

Le manuel de la librairie **WeightedCluster**

Un guide pratique pour la création de typologies de trajectoires
en sciences sociales avec **R**

Matthias Studer

Institute for Demographic and Life Course Studies
University of Geneva

Abstract

Ce manuel poursuit un double but: présenter la librairie **WeightedCluster** et proposer un guide pas à pas à la création de typologie de séquences pour les sciences sociales. Cette librairie permet notamment de représenter graphiquement les résultats d'une analyse en clusters hiérarchique, de regrouper les séquences identiques afin d'analyser un nombre de séquences plus important, de calculer un ensemble de mesure de qualité d'une partition, ainsi qu'un algorithme PAM optimisé prenant en compte les pondérations. La librairie offre également des procédures pour faciliter le choix d'une solution de clustering particulière et définir le nombre de groupes.

Outre les méthodes, nous discutons également de la construction de typologie de séquences en sciences sociales et des hypothèses sous-jacentes à cette démarche. Nous clarifions notamment la place que l'on devrait donner à la construction de typologie en analyse de séquences. Nous montrons ainsi que ces méthodes offrent un point de vue descriptif important sur les séquences en faisant ressortir des patterns récurrents. Toutefois, elles ne devraient pas être utilisées dans une optique confirmatoire, car elles peuvent conduire à des conclusions trompeuses.

Keywords: Analyse de séquences, trajectoire, parcours de vie, optimal matching, distance, cluster, typologie, pondération, mesure de qualité d'une partition, R.

Pour citer ce document ou la librairie **WeightedCluster**, merci d'utiliser :

Studer, Matthias (2012). *Étude des inégalités de genre en début de carrière académique à l'aide de méthodes innovatrices d'analyse de données séquentielles*, Chapitre : Le manuel de la librairie **WeightedCluster** : Un guide pratique pour la création de typologies de trajectoires en sciences sociales avec R. Thèse SES 777, Faculté des sciences économiques et sociales, Université de Genève.

1. Introduction

Ce manuel poursuit un double but. Il présente les fonctionnalités offertes par la

librairie **WeightedCluster** pour la construction et la validation de clustering de données pondérées dans R. En même temps, nous appliquons, tout au long de ce manuel, les méthodes présentées à l'analyse de séquences en sciences sociales, ce qui en fait également un guide pas à pas de la construction de typologie de séquences. Nous discutons également des implications et des hypothèses sociologiques sous-jacentes à ces analyses.

D'une manière générale, l'analyse en clusters a pour but de construire un regroupement d'un ensemble d'objets de telle manière que les groupes obtenus soient les plus homogènes possible et les plus différents possible les uns des autres. Il existe beaucoup de méthodes différentes pour réaliser cette opération dont la pertinence dépend notamment des objets analysés. Les méthodes présentées dans ce manuel et disponibles dans la librairie **WeightedCluster** se basent sur une mesure de dissimilarité entre les objets, ce qui permet de *comparer* les objets en quantifiant leur similarité.

Nous présentons deux étapes de l'analyse en clusters basée sur des dissimilarités : les algorithmes de regroupement des objets avant d'aborder la mesure de la qualité des résultats obtenus. Cette dernière étape est essentielle puisque toutes les analyses en cluster présentées produisent des résultats que celui-ci soit pertinent ou non (Levine 2000). Ces mesures fournissent également une aide précieuse pour choisir le meilleur regroupement parmi les solutions issues de différents algorithmes ou pour sélectionner le nombre de groupes optimal. Pour ce faire, nous utilisons ici les méthodes offertes par la librairie **WeightedCluster** telle qu'un algorithme PAM hautement optimisé ou encore le calcul et la visualisation de la qualité d'un ensemble de solutions de clustering.

La particularité de la librairie **WeightedCluster** est la prise en compte de la pondération des observations dans les deux phases de l'analyse décrites précédemment. Il y a aux moins deux cas de figure où l'utilisation de pondération se révèle indispensable. Premièrement, la pondération permet de regrouper les cas identiques, ce qui réduit considérablement la mémoire et le temps de calcul utilisé. La section A présente en détail les fonctionnalités proposées par la librairie **WeightedCluster** pour automatiser ce regroupement. Deuxièmement, les données d'enquête sont souvent pondérées pour corriger les biais de représentativité. Dans ces cas de figure, il est essentiel d'utiliser les poids dans les procédures de clustering pour que les résultats ne soient pas biaisés. La librairie **WeightedCluster** offre des fonctions pour inclure les pondérations uniquement pour les analyses où il n'existe pas, à l'heure actuelle et à notre connaissance, d'autres librairies qui le font déjà. Si ceci devait être le cas, comme pour les procédures de clustering hiérarchiques, nous présentons les solutions existantes.

Comme mentionné précédemment, ce manuel se veut également un guide pas à pas pour la construction de typologie de trajectoires dans R. En tant que tel, ce manuel se destine à un public large, c'est pourquoi nous avons mis en annexe les parties plus techniques ou plus avancées¹. Cette optique nous amènera également à discuter des implications et des hypothèses sociologiques sous-jacentes

1. Toutefois, une connaissance des bases de l'analyse de séquences est supposée connue. Dans le cas contraire, une introduction à leurs mise en pratique avec **TraMineR** est disponible dans Gabadinho *et al.* (2011).

à l’analyse en clusters.

Dans ce manuel, nous illustrons les méthodes présentées à l’aide des données issues de l’étude de [McVicar and Anyadike-Danes \(2002\)](#) qui sont disponibles dans **TraMineR** ([Gabadinho *et al.* 2011](#)) ce qui permettra au lecteur de reproduire l’ensemble des analyses présentées. Ces données décrivent sous forme de séquences de statuts mensuels la transition vers l’emploi de jeunes nord-irlandais ayant terminé l’école obligatoire. Le but original de l’étude était d’“identify the ‘at-risk’ young people at age 16 years and to characterize their post-school career trajectories”. Ce jeu de donnée est pondéré pour corriger les biais de représentativité.

La suite de ce manuel est organisée de la manière suivante. Nous commençons par présenter les enjeux théoriques de la construction de typologie de séquences en sciences sociales avant d’aborder les méthodes d’analyse en clusters à proprement parler. Nous reviendrons ensuite brièvement sur les différentes étapes de l’analyse en clusters avant de présenter plusieurs algorithmes de clustering disponibles dans R pour les données pondérées. Nous présentons ensuite plusieurs mesures de la qualité d’un clustering et les principales utilisations que l’on peut en faire. Finalement, nous discutons des enjeux de l’interprétation de l’analyse en clusters et des risques que cela comporte lorsque l’on analyse les liens entre types de trajectoires et des facteurs explicatifs, une pratique courante en sciences sociales.

2. Typologie de séquences en sciences sociales

La création d’une typologie est la méthode la plus utilisée pour analyser des séquences ([Hollister 2009](#); [Aisenbrey and Fasang 2010](#); [Abbott and Tsay 2000](#)). Cette procédure a pour but d’identifier les patterns récurrents dans les séquences ou, en d’autres termes, les successions d’états typiques par lesquelles passent les trajectoires. Les séquences individuelles se distinguent les une des autres par une multitude de petites différences. La construction d’une typologie des séquences a pour but de gommer ces petites différences afin d’identifier des types de trajectoires homogènes et distincts les uns des autres.

Cette analyse doit faire ressortir des patterns récurrents et/ou des “séquences idéales-typiques” ([Abbott and Hrycak 1990](#)). Ces patterns peuvent également s’interpréter comme des interdépendances entre différents moments de la trajectoire. La recherche de tels patterns est ainsi une question importante dans plusieurs problématiques de sciences sociales ([Abbott 1995](#)). Elle permet notamment de mettre en lumière les contraintes légales, économiques ou sociales qui encadrent la construction des parcours individuels. Comme le notent [Abbott and Hrycak \(1990\)](#), si les séquences types peuvent résulter de contrainte que l’on redécouvre, ces séquences typiques peuvent également agir sur la réalité en servant de modèles aux acteurs qui anticipent leur propre futur. Ces différentes possibilités d’interprétations font de la création de typologie un outil puissant. Dans l’étude de [McVicar and Anyadike-Danes \(2002\)](#) par exemple, une telle analyse devrait permettre d’identifier les successions d’états qui mènent à des

situations ‘at-risk’, c’est-à-dire marquées par un fort taux de chômage.

Cette procédure de regroupement repose sur une *simplification* des données. Elle offre ainsi un point de vue descriptif et exploratoire sur les trajectoires. En simplifiant l’information, on peut identifier les principales caractéristiques et les motifs des séquences. Toutefois, il existe un risque que cette *simplification* soit *abusive* et ne corresponde pas à une réalité des données. En d’autres termes, il est possible que les types identifiés ne soient pas clairement séparés les uns des autres ou qu’ils ne soient pas suffisamment homogènes. Ce risque a souvent été négligé dans la littérature. Comme le fait remarquer [Levine \(2000\)](#) notamment, toutes les analyses en cluster produisent un résultat, que celui-ci soit pertinent ou non. Il est donc toujours possible d’interpréter les résultats et d’en faire une théorie qui se proclame souvent “sans hypothèses préalables” ou “issue des données”, alors que cette typologie peut également être le fruit d’un artifice statistique.

Selon [Shalizi \(2009\)](#), la pertinence d’une typologie dépend notamment de trois conditions. Premièrement, et c’est la plus importante, la typologie ne doit pas être dépendante de l’échantillonnage, ce qui signifie qu’elle doit être généralisable à d’autres observations. Deuxièmement, une typologie devrait s’étendre à d’autres propriétés. Finalement, la typologie obtenue devrait également être fondée par une théorie du domaine analysé. [Shalizi \(2009\)](#) cite deux exemples pour illustrer ses propos. La classification des espèces animales, créées sur la base de caractéristiques physiques, s’applique également à d’autres caractéristiques telles que le chant d’un oiseau. Par contre, la classification des étoiles en constellations a permis de construire des théories sans fondements.

À notre connaissance, il n’existe pas de méthode pour *attester* de la pertinence théorique d’une typologie. Toutefois, nous présentons ici plusieurs indices pour mesurer la qualité statistique d’une partition obtenue à l’aide d’une procédure de regroupement automatique. L’utilisation de telles mesures est à notre sens une étape essentielle pour valider les résultats et, d’une manière plus générale, rendre les résultats de l’analyse de séquences utilisant le clustering plus crédibles.

Maintenant que nous avons présenté les enjeux théoriques de l’analyse en clusters, nous passons à la pratique en présentant les différentes étapes de l’analyse en clusters.

3. Installation et chargement

Pour utiliser la librairie **WeightedCluster**, il est nécessaire de l’installer et de la charger. Il suffit de l’installer une seule fois, mais elle devra être rechargée avec la commande `library` à chaque fois que R est lancé. Ces deux étapes sont réalisées de la manière suivante.

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

4. Étapes de l'analyse en clusters

D'une manière générale, l'analyse en clusters se déroule en quatre étapes sur lesquelles nous reviendrons plus en détail. On commence par calculer les *dissimilarités* entre séquences. On utilise ensuite ces dissimilarités pour regrouper les séquences similaires en types les plus homogènes possible et les plus différents possible les uns des autres. On teste généralement plusieurs algorithmes et nombres de groupes différents. On calcule ensuite pour chacun des regroupements obtenus des mesures de qualité. Ces mesures permettent de guider le choix d'une solution particulière et de la valider. Finalement, on interprète les résultats de l'analyse avant de mesurer éventuellement l'association entre le regroupement obtenu et d'autres variables d'intérêt.

Dans ce manuel, nous discuterons des trois dernières étapes de l'analyse. Nous n'offrons ici qu'une introduction au calcul de dissimilarités entre séquences. Rappelons notamment qu'une mesure de dissimilarité est une quantification de l'éloignement de deux séquences, ou d'une manière plus générale, de deux objets. Cette quantification permet ensuite de *comparer* les séquences. Dans l'analyse en clusters par exemple, cette information est nécessaire pour regrouper les séquences les plus similaires. On utilise généralement une matrice de dissimilarités qui contient l'ensemble des dissimilarités deux à deux ou, en d'autres termes, la quantification de toutes les comparaisons possibles.

