

Cahier thématique

Recommandation de Musique Basé sur le filtrage Collaboratif

**Manipulation 1** : Description des données

**Manipulation 2** : Construire l’algorithme du Item-based Collaboratif filtering  
**Manipulation 3** : Appliquer l’algorithme de recommandation

**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 votre répertoire de travail

## Préliminaire

* R-studio est disponible.

## Objectif

Construire un système de recommandation basé sur le filtrage collaboratif pour suggérer des chansons d’artistes aux utilisateurs. Pour cela, nous allons appliquer l’approche du **Item-based collaborative filtering** :

* Recommander des chansons d’artiste similaires

# Description : LastFM

L'ensemble de données contient des informations sur les utilisateurs, leur sexe, leur âge, et les artistes qu'ils ont écoutés sur Last.FM.

Les données contiennent : 1257 user pour 286 items.

Les données sont extraites du site de LastFM: http://labrosa.ee.columbia.edu/millionsong/lastfm

## Manipulation 1 : Description des données

### Définir le répertoire de travail sous R

getwe()

setwd("~/folder/")

### Charger les données

data.germany <- read.csv(file="lastfm-matrix-germany.csv")

dim(data.germany)

### dimension

dim(data.germany)

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

[1] 1257 286

### Description des variables

names(iris)

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

[1] "user" "a.perfect.circle" "abba" "ac.dc"

[5] "adam.green" "aerosmith" "afi" "air"

[9] "alanis.morissette" "alexisonfire" "alicia.keys" "all.that.remains"

[13] "amon.amarth" "amy.macdonald" "amy.winehouse" "anti.flag"

1. Structure

str(data.gernamy)

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

'data.frame': 1257 obs. of 286 variables:

$ user : int 1 33 42 51 62 75 130 141 144 150 ...

$ a.perfect.circle : int 0 0 0 0 0 0 0 0 0 0 ...

$ abba : int 0 0 0 0 0 0 0 0 0 0 ...

$ ac.dc : int 0 0 0 0 0 0 0 0 0 0 ...

$ adam.green : int 0 1 0 0 0 0 0 0 0 0 ...

$ aerosmith : int 0 0 0 0 0 0 0 0 0 0 ...

$ afi : int 0 0 0 0 0 0 0 0 0 0 ...

$ air : int 0 0 0 0 0 0 0 0 0 0 ...

$ alanis.morissette : int 0 0 0 0 0 0 0 0 0 0 ...

### Afficher les données

# les premiers lignes

head(data.germany[,c(1,3:8)])

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

user abba ac.dc adam.green aerosmith afi air

1 1 0 0 0 0 0 0

2 33 0 0 1 0 0 0

3 42 0 0 0 0 0 0

4 51 0 0 0 0 0 0

5 62 0 0 0 0 0 0

6 75 0 0 0 0 0 0

## Manipulation 2 : Construire l’algorithme du Item-based Collaboratif filtering

1. **Préparation des données**

Dans filtrage collaboratif basé sur les items nous ne nous soucions pas vraiment des utilisateurs. Donc, la première chose que nous devrions faire est de supprimer la colonne d'utilisateur de nos données

*# Drop any column named "user"*

data.germany.ibs <- (data.germany[,!(names(data.germany) %in% c("user"))])

1. **Calculer la similarité item-item avec le cosinus**

Nous voulons ensuite calculer la similarité de chaque chanson avec le reste des chansons. Cela signifie que nous voulons comparer chaque colonne dans nos "data.germany.ibs" fixés avec chaque autre colonne dans le jeu de données de données. Plus précisément, nous allons comparer ce qui est connu comme le «Cosinus Similarity».

La similitude cosinus, en pratique prend le produit de la somme de la première et la deuxième colonne, et diviser par le produit de la racine carrée de la somme des carrés de chaque colonne.

La chose importante à savoir est le nombre obtenu représente la «similaire» la première colonne est avec la deuxième colonne.

