Parallelised Time Series Spike Detection on Hadoop

FlexiScore: Identifying Energy Supply Points with Flexible Electricity Consumption

Timothy Wong[1], Neal Coady[1], Thomas Gerner[2,6], Kathy Chen[3,6], Caroline Alexander[4,6], Ondrej Urban[5,6]

[1] Centrica plc (British Gas)
[2] Heidelberg University
[3] Université de Nantes
[4] National Aeronautics and Space Administration (NASA)
[5] Stanford University
[6] S2DS Pivigo

Correspondence email: timothy.wong@centrica.com

useR! 2017 Brussels

Background

In Great Britain's energy market, there are two roughly equal key drivers for the cost of electricity supplied : (1) wholesale energy price and (2) transmission and distribution cost ("T&D").

Wholesale energy price varies at different times of the day. For instance, power price at peak periods can be over ten times more expensive than off-peak periods within the same day. This creates incentives for suppliers to encourage consumers to shift their energy load to periods where wholesale prices are considerably lower.

With smart meters, energy consumption at every supply point is recorded at half-hourly interval across Great Britain. Such data can be used to establish an individual load profile for each and every supply point. Supply points with flexible consumption can be identified if their consumption deviates from their usual behaviour (i.e. does not conform very well to its previously observed pattern). *FlexiScore* is used to quantify the degree of flexibility and allow comparison between different supply points across the entire portfolio.

Data Preprocessing

Data in the real world is rarely perfect. For instance, duplicate and missing values can be found in many places. This can be due to many legitimate reasons, such as meter removal, signal interruption, system upgrade… etc.

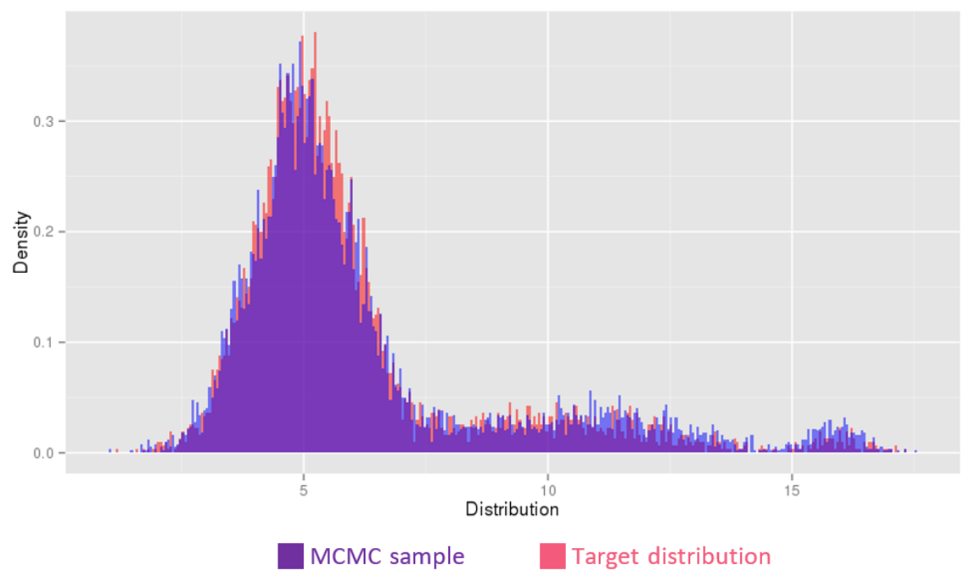
A layer of data preprocessing is introduced to mitigate the problem of imperfect data. Duplicate data is systematically resolved and missing data is permuted. This process creates a continuous (i.e. gapless) stream of regularly-spaced time series data.

Metropolis-Hastings Algorithm (MCMC Sampler)

Metropolis-Hastings sampling algorithm can be used to draw sample from a distribution. The key feature of the sampling algorithm is that the resampled values imitate the original target distribution. This makes it a preferable tool to permute missing data.