Le choix de la mesure de dissimilarité utilisée pour quantifier les différences entre séquences est une question importante, mais qui dépasse le cadre de ce manuel. Plusieurs articles adressent cette question ([Aisenbrey and Fasang 2010](#); [Hollister 2009](#); [Lesnard 2010](#)). Ici, nous utilisons la distance d'optimal matching en utilisant les coûts utilisés dans l'article original de [McVicar and Anyadike-Danes \(2002\)](#).

Dans R, les distances entre séquences d'états peuvent être calculées à l'aide de la librairie **TraMineR** ([Gabadinho et al. 2011](#)). Pour calculer ces distances, il faut premièrement créer un objet "séquence d'états" à l'aide de la fonction `seqdef`. On calcule ensuite les distances avec la fonction `seqdist`. L'article [Gabadinho et al. \(2011\)](#) présente en détail ces différentes étapes.

Ici, nous commençons par charger le fichier de donnée exemple avec la commande `data`. Nous construisons ensuite l'objet séquence en spécifiant les colonnes qui contiennent les données (17 à 86) ainsi que la variable de pondération des observations. La fonction `seqdist` calcule la matrice des distances entre séquences.

```
R> data(mvad)
R> mvad.alphabet <- c("employment", "FE", "HE", "joblessness", "school",
  "training")
R> mvad.labels <- c("Employment", "Further Education", "Higher Education",
  "Joblessness", "School", "Training")
R> mvad.scodes <- c("EM", "FE", "HE", "JL", "SC", "TR")
R> mvadseq <- seqdef(mvad[, 17:86], alphabet = mvad.alphabet, states = mvad.scodes,
  labels = mvad.labels, weights=mvad$weight, xtstep=6)
R> ## Defining the custom cost matrix
```

```
R> subm.custom <- matrix(
  c(0, 1, 1, 2, 1, 1,
    1, 0, 1, 2, 1, 2,
    1, 1, 0, 3, 1, 2,
    2, 2, 3, 0, 3, 1,
    1, 1, 1, 3, 0, 2,
    1, 2, 2, 1, 2, 0),
  nrow = 6, ncol = 6, byrow = TRUE)
R> ## Computing the OM dissimilarities
R> mvaddist <- seqdist(mvadseq, method="OM", indel=1.5, sm=subm.custom)
```

La suite de ce manuel traite des trois étapes suivantes. Nous commençons par discuter des méthodes de clustering disponibles pour les données pondérées. Nous présenterons ensuite les mesures de la qualité d'un clustering offert par la librairie **WeightedCluster**. Ces mesures nous permettront de choisir une solution particulière et de mesurer sa validité. Finalement, nous discuterons des problèmes d'interprétation des résultats de l'analyse en clusters et du calcul des liens entre typologies et d'autres variables d'intérêts.

5. Le clustering

Il existe beaucoup d'algorithmes de clustering différents. Nous présentons ici des méthodes issues de deux logiques différentes : les méthodes de regroupement hiérarchique et celles de partitionnement en un nombre prédéfini de groupes. Nous concluons sur les interactions possibles entre ces types d'algorithmes.

5.1. Clustering hiérarchique

Nous présentons ici les procédures de regroupements hiérarchiques ascendantes qui fonctionnent ainsi. On part des observations, chacune d'entre elles étant considérée comme un groupe. À chaque itération, on regroupe les deux groupes (ou observations au départ) les plus proches, jusqu'à ce que toutes les observations ne forment plus qu'un seul groupe. Le schéma agglomératif, c'est-à-dire la succession des regroupements effectués, représente la procédure de clustering sous la forme d'un arbre que l'on appelle dendrogramme. Une fois le schéma agglomératif construit, on sélectionne le nombre de groupes. La partition retenue est obtenue en "coupant" l'arbre de regroupement au niveau correspondant ².

Dans R, le schéma agglomératif (la succession des regroupements effectués) est créé avec la fonction `hclust` ³. Cette fonction prend les paramètres suivants : la matrice de distances, la méthode (ici "ward", nous y reviendrons) ainsi que le vecteur de poids `members`.

2. Il existe également des procédures dites descendantes (ou divisées) qui procèdent de la même manière, mais en sens inverse. Au lieu de démarrer des observations que l'on considère comme des groupes, la procédure divise à chaque pas un des groupes jusqu'à ce que chaque observation corresponde à un groupe. Un algorithme est disponible dans R avec la fonction `diana` de la librairie `cluster`. Tout ce que nous présentons ici est également compatible avec l'objet retourné par cette fonction.

3. d'autres possibilités existent.

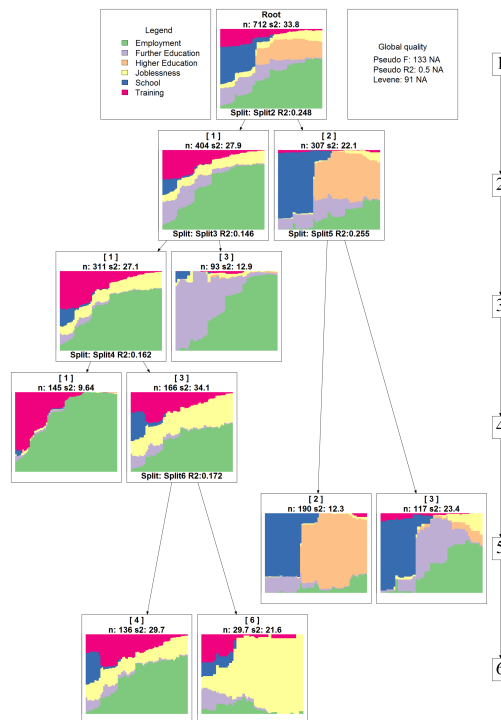
```
R> wardCluster <- hclust(as.dist(mvaddist), method="ward", members=mvad$weight)
```

Une fois le schéma agglomératif créé, on peut visualiser les dernières étapes de ce schéma à l'aide de la librairie **WeightedCluster**. Pour ce faire, on procède en deux temps. On commence par construire un arbre de séquences à partir du schéma agglomératif avec la commande `as.seqtrees`. Cette fonction prend les arguments suivant : la procédure de clustering (`wardCluster` dans notre cas), l'objet séquence (argument `seqdata`), la matrice de distance (argument `diss`) et le nombre maximum de regroupements à représenter (`ncluster`).

```
R> wardTree <- as.seqtrees(wardCluster, seqdata=mvadseq, diss=mvaddist, ncluster=6)
```

Une fois l'arbre construit, la fonction `seqtreedisplay` de la librairie **TraMineR** (Studer *et al.* 2011) permet de le représenter graphiquement. L'option `showdepth=TRUE` affiche les niveaux des regroupements sur la droite du graphique.

```
R> seqtreedisplay(wardTree, type="d", border=NA, showdepth=TRUE)
```



Cette figure présente le résultat de cette procédure et permet de visualiser les logiques de regroupement des séquences. La première distinction, censée être la plus importante, sépare les jeunes irlandais qui se dirigent vers l'école supérieure des autres. Les distinctions qui se font lorsque l'on passe de quatre à cinq groupes nous permettent toutefois de mettre en lumière que ce groupe

amalgame deux logiques : ceux qui vont vers l'école supérieure et ceux qui vont vers la "Further Education". Ce graphique fournit une aide importante pour identifier les distinctions pertinentes pour l'analyse. Ainsi, si l'on retient une solution en quatre groupes, on ne fera plus de distinction entre les deux logiques ("École supérieure" et "Further Education"). Par contre, la distinction en cinq groupes permet de l'identifier.

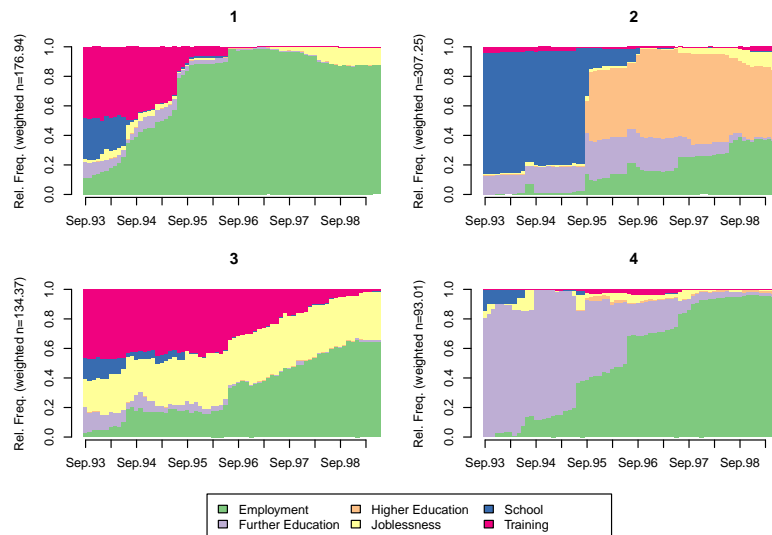
Cet exemple illustre la *simplification* des données effectuées par les procédures de clustering. En ne retenant que quatre groupes, on amalgame deux logiques qui pourraient (ou non) être pertinentes pour l'analyse. Notons qu'en ne retenant que cinq groupes, on amalgame les distinctions qui se font à un niveau inférieur. On effectue ainsi toujours une simplification.

Pour obtenir un regroupement en un nombre de groupes particulier, on coupe l'arbre à un niveau donné. Ceci signifie que l'on garde tous les nœuds terminaux si l'arbre s'arrêtait au niveau présenté sur la droite. Dans R, on récupère cette information à l'aide de la fonction `cutree`. Par exemple, pour utiliser le regroupement en 4 classes distinctes.

```
R> clust4 <- cutree(wardCluster, k=4)
```

On peut ensuite utiliser cette information dans d'autres analyses, par exemple, pour représenter les types obtenus à l'aide d'un chronogramme. Sans surprise, les graphiques obtenus correspondent aux nœuds terminaux de l'arbre au niveau quatre.

```
R> seqdplot(mvadseq, group=clust4, border=NA)
```



Comme nous l'avons mentionné, la fonction `hclust` propose sept algorithmes hiérarchiques différents que l'on spécifie avec l'argument `method`. La librairie **fastcluster** fournit une version optimisée de cette fonction (Müllner 2011). Les algorithmes hiérarchiques **diana** Kaufman and Rousseeuw (procédure descen-

dante voir 1990) et le bêta-flexible clustering avec la fonction `agnes` sont disponibles dans la librairie `cluster` (Maechler *et al.* 2005)⁴. Le tableau 1 liste ces algorithmes en précisant le nom de la fonction à utiliser, la prise en compte des pondérations (l’entrée “Indep” signifie que l’algorithme est insensible aux pondérations) et l’interprétation de la logique de clustering. Une présentation plus détaillée de ces algorithmes est disponible dans Müllner (2011) et Kaufman and Rousseeuw (1990).

TABLE 1 – Algorithmes de regroupements hiérarchiques.

Nom	Fonction	Poids	Interprétation et notes.
single	<code>hclust</code>	Indep	Fusion des groupes avec les observations les plus proches.
complete	<code>hclust</code>	Indep	Minimisation du diamètre de chaque nouveau groupe (très sensible aux données atypiques).
average (ou UPGMA)	<code>hclust</code>	Oui	Moyenne des distances.
McQuitty (ou WPGMA)	<code>hclust</code>	Indep	Dépend des fusions précédentes.
centroid	<code>hclust</code>	Oui	Minimisation des distances entre médoïdes.
median	<code>hclust</code>	Indep	Dépend des fusions précédentes.
ward	<code>hclust</code>	Oui	Minimisation de la variance résiduelle.
beta-flexible	<code>agnes</code>	Non	Pour une valeur de β proche de -0.25 , utiliser par <code>.method=0.625</code> .