Écrire la fonction suivante pour le produit Cosinus Similarity :

|  |
| --- |
| *# Create a helper function to calculate the cosine between two vectors*  getCosine <- function(x,y)  {  this.cosine <- sum(x\*y) / (sqrt(sum(x\*x)) \* sqrt(sum(y\*y)))  return(this.cosine)  } |

1. **Préparation des matrices item-item**

Nous allons créer un espace réservé dataframe qui va contenir la matrice item-item

*# Create a placeholder dataframe listing item vs. item*

holder <- matrix(NA, nrow=ncol(data.germany.ibs),ncol=ncol(data.germany.ibs),dimnames=list(colnames(data.germany.ibs),colnames(data.germany.ibs)))

data.germany.ibs.similarity <- as.data.frame(holder)

inspecter le contenu du data.germany.ibs.similarity

data.germany.ibs.similarity

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

|  |
| --- |
| a.perfect.circle abba ac.dc adam.green aerosmith afi  a.perfect.circle NA NA NA NA NA NA  abba NA NA NA NA NA NA  ac.dc NA NA NA NA NA NA  adam.green NA NA NA NA NA NA  aerosmith NA NA NA NA NA NA  afi NA NA NA NA NA NA |

1. Générer la matrice item-item avec les valeurs de similarités

# Lets fill in those empty spaces with cosine similarities

# Loop through the columns

for(i in 1:ncol(data.germany.ibs)) {

# Loop through the columns for each column

for(j in 1:ncol(data.germany.ibs)) {

# Fill in placeholder with cosine similarities

data.germany.ibs.similarity[i,j] <- getCosine(as.matrix(data.germany.ibs[i]),as.matrix(data.germany.ibs[j]))

}

}

# Back to dataframe

data.germany.ibs.similarity <- as.data.frame(data.germany.ibs.similarity)

1. Afficher la matrice de similarité item-item

data.germany.ibs.similarity

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

|  |
| --- |
| a.perfect.circle abba ac.dc adam.green aerosmith afi air alanis.morissette alexisonfire  a.perfect.circle 1.00000000 0.00000000 0.01791723 0.05155393 0.06277648 0.0000000 0.05175492 0.06071767 0.0000000  abba 0.00000000 1.00000000 0.05227877 0.02507061 0.06105625 0.0000000 0.01677890 0.02952693 0.0000000  ac.dc 0.01791723 0.05227877 1.00000000 0.11315371 0.17715300 0.0678942 0.07572991 0.03807625 0.0000000  adam.green 0.05155393 0.02507061 0.11315371 1.00000000 0.05663655 0.0000000 0.09338592 0.00000000 0.0000000  aerosmith 0.06277648 0.06105625 0.17715300 0.05663655 1.00000000 0.0000000 0.11371471 0.10005560 0.0000000  afi 0.00000000 0.00000000 0.06789420 0.00000000 0.00000000 1.0000000 0.00000000 0.05751973 0.0836242 |

## Manipulation 3 : Appliquer l’algorithme de recommandation

Nous sommes maintenant en mesure de faire des recommandations !

Nous regardons les 10 premiers voisins de chaque chanson qui seront recommandés aux utilisateurs qui écoutes ces chansons.

1. Nous commençons par créer un espace réservé :

data.germany.neighbours <- matrix(NA, nrow=ncol(data.germany.ibs.similarity),ncol=11,dimnames=list(colnames(data.germany.ibs.similarity)))

1. Afficher la matrice de similarité data.germany.neighbours

data.germany.neighbours

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

|  |
| --- |
| [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11]  a.perfect.circle NA NA NA NA NA NA NA NA NA NA NA  abba NA NA NA NA NA NA NA NA NA NA NA  ac.dc NA NA NA NA NA NA NA NA NA NA NA  adam.green NA NA NA NA NA NA NA NA NA NA NA  aerosmith NA NA NA NA NA NA NA NA NA NA NA  afi NA NA NA NA NA NA NA NA NA NA NA |

1. Trouver les items similaires

for(i in 1:ncol(data.germany.ibs))

{

data.germany.neighbours[i,] <- (t(head(n=11,rownames(data.germany.ibs.similarity[order(data.germany.ibs.similarity[,i],decreasing=TRUE),][i]))))

}

1. Afficher les recommandations

Recommandation<- as.data.frame(data.germany.neighbours)

Recommandation

En sortie sur la console, on devrait avoir un résultat similaire au suivant :

|  |
| --- |
| [,1] [,2] [,3]  a.perfect.circle "a.perfect.circle" "tool" "dredg"  abba "abba" "madonna" "robbie.williams"  ac.dc "ac.dc" "red.hot.chilli.peppers" "metallica"  adam.green "adam.green" "the.libertines" "the.strokes"  aerosmith "aerosmith" "u2" "led.zeppelin"  afi "afi" "funeral.for.a.friend" "rise.against" |