Manuel d'utilisation des fonctions R

Soraya Popic

Stage de fin d'études
Licence SID, UPS - 2011/2012

Objet du document : Ce manuel présente une description de chacune des fonctions ainsi qu'une explication des paramètres d'entrée et/ou de sortie. Toutes les fonctions présentées dans ce document sont agrémentées d'un exemple d'analyse basé sur le jeu de données seita-simulation.csv, qui contient les valeurs des simulations obtenues via Soclab pour le cas SEITA.

Table des matières

1 Fonction `CORR()`

1.1 Description

La fonction `CORR()` permet de tracer un graphique des corrélations (basé sur une matrice des corrélations) entre plusieurs variables.

1.2 Usage

```
CORR(delVar=0,delim="\t",save=FALSE)
```

1.3 Paramètres

`delVar` Grâce à ce paramètre, on peut exclure une ou plusieurs variables de l'analyse des corrélations.

`delim` Ce paramètre est utilisé pour modifier le type du séparateur de colonnes du fichier csv. Par défaut, le séparateur est une tabulation. Pour le modifier, on procède comme suit : `delim=";"`.

`save` Ce paramètre permet de spécifier si l'on veut effectuer une sauvegarde automatique du fichier de sortie ou non. Par défaut, ce paramètre est initialisé à `FALSE`, c'est-à-dire que, par défaut, le graphique ne sera pas sauvegardé. Si l'on désire une sauvegarde, il faut donc mettre le paramètre à `TRUE`.

Quelques exemples de fonctions sur le jeu de données SEITA :

- Tracé de la matrice de corrélation sans les seuils de satisfaction, avec sauvegarde du graphique :

```
CORR(delVar="ChefAtelier.SeuilSatisfaction,OuvEnt.SeuilSatisfaction
    ,OuvProd.SeuilSatisfaction", save=TRUE)
```

NB : Il faut bien prendre en compte **l'ordre d'apparition** des paramètres ou les **nommer explicitement**.

1.4 Sauvegarde

Le graphique sera enregistré en format .png dans le dossier contenant le jeu de données, sous le nom *nomDuJeuDeDonnées*-`CORRgraph.png`.

1.5 Exemple d'analyse

```
CORR(save=TRUE)
```

On obtient le graphique de la figure 1.
La force du lien entre les variables peut être lue via les couleurs (plus la couleur est foncée, plus le lien est fort) ou par la forme du cercle (plus il est aplati, plus la corrélation est forte).
Le sens de la corrélation peut, de même, être interprété selon la couleur (le bleu indique une corrélation négative alors que le rouge dénote d'une corrélation positive) ou selon l'orientation de l'ovale (montant si la corrélation est positive, descendant sinon).
Pour le cas SEITA, on pourrait interpréter les données comme suit :

- On constate que le nombre de pas (`nb_step`) est corrélé négativement avec toutes les autres variables. Cela signifie que plus l'algorithme a besoin d'itérations pour trouver une situation convergente, plus la satisfaction et le seuil de satisfaction des différents acteurs baissent et plus les résultats des relations sont bas. En fait, cela veut dire que lorsque l'organisation a des difficultés à se stabiliser (d'où un grand nombre d'itérations), il faudra certainement passer par une dégradation des satisfactions des différents agents afin de trouver une stabilité.

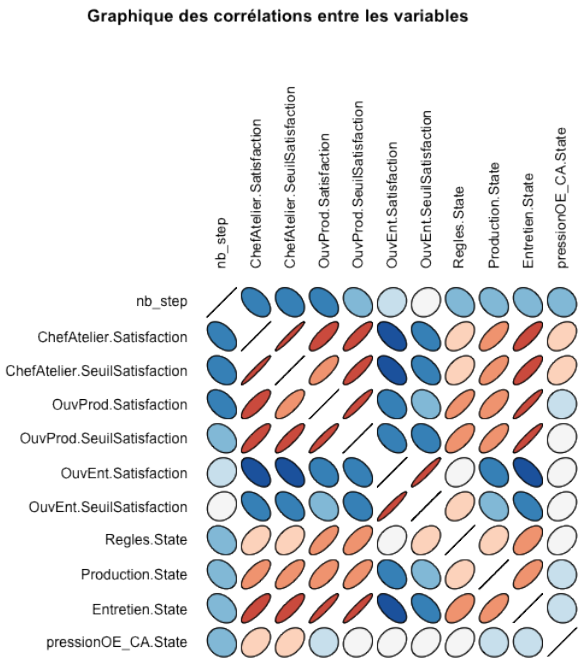

FIGURE 1: Graphique des corrélations obtenu via la fonction `CORR` pour le jeu de données `simulation-Seita.csv`

- Si l'on s'intéresse aux satisfactions des agents, on peut tout de suite voir que l'ouvrier d'entretien est corrélé négativement avec la satisfaction des autres agents. Cela signifie que plus les autres agents sont satisfaits, moins l'ouvrier d'entretien l'est et inversement. On voit tout de suite que cet agent s'oppose aux autres et peut donc être un facteur de mésentente dans l'organisation. De plus, cette tendance est encore plus accentuée quand il s'agit d'étudier les relations entre l'ouvrier d'entretien et le chef d'atelier.
- En ce qui concerne les relations, elles sont toutes corrélées positivement entre elles, ce qui nous amène à penser que, par exemple, plus le respect des règles est fort, plus l'entretien et la production sont élevés.
- Enfin, il est intéressant de voir que l'entretien des machines est la relation qui influence le plus la satisfaction des différents agents.

2 Fonction `MDS()`

2.1 Description

La fonction `MDS()` permet d'obtenir un nuage de points représentant la distance entre chacun des individus. Chacun d'entre eux est ensuite coloré en fonction de la valeur d'une certaine variable. Par ailleurs, elle permet également d'obtenir un « bagplot », c'est-à-dire un nuage de points en deux dimensions permettant de détecter des valeurs aberrantes ou simplement des individus atypiques.

2.2 Usage

```
MDS(delVar=0,delCol=delVar,bagPlot=FALSE,delim="\t",save=FALSE)
```

2.3 Paramètres

`delVar` Ce paramètre permet de spécifier quelles sont les variables que l'on ne veut pas utiliser pour tracer le nuage de points. Par défaut, ce sont toutes les variables présentes dans le fichier qui seront utilisées. Les noms des variables doivent être ceux que l'on retrouve dans le fichier de base : seule la casse n'est pas prise en compte.

`delCol` Le second paramètre sert à indiquer s'il y a des variables que l'on souhaite exclure pour la coloration du nuage de points. Par défaut, ce paramètre est égal au premier. C'est-à-dire que les variables à utiliser pour tracer le graphique seront aussi utilisées pour colorer les points du nuage de points.

`bagPlot` Initialisé à `FALSE`, ce paramètre permet de visualiser le bagplot des observations. Pour l'enregistrer, il suffit de modifier le paramètre `save`.

`delim` Ce paramètre est utilisé pour modifier le type du séparateur de colonnes du fichier csv. Par défaut, le séparateur est une tabulation.

`save` Ce paramètre permet de spécifier si l'on veut effectuer une sauvegarde automatique des fichiers de sortie ou non. Par défaut, ce paramètre est initialisé à `FALSE`, c'est-à-dire que, par défaut, le graphique ne sera pas sauvegardé. Si l'on désire une sauvegarde, il faut donc mettre le paramètre à `TRUE`.

Quelques exemples de fonctions sur le jeu de données SEITA :

- Nuage de points avec toutes les variables sauf celles concernant les seuils de satisfaction des différents agents. La coloration du nuage de points se fera, a fortiori, sans les seuils de satisfaction. Enfin, on sauvegarde le graphique :

```
MDS(delVar="ChefAtelier.SeuilSatisfaction,OuvEnt.SeuilSatisfaction,
OuvProd.SeuilSatisfaction",save=TRUE)
```

- Il est également possible de ne faire varier que certains paramètres comme dans les exemples ci-dessous : ici, on souhaite enlever le chef d'atelier mais seulement pour la coloration :

```
MDS(delCol="ChefAtelier.Satisfaction,ChefAtelier.SeuilSatisfaction"
    ,save=TRUE)
```

NB : Il faut bien prendre en compte **l'ordre d'apparition** des paramètres ou les **nommer explicitement**.

