Université Paul Sabatier Toulouse III
Faculté des sciences et Ingénierie
Département Mathématiques
Parcours Statistiques et Informatique Décisionnelle (SID)

Institut de Recherche en Informatique de Toulouse
Université des Sciences Sociales (UT1)
Place Anatole-France
31042 Toulouse Cedex 9

Analyse Statistique de résultats de simulations

issus de la plateforme SocLab

POPIC Soraya

Mémoire de stage L3 SID
Stage effectué du 2 avril au 29 juin 2012
Tuteurs entreprise : SIBERTIN-BLANC C., VILLA-VIALANEIX N.
Tuteur pédagogique : VILETTE V.
Date de soutenance: 02/07/12

Historique du document

Date de mise à jour	Objet	Version
05/04/2012	Définition des contours de l'étude But du projet Conception du plan d'étude Planning prévisionnel	1.0
09/04/2012	Développement de la méthode Présentation du cas SEITA Analyse des corrélations du cas SEITA	1.1
13/04/2012	Analyse de la dispersion du cas SEITA Explication de la méthode MDS	1.2
24/04/2012	Ajout du bagplot dans l'analyse de la dispersion SEITA Analyse de la classification du cas SEITA Explication de la méthode de classification	1.3
27/04/2012	Présentation de Git Présentation du cas Bolet	2.0
21/05/12	Analyse des cartes auto-organisatrices du cas SEITA Analyse des corrélations du cas Bolet Analyse de la dispersion du cas Bolet Analyse de la classification du cas Bolet	2.1
23/05/12	Analyse des cartes auto-organisatrices du cas Bolet Développement informatique	2.2
29/05/12	Explication de la méthode des cartes auto-organisatrices	2.3
05/06/12	Présentation de LaTeX Assurance et contrôle qualité	3.0
08/06/12	Développement informatique Présentation du cas SEITA-500 simulations	3.1
12/06/12	Analyse des corrélations du cas SEITA-500 simulations Analyse de la dispersion du cas SEITA-500 simulations Analyse de la classification du cas SEITA-500 simulations	3.2
13/06/12	Analyse des cartes auto-organisatrices du cas Touch Présentation du cas Touch	3.3
14/06/12	Analyse des corrélations du cas Touch Analyse de la dispersion du cas Touch Analyse de la classification du cas Touch	4.0
19/06/12	Vérification des annexes Rédaction des résumés Rédaction du bilan et de la conclusion	4.1

Table des matières

Index des figures

Résumé

Le but de mon travail sur les jeux de données issus de SocLab -un environnement informatique qui simule des relations sociales entre agents- est la mise en place d'outils statistiques adaptés afin de répondre à des questions telles que : comment fonctionnent les organisations ? Quelles relations ou acteurs prédominants influencent les organisations ? Existe-t-il différents modes d'organisation ?
Dans cette optique, j'ai codé différentes fonctions via le logiciel R. Mon plan d'étude inclut des analyses des corrélations, de la dispersion mais également des méthodes de classification et la création de cartes auto-organisatrices (ou cartes de Kohonen).
Ensuite, j'ai réalisé des exemples d'analyse pour chacune des fonctions, basés sur des cas d'école ou des cas pratiques.

Abstract

The SocLab environment allows us to simulate the behavior of social actors. It is based on a formalization of the Sociology of The Organized Action, a theory that explains the different behaviors of the agents forming a social organization.
The aim of my work is to analyse SocLab data in order to discover various types of organization and to find actors and resources that affect the most the organization state.
To this end, I have created different fonctions thanks to the R software environment. These functions allows us to analyse correlations between variables as well as scattering. They also make statistical clusterings and self-organizing map.
Furthermore, I have produced analysis based upon well-known cases such as the SEITA case or the Bolet case so as to give an interpretation example.

I. Présentation du projet

1. Objet du document

Dans le cadre de cette étude, nous travaillons sur des jeux de données issus du logiciel Soclab. Ce dernier est un environnement informatique permettant de simuler des relations multi-agents à partir d'un modèle d'organisation sociale renseignée par l'utilisateur. En fait, ce logiciel s'appuie sur la sociologie de l'action organisée. Afin de bien comprendre le sujet, il convient de définir plusieurs notions.

Soclab se propose d'étudier des relations entre différents agents et plus particulièrement, des relations entre différentes organisations.
On a constaté d'une manière générale que toute organisation a besoin d'observer la stabilité des différents agents qui la composent afin de pouvoir réaliser ses objectifs et par là-même, de perdurer.
La sociologie de l'action organisée (SAO) se propose justement d'étudier la manière dont les organisations trouvent cette stabilité, se régulent. Les différents acteurs ajustant leur comportement les uns aux autres afin de trouver un équilibre.
Les principes de la SAO ont été jetés par M. Crozier et E. Friedberg (Crozier & Friedberg, 1977).
Soclab permet de recréer un système multi-agents puis, grâce à un algorithme, de simuler les comportements que peuvent avoir les agents entre eux afin de trouver une situation d'équilibre, c'est-à-dire, la stabilité des organisations.

a. Principe de base

Il existe différents agents qui chacun, contrôlent certaines ressources et dépendent d'autres. Ils se servent logiquement de celles qu'ils contrôlent comme d'un levier pour influencer le comportement des agents qui dépendent de ces ressources. Au final, ils peuvent décider d'être plus ou moins coopératifs entre eux, selon qu'ils ont besoin de la ressource détenue par l'autre ou non.
Par ailleurs, chacun des acteurs poursuit un but, a une ambition qui lui est propre. Lorsque cet objectif est atteint, l'acteur est alors satisfait.
L'organisation se trouvera dans un état de stabilité lorsque tous les acteurs auront trouvé satisfaction.

b. L'algorithme

L'algorithme fonctionne sur le principe d'apprentissage par essais-erreurs. On considère qu'à la base, un acteur ne sait pas comment se comporter face aux autres car il ne connaît pas leur manière de réagir. Il va donc devoir tenter des actions au hasard et on tirer des conclusions positives ou négatives. Ensuite, fort de son expérience, il pourra choisir selon le cas de réitérer une action qui avait produit ses fruits par le passé ou de tenter une nouvelle action, toujours dans le but d'atteindre ses propres objectifs.

Chaque acteur est définit en fonction de variables, dont les valeurs sont définies par l'utilisateur du logiciel : le sociologue. Ainsi, on peut définir pour un acteur son seuil de satisfaction (seuil à partir duquel il commencera à être satisfait) variant de 1 à 100 mais également les ressources qu'il contrôle, appelées « relations », notées de 1 à 10.
On trouve également des « paramètres », variant de 1 à 10 tels que sa ténacité (sa propension à chercher de nouvelles solutions ou au contraire à exploiter ses connaissances), sa réactivité (sa capacité à réagir rapidement face aux stimuli des autres acteurs ou pas), son discernement (son aptitude à bien évaluer les situations ou non).
L'algorithme s'arrête lorsqu'il converge, c'est-à-dire lorsque un état de stabilité a été atteint (*i.e.*: tous les acteurs sont satisfaits donc ils gardent une ligne de conduite bien définie et ne cherchent

plus à modifier leurs comportements). Il est à noter que l'algorithme ne converge pas systématiquement car les acteurs ne finissent pas toujours par trouver une entente et que lorsqu'il converge, ce n'est pas toujours vers un optimum global. Cela veut dire qu'on n'arrive pas forcément à un état du jeu où la somme des capacités de chacun des acteurs serait maximale.
Enfin, il a été démontré par Barel (Barel, 1979) que certaines organisations peuvent trouver plusieurs modalités selon lesquelles elles pourraient fonctionner, même si, empiriquement, une seule de ces situations est observée.

c. But du projet

Le but, dans ce cadre, est de présenter une méthode permettant l'exploitation des données issues de Soclab qui doit être utilisable par une personne non initiée au logiciel R.
Pour cela, je suis amenée à coder différentes fonctions, chacune permettant de mettre en œuvre une méthode statistique différente. Chacune de ces fonctions sera ensuite implémentée dans une interface Java programmée par un précédent stagiaire. Le but étant qu'un sociologue puisse utiliser facilement mes programmes, je dois également rédiger un manuel utilisateur (disponible en fichier PDF) précisant les résultats obtenus pour chacune des fonctions et présentant un exemple d'interprétation de résultats. Par ailleurs, je dois aiguiller l'utilisateur sur ce que chaque fonction peut mettre en lumière et sur ce qui doit particulièrement attirer son regard.
En outre, le travail du statisticien doit permettre de mettre à jour différents modes de stabilité d'une organisation (théorie mise en lumière par Barel) et de révéler quels sont les acteurs et les relations les plus influents. Jusqu'ici, aucune étude n'avait été menée au delà de l'ACP et nous espérons que la classification pourra donner de nouveaux résultats.

2. But du document

Ce document servira d'appui pour comprendre les différentes méthodes statistiques mises en œuvre afin d'étudier un jeu de données issu de Soclab et permettra à l'utilisateur d'en tirer des conclusions pertinentes. Par ailleurs, il pourra aussi permettre de faciliter la mise à jour ou la modification des fonctions R.

3. Domaine d'application

Comme défini précédemment, le domaine d'application est avant tout la sociologie. Les méthodes statistiques développées dans ce document correspondent au traitement de données issues de Soclab et des besoins en information de ses développeurs. Enfin, les résultats pourront également être utilisés par les sociologues afin de faciliter la compréhension de leur modèle d'organisation.

4. Présentation du document

Après avoir présenté les différents documents qui m'ont permis de comprendre le fonctionnement de l'environnement SocLab et de m'approprier quelques théories sur la sociologie de l'action organisée, je présenterai le planning prévisionnel ainsi que le plan d'étude.
La partie suivante permettra d'exposer les résultats statistiques obtenus avant d'expliquer comment les fonctions ont été codées.
Pour finir, je présenterai les méthodes et outils qui m'ont été nécessaires pour mener à bien cette étude avant de présenter quelques tests unitaires que j'ai réalisés sur mes scripts.
Ce document est présenté dans sa version finale, à savoir la version 4.1.

II. Documents et références

1. Documents applicables

En tout début de stage, un document m'a été remis, expliquant les enjeux et objectifs de ce stage (consultable en annexe). Il s'agit d'un document informel. C'est le seul document applicable à proprement parler car les besoins et l'objet du projet ont été affinés ensuite, à l'oral.

2. Vocabulaire de base

a. Abréviations

BE : Bureau d'études
CAH : Classification Ascendante Hiérarchique
GNU : GNU's Not UNIX (trad. : GNU n'est pas Unix)
MDS : Multi-Dimensional Scaling
SAO : Sociologie de l'action organisée
SEITA : Service d'Exploitation Industrielle des Tabacs et Allumettes
SIG : Système d'Information Géographique

b. Définitions

Apprentissage automatique : Fait partie des techniques liées à l'intelligence artificielle. Elle permet de développer, analyser, implémenter des méthodes automatiques permettant à une machine d'évoluer grâce à un processus d'apprentissage. On dit qu'une machine évolue ou « apprend » dès lors qu'elle est capable de modifier sa structure de sorte que ses performances futures soient meilleures. Il existe des apprentissages supervisés et non supervisés.
Apprentissage automatique non supervisé : En informatique, il s'agit pour un logiciel ou une fonction de réaliser un tri avec des données hétérogènes de manière à regrouper les individus considérés comme similaires dans un même groupe et à l'inverse, de mettre les observations considérées comme différentes dans des groupes distincts.
Apprentissage automatique supervisé : Dans cette méthode d'apprentissage, il existe une sortie définie *a priori*. Ici, l'apprentissage est basé sur une base de données d'apprentissage contenant des cas déjà traités et validés.
Bug (ou bogue) : Défaut dans le programme.
Device : Terme utilisé dans les commentaires des scripts. « Device » ou « output device » désigne le périphérique de sortie : ici, il s'agit d'une fenêtre graphique.
Distance euclidienne : Pour deux points x et x' de Rd, la distance euclidienne est: $\sqrt{\sum_{i=1}^{d}(x_i - x_i')^2}$.
Intelligence artificielle : L'intelligence artificielle est une discipline scientifique visant à doter un système informatique de capacités intellectuelles comparables à celle d'un être humain telles que le raisonnement, l'adaptation, l'apprentissage, la compréhension, etc.
Logiciel de gestion de versions : Logiciel permettant de conserver la chronologie des modifications apportées à un ensemble de fichier, ainsi que les modifications multiutilisateurs.
Organisation : Ensemble assez fortement codifié, délimité, où les acteurs sont durablement ensemble et pour lequel il existe un objectif.
Outlier : Observation très différente des autres. Elle est numériquement distante du reste des données. (trad. : marginal, exceptionnel).
Package : Un package est un ensemble complet et documenté de programmes conçu pour être fourni à plusieurs utilisateurs, en vue d'une même application ou d'une même fonction.
Stimulus : Un stimulus désigne un facteur qui déclenche une réaction sur un organisme vivant. Par exemple, on peut observer des stimuli visuels, thermiques, auditifs, etc.

Test : [Informatique] Vérification par exécution.

3. Documents de références

Outre les articles écrits par les membres de l'équipe que j'ai intégrée, (El Gemayel *et al.*, 2011) j'ai reçu d'autres documents permettant de mieux cibler le sujet tels que les travaux réalisés antérieurement par de précédents stagiaires. En effet, le projet que je dois réaliser est la suite d'un projet déjà entamé par un autre étudiant.
Ainsi, son rapport présente des analyses statistiques de base telles que :
L'analyse statistique univariée des différentes variables, comprenant l'étude des quartiles, boîtes à moustache, histogrammes.
L'analyse bivariée avec notamment l'étude de nuages de points, de boîtes à moustaches entre variables qualitatives et quantitatives (découpées en classes).
Par ailleurs, des régressions linéaires simples et multiples ont été réalisées où l'on a par exemple essayé d'expliquer le nombre de pas de l'algorithme en fonction de tous les autres paramètres.
Enfin, une analyse en composantes principales a été menée sur toutes les variables puis les codes R permettant d'obtenir des ACP pour la satisfaction des acteurs ou encore le nombre de pas et les satisfactions ont été donnés.
Par ailleurs, j'ai également utilisé des extraits de livre de sociologie afin de mieux délimiter le sujet de l'étude au travers de cas d'école réputés en sociologie tels que le cas Bolet et le cas SEITA. J'ai également lu des rapports de recherches permettant de comprendre comment a été mise en place la modélisation de cas concrets.
Enfin, les documents de cours relatifs à R reçus lors de l'année scolaire m'ont été d'un grand secours.

III. Organisation du travail et planning

1. Conception du plan d'étude

Pour mener une étude statistique sur plusieurs mois, il est inévitable d'organiser un plan d'étude. Pour cela, il faut tout d'abord bien comprendre les besoins des utilisateurs mais aussi s'approprier le travail déjà fait sur le sujet.
Après avoir lu les travaux des stagiaires précédents, je me suis aperçue que des fonctions permettant de réaliser des statistiques univariées, bivariées et des ACP avaient déjà été créées, permettant de travailler sur les données obtenues en sortie du logiciel Soclab: c'est pourquoi, il n'était pas pertinent de refaire ces mêmes analyses.
Nous avons donc mis au point un plan d'étude prévoyant d'employer des méthodes statistiques qui n'avaient pas encore été employées pour ce sujet.
En conséquence, il a été décidé de commencer par une analyse des corrélations suivie d'une analyse de la dispersion. Cette dernière sera effectuée via la méthode MDS, méthode statistique qui m'était inconnue.
Suivra ensuite une classification réalisée avec les méthodes courantes (k-means, CAH) et enfin, la mise au point de cartes auto-organisatrices.
En outre, il ne faut pas perdre de vue l'objectif principal, à savoir, trouver différentes configurations permettant à l'organisation de trouver sa stabilité, si elles existent.

2. Planning

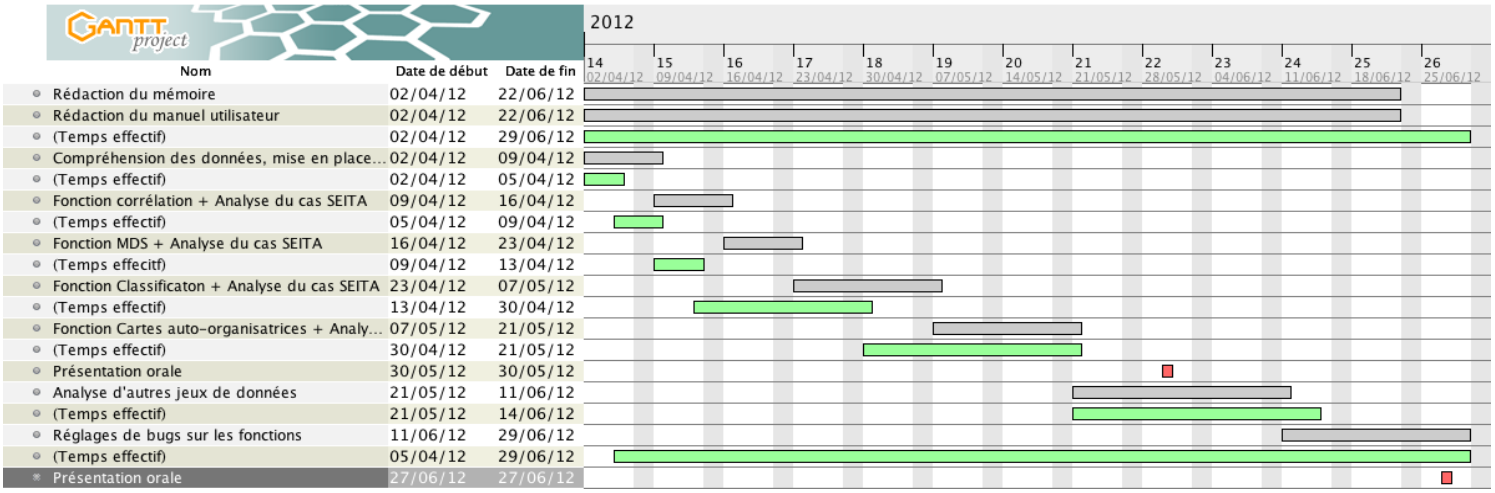


Figure 1 : Diagramme de Gantt

Les tâches colorées en gris représentent le planning prévisionnel tandis que celles en vert sont les délais effectifs. On peut constater qu'au début, la programmation des fonctions a été plus rapide que prévu, avant de prendre finalement davantage de temps à partir de la fonction de classification.
Les cartes auto-organisatrices m'ont demandé beaucoup de temps car je ne connaissais pas du tout cette méthode et il m'a d'abord fallu comprendre l'aspect théorique avant de lire des cas d'interprétation.
Au début, je pensais que le réglage de bugs allait pouvoir s'effectuer à la fin, de manière méthodique. En réalité, dès la première fonction codée, je n'ai cessé d'apporter des corrections aux fonctions. En effet, c'est d'abord en utilisant les fonctions que l'on trouve leurs failles plutôt qu'en mettant en place des tests. Ces derniers permettent d'éviter les principaux écueils mais il est impossible de penser à toutes les situations.

La rédaction du rapport a été commencée dès le début du stage car pour moi, il est assimilable à un carnet de bord qui permet à la fois de construire une réflexion cohérente autour du projet mais également de visualiser l'avancement du projet. De plus, dans le cadre de mon sujet, l'étude de cas représente la moitié du travail fourni.
Enfin, je n'ai pas représenté chacune des tâches dans ce diagramme, cela aurait été trop fastidieux et donnerait au final assez peu d'informations supplémentaires. Je n'ai pas noté par exemple la rédaction en LateX du manuel utilisateur, la création de sous-fonctions nécessaires au fonctionnement des fonctions principales, la rédaction des parties hors développement du mémoire, la préparation des présentations orales, etc. J'ai en effet présenté mon travail à deux reprises aux membres de l'équipe (résumé de ma présentation disponible en annexe).
Finalement, j'estime avoir assez bien suivi mon planning et avoir bénéficié d'assez de temps pour apporter une finition soignée à l'ensemble de mon travail.

IV. Développement de la méthode

Afin d'organiser une méthode d'analyse de résultats statistiques et d'implémenter les fonctions, je me suis basée sur un seul jeu de données issu de simulations sur Soclab. Il s'agit du jeu de données SEITA. Par la suite, les fonctions créées ont été testées sur de nouveaux jeux de données pour vérifier d'une part, que les fonctions étaient bien adaptées à différentes configurations et d'autre part, que les résultats attendus étaient bien conformes aux connaissances préalables sur ces données déjà familières aux informaticiens et sociologues.
J'attire votre attention sur le fait que la partie informatique sera moins développée dans ce mémoire que l'analyse statistique qui est le cœur du projet.
Le manuel utilisateur, disponible en PDF, explique le rôle de chacune des fonctions et détaille chacun des paramètres ainsi que leur utilisation. Par ailleurs, tous les scripts ont été commentés et sont disponibles en annexe. Enfin, il est également possible de consulter l'arborescence des fonctions, présentant les liens entre chacune d'elles.

1. Description des jeux de données

a. Jeu de données principal : le cas SEITA

Rappelons que ce jeu de données est issu de l'exemple SEITA, un cas célèbre élaboré par M. Crozier, dont les résultats forment un cas d'école bien connu. SEITA est une entreprise industrielle française, produisant des articles de consommation courante, en situation de monopole.
Chacun de ses ateliers est composé de trois catégories de personnel que l'on nommera ici des acteurs et qui effectuent chacun des tâches bien définies au sein de l'entreprise.
Le jeu de données de base généré via Soclab est nommé simulation-seita.csv. Ce dernier contient 100 simulations à modèle fixé ; ici, on a 3 acteurs et 4 relations.
Les acteurs sont :

- les chefs d'ateliers,
- les ouvriers d'entretien,
- les ouvriers de production.

En particulier, les variables dont on tiendra compte sont la satisfaction et le seuil de satisfaction de chacun des acteurs. Le seuil de satisfaction est le seuil au delà duquel l'acteur est satisfait alors que la satisfaction est le niveau de satisfaction finale de l'acteur à l'issue de la simulation. Ces dernières sont notées comme suit : `OuvProd.Satisfaction, OuvProd.SeuilSatisfaction.`
Les différentes relations sont les suivantes :

- `Regles.State` : niveau du respect des règles à l'intérieur de l'entreprise atteint à la fin de la simulation. Ces règles sont impersonnelles et prévoient une solution à chaque problème potentiel.
- `Production.State` : niveau de la production atteint à la fin de la simulation.
- `Entretien.State` : niveau d'entretien des machines atteint à la fin de la simulation. Ce sont les ouvriers d'entretien qui contrôlent cette relation.
- `pressionOE_CA.State` : niveau de pression des ouvriers d'entretien sur le chef de projet à la fin de la simulation.

On a également une variable notée `nb_step` qui correspond au nombre d'itérations de la boucle avant de trouver un état stable (*i.e.*: tous les acteurs sont satisfaits).
Enfin, il existe une variable notée `run`, qui donne le numéro du run correspondant (de 1 à 100) mais celle-ci n'étant qu'un identifiant, elle n'est pas prise en compte lors de l'étude.

b. Jeu de données secondaire : le cas Bolet

Le cas Bolet est, tout comme SEITA, un cas d'école bien connu des sociologues. Ici, au lieu d'un cas de monopole industriel, il s'agit d'une PMI familiale.
M. Bolet est le fondateur de la société Charicoup qui produit du matériel pour scierie. L'entreprise ne connaît pas de difficultés de recrutement et a connu une expansion rapide, passant de 20 à 200 personnes en 15 ans.
Lors d'un voyage en Suède, le directeur commercial a découvert un nouveau matériel de fabrication : une machine qui serait beaucoup plus performante que celle utilisée actuellement dans l'entreprise. Le directeur commercial n'est autre que le fils de M. Bolet, Jean Bolet. Il gère les relations avec les clients grâce auxquels il détermine les contraintes de production qu'il souhaite voir appliquer.
Par ailleurs, le second fils de M. Bolet, André, travaille également dans l'entreprise familiale au poste de directeur de fabrication. Il peut alors décider d'appliquer ou de contester les prescriptions s'appliquant à la production mises en place par son frère avec le bureau d'études.
Au final, c'est toujours le père qui prend la décision finale : ce dernier est plutôt conciliant et prend avant tout en compte la satisfaction de chacun.
Le jeu de données sensibility-bolet.csv a pour but d'étudier l'influence de certains paramètres sur le modèle. Il a été réalisé sur 100 simulations.
On y retrouve le niveau de satisfaction de 4 acteurs, à savoir :

- `CA_satis`, la satisfaction du chef d'atelier,
- `Pere_satis`, la satisfaction du fondateur de la société,
- `Andre_satis`, la satisfaction du premier fils,
- `Jean-BE_satis`, la satisfaction du deuxième fils couplé au bureau d'études.

Une variable notée `param0` à également été fixée. Cette dernière permet de faire varier la ténacité du chef d'atelier. On trouve ensuite 6 relations qui prennent leurs valeurs dans l'intervalle [-10 ;10]:

- `decision-achat_staterel`, la décision d'achat de la machine : c'est une relation contrôlée par le père.
- `application-prescription_staterel`, montre à quel point les prescriptions du bureau d'études sont appliquées ou non. Cette relation est détenue par le chef d'atelier, qui peut décider, au niveau de la production d'appliquer correctement les prescriptions du BE ou non.
- `investissement-dans-prod_staterel`, l'investissement dans la production est contrôlé par le chef d'atelier. Soit l'équipe de production est vive et la production est élevée, soit l'équipe met peu d'ardeur à effectuer son travail et la production est faible.
- `controle-application-presc_staterel`, le niveau de contrôle d'application de la prescription du BE. C'est André, qui, en tant que directeur de fabrication peut augmenter les contrôles au niveau de la production pour s'assurer que les prescriptions du BE sont suivies.
- `nature-prescription_staterel`, contrôlée par Jean. Le BE (et donc Jean) détermine les besoins de la clientèle et donc définit les prescriptions de production. La nature de sa prescription va de « complètement rationnelle » à « la moins rationnelle possible ». En fait, soit il prend des décisions complètement rationnelles dictées par les lois du marché, soit il s'éloigne de cette rationalité et laisse d'autres facteurs interférer.
- `controle-nature-presc_staterel` est une relation détenue par André. Soit André contrôle la décision du BE (la prescription), soit il ne la vérifie pas du tout.

On constate que les relations détenues par André sont différentes des autres car en réalité, elles portent sur la marge de manœuvre de la maîtrise d'autres relations.
Exemple : Si André décide de faire un contrôle tatillon de l'application de la prescription, il réduit la marge de manœuvre du chef d'atelier qui, lui, contrôle l'application de la prescription dans son atelier.

Enfin, on trouve également la satisfaction globale des acteurs à la fin de la simulation (`total_satis`) et le nombre de pas nécessaires pour converger, noté `nb_step`.

c. Jeu de données secondaire : SEITA à 500 simulations

Ce jeu de données est un « extension » du premier jeu de données SEITA déjà étudié. La principale différence réside dans le fait que celui-ci est composé de 500 simulations au lieu de 100 seulement même si le modèle a également été soumis à quelques modifications.
On va chercher à tester la robustesse de la première étude mais également à voir ce qui va changer dans l'interprétation des résultats. Par ailleurs, certaines variables ont été supprimées, d'autres ajoutées. Voyons ce qu'il en est des acteurs :
NB : Dans ce fichier de données, les acteurs étaient mentionnés en anglais et les relations n'avaient pas exactement le même nom que dans le premier jeu de données. Par souci de compréhension, j'ai renommé toutes les variables à l'identique.

- ChefAtelier.Satisfaction
- OuvProd.Satisfaction
- OuvEnt.Satisfaction

On constate que dans cette version, nous n'avons plus accès aux seuils de satisfaction de chacun des acteurs.
Les 4 relations sont exactement les mêmes que dans le jeu de données originel (`Regles.State, Production.State, Entretien.State, pressionOE_CA.State`).
On a également une variable qui représente la satisfaction globale des acteurs (`total_satis`) et une autre, le nombre de pas nécessaires à l'algorithme pour converger (`nb_step`).
Enfin, on a 4 paramètres qui varient d'une simulation à une autre : normalement, ces paramètres sont nommés param*n* (avec *n* variant de 1 au nombre de paramètres) mais pour des raisons de lisibilité, j'ai préféré les renommer :

- `CA.Tenacite`, pour la ténacité du chef d'atelier
- `OE.Tenacite`, *idem* pour l'ouvrier d'entretien
- `OP.Tenacite`, *idem* pour l'ouvrier de production
- `Reactivite`, pour la réactivité de l'ensemble des agents. On rappelle que la réactivité est la capacité à réagir rapidement face aux stimuli des autres acteurs.

d. Jeu de données secondaire : la gestion du Touch

La particularité principale de ce jeu de données est qu'il présente une organisation avec 10 acteurs et 10 relations. C'est donc une organisation particulièrement complexe.
En fait, le Touch est une rivière de 75 km, affluent de la Garonne. Son débit a d'importantes fluctuations saisonnières et il est sujet à des crues, inondant les communes situées en aval. Aujourd'hui, les différents agents ont pour projet de s'associer afin de mettre en place une gestion globale et solidaire de l'eau. En effet, un projet d'aménagement suite aux crues de février 2003 doit être mis en place, prévoyant la rénovation de digues existantes, l'entretien du lit de la rivière et principalement la création d'une deuxième digue. En conséquence, on attend tout particulièrement un geste des communes situées en amont du Touch pour tenter de « retenir » une partie de l'eau.
Parmi ces acteurs, on trouve :

- DDT_police : Direction Départementale des Territoires de la Haute-Garonne. C'est l'autorité au niveau préfectoral
- ONEMA : Office National de l'Eau et des Milieux Aquatiques
- Agence de l'eau : Agence de l'eau Adour Garonne qui dispose des ressources financières du bassin Adour Garonne
- ARTESA : Association des Riverains du Touch Et de Ses Affluents
- Communes en amont du Touch
- Communes en aval du Touch

- SIAH : Syndicat Inter-Communal pour l'aménagement hydraulique. C'est dans cette structure que l'on décide des aménagements à réaliser
- CR : Conseil Régional (Midi-Pyrénées)
- CG : Conseil Général de la Haute-Garonne
- SOGREAH : Bureau d'étude faisant partie d'une société d'ingénierie intervenant dans les domaines de l'eau, de l'énergie et de l'environnement

Ce sont, à chaque fois les satisfactions des différents acteurs que l'on va prendre en compte. Ensuite, on étudiera les différentes relations :

- `validation_dossier_travaux.State [-10; 10]`: Suivi, instruction et vérification de la conformité. On autorise les travaux ou pas en fonction notamment de ses critères écologiques. Cette relation est détenue par DDT_police.
- `Etude_suivi_conseil.State [-8; 8]`: Cette relation est contrôlée par l'ONEMA. Elle réalise des expertises et peut intervenir grâce à ses pouvoirs de police (contraventions). A l'issue de cette étude, l'ONEMA peut donner un avis défavorable ou favorable au lancement des travaux, sachant que l'ONEMA est particulièrement sensible à la problématique écologique.
- `Financement_conseil_suivi.State [-8; 8]`: L'agence de l'eau peut décider de financer les projets jugés écologiques du SIAH jusqu'à 75%. Cette agence est très attachée au caractère écologique des travaux et ne trouvera satisfaction que si les travaux se réalisent, entérinant ainsi l'influence de l'agence.
- `Connaissance_pression_proprietaires.State [-10; 10]`: C'est l'ARTESA qui contrôle cette relation. Certaines rives appartiennent à des particuliers (des agriculteurs principalement) qui ont alors à leur égard des droits d'usages et des devoirs d'entretien. Ces particuliers sont épaulés par un comité d'experts qui produisent notes et rapports. C'est notamment grâce à cela que ces particuliers peuvent exercer un lobby sur les élus. Les revendications de ces propriétaires n'étant pas forcément en harmonie avec une idéologie écologique, l'ONEMA et l'Agence de l'eau par exemple, ont intérêt à limiter l'impact de l'ARTESA. Cependant, il est difficile de composer sans eux car ils sont propriétaires d'une partie des terrains inondables.
- `Maitrise_de_terrains_pour_expansion_des_crues.State [-8; 8]`: Ce sont les commune en amont qui disposent de cette relation. Elle démontre la capacité des communes en amont à pouvoir discuter des aménagements. Le cas échéant, elles se voient imposer une zone d'expansion de crue. On rappelle qu'une zone d'expansion de crue est une zone inondable mais non urbanisée, qui peut être sujette à des aménagements en vue de prévenir des inondations en aval.
- `Ressources_financieres.State [-8; 8]`: Les communes en aval, grâce à des ressources financière plus importantes que les communes en amont ont le pouvoir de financer les travaux au travers du SIAH. Plus cette relation a une valeur faible, plus les communes en aval sont contraintes de s'engager dans cette gestion des risques en aménageant des espaces. A l'inverse, une ressource élevée permet aux communes en aval de financer des aménagements en amont sans intervention en aval.
- `Exercice_des_competences_rivieres_des_communes.State [-8; 8]`: Cette relation décrit l'influence que peut avoir le SIAH. Soit sa valeur est faible et le SIAH se contente d'assurer l'entretien de la rivière, soit sa valeur est forte et ses activités tendent davantage vers la gestion de risques.
- `Ressources_financieres_complementaires_CR.State [-7; 7]`: Plus la valeur de cette relation augmente, plus la capacité d'action (financement) du conseil régional est forte et plus les travaux tendent à être écologique.
- `Ressources_financieres_complementaires_CG.State [-6; 6]`: Le conseil Général peut décider de financer les travaux, en concurrence avec l'agence de l'eau (qui peut financer jusqu'à 75% des travaux). Les contraintes du Conseil Général sont fortes car les

financements sont apportés en fonction de l'application du règlement intérieur, suivant une logique routinière et bureaucratique. Plus cette relation est élevée, plus les financements tendent à être moins « automatiques » et inscrits dans une logique écologique.

