BIOINFORMATICS APPLICATIONS NOTE Vol. 27 no. 17 2011, pages 2463–2464
doi:10.1093/bioinformatics/btr406

Data and text mining

Advance Access publication July 6, 2011

APCluster: an R package for affinity propagation clustering

Ulrich Bodenhofer*, Andreas Kothmeier and Sepp Hochreiter

Institute of Bioinformatics, Johannes Kepler University, Linz, Austria

Associate Editor: Jonathan Wren

ABSTRACT

Summary: Affinity propagation (AP) clustering has recently gained increasing popularity in bioinformatics. AP clustering has the advantage that it allows for determining typical cluster members, the so-called exemplars. We provide an R implementation of this promising new clustering technique to account for the ubiquity of R in bioinformatics. This article introduces the package and presents an application from structural biology.

Availability: The R package `apcluster` is available via CRAN—The Comprehensive R Archive Network:
http://cran.r-project.org/web/packages/apcluster

Contact: apcluster@bioinf.jku.at; bodenhofer@bioinf.jku.at

Received on April 21, 2011; revised on June 10, 2011; accepted on July 2, 2011

1 INTRODUCTION

Affinity propagation (AP) is a relatively new clustering algorithm that has been introduced by Frey and Dueck (2007). AP clustering determines a so-called *exemplar* for each cluster, that is, a sample that is most representative for this cluster. Like agglomerative clustering, AP has the advantage that it works for any meaningful measure of similarity between data samples. Unlike most prototype-based clustering algorithms (like, e.g. k-means), AP does not require a vector space structure and the exemplars are chosen among the observed data samples and not computed as hypothetical averages of cluster samples. These characteristics make AP clustering particularly suitable for applications in bioinformatics: (i) many similarity measures used in bioinformatics cannot be linked to explicit vectorial descriptions (e.g. sequence or structure alignment scores); (ii) the opportunity to identify a small set of exemplars provides new potentials for exploratory analysis of biological data.

AP clustering has been used successfully for clustering microarray/gene expression data (Frey and Dueck, 2007; Kiddle *et al.*, 2010; Leone *et al.*, 2007; Liu *et al.*, 2010; Tang *et al.*, 2010), in structural biology (North *et al.*, 2011; Pandit and Skolnick, 2010; Wang *et al.*, 2010), biological network analysis (Pavlopoulos *et al.*, 2008; Vasblom and Wodak, 2009; Woźniak *et al.*, 2010) and sequence analysis (Yang *et al.*, 2010).

To date, no R implementation has been available. In order to leverage affinity propagation for bioinformatics applications, we have implemented affinity propagation as an R package along with visualization tools for analyzing the results.

*To whom correspondence should be addressed.

2 PACKAGE DESCRIPTION

The most important function is `apcluster()`. In the simplest form, this function can be called with only one argument, a quadratic similarity matrix. The function returns an S4 object which contains the clustering result in a convenient format that facilitates further processing.

Number of clusters and choice of input preferences: AP determines a suitable number of clusters by itself, depending on a vector of *input preferences*. In terms of the choice of input preferences, our package mainly implements the strategies and functions of Frey and Dueck (2007): by default, `apcluster()` uses the median of similarities as input preference for all samples. Frey and Dueck (2007) further suggest to use the minimum of similarities as input preference if a smaller number of clusters is prefered. In order to accommodate this strategy too and to allow for a smooth transition between these two strategies, our implementation offers a new optional argument `q` whose value determines which quantile of the similarities should be used as input preference. For determining a meaningful range of input preferences, the function `preferenceRange()` is available. If a fixed number of clusters is desired, our package further provides the function `apclusterK()`, which uses a search algorithm that adjusts the input preference to achieve the desired number of clusters.

Exemplar-based agglomerative clustering: the package also provides a new agglomerative clustering algorithm `aggExCluster()` that is geared towards the identification of meaningful exemplars. Most importantly, this function can be used to create a hierarchy of clusters from an AP result.

Visualization tools: in order to provide the user with tools for analyzing clustering results, the package offers various functions for plotting clusters (2D data only), heatmaps and dendrograms.