2.4 Sauvegarde

Le MDS sera enregistré en format .png dans le dossier contenant le jeu de données, sous le nom *nomDuJeuDeDonnées*`-MDSgraph.png`.
Le bagplot sera enregistré en format .png également, dans le même dossier, sous le nom *nomDuJeuDeDonnées*`-MDSbagplot.png`.

2.5 Exemple d'analyse

```
MDS(bagPlot=TRUE,save=TRUE)
```

On prends toutes les variables pour tracer le nuage de points et pour colorer.

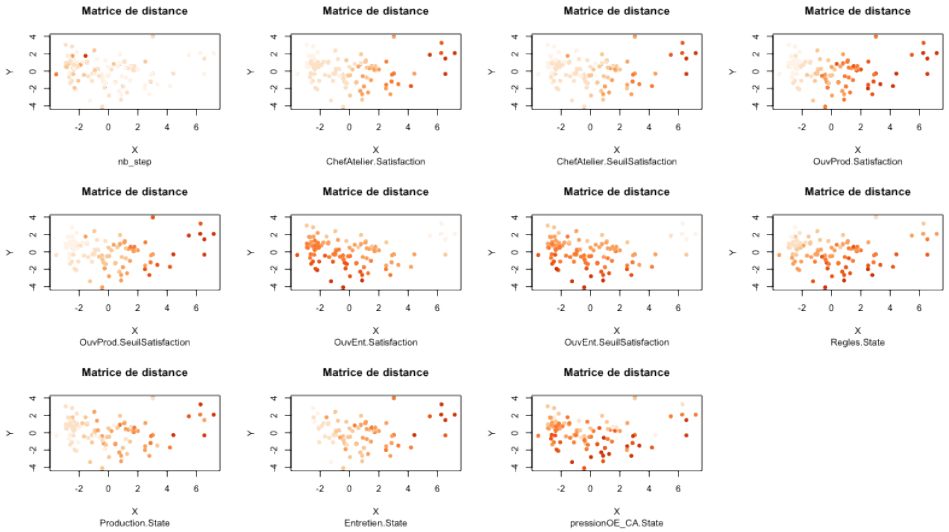

FIGURE 2: Matrice de nuages de points obtenue via la fonction `MDS` pour le jeu de données `simulation-Seita.csv`

Le nuage de points est la représentation en deux dimensions de la matrice de distances entre individus. Il est construit de telle manière que la distance entre deux points dans la projection est approximativement égale à la valeur de leur distance dans l'espace de départ (qui est un espace dont la dimension est égal au nombre de variables ; on a ici muni cet espace de la distance euclidienne usuelle dans $\mathbb{R}^p$ où p est le nombre de variables, sur l'espace des variables, dont les valeurs ont préalablement été centrées et réduites). Autrement dit, deux points proches sur le graphique ont des valeurs plutôt semblables tandis que deux individus éloignés sont bien différents.
L'interprétation de ce graphique va nous permettre de visualiser mais également de comprendre, à l'aide des couleurs, la répartition des individus.
Le nuage de points a été reproduit à l'identique pour chacune des variables choisies (liste modifiable avec `delCol`) puis a été coloré en fonction de l'intensité de cette variable pour chacun des individus. Plus on s'approche du rouge, plus la variable concernée est forte pour cet individu. A l'inverse, un orange pâle dénote une faible valeur pour la variable concernée. Il faut donc être attentif aux variations de couleurs dans les groupes de points qui sont visuellement séparés.
Dans notre exemple, ces groupes sont principalement organisés selon l'axe des abscisses (2 groupes principaux et des simulations « isolées » à droite du graphique).
On interprète les données comme suit :

- D'une manière générale, on peut voir 3 groupes avec un regroupement d'orange pâle, un autre d'orange et un dernier orange foncé. Cette observation est assez subjective : il faut chercher visuellement des concentrations de couleur. On cherche ensuite les variables qui permettraient d'expliquer ces trois groupes.
- On remarque que les satisfactions et seuils de satisfaction des chefs d'ateliers et ouvriers de production expliquent bien les trois groupes : en effet, on voit une séparation nette entre les 3 groupes le long de l'axe des abscisses.

- Aussi, la variable `entretien` rend bien compte de l'existence de 3 groupes avec deux groupes proches (points oranges pâles et oranges) et un groupe d'individus atypiques (points oranges foncés).
- On peut également s'apercevoir qu'à l'inverse, les variables `production`, `règles` et `pression` ne permettent pas d'expliquer les trois groupes. Par exemple pour `règles`, on voit des points foncés tout le long de l'axe des abscisses, ce qui signifie qu'ils ne se sont pas regroupés à un endroit précis et donc, cette variable n'explique pas l'existence de groupes.

L'étude du bagplot va nous permettre d'en savoir davantage sur la distribution.

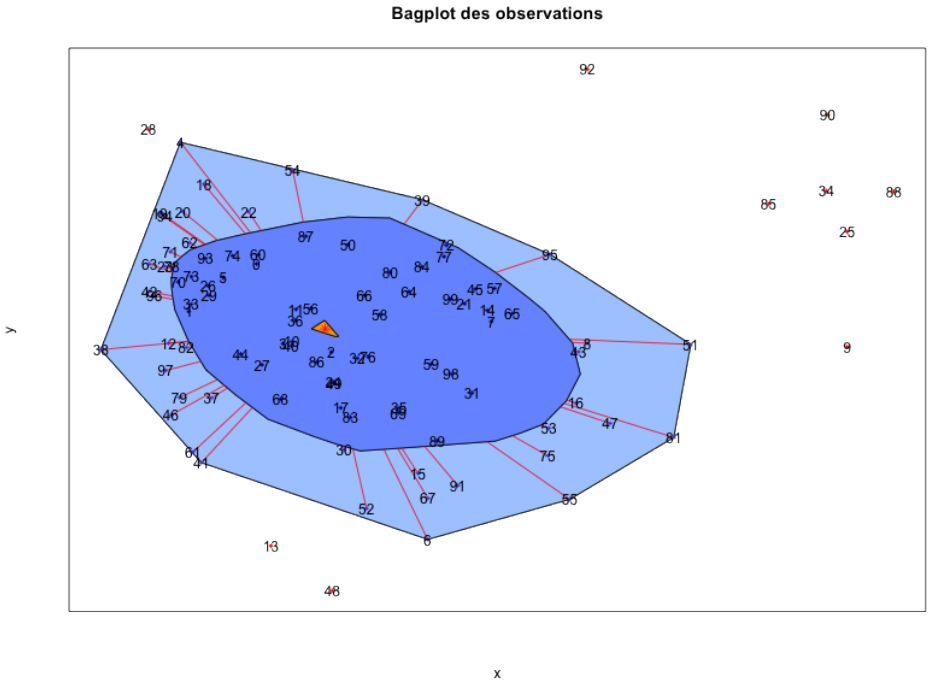

FIGURE 3: Bagplot des données obtenu via la fonction `MDS` pour le jeu de données `simulation-Seita.csv`

Chacun des individus, qui correspondent aux 100 simulations étudiées, a été à nouveau représenté sur un nuage de points en deux dimensions, représentant les distances entre chaque individu.
A l'intérieur de la région bleue foncée, on voit tout d'abord une étoile rouge surmontée d'une flèche jaune : ce symbole représente la médiane bivariée des observations. A l'intérieur de cette surface on retrouve 50% des observations. La surface plus claire représente une extension de la première : ici elle permet de regrouper 90% des observations. En dehors de cette surface, les quelques points restant peuvent être identifiés comme outliers, c'est-à-dire des points atypiques du jeu de données. Le bagplot permet de rendre compte de la forme du nuage de points : en effet, on cherche à y inclure 50% des observations, c'est pourquoi, plus la concentration en variables est forte dans une direction, moins il faudra aller loin pour les rencontrer.

