Capturing Energy Consumption Patterns in ENDESA's customer network

Endesa Datathon Contest

Paulino Tardáguila

April 2016

Contents

The Idea

Big Data Empire

In the last years, a new set of concepts and ideas have landed among us. Terms such as "The Fourth Industrial Revolution" are now common in everyday language. Undoubtedly, the rise of technological capabilities and its impact is one of the socioeconomic main drivers in nowadays world. The astronomical increase of collected and available data, the so called 'Big Data Era' is probably the most patent outcome of the technological revolution involving many aspects of life.

Focusing on the energy world, Power Utilities start to regard this availability of information not only as a valuable analysis resource, but as the most important pathway to face upcoming challenges in energy generation, distribution and efficient use. The recent massive integration of smart metering devices complements other sources of information affecting energy processes, such as market prices, renewable generation, load curves, meteorological information, etc. Smart grid operation and future energy management will be highly data intensive, and therefore, finding valuable ways to process as much information as possible becomes the main challenge for Utilities in the current times.

A common mistake when approaching so called 'Big Data' problems is to focus on data volume rather than the useful information within. In other words, getting insightful patterns is the main advantage to be obtained from vast amount of data, and the most useful tool towards turning this information availability into better decisions.

Clustering Energy Use Patterns

All that said, the idea presented in this project has to do with extracting useful power use patterns from the outputs of the customer's smart-meter network. Grouping individual consumption profiles into common patterns constitutes an useful tool for large utilities in two ways:

- Provides easily interpretable insights about the actual use of the energy distributed in time and space, and
- Acts as an starting point for the integration of distributed load forecast in Smart Grid planning at a regional level.

This project proposes a clustering system to be applied on the active energy consumption series of individual customers within a time period, aimed at the distribution of them into coherent clusters outlining the different energy use patterns present on the ENDESA's customer network. This system is articulated in two parts:

- Raw smart-meter data processing
- Clustering process

Added value for the Community

The system presented in this report can be regarded as a multi-purpose tool. Getting representative actual profiles of energy consumption together with the features of the members of each group provides a useful instrument for many applications.

Spanish retail energy rates policy

In 2014, Spanish energy regulations included some important modifications in the electricity pricing schemes for retail users. Real market prices were to be applied to small customers consumption. However, the proportion of customers using smart metering devices capable to log hourly consumption is still low. Concerning ENDESA, currently around 4 million out of the total 11 million customers, provide hourly consumption data for applying market prices on their bills. Although there are plans to get a full smart-meter coverage on the customer network by 2018, most of the customers cannot be charged market energy prices because of the lack of hourly data.

To solve this issue, the mentioned regulation provides a pricing scheme to be applied when no hourly consumption data are available. It consists on distributing total measured consumption (normally on a monthly basis) according to "official load profiles" constructed using generic load curves, modified by the actual global load profile during the billing period. This fixed scheme prevents this huge group of customers from taking advantage of prices fluctuations due to energy availability, and consequently affects the whole system by not promoting efficient use of the energy resource.

Observed energy use patterns could be applied to provide hourly load curves based on the actual behavior of the group each customer belongs to, forced not only by general electric load drivers, such as meteorology or calendar variables, but also depending on particular customer typology, such as habits on energy consumption.

Distributed Load Forecast System

Smart Grids are systems designed for the efficient management of increasing complexity energy networks. The growing importance of non scheduled renewable energy sources into the system, turns load forecast into one of the key aspects in Smart Grid design problems. Reliable electricity demand information at a higher spatial distributed level becomes quite relevant in order to manage generation from many different sources.

The possibility of quantifying the amount of installed capacity associated to a certain consumption pattern, and exploring driver variables explaining them, make clustering algorithms a useful tool for load forecasting. By applying grouping techniques, the overall load curve for a region can be more effectively explained by splitting it not only spatially, but according to differentiated patterns too.

Both applications are further discussed in the corresponding section.

