

Cahier thématique

Recommandation de Films Basé sur le filtrage Collaboratif - Approche du Item-based

**Manipulation 1** : Fonctions du package recommenderlab

**Manipulation 2** : Exploration des données

**Manipulation 3 :** Préparation des données

**Manipulation 4 :** Préparation des données d’apprentissage et de test

**Manipulation 5 :** Construire le modèle de recommandation item-based

**Manipulation 6 :** Application du modèle de recommender sur l'ensemble de test

**Préparé par :** Nesrine Zemirli  
  
© Nesrine Zemirli et Hafed Benteftifa 2015-2016

Ce document ne peut être utilisé dans le cadre d’une formation, publication papier, site internet ou tout support sans mon accord express. Aucune reproduction, même partielle, ne peut être faite de ce document et de l'ensemble de son contenu : textes, images, etc. sans mon autorisation express. Pour toutes informations, communiquer avec moi sur info@degenio.com.

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Changement** |
| 20 Juin 2016 | 1.0 | Version initiale |
|  |  |  |
|  |  |  |

# Mise en contexte et présentation du projet

# Format : individuel

# Jeux de données : on a à notre disposition une source de données disponible sur RStudio

## Préliminaire

* R-studio est disponible.
* Utilisation du package « recommenderlab » Contenant lesfonction pour Développer et Tester les algorithmes de recommandation.

## Objectif

Construire un système de recommandation basé sur le filtrage collaboratif pour suggérer des films aux utilisateurs. Pour cela, nous allons appliquer une des approches du filtrage collaborative :

* **Item-based collaborative filtering** : le filtrage collaboratif basée sur les items recommande à un utilisateur les items qui ressemblent le plus à ce qu’il a déjà vue

# Description : MovieLens

Les données du TP sont issues de la collection des films de MovieLens, projet de l’université du Minnesota, représentent les avis des utilisateurs sur des films. Chaque film a un certain nombre de commentaires. Les avis (critique) sont des notes de 1 à 5 : 1 pour pas intéressant et 5 pour très intéressant.

Les données sont sur le site de MovieLens : <http://grouplens.org/datasets/movielens/100k/>

Cet ensemble contient les notes que les utilisateurs ont donnés aux films. Chaque ligne de MovieLense correspond à un utilisateur, et chaque colonne correspond à un film. Il y a : 943 utilisateurs pour 1664 évaluation.

Il y a donc plus de 1.500.000 combinaisons entre un utilisateur et un film. Par conséquent, le stockage de la matrice complète nécessiterait plus de 1.500.000 cellules. Cependant, les utilisateur n’ont pas évaluée tous les films (environ 20 films par utilisateur). Par conséquent, il y a moins de 100.000 notes, et la matrice est creuse.

## Manipulation 1 : Fonctions du package recommenderlab

### **Objectif** : présenter un aperçu rapide des fonctions du package « recommenderlab »

## Charger le packages

#### Tout d'abord, nous allons l'installer, si on n'a pas déjà fait :

**if**(**!**"recommenderlab" %in% rownames(installed.packages())){ install.packages("recommenderlab")}

Maintenant, nous pouvons charger le package. Lorsque, en utilisant la fonction d'aide, nous pouvons jeter un coup d'oeil à sa documentation : help(package ="recommenderlab")

library("recommenderlab")

help(package ="recommenderlab")

Les données manipulé dans ce TP contiennent des composantes aléatoires.

Pour ne pas reproduire le même résultat on va utiliser set.seed().

​

Pratique lorsqu'on fait des simulations afin que ce ne soient pas les mêmes à chaque fois.

set.seed(1)

## Jeux de données

Comme beaucoup d'autres packages R, recommenderlab contient des ensembles de données qui peuvent être utilisé directement sous R :

data\_package <- data(package = "recommenderlab")

data\_package$results[, "Item"]

Le résultat sur la console devrait être :

Jester5k, MSWeb, and MovieLense

## Charger le jeu de données MovieLens

data(MovieLense)

MovieLense

Le résultat sur la console devrait être :

943 x 1664 rating matrix of class ‘realRatingMatrix’ with 99392 ratings

## La classe de rating matrice

Dans cette section, nous allons faire une exploration sommaire MovieLens pour se familliariser avec les fonctions du "recommenderlab"

class(MovieLense)

## [1] "realRatingMatrix"

## attr(,"package")

## [1] "recommenderlab"

La classe realRatingMatrix contient des matrices creuses de ratings.

Jetons un regard sur les méthodes que nous pouvons appliquer sur les objets de cette classe :

methods(class = class(MovieLense))

[1] [ [<- binarize

[4] calcPredictionAccuracy coerce colCounts

[7] colMeans colSds colSums

[10] denormalize dim dimnames

[13] dimnames<- dissimilarity evaluationScheme

[16] getData.frame getList getNormalize

[19] getRatingMatrix getRatings getTopNLists

[22] image normalize nratings

[25] Recommender removeKnownRatings rowCounts

[28] rowMeans rowSds rowSums

[31] sample show similarity

Certains procédés qui sont applicables à des matrices ont été redéfinies de manière plus optimale. Par exemple, nous pouvons utiliser "dim" pour extraire le nombre de lignes et de colonnes, et "colSums" pour calculer la somme de chaque colonne.

En outre, il existe de nouvelles méthodes qui sont spécifiques pour les systèmes de recommandation.

Habituellement, matrices de rating sont des matrices creuses. Pour cette raison, la classe realRatingMatrix supporte un stockage compact des matrices creuses.