Dans leur article, Milligan and Cooper (1987) reprennent les résultats de différentes simulations effectuées afin d’évaluer les performances de certains de ces algorithmes. Notons que ces simulations ont été réalisées avec des données numériques et la mesure de distance euclidienne. L’extension de ces résultats à d’autres types de distances est sujette à discussion. Ils reportent ainsi des résultats plutôt mauvais pour les méthodes “single”, “complete”, “centroid” et “median”. La méthode “Ward” fait en général assez bien sauf en présence de données extrêmes qui biaisent les résultats. Ils reportent des résultats très variables pour la méthode “average”. Finalement, la méthode “beta-flexible” avec une valeur de bêta proche de -0.25 donne de bons résultats en présence de différentes formes d’erreur dans les données (Milligan 1989). Les meilleurs résultats sont obtenus par l’algorithme “flexible UPGMA” qui n’est pas disponible à l’heure actuelle dans R (Belbin *et al.* 1992).

Plusieurs critiques peuvent être adressées aux procédures hiérarchiques. Premièrement, et c’est la plus importante, la fusion de deux groupes se fait en maximisant un critère local. Ces procédures optimisent un critère local, c’est-à-dire qu’on estime localement la perte d’information due à un regroupement. Or, ces choix locaux peuvent mener à de grandes différences à des niveaux plus élevés et il n’est pas garanti qu’il soit les meilleurs d’un point de vue global. En d’autres termes, il arrive souvent qu’un choix bon au niveau local conduise à

4. Si l’on utilise ces fonctions, il est nécessaire de spécifier l’argument `diss=TRUE`, pour que l’algorithme utilise la matrice de distance passée en argument.

des résultats médiocres à un niveau de regroupement supérieur. Deuxièmement, les procédures ascendantes ne sont pas déterministes en particulier lorsque la mesure de distance ne prend que peu de valeurs différentes donnant lieu à des égalités entre lesquelles il faut trancher, ce qui peut notamment être le cas avec l'appariement optimal ou la distance de Hamming. Bien qu'une version particulière de cet algorithme produise généralement le même dendrogramme à chaque analyse⁵, plusieurs versions de ce même algorithme peuvent conduire à des résultats divergents. De plus, ce cas de figure peut pénaliser la procédure qui n'a pas de critère pour effectuer un choix (Fernández and Gómez 2008). Nous présentons à présent l'algorithme PAM qui a l'avantage de chercher à maximiser un critère global.

5.2. Partitioning Around Medoids

L'algorithme PAM pour "Partitioning Around Medoids" suit une autre logique que les algorithmes hiérarchiques (Kaufman and Rousseeuw 1990). Il vise à obtenir la meilleure partition d'un ensemble de données en un nombre k prédéfini de groupes. Par rapport aux autres algorithmes présentés, cet algorithme a l'avantage de maximiser un critère global et non uniquement un critère local.

Le but de l'algorithme est d'identifier les k meilleurs représentants de groupes, appelés médoïdes. Plus précisément, un médoïde correspond à l'observation d'un groupe ayant la plus petite somme pondérée des distances aux autres observations de ce groupe. Cet algorithme cherche ainsi à minimiser la somme pondérée des distances au médoïde.

Brièvement, on peut décrire le fonctionnement de cet algorithme en deux phases. Dans un premier temps, on initialise l'algorithme en cherchant les observations qui diminuent le plus la somme pondérée des distances aux médoïdes existants, en choisissant le médoïde de l'ensemble des données au départ. Une fois la solution initiale construite, la deuxième phase de l'algorithme, appelée "échange", commence. Pour chaque observation, on calcule le gain potentiel si l'on remplaçait l'un des médoïdes existants par cette observation. Le gain est calculé au niveau global et en fonction des distances pondérées aux médoïdes les plus proches. On remplace ensuite le médoïde par l'observation qui conduit au plus grand gain possible. On répète ces opérations jusqu'à ce qu'il ne soit plus possible d'améliorer la solution courante.

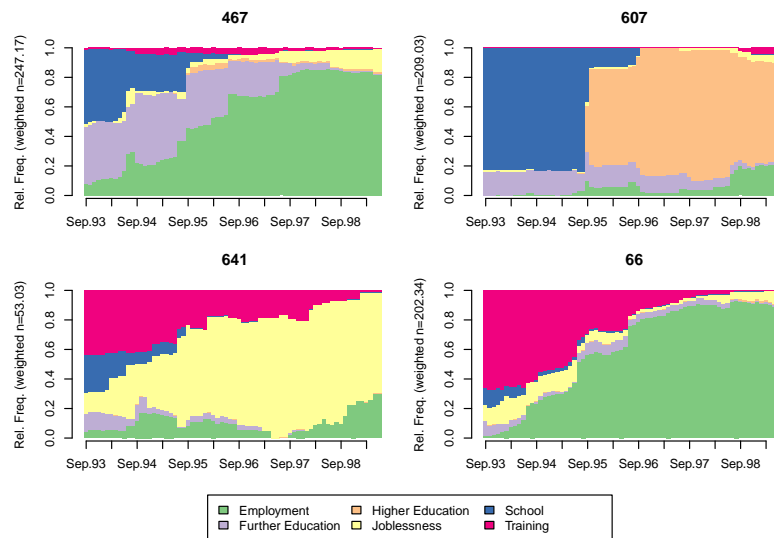
L'algorithme est disponible dans la librairie R **cluster** (Maechler *et al.* 2005; Struyf *et al.* 1997), mais il ne permet pas d'utiliser des données pondérées. Basée en partie sur le code disponible dans la librairie **cluster**, la fonction **wcKMedoids** de librairie **WeightedCluster** prend en compte les pondérations et implémente également les optimisations proposées par Reynolds *et al.* (2006), ce qui la rend plus rapide. Cette fonction consomme également deux fois moins de mémoire ce qui la rend adéquate pour analyser de très grands ensembles de données. L'annexe B présente plus en détail les gains en temps de calcul.

5. les algorithmes font généralement un choix qui dépend de l'ordre des observations, ce qui le rend reproductible pour autant que l'ordre soit inchangé.

```
R> pamclust4 <- wcKMedoids(mvaddist, k=4, weights=mvad$weight)
```

L'élément `clustering` contient l'appartenance aux clusters de chaque observation. On peut, par exemple, utiliser cette information pour représenter les séquences à l'aide d'un chronogramme.

```
R> seqdplot(mvadseq, group=pamclust4$clustering, border=NA)
```



Le numéro assigné au groupe correspond à l'index du médoïde de ce groupe. On peut ainsi récupérer les médoïdes en utilisant la commande `unique(pamclust4$clustering)`. La commande suivante utilise cette possibilité pour afficher les séquences médoïdes de chaque groupe.

```
R> print(mvadseq[unique(pamclust4$clustering), ], format="SPS")
```

```
Sequence
66  (TR,22)-(EM,48)
607 (SC,25)-(HE,45)
467 (SC,10)-(FE,12)-(EM,48)
641 (TR,22)-(JL,48)
```

L'algorithme PAM a plusieurs désavantages. Le premier est de créer des groupes "sphériques"⁶ centrés autour de leur médoïde, ce qui ne correspond pas nécessairement à la réalité des données. Deuxièmement, il est nécessaire de spécifier le nombre k de groupes au préalable. En testant plusieurs tailles k de partition, on n'est pas assuré que les types obtenus s'emboîtent comme dans le cas des procédures hiérarchiques et le temps de calcul peut être conséquent. Finalement, l'algorithme est dépendant du choix des médoïdes de départ qui n'est pas toujours effectué de manière optimale.

6. Ce qui est également le cas du critère de "Ward" dans les procédures hiérarchiques.

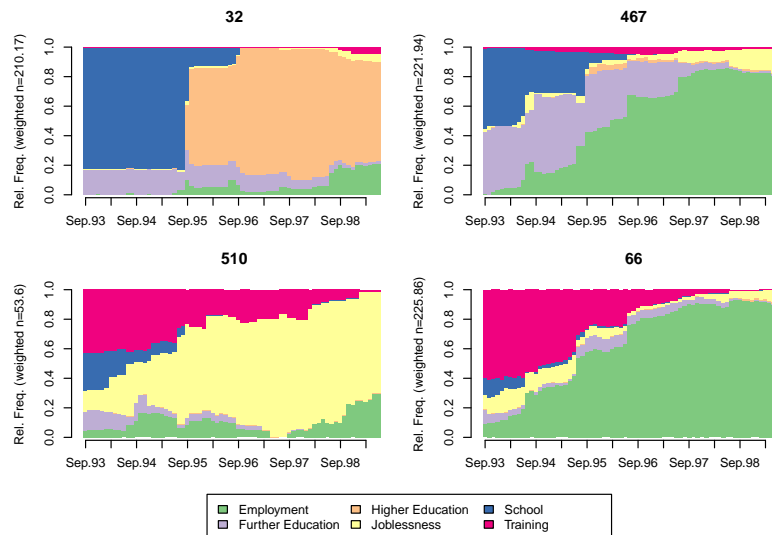
6. Combiner les algorithmes

Les deux procédures de clustering peuvent être combinées, ce qui produit parfois de meilleurs résultats. Pour ce faire, on spécifie comme point de départ de la fonction `wcKMedoids` le clustering obtenu par la méthode hiérarchique (l'argument `initialclust=wardCluster`).

```
R> pamwardclust4 <- wcKMedoids(mvaddist, k=4, weights=mvad$weight, initialclust=wardCluster
```

Ceci aboutit ici à une solution légèrement différente, mais avec une meilleure qualité.

```
R> seqdplot(mvadseq, group=pamwardclust4$clustering, border=NA)
```



Nous avons présenté plusieurs types d'analyses en cluster qui nous ont conduits à des solutions différentes. Comment faire un choix entre ces solutions ? Cette question est d'autant plus importante que nous aurions également pu sélectionner des nombres de groupes différents pour chaque procédure, ce qui nous amènerait à un grand nombre de possibilités. Les mesures de la qualité d'une partition que nous présentons maintenant aident à réaliser ce choix en offrant une base de comparaison de ces différentes solutions.

7. Mesurer la qualité d'une partition

Les mesures de la qualité d'une partition ont deux objectifs. Premièrement, certaines d'entre elles donnent une idée de la qualité statistique de la partition. Deuxièmement, ces mesures aident au choix de la meilleure partition d'un point de vue statistique. Elles sont ainsi d'une aide précieuse pour sélectionner le nombre de groupes ou le meilleur algorithme.

7.1. Présentation des mesures

La librairie **WeightedCluster** propose plusieurs mesures de la qualité d’une partition qui sont listées dans le tableau 2. Le choix de ces mesures est largement inspiré par [Hennig and Liao \(2010\)](#) que nous avons complété avec le “C-index” qui figurait parmi les meilleurs indices selon [Milligan and Cooper \(1985\)](#). La présentation que nous faisons ici se centre sur les concepts. Le lecteur intéressé pourra toutefois se référer à l’annexe C qui présente les détails mathématiques de ces mesures ainsi que les ajustements effectués pour prendre en compte la pondération des observations. Outre le nom des mesures de qualités, le tableau 2 présente leurs principales caractéristiques à l’aide des informations suivantes :

- Abrv : abréviation utilisée dans la librairie **WeightedCluster**.
- Étendu : intervalle des valeurs possibles.
- Min/Max : Est-ce qu’une bonne partition minimise ou maximise cette mesure ?
- Interprétation de la valeur.

TABLE 2 – Mesures de la qualité d’un regroupement.