Design principles

Along the design and implementation stages of this work, we have stick ourselves to a simplicity principle. Both in terms of statistical procedures and computing structure, decisions have been made seeking the simplest approach that fulfilled required performance criteria. The advantages associated to this are multiple and they are aimed to minimize the hidden technical debt[1] of a potential future implementation of this system:

- Using well tested, simple statistical methods provides robust machine learning models, which tend to behave in a more stable way upon a wider range of situations affecting quality of the inputs.
- Simple architectures, either at software and hardware levels, provide easier (=cheaper) maintenance systems, making them more feasible and scalable.

Dataset Exploration & Preparation

The initial contact with the dataset consisted on the extraction of the customer attributes and the isolation of time series of ACTIVA and REACTIVA. For this purpose, we transformed the dataset from wide (one row per day with column hourly values) to long format (one row per hour), making the access and display of single time series more efficient. During this process some issues related to data quality were identified:

[1] http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf

- `HORA 25` field contained only 0's, therefore, it was removed from the dataset.
- A significant number of customers included duplicated records: duplicated tuples of `IDENTIFICADOR/DIA` fields with different values of `ACTIVA` and `REACTIVA`, as if somehow, one single customer had more than one active metering devices at the same time. Approximately 13% of the 100.000 customers presented this effect, with up to 4 different series per customer id. As an example, querying DIA 20150603 for IDENTIFICADOR 501 we obtained:

```
##          DIA ACTIVA_H1 ACTIVA_H2 ACTIVA_H3 ACTIVA_H4 ACTIVA_H5 ACTIVA_H6
## 1: 20150603       388       319       286       271       267       273
## 2: 20150603       395       325       291       276       272       278
```

As a straightforward way to deal with this, we just account for the first record of each `IDENTIFICADOR/DIA` combination.

- Low quality series. As expected in a dataset of device measurements, it is not uncommon to find outliers, missing values, spans with repeated values or strange shifts in recorded energy consumption. To minimize the noise associated, we have applied a battery of filters in order to remove series not fulfilling some minimum quality standards:
  - **Maximum missing rate filter**: Series with available data below 80% in the working period have been discarded from the clustering analysis.
  - **Minimum variance filter**: To automatically skip series with zeros or repeated values, the 1.5% of series with lowest variances have been removed from the sample. This threshold has been manually tuned to try to avoid the removal of valid series.

As a result, we obtained a filtered set of customer's energy consumption series covering almost one year of hourly data (from 2014/11/01 to 2015/09/30), which constitutes the basic dataset for the clustering process. Around 3800 from the initial 100000 ACTIVA series where removed from the original sample.

Since the clustering matrix cannot include missing values, the last preprocessing step has been to fill in the remaining gaps with simulated data. As a lightweight solution, hourly averages for each customer have been used to complete their corresponding series. Since we had previously limited the maximum rate of missings, the effect of this *rough* filling process remains limited.

Contracted Capacity

We will see later that the clustering technique required normalized time series. Consequently, we need a reference value for each customer. At this point, we have used the **99th percentile of the recorded ACTIVA** values as "Contracted Capacity" approximation. This assumption serves several objectives: while P99 is valid as a reference level for normalization, it helps to filter observed outliers. Besides, it provides an approximation of the power supply level of each user which will be later used for classification purposes.

Reference Scripts

Code corresponding to data preparation can be found in the repository contained in the following scripts:

- `R/funciones_tratamiento_datos.R`: compilation of customized functions for reading, rearranging and displaying different datasets used along the project.
- `R/genera_matriz_cluster_secuencial.R`: Receives the raw initial csv file, performs the explained steps and generates single text files for each customer, each one corresponding to a single row of the clustering matrix, containing normalized hourly values of active energy consumption. This set of files was manually uploaded to an AWS S3 bucket using the `s3cmd` suite, from where it was made available for the rest of the project's applications: as an example, the link https://s3.eu-central-1.amazonaws.com/mdcendesa/user_rowfiles/100000_filtered_nactiva.txt corresponds to the file containing data of customer with `IDENTIFICADOR` = 100000.