##### **Comparons la taille de MovieLense avec la matrice R correspondant*:***

object.size(MovieLense)

1388448 bytes

object.size(as(MovieLense, "matrix"))

12740464 bytes

## Calcule des similarités de matrices

Les algorithmes de filtrage collaboratif sont basés sur la mesure de la similitude entre les utilisateurs ou entre les items.

A cet effet, recommenderlab contient la fonction de similarité.

Les méthodes prises en chargé pour calculer les similitudes sont cosinus, pearson et jaccard.

Par exemple, nous pourrions déterminer la similarité des cinq premiers utilisateurs.

**Calculons ce en utilisant la distance cosinus:**

similarity\_users <- similarity(MovieLense[1:4, ], method =

"cosine", which = "users")

Le résultat sera

similarity\_users

1 2 3

2 0.16893670

3 0.03827203 0.09706862

4 0.06634975 0.15310468 0.33343036

Nous pouvons également convertir similarity\_users dans une matrice et le visualiser:

as.matrix(similarity\_users)

Le résultat sear :

-------------------------------

1 2 3 4

------- ------- ------- -------

0 0.1689 0.03827 0.06635

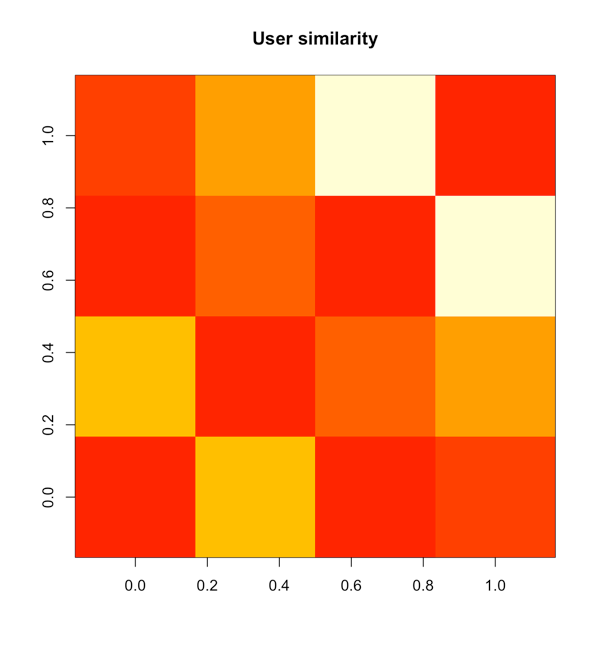
0.1689 0 0.09707 0.1531

0.03827 0.09707 0 0.3334

0.06635 0.1531 0.3334 0

En utilisant image, nous pouvons visualiser la matrice. Chaque rangée et chaque colonne correspond à un utilisateur, et chaque cellule correspond à la similarité entre deux utilisateurs :

image(as.matrix(similarity\_users), main = "User similarity")



En utilisant la même approche, nous pouvons calculer et de visualiser la similitude entre les quatre premiers items :

similarity\_items <- similarity(MovieLense[, 1:4], method ="cosine", which = "items")

as.matrix(similarity\_items)

Le résultat sera :

Toy Story (1995) GoldenEye (1995) Four Rooms (1995) Get Shorty (1995)

Toy Story (1995) 0.0000000 0.4023822 0.3302448 0.4549379

GoldenEye (1995) 0.4023822 0.0000000 0.2730692 0.5025708

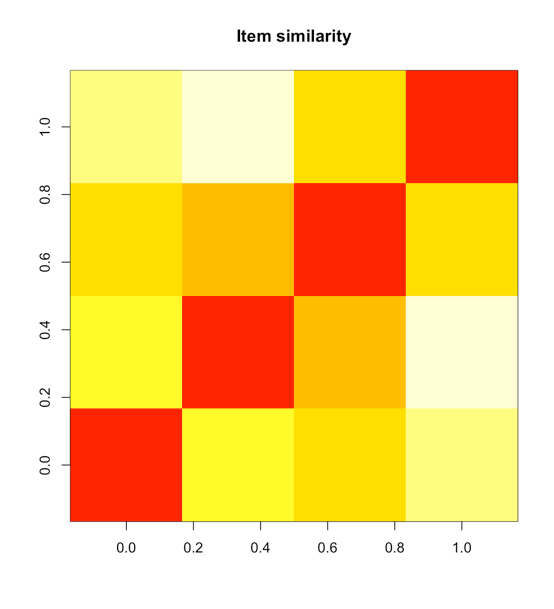
Four Rooms (1995) 0.3302448 0.2730692 0.0000000 0.3248664

Get Shorty (1995) 0.4549379 0.5025708 0.3248664 0.0000000

Visualiser la matrice de similarité

# Visualiser

image(as.matrix(similarity\_items), main = "Item similarity")



## Les algorithmes de recommandation

Le package recommenderlab contient quelques options pour l'algorithme de recommandation. Nous pouvons afficher le modèle applicable au realRatingMatrix objets en utilisant Registry recommender $ get\_entries:

recommender\_models <- recommenderRegistry$get\_entries(dataType =

"realRatingMatrix")

Pour une belle visualisation des résultats, nous allons utiliser le package pander

install.packages("pander")

library(pander)

Afficher la liste des algorithmes

df\_models <- data.frame(model = names(recommender\_models))

pander(df\_models)

Le résultat sera :

