

Cahier thématique

Évaluation des systèmes de Recommandation

**Manipulation 1** : Préparation des données pour évaluer les modèles

**Manipulation 2 :** Évaluer des ratings

**Manipulation 2 :** Évaluer de la 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 RStudio

## Préliminaire

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

## Objectif

Le TPs précédents vous construire des systèmes recommandation. Dans ce TP nous allons voir comment évaluer les systèmes recommandation,.

## Démarche :

* Préparation des données pour évaluer la performance
* L'évaluation de la performance de la prédiction des ratings
* L’évaluation de la performance de la recommandation

# Description : MovieLens

Les données du TP sont issues de la collection des films de MovieLens,

## Manipulation 1 : Préparation des données pour évaluer les modèles

### **Objectif** :

### Pour évaluer les modèles, vous devez les construire avec des données et de les tester sur d'autres données. Cette vous montrera comment préparer les deux ensembles de données. Le package recommenderlab contient des outils prédéfinis qui aident dans cette tâche.

L'objectif est de définir deux ensembles de données, qui sont les suivants :

* Jeu d’apprentissage
* Jeu de test

Évaluer le modèle en comparant les ratings prédits par le modèle avec leurs vrais valeurs.

## Charger les données

library(pander)

set.seed(1)

library(recommenderlab)

library(ggplot2)

data(MovieLense)

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

colCounts(MovieLense) > 100]

ratings\_movies

## Validation croisé k-fold

Préparer les données de test pour l’évaluation

n\_fold <- 4

eval\_sets <- evaluationScheme(data = ratings\_movies,

method = "cross-validation",

k = n\_fold,

given = items\_to\_keep,

goodRating = rating\_threshold)

Tel que :

* data: dataset initiale
* method : la façon dont les données sont découpées. Dans ce cas c’est pas cross validation
* given: nombre d’item utilisé pour l’évaluation par utlisateur
* goodRating : : Seuil de valeur minimum pour considérer qu’un rating comme bon
* k : le nombre de fois pour exécuter l'évaluation

## Manipulation 2 : Évaluer des ratings

Nous devons définir le modèle à évaluer. Par exemple, nous pouvons évaluer le Item-Based Recommender.

Nous avons besoin de spécifier le nom du modèle et la liste de ses paramètres. Si nous utilisons leurs valeurs par défaut, alors il est NULL :

model\_to\_evaluate <- "IBCF"

model\_parameters <- NULL

1. Construire le modèle

eval\_recommender <- Recommender(data = getData(eval\_sets, "train"), method = model\_to\_evaluate,parameter = model\_parameters)

Le IBCF peut recommander de nouveaux items et de prédire leurs ratings. Afin de construire le modèle, nous avons besoin de spécifier le nombre d'éléments que nous voulons recommander, par exemple, 10, même si on n'a pas besoin d'utiliser ce paramètre dans l'évaluation :

items\_to\_recommend <- 10

1. Construire la matrice avec les ratings prédit

eval\_prediction <- predict(object = eval\_recommender, newdata =

getData(eval\_sets, "known"), n = items\_to\_recommend, type = "ratings")

class(eval\_prediction)

## realRatingMatrix

## Inspection

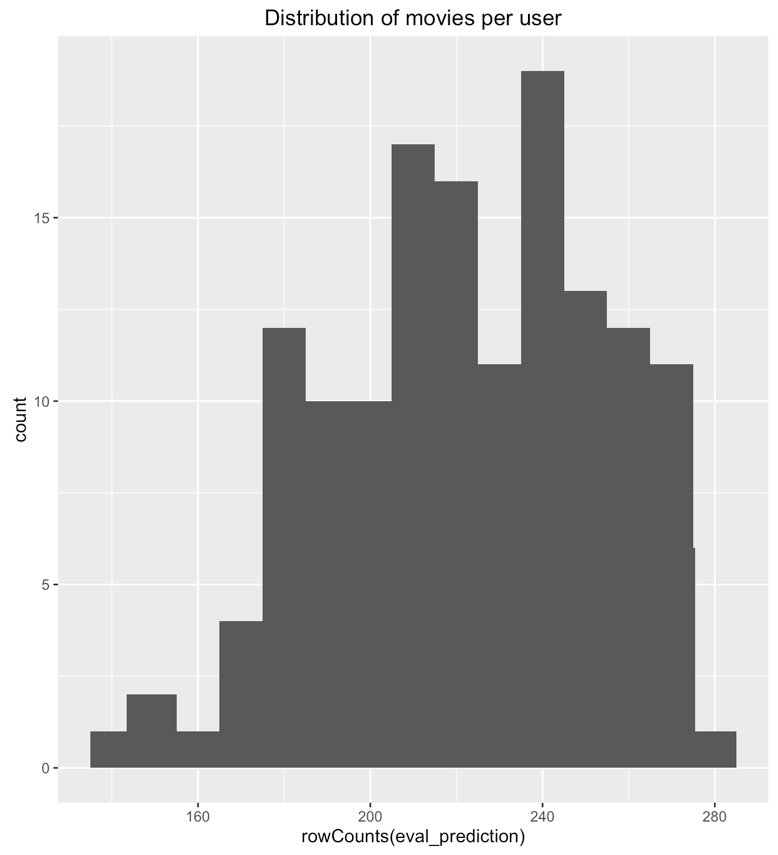
Combien de films nous recommandons à chaque utilisateur ? A cet effet, nous pouvons visualiser la répartition du nombre de films par utilisateur :

qplot(rowCounts(eval\_prediction)) +

geom\_histogram(binwidth = 10) +

ggtitle("Distribution of movies per user")

Le résultat sera :



Le nombre de films par utilisateur est à peu près entre 150 et 300.