- `production_etude.State [-8; 8]`: Cette étude réalisée par la SOGREAH n'intervient que si elle a été désignée comme fournisseur d'étude. Le type d'étude menée sera de type hydromorphologique (valeurs positives) ou hydraulique (valeurs négatives).
  L'hydraulique mène à la construction de digues, barrages ou déversoirs. Cette approche est jugée moins écologique que l'hydromorphologie et est donc condamnée par un certain nombre d'acteurs. En effet, l'hydromorphologie part du principe que depuis des siècles, l'homme a façonné les cours d'eau à sa manière, rompant l'équilibre et la diversité des milieux. Cette science a donc pour but de s'appuyer sur le fonctionnement et la forme du cours d'eau pour mieux appréhender ses débordements.

2. Analyse des corrélations

En premier lieu, une analyse des corrélations sera réalisée afin de détecter s'il existe des liens entre les variables. Cette analyse donnera lieu à l'écriture d'une fonction, permettant d'exécuter le traitement de manière automatique. En sortie de cette fonction, on obtient un graphique des corrélations très visuel puisqu'il se lit principalement grâce à la couleur et à la forme du symbole rond.
La fonction est nommée CORR(): son script et sa description sont consultables en annexes.

a. Jeu de données SEITA

Pour le jeu de données SEITA, nous avons par exemple ce graphique en sortie :

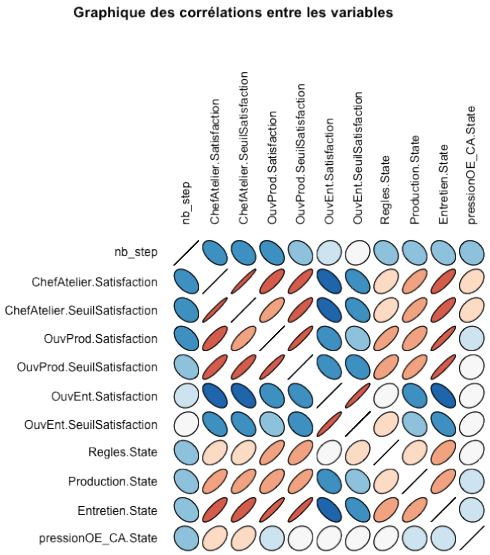


Figure 2: Graphique des corrélations du cas SEITA

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 baisse 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.
Si l'on s'intéresse aux satisfactions des différents 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 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.

b. Jeu de données Bolet

L'analyse des corrélations nous donne ce graphique :

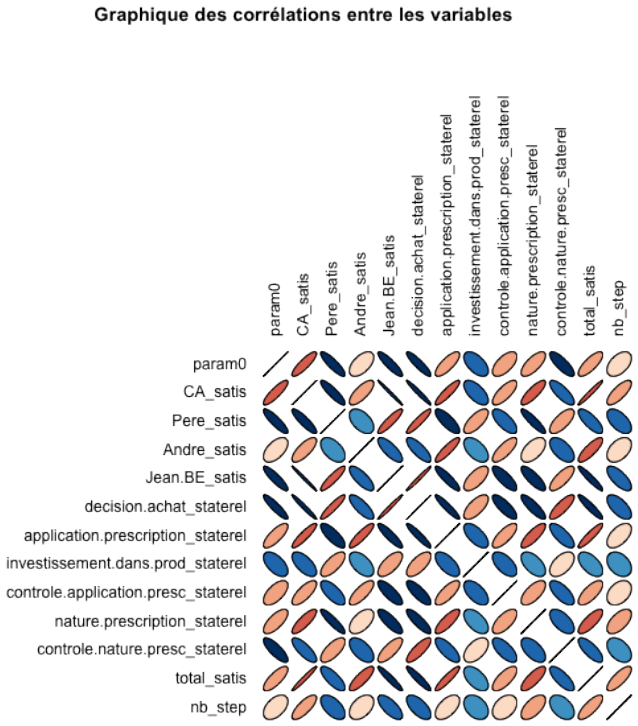


Figure 3: Graphique des corrélations du cas Bolet

A première vue, on constate que les corrélations entre les différentes variables sont nombreuses et assez intenses.

Commençons par étudier les corrélations entre les satisfactions des différents agents.

La corrélation entre la satisfaction du chef d'atelier et celle du père est forte et négative à la fois. Cela signifie que ces deux satisfactions sont assez antagonistes. On remarque qu'André aussi à une satisfaction corrélée négativement à celle de son père. En revanche, elle est positive avec celle du chef d'atelier. En fait, on voit deux groupes de corrélations qui se forment : d'un côté, le père et Jean qui ont tendance à voir leur satisfaction personnelle augmenter quand celle de l'autre augmente et de l'autre André et le chef d'atelier qui sont dans la même situation.

Le nerf de la guerre étant l'achat ou non de la nouvelle machine, en regardant comment cette dernière est corrélée avec les agents, on comprend mieux le schéma sous-jacent : si la machine est achetée, le père et Jean (initiateur de l'achat) voient leur satisfaction augmenter. En revanche, André (directeur de la fabrication) et le chef d'atelier sont insatisfaits car cette nouvelle machine va les obliger à modifier leur manière de produire (augmentation de la planification, travail posté, etc.).

Par ailleurs, on constate que l'achat de la machine influence d'autres relations : si la décision d'achat est forte on constate que l'investissement dans la production l'est également.

En revanche, cette décision diminue la rationalité des prescriptions du BE, forçant André à augmenter les contrôles sur la nature des prescriptions. Une fois que le BE a définit ses prescriptions, elles ont tendance à être moins appliquées par le chef d'atelier tandis qu'André baisse ses contrôles à ce niveau. La décision d'achat a donc des conséquences importantes dans l'entreprise, d'autant plus que globalement, elle fait baisser la satisfaction totale des agents.

Si l'on s'intéresse aux relations, on peut voir que pour augmenter la satisfaction du chef d'atelier, il faut principalement que les prescriptions du BE soient appliquées et que la nature de la prescription

soit la plus rationnelle possible. Sa satisfaction est corrélée positivement avec la satisfaction globale des agents. Si l'on augment la ténacité du chef d'atelier (rappel : sa propension à chercher de nouvelles solutions ou au contraire à exploiter ses connaissances), les contrôles effectués par André au niveau des décision du BE diminuent, l'investissement dans la production est plus faible et la machine a peu de chance d'être achetée.
André est proche du chef d'atelier puisque lui aussi voit sa satisfaction augmenter lorsque les décisions du BE sont bien appliquées.
À l'inverse, on constate que la satisfaction de Jean est principalement provoquée par une mauvaise application des prescriptions du BE, un contrôle des décisions du BE faible et des prescriptions qui tendent à être irrationnelles. C'est tout à fait contradictoire puisqu'il est satisfait lorsque ses propres décisions ne sont pas suivies. On peut alors se demander s'il n'existe pas un effet de « bénéfice transactionnel », qui ferait que Jean obtient finalement plus de bénéfices sur une autre relation en laissant le chef d'atelier mal appliquer les décisions du BE. Tout cela est certainement lié à la décision d'achat, très importante pour Jean, qui pourrait justifier ce lâcher prise au niveau de l'application des décisions qu'il a lui même formulées. On constate d'ailleurs que plus la décision d'achat est forte, plus l'application de la prescription baisse. On pourrait penser que le chef d'atelier utilise le levier qu'il contrôle, l'application des prescriptions du BE, pour faire comprendre à André qu'il rejette la décision d'achat de la machine. Le chef d'atelier entre ainsi en conflit avec André même si ce dernier accepte cette situation car il préfère voir l'achat de la machine se réaliser en dépit d'une mauvaise application de ses décisions plutôt que de voir la décision d'achat diminuer.
Le père présente à peu près les mêmes caractéristiques et la satisfaction de ces deux agents entraine une baisse de la satisfaction totale des acteurs.
Pour finir, si l'on voulait obtenir la meilleure satisfaction globale des agents, il faudrait que la machine ne soit pas achetée, au détriment de père et de Jean et que les prescriptions du BE soit à la fois rationnelles et appliquées au niveau de la chaîne de production.

c. Jeu de données SEITA à 500 simulations

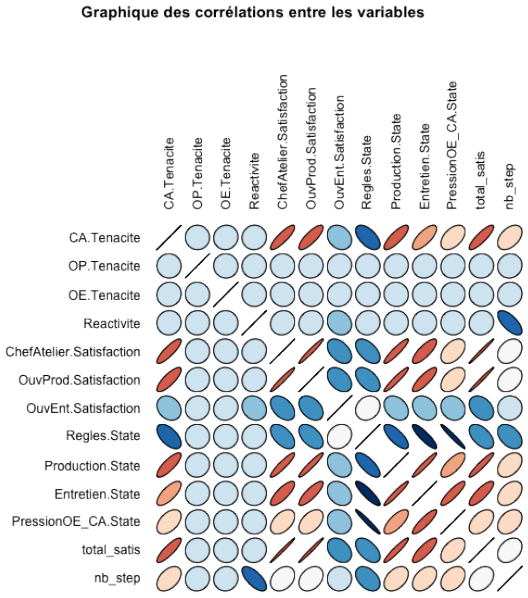


Figure 4 : Graphique des corrélations du cas SEITA à 500 simulations

Au premier regard, on s'aperçoit que le graphique obtenu est plutôt différent du premier. On ne voit plus l'ouvrier d'entretien s'opposer nettement et d'une manière générale, on voit que la ténacité des différents agents n'a pas influencé les autres variables sauf pour le chef d'atelier. Voyons en détail ce que l'on peut tirer de ce graphique :

La satisfaction du chef d'atelier est toujours corrélée positivement à la satisfaction de l'ouvrier de production et négativement à l'ouvrier d'entretien. De même, l'entretien est toujours une relation dominante qui influence fortement la satisfaction des agents même si la production est davantage corrélée aux satisfactions. Si du côté des acteurs, le nombre de simulations n'a pas vraiment changé la donne, il en est autrement des relations.

En effet, si le niveau de respect des règles n'influençait pas vraiment le premier modèle, on voit ici que cette relation est fortement liée à la production, l'entretien et le pression de l'ouvrier d'entretien sur le chef d'atelier. Plus les règles sont respectées, moins les autres relations sont bonnes et inversement. Aussi, plus l'ouvrier d'entretien exerce de pression sur le chef d'atelier et plus l'entretien est fort. On pourrait penser qu'il y a un effet « rébellion » car plus les employés doivent respecter les règles et moins l'usine est productive tandis que plus l'ouvrier d'entretien exerce de pression sur son supérieur, plus il s'applique dans son travail. Il est difficile de trouver une explication à ce comportement des simulations et on peut espérer trouver une réponse grâce aux autres méthodes statistiques.

Concernant les paramètres (ténacité et réactivité), à part la ténacité du chef d'atelier, ils ne semblent pas beaucoup influencer les simulations. La ténacité du chef d'atelier, quant elle est élevée, permet de faire évoluer de manière positive sa satisfaction et celle de l'ouvrier de production, la production, l'entretien ainsi que la satisfaction totale. En fait quand il est tenace, il arrive plus facilement à ses fins. Cependant, on pourrait se demander ce qui fait qu'il n'en est pas de même pour les autres acteurs : seul le chef parvient à s'imposer de cette manière.

d. Jeu de données Touch

Pour étudier ce jeu de données, nous allons procéder à une suppression de variables. En effet, ce jeu de données contient beaucoup d'acteurs et de relations, c'est pourquoi on peut supprimer les seuils de satisfaction des différents acteurs afin d'obtenir un schéma plus lisible sans perte d'information. Cette opération sera effectuée pour chacune des méthodes.

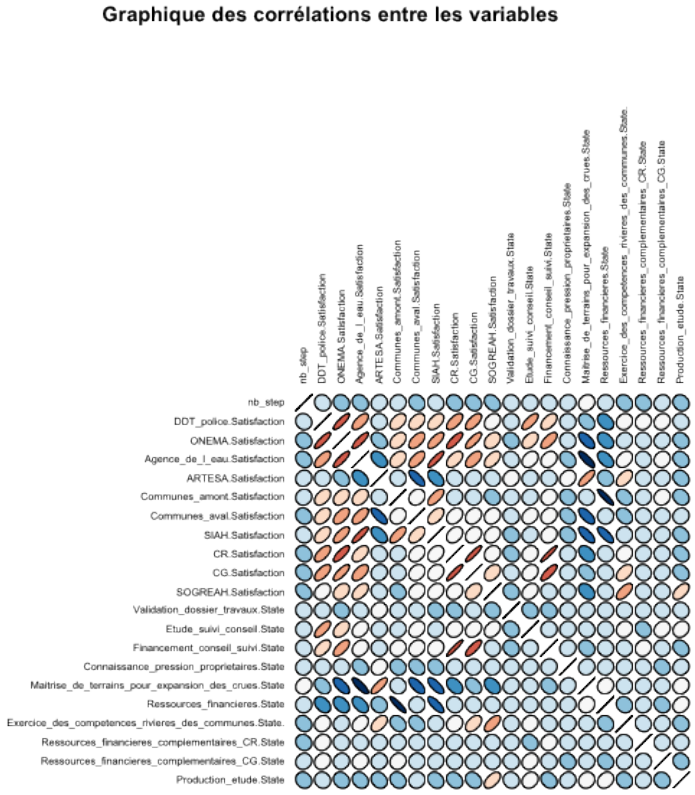


Figure 5 : Graphique des corrélations du cas Touch

Les variables étant nombreuses, on va se concentrer sur les corrélations les plus importantes, en commençant par celles qui existent entre les acteurs.

La satisfaction de la direction départementale des territoires est corrélée positivement à celle de l'ONEMA (Office National de l'Eau et des Milieux Aquatiques). Ces deux institutions sont favorables au lancement des travaux quand ces derniers sont à dominante écologique, ce qui explique leur lien positif.

L'ONEMA présente également une corrélation positive avec l'agence de l'eau et le conseil régional. En effet, plus le conseil régional est impliqué dans les travaux, plus ils seront de nature écologique. De même, l'agence de l'eau soutient un projet non-hydraulique et suit donc les paradigmes de l'ONEMA. En fait, cette institution est satisfaite quand ses projets écologiques trouvent un écho financier auprès de ses partenaires.

On trouve une nouvelle corrélation forte et positive entre le SIAH (Syndicat pour l'Aménagement Hydraulique) et l'agence de l'eau. En effet, si le SIAH parvient à développer ses compétences au delà du simple entretien de la rivière, il pourra atteindre ses objectifs environnementaux chers à l'agence de l'eau.

Ensuite, on constate que l'ARTESA (Associations des Riveraines du Touch Et de Ses Affluents) est corrélée négativement avec tous les autres acteurs. Cette association se révèle être un acteur qui bloque les projets d'aménagements quels qu'ils soient. Cette corrélation négative est encore plus importante avec les communes situées en aval du Touch. On rappelle que ces communes sont celles

qui sont inondées par le Touch mais également celles qui détiennent le plus de ressources financières (par rapport aux communes en amont) pour faire avancer les projets.
Enfin, on peut voir que les deux conseils, général et régional sont corrélés positivement ce qui sous-entend bien que ces deux institutions poursuivent le même but.
Au niveau des relations, on peut voir que la plupart d'entre elles n'affectent que très peu le modèle. Parmi les relations influentes, on peut voir le `financement_conseil_suivi`, détenue par l'agence de l'eau. Elle est fortement corrélée à la satisfaction du CG (Conseil Général) et du CR (Conseil Régional). En fait, plus le financement de l'agence de l'eau est fort, plus les conseils sont satisfaits, certainement parce qu'en conséquence, ils devront mobiliser moins de moyens financiers, tout en sachant que les aménagements réalisés seront à dominante écologique.
La relation `Maitrise_des_terrains_pour_expansion_des_crues` semble être assez problématique puisqu'elle est corrélée négativement avec un certain nombre d'agents : l'ONEMA, l'agence de l'eau, le SIAH, les communes en aval. Rappelons que cette relation définit la capacité des communes en amont à donner leur avis en terme d'aménagement. Plus cette maîtrise est forte, plus les communes peuvent intervenir dans la prise de décision et ainsi, empêcher la mise en place de terrains d'expansion de crues, jugés nécessaires par les autres agents. Un terrain d'expansion de crue est un espace naturel ou aménagé permettant aux eaux de débordement de se répandre lors d'une crue. Les communes en amont, si elles le peuvent, préféreront garder ces terrains pour se développer.
Les ressources financières gérées par les communes en aval sont également problématiques : plus elles sont importantes, plus la satisfaction du SIAH et des communes en amont sont faibles. En effet, l'apport de ressources représente un pouvoir décisionnel accru et le syndicat de propriétaires comme les communes en amont ne souhaitent pas se voir imposer de décisions.

3. Analyse de la dispersion

L'analyse de la dispersion permet de voir quelle est la variabilité entre les individus, c'est-à-dire l'étendue des valeurs que peut prendre la variable. On pourra savoir si les individus sont groupés ou dispersés (*i.e.*: si les valeurs sont homogènes ou hétérogènes) et repérer les individus atypiques.
Pour cela, nous allons utiliser une méthode connue sous l'abréviation MDS (méthode expliquée dans la section « Méthodes et outils ») qui va permettre de tracer le nuage de points de la matrice centrée réduite des distances entre les observations.

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 est approximativement égale à la valeur de leur dissimilarité. 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.
Afin de pouvoir détecter la présence d'individus atypiques ou de données aberrantes, il a été décidé, dans un second temps, d'ajouter le tracé d'un bagplot aux différents MDS. Ce dernier est construit à partir de la matrice obtenue via la méthode MDS et permet de visualiser rapidement s'il existe des outliers.

Cette fonction se nomme MDS() et elle accepte en entrée le fichier au format csv regroupant les données. Il est possible de préciser quelles variables on souhaite exclure de l'étude ou de la coloration. Le bagplot est optionnel, il faut passer le booléen à « TRUE » pour le voir s'afficher. Enfin, il est possible de sauvegarder les sorties grâce à un dernier paramètre d'entrée. Je rappelle que le script est disponible en annexe et que le manuel utilisateur est disponible en PDF.

a. Jeu de données SEITA

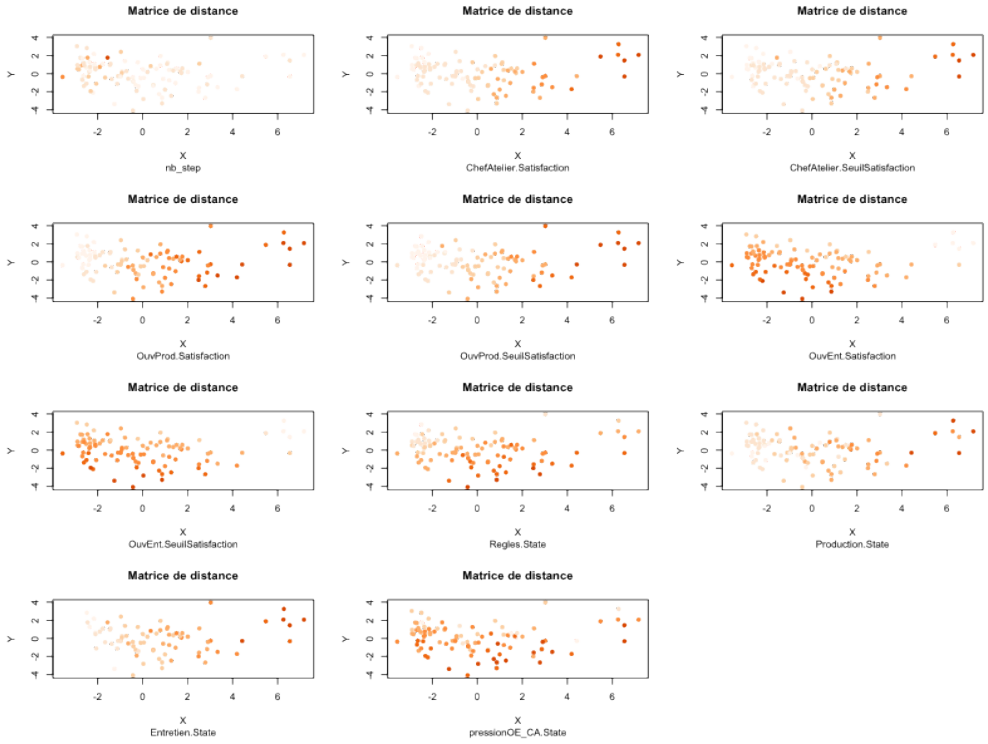


Figure 6: Méthode MDS sur SEITA

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 le paramètre 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. On va donc chercher les variables qui permettraient d'expliquer les trois groupes.
On remarque que les satisfactions et seuils de satisfaction des chefs d'ateliers et ouvriers de production expliquent bien la formation des 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 le respect des 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.

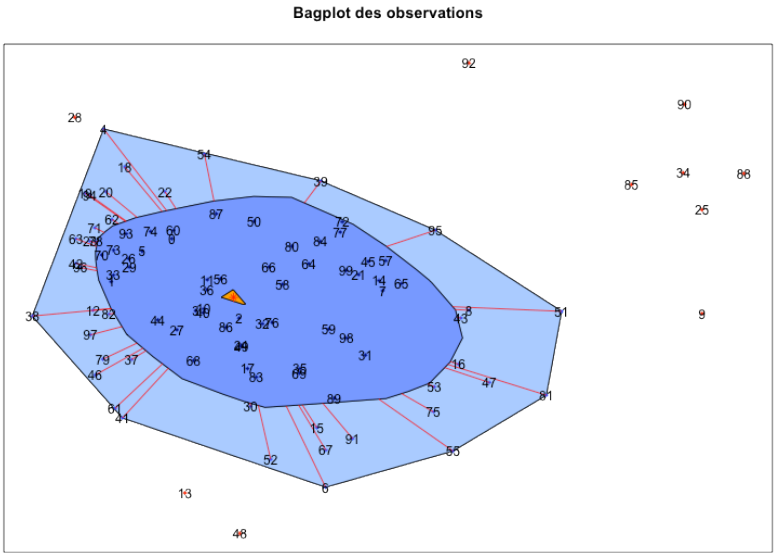


Figure 7: Bagplot du jeu de données SEITA

Chacun des individus, qui correspondent aux 100 simulations étudiées, ont été à nouveau représentés sur un nuage de points en deux dimensions, présentant les distances entre chaque individu.
À 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.

b. Jeu de données Bolet

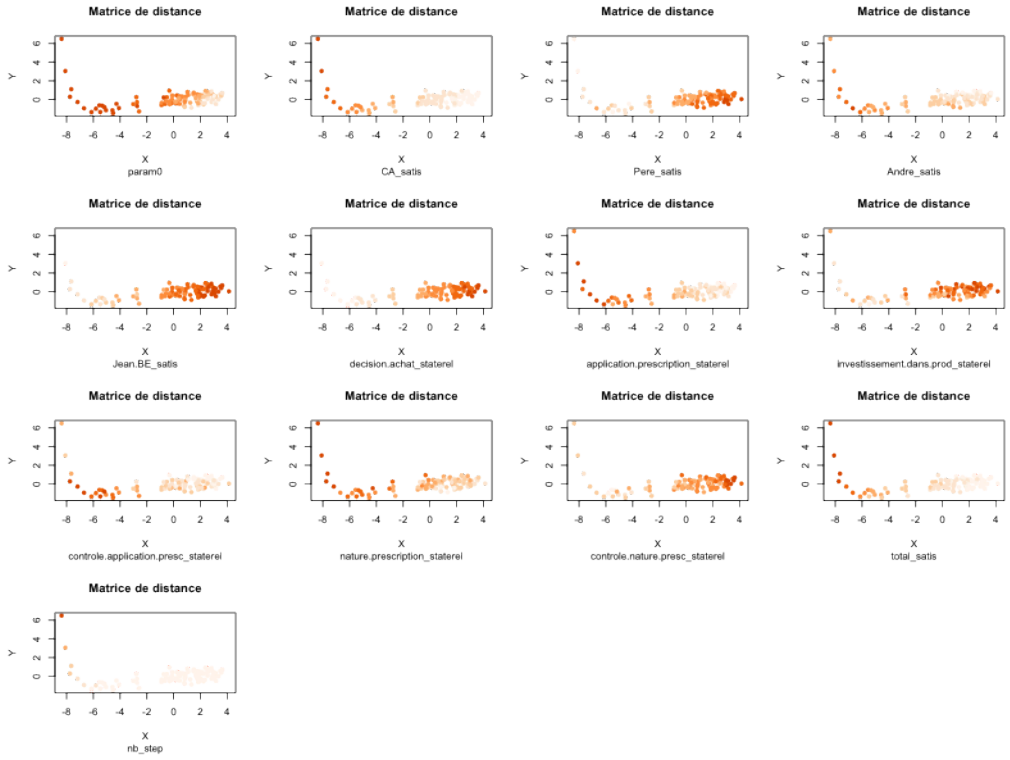


Figure 8 : Méthode MDS sur Bolet

Concernant la forme globale du nuage de points, on constate que celui-ci semble avoir une « queue » et propose des individus atypiques.
En regardant les couleurs des différentes variables, on s'aperçoit qu'il est possible de déceler une organisation sous-jacente à cette forme de nuage.
On peut voir que les variables qui semblent le plus influencer la forme du nuage de points sont la satisfaction du chef d'atelier et la décision d'achat. En effet, la variable `décision-achat` coupe nettement l'agglomérat de points en deux, tandis que la queue du nuage de points propose un dégradé de couleur. À l'inverse, la satisfaction du chef d'atelier sépare la queue en deux.
La concentration d'individus localisée à droite est principalement caractérisée par une faible satisfaction du chef d'atelier causée par une décision d'achat forte. Plus l'on se décale à gauche sur l'axe des abscisses, plus la décision d'achat de la machine faiblit.
On constate que certaines variables comme l'investissement dans la production, le contrôle de l'application des prescriptions, la nature de la prescription et le contrôle de la nature de la prescription ne semble pas influencer l'organisation du nuage de points.
Les autres variables permettent de comprendre que la queue est composée des simulations pour lesquelles la décision d'achat est très faible entraînant une forte satisfaction du chef d'atelier et d'André ainsi qu'une application des prescriptions élevée. Les simulations de la queue présentent une satisfaction totale supérieure aux autres simulations tandis qu'elles tendent à avoir besoin de plus d'itérations pour converger. Enfin, on constate que les simulations où le chef d'atelier est le plus tenace sont massées à gauche du nuage de points. En fait, on pourrait en déduire que plus le

chef d'atelier est tenace, plus il a de chances de voir son souhait se réaliser : la machine ne sera pas achetée. Par ailleurs, dans cette configuration, il a tendance à appliquer correctement les prescriptions du BE, dictées par Jean alors que, paradoxalement, son équipe devient moins productive.