----------------------------

model

----------------------------

IBCF\_realRatingMatrix

POPULAR\_realRatingMatrix

RANDOM\_realRatingMatrix

RERECOMMEND\_realRatingMatrix

SVD\_realRatingMatrix

SVDF\_realRatingMatrix

UBCF\_realRatingMatrix

Ce qui nous intéresse c'est "IBCD"et"UBCF"

df\_parameters <- data.frame(

parameter = names(recommender\_models$IBCF\_realRatingMatrix$parameters),

default = unlist(recommender\_models$IBCF\_realRatingMatrix$parameters)

)

rownames(df\_parameters) <- NULL

pander(head(df\_parameters))

# ou simplement avec

#recommender\_models$IBCF\_realRatingMatrix$parameters



## Manipulation 2 : Exploration des données MovieLens

## Dimension

dim(MovieLense)

# Résultat

## [1] 943 1664

## Structure

str(MovieLense)

# Résultat

Formal class 'realRatingMatrix' [package "recommenderlab"] with 2 slots

..@ data :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots

.. .. ..@ i : int [1:99392] 0 1 4 5 9 12 14 15 16 17 ...

.. .. ..@ p : int [1:1665] 0 452 583 673 882 968 994 1386 1605 1904 ...

.. .. ..@ Dim : int [1:2] 943 1664

.. .. ..@ Dimnames:List of 2

.. .. .. ..$ : chr [1:943] "1" "2" "3" "4" ...

.. .. .. ..$ : chr [1:1664] "Toy Story (1995)" "GoldenEye (1995)" "Four Rooms (1995)" "Get Shorty (1995)" ...

.. .. ..@ x : num [1:99392] 5 4 4 4 4 3 1 5 4 5 ...

.. .. ..@ factors : list()

..@ normalize: NULL

## Explorer les valeurs des ratings

Convertir la matrice dans un vecteur et d'explorer ses valeurs :

vector\_ratings <- as.vector(MovieLense@data)

unique(vector\_ratings)

5 4 0 3 1 2

Le résultat sera :

5 4 0 3 1 2

Visualiser sous forme de tableau

table\_ratings <- table(vector\_ratings)

df\_ratings <- data.frame(

rating = names(table\_ratings),

occurrences = as.vector(table\_ratings)

)

pander(head(df\_ratings))

Résultat :

----------------------

rating occurrences

-------- -------------

0 1469760

1 6059

2 11307

3 27002

4 33947

5 21077

Selon la documentation, une note égale à 0 représente une valeur manquante, afin que nous puissions les supprimer de notes de vecteur :

vector\_ratings <- vector\_ratings[vector\_ratings != 0]

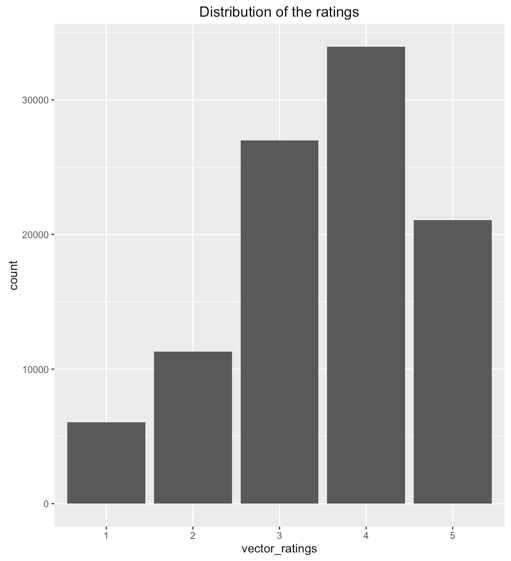
## Visualiser graphiquement les valeurs des ratings

vector\_ratings <- factor(vector\_ratings)

library("ggplot2")

qplot(vector\_ratings) + ggtitle("Distribution of the ratings")

Le résultat sera similaire au suivant :



**Ce que l’on peut déduite : La plupart des notes sont supérieures à 2, et le plus commun est 4.**

## Explorer quels films ont été visionnés

A partir de MovieLense, nous allons extraire des statistique en utilisant des méthodes suivantes:

* colCounts: Ceci est le nombre de valeurs non manquantes pour chaque colonne
* colMeans: Ceci est la valeur moyenne pour chaque colonne

Par exemple, quels sont les films les plus vus ? Nous pouvons utiliser colCounts pour cette objectif.

Tout d'abord, nous allons compter les vues pour chaque film :

views\_per\_movie <- colCounts(MovieLense)

Ensuite, nous pouvons trier les films par nombre de vues :

table\_views <- data.frame(

movie = names(views\_per\_movie),

views = views\_per\_movie

)

table\_views <- table\_views[order(table\_views$views, decreasing = TRUE), ]

