By clustering of consumers of electricity load, we can extract typical load **profiles**, improve the accuracy of consequent electricity consumption forecasting, detect anomalies or monitor a whole smart grid (grid of consumers) (Laurinec et al. (2016), Laurinec and Lucká (2016)). I will show you the first use case, the extraction of typical electricity load profiles by [**K-medoids**](https://en.wikipedia.org/wiki/K-medoids) clustering method.

Firstly, let’s load the required packages, data (elec\_load) and return the dimensions of the dataset.

library(TSrepr)

library(ggplot2)

library(data.table)

library(cluster)

library(clusterCrit)

data("elec\_load")

dim(elec\_load)

## [1] 50 672

There are 50 time series (consumers) of length 672, specifically time series of length 2 weeks of electricity consumption.  
These measurements are from smart meters from an unknown country.

It is obvious that dimensionality is too high and the [**curse of dimensionality**](https://en.wikipedia.org/wiki/Curse_of_dimensionality) can happen.  
For this reason, we have to reduce dimensionality in some way. One of the best approaches is to use time series representations in order to reduce dimensionality, reduce noise and emphasize the main characteristics of time series.

For double seasonal time series of electricity consumption (daily and weekly seasonality), the model-based representation approaches seem to have best ability to extract typical profiles of consumption.  
However, I will use some other representation methods implemented in **TSrepr** too.

Let’s use one of the basic model-based representation methods – **mean seasonal profile**. It is implemented in the repr\_seas\_profile function and we will use it alongside repr\_matrix function that computes representations for every row of a matrix of time series. One more very important notice here, normalisation of time series is a necessary procedure before every clustering or classification of time series. It is due to a fact that we want to extract typical curves of consumption and don’t cluster based on an amount of consumption.  
By using the **TSrepr** package, we can do it all in one function – repr\_matrix. We will use z-score normalisation implemented in norm\_z function.  
We have 48 measurements during a day, so set freq = 48.

data\_seasprof <- repr\_matrix(elec\_load, func = repr\_seas\_profile,

args = list(freq = 48, func = mean),

normalise = TRUE, func\_norm = norm\_z)

dim(data\_seasprof)

## [1] 50 48

We can see that dimensionality was in fact reduced significantly. Now, let’s use the K-medoids (pam function from cluster package) clustering method to extract typical consumption profiles.  
Since we don’t know a proper number of clusters to create, i.e. a priori information about a number of clusters, an [internal validation index](https://en.wikipedia.org/wiki/Cluster_analysis#Internal_evaluation) is have to be used to determine an optimal number of clusters.  
I will use well known [Davies-Bouldin index](https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index) for this evaluation. By Davies-Bouldin index computation, we want to find a minimum of its values.

I will set the range of number of clusters to 2-7.

clusterings <- lapply(c(2:7), function(x)

pam(data\_seasprof, x))

DB\_values <- sapply(seq\_along(clusterings), function(x)

intCriteria(data\_seasprof, as.integer(clusterings[[x]]$clustering),

c("Davies\_Bouldin")))

Let’s plot results of internal evaluation.

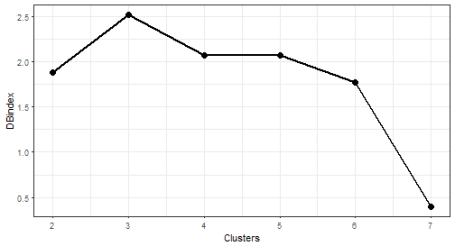
ggplot(data.table(Clusters = 2:7, DBindex = unlist(DB\_values)),

aes(Clusters, DBindex)) +

geom\_line(size = 1) +

geom\_point(size = 3) +

theme\_bw()



The “best” number of clusters is 7 (lowest value of the index).

Let’s plot the results of clustering with 7 number of clusters. I will use computed medoids stored in the object clustering. For a visualization, the facet\_wrap function is very handy here.

