Institute of Bioinformatics
Johannes Kepler University Linz

# Introduction to `apcluster`

Ulrich Bodenhofer

*bodenhofer@bioinf.jku.at*

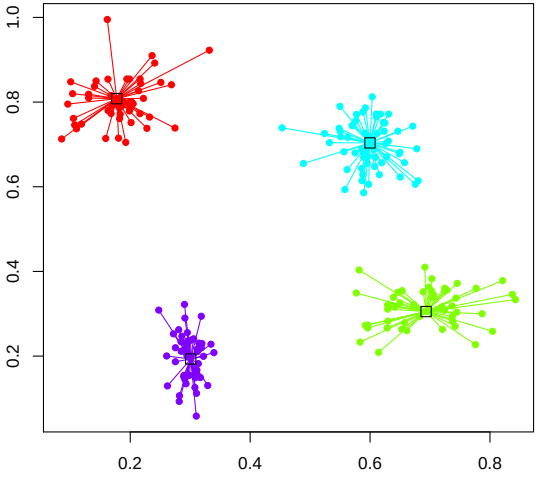
JKU
JOHANNES KEPLER
UNIVERSITY LINZ

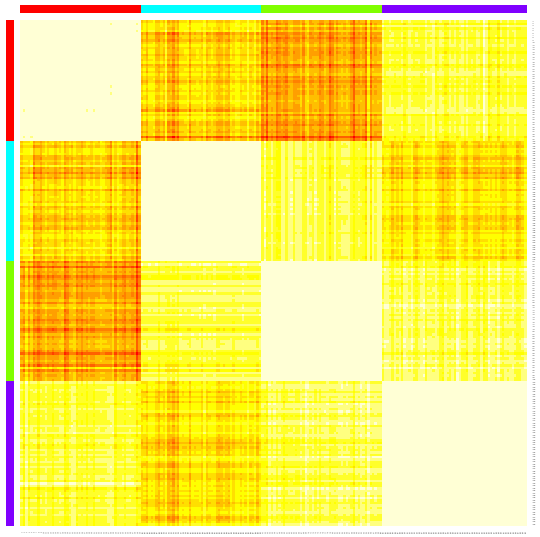
# Outline

1. Introduction to affinity propagation (AP) clustering
2. The `apcluster` package, its algorithms, and visualization tools
3. Live `apcluster` demonstration
4. Question and Answer period

# Affinity Propagation (AP) Clustering

- Affinity propagation (AP) is an emerging new clustering technique with increasing importance in many fields (in particular, bioinformatics).
- AP uses **pairwise similarities** as inputs and iteratively determines clusters along with samples that are representative for the clusters, so-called **exemplars**.
- AP is based on an iterative message passing scheme in which samples compete for becoming exemplars.



# Affinity Propagation: What's Special About It?

- Efficiently finds approximate exemplars (to find an optimal solution is actually NP-hard).
- Deterministic, initialization-independent algorithm with very simple update rules (can be viewed as max-sum algorithm in a factor graph).
- Published in Science:

  B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, **315**:972–976, 2007. DOI: 10.1126/science.1136800.

# Affinity Propagation: Message Matrices

**Responsibility matrix $R$:** $R(i,k)$ is sent from $i$ to $k$ and corresponds to the accumulated evidence for how well-suited $k$ is to serve as the exemplar for $i$ (taking into account other possible exemplars for $i$).

**Availability matrix $A$:** $A(i,k)$ is sent from $k$ to $i$ and corresponds to the accumulated evidence for how appropriate it is for $i$ to choose $k$ as its exemplar (taking into account other points that would choose $k$ as exemplar).

# Message Passing / Update Rules

Repeatedly perform the following updates ($A$ is initialized with zeros):

- For $i, k \in \{1, \ldots, l\}$:

$$R(i,k) \leftarrow S(i,k) - \max_{k' \neq k} \left( A(i,k') + S(i,k') \right)$$

- For $i, k \in \{1, \ldots, l\}$ s.t. $i \neq k$:

$$A(i,k) \leftarrow \min \left( 0, R(k,k) + \sum_{j \notin \{i,k\}} \max \left( 0, R(j,k) \right) \right)$$

