Technical notes on the `anticlust` package

Martin Papenberg

2019-03-07

This document explains the technical and algorithmical background of the `R` package `anticlust`. **It is still a work in progress.** The following topics are covered:

1. A formalization of the anticlustering problem
2. A description of the objective functions used to measure anticluster similarity
3. A documentation of the algorithms used to optimize the objective funtions

1 Problem formalization

A set of n d-dimensional data points $X = \{x_i\}$ $(i \in \{1, ..., n\})$ has to be partitioned into K clusters $C = \{c_k, k = 1, ..., K\}$, satisfying the following restrictions:

$$\bigcup_{k=1}^{K} c_k = X \tag{1}$$

$$S_k \cap S_j = \emptyset,\ \forall k, j \in \{1, ..., K\},\ k \neq j \tag{2}$$

$$|c_k| = |c_j|,\ \forall k, j \in \{1, ..., K\} \tag{3}$$

Restriction (1) ensures that each element from the underlying set X is assigned to an anticluster; restriction (2) ensures that each element is assigned to only one anticluster; restriction (3) ensures that each anticluster contains the same number of elements. It follows that $|c_k| = \frac{n}{K}$ $\forall k \in \{1, ..., K\}$. Restriction (3) is currently implemented for all methods in the `anticlust` package, but it is not an obligatory restriction for anticlustering in general.The objective is to select a partitioning that maximizes the similarity of the K anticlusters.

2 Objective functions

This section presents definitions of optimal anticluster similarity as assumed in the `anticlust` package.

2.1 The variance objective

Späth (1986) and Valev (1998) independently proposed to maximize the variance criterion used in k-means clustering as the objective in anticlustering. The variance criterion is given by sum of the squared errors between cluster centers (μ_k) and individual data points (Jain, 2010):

$$\sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2 \tag{4}$$

The following plot graphically illustrates efforts to maximize and to minimize the variance criterion in a 2-dimensional feature space for three (anti)clusters, respectively. The partitions employ restrictions (1) - (3), including the restriction of equal (anti)cluster sizes. Optimizing the variance objective is a computationally difficult problem that is usually tackled using heuristic methods (Jain, 2010; Späth, 1986; Valev, 1998).

