WeightedCluster Preview

Matthias Studer
Institute for Demographic and Life Course Studies
University of Geneva
matthias.studer@unige.ch

1 Installation

Some functions of WeightedCluster require the free GraphViz program (Gansner and North, 1999). It needs to be installed before launching R for these functions to work properly. You can download it here: `http:\www.graphviz.org`.

The **WeightedCluster** library can be installed and loaded using the following commands:

```
R> install.packages("WeightedCluster")
R> library(WeightedCluster)
```

2 An illustrative example

In this preview, we use the dataset from McVicar and Anyadike-Danes (2002) which is distributed with the **TraMineR** library (Gabadinho *et al.*, 2011). This dataset contains sequences of school-to-work transitions in Northern Ireland. The dataset is loaded using :

```
R> data(mvad)
```

`wcAggregateCases` allows us to identify and aggregate identical state sequences (which are in columns `17:86`). We print out the basic information about the aggregation and create the `uniqueMvad` object which contains only unique sequences.

```
R> aggMvad <- wcAggregateCases(mvad[, 17:86])
R> print(aggMvad)

Number of disaggregated cases:  712
Number of aggregated cases:  490
Average aggregated cases:  1.45
Average (weighted) aggregation:  1.45

R> uniqueMvad <- mvad[aggMvad$aggIndex, 17:86]
```

Using the unique sequence dataset, we build a sequence object and compute dissimilarities between sequences (see Gabadinho *et al.*, 2011, for more on this topics). The vector `aggMvad$aggWeights` store the number of replication of each unique sequence. It is thus used as unique sequence weight.

```
R> mvad.seq <- seqdef(uniqueMvad, weights=aggMvad$aggWeights)
R> ## Computing Hamming distance between sequence
R> diss <- seqdist(mvad.seq, method="HAM")
```

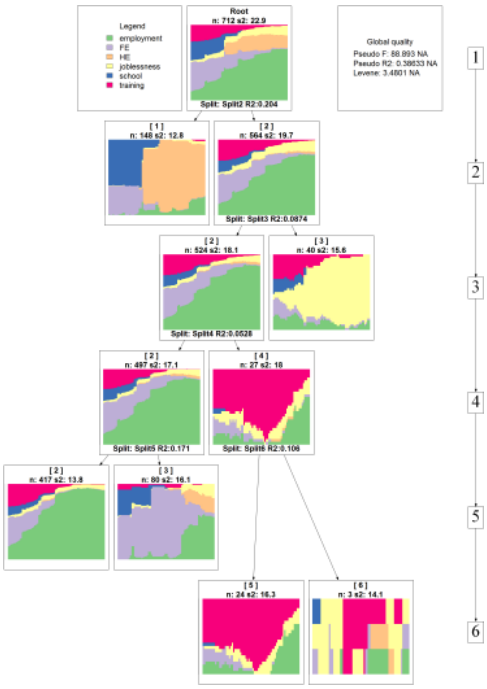
3 Hierarchical clustering

We can regroup similar sequences using hierarchical clustering with `"average"` method using weights (`aggMvad$aggWeights`) (any method may be used).

```
R> averageClust <- hclust(as.dist(diss), method="average", members=aggMvad$aggWeights)
```

The agglomeration schedule can be represented graphically as a tree using:

```
R> averageTree <- as.seqtree(averageClust, seqdata=mvad.seq, diss=diss, nclus-
ter=6)
R> seqtreedisplay(averageTree, type="d", border=NA,  showdepth=TRUE)
```



4 Cluster quality

We can automatically compute several clustering quality measures (presented in table 1) for a range of numbers of groups: 2 until ncluster=10.