- Pour le jeu de données SEITA, on constate qu'il y a un groupe de points atypiques : c'est celui qui a déjà été identifié lors de l'étude des MDS. On remarque d'autres simulations telles que les simulations 13, 48, 28 et 92 qui présentent un caractère inhabituel.

3 Fonction `CAH()`

3.1 Description

La fonction `CAH()` permet de réaliser une classification ascendante hiérarchique sur un jeu de données défini. Le but de la classification est de simplifier le profil des individus en le regroupant. Chaque individu appartient alors à une classe, un groupe, parmi d'autres individus qui lui ressemblent et pour lesquels, on pourra dresser un profil type.
On pourra fixer un nombre de classes a priori, ou chercher la meilleure représentation.
Cette fonction affiche plusieurs graphiques en sortie : tout d'abord, elle produit un arbre hiérarchique des classes, aussi nommé « dendrogramme » permettant de choisir le meilleur nombre de classes ainsi qu'un MDS présentant les individus selon la classe choisie. Aussi, afin de dégager des profils, on utilisera le tableau d'analyse des moyennes et la matrice de boîtes à moustaches.

3.2 Usage

```
CAH(class=0,delVar=0,manSelect=FALSE,MAV=FALSE,bPlot=FALSE,delim="\t",
    save=FALSE)
```

3.3 Paramètres

`class` Ce paramètre permet d'indiquer en combien de classes on souhaite partager les individus. Il est possible de fixer un seul nombre de classes ou d'en tester plusieurs dans une même fenêtre graphique. Si l'on souhaite tester plusieurs classes à la fois, il est nécessaire de faire précéder ces chiffres par un « c », tel que : `class=c(2,3,4)`.

`delVar` Grâce à ce paramètre, on peut exclure une ou plusieurs variables de la classification

`manSelect` Par défaut à `FALSE`, `manSelect` permet de faire une sélection manuelle sur le graphique (dendrogramme) du nombre de classes. La sélection s'effectue avec le bouton gauche de la souris. Une fois les classes sélectionnées, il faut cliquer sur le bouton droit de la souris pour obtenir les sorties supplémentaires.

`bPlot` Permet d'obtenir une matrice de boîtes à moustaches : pour chacune des variables de l'étude, on obtient la boîte à moustaches de chacune des classes.

`MAV` Permet d'obtenir dans un fichier au format CSV les valeurs numériques du tableau d'analyse des moyennes (uniquement disponible pour un nombre de classes unique ou une sélection manuelle, cf. `Sauvegardes`).

`delim` Ce paramètre est utilisé pour modifier le type du séparateur de colonnes du fichier csv. Par défaut, le séparateur est une tabulation.

`save` Ce paramètre permet de spécifier si l'on veut effectuer une sauvegarde automatique du fichier de sortie ou non. Par défaut, ce paramètre est initialisé à `FALSE`, c'est-à-dire que, par défaut, le graphique ne sera pas sauvegardé. Si l'on désire une sauvegarde, il faut donc mettre le paramètre à `TRUE`.

Quelques exemples de fonctions sur le jeu de données SEITA :

- Tracé du dendrogramme et du MDS avec 2 et 3 classes, sauvegarde du graphique :

```
CAH(class=c(2,3),save=TRUE)
```

- Tracé du dendrogramme, MDS et des boîtes à moustaches pour une classification en 3 classes sans les seuils de satisfaction des différents agents :

```
CAH(3,delVar="ChefAtelier.SeuilSatisfaction,OuvEnt.
    SeuilSatisfaction,OuvProd.SeuilSatisfaction",bPlot=TRUE)
```

NB : Il faut bien prendre en compte **l'ordre d'apparition** des paramètres ou les **nommer explicitement**.

3.4 Sauvegarde

On considère pour cette fonction, 3 manières d'obtenir le dendrogramme et le MDS :

- Choix de plusieurs nombres de classes
- Choix d'un nombre de classes unique
- Choix du nombre de classes manuellement

Le choix d'un nombre de classes multiples sert à comparer différents « découpages » du jeu de données afin de choisir la meilleure partition et ne doit pas être utilisé pour l'interprétation des profils. En conséquence, une commande du type `CAH(class=c(2,3))` ne peut permettre d'obtenir en sortie le tableau d'analyse des moyennes ou encore la matrice de boîtes à moustaches, utiles pour la description des classes.
Une fois qu'on a fixé le nombre de classes désiré, on utilise soit la sélection manuelle `CAH(manSelect=TRUE)`, soit la sélection unique `CAH(class=3)` pour obtenir le tableau d'analyse des moyennes et selon option, les boîtes à moustaches. En conséquence, il n'y a qu'avec ces configurations que l'on peut enregistrer ces sorties.
Les graphiques (dendrogramme/MDS et matrice de boîtes à moustaches) seront enregistrés en format .png. Le tableau d'analyse des moyennes est disponible sous 2 formes : en format .csv (paramètre `MAV`), contenant les valeurs numériques et en format .png, représentant l'analyse des moyennes de manière graphique.

- *nomDuJeuDeDonnées*-`CAHgraph`*n*`Cl.png` pour un graphique tracé avec un seul nombre de classes. *n* est ce nombre de classes.
- *nomDuJeuDeDonnées*-`CAHgraphMulti.png` pour un graphique tracé avec différents nombres de classes.
- *nomDuJeuDeDonnées*-`CAHanalyseMoy`*n*`Cl.csv` pour le tableau d'analyse des moyennes en format csv, avec *n*, le nombre de classes.
- *nomDuJeuDeDonnées*-`CAHanalyseMoy`*n*`Cl.png` pour le tableau d'analyse graphique des moyennes, avec *n*, le nombre de classes.
- *nomDuJeuDeDonnées*-`CAHboxPlot.png` pour la matrice de boîtes à moustaches.

Remarque importante : Il n'est pas possible de sauvegarder un graphique si le nom de celui-ci existe déjà. Par exemple, si un graphique à 3 classes a déjà été sauvegardé, on ne pourra sauvegarder automatiquement un autre graphique à 3 classes, sans les variables seuils par exemple. Pour régler ce conflit, on peut enregistrer manuellement les graphiques, soit en faisant *Clic droit* → *Enregistrer*, soit en allant dans *Fichier* → *Enregistrer sous*.

3.5 Exemple d'analyse

Choix du nombre de classes

Tout d'abord, pour commencer l'étude d'un jeu de données, il va falloir choisir le nombre de classes le plus pertinent. La contrainte généralement fixée est de choisir un nombre de classe minimal pour une inertie intra-groupe minimale. Il est toutefois possible de ne prendre en compte qu'une exigence technique qui amènerait par exemple à souhaiter que les individus ne soient regroupés qu'en 3 profils uniquement. Enfin, on rappelle que la CAH trie les individus en classes tel que :

- Les individus d'un même groupe ont des comportements très proches les uns des autres : la variance intra-groupe (à l'intérieur du groupe) est faible.
- Les individus de deux groupes différents ont des comportements très éloignés : la variance inter-groupe (la variance entre les groupes) est la plus élevée possible.

Dans le cas des données SEITA, l'étude du MDS nous avait démontré l'existence de 3 groupes potentiels. Pour affirmer cela, nous allons, dans un premier temps, tracer le dendrogramme et le MDS du jeu de données partitionné en 2, 3 et 4 groupes, afin de voir si le découpage des individus en 3 classes est bien le plus pertinent.

```
CAH(class=c(2,3,4),save=TRUE)
```

On constatera que le tableau d'analyse des moyennes ne peut être sauvegardé en sortie : en effet, ce dernier permettant de décrire les classes créées, il n'est pas pertinent tant que ce nombre de classes n'a pas été fixé.

