Harmonisation de données statistiques et algorithme de recommandation

Benjamin Clamme

Université de Strasbourg

Rapport de stage

InTech S.A.

du 03 février au 31 juillet 2014

Auteur :
Benjamin Clamme

Entreprise :
InTech S.A.

31 juillet 2014

Table des matières

1 Remerciements

Je tiens tout d'abord à remercier François Bocci, Michel Sanitas et Alain Pucar directeurs d'InTech Group pour m'avoir donné l'opportunité d'effectuer mon stage de fin d'étude dans leur entreprise. Je souhaite aussi remercier Nathalie Milair, Fabrice Croiseaux et Sébastien Larose, directeurs opérationnels d'InTech S.A., ainsi que Thierry Kruten, directeur du LIS, pour la confiance qu'ils m'ont accordés.

Un grand remerciement à Florian Collignon et Antoine Detante qui ont pu m'apporter leur aide et un soutien technique tout au long de mes deux projets.

Je remercie aussi Nicolas Sanitas, que j'ai rencontré au forum "Stage en poche" de Metz qui m'a permis de découvrir la société InTech et les opportunités de stage qu'elle propose.

Je remercie enfin tous les collaborateurs et stagiaires de l'entreprise dont la bonne humeur m'a rendu ce stage que plus plaisant.

2 Introduction

2.1 Un fil conducteur...

Mon stage au sein d'InTech a été divisé en deux parties : j'ai tout d'abord travaillé chez un client de la société, le LIS puis à l'agence pour un projet en interne. Mes deux sujets, bien que différents, présentent un dénominateur commun : la gestion d'un volume important de données dont on souhaite extraire l'information essentielle afin de produire des indicateurs simples et compréhensibles par l'utilisateur.

Cette problématique est à la fois typique des statistiques -qui permettent de résumer des tableaux de données volumineux en une série d'indicateurs et de représentations graphiques plus intelligibles- mais aussi de l'informatique. En effet, l'usage d'internet a induit un flux de données rapide et volumineux et dont l'approche dans sa globalité est, sans outils spécifiques, tout à fait impossible. Ceci est particulièrement vrai pour les réseaux sociaux (Facebook, Twitter...) où l'utilisateur est confronté à un nombre incalculable de données émises par de nombreux utilisateurs.

Le rôle du statisticien est ici de guider l'utilisateur à travers ce flux et de ne lui présenter qu'un contenu adapter à ces préférences et à ses goûts, tout en prenant soin de s'adapter à l'évolution de son comportement d'internaute.

2.2 Déroulement du stage

InTech S.A., l'entreprise qui m'accueille pour mon stage, est une société de service informatique. A ce titre, elle répond à des offres de clients qui souhaitent intégrer des solutions IT dans leur chaîne de production.

Afin de mieux répondre aux besoins de l'entreprise, mon stage a été découpé en 2 parties. Dans un premier temps, je vais travailler 3 mois pour LIS (un centre de recherche et d'archives de données luxembourgeois, voir présentation ci-dessous) qui est un client d'InTech, puis durant les 3 mois restants, me consacrer à un projet interne au siège d'InTech.

3 Présentation des entreprises

3.1 Présentation d'InTech

Fondée en 1995 au Luxembourg, InTech compte aujourd'hui 78 collaborateurs spécialisés dans la conception et la réalisation de solutions métiers construites à partir de composants industriels fondés sur les Nouvelles Technologies de l'Information.

InTech est positionnée sur le segment des Grands Comptes et opère principalement dans les secteurs des services financiers, de l'industrie, des administrations luxembourgeoises, des services et de la santé. L'entreprise travaille essentiellement en mode projet, en associant des compétences pluridisciplinaires de management de projet, de conseil en architectures informatiques, d'expertise technique et de développement.

3.2 Présentation de LIS

Le "Luxembourg Income Study" (LIS), organisation à but non-lucratif, a pour but de collecter puis d'harmoniser des données socio-économiques d'enquêtes provenant d'une cinquantaine de pays. Les données harmonisées sont ensuite mises à disposition de chercheurs et permettent principalement de mener à bien des études comparatives internationales dans le domaine des inégalités de revenu au sein d'un seul ou d'un ensemble de pays.

4 Projet Metadata

4.1 Objectifs et enjeux

Les données traitées par LIS parviennent d'enquêtes effectuées par les organismes statistiques propres à chaque pays (par exemple en France : l'INSEE). Cependant ce ne sont pas ces données brutes qui sont directement accessibles aux utilisateurs finaux. Ces derniers ont uniquement accès :

1. A une interface, leur permettant d'effectuer des requêtes sur la base de données (cette partie des outils n'a pas été utilisée dans le stage).
2. A un ensemble d'informations descriptives (codebooks, matrices de disponibilité des variables ("variable availability matrix") et à des indicateurs ("Inequality Key Figures")

Actuellement, les informations descriptives comprennent un agrégat de fichiers texte (les codebooks) de classeur Excel (les matrices de disponibilité) et une application en ligne (pour les Inequality Key Figures). Afin de structurer l'ensemble de ces informations, le LIS a fait appel à InTech pour développer une application web, appelée Metadata, directement interrogeable par les utilisateurs et qui regroupe l'ensemble des données disponibles, citées ci-dessus.

En relation avec le développement de cette application, mon travail a consisté à :

1. Générer les matrices de disponibilité à partir des données harmonisées
2. Extraire les différentes modalités (code et value-labels) des variables qualitatives
3. Calculer des statistiques descriptives pour les variables présentes dans les données harmonisées
4. Calculer les "Inequality Key Figures" à partir des données harmonisées

4.2 Cahier des charges

4.2.1 L'application "Metadata"

L'application Metadata permet aux utilisateurs finaux de prendre connaissance rapidement de l'ensemble des données disponibles aux seins des deux databases de LIS : "LIS" et "LWS". La version finale de cette application doit permettre aux utilisateurs d'avoir accès, pour l'ensemble des Dataset existants,

à la liste des variables effectivement présentes dans le dataset, à leur modalités (pour les qualitatives), à des statistiques sur ces variables et aux Inequality Key Figures (ainsi qu'à d'autres données qui ne sont pas en relation avec mon travail). L'application Metadata stocke ces informations dans une base de donnée SQL cependant, elle n'a pas accès directement aux micro-données. Mon travail consiste donc à créer une passerelle entre ces données brutes et Metadata (ou en terme de stockage de l'information, entre les fichiers stata et la base SQL) via des fichiers textes.

4.2.2 Organisation des données

Afin de mieux situer les termes techniques utilisés, voici une liste expliquant l'organisation des données de LIS :

- Niveau 1 : Base de données (LIS/LWS)
  - Niveau 2 : Dataset
    - Niveau 3 : Datafile
      - Niveau 4 : Variable

Du point de vue du travail à effectuer, un datafile n'est rien d'autre qu'un fichier stata, qui englobe les micro-données.

4.2.3 Travail à effectuer

Le travail consiste à développer un programme permettant de :

1. Lire les fichiers stata contenant les données harmonisées
2. En extraire :
   (a) Les matrices de disponibilité des variables ("variable availability matrix"-VAM)
   (b) Les différents libellés et leurs codes pour les variables qualitatives (VL)
   (c) Des statistiques descriptives (STATS)
   (d) Les "Inequality Key Figures" (IKF)
3. Ecrire les résultats dans un fichier texte avec un encodage spécifique

Pour cela, les logiciels suivants ont été utilisés :

- R pour la programmation en générale
- stata pour l'accès aux données harmonisées
- HeidiSQL pour interagir avec Metadata

Chaque tâche doit être effectuée par un pull-program différent. Le programme final sera donc composé de quatre pull-programs indépendants. Par ailleurs l'utilisateur doit disposer d'une totale liberté quant à leur utilisation, c'est-à-dire lancer aussi bien un seul pull-program sur un dataset ou à l'inverse tous les pull-programs sur tous les datasets.

Par ailleurs, les résultats doivent être stockés dans un ou plusieurs fichiers textes avec le symbole |$| comme séparateur de données.

Chaque pull-program doit aussi répondre à un cahier des charges spécifique détaillé ci-dessous.

4.2.4 VAM

Pour un dataset donné, la Variable Availability Matrix indique, pour chacune des variables traitées par le LIS, si elle est effectivement présente dans le dataset (i.e. s'il y a au moins une observation différente de N.A.).

Les résultats doivent être stockés dans un fichier texte contenant, pour un dataset :

- Une ligne de référencement comprenant les identificateurs des variables
- Une ligne de résultat, commençant par l'identificateur du dataset suivi de la disponibilité des variables

La disponibilité d'une variable suit un codage binaire (0=absence, 1=présence).

4.2.5 VL

Ce pull-program concerne uniquement les variables qualitatives. On cherche à extraire, pour chaque modalité, ses différents "value-labels" et le code correspondant.

Les variables "non country-specific" possèdent des value-labels et codes standardisés déjà stockés dans la base Metadata. Il est donc nécessaire de lire les micro-données uniquement pour les variables "country specific".

Les résultats doivent ensuite être écrits dans un fichier texte contenant, pour un dataset :

- Une ligne par variable comprenant dans l'ordre :
  - l'identificateur de la variable
  - un code de "value-label"
  - le "value label" correspondant

Ainsi, une variable comprenant 4 modalités aura une ligne comprenant 9 éléments.

4.2.6 STATS

Les statistiques descriptives produites dépendent du type de variables.

Pour les variables quantitatives, on cherche à calculer :

1. Le nombre d'observations (référence 1)
2. Le minimum des observations (référence 2)
3. Le maximum des observations (référence 3)
4. La moyenne sur l'ensemble des observations (référence 4)
5. La moyenne pour les observations non-nuls (référence 5)
6. L'écart-type sur l'ensemble des observations (référence 6)
7. L'écart-type pour les observations non-nuls (référence 7)

Pour les variables qualitatives, on cherche à calculer :

- le nombre d'observations pour chaque modalité (référence 8)

Les résultats doivent ensuite être stockés dans un fichier texte contenant, pour un dataset et pour chaque variable :

- Dans le cas d'une variable quantitative :
  - Une ligne contenant les références des statistiques
  - Une ligne contenant l'identificateur de la variable et les valeurs des statistiques
- Dans le cas d'une variable qualitative
  - Une ligne contenant la référence de la statistique
  - Une ligne par modalité comprenant, l'identificateur de la variable, le code de la modalité et la valeur de la statistique

4.2.7 IKF

Les Inequality Key Figures, au nombre de 23, permettent de se faire une idée sur les différences de niveau de vie au sein d'un pays. Elles sont calculées à partir de plusieurs variables présentes dans les deux datafiles (household et person).