Clustering Methodology

The overall purpose of the clustering process is to group customers according to similar energy use patterns. We have focused on two interrelated aspects of energy use:

- Distribution of the energy usage along time (i.e. when does it take place)
- Amount of energy used, relative to the maximum power supply (i.e. how intensive it is)

In the end, our clustering problem is reduced to find similarities in both the shape and the relative values of the consumption profiles. In practical terms, clustering is performed the other way around: measuring dissimilarities among observations to group closest ones by means of an algorithm. Thus, the two main components of a clustering process are the Dissimilarity Measure and the Clustering Algorithm.

Dissimilarity Measure

In our study case, we consider two profiles as similar when they tend to follow a common shape (peaks and valleys are synchronous). In that sense, dissimilarity measures based on correlation between series could be applied. However, these approaches would only focus on the profiles' shape, disregarding proximity on absolute consumption levels. We are interested on the intensity of the energy usage, and for that reason, the selected dissimilarity measure must also seek time series with close values. All this said, we consider the Minkowsky distance of order 2, better known as **Euclidean distance**, a dissimilarity metric that fulfills the problem requirements, with some additional advantages: simple calculation and interpretation, plus is the distance metric upon which K-Means clustering algorithm is defined. To be able to compare profiles of users with different levels of supplied capacity (i.e. different scales), we work with normalized series. Consumption series are represented relative to the 'contracted capacity' of each customer, with values ranging from 0 to 1 (full capacity).

Clustering Algorithm

The selection of the Euclidean Distance as dissimilarity measure allows the use of the **K-Means** clustering algorithm, a widely used iterative method. As a common standard, it is implemented in some Big Data frameworks. Such is the case of Apache Spark's MLLIB. Another advantage of K-Means in opposition to other simple clustering methods like Hierarchical Clustering, is that each cluster is represented by the 'centroid' of its members. When dealing with time series, centroids can be interpreted as *representative profiles* of each group, providing a useful output in terms of graphical interpretation.

Reference Scripts

Code corresponding to the clustering process is contained in the following script:

- `spark/makeclusters.py`: This Python script manages the clustering process from the master node of the computing distributed system. Initializes the Spark context, retrieves the matrix from the S3 bucket in form of Spark's distributed dataset (RDD), performs the K-means clustering algorithm and saves the output (cluster assignation and centroids) into local text files in the `output` directory.

Some Results

The results shown here correspond to the execution of the K-Means algorithm on the full customer matrix. Series have been aggregated into 20 clusters performing 10 runs with randomized starting point. The number of clusters has been manually chosen from test runs on smaller subsets, and provides a first approach of the clustering process capabilities.

For a comprehensive analysis of the clustering results, a set of interactive visualizations has been included in a dashboard available through the following link: https://paulino-tardaguila.shinyapps.io/ClusterViz/

The information in this web app is distributed in 3 tabs:

- **Time Series**: Interactive chart with representative series of the selected clusters. Provides a quick view of the clustered patterns along the study period. It allows to show moving averages, useful to explore patterns from hourly to smaller time resolution (daily, weekly, etc). We can clearly see how the clusters are configured, with patent emerging time patterns within day, week and seasonal variation. Clusters also differentiate levels of consumption intensity. The charts in the lower row provide information about the cluster's composition: the one on the left depicts cluster composition in terms of number of customers and accumulated power within each one, classified by some user attributes. The one on the right displays the deviation of each class frequency within cluster from the overall frequency (full customer group). It effectively shows composition patterns within the clusters. For example: regarding customer type (CNAE field), we see how commercial customers frequency is higher in clusters 1, 2, 3, 8, 16 and 17. On the opposite, clusters 4, 5, 7, 13, 14 present higher density of household clients. Classifying by contracted capacity class, the same pattern emerges by linking commercial customers to higher capacity classes, which is logical.
- **Averaged Patterns**: In this section, weekly and seasonal patterns are displayed through day-of-week and month average values and deviation from average values. Clusters retaining week-end patterns (1, 16, 18) emerge clearly, as well as the different seasonal variation curves present across ENDESA's customer network. For example, clusters with increased consumption in winter are likely to group customers from colder areas with electric heating systems. On the other side, cluster 15 exhibits a quite differentiated profile with a marked summer increase in demand due to high temperatures.
- **Hourly Averaged Patterns**: A single faceted chart is presented here, showing the relationship of intra-day patterns across months for each cluster. Seasonal differences are effectively outlined: we can see how in cluster 2, which is clearly grouping "nightly consumers", the beginning of the consumption period is delayed during summertime. This pattern could be linked to electricity used for lightning. In other cases, we can evaluate how clusters with similar hourly patterns show different seasonal variations: 4 and 6, for example. There is actually a huge amount of information to be extracted and analyzed in this plot.

Reference Scripts

The source code of the Visualization app is located in the `R/report/ClusterViz` directory in the repository.

- `Index.Rmd`: source code of the website. It is written using Rmarkdown syntax, and is automatically parsed into html by shiny-server on the host.
- `aux.R`: auxiliary R script to construct the charts from the clustering output files, which for simplicity reasons have been included in the app directory under the `output` folder.

This report's source code can be accessed in:

- `R/report/paulino_tardaguila_report.Rmd`

Technology

The implementation of this proposal comprised three phases in which different combination of software and hardware resources have been used, always keeping in mind a design aimed at effortless scaling up the system to meet the requirements of a real world problem.

Data Preparation Stage

- **Software**: This part of the work has been built upon R as programming language and graphical framework. To deal with the full dataset, instead of common data.frames, data.table objects have been used, which are better for handling large objects in memory. Graphical analysis of the information has been carried out mainly using dygraphs interactive plots. The synchronization between local and cloud data was performed using the **s3cmd** package.
- **Hardware**:
  - Preliminary exploratory analyses, proofs of concept tests and creation of the clustering matrix has been executed on a local Ubuntu 14.04 LTS system (64 GB RAM, 32 cores)
  - The distributed filesystem containing the full clustering matrix is located in a S3 Bucket, hosted by Amazon Web Services. This type of storage provides secure and reliable access, being specially efficient to work in association which cloud computing systems from the same vendor.

Clustering Stage

This part of the work has been entirely conducted using Apache Spark infrastructure on a AWS cloud computing platform specially dimensioned for the size of the task.

- **Software**:
  - Big data framework: **Spark 1.4 on Hadoop 2.4**. Hadoop version was chosen to avoid some bugs related to the generation of Distributed Datasets from data stored in S3, experienced using the latest version (2.6). A slight downgrade solved this little inconvenience.
  - Python's pyspark package has been used, specially the clustering functions implemented in its MLlib package: **pyspark.mllib.clustering**
- **Hardware**: We have chosen AWS Elastic Cloud Computing (EC2) service as the supporting infrastructure. Spark installation provides an *spark-ec2* script with built-in functions to easily launch, configure and re-size Spark clusters based on EC2 instances. In this case, we set a cluster of 1 master + 6 slave nodes (m3.xlarge type, with 4 cores and 15GB RAM each). For clusters of this size, Spark's Standalone Cluster Manager is enough. The experienced computational performance was outstanding regarding the costs. Executing K-means algorithm on an approx. 96500 * 8760 matrix took roughly 2 hours.

Results Analysis Stage

In this section, an interactive visualization application has been built. The utilities used to build up this report can be included here too.