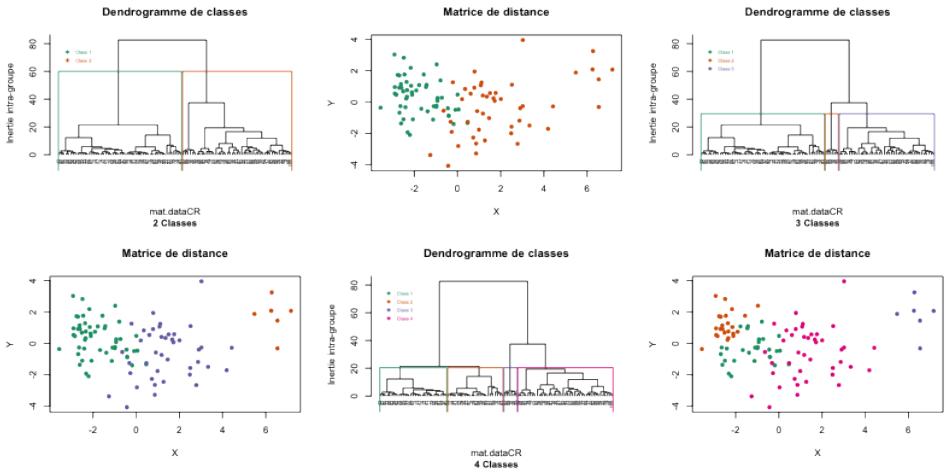

FIGURE 4: Partition du jeu de données en classes multiples obtenue via la fonction `CAH` pour le jeu de données `simulation-Seita.csv`

Le dendrogramme est obtenu grâce à un algorithme qui, à chaque itération, relie les deux individus les plus proches, jusqu'à l'obtention d'une seule et même classe. C'est cette information que l'on peut lire sur l'axe des abscisses. Une classe correspond à une branche de l'arbre : si l'on trace un trait horizontal sur le dendrogramme et que l'on compte le nombre de fois où il coupe une branche du dendrogramme, on obtient le nombre de classes.
Sur l'axe des ordonnées, on lit la valeur de l'inertie intra-groupes que l'on souhaite la plus faible possible. En effet, cette dernière n'étant autre que la variance à l'intérieur des groupes, on la veut très basse afin que des individus formant une classe soient les plus « proches » possible.
On constate que plus l'on descend le long de cet axe, plus le nombre de classes est important, or on souhaite également un nombre de classes faible.
Pour notre jeu de données, selon que la partition soit de deux, trois ou quatre classes, l'inertie intra-groupes est respectivement de 60, 30 puis 20. En effet, l'inertie intra-classes est une fonction décroissante du nombre de classes.
Le choix du nombre de classes est finalement subjectif puisque l'on va le choisir le nombre de classes k tel que le passage de k à $k+1$ classes ne diminue plus significativement l'inertie intra-groupes.
Dans le cas des données SEITA, on va s'attacher à avoir un nombre de classes faibles et au vue des données, on choisira le résultat produit par le MDS, c'est-à-dire trois classes. En effet, démontrer l'existence de 3 groupes de simulations distincts et a fortiori de 3 modèles de fonctionnement de l'organisation est suffisant. Par ailleurs, l'inertie intra-groupes est assez basse.

Description des classes d'une partition

Maintenant que l'on a fixé le nombre de classes désiré, à savoir 3, on va pouvoir obtenir des données supplémentaires sur ce découpage en groupes, notamment grâce au tableau d'analyse des moyennes.

```
CAH(class=3, bPlot=TRUE, MAV=TRUE, save=TRUE)
```

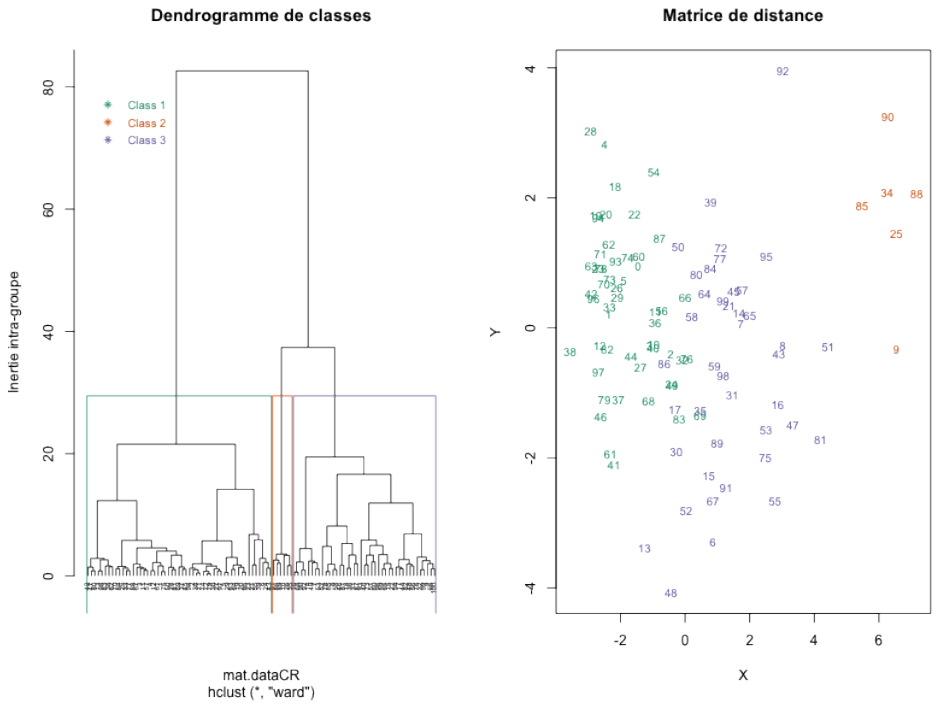

FIGURE 5: Partition du jeu de données en 3 classes obtenue via la fonction `CAH` pour le jeu de données `simulation-Seita.csv`

Ici, on a donc les 100 simulations réparties selon 3 classes. Sur la matrice de distances, les points ont été remplacés par le numéro des individus. Il convient maintenant de comprendre comment ces classes ont été créées, quelles sont les caractéristiques de chacune des classes. Pour cela, on utilisera le tableau d'analyse des moyennes, disponible sous cette forme en .csv :

```
"";"nb_step";"ChefAtelier.Satisfaction";"ChefAtelier.SeuilSatisfaction";"OuvProd.Satisfaction";"OuvP
"1";5824,4;-17,36;-17,71;-24,04;-32,56;20,62;17,31;0,86;-1,88;-5,42;-0,53;53
"2";2433,33;46,79;43,04;64,45;57,86;-9,04;-9,56;3,52;6,53;8,6;2,02;6
"3";4085,07;3,67;0,06;32,83;8,24;17,6;16,45;5,11;0,68;0,76;0,06;41
```

Pour chacun des groupes, on a calculé la moyenne de chacune des variables. En les comparant les unes aux autres, on dressera les profils de chacune des classes. On peut simplifier la lecture de ce tableau en remplaçant les valeurs moyennes par un code couleur simple : si la moyenne considérée est basse par rapport aux autres, elle sera représentée par une pastille bleue (allant du foncé au clair), si elle est forte par rapport aux autres, elle sera rouge : c'est ce que l'on obtient dans la deuxième fenêtre graphique et que l'on peut sauvegarder en .png.
On peut décrire les classes de cette manière (voir figure 6)

- *Classe 1* : La première classe regroupe les simulations qui ont été les plus longues à converger. On peut remarquer que les satisfactions et seuils de satisfaction des chefs d'ateliers et des ouvriers de production sont les plus faibles. En revanche, la satisfaction de l'ouvrier d'entretien, pour ces simulations, est maximale.