Les résultats doivent être stockés dans un fichier texte contenant, pour un dataset :

- Un en-tête reprenant les numéros des IKF
- Une ligne comprenant l'identificateur du dataset et les résultats des calculs

4.3 Travail effectué

4.3.1 Création des programmes

Structure du programme final

Le programme final se présente sous la forme d'un dossier nommé "pull" à l'arborescence suivante :

- Dossier "config"
  - fichier "conf.txt"
- Dossier "docs"
  - fichier "user_guide.doc"
  - fichier "tech_guide.doc"
- Dossier "Tech"
  - fichier "tools_function.r"
  - fichier "ikf_functions.r"
  - fichier "core_functions.r"
- Dossier "Usr"
  - Dossier "Log"
  - Dossier "Out"
  - fichier "guideline.csv"
  - fichier "main.r"

Fonctionnement du programme

Le programme s'appuie sur les 3 scripts R qui se trouvent dans le dossier pull/Tech. Le script core_functions.r est le coeur du programme. Il est divisé en 4 functions (VAM, VL, STATS et IKF) qui effectuent chacune une tâche spécifique. Ces fonctions s'appuient sur d'autres fonctions "outils" qui se trouvent dans le script "tool_functions.r". Le script "ikf_functions.r" regroupe 23 fonctions qui contiennent les formules mathématiques permettant de calculer les IKF.

Par ailleurs, le programme doit avoir accès au micro-données (les fichiers stata) ainsi qu'à une copie de la base de donnée SQL de Metadata, ce qui suppose l'utilisation des packages "foreign" et "RODBC". L'accès à une base de donnée "Metadata-like" permet de retrouver les identificateurs des datasets, des variables et de connaître la structure des données.

Le script "main.r" constitue l'interface entre l'utilisateur et le programme. C'est ce script qui doit être chargé sur R après avoir été configuré. En effet, il faut préciser à l'intérieur du script où est stocké le dossier "pull" (ce chemin est appelé le root). Le script contient une seule fonction, appelée main, qui doit être exécutée pour lancer le programme. Cette fonction s'appuie sur deux fichiers :

"config.txt" et "guideline.csv". Le premier permet à l'utilisateur de configurer les chemins d'accès vers les micro-données (appelé stataPath) et de préciser le nom de la base "Metadata-like" utilisée (la user_DB). Le second permet à l'utilisateur de préciser quelles tâches il souhaite effectuer et sur quels datasets.

Ci-dessous, un exemple de guideline :

	A	B	C	D	E	F
1	Database	Dataset	vam	stats	vl	ind
2	LIS	IL79	0	0	0	0
3	LIS	DE81	0	0	0	0
4	LIS	UK79	0	0	0	0
5	LIS	NO79	0	0	0	0
6	LIS	CA81	0	0	0	0

La première colonne correspond au nom de la database, la deuxième colonne au CCYY du dataset. Les quatre colonnes suivantes correspondent aux pull-programs (de VAM à IKF) avec un codage binaire (1=lancer le pull-program sur le dataset).

S'il est correctement configuré, l'exécution du programme se déroule ensuite en trois temps :

1. lecture du guideline
2. lecture des fichiers stata
3. execution des pull-programs

Lecture du guideline L'utilisateur possède en fait deux mode d'exécution des pull-programs : totale ou partielle. Pour une exécution totale, c'est-à-dire sur tous les dataset d'une database, il lui suffit de renseigner "all", plutôt que le nom du dataset dans la deuxième colonne. Le programme se charge alors de vérifier quels sont les datasets disponibles en se connectant à la base Metadata-like. A noter que l'utilisateur peut toujours choisir quels pull-programs il souhaite jouer. Une exécution partielle correspond à l'exemple de guideline précédant. Il s'agit alors d'une lecture classique de fichier csv avec stockage des informations dans une matrice R.

Chaque ligne de cette matrice permet ensuite de générer un objet de type liste, appelé "ref" qui contient :

1. Le nom de la database
2. Le nom du dataset
3. L'identificateur du dataset
4. Des informations techniques corolaires

Il permet ensuite de guider les pull-programs. Les informations qu'il contient sont actualisées pour chaque ligne du guideline (pour chaque dataset).

Lecture des fichiers stata Les fichiers stata peuvent être lu de deux manières. En effet, les modalités des variables qualitatives peuvent soit être déclarées en tant que facteur soit en tant qu'integer. La première option permet d'accéder à la fois au code et au value label. La seconde renvoi uniquement le code. Pour des raisons techniques certains pull-program ne peuvent être lancé qu'avec des modalités sous forme d'integer (IKF notamment). A contrario, on ne peut évidemment pas se passer des value labels pour VL.

Pour éviter de lire inutilement les fichiers stata, le programme vérifie dans le guideline si, pour chaque dataset :

1. Le pull-programs VL doit être joué
2. Un des autres pull-programs doit être joué

Si 1. est vrai le fichier sera lu avec les modalités en facteur. Si 2. est vrai, il sera lu avec des integer. Il peut donc il y avoir soit une, soit deux lectures du fichier stata par dataset.

C'est la commande read.dta qui permet de lire un fichier stata. L'option convert.factor=TRUE/FALSE permet de contrôler si les modalités doivent être sous forme de factors ou non (TRUE=oui).

Un fichier stata correspondant à un datafile, il y a donc deux fichiers stata à lire par dataset. Les résultats de cette lecture sont stockés sous R sous forme d'une liste contenant deux dataframe.

Exemple de code :

```
>stata<-list()
>stata[[1]]<-read.dta(paste(stataPath,"CCYYih.dta",sep=""),convert.factor=TRUE)
>stata[[2]]<-read.dta(paste(stataPath,"CCYYip.dta",sep=""),convert.factor=TRUE)
```

"CCYYih" correspond au fichier "household" du dataset CCYY (ex FR04ih), et "CCYYip" au fichier "person".

Exécution des pull-programs Il existe deux types de pull-program vis-à-vis de la structure :

1. Les programmes à datafiles séparés, c'est-à-dire qui traitent indépendamment les deux datafiles d'un dataset.
2. Les programmes à datafiles recoupés, qui utilisent des informations provenant des deux datafiles pour le calcul des résultats.

IKF est le seul programme à "datafiles recoupés" et il est donc structuré de manière unique. Les 3 autres pull-programs possèdent une structure identique.

Précision sur le pull-program STATS Le pull program STATS est en fait la concaténation de deux fonctions. La première opère sur les variables quantitatives, la seconde sur les variables qualitatives. Ces deux fonctions écrivent tout de même leurs résultats dans un même fichier texte et prennent leurs instructions de la même colonne du guideline. L'utilisateur a donc l'illusion d'un seul pull-program (et il n'a donc pas la possibilité de les exécuter séparément).

Pour clarifier les choses, on fera désormais référence à STATS0 comme étant la fonction opérant sur les variables quantitatives et à STATS1 celle sur les variables qualitatives.

Structure des pull-programs à Datafiles séparés

L'éxécution de ces pull-programs passe invariablement les étapes suivantes :

1. Lecture des arguments
2. Création des objets intermédiaires
   (a) Travail sur le datafile "household"
      i. Connection à la "user_DB", récupération des informations d'identification
      ii. Extraction des résultats des micro-données
   (b) Travail sur le datafile "person" (idem que précédemment)
3. Concaténation des résultats pour les deux datafiles
4. Ecriture des résultats

Arguments des fonctions Les arguments utilisés sont :

- La liste d'informations "ref"
- La liste "stata" contenant les micro-données
- Le "root"
- La "user_DB"

Ces objets sont automatiquement renseignés par l'intermédiaire de la fonction main.

Création des objets intermédiaires Il s'agit de deux listes qui permettront de stocker les sorties de la user_DB (une par datafile) et les résultats des micro-données (issus des deux fichiers stata).

Connection à la "user_DB" Il s'agit d'extraire :

- Les noms des variables
- Leurs identificateurs

Avec un ciblage (dans la requête SQL) qui dépend du pull program

- Pour VAM : on cible l'ensemble des variables
- Pour VL : on se concentre sur les variables qualitatives, country-specific
- Pour STATS0 : on se concentre sur les variables quantitatives
- Pour STATS1 : on se concentre sur les variables qualitatives

A noter que dans tous les cas, on procède datafile par datafile (ce qui suppose un ciblage).

Extraction des résultats Il s'agit simplement de calculer les résultats attendus.

Concaténation et écriture Les résultats sont ensuite concaténés aux extractions de la "user_DB". Cela permet uniquement de récupérer les identificateurs des variables. En effet, dans les micro-données, on ne possède que les noms des variables. On fait donc une concaténation par nom.

Les résultats sont ensuite écrits selon les modalités inscrits dans le cahier des charges.

Un exemple de sortie Voici un aperçu des résultats du pull program STATS :

DATASET|$|VARIABLE|$|VALUE_LABEL_CODE|$|1|$|2|$|3|$|4|$|5|$|6|$|7

230|$|1|$||$|26745|$|1|$|26745|$|13373|$|13373|$|7720.76|$|7720.76

DATASET|$|VARIABLE|$|VALUE_LABEL_CODE|$|8

230|$|18|$|110|$|18048

230|$|18|$|120|$|3149

230|$|18|$|200|$|636

230|$|18|$|210|$|4834

230|$|18|$|NA|$|78

La première ligne correspond à l'en-tête pour une variable quantitative. La seconde ligne est la sortie pour la variable 1 du dataset 230. Elle ne possède pas de VALUE_LABEL_CODE (étant quantitative). La troisième ligne correspond à l'en-tête pour une variable qualitative. Les lignes suivantes sont les sorties

pour la variable 18 du dataset 23O. Chaque ligne correspond à une modalité différente, identifiée par son code.

Structure du programme IKF

Le programme IKF se concentre uniquement sur quelques variables qui sont principalement :

- Des variables de revenus
- Des variables de composition des foyers

Ces variables se trouvent à la fois dans le datafile "household" (pour la plupart) et dans le datafile "person" (pour le sexe des individus par exemple). Il faut donc, avant de lancer les calculs, extraire les variables attendues et les stocker dans un même objet. Cette étape est assurée par une "tool_function" qui renvoie ensuite un data-frame (observations x variables) à la "core_function" IKF.

Le calcul des 23 IKF à proprement parlé est assuré par les 23 functions sous-fonctions IKF1,...,IKF2 qui se situent dans le script "ikf_functions.r". Ces fonctions ont été créées indépendantes afin de faciliter leur éventuelle modification ultérieure (en cas de changement de méthode de calcul par exemple). Les résultats bruts sont couplés avec un codage qui indique grossièrement le nombre d'observations utilisées pour les obtenir. Il s'agit de :

- '***' pour une absence de résultat (0 observations ou impossibilité de calcul)
- '**' pour un nombre d'observations inférieure à 15.
- '*' si le nombre d'observations est compris entre 15 et 30.

Pour plus de 30 observations, il n'y a pas de codage spécifique. Les résultats codés sont ensuite écrits dans un fichier texte défini dans le cahier des charges. Ci-dessous, un exemple de sortie partielle du pull-program :

DATASET|$|1|$|2|$|3|$|4|$|5|$|6| 403|$|0.25|$|0.056|$|***|$|**|$|1.665|$|1.997*|

On observe ici les 6 premiers IKF pour le dataset 403. On remarque que l'IKF 3 n'est pas calculable. L'IKF 4 est calculé mais sur moins de 15 observations, il n'est donc pas affiché car peu fiable. L'IKF 6 est calculé avec peu d'observations (entre 15 et 30), il est donc suivi d'une étoile.