Nom	Abrv.	Étendue	Min/Max	Interprétation
Point Biserial Correlation	PBC	$[-1; 1]$	Max	Mesure de la capacité du clustering à reproduire les distances.
Hubert’s Gamma	HG	$[-1; 1]$	Max	Mesure de la capacité du clustering à reproduire les distances (ordre de grandeur).
Hubert’s Somers D	HGSD	$[-1; 1]$	Max	Mesure de la capacité du clustering à reproduire les distances (ordre de grandeur) avec prise en compte des égalités sur les distances.
Hubert’s C	HC	$[0; 1]$	Min	Écart entre la partition obtenue et la meilleure partition qu’il serait théoriquement possible d’obtenir avec ce nombre de groupes et ces distances.
Average Silhouette Width	ASW	$[-1; 1]$	Max	Cohérence des assignations. Une cohérence élevée indique des distances inter-groupes élevées et une forte homogénéité intragroupe.
Average Silhouette Width (weighted)	ASWw	$[-1; 1]$	Max	Idem que précédant, si l’unité des poids n’a pas un sens explicite.
Calinski-Harabasz index	CH	$[0; +\infty[$	Max	Pseudo F calculé à partir des distances.
Calinski-Harabasz index	CHsq	$[0; +\infty[$	Max	Idem que précédant, mais en utilisant les distances <i>au carré</i> .
Pseudo R^2	R2	$[0; 1]$	Max	Part de la dispersion expliquée par la solution de clustering (uniquement pour comparer des partitions avec nombre de groupes identiques).
Pseudo R^2	R2sq	$[0; 1]$	Max	Idem que précédant, mais en utilisant les distances <i>au carré</i> .

Les trois premières mesures, à savoir “Point Biserial Correlation” ([Milligan and Cooper 1985](#); [Hennig and Liao 2010](#)), “Hubert’s Gamma” et “Hubert’s Somers D” ([Hubert and Arabie 1985](#)), suivent la même logique. Elles mesurent la capacité d’une partition des données à reproduire la matrice des distances. Si

la première mesure la capacité à reproduire la valeur exacte des distances, les deux suivantes se basent sur les concordances. Ceci implique que, selon ces deux derniers indices, une partition est bonne si les distances entre les groupes sont plus grandes que celles à l'intérieur des groupes. Techniquement, on mesure cette capacité en calculant l'association entre la matrice de distance et une deuxième mesure de distance qui prend la valeur 0 pour les observations qui sont dans le même groupe et 1 sinon. On utilise la corrélation de Pearson ("Point Biserial Correlation"), Gamma de Goodman et Kruskal ("Hubert's Gamma") ou le D de Somers ("Hubert's Somers D").

L'indice "Hubert's C" met en parallèle la partition obtenue et la meilleure partition que l'on aurait pu obtenir avec ce nombre de groupe et cette matrice de distance. Contrairement aux autres indices, une petite valeur indique une bonne partition des données.

Les indices de Calinski-Harabasz (Calinski and Harabasz 1974) se basent sur la statistique F de l'analyse de variance. Cette mesure a donné de très bons résultats dans les simulations de Milligan and Cooper (1985). Toutefois, son extension à des données non numériques est sujette à discussion (Hennig and Liao 2010). On peut discuter de sa pertinence si la mesure de distance est euclidienne (ou euclidienne au carré) auquel cas, cette mesure revient à utiliser la statistique F sur les coordonnées que l'on peut associer aux observations, par exemple avec une analyse en coordonnées principales. Pour ce cas de figure, la librairie **WeightedCluster** fournit cette statistique en utilisant les carrés des distances lorsque la distance est euclidienne ou la distance elle-même lorsque la mesure est déjà une distance euclidienne au carré, comme la distance de Hamming par exemple.

Le "R-square" calcule la part de la dispersion expliquée par une partition (Studer *et al.* 2011). Cette mesure n'est pertinente que pour comparer des partitions comportant le même nombre de groupe, car elle ne pénalise pas la complexité.

Finalement, la mesure "Average Silhouette Width" proposée par Kaufman and Rousseeuw (1990) est particulièrement intéressante. Elle se base sur la cohérence de l'assignation d'une observation à un groupe donné en mettant deux éléments en parallèle : la distance moyenne pondérée d'une observation aux autres membres de son groupe et la distance moyenne pondérée au groupe le plus proche. Une valeur de silhouette est calculée pour chaque observation. Si cette valeur est négative, l'observation est mal classée, car elle est plus proche d'un autre groupe que du sien. Au contraire, une valeur proche de 1 signifie que l'observation est proche de son groupe et loin de tous les autres. Généralement, on regarde plutôt la silhouette moyenne. Si celle-ci est faible, cela signifie que les groupes ne sont pas clairement séparés les uns des autres ou que l'homogénéité des groupes est faible. De manière intéressante, Kaufman and Rousseeuw (1990) proposent des ordres de grandeur pour interpréter cette mesure que nous reproduisons dans le tableau 3.

La formulation originale de Kaufman and Rousseeuw (1990) suppose que l'unité des pondérations corresponde à une observation, ce qui est le cas si les pondérations résultent d'agrégation (voir l'annexe A) ou si les données ne sont pas

TABLE 3 – Ordres de grandeur pour interpréter la mesure *ASW*

<i>ASW</i>	Interprétation proposée
0.71-1.00	Structure forte identifiée.
0.51-0.70	Structure raisonnable identifiée.
0.26-0.50	La structure est faible et pourrait être artificielle. Essayer d'autres algorithmes.
≤ 0.25	Aucune structure.

pondérées. Pour le cas où les pondérations visent à corriger la représentativité de données d'enquêtes (comme c'est le cas ici), nous proposons une variante de cette mesure appelée "ASWw" détaillée dans l'annexe C.1. D'une manière générale, les résultats entre ces deux variantes sont très similaires, "ASWw" tendant à attribuer une qualité légèrement plus élevée.

Ces mesures sont calculées avec la fonction `wcClusterQuality`. La valeur retournée par la fonction est une liste qui comprend deux éléments. L'élément `stats` contient les valeurs des mesures de qualité.

```
R> clustqual4 <- wcClusterQuality(mvaddist, clust4, weights=mvad$weight)
R> clustqual4$stats
```

PBC	HG	HGSD	ASW	ASWw	CH	R2	CHsq	R2sq	HC
0.46	0.62	0.61	0.24	0.24	144.83	0.38	310.58	0.57	0.17

Selon la mesure $ASWw=0.24$, la solution en quatre groupes obtenue avec le critère de Ward pourrait être un artifice statistique, puisqu'elle est inférieure à 0.25.

L'élément `ASW` de l'objet `clustqual4` contient les deux variantes de la silhouette moyenne de chaque groupe pris séparément. Selon ces mesures, le groupe 3 est particulièrement mal défini puisque sa silhouette moyenne est négative.

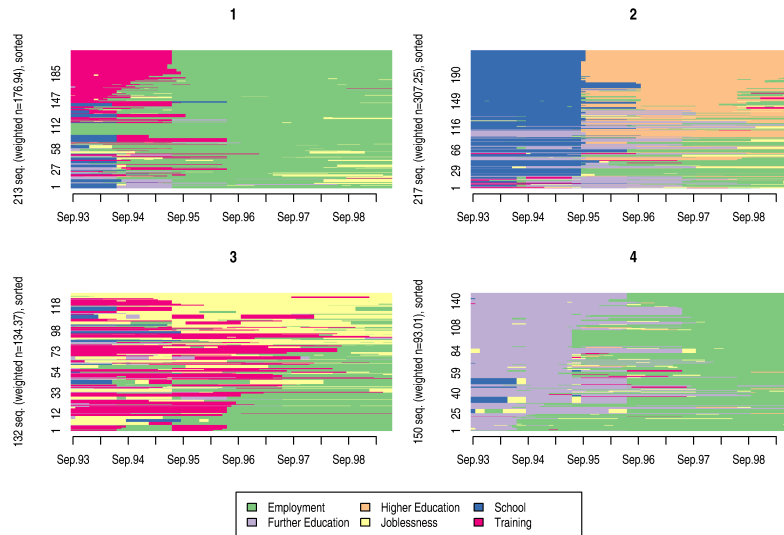
```
R> clustqual4$ASW
```

	ASW	ASWw
1	0.3621	0.3656
2	0.2313	0.2335
3	-0.0431	-0.0375
4	0.4184	0.4246

7.2. Utiliser la silhouette pour représenter les clusters

La silhouette peut être calculée séparément pour chaque séquence, ce qui permet d'identifier les séquences caractéristiques d'un regroupement (silhouette proche de un). La fonction `wcSilhouetteObs` calcule ces valeurs. Dans l'exemple ci-dessous, nous utilisons les silhouettes (avec la variante `measure="ASWw"`) pour ordonner les séquences dans des index-plots.

```
R> sil <- wcSilhouetteObs(mvaddist, clust4, weights=mvad$weight, measure="ASWw")
R> seqIplot(mvadseq, group=clust4, sortv=sil)
```



Les séquences les plus caractéristiques de chaque cluster sont représentées en haut de chaque graphique. Selon la définition de la silhouette, les séquences que nous appelons “caractéristiques” sont celles qui sont proches du centre de leur groupe tout en étant éloignées du groupe le plus proche. Dans le groupe 1 par exemple, la séquence caractéristique est d’accéder à l’emploi après deux ans d’apprentissage. Au contraire, les séquences au bas de chaque graphique sont mal représentées et/ou mal assignées. Ainsi, dans le groupe 3 par exemple, les séquences “école – apprentissage – emploi” sont plus proches d’un autre groupe (le 1 vraisemblablement) que de son propre groupe.

7.3. Choix d’une partition

Les mesures de la qualité d’une partition facilitent le choix de la meilleure partition parmi un ensemble de possibilités. On peut ainsi les utiliser pour identifier l’algorithme qui donne les meilleurs résultats. La fonction `wcKmedoids` utilisée pour PAM calcule directement ces valeurs qui sont stockées dans les éléments `stats` et `ASW` comme précédemment. Le code suivant affiche les mesures de qualités pour la partition identifiée à l’aide de PAM. Cette partition semble ainsi meilleure que celle obtenue avec Ward.

```
R> pamclust4$stats
```

PBC	HG	HGSD	ASW	ASWw	CH	R2	CHsq	R2sq	HC
0.6	0.8	0.8	0.4	0.4	198.2	0.5	534.6	0.7	0.1

Ces mesures permettent également de comparer des partitions avec un nombre de groupes différents. Seul le pseudo R^2 ne devrait pas être utilisé dans ce but, car il ne pénalise pas pour la complexité. Le calcul de la qualité de toutes

ces différentes possibilités s'avère vite laborieux. La fonction `as.clustrange` de la librairie **WeightedCluster** calcule automatiquement ces valeurs pour un ensemble de nombres de groupes issus d'une même procédure de regroupement hiérarchique (`wardCluster` dans notre exemple). Cette fonction requiert les arguments suivants : la matrice des dissimilarités (`diss`), les pondérations (optionnel, argument `weights`) ainsi que le nombre maximum de cluster que l'on entend conserver (`ncluster`). Dans l'exemple suivant, nous estimons la qualité de clustering pour les regroupements en 2, 3, ..., `ncluster = 20` groupes.

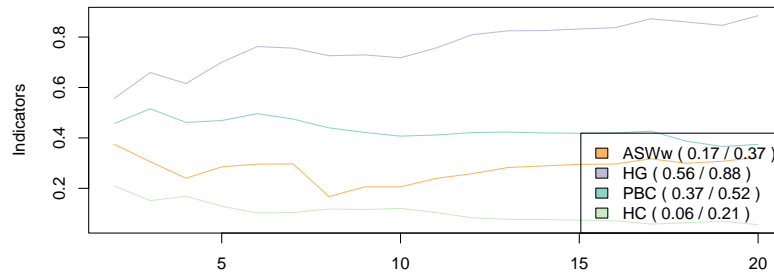
```
R> wardRange <- as.clustrange(wardCluster, diss=mvaddist, weights=mvad$weight, ncluster=20)
R> summary(wardRange, max.rank=2)
```

	1. N groups	1. stat	2. N groups	2. stat
PBC	3	0.5155	6	0.4964
HG	20	0.8847	17	0.8724
HGSD	20	0.8830	17	0.8707
ASW	2	0.3725	20	0.3042
ASWw	2	0.3742	20	0.3227
CH	2	234.1011	3	174.3984
R2	20	0.6972	19	0.6823
CHsq	2	469.5188	3	388.3765
R2sq	20	0.8630	19	0.8438
HC	20	0.0558	17	0.0584

La fonction `summary` présente le meilleur nombre de groupes selon chaque mesure de qualité ainsi que la valeur de ces statistiques. L'argument `max.rank` spécifie le nombre de partitions à afficher. Selon la mesure "Point Biserial Correlation" (PBC), la partition en six groupes est la meilleure des partitions, alors que pour l'indice "ASW" une solution en deux groupes paraît préférable. La valeur maximale identifiée pour ce dernier indice indique qu'il s'agit peut-être d'artifices statistiques (voir tableau 3). Rappelons que le pseudo- R^2 et sa version basée sur les distances élevées aux carrés donneront toujours un maximum pour le nombre de groupes le plus élevé, car ces statistiques ne peuvent pas diminuer.