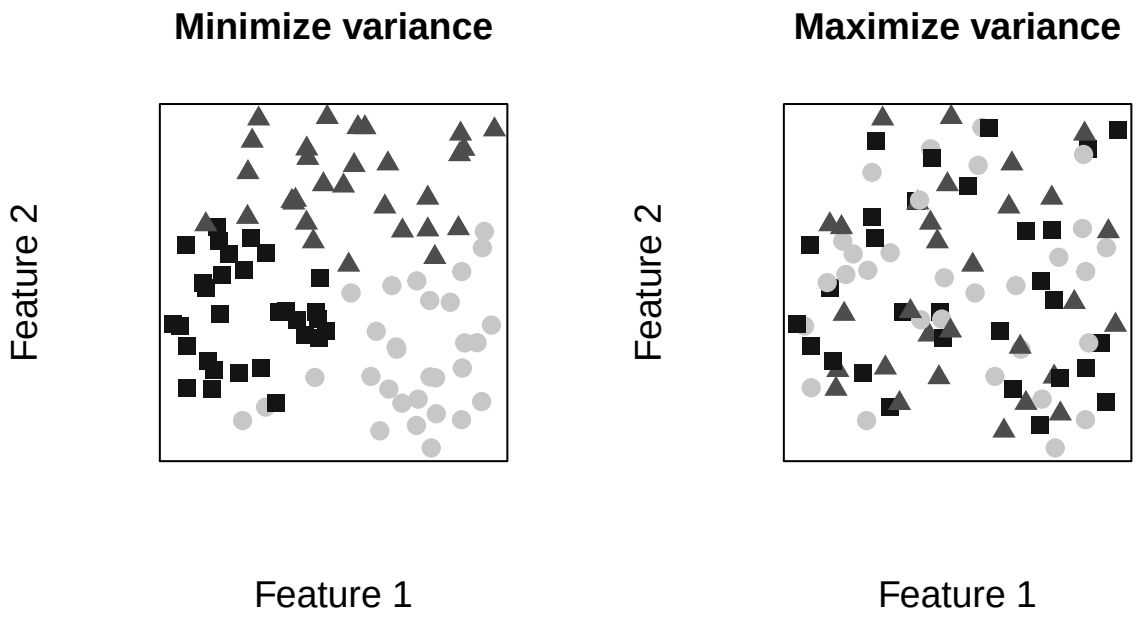

Figure 1: Attempts to minimize and maximize the variance objective, leading to clustering and anticlustering partitions, respectively. **TODO**: add cluster centers to illustrate the k-means variance objective; in anticlustering, the cluster centers are close to each other.

2.2 The distance objective

In addition to the variance criterion, the `anticlust` package introduces another clustering objective to the anticlustering application. The objective has been developed in the problem domain of cluster editing and is based on a measure of the pairwise dissimilarity between data points (Böcker & Baumbach, 2013; Rahmann et al., 2007).[1] In weighted cluster editing, the optimal objective is found when the sum of all pairwise dissimilarities within-clusters is minimized; for the anticlustering application, the sum of pairwise dissimilarities is maximized instead.

To formalize the cluster editing objective, we use variables x_{ij} to encode whether two data points x_i and x_j belong to the same anticluster c_k:

$$x_{ij} = \begin{cases} 1 & \text{if } x_i \in c_k \wedge x_j \in c_k \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Assume that d_{ij} represents a measure of the dissimilarity between two data points x_i and x_j, for example given as the euclidean distance. The cluster editing distance objective is then given as follows (Grötschel & Wakabayashi, 1989; Miyauchi & Sukegawa, 2015):

$$\sum_{1 \leq i < j \leq n} d_{ij}\, x_{ij} \tag{6}$$

I refer to this objective function as the "distance objective" as opposed to the "variance objective" in (4).

Maximizing the distance objective corresponds to minimizing the average linkage distance between anticlusters. In hierarchical cluster algorithms, the average linkage distance is a quantification of the similarity of two

[1] Cluster editing has also been studied under different names such as correlation clustering (Bansal, Blum, & Chawla, 2004), clique partition problem (Grötschel & Wakabayashi, 1989), and transitivity clustering (Wittkop et al., 2010).

clusters (Bacon, 2001). To appreciate the correspondence of the distance objective and the average linkage method, consider the total sum of all pairwise dissimilarities. The total sum can be partitioned into the sum of distances within-clusters and the sum of distances between-clusters:

$$\sum_{1 \leq i < j \leq n} d_{ij} = \sum_{1 \leq i < j \leq n} d_{ij}\, x_{ij} + \sum_{1 \leq i < j \leq n} d_{ij}\, (1 - x_{ij}) \tag{7}$$

The total sum of distances is not influenced by the concrete partitioning x_{ij}; hence, the following optimizations lead to the same partitions, i.e., assignments of elements to anticlusters:

$$\text{Maximize} \sum_{1 \leq i < j \leq n} d_{ij}\, x_{ij}$$

$$\text{Minimize} \sum_{1 \leq i < j \leq n} d_{ij}\, (1 - x_{ij})$$

In the special case of $K = 2$ where we have two partitions A and B, the sum of the between-cluster distances can be formulated as follows:

$$\sum_{i \in A} \sum_{j \in B} d_{ij} \tag{8}$$

This formulation is very close to the average linkage objective that additioanlly incorporates the cardinalities of the sets A and B (Guha, Rastogi, & Shim, 1998):

$$\frac{1}{|A|\,|B|} \sum_{i \in A} \sum_{j \in B} d_{ij} \tag{9}$$

Given restriction (3) for the anticlustering problem, the partitions A and B are of equal size; therefore, (8) is minimized whenever (9) is minimized. Hence, the cluster editing objective is a generalization of the average linkage measure on more than two clusters: maximizing the distance criterion maximizes cluster similarity as measured by the average linkage method.