- **Software**: In this part of the work we have turned back to R, as it provides excellent tools for data visualization and interactive web development and deployment. ggplot2, dygraphs, RColorBrewer and plotly packages were used to produce graphical summaries of the information contained in the clusters. The visualization app was developed with the excellent flexdashboard package, based on rmarkdown and shiny libraries. This report has been designed and compiled using the **knitr** package, based on rmarkdown too.
- **Hardware**: The visualization web app was deployed on the shinyapps.io server, which allows users to share R based web applications made using shiny and related libraries.

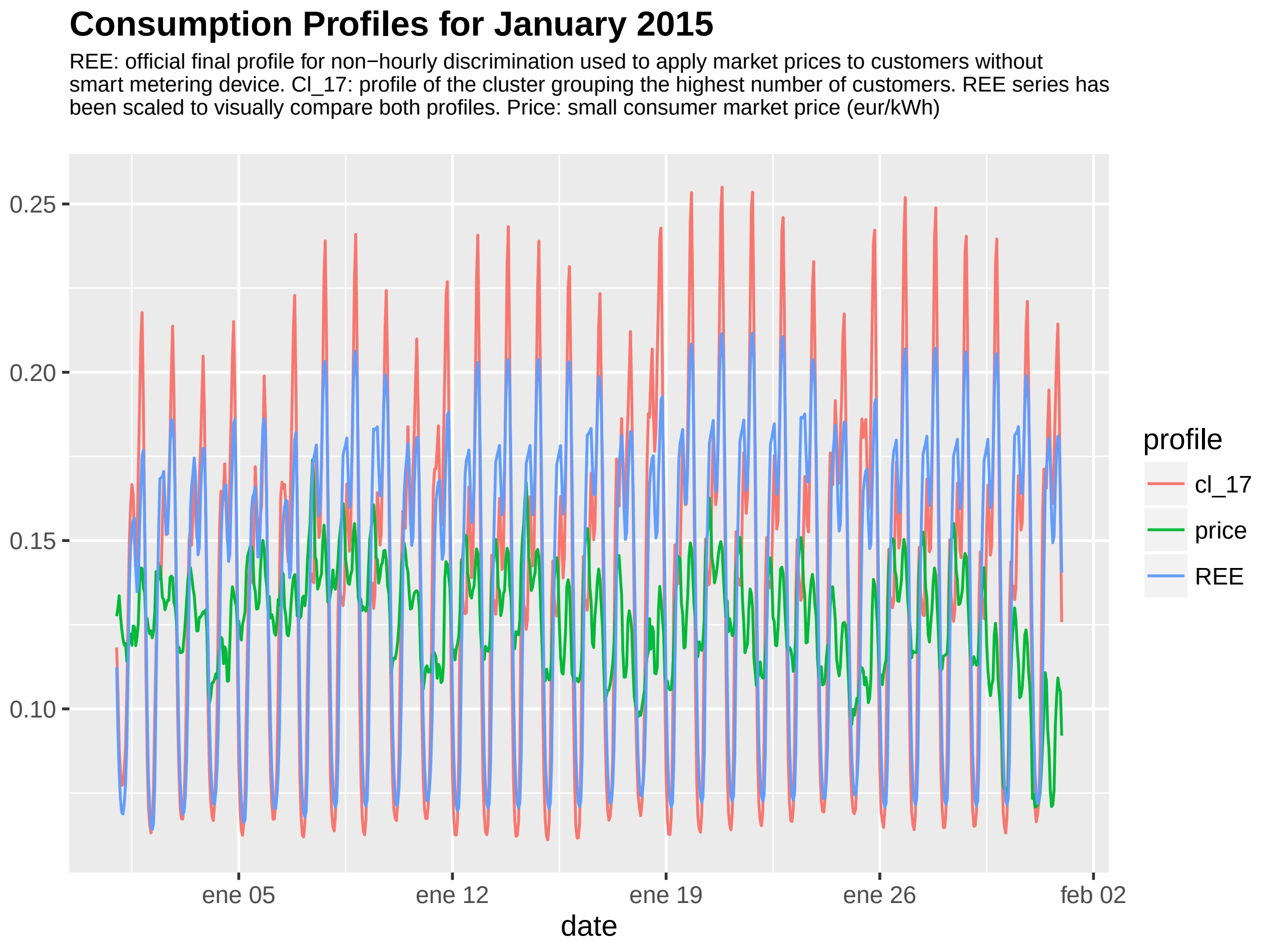
Applications

In this section we outline some applications of the clustering results that were previously introduced in the first part of the report. None of them has been developed in depth, therefore, what presented here constitutes a summary of possible further developments using the clustering outcome as starting point.

Enhanced consumption profiles for hourly pricing

One of the outputs of the clustering process is the assignation of each customer to one of the defined clusters. Indirectly, a qualitative variable classifying customers according to the energy behavior has been defined for the customers dataset. We could then construct a classifications aimed at assigning customers without recorded consumption profiles to a certain cluster, according to some explanatory variables such as customer type, contracted capacity, location and many other present in ENDESA's data base. In any case, a more detailed assessment of the quality of the cluster model might be necessary. Since we would be constructing this classification system upon a derived variable, we have to obtain a clear estimation of the effectiveness of the initial grouping model.

After the classification, an hourly usage profile can be assigned to every single customer, representing the average consumption pattern exhibited by similar customers. This is likely to reflect in a better way the actual load distribution of each customer than the generic profile. Moreover, if this information is made available to the customer through integrating currently available price curves with tailored real consumption profiles, users without smart meters could also benefit from market information to optimize their energy usage. Official REE consumption profiles are publicly available on line.



Virtual node network for distributed load forecast