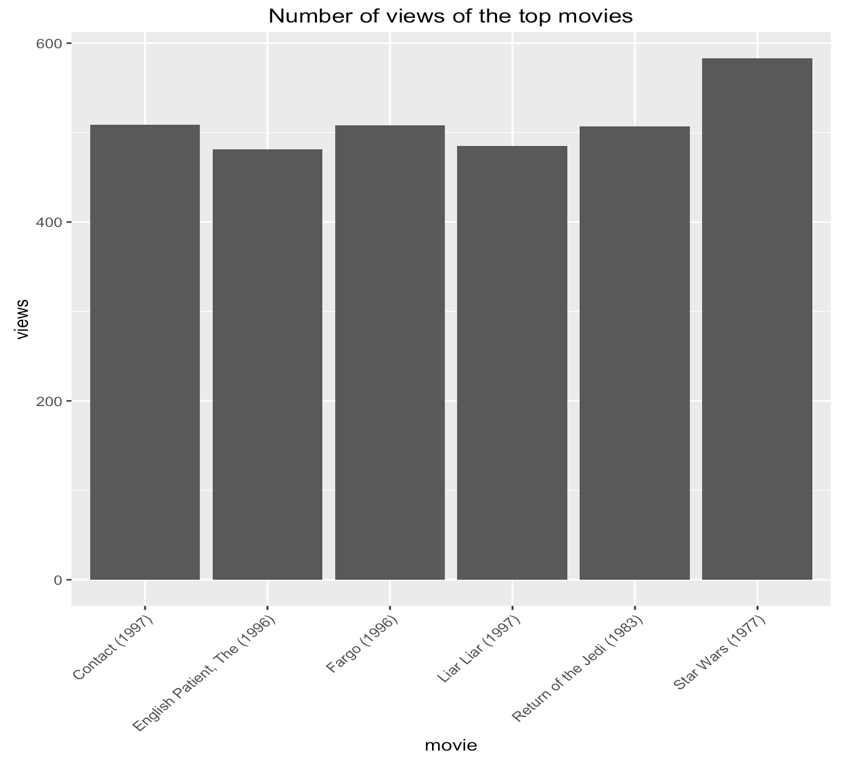
Visualiser

ggplot(table\_views[1:6, ], aes(x = movie, y = views)) +

geom\_bar(stat="identity") + theme(axis.text.x =

element\_text(angle = 45, hjust = 1)) + ggtitle("Number of views

of the top movies")

 Dans le tableau, vous pouvez remarquer que Star Wars (1977) est le film le plus vu, dépassant les autres d'environ 100 vues.

## Explorer les notes moyennes

Nous pouvons identifier les meilleurs films notés en calculant la note moyenne de chacun d'eux. A cet effet, nous pouvons utiliser colMeans; il ignore automatiquement les 0s, car ils représentent des valeurs manquantes. Jetons un coup d'œil à la distribution de la moyenne des rating film :

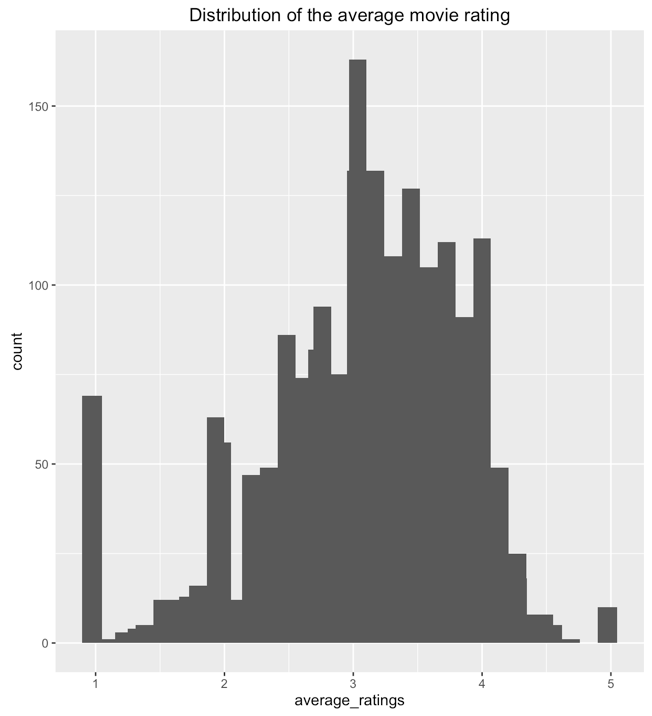
average\_ratings <- colMeans(MovieLense)

Visualiser

qplot(average\_ratings) + stat\_bin(binwidth = 0.1) +

ggtitle("Distribution of the average movie rating")

Le résultat sera similaire au suivant :



La valeur la plus élevée est d'environ 3, et il y a quelques films dont la notation est 1 ou 5. Probablement, la raison est que ces films ont reçu une note de seulement quelques personnes, donc il ne faut pas les prendre en compte. Nous pouvons supprimer les films dont le nombre de vues est inférieur à un seuil défini, par exemple, en dessous de 100:

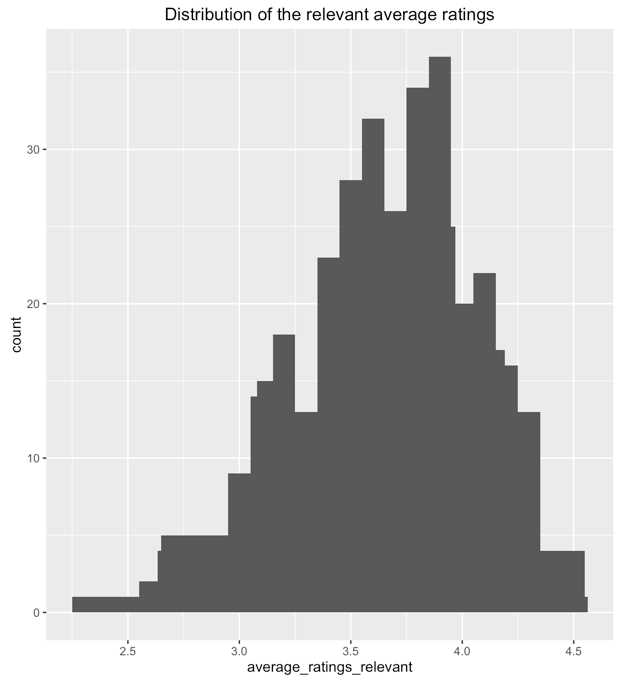
Filtrer les ratings

average\_ratings\_relevant <- average\_ratings[views\_per\_movie > 100]