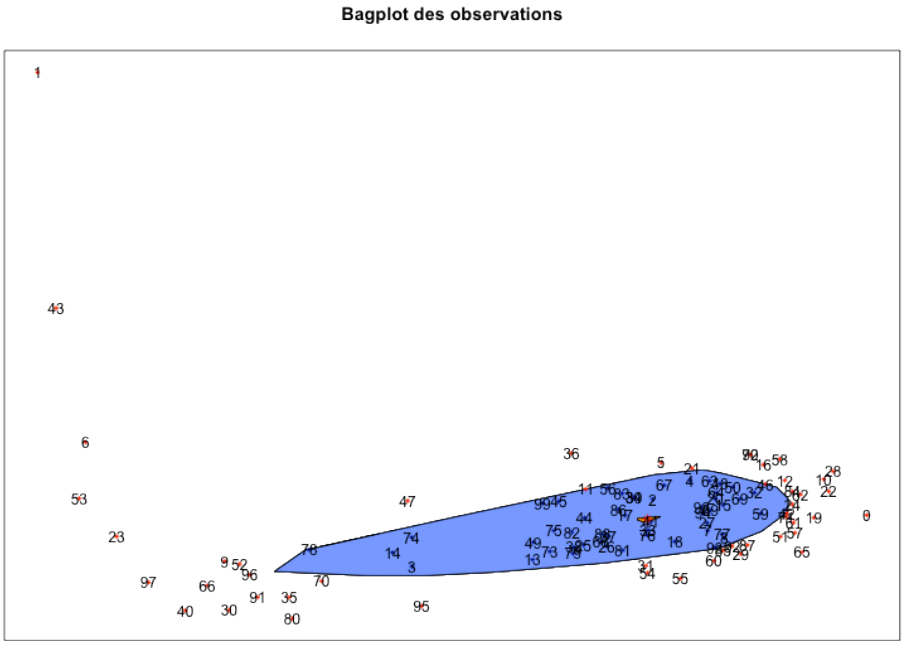


Figure 9 : Bagplot du jeu de données Bolet

Le bagplot montre bien la forme allongée de ce nuage points. On voit qu'il y a beaucoup de simulations concentrées à droite de l'axe des abscisses et plus on va vers la gauche, plus les simulations ont tendance à être éparses et à s'élever sur l'axe des ordonnées. La médiane bivariée est située au centre de la concentration de simulations. A l'opposée de cette dernière, on trouve la simulation numéro 1 qui semble être très atypique.

c. Jeu de données SEITA à 500 simulations

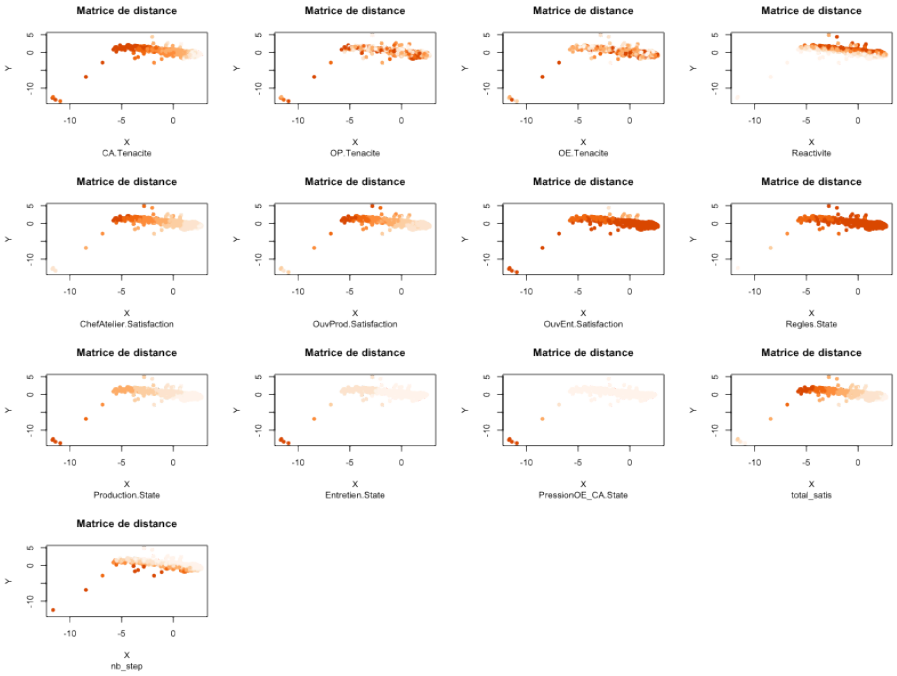


Figure 10 : Méthode MDS sur SEITA à 500 simulations

On peut tout d'abord confirmer que les 3 paramètres ténacité de l'ouvrier de production, ténacité de l'ouvrier d'entretien et réactivité de l'ensemble des acteurs n'influencent pas la forme du nuage de points.
Si l'on regarde les acteurs, il y a une information très importante : dans ces simulations, l'ouvrier d'entretien est presque toujours satisfait. Si les simulations où l'ouvrier d'entretien n'était pas satisfait étaient marginales dans le premier jeu de données, cette situation devient extrêmement rare ici. De la même manière, le respect des règles s'est imposé dans quasiment toutes les simulations.
Les variables qui ordonnent le nuage de points sont la ténacité du chef d'atelier, sa satisfaction, la satisfaction de l'ouvrier de production et la satisfaction totale des agents. Ces quatre variables étant toutes corrélées positivement, on voit le même dégradé de couleur sur tous les nuages. On pourrait voir 4 classes en tout.
Dans le coin inférieur gauche, on distingue quelques points isolés. Pour ces simulations, la production, l'entretien et la pression sont élevé est le chef d'atelier et l'ouvrier de production ne sont pas satisfaits.
Enfin, on peut constater que la réactivité et le nombre de pas ont deux nuages de points qui semblent se compléter : on pourrait dire que plus la réactivité augmente, plus le nombre de pas diminue. Cela semblerait tout à fait logique car si les agents sont prompts à réagir rapidement face aux décisions des autres agents, l'algorithme ne devrait pas avoir besoin de nombreuses itérations pour converger. On constate que cette relation avait bien été mise en lumière sur le graphique des corrélations.
Voyons maintenant le bagplot :

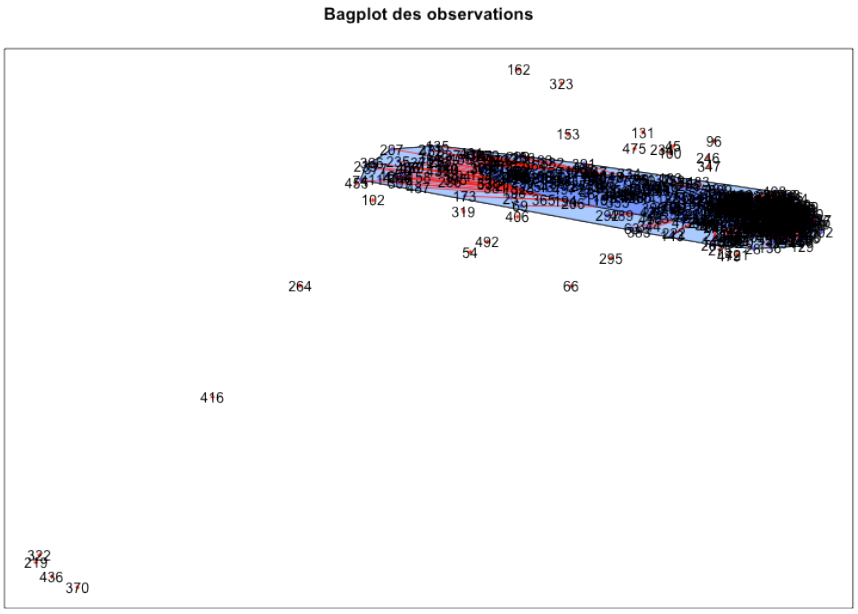


Figure 11 : Bagplot du jeu de données SEITA à 500 simulations

Si l'on prend en compte le fait qu'il y ait 500 simulations, on peut constater qu'au final, il y assez peu d'individus marginaux. On voit une forte concentration de simulations dans la partie droite du nuage de points tandis que plus l'on va vers la gauche, plus on rencontre de points atypiques.

d. Jeu de données Touch

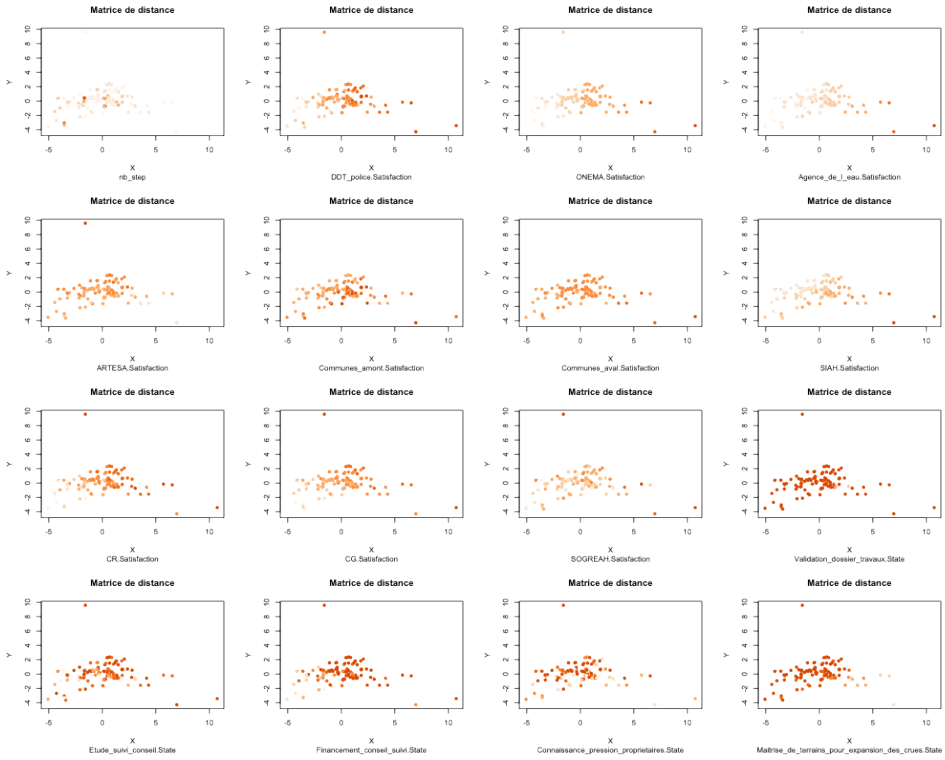


Figure 12 : Méthode MDS sur jeu de données Touch (1)

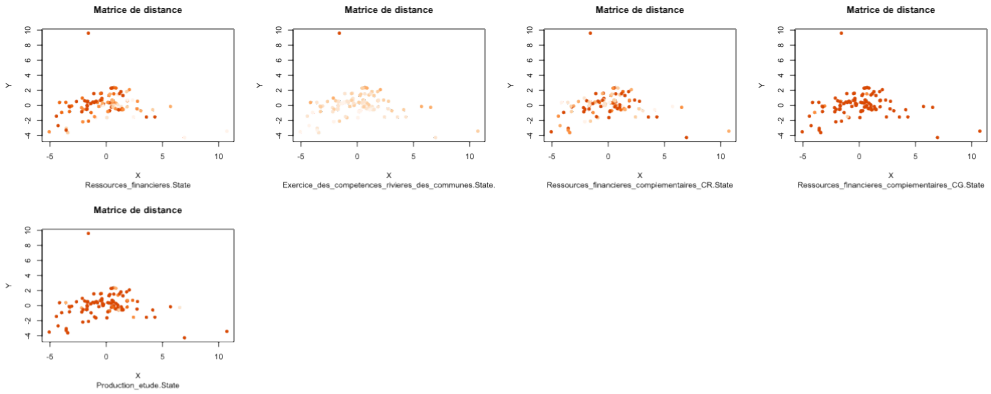


Figure 13 : Méthode MDS sur jeu de données Touch (2)

Parmi les acteurs, ceux qui semblent le plus influencer la forme du nuage de points sont l'ONEMA, le SIAH et l'Agence de l'eau. Ils divisent le nuage de points en 3 groupes le long de l'axe des abscisses avec, à gauche, les valeurs faibles et à droite, les simulations pour lesquelles leur satisfaction est bonne. L'Agence de l'eau et la SIAH présentent finalement assez peu de simulations avec une satisfaction très élevée.

Cependant, du côté des relations, on remarque qu'il y en a plusieurs pour lesquelles les valeurs obtenues pour les différentes simulations sont très élevées pour quasiment toutes les simulations, bien que cela ne permette pas d'expliquer l'organisation du nuage de points. Cela est certainement du à grande dissymétrie des observations : en effet, les classes ont la même amplitude et la présence d'une ou deux valeurs négatives fortes perturbent fortement cette échelle. Pour autant, il est

possible que la variable influence bien la formation du nuage de points mais qu'à cause de ces valeurs extrêmes, on ne puisse le voir.
Parmi ces relations, on trouve par exemple la validation_dossier_travaux qui correspond à l'autorisation des travaux, la maîtrise des terrains pour l'expansion des crues, ce qui signifie que les communes en amont ont globalement une grande capacité à discuter les aménagements, les ressources financières du CG qui ont tendance à être importante ou encore la production d'étude qui a davantage tendance à être de type hydromorphologique plutôt qu'hydraulique.
A l'inverse, on voit que les compétences du SIAH ont tendance à rester cantonnée à l'entretien de la rivière (variable `Exercice_des_competences_rivieres_des_communes.State`). On retiendra donc de l'étude du MDS que quelques variables organisent le nuage de points en trois groupes tandis que l'on voit certaines autres variables obtenir dans quasiment toutes les simulations de fortes valeurs.

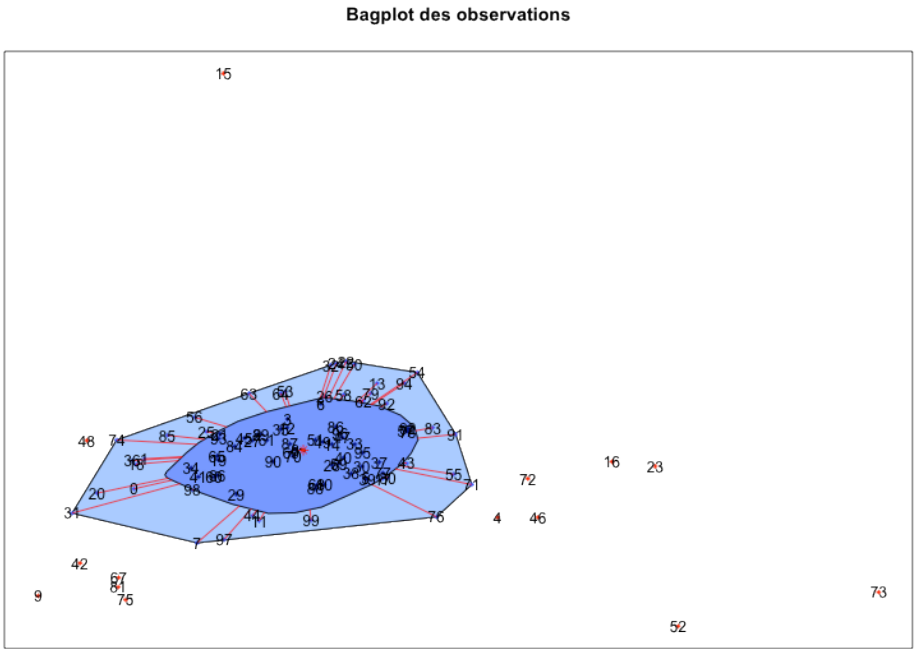


Figure 14 : Bagplot du jeu de données Touch

Le tracé de ce bagplot met particulièrement en valeur les simulations 15 et 17 ainsi que deux groupes de données situés à droite et à gauche du nuage de points. On peut voir que ce dernier est étendu le long de l'axe des abscisses.

4. Classification hiérarchique

Le but de la classification est de simplifier le profil des individus en les 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 (méthode expliquée dans la partie « Méthodes et outils »). La méthode statistique utilisée pour la construction de classes est la classification ascendante hiérarchique (CAH).

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 classes 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 telles 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.

Une fonction nommée CAH() a été créée afin de réaliser cette étude de manière automatique. Grâce à celle-ci, il est possible de spécifier le nombre de classes que l'on souhaite étudier (on peut choisir de ne fixer qu'une classe pour une étude en profondeur ou d'en tester plusieurs à la fois pour fixer le nombre de classes le plus pertinent).
Une fois le nombre de classes fixé, il est possible d'obtenir le tableau d'analyse des moyennes, soit en format csv, soit de manière résumée sous forme graphique. Enfin, il est possible de sauvegarder chacune de ces sorties.

a. Jeu de données SEITA

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.

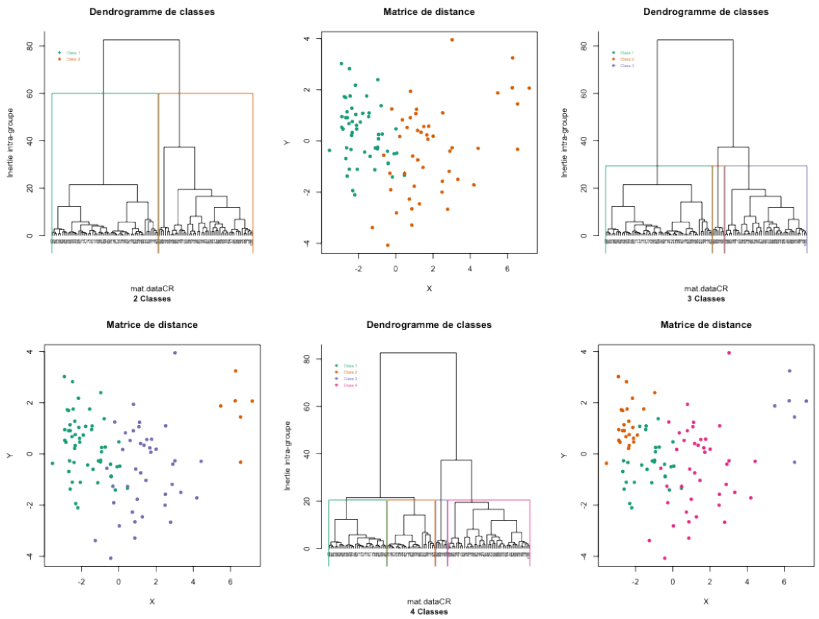
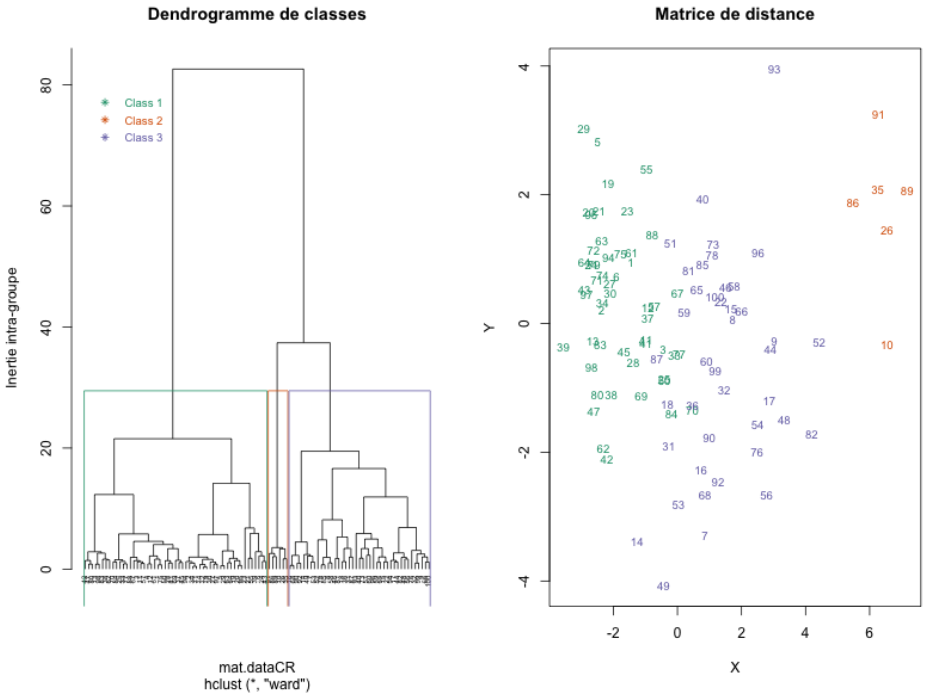


Figure 15: Plusieurs classifications possibles pour le jeu de données SEITA

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 peut lire la valeur de l'inertie intra-groupe 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 faibles.

Selon que la partition ait été réalisée pour deux, trois ou quatre classes, l'inertie intra-groupe est respectivement de 60, 30 puis 20. En effet, l'inertie intra-classe est une fonction décroissante du nombre de classes.
Le choix du nombre de classe est finalement subjectif puisque l'on va le choisir le nombre de classes *k* tel que le passage de *k* à *k+1* ne diminue plus de manière forte l'inertie intra-groupe.
Dans le cas des données SEITA, on va s'attacher à avoir un nombre de classes faibles et à la 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* 3 types de fonctionnement de l'organisation est suffisant. Par ailleurs, l'inertie intra-groupe est assez basse.
Maintenant que l'on a fixé le nombre de classes, on va tenter de décrire chacune d'entre elle afin



d'en connaître plus sur les différentes modalités de fonctionnement de l'organisation.

Figure 16: Classification ascendante hiérarchique à 3 classes du jeu de données SEITA

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, j'ai mis en place un outil, un tableau d'analyse des moyennes, disponible sous deux formes : l'une sous forme numérique (format csv), l'autre sous forme graphique (format png).

	nb_step	ChefAtelier.Satisfaction	ChefAtelier.SeuilSatisfaction	OuvProd.Satisfaction	OuvProd.SeuilSatisfaction	OuvEnt.Satisfaction	OuvEnt.SeuilSatisfaction	Regles.State	Production.State	Entretien.State	pressionOE_CA.State	Effectifs
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

Tableau 1: Tableau d'analyse des moyennes du jeu de données SEITA

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.
La lecture de ce tableau n'étant pas directe, j'ai mis en place une seconde version de ce dernier, beaucoup plus visuelle :

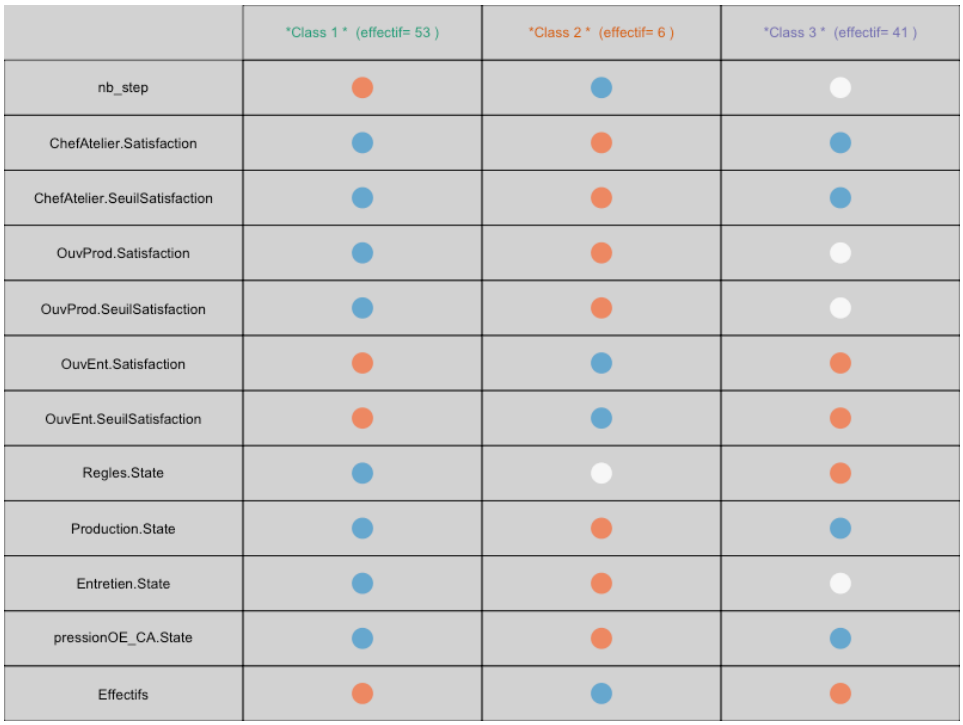


Figure 17 : Tableau d'analyse des moyennes du cas SEITA

J'ai opté pour un code couleur simple : plus la couleur tend vers le rouge, plus la valeur de la moyenne est élevée par rapport aux autres. A l'inverse, plus la pastille avance dans les nuances de bleu, plus la moyenne est faible par rapport aux autres.
La pastille blanche n'existe que pour les nombres de classes impairs : elle correspond aux valeurs centrales de la moyenne.

On peut décrire les classes de cette manière:
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 de 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 2 présente les caractéristiques inverses de la classe 1.
En effet, pour ces simulations qui convergent très rapidement, les satisfactions des ouvriers de production et du chef d'atelier sont très bonnes. On peut s'apercevoir que l'entretien et 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 ce type de simulation n'apparait 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.

En fait, en étudiant l'analyse de cas réalisée par les sociologues pour le cas SEITA, on sait qu'entre les ouvriers d'entretien et les chefs d'atelier, il existe des relations que l'on peut qualifier d'hostiles. Pour chacun des agents, la charge émotionnelle est forte et l'on constate que les ouvriers d'entretien sont davantage satisfaits de leur travail quand ils sont agressifs à l'égard de leur chef d'atelier. Par ailleurs, on sait également que ce sont en réalité les ouvriers d'entretien qui sont les « vrais chefs » de l'entreprise, puisqu'ils contrôlent la seule ressource aléatoire de l'usine : les machines. En effet, lorsque ces dernières tombent en panne, c'est l'ouvrier d'entretien qui possède le pouvoir de bien la réparer ou non, de le faire plus ou moins rapidement, etc. De cette manière, il contrôle la production de l'usine et a donc le plus grand pouvoir.

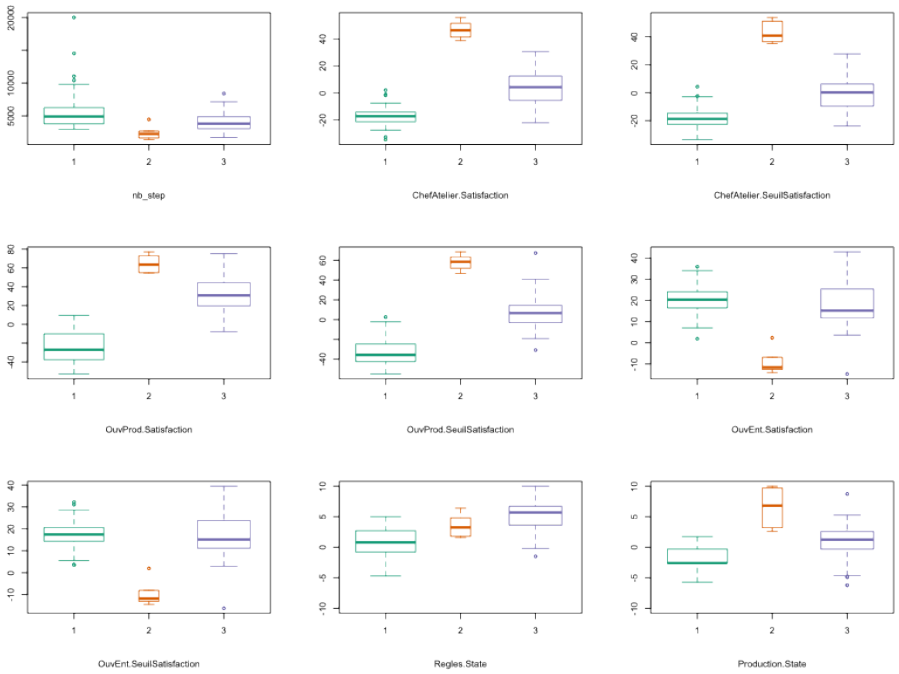


Figure 18 : Boîtes à moustache du jeu de données SEITA (1)

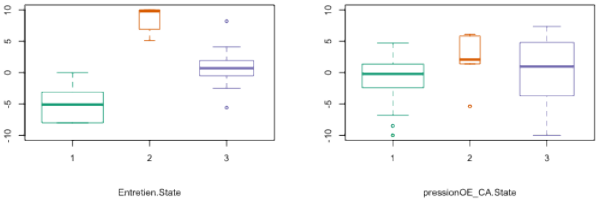


Figure 19 : Boîtes à moustache du jeu de données SEITA (2)

Chaque boîte à moustaches 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 boîte à moustaches concernant la satisfaction des ouvriers d'entretien dans les classes 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.
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.

b. Jeu de données Bolet

L'étude du MDS nous avait montré l'existence de deux groupes de simulations, l'un correspondant à un agglomérat de points, l'autre à une queue. On a vu que le premier était scindé en deux à cause de deux variables principalement : la satisfaction du chef d'atelier et la décision d'achat. Ensuite, on a pu constater que la queue du nuage de points avait tendance à monter le long de l'axe des ordonnées.
Nous allons donc, dans un premier temps, tester des classifications pour le jeu de données Bolet avec 3, 4 et 5 classes. En effet, il est possible qu'il soit pertinent de couper la queue en deux groupes.

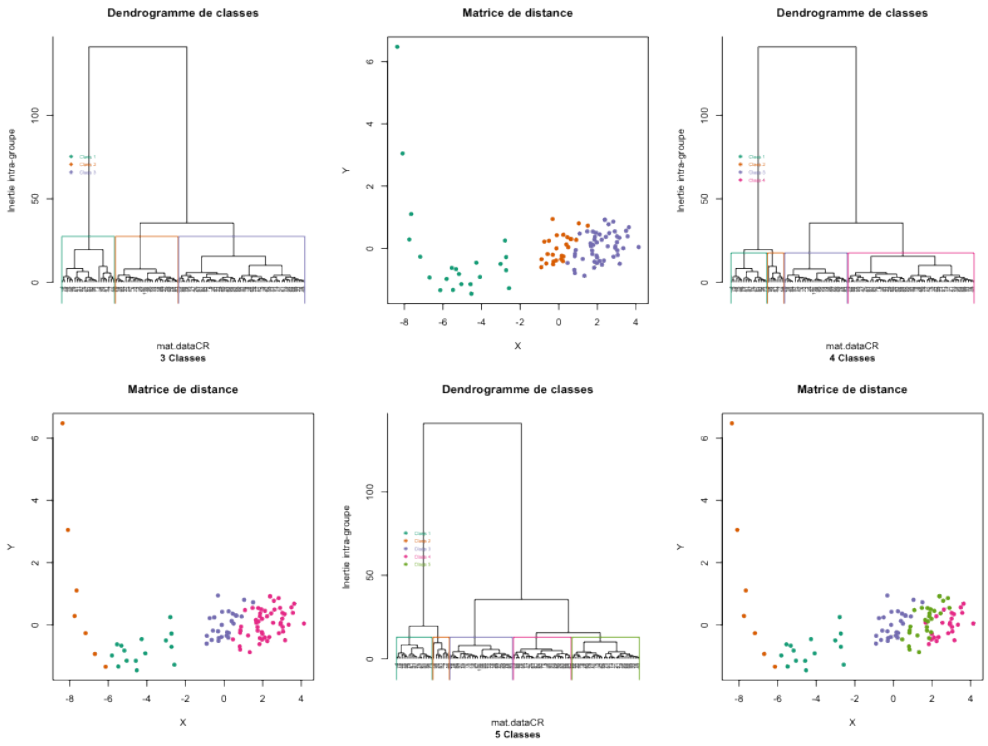


Figure 20 : Plusieurs classifications possibles pour le jeu de données Bolet

On constate que, comme prévu, le passage à 4 classes provoque la séparation de la queue en 2 classes. Cette organisation paraît préférable car, en plus de faire diminuer la variance intra-groupe, on s'aperçoit que ces simulations sont bien différentes : en effet, la classe 2 (orange) semble présenter des observations isolées qui montent progressivement le long de l'axe des ordonnées alors que la classe 1 (verte) semble être davantage la continuité de l'agglomérat de points situé dans l'angle à droite.
Maintenant, on pourrait se demander si le passage à 5 classes est pertinent. Su le graphique, on voit que les 5 classes sont bien constituées, c'est-à-dire qu'elles sont correctement délimitées et qu'elles ne s'emboîtent pas les unes dans les autres. Cependant, l'étude du MDS nous avait montré une tendance de ce nuage à se scinder en deux et pas en trois classes, c'est pourquoi, il semble préférable d'en rester à 4 classes. Enfin, la diminution de l'inertie intra-groupe est bien moins significative que lors du passage de 3 à 4 classes.
Pour la suite de l'étude, on choisira donc d'étudier 4 classes.

