1. Proof A

This section provides the proof that single-linkage clustering (SL) with correlation coefficients as similarity measure produces the same results as our clustering algorithm with the correlation edge measure (CEM). In the following, $X = \{x_1, \ldots, x_n\}$ denotes the set of points getting clustered and ρ_{ij} is the correlation between two points $x_i, x_j \in X$.

In every iteration step, SL merges the two clusters A and B with the highest similarity, which is defined as the highest similarity between two points $x_i \in A, x_j \in B$. Therefore, it finds the pair of yet unmerged points that have the highest correlation. CEM does exactly the same by first sorting all edges by similarity and then merging them successively. The only difference is that instead of considering all possible point pairs, CEM only merges along edges of the beta-skeleton with $1 \leq \beta \leq 2$. Since graphs resulting from smaller β are always subgraphs of ones with higher β, we only have to prove similarity for the neighborhood graph with the fewest edges, i.e., the relative neighborhood graph (RNG) with $\beta = 2$.

Both clustering algorithms start with each point in its own separate cluster. Assume that up to a certain iteration, both clustering hierarchies are equivalent and $C(x) \subseteq X$ denotes the cluster containing point $x \in X$. Let $A \subset X$ and $B \subset X$ be two different clusters with points $x_i \in A$ and $x_j \in B$. To prove equality of the clusterings, let us compare the two following statements:

I the next two cluster getting merged by SL are A and B due to the highest similarity between x_i and x_j

II the next two clusters getting merged by CEM are A and B due to the edge between x_i and x_j corresponding to the highest correlation

I $\rightarrow$ II

Let us assume statement (I) is true. It follows that

$$\rho_{ij} = \max\{\rho_{ab} \mid C(x_a) \neq C(x_b)\}. \tag{9}$$

To show that these points have to be connected in the RNG, we have to look at the beta-lune. Two points are connected in the RNG if and only if their beta-lune does not contain a third point x_k. As of Formula 5, the test for containment in the hyperspherical lune reduces to

$$\begin{aligned} & (2-\beta)(1-\rho_{ki}) + \beta(\rho_{ij} - \rho_{kj}) < 0 \\ \wedge \quad & (2-\beta)(1-\rho_{kj}) + \beta(\rho_{ij} - \rho_{ki}) < 0. \end{aligned}$$

For $\beta = 2$, this simplifies to

$$\begin{aligned} & 2 \cdot (\rho_{ij} - \rho_{kj}) < 0 \quad \wedge \quad 2 \cdot (\rho_{ij} - \rho_{ki}) < 0 \\ \Leftrightarrow \quad & \rho_{ij} < \rho_{kj} \quad \wedge \quad \rho_{ij} < \rho_{ki}. \end{aligned}$$

If there would be a point x_k fulfilling this criterion, thus causing the edge between x_i and x_j to not be part of the neighborhood graph, either one of the two conditions would contradict Implication 9. There also can be no edge with higher correlation, as this would also violate Implication 9.

$\neg$ I $\rightarrow \neg$ II

To complete the proof, let us now assume that statement (I) is false. That means there has to be a pair of points x_k, x_l with $C(x_k) \neq C(x_l)$ and

$$\rho_{kl} = \max\{\rho_{ab} \mid C(x_a) \neq C(x_b)\} < \rho_{ij}.$$

With the same implications as above, there has to be an edge between x_k and x_l in the neighborhood graph, which causes statement (II) to be false as well.

This proves equality of the two clusterings. □

2. Agglomorative Correlation Clustering

Algorithm 1: Agglomorative Correlation Clustering

input : $G = (V, E)$ - Neighborhood Graph with vertices $V = \{v_1, \ldots, v_n\}$ and edges $E \subseteq V \times V$
measures - Array of edge measures
output: Dendrogram

```
// Initialize data structures
d ← Dendrogram with n nodes;
uf ← UnionFind of size n;
repr ← Array of size n;
for i ← 1 to n do
    repr[i] = i;

// Main loop
sort edges E by measures;
foreach edge (v1, v2) in E do
    r1 ← uf.root(v1);
    r2 ← uf.root(v2);
    if r1 ≠ r2 then
        newRoot ← uf.merge(r1,r2);
        newVertex ← d.addVertex();
        d.connect(newVertex,repr[r1]);
        d.connect(newVertex,repr[r2]);
        repr[newRoot] ← newVertex;
return d;
```