Visualiser

qplot(average\_ratings\_relevant) + stat\_bin(binwidth = 0.1) + ggtitle(paste("Distribution of the relevant average ratings"))

Le résultat sera similaire à :



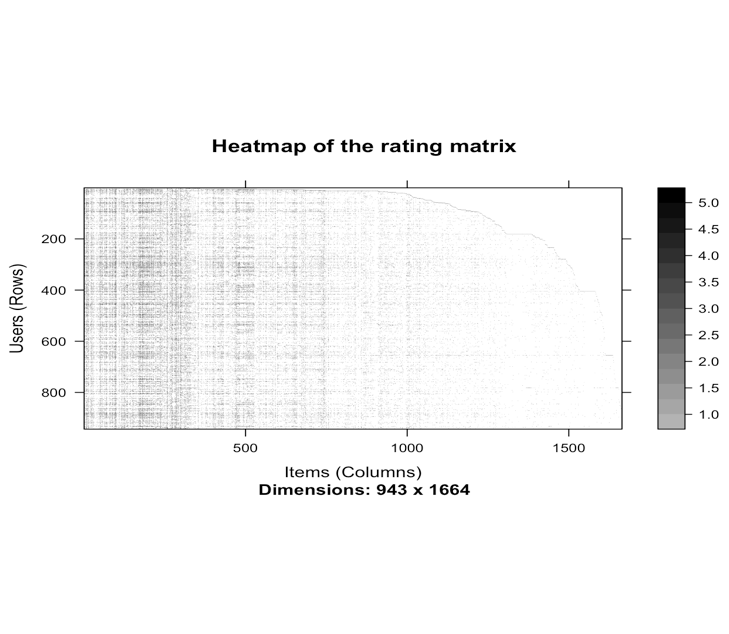
## Visualiser la matrice

Nous pouvons visualiser la matrice en construisant une carte de chaleur dont les couleurs représentent les ratins. Chaque ligne de la matrice correspond à un utilisateur, chaque colonne pour un film, et chaque cellule à sa valeur de rating. A cet effet, nous pouvons utiliser la méthode générique: image. Le package recommenderlab redéfini l'image de la méthode pour les objets realRatingMatrix.

Construisons l'heatmap utilisant l'image:

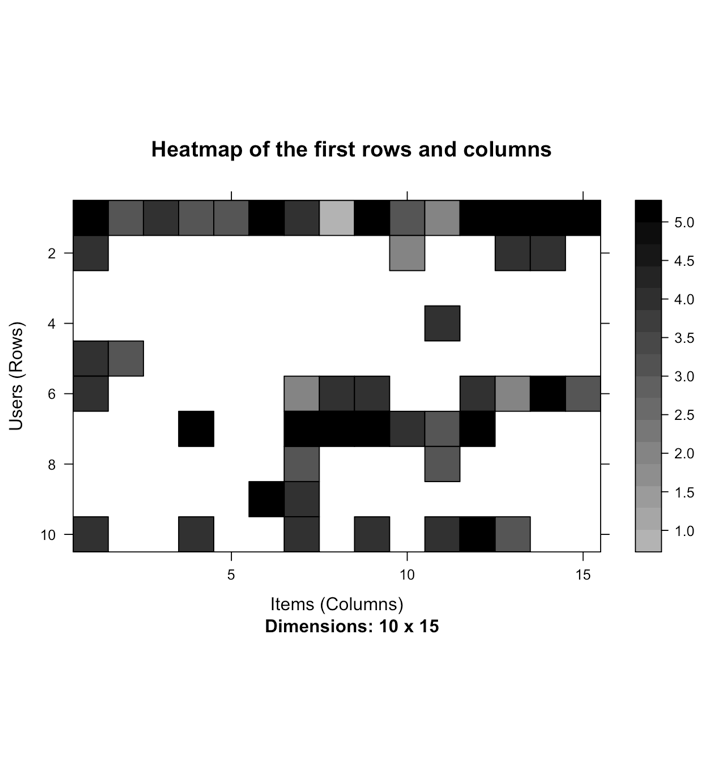
image(MovieLense, main = "Heatmap of the rating matrix")

Le résultat sera similaire à :



Si on veut zoomer et voir l’espace de la matrice avec une dimension réduite :

image(MovieLense[1:10, 1:15], main = "Heatmap of the first rows and columns")



## Manipulation 3 : Préparation des données

**Objectif**: préparer les données à utiliser par les algorithmes de recommandation

**Démarche**:

1. Sélectionnez les données pertinentes.
2. Normaliser les données.

## Sélection de données les plus pertinentes

Lorsque nous avons exploré les données, nous avons remarqué que la table contient

* Les films qui ont été vus que quelques fois. Leurs notes pourraient être biaisée en raison du manque de données.
* Les utilisateurs qui ont évalué seulement quelques films. Leurs notes pourraient être biaisés.

Nous avons besoin de déterminer le nombre minimum d'utilisateurs par film et vice versa.

Nous allons définir ratings\_movies contenant la matrice que nous allons utiliser. Ça prend compte :

* Les utilisateurs qui ont noté au moins 50 films
* Les films qui ont été regardés au moins 100 fois

Les points précédents sont définis dans le code suivant :

ratings\_movies <- MovieLense[rowCounts(MovieLense) > 50,

colCounts(MovieLense) > 100]

ratings\_movies

Le résultat est :

560 x 332 rating matrix of class ‘realRatingMatrix’ with 55298 ratings.