  Toutes les relations sont très faibles, cela signifie que le respect des règles, la production, le niveau d'entretien et la pression sont très faibles.

  On peut aussi interpoler les résultats et essayer d'aller plus loin dans l'analyse : l'ouvrier d'entretien est très satisfait car l'entretien est faible, il ne réalise donc pas ou peu de tâches. A l'inverse, l'ouvrier de production ne peut plus travailler correctement car l'entretien des machines est mauvais, ce qui entraîne une baisse de la satisfaction du chef d'atelier car la production est trop basse.

- *Classe 2* : La classe deux présente les caractéristiques inverses de la classe 1. En effet, pour ces simulations qui convergent très rapidement (moyenne faible de `nb_step`), les satisfactions des ouvriers de production et du chef d'atelier sont très bonnes. On peut s'apercevoir que l'entretien et

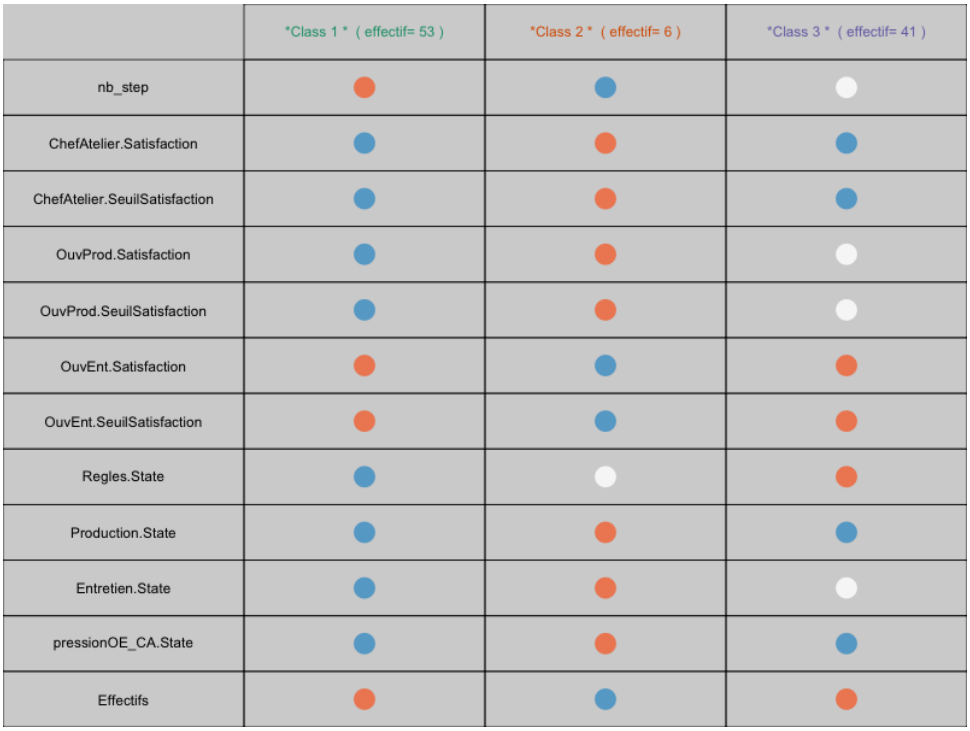

FIGURE 6: Tableau d'analyse des moyennes obtenu via la fonction `CAH` pour le jeu de données `simulation-Seita.csv`

la production ont des valeurs maximales. En conséquence, l'ouvrier d'entretien est très mécontent puisqu'il a un surplus de travail. Le respect de règles a une valeur centrale tandis que la pression des ouvriers d'entretien sur le chef d'atelier est la plus élevée. Cette classe est la plus petite de toute, cela veut dire que c'est un groupe contenant des simulations qui n'occurrent que rarement.

- *Classe 3* : Enfin, cette dernière classe est en quelque sorte le groupe « médian ». Toutes les satisfactions sont moyennes sauf pour l'ouvrier d'entretien qui présente une satisfaction assez élevée, proche de la satisfaction maximale observée dans la classe 2. Le respect des règles est maximal, tandis que `pression`, `entretien` et `production` sont faibles. Dans ces simulations, c'est l'ouvrier d'entretien qui est le plus satisfait, suivi de l'ouvrier de production. On peut constater que, de manière générale, il n'est pas possible d'avoir une bonne satisfaction pour le chef d'atelier et pour l'ouvrier d'entretien.

Chaque boîte à moustaches (figures 7 et 8) est colorée selon la couleur qui a été attribuée à sa classe. On peut aussi voir que la largeur de la boîte est proportionnelle à son effectif permettant de ne pas perdre de vue l'importance relative de chacune des classes. Il est intéressant de constater que :

- Pour la variable `nb_step`, les données à l'intérieur de chacune des classes sont bien hétérogènes : l'étendue des valeurs est faible même si pour chacune des classes, on constate la présence d'outliers. La classe 1 est celle qui propose les plus fortes valeurs de `nb_step` et la plus grande étendue.

- Si l'on s'intéresse au niveau des satisfactions des agents, on peut voir en premier lieu que dans chacune des boîtes à moustaches on constate des valeurs bien réparties autour de la médiane ce qui signifie que les données sont bien réparties. Seules les boîtes à moustaches concernant la satisfaction des ouvriers d'entretien dans les classe 2 et 3 ont tendance à avoir des valeurs supérieures peu concentrées. En outre, la classe 3 semble la plus dispersée.

  Il est intéressant de voir que dans la classe 3, l'ouvrier de production a une valeur médiane de satisfaction égale à 3 contre 15 pour l'ouvrier d'entretien. De même, la valeur de satisfaction maximale atteinte revient à l'ouvrier de production dans les classes 2 et 3, avec 80 contre 40 pour l'ouvrier d'entretien dans la classe 3. En fait, on s'aperçoit que même si c'est l'ouvrier d'entretien qui a le plus souvent la meilleure satisfaction, les autres agents, lorsqu'ils sont satisfaits, le sont avec de meilleures valeurs, surtout l'ouvrier de production.

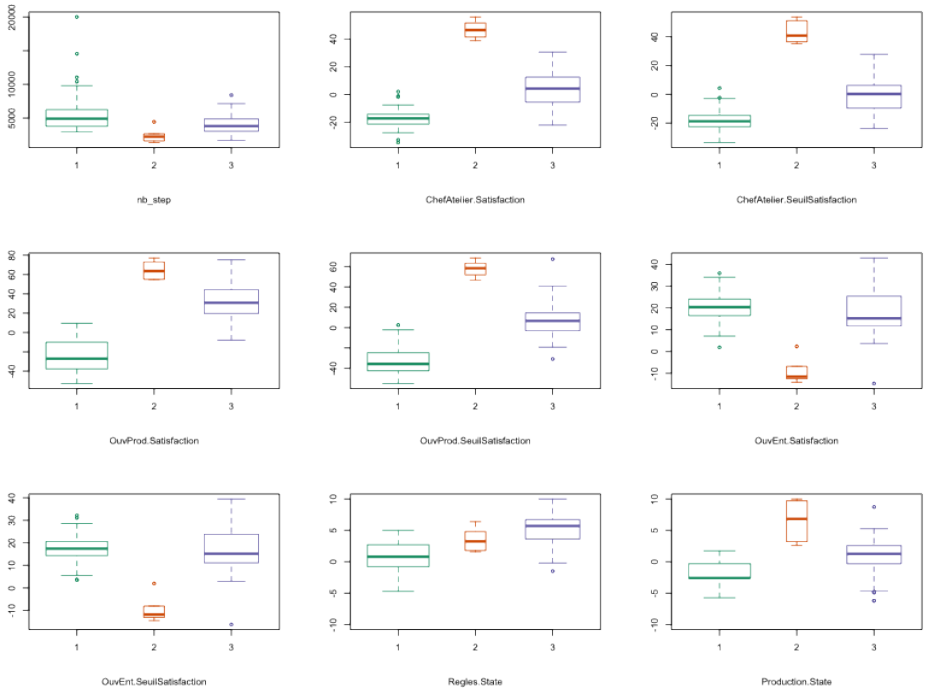

FIGURE 7: Matrice de boîtes à moustache obtenue via la fonction `CAH` pour le jeu de données `simulation-Seita.csv`