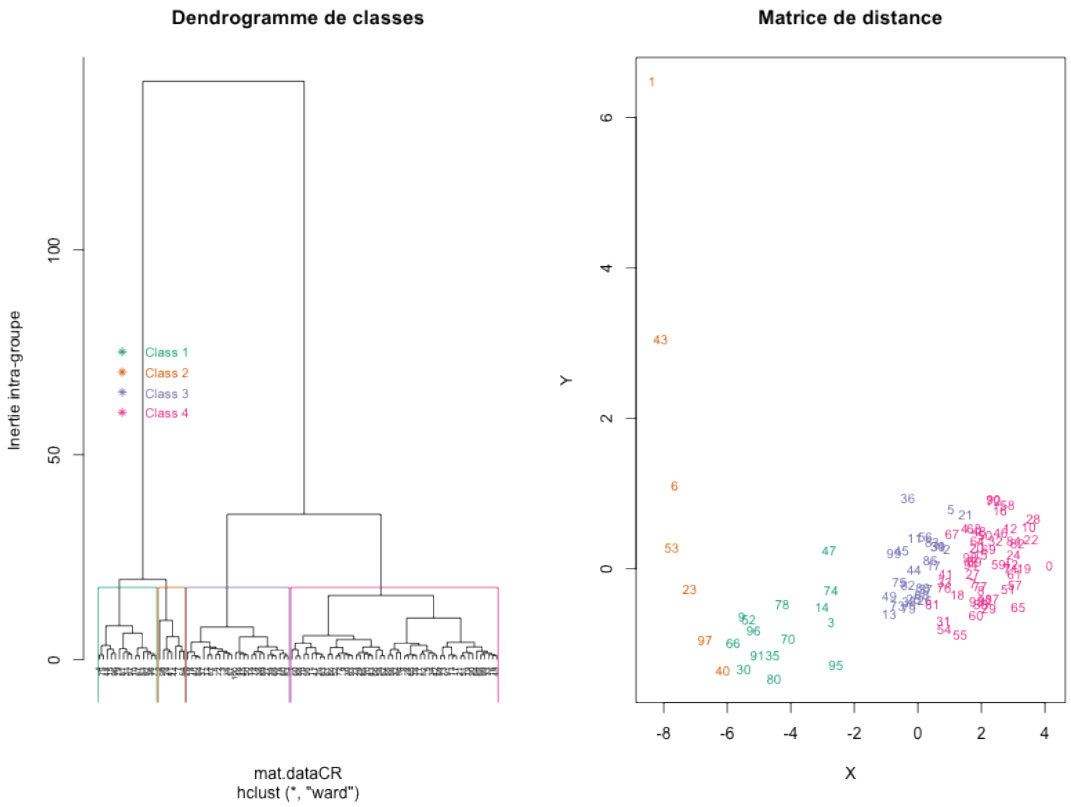


Figure 21 : Classification ascendante hiérarchique à 4 classes du jeu de données Bolet

Pour étudier en détails cette classification, utilisons le tableau d'analyse des moyennes :

	*Class 1 * (effectif= 15)	*Class 2 * (effectif= 7)	*Class 3 * (effectif= 26)	*Class 4 * (effectif= 52)
param0	●	●	●	●
CA_satis	●	●	●	●
Pere_satis	●	●	●	●
Andre_satis	●	●	●	●
Jean.BE_satis	●	●	●	●
decision.achat_staterel	●	●	●	●
pplication.prescription_statere	●	●	●	●
vestissement.dans.prod_state	●	●	●	●
ntrole.application.presc_state	●	●	●	●
nature.prescription_staterel	●	●	●	●
controle.nature.presc_staterel	●	●	●	●
total_satis	●	●	●	●
nb_step	●	●	●	●
Effectifs	●	●	●	●

Figure 22 : Tableau d'analyse des moyennes du cas Bolet

On pourrait donc décrire les classes de cette manière :

Classe 1 : C'est une classe d'envergure moyenne puisqu'elle regroupe 15 individus formant le début de la queue du nuage de points. Ces simulations ont mis peu de temps à converger. Cette classe présente une ténacité du chef d'atelier et un contrôle de l'application des prescriptions maximaux. Pour cette classe, la décision d'achat est faible, entraînant l'insatisfaction du père et de Jean. On constate que la nature des prescriptions tend à être rationnelle tandis que les contrôles sur cette dernière ont un niveau très bas. Si le chef d'atelier est assez satisfait, l'investissement dans la production a un niveau bas même si les recommandations du BE sont assez bien suivies. Enfin, la satisfaction globale des agents est plutôt mauvaise.
Classe 2 : Cette classe qui forme ce que l'on a nommé la queue du nuage de points présente à peu près les mêmes caractéristiques que la classe précédente sauf que la valeur des paramètres est exacerbée. Il n'y a pas de valeurs moyennes dans cette classe en comparaison aux autres. C'est dans cette classe qu'André et le chef d'atelier obtiennent la meilleure satisfaction puisque la machine a très peu de chance d'être achetée. Quant aux relations que ces deux agents contrôlent, on constate que le chef d'atelier fait appliquer à la lettre les recommandations du BE alors que son équipe est très peu investie dans la production. Du côté d'André, on constate qu'il fait un contrôle tatillon des prescriptions du BE alors qu'au même moment, il n'exerce que peu de contrôle sur ces mêmes décisions. Peut-être que lorsque le chef d'atelier est très tenace, il fait l'objet d'une surveillance rapprochée de la part d'André, qui vérifie que les prescriptions sont bien appliquée tandis que l'équipe de production devient moins productive.
Par ailleurs, c'est cette classe qui met le plus de temps à converger certainement à cause de la ténacité accrue du chef d'atelier alors que, simultanément, la satisfaction totale des agents est la meilleure.
Classe 3 : La classe 3 forme la partie gauche de l'agglomérat de points. C'est une classe d'effectif moyen dans laquelle le chef d'atelier est moyennement tenace. La décision d'achat de la machine est plutôt favorable à l'encontre des désirs du chef d'atelier et d'André.
On remarque que pour ces simulations, l'investissement dans la production est très élevée tandis que les prescriptions du BE, qui ont tendance à être peu rationnelles, ont tendance à être ignorées, cette situation étant favorisée par un faible taux de contrôle à ce niveau. Ces simulations ont convergé assez rapidement pour une satisfaction totale des agents insuffisante.
Classe 4 : On repère rapidement que la classe 4, qui comprend un peu plus de 50% des effectifs totaux est l'exact contraire de la classe 2, la queue du nuage de points.
Ici, le chef d'atelier a abandonné sa ténacité provoquant une décision d'achat de la machine forte. En conséquence, Jean et son père sont très satisfaits.
En ce qui concerne les relations, les décisions du BE sont surveillées de près par André même si cela n'empêche pas ces dernières de tendre vers l'irrationalité. La productivité est bonne même si André à tendance à ne pas contrôler l'application des prescriptions du BE au niveau de la production.

c. Jeu de données SEITA à 500 simulations

Après avoir longuement hésité entre 3 et 4 classe, j'ai décidé, après visualisation des dendrogrammes, de prendre 3 classes. Cela devrait être suffisant pour faire apparaître des comportements typiques et le passage à 4 classes ne diminuait plus « significativement » l'inertie intra-groupe.

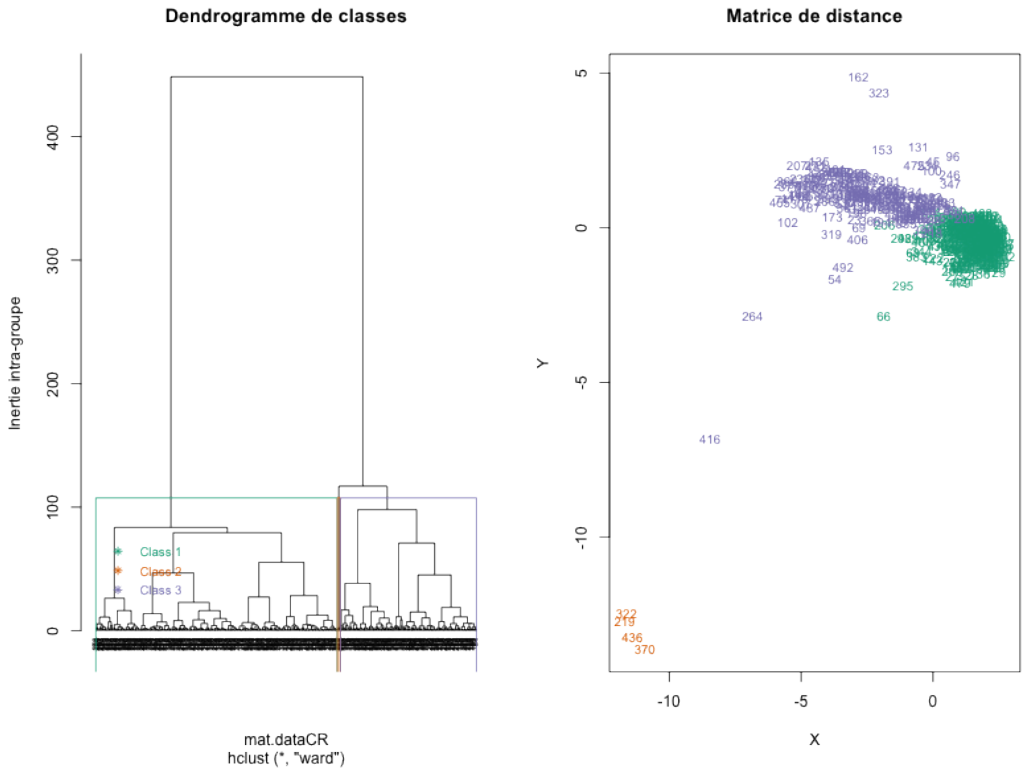


Figure 23 : Classification ascendante hiérarchique à 3 classes du jeu de données SEITA à 500 simulations

On constate qu'en raison du grand nombre de données, l'inertie intra-groupe reste élevée.

	*Class 1 * (effectif= 317)	*Class 2 * (effectif= 4)	*Class 3 * (effectif= 179)
CA.Tenacite	●	●	●
OP.Tenacite	●	●	●
OE.Tenacite	○	●	●
Reactivite	●	●	●
ChefAtelier.Satisfaction	●	●	●
OuvProd.Satisfaction	●	●	●
OuvEnt.Satisfaction	●	●	●
Regles.State	●	●	●
Production.State	●	●	●
Entretien.State	●	●	●
PressionOE_CA.State	●	●	●
total_satis	●	●	●
nb_step	●	●	●
Effectifs	●	●	○

Figure 24 : Tableau d'analyse des moyennes du cas SEITA à 500 simulations

Classe 1 : La classe 1 est l'ensemble de points qui a été repéré lors de l'étude du MDS. C'est la classe la plus importante numériquement. Ce groupe est caractérisé par une bonne réactivité des différents agents même si seul l'ouvrier d'entretien est satisfait. La production, l'entretien et la pression sont faibles pour ces simulations qui ont tendance à converger rapidement. En fait, cette classe est l'association des simulations qui tournent à l'avantage de l'ouvrier d'entretien.
Classe 2 : Cette classe regroupe les simulations où les agents sont les plus tenaces. Cette ténacité entraîne un grand nombre de pas avant la convergence. C'est une classe qui est composée d'à peine 4 individus. En regardant de plus près les valeurs des variables de ces individus, on se rend compte que 3 d'entre eux ont une valeur égale à 0 pour l'entretien, la production, le respect des règles et la pression, tandis que pour toutes les autres simulations, toutes ces valeurs ont tendance à être négatives. Cela fait de cette classe celle qui a les plus grandes valeurs pour les relations même si, en réalité, elles sont tout de même très faibles.
Le nombre de pas pour parvenir à une convergence est très élevé.
L'ouvrier d'entretien est encore l'agent pour qui la satisfaction est la meilleure.
Ces simulations présentant des agents particulièrement tenaces est très atypique puisque seulement 4 simulations sur 500 ont mené à ce résultat pour ce test.
Classe 3 : Ce troisième groupe est de taille moyenne. Seul le chef d'atelier est tenace et comme on l'avait vu dans le graphique des corrélations, cette ténacité engendre la satisfaction de cet agent et celle de l'ouvrier de production. Attention néanmoins à bien interpréter les données : on voit ici que les satisfactions de ces agents sont maximales dans ces groupes, contrairement à l'ouvrier d'entretien. Cela ne veut pas dire que la satisfaction de l'ouvrier de production est meilleure que celle de l'ouvrier d'entretien. Au contraire, si l'on regarde les chiffres, on voit que c'est toujours l'ouvrier d'entretien le plus satisfait : il faut bien garder à l'esprit que cette coloration est relative avec les valeurs obtenues dans les autres classes.
Même si on constate une légère amélioration des satisfactions de ces agents, entraînant l'augmentation de la satisfaction globale, on constate que l'ouvrier d'entretien est encore le gagnant de ces simulations.

On a l'impression que le fait d'être passés de 100 à 500 simulations a exacerbé l'hégémonie de l'ouvrier d'entretien. Il est plus fort que jamais puisqu'il est pleinement satisfait quasiment à chaque fois.

Classe	ChefAtelier.Satisfaction	OuvProd.Satisfaction	OuvEnt.Satisfaction
1	-53,49	-5,16	81,77
2	-54,5	-4,42	81,97
3	-48,73	-1,99	81,34

Figure 25 : Extrait du tableau d'analyse des moyennes

Il n'y a plus de situation de compromis où l'ouvrier de production et le chef d'atelier obtiennent gain de cause.
Pour toutes les relations sauf le respect des règles, la valeur moyenne est négative dans toutes les classes. Cela veut dire que même si, par exemple, on a constaté de meilleures valeurs pour les relations dans la classe 2, en réalité, ces valeurs sont négatives.
On a l'impression que les résultats de ces 500 simulations sont la version radicale des résultats à 100 simulations.

d. Jeu de données Touch

Après avoir essayé différentes configurations (3, 4 puis 5 classes), j'ai décidé de baser mon analyse sur l'étude de 3 classes pour les raisons usuelles qui sont une inertie intra suffisamment faible ainsi qu'un nombre de classes satisfaisant. Comme pour la construction du graphique des corrélations et du bagplot, on supprime de l'étude les variables concernant les seuils de satisfaction des acteurs à l'aide du paramètre inclus dans la fonction.

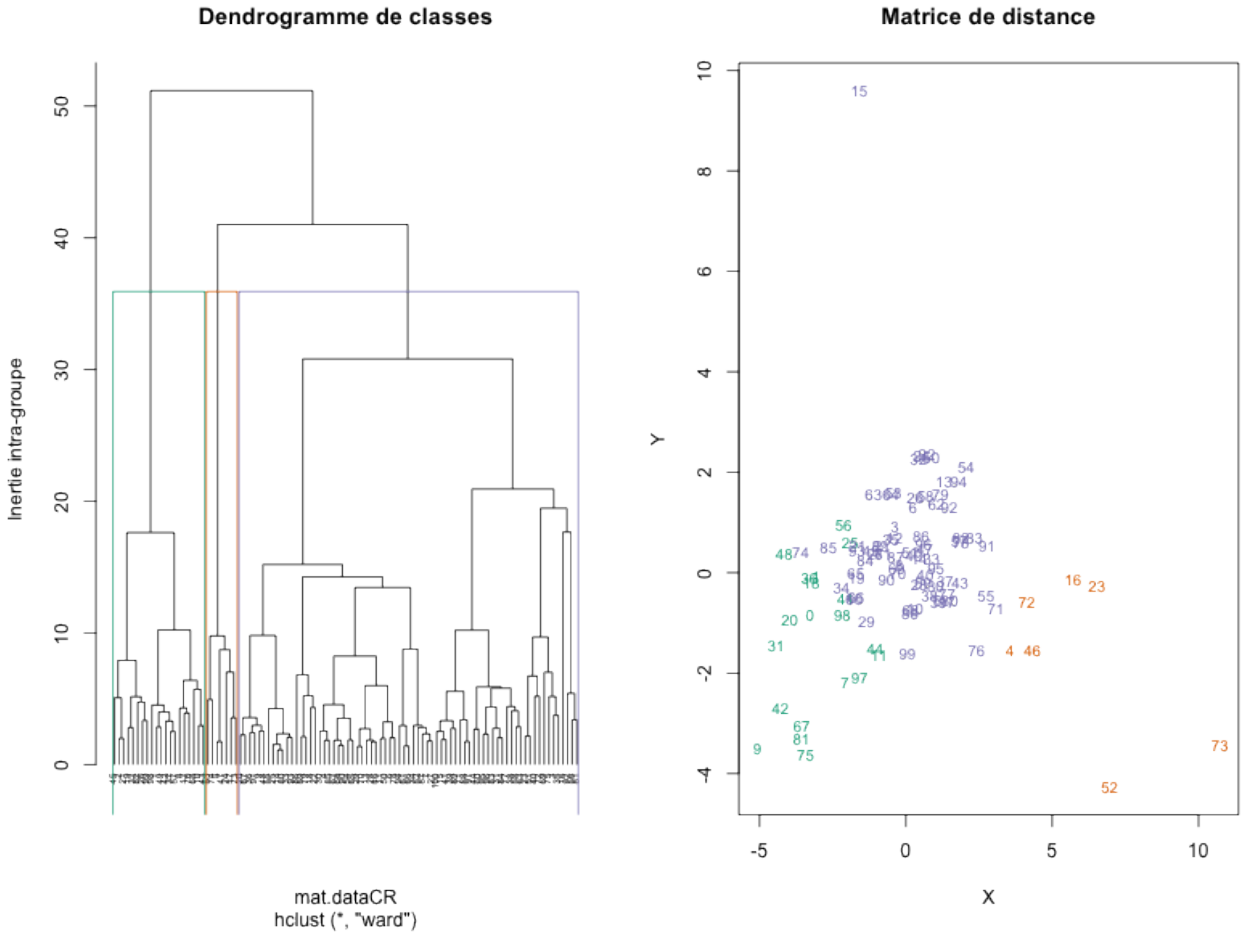


Figure 26 : Classification ascendante hiérarchique à 3 classes du jeu de données Touch

Le passage à 4 classes divise le nuage de points violet (classe 3) en deux groupes qui se chevauchent, prouvant qu'il vaut mieux rester à 3 classes.

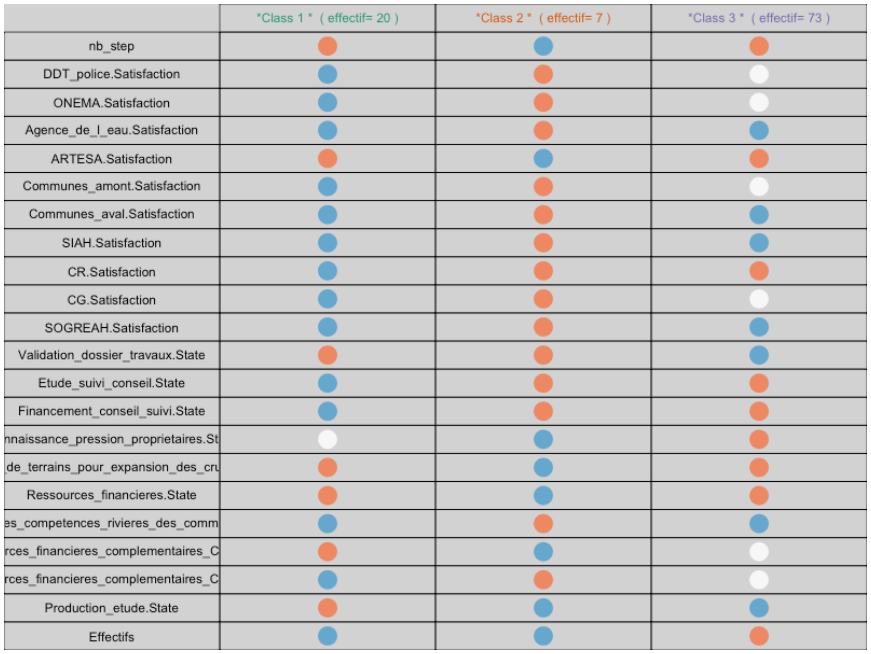


Figure 27 : Tableau d'analyse des moyennes du cas Touch

Classe 1 : La classe 1 est un classe d'effectif moyen puisqu'elle concentre un cinquième des simulations. La convergence de ces simulations est lente puisque `nb_step` est élevé. C'est dans cette classe que l'ARTESA est la plus satisfaite alors que tous les autres acteurs ont une satisfaction faible. Je rappelle que l'ARTESA est l'association de propriétaires. Voyons ce qui peut provoquer cette situation : on voit que les travaux auront lieu même si l'ONEMA semble donner un avis moins favorable au lancement des travaux, probablement jugés moins respectueux de l'environnement (relation `Etude_suivi_conseil`) que dans les autres classes. Pour les mêmes raisons, on trouve un investissement frileux de la part de l'Agence de l'eau (relation `Financement_conseil_suivi`). En revanche, le financement est assuré par les communes en aval (`Ressources_financieres`) et le conseil régional. Enfin, les communes en amont ont une grande capacité à influencer les décisions concernant les aménagements. Ainsi, même si les études menées par la SOGREAH se tournent vers l'hydromorphologie, l'ARTESA est satisfaite.

Classe 2 : Cette classe regroupe à peine 7 simulations, cela revient à dire que cette organisation n'est pas courante. Ces simulations qui ont rapidement convergé sont presque l'inverse de la classe 1. Ici, tous les acteurs sont satisfaits sauf l'ARTESA. En effet, le projet d'aménagement est validé par l'ONEMA (*a fortiori*, il est écologique) et largement soutenu financièrement par l'Agence de l'eau. La pression des propriétaires est ici minimale, tandis que les communes en amont ont moins de pouvoir décisionnel concernant le type d'aménagement. Dans ces simulations, on constate que le SIAH, c'est-à-dire le Syndicat pour l'Aménagement Hydraulique est parvenu à développer ses activités au delà du simple entretien de la rivière, devenant ainsi un acteur incontournable de la gestion de risques. La production d'étude a tendance à avoir une valeur plus faible mais après comparaison auprès des valeurs numériques, on s'aperçoit qu'elle n'est faible que comparativement.

Classe 3 : La classe 3 est la plus importante, avec 73 simulations. Parmi les acteurs qui ont la plus grande satisfaction, on trouve l'ARTESA et le conseil régional. A l'inverse, les moins satisfaits sont l'Agence de l'eau, les communes en aval, le SIAH et la SOGREAH. On voit qu'à nouveau l'ARTESA se différencie du reste des acteurs. On peut également remarquer que les communes en amont et en aval ne sont pas vraiment satisfaites de la situation. Du côté des relations, le projet a obtenu l'aval de l'ONEMA et de l'Agence de l'eau. La pression des propriétaires est assez forte et les communes en amont ont la possibilité d'imposer leurs décisions tandis que celles en aval ont les moyens de financer des aménagements en amont sans intervenir dans leurs propres communes.

En fait, après vérification des valeurs numériques dans le fichier d'analyse des moyennes (fichier csv), on se rend compte qu'en réalité toutes les valeurs sont très proches donc ses résultats sont à modérer. En réalité un point bleu n'est pas nécessairement une valeur faible, c'est juste la plus faible et la plupart du temps, pour le jeu de données Touch, elle n'est pas très éloignée de la plus haute. Par ailleurs, absolument toutes les moyennes sont positives, ce qui signifie que tous les projets proposés tendent à être écologiques et les ressources financières ont tendance à suivre le projet. Les boîtes à moustaches devraient révéler ce phénomène.

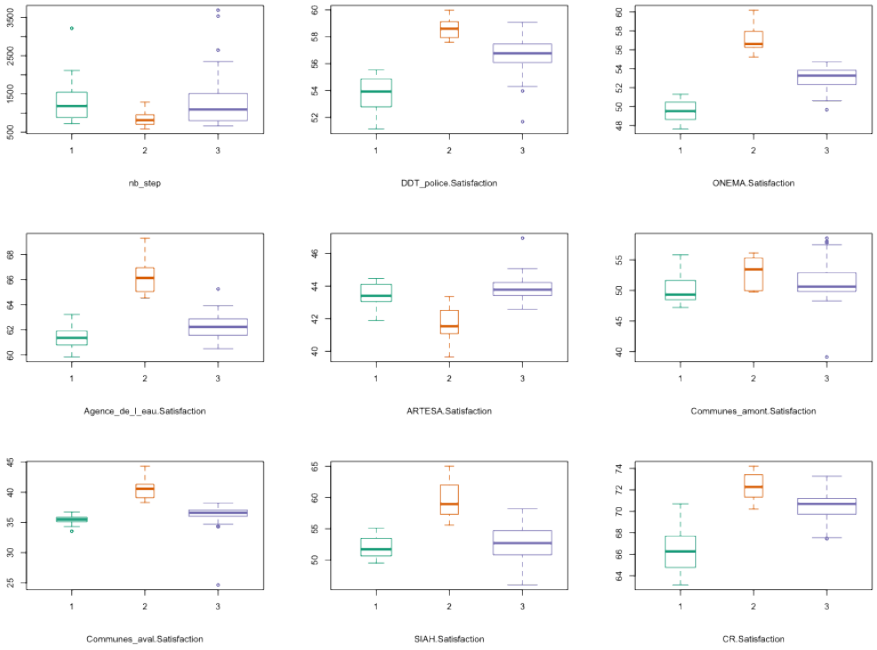


Figure 28 : Boîtes à moustaches du jeu de données Touch (1)

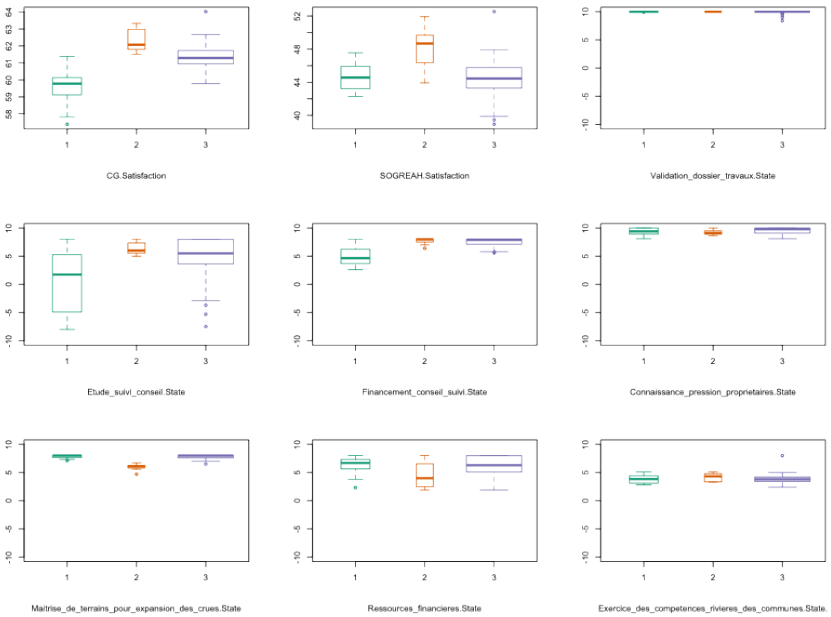


Figure 29 : Boîtes à moustaches du jeu de données Touch (2)

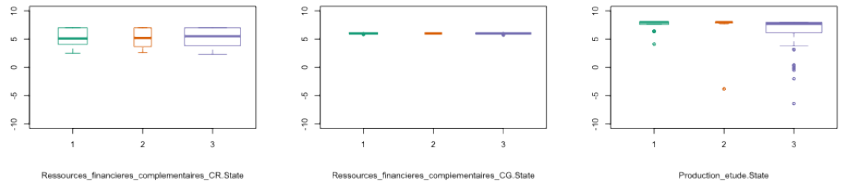


Figure 30 : Boîtes à moustaches du jeu de données Touch (3)

Pour tous les acteurs, on voit bien que les satisfactions ont tendance à être élevées. Au final, on ne retrouve entre les classes qu'une faible variation des valeurs. De manière générale, les données sont bien réparties autour de la médiane avec assez peu d'individus atypiques. On peut voir que les communes n'obtiennent pas des résultats radicalement différents d'une classe à une autre et ont tendance à avoir toujours à peu près le même niveau de satisfaction.
On s'aperçoit que la classe 2 semble se différencier davantage des autres classes car elle fonctionne un peu à « contre-courant ».
Du côté des relations, on voit tout de suite que beaucoup d'entre elles présentent des valeurs médianes égales (`validation_dossier_travaux`, `Connaissance_pression_proprietaires`, `Maitrise_de_terrains_pour_expansion_des_crues`, `Exercice_des_competences_rivieres_des_communes`, `Ressources_financieres_complementaires_CR`, `Ressources_financieres_complementaires _CG`, `production_etude`). Cela signifie que toutes ces variables ne participent pas à la construction des groupes, mais surtout que les acteurs qui les détiennent n'ont au final qu'un pouvoir décisionnel très restreint dans cette organisation.
Par ailleurs, la relation qui semble la plus déterminante dans cette construction est `Etude_suivi_conseil`, détenue par l'ONEMA qui décide de donner un avis favorable ou défavorable aux travaux selon leur caractère écologique. Même si la médiane est quasiment égale dans chacun des groupes, la première classe présente une grande boîte dissymétrique ce qui signifie que les valeurs ne sont pas équiréparties autour de la médiane et tendent particulièrement vers les faibles valeurs. Dans la troisième classe, on retrouve le même schéma sauf que cette fois-ci, c'est la longueur des moustaches qui est important, dénotant de valeurs dispersées avec une forte tendance à tendre vers le négatif.
C'est donc bien la classe 2 qui présente le meilleur projet écologique même si on voit que les classes ont tendance à se ressembler, avec des valeurs proches pour quantité de variables.
L'étude des cartes auto-organisatrices devrait permettre d'affiner les profils et de permettre la détection de comportements plus « tranchés ».

5. Carte auto-organisatrice

Nous allons compléter les résultats de la classification ascendante hiérarchique en réalisant une carte auto-organisatrice des jeux de données. 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 les dimensions, longueur et largeur) définissant le nombre de classes souhaitées. Par défaut, 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.

La fonction permettant de réaliser cette étude se nomme SOM() et ses paramètres permettent entre autres de modifier la taille de la grille et son type (rectangulaire, hexagonal).

a. Jeu de données SEITA

La fonction SOM fournit 5 graphiques différents. Il faudra donc croiser les informations afin d'obtenir une bonne analyse.
Nous allons commencer par étudier la composition des classes créées par la méthode.

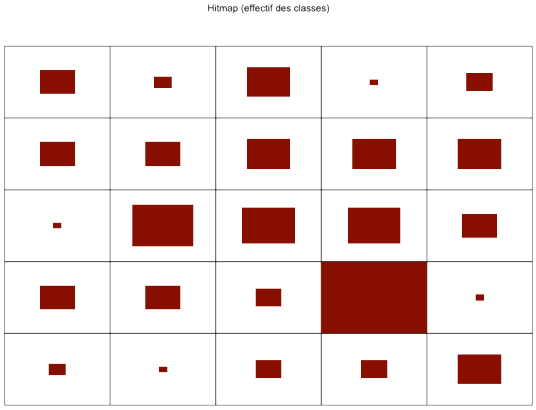


Figure 31 : Carte des effectifs du jeu de données SEITA

Sur cette grille, 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.

Pour pouvoir décrire les différentes classes, on peut commencer par regarder la matrice de graphiques ci-dessous:

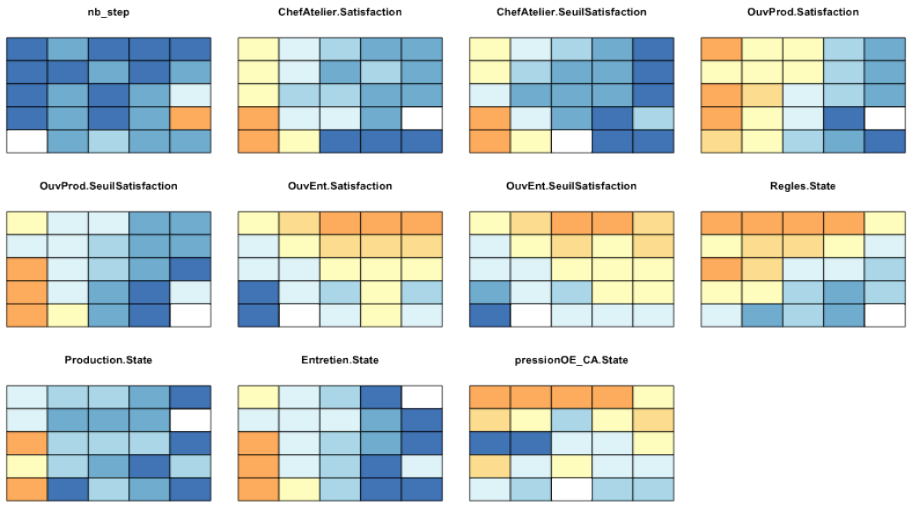


Figure 32 : Cartes colorées selon les valeurs des moyennes de chaque variable du jeu de données SEITA

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 valeur de la moyenne de la variable pour cette classe. Plus la valeur de la variable est forte, plus la couleur utilisée sera foncée.
Par exemple, pour `nb_step`, on a calculé la moyenne du nombre de pas pour chacun des neurones. On obtient donc un ensemble de 25 valeurs. On crée ensuite une séquence entre le minimum et le maximum de ces valeurs que l'on va découper en un nombre défini de classes (ici, 9). Enfin, on prend une à une les moyennes des neurones que l'on va assigner à l'une des 9 classes, sachant que la plus petite classe se voit octroyer la couleur bleu foncé alors que la plus grande est orange foncé. Ainsi, la présence d'une carte monochromatique ou presque est due à la présence d'une valeur atypique.
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.