## Exploration de données les plus pertinentes

En utilisant la même approche que nous avons fait dans la section précédente, nous allons visualiser le top 2 pour cent des utilisateurs et des films dans la nouvelle matrice :

# visualize the top matrix

min\_movies <- quantile(rowCounts(ratings\_movies), 0.98)

min\_users <- quantile(colCounts(ratings\_movies), 0.98)

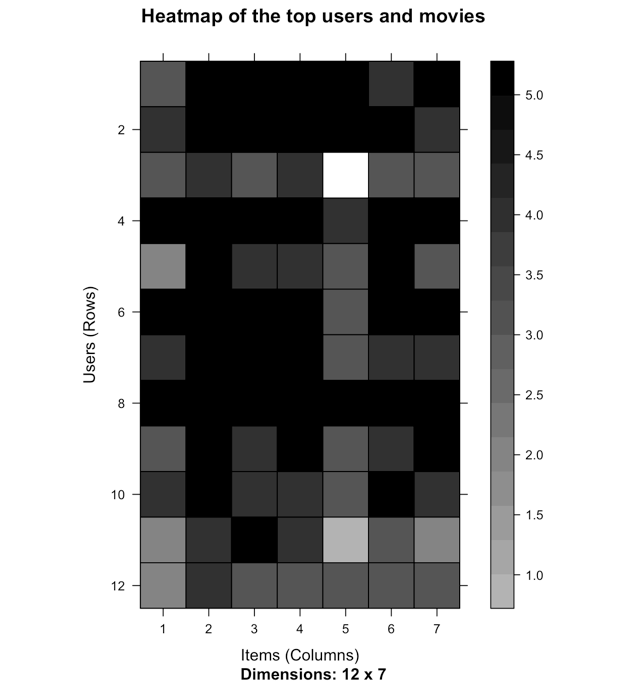
Visualiser graphiquement :

image(ratings\_movies[rowCounts(ratings\_movies) > min\_movies,

colCounts(ratings\_movies) > min\_users], main = "Heatmap of the top

users and movies")

Le résultat sera :



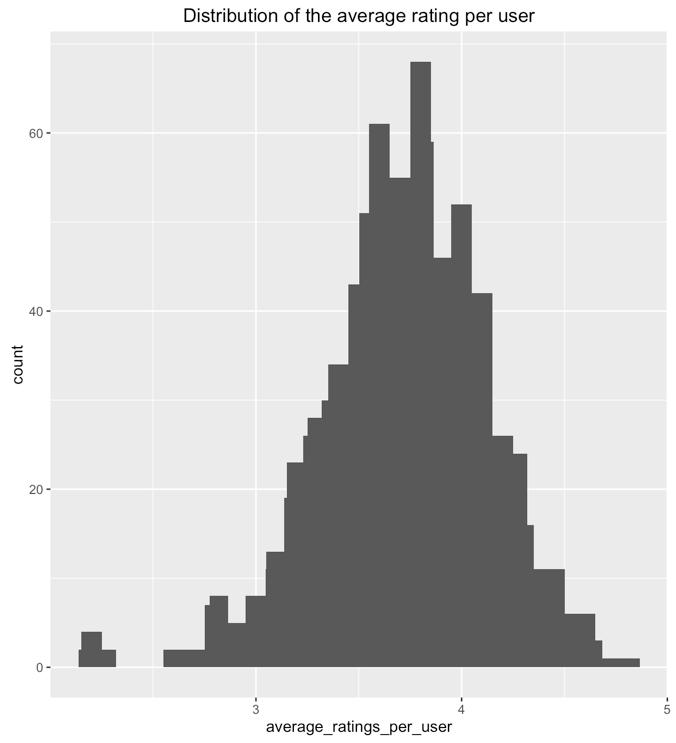
Comme nous l'avons déjà remarqué, certaines lignes sont plus sombres que les autres. Cela pourrait signifier que certains utilisateurs donnent des notes plus élevées à tous les films. Cependant, nous avons visualisé les meilleurs films seulement. Afin d'avoir un aperçu de tous les utilisateurs, nous allons jeter un coup d'œil à la distribution de la valeur du rating moyenne par l'utilisateur :

average\_ratings\_per\_user <- rowMeans(ratings\_movies)

Visualiser

qplot(average\_ratings\_per\_user) + stat\_bin(binwidth = 0.1) +

ggtitle("Distribution of the average rating per user")



## Normaliser les données

Avoir les utilisateurs qui donnent des notes élevées (ou faible) à tous leurs films pourrait biaiser les résultats.

Nous pouvons éliminer cet effet en normalisant les données de telle sorte que la note moyenne de chaque utilisateur est 0. On utilise pour cela la fonction suivante :

ratings\_movies\_norm <- normalize(ratings\_movies)

Regardons à la note moyenne des ratings par utilisateurs:

sum(rowMeans(ratings\_movies\_norm) > 0.00001)