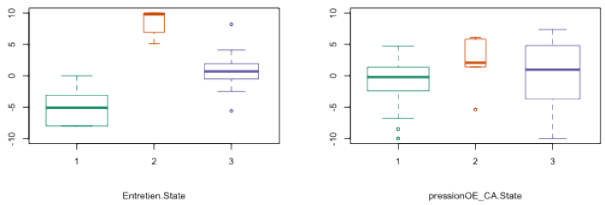

FIGURE 8: Matrice de boîtes à moustache obtenue via la fonction `CAH` pour le jeu de données `simulation-Seita.csv`

- Les relations sont comprises dans l'intervalle [-10 ;10]. Globalement, on voit que la classe 2 présente de bons résultats pour chacune des relations puisque toutes les boîtes à moustaches sont dans la partie supérieure du graphique ([0 ;10]) même si la production présente des valeurs particulièrement étendues. La relation `pression` ne semble pas être intervenue dans la construction des classes, avec des valeurs hétérogènes dans chacune des classes.

4 Fonction `SOM()`

4.1 Description

La fonction `SOM()` permet de réaliser la carte auto-organisatrice d'un jeu de données. En fait, sur ces cartes, les individus sont regroupés en classe comme pour une classification ascendante hiérarchique. Parallèlement, ces classes sont organisées de manière à ce que deux classes qui présentent de fortes similarités soient côte à côte alors que deux classes éloignées le seront également physiquement sur la grille. L'avantage de la carte auto-organisatrice est qu'elle propose davantage de classes et permet donc de mettre à jour des comportements moins évidents, plus particuliers.
En sortie de cette fonction, on obtient 5 graphiques qui permettront d'étudier cette classification :

- La carte des effectifs, aussi nommée « hitmap », indique, pour chaque neurone (ou classe), son effectif.
- La carte des individus, pour savoir exactement quels individus sont présents dans telle ou telle classe.
- La carte des distances, qui révèle par des variations de couleurs la distance entre les neurones.
- Les dernières cartes présentent une matrice de graphiques : la première affiche la valeur moyenne des variables dans les différentes classes tandis que la seconde en fait de même pour l'écart-type.

4.2 Usage

```
SOM(save=FALSE,grid.length=5,grid.type=1,delim="\t",delVar=0)
```

4.3 Paramètres

`grid.length` Pour créer la carte, on doit fixer les paramètres de cette dernière. Ici, `grid.length` permet de modifier la taille de la grille. Par défaut, c'est une grille 5x5, c'est-à-dire une grille comprenant 25 neurones, soit 25 classes différentes.

`grid.type` De même, on peut changer la forme de la grille. Au lieu d'obtenir une grille carrée, on peut donner aux neurones la forme hexagonale. Il faut par ailleurs comprendre que l'hexagone, du fait de sa forme, permet à une classe d'avoir un nombre de classes voisines plus importante. Le type 1 correspond à une grille composée de neurones carrés tandis que le type 2 engendre une grille de neurones hexagonaux. Par défaut, la grille est carrée.

`delVar` Il est possible d'écarter certaines variables de l'analyse grâce à ce paramètre.

`delim` Ce paramètre est utilisé pour modifier le type du séparateur de colonnes du fichier csv. Par défaut, le séparateur est une tabulation.

`save` Ce paramètre permet de spécifier si l'on veut effectuer une sauvegarde automatique du fichier de sortie ou non. Par défaut, ce paramètre est initialisé à `FALSE`, c'est-à-dire que, par défaut, le graphique ne sera pas sauvegardé. Si l'on désire une sauvegarde, il faut donc mettre le paramètre à `TRUE`.

Quelques exemples de fonctions sur le jeu de données SEITA :

- Création d'une carte auto-organisatrice de taille 4x4 avec des neurones hexagonaux :

```
SOM(grid.type=2,grid.length=4)
```

- Création d'une carte auto-organisatrice sans les seuils de satisfaction :

```
SOM(3,delVar="ChefAtelier.SeuilSatisfaction,OuvEnt.
    SeuilSatisfaction,OuvProd.SeuilSatisfaction")
```

NB : Il faut bien prendre en compte **l'ordre d'apparition** des paramètres ou les **nommer explicitement**.

4.4 Sauvegarde

Les graphiques seront enregistrés en format .png. Les noms des fichiers seront les suivants :

- *nomDuJeuDeDonnées*-`SOMcarteEff.png` pour la carte des effectifs.
- *nomDuJeuDeDonnées*-`SOMcarteDist.png` pour la carte des distances.
- *nomDuJeuDeDonnées*-`SOMcarteInd.png` pour la carte des individus.
- *nomDuJeuDeDonnées*-`SOMCarteNivMoy.png` pour la matrice représentant la valeur moyenne des variables dans chacune des classes. Son équivalent basé sur le calcul de l'écart-type est enregistré sous ce nom : *nomDuJeuDeDonnées*-`SOMCarteNivSD.png`

4.5 Exemple d'analyse

Nous allons compléter les résultats de la classification ascendante hiérarchique en réalisant une carte auto-organisatrice sur le jeu de données SEITA. Cette dernière permettra de compléter notre analyse en proposant davantage de classes que la `CAH` (et donc plus de modalités de fonctionnement pour l'organisation) et en donnant pour chaque classe celles qui ont une organisation proche de la sienne.
En fait, on pourrait considérer que la carte auto-organisatrice est une combinaison de méthodes de classification (comme dans la `CAH`) et de visualisation (comme dans le `MDS`). La carte auto-organisatrice est basée sur la création d'une grille (dont l'utilisateur choisit le dimensions) définissant le nombre de classes souhaitées. Dans notre exemple, c'est une grille 5x5 qui est utilisée, fixant a fortiori 25 classes, aussi appelées neurones.
Chaque neurone est représenté par un prototype. Le prototype est une observation artificielle considérée comme représentative des observations d'origine. A chaque itération, on associe à une observation le neurone dont le prototype est le plus proche. Ensuite, le prototype est mis à jour afin de prendre en compte la nouvelle observation et ainsi de suite pour tous les individus.

```
SOM(save=TRUE)
```

Pour étudier le jeu de données, la fonction `SOM` fournit 5 graphiques différents. Il faut croiser les informations provenant de tous les graphiques pour obtenir une bonne analyse. Nous allons commencer par étudier la composition des classes créées par la méthode.