Ainsi, pour le jeu de données SEITA, on peut constater qu'un ensemble de classes placées 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 qu'il exerce sur les chefs d'atelier 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.

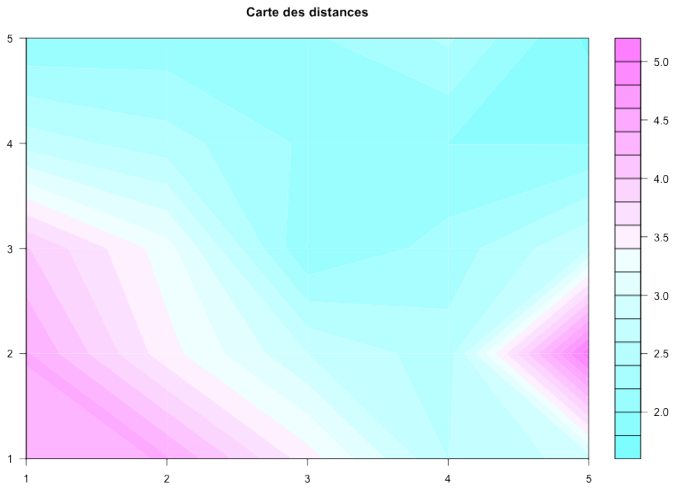


Figure 33 : Carte des distances du jeu de données SEITA

Il s'agit de la carte de base colorée selon la distance entre les classes: plus la teinte tend vers le rose foncé, plus la distance entre les prototypes des classes est élevée. En effet, la carte auto-organisatrice à 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.
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.

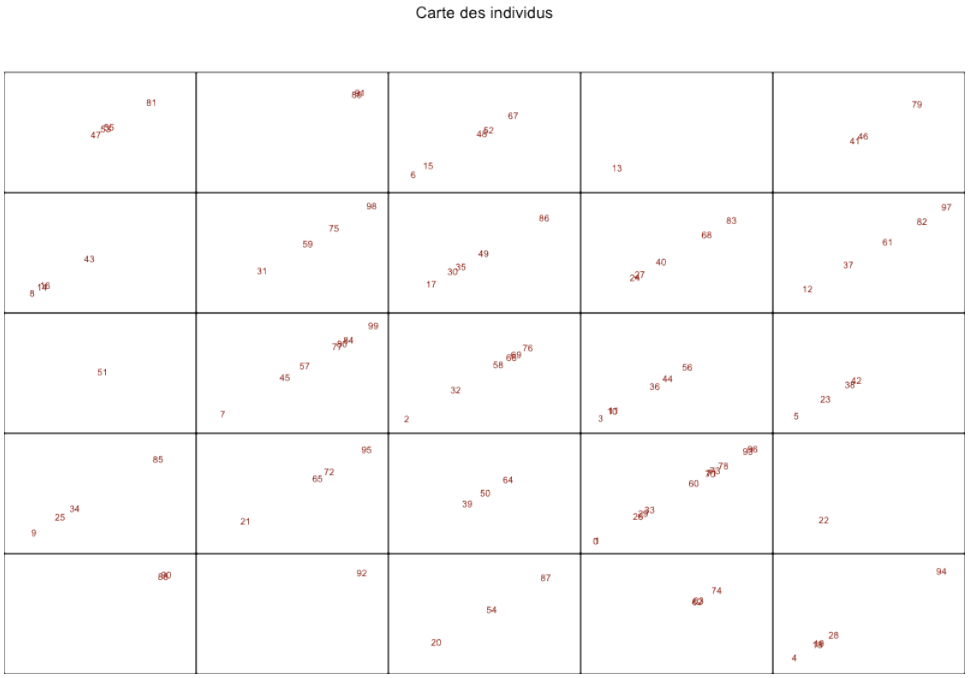


Figure 34 : Carte des individus du jeu de données SEITA

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, 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'elle 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 graphiques que précédemment sauf que les classes ont été colorées en fonction de la valeur de leur écart-type pour cette classe.

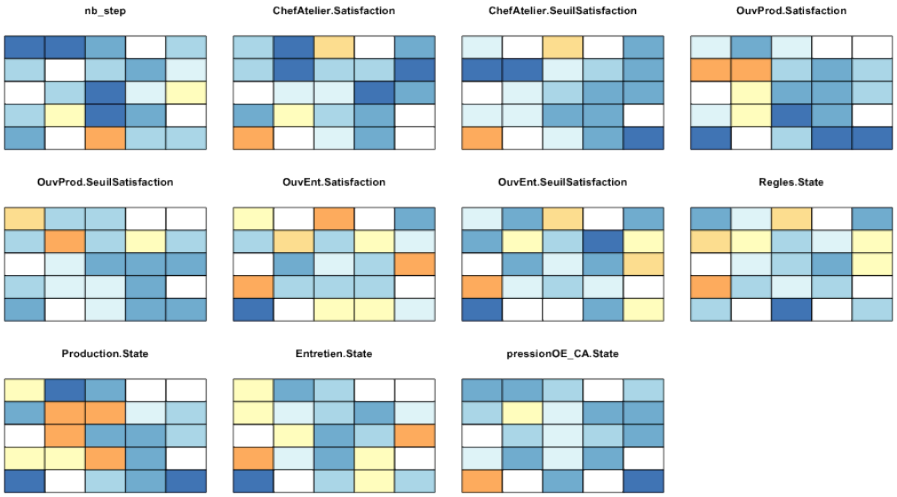


Figure 35 : Cartes colorées selon les valeurs des écart-types de chaque variable du jeu de données SEITA

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 compose 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.

b. Jeu de données Bolet

Durant l'étude de ce jeu de données, nous avons pu constater qu'il y a une séparation très nette du jeu de données en 4 parties distinctes et que l'achat de la machine est un facteur déterminant pour la satisfaction des différents agents.
Pour l'étude de la carte auto-organisatrice, deux grilles ont été testées ; l'une composée de 25 classes et l'autre de 16. La première ayant pour défaut de proposer beaucoup de classes de même effectif avec des caractéristiques quasiment identiques, il est préférable d'opter pour une grille contenant moins de classes. En fait, c'est l'agglomérat de points qui a été découpé en plusieurs classes en raison de légères variations sur l'une ou l'autre des variables.
Ainsi, la grille choisie est donc une grille 4x4.

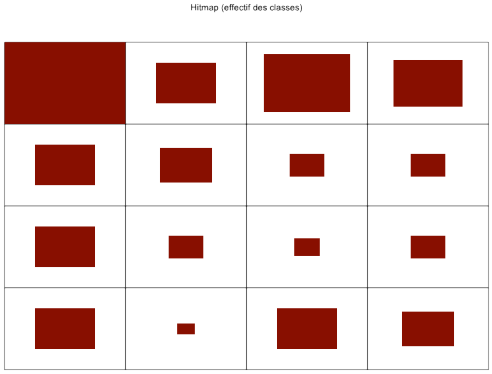


Figure 36 : Carte des effectifs du jeu de données Bolet

Dans cette organisation, on constate qu'il y a une classe prépondérante par son effectif, entourée de classes d'effectifs plus modestes. À l'opposé de la première, on trouve 2 classes d'effectifs moyens. On peut supposer qu'il s'agit d'une part de l'agrégat de points et d'autre part de la queue du nuage de points.

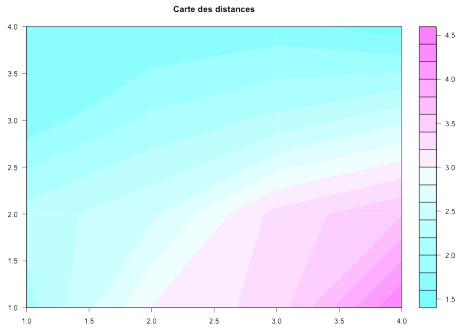


Figure 37 : Carte des distances du jeu de données Bolet

Cette carte nous apprend que les classes situées dans le coin inférieur droit présentent des caractéristiques très différentes des autres, c'est pourquoi elles sont plus distantes du reste des classes.

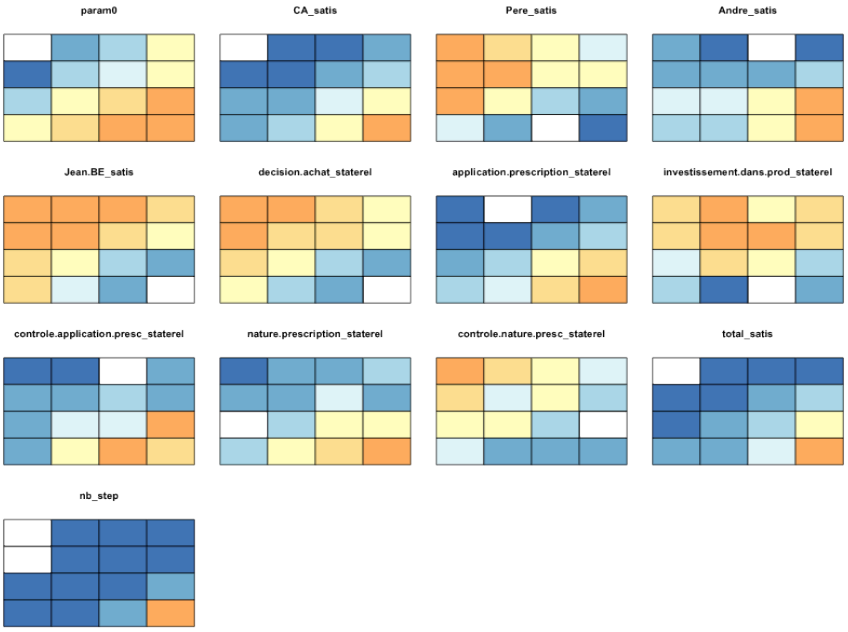


Figure 38 : Cartes colorées selon les valeurs des moyennes de chaque variable du jeu de données Bolet

En étudiant les moyennes de chaque variable, on constate une nette divergence : elles ont tendance à prendre des valeurs fortes soit dans le coin supérieur gauche soit dans le coin inférieur droit. Il n'y a que la variable investissement dans la production qui n'observe pas exactement ce schéma.
On constate que les agents qui trouvent le plus souvent satisfaction sont le père et Jean car de manière générale il y a davantage de classes comprenant de forts effectifs pour lesquelles la décision d'achat de la machine semble plutôt positive. Pour ces mêmes classes, on constate que la nature des prescriptions a tendance à être contrôlée tandis que l'investissement dans la production est plutôt bon. Par ailleurs, cette carte met en évidence le fait que la satisfaction de Jean est très liée à l'achat de la machine, du moins plus que pour son père. En effet, la satisfaction de Jean suit exactement le même motif que l'achat de la machine alors que la satisfaction du père semble être davantage influencée par l'achat de la machine et l'investissement dans la production à la fois.
Si l'on s'intéresse aux classes mises en évidence par la carte des distances et particulièrement celle occupant le coin inférieur droit, on voit que c'est une classe pour laquelle la ténacité du chef d'atelier est forte. Cela entraîne une valeur élevée pour la variable `nb_step`. En effet, on rappelle que la ténacité du chef d'atelier est sa propension à explorer ou à utiliser ses connaissances pour prendre des décisions. Dans le cas d'un individu tenace, plus il veut explorer, plus il y aura besoin d'itérations pour parvenir à une stabilisation de l'organisation.
Ces classes sont différentes des autres car peu de simulations mènent à une décision d'achat aussi faible. Pour ces classes, on constate également que les prescriptions du BE, qui sont les plus rationnelles, sont bien appliquées au niveau de la production peut-être car le contrôle effectué à ce niveau est pointilleux.

c. Jeu de données SEITA à 500 simulations

Pour étudier ce jeu de données, j'ai testé plusieurs configurations de grilles. Mon choix s'est arrêté sur une grille 7x7 (ce qui ferait environ 10 individus par case si les données étaient distribuées de manière égale), de type hexagonal. J'ai choisi de changer la forme pour donner un aperçu des différentes sorties que l'on peut obtenir, mais également en raison du grand nombre d'individus.

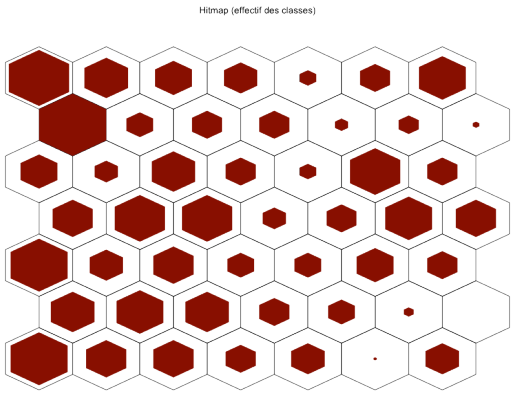


Figure 39 : Carte des effectifs du jeu de données SEITA à 500 simulations

On peut voir que les données semblent légèrement massées vers le côté gauche de la carte.

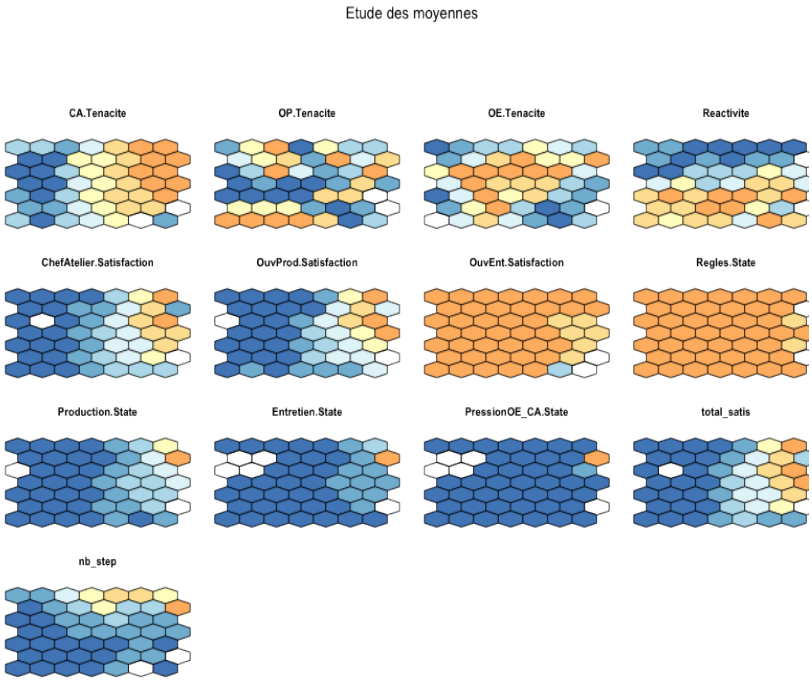


Figure 40 : Cartes colorées selon les valeurs des moyennes de chaque variable du jeu de données SEITA-500

Au premier abord, on voit tout de suite que, pour ces simulations, l'ouvrier d'entretien finit quasiment toujours par obtenir une grande satisfaction. A l'inverse, pour les autres acteurs, cela devient très marginal. Dans la première version de SEITA (100 simulations), même si l'ouvrier d'entretien obtenait plus souvent satisfaction que ses collègues, ces deux derniers arrivaient à mieux s'imposer. On s'aperçoit que l'ouvrier de production est devenu complètement dépendant du chef d'atelier : dans les précédentes simulations, il y avait des situations pour lesquelles, bien que le chef

d'atelier n'était pas vraiment satisfait, l'ouvrier de production arrivait à l'être. Or ici, l'ouvrier de production dépend de son chef et plus particulièrement de la ténacité de ce dernier. On avait bien vu lors de l'étude des corrélations que la ténacité du chef d'atelier était corrélée positivement avec sa satisfaction et celle de l'ouvrier de production. Cette tendance est complètement affirmée sur les cartes auto-organisatrices.
Ce qui est assez étonnant, c'est que seule la ténacité du chef d'atelier ait des répercussions sur l'état des simulations. Il est possible que cela vienne du fait que l'ouvrier de production ne détienne pas de « relations clefs » et que, même en étant tenace, il ne puisse pas vraiment influencer les autres acteurs. Quant à l'ouvrier d'entretien, à l'inverse il détient la relation la plus « puissante » et qu'il soit tenace ou non ne déteint pas sur les résultats, puisque c'est le relation « entretien » qui lui permet d'obtenir satisfaction.
On peut voir aussi que le respect des règles est maximum partout. Or, en regardant le graphique des corrélations, on s'aperçoit que le respect des règles n'est pas, contrairement à ce que le graphique laisse entendre, corrélé à la satisfaction de l'ouvrier d'entretien, que ce soit négativement ou positivement. A l'inverse, le respect des règles est fortement lié aux 3 autres relations : plus le respect des règles augmente, plus la production, l'entretien et la pression diminuent.
Par ailleurs, on voit qu'il y a quelques classes pour lesquelles la satisfaction de tous les agents paraît bonne. Seulement, il faut garder à l'esprit que la couleur orange signifie « fort par rapport aux autres simulations », ce qui ne veut pas forcément dire une « bonne satisfaction ».
Il vaut mieux dire qu'il y a quelques classes pour lesquelles les satisfactions des ouvriers de production et du chef d'atelier sont meilleures, même si elles restent faibles.
Un autre résultat est assez étonnant : il s'agit de la réactivité. On voit bien que cette variable est organisée sur la carte mais cela ne semble pas avoir eu de conséquence sur l'organisation des autres variables, comme si la réactivité des agents n'influençait pas le modèle.

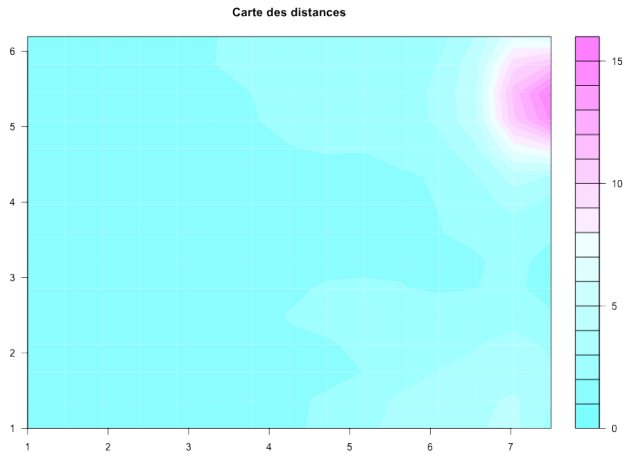


Figure 41 : Carte des distances du jeu de données SEITA à 500 simulations

La carte des distances met en valeur ces quelques classes, qui voient la satisfaction du chef d'atelier et de l'ouvrier de production augmenter mais également la classe pour laquelle tous les agents sont tenaces. Ces simulations ont été très longues à converger et ont la particularité de présenter des valeurs plus élevées que les autres classes pour toutes les relations sauf pour le respect des règles, qui est plus bas qu'habituellement.
Pour conclure, je pense que ce jeu de données à 500 simulations est moins intéressant que le premier étudié dans le sens où il exacerbe des comportements connus et ne permet pas de détecter des comportements plus marginaux. A mon sens, pour le cas SEITA, il est plus pertinent d'utiliser un jeu de données contenant moins de simulations.

d. Jeu de données Touch

Pour l'étude de ce jeu de données, j'ai conservé la configuration de base concernant le nombre de cellules mais j'ai modifié leur type. En effet, les données du cas Touch sont plus nombreuses et complexes que pour les autres cas, c'est pourquoi on s'autorise un voisinage plus important par cellule.

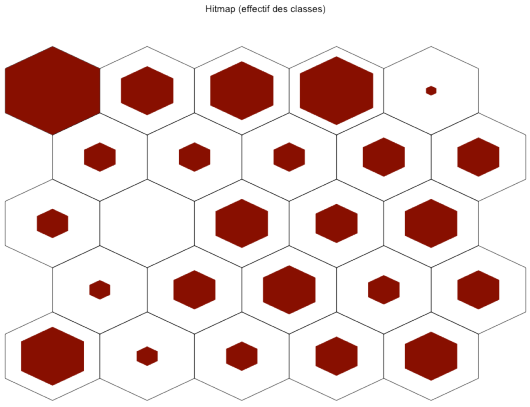


Figure 42: Carte des effectifs du jeu de données Touch

On remarque une certaine disparité entre les effectifs des cellules puisque certains ne contiennent aucun ou très peu d'individus alors que d'autres sont plus importantes.

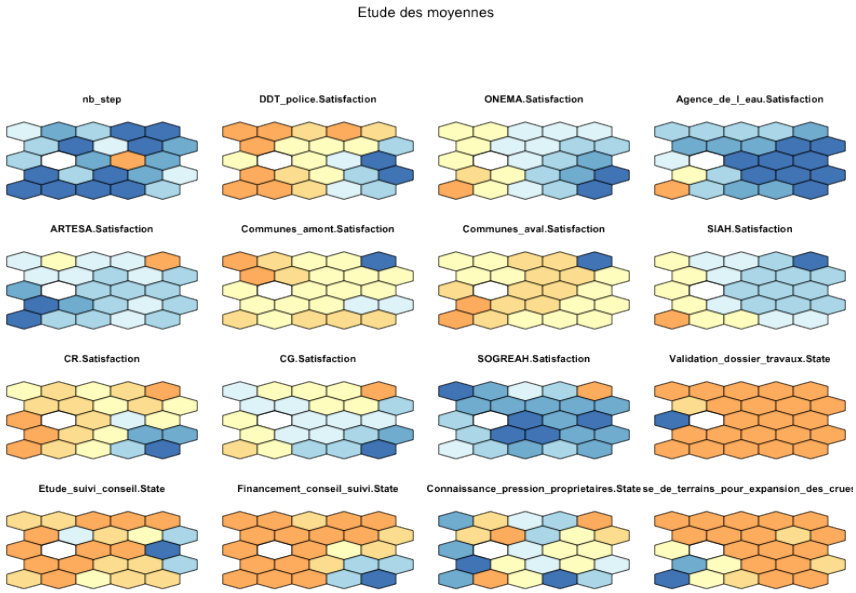


Figure 43 : Cartes colorée selon les valeurs des moyennes de chaque variable du jeu de données Touch (1)

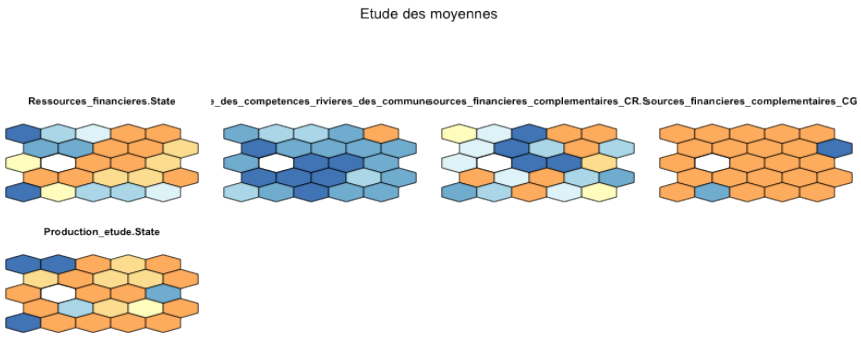


Figure 44 : Cartes colorées selon les valeurs des moyennes de chaque variable du jeu de données Touch (2)

Ce que l'on voit immédiatement sur ces cartes, c'est qu'elles semblent avoir des couleurs plus atténuées que les autres cartes auto-organisatrices, ce qui signifie qu'elles ont tendance à avoir des valeurs proches les unes des autres et à tendance centrale.
Par ailleurs, sur beaucoup de cartes, on voit que les variables représentées ne participent pas à l'organisation de la carte (on ne voit pas de dégradés de couleurs, les couleurs sont éparses).
Au niveau des acteurs, on constate que l'agence de l'eau est globalement peu satisfaite, notamment dans les cellules situées à droite de la carte auto-organisatrice. Il n'y a qu'une cellule pour laquelle l'Agence de l'eau obtient une bonne satisfaction comparativement aux autres moyennes. Il s'agit de la cellule qui regroupe la plupart des individus qui formaient la classe 2 (effectif très faible, satisfaction de tous les agents sauf l'ARTESA).
Les communes en aval et en amont ont l'air d'avoir des satisfactions toujours à peu près égales et de valeurs centrales.
Une cellule particulière est placée en haut à droite des cartes : il s'agit d'un neurone ne contenant que la simulation 15, qui est un individu atypique, remarqué lors de l'étude du MDS. Cette simulation atypique présente des satisfactions particulièrement basses pour les communes, le SIAH et dans une moindre mesure, l'agence de l'eau. On dirait que c'est une simulation qui a conduit à prendre des décisions néfastes pour les habitants. L'ARTESA et la SOGREAH ont trouvé leur compte dans cette simulation car leurs satisfactions sont très bonnes.
Encore une fois, on constate que malheureusement les différentes relations ne permettent pas bien d'expliquer les satisfactions des acteurs. Aucune d'entre elle ne présente un dégradé de couleurs et les cellules à valeurs fortes côtoient celles à valeurs faibles.
On voit quand même que la validation des travaux est effective dans la majorité des cas, que la SIAH finance les projets dans de nombreuses simulations, que les communes en amont ont tendance à avoir un pouvoir décisionnel fort et que le CG participe très souvent au financement des travaux.
Ces simulations présentent des caractères moins « tranchés » que les cas étudiés précédemment. Je pense que ce phénomène vient du fait que l'on est passés d'un cas d'école à un cas pratique. En effet, le modèle a peut-être été moins étudié et peut présenter quelques failles. On a d'ailleurs remarqué que beaucoup de relations ne participaient pas à la construction de l'organisation, ce qui démontrait « l'inutilité » de certains acteurs.
De plus, c'est tout de même un modèle très compliqué, avec de nombreuses interactions entre les agents, ce qui permet une grande variété de cas, difficiles à contenir même dans un réseau de neurones.

V. Développement informatique

Le développement informatique a fait partie intégrante de mon travail et a absorbé une bonne partie de mon temps. Comme cela a été expliqué précédemment, chacune des méthodes statistiques a été codée à l'aide du logiciel R. À terme, ces scripts seront insérés dans une interface préexistante, implémentée en Java. A ce jour, cette interface permet de faire des statistiques univariées et des ACP sur les données issues de SocLab.
Mes fonctions doivent en outre être simples à utiliser pour une personne non initiée à l'informatique (nom de paramètres simples et compréhensibles, explication précises, etc.) et facile à comprendre pour un programmeur désireux de faire des modifications *a posteriori*.

1. Architecture

Tout d'abord, pour mieux comprendre l'architecture de mes fonctions, en voici l'illustration :

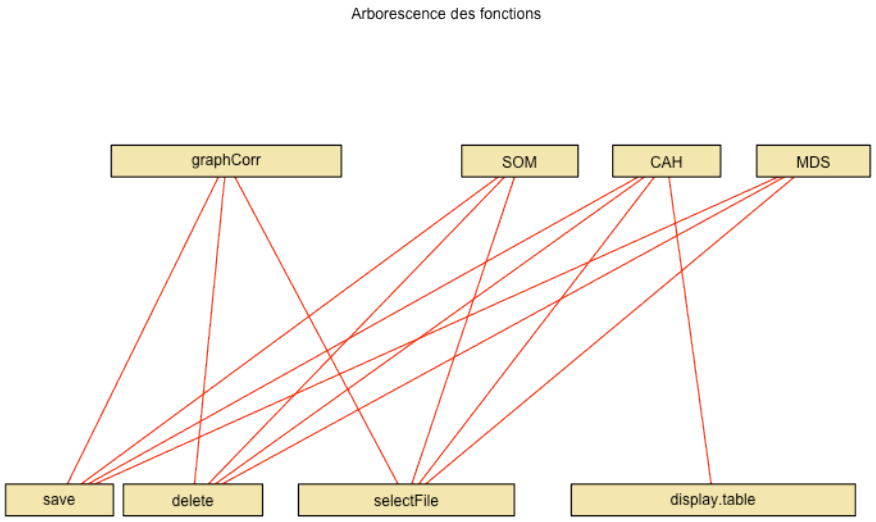


Figure 45: Arborescence des fonctions

Cette architecture est obtenue grâce à la fonction foodweb implémentée sur R, qui traduite en français signifierait « chaîne alimentaire ». En effet, la partie supérieure montre les fonctions principales permettant d'appliquer des méthodes statistiques. Chacune est alimentée par une ou plusieurs sous-fonctions. La fonction foodweb permet également de comprendre l'architecture d'un package inconnu par exemple et peut se révéler fort utile pour un programmeur.

