fdacluster: Clustering for Functional Data

A. Stamm[1] L. Bellanger[1] L.M. Sangalli[2] P. Secchi[2] S. Vantini[2]

[1] Laboratoire de Mathématiques Jean Leray, UMR CNRS 6629, Nantes Université, France.
[2] Department of Mathematics F. Brioschi, Politecnico di Milano, Italie.

Abstract

The **fdacluster** package provides implementations of the popular k-means and hierarchical agglomerative clustering (HAC) methods for functional data (Ramsay and Silverman 2005). Variability in functional data can be divided into three components: *amplitude*, *phase* and *ancillary* variability (Vantini 2012; Marron et al. 2015). The first two sources of variability can be captured with a statistical analysis that integrates a *curve alignment* step. The k-means and HAC algorithms implemented in **fdacluster** provide clustering structures that are based either on amplitude variation (default behavior) or phase variation (Marron et al. 2014). This is achieved by jointly performing clustering and alignment of a functional data set. The two main related functions are `fdakmeans()` for the k-means method which implements mainly the algorithm proposed in Sangalli et al. (2010) and `fdahclust()` for HAC methods. All methods handle *multivariate Euclidean* codomains.

Keywords: functional data – amplitude & phase variability – k-means – hierarchical clustering – R package.

Development

The **fdacluster** package is structured into 3 main items:

1. A dedicated `S3` class for storing results of clustering with amplitude and phase separation;
2. Implementations of clustering methods for functional data;
3. Tools for comparing various clustering strategies.

The `caps` class

The `caps` class is a novel `S3` class which aims at storing in a standardized fashion the output of any functional clustering method that relies on phase and amplitude variability separation. This class is meant to encapsulate them into a single object for providing dedicated `S3` methods for e.g. plotting, summarizing, etc. The name of the class stems from **C**lustering with **A**mplitude and **P**hase **S**eparation. Dedicated visualization tools are implemented for `caps` objects via an `S3` specialization of `ggplot2::autoplot()` which returns a `ggplot2::ggplot` object that can be further customized and via an `S3` specialization of `graphics::plot()` as well for convenience. An additional visualization function called `diagnostic_plot()` plots the distance-to-center and silhouette values for each observed curves as a diagnostic tool.

Clustering methods

The **fdacluster** package provides implementations of the k-means algorithm and of a proposed hierarchical agglomerative clustering (HAC) method for functional data. Both methods are maximally parallelized using the **futureverse** framework. Similarly, the user can perform clustering with a progress bar which can be useful when computation time is expected to be large. This means that, when the user is satisfied with the code (s)he produced, (s)he can turn on parallelization and progress bar as follows:

```
library(future)
library(progressr)
handlers("rstudio") # Defines RStudio progress bar style
plan(multisession, workers = 8) # Turns ON parallelization
```

```
with_progress(
  # User-defined code for clustering
)
plan(sequential) # Turns OFF parallelization
```

k-means

The k-means algorithm is implemented in the function `fdakmeans()` which mainly follows Sangalli et al. (2010). Most of the implementation is programmed in C++ using the **RcppArmadillo** package which wraps the **armadillo** library. The normalization step which pertains to normalizing estimated warping functions is however implemented according to Vantini (2012) instead of the original proposal in Sangalli et al. (2010) because it generalizes better and seems to be the most theoretically sound approach. A number of seeding strategies are implemented as well for choosing the initial centroids. The best trade-off between computation time and global convergence of the algorithm seems to be the so-called `exhaustive-kmeans++` strategy which implements the k-means++ strategy described in Arthur and Vassilvitskii (2007) starting with every observed curve as first centroid and picking the one yielding the lowest total WSS.

The alignment is performed by minimizing a chosen distance (currently L^2 or Pearson-derived) over a set of warping functions. For functional data defined on the real line, warping classes can be either `shift`, `dilation` or `affine`. For functional data defined on a specific interval, the square-root slope function (SRSF) framework described in Tucker, Wu, and Srivastava (2013) is used which minimizes the L^2 distance between the SRSFs of the original curvesa over the set of all boundary-preserving warping functions. For computing the cluster centroids, the `fdakmeans()` function provides 4 options: `mean`, `medoid`, `lowess` and `poly`.

HAC

The `fdahclust` function implements functional HAC:

1. Functional HAC needs to account for the intrinsic amplitude and phase variability inherent to functional data. It is therefore natural when computing the distance between two curves to minimize such a distance over all possible warping functions in a chosen class, i.e. to integrate alignment when computing the distance matrix. The idea is to assess how far two curves are after optimal phase alignment. This step is achieved through the `fdacluster::fdadist()`.
2. The process of building up the hierarchy of possible clustering structures does not change once the distance matrix D is known. In fact, from a practical point of view with `R`, it is still achieved via a call to `stats::hclust()`. Here, the user can choose which linkage criterion to use for assessing distances between sets of curves. For the sake of simplicity, only a subset of choices from the `stats::hclust()` function is actually available in **fdacluster**, namely `complete`, `average`, `single` and `ward.D2`.
3. Once the dendrogram is built, a big difference between multivariate HAC and functional HAC stems from the fact that the latter requires the choice of the number of clusters, hence making Step 3 mandatory as well. This is because, once the grouping structure has been chosen, all the curves need to undergo a last alignment to the centroid of the cluster they have been assigned to. Then, and only then, within-cluster sum of squared distances to centroid and silhouette values can be assessed. This step is achieved via `fdacluster::fdakmeans()` with $k = 1$ and cluster medoid as initial centroid.

References

Arthur, D., and S. Vassilvitskii. 2007. "K-Means++ the Advantages of Careful Seeding." In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1027–35.

Marron, J. S., J. O. Ramsay, L. M. Sangalli, and A. Srivastava. 2014. "Statistics of Time Warpings and Phase Variations."

———. 2015. "Functional Data Analysis of Amplitude and Phase Variation." *Statistical Science*, 468–84.

Ramsay, J., and B. W. Silverman. 2005. *Functional Data Analysis*. Springer Series in Statistics. Springer.

Sangalli, L. M., P. Secchi, S. Vantini, and V. Vitelli. 2010. "K-Mean Alignment for Curve Clustering." *Computational Statistics & Data Analysis* 54 (5): 1219–33.

Tucker, J. D., W. Wu, and A. Srivastava. 2013. "Generative Models for Functional Data Using Phase and Amplitude Separation." *Computational Statistics & Data Analysis* 61: 50–66.

Vantini, S. 2012. "On the Definition of Phase and Amplitude Variability in Functional Data Analysis." *Test* 21 (4): 676–96. https://doi.org/10.1007/s11749-011-0268-9.