The algorithm first calculates the proposal distribution using the original data (i.e. non-missing data). Afterwards, the Markov chain begins with a randomly selected position alongside a proposed jump. The proposed jump will be accepted if the randomly drawn number is smaller than the log-acceptance ratio, otherwise the proposed jump will be rejected and the chain stays at its original position. The Markov chain process repeats many times until sufficient numbers are drawn. In many cases, a predetermined 'burn in' value is set and the first few members from the chain are discarded. This is because earlier part of the chain can be located at a low density part of the distribution. Throwing away the head of the chain would ensure

```
Determine burn-in value $B$
Start at a randomly selected parameter where $u \in \mathbb{R}^k$
FOR $n = 1, 2, 3 \dots N$
    Determine jump $u' \in \mathbb{R}^n$ given the proposal distribution $P(u'|u)$
     Calculate log-likelihood of the proposed jump $u'$
     Draw a random number $a$ from uniform distribution $[0, 1]$
     IF $\log a$ is smaller than the log-acceptance ratio THEN
          Yield $u'$ and set $u = u'$
     ELSE
          Yield $u$
     END IF
END FOR
Discard items $i_1, i_2, i_3 \dots i_{B-1}$
Return items $i_B, i_{B+1}, i_{B+2} \dots i_N$
```



The following histogram shows an imbalanced trimodal distribution (in purple) and the resampled values using MCMC Metropolis-Hastings algorithm. The MCMC sampler is the preferable way for missing data permutation in the preferred implementation of *FlexiScore*. As the generated replacement values and original values have the same underlying distribution, the impact of using permuted data is minimised. One of the major disadvantages of this algorithm is that it adds significant amount of computation time to the distributed computation process.

Data Aggregation

Once the algorithm completes the permutation process, the time series should now be continuous and free from any missing values. As the source data is at half-hourly granularity, we would expect a gapless time series data with exactly 48 valid consumption values per day. This data can be reduced into lower granularity (i.e. fewer values) through sum aggregation. This serves several purposes.

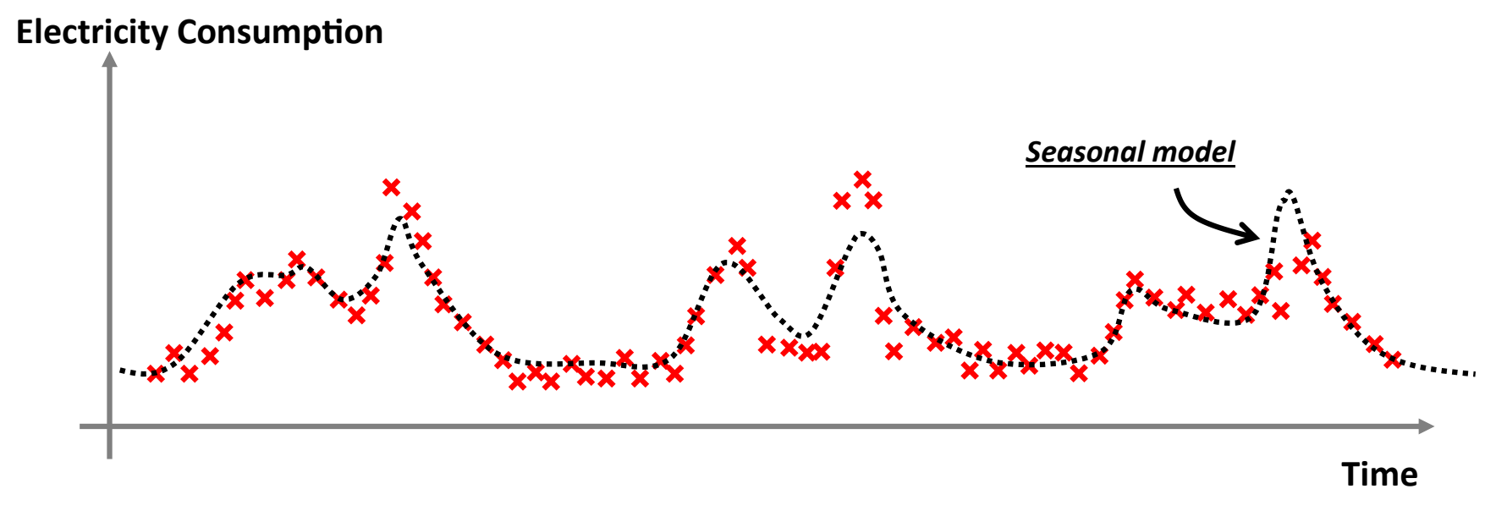
First, it reduces the amount of data we are dealing with, which can marginally speed up the computation process. Secondly, by aggregating consumption values into a wider window (e.g. 1 hour, 2 hours, or more) can effectively adjust the sensitivity of flexibility measurement. Clearly, this is a parameter of the algorithm which is largely determined by the relevant business context. The current *FlexiScore* implementation uses data aggregated to hourly level (i.e. 24 values per day). Various level of aggregation can be tried and tested.

