LOG6308 - Laboratoire 1 - Rapport

Erwan Marchand - 1659273

September 10, 2017

Contents

Presentation du laboratoire

Ce laboratoire a pour objectif de nous familiariser à l'estimation de votes d'utilisateurs par utilisation de filtre collaboratif mais également par une approche contenu. Afin de répondre aux questions de ce laboratoire, nous aurons à trouver par calcul les films les plus similaires à un film donné à l'aide de la mesure de cosinus ou de corrélation.

La remise de ce rapport contient 3 fichiers :

- Le fichier .Rmd qui est une version finale du code crée à l'aide de Rmarkup, elle contient les commentaires et résultats.
- Le fichier .pdf qui est le pdf généré via le fichier .Rmd.
- Le fichier .html qui est le html généré via le fichier .Rmd.

Ces 3 fichiers contiennent donc exactement les memes données, seul le format change afin de laisser la liberté de choix pour la correction.

Création des données

```
rm(list = ls())

library(Matrix)

u.data <- read.csv(file='u.data.csv', sep='|', header=T)
u.item <- read.csv(file='u.item.csv', sep='|', header=T)
u.user <- read.csv(file='u.user.csv', sep='|', header=T)

m.sparse <- sparseMatrix(u.data$user.id,u.data$item.id,x=u.data$rating)
rownames(m.sparse) <- paste('u', 1:nrow(m.sparse), sep='')
colnames(m.sparse) <- paste('i', 1:ncol(m.sparse), sep='')

m <- as.matrix(m.sparse)

m[m==0] <- NA


mx <- merge(u.user, u.data, 1)
```

1) Quelle est la moyenne des votes par profession ("job") et par âge?

Moyenne par profession

Le code ci-dessous nous permet d'obtenir la liste des moyenne de classement par profession, afin d'alléger le rapport, nous n'en affichons que 6. De meme pour la liste des moyennes de classement par age.

```
meanPerJob <-aggregate(mx[, 7], list(mx$job), mean)
colnames(meanPerJob) <- c("job","mean Rating")
head(meanPerJob)
```

job	mean Rating
administrator	3.635647
artist	3.653379
doctor	3.688889
educator	3.670621
engineer	3.541407
entertainment	3.441050

Moyenne par age

```
meanPerAge <-aggregate(mx[, 7], list(mx$age), mean)
colnames(meanPerAge) <- c("age","mean Rating")
head(meanPerAge)
```

age	mean Rating
7	3.767442
10	3.387097
11	2.925926
13	3.470825
14	3.375000
15	3.264484

2) Quels sont les 10 films les plus similaires à “Star Trek V: The Final Frontier (1989)” selon respectivement la mesure du cosinus et de la corrélation avec la matrice de votes

Afin de répondre a cette question, il est necessaire de creer plusieurs fonctions.

Creation des fonctions

Cosinus entre un vecteur v et chaque colonne de la matrice m

```
cosinus.vm <- function(v,m) {
  n <- sqrt(colSums(m^2))
  return ((v %*% m)/(n * sqrt(sum(v^2))))
}
```

Trouve les indexes des premieres ‘n’ valeurs maximales/minimales d’une matrice

```
max.nindex <- function(m, n=5) {
  i <- order(m, decreasing=TRUE)
  return(i[1:n])
}

min.nindex <- function(m, n=5) {
```

```
  i <- order(m)
  return(i[1:n])
}
```

Les 10 films les plus similaires selon la mesure du cosinus avec la matrice de votes

Sachant que le film le plus similaire a Star Trek V selon la matrice de vote sera toujours lui meme, il faut penser a le supprimer du resultat.

