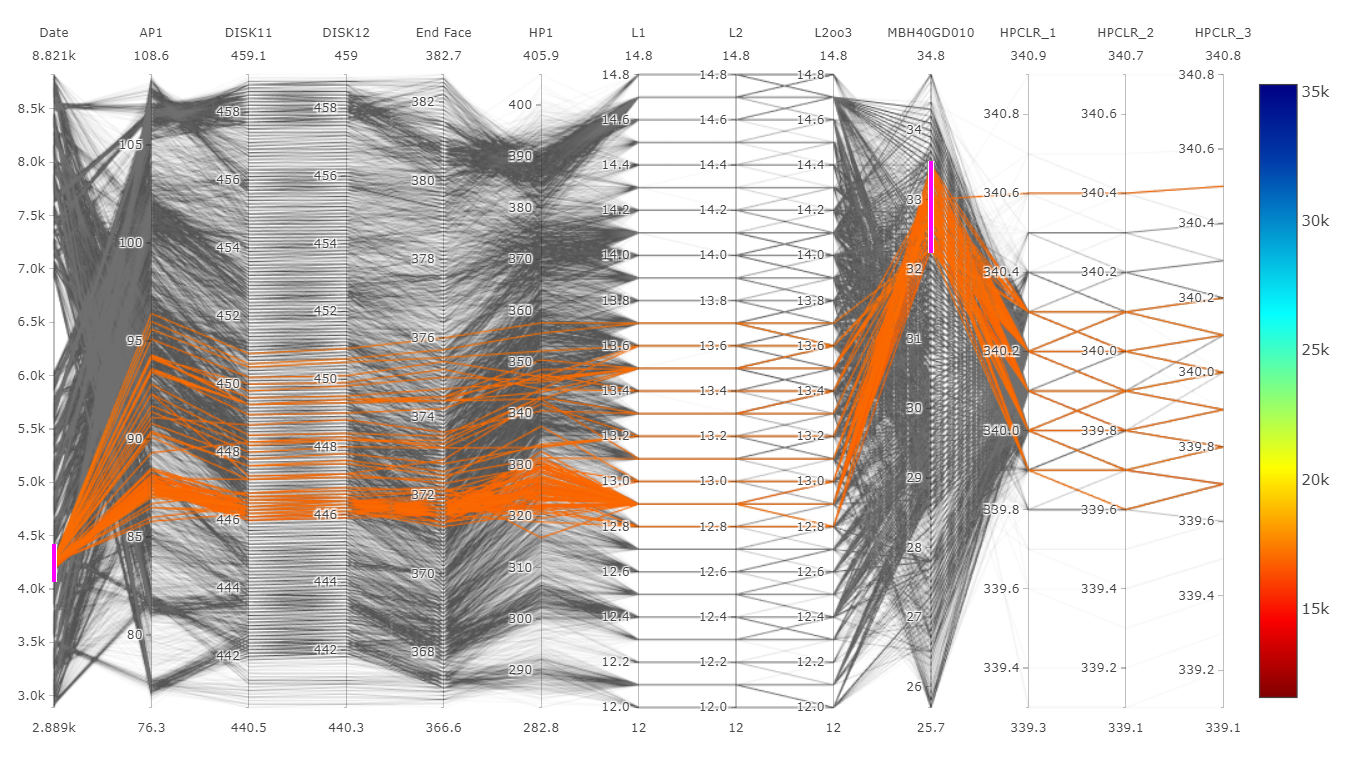
**HOW TO USE THE PLOT**

* Allowable selections:
  1. Function System
  2. Range of dates
  3. Date (reference – min. of Day Window)
  4. Exclude date range
  5. Day window (1 day, 3 days, 5 days, 7 days, 14 days, All days)
  6. Time interval (Every 5min, Every hour, Every day)
* Interactive elements:
  1. Drag along one or multiple axis to highlight data
  2. Double-click to dismiss selection
  3. Drag and re-arrange order of vertical axis
* EDA:
  1. Select the desired window of time you wish to examine from ‘Day Window’.
  2. Drag the slider from ‘Date (reference – min of Day Window) horizontally in order to examine the sensor readings with respect to time. The date on the slider denotes the earliest date in the time window selected.
  3. Alternatively, select and drag the range of data on the left-most axis upwards slowly.
  4. Lowest values on the date axis denotes the start date in the range of dates selected.
  5. Take note of lines that display inter-relationships that are abnormal with comparison to data set within the same period.
  6. These abnormalities suggest potential sensors and date period that are worthy of further investigation.