Seasonal Model

Using the regularly-spaced time series data, seasonal models can be established and then the residuals can be passed on to the next stage for further processing. Depending on the context, any time series technique can be used such as *ARMA models*. Our implementation chose to use Fourier transform instead due to its fast performance over large amount of data. Fourier transform is traditionally used in audio analysis but it can also be used to establish seasonal time series models.

To begin with, the FFT function is used to compute the complex vector of the Discrete Fourier Transform. Afterwards, the Fourier modes are sorted and the weakest ones are discarded incrementally. The remaining Fourier modes are then inversed via FFT function and the real portion of the complex vector is used to recreate time domain signal which acts as the seasonal model. It is then subtracted from the original data in order to calculate the residuals. The residuals then feed through the autocorrelation function (ACF) to check for any remaining seasonality. Statistically significant peaks in the ACF are calculated as a percentage. This process repeats iteratively until the percentage exceeds the predefined threshold. Once the iterative process finishes, final seasonal model is returned alongside the model residuals.

```
Determine increment value $I$ and cut-off percentage $C$
Obtain the Fast Fourier Transform of the time series
Normalise the FFT complex vector
Set $M = 0$ and $p = 1$
WHILE $p > C$ THEN
     Set $M = M + I$
     Rank the complex vector by real portion and set the weakest M modes to zero
     Compute inverse FFT using the updated complex vector
     Obtain the real portion of the inverse FFT and compute residual
     Compute autocorrelation function (ACF) of residual
     Set $p$ as the percentage of lags in ACF exceeding threshold
Return the real portion of the complex vector as model fit
```



Spike Detection

After model residual is computed in the above steps, all negative residuals are set to zero as the algorithm is aimed at identifying upward power spikes. With the remaining residual, the first order difference is calculated. For spike detection, the algorithm retains the differenced residual and searches for rising edge (alternatively, falling edge instead). Only spikes with magnitude exceeding the user-defined threshold is retained. The threshold value is set at 1 kWh level for the current implementation of *FlexiScore* but it can be adjusted for different level of sensitivity. In addition, spikes with magnitude weaker than its immediate neighbour is removed. If spikes are close to each other, the weaker one will be taken out as well.

```
Identify rising (or falling) edges
Remove spikes whose magnitude is less than the predefined height
Remove spikes whose magnitude is less than that of its immediate neighbour
Sort the spikes in descending order
FOR $n = N$ where $N$ is the total number of spikes
     If the $n^{th}$ spike is not yet removed THEN
          Remove neighbouring spikes within the predefined distance
     END IF
```

Score Calculation

FlexiScore is simply defined as the percentage of spikey days within a given period. A spikey day indicate the supply point is occupied on the day and the occupants are capable of creating load at an unusual time. *F* value can be calculated as:

$$F = \frac{\text{Number of days with at least 1 spike}}{\text{Number of days}} \quad \text{where } F \in [0\,,\ 1]$$

The *F* value can be calculated in greater detail by looking at certain time period of the day. This allows users to assess flexibility of a specific time period. For instance, *FlexiScore* of peak period can be defined as:

$$F_{peak} = \frac{\text{Number of days with at least 1 spike during peak hours}}{\text{Number of days}}$$