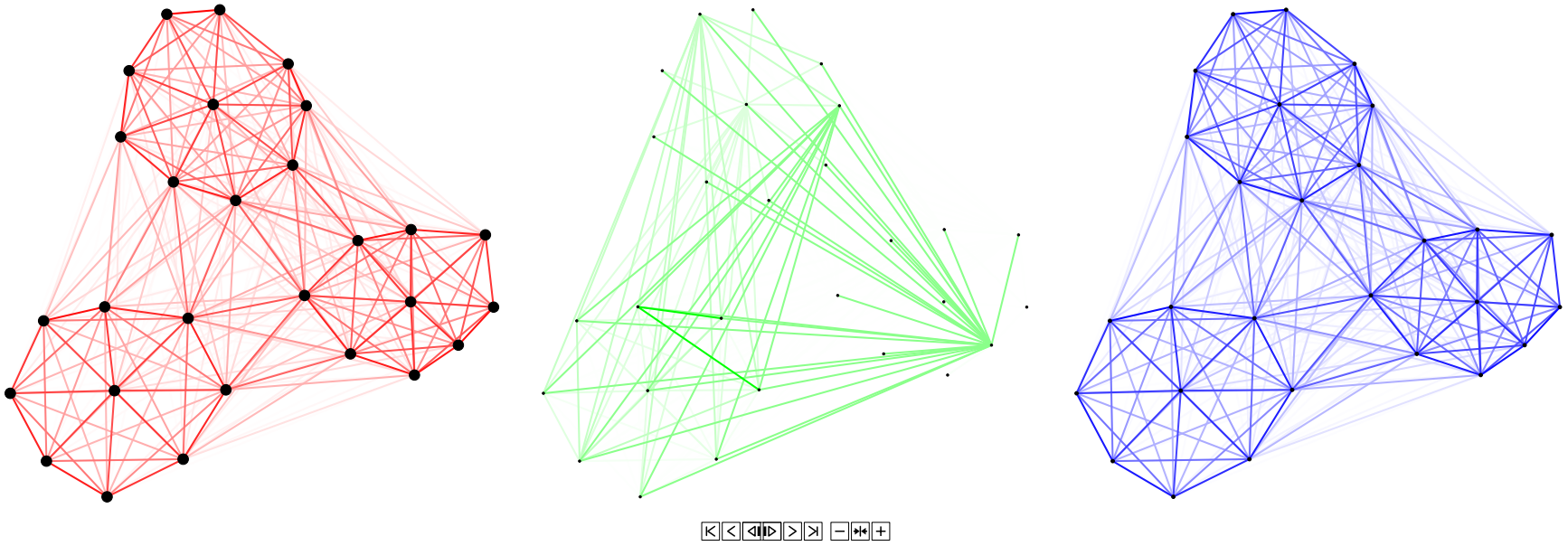
- For $k \in \{1, \ldots, l\}$:

$$A(k,k) \leftarrow \sum_{j \neq k} \max \left( 0, R(j,k) \right)$$

The "self-similarities" $S(k,k)$ are called *input preferences*. They determine the individual tendencies of samples to become exemplars.

# How Exemplars Emerge: An Example

$R$ $A$ $R + A$



# Function `apcluster()`

```
apcluster(s, x, p, q, ...)
```

`s` ... quadratic similarity matrix or similarity measure (function)

`x` ... data (vector or list; only if `s` was a function)

`p` ... input preference (per sample or one value for all)

`q` ... sets input preference to quantile of similarities ($\texttt{q} \in [0, 1]$)

The function returns an `APResult` object that contains the clustering result.

# Affinity Propagation: Pro's & Con's


+ Exemplars are real samples, no hypothetical averages.
+ Only pairwise similarities necessary; the similarity measures not even need to satisfy symmetry or the triangle inequality (applicable to all sorts of kernels and correlation measures).
+ Algorithm (almost) deterministic, not sensitive to initialization.
+ Number of clusters need not be pre-specified.
– Adjustment of input preference parameter can be tricky.
– Quadratic similarity matrices required: does not scale to large data sets.

# Adjustment of Input Preference / Number of Clusters

The `apcluster` package further offers:

- Function `preferenceRange(s)` computes input preference bounds for given similarity matrix `s`:
  - Lower bound: 1–2 clusters
  - Upper bound: as many clusters as samples
- Function `apclusterK()` performs an inverse search for an input preference that facilitates a desired number of clusters $K$.
- Function `aggExCluster()` performs agglomerative clustering on top of AP clustering and allows for specifying cut levels with a desired number of clusters.

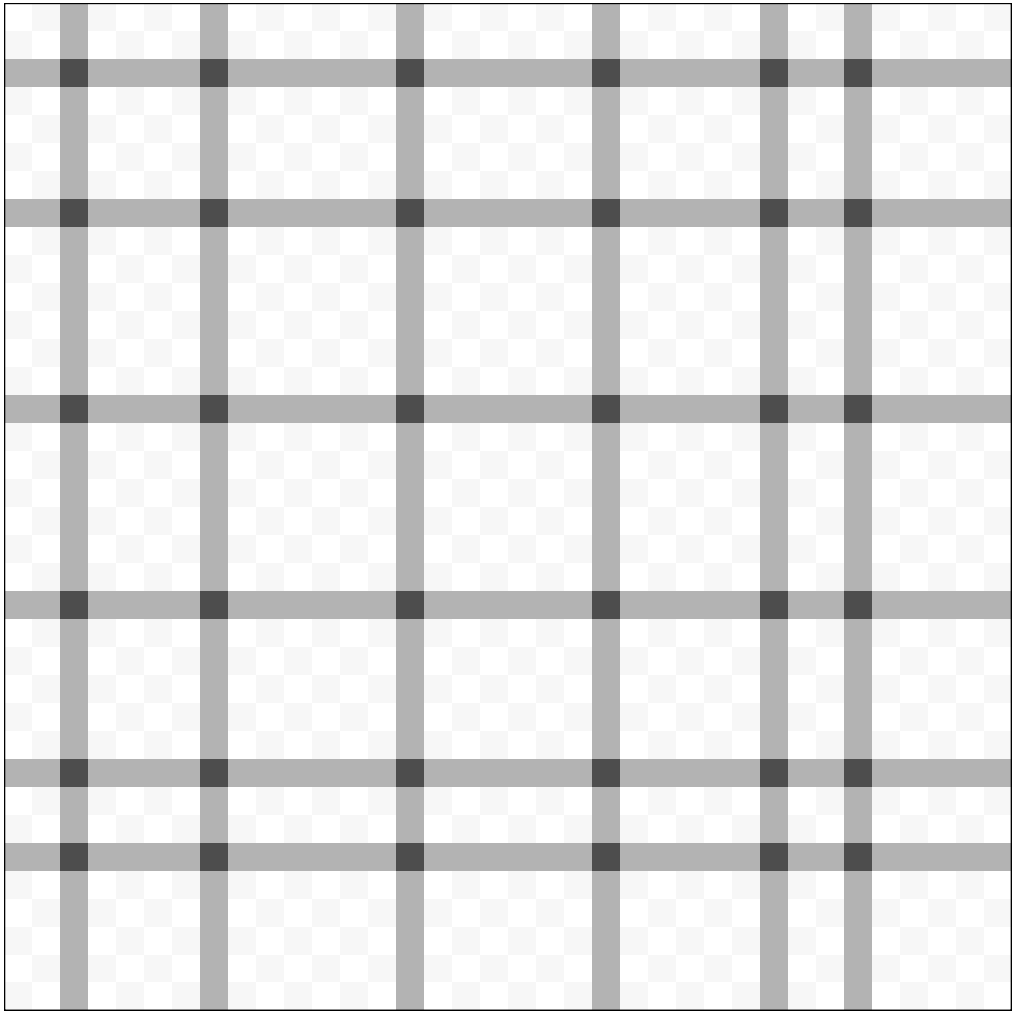
# So How to Scale AP to Large Datasets Then?