4.3.2 Création d'un log

Afin de faciliter le suivi de l'exécution des pull-programs, on a choisi de créer un journal (log) qui contient :

- des messages standards confirmant la bonne exécution des tâches

- une redirection des messages d'erreurs générés par R

Pour rediriger les messages d'erreur de R, il faut créer un "sink", le principe étant de lier la session R à un fichier texte. La procédure est la suivante :

```
>error_log<-file("chemin_du_fichier",open='a')
>sink(file=error_log,append=T,type="message")
```

Le fichier est ouvert avec l'option "a" pour éviter que l'écriture d'un message efface les entrées précédentes. On redirige ensuite les uniquement les sorties de type "message", avec l'option append mise à TRUE pour éviter d'effacer le log d'une session sur l'autre.

4.3.3 Tests de conformité

La programmation terminée, il s'agit de vérifier que les sorties des programmes sont conformes au cahier des charges. Cette étape de vérification a été conduite de manière quasi-automatique. En effet, les données produites par les pull-programs sont destinées à être uploadées sur la base de données Metadata, par l'intermédiaire d'un parser. Les fichiers textes non conformes sont alors automatiquement rejetés ce qui a permis de remonter quelques erreurs de programmation.

L'étape de test unitaire a été considérée comme achevée quand la plupart fichiers de résultat (pour de nombreux dataset) ont pu être "parsés".

4.3.4 Tests fonctionnels

Les tests fonctionnels consistent à prendre les 4 sorties pour quelques datasets et à vérifier la conformité des résultats (du fond et non de la forme) vis à vis des résultats existants. Le plan de test est le suivant :

- Comparaison des sorties de VAM avec les availability matrix disponibles sur le site du LIS (sous forme de classeur Excel) pour 4 datasets
- Comparaison des sorties de VL avec les codebooks disponibles sur le site du LIS (sous forme de fichier PDF) sur 2 datasets
- Comparaison des sorties de STATS avec les codebooks disponibles sur le site du LIS (sous forme de fichier PDF) sur 3 datasets
- Comparaison des sorties de IKF avec les données disponibles sur le site du LIS (via l'interface de calcul intégrée) sur 5 datasets en maximisant la couverture temporelle

A ces vérifications de base s'ajoutent quelques complexités. En effet, dans les codebooks, les statistiques descriptives disponibles pour chaque variable sont :

- Le nombre d'occurrence
- Le minimum

- Le maximum
- La moyenne
- L'écart type

Ainsi, la moyenne et l'écart type zéros exclus doivent être recalculés directement à partir des micro-données. Pour cela, on utilise directement stata.

Ces vérifications terminées, les programmes sont considérés fonctionnels et leurs résultats peuvent être définitivement chargés sur la base Metadata.

5 L'Algorithme de recommandation

5.1 Un intérêt commercial...

De manière générale, un algorithme de recommandation permet de proposer à un utilisateur un contenu adapté à ces goûts ou à son activité (les deux étant souvent liés). En termes d'objectifs commerciaux, l'algorithme permet d'augmenter le taux de transformation, i.e. de maximiser les ventes après présentation du produit. Dans le cadre de Twitter, aucun produit n'est à vendre. Cependant, il existe un intérêt commercial sous-jacent. Pour une entreprise comme InTech, axé sur l'IT, la présence sur les réseaux sociaux est perçue comme une évidence. En effet, le site est une source potentielle de clients et de prospects dont la détection manuelle peut s'avérer fastidieuse. Ainsi, en se basant sur les relations existantes entre InTech et ses clients, un algorithme pourrait fournir de précieuses indications sur de nouvelles personnes ou entités éventuellement intéressées par la gamme de services proposée par la société.

5.2 Théorie

Dans le cadre d'un service numérique, un algorithme de recommandation permet à un prestataire (organisme de vente, de streaming vidéo etc...) de proposer à ses clients des produits adaptés à leurs goûts ou à leurs besoins (et ce afin de maximiser son taux de vente).

La problématique adressée se place donc (la plupart du temps) dans un cadre utilisateur-objet. La recommandation peut s'appuyer soit sur les interactions entre les utilisateurs (en partant du principe qu'un utilisateur au sein d'un groupe tendra à partager les goûts les plus prononcés du groupe) soit sur les similarités entre les objets.

La première méthode -sociale- est appelée "Collaborative filtering". Ce sont ses principes de base que nous avons tenté d'adapter à notre problème et c'est donc cette méthode que l'on se propose de présenter.

Sa mise en place se divise en trois grandes étapes :

1. Caractérisation des utilisateurs selon leurs comportements
2. Mesure de la similarité entre les utilisateurs
3. Recommandation d'objets en fonction des relations inter-utilisateurs

5.2.1 La matrice d'utilité

La méthode de filtrage collaboratif s'appuie sur les relations entre utilisateurs du point de vue de leurs comportements vis-à-vis des objets.

L'idée est d'établir une "Utility Matrix" qui regroupe une forme d'appréciation des objets par les utilisateurs. Si on prend l'exemple d'un site de streaming vidéo. On peut imaginer que chaque utilisateur est amené à noter les films/séries (ou autre) qu'il a visionné. La matrice d'utilité sera alors le regroupement des n vecteurs de notation (on a n utilisateurs) pour une liste potentiellement très importante de films (tous les films disponibles sur le site...).

Cette forme de notation demande la participation active de l'utilisateur (il doit faire la démarche de noter la vidéo et dans la plupart des cas, il ne le fera probablement pas) mais on peut très bien imaginer une notation implicite, binaire, qui vaut 1 si l'utilisateur a visionné le contenu.

Voici un exemple de matrice d'utilité adaptée à la recommendation de contenu vidéo :

	Film 1	Film 2	Film 3	Film 4	Film 5	Film 6
Pierre	5	1		4		2
Sophie	4	2				1
Patrick	1	5	3			
Nicolas				3	3	5
Marie	2	2		5		

Les notes vont de 1 à 5.

Dans cet exemple on voit que Pierre et Marie ont des avis similaires sur les films 1, 2 et 6. En se basant sur cette similarité, on peut conseiller à Marie le film 4, qu'elle n'a pas noté mais que Pierre a bien noté.

En pratique, la matrice est beaucoup plus vaste et contient majoritairement des blancs, c'est-à-dire une absence de notation, rendant la détection d'utilisateurs similaires d'autant plus difficile.

5.2.2 Une mesure de similarité

La matrice d'utilité établie, il s'agit maintenant d'établir une mesure de similarité entre les lignes (les utilisateurs) de la matrice. Le choix de cette mesure dépend principalement des données. Pour des données binaires, la mesure de Jaccard peut être adaptée. Pour des données plus "complexes" (comme les notations) on privilégiera le coefficient de similarité de Pearson.

On rappelle ici sa définition :

$$\rho(x,y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2 \sum_{i=1}^{n}(y_i-\bar{y})^2}}$$

A partir de cette mesure, on est en mesure d'identifier un groupe d'utilisateurs proche de notre utilisateur cible (ce sont ceux dont la similarité est la plus élevée). On utilise ensuite leurs lignes de la matrice d'utilité pour le processus de recommandation.

5.2.3 Recommendation...

Reprenons l'exemple de la recommandation de contenus vidéo. On a cette fois, isolé les utilisateurs les plus proches de Pierre :

	Film 1	Film 2	Film 3	Film 4	Film 5	Film 6
Pierre	5	1		4		2
Jean	5	2	4			1
Charlotte	4		3	3		2
Léa	2				5	1

Ici les données n'étant pas binaires, on propose le protocole de recommandation suivant :

1. On fixe une notation seuil, au delà de laquelle un objet est considérée comme recommandable
2. On calcule des notes prédictives pour les objets non notés par l'utilisateur cible (ici Pierre)
3. On recommande les objets dont la note prédictive dépasse le seuil

N.B : Il faut bien garder à l'esprit que ce tableau ne représente que les utilisateurs proches de Pierre et non des autres utilisateurs. En général, deux utilisateurs même très proches ne possèdent pas strictement le même voisinage proche.

On a compléter le tableau précédant avec les notes prédictives pour Pierre :

	Film 1	Film 2	Film 3	Film 4	Film 5	Film 6
Pierre	5	1	3.5	4	5	2
Jean	5	2	4			1
Charlotte	4		3	3		2
Léa	2				5	1

On a ici utilisé une simple moyenne (sur les données non manquantes) pour prédire les notes. On aurait pu utiliser une moyenne pondérée avec comme poids la similarité entre l'utilisateur X et Pierre ou tout autre fonction qu'on juge pertinente.

Considérons que le seuil de recommandation est fixé à 3.5, on peut donc recommander à Pierre les deux films 5 et 3 avec une préférence pour le 5.

5.3 Le projet TR (Twitter Recommandation)

5.3.1 Domaine d'application

On souhaite développer un algorithme adapté au cadre de Twitter. On rappelle brièvement le fonctionnement de ce réseau :

- Chaque utilisateur crée un compte afin de partager des messages brefs (moins de 140 caractères) appelés Tweets
- Les tweets d'un utilisateur apparaissent sur sa page personnelle
- Les messages peuvent être caractérisés à l'aide de hashtag (#), par exemple : #Rapportdestage. Ce sont des objets cliquables qui permettent d'accéder à tous les autres tweets contenant le même hashtag
- Un utilisateur peut adresser un tweet à un autre utilisateur en utilisant la balise @, par exemple @NicolasPoulin
- Les tweets peuvent aussi être accompagnés de liens ou d'images
- Les relations sociales ne sont pas symétriques (contrairement à Facebook où une demande d'ami est un "contrat" réciproque entre les deux parties). Ici se sont des liens de suivi. Un utilisateur choisit qu'il veut suivre mais ne choisit pas qui le suit
- Un utilisateur peu retweeter le tweet d'un utilisateur tiers. Il fait alors apparaitre le message sur son propre mur

5.3.2 Objectifs

L'algorithme final devra entre autre :

- Collecter les informations publiques d'un utilisateur et en extraire l'information essentielle
- Caractériser le réseau existant de l'utilisateur
- Repérer des candidats potentiels à la recommandation, établir des similarités
- Recommander un candidat (ou une liste) pour un éventuel following

5.3.3 Spécifications techniques

Les données brutes sont extraites de Twitter via son API. Elles sont ensuite stockées dans une base de données puis utilisées par un logiciel tiers.

Les outils utilisés sont les suivants :

-  Le logiciel R pour l'exploitation des données
-  La technologie MySql pour la gestion de la base de données

Le logiciel R permet de se connecter directement à la base SQL via une connexion ODBC (on utilise pour cela le package RODBC).