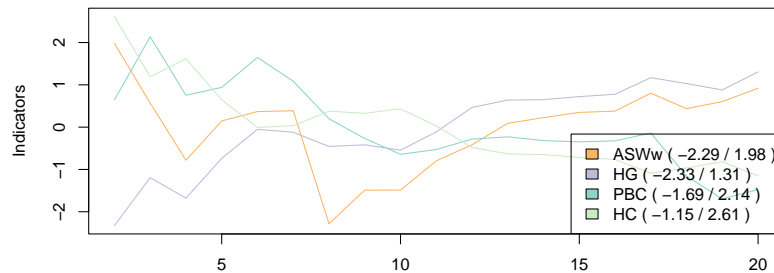
La présentation offerte par `summary` est utile pour comparer les résultats de deux procédures (voir ci-dessous). Toutefois, elle ne présente que deux ou trois solutions et il est souvent utile d'observer l'évolution de ces mesures pour identifier les points de décrochages et les partitions qui offrent le meilleur compromis entre plusieurs mesures. La fonction `plot` associée à l'objet retourné par `as.clustrange` représente graphiquement cette évolution. Au besoin, l'argument `stat` spécifie la liste des mesures à afficher ("`all`" les affiche toutes).

```
R> plot(wardRange, stat=c("ASWw", "HG", "PBC", "HC"))
```



La solution en six groupes est ici un maximum local pour les mesures “HC”, “PBC” et “HG”, ce qui en fait une bonne si l’on souhaite garder un nombre de groupe restreint. La lecture du graphique est parfois un peu difficile, car les valeurs moyennes de chaque mesure diffèrent. Pour pallier ce problème, l’argument `norm="zscore"` standardise les valeurs, ce qui permet de mieux identifier les maximums et minimums⁷. La figure ci-dessous facilite l’identification des partitions qui donnent de bons résultats. Les solutions en six et en 17 groupes paraissent bonnes dans notre cas de figure.

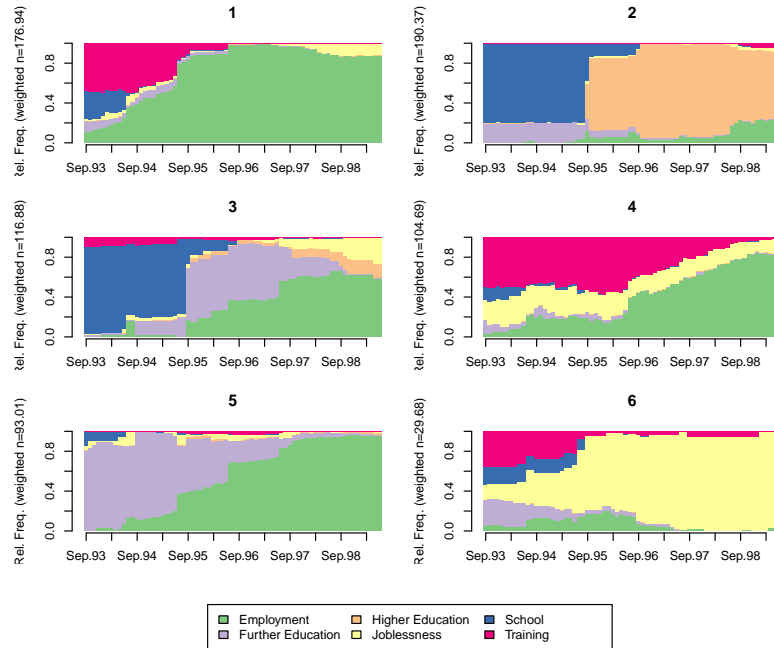
```
R> plot(wardRange, stat=c("ASWw", "HG", "PBC", "HC"), norm="zscore")
```



L’objet retourné par la fonction `as.clustrange` contient également un `data.frame` contenant toutes les partitions testées dans l’élément `clustering`. On peut ainsi utiliser directement `as.clustrange` plutôt que `cutree`. La solution en six groupes peut être affichée de la manière suivante.

```
R> seqdplot(mvadseq, group=wardRange$clustering$cluster6, border=NA)
```

7. on peut utiliser `norm="zscoremed"` pour une standardisation plus robuste basée sur la médiane.



La fonction `as.clustrange` accepte divers types d'arguments, dont l'ensemble des procédures de clustering disponibles dans la librairie `cluster`, même si ces dernières n'acceptent pas de pondération. Toutefois, il n'est pas possible de l'utiliser directement avec les algorithmes non hiérarchiques tels que PAM. Pour ce faire, on utilise la fonction `wcKMedRange` qui calcule automatiquement les partitions pour une série de valeurs de k (nombre de groupes). L'objet retourné est le même que celui présenté précédemment pour les procédures de regroupements hiérarchiques et l'on peut donc utiliser les mêmes présentations que précédemment. Cette fonction prend en paramètre une matrice de distance, `kvals` un vecteur contenant le nombre de groupes des différentes partitions à créer, et un vecteur de poids. Les arguments supplémentaires sont passés à `wcKMedoids`.

```
R> pamRange <- wcKMedRange(mvaddist, kvals=2:20, weights=mvad$weight)
R> summary(pamRange, max.rank=2)
```

	1. N groups	1. stat	2. N groups	2. stat
PBC	4	0.565	5	0.5323
HG	20	0.910	19	0.8936
HGSD	20	0.908	19	0.8920
ASW	2	0.389	20	0.3716
ASWw	20	0.390	2	0.3900
CH	2	252.008	3	205.7979
R2	20	0.712	19	0.7024
CHsq	4	534.585	2	483.1366
R2sq	20	0.887	19	0.8808
HC	20	0.045	19	0.0528

La présentation offerte par la fonction `summary` est particulièrement utile pour comparer les solutions de différentes procédures de clustering. On peut ainsi re-

marquer que les résultats de PAM sont généralement plus performants. Outre la solution en deux groupes, peut-être un peu trop simplificatrice, celle en quatre groupes paraît ici adaptée. Remarquons tout de même que ces partitions pourraient toujours être le fruit d’un artifice statistique puisque l’ASW est inférieur à 0.5.

Les différents outils que nous avons présentés nous ont permis de construire une typologie des séquences. Pour ce faire, nous avons testé divers algorithmes et nombres de groupes avant de retenir une partition en quatre groupes créée à l’aide de la méthode PAM. D’une manière générale, nous suggérons aux lecteurs de tester un plus grand nombre d’algorithmes que ce que nous avons fait ici. Lesnard (2006) suggère par exemple d’utiliser la méthode “average” ou la méthode “flexible” (voir tableau 1). Les outils présentés permettent de faire ces comparaisons.

7.4. Nommer les clusters

Une fois la typologie créée, il est d’usage de nommer les types obtenus afin de rendre leur interprétation plus facile. La fonction `factor` permet de créer une variable catégorielle. Pour ce faire, on spécifie la variable des types (ici `pamclust4$clustering`), les valeurs que prend cette variable avec l’argument `levels` (on peut retrouver ces valeurs dans les graphiques précédents), ainsi que les labels que l’on souhaite attribuer à chacun de ces types avec l’argument `labels`. Les `labels` doivent être spécifiés dans le même ordre que l’argument `levels` de sorte que la première entrée de `levels` (ici 66) corresponde à la première entrée de `labels` (ici “Apprentissage-Emploi”).

```
R> mvad$pam4 <- factor(pamclust4$clustering, levels=c(66, 467, 607, 641), labels=c("Appr.-E
```

La fonction `seqclustname` permet de nommer automatiquement les groupes en utilisant leur médioïde. On doit spécifier l’objet séquence (argument `seqdata`), le clustering (argument `group`), la matrice de distance (argument `diss`). Si `weighted=TRUE` (par défaut), les poids de l’objet `seqdata` sont utilisés pour trouver les médioïdes. Finalement, l’option `perc=TRUE` ajoute le pourcentage d’observation dans chaque groupe aux labels.

```
R> mvad$pam4.auto <- seqclustname(mvadseq, pamclust4$clustering, mvaddist)
R> table( mvad$pam4.auto, mvad$pam4)
```

	Appr.-Empl.	Ecole-Empl.	Ecole sup.	Sans empl.
TR/22-EM/48	189	0	0	0
SC/10-FE/12-EM/48	0	307	0	0
SC/25-HE/45	0	0	160	0
TR/22-JL/48	0	0	0	56

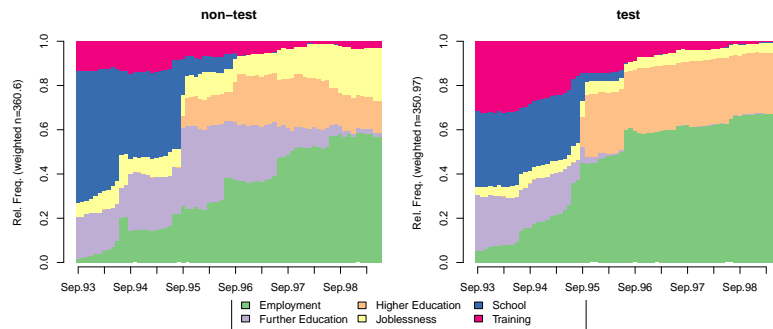
Une fois les clusters nommés, la typologie est construite. Avant de conclure, nous revenons plus en détail sur la question de la simplification induite par l’analyse en clusters et les biais que celle-ci peut introduire dans l’analyse.

Nous illustrons cette question en présentant le calcul des liens entre typologie et facteurs explicatifs et en montrant comment ceci peut biaiser les résultats.