# So How to Scale AP to Large Datasets Then?

Sub-sampling?

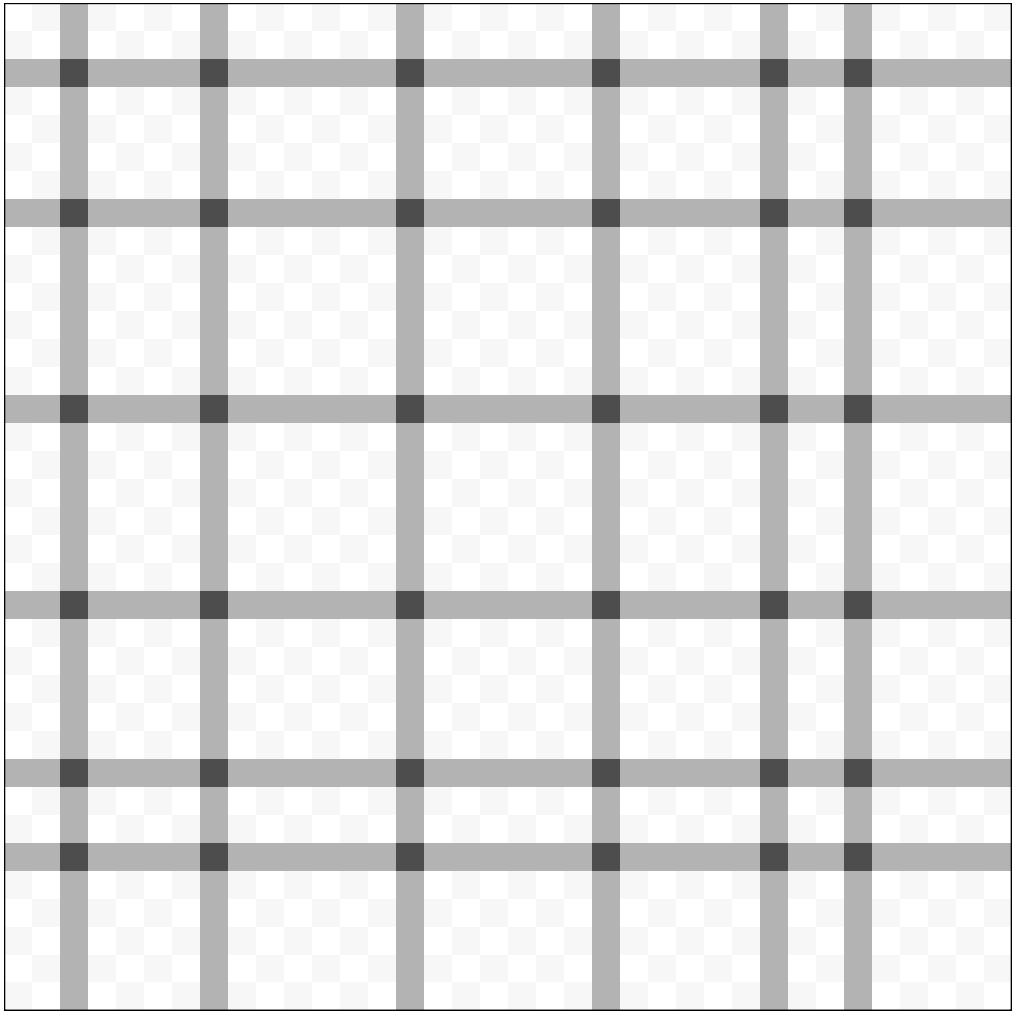
# So How to Scale AP to Large Datasets Then?

Sub-sampling?



# So How to Scale AP to Large Datasets Then?

Sub-sampling?



**Massive loss of information!**

# So How to Scale AP to Large Datasets Then?

**Reduce column objects . . .**

# So How to Scale AP to Large Datasets Then?

**Reduce column objects . . .**

# So How to Scale AP to Large Datasets Then?

**Reduce column objects . . .**

**but keep all row objects**

# Leveraged Affinity Propagation

- Start with a small, but reasonable, sub-sample of columns (potential exemplars).
- Iteratively repeat AP on such sub-samples, keeping the exemplars of the previous iteration. Messages are exchanged between the row objects (all) and the potential exemplars/column objects (sub-sample).
- No need to calculate the whole similarity matrix in advance. Instead, only the similarities with the sub-samples need to be computed.