# prepare data for plotting

data\_plot <- data.table(melt(data.table(class = as.factor(clusterings[[6]]$clustering),

data\_seasprof)))

data\_plot[, Time := rep(1:ncol(data\_seasprof), each = nrow(data\_seasprof))]

data\_plot[, ID := rep(1:nrow(data\_seasprof), ncol(data\_seasprof))]

# prepare medoids

centers <- data.table(melt(clusterings[[6]]$medoids))

setnames(centers, c("Var1", "Var2"), c("class", "Time"))

centers[, ID := class]

# plot the results

ggplot(data\_plot, aes(Time, value, group = ID)) +

facet\_wrap(~class, ncol = 2, scales = "free\_y") +

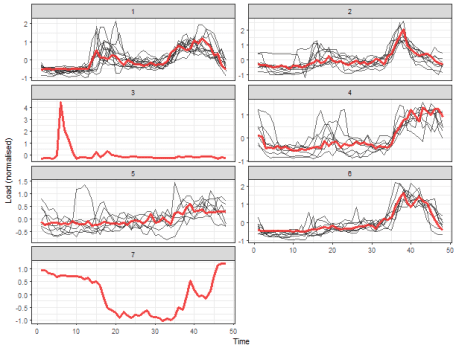
geom\_line(color = "grey10", alpha = 0.65) +

geom\_line(data = centers, aes(Time, value),

color = "firebrick1", alpha = 0.80, size = 1.2) +

labs(x = "Time", y = "Load (normalised)") +

theme\_bw()



We can see 5 typical extracted profiles – red lines (medoids of clusters). The next two clusters are one-elemental, so they can be referred as outliers.

Now, let’s try some more sophisticated method for the extraction of seasonal profiles – **GAM regression coefficients**. By repr\_gam function, we can extract double seasonal regression coefficients – daily and weekly (you can read more about the [analysis of seasonal time series with **GAM** here](https://petolau.github.io/Analyzing-double-seasonal-time-series-with-GAM-in-R/)).  
However, the number of weekly seasonal regression coefficients will be only 6, because the number of the second seasonality coefficients is set to freq\_2 / freq\_1, so 48\*7 / 48 = 7, and 7 – 1 = 6 because of splines computation (differentiation). I will again use repr\_matrix function.

data\_gam <- repr\_matrix(elec\_load, func = repr\_gam, args = list(freq = c(48, 48\*7)),

normalise = TRUE, func\_norm = norm\_z)

dim(data\_gam)

## [1] 50 53

So the dimension is 47 + 6 = 53 because of usage of splines in GAM method. Let’s cluster the data and visualize results of it.

clusterings <- lapply(c(2:7), function(x)

pam(data\_gam, x))

DB\_values <- sapply(seq\_along(clusterings), function(x)

intCriteria(data\_gam, as.integer(clusterings[[x]]$clustering),

c("Davies\_Bouldin")))

Let’s plot results of internal evaluation.

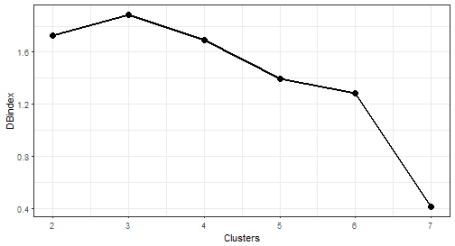
ggplot(data.table(Clusters = 2:7, DBindex = unlist(DB\_values)),

aes(Clusters, DBindex)) +

geom\_line(size = 1) +

geom\_point(size = 3) +

theme\_bw()



The optimal number of clusters is again 7. Let’s plot results.

# prepare data for plotting

data\_plot <- data.table(melt(data.table(class = as.factor(clusterings[[6]]$clustering),

data\_gam)))

data\_plot[, Time := rep(1:ncol(data\_gam), each = nrow(data\_gam))]

data\_plot[, ID := rep(1:nrow(data\_gam), ncol(data\_gam))]

# prepare medoids

centers <- data.table(melt(clusterings[[6]]$medoids))

setnames(centers, c("Var1", "Var2"), c("class", "Time"))

centers[, ID := class]