## Métrique d’évaluation

La fonction de mesure la précision du modèle (« accuracy » ) est calcPredictionAccuracy et il calcule les métriques suivantes:

* Root mean square error (RMSE) : Ceci est la déviation standard de la différence entre les ratings réelles et prédits.
* Mean squared error (MSE) : Ceci est la moyenne de la différence au carré entre les les ratings réelles et prédits. Il est le carré de RMSE, il contient les mêmes informations.
* Mean absolute error (MAE) : Ceci est la moyenne de la différence absolue entre les ratings réelles et prédits.

Calculer ces mesures sur chaque utilisateur :

eval\_accuracy <- calcPredictionAccuracy(

x = eval\_prediction,

data = getData(eval\_sets, "unknown"),

byUser = TRUE)

pander(head(eval\_accuracy))

Le résultat sera :

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

&nbsp; RMSE MSE MAE

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

\*\*1\*\* 1.217 1.481 0.8205

\*\*2\*\* 0.908 0.8244 0.727

\*\*6\*\* 1.172 1.374 0.903

\*\*14\*\* 1.405 1.973 1.027

\*\*15\*\* 1.601 2.562 1.243

\*\*18\*\* 0.8787 0.7721 0.633

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

## Manipulation 3 : Évaluer de la recommandation

Objectif : Une autre façon de mesurer exactitudes est en comparant les recommandations avec les items ayant une note positive.

## Évaluer

On utilise la fonciton evaluate avec les paramètres suivants :

* x: l'objet contenant le système d'évaluation.
* Méthode: Ceci est la technique de recommandation.
* n: Ceci est le nombre d'items de recommander à chaque utilisateur.

results <- evaluate(x = eval\_sets, method = model\_to\_evaluate, n = seq(10, 100, 10))

class(results)

## evaluationResults

## Matrice de confusion

Extraire une liste des matrices de confusion. Chaque élément de la liste correspond à une autre division de k fold.

pander(head(getConfusionMatrix(results)[[1]]))

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

&nbsp; TP FP FN TN precision recall TPR FPR

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

\*\*10\*\* 3.443 6.557 70.61 236.4 0.3443 0.04642 0.04642 0.02625

\*\*20\*\* 6.686 13.31 67.36 229.6 0.3343 0.09175 0.09175 0.05363

\*\*30\*\* 10.02 19.98 64.03 223 0.334 0.1393 0.1393 0.08075

\*\*40\*\* 13.29 26.71 60.76 216.2 0.3323 0.1849 0.1849 0.1081

\*\*50\*\* 16.43 33.57 57.62 209.4 0.3286 0.2308 0.2308 0.1362

\*\*60\*\* 19.61 40.39 54.44 202.6 0.3268 0.2759 0.2759 0.164

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

Les quatre premières colonnes contiennent les vrais-faux des valeurs positifs / négatifs , et ils sont comme suit:

* Positifs Vrai (TP) : items recommandés qui ont été achetés
* Faux Positifs (FP) : Items recommandés qui n'ont pas été achetés
* Faux Négatifs (FN) : items qui ne sont pas recommandées qui ont été achetés
* Négatifs Vrai (TN) : qui ne sont pas recommandées et qui n'ont pas été achetés

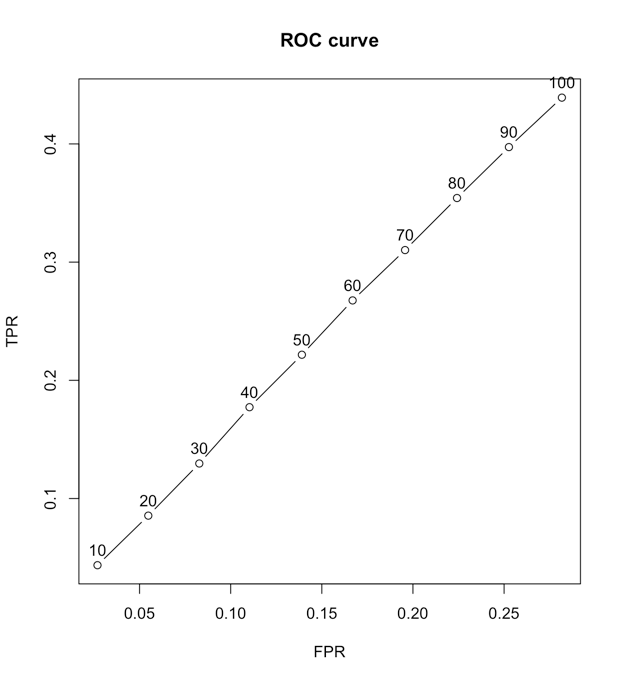
## Coubre de ROC, Recall et Precision

Tout d'abord, nous allons construire la courbe ROC. Il affiche ces facteurs :

* Taux de vrais positifs (TPR): Ceci est le pourcentage d'articles achetés que ont été recommandées. Il est le nombre de TP divisé par le nombre de articles achetés (TP + FN).
* Taux de faux positifs (FPR): Ceci est le pourcentage d'articles non pas été recommandées. Il est le nombre de FP divisé par le nombre de pas articles achetés (FP + TN).

plot(results, annotate = TRUE,main = "ROC curve")

Résultat :



Courbe de précision et rappel

plot(results, "prec/rec", annotate = TRUE, main = "Precision-recall")