# Function `apclusterL()`

```
apclusterL(s, x, frac, sweeps, p, q, ...)
```

`s` ... similarity measure (function)

`x` ... data (vector or list)

`frac` ... fraction of samples to be considered in each iteration

`sweeps` ... number of iterations

`p, q` ... analogous to `apcluster()`

The function returns an `APResult` object that contains the clustering result.

# Functions for Visualizing Results

- `plot(x)` with `x` being an `APResult` object: performance graphs for assessing convergence
- `plot(x)` with `x` being an `AggExResult` object: dendrogram of agglomerative clustering
- `plot(x, y)` with `x` being an `APResult` object and `y` being original data: plot data along with cluster structure)
- `heatmap(x)` with `x` being an `APResult` or `AggExResult` object: plot heatmap (dendrograms and color coding of clusters switchable)

# Similarity Measures

The `apcluster` package provides the following similarity measures:

`negDistMat()`: negative distances (resp. power thereof), interface analogous to `dist()`

`expSimMat()`: generalization of Gauss/Laplace similarity (RBF/Laplace kernel)

`linSimMat()`: Łukasiewicz similarity, interface analogous to `dist()`

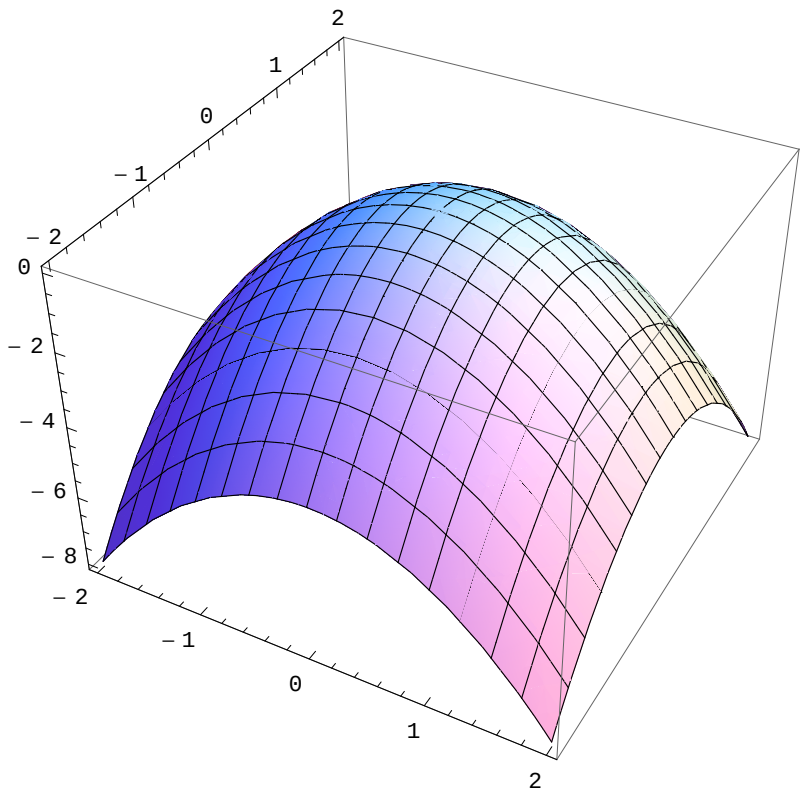
`corSimMat()`: correlation, interface analogous to `cor()`

All functions in the `apcluster` package can also be used with custom-made similarity measures.