**EXAMPLES**

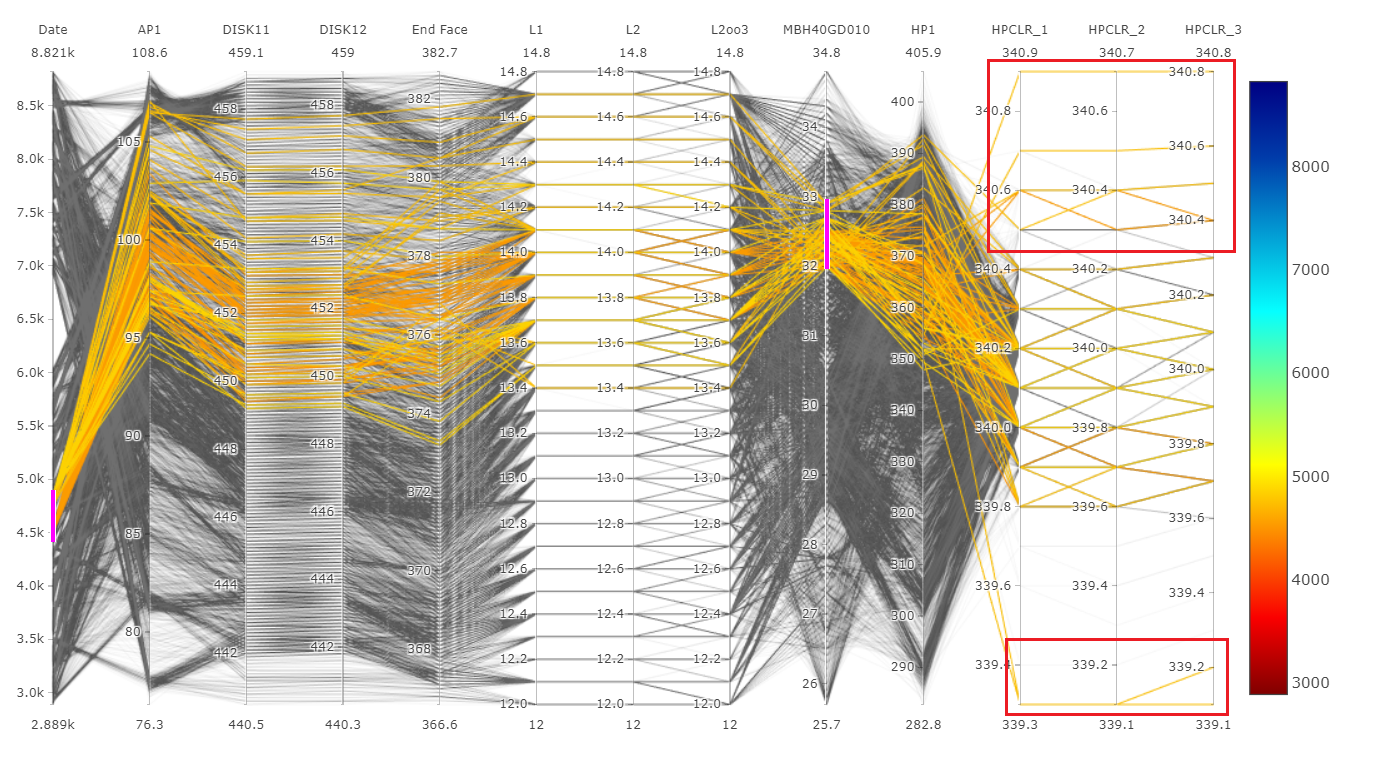






Around 2007-07-16 to 2007-07-17, noticed a wider variance of values for HPCLR\_1, HPCLR\_2,HPCLR\_3 that was not usual compared to the normal range deviation.

Sensors spotted with unusual readings deviation



**PACKAGES COMPARISON**

1. **GGally**

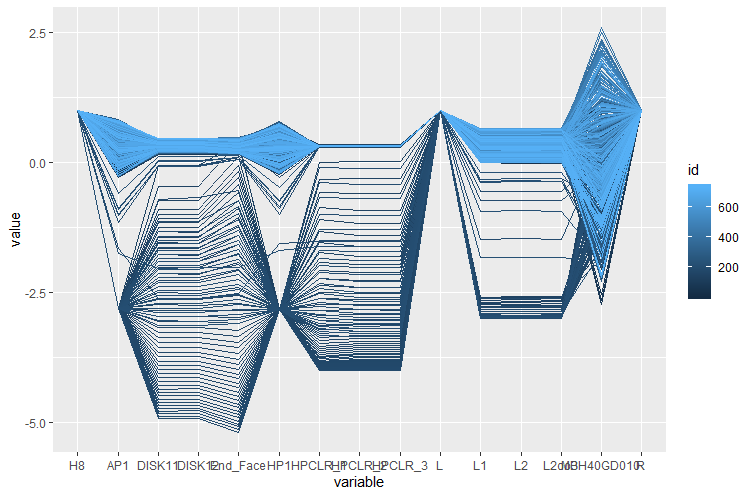
ggparcoord(per\_hour %>%

filter(FunSys\_item==40) %>%

select(-c("date")),

columns = 6:20,

groupColumn = "id")