Dans un premier temps, on recupere l'indice du film considere ainsi que les notes qui y sont associes

```
name.star.trek <- "Star Trek V: The Final Frontier (1989)"
indice.star.trek <- u.item[u.item$movie.title == name.star.trek, ]$movie.id
notes.star.trek <- m.sparse[, indice.star.trek]
```

Puis on effectue les calculs pour la mesure du cosinus

```
wcos <- cosinus.vm(notes.star.trek, m.sparse)
similarMoviesCos.index <- max.nindex(wcos, 11)
similarMoviesCos.index <- similarMoviesCos.index[similarMoviesCos.index != indice.star.trek]
similarMoviesCos.names <- u.item[similarMoviesCos.index, ]$movie.title
similarMoviesCos.names <- data.frame(similarMoviesCos.names)
colnames(similarMoviesCos.names) <- c(paste("Most similar movies from", name.star.trek,
    sep = " : "))
similarMoviesCos.names
```

Most similar movies from : Star Trek V: The Final Frontier (1989)
Star Trek: The Motion Picture (1979)
Star Trek VI: The Undiscovered Country (1991)
Star Trek III: The Search for Spock (1984)
Star Trek IV: The Voyage Home (1986)
Star Trek: The Wrath of Khan (1982)
Stargate (1994)
Star Trek: Generations (1994)
Die Hard 2 (1990)
Escape from New York (1981)
Conan the Barbarian (1981)

Les 10 films les plus similaires selon la mesure de la correlation avec la matrice de votes

```
wcor <- as.vector(cor(notes.star.trek, as.matrix(m.sparse)))
similarMoviesCor.index <- max.nindex(wcor, 11)
similarMoviesCor.index <- similarMoviesCor.index[similarMoviesCor.index != indice.star.trek]
similarMoviesCor.names <- u.item[similarMoviesCor.index, ]$movie.title
similarMoviesCor.names <- data.frame(similarMoviesCor.names)
colnames(similarMoviesCor.names) <- c(paste("Most similar movies from", name.star.trek,
    sep = " : "))
similarMoviesCor.names
```

Most similar movies from : Star Trek V: The Final Frontier (1989)
Star Trek: The Motion Picture (1979)
Star Trek VI: The Undiscovered Country (1991)
Star Trek III: The Search for Spock (1984)
Star Trek IV: The Voyage Home (1986)
Star Trek: The Wrath of Khan (1982)
Stargate (1994)
Star Trek: Generations (1994)
Die Hard 2 (1990)
Escape from New York (1981)
Conan the Barbarian (1981)

On peut constater qu'on obtient dans les 2 cas la meme liste de films.

3) Utilisez une approche item-item pour calculer le vote au film "Star Trek V: The Final Frontier (1989)" des utilisateurs qui n'ont pas de vote pour celui-ci. Prenez les 20 voisins les plus rapprochés selon la distance euclidienne et utilisez les cosinus comme poids. Si aucun vote commun existe, alors la valeur prédite est fixée à NA

Recuperation des 20 voisins les plus rapprochés selon la distance euclidienne du film "Star Trek V"

Il est tout d'abord important de noter que nous devons considerer les 21 voisins les plus proches, sachant que le voisin le plus proche de Star Trek V sera toujours lui-meme tel que mentionne precedemment.

```
n.voisins <- 20 + 1
```

On peut ensuite recuperer le nombre de votes commun entre Star Trek V et chaque film, puis la distance euclidienne entre les votes de Star Trek et chaque film.

```
votes.communs <- (colSums((m.sparse[,indice.star.trek] * m.sparse) > 0))
distance.star.trek <- sqrt(colSums((m.sparse[,indice.star.trek] - m.sparse)^2))
i.distance.star.trek <- min.nindex(distance.star.trek, n.voisins)
i.distance.star.trek <- i.distance.star.trek[i.distance.star.trek!=indice.star.trek]
```

Verifications des données récupérées

On peut maintenant verifier que la distance entre Star Trek et le meme est bien de 0.

```
distance.star.trek[indice.star.trek]
```

```
## i450
##    0
```