3 Algorithmic approaches

Finding optimal data partitions usually corresponds to NP-complete problems (Arabie & Hubert, 1996; Bansal et al., 2004; Jain, 2010). For NP-complete problems, it is often infeasible to find the optimal objective, especially when n is large. To find the optimal solution for moderately large instances, `anticlust` employs integer linear programming. To process larger problem instances, `anticlust` uses heuristic methods based on repeated random sampling.

3.1 NP-completeness

In the following, I show that distance anticlustering is NP-complete. First, distance anticlustering is in NP because the distance objective can be computed in polynomial time for a given partitioning; the summation of all distance values d_{ij} is in $O(n^2)$.

Second, I show that if an efficient algorithm exists to solve distance anticlustering in polynomial time, it is also possible solve the NP-complete balanced number partitioning problem in polynomial time (Mertens, 2001). In the number partitioning problem, we have a list of positive integers $a_1, a_2, ..., a_n$ and try to find a subset $A \subset \{1, ..., n\}$ that minimizes the partition difference

$$E(A) = \left| \sum_{i \in A} a_i - \sum_{j \notin A} a_j \right| \tag{10}$$

In the balanced version of number partitioning, we may impose the restriction of $|A| = \frac{n}{2}$ – assuming that n is even – corresponding to restriction (3) of equal cluster sizes in anticlustering (Mertens, 2001).

To convert the number partitioning formulation into a formulation of distance anticlustering, we define d_{ij} as the absolute difference representing the dissimilarity of two numbers:

$$d_{ij} := |a_i - a_j| \tag{11}$$

We thus obtain

$$E(A) = \sum_{i \in A} \sum_{j \notin A} d_{ij} \tag{12}$$

Using variables $x_{ij} \in \{0, 1\}$ to represent whether two numbers are both either in A or not, i.e.,

$$x_{ij} = \begin{cases} 1 & \text{if } (x_i \in A \wedge x_j \in A) \vee (x_i \notin A \wedge x_j \notin A) \\ 0 & \text{otherwise} \end{cases}$$

we obtain $E(A)$ as the distance anticlustering objective:

$$E(A) = \sum_{1 \leq i < j \leq n} d_{ij}\, x_{ij} \tag{13}$$

Hence, balanced number partitioning is a special case of distance anticlustering where

a) $K = 2$
b) each element is described by exactly one integer
c) d_{ij} is the absolute difference

If a polynomial-time algorithm exists that solves distance anticlustering, we can therefore solve the NP-complete balanced number partitioning in polynomial time. Hence, distance anticlustering is NP-complete.

3.2 Integer linear programming

Despite the NP-complete nature of cluster editing, integer linear programming has been successfully used to find optimal solutions even for relatively large problem instances (Böcker, Briesemeister, & Klau, 2011; L. H. N. Lorena, Quiles, Carvalho, & Lorena, 2018). Integer linear programming identifies the values of *decision variables* that optimize a linear objective function while employing constraints on the decision variables that are implemented as mathematical inequalities. In the case of anticlustering, integer linear programming can

be used to maximize the distance objective in (6) while inequalities ensure that the anticlustering conditions in (1) - (3) are met.

For the integer linear programming formulation of distance anticlustering, a problem instance is represented as an undirected complete graph $G = (V, E)$ (cf. Schaeffer, 2007). Each vertex $v \in V(1, ..., n)$ represents an element from the input data that has to be assigned to an anticluster. Edges are unordered pairs $\{i, j\} \in E$; the short form ij will be used to refer to edges. A cost function $w : E \to \mathbb{R}$ assigns a weight to each edge. In distance anticlustering, an edge weight is given by the distance between the two elements that are connected by the edge, i.e., $w(ij) = d_{ij}$.