**Pros:**

* Easy to implement with few lines of code

**Cons:**

* Not interactive

1. **cdparcoord**

# Categorize values into 10 buckets

wb <- discretize(per\_hour %>%

filter(FunSys\_item==40) %>%

select(-c("

date","FunSys\_item","H11","H2","H5","H8","L","R","T1","T3","X1","id")),

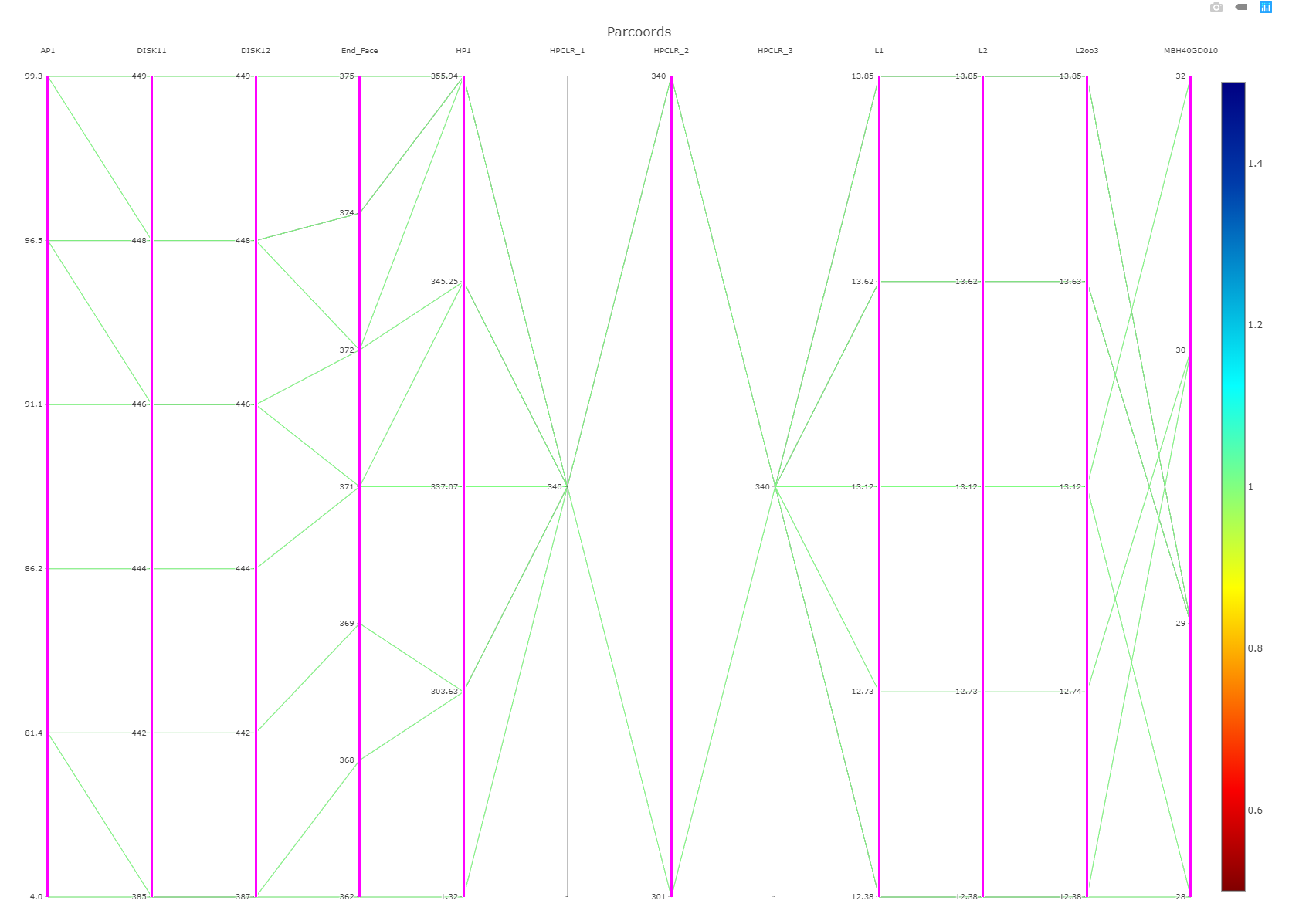
nlevels=10)

# Plot k=-10 least frequent tuples

discparcoord(wb,

k = -10,

saveCounts=FALSE)



**Pros:**

* Less visual clutter - graph only the most frequent lines, with frequency defined as actual tuple count. (In a mixed continuous-categorical setting, the continuous variables are discretized.)
* Interactive

**Cons:**

* Although able to pick outliers, still find it hard to observe abnormal patterns versus normal patterns

**lubridate versus openair**

There is essentially no difference in results between the use of lubridate or openair in aggregating time series except for the syntax.

For example:

Lubridate

df %>%

mutate(date = month(date)) %>%

group\_by(date) %>%

summarize(mean\_X1 = mean(X1))

openair

df = timeAverage(df, avg.time=”month”, statistic=”mean”)

The ‘openair’ package is a compilation of “tools for the analysis of air pollution data” developed by the Environmental Research Group at King’s College London. It is a very well maintained package with its own website at [http://www.openair-project.org](http://www.openair-project.org/).

timeAverage() should be useful in many circumstances where it is necessary to work with different time average data. For example, hourly air pollution data and 15-minute meteorological data. To merge the two data sets timeAverage can be used to make the meteorological data 1-hour means first. Alternatively, timeAverage can be used to expand the hourly data to 15 minute data.