- delete permet de supprimer des données inutilisées lors de l'étude des jeux de données,
- selectfile déclenche l'ouverture du gestionnaire de fichiers pour permettre à l'utilisateur de choisir facilement le jeu de données sur lequel il souhaite travailler
- display.table est utilisée pour afficher le tableau d'analyse des moyennes
- save permet une sauvegarde simple de graphique (opposé à une sauvegarde multiple dans les cas où la sortie est une matrice de graphiques qui, pour des questions de lisibilité doit être affichée sur plusieurs fenêtres)

2. Codage

Le codage des différentes fonctions a occupé la moitié du temps de stage tandis que l'autre moitié a été dédiée aux analyses des sorties. Chacune des fonctions a été testée sur plusieurs jeu de données et j'ai essayé d'anticiper autant que possible les différents conflits possibles afin de pérenniser mes fonctions.
Aussi, j'ai essayé de rendre chacune des fonctions « user friendly », c'est-à-dire agréables à manipuler pour les utilisateurs tant au niveau des fonctionnalités que du choix des couleurs et des paramètres graphiques. Par exemple, le choix du fichier de données à étudier se fait via la gestionnaire de fichier et non via une commande R. Aussi, la fonction CAH permet à l'utilisateur de faire une sélection manuelle du nombre de classes qu'il désire : un dendrogramme vierge s'affiche et grâce au clic gauche de la souris, l'utilisateur affiche les différentes classe. Un clic droit permet ensuite d'obtenir les graphiques utiles à la description des classes. Cette sélection manuelle permet une interaction certes de faible niveau, mais effective entre l'utilisateur et l'outil statistique.
Par ailleurs, j'ai apporté un soin tout particulier à la présentation de mes fonctions (indentation, cartouche) et à la clarté de mes commentaires. Aussi, en cas d'arrêt d'exécution, la fonction retourne toujours un message d'erreur clair à l'utilisateur, lui rappelant parfois les choix possibles pour un paramètre donné.
Enfin, j'aimerais en particulier expliquer dans cette partie le codage de certaines fonctionnalités, régulièrement citées dans le mémoire ou dans le manuel des fonctions. Je conseille vivement la lecture de ce dernier afin de bien comprendre la structure des fonctions et de voir les différents paramètres proposés.

a. Suppression de variables

C'est un paramètre que l'on retrouve dans toutes les fonctions. En effet, il sert principalement à supprimer les seuils de satisfaction des agents car ils donnent quasiment les mêmes informations que les satisfactions.
Dans la pratique, le paramètre (`delVar`) est égal à zéro par défaut. Si l'utilisateur précise qu'il veut supprimer des variables, l'algorithme fonctionne ainsi :

```
Si le paramètre delVar≠0 {
	Récupération du motif auquel on ajoute une virgule au début et à la fin
	On cherche le nombre de virgules et leur position
	Le nombre de variables est égal au nombre de virgules -1
	Pour chacune des variables {
		On récupère le(s) motif(s) entre les virgules et on l'affecte à une variable
	}
	Pour chacun des motifs {
		Si on trouve le motif parmi les noms de variables initiaux (sans prendre en
		compte la casse) {
			On récupère le numéro de colonne de la variable
			Le nouveau jeu de donnée = l'ancien jeu de données – la colonne
			concernée
		}
		Sinon {
			Message d'erreur informant l'utilisateur que la variable spécifiée
			n'existe pas
		}
	}
}
```

b. Coloration selon la valeur de la variable

La coloration selon la valeur de la variable a été utilisée a plusieurs reprises : elle permet de bien montrer le niveau d'une variable dans une classe par exemple ou comme dans le MDS, des regroupements d'individus en fonction de la valeur d'une variable.
Pour se faire, l'idée de base est d'affecter une classe à chacun des individus :

```
On stocke dans une variable plusieurs couleurs allant de la plus claire à la plus foncée
(généralement 8 ou 9 couleurs)
Pour chaque variable du jeu de données {
	Récupération du maximum et du minimum
	Création d'une séquence entre ces deux bornes
	On organise cette séquence en classes (généralement 8 ou 9 classes de mêmes
amplitudes)
	A chaque donnée, on associe la classe correspondante
	Les valeurs des couleurs sont affectées aux classes dans l'ordre croissant
}
```

C'est une méthode à laquelle je n'avais jamais pensé auparavant mais qui permet de donner à un nuage de points d'autres dimensions de lecture tout à fait intéressantes.

c. Tableau d'analyse des moyennes (CAH)

La création du tableau d'analyse des moyennes a été problématique notamment parce qu'il n'a pas été simple d'affecter le bon numéro de classe à chaque données. On effet, R fournit après la classification un vecteur qui coupe bien les données dans le nombre de classes souhaité sauf qu'il assigne un numéro de classe dans l'ordre croissant.
Par exemple, admettons que l'individu 1 fait partie de la classe 2, le 2 de la classe 3 et le 3 de la classe 1, R retourne un vecteur de la forme : `[1] 1 2 3` et pas `[1] 2 3 1`.
C'est une erreur que je n'ai pas décelée tout de suite mais lors de l'interprétation du SOM en comparant les profils des individus.
La difficulté est de construire le « vrai » vecteur de classes, cependant je n'ai pas pu détailler ici l'algorithme permettant de l'obtenir car il utilise les résultats de classification fournis par R et il aurait fallut expliquer ce que contenait chaque vecteur ainsi que sa structure. L'algorithme est toutefois disponible en annexes.

```
Création du vecteur contenant les numéros de classe des individus
Fusion de ce vecteur avec le jeu de données (on ajoute donc une colonne)
On agrège les données selon le numéro de classe et on calcule la moyenne
Utilisation de la fonction displayTable pour mettre en forme le tableau obtenu
```

d. Cartes des individus et d'étude des moyennes et écarts types (SOM)

Le package Yasomi (dont la référence est disponible dans la bibliographie) que j'utilise pour la fonction SOM est actuellement en cours de développement, c'est pourquoi, malgré toutes ses fonctionnalités il peut persister quelques erreurs.
A la base, la fonction SOM faisait apparaître un autre graphique proposé par le package : il s'agissait d'une matrice de graphiques colorant les neurones en fonction de la valeur du prototype de la classe. J'ai alors découvert, en même temps que mon erreur concernant les classes de CAH, que la coloration était inversée : les couleurs pâles étaient affectée aux fortes valeurs et inversement. Malgré l'intérêt d'un tel graphique, je n'ai donc pas pu l'utiliser et il a été remplacé par une matrice de graphiques avec les classes colorées en fonction de la valeur moyenne de la variable au lieu du prototype.
J'ai donc crée « à la main » trois cartes sur cinq présentes proposées en sortie de la fonction SOM.
Voici la création de la carte des individus (chaque classe contient les numéros des individus qui la composent) :

On part d'une fenêtre graphique vierge donc, avant de tracer quoi que ce soit, il convient de définir des zones de traçage afin de pouvoir placer un titre par exemple. Pour cela, on utilise une simple matrice que l'on affectera au paramètre « `layout` ». Admettons que je souhaite avoir un titre puis une matrice de graphiques 4x4. Je crée donc cette matrice : `matrix(c(rep(1,4),2 :(4^2+1)),ncol=4,byrow=T)` qui, si on l'affiche sur le terminal R donne :

```
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2    3    4    5
[3,]    6    7    8    9
[4,]   10   11   12   13
[5,]   14   15   16   17
```

En fait, le premier graphique que l'on trace prendra place sur les case notées « 1 », le second sur la case « 2 » et ainsi de suite. Dans ce cas, on demandera en premier d'imprimer le titre puis chacune des cases de la matrice de graphique.
Je ne connaissais pas cette technique et elle s'avère très pratique pour les cas où comme ici, on souhaite reconstruire un graphique à la main ou encore si l'on veut afficher des graphiques de nature différente sur une même fenêtre.
Pour construire la grille à proprement parler, on procède ainsi :

```
Définition des zones de traçage de la fenêtre graphique
Affichage du titre
Pour chaque classe du SOM (chaque case de la grille) {
	Si la case n'est pas vide {
		Récupération du numéro des individus
		Construction d'un graphique allant de 1 au nombre total d'individus en ordonnées et en abscisse (dans tracer les axes)
		Le numéro de l'individu devient ses coordonnées et on y écrit son numéro
		Encadrement du graphique
	}
	Sinon {
		Création d'un graphique vide
		Encadrement du graphique
	}
}
```

Le fait que le numéro de l'individu devienne ses coordonnées permet de faciliter la lecture du graphique : en effet, on voyant sa position par rapport aux autres, on peut avoir une idée de son numéro même si il est en partie dissimulé par d'autres individus.
Ce graphique présente un seul défaut : il ne s'adapte pas, contrairement à tous les autres, à la forme de la grille. Si l'utilisateur opte pour une grille hexagonale, la carte des individus sera tout de même rectangulaire.
Pour l'étude des moyennes et des écarts types, il a été possible d'utiliser la commande qui trace automatiquement la grille : contrairement à la carte des individus, il n'y a rien à tracer dans chacune des cases, juste une couleur à assigner, c'est pourquoi il a été possible d'automatiser la création de la grille. Techniquement, pour faire ces graphiques, il suffit de récupérer les individus de chaque classe, de calculer leurs moyennes pour chacune des variables et de colorer les cases selon cette valeur (technique des classes expliquées précédemment).

3. Visualisation

Titulaire d'un premier DUT en Information-Communication, j'ai été sensibilisée aux problèmes de communication dus à une information visuelle peu claire, trop chargée, etc. J'ai appris qu'un mauvais choix de couleurs ou qu'une mise en page hasardeuse pouvaient être porteurs de messages sous-jacents et influencer de manière illégitime l'interprétation que peut faire l'usager d'une image.
J'estime que, dans le domaine de la statistique, cette problématique est particulièrement importante bien que largement sous-traitée dans les cursus universitaires. C'est pourquoi, j'ai décidé de rendre son importance aux efforts que j'ai fournis tout au long de la création des graphiques afin de transmettre une information claire et aérée à l'utilisateur.

a. Couleurs

Chaque fois qu'un graphique nécessite l'utilisation de couleurs, je fais appel à la libraire RcolorBrewer. C'est un package dédié à la coloration de graphiques, inspiré par les travaux de recherches de Cynthia Brewer, spécialisée entre autre en design de cartes et en théorie des couleurs en cartographie.
Dans son article « Practical Rules for Using Color in Charts » (Few, 2008), Stephen Few, spécialiste en communication et visualisation des données nous conseille l'utilisation du package RcolorBrewer et nous rappelle quelques règles de bon sens pour réaliser des graphiques telles que :

Rule #2 If you want objects in a table or graph to be easily seen, use a background color that contrasts sufficiently with the object.
Rule #3 Use color only when needed to serve a particular communication goal.
Rule #9 Avoid using visual effects in graphs.
(Source disponible dans la bibliographie)

On trouve également une règle plus atypique qui nous déconseille d'utiliser le rouge et le vert dans un même graphique car approximativement 10% des hommes américains (8% pour les européens) souffre de daltonisme, ce qui les empêchent de distinguer ces deux couleurs.
Stephen Few qui a crée un cabinet conseil spécialisé en visualisation de données à entre autres clients Microsoft, Apple, SAS Institute, la NASA, l'US Navy ou encore l'UNSECO et Kraft Foods.
Le package RcolorBrewer est composé de trois types de palettes différentes :

- Palettes qualitatives : elles sont utilisées pour représenter des données qualitatives nominales ou catégorielles. Ce sont ces palettes que j'utilise pour représenter les différentes classes par exemple.
  Les palettes contiennent un maximum de 12 couleurs car l'œil humain n'est plus capable de discerner correctement les différentes couleurs au delà.

Figure 46 : Palette qualitative de RColorBrewer

- Palettes séquentielles : ce sont des palettes qui contiennent des couleurs ordonnées de la plus pâle à la plus foncée. On les utilise pour représenter une variable qui a des valeurs croissantes.
  Dans mes fonctions, j'utilise cette palette pour colorer le MDS selon l'intensité des variables.

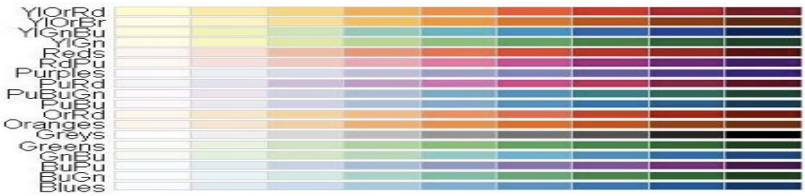


Figure 47 : Palette séquentielle de RColorBrewer

- Palettes divergentes : ce sont 2 palettes de couleurs séquentielles qui sont réunies mais qui baissent d'intensité au milieu de la palette. On utilise ce genre de palette pour représenter une information quantitative qui prend d'autant plus de sens que la couleur est éloignée des points centraux. J'ai utilisé ces palettes pour colorer le tableau d'analyse des moyennes ou les cartes auto-organisatrices représentées selon la valeur moyenne de chacune des variables. Elles permettent de voir immédiatement des regroupements faibles ou forts, alors que les valeurs centrales et donc peu importantes dans ces études sont beaucoup moins visibles.

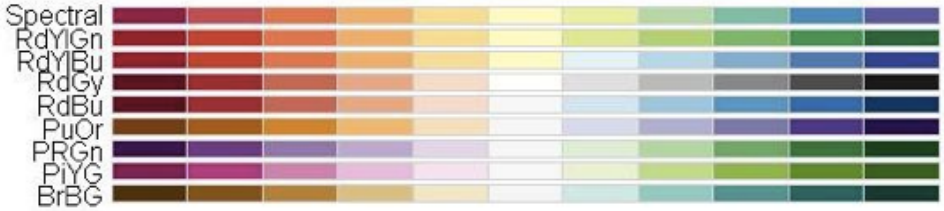


Figure 48 : Palette divergente de RColorBrewer

b. Affichage des graphiques

La plupart des graphiques que j'ai crées se modifient selon le nombre de variables, la quantité d'informations à donner. En effet, les fenêtres graphiques susceptibles d'afficher plusieurs graphiques sont plafonnées à un affichage 4x4 ou 5x5 selon les cas. Au delà de ces valeurs, l'affichage continue sur une second fenêtre. De même, le graphique des corrélations par exemple, voit sa taille de police évoluer en fonction du nombre de variables. Si il y a peu de variables, on en profite pour augmenter la taille et rendre le graphique plus lisible. J'ai essayé de rendre mes fonctions adaptables selon les données.

Poursuivant le même but de lisibilité, j'ai rendu certains graphiques facultatifs (boîtes à moustaches, bagplot) ceci dans le but de ne pas noyer l'utilisateur sous les informations. En effet, il peut ainsi lancer la fonction « de base », en étudier les premières sorties puis, lorsque les informations sont claires et bien comprises, il peut y ajouter l'étude d'un nouveau graphique.

Enfin, la fonction SOM permet de modifier le type de la grille (hexagonal ou rectangulaire). En effet, nous avons constaté que la forme de la grille avait son importance pour l'utilisateur : certaines personnes visualisent mieux la carte auto-organisatrice quand elle est composée d'hexagones.

VI. Méthodes et outils utilisés

1. Méthodes

a. Méthode MDS

Cette méthode fait partie des méthodes d'analyse factorielle des données. Le Multi-Dimensional Scaling retourne, à partir d'un ensemble de dissimilarités, un ensemble de points dans le plan 2D construits de telle manière que la distance entre les points dans le plan est approximativement égale à la valeur de leur dissimilarité (ou distance) dans l'espace initial.
Elle permet de rendre compte des similarités ou dissimilarités des données. En effet, pour chaque couple d'individus, on va calculer un indice (de dissimilarité, de similarité ou de distance) qui sera ensuite stocké dans une matrice. L'objectif du MDS est de construire, à partir de cette matrice une représentation euclidienne des individus, idéalement dans un espace à 2 dimensions, afin d'en obtenir une représentation graphique.
On peut choisir par exemple de construire la matrice de distances selon la distance euclidienne de cette manière :
On considère les observations pour un individu (x_i, y_i,...) pour le $i^{ème}$ individu. Si l'on considère seulement deux variables, chaque individu est un point M_i de l'espace de coordonnées (x_i, y_i). La distance entre les individus M_i et M_j est telle que :

$$d^2(M_i, M_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Plus l'indice est petit, plus la distance entre individus est minimale et, à l'inverse, plus la distance est élevée, plus les individus sont différents. Ces résultats sont ensuite insérés dans une matrice dite « de distance » à *n* (avec *n*, l'effectif total) lignes et *n* colonnes, symétrique, dont les coefficients sont positifs.
On cherche ensuite une configuration de n points dans un espace de dimension q telle que le point $\vec{x_i}$ représente de façon unique l'objet i et la distance euclidienne entre les points $\vec{x_i}$ et $\vec{x_j}$.
On obtient la solution en réalisant ces différentes étapes:

- On construit la matrice de terme général $-\frac{1}{2}d^2(M_i, M_j)$
- Construction de la matrice de produits scalaires grâce à un double centrage. On obtient la matrice B=HAH
- On diagonalise $B=U\Delta U^T$
- Les coordonnées principales, c'est-à-dire les coordonnées d'une configuration sont les lignes de la matrice $X=U\Delta^{1/2}$

b. Classification hiérarchique

L'objectif global de la classification est de regrouper les individus ayant un comportement similaire dans une même classe. Dans chacun des groupes, les individus doivent être aussi proches que possible alors que, pour deux groupes distincts, les individus devront être aussi différents que possible. On parle également de « similarités » et de « dissimilarités » entre les individus.
On définit la similarité comme une quantité qui est d'autant plus élevée que les individus sont semblables, alors que pour la dissimilarité, c'est l'inverse (plus les individus sont différents, plus la quantité est élevée). La distance euclidienne par exemple est une dissimilarité.
En classification, un indice est particulièrement important : il s'agit de l'inertie.
On rappelle que l'inertie intra peut se décomposer de cette manière :

Inertie totale= inertie intra-classes + inertie inter-classes

L'inertie intra-classes est la moyenne des variances à l'intérieur de chacune des classes alors que l'inertie inter-classes et la variance des moyennes de chacune des classes.

Pour la problématique de la classification, on cherche à avoir une variance intra-classes faible (*i.e.*: les individus d'un même groupe doivent être semblables) et une variance inter-classes forte (*i.e.*: les individus appartenant à des groupes différents doivent présenter une forte dissimilarité).
La décomposition de l'inertie met en lumière le fait qu'augmenter l'inertie intra-classes revient automatiquement à diminuer l'inertie inter-classes et inversement.

Il existe plusieurs méthodes de classification : ici, nous verrons la classification hiérarchique, bien qu'il existe des méthodes non hiérarchique.
La classification hiérarchique consiste à créer une hiérarchie de partitions allant de N à 1 partie selon des indices de similarités ou de dissimilarités.

Exemple : Prenons N=5 individus, formant un ensemble E d'individus. E={e_1, e_2, e_3, e_4, e_5}.
Soit P, une partition de E.
Une hiérarchie de partitions possible serait :

P1={{e_1, e_2, e_3, e_4, e_5}}
P2={{e_1},{e_2, e_3, e_4, e_5}}
P3={{e_1},{e_2, e_3},{e_4,e_5}}
P4={{e_1},{e_2, e_3},},{e_4},{e_5}}
P5={{e_1},{e_2},{e_3},{e_4},{e_5}}

On qualifie particulièrement cette classification de « classification descendante hiérarchique » : elle va d'une partition à un ensemble à une partition de singletons.
À l'inverse, la technique que nous avons utilisée pour traiter les jeux de données de Soclab est une classification ascendante hiérarchique. La méthode itérative commence sur une partition de singletons pour relier petit à petit les individus présentant de fortes similarités. Ces informations peuvent être visualisées sur un dendrogramme.

c. Méthode des cartes auto-organisatrices

Les cartes auto-organisatrices, également nommées cartes auto-adaptatives ou cartes de Kohonen sont des méthodes issues de l'intelligence artificielle et plus particulièrement de l'apprentissage automatique.
Le principe de cette méthode est inspiré du fonctionnement des systèmes nerveux de perception des mammifères. En effet, on a constaté que certaines zones du cerveau présentent la même topologie que le capteur sensoriel qui leur est associé : si l'on prend en exemple le cortex visuel et son capteur associé, l'œil et en particulier la rétine, on constate que deux zones proches dans le cortex visuel le sont également au niveau de la rétine. On peut faire le même constat en ce qui concerne l'appareil auditif ou le bulbe olfactif.
Par ailleurs, on remarque que des *stimuli* de même nature excitent une même région du cerveau : chaque neurone est à la fois spécialisé et connecté à son voisin de sorte à pouvoir recevoir toutes les informations possibles.
C'est donc là le principe de base des cartes mis en place par Teuvo Kohonen en 1984. Chaque neurone de la carte représente donc un groupe bien particulier de données et ses neurones voisins présentent des groupes à la fois différents et proches de ce dernier.
Les cartes de Kohonen sont organisées en grille de structure variable (carrée, rectangulaire, cylindrique, hexagonale, etc.) généralement en deux dimensions à laquelle on applique l'algorithme de Kohonen. C'est un algorithme dit d'apprentissage non supervisé qui cherche à regrouper les observations en classes en respectant le fait que des observations voisines appartiennent à la même classe ou à des classes voisines.
Cet algorithme itératif est organisé de la manière suivante :

- Initialisation aléatoire des prototypes de chaque neurone.

- A chaque étape, une observation est choisie au hasard puis comparée à chaque neurone afin de déterminer quel vecteur code est le plus proche de la donnée. Cette phase est appelée « phase de compétition » et le vecteur le plus proche est nommé classe ou neurone gagnant.
- On modifie la valeur du neurone gagnant de sorte qu'il se rapproche davantage de la valeur de l'observation. Ainsi, il répondra encore mieux à un prochain *stimulus* de même nature. De même, les valeurs des neurones voisins sont également affectées mais de manière plus légère : l'adaptation est d'autant plus forte que les classes sont proches du neurone gagnant. Cette étape est la « phase de coopération » puisqu'elle permet d'adapter et de spécialiser toute une partie de la carte au voisinage du neurone gagnant.
- L'algorithme s'achève lorsque les neurones ne bougent plus ou très peu : la carte représente alors toutes les topologies de données possibles.

Formalisation mathématique

On a deux espaces indépendants ; l'espace des données de grande dimension (espace d'entrée) et l'espace des représentations (ou carte, espace de sortie) de dimension réduite. On cherche à trouver une projection entre ces deux espaces conservant la topologie des données (fixée *a priori*) en adaptant des prototypes W de manière à ce que des observations proches dans l'espace d'entrée soient associées au même neurone ou à un neurone voisin dans l'espace de sortie.
Définition du nombre de classes notées n.
Pour chaque unité i (i=1, …, n), on initialise de manière aléatoire des prototypes W.
A chaque itération t,

- Choix d'une observation X(t), au hasard
- Détection du neurone j* gagnant : celui dont le prototype est le plus proche de l'observation X(t) : j*=min $||X(t)-W_j(t)||$
- On met à jour les vecteurs codes voisins et le vecteur code du neurone gagnant de manière à ce que l'adaptation soit d'autant plus forte que les neurones sont voisins de j*

Le nouveau prototype est définit par : $W_j(t+1)=W_j(t)+\Delta W_j(t)$
Avec $\Delta W_j(t) = \varepsilon(t) \cdot hj^*(j,t) \cdot (X(t)-W_j(t))$
ε(t), le paramètre d'adaptation qui contrôle la vitesse d'apprentissage. Si ε est trop petit, le modèle ne s'adapte pas assez aux données et s'il est trop grand, le modèle peut devenir instable.
hj*(j,t), la fonction de voisinage qui est une fonction continue de forme gaussienne

Il existe de nombreuses variantes pour l'algorithme des cartes auto-organisatrices et celle utilisée pour le codage de la fonction SOM() n'est pas tout à fait la même (El Golli *et al.*, 2006). Voici l'algorithme en question :

Algorithme 1 La version *batch* de l'algorithme SOM

1: Choisir une valeur initiale pour les prototypes $(\mathbf{p}_c)_{c\in C}$ {Étape d'initialisation}
2: **Pour** $l = 1$ à L **faire**
3: **Pour tout** élément $\mathbf{x}$ de Ω **faire** {Étape d'affectation}
4: calculer

$$f(\mathbf{x}) = \arg\min_{r\in C} \gamma^T(\mathbf{x}, r)$$

5: **Fin pour**
6: **Pour tout** neurone $c \in C$ **faire** {Étape de représentation}
7: calculer

$$\mathbf{p}_c = \frac{\sum_{\mathbf{x}_i\in\Omega} K^T(\delta(f(\mathbf{x}_i), c))\,\mathbf{x}_i}{\sum_{\mathbf{x}_i\in\Omega} K^T(\delta(f(\mathbf{x}_i), c))}$$

8: **Fin pour**
9: **Fin pour**

Figure 49 : Algorithme de la fonction SOM()

Avec Ω, un ensemble de n vecteurs,
C, les M neurones de la carte,
p_c, les prototypes choisis dans C,
δ(c,r), la longueur du plus court chemin entre c et r, et

$$\gamma^T(x,r) = \sum_{c \in C} K^T(\delta(r,c)) \| p_c - x \|^2$$

avec K^T, une fonction telle que lorsque T est élevé, $K^T(x)$ est proche de 1 tandis que lorsque T est bas, K^T tend très vite vers 0. Le but recherché est de transformer la fonction δ(c,r) en mesure de proximité effective entre c et r. Par ailleurs, au fur et à mesure des itérations, T décroît afin de mener à la stabilité de la solution.

2. Outils

a. R

R est un environnement logiciel utilisé pour le traitement des données et l'analyse statistique. C'est un logiciel libre (donc gratuit) distribué par GNU Public Licence. Existant depuis plus de 10 ans, il a été influencé par le langage S. Disponible sous toutes plateformes, Linux/GNU, Windows, Mac OS X, NetBSD, etc., le logiciel R est devenu l'un des enjeux du projet GNU.
Après téléchargement du logiciel, on possède la version de base de R. A cette version basique peuvent s'ajouter de multiples packages. L'une des principale force de R réside dans ce concept : étant un logiciel libre, c'est la communauté des utilisateurs de R qui produisent ces packages. Contrairement à d'autres logiciels, celui-ci n'a pas besoin d'attendre des subventions pour développer tel ou tel module complémentaire : les utilisateurs s'en chargent eux-mêmes. De plus, les mises à jour sont rapidement partagées sur le web et le logiciel évolue en temps réel.
De nombreux sites collaboratifs de partage de scripts sont dédiés à R, tels que RBloggers, AddictedToR (tout deux en anglais), Abcd'R (en français), etc.
Il existe aujourd'hui, environ 3745 packages pour le logiciel R.
De plus, ce logiciel n'est pas resté cantonné à ses fonctions statistiques de base puisqu'il est également utilisé en mathématiques, en SIG et permet même de réaliser de la visualisation 3D.
Enfin, certains utilisateurs ont développé une activité bien particulière, que l'on pourrait nommer du « R art », dont la figure ci-dessous est une jolie illustration.

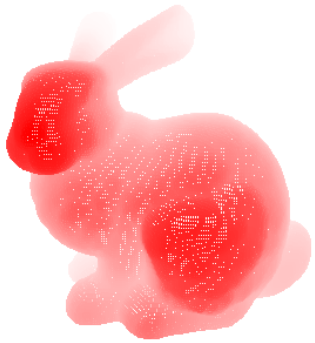


Figure 50: Exemple de "R art"

b. Git

Pour le suivi de ce projet, Mme VILLA-VIALANEIX a mis en place un outil permettant la gestion de versions. Il s'agit de Git, un logiciel libre sous licence GNU GPL.
Git permet de garder une trace des modifications successives apportées à chacun des scripts et permet, si besoin, le retour à une version antérieure de l'un d'entre eux.
Pour ce projet, il a été principalement utilisé pour suivre l'avancement des travaux mais également pour faciliter les corrections.
Le dépôt bénéficie d'une interface graphique permettant de gérer simplement les différentes versions :

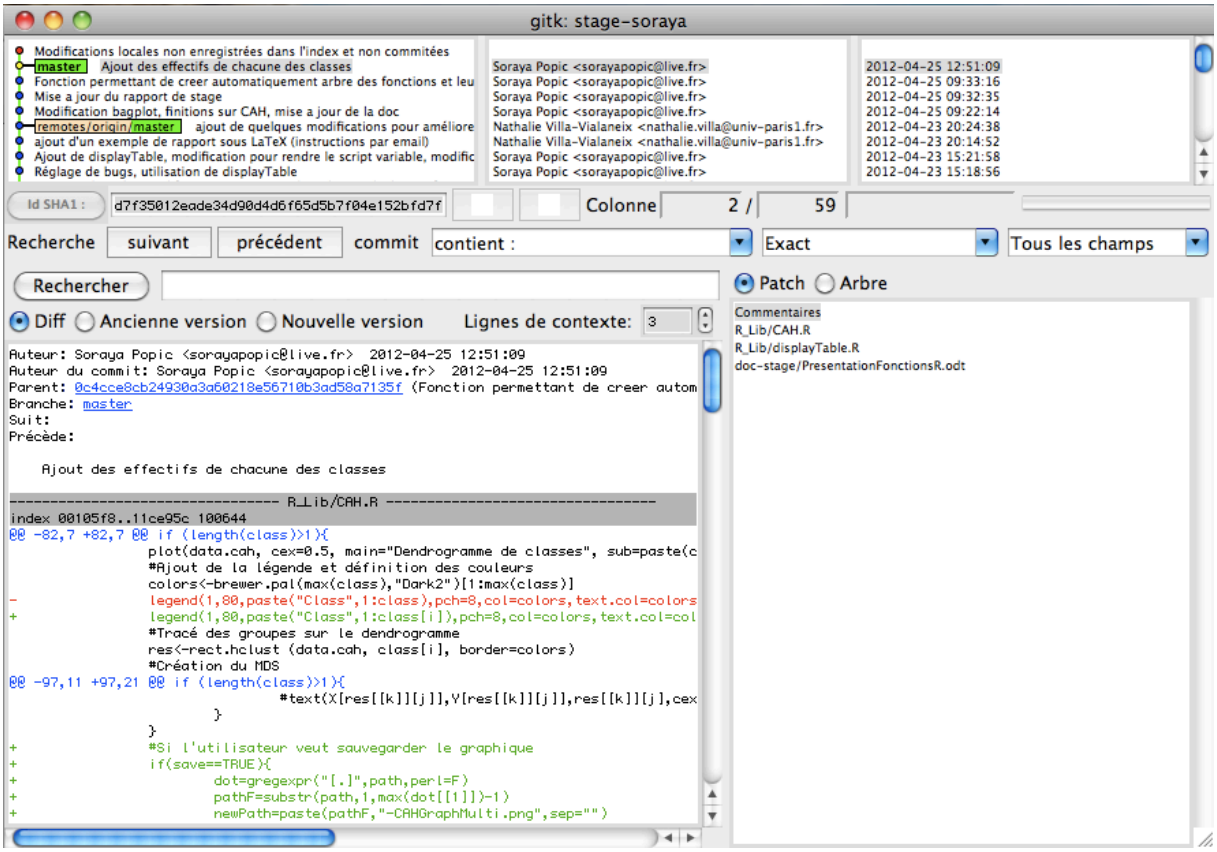


Figure 51: Interface graphique de Git

Pour déposer, mettre à jour, ajouter des fichiers, il suffit d'utiliser le terminal et d'y entrer les commandes git associées.

c. LaTeX

Figure 52 : Logo LaTeX

LaTeX est un langage crée en 1983 par Leslie Lamport permettant de composer des documents. En fait, il s'agit d'un ensemble de macros permettant de faciliter l'utilisation de TeX, un logiciel de composition de documents.

TeX est le fruit du travail de Donald Knuth, par ailleurs connu pour l'écriture de l'ouvrage « The Art of Computer Programming » qui aujourd'hui encore, malgré la rapidité d'évolution d'une science

comme l'informatique, fait toujours office de référence.
La création de TeX est justement lié à cet ouvrage, publié en plusieurs volumes : insatisfait de la manière dont ses livres étaient imprimés, il a commencé en 1977 à créer un logiciel particulièrement adapté à la mise en forme des formules mathématiques. Le but est d'obtenir le meilleur rendu pour un minimum d'effort. Ainsi TeX s'utilise via des balises et s'éxecute via un compilateur. L'utilisation de TeX étant toutefois assez compliquée pour un utilisateur *lambda,* Leslie Lamport a crée LaTeX (pour Lamport TeX) qui est en fait un jeu de macros-commandes qui, grâce à sa simplicité d'utilisation a su s'imposer pour la rédaction de documents scientifiques employant TeX.
LaTeX est principalement utilisé en informatique, mathématiques, physique, ingénierie mais également en musique.
J'ai utilisé cet outil pour la rédaction du manuel utilisateur.