# plot the results

ggplot(data\_plot, aes(Time, value, group = ID)) +

facet\_wrap(~class, ncol = 2, scales = "free\_y") +

geom\_line(color = "grey10", alpha = 0.65) +

geom\_line(data = centers, aes(Time, value),

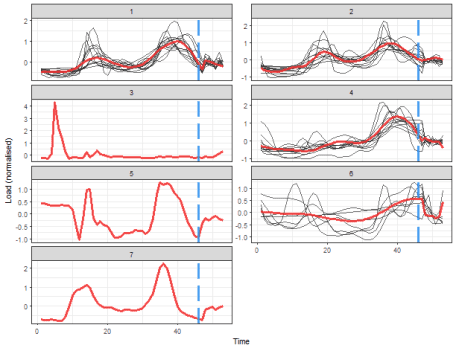
color = "firebrick1", alpha = 0.80, size = 1.2) +

geom\_vline(xintercept = 46, color = "dodgerblue2",

size = 1.4, linetype = 5, alpha = 0.8) +

labs(x = "Time", y = "Load (normalised)") +

theme\_bw()



That is a more delightful result, isn’t it? Extracted consumption profiles are smoother than in the case of average seasonal profiles. The blue dashed line borders daily and weekly seasonal coefficients.  
The K-medoids now extracted 4 typical profiles and determined 3 one-element clusters.

I will show you also result of clustering of some nondata adaptive representation, let’s pick for example **DFT** (Discrete Fourier Transform) method and extract first 48 DFT coefficients.

data\_dft <- repr\_matrix(elec\_load, func = repr\_dft, args = list(coef = 48),

normalise = TRUE, func\_norm = norm\_z)

dim(data\_dft)

## [1] 50 48

clusterings <- lapply(c(2:7), function(x)

pam(data\_dft, x))

DB\_values <- sapply(seq\_along(clusterings), function(x)

intCriteria(data\_dft, as.integer(clusterings[[x]]$clustering),

c("Davies\_Bouldin")))

Let’s plot results of internal evaluation.

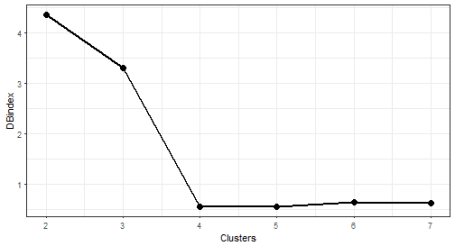
ggplot(data.table(Clusters = 2:7, DBindex = unlist(DB\_values)),

aes(Clusters, DBindex)) +

geom\_line(size = 1) +

geom\_point(size = 3) +

theme\_bw()



We can see nice [“elbow”](https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set#The_elbow_method) here at 4 number of clusters.

# prepare data for plotting

data\_plot <- data.table(melt(data.table(class = as.factor(clusterings[[3]]$clustering),

data\_dft)))

data\_plot[, Time := rep(1:ncol(data\_dft), each = nrow(data\_dft))]

data\_plot[, ID := rep(1:nrow(data\_dft), ncol(data\_dft))]

# prepare medoids

centers <- data.table(melt(clusterings[[3]]$medoids))

setnames(centers, c("Var1", "Var2"), c("class", "Time"))

centers[, ID := class]

# plot the results

ggplot(data\_plot, aes(Time, value, group = ID)) +

facet\_wrap(~class, ncol = 2, scales = "free\_y") +

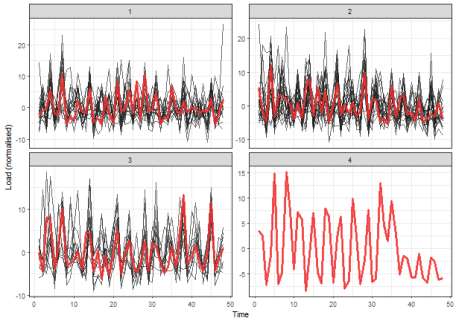
geom\_line(color = "grey10", alpha = 0.65) +

geom\_line(data = centers, aes(Time, value),

color = "firebrick1", alpha = 0.80, size = 1.2) +

labs(x = "Time", y = "Load (normalised)") +

theme\_bw()



The interpretability of these results would be difficult. Therefore, model-based time series representations are very effective in this use case (so the typical profiles extraction).