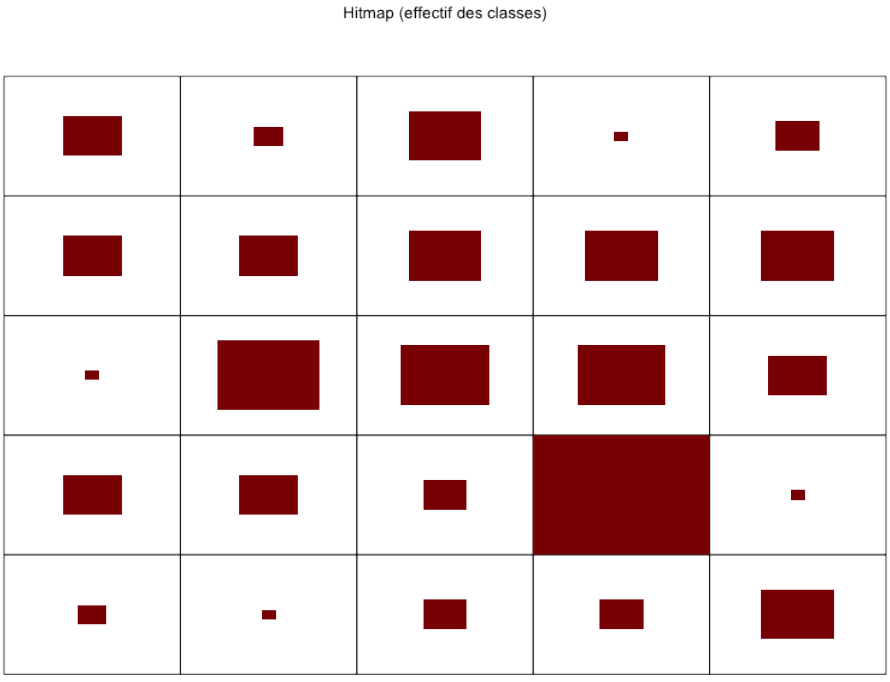

FIGURE 9: Carte des effectifs obtenue via la fonction `SOM` pour le jeu de données `simulation-Seita.csv`

Sur la hitmap (figure 9), nous pouvons voir les 25 classes composant la carte organisatrice. On rappelle qu'une classe regroupe des individus qui sont "proches" les uns des autres.
À l'intérieur de chacune d'entre elles, un rectangle proportionnel à l'effectif de la classe est dessiné. Ainsi, on peut s'apercevoir sur ce graphique que la classe dans le coin inférieur droit regroupe beaucoup d'individus tandis que d'autres classes ont un très petit effectif.

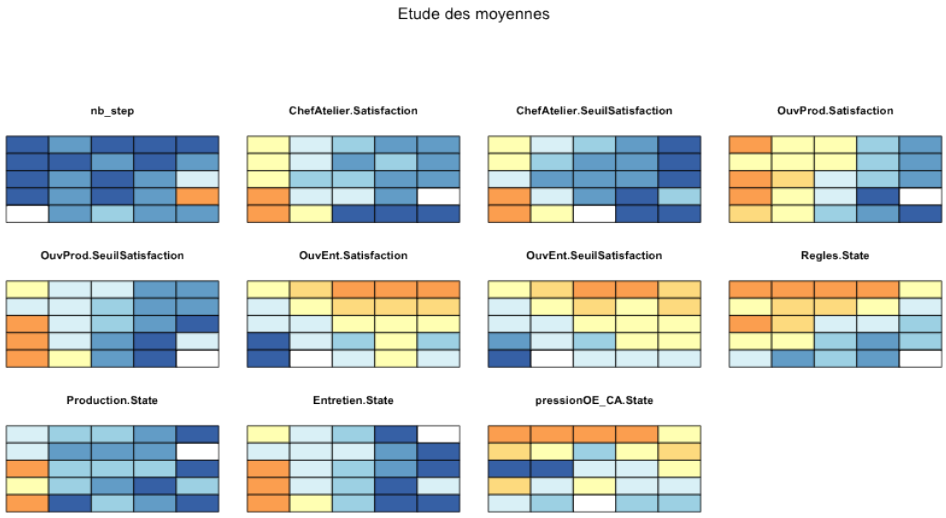

FIGURE 10: Matrice de cartes auto-organisatrices colorée selon les moyennes des variables obtenue via la fonction `SOM` pour le jeu de données `simulation-Seita.csv`

Pour pouvoir décrire les différentes classes, on peut commencer par regarder la matrice de graphiques de la figure 10. Pour chacune des variables, on a représenté la carte auto-organisatrice telle que la méthode l'a créée. Ensuite, chacune des classes a été colorée en fonction de la moyenne de la variable pour cette classe. Plus la valeur de la variable est forte, plus la couleur utilisée sera foncée. Ce qu'il est crucial de comprendre dans cette étude, c'est qu'une couleur bleue dénote une valeur faible de la moyenne relativement aux autres moyennes. Cela ne veut en aucun cas dire que la moyenne est négative par exemple, elle est juste faible par rapport aux autres moyennes. Pour le jeu de données SEITA, on peut constater que :

- Un ensemble de classes placé à droite de la carte présente des satisfactions faibles pour le chef d'atelier et le chef de production. En regardant les relations, on remarque que ces classes sont principalement caractérisées par une production et un niveau d'entretien bas.
- Si l'on s'intéresse à l'ouvrier d'entretien, on se rend compte que pour celui-ci, il y a essentiellement 3 classes pour lesquelles sa satisfaction est élevée. En regardant ces trois classes sur les autres schémas, il est possible de dégager les configurations permettant à l'ouvrier d'entretien d'être satisfait. Ainsi, ce dernier est à l'aise dans les situations où les règles et la pression sont relativement élevées pourvu que le niveau d'entretien soit faible.
- L'étude de ces cartes nous prouve qu'il est plus facile de satisfaire l'ouvrier d'entretien que les autres agents mais qu'il est compliqué de satisfaire les 3 agents simultanément.

Un autre graphique est susceptible de nous apporter de l'information.
Il s'agit de la carte de base colorée selon la distance entre les classes (figure 11) : plus la teinte tend vers le rose foncé, plus la distance entre les prototypes des classes est élevée. En effet, la carte auto-organisatrice a pour but de mettre côte à côte des classes qui sont proches le plus possible les unes des autres. Cependant, elles ne sont pas toutes aussi "proches" les unes des autres, il existe des distorsions et cette carte a pour but de le montrer. Pour mieux comprendre, revenons à notre exemple.

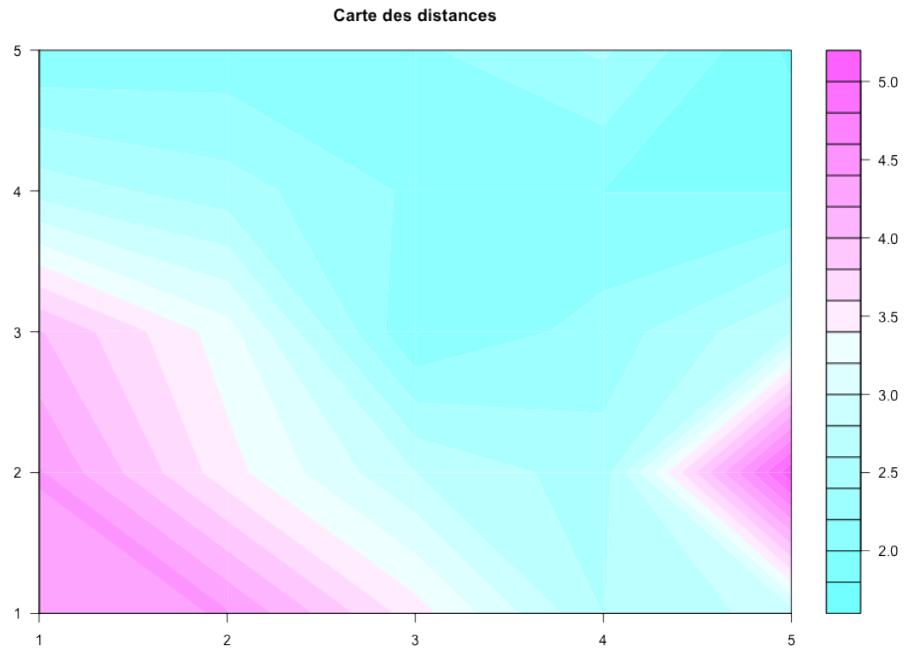

FIGURE 11: Carte des distances obtenue via la fonction `SOM` pour le jeu de données `simulation-Seita.csv`

- On constate que, dans le coin inférieur gauche, il y a une grande distance entre les classes. Ces trois classes formant le coin sont plus distantes des autres classes que ne le laisserait penser la grille. Si l'on s'intéresse davantage à ce trio, on constate qu'il s'agit des simulations 9, 25, 34, 85, 88, 90 et 92 (figure 12).