5.4 Travail effectué

5.4.1 Les données disponibles

On dispose de 277 utilisateurs dans la base et pour chacun d'entre eux :

- De sa liste complète de Tweets, leurs dates de création et combien de fois ils ont été retweetés
- De sa liste de followings (les utilisateurs qu'il suit)
- De sa liste de followers (les utilisateurs qui le suivent)

Il y a au total 163 100 tweets dans la base.

5.4.2 Aperçu statistique des données

Le volume de tweets

Le volume de tweet est tout simplement le nombre de tweets postés par un utilisateur depuis la création de son compte.
Moyenne : 611 tweets
Ecart type : 407 tweets

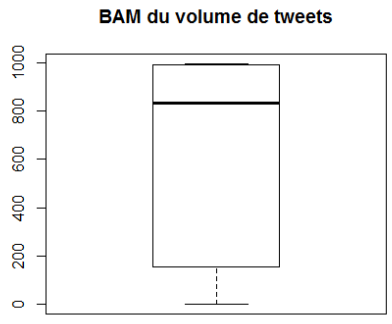


FIGURE 5.1 – La boîte à moustache de la série

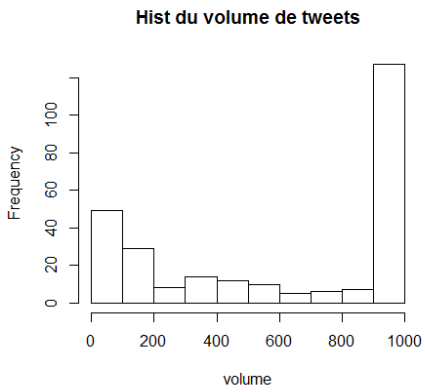


FIGURE 5.2 – L'histogramme de la série

La distribution du volume est plutôt large avec un écart interquartiles de 838 tweets. La médiane est de 834 tweets ce qui est largement au-dessus de la

moyenne. Comme l'indique l'histogramme, la série est dominée par ses valeurs élevées, 48% des utilisateurs ayant un volume supérieur à 900 tweets.

Le taux d'inactivité

Le taux d'inactivité est la proportion de jours sans tweets pour un utilisateur donné.
Moyenne : 53%
Ecart-type : 34%

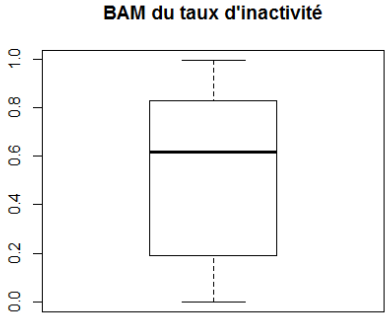


FIGURE 5.3 – La boîte à moustache de la série

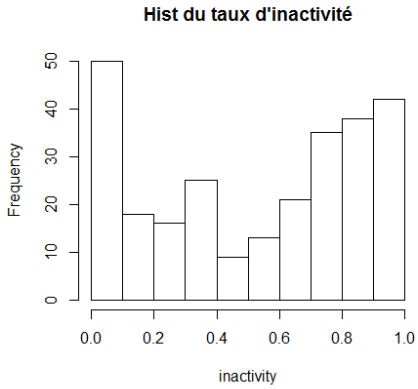


FIGURE 5.4 – L'histogramme de la série

La série est assez étendue avec un écart interquartiles de 0.64. Elle est influencée par des valeurs élevées, la médiane étant de 0.62 ce qui est supérieur à la moyenne. 30% des utilisateurs ont un taux d'inactivité de 80% ce qui signifie qu'ils ne tweetent qu'un jour sur cinq. L'utilisateur moyen tweete quand à lui un jour sur deux. Par ailleurs, la classe la plus représentée est celle des valeurs très faibles (inférieurs à 0.1) qui permet d'équilibrer la série.

La profondeur du réseau

On appelle profondeur du réseau, la somme des followers et followings d'un utilisateur donné.
Moyenne : 277 utilisateurs
Ecart-type : 111 utilisateurs

La série est fortement influencée par ses valeurs élevées : le premier quartile est à 211 utilisateurs ce qui est plutôt proche de la moyenne. La série est peu

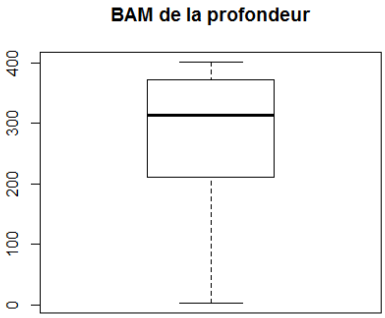


FIGURE 5.5 – La boîte à moustache de la série

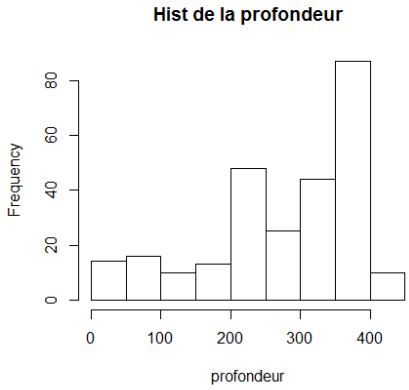
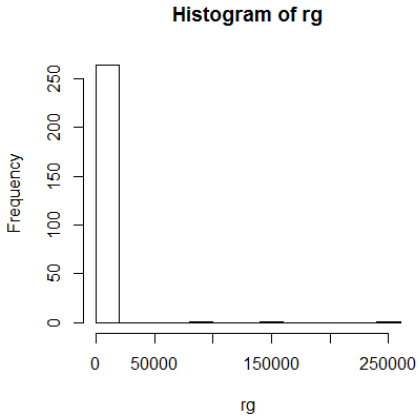


FIGURE 5.6 – L'histogramme de la série

étendue avec un écart interquartiles de 161 utilisateurs. La valeur médiane est 313 utilisateurs. Par ailleurs, 36% des utilisateurs ont une profondeur supérieure à 350.

L'équilibre follower/following

L'équilibre follower/following est le rapport entre le nombre de followers et le nombre de following (followers/followings). Dans le cas où l'utilisateur n'a pas de followings, l'équilibre est égale au nombre de followers. Moyenne : 2063.97
Ecart-type : 18941.31
L'histogramme atteste d'une série complètement déséquilibrée (ce qui n'est pas étonnant vu le rapport moyenne/écart-type) :



La BAM correspondante est illisible (c'est un simple trait, elle est « écrasée » par les valeurs extrêmes). En fait, 53% des valeurs sont inférieures ou égales à 1, 90% des valeurs sont inférieurs à 86.39 et 95% sont inférieures à 686.3 ! La médiane est à 0.88. Dans le cas d'une série comme ceci avec des valeurs très élevées (quasi disproportionnées) mais très rare, la médiane est un meilleur indicateur que la moyenne quant au profil de l'utilisateur typique. Ici, on remarque que les réseaux tendent à être légèrement déficitaires en followers, ce qui indique qu'un équilibre r/g excédentaire est un « privilège » ou tout du moins un bon indicateur de la popularité de l'utilisateur.

Le nombre de retweet

Le nombre de retweet est calculé sur les tweets originaux d'un utilisateur (i.e. qu'il n'a pas lui-même retweetés).
Moyenne : 2185
Ecart-type : 13462
On ne présentera même pas l'histogramme. On a à nouveau une série complètement déséquilibrée (à cause de ses valeurs élevées) : la médiane est de 79 retweet, et 90% des utilisateurs ont moins de 1673 retweet. La valeur maximale quant à elle est de 161633 retweets.

Le taux d'utilisation des objets

On définit le taux d'utilisation des objets comme le pourcentage de tweet comprenant au moins un objet, à savoir un #, un @ ou un lien.
Moyenne : 91%
Ecart-type : 17%

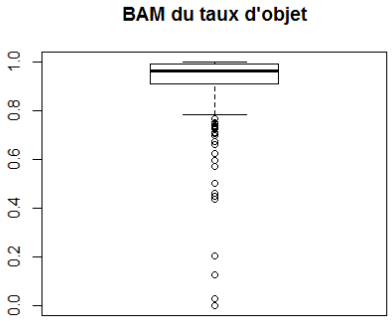


FIGURE 5.7 – La boîte à moustache de la série

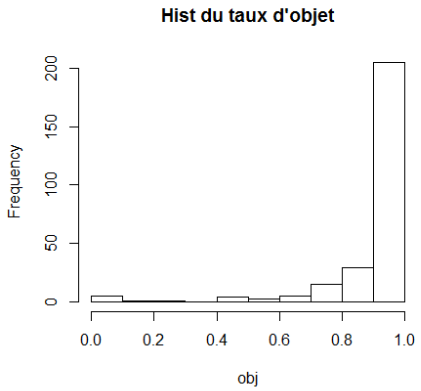


FIGURE 5.8 – L'histogramme de la série

La série est très déséquilibrée. 77% des valeurs sont supérieures ou égales à 0.9. La médiane est à 0.96. La série est « perturbée » par une série de valeurs plus faibles. A titre d'information, la deuxième barre la plus haute de l'histogramme (les valeurs comprises entre 0.8 et 0.9) ne représente que 11% de l'échantillon.

Un utilisateur typique...

- Un volume moyen de 611 tweets
- Un taux d'inactivité de 53
- Un réseau fort de 277 followers+followings
- Un réseau déficitaire en followers avec un équilibre autour de 0.88
- 79 retweets sur ses tweets originaux
- 9 tweets sur 10 comportent au moins un objet

5.4.3 Méthode de recommandation

Twitter n'offre pas un cadre utilisateur-objet classique comme présenté dans la partie théorique. Ainsi, il nous faut adapter la méthode de filtrage collaboratif à nos données.

On a choisi d'appliquer la méthodologie suivante (un utilisateur cible étant fixé)

1. On sépare les utilisateurs présents dans la base de données en deux groupes :
   - les candidats, i.e. les utilisateurs qui ne sont pas suivis par la cible
   - les "autres", i.e. les followings de la cible présents en base
2. On établit les profils de tous les utilisateurs
3. On cherche parmi les utilisateurs "autres" les p plus proches de la cible (ce sont les "top-sociaux", p à fixer)
4. On compare les profils des tops-sociaux aux candidats et on retient ceux les plus proches du groupe des "tops-sociaux" selon un critère à définir.

En pratique, on choisit comme mesure de similarité le coefficient de corrélation de Pearson. Les profils sont obtenus selon la méthodologie exposée ci-dessous.

5.4.4 Création du profil utilisateur

On va traiter les tweets des utilisateurs de la manière suivante :

- On identifie les "objets", c'est-à-dire les #, les @ et les liens dans chaque tweet
- On purge le reste du tweet (le texte brut) de tous les mots vides de sens (les "stop-words", comme par exemple le, la, les, aux etc...)
- On compte ensuite le nombre d'occurrences pour chaque #, chaque @ et chaque mot porteur de sens.