The integer linear program returns a subgraph $G' = (V, E')$. The decision variables x_{ij} encode wether two vertices i and j are connected by an edge in the output graph G':

$$x_{ij} = \begin{cases} 1 & \text{if } ij \in E' \\ 0 & \text{otherwise} \end{cases}$$

If two vertices are connected by an edge in G', the integer linear program ensures that they are also part of a *clique*, that is, a subgraph whose vertices are all connected. Cliques in G' represent anticlusters – hence, the following two conditions are equivalent in the integer linear programming formulation of distance anticlustering:

1. $ij \in E'$
2. Vertices i and j belong to the same anticluster.

The `anticlust` package employs an integer linear program on the basis of formulations proposed by Grötschel and Wakabayashi (1989) to solve cluster editing:

$$\text{Maximize} \sum_{1 \leq i < j \leq n} w(ij)\, x_{ij} \tag{14}$$

$$-x_{ij} + x_{ik} + x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq n, \tag{15}$$

$$x_{ij} - x_{ik} + x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq n, \tag{16}$$

$$x_{ij} + x_{ik} - x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq n, \tag{17}$$

$$\sum_{1 \leq i < j \leq n} x_{ij} + \sum_{1 \leq k < i \leq n} x_{ki} = \frac{n}{K} - 1, \qquad \forall i \in \{1, ..., n\} \tag{18}$$

$$x_{ij} \in \{0, 1\},\ \forall\, 1 \leq i < j \leq n \tag{19}$$

The inequalities (15) - (17) are called triangular constraints; they ensure that the output graph is a union of disjoint cliques (Grötschel & Wakabayashi, 1989). Constraint (18) ensures that K cliques are returned, each of cardinality $\frac{n}{K}$. Constraint (19) ensures that the decision variables x_{ij} are binary. Cliques are created under the objective to maximize the sum of the edge weights within cliques.

In the `anticlust` package, one of the commercial solvers `gurobi` or `CPLEX`, or the open source GNU linear programming kit can be used to solve the ILP in (14) - (19) optimally.

3.2.1 Preprocessing

To expand its applicability to larger problem sizes, additional constraints were added to the integer linear program through a redefinition of the cost function w. The redefinition resulted from two considerations:

1. The run time of the weighted cluster editing ILP is improved if there is an uneven distribution of edge weights (Böcker & Baumbach, 2013; Böcker et al., 2011)
2. It is possible to prevent very similar elements from joining the same anticluster without impairing the quality of the solution strongly

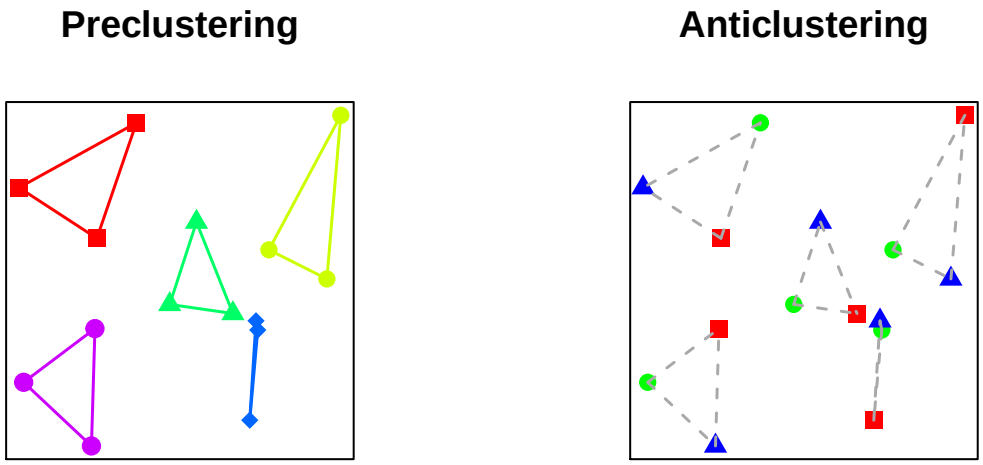

Figure 2: The left-hand plot shows the preclusters each consisting of three elements. The right-hand plot shows the optimal anticlustering under the preclustering restrictions. Elements that are part of the same precluster cannot be part of the same anticluster