#Résultat

## [1] 0

Visualiser la matrice normalisée

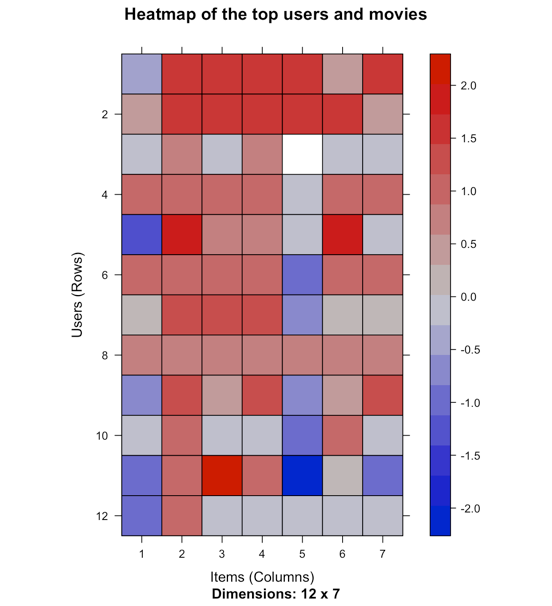
# visualize the normalized matrix

image(ratings\_movies\_norm[rowCounts(ratings\_movies\_norm) > min\_movies,

colCounts(ratings\_movies\_norm) > min\_users], main = "Heatmap of the

top users and movies")

Le résultat sera similaire à :



## Manipulation 4 : Préparation des données d’apprentissage et de test

**Objectif**: construire le modèle en utilisant une partie de l'ensemble de données MovieLense comme ensemble d’apprentissage et l'appliquer sur l'autre partie (le jeu de test).

**Démarche**

Les deux ensembles sont les suivantes :

• Jeu d’apprentissage : Cet ensemble comprend les utilisateurs à partir de laquelle le modèle apprend

• Jeu de test : Cet ensemble comprend les utilisateurs auxquels nous recommandons des films

L'algorithme normalise automatiquement les données, afin que nous puissions utiliser ratings\_movies qui contient les utilisateurs et les films de MovieLense pertinents. Nous avons défini ratings\_movies dans la section précédente qui contient les données pertinentes à prendre en considération : le sous-ensemble d'utilisateurs MovieLense qui ont classé au moins 50 films et des films qui ont été notés au moins 100 fois.

## Donnée d’apprentissage

Premièrement, nous définissons au hasard le vecteur which\_train qui est vrai pour les utilisateurs dans l'ensemble d’apprentissage et FALSE pour les autres. Nous allons définir la probabilité de la formation définie comme 80 pour cent :

which\_train <- sample(x = c(TRUE, FALSE), size = nrow(ratings\_movies),

replace = TRUE, prob = c(0.8, 0.2))

head(which\_train)

## [1] TRUE TRUE TRUE FALSE TRUE FALSE

Puis nous construisons les jeux de données respectivement

#Let's define the training and the test sets:

recc\_data\_train <- ratings\_movies[which\_train, ]

recc\_data\_test <- ratings\_movies[!which\_train, ]

recc\_data\_train

451 x 332 rating matrix of class ‘realRatingMatrix’ with 43846 ratings.

recc\_data\_test

109 x 332 rating matrix of class ‘realRatingMatrix’ with 11452 ratings.

## Manipulation 5 : Construire le modèle de recommandation Item-Based Collaboratif Filtering

Objectif : construire la fonction qui implémente l’algorithme de recommandation modèles, ses entrées sont les suivantes :

* Données: Ceci est l'ensemble de données d’apprentissage
* Méthode: Ceci est le nom de la technique
* Paramètres: Ce sont des paramètres optionnels de la technique

## Fonction du Item-based collaboratif

La fonction que l’on va utiliser est la suivante :

recommender\_models <- recommenderRegistry$get\_entries(dataType = "realRatingMatrix")

Pour connaitre ses paramètres :

list\_parameters <- recommender\_models$IBCF\_realRatingMatrix$parameters

df\_paramters <- data.frame(

parameter = names(list\_parameters),

default = unlist(list\_parameters)

)

rownames(df\_paramters) <- NULL

pander(df\_paramters)

Résultat :

------------------------------

parameter default

-------------------- ---------

k 30

method Cosine

normalize center

normalize\_sim\_matrix FALSE

alpha 0.5

na\_as\_zero FALSE

------------------------------

## Construction du modèle de recommandation

Certains paramètres importants sont les suivants :

* **k** : Dans la première étape, l'algorithme calcule les similitudes entre chaque paire d'items. Ensuite, pour chaque item, il identifie ses k items les plus similaires et les stocke.
* **Méthode** : Ceci est la fonction de similarité. Par défaut, il est Cosinus. Un autre option populaire est pearson.

nous fixons k = 30, ce qui est la valeur par défaut. Nous sommes maintenant prêt à construire un modèle de recommender, comme cela :

recc\_model <- Recommender(data = recc\_data\_train,

method = "IBCF",

parameter = list(k = 30))

recc\_model

class(recc\_model)

Résultat sera :

Recommender of type ‘IBCF’ for ‘realRatingMatrix’

learned using 451 users.

'Recommender'

## Manipulation 6 : Application du modèle de recommender sur l'ensemble de test

**Objectif**: Recommander des films aux aux utilisateurs dans l'ensemble de test.

**Démarche**: Nous allons définir n\_recommended qui spécifie le nombre d'éléments de recommander à chaque utilisateur.

1. Pour chaque utilisateur, l'algorithme extrait ses films notés.
2. Pour chaque film, il identifie tous ses éléments similaires, à partir de la matrice de similarité.
3. Ensuite, l'algorithme classe chaque item similaire de cette manière:

* Extraire le rating d’utilisateur pour chaque item associé à cet item. La valeur du rating est utilisée en tant que poids.
* Extraire la similitude de l’item avec les items qui lui sont associé
* Multiplier chaque poids avec la similitude connexe.
* Faire la somme

Ensuite, l'algorithme identifie les **n** premières recommandations

## Appliquer le modèle de recommandation sur le jeu de test

Dans notre ça nous souhaitons recommander à chaque utilisateur 6 films.