A partir de ces données on peut établir un profil utilisateur en ne retenant que les k mots, # et at les plus fréquents. Par exemple, si on prend k=5, le profil d'un utilisateur quelconque est le suivant :

milan	0.0072	#forzamilan	0.0221	@lildada	0.1129
monde	0.0054	#calcio	0.0200	@mcanzeno	0.0965
énorme	0.0052	#sochi2014	0.0168	@ffponey	0.0575
news	0.0046	#vieetudiante	0.0158	@gazzetta_it	0.0411
france	0.0039	#forzaitalia	0.0116	@luigichone	0.0329

On peut aussi envisager de prendre un paramètre différent pour chaque type d'objet, i.e. de retenir k_1 mots, k_2 hashtags et k_3 at.

5.4.5 En résumé...

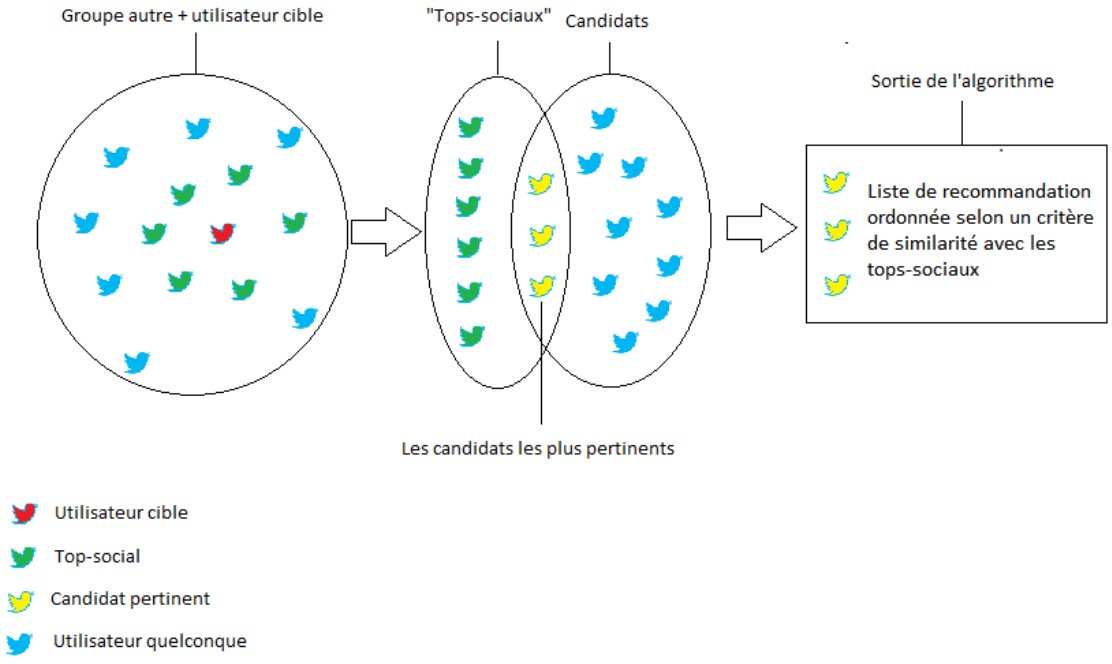


FIGURE 5.9 – Un résumé du protocole

5.4.6 Quelques tests

En guise de tests, on propose la méthodologie suivante :

1. On déplace manuellement un utilisateur de la liste autre vers la liste candidat. On choisira préférablement un utilisateur avec une forte similarité vis-à-vis de la cible.
2. On fait tourner l'algorithme. On considère le test réussi si l'élément déplacé apparait en bonne place dans la recommandation.

Pour ces tests, on a établi les profils en sélectionnant les 20 mots, # et @ les plus utilisés par un utilisateur. On à par ailleurs garder les 5 autres utilisateurs les plus proches de l'utilisateur cible.

Ci-dessous, les 24 premiers autres utilisateurs en termes de coefficient de pearson vis-à-vis de « antoined » :

nsanitas 0.6992	H4st0 0.6813	matleclaire 0.5069	oghma777 0.3618	naikyworld 0.3311	njl69 0.2988	fXzo 0.2694	Kyoku57 0.2654
macsroustan 0.2528	klein_stephane 0.2063	aagahi 0.1874	NicolasLabrot 0.1868	nmartignole 0.186	aheritier 0.1829	DevoxxFR 0.1798	guillaumebort 0.1773
lambdadevfr 0.1739	gbougeard 0.1705	adericbourg 0.1636	fsarradin 0.1578	Knolspeak 0.1578	CodeStory 0.157	gfred0404 0.1557	morlhon 0.1548

Les résultats :

- Dans le cas du transfert de "nsanitas"

nsanitas	**courieti**	**dupot_org**	**aadlani**	**toutluxembourg**
0.383	0.0823	0.0531	0.0257	0.0251

- Dans le cas du transfert de "H4st0"

H4st0	**courieti**	**dupot_org**	**ealliaume**	**toutluxembourg**
0,4784	0,0951	0,0733	0,042	0,0247

- Dans le cas du transfert de "matleclaire"

matleclaire	**courieti**	**dupot_org**	**ealliaume**	**aadlani**
0.4293	0.1114	0.0918	0.0367	0.0233

- Dans le cas du transfert de "oghma777"

oghma777	**courieti**	**dupot_org**	**ealliaume**	**aadlani**
0.3184	0.116	0.093	0.0447	0.0295

- Dans le cas du transfert de "naikyworld"

naikyworld	**courieti**	**dupot_org**	**ealliaume**	**aadlani**
0.1281	0.0837	0.0835	0.0351	0.0302

- Dans le cas du transfert de "njl69"

njl69	**courieti**	**dupot_org**	**ealliaume**	**aadlani**
0.1995	0.1082	0.0825	0.0426	0.0261

- Dans le cas du transfert de "fXzo"

fXzo	**courieti**	**dupot_org**	**ealliaume**	**aadlani**
0.1714	0.1082	0.0824	0.0426	0.0261

- Dans le cas du transfert de "Kyoku57"

Kyoku57	courieti	dupot_org	ealliaume	aadlani
0.1494	0.1081	0.0824	0.0425	0.026

- Dans le cas du transfert de "macsroustan"

macsroustan	courieti	dupot_org	ealliaume	aadlani
0.1494	0.1082	0.0825	0.0427	0.0262

- Dans le cas du transfert de "klein_stephane"

klein_stephane	courieti	dupot_org	ealliaume	aadlani
0.1656	0.1082	0.0824	0.0426	0.0261

- Dans le cas du transfert de "aagahi"

aagahi	courieti	dupot_org	ealliaume	aadlani
0.1523	0.1082	0.0824	0.0426	0.0261

Pour les 24 premiers tops-sociaux le test s'avère concluant puisqu'en cas de transfert, ils se retrouvent en première place de la liste de recommandation.

C'est à partir de l'utilisateur numéro 30 (en termes de coefficient de pearson) que l'algorithme ne renvoie plus cet utilisateur à la première place.

Sa similarité avec antoined est :

sirthias
0.1391

Et le résultat de l'algorithme :

courieti	sirthias	dupot_org	ealliaume	aadlani
0.1083	0.0949	0.0825	0.0427	0.0262

En conclusion, ces tests s'avèrent plutôt concluants, l'algorithme est tout d'abord peu sensible à un changement ponctuel dans les top-sociaux (on remarque que les recommandations 2, 3 et 4 sont toujours les mêmes et que sauf pour le transfert 1 et 2, la recommandation 5 ne bouge pas non plus) et, en cas de suppression d'un following pertinent (proche de l'utilisateur cible) l'algorithme l'identifie à nouveau comme un candidat potentiel.

5.4.7 Clusterisation des données

En fonction du comportement des utilisateurs

L'algorithme de recommandation présenté précédemment permet de générer des recommandations personnalisées, c'est-à-dire qui s'appuient sur l'activité passée de l'utilisateur. Il existe une autre approche de la recommandation, plus simple, qui consiste à conseiller à un utilisateur ce qui est populaire. Pour cela, il suffit de mettre en place une liste des objets les plus vendus/consultés/aimés (tout dépend du contexte) et de présenter cette liste à l'utilisateur. Cette méthode présente tout de même quelques inconvénients :

- Elle ne propose pas de contenu original et est donc plus susceptible de présenter à l'utilisateur des objets qu'il connait déjà
- Elle est très peu adaptable

Cependant, des améliorations sont faciles à apporter. Prenons l'exemple d'un site web proposant du streaming vidéo. Si un utilisateur regarde beaucoup de comédie et de films policier, la recommandation pourra être les comédies et les polars les plus streamés plutôt que simplement les films les plus streamés en général. Une idée originale est de combiner un algorithme de recommandation personnalisé avec une méthode basée sur la popularité. Cependant, si dans un contexte marchand la popularité est facile à définir, cela semble être un peu plus complexe sur Twitter. On a défini précédemment (section filtrage en amont) quelques critères qui pourraient permettre de sélectionner les utilisateurs les plus « populaires ». On a calculé chacun de ces critères pour notre échantillon de 277 utilisateurs et réalisé une ACP.

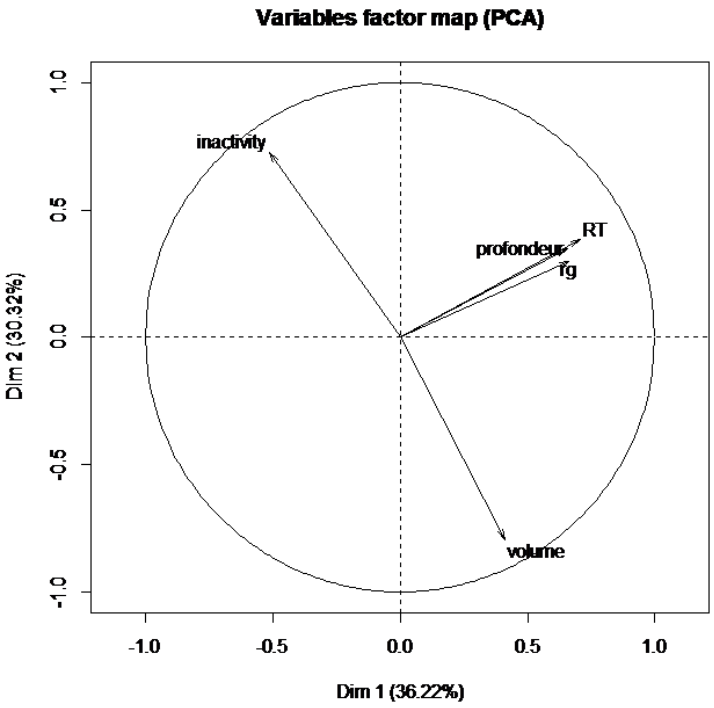


FIGURE 5.10 – Le cercle des corrélations

Le plan factoriel explique 66.54% de la variabilité entre les individus. La projection est satisfaisante. Néanmoins, les variables "RT" et "profondeur" sont très fortement corrélées, de même les variables "inactivity" et "volume" sont inversement corrélées. On a une redondance dans l'information. On décide donc de supprimer la variable "profondeur" (loin du cercle unité) et la variable "in-

activity".