VII. Assurance et contrôle qualité

1. Tests unitaires

Dans cette partie, il s'agit de déceler le maximum de bugs en essayant d'imaginer les différents cas possibles afin d'empêcher le programme de planter ou à défaut, de stopper son exécution et délivrer à l'utilisateur un message d'erreur explicite.
On pourrait dire qu'il existe deux types de test : ceux qui permettant de vérifier que les fonctions réalisent leurs objectifs quelques soient les circonstances et ceux relatifs à la gestion de conflits.
Tous les tests n'ont pas été reportés ici : par exemple, les tests concernant la sauvegarde des fichiers de sortie ont été faits mais n'ont pas fait l'objet d'une fiche par souci de lisibilité mais également pour ne garder que les cas les plus intéressants. Toutes les fonctionnalités de base ont été testées sur plusieurs jeux de données et fonctionnent correctement (il est possible de consulter davantage de tests en annexe, à la fin des scripts de chacune des fonctions).

a. Fiches de test de la fonction CORR()

Identification	Crée le 05/06/12 POPIC Soraya
Fonctionnalité testée	**Suppression de variables** pour l'étude (variables correctement nommées, casse non respectée)
Jeu de données associé	simulation-SEITA.csv
Résultats attendus	Les variables choisies n'apparaissent pas sur le graphique des corrélations
Résultats obtenus	OK

Identification	Crée le 05/06/12 POPIC Soraya
Fonctionnalité testée	**Suppression de variables** pour l'étude (variables mal nommées)
Jeu de données associé	simulation-SEITA.csv
Résultats attendus	L'exécution se stoppe et un message d'erreur avertit l'utilisateur que la variable n'existe pas
Résultats obtenus	OK

b. Fiches de test de la fonction MDS()

Identification	Crée le 05/06/12 POPIC Soraya
Fonctionnalité testée	**Suppression de variables** pour l'étude (variable mal nommées)
Jeu de données associé	simulation-SEITA.csv
Résultats attendus	L'exécution se stoppe et un message d'erreur avertit l'utilisateur que la variable n'existe pas
Résultats obtenus	OK

Identification	Crée le 05/06/12 POPIC Soraya
Fonctionnalité testée	**Suppression de variables** à la fois pour la construction de MDS et pour la coloration, casse non respectée

Jeu de données associé	simulation-SEITA.csv
Résultats attendus	Le programme s'exécute et retourne le MDS construit et coloré avec les variables choisies
Résultats obtenus	OK

Identification	Crée le 05/06/12 POPIC Soraya
Fonctionnalité testée	**Affichage des MDS** sur plusieurs fenêtres graphiques dès lors que l'on dépasse l'affichage 4x4
Jeu de données associé	simulation-Touch.csv
Résultats attendus	Production de plusieurs fenêtres graphique comprenant chacune un maximum de 16 MDS
Résultats obtenus	OK

c. Fiches de test de la fonction CAH()

Identification	Crée le 06/06/12 POPIC Soraya
Fonctionnalité testée	**Affichage des boîtes à moustaches** en sélection multiple
Jeu de données associé	simulation-SEITA.csv
Résultats attendus	L'exécution se stoppe et un message d'erreur avertit l'utilisateur que l'on ne peut afficher de graphique en sélection de classes multiples
Résultats obtenus	OK

Identification	Crée le 06/06/12 POPIC Soraya
Fonctionnalité testée	**Suppression de variables** en sélection manuelle, simple ou multiple, casse non respectée
Jeu de données associé	simulation-SEITA.csv
Résultats attendus	L'exécution se déroule normalement et l'on obtient en sortie les graphiques demandés
Résultats obtenus	OK

Identification	Crée le 06/06/12 POPIC Soraya
Fonctionnalité testée	**Choix de sélection impossible** : demande manuelle et automatique à la fois
Jeu de données associé	simulation-SEITA.csv
Résultats attendus	L'exécution se stoppe et le message d'erreur précise à l'utilisateur de choisir entre l'une ou l'autre des méthodes
Résultats obtenus	OK

d. Fiches de test de la fonction SOM()

Identification	Crée le 06/06/12 POPIC Soraya
Fonctionnalité testée	**Choix d'un type de grille inexistant**

Jeu de données associé	simulation-SEITA.csv
Résultats attendus	L'exécution se stoppe et le message d'erreur précise à l'utilisateur les choix de grille existants
Résultats obtenus	OK

Identification	Crée le 06/06/12 POPIC Soraya
Fonctionnalité testée	**Affichage des études des moyennes et écarts types** sur plusieurs fenêtres graphiques dès lors que l'on dépasse l'affichage 4x4
Jeu de données associé	simulation-Touch.csv
Résultats attendus	Production de plusieurs fenêtres graphique comprenant chacune un maximum de 16 grilles
Résultats obtenus	OK

VIII. Bilan

J'ai trouvé ce stage très intéressant dans le sens où il est à la croisée des chemins entre trois matières : la sociologie, l'informatique et les mathématiques.

Avant d'étudier la statistique, j'ai obtenu un diplôme dans le domaine de l'Information-Communication. Ce type de cursus faisant la part belle aux sciences sociales, j'ai été sensibilisée au domaine de la sociologie. C'est une science qui m'a toujours intéressée et j'ai pu mettre à profit mes quelques connaissances afin de mieux appréhender le sujet.

Ce stage a également été l'occasion de découvrir comment il était possible de formaliser, théoriser, des comportements humains et des relations sociales. Cela m'a permis de comprendre comment était faite la jointure entre les sciences humaines et les sciences exactes et grâce à cela, j'appréhende mieux aujourd'hui les concepts d'intelligence artificielle, concepts qui m'apparaissaient de manière floue auparavant. C'est d'ailleurs toute une nouvelle facette de l'informatique que j'ai découverte en comprenant que beaucoup de phénomènes pouvaient être simulé par un algorithme.

Enfin, en ce qui concerne les mathématiques, j'ai découvert principalement deux nouvelles méthodes : le Multi-Dimensionnal Scaling et les cartes auto-organisatrices. De manière plus générale, cela m'a également forcée à ouvrir mes horizons et à na pas rester cantonnée aux quelques méthodes que je connais et maîtrise. De plus, j'ai pu réutiliser des méthodes apprises en DUT et qui n'ont pas forcément été revues en troisième année de licence.

IX. Conclusion

Le manuel utilisateur ainsi que ce mémoire ont pour but de permettre aux sociologues et informaticiens de réaliser de nouvelles analyses statistiques simplement.
Parmi les méthodes mises en place, certaines peuvent faire l'objet d'une interprétation « libre », c'est-à-dire que même si l'on présentait les résultats à des statisticiens, il est possible que chacun d'entre eux fasse des choix différents. Je pense notamment au choix d'un nombre de classes ou de cellules pour la classification et les cartes auto-organisatrices ou encore à l'étude du MDS qui passe par des observations plus subjectives, dépendant de la manière qu'à un individu de visualiser des couleurs, des données et d'en synthétiser l'information.
Bien que cela ne change pas du tout au tout l'analyse, elle sera tout de même sujette à des variations. C'est la limite de mon étude et c'est dans ce but que j'ai essayé de donner tous les outils possibles afin que les analyses et les choix réalisés soient les plus justes et les meilleurs possibles.

Par ailleurs, le plan d'étude mis en place au début de l'étude a été efficace puisqu'il n'a pas ou peu été sujet à modifications au cours du stage.
Chacune des fonctions a permis de mettre à jour des comportements de simulations différents et de révéler certains aspects jusqu'alors inconnus des jeux de données. En effet, l'outil que j'ai mis en place est le premier qui permette de répondre à une problématique ciblée concernant les données issues de SocLab et j'espère que mon travail permettra de mieux comprendre les relations multi-agents tout en facilitant l'amélioration des modèles conçus.

X. Annexes

1. Document applicable

Ce à quoi on s'intéresse :

On s'intéresse :

- A l'analyse des résultats de simulation, éventuellement d'analyse de sensibilité des paramètres du modèle de simulation, pour une organisation précise : comment interpréter ces résultats, quelles connaissances peut-on en extraire sur le fonctionnement de cette organisation ?
- Aux propriétés de l'algorithme de simulation : étant données les propriétés de la structure d'une organisation et la valeur des paramètres de simulation, que peut-on (pré)dire sur les résultats de simulation ?

Satisfaction globale : c'est la somme (ou la moyenne) de la satisfaction de tous les acteurs.
Elle ne figure pas dans les fichiers de résultats, mais c'est une donnée importante.
Elle a un min et un max qui sont déterminés par la structure du jeu et que l'on a par le module « state analysis » de SocLab. On est intéressé par la valeur en quantité de la satis globale, mais aussi la valeur relative, en pourcentage, par le ratio qui est produit par l'algorithme : (satis – satis_min)/(satis_max – satis_min) que l'on a aussi par state analysis.
(satis_max – satis_min) est une propriété significative d'une organisation, elle détermine dans quelle mesure la coopérativité des acteurs est payante.

Performance de l'algorithme : quels sont les déterminants du ratio d'une organisation ? Ne peut être déterminé qu'en comparant les résultats produits par plusieurs organisations !
Influence respective de la ténacité et de la réactivité.

Relations déterminantes : Les relations qui ont le plus d'influence sur les résultats de simulation. Celles qui interviennent les premières composantes de l'ACP, celles qui ont l'écart type le plus important ?
Quantifier l'importance relative de chaque relation (somme de l'importance des relations = 1).
En déduire l'importance relative de chaque acteur.
((mettre en relation cette importance avec le pouvoir/influence de l'acteur))
((mettre en relation importance des relations avec leur force))

Graphe des corrélations entre les relations : quelle relation détermine la valeur prise par quelles autres, et pour combien ?
En déduire la même chose pour les acteurs.

Dispersion des résultats de simulation : Ce serait bien de trouver un indicateur de la focalisation/dispersion des résultats de simulation, dans quelle mesure ils sont groupés comme par exemple dans DP4Synthèse.xls provenant de « Conflict1<->3.org ». Un étudiant a fait un rapport sur le sujet, « concordance.pdf », mais sans conclure.
Interprétation : moins c'est dispersé, plus les acteurs semblent être contraints par la structure de l'organisation.

Existence de différents modes : dans quelle mesure peut-on identifier des pôles autour desquelles les résultats se regroupent.

Si c'est le cas, on peut considérer que l'organisation a une certaine variété, *ie* : elle peut fonctionner de différentes façons.

Fichiers résultats de simulation

- les valeurs de chaque variable <acteur>satisfaction se déduisent analytiquement de celles des variables <relation>état. Les choses intéressantes à trouver sont donc entre les variables <relation> et entre les variables <acteur>
- nb-step peut être considérée comme une variable indépendante : les résultats dépendent souvent de la longueur des simulations
- Une simulation s'arrête, elle « converge » lorsque, pour chaque acteur, satisfaction ≥ seuil. on peut donc présumer que l'acteur pour lequel seuil-satisfaction est le plus faible est le dernier à avoir satisfait cette condition et est donc responsable de la durée de la simulation. je ne sais pas s'il y a grand-chose à en tirer.

Analyse de Sensibilité :

- les variables param-i sont celles indépendantes, qui varient d'une expérience à l'autre. Par exemple dans le jeu de données Bolet, on avait une variable param0 qui resprésentait la ténacité du chef d'atelier. Pour chaque simulation, on faisait varier ce paramètre afin de voir l'impact obtenu sur les autres variables.
- chaque expérience est la moyenne de n runs. Les premières expériences prennent les valeurs min et max des variables, les valeurs des autres expériences sont choisies aléatoirement

Les modèles d'organisation

Seita : les simulations donnent des écarts types importants

DP 2 : dilemme du prisonnier
on a fait varier la répartition des enjeux, c'est à dire l'intensité du conflit.
voir chapitre 5 de la thèse et le fichier dilemme2-Comparaison.odt : quelle est la relation entre nombre de pas et la satisfaction ?

Conflict : modèle linéaire, 1 acteurs contre 3 autres
version à 4 relations : (résultats obtenus avec des versions différentes de l'algo de simulation)
2 simulations, avec réactivité de 3 et de 8
1 analyse de sensibilité sur réactivité et ténacité variant de 1 à 10
DP4Synthèse : résultats qui convergent tous vers 3 situations différentes bien caractérisées
version à 6 relations : A1 a une relation pour chacun des 3 autres

Plan International
chap. 5 thèse de Paul, on peut ne pas tenir compte de l'acteur COM, l'acteur BD a lui aussi un statut un peu extérieur au jeu

Touch : modèle à 10 acteurs

Bolet : exemple article AAECS

2. Présentation de l'intervention orale

Analyse statistique de résultats de simulation

Mercredi 27 juin 2012

POPIC Soraya
Etudiante en troisième année de Statistique et Informatique Décisionnelle (SID) à l'Université Paul Sabatier de Toulouse

Résumé : Le but de mon travail sur les jeux de données issus de SocLab est la mise en place d'outils statistiques adaptés afin de répondre à des questions telles que : comment fonctionnent les organisations ? Quelles relations ou acteurs prédominants influencent les organisations ? Existe-t-il différents modes d'organisation ?
Lors de ma présentation, j'exposerai les différentes fonctions que j'ai créées via le logiciel R, ainsi que leurs paramètres. Mon plan d'étude inclut des analyses des corrélations, de la dispersion mais également des méthodes de classification et la création de cartes auto-organisatrices (ou cartes de Kohonen).
Chacune des méthodes sera étayée d'un exemple d'analyse basé sur le cas SEITA.

Figure 53 : Résume de la présentation orale

3. Script des fonctions

a. Fonction selectFile()

Cette fonction permet de récupérer l'adresse complète spécifiant le chemin d'accès au fichier sélectionné (ex: "/Users/smac/Desktop/stage-soraya/data1/simulation-Seita.csv").
Par la suite, cette adresse me permet également d'enregistrer les sorties sans redemander à l'utilisateur où il souhaite enregistrer ; j'extrais le .csv et à la place, j'insère « -*nomGraphique*.png »(ex:"/Users/smac/Desktop/stage-soraya/data1/simulation-Seita-*nomGraphique*.png"). Ainsi, l'utilisateur retrouvera le graphique qu'il a souhaité réaliser au même endroit que son fichier de données.

```
##################################
#Description: Cette fonction permet de selectionner un fichier via l'arborescence de
 fichiers
#Parametres: Il n'y a pas de parametre d'entree. On obtient en sortie le chemin complet
 menant au fichier selectionne
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120404 | POPIC            | Creation de la premiere version
##################################

#Fontion permettant de selectionner un fichier via l'arborescence de fichiers
```

```
selectFile<-function(){
 file<-file.choose()
 return(file)
}
```

b. Fonction delete()

```
#################################
#Description: Cette fonction permet du supprimer les donnees inutiles issues des sorties
 de SocLab
#Parametres: Elle accepte en parametre d'entree le fichier contenant les donnees
#En sortie, elle renvoit le fichier de donnees nettoye
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120413 | POPIC            | Creation de la premiere version
#################################

delete<-function(data){
 #Suppression des colonnes "run" (inutile), "X" (que des valeurs manquantes), "num_exp"
 if(isTRUE(grep("runs", names(data), ignore.case=TRUE)!=0)){
      col<-grep("runs", names(data), ignore.case=TRUE)
      data<-data[,-col]
 }

 if(isTRUE(grep("num_exp", names(data), ignore.case=TRUE)!=0)){
      col<-grep("num_exp", names(data), ignore.case=TRUE)
      data<-data[,-col]
 }

 if(isTRUE(grep("X", names(data), ignore.case=TRUE)!=0)){
      col<-grep("X", names(data), ignore.case=TRUE)
      data<-data[,-col]
 }

return(data)
}
```

c. Fonction displayTable()

```
#################################
#Description: Cette fonction permet de creer le tableau d'analyse des moyennes dans une
 fenêtre graphique
#Parametres: Elle accepte en parametre d'entree le contenu du tableau
#En sortie, elle renvoit le tableau d'analyse de la moyenne simplifie avec des symboles
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120424 | POPIC            | Creation de la premiere version
#################################
#Librairie:
library(RColorBrewer)

display.table<-function(Tab){
 x11()
 #Definition des couleurs de classes
 colors<-brewer.pal(max(nrow(Tab)),"Dark2")[1:max(nrow(Tab))]
 #Definition des couleurs des symboles
```

```
 the.cols<-brewer.pal(nrow(Tab),"RdBu")[nrow(Tab):1]
 par(mar=rep(0.1,4),bg="lightgrey",mfrow=c(ncol(Tab)+1,nrow(Tab)+1))
 plot.new()
 #Creation de l'intitule des colonnes
 for (ind.c in 1:nrow(Tab)) {
      label<-paste("*Class",rownames(Tab)[ind.c],"*")
      effectif<-paste(" ( effectif=",Tab[,(ncol(Tab))][ind.c],")")
      label<-paste(label,effectif)
      plot(0,0,axes=F,type="n",xlab="",ylab="",main="")
      text(0,0,label,cex=1.5,lwd=3,col=colors[ind.c])
      box("figure")
 }
 #Remplissage des lignes
 for (ind.v in 1:ncol(Tab)) {
      plot(0,0,axes=F,type="n",xlab="",ylab="",main="")
      text(0,0,colnames(Tab)[ind.v],cex=1.5,lwd=2)
      box("figure")
      classes.v<-cut(Tab[,ind.v],nrow(Tab),labels=F,include.lowest=TRUE)
      for (ind.c in 1:nrow(Tab)) {
           plot(0,0,axes=F,type="n",xlab="",ylab="",main="")
           points(0,0,pch=16,cex=5,col=the.cols[classes.v[ind.c]])
           box("figure")
      }
 }
}

#display.table(tabMoy)
```

d. Fonction save()

```
#################################
#Description: Cette fonction permet de sauvegarder les graphiques obtenus en sortie des
 fonctions
#Parametres: Son parametre d'entree est le nom du fichier a sauvegarder
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120523 | POPIC            | Creation de la premiere version
#################################

graphSave<-function(name,vPath){
 dot<-gregexpr("[.]",vPath,perl=F)
 pathF<-substr(vPath,1,max(dot[[1]])-1)
 newPath<-paste(pathF,name,sep="")
 dev.print(png, file=newPath, width=800, height=600)
}

#save(name="-graphName.png")
```

e. Fonction CORR()

```
#################################
#Description: Cette fonction permet de generer automatiquement un graphique des
 correlations
#Parametres: save permet de sauvegarder le graphique
#delVar permet de supprimer des variables de l'etude
#delim permet de changer le separateur de colonnes
# Suivi version :
```

```
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|---------------------------------------
#  1.0 | 20120405 | POPIC            | Creation de la premiere version
#################################
#Librairies
library(ellipse)
library(RColorBrewer)
#display.brewer.all() pour voir toutes les palettes de couleurs

CORR<-function(delVar=0,delim="\t",save=FALSE){
 #Recuperation du fichier contenant les donnees avec selection via arborescence
 path<-selectFile()
 data<-read.delim(path,header=TRUE,sep=delim)

 #Suppression des colonnes "run" (inutile), "X" (que des valeurs manquantes), "num_exp",
 "nb_step"
 data<-delete(data)
 #Traitement d'eventuelles donnees manquantes
 data<-na.omit(data)

 #Suppression des variables indesirables
 if(delVar!=0){
      #Traitement des variables a supprimer
      delVar1<-paste(",",delVar,",",sep="")
      #On obtient les variables sous cette forme : ",Var1,Var2,...,VarN,"
      virgule<-gregexpr("[,]",delVar1,perl=F)
      nbvirgule<-length(virgule[[1]])
      nbVar<-nbvirgule-1
      Var<-0

      #Extraction des noms des variables
      for(i in 1:nbVar){
           Var[i]<-substr(delVar,(virgule[[1]][i]),(virgule[[1]][i+1])-2)
      }

      #Suppression des variables que l'on souhaite exclure de l'etude
      for(i in 1:nbVar){
           if(isTRUE(grep(Var[i], names(data), ignore.case=TRUE)!=0)){
                col<-grep(Var[i], names(data), ignore.case=TRUE)
                data<-data[,-col]
           }
           else{
                stop ("Le nom de variable ",Var[i]," n'existe pas")
           }
      }
 }

 #Creation de la matrice de correlation
 correlation<-cor(data,use="complete.obs")

 #Creation d'intervalles et gestion des couleurs
 colors<-brewer.pal(11,"RdBu")[11:1]
 bornes<-cut(correlation,breaks=seq(-1,1,length=10),include.lowest=TRUE)

 #Creation du graphique
 #Variation de la taille de la police selon le nombre de variables
 if (length(data)>20){
      plotcorr(correlation, col=colors[bornes], main="Graphique des corrélations entre
```

```
 les variables",cex.lab=0.6,asp=1,mar=0.1+c(1,1,3,1))
 }
 else{
     plotcorr(correlation, col=colors[bornes], main="Graphique des corrélations entre
 les variables",cex.lab=1,asp=1,mar=0.1+c(1,1,3,1))
 }

 #Si l'utilisateur veut sauvegarder le graphique
 if(save==TRUE){
     graphSave(name="-CORRgraph.png",vPath=path)
 }
}

##### QUELQUES TESTS #####
### Doit fonctionner ##
#CORR()
#CORR(delVar="ChefAtelier.SeuilSatisfaction,OuvProd.SeuilSatisfaction,OuvEnt.SeuilSatisfa
 ction")
## Doit planter ##
#CORR(delVar="Enretien")
```

f. Fonction MDS()

```
#################################
#Description: Cette fonction permet de tracer un graphique representant la distance
 (euclidienne) entre les individus
#Parametres:Elle possede 5 parametres d'entree
#delVar qui permet de specifier les variables que l'on veut exclure du nuage de points
#delCol qui correspond aux variables que l'on va utiliser pour les couleurs. Par defaut,
 delCol est egale a delVar
#bagPlot permet de tracer le bagplot des donnees
#delim permet de changer le separateur de colonnes
#save permet de dire si l'on veut sauvegarder ou non; par defaut, il est a FALSE
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120413 | POPIC            | Creation de la premiere version
#################################
#Librairies:
library(RColorBrewer)
library(aplpack)

MDS<-function(delVar=0,delCol=0,bagPlot=FALSE,delim="\t",save=FALSE){

 #Recuperation du fichier contenant les donnees avec selection via arborescence
 path<-selectFile()
 data<-read.delim(path,header=TRUE,sep=delim)

 #Suppression des colonnes "run" (inutile), "X" (que des valeurs manquantes), "num_exp"
 dataProj<-delete(data)
 #Traitement d'eventuelles donnees manquantes
 dataProj<-na.omit(dataProj)

 if(delVar!=0){
     #Traitement des variables a supprimer
     vProj1<-paste(",",delVar,",",sep="")
     #On obtient les variables sous cette forme : ",Var1,Var2,...,VarN,"
     virgule<-gregexpr("[,]",vProj1,perl=F)
```

```
     nbvirgule<-length(virgule[[1]])
     nbVar<-nbvirgule-1
     Var<-0

     #Extraction des noms des variables
     for(i in 1:nbVar){
          Var[i]<-substr(delVar,(virgule[[1]][i]),(virgule[[1]][i+1])-2)
     }

     #Suppression des variables que l'on souhaite exclure de l'etude
     for(i in 1:nbVar){
          if(isTRUE(grep(Var[i], names(dataProj), ignore.case=TRUE)!=0)){
               col<-grep(Var[i], names(dataProj), ignore.case=TRUE)
               dataProj<-dataProj[,-col]
          }
          else{
               stop("Le nom de Variable ",Var[i]," n'existe pas")
          }
     }
}

#Creation de la matrice de couleur
dataCoul<-dataProj

if(delCol!=0){
     #Traitement des variables a supprimer de la coloration
     vCoul1<-paste(",",delCol,",",sep="")
     #On obtient les Variables sous cette forme : ",Var1,Var2,...,VarN,"
     virgule<-gregexpr("[,]",vCoul1,perl=F)
     nbvirgule<-length(virgule[[1]])
     nbVar<-nbvirgule-1
     Var<-0

     #Extraction des noms des variables
     for(i in 1:nbVar){
          Var[i]<-substr(delCol,(virgule[[1]][i]),(virgule[[1]][i+1])-2)
     }

     #Suppression des variables que l'on ne souhaite pas mettre en couleur
     for(i in 1:nbVar){
          if(isTRUE(grep(Var[i], names(dataCoul), ignore.case=TRUE)!=0)){
               col<-grep(Var[i], names(dataCoul), ignore.case=TRUE)
               dataCoul<-dataCoul[,-col]
          }
          else{
               stop("Le nom de Variable ",Var[i]," n'existe pas ou la variable a déjà
été supprimée\n")
          }
     }
}

#Centrage et reduction de la matrice de projection
dataCR<-scale(dataProj)
#Creation d'une matrice de distance
mat.dataCR<-dist(dataCR, method = "euclidean")
#Creation des axes du nuage de points:
X<-cmdscale(mat.dataCR)[,1]
Y<-cmdscale(mat.dataCR)[,2]
```

```
#Choix des options pour la coloration
nbvCoul<-length(dataCoul)
coloring<-brewer.pal(9,"Oranges")[1:9]

#Trace des graphiques
#Traitement de l'affichage graphique
nbLig<-ceiling(sqrt(nbvCoul))

#Si l'on doit afficher plus de 16 MDS
if (nbLig>4){
     nbdevice<-ceiling(nbvCoul/16)
     k<-1
     l<-16
     #Trace des MDS
     for (i in 1:nbdevice){
          x11()
          par(mfrow=c(4,4))
          for (j in k:l){
               bornes<-
cut(dataCoul[,j],breaks=seq(range(dataCoul[,j],na.rm=TRUE)[1],range(dataCoul[,j],na.rm=T
RUE)[2],length=8),include.lowest=TRUE)
               plot(X,Y,col=coloring[bornes],type="p",pch=16,main="Matrice de
distance",sub=names(dataCoul[j]))
          }
          k<-l+1
          l<-l+16
          if(i==nbdevice-1){
               l<-ncol(dataCoul)
          }
          #Si l'utilisateur veut sauvegarder les graphiques
          if(save==TRUE){
               dot<-gregexpr("[.]",path,perl=F)
               pathF<-substr(path,1,max(dot[[1]])-1)
               path1<-paste(pathF,"-MDSGraph",sep="")
               path2<-paste(path1,i,sep="")
               newPath<-paste(path2,".png",sep="")
               dev.print(png, file=newPath, width=1000, height=800)
          }
     }
}

#Si l'on doit afficher 16 MDS ou moins
if (nbLig<=4){
     par(mfrow=c(nbLig,nbLig))
     #Trace des MDS
     for (i in 1:nbvCoul){
          bornes<-
cut(dataCoul[,i],breaks=seq(range(dataCoul[,i],na.rm=TRUE)[1],range(dataCoul[,i],na.rm=T
RUE)[2],length=8),include.lowest=TRUE)
          plot(X,Y,col=coloring[bornes],type="p",pch=16,main="Matrice de
distance",sub=names(dataCoul[i]))
     }
     #Si l'utilisateur veut sauvegarder le graphique
     if(save==TRUE){
          graphSave(name="-MDSgraph.png",vPath=path)
     }
}
```

```
 #Trace du bagplot, a la demande
 if(bagPlot==TRUE){
      x11()
      bagplot(X,Y,factor=2,dkmethod=1,main="Bagplot des
 observations",axes=F,pch=8,show.looppoints=TRUE, show.outlier=TRUE,show.bagpoints=TRUE,
 show.whiskers=TRUE,show.loophull=TRUE,show.baghull=TRUE, verbose=FALSE)
      box()
      text(X,Y,as.character(data[,1]),cex=1)
      if(save==TRUE){
           graphSave(name="-MDSbagplot.png",vPath=path)
      }
 }
}

##### QUELQUES TESTS #####
### Doit fonctionner ##
#MDS()
#MDS(delVar="ChefAtelier.SeuilSatisfaction,OuvProd.SeuilSatisfaction,OuvEnt.SeuilSatisfac
 tion")
#MDS(bagPlot=TRUE)
#MDS("ChefAtelier.Satisfaction,ChefAtelier.SeuilSatisfaction",bagPlot=TRUE)
#MDS(delVar="ChefAtelier.Satisfaction,ChefAtelier.SeuilSatisfaction",delCol="entretien",s
 ave=TRUE)
## Doit planter ##
#MDS(delVar="Enretien")
```