Similarity measures: the package further features several functions for computing similarities for vectorial data, such as, negative distances, RBF and Laplace kernels, etc.

Performance issues: the code made available by Frey and Dueck (2007) is mainly based on loops over rows and/or columns. We have re-engineered the function `apcluster()` to use matrix operations only. This led to a 15–20% reduction of runtimes, at the cost of additional memory consumption. If the user's R system is configured to distribute matrix operations over several cores/processors, our code can make use of this parallelization instantly without any interaction or configuration from the user's side. For large datasets and memory-tight systems, a memory-efficient implementation *à la* Frey and Dueck is still available as the function `apclusterLM()`.

3 EXAMPLE: CLUSTERING COILED COILS

We consider the exploratory analysis of coiled coil sequences as a brief example. Coiled coils are an important structural motif in which two or more α-helices are coiled together. Approximately 6% of the proteins in the Protein Data Bank (PDB) contain a coiled coil motif, many of which fulfill important biological functions, e.g. gene transcription or viral membrane fusion. Dimers (two helices) and trimers (three helices) are the most important types and together

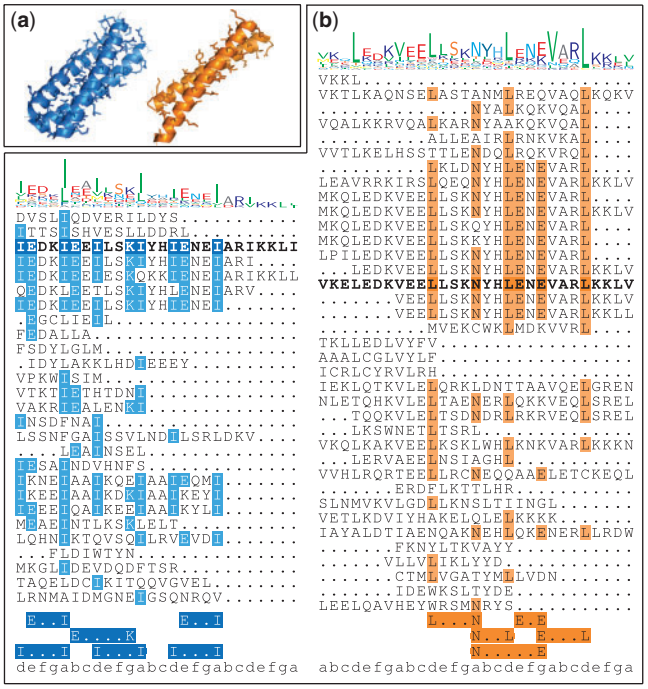

Fig. 1. (**a**) Structures of a typical trimer (left) and typical dimer (right). (**b**) Two clusters of coiled coil sequences determined by affinity propagation: multiple alignments of trimer cluster no. 3 (left) and dimer cluster no. 7 (right) with exemplars highlighted in boldface. Occurrences of top trimer patterns and top dimer patterns are indicated below the alignments in blue and orange, respectively.

account for ∼95% of known coiled coils; see Figure 1a for typical 3D structures of dimeric and trimeric coiled coils. Coiled coil segments are usually made up of repeats of blocks of seven amino acids, the so-called heptad repeats. Those seven residues are usually annotated with letters *a*–*g*, where positions *a* and *d* are often occupied by hydrophobic residues and positions *e* and *g* are often occupied by charged residues (Mason and Arndt, 2004).

Mahrenholz *et al.* (2011) have studied which sequence characteristics determine whether a given coiled coil segment tends to form a dimer or trimer. Mahrenholz *et al.* (2011) found out that pair interactions of amino acids are the key to understanding coiled coil oligomerization and, consequently, propose a custom string kernel based on pair interactions, the *coiled coil kernel*, upon which they base their prediction tool *PrOCoil*.