8. Mettre en lien trajectoires-types et facteurs explicatifs

Beaucoup de questions de recherche des sciences sociales mettent en lien des trajectoires (ou des séquences) avec des facteurs explicatifs. Ainsi, on peut se demander si les trajectoires professionnelles des hommes diffèrent significativement de celles des femmes. De manière similaire, [Widmer *et al.* \(2003\)](#) cherchent à mettre en évidence l'apparition de nouvelles trajectoires de construction de la vie familiale. Pour ce faire, on met généralement en lien une typologie des séquences familiales avec la cohorte des individus à l'aide de test du khi-carré ou de régression logistique ([Abbott and Tsay 2000](#)). Dans cette section, nous présentons cette technique ainsi que les dangers qu'elle comporte pour l'analyse. Supposons que l'on cherche à mesurer les liens entre la variable `test` que nous avons créée pour l'occasion⁸ et nos trajectoires. Cette variable prend deux modalités, “test” et “non-test”, et regroupe les séquences de la manière suivante.

```
R> seqdplot(mvadseq, group=mvad$test, border=NA)
```



Les individus “non-test” semblent notamment avoir une plus forte probabilité de connaître une période sans emploi. Est-ce que cette différence est significative? On peut faire un test du khi-carré entre la variable `test` et `pam4` (qui reprend nos types) de la manière suivante pour des données pondérées. On commence par construire un tableau croisé avec la fonction `xtabs`⁹. Cette fonction prend en paramètre une formule, dont le côté gauche de l'équation fait référence aux pondérations et le côté droit liste les facteurs qui forment le tableau croisé. L'argument `data` spécifie où il faut chercher les variables qui apparaissent dans la formule. On calcule ensuite un test du khi-carré sur ce tableau.

```
R> tb <- xtabs(weight~test+pam4, data=mvad)
R> chisq.test(tb)
```

8. Le détail de la création de cette variable est donné dans l'annexe D.

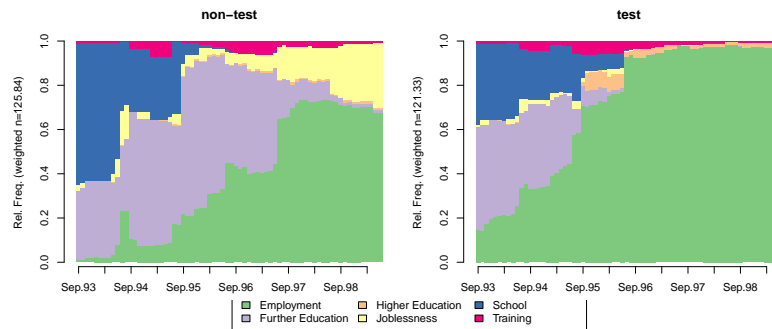
9. Pour des données non pondérées, on peut utiliser la fonction `table`.

```
Pearson's Chi-squared test

data:  tb
X-squared = 0.02, df = 3, p-value = 1
```

Le résultat est non significatif, ce qui signifie que selon cette procédure les trajectoires ne diffèrent pas selon la variable `test`. Quel est le problème ? Est-ce vraiment le cas ? Non, le problème vient de la simplification de l'analyse en clusters. En utilisant la typologie dans le test du khi-carré, on fait implicitement l'hypothèse que cette typologie est suffisante pour décrire la complexité des trajectoires, or ce n'est pas le cas ici. En effet, la variable `test` explique la variation des trajectoires *au sein des types*. À titre d'exemple, le graphique suivant montre les différences de trajectoires selon la variable `test` pour les individus classés dans le type "École-Emploi". Les individus "non-test" semblent ainsi avoir un risque plus élevé de connaître un épisode "Sans emploi". La variabilité au sein de ce type ne semble donc pas négligeable.

```
R> EcoleEmploi <- mvad$pam4=="Ecole-Empl."
R> seqdplot(mvadseq[EcoleEmploi, ], group=mvad$test[EcoleEmploi], border=NA)
```



Revenons sur l'hypothèse sous-jacente dont nous avons parlé et tâchons de l'explicitier. En utilisant la typologie, on assigne à chaque séquence son type. On ignore ainsi l'écart entre la trajectoire réellement suivie par un individu et son type. Une des justifications possibles de cette opération serait de dire que les types obtenus correspondent aux modèles qui ont effectivement généré les trajectoires. Les écarts entre les trajectoires suivies et ces modèles (c'est-à-dire les types) seraient ainsi assimilables à une forme de terme d'erreur aléatoire ne contenant aucune information pertinente. Cette méthode revient donc à faire l'hypothèse que les trajectoires sont générées par des modèles établis et clairement distincts les uns des autres. De plus, nous faisons implicitement l'hypothèse que nous avons effectivement réussi à retrouver les vrais modèles à l'aide de l'analyse en clusters.

Parallèlement, on fait également une deuxième hypothèse, à savoir que les types obtenus sont également différents les uns des autres¹⁰. Or, ce n'est pas le cas.

10. Les cartes de Kohonen permettent de visualiser ces distances entre types de trajectoires (Rousset and Giret 2007).

Les types “Apprentissage-Emploi” et “Ecole-Emploi” sont plus proches, car ils partagent une fin de trajectoires identiques. Quant à eux, les types “école supérieure” et “sans emploi” sont particulièrement éloignés.

Ces deux hypothèses sont des hypothèses fortes. On postule l’existence de modèles qui ont effectivement généré les trajectoires, ce qui est discutable d’un point de vue sociologique. Dans la lignée du paradigme des parcours de vie, on peut ainsi penser que les individus sont soumis à des influences et des contraintes diverses qui participent, chacune à leur manière, à la construction de la trajectoire (ou d’une partie de la trajectoire). On est bien loin de la recherche de modèles de trajectoires.

Encore une fois, ces hypothèses peuvent s’avérer pertinentes si les groupes obtenus sont très homogènes et très différents les uns des autres. Une situation où la silhouette moyenne devrait être relativement élevée ($ASW > 0.7$ par exemple). Mais ce n’est pas le cas ici, puisque la silhouette moyenne est inférieure à 0.5.

La solution à ce problème consiste à utiliser l’analyse de dispersion (Studer *et al.* 2011). Ce type d’analyse permet de mesurer la force du lien en fournissant un pseudo- R^2 , c’est-à-dire la part de la variation expliquée par une variable, ainsi que la significativité de l’association. On s’affranchit ainsi de l’hypothèse des modèles de trajectoires en calculant directement le lien, sans clustering préalable. On trouvera dans Studer *et al.* (2011) une introduction à sa mise en pratique dans R ainsi qu’une présentation générale de la méthode. Nous n’offrons donc ici qu’un survol destiné à étayer notre propos.

Brièvement, un test bivarié de l’association entre les séquences et la variable `test` peut être calculé avec la fonction `dissassoc` disponible dans la librairie **TraMineR**. On lit les résultats qui nous intéressent ici sur la ligne Pseudo R^2 . On remarque ainsi que la variable `test` permet d’expliquer 3.6% de la variabilité des trajectoires et la p -valeur est largement significative.

```
R> set.seed(1)
R> dsa <- dissassoc(mvaddist, mvad$test, weights=mvad$weight, weight.permutation="diss", R=500)
R> print(dsa$stat)
```

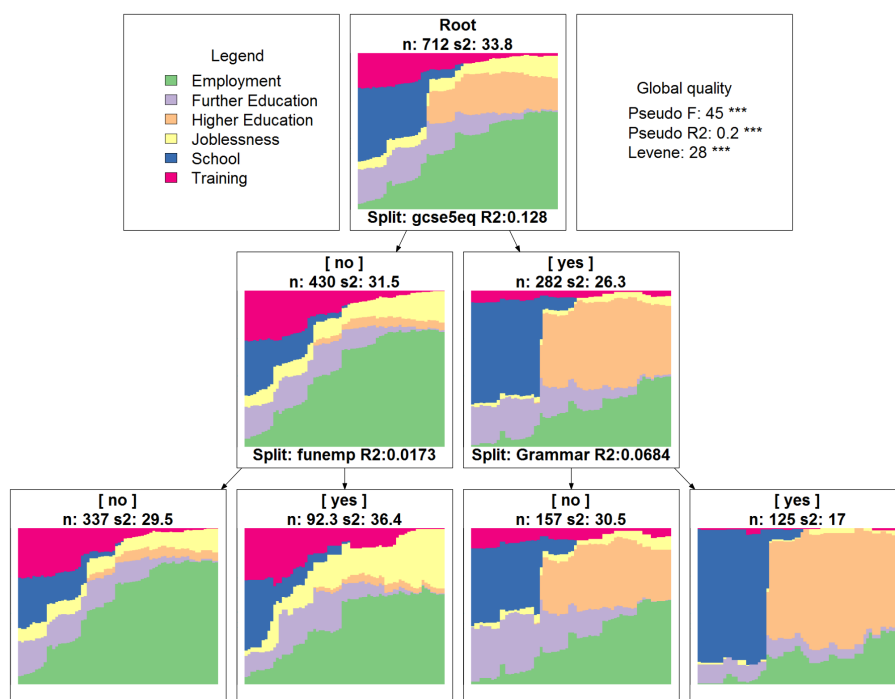
	t0	p.value
Pseudo F	25.8285	0.0002
Pseudo Fbf	25.9594	0.0002
Pseudo R2	0.0351	0.0002
Bartlett	2.4336	0.8050
Levene	20.0631	0.0006

On peut inclure plusieurs variables dans l’analyse à l’aide d’un arbre de régression sur les séquences¹¹. Pour ce faire, on utilise la fonction `seqtree`. Cette analyse met en évidence les variables les plus importantes ainsi que leurs interactions. Ici, la variable `gcse5eq` (bon résultats à la fin de l’école obligatoire) à l’effet le plus important. Pour ceux qui ont eu de bon résultats, avoir suivi une école obligatoire de type **Grammar** favorise l’accès à l’école supérieure. Par

11. Il est également possible d’analyser les effets conjoints avec une approche multifacteur.

contre, pour ceux qui ont eu de mauvais résultats, c'est d'avoir un père au chômage qui est le plus significatif. On semble alors avoir plus de chance de se retrouver au sans emploi.

```
R> tree <- seqtree(mvadseq~gcse5eq+Grammar+funemp, data=mvad, diss=mvaddist, weight.permuta
R> seqtreedisplay(tree, type="d", border=NA)
```



L'analyse de dispersion conduit à un changement de paradigme et à s'affranchir du concept de modèle de trajectoire. Plutôt que de se baser sur la recherche de modèles, nous considérons que les trajectoires s'insèrent dans un contexte qui influence à sa manière la construction de la trajectoire. En d'autres termes, nous cherchons à comprendre dans quelle mesure la variabilité interindividuelle est expliquée par un contexte tout en rendant compte de la diversité des chemins empruntés. D'un point de vue conceptuel, les hypothèses sous-jacentes aux méthodes à l'analyse de dispersion sont très proches des principes mis en avant par le paradigme des parcours de vie. [Elder \(1999\)](#) insiste ainsi sur la nécessité d'insérer les parcours dans des temps et des lieux (le contexte) tout en préservant la variabilité interindividuelle et l'internationalité des acteurs.

À notre sens, la construction de typologie de séquence est une méthode puissante qui a l'avantage d'offrir un point de vue descriptif sur les séquences en réduisant la complexité de l'analyse. Toutefois, son utilisation en lien avec des méthodes inférentielles doit être faite avec prudence, puisqu'elle peut conduire à des conclusions trompeuses, comme illustré avec la variable `test`.

9. Conclusion

Ce manuel poursuivait un double but : présenter la librairie **WeightedCluster** et proposer un guide pas à pas à la création de typologie de séquences pour les sciences sociales. Cette librairie permet notamment de représenter graphiquement les résultats d’une analyse en clusters hiérarchique, de regrouper les séquences identiques afin d’analyser un nombre de séquences plus important ¹², de calculer un ensemble de mesure de qualité d’une partition ainsi qu’une version optimisée de l’algorithme PAM prenant en compte les pondérations. La librairie offre également des procédures pour faciliter le choix d’une solution de clustering particulière et définir le nombre de groupes.

Outre les méthodes, nous avons également discuté de la construction de typologie de séquences en sciences sociales. Nous avons ainsi argumenté que ces méthodes offrent un point de vue descriptif important sur les séquences. L’analyse en clusters permet de faire ressortir des patterns récurrents et/ou des “séquences idéales-typiques” (Abbott and Hrycak 1990). La recherche de tels patterns est une question importante dans plusieurs problématiques de sciences sociales (Abbott 1995). Elle permet notamment de mettre en lumière les contraintes légales, économiques ou sociales qui encadrent la construction des parcours individuels. Comme le notent Abbott and Hrycak (1990), si les séquences types peuvent résulter de contrainte que l’on redécouvre, ces séquences typiques peuvent également agir sur la réalité en servant de modèles aux acteurs qui anticipent leur propre futur. Ces différentes possibilités d’interprétations font de la création de typologie un outil puissant.

Toutes les analyses en cluster produisent des résultats, quelle qu’en soit la pertinence (Levine 2000). Il est donc nécessaire de discuter de sa qualité afin de préciser la portée des résultats et de ne pas faire de généralisation abusive. À notre sens, cette étape est trop souvent absente des analyses en cluster. Dans notre cas, cette qualité était faible, ce qui est courant en analyse de séquences. Avec une qualité plus élevée (c’est-à-dire $ASW > 0.7$ par exemple), on pourra se montrer plus affirmatif, car la partition est vraisemblablement le reflet d’une structure forte identifiée dans les données.

