Requêtes avec jointures sous R

Tutoriel Tanagra

1. Introduction

Dans ma pratique usuelle, lorsque je dois traiter des bases multi-tables dans un processus de modélisation, j'effectue une partie du pre-processing avec des SGBD (système de gestion de base de données). Avec le langage SQL (structured query language), on y est à l'aise pour effectuer des requêtes mettant en œuvre des jointures complexes entre plusieurs sources. Au final, une table unique propice à l'analyse est produite, que j'importe ensuite dans le logiciel d'analyse statistique, que ce soit R ou Python.

Cette approche n'est pas toujours adaptée lorsque les sources initiales sont susceptibles de mises à jour fréquentes. Une modification des données nécessiterait la réexécution des requêtes en amont avant de pouvoir relancer le processus de modélisation. Dans ce cas, il est plus judicieux d'intégrer le code de la phase de requêtage dans le programme réalisant le traitement statistique.

Dans ce tutoriel, nous étudions les différentes solutions à notre disposition sous R pour effectuer des requêtes avec jointures. Elles ont toutes permis de répondre au cahier des charges, avec plus ou moins de facilité. Finalement, il nous appartient de choisir celle qui est la plus adaptée par rapport à notre cahier des charges.

2. Données "MovieLens"

a. Présentation

La base de données provient du site MovieLens (https://grouplens.org/datasets/movielens/). J'ai pris une version réduite d'un des datasets. Deux tables ont été regroupées dans un classeur Excel "**movies_dataset.xlsx**", composé de deux feuilles.

USER RATINGS. La première (user ratings) décrit les notes attribuées à des films par des utilisateurs. Un utilisateur ne peut pas évaluer plusieurs fois le même film ; il n'a pas évalué tous les films existants.

Les premières lignes de la base se présentent comme suit :

user	title	rating
1	Toy Story (1995)	5
1	GoldenEye (1995)	3
1	Four Rooms (1995)	4
1	Get Shorty (1995)	3
1	Copycat (1995)	3
1	Twelve Monkeys (1995)	4
1	Babe (1995)	1
1	Dead Man Walking (1995)	5
1	Richard III (1995)	3
2	Toy Story (1995)	4
2	Richard III (1995)	2
2	Mighty Aphrodite (1995)	4

On y lit par exemple que l'utilisateur 1 a vu le film "Toy Story (1995)" et l'a noté 5 ; il a également visionné le film "GoldenEye (1995)" avec la note 3 ; etc.

MOVIES TYPES. La seconde (movies types) recense la catégorie attribuée aux films. Chaque film peut être classé dans plusieurs catégories.

title	unknown	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	Fantasy	FilmNoir	Horror	Musical	Mystery	Romance	SciFiction	Thriller	War	Western
U Turn (1997)	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0
Chungking Express (1994)	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
Rainmaker, The (1997)	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
I Can t Sleep (J ai pas sommeil) (1994)	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0
Good Man in Africa, A (1994)	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bonheur, Le (1965)	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Jingle All the Way (1996)	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Babyfever (1994)	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
Liar Liar (1997)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Fools Rush In (1997)	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0

"U Turn (1997)" est rattaché aux films d'action, de Crime et de Mystery ; "Chunking Express (1994)" aux genres Drama, Mystery et Romance ; etc.

On observe tout de suite que la jonction entre ces deux tables peut s'appuyer sur le champ "title" qui leur est commune. Cependant, certains films évalués dans la table "user ratings" n'apparaissent pas dans la seconde table "movies types". L'inverse peut être vrai également c.-à-d. certains films référencés n'ont jamais reçu de note. Il faudra être très vigilant lorsqu'on aura à établir les jointures.

b. Importation des données et quelques vérifications

Nous utilisons le package “readxl” pour importer les données. Il faut préalablement l’installer s’il n’est pas présent sur notre machine.