# set variable value

n\_recommended <- 6

recc\_predicted <- predict(object = recc\_model, newdata = recc\_data\_test, n = n\_recommended)

recc\_predicted

Résultat sera :

Recommendations as ‘topNList’ with n = 6 for 109 users.

## Inspecter les résultats

L’objet recc\_predicted contient les recommandations. Inspecter sa structure :

class(recc\_predicted)

slotNames(recc\_predicted)

## [1] "topNList"

## [1] "items" "itemLabels" "n"

Les slotNames sont:

* items : Ceci est la liste avec les indices des items recommandés pour chaque utilisateur
* itemLabels: Ceci est le nom des items
* n: Ceci est le nombre de recommandations

Par exemple, ce sont les recommandations pour le premier utilisateur :

recc\_predicted@items[[1]]

278 85 165 271 193 97

Nous pouvons extraire les noms des films recommandés avec recc\_predicted@item

recc\_user\_1 <- recc\_predicted@items[[1]]

movies\_user\_1 <- recc\_predicted@itemLabels[recc\_user\_1]

# visualiser

table\_user\_1 <- data.frame(index = recc\_user\_1,

movie = movies\_user\_1)

pander(table\_user\_1)

Le résultat sera :

index movie

------- --------------------------

278 Killing Fields, The (1984)

85 Reservoir Dogs (1992)

165 Contact (1997)

271 Annie Hall (1977)

193 L.A. Confidential (1997)

97 Brazil (1985)

----------------------------------

Nous pouvons définir une matrice avec les recommandations pour chaque utilisateur :

recc\_matrix <- sapply(recc\_predicted@items, function(x){

colnames(ratings\_movies)[x]

})

dim(recc\_matrix)

# 6 109

Puis, visualiser les recommandations pour les 4 premiers utilisateurs :

pander(recc\_matrix[, 1:4])

--------------------------------------------------------------------

5 7 8

-------------------------- ------------------------- ---------------

Killing Fields, The (1984) Good Will Hunting (1997) Babe (1995)

Reservoir Dogs (1992) As Good As It Gets (1997) Mighty Aphrodite (1995)

Contact (1997) This Is Spinal Tap (1984) Ed Wood (1994)

Annie Hall (1977) Trainspotting (1996) Hoop Dreams (1994)

L.A. Confidential (1997) Titanic (1997) Fargo (1996)

Brazil (1985) Amistad (1997) Lone Star (1996)

## Les films les plus recommandés

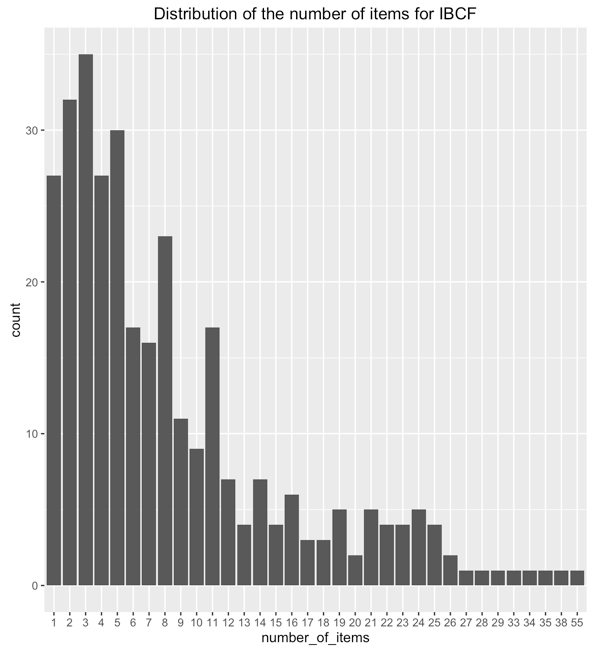
Maintenant, nous pouvons identifier les films les plus recommandés. A cet effet, nous allons définir un vecteur avec toutes les recommandations, et nous allons construire un tracé de fréquence:

number\_of\_items <- factor(table(recc\_matrix))

chart\_title <- "Distribution of the number of items for IBCF"

Visualiser

qplot(number\_of\_items) + ggtitle(chart\_title)



La plupart des films ont été recommandés que quelques fois, et quelques films ont été recommandées à plusieurs reprises. Voyons quels sont les films les plus populaires :

number\_of\_items\_sorted <- sort(number\_of\_items, decreasing = TRUE)

number\_of\_items\_top <- head(number\_of\_items\_sorted, n = 4)

table\_top <- data.frame(names(number\_of\_items\_top),

number\_of\_items\_top)

table\_top

Le résultat sera :

---------------------------------------------------------------------------------------------------

&nbsp; names.number\_of\_items\_top.

----------------------------------------------------------------------- ----------------------------

\*\*Mr. Smith Goes to Washington (1939)\*\* Mr. Smith Goes to Washington (1939)

\*\*Babe (1995)\*\* Babe (1995)

\*\*Maltese Falcon, The (1941)\*\* Maltese Falcon, The (1941)

\*\*L.A. Confidential (1997)\*\* L.A. Confidential (1997)

---------------------------------------------------------------------------------------------------------

Table: Table continues below

-----------------------------------------------------------------------------------------------

&nbsp; number\_of\_items\_top

-------------------------------- ---------------------------------------------------------------

\*\*Mr. Smith Goes to Washington (1939)\*\* 55

\*\*Babe (1995)\*\* 38

\*\*Maltese Falcon, The (1941)\*\* 35

\*\*L.A. Confidential (1997)\*\* 34

-----------------------------------------------------------------------------------------------