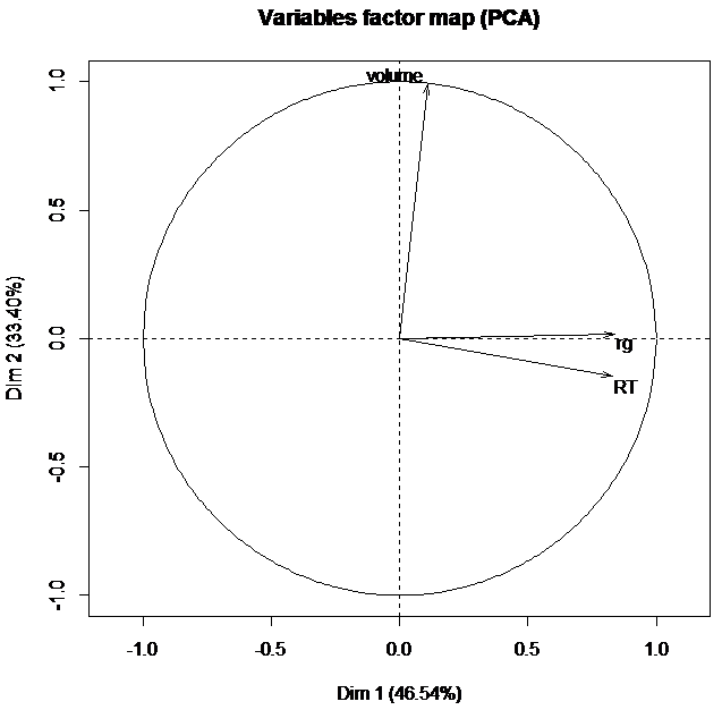


FIGURE 5.11 – Le 2ème cercle des corrélations

La qualité de la projection est meilleure (79.94% de la variabilité est expliquée) ce qui est logique. Les résultats restent cohérents.

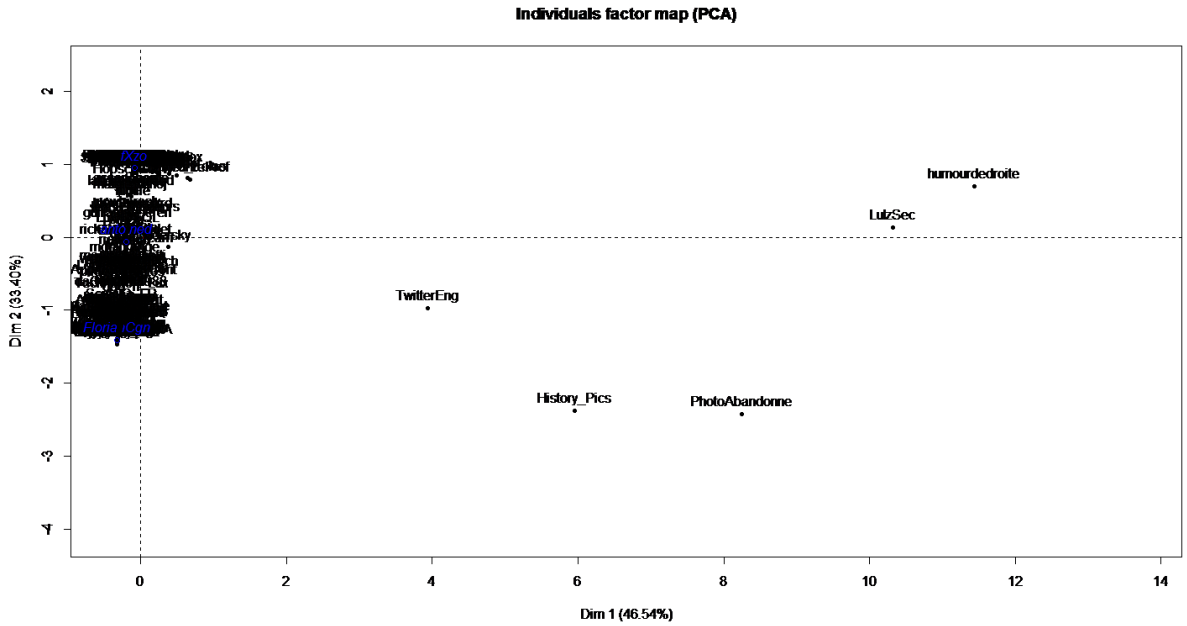


FIGURE 5.12 – Le plan factoriel

Ci-dessus, le plan factoriel des individus. Les observations ont été centrées et réduites. On observe clairement deux groupes (ou 3). A gauche du plan, un groupe sur concentré. C'est le groupe des utilisateurs moyens. Ceux-ci ne se démarquent qu'en fonction de la seconde dimension. Ils sont donc différents du point de vue de la variable volume. En bas de ce nuage, ce sont les individus peu présents sur le réseau, un exemple est "FlorianCgn". Au milieu, des individus moyens, comme "antoined", c'est un utilisateur "typique" de Twitter (du point de vue de notre échantillon en haut, les utilisateurs actifs, avec un volume important comme "fXzo". Le point commun entre tous les utilisateurs de ce groupe est qu'ils ne sont peu retweetés et qu'ils n'ont pas un équilibre follower/following excédentaire, en clair, leur impact sur le réseau est limité –bien qu'ils puissent avoir un impact local sur le réseau, comme c'est probablement le cas pour "fXzo".

C'est le long de la première dimension que se démarquent les groupes. A droite du nuage d'utilisateurs moyens, on, remarque deux groupes : les utilisateurs très populaires avec un volume important (comme "humourdedroite" ou "LulzSec") qui se situent en haut à droite et les utilisateurs populaires avec un volume plus faible (comme "History_Pics" ou "PhotoAbandonne") qui se situent en bas à droite.

Ce qu'on peut constater, c'est que les individus qui se démarquent le long de la première dimension ne sont pas de simples utilisateurs, mais plutôt des comptes

officiels ou liés à un contenu extérieur. On peut supposer que les figures publiques (François Hollande, Barack Obama, Karim Benzema etc. . .) se placeraient aussi à cet endroit du plan. Cette constatation nous amène à la réflexion suivante : si on filtre les recommandations selon la "popularité" des utilisateurs, en partant du principe que celle-ci est proportionnelle à l'équilibre rg et au nombre de RT alors, bien vite, on ne recommandera plus que des comptes officiels, des personnages publiques ou des célébrités. . . Ces recommandations, bien que probablement très pertinentes, ne sont pas forcément "intéressante". Tout dépend de ce qu'on cherche à accomplir avec l'algorithme.

1. Si on cherche avant tout à recommander du contenu de qualité et "approuvé" par le réseau alors un filtrage par popularité peut être pertinent.
2. Si on cherche à connecter des utilisateurs "typiques" partageant des centres d'intérêts et susceptibles d'interagir de manière plus symétrique alors le filtrage doit être évité.

Une approche hybride peut aussi être envisagée : on regroupe les individus en n clusters et on recommande des individus des différents clusters.

On choisit d'appliquer cette méthode et on a donc effectué une CAH sur les composantes principales de l'ACP.