Puis on peut regarder le nombre de votes communs entre les 20 films les plus proches de Star Trek V et Star Trek V.

```
temp <- as.matrix(votes.communs[i.distance.star.trek])
temp <- data.frame(temp)
colnames(temp) <- c("nombre de votes communs")
```

```
rownames(temp) <- u.item[gsub('i','',rownames(temp)),]$movie.title
temp
```

	nombre de votes communs
Robocop 3 (1993)	8
Mr. Magoo (1997)	5
Vampire in Brooklyn (1995)	9
Jerky Boys, The (1994)	3
Across the Sea of Time (1995)	3
Blood For Dracula (Andy Warhol's Dracula) (1974)	3
Fair Game (1995)	7
Phat Beach (1996)	2
S.F.W. (1994)	2
Window to Paris (1994)	1
Farmer & Chase (1995)	1
Fire on the Mountain (1996)	1
Open Season (1996)	1
Tough and Deadly (1995)	1
Daniel Defoe's Robinson Crusoe (1996)	1
Blood Beach (1981)	4
Paris Was a Woman (1995)	1
Bad Girls (1994)	4
Mad Dog Time (1996)	1
Shooter, The (1995)	2

On peut s'appercevoir que par chance, aucun des 20 films les plus proche en distance euclidienne de Star Trek V n'a aucun vote en commun avec celui-ci.

Estimation des votes des utilisateurs n'ayant pas votés pour "Star Trek V"

On peut enfin effectuer les opérations nécessaires afin de calculer les votes des utilisateurs n'ayant pas votés pour le film "Star Trek V"

```
wcos.voisin <- t(as.matrix(cosinus.vm(m.sparse[, indice.star.trek], m.sparse[,
    i.distance.star.trek])))
m.sparse.star.trek.na <- as.matrix(m.sparse[which(is.na(m[, indice.star.trek])),
    i.distance.star.trek])  # on ne considere pas les utilisateurs ayant deja votés pour star trek

m.star.trek.na <- as.matrix(m.sparse.star.trek.na)
m.star.trek.na[m.star.trek.na == 0] <- NA

wcos.sums.star.trek.na <- m.star.trek.na
wcos.sums.star.trek.na[!is.na(wcos.sums.star.trek.na)] <- 1
wcos.sums.star.trek.na[is.na(wcos.sums.star.trek.na)] <- 0
wcos.sums.star.trek.na <- abs(wcos.sums.star.trek.na %*% (as.matrix(wcos.voisin)))
wcos.sums.star.trek.na[wcos.sums.star.trek.na == 0] <- NA

mean.star.trek = mean(m[, indice.star.trek], na.rm = TRUE)

means.voisins.na = colMeans(m.star.trek.na, na.rm = TRUE)  # moyenne pour chaque voisin sans vote des u
means.voisins.na[is.nan(means.voisins.na)] <- NA
```

```r
temp <- t(t(m.star.trek.na) - means.voisins.na)
temp[is.na(temp)] <- 0
notes.star.trek.predicted.na <- temp %*% wcos.voisin/wcos.sums.star.trek.na
notes.star.trek.predicted.na[notes.star.trek.predicted.na == 0] <- NA
notes.star.trek.predicted.na <- mean.star.trek + notes.star.trek.predicted.na

notes.star.trek.predicted.na.no.na <- merge(rownames(notes.star.trek.predicted.na)[!is.na(notes.star.tr
    as.matrix(notes.star.trek.predicted.na[!is.na(as.matrix(notes.star.trek.predicted.na))]),
    by = "row.names")

colnames(notes.star.trek.predicted.na.no.na) <- c(" ", "Identifiant utilisateur",
    "vote prédit")
notes.star.trek.predicted.na.no.na[1] <- NULL
notes.star.trek.predicted.na.no.na
```

Identifiant utilisateur	vote prédit
u21	0.8968254
u617	3.8968254
u642	2.6468254
u729	1.8253968
u749	2.7301587
u756	2.7301587
u805	1.3968254
u811	4.8253968
u813	1.8253968
u833	1.7301587
u863	1.8253968
u110	2.5462060
u893	2.6468254
u358	2.8968254
u404	2.8253968
u425	1.6468254
u537	1.8253968
u577	3.3968254
u609	1.8253968
u615	1.8968254

On peut constater que les résultats semblent cohérents car ils sont tous compris entre 0 et 5, de plus la moyenne de ces estimations est d'environ 2.39 ce qui est raisonnable.