Si l’outil est puissant, il est également risqué. En donnant un nom unique à chaque groupe, on tend à supprimer de l’analyse la diversité des situations rencontrées à l’intérieur de chaque groupe. À titre d’exemple, nous avons noté que la durée des périodes sans emploi varie sensiblement au sein du groupe que nous avons nommé “Sans emploi” et que l’état se rencontre également dans d’autres groupes. Cette simplification fait sens si le but est de faire émerger les patterns récurrents dans une optique descriptive.

Toutefois, les typologies ne devraient pas être utilisées dans une démarche explicative. En effet, ceci revient à postuler l’existence de modèles clairement définis qui auraient effectivement généré les trajectoires et que l’on aurait identifiés grâce à l’analyse en clusters. Outre le fait que cette démarche peut conduire à des conclusions trompeuses si ces hypothèses ne se vérifient pas, ces hypothèses sont discutables d’un point de vue sociologique. Dans la lignée du paradigme

12. Le détails de cette procédure est explicité dans l’annexe A.

des parcours de vie, on peut penser que les individus sont soumis à des influences et des contraintes diverses qui participent, chacune à leur manière, à la construction de la trajectoire (ou d'une partie de la trajectoire). On est ainsi bien loin de la recherche de modèles de trajectoires clairement définis.

Références

- Abbott A (1995). “Sequence Analysis : New Methods for Old Ideas.” *Annual Review of Sociology*, **21**, 93–113.
- Abbott A, Hrycak A (1990). “Measuring Resemblance in Sequence Data : An Optimal Matching Analysis of Musicians’ Careers.” *American Journal of Sociology*, **96**(1), 144–185.
- Abbott A, Tsay A (2000). “Sequence Analysis and Optimal Matching Methods in Sociology, Review and Prospect.” *Sociological Methods and Research*, **29**(1), 3–33. (With discussion, pp 34–76).
- Aisenbrey S, Fasang AE (2010). “New Life for Old Ideas : The “Second Wave” of Sequence Analysis Bringing the “Course” Back Into the Life Course.” *Sociological Methods and Research*, **38**(3), 430–462.
- Belbin L, Faith DP, Milligan GW (1992). “A Comparison of Two Approaches to beta-Flexible Clustering.” *Multivariate Behavioral Research*, **3**(3), 417–433.
- Calinski RB, Harabasz J (1974). “A Dendrite Method for Cluster Analysis.” *Communications in Statistics*, **3**, 1–27.
- Elder GH (1999). *Children of the Great Depression*. Westview Press, Boulder.
- Fernández A, Gómez S (2008). “Solving Non-Uniqueness in Agglomerative Hierarchical Clustering Using Multidendrograms.” *Journal of Classification*, **25**, 43–65. ISSN 0176-4268. 10.1007/s00357-008-9004-x, URL <http://dx.doi.org/10.1007/s00357-008-9004-x>.
- 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. doi:<https://doi.org/10.18637/jss.v040.i04>.
- Hennig C, Liao TF (2010). “Comparing latent class and dissimilarity based clustering for mixed type variables with application to social stratification.” *Technical report*, Department of Statistical Science, UCL, Department of Sociology, University of Illinois.
- Hollister M (2009). “Is Optimal Matching Suboptimal?” *Sociological Methods Research*, **38**(2), 235–264. doi:[10.1177/0049124109346164](https://doi.org/10.1177/0049124109346164).
- Hubert L, Arabie P (1985). “Comparing Partitions.” *Journal of Classification*, **2**, 193–218.

- Hubert L, Levin J (1976). “A general statistical framework for assessing categorical clustering in free recall.” *Psychological Bulletin*, **83**, 1072–1080.
- Kaufman L, Rousseeuw PJ (1990). *Finding groups in data. an introduction to cluster analysis*. Wiley, New York.
- Lesnard L (2006). “Optimal Matching and Social Sciences.” *Manuscript*, Observatoire Sociologique du Changement (Sciences Po and CNRS), Paris.
- Lesnard L (2010). “Setting Cost in Optimal Matching to Uncover Contemporaneous Socio-Temporal Patterns.” *Sociological Methods Research*, **38**(3), 389–419. doi:10.1177/0049124110362526. URL <http://smr.sagepub.com/cgi/content/abstract/38/3/389>.
- Levine J (2000). “But What Have You Done For Us Lately.” *Sociological Methods & Research*, **29** (1), pp. 35–40.
- Maechler M, Rousseeuw P, Struyf A, Hubert M (2005). “Cluster Analysis Basics and Extensions.” Rousseeuw et al provided the S original which has been ported to R by Kurt Hornik and has since been enhanced by Martin Maechler : speed improvements, silhouette() functionality, bug fixes, etc. See the ‘Changelog’ file (in the package source).
- 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.
- Milligan G (1989). “A study of the beta-flexible clustering method.” *Multivariate Behavioral Research*, **24**(2), 163–176.
- Milligan G, Cooper M (1985). “An examination of procedures for determining the number of clusters in a data set.” *Psychometrika*, **50**(2), 159–179. URL <http://ideas.repec.org/a/spr/psycho/v50y1985i2p159-179.html>.
- Milligan GW, Cooper MC (1987). “Methodology Review : Clustering Methods.” *Applied Psychological Measurement*, **11**(4), 329–354. doi:10.1177/014662168701100401.
- Müllner D (2011). *fastcluster : Fast hierarchical clustering routines for R and Python*. Version 1.1.3, URL <http://math.stanford.edu/~muellner>.
- Oksanen J, Blanchet FG, Kindt R, Legendre P, Minchin PR, O’Hara RB, Simpson GL, Solymos P, Stevens MHH, Wagner H (2012). *vegan : Community Ecology Package*. R package version 2.0-3, URL <http://CRAN.R-project.org/package=vegan>.
- Reynolds A, Richards G, de la Iglesia B, Rayward-Smith V (2006). “Clustering Rules : A Comparison of Partitioning and Hierarchical Clustering Algorithms.” *Journal of Mathematical Modelling and Algorithms*, **5**, 475–504. ISSN 1570-1166. 10.1007/s10852-005-9022-1, URL <http://dx.doi.org/10.1007/s10852-005-9022-1>.

Rousset P, Giret JF (2007). “Classifying qualitative time series with SOM : the typology of career paths in France.” In *Proceedings of the 9th international work conference on Artificial neural networks*, IWANN’07, pp. 757–764. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-540-73006-4. URL <http://dl.acm.org/citation.cfm?id=1768409.1768513>.

Shalizi C (2009). “Distances between Clustering, Hierarchical Clustering.” *Lectures notes*, Carnegie Mellon University.

Struyf A, Hubert M, Rousseeuw P (1997). “Clustering in an Object-Oriented Environment.” *Journal of Statistical Software*, **1**(4), 1–30. ISSN 1548-7660. URL <http://www.jstatsoft.org/v01/i04>.

Studer M, Ritschard G, Gabadinho A, Müller NS (2011). “Discrepancy Analysis of State Sequences.” *Sociological Methods and Research*, **40**(3), 471–510. doi: [10.1177/0049124111415372](https://doi.org/10.1177/0049124111415372).

Widmer ED, Levy R, Pollien A, Hammer R, Gauthier JA (2003). “Entre standardisation, individualisation et sexuation : une analyse des trajectoires personnelles en Suisse.” *Revue suisse de sociologie*, **29**(1), 35–67.

A. Agréger les séquences identiques

Les algorithmes que nous avons présentés prennent tous en compte la pondération des observations. Ceci permet de regrouper les observations identiques en leur donnant une pondération plus élevée. On peut ensuite effectuer l’analyse en clusters sur ces données regroupées, ce qui diminue considérablement temps de calcul et la mémoire utilisée¹³. On termine l’analyse en “désagrégeant” les données afin de réintégrer la typologie dans les données initiales.

Ces différentes opérations sont réalisées aisément avec la fonction `wcAggregateCases` de la librairie **WeightedCluster**. Cette fonction identifie les cas identiques afin de les regrouper. Dans un deuxième temps, l’objet retourné permet également de réaliser l’opération inverse, c’est-à-dire de désagréger les données.

Reprenons l’exemple que nous avons utilisé depuis le début de ce manuel. Le code suivant permet d’identifier les séquences identiques. La fonction `wcAggregateCases` prend deux paramètres : un `data.frame` (ou une matrice) qui contient les cas à agréger ainsi qu’un vecteur de poids optionnel.

```
R> ac <- wcAggregateCases(mvad[, 17:86], weights=mvad$weight)
R> ac
```

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

13. La quantité de mémoire nécessaire évolue de manière quadratique.

L'objet renvoyé par la fonction `wcAggregateCases` (ac ici) donne quelques informations de base sur le regroupement. On peut ainsi noter que le jeu de donnée initial comporte 712 observations, mais seulement 490 séquences différentes. L'objet retourné contient trois éléments particulièrement intéressants.

- `aggIndex` : index des objets uniques.
- `aggWeights` : Nombre de fois (éventuellement pondéré) que chaque objet unique de `aggIndex` apparaît dans les données.
- `disaggIndex` : index des objets initiaux dans la liste des objets uniques. Cette information permet de désagréger les données. On donnera plus tard un exemple d'utilisation.

À l'aide de ces informations, nous pouvons créer un objet `uniqueSeq` des séquences uniques pondérées par le nombre de fois qu'elles apparaissent dans les données. Dans le code suivant, `ac$aggIndex` permet de sélectionner les séquences uniques et le vecteur `ac$aggWeights` contient la pondération de chacune de ces séquences.

```
R> uniqueSeq <- seqdef(mvad[ac$aggIndex, 17:86], alphabet = mvad.alphabet,
  states = mvad.scodes, labels = mvad.labels, weights=ac$aggWeights)
```

À partir de là, nous pouvons calculer différentes solutions de clustering. Ici, nous calculons la matrice des distances avant d'utiliser cette information pour la fonction `wcKMedoids`. Comme précédemment, nous utilisons ici le vecteur `ac$aggWeights` pour utiliser les pondérations.

```
R> mvaddist2 <- seqdist(uniqueSeq, method="OM", indel=1.5, sm=subm.custom)
R> pamclust4ac <- wcKMedoids(mvaddist2, k=4, weights=ac$aggWeights)
```

Une fois le clustering réalisé, l'information contenue dans `ac$disaggIndex` permet de revenir en arrière. On peut par exemple ajouter la typologie dans le `data.frame` original (non agrégé) à l'aide du code suivant. Le vecteur `pamclust4ac$clustering` contient l'appartenance de chaque séquence unique aux clusters. En utilisant l'indice `ac$disaggIndex`, on revient au jeu de donnée original, c'est-à-dire que l'on obtient l'appartenance de chaque séquence (non unique) aux clusters.

```
R> mvad$acpam4 <- pamclust4ac$clustering[ac$disaggIndex]
```

Le tableau suivant donne la répartition des cas originaux entre la typologie que nous avons obtenue à partir des cas désagrégés (variable `pam4`) et celle obtenue à partir des données agrégées (variable `acpam4`). On remarquera que les deux solutions contiennent les mêmes cas, seuls les labels diffèrent.

```
R> table(mvad$pam4, mvad$acpam4)
```

	28	87	414	444
Appr.-Empl.	0	0	189	0
Ecole-Empl.	0	307	0	0
Ecole sup.	160	0	0	0
Sans empl.	0	0	0	56

L'agrégation de cas identiques est une fonctionnalité très utile pour les grands jeux de données. Il est fréquent que l'on ne puisse pas calculer l'ensemble des distances pour cause de mémoire insuffisante. Dans de tels cas, l'utilisation de `wcAggregateCases` pourrait bien résoudre le problème.