Nombre idéal de cluster : 3

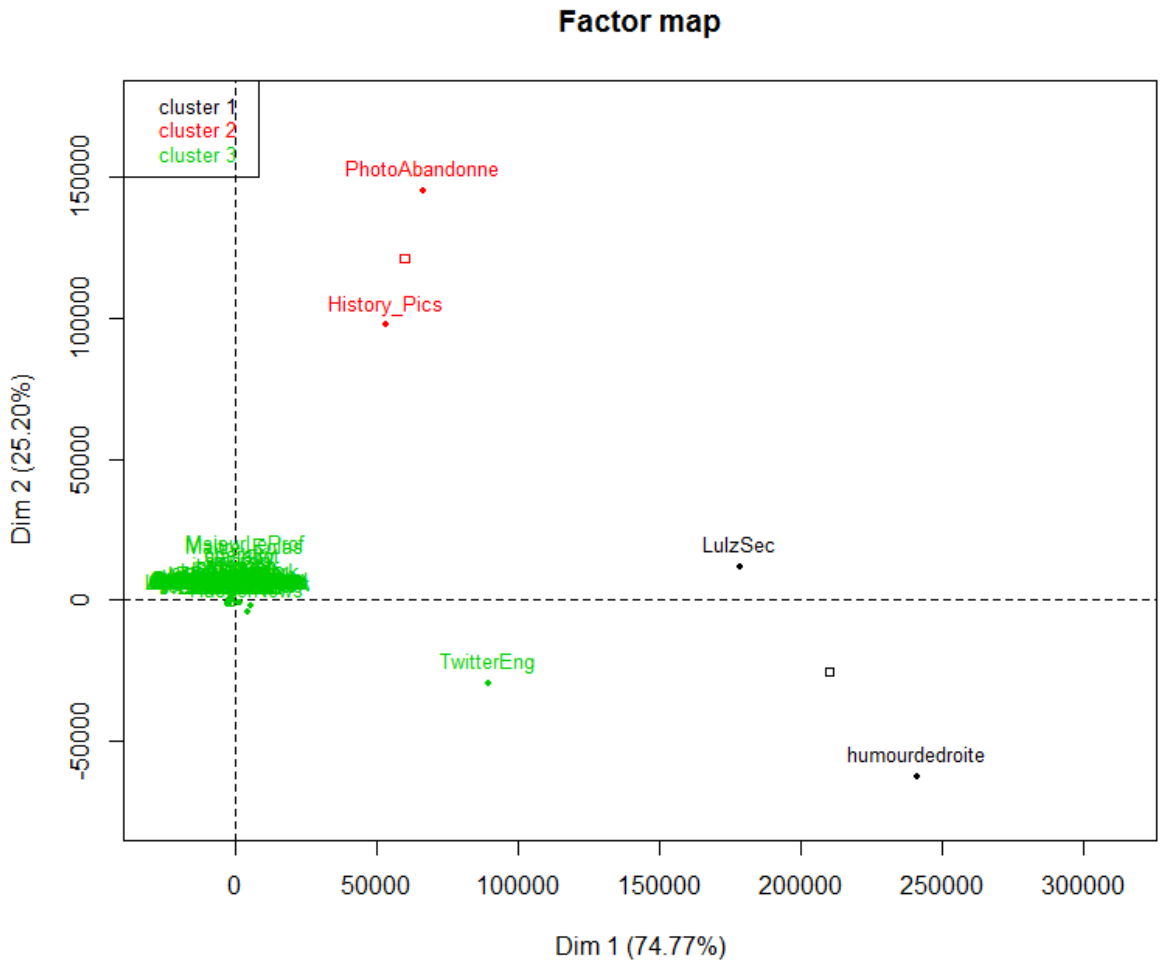


FIGURE 5.13 – Le plan factoriel et les clusters

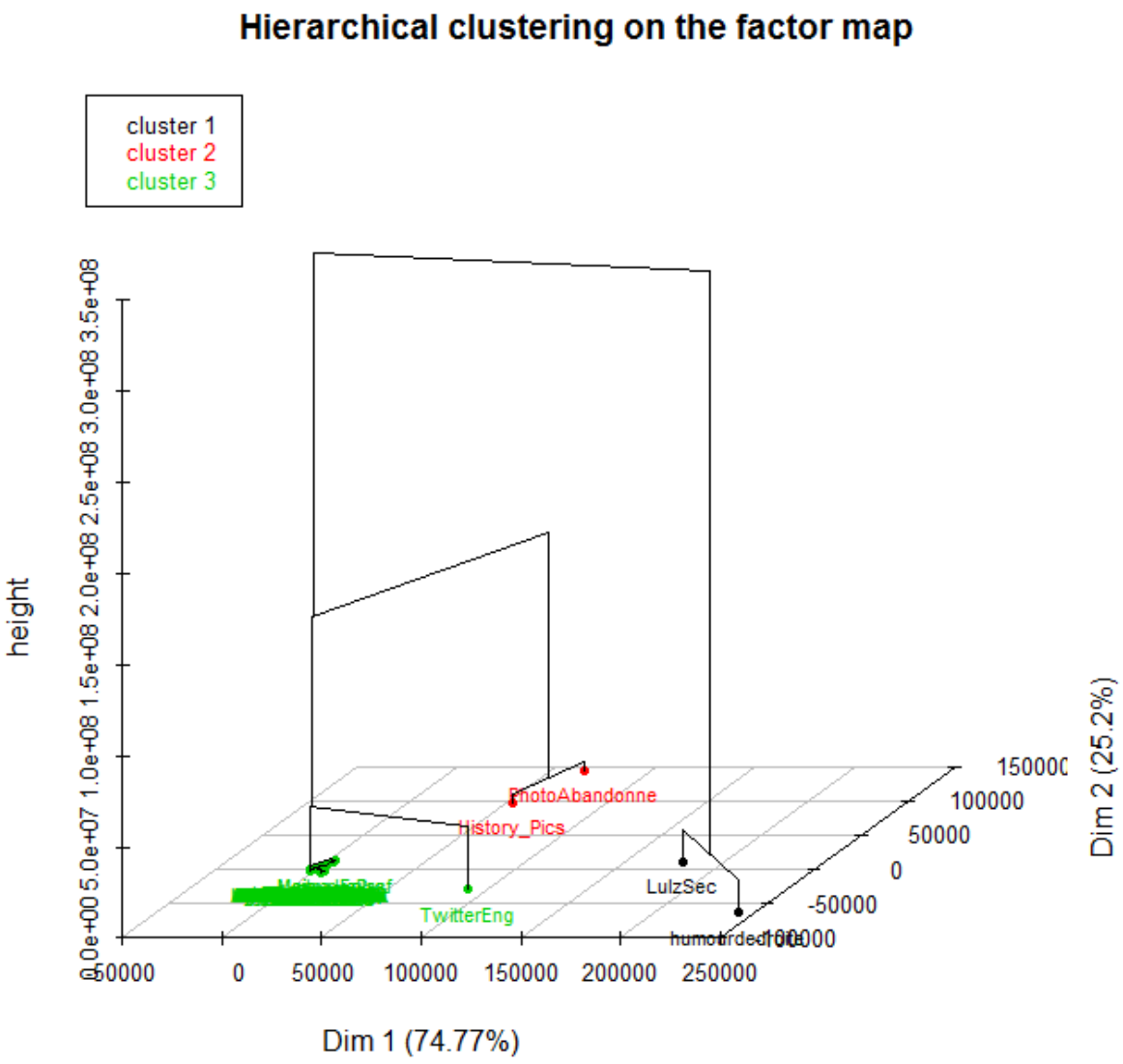


FIGURE 5.14 – L'arbre

On Remarque que le classement ne se fait que principalement selon la première dimension qui retient 74.77% de l'information (la seconde dimension n'est cependant pas négligeable avec 25.20% de l'information).

En fonction des centres d'intérêts

Prenons le profil d'un utilisateur quelconque en termes de mots et de hashtags :

Mots	freq	Hashtags	freq
milan	0.71	#forzamilan	2.21
monde	0.53	#calcio	2
énorme	0.51	#sochi2014	1.69
news	0.46	#vieetudiante	1.58
france	0.38	#forzaitalia	1.16
allociné	0.35	#senscritique	1.16
iphone	0.35	#nancy	1.05
matin	0.29	#karatespain2013	1.05
apple	0.29	#francetvsport	1.05
jeu	0.26	#lesjoiesducode	0.95
match	0.26	#gtav	0.95
série	0.26	#karaté	0.95
coupe	0.26	#motscroises	0.95
di	0.24	#football	0.84
films	0.24	#tpmp	0.84

On remarque que l'utilisateur n'aborde pas qu'un seul thème : le football est largement présent (#calcio, milan etc...) , le karaté, le sport en général, le cinéma (sens critique, films) etc... Or, dans le cadre d'une recommandation, il est peu probable que l'on trouve un utilisateur abordant exactement les mêmes sujets. En réalité, on pourra exhiber des utilisateurs avec seulement une partie d'intérêts communs. Ainsi, supposons que l'on puisse regrouper les candidats à la recommandation en groupes en fonction des sujets abordés. Chaque groupe aurait alors quelques similarités avec "MCanzerini". Une vision très simplifiée de la chose est la suivante :

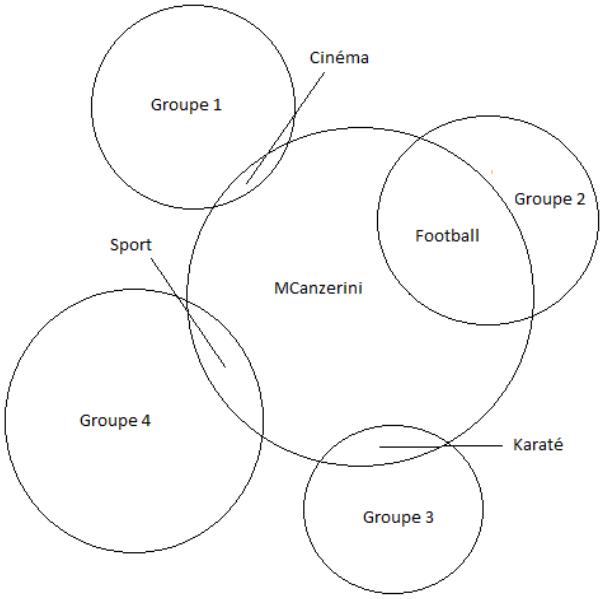


FIGURE 5.15 – Les différents groupes d'intérêts

Schématiquement, la taille des cercles représente le cardinal des groupes ("MCanzerini" est lui aussi représenté par un cercle pour des raisons pratiques). La taille de leurs intersections avec "MCanzerini" représente l'importance de la similarité.

Ainsi, si on devait manuellement faire une recommandation on serait naturellement tenté de choisir un élément du groupe 2. En effet, le profil de "MCanzerini" est très marqué par le football, et son intersection avec le groupe 2 est importante. Dans le cadre d'une recommandation automatisé, cela peut conduire à une redondance : on ne souhaite pas recommander uniquement des utilisateurs du groupe 2, bien que celui-ci domine les autres (du point de vue de la similarité).

Afin de prévenir cet effet, il "suffit" de changer de groupe ! Cependant, en réalité, on ne dispose pas de tels groupes bien définis. En effet, s'il est simple pour un humain de regrouper des mots en sujets (milan, calcio et forzaitalia dans le groupe football par exemple) il est beaucoup plus complexe d'automatiser cette démarche.

Plutôt que de se lancer dans une procédure de reconnaissance d'information, on va s'appuyer sur la similarité entre les candidats et les top-sociaux.

Les idées qui soutiennent cette démarche sont les suivantes :

- Les tops sociaux sont un bon indicateur de ce que l'utilisateur cible apprécie ou peu apprécier
- Les tops sociaux abordent suffisamment de sujets distincts pour que l'on puisse former des groupes différents

On établit donc une matrice NxP où :

- N est le nombre de candidats
- P est la précision, i.e. le nombre de tops sociaux

Une case de la matrice est la similarité entre le candidat i ($1 \leq i \leq N$) et le top social j ($1 \leq j \leq P$) On peut voir cette matrice comme un tableau classique individu x variables. On effectue sur ce tableau une HCPC sur ses composantes principales (i.e. ACP + CAH).
Dans notre exemple, N=38 et P=5. L'utilisateur cible est "antoined". L'analyse suggère 7 groupes :