4) Calculez l'erreur quadratique moyenne des prédiction de l'approche item-item à la question précédente en la comparant aux valeurs observées

Dans un premier temps, on doit recalculer les vote estimés, cette fois-ci pour les utilisateurs ayant deja votés.

Calcul des votes estimés pour les utilisateurs ayant déjà votés.

```
m.sparse.star.trek <- as.matrix(m.sparse[which(!is.na(m[, indice.star.trek])),
    i.distance.star.trek])  # on ne considere que les utilisateurs ayant deja votés pour star trek

m.star.trek <- as.matrix(m.sparse.star.trek)
m.star.trek[m.star.trek == 0] <- NA

wcos.sums.star.trek <- m.star.trek
wcos.sums.star.trek[!is.na(wcos.sums.star.trek)] <- 1
wcos.sums.star.trek[is.na(wcos.sums.star.trek)] <- 0
wcos.sums.star.trek <- abs(wcos.sums.star.trek %*% (as.matrix(wcos.voisin)))
wcos.sums.star.trek[wcos.sums.star.trek == 0] <- NA

means.voisins = colMeans(m.star.trek, na.rm = TRUE)  # moyenne pour chaque voisin avec vote des utilisa
means.voisins[is.nan(means.voisins)] <- NA

temp <- t(t(m.star.trek) - means.voisins)
temp[is.na(temp)] <- 0
notes.star.trek.predicted <- temp %*% wcos.voisin/wcos.sums.star.trek
notes.star.trek.predicted[notes.star.trek.predicted == 0] <- NA
notes.star.trek.predicted <- mean.star.trek + notes.star.trek.predicted
```

On peut ensuite afficher les votes calculés par rapport aux votes réels.

Comparaison des votes estimés aux votes réels.

```
notes.star.trek.predicted.no.na <- merge(as.matrix(notes.star.trek.predicted),
    as.matrix(notes.star.trek), by = "row.names")
colnames(notes.star.trek.predicted.no.na) <- c("Identifiant utilisateur", "vote prédit",
    "vote réel")
notes.star.trek.predicted.no.na <- na.omit(notes.star.trek.predicted.no.na)
rownames(notes.star.trek.predicted.no.na) <- NULL
notes.star.trek.predicted.no.na
```

Identifiant utilisateur	vote prédit	vote réel
u127	4.9968254	5
u13	1.9473399	3
u130	1.8758358	2
u145	0.9968254	3
u267	2.6190476	2
u276	1.5489537	1
u279	3.1044682	4
u303	2.1706133	3
u394	3.7301587	3
u399	2.0822519	2
u405	1.2789724	1
u417	2.3910664	2
u497	3.6190476	2
u532	2.3912642	2
u600	2.6013843	4
u650	1.6468254	1

Identifiant utilisateur	vote prédit	vote réel
u7	3.1549301	4
u774	1.6343846	2
u807	3.6797595	4

On peut constater que les votes estimés sont souvent proches des votes réels malgre parfois un ecart de 1 voir 2 points.

Calcul de l'erreur quadratique moyenne.

```
erreur.quadratique <- sqrt(mean((notes.star.trek.predicted.no.na[, "vote prédit"] -
    notes.star.trek.predicted.no.na[, "vote réel"])^2, na.rm = TRUE))
erreur.quadratique
```

```
## [1] 0.8639139
```

On peut constater que l'erreur quadratique de nos estimations est d'environ 0.86, ce qui est explicable par le faible nombre de votes communs parfois utilisé afin de générer ces estimations.

5) Un utilisateur a coté la note la plus faible (1) à tous les films de Star Wars et la note la plus forte (5) à tous les films de Star Trek. Quels 10 films lui conseillez-vous? Utilisez une approche utilisateur-utilisateur pour la réponse et 20 voisins rapprochés

Pour répondre à cette question, on peut obter pour une approche similaire à celle de la question 3.