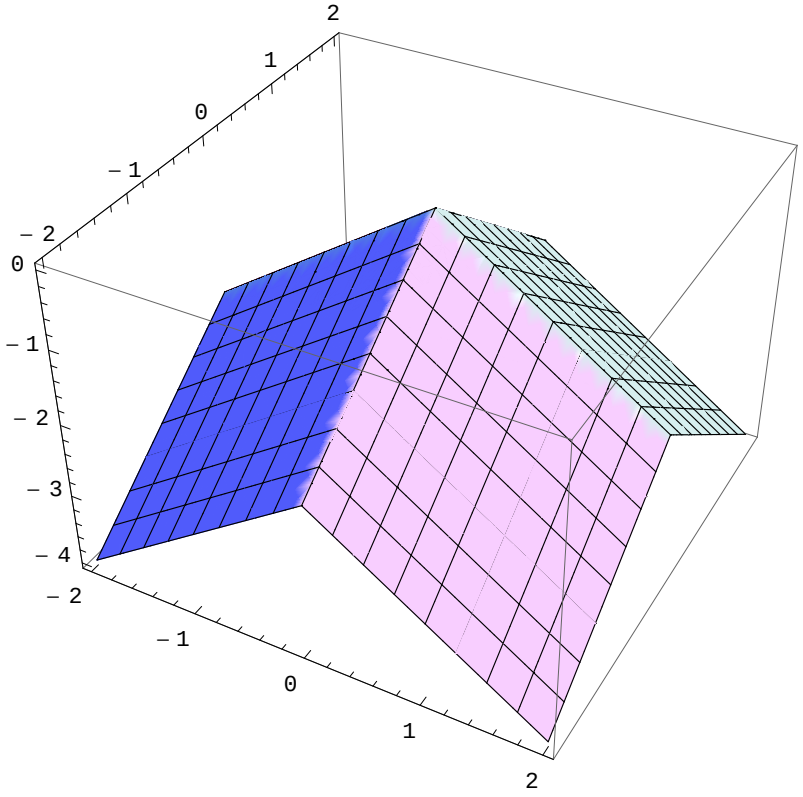
# Similarity Measures: Examples

## [similarity of $(x, y)$ and $(0, 0)$]

`negDistMat(r=2, method="euclidean")`



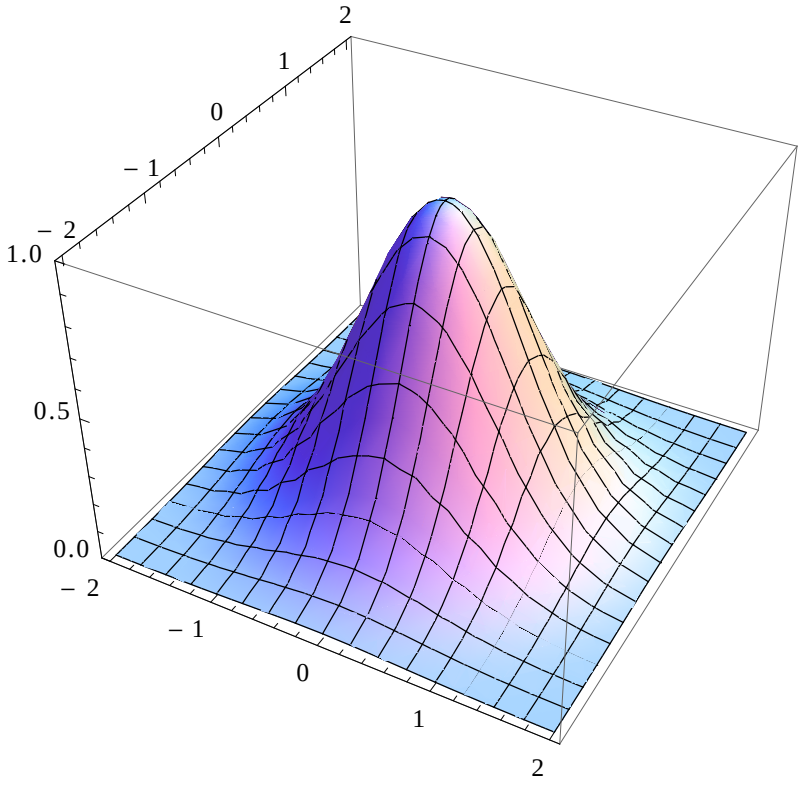

`negDistMat(r=1, method="manhattan")`



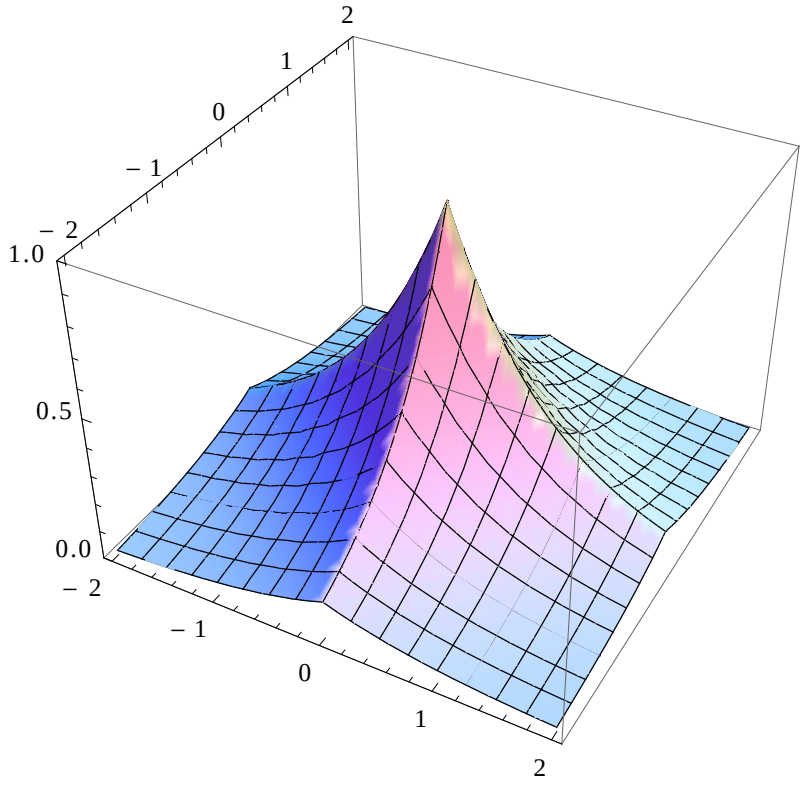
# Similarity Measures: Examples

## [similarity of $(x, y)$ and $(0, 0)$]

`expSimMat(r=2, w=1, method="euclidean")`



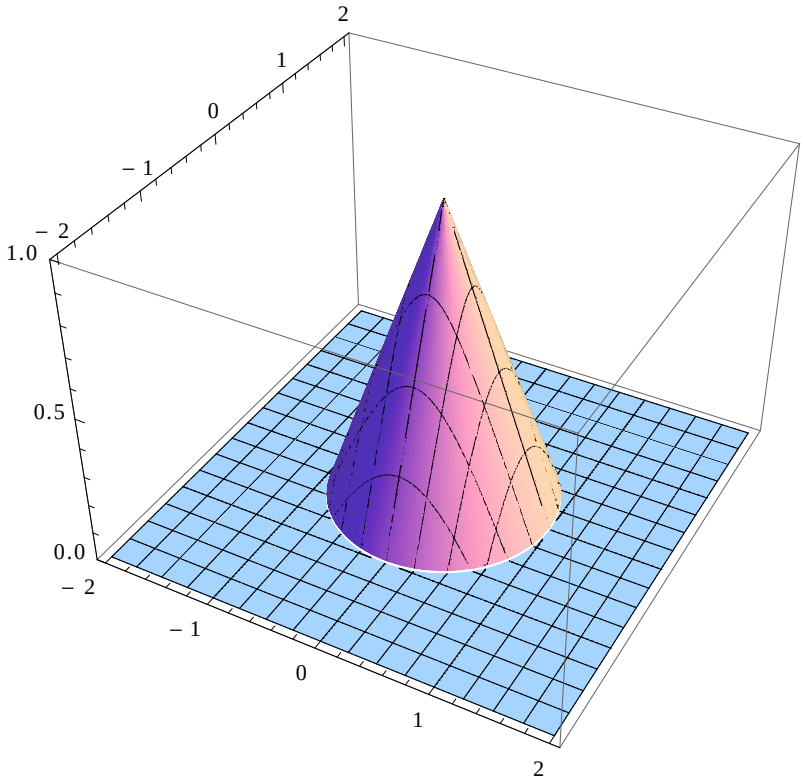

`expSimMat(r=1, w=1, method="manhattan")`