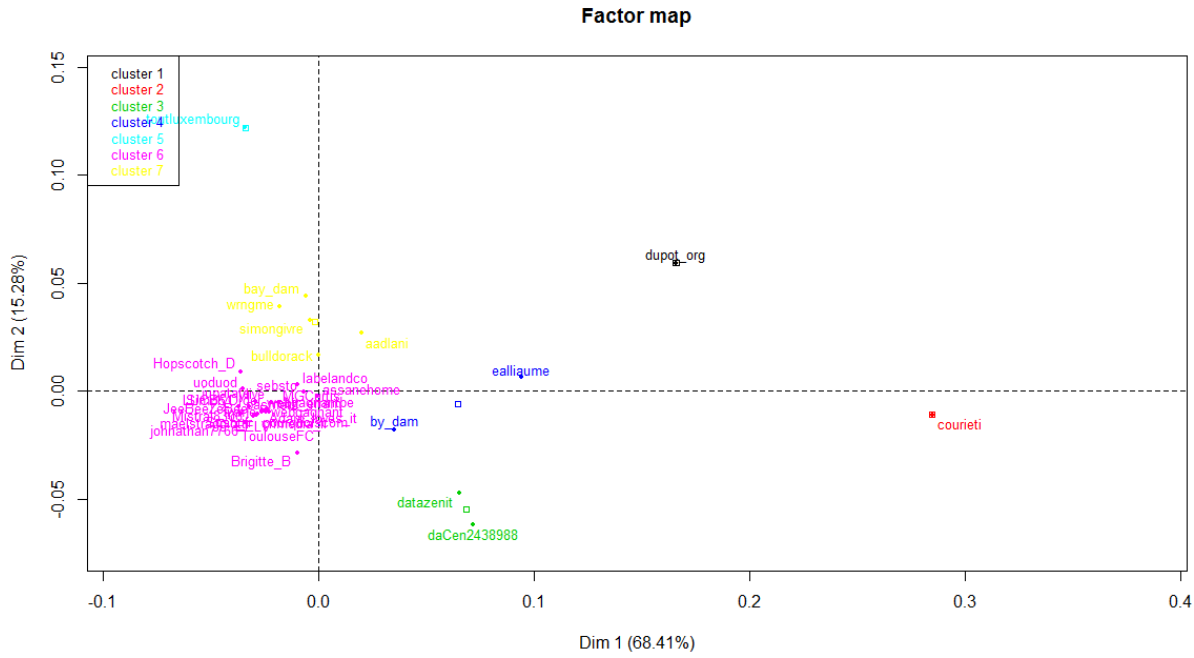


FIGURE 5.16 – Le plan factoriel et les clusters

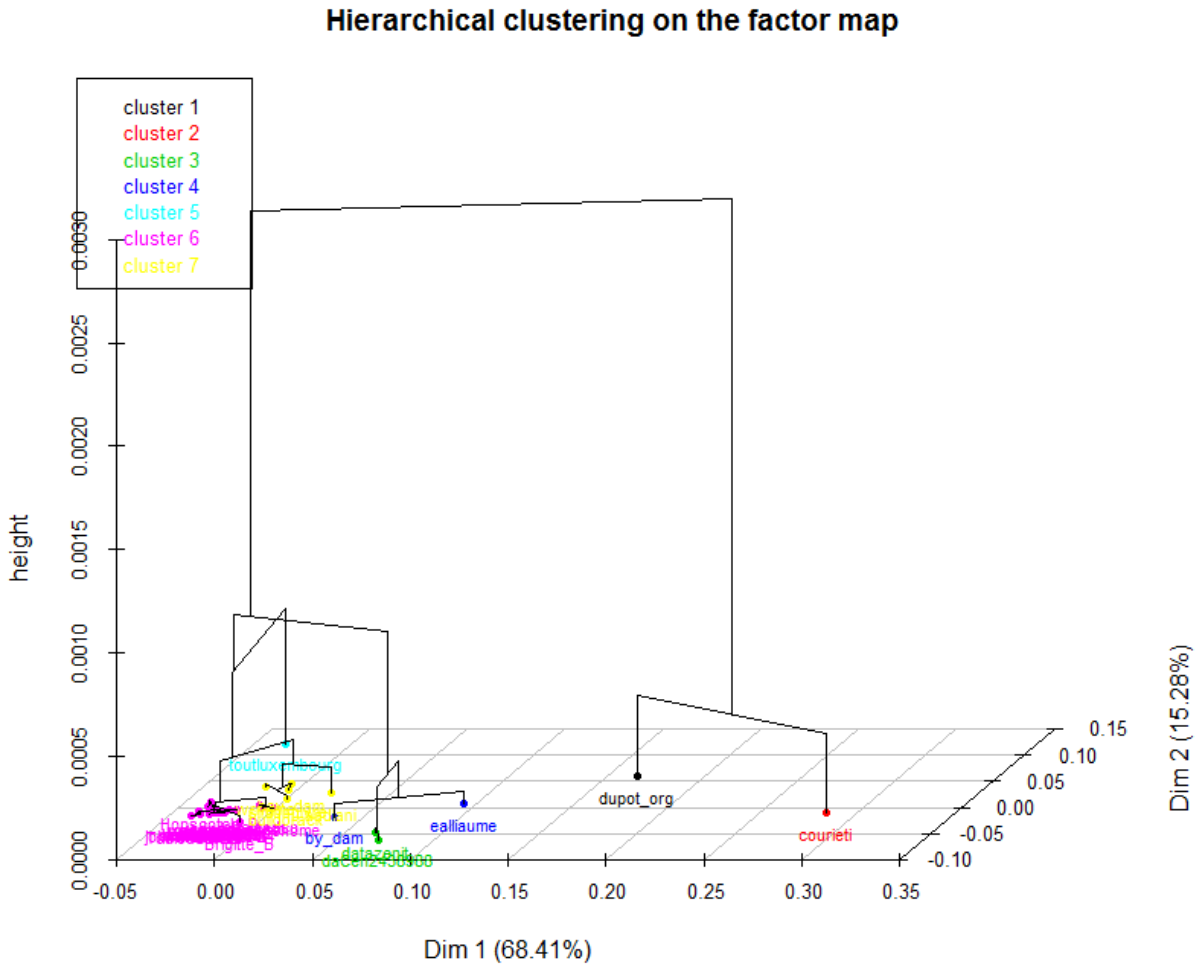


FIGURE 5.17 – L'arbre

On remarque que les deux dimensions 83,69% de l'inertie, ce qui indique que la représentation est plutôt bonne.
Ici, le résultat peut paraitre décevant. En effet, les groupes sont déséquilibré, avec trois groupes de cardinal 1, deux de cardinal 2 et un groupe monopolisant une majorité des candidats. Cependant, on espère qu'avec plus des données et en particulier des données moins centrées autour de « antoined » (et donc moins marquées par l'IT et InTech), on arrivera à des clusters plus équilibrés.

6 Conclusion

Ce stage au sein d'InTech a été pour moi une expérience très enrichissante. En effet, j'ai eu l'occasion de travailler sur deux sujets différents mais également stimulant. Le premier, plus technique, m'a permis de compléter mes connaissances du logiciel R et de maîtriser au mieux la création d'un programme via ce langage.
Le second, à la fois technique et théorique, m'a fait découvrir le monde des algorithmes de recommandation. J'ai ainsi pu me familiariser avec des méthodes pour moi jusque-là inconnu puis j'ai eu l'occasion de construire mon propre algorithme pour une application concrète sur le réseau Twitter.

Par ailleurs, toujours d'un point de vue technique, j'ai pu me familiariser avec la gestion d'une base de données SQL, outil au combien pratique pour le stockage de gros volumes de données.

D'un point de vue humain, l'excellente ambiance qui règne au sein de l'entreprise, le dynamise des collègues et l'esprit de groupe -qui se manifeste lors des évènements extra-professionnels- ont été des composantes essentielles au plaisir que j'ai trouvé à effectuer mon stage.

J'espère pouvoir être amené, dans le futur, à collaborer à nouveau avec InTech.

Appendices

A Lexique du projet Metadata

- CCYY : Le code permettant d'identifier un dataset. Il est en fait composé du code iso (2-digits du pays (CC, par exemple FR pour France) et des deux derniers chiffres de l'année (YY par exemple 04 pour 2004).
- Database : Un ensemble de datasets. Il existe deux database "LIS" et "LWS" (toutes deux créées par le LIS).
- Datafiles : (Household ou Person). Le premier regroupe les micro-données au niveau des ménages pour un pays pour une année donnée. Le second les micro-données au niveau des personnes.
- Dataset : Ils sont composés de deux Datafiles (household et person). C'est l'ensemble des micro-données disponibles pour un pays à une année donnée.
- Micro-données : les données harmonisées (fournies par les organismes de statistiques) et standardisées par le LIS. Elles se présentent sous forme de fichiers stata.
- Variables : l'ensemble des variables traités par le LIS. Il en existe 3 types :
  - Les variables quantitatives, par exemple le revenu disponible d'un foyer.
  - Les variables qualitatives, par exemple la nationalité d'un sondé, qui peuvent soit êtres :
    - "Country specific", i.e. spécifiques au pays étudié
    - Ou "non country specific", communes à tous les pays

    A noter que certaines variables sont communes aux deux datafiles. Par ailleurs, pour un dataset donné l'ensemble des variables ne sont pas toujours existantes
- Pull-programs : les programmes permettant de faire le lien entre les micro-données et l'application Metadata

B Compléments sur le projet TR

B.1 A propos du premier clustering

On donne ci-dessous deux tableaux supplémentaires. Le premier présente le "êta-carré" pour les variables RT et rg (accompagné du résultat au test de significativité, i.e. la p-value). Le second est une description des différents clusters par ces deux variables.

	eta2	p-value
RT	0.9357768	4.124094e-158
rg	0.8619052	3.187058e-114

Ce tableau nous montre que la quasi-totalité de la variance à l'intérieur de l'échantillon (93.58%) est expliquée par la variable RT. La variable rg explique 86.19% de la variance.

Cluster 1

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
rg	15.133922	204000.0	2063.971	44000.0	18905.81	9.675387e-52
RT	6.664697	65388.5	2185.449	20871.5	13436.64	2.652125e-11

Cluster 2

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
RT	14.24959	137318	2185.449	24315	13436.64	4.509302e-46

Cluster 3

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
rg	-11.00213	491.1354	2063.971	5940.059	18905.81	3.732080e-28
RT	-14.84476	677.1939	2185.449	1990.666	13436.64	7.522694e-50

B.2 A propos du second clustering

	Eta2	P-value
naikyworld	0.9709033	2.113469e-22
nsanitas	0.9431830	6.401216e-18
oghma777	0.9261717	3.585320e-16
H4st0	0.8543460	1.156714e-11
matleclaire	0.6646374	2.996687e-06

Cluster 1

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
nsanitas	3.830197	0.1651927	0.009140987	NA	0.04074247	0.0001280407
H4st0	3.478976	0.1026020	0.006131839	NA	0.02772947	0.0005033335

Cluster 2

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
naikyworld	4.535105	0.22423422	0.009121396	NA	0.04743282	5.757489e-06
H4st0	3.935940	0.11527338	0.006131839	NA	0.02772947	8.287162e-05
nsanitas	3.577833	0.15491072	0.009140987	NA	0.04074247	3.464550e-04
matleclaire	3.220153	0.04529552	-0.003601430	NA	0.01518467	1.281222e-03

Cluster 3

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
naikyworld	3.214784	0.1154784	0.009121396	0.01587997	0.04743282	0.001305427

Cluster 4

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
matleclaire	2.921479	0.02734021	-0.00360143	0.01254749	0.01518467	0.003483739

Cluster 5

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
oghma777	5.424166	0.1581497	0.000777632	NA	0.02901313	5.82257e-08

Cluster 6

	v.test	Mean in category	Overall mean	sd in category	Overall sd	p.value
oghma777	-2.259143	-0.006542879	0.000777632	0.006171322	0.02901313	0.0238745128
matleclaire	-3.090858	-0.008843313	-0.003601430	0.005990579	0.01518467	0.0019957900
H4st0	-3.131708	-0.003567138	0.006131839	0.006380749	0.02772947	0.0017379265
naikyworld	-3.249390	-0.008092685	0.009121396	0.007585319	0.04743282	0.0011565297
nsanitas	-3.717245	-0.007773996	0.009140987	0.008046835	0.04074247	0.0002014072

Le 7ème cluster n'est pas caractérisé par une variable en particulier.

Bibliographie

[1] http ://www.lisdatacenter.org Le site web du LIS

[2] http ://www.lisdatacenter.org/lis-ikf-webapp/app/search-ikf-figures L'application qui permet d'accéder aux Inequality Key Figures

[3] http ://www.lisdatacenter.org/wp-content/uploads//our-lis-documentation-by-lu10-hcodebook.txt Un exemple de codebook

[4] http ://infolab.stanford.edu/ ullman/mmds/ch9.pdf Une introduction complète sur les algorithmes de recommandation

[5] http ://www.persane.com/ Quelques détails sur les méthodes statistiques utilisées

[6] http ://knight.cis.temple.edu/ vasilis/Courses/CIS750/Papers/jain99data.pdf Un article complet sur les méthodes de clustering