Initialisation du vecteur de votes du nouvel utilisateur

```
indices.star.trek <- grep("trek", as.character(u.item$movie.title), ignore.case = TRUE)
indices.star.wars <- c(172, 181)

votes.new.user <- user.votes <- rep(0, ncol(m.sparse))
votes.new.user[indices.star.trek] <- 5
votes.new.user[indices.star.wars] <- 1
```

Recuperation des 20 voisins les plus rapprochés selon la distance euclidienne du nouvel utilisateur

```
votes.communs.new.user <- (rowSums((votes.new.user * m.sparse) > 0))
distance.new.user <- sqrt(rowSums((votes.new.user - m.sparse)^2))
i.distance.new.user <- min.nindex(distance.new.user, 20)  #pas besoin de supprimer notre utilisateur de
```

Estimation des notes du nouvel utilisateur pour les films qu'il n'a pas vu

```
wcos.new.user.voisins <- as.vector(cosinus.vm(votes.new.user, t(m.sparse[i.distance.new.user,
    ])))

m.sparse.new.user <- as.matrix(m.sparse[i.distance.new.user, ])

m.new.user <- as.matrix(m.sparse.new.user)
m.new.user[m.new.user == 0] <- NA

wcos.sums.new.user <- m.new.user
wcos.sums.new.user[!is.na(wcos.sums.new.user)] <- 1
wcos.sums.new.user[is.na(wcos.sums.new.user)] <- 0
wcos.sums.new.user <- abs(t(wcos.sums.new.user) %*% (as.matrix(wcos.new.user.voisins)))
wcos.sums.new.user[wcos.sums.new.user == 0] <- NA

mean.new.user = mean(votes.new.user[votes.new.user > 0], na.rm = TRUE)

means.voisins.new.user = rowMeans(m.new.user, na.rm = TRUE)  # moyenne pour chaque voisin
means.voisins.new.user[is.nan(means.voisins.new.user)] <- NA

temp <- t(t(m.new.user) - means.voisins)
temp[is.na(temp)] <- 0

notes.new.user.predicted <- t(temp) %*% wcos.new.user.voisins/wcos.sums.new.user
notes.new.user.predicted[notes.new.user.predicted == 0] <- NA
notes.new.user.predicted <- mean.new.user + notes.new.user.predicted
```

Affichage des 10 films auquels l'utilisateur est susceptible de donner les meilleurs notes

```
indices.movies.recommended.new.user <- min.nindex(distance.new.user, 10)
title.movies.recommended.new.user <- as.matrix(u.item$movie.title[indices.movies.recommended.new.user])
title.movies.recommended.new.user <- data.frame(title.movies.recommended.new.user)
colnames(title.movies.recommended.new.user) <- c("Recommended movies")
title.movies.recommended.new.user
```

Recommended movies
Executive Decision (1996)
Great White Hype, The (1996)
L.A. Confidential (1997)
Michael (1996)
Dirty Dancing (1987)
Wyatt Earp (1994)
M (1931)
Groundhog Day (1993)
Demolition Man (1993)
Picture Perfect (1997)

6) Je suis un nouvel utilisateur. Vous connaissez ma profession, mon sexe et mon âge. Développez un algorithme bayésien pour recommander 10 films sur la base de ces trois catégories

Afin de répondre à cette question, il est nécessaire de "discretiser" la variable pour l'age de l'utilisateur, sans quoi les nombre de comparaisons avec d'autres utilisateurs du meme age sera trop faible. A cette fin, l'approche vue en cours sera utilisée : les ages seront séparés en 2 groupes : les moins de 50 ans et les 50 ans et plus. Cette approche peut sembler diminuer fortement la precision de l'algorithme, cependant les 2 autres precision (sexe et profession) permettront d'obtenir des resultats suffisament precis pour globaliser fortement l'age du nouvel utilisateur.

Fonctions utilitaires

On peut commencer par créer 2 fonctions utilitaires.

```
ratio.chances <- function(rating.vec, seuil = 3) {
    sum(rating.vec > seuil)/sum(rating.vec <= seuil)
}

OddsToP <- function(o) {
    o/(1 + o)
}
```

On peut ensuite écrire la fonction principale qui, selon les caracteristique de l'utilisateur, lui retournera 10 filns conseillés.

