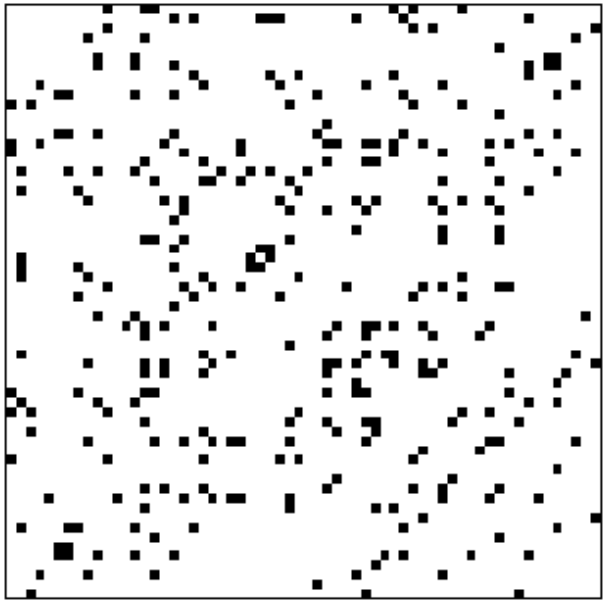
Travailler avec des matrices creuses sous R

3èmes Rencontres R - Montpellier

B. Thieurmel - bt@datak.fr

27 Juin 2014

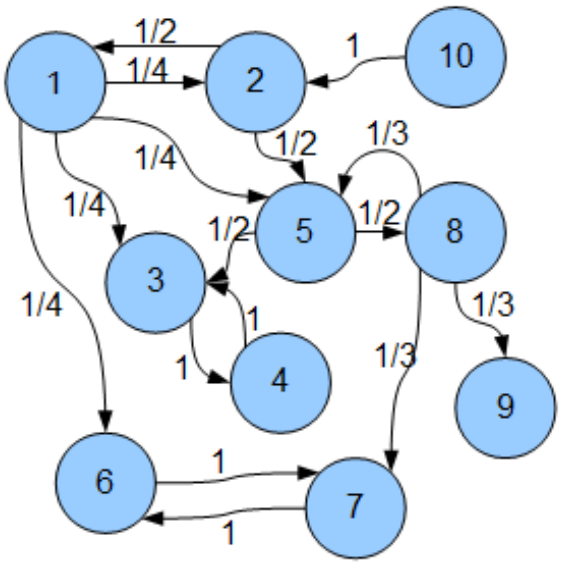
DATAKnowledge



Definition :

- Une matrice contenant beaucoup de zéros
- Des zéros, et non des données manquantes...
- Beaucoup ? 10%, 20%, 50%, 66%, 80%, ... ?

- Résolution d'équations aux dérivées partielles
- Traitement de données de compositions
- Théorie des réseaux / Système de recommendations
- Text mining



$$Q = \begin{pmatrix} 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1/4 & 0 & 0 & 1 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/2 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 \\ 1/4 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

.

Idée :

- Réduire l'information stockée aux valeurs non-nulles de la matrice

Principales méthodes de stockage :

- La plus intuitive / simple :
  - un ensemble de triplets (ligne, colonne, valeur)
- Plus performante / moins gourmande :
  - en utilisant un système de stockage compressé sur les lignes