g. Fonction CAH()

```
##################################
#Description: Cette fonction permet de realiser une classification ascendante
 hierarchique
#Parametres: class, choix du nombre de classes dessinees sur le dendrogramme
#manSelect, pour selectionner a la main, sur le dendrogramme, un nombre de classes
 (selection manuelle)
#save permet de sauvegarder le graphique (dendrogramme+MDS)
#delVar permet de supprimer des variables de l'etude
#MAV permet d'importer le tableau d'analyse des moyennes en fichier csv
#bPlot  permet d'obtenir les boites a moustaches de chaque classe pour chaque variable
#delim permet de changer le separateur de colonne
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120531 | POPIC            | Creation de la premiere version
##################################
#Librairies:
library(stats)
library(RColorBrewer)
#display.brewer.all() pour voir toutes les palettes de couleurs
library(xtable)

CAH<-
 function(class=0,delVar=0,manSelect=FALSE,MAV=FALSE,bPlot=FALSE,delim="\t",save=FALSE){

 #Gestion de conflit
 if ((class>=1)&&(manSelect==TRUE)){
```

```
     stop("Impossible de spécifier un nombre de classe de manière automatique et
manuelle à la fois")
}
if((class==0)&&(manSelect==FALSE)){
     stop("Veuillez spécifier un nombre de classe ou effectuer une sélection manuelle")
}

#Recuperation du fichier contenant les donnees avec selection via arborescence
path<-selectFile()
data<-read.delim(path,header=TRUE,sep=delim)

#Suppression des colonnes "run" (inutile), "X" (que des valeurs manquantes), "num_exp"
data<-delete(data)
#Traitement d'eventuelles donnees manquantes
data<-na.omit(data)

#Suppression des variables indesirables
if(delVar!=0){
     #Traitement des variables a supprimer
     delVar1<-paste(",",delVar,",",sep="")
     #On obtient les variables sous cette forme : ",Var1,Var2,...,VarN,"
     virgule<-gregexpr("[,]",delVar1,perl=F)
     nbvirgule<-length(virgule[[1]])
     nbVar<-nbvirgule-1
     Var<-0

     #Extraction des noms des variables
     for(i in 1:nbVar){
          Var[i]<-substr(delVar,(virgule[[1]][i]),(virgule[[1]][i+1])-2)
     }

     #Suppression des variables que l'on souhaite exclure de l'etude
     for(i in 1:nbVar){
          if(isTRUE(grep(Var[i], names(data), ignore.case=TRUE)!=0)){
               col<-grep(Var[i], names(data), ignore.case=TRUE)
               data<-data[,-col]
          }
          else{
               stop ("Le nom de Variable ",Var[i]," n'existe pas")
          }
     }
}

#Centrage et reduction des donnees
dataCR<-scale(data)
#Creation d'une matrice de distance:
mat.dataCR<-dist(dataCR, method="euclidean")
#Creation des axes du nuage de points:
X<-cmdscale(mat.dataCR)[,1]
Y<-cmdscale(mat.dataCR)[,2]

#Mise en oeuvre de la methode de classification ascendante hierarchique
data.cah<-hclust(mat.dataCR, method="ward")

#Trace du dendrogramme avec class, le nombre de classes choisies
#Si l'utilisateur a choisit plusieurs valeurs de classes
if (length(class)>1){
     if((MAV==TRUE)||(bPlot==TRUE)){
```

```
          stop("Impossible d'enregistrer les données relatives à l'analyse des moyennes
ou d'afficher les boîtes à moustaches en sélection de classes mutliple")
     }
     vClass<-0
     nbLig<-ceiling(sqrt(length(class)*2))
     par(mfrow=c(nbLig,nbLig))
     for (i in 1:length(class)){
          #Creation du dendrogramme
          plot(data.cah, cex=0.5, main="Dendrogramme de classes",
sub=paste(class[i],"Classes"), ylim=c(0,100), ylab="Inertie intra-
groupe",font.sub=2,hang=-1)
          #Ajout de la legende et definition des couleurs
          colors<-brewer.pal(max(class),"Dark2")[1:max(class)]

     legend(1,80,paste("Class",1:class[i]),pch=8,col=colors,text.col=colors,bty="n",cex=
0.5)
          #Trace des groupes sur le dendrogramme
          res<-rect.hclust (data.cah, class[i], border=colors)
          #Creation du MDS
          vClass<-cutree(data.cah,class[i])
          #Coloration des classes
          plot(X,Y,type="n",main="Matrice de distance")
          #k=nombre de classes
          for(k in 1:(length(res))){
               #i=nombre d'individus dans la classe
               for(j in 1:(length(res[[k]]))){

     points(X[res[[k]][j]],Y[res[[k]][j]],type="p",sub=paste(class[i],"Classes"),font.su
b=2,pch=16,col=colors[k])
               }
          }
          #Si l'utilisateur veut sauvegarder le graphique
          if(save==TRUE){
               graphSave(name="-CAHgraphMulti.png",vPath=path)
          }
          #Creation du tableau d'analyse des moyennes
          #Creation d'un vecteur contenant le numero de classe de chaque individu
          vect<-0
          #Creation d'un vecteur content le nombre d'individu dans chaque classe
          eff<-0
          #k=nombre de classes
          for(k in 1:(length(res))){
               eff[k]<-length(res[[k]])
               #j=nombre d'individus dans la classe
               for(j in 1:(length(res[[k]]))){
                    #i=nombre d'individus au total
                    for(i in 1:length(vClass)){
                         if(res[[k]][j]==i){
                         vect[i]<-k
                         }
                    }
               }
          }
          #On ajoute une colonne contenant les numeros de classe au jeu de donnees
          step1<-merge(data,vect,by="row.names",sort=FALSE)
          tabMoy<-aggregate(step1[,-1], list(step1$y), mean, na.rm=TRUE)
          tabMoy<-round(tabMoy,2)
          #Suppression de la colonne "y", utilisee pour la creation du tableau
```

```
          tabMoy<-tabMoy[,-(ncol(tabMoy))]
          #Ajout d'une colonne contenant les effectifs
          tabMoy[,(ncol(tabMoy))+1]<-eff
          names(tabMoy)[ncol(tabMoy)]<-"Effectifs"
     }
}
#Si l'utilisateur n'a choisit qu'une valeur pour definir la classe
else{
     if(!manSelect){
          par(mfrow=c(1,2))
          vClass<-0
          #Creation du dendrogramme
          plot(data.cah, cex=0.5, main="Dendrogramme de classes", ylim=c(0,100),
ylab="Inertie intra-groupe",hang=-1)
          #Ajout de la legende et definition des couleurs
          colors<-brewer.pal(class,"Dark2")[1:class]

     legend(1,80,paste("Class",1:class),pch=8,col=colors,text.col=colors,bty="n",cex=0.8
)
          #Trace des groupes sur le dendrogramme
          res<-rect.hclust(data.cah, class, border=colors)
          #Creation du MDS
          vClass<-cutree(data.cah,class)
          plot(X,Y,type="n",main="Matrice de distance")
          #Coloration des classes
          #k=nombre de classes
          for(k in 1:(length(res))){
               #i=nombre d'individus dans la classe
               for(i in 1:(length(res[[k]]))){
                    text(X[res[[k]][i]],Y[res[[k]][i]],res[[k]][i]-
1,cex=0.8,col=colors[k])
               }
          }
          #Si l'utilisateur veut sauvegarder le graphique
          if(save==TRUE){
               dot<-gregexpr("[.]",path,perl=F)
               pathF<-substr(path,1,max(dot[[1]])-1)
               name<-paste("-CAHgraph",class,"Cl.png",sep="")
               newPath<-paste(pathF,name,sep="")
               dev.print(png,file=newPath,width=800,height=600)
          }
          #Creation du tableau d'analyse des moyennes
          #Creation d'un vecteur contenant le numero de classe de chaque individu
          vect<-0
          #Creation d'un vecteur contenant le nombre d'individu dans chaque classe
          eff<-0
          #k=nombre de classes
          for(k in 1:(length(res))){
               eff[k]<-length(res[[k]])
               #j=nombre d'individus dans la classe
               for(j in 1:eff[k]){
                    #i=nombre d'individus au total
                    for(i in 1:length(vClass)){
                         if(res[[k]][j]==i){
                              vect[i]<-k
                         }
                    }
               }
```

```
          }
          step1<-merge(data,vect,by="row.names",sort=FALSE)
          tabMoy<-aggregate(step1[,-1], list(step1$y), mean, na.rm=TRUE)
          tabMoy<-round(tabMoy,2)
          #Suppression de la colonne "y", utilisee pour la creation du tableau
          tabMoy<-tabMoy[,-(ncol(tabMoy))]
          #Ajout d'une colonne contenant les effectifs
          tabMoy[,(ncol(tabMoy))+1]<-eff
          names(tabMoy)[ncol(tabMoy)]<-"Effectifs"
          #Suppression de la colonne "group.1" pour l'analyse de la moyenne avec les
symboles
          tabMoy<-tabMoy[,-1]
          display.table(tabMoy)
          #Si l'utilisateur veut sauvegarder le tableau d'analyse des moyennes en csv
          if(MAV==TRUE){
               dot<-gregexpr("[.]",path,perl=F)
               pathF<-substr(path,1,max(dot[[1]])-1)
               tabMoy.table<-xtable(tabMoy)
               #Importation de ce tableau en fichier csv
               name<-paste("-CAHanalyseMoy",class,"Cl.csv",sep="")
               newPath<-paste(pathF,name,sep="")
               write.csv2(tabMoy,file=newPath)
          }
          #Si l'utilisateur veut sauvegarder le tableau d'analyse des moyennes avec les
symboles, en .png
          if(save==TRUE){
               dot<-gregexpr("[.]",path,perl=F)
               pathF<-substr(path,1,max(dot[[1]])-1)
               tabMoy.table<-xtable(tabMoy)
               #Importation de ce tableau en fichier png
               name<-paste("-CAHanalyseMoy",class,"Cl.png",sep="")
               newPath<-paste(pathF,name,sep="")
               dev.print(png,file=newPath,width=800,height=600)
          }
          #Creation de la matrice de boites a moustaches
          if(bPlot==TRUE){
               nbLig<-ceiling(sqrt(length(data)*2))
               #Si l'on a plus de 9 boites a moustaches (affichage 3x3)
               if(nbLig>3){
                    nbdevice<-ceiling(ncol(data)/9)
                    l<-9
                    k<-1
                    for(i in 1:nbdevice){
                         x11()
                         par(mfrow=c(3,3))
                         for(k in k:l){
                              #Definition de la longueur de l'axe des ordonnees
                              limInf<-0
                              limSup<-0
                              #Une satisfaction est comprise entre -100 et 100
                              if(isTRUE(grep("satisfaction", names(data[k]),
ignore.case=TRUE)!=0)){
                                   col<-grep("satisfaction", names(data[k]),
ignore.case=TRUE)
                                   limInf<--100
                                   limSup<-100
                              }
                              #Une relation est comprise entre -10 et 10
```

```
                              if(isTRUE(grep("state", names(data[k]),
ignore.case=TRUE)!=0)){
                                   col<-grep("state", names(data[k]), ignore.case=TRUE)
                                   limInf<--10
                                   limSup<-10
                              }
                              #Si c'est autre chose, on prend le maximum et le minimum
                              else{
                                   limInf<-min(data[,k],na.rm=TRUE)
                                   limSup<-max(data[,k],na.rm=TRUE)
                              }

    boxplot(data[,k]~step1[,(length(step1))],sub=names(data[k]),ylim=c(limInf,limSup),v
arwidth=TRUE,border=c(colors))
                         }
                         k<-l+1
                         l<-l+9
                         if(i==nbdevice-1){
                              l<-ncol(data)
                         }
                         #Sauvegarde
                         if(save==TRUE){
                              dot<-gregexpr("[.]",path,perl=F)
                              pathF<-substr(path,1,max(dot[[1]])-1)
                              path1<-paste(pathF,"-CAHboxPlot",sep="")
                              path2<-paste(path1,i,sep="")
                              newPath<-paste(path2,".png",sep="")
                              dev.print(png, file=newPath, width=800, height=600)
                         }
                    }
               }
               else{
                    x11()
                    par(mfrow=c(nbLig,nbLig))
                    for(i in 1:ncol(data)){
                         #Definition de la longueur de l'axe des ordonnees
                         limInf<-0
                         limSup<-0
                         #Une satisfaction est comprise entre -100 et 100
                         if(isTRUE(grep("satisfaction", names(data[i]),
ignore.case=TRUE)!=0)){
                              col<-grep("satisfaction", names(data[i]),
ignore.case=TRUE)
                              limInf<--100
                              limSup<-100
                         }
                         #Une relation est comprise entre -10 et 10
                         if(isTRUE(grep("state", names(data[i]), ignore.case=TRUE)!=0)){
                              col<-grep("state", names(data[i]), ignore.case=TRUE)
                              limInf<--10
                              limSup<-10
                         }
                         #Si c'est autre chose, on prend le maximum et le minimum
                         else{
                              limInf<-min(data[,i],na.rm=TRUE)
                              limSup<-max(data[,i],na.rm=TRUE)
                         }
```

```
     boxplot(data[,i]~step1[,(length(step1))],sub=names(data[i]),ylim=c(limInf,limSup),v
arwidth=TRUE,border=c(colors))
                    }
                    #Sauvegarde
                    if(save==TRUE){
                         graphSave(name="-CAHboxPlot.png",vPath=path)
                    }
               }
          }
     }
}

#Selection manuelle des classes:
if(manSelect){
     par(mfrow=c(1,2))
     plot(data.cah, cex=0.5, main="Dendrogramme de classes", ylim=c(0,100),
ylab="Inertie intra-groupe", hang=-1)
     data.cah.id<-identify(data.cah)
     #data.cah.id contient les differents individus dans chaque classe
     #Ajout de la legende et definition des couleurs
     colors<-brewer.pal(length(data.cah.id),"Dark2")[1:length(data.cah.id)]

     legend(1,80,paste("Class",1:length(data.cah.id)),pch=8,col=colors,text.col=colors,b
ty="n",cex=0.8)
     #Trace des groupes sur le dendrogramme
     res<-rect.hclust(data.cah, length(data.cah.id), border=colors)
     #Creation du MDS
     vClass<-cutree(data.cah,length(data.cah.id))
     plot(X,Y,type="n",main="Matrice de distance")
     #Coloration des classes
     #k=nombre de classes
     for(k in 1:(length(res))){
          #i=nombre d'individus dans la classe
          for(i in 1:(length(res[[k]]))){
               text(X[res[[k]][i]],Y[res[[k]][i]],res[[k]][i]-1,cex=0.8,col=colors[k])
          }
     }
     #Si l'utilisateur veut sauvegarder le graphique
     if(save==TRUE){
          graphSave(name="-CAHgraph.png",vPath=path)
     }
     #Creation du tableau d'analyse des moyennes
     #Creation d'un vecteur contenant le numero de classe de chaque individu
     vect<-0
     #Creation d'un vecteur content le nombre d'individu dans chaque classe
     eff<-0
     #k=nombre de classes
     for(k in 1:(length(res))){
          eff[k]<-length(res[[k]])
          #j=nombre d'individus dans la classe
          for(j in 1:(length(res[[k]]))){
               #i=nombre d'individus au total
               for(i in 1:length(vClass)){
                    if(res[[k]][j]==i){
                         vect[i]<-k
                    }
               }
          }
```

```
    }
    step1<-merge(data,vect,by="row.names",sort=FALSE)
    tabMoy<-aggregate(step1[,-1],by=list(step1$y),mean, na.rm=TRUE)
    tabMoy<-round(tabMoy,2)
    #Suppression de la colonne "y", utilisee pour la creation du tableau
    tabMoy<-tabMoy[,-(ncol(tabMoy))]
    #Ajout d'une colonne contenant les effectifs
    tabMoy[,(ncol(tabMoy))+1]<-eff
    names(tabMoy)[ncol(tabMoy)]<-"Effectifs"
    #Suppression de la colonne "group.1" pour l'analyse de la moyenne avec les symboles
    tabMoy<-tabMoy[,-1]
    display.table(tabMoy)
    #Si l'utilisateur veut sauvegarder le tableau d'analyse des moyennes en csv
    if(MAV==TRUE){
        dot<-gregexpr("[.]",path,perl=F)
        pathF<-substr(path,1,max(dot[[1]])-1)
        #Importation de ce tableau en fichier csv
        name<-paste("-CAHanalyseMoy",length(data.cah.id),"Cl.csv",sep="")
        newPath<-paste(pathF,name,sep="")
        write.csv2(tabMoy,file=newPath)
    }
    #Si l'utilisateur veut sauvegarder le tableau d'analyse des moyennes avec les
symboles, en .png
    if(save==TRUE){
        dot<-gregexpr("[.]",path,perl=F)
        pathF<-substr(path,1,max(dot[[1]])-1)
        #Importation de ce tableau en fichier png
        name<-paste("-CAHanalyseMoy",length(data.cah.id),"Cl.png",sep="")
        newPath<-paste(pathF,name,sep="")
        dev.print(png,file=newPath,width=800,height=600)
    }
    #Creation de la matrice de boites a moustaches
    if(bPlot==TRUE){
        nbLig<-ceiling(sqrt(length(data)*2))
        if(nbLig>3){
            nbdevice<-ceiling(ncol(data)/9)
            l<-9
            k<-1
            for(i in 1:nbdevice){
                x11()
                par(mfrow=c(3,3))
                for(k in k:l){
                    #Definition de la longueur de l'axe des ordonnees
                    limInf<-0
                    limSup<-0
                    #Une satisfaction est comprise entre -100 et 100
                    if(isTRUE(grep("satisfaction", names(data[k]),
ignore.case=TRUE)!=0)){
                        col<-grep("satisfaction", names(data[k]),
ignore.case=TRUE)
                        limInf<--100
                        limSup<-100
                    }
                    #Une relation est comprise entre -10 et 10
                    if(isTRUE(grep("state", names(data[k]), ignore.case=TRUE)!=0)){
                        col<-grep("state", names(data[k]), ignore.case=TRUE)
                        limInf<--10
                        limSup<-10
```

```
                        }
                        #Si c'est autre chose, on prend le maximum et le minimum
                        else{
                              limInf<-min(data[,k])
                              limSup<-max(data[,k])
                        }

    boxplot(data[,k]~step1[,(length(step1))],sub=names(data[k]),ylim=c(limInf,limSup),v
arwidth=TRUE,border=c(colors))
                  }
                  k<-l+1
                  l<-l+9
                  if(i==nbdevice-1){
                        l<-ncol(data)
                  }
                  #Sauvegarde
                  if(save==TRUE){
                        dot<-gregexpr("[.]",path,perl=F)
                        pathF<-substr(path,1,max(dot[[1]])-1)
                        path1<-paste(pathF,"-CAHboxPlot",sep="")
                        path2<-paste(path1,i,sep="")
                        newPath<-paste(path2,".png",sep="")
                        dev.print(png, file=newPath, width=800, height=600)
                  }
            }
      }
      else{
            x11()
            par(mfrow=c(nbLig,nbLig))
            for(i in 1:ncol(data)){
                  #Definition de la longueur de l'axe des ordonnees
                  limInf<-0
                  limSup<-0
                  #Une satisfaction est comprise entre -100 et 100
                  if(isTRUE(grep("satisfaction", names(data[i]),
ignore.case=TRUE)!=0)){
                        col<-grep("satisfaction", names(data[i]), ignore.case=TRUE)
                        limInf<--100
                        limSup<-100
                  }
                  #Une relation est comprise entre -10 et 10
                  if(isTRUE(grep("state", names(data[i]), ignore.case=TRUE)!=0)){
                        col<-grep("state", names(data[i]), ignore.case=TRUE)
                        limInf<--10
                        limSup<-10
                  }
                  #Si c'est autre chose, on prend le maximum et le minimum
                  else{
                        limInf<-min(data[,i])
                        limSup<-max(data[,i])
                  }

    boxplot(data[,i]~step1[,(length(step1))],sub=names(data[i]),ylim=c(limInf,limSup),v
arwidth=TRUE,border=c(colors))
            }
            #Sauvegarde
            if(save==TRUE){
                  graphSave(name="-CAHboxPlot.png",vPath=path)
```

```
                }
            }
        }
    }
}

##### QUELQUES TESTS #####

### SELECTION MULTIPLE ###
## Doit fonctionner ##
#CAH(class=c(2,3,4),save=TRUE)
#CAH(class=c(2,3))
## Doit planter ##
#CAH(class=c(2,3),MAV=TRUE)

### SELECTION SIMPLE ###
## Doit fonctionner ##
#CAH(3,save=TRUE,MAV=TRUE)
#CAH(3)
#CAH(3,MAV=TRUE)

### SELECTION MANUELLE ###
## Doit fonctionner ##
#CAH(manSelect=TRUE)
#CAH(manSelect=TRUE,save=TRUE,MAV=TRUE)
#CAH(manSelect=TRUE)
#CAH(manSelect=TRUE,delVar="ChefAtelier.SeuilSatisfaction,OuvProd.SeuilSatisfaction,OuvEn
 t.SeuilSatisfaction")
## Doit planter ##
#CAH(manSelect=TRUE,class=3)
#CAH()
```

h. Fonction SOM()

```
##################################
#Description: Cette fonction permet de réaliser la carte auto-organisatrice (self-
 organizing map) des données
#Parametres: save:permet de sauvegarder les graphiques en sortie
#grid.length permet de définir les dimensions dela grille (par défaut 5x5)
#grid.type permet de définir la forme de la grille. Le type 1 correspond à rectangulaire,
 le type 2 correspond a hexagonal
#delim permet de changer le separateur de colonnes
# Suivi version :
#  V   | Date     | Auteur           | Description des modifications
# -----|----------|------------------|----------------------------------------
#  1.0 | 20120516 | POPIC            | Creation de la premiere version
##################################
library(yasomi)
library(RColorBrewer)

SOM<-function(grid.length=5,grid.type=1,delVar=0,delim="\t",save=FALSE){

 #Recuperation du fichier contenant les donnees avec selection via arborescence
 path<-selectFile()
 data<-read.delim(path,header=TRUE,sep=delim)

 #Suppression des colonnes "run" (inutile), "X" (que des valeurs manquantes), "num_exp"
 data<-delete(data)
```

```
#Traitement d'eventuelles donnees manquantes
data<-na.omit(data)

if(delVar!=0){
     #Creation de la nouvelle matrice de donnees
     #Traitement des variables a supprimer
     delVar1<-paste(",",delVar,",",sep="")
     #On obtient les variables sous cette forme : ",Var1,Var2,...,VarN,"
     virgule<-gregexpr("[,]",delVar1,perl=F)
     nbvirgule<-length(virgule[[1]])
     nbVar<-nbvirgule-1
     Var<-0

     #Extraction des noms des variables
     for(i in 1:nbVar){
          Var[i]<-substr(delVar,(virgule[[1]][i]),(virgule[[1]][i+1])-2)
     }

     #Suppression des variables que l'on souhaite exclure de l'etude
     for(i in 1:nbVar){
          if(isTRUE(grep(Var[i],names(data),ignore.case=TRUE)!=0)){
               col<-grep(Var[i],names(data),ignore.case=TRUE)
               data<-data[,-col]
          }
          else{
          stop ("Le nom de Variable ",Var[i]," n'existe pas")
          }
     }
}

#Centrage et reduction de la matrice de donnees
Sdata<-scale(data)

#Construction de la grille
#Definition du type de grille
if(grid.type==1){
     grid.type="rectangular"
}
else if(grid.type==2){
     grid.type="hex"
}
else{
     stop("Le type de grille ", grid.type, " n'existe pas\n 1: Grille rectangulaire\n 2:
Grille hexagonale ")
}
#Définition de la grille
grid<-somgrid(xdim=grid.length,ydim=grid.length,topo=grid.type)
#Avec nradii, le nombre de configurations testees et criterion, le critere permettant de
dire quel SOM est le meilleur
somT<-
som.tune(Sdata,grid,som.tunecontrol(grid,radii=c(1,grid$diam),nradii=40,criterion=error.
kaskilagus))
som<-somT$best.som

#Construction de la carte representant les effectifs pour chaque neurone (hitmap)
x11()
layout(matrix(c(1,2,2,2,2,2,2,2,2,2,2),ncol=1))
par(mai=rep(0.1,4))
```

```
plot(0,0,type="n",axes=F,xlab="",ylab="")
text(0,0,"Hitmap (effectif des classes)",cex=1.7)
hitMap(som,col="darkred")
#Sauvegarde
if(save==TRUE){
    graphSave(name="-SOMcarteEff.png",vPath=path)
}

#Construction de la carte des distances
pdist<-prototype.distances(som)
pgrid<-distance.grid(pdist)
x11()
filled.contour(pgrid,main="Carte des distances")
#Sauvegarde
if(save==TRUE){
    graphSave(name="-SOMcarteDist.png",vPath=path)
}

#Reconstruction de la grille en explicitant les individus composant chaque classe
x11()
layout(matrix(c(rep(1,grid.length),2:((grid.length^2)+1)),byrow=T,ncol=grid.length))
par(mai=rep(0.1,4))
plot(0,0,type="n",axes=F,xlab="",ylab="")
text(0,0,"Carte des individus",cex=1.7)
#Pour chacune des classes du SOM
for (ind in 1:(grid.length^2)){
    #Si la case n'est pas vide
    if (sum(som$classif==ind)>0){
         #On récupère le numéro des individus
         som.ind<-which(som$classif==ind)
         par(mai=rep(0.1,4))

    plot(1:length(som$classif),1:length(som$classif),type="n",main="",xlab="",ylab="",a
xes=F)
         text(som.ind,som.ind,som.ind-1,lwd=2,col="darkred")
         box("figure",col="black")
    }
    #Si la case est vide
    else{
         plot(0,0,type="n",axes=F,main="",xlab="",ylab="")
         box("figure",col="black")
    }
}
if(save==TRUE){
    graphSave(name="-SOMcarteInd.png",vPath=path)
}

#Construction d'une matrice de graphique contenant la valeur moyennes des variables pour
chacune des classes
nbVar<-length(names(data))
#Creation d'un tableau contenant les moyennes
#on aura tab.mean[individu,variable]
tab.mean<-array(dim=c(grid.length^2,nbVar))
#Creation d'un tableau contenant les ecart-types
#on aura tab.sd[individu,variable]
tab.sd<-array(dim=c(grid.length^2,nbVar))

### Etude de la moyenne ###
```

```
#Pour chacune des variables choisies
for(k in 1:nbVar){
     #Pour chacune des classes du SOM
     for(ind in 1:(grid.length^2)){
          if(sum(som$classif==ind)>0){
               #On recupere le numero des individus
               som.ind<-which(som$classif==ind)
               vect.value<-0
               #Pour chacune des valeurs de la classe
               for(j in 1:length(som.ind)){
                    #On calcule la moyenne des individus pour la variable souhaitee
                    vect.value<-vect.value+som$data[som.ind[j],k]
                    mean.value<-vect.value/(length(som.ind))
                    mean.value<-round(mean.value,2)
               }
               #On incremente le tableau contenant les moyennes
               tab.mean[ind,k]<-mean.value
          }
     }
}
#Construction de la grille
coloring<-brewer.pal(9,"RdYlBu")[9:1]
nbLig<-ceiling(sqrt(ncol(data)))
#Si l'on a plus de 4*4 graphiques par device
if(nbLig>4){
     nbdevice<-ceiling(ncol(data)/16)
     l<-16
     k<-1
     for(i in 1:nbdevice){
          x11()
          par(mai=rep(0.1,4))
          layout(matrix(c(rep(1,4),2:(4^2+1)),ncol=4,byrow=T))
          plot(0,0,type="l",axes=F,xlab="",ylab="")
          text(0,0,"Etude des moyennes",cex=1.7)
          for(k in k:l){
               bornes<-
cut(tab.mean[,k],breaks=seq(range(tab.mean[,k],na.rm=TRUE)[1],range(tab.mean[,k],na.rm=T
RUE)[2],length=8),include.lowest=TRUE)
               plot(grid,col=coloring[bornes],main=names(data[k]))
          }
          k<-l+1
          l<-l+16
          if(i==nbdevice-1){
               l<-ncol(data)
          }
          #Sauvegarde
          if(save==TRUE){
               dot<-gregexpr("[.]",path,perl=F)
               pathF<-substr(path,1,max(dot[[1]])-1)
               path1<-paste(pathF,"-SOMcarteNivMoy",sep="")
               path2<-paste(path1,i,sep="")
               newPath<-paste(path2,".png",sep="")
               dev.print(png, file=newPath, width=800, height=600)
          }
     }
}
else{
     x11()
```

```
     par(mai=rep(0.1,4))
     layout(matrix(c(rep(1,nbLig),2:(nbLig^2+1)),ncol=nbLig,byrow=T))
     plot(0,0,type="l",axes=F,xlab="",ylab="")
     text(0,0,"Etude des moyennes",cex=1.7)
     for(k in 1:nbVar){
          bornes<-
cut(tab.mean[,k],breaks=seq(range(tab.mean[,k],na.rm=TRUE)[1],range(tab.mean[,k],na.rm=T
RUE)[2],length=8),include.lowest=TRUE)
          plot(grid,col=coloring[bornes],main=names(data[k]))
     }
     #Sauvegarde
     if(save==TRUE){
          graphSave(name="-SOMcarteNivMoy.png",vPath=path)
     }
}
### Etude de l'ecart-type ###
#Pour chacune des variables choisies
for(k in 1:nbVar){
     #Pour chacune des classes du SOM
     for(ind in 1:(grid.length^2)){
          if(sum(som$classif==ind)>0){
               #On recupere le numero des individus
               som.ind<-which(som$classif==ind)
               vect.value<-0
               #Pour chacune des valeurs de la classe
               for(j in 1:length(som.ind)){
                    #On recupere les valeurs de chacun des individus
                    vect.value[j]<-som$data[som.ind[j],k]
               }
               #on calcule l'ecart-type des variales constituant la classe
               sd.value<-sd(vect.value)
               tab.sd[ind,k]<-round(sd.value,2)
          }
     }
}
#Construction de la grille
coloring<-brewer.pal(9,"RdYlBu")[9:1]
nbLig<-ceiling(sqrt(ncol(data)))
#Si l'on a plus de 4*4 graphiques par device
if(nbLig>4){
     nbdevice<-ceiling(ncol(data)/16)
     l<-16
     k<-1
     for(i in 1:nbdevice){
          x11()
          par(mai=rep(0.1,4))
          layout(matrix(c(rep(1,4),2:(4^2+1)),ncol=4,byrow=T))
          plot(0,0,type="l",axes=F,xlab="",ylab="")
          text(0,0,"Etude des écarts types",cex=1.7)
          for(k in k:l){
               bornes<-
cut(tab.sd[,k],breaks=seq(range(tab.sd[,k],na.rm=TRUE)[1],range(tab.sd[,k],na.rm=TRUE)[2
],length=8),include.lowest=TRUE)
               plot(grid,col=coloring[bornes],main=names(data[k]))
          }
          k<-l+1
          l<-l+16
          if(i==nbdevice-1){
```

```
                l<-ncol(data)
          }
          #Sauvegarde
          if(save==TRUE){
                dot<-gregexpr("[.]",path,perl=F)
                pathF<-substr(path,1,max(dot[[1]])-1)
                path1<-paste(pathF,"-SOMcarteNivSD",sep="")
                path2<-paste(path1,i,sep="")
                newPath<-paste(path2,".png",sep="")
                dev.print(png, file=newPath, width=800, height=600)
          }
     }
 }
 else{
     x11()
     par(mai=rep(0.1,4))
     layout(matrix(c(rep(1,nbLig),2:(nbLig^2+1)),ncol=nbLig,byrow=T))
     plot(0,0,type="l",axes=F,xlab="",ylab="")
     text(0,0,"Etude des écarts types",cex=1.7)
     for(k in 1:nbVar){
          bornes<-
 cut(tab.sd[,k],breaks=seq(range(tab.sd[,k],na.rm=TRUE)[1],range(tab.sd[,k],na.rm=TRUE)[2
 ],length=8),include.lowest=TRUE)
          plot(grid,col=coloring[bornes],main=names(data[k]))
     }
     #Sauvegarde
     if(save==TRUE){
          graphSave(name="-SOMcarteNivSD.png",vPath=path)
     }
 }
}

##### QUELQUES TESTS #####

## Doit fonctionner ##
#SOM(grid.type=2,grid.length=9)
#SOM(delVar="ChefAtelier.SeuilSatisfaction,OuvProd.SeuilSatisfaction,OuvEnt.SeuilSatisfac
 tion")
## Doit planter ##
#SOM(grid.type=3)
```

XI. Bibliographie

1. Références

- EL GEMAYEL J., CHAPRON P., ADREIT F., et SIBERTIN-BLANC C., « Quand et comment les acteurs sociaux peuvent-ils coopérer ? Un algorithme de simulation pour la négociation de leurs comportement » in *Revue d'Intelligence Artificielle,* 2011.
- EL GEMAYEL J., CHAPRON P., ADREIT F., et SIBERTIN-BLANC C., « Impact of Tenacity upon the Behaviors of Social Actors », 2011.
- EL GOLLI A., ROSSI F., CONAN-GUEZ B., et LECHEVALLIER Y., « Une adaptation des cartes auto-organisatrices pour des données décrites par un tableau de dissimilarités », in *Revue de statistique appliquée*, 2006.
- ADREIT F., MAILLARD M., ROGGERO P., SIBERTIN-BLANC C., et VAUTIER C., « Formalisation de la SAO : modélisation du cas Bolet », 2007.

2. Sites

- BOUCHIER A., « L'analyse des données multivariées à l'aide du logiciel R » [PDF]. 2012. http://rstat.ouvaton.org/?article10/classification-hierarchique
- CARPENTIER F.G., « MDS classique avec R » [en ligne]. 25-11-11. http://geai.univ-brest.fr/~carpenti/2006-2007/Documents-R/MDS-simple.html
- CARPENTIER F.G., « Classification ascendante hiérarchique avec R» [en ligne]. 28-05-07. http://geai.univ-brest.fr/~carpenti/2006-2007/Documents-R/CAH-avec-R.html
- FEW S., « Visual Business Intelligence for enlightening analysis and communication » [Site Officiel]. 2012. http://www.perceptualedge.com (tapez Practical Rules for Using Color in Charts dans la barre de recherche pour lire l'article cité dans le mémoire)
- GNU, « The compréhensive R archive network » [Site Officiel]. 2012. http://cran.r-project.org/
- IRIT, « SocLab, social laboratory » [en ligne]. 2012. http://www.irit.fr/socLab,1071
- KAUFFMANN M., « Aide mémoire R. Référence des fonctions de R les plus courantes» [PDF]. 2009. cran.r-project.org/doc/contrib/Kauffmann_aide_memoire_R.pdf`
- ROSSI F., « Package Yasomi » [en ligne]. https://r-forge.r-project.org/projects/yasomi/

3. Ouvrages

- BAREL Y., *Le paradoxe et le système, Essai sur l'imaginaire social,* Grenoble, PUG, 1979.
- CROZIER, M. *Le Phénomène Bureaucratique*, Le Seuil, Paris, France, collection points et essais édition, 1964.
- CROZIER M., FRIEDBERG E., *L'acteur et le système*. *Contraintes de l'action collective*, Paris, Seuil, 1977.