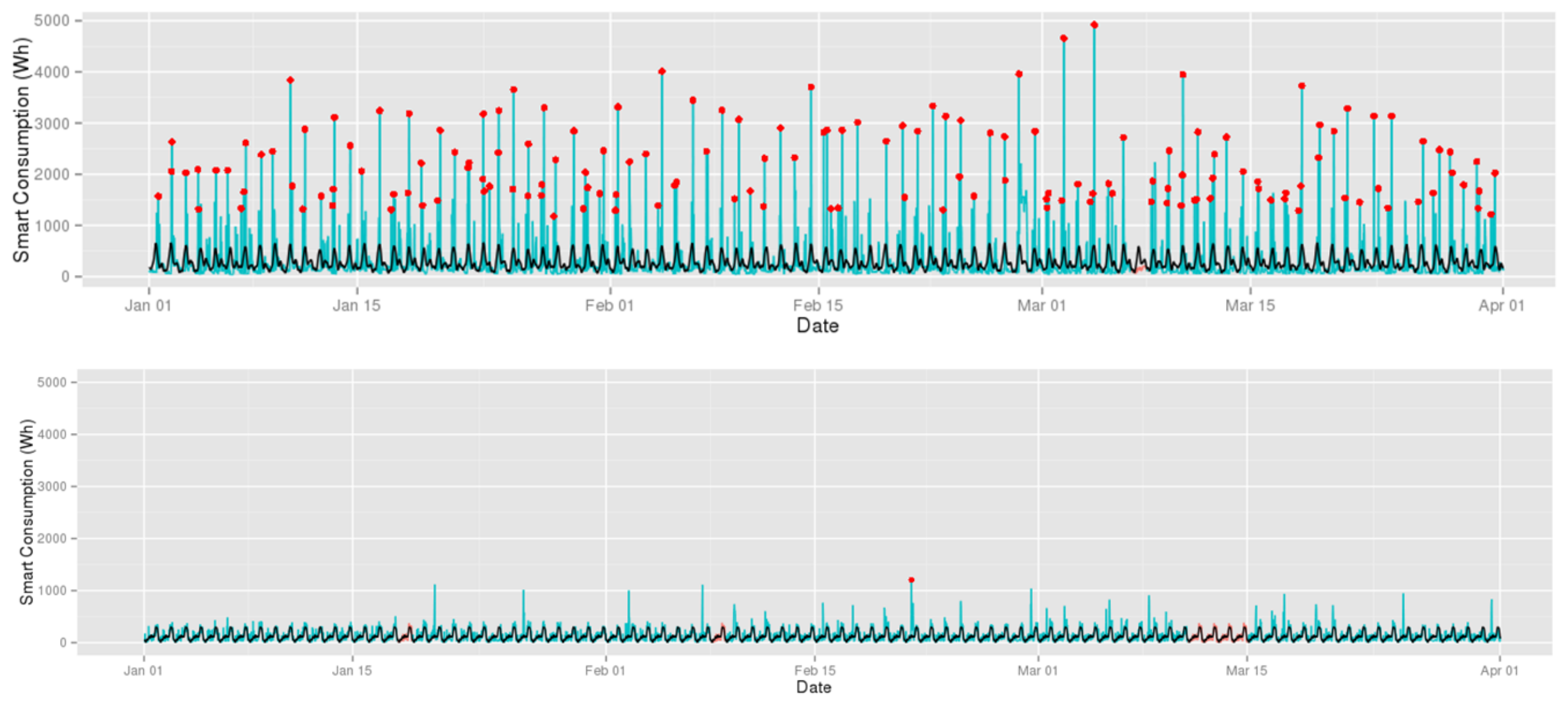
Selected Examples

Example 1: High FlexiScore (F=0.967)