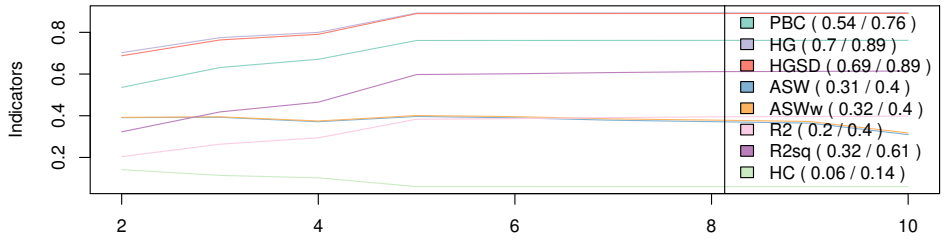
```
R> avgClustQual <- as.clustrange(averageClust, diss, weights=aggMvad$aggWeights, nclus-
ter=10)
```

The results can be plotted and used to identify the best number of groups (you can also print them).

```
R> plot(avgClustQual)
```

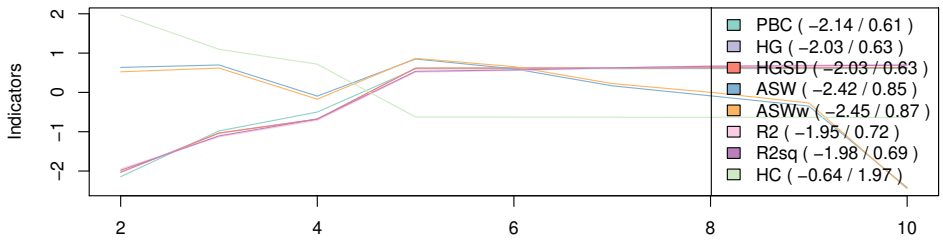
Table 1: Cluster Quality Measures Available in WeightedCluster

Name	Abrv.	Range	Min/Max	Interpretation
Point Biserial Correlation	PBC	$[-1;1]$	Max	Capacity of the clustering to reproduce the original distance matrix.
Hubert's Gamma	HG	$[-1;1]$	Max	Capacity of the clustering to reproduce the original distance matrix (Order of magnitude).
Hubert's Somers D	HGSD	$[-1;1]$	Max	Same as above, taking into account ties in the distance matrix.
Hubert's C	HC	$[0;1]$	**Min**	Gap between the current quality of clustering and the best possible quality for this distance matrix and number of groups.
Average Silhouette Width	ASW	$[-1;1]$	Max	Coherence of the assignments. A high coherence indicates high between groups distances and high intra group homogeneity.
Calinski-Harabasz index	CH	$[0;+\infty[$	Max	Pseudo F computed from the distances.
Calinski-Harabasz index	CHsq	$[0;+\infty[$	Max	Idem, using the *squared* distances.
Pseudo R^2	R2	$[0;1]$	Max	Share of the discrepancy explained by the clustering.
Pseudo R^2	R2sq	$[0;1]$	Max	Idem, using the *squared* distances.



It is usually easier to choose the number of groups based on standardized scores. Here, five groups seems to be a good solution.

```
R> plot(avgClustQual, norm="zscore")
```



Alternatively, we can retrieve the two best solutions according to each quality measure:

```
R> summary(avgClustQual, max.rank=2)

     1. N groups 1.  stat 2. N groups 2.  stat
PBC            10   0.7616            9    0.761
HG             10   0.8939            9    0.893
HGSD           10   0.8910            9    0.890
ASW             5   0.3966            3    0.393
ASWw            5   0.4010            6    0.396
CH              2 181.9886            3  126.780
R2             10   0.3980            9    0.396
CHsq            2 338.5174            5  262.451
R2sq           10   0.6145            9    0.613
HC             10   0.0598            9    0.060
```

5 PAM clustering

The **WeightedCluster** library also provides an optimized PAM algorithm. We can automatically compute PAM cluster for a range of numbers of groups using:

```
R> pamClustRange <- wcKMedRange(diss, kvals=2:10, weights=aggMvad$aggWeights)
```

As before, we can plot the quality measures of each solution (not shown here) or retrieve the two best solutions according to each quality measure using:

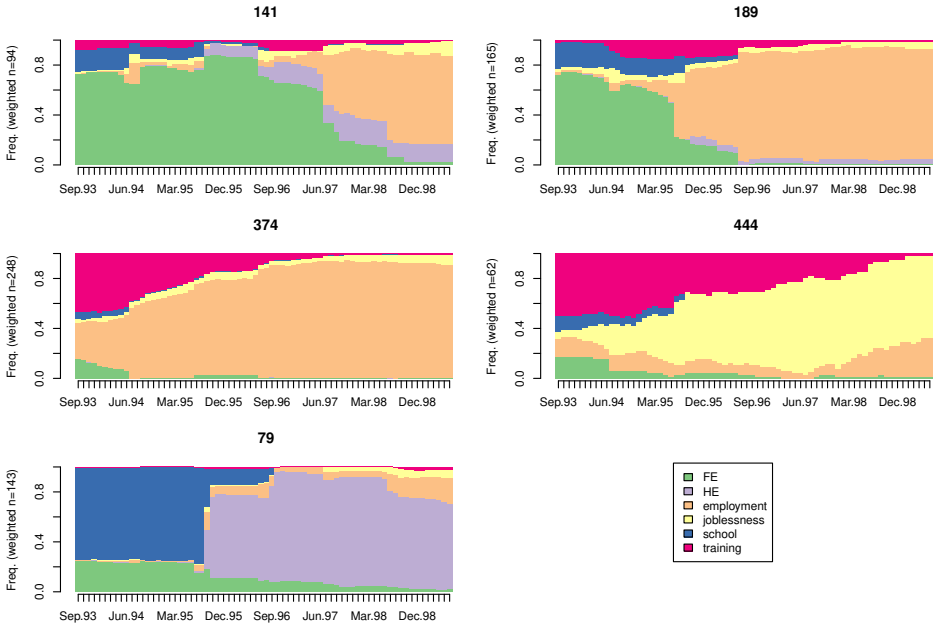
```
R> summary(pamClustRange, max.rank=2)

     1. N groups 1.  stat 2. N groups 2.  stat
PBC             2   0.619            4    0.618
HG             10   0.845            9    0.845
HGSD           10   0.842            9    0.842
ASW             2   0.411            9    0.370
ASWw            2   0.412            9    0.378
CH              2 200.286            3  151.245
R2             10   0.590            9    0.576
CHsq            2 394.893            4  310.881
R2sq           10   0.786            9    0.774
HC              9   0.100           10    0.104
```

6 Keeping a solution

The objets returned by `as.clustrange` or `wcKMedRange` contain a `data.frame` with cluster membership (named `clustering`). For instance, we can plot the sequences

according to PAM clustering in 5 groups using:

```
R> seqdplot(mvad.seq, group=pamClustRange$clustering$cluster5, border=NA)
```



7 Disaggregating data

Once the sequences have been regrouped, it is often useful to "disaggregate" the data. For instance, we may want to add the cluster membership in the original data set (i.e. before unique sequences were identified). This allows us to cross-tabulate cluster membership and father unemployment (variable `funemp`). This operation is performed using `aggMvad$disaggIndex` which stores the index of each unique sequence in the original dataset.

```
R> uniqueCluster5 <- avgClustQual$clustering$cluster5
R> mvad$cluster5 <- uniqueCluster5[aggMvad$disaggIndex]
R> table(mvad$funemp, mvad$cluster5)


        1   2   3   4   5
  no  134 348  70  24  19
  yes  14  69  10  16   8
```

References

Gabadinho A, Ritschard G, Müller NS, Studer M (2011). "Analyzing and visualizing state sequences in R with TraMineR." *Journal of Statistical Software*, **40**(4), 1–37. URL `http://www.jstatsoft.org/v40/i04/`.

Gansner ER, North SC (1999). "An Open Graph Visualization System and Its Applications to software engineering." *Software - Practice and Experience*, **30**, 1203–1233.

McVicar D, Anyadike-Danes M (2002). "Predicting successful and unsuccessful transitions from school to work using sequence methods." *Journal of the Royal Statistical Society A*, **165**(2), 317–334.