An ideal distributed load forecast system should be able to provide information of electricity demand timely and spatially distributed. One common approach for this task is to define a spatial network and estimate values at the node scale. This nodes are placed according to the layout of the transmission grid. In this scenario, clustering processes could be applied on every single spatial node. By grouping customers connected to a given node, total load on it can be decomposed on a "sub-node" level according to the existing patterns within that node and the relative importance of each one in terms of aggregated capacity. Simple load forecast models could be adjusted to each "sub-node" in order to aggregate their output on the node level.

Scaling Up

One of the main features of a good prototype of a Big Data system is the possibility to scale it up as seamlessly as possible to meet future growing requirements in terms of amount of processed information, while keeping computing and results delivery times within a reasonable range. For this purpose, it becomes essential to effectively combine tools that are scalable by themselves with processes that are robust and reliable when applied to bigger inputs. We have followed this principle along the design phase of this project, and tried to do so at the implementation stage.

A potential upgrade of the proposed system would comprise two different parts:

- **Data integration and processing**: The data preparation workflow presented so far might undergo some adaptation to face the real big data scenario. The current implementation (mainly based on data.table objects on R) is supposed to be suitable for managing datasets of a size up to some 100 GB. Nevertheless, more efficient paralellization techniques should be integrated, preferably on a distributed computing environment. In any case, for a full-sized problem, the best approach would be to rely on Spark also for this phase of the work. The integration of Spark Streaming module would make data assimilation quite efficient, and would enhance real time functionalities, quite interesting for the applications mentioned in the former section.
- **Time series clustering**: The current design of the clustering model is directly scalable to bigger problems by scaling up the computing platform. This is actually a trivial task for which AWS systems are specially designed. It may be useful to tackle data storage necessities using some more specialized architectures than raw text filesystems. Nevertheless, as already said, this design is robust enough to manage significantly bigger amounts of data.

One Final Note

In the development of the this work, some aspects included on the first round proposal have been only slightly approached or even skipped. This shortage with respect to the original plan has been forced by circumstances that fall outside the scope of this report. In any case, the remaining parts that were outlined in the original proposal could be interpreted as future developments to tackle as the evolution of this work.

Madrid, April 30th 2016