B. Notes sur les performances

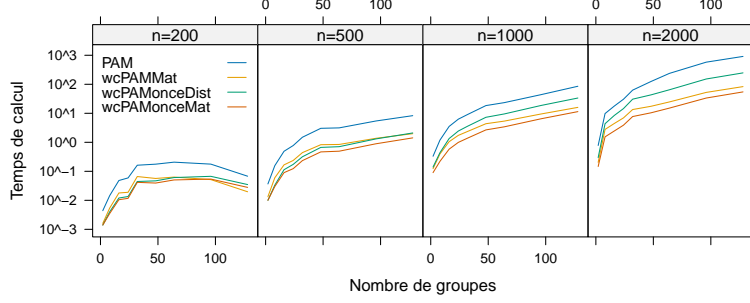
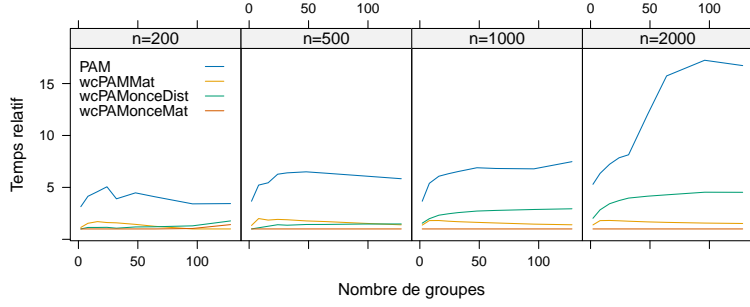
Les algorithmes de partitionnement autour de centres mobiles disponibles dans la librairie **WeightedCluster** sont hautement optimisés. En interne, la librairie propose plusieurs variantes de l'algorithme PAM. Le choix entre ces variantes dépend du type de l'objet distance passé à la fonction ainsi que de l'argument méthode.

L'argument `diss` peut être une matrice de distance ou un objet `dist`. Dans le premier cas, chaque distance est enregistrée à double ce qui peut rapidement poser des problèmes de quantité de mémoire disponible, mais l'algorithme est généralement plus rapide (voir les comparaisons de performances ci-dessous). Si l'argument est de type `dist`, seul le triangle inférieur de la matrice de distance est stocké en mémoire, mais ce gain se fait au détriment de la rapidité de l'algorithme.

Contrairement à l'algorithme PAM disponible dans la librairie **cluster**, la matrice des distances n'est pas copiée en interne, il est donc possible de réaliser un clustering d'un nombre nettement plus important d'objets avant d'atteindre les limites mémoires de la machine.

L'argument `method` spécifie l'algorithme utilisé. Deux versions sont disponibles : la version originale de l'algorithme PAM et PAMonce. L'algorithme "PAMonce" implémente les optimisations proposées par [Reynolds et al. \(2006\)](#) qui consiste à n'évaluer qu'une seule fois (plutôt que n fois dans la version originale) le coût de la suppression d'un médoïde. Nous avons également inclus dans cet algorithme une deuxième optimisation qui consiste à ne pas évaluer le gain du remplacement d'un médoïde par un objet donné si la distance entre ceux-ci est nulle. Cette optimisation est cohérente avec la définition mathématique d'une mesure de distance selon laquelle deux objets sont identiques si, et seulement si, leur distance est nulle. Cette optimisation ne fait sens que dans la mesure où les objets à regrouper contiennent des doublons et surtout si la mesure de dissimilarité utilisée respecte cette condition. Notons que les mesures de dissimilarités la respectent généralement.

Afin de mesurer l'impact sur les performances de ces optimisations, nous avons conduit plusieurs simulations. Dans celles-ci, il s'agissait de regrouper en $k \in (2, 8, 16, 24, 32, 48, 64, 96, 128)$ groupes un ensemble de $n \in (200, 500, 1000, 2000)$ observations dont les coordonnées x et y ont été générées aléatoirement selon une distribution uniforme. La figure 1 présente l'évolution du temps de calcul (sur une échelle logarithmique) en fonction des valeurs de n et k . Quant à elle, la figure 2 présente l'évolution du temps total relatif, c'est-à-dire divisé par le temps mis par l'algorithme le plus rapide pour cette solution, en fonction des mêmes paramètres.

FIGURE 1 – Évolution du temps de calcul en fonction de n et k FIGURE 2 – Évolution du temps relatif en fonction de n et k 

Les solutions proposées par **WeightedCluster** sont plus rapides et les différences augmentent à mesure que k et n augmentent. L'utilisation d'une matrice de distances plutôt que l'objet `dist` permet d'accélérer les temps de calcul. Le gain de mémoire se fait donc bien au détriment du temps de calcul. Il en va de même pour les optimisations de "PAMonce" qui apporte un gain significatif. Par rapport à la librairie **cluster**, les gains sont particulièrement importants (15 fois environ) lorsque n est plus grand que 1'000 et que k est grand.

Rappelons encore que, si les données contiennent des cas identiques, les gains sont potentiellement encore plus importants, car ces cas peuvent être regroupés en utilisant `wcAggregateCases`. Cette opération réduit considérablement la mémoire nécessaire et le temps de calcul.

C. Détails des mesures de qualité

C.1. Average Silhouette Width (ASW)

Originellement proposée par [Kaufman and Rousseeuw \(1990\)](#), cet indice se base sur une notion de cohérence de l'assignation d'une observation à un groupe donné qui est mesurée en mettant en parallèle la distance moyenne pondérée, notée a_i , d'une observation i aux autres membres de son groupe et la distance

moyenne pondérée au groupe le plus proche, notée b_i .

Formellement, ces distances moyennes sont définies de la manière suivante. Soit k le groupe de l'observation i , W_k la somme des pondérations des observations appartenant au groupe k , w_i le poids de l'observation i et ℓ un des autres groupes, la silhouette d'une observation se calcule de la manière suivante.

$$a_i = \frac{1}{W_k - 1} \sum_{j \in k} w_j d_{ij} \quad (1)$$

$$b_i = \min_{\ell} \frac{1}{W_{\ell}} \sum_{j \in \ell} w_j d_{ij} \quad (2)$$

$$s_i = \frac{b_i - a_i}{\max(a_i, b_i)} \quad (3)$$

L'équation 1 suppose que l'unité des pondérations corresponde à une observation. Cette hypothèse découle de l'utilisation de $W_k - 1$ dans l'équation 1. Cette hypothèse ne pose aucun problème quand les pondérations résultent d'agrégation (voir l'annexe A) ou si les données ne sont pas pondérées. Lorsque certains w_i ou si W_k sont inférieurs à un cependant, les valeurs a_i sont indéfinies et la silhouette ne peut donc être interprétée. Ces cas de figure sont fréquents quand lorsque les pondérations visent à corriger la représentativité de données d'enquêtes (comme c'est le cas dans notre base de donnée exemple). Pour pallier ce problème, nous proposons de remplacer a_i par aw_i qui se calcule de la manière suivante :

$$aw_i = \frac{1}{W_k} \sum_{j \in k} w_j d_{ij} \quad (4)$$

la valeur aw_i peut s'interpréter de deux manières. Elle correspond à la distance entre l'observation et son propre groupe, en considérant que toute les observations du groupes doivent être utilisée pour définir le groupe. Dans la formulation originale, l'observation est retirée du groupe pour calculer cette distance. La deuxième interprétation consiste à dire que l'unité des pondérations est aussi petite que possible, c'est-à-dire qu'elle tend vers zéro.

Dans les deux cas de figure, l'indice finalement utilisé correspond à la moyenne pondérée des silhouettes s_i . Cette valeur est retournée dans l'élément **stats**. La moyenne pondérée des silhouettes de chaque groupe est donnée dans l'élément **ASW** et mesure séparément la cohérence de chaque groupe.

C.2. C index (HC)

Développé par [Hubert and Levin \(1976\)](#), cet indice met en parallèle la partition obtenue et la meilleure partition que l'on aurait pu obtenir avec ce nombre de groupe et cette matrice de distance. L'indice varie entre 0 et 1, une petite valeur indiquant une bonne partition des données. Plus formellement, il est défini de la manière suivante. Soit S la somme des distances intragroupes pondérée par

le produit des poids de chaque observation, W la somme des poids des distances intragroupes, S_{min} la somme pondérée des W plus petites distances, et S_{max} la somme pondérée des W plus grandes distances :

$$C_{index} = \frac{S - S_{min}}{S_{max} - S_{min}} \quad (5)$$

C.3. “Hubert’s Gamma” (HG, HGSD) et “Point Biserial Correlation” (PBC)

Ces indices mesurent la capacité d’une partition à reproduire la matrice des distances en calculant l’association entre la distance originale d et une deuxième matrice d_{bin} prenant la valeur 0 pour les observations classées dans la même partition et 1 sinon. Pour utiliser les pondérations, nous utilisons W^{mat} la matrice du produit des pondérations $W_{ij}^{mat} = w_i \cdot w_j$.

Hubert and Arabie (1985) proposent de mesurer cette association en utilisant le Gamma de Goodman et Kruskal (HG) ou le D de Somers (HGSD) afin de tenir compte des égalités dans la matrice des distances. **WeightedCluster** utilise les formules suivantes basées sur le fait que d_{bin} ne prend que deux valeurs différentes. Soit T le tableau croisé pondéré entre les valeurs de d en ligne (ℓ lignes) et de d_{bin} en deux colonnes (0 ou 1) calculé en utilisation les pondérations W^{mat} , le nombre de paires concordantes C , celui de paires discordantes D , celui des égalités sur d E sont définis de la manière suivante :

$$\begin{aligned} C &= \sum_{i=1}^{\ell} \sum_{i'=1}^{i-1} T_{i'0} & D &= \sum_{i=1}^{\ell} \sum_{i'=1}^{i-1} T_{i'1} & E &= \sum_{i=1}^{\ell} T_{i0} + T_{i1} \\ HG &= \frac{C - D}{C + D} & HGSD &= \frac{C - D}{C + D + E} \end{aligned}$$

Hennig and Liao (2010) proposent d’utiliser la corrélation de Pearson plutôt, solution également connue sous le nom de “Point Biserial Correlation” (PBC équation 6) (Milligan and Cooper 1985). Soit s_d et $s_{d_{bin}}$ l’écart-type pondéré par W^{mat} de d et d_{bin} respectivement, $s_{d,d_{bin}}$ la covariance pondérée par W^{mat} entre d et d_{bin} , cette corrélation est calculée de la manière suivante :

$$PBC = \frac{s_{d,d_{bin}}}{s_{d_{bin}} \cdot s_{d_{bin}}} \quad (6)$$

C.4. CH index (CH, CHsq, R2, R2sq)

Calinski and Harabasz (1974) proposent d’utiliser la statistique F de l’analyse de variance en le calculant à partir des distances euclidiennes aux carrés. Sur les mêmes bases, on peut calculer un pseudo R^2 , soit la part de la variabilité expliquée par une partition. Studer *et al.* (2011) proposent une extension aux données pondérées de ces mesures.

D. Construction de la variable test

La variable `test` est construite de façon à expliquer la variabilité à l'intérieur des clusters. Pour ce faire, on utilise un MDS pour données pondérées (Oksanen *et al.* 2012).

```
R> library(vegan)
R> worsq <- wcmdscale(mvaddist, w=mvad$weight, k=2)
```

L'analyse en clusters explique principalement les différences sur le premier axe. On crée donc la variable `test` en fonction du deuxième axe. Afin que le test du khi-carré soit proche de zéro, il faut que les proportions de “test” et de “non-test” soient équivalentes dans chaque groupe. Pour satisfaire ces deux contraintes, on utilise la médiane du deuxième axe du MDS dans chaque groupe comme point de séparation.

```
R> library(isotone)
R> mvad$test <- rep(-1, nrow(mvad))
R> for(clust in unique(pamclust4$clustering)){
  cond <- pamclust4$clustering == clust
  values <- worsq[cond, 2]
  mvad$test[cond] <- values > weighted.median(values, w=mvad$weight[cond])
}
R> mvad$test <- factor(mvad$test, levels=0:1, labels=c("non-test", "test"))
```

Affiliation:

Matthias Studer
 Institute for Demographic and Life Course Studies
 University of Geneva
 CH-1211 Geneva 4, Switzerland
 E-mail: matthias.studer@unige.ch
 URL: <http://mephisto.unige.ch/weightedcluster/>