  En comparant ces numéros avec les résultats de la `CAH`, on retrouve les 6 individus de la classe 2 (classe minoritaire caractérisée par une faible satisfaction des ouvriers d'entretien). En effet, ces individus étant proches les uns des autres et par ailleurs très différents des autres individus du jeu de données (dans le sens où peu de simulations mènent à l'insatisfaction de l'ouvrier d'entretien), ils sont les bons candidats pour former une classe.

- Sur la carte des distances (figure 11), il y a une autre zone qui présente des distances entre classes élevées. En regardant la matrice de représentation de la carte en fonction des variables, on constate que certaines de ces classes sont particulières puisqu'elles ont tendance à présenter des valeurs similaires (très faibles) pour chacune des variables.

  Les simulations qui composent ces classes font partie de ce qui a été la classe 1 de la `CAH` (nombre d'individus important, satisfaction élevée de l'ouvrier d'entretien au détriment de ses collaborateurs, niveau des relations faible). La carte auto-organisatrice a permis de révéler l'existence d'un comportement assez marginal de l'organisation permettant de trouver une situation où aucun des acteurs n'est satisfait. L'organisation n'a pu trouver de stabilité pour ces simulations, c'est pourquoi pour l'une de ces classes (composée d'un seul individu) on a une valeur de `nb_step` très importante. Par ailleurs, on voit bien ici le lien de complémentarité existant entre l'interprétation de la `CAH` et celle des cartes auto-organisatrices.

- Si l'on revient à la case contenant le plus grand nombre d'individus et que l'on cherche le numéro des individus sur le nuage de points de la `CAH`, on voit qu'il représente le « noyau dur »de la classe 1. Ces observations sont très proches les unes des autres et forment le schéma le plus classique de l'organisation SEITA avec une satisfaction positive de l'ouvrier d'entretien (corrélée à un entretien très faible et donc une production très faible) entraînant une satisfaction très basse des autres acteurs notamment du chef de production.

Enfin, on peut utiliser un dernier outil pour étudier ce jeu de données : il s'agit de la même matrice de

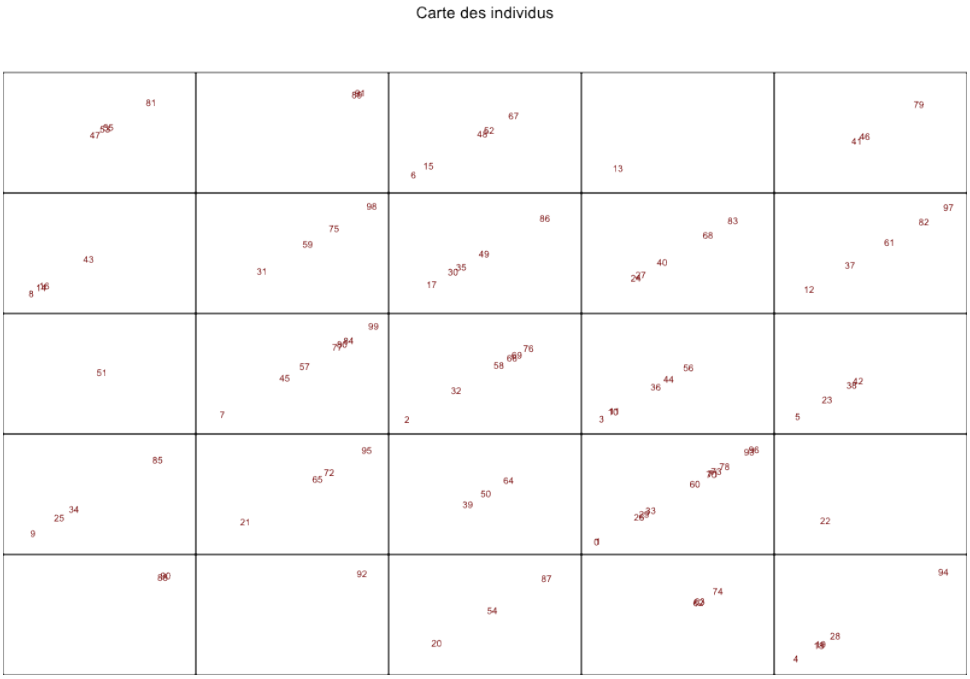

FIGURE 12: Carte des individus obtenue via la fonction `SOM` pour le jeu de données `simulation-Seita.csv`

graphiques que précédemment (figure 10) sauf que les classes ont été colorées en fonction de la valeur de leur écart-type pour cette classe.
On rappelle que l'écart-type permet de mesurer la dispersion des données, c'est-à-dire que l'on cherche à savoir si les simulations d'une même classe sont proches les unes des autres ou non. En fait, cela pourrait paraître contradictoire car le but d'une classe est de regrouper des individus très proches les uns des autres et donc avec une variance très faible. Dans la pratique, c'est globalement ce qui se passe pour nos classes sauf qu'elles ne peuvent pas avoir une variance faible pour absolument toutes les variables mais plutôt une variance faible généralement (en moyenne).
Lors de l'interprétation de ces cartes, il faut bien vérifier le nombre d'individus composant la classe pour ne pas confondre un écart-type faible et un écart-type inexistant (si il y a un seul individu dans la classe, on ne peut calculer d'écart-type).

- On peut voir que la classe qui contenait l'effectif le plus important a un écart-type faible pour chacune des variables, ce qui veut dire que les simulations qui la composent sont vraiment proches les unes des autres et fonctionnent exactement de la même manière.

- Si l'on regarde les classes dans le coin inférieur gauche (celles pour lesquelles la satisfaction du chef d'atelier et de l'ouvrier de production est maximale contrairement à l'ouvrier d'entretien), on remarque que la satisfaction de l'ouvrier d'entretien mais également les relations comme l'entretien, la pression et le niveau de respect des règles ont des écart-types élevés. Comme l'on sait que le niveau d'entretien des machines est fortement corrélé à la satisfaction de l'ouvrier d'entretien, on peut penser qu'il existe peut-être dans ces classes, quelques situations pour lesquelles la satisfaction de l'ouvrier d'entretien est bonne bien que les conditions ne lui soient pas favorables.

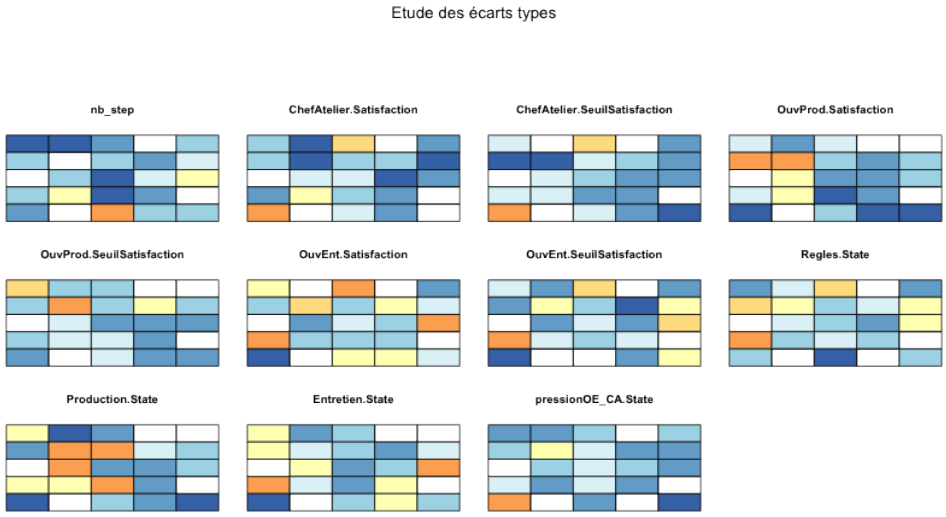

FIGURE 13: Matrice de cartes auto-organisatrices colorée selon les moyennes des écarts-types obtenue via la fonction `SOM` pour le jeu de données `simulation-Seita.csv`