Fonction principale

```
recommended.movie.content.base <- function(job, gender, age) {

    probabilities.to.like = rep(0, nrow(u.item))

    for (movie in c(1:nrow(u.item))) {

        i <- (mx$item == movie & mx$rating > 3)  # ceux qui aiment
        ni <- (mx$item == movie & mx$rating <= 3)  # et ceux qui n'aiment pas

        # on ne considere le film que s'il y a au minimum 3 votes, sinon on laisse
        # la probabilité à 0
        if (sum(i) + sum(ni) >= 3) {

            E_11 <- (table(mx[i, "job"])/sum(table(mx[i, "job"])))[job]
            E_12 <- (table(mx[ni, "job"])/sum(table(mx[ni, "job"])))[job]

            if (is.na(E_12) | E_12 == 0) {
                E_1 <- 0
            } else {
                E_1 <- E_11/E_12
            }

            if (age < 50) {
```

```
                E_21 <- table(mx[i, "age"] < 50)/sum(table(mx[i, "age"] < 50))
                E_22 <- table(mx[ni, "age"] < 50)/sum(table(mx[ni, "age"] <
                  50))
            } else {
                E_21 <- table(mx[i, "age"] >= 50)/sum(table(mx[i, "age"] >=
                  50))
                E_22 <- table(mx[ni, "age"] >= 50)/sum(table(mx[ni, "age"] >=
                  50))
            }

            # si on a pas suffisament de votes on arrete
            if (nrow(E_21) < 2 | nrow(E_22) < 2) {
                E_2 <- 0
            } else {
                E_2 <- (E_21/E_22)["TRUE"]
            }

            E_31 <- (table(mx[i, "gender"])/sum(table(mx[i, "gender"])))[gender]
            E_32 <- (table(mx[ni, "gender"])/sum(table(mx[ni, "gender"])))[gender]

            if (is.na(E_32) | E_32 == 0) {
                E_3 <- 0
            } else {
                E_3 <- E_31/E_32
            }

            probabilities.to.like[movie] <- OddsToP(ratio.chances(mx[mx$item.id ==
                movie, "rating"]) * E_1 * E_2 * E_3)

        } else {
            probabilities.to.like[movie] <- 0
        }

    }

    probabilities.to.like[is.nan(probabilities.to.like)] <- 0

    indices.movies.recommended.new.user <- max.nindex(probabilities.to.like,
        10)
    recommended.movies <- data.frame(u.item$movie.title[indices.movies.recommended.new.user])
    colnames(recommended.movies) <- c("Recommended movies")
    recommended.movies
}
```

Essais avec différents utilisateurs

On peut finalement tester cette fonction avec differents cas.

```
recommended.movie.content.base('engineer', 'M', '40')
```

Recommended movies
L.A. Confidential (1997)

Recommended movies
Star Wars (1977)
Maltese Falcon, The (1941)
Princess Bride, The (1987)
Vertigo (1958)
Casablanca (1942)
Raiders of the Lost Ark (1981)
Man Who Would Be King, The (1975)
Rear Window (1954)
Empire Strikes Back, The (1980)

```
recommended.movie.content.base('student', 'F', '20')
```

Recommended movies
Casablanca (1942)
Vertigo (1958)
Jane Eyre (1996)
Nikita (La Femme Nikita) (1990)
Secrets & Lies (1996)
Manchurian Candidate, The (1962)
Shawshank Redemption, The (1994)
Much Ado About Nothing (1993)
Some Folks Call It a Sling Blade (1993)
Hamlet (1996)

```
recommended.movie.content.base('writer', 'M', '60')
```

Recommended movies
Amadeus (1984)
Maltese Falcon, The (1941)
Looking for Richard (1996)
High Noon (1952)
Peacemaker, The (1997)
Godfather, The (1972)
Crumb (1994)
Priest (1994)
Glory (1989)
Apocalypse Now (1979)