In the cluster editing integer linear programming framework, setting $w(ij) = -\infty$ prevents two vertices i and j from joining the same clique in the output graph (Böcker et al., 2011). This recalculation of some edge weights induces a strong enevenness among edge weights, increasing the problem sizes the integer linear program can be applied to. If only very similar elements are prevented from joining the same anticluster, the quality of the solution is not impaired by much because the anticlustering objective is based on the idea that similar elements should be in different anticlusters.

Therefore, the `anticlust` package realizes a preprocessing step in which very similar elements are grouped into *preclusters*; elements of a precluster are prevented from joining the same anticluster thereafter. The preclustering step is formalized by the ILP defined in (20) - (25).

$$\text{Minimize} \sum_{1 \leq i < j \leq n} w(ij)\, x_{ij} \tag{20}$$

$$-x_{ij} + x_{ik} + x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq n, \tag{21}$$

$$x_{ij} - x_{ik} + x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq n, \tag{22}$$

$$x_{ij} + x_{ik} - x_{jk} \leq 1, \qquad \forall\, 1 \leq i < j < k \leq n, \tag{23}$$

$$\sum_{1 \leq i < j \leq n} x_{ij} + \sum_{1 \leq k < i \leq n} x_{ki} = K - 1, \qquad \forall i \in \{1, ..., n\} \tag{24}$$

$$x_{ij} \in \{0, 1\},\ \forall\, 1 \leq i < j \leq n \tag{25}$$

By minimizing the distance objective, the preclustering step solves weighted cluster editing under the restriction of a cluster size K, resulting in $\frac{n}{K}$ clusters.[2] The number of elements per precluster corresponds to the number of anticlusters. For $K = 2$, the preclustering corresponds to the minimum weight perfect matching problem (Gerards, 1995). The left-hand plot in Figure 2 illustrates the preprocessing step for $K = 3$ and $n = 15$.

In a second step, the cost function is redefined to prevent elements from the same preclustering from joining the same anticluster:

[2]This is a feasible preprocessing step because minimizing the distance objective works faster than maximization for the same problem size. That is, cluster editing is solved more efficiently using ILP than anticlustering.

$$w(ij) = \begin{cases} -\infty & \text{if } x_{ij} = 1 \\ d_{ij} & \text{if } x_{ij} = 0 \end{cases}$$

Using the redefined edge weights, we solve distance anticlustering using the ILP defined in (14) - (19). The results of an anticlustering employing the preclustering restrictions is shown in the right-hand plot of Figure 2. Because the optimal solution sometimes sometimes requires to join elements that are part of the same precluster, the preprocessing sometimes precludes an optimal solution.

3.3 Heuristic anticlustering

To solve larger problem instances that cannot be processed using integer linear programming, two heuristic methods based on random search are employed in the `anticlust` package: random sampling and simulated annealing. Both methods may employ a preclustering step that usually improves the quality of the output.

3.3.1 Random sampling

A simple random sampling approach may be used to optimize the variance or distance objective. That is, across a user-specified number of runs, each element is randomly first assigned to an anticluster, then the objective value is computed and in the end, the best assignment is returned as the output. When a preclustering is employed, the random assignment is conducted under the restriction that preclustered elements cannot be part of the same anticluster.

3.3.2 Simulated annealing

The `anticlust` package also employs a simulated annealing approach to optimize the variance or distance objective. If no preclustering is employed, anticlusters are initialized randomly. The exchange step is realized by changing the anticluster affiliations of two random elements – this step is repeated if two elements of the same anticluster are swapped.

If preclustering is employed, the initialization of cluster assignments incorporates the condition that elements from the same precluster are not part of the same anticluster. The exchange step is realized as follows: a precluster is selected at random. From this precluster, two elements exchange their anticluster affiliation.