La table “user ratings”

Dans un premier temps, nous chargeons la feuille “user ratings”. Nous affichons la liste des champs puis les dix premières observations.

```
#librairie pour chargement de fchiers Excel
library(readxl)

#charger les dfRatings des users
dfRatings <- read_excel("movies_dataset.xlsx",sheet="user ratings")

#vérification
print(str(dfRatings))
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    1190 obs. of  3 variables:
##  $ user  : num  1 1 1 1 1 1 1 1 2 ...
##  $ title : chr  "Toy Story (1995)" "GoldenEye (1995)" "Four Rooms (1995)" "Get
Shorty (1995)" ...
##  $ rating: num  5 3 4 3 3 4 1 5 3 4 ...
## NULL
```

```
#affichage des 10 premières lignes
print(head(dfRatings,10))
```

```
## # A tibble: 10 x 3
##     user title                   rating
##    <dbl> <chr>                    <dbl>
##  1    1. Toy Story (1995)            5.
##  2    1. GoldenEye (1995)            3.
##  3    1. Four Rooms (1995)           4.
##  4    1. Get Shorty (1995)           3.
##  5    1. Copycat (1995)              3.
##  6    1. Twelve Monkeys (1995)       4.
##  7    1. Babe (1995)                 1.
##  8    1. Dead Man Walking (1995)     5.
##  9    1. Richard III (1995)          3.
## 10    2. Toy Story (1995)            4.
```

Il y a 1190 évaluations dans la table.

Calculons quelques statistiques pour mieux appréhender le contenu du fichier :

- La liste des utilisateurs et le nombre de films visionnés par chacun d'entre eux.

```
#nombre de films vu par chaque utilisateur
print(table(dfRatings$user))
```

```
##
##   1   2   3   4   5   6   7   8   9  10
##   9  59  51  23 175 208 400  59  22 184
```

Il y a 10 utilisateurs distincts. Le premier utilisateur (user == 1) a vu 9 films, le second 59, etc.

- Les notes moyennes attribuées par chaque utilisateur.

```
#notes moyennes des utilisateurs
print(aggregate(dfRatings$rating,by=list(dfRatings$user),mean))
```

```
##    Group.1        x
## 1        1 3.444444
## 2        2 3.745763
## 3        3 2.764706
## 4        4 4.304348
## 5        5 2.874286
## 6        6 3.639423
## 7        7 3.965000
## 8        8 3.796610
## 9        9 4.272727
## 10      10 4.206522
```

Le plus "pingre" est l'utilisateur n°3, le plus généreux est le n°4.

- Le nombre de films qui ont été notés.

```
#nombre de films notés
print(length(unique(dfRatings$title)))
```

```
## [1] 642
```

642 films distincts ont été évalués. Chaque film a donc été en moyenne visionné 1190/642 = 1.85 fois.

La table "movie types"

Nous chargeons la seconde feuille du classeur Excel.

```r
#charger le type des films
dfTypes <- read_excel("movies_dataset.xlsx",sheet="movies types")

#vérification
print(str(dfTypes))
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    100 obs. of  20 variables:
##  $ title      : chr  "U Turn (1997)" "Chungking Express (1994)" "Rainmaker, The
(1997)" "I Can t Sleep (J ai pas sommeil) (1994)" ...
##  $ unknown    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Action     : num  1 0 0 0 1 0 0 0 0 0 ...
##  $ Adventure  : num  0 0 0 0 1 0 1 0 0 0 ...
##  $ Animation  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Children   : num  0 0 0 0 0 0 1 0 0 0 ...
##  $ Comedy     : num  0 0 0 0 0 0 1 1 1 1 ...
##  $ Crime      : num  1 0 0 0 0 0 0 0 0 0 ...
##  $ Documentary: num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Drama      : num  0 1 1 1 0 1 0 1 0 0 ...
##  $ Fantasy    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ FilmNoir   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Horror     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Musical    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Mystery    : num  1 1 0 0 0 0 0 0 0 0 ...
##  $ Romance    : num  0 1 0 0 0 0 0 0 0 1 ...
##  $ SciFiction : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Thriller   : num  0 0 0 1 0 0 0 0 0 0 ...
##  $ War        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Western    : num  0 0 0 0 0 0 0 0 0 0 ...
## NULL
```

100 films différents sont recensés, caractérisés par 20 catégories.

Calculons le nombre de films par genre.

```r
#nombre de films par genre
print(colSums(dfTypes[-1]))
```

```
##     unknown      Action   Adventure   Animation    Children      Comedy
##           0          17           8           4          11          32
##       Crime Documentary       Drama     Fantasy    FilmNoir      Horror
##           2           1          51           0           1           4
##     Musical     Mystery     Romance  SciFiction    Thriller         War
##           4           5          17           5          13           5
```

```
##     Western
##           0
```

Les films les plus représentés dans cette typologie sont : Drama, Comedy, Action et Romance. Nous étudierons en priorité ces catégories.