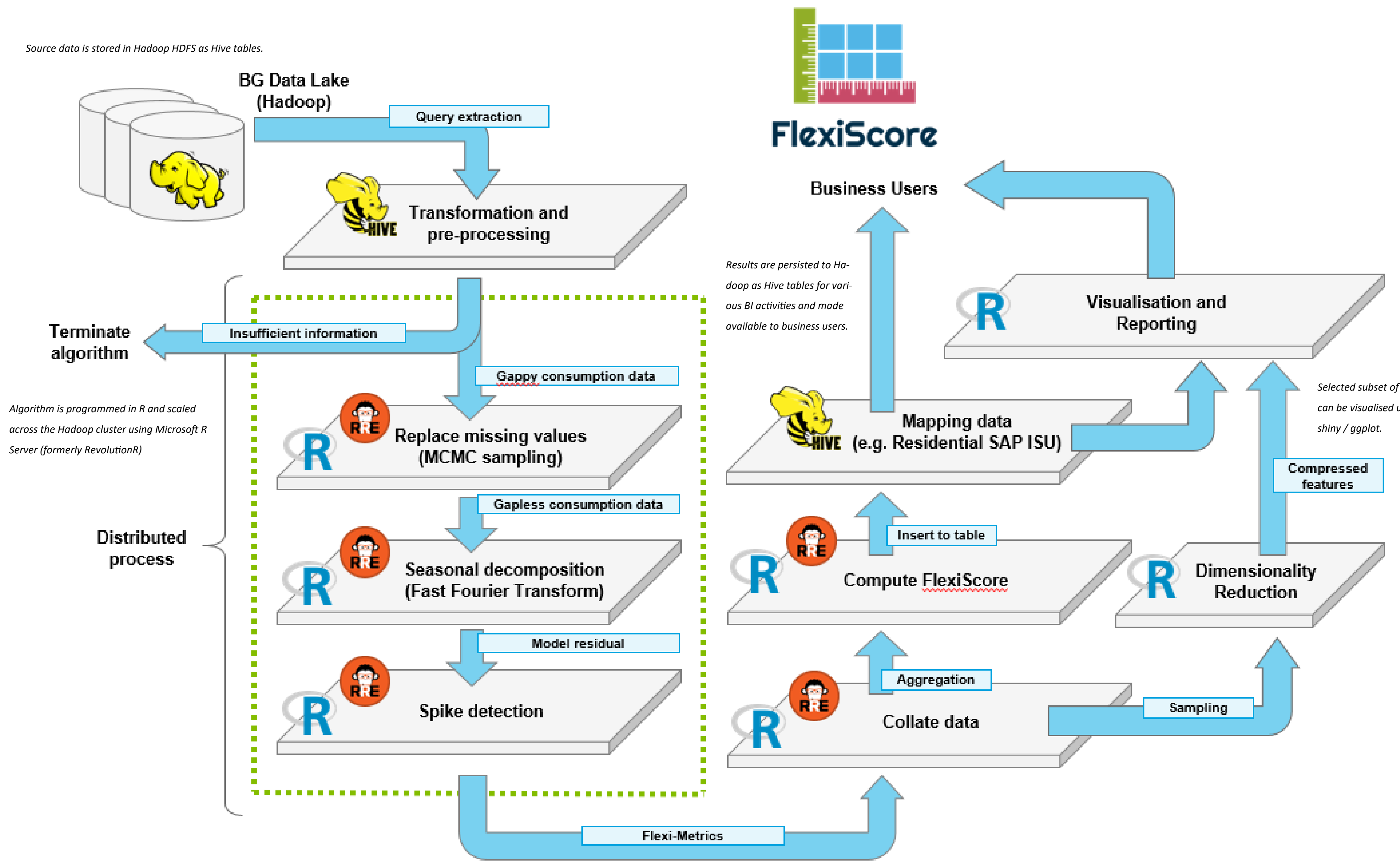
Figure on the left shows an electricity supply point with very high *F* value. Positive model residual which exceed threshold value are considered as spikes and are tagged with red dots. It's observable that the spikes have variable positions over time and variable heights as well. This indicates a less predictable supply point with higher flexibility.

Example 2: Low FlexiScore (F=0.011)

The right figure shows another supply point with fairly low *F* value. The seasonal model already captures most of the variation, meaning this supply point is less flexible.



Technical Architecture


Source data is stored in Hadoop HDFS as Hive tables.
BG Data Lake
(Hadoop)
Query extraction
Transformation and
pre-processing
Terminate
algorithm
Insufficient information
Algorithm is programmed in R and scaled across the Hadoop cluster using Microsoft R Server (formerly RevolutionR)
Gappy consumption data
Replace missing values
(MCMC sampling)
Gapless consumption data
Distributed
process
Seasonal decomposition
(Fast Fourier Transform)
Model residual
Spike detection
Flexi-Metrics
FlexiScore
Business Users
Results are persisted to Hadoop as Hive tables for various BI activities and made available to business users.
Visualisation and
Reporting
Mapping data
(e.g. Residential SAP ISU)
Selected subset of data can be visualised using shiny / ggplot.
Compressed
features
Insert to table
Compute FlexiScore
Dimensionality
Reduction
Aggregation
Collate data
Sampling