I will show you the usage of one more representation method – **FeaClip**. The **FeaClip** is feature extraction method from a clipping representation. The windowing approach alongside FeaClip is recommended to use for every day of time series. The big advantage is that normalisation is not needed alongside FeaClip method. Let’s use it in our case. The windowing method is also directly implemented in repr\_matrix function.

data\_feaclip <- repr\_matrix(elec\_load, func = repr\_feaclip,

windowing = TRUE, win\_size = 48)

dim(data\_feaclip)

## [1] 50 112

clusterings <- lapply(c(2:7), function(x)

pam(data\_feaclip, x))

DB\_values <- sapply(seq\_along(clusterings), function(x)

intCriteria(data\_feaclip, as.integer(clusterings[[x]]$clustering),

c("Davies\_Bouldin")))

Let’s plot results of internal evaluation.

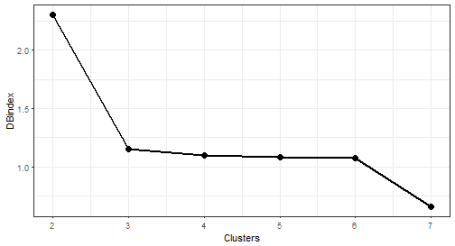
ggplot(data.table(Clusters = 2:7, DBindex = unlist(DB\_values)),

aes(Clusters, DBindex)) +

geom\_line(size = 1) +

geom\_point(size = 3) +

theme\_bw()



We can see that now 2 “elbows” appeared. The biggest change is between 2 and 3 number of clusters, so I will choose number 3.

# prepare data for plotting

data\_plot <- data.table(melt(data.table(class = as.factor(clusterings[[2]]$clustering),

data\_feaclip)))

data\_plot[, Time := rep(1:ncol(data\_feaclip), each = nrow(data\_feaclip))]

data\_plot[, ID := rep(1:nrow(data\_feaclip), ncol(data\_feaclip))]

# prepare medoids

centers <- data.table(melt(clusterings[[2]]$medoids))

setnames(centers, c("Var1", "Var2"), c("class", "Time"))

centers[, ID := class]

# plot the results

ggplot(data\_plot, aes(Time, value, group = ID)) +

facet\_wrap(~class, ncol = 2, scales = "free\_y") +

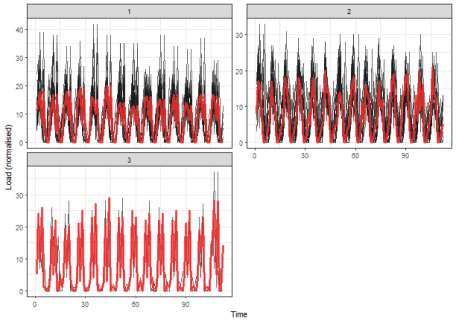
geom\_line(color = "grey10", alpha = 0.65) +

geom\_line(data = centers, aes(Time, value),

color = "firebrick1", alpha = 0.80, size = 1.2) +

labs(x = "Time", y = "Load (normalised)") +

theme\_bw()



It seems like good separability of clusters, better than in the case of DFT.  
But you can check also other results of clusterings with the different number of clusters.

**Conclusion**

In this tutorial, I showed you the usage of time series representation methods to create more characteristic profiles of consumers.  
Then, time series representations, calculated by **TSrepr** package, were clustered by K-medoids and typical consumption profiles were extracted from created clusters.

Time series representations can be helpful also in other use cases as classification or time series indexing.  
You can check how I use time series representations in my dissertation thesis in more detail on the [**research** section](https://petolau.github.io/research) of this site.

**References**

Laurinec, Peter, and Mária Lucká. 2016. “Comparison of Representations of Time Series for Clustering Smart Meter Data.” In Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering and Computer Science 2016, 458–63.

Laurinec, Peter, Marek Lóderer, Petra Vrablecová, Mária Lucká, Viera Rozinajová, and Anna Bou Ezzeddine. 2016. “Adaptive Time Series Forecasting of Energy Consumption Using Optimized Cluster Analysis.” In Data Mining Workshops (Icdmw), 2016 Ieee 16th International Conference on, 398–405. IEEE.