3. Le package "sqldf"

Le package "sqldf" (https://cran.r-project.org/web/packages/sqldf/index.html) permet d'exécuter des requêtes SQL appliquées sur des data frame dans du code R. Il nous permet de profiter pleinement de la puissance du langage SQL tout en restant dans notre environnement de travail habituel, et enchaîner directement avec les traitements statistiques car l'objet issu des requêtes est lui également un data frame.

Après l'avoir installé, nous le chargeons avec la commande library().

```
#librairie sqldf
library(sqldf)

## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

a. Associer le genre à chaque couple utilisateur-film

L'objectif est d'associer à chaque binôme utilisateur-film de la table "ratings" les catégories Action, Comedy, Drama et Romance. L'association est basée sur le champ "title" commune aux deux tables. Nous effectuons une jointure à gauche pour que chaque enregistrement de "ratings" soit caractérisé même si le titre correspondant n'est pas présent dans la table "types".

Nous procédons en deux temps : définition de la requête sous la forme d'une chaîne de caractères, exécution de la requête avec la commande sqldf().

```
#définition de la requête
q1 <- "SELECT dfRatings.User, dfTypes.title, dfTypes.Action, dfTypes.Comedy,
dfTypes.Drama, dfTypes.Romance
FROM dfRatings LEFT JOIN dfTypes ON dfRatings.title = dfTypes.title;"

#exécution de la requête
dfq1 <- sqldf(q1)
```

```
#vérification
print(dim(dfq1))
```

```
## [1] 1190    6
```

Nous disposons d'un nouveau data frame avec 1190 lignes, autant que dans la table "ratings", et 6 colonnes, les 4 catégories concernées ont été accolées aux champs "user" et "title". Affichons les dix premières lignes.

```
#affichage des 10 premières lignes du fruit de la requête
print(head(dfq1,10))
```

```
##    user            title Action Comedy Drama Romance
## 1     1             <NA>     NA     NA    NA      NA
## 2     1 GoldenEye (1995)      1      0     0       0
## 3     1             <NA>     NA     NA    NA      NA
## 4     1             <NA>     NA     NA    NA      NA
## 5     1             <NA>     NA     NA    NA      NA
## 6     1             <NA>     NA     NA    NA      NA
## 7     1             <NA>     NA     NA    NA      NA
## 8     1             <NA>     NA     NA    NA      NA
## 9     1             <NA>     NA     NA    NA      NA
## 10    2             <NA>     NA     NA    NA      NA
```

Pour l'utilisateur n°1, seul le film "Golden Eye" est reconnu dans la base "types". Nous disposons de sa description. Les autres films notés ("Toy STory", "Four Rooms", etc.) ne sont pas recensés, d'où les valeurs NA.

Il est facile de restreindre dfq1 aux films effectivement caractérisés.

```
print(nrow(dfq1[!is.na(dfq1$title),]))
```

```
## [1] 80
```

On dispose de la typologie de 80 couples utilisateurs-films.

b. Association restreinte aux films reconnus

Plutôt que de passer par un post-traitement, nous pouvons intégrer cette contrainte dans la définition de la requête en utilisant une jointure interne.

```
#association uniquement avec les films effectivement caractérisés
q2 <- "SELECT dfRatings.User, dfTypes.title, dfTypes.Action, dfTypes.Comedy,
dfTypes.Drama, dfTypes.Romance
FROM dfRatings INNER JOIN dfTypes ON dfRatings.title = dfTypes.title;"
```

```
#lancement
dfq2 <- sqldf(q2)

#dimensions
print(dim(dfq2))
```

```
## [1] 80  6
```

```
#affichage des 10 premiers
print(head(dfq2,10))
```

```
##    user                         title Action Comedy Drama Romance
## 1     1             GoldenEye (1995)      1      0     0       0
## 2     2          Jerry Maguire (1996)      0      0     1       1
## 3     2               Sabrina (1995)      0      1     0       1
## 4     2 Sense and Sensibility (1995)      0      0     1       1
## 5     2             Liar Liar (1997)      0      1     0       0
## 6     2         Rainmaker, The (1997)      0      0     1       0
## 7     3             Liar Liar (1997)      0      1     0       0
## 8     3                Mother (1996)      0      1     0       0
## 9     3                U Turn (1997)      1      0     0       0
## 10    4             Liar Liar (1997)      0      1     0       0
```

Nous disposons de 80 enregistrements. Le résultat est cohérent avec celui de la section précédente (après post traitement de la requête LEFT JOIN).

Pour l'utilisateur n°1, seul le film "Golden Eye" est référencé dans la table "types".

c. Genre des films visionnés par les utilisateurs

L'objectif dans cette section est de mesurer l'appétence des utilisateurs aux différentes catégories de films. Il y a un regroupement (par utilisateur) et des calculs récapitulatifs (par genre) à réaliser.

```
#genre de films visionnés par chaque utilisateur
q3 <- "SELECT dfRatings.User, Sum(dfTypes.Action) AS SAction, Sum(dfTypes.Comedy) AS
SComedy, Sum(dfTypes.Drama) AS SDrama, Sum(dfTypes.Romance) AS SRomance
FROM dfRatings INNER JOIN dfTypes ON dfRatings.title = dfTypes.title
GROUP BY dfRatings.User;"

#exécution de la requête
dfq3 <- sqldf(q3)
```

```
#affichage des résultats
print(dfq3)
```

```
##    user SAction SComedy SDrama SRomance
## 1     1       1       0      0        0
## 2     2       0       2      3        3
## 3     3       1       2      0        0
## 4     4       1       2      0        0
## 5     5       2       3      0        0
## 6     6       1       6     11        6
## 7     7       4       6      8        4
## 8     8       2       2      1        0
## 9     9       0       1      1        0
## 10   10       2       4      8        2
```

L'utilisateur n°6 par exemple semble être particulièrement attiré par les films dramatiques (Drama). Il faut quand même mettre un bémol à cette affirmation car nous nous en tenons uniquement aux films référencés dans la base "types" ici. En réalité, cet utilisateur a visionné 208 films.

d. Notes moyennes attribués aux films dramatiques visionnés

Nous souhaitons calculer les notes moyennes attribués par les utilisateurs aux films dramatiques qu'ils ont visionnés. Il y a un regroupement à faire (selon les utilisateurs), un calcul récapitulatif (moyenne des notes) et une restriction (films dramatiques uniquement).

```
#note moyenne attribué par l'utilisateur selon le genre Drama
q4 <- "SELECT dfRatings.user, Count(dfTypes.Drama) AS CDrama, Avg(dfRatings.rating)
AS Mrating
FROM dfRatings INNER JOIN dfTypes ON dfRatings.title = dfTypes.title
GROUP BY dfRatings.user, dfTypes.Drama
HAVING (((dfTypes.Drama)=1));"

#exécution de la requête
dfq4 <- sqldf(q4)

#affichage
print(dfq4)
```

```
##   user CDrama  Mrating
## 1    2      3 4.333333
## 2    6     11 3.545455
## 3    7      8 4.500000
```

```
## 4    8      1 5.000000
## 5    9      1 3.000000
## 6   10      8 4.375000
```

Les utilisateurs n°1, 3, 4 et 5 n'ont pas vu de films dramatiques, ils n'apparaissent pas dans les résultats. L'utilisateur n°8 est celui qui apprécie le plus ce type de film, il n'en a vu qu'un seul ceci étant.

4. Fonctions natives de R

Pour les personnes réfractaires au SQL, est-ce qu'il y a une alternative sous R ? Dans cette section, nous nous intéressons à la fonction merge() de R (https://stat.ethz.ch/R-manual/R-devel/library/base/html/merge.html). Elle semble pouvoir répondre à nos attentes car elle permet de produire des requêtes avec jointures entre data frame.

a. Requête 1 : Associer le genre à chaque couple utilisateur-film

La jointure à gauche est définie comme suit :

```
#utilisation de merge LEFT JOIN
dmq1 <- merge(dfRatings,dfTypes,by="title",all.x=TRUE,sort=FALSE)

#affichage pour le premier utilisateur
print(dmq1[dmq1$user==1,c("user","title","Action","Comedy","Drama","Romance")])
```

```
##    user                    title Action Comedy Drama Romance
## 2     1         GoldenEye (1995)      1      0     0       0
## 81    1         Toy Story (1995)     NA     NA    NA      NA
## 83    1        Four Rooms (1995)     NA     NA    NA      NA
## 84    1        Get Shorty (1995)     NA     NA    NA      NA
## 85    1           Copycat (1995)     NA     NA    NA      NA
## 86    1    Twelve Monkeys (1995)     NA     NA    NA      NA
## 87    1              Babe (1995)     NA     NA    NA      NA
## 88    1 Dead Man Walking (1995)     NA     NA    NA      NA
## 89    1       Richard III (1995)     NA     NA    NA      NA
```

Les trois premiers paramètres sont respectivement les deux tables sources et le champ de jointure. Si ce dernier porte des noms différents dans les deux sources, il est possible de les indiquer spécifiquement.

L'option all.x = TRUE joue un rôle important. La fonction recherche les correspondances pour tous les éléments de la première table (dfRatings). Quand elles n'existent pas, les valeurs NA sont introduites.

Enfin, l'option sort=FALSE évite de trier la table résultante selon le champ de jointure. Opération superflue en ce qui nous concerne.

De fait, nous avons bien les 1190 enregistrements de la table "ratings".

```
#dimension du data frame issu de la fusion
print(dim(dmq1))
```

```
## [1] 1190   22
```

b. Requête 2 : Association restreinte aux films reconnus

L'INNER JOIN va être spécifié par l'option all = FALSE.

```
#utilisation de merge INNER JOIN
dmq2 <- merge(dfRatings,dfTypes,by="title",all=FALSE,sort=FALSE)

#affichage pour le premier utilisateur
print(dmq2[dmq2$user==1,c("user","title","Action","Comedy","Drama","Romance")])
```

```
##   user            title Action Comedy Drama Romance
## 2    1 GoldenEye (1995)      1      0     0       0
```

Seul le film "Golden Eye" apparaît pour le premier utilisateur.

Et nous disposons bien cette fois-ci de 80 enregistrements dans la table résultante.

```
#dimension
print(dim(dmq2))
```

```
## [1] 80 22
```

c. Requête 3 : Genre des films visionnés par les utilisateurs

La fonction merge() n'est pas conçue pour effectuer des regroupements. Mais en réalité, nous disposons déjà de toutes les informations nécessaires dans le data frame **dmq2**. Il ne nous reste plus qu'à utiliser les fonctions usuelles de regroupement de R pour réaliser les consolidations. Dans notre cas, nous nous appuyons sur aggregate().

```
#nombre de films par genre
print(aggregate(dmq2[c("Action","Comedy","Drama","Romance")],by=list(dmq2$user),sum))
```

```
##   Group.1 Action Comedy Drama Romance
## 1       1      1      0     0       0
## 2       2      0      2     3       3
```

```
## 3        3      1      2     0       0
## 4        4      1      2     0       0
## 5        5      2      3     0       0
## 6        6      1      6    11       6
## 7        7      4      6     8       4
## 8        8      2      2     1       0
## 9        9      0      1     1       0
## 10      10      2      4     8       2
```

d. Requête 4 : Notes moyennes attribués aux films dramatiques visionnés

Nous opérons de la même manière pour calculer les notes moyennes des films dramatiques.

Nous créons dans un premier temps une variable intermédiaire pour distinguer les notes associées aux films dramatiques.

```
#notes des films dramatiques, NA si pas concerné
NotesDrama <- ifelse(dmq2$Drama>0, dmq2$rating,NA)
```

Puis, nous réalisons la consolidation.

```
#moyennes des notes par utilisateur des films dramatiques
print(aggregate(NotesDrama,by=list(dmq2$user),mean,na.rm=TRUE))
```

```
##    Group.1        x
## 1        1      NaN
## 2        2 4.333333
## 3        3      NaN
## 4        4      NaN
## 5        5      NaN
## 6        6 3.545455
## 7        7 4.500000
## 8        8 5.000000
## 9        9 3.000000
## 10      10 4.375000
```

La valeur NaN est associée aux utilisateurs qui n'ont pas visionné de films dramatiques (Nan parce qu'une moyenne a été calculée sur 0 observations, nous avons une division par zéro).

5. Le package “dplyr”

J’avais identifié le package “dplyr”. Après avoir regardé succinctement, j’en étais arrivé à la conclusion qu’il permettait certes de faciliter certaines tâches, mais sans vraiment introduire un gap significatif par rapport aux commandes de base de R et un peu jugeotte.

Plusieurs étudiants m’en ont reparlé récemment, mettant en avant son aspect pratique et ses performances sur les grandes bases. Je me suis dit qu’éprouver ses capacités dans notre contexte constituerait un test intéressant.

Après avoir installé le package, nous le chargeons.

```
#chargement du package dplyr
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

a. Requête 1 : Associer le genre à chaque couple utilisateur-film

Pour la première requête, nous disposons d’une fonction dédiée.

```
#requête avec jointure à gauche
dp1 <- dplyr::left_join(dfRatings,dfTypes,by="title")

#10 premières observations
print(head(dp1[c("user","title","Action","Comedy","Drama","Romance")],10))
```

```
## # A tibble: 10 x 6
##     user title                    Action Comedy Drama Romance
##    <dbl> <chr>                     <dbl>  <dbl> <dbl>   <dbl>
##  1    1. Toy Story (1995)           NA     NA    NA      NA
##  2    1. GoldenEye (1995)            1.     0.    0.      0.
##  3    1. Four Rooms (1995)          NA     NA    NA      NA
##  4    1. Get Shorty (1995)          NA     NA    NA      NA
##  5    1. Copycat (1995)             NA     NA    NA      NA
```

```
##  6    1. Twelve Monkeys (1995)      NA     NA    NA      NA
##  7    1. Babe (1995)                NA     NA    NA      NA
##  8    1. Dead Man Walking (1995)    NA     NA    NA      NA
##  9    1. Richard III (1995)         NA     NA    NA      NA
## 10    2. Toy Story (1995)           NA     NA    NA      NA
```

```
#dimension de la table résultante
print(dim(dp1))
```

```
## [1] 1190   22
```

Les paramètres de left_join() sont pour le moins explicites. Les résultats concordent en tous points aux deux approches précédentes.

b. Requête 2 : Association restreinte aux films reconnus

Ici également, la fonction dédiée inner_join() fait l'affaire pour la jointure interne.

```
#requête avec jointure interne
dp2 <- dplyr::inner_join(dfRatings,dfTypes,by="title")

#10 premiers enregistrements
print(head(dp2[c("user","title","Action","Comedy","Drama","Romance")],10))
```

```
## # A tibble: 10 x 6
##     user title                        Action Comedy Drama Romance
##    <dbl> <chr>                         <dbl>  <dbl> <dbl>   <dbl>
##  1    1. GoldenEye (1995)                1.     0.    0.      0.
##  2    2. Jerry Maguire (1996)            0.     0.    1.      1.
##  3    2. Sabrina (1995)                  0.     1.    0.      1.
##  4    2. Sense and Sensibility (1995)    0.     0.    1.      1.
##  5    2. Liar Liar (1997)                0.     1.    0.      0.
##  6    2. Rainmaker, The (1997)           0.     0.    1.      0.
##  7    3. Liar Liar (1997)                0.     1.    0.      0.
##  8    3. Mother (1996)                   0.     1.    0.      0.
##  9    3. U Turn (1997)                   1.     0.    0.      0.
## 10    4. Liar Liar (1997)                0.     1.    0.      0.
```

```
#dimension de la table résultante
print(dim(dp2))
```

```
## [1] 80 22
```

Remarque : Notons qu'il est possible de filtrer une table en fonction d'une autre, sans réaliser explicitement la requête. Dans notre cas, si nous souhaitons restreindre la table "ratings"" en fonction des films présents dans "types", nous utiliserions la commande semi_join().

```
#requête 2 - semi join - filtrage des enregistrements de ratings
dp2b <- dplyr::semi_join(dfRatings,dfTypes,by="title")

#print
print(head(dp2b,10))
```

```
## # A tibble: 10 x 3
##     user title                          rating
##    <dbl> <chr>                           <dbl>
##  1    1. GoldenEye (1995)                  3.
##  2    2. Jerry Maguire (1996)              4.
##  3    2. Sabrina (1995)                    3.
##  4    2. Sense and Sensibility (1995)      5.
##  5    2. Liar Liar (1997)                  1.
##  6    2. Rainmaker, The (1997)             4.
##  7    3. Liar Liar (1997)                  2.
##  8    3. Mother (1996)                     5.
##  9    3. U Turn (1997)                     3.
## 10    4. Liar Liar (1997)                  5.
```

Et nous avons bien les 80 enregistrements.

```
print(dim(dp2b))
```

```
## [1] 80  3
```

c. Requête 3 : Genre des films visionnés par les utilisateurs

J'attendais un peu le package dplyr au tournant ici. J'ai eu la joie de mieux faire connaissance avec l'instruction sibylline **%>%** qui signifie *grosso modo* : "envoyer comme premier paramètre de la fonction située à droite de %>% la résultante de la commande située à gauche".

Dans notre cas, voici la séquence :

```
#nombre de films par genre
dp2[c("user","Action","Comedy","Drama","Romance")] %>% group_by(user) %>% summarize_all(sum)
```

```
## # A tibble: 10 x 5
##     user Action Comedy Drama Romance
##    <dbl>  <dbl>  <dbl> <dbl>   <dbl>
```

```
##  1    1.     1.     0.    0.      0.
##  2    2.     0.     2.    3.      3.
##  3    3.     1.     2.    0.      0.
##  4    4.     1.     2.    0.      0.
##  5    5.     2.     3.    0.      0.
##  6    6.     1.     6.   11.      6.
##  7    7.     4.     6.    8.      4.
##  8    8.     2.     2.    1.      0.
##  9    9.     0.     1.    1.      0.
## 10   10.     2.     4.    8.      2.
```

Lu de gauche à droite, voici le détail des opérations :

- Nous effectuons une projection sur les champs de dp2 en nous restreignant aux champs {user, Action, Comedy, Drama, Romance} ;
- Puis nous réalisons un regroupement des enregistrements selon le champ user à l'aide de l'instruction group_by() ;
- Et nous calculons la somme consolidée des valeurs pour les champs restants.

Le résultat est conforme. Aucun doute là-dessus. Mais voilà le type même de commande qui flatte les initiés et effraie les néophytes. Elle est nettement moins glamour écrite sous la forme suivante :

```
summarize_all(group_by(dp2[c("user","Action","Comedy","Drama","Romance")],user),sum)
```

```
## # A tibble: 10 x 5
##     user Action Comedy Drama Romance
##    <dbl>  <dbl>  <dbl> <dbl>   <dbl>
##  1    1.     1.     0.    0.      0.
##  2    2.     0.     2.    3.      3.
##  3    3.     1.     2.    0.      0.
##  4    4.     1.     2.    0.      0.
##  5    5.     2.     3.    0.      0.
##  6    6.     1.     6.   11.      6.
##  7    7.     4.     6.    8.      4.
##  8    8.     2.     2.    1.      0.
##  9    9.     0.     1.    1.      0.
## 10   10.     2.     4.    8.      2.
```

Et pourtant, nous avons le bon résultat et, surtout, nous observons différemment (mieux ? moins bien ? question de point de vue) l'enchaînement des instructions.

d. Requête 4 : Notes moyennes attribués aux films dramatiques visionnés

Maintenant que nous savons utiliser **%>%**, nous l'exploitons pleinement dans la dernière requête.

```
#moyenne des notes par utilisateur pour Drama
dp2[c("Drama","user","rating")] %>% filter(Drama > 0) %>% group_by(user) %>% summarize_at(c("rating"),mean)
```

```
## # A tibble: 6 x 2
##    user rating
##   <dbl>  <dbl>
## 1    2.   4.33
## 2    6.   3.55
## 3    7.   4.50
## 4    8.   5.00
## 5    9.   3.00
## 6   10.   4.38
```

Ecrire des commandes complexes en une seule ligne de code est toujours très flatteur, il faut le reconnaître.

6. Conclusion

Qu'importe le flacon pourvu qu'on ait l'ivresse disait un sage (ou un soiffard, question de point de vue également). Notre objectif était de montrer qu'il était possible de réaliser des requêtes avec jointures directement sous R. Le contrat est rempli et c'est une très bonne chose.

Après, le choix des outils dépend du cahier des charges, des considérations de performances, etc. Si nous voulons effectuons une requête avec 3 tables par exemple, on doit pouvoir le faire directement en SQL. Avec les outils natifs de R ou avec dplyr, il faudrait procéder en deux temps. A contrario, il doit y avoir des cas où merge() ou encore les outils de dplyr sont plus adéquats. Bref, il nous appartient de déterminer l'outil le plus adapté en fonction de notre contexte et de nos attentes. Comme toujours…