Affinity propagation (AP) is a relatively new clustering algorithm (Frey and Dueck, 2007), in structural biology (Wang et al., 2010; North et al., 2010; Leone et al., 2011; Pandit and Skolnick, 2010, 2011; Tang et al., 2010) and microarray/gene expression data (Frey and Dueck, 2007; Kiddle et al., 2010; Lin 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; Vashblom and Wodak, 2009; Wozniak 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.

---

**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.

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 preferred. 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.

**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...
based on pair interactions, the oligomerization and, consequently, propose a custom string kernel interactions of amino acids are the key to understanding coiled coil to determine reasonable input preferences. We decided to use a sequence similarity measure, we first used base their prediction tool consisting of a selection of 477 confirmed coiled coils (385 dimers too. We used the coiled coil dataset of Mahrenholz kernel is an ideal choice for clustering this type of sequences, to coiled coil sequences and has proven suitable for predicting with letters so-called heptad repeats. Those seven residues are usually annotated are usually made up of repeats of blocks of seven amino acids, the structures of dimeric and trimeric coiled coils. Coiled coil segments hydrophobic residues and positions 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, L...N, N...e, E...E and N...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...e, E...K0 and L...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.

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.

REFERENCES