The coiled coil kernel is a similarity measure that is tailored to coiled coil sequences and has proven suitable for predicting oligomerization with high accuracy. Therefore, the coiled coil kernel is an ideal choice for clustering this type of sequences, too. We used the coiled coil dataset of Mahrenholz *et al.* (2011) consisting of a selection of 477 confirmed coiled coils (385 dimers and 92 trimers) from the PDB and proceeded as follows to identify most typical coiled coil sequences: adopting the coiled coil kernel with the same settings as in (Mahrenholz *et al.*, 2011) as sequence similarity measure, we first used `preferenceRange()` to determine reasonable input preferences. We decided to use a default input preference of −1 for all further studies. We then used `apcluster()` to cluster dimeric and trimeric sequences separately. This resulted in 13 clusters of dimeric sequences and 4 clusters of trimeric sequences. As examples, let us consider trimer cluster no. 3 (27 sequences) and dimer cluster no. 7 (37 sequences) in the following. Figure 1 shows multiple alignments of both clusters (insignificant positions have been omitted). The exemplars of both clusters are highlighted in boldface. It is quite obvious that the two exemplars are exactly those sequences that have the highest similarity to all other sequences in the cluster. This is also clearly visible from the excellent matching of the exemplars with the sequence logos displayed on top of the alignments. Below the alignments, we displayed the heptad annotation and which of the most indicative oligomerization patterns according to Mahrenholz *et al.* (2011) match the majority of sequences in the cluster. We see that all top five dimer patterns E...L*g*, L...N*d*, N..L*a*, E.E*e* and N.....E*e* are well represented in dimer cluster no. 7 and its exemplar sequence. Of the top five trimer patterns (Mahrenholz *et al.*, 2011), three occur prominently in trimer cluster no. 3: E..I*e*, E....K*b* and I...I*d*, two of which occur twice or more often in the cluster and in the corresponding exemplar.

This example illustrates that AP successfully identifies suitable exemplars for clusters of sequences. We are convinced that using AP in such a way has great potential for exploratory sequence analysis. AP provides the user with a small set of exemplars that can be studied in more detail, while the user can rely on the fact that biological knowledge obtained from investigating these exemplars is largely valid for all sequences in the dataset considered.

REFERENCES

Frey,B.J. and Dueck,D. (2007) Clustering by passing messages between data points. *Science*, **315**, 972–976.

Kiddle,S.J. *et al.* (2010) Temporal clustering by affinity propagation reveals transcriptional modules in *Arabidopsis thaliana*. *Bioinformatics*, **26**, 355–362.

Leone,M. *et al.* (2007) Clustering by soft-constraint affinity propagation: applications to gene-expression data. *Bioinformatics*, **23**, 2708–2715.

Liu,H. *et al.* (2010) Detecting microarray data supported microRNA-mRNA interactions. *Int. J. Data Min. Bioinform.*, **4**, 639–655.

Mahrenholz,C.C. *et al.* (2011) Complex networks govern coiled coil oligomerization — predicting and profiling by means of a machine learning approach. *Mol. Cell. Proteomics*, **10**, M110.004994.

Mason,J.M. and Arndt,K.M. (2004) Coiled coil domains: stability, specificity, and biological implications. *ChemBioChem*, **5**, 170–176.

North,B. *et al.* (2011) A new clustering of antibody CDR loop conformations. *J. Mol. Biol.*, **406**, 228–256.

Pandit,S.B. and Skolnick,J. (2010) TASSER_low-zsc: an approach to improve structure prediction using low z-score-ranked templates. *Proteins*, **78**, 2769–2780.

Pavlopoulos,G.A. *et al.* (2008) Arena3D: visualization of biological networks in 3D. *BMC Syst. Biol.*, **2**, 104.

Tang,D. *et al.* (2010) A Poisson-based adaptive affinity propagation clustering for SAGE data. *Comput. Biol. Chem.*, **34**, 63–70.

Vasblom,J. and Wodak,S.J. (2009). Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC Bioinformatics*, **10**, 99.

Wang,C.W. *et al.* (2010) iPARTS: an improved tool of pairwise alignment of RNA tertiary structures. *Nucleic Acids Res.*, **38**, W340–W347.

Woźniak,M. *et al.* (2010) MODEVO: exploring modularity and evolution of protein interaction networks. *Bioinformatics*, **26**, 1790–1791.

Yang,F. *et al.* (2010) Using affinity propagation combined post-processing to cluster protein sequences. *Protein Pept. Lett.*, **17**, 681–689.