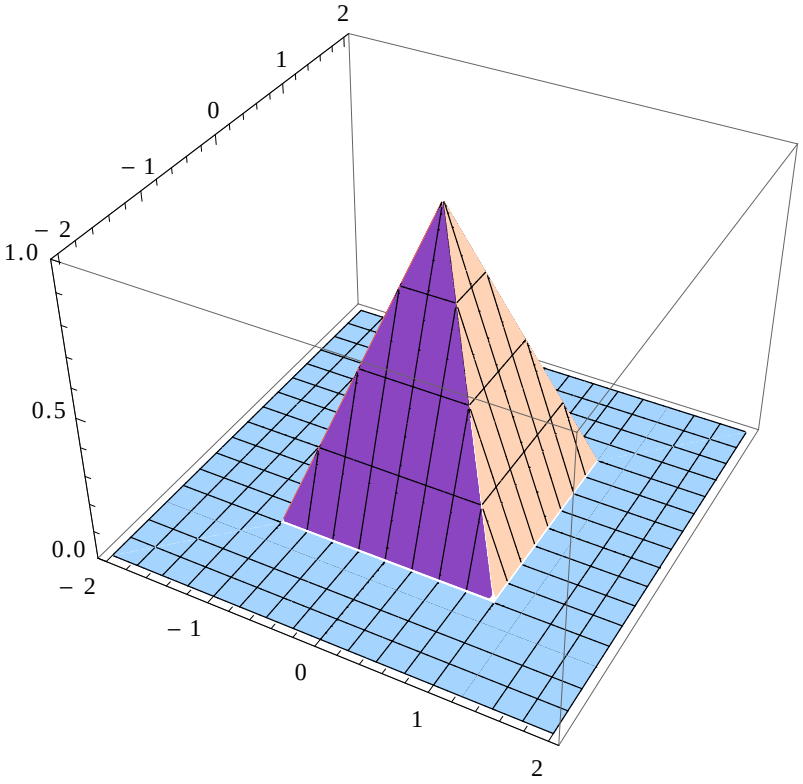
# Similarity Measures: Examples

## [similarity of $(x, y)$ and $(0, 0)$]

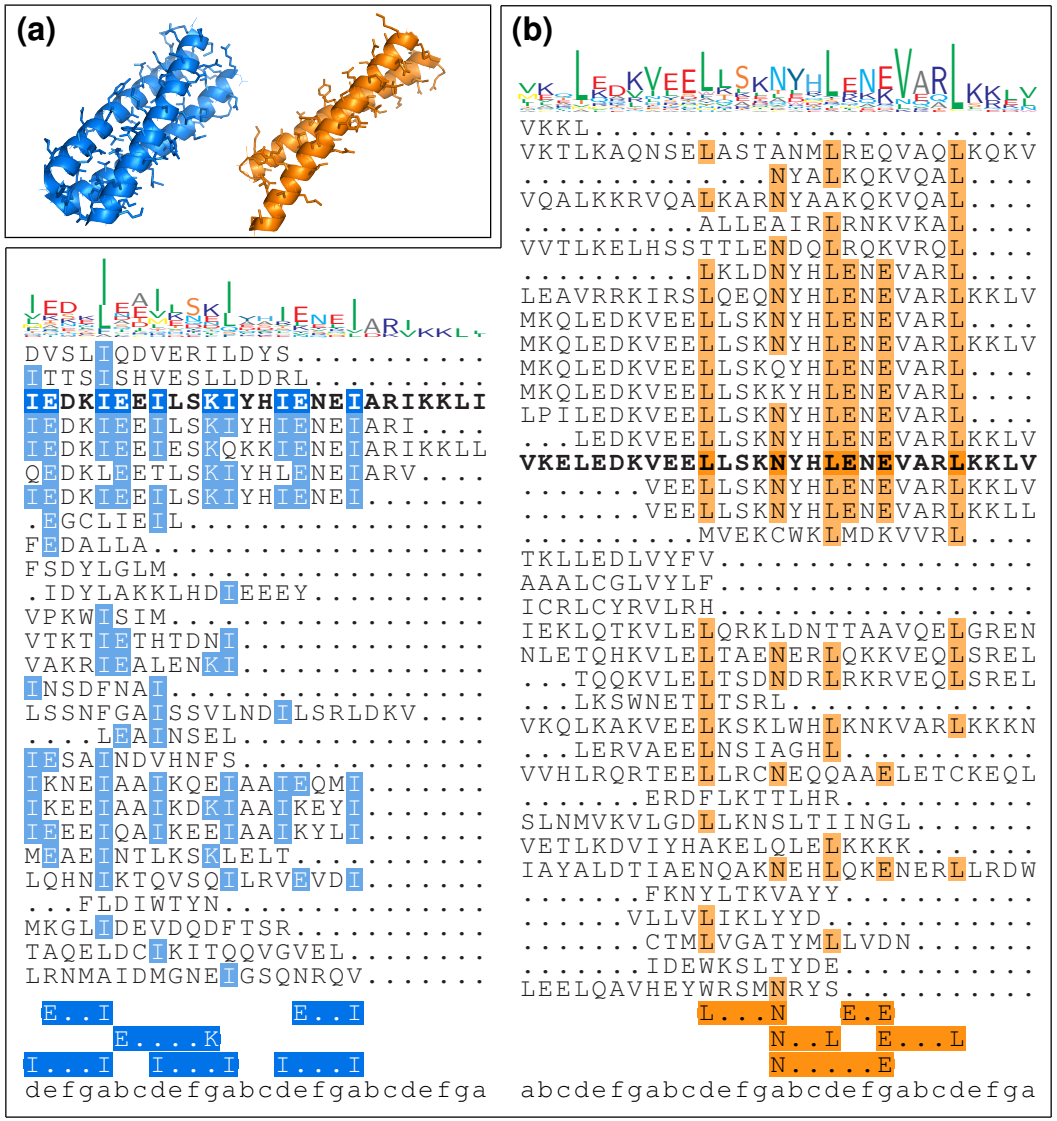
`linSimMat(w=1, method="euclidean")`



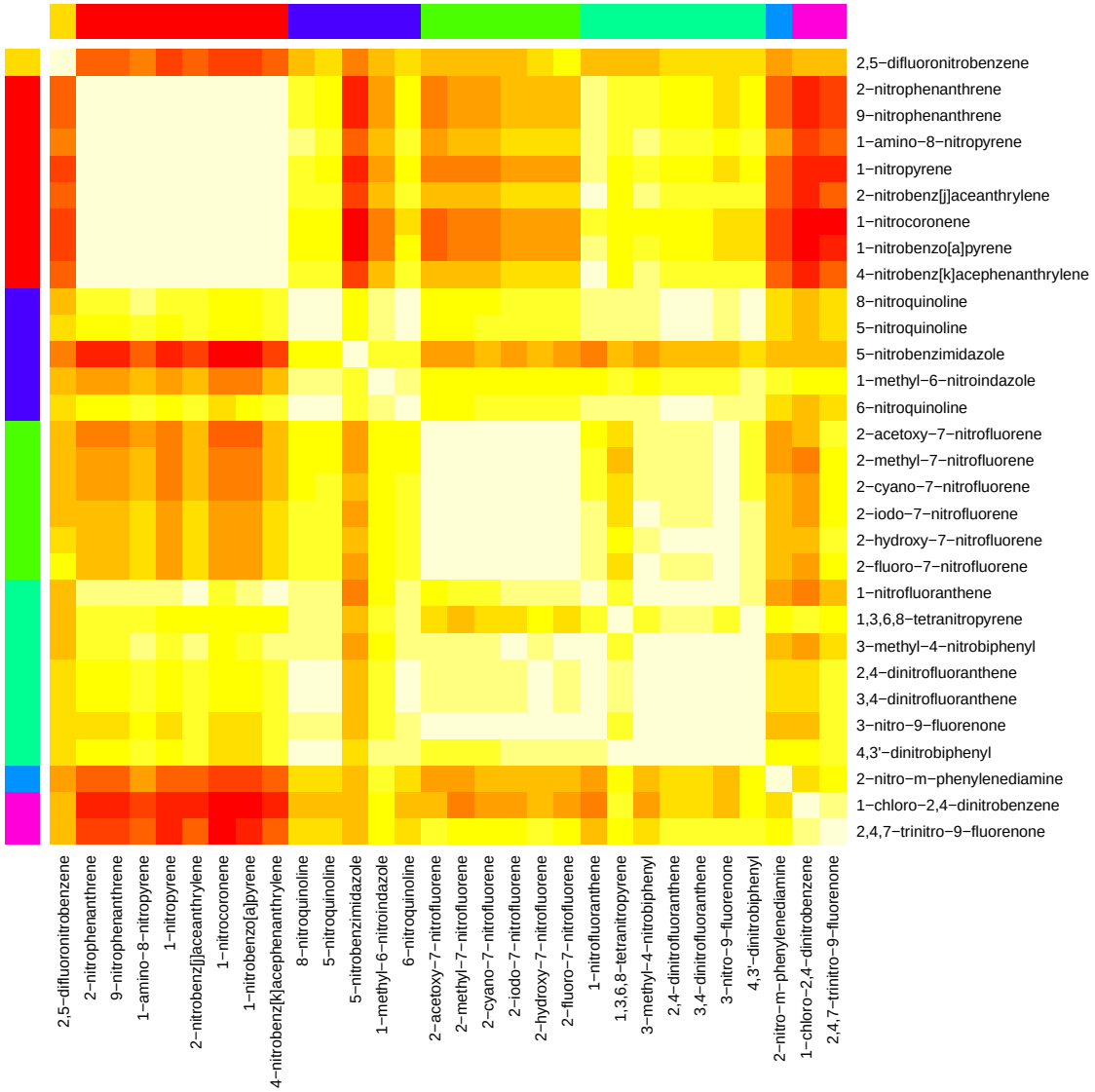

`linSimMat(w=1, method="maximum")`



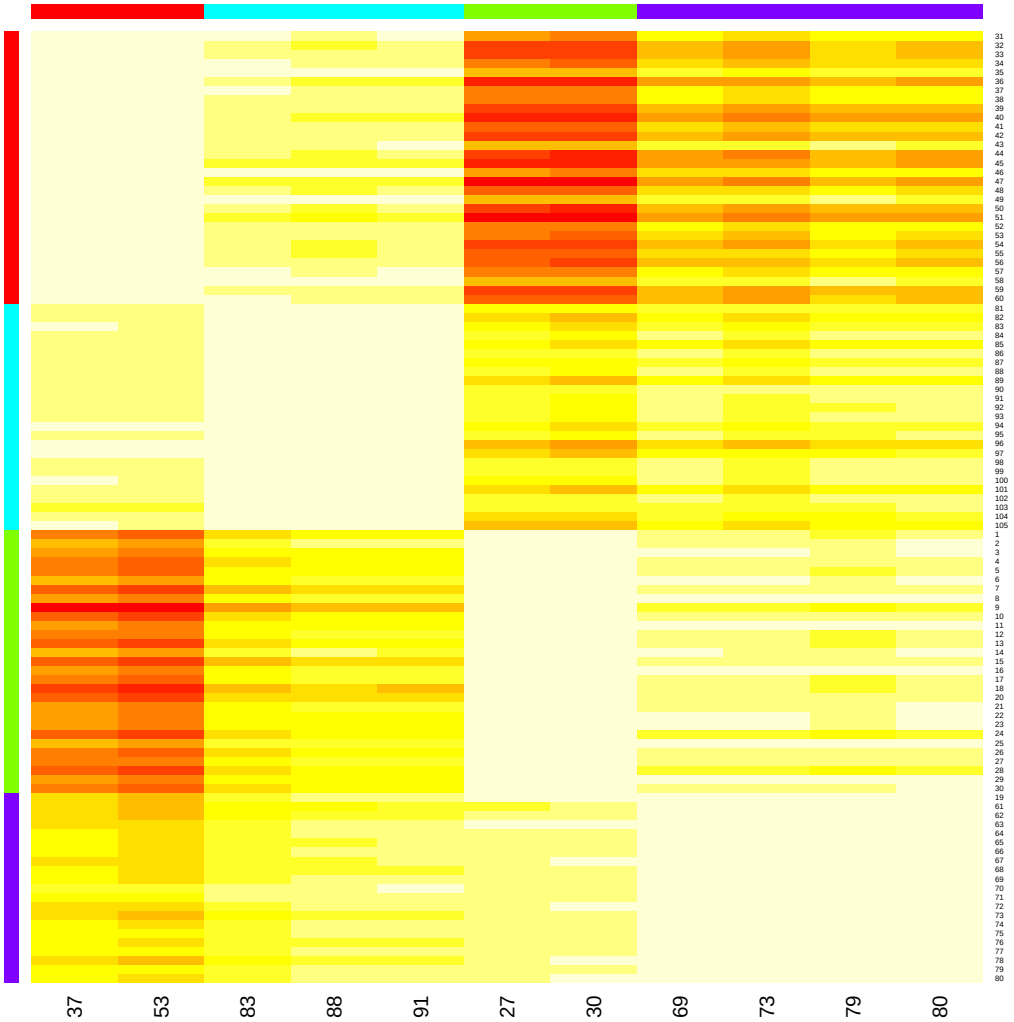
# Example:
# Two Clusters of Coiled Coil Sequences



# Example:
# AP Clustering of Chemical Compounds



# Example: Leveraged Affinity Propagation



# And Now . . .

# And Now ...


## ... let's see `apcluster` at work!

# Further Information

- Paper about package:

  U. Bodenhofer, A. Kothmeier, and S. Hochreiter. APCluster: an R package for affinity propagation clustering. *Bioinformatics*, **27**(17):2463–2464, 2011. DOI: 10.1093/bioinformatics/btr406.

- Package homepages:

  `http://www.bioinf.jku.at/software/apcluster/`
  `http://cran.r-project.org/web/packages/apcluster/index.html`

# Acknowledgments


Package co-authors:

- Andreas Kothmeier
- Johannes Palme

# Acknowledgments

Package co-authors:

- Andreas Kothmeier
- Johannes Palme

Co-workers in related applications and projects:

- Sepp Hochreiter
- Günter Klambauer
- Andreas Mayr

# Acknowledgments

Package co-authors:

- Andreas Kothmeier
- Johannes Palme

Co-workers in related applications and projects:

- Sepp Hochreiter
- Günter Klambauer
- Andreas Mayr

For the invitation to and organization of this webinar:

- Ray DiGiacomo, Jr., and the Orange County R User Group