The simulated annealing uses the function `optim` from base `R`; the following parameters are used:

- Temperature: 2000
- tmax: 20

3.3.3 Heuristic preclustering

A preclustering step may be used to improve the quality of the random search procedures. To this end, a heuristic clustering algorithm is employed. The algorithm is based on k-means clustering and enforces equal cluster sizes. First, $\frac{n}{K}$ cluster centers are initialed using k-means; then, elements are sequentially assigned to the nearest cluster centers while ensuring that each of the clusters is filled with K elements. The following pseudo code formalizes the heuristic:

```
E = list of n elements
C = list of K/n cluster centers, initialized using k-means
// Iterate over the number of elements per cluster
Repeat K times {
  // Iterate over clusters in random order
  S <- random sequence of C
```

```
  for (j in S) {
    1. e = element that is closest to cluster center j
    2. Add e to cluster of j
    3. Remove e from E
  }
}
```

4 References

Arabie, P., & Hubert, L. J. (1996). An overview of combinatorial data analysis. In P. Arabie, L. J. Hubert, & G. De Soet (Eds.), *Clustering and classification* (pp. 5–63).

Bacon, D. R. (2001). An evaluation of cluster analytic approaches to initial model specification. *Structural Equation Modeling*, *8*(3), 397–429.

Bansal, N., Blum, A., & Chawla, S. (2004). Correlation clustering. *Machine Learning*, *56*(1-3), 89–113.

Böcker, S., & Baumbach, J. (2013). Cluster editing. In *Conference on Computability in Europe* (pp. 33–44). Springer.

Böcker, S., Briesemeister, S., & Klau, G. W. (2011). Exact algorithms for cluster editing: Evaluation and experiments. *Algorithmica*, *60*(2), 316–334.

Gerards, A. M. H. (1995). Matching. In M. O. Ball, T. L. Magnanti, C. L. Monma, & G. L. Nemhauser (Eds.), *Network models* (pp. 135–224).

Grötschel, M., & Wakabayashi, Y. (1989). A cutting plane algorithm for a clustering problem. *Mathematical Programming*, *45*(1-3), 59–96.

Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An efficient clustering algorithm for large databases. In *ACM sigmod record* (Vol. 27, pp. 73–84). ACM.

Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, *31*(8), 651–666.

Lorena, L. H. N., Quiles, M. G., Carvalho, A. C. P. de L. F. de, & Lorena, L. A. N. (2018). Preprocessing technique for cluster editing via integer linear programming. In D.-S. Huang, V. Bevilacqua, P. Premaratne, & P. Gupta (Eds.), *Intelligent computing theories and application* (pp. 287–297). Cham: Springer International Publishing.

Mertens, S. (2001). A physicist's approach to number partitioning. *Theoretical Computer Science*, *265*(1-2), 79–108.

Miyauchi, A., & Sukegawa, N. (2015). Redundant constraints in the standard formulation for the clique partitioning problem. *Optimization Letters*, *9*(1), 199–207.

Rahmann, S., Wittkop, T., Baumbach, J., Martin, M., Truss, A., & Böcker, S. (2007). Exact and heuristic algorithms for weighted cluster editing. In *Computational systems bioinformatics: (Volume 6)* (pp. 391–401). World Scientific.

Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, *1*(1), 27–64.

Späth, H. (1986). Anticlustering: Maximizing the variance criterion. *Control and Cybernetics*, *15*(2), 213–218.

Valev, V. (1998). Set partition principles revisited. In *Joint IAPR international workshops on statistical techniques in pattern recognition (SPR) and structural and syntactic pattern recognition (SSPR)* (pp. 875–881). Springer.

Wittkop, T., Emig, D., Lange, S., Rahmann, S., Albrecht, M., Morris, J. H., ... Baumbach, J. (2010). Partitioning biological data with transitivity clustering. *Nature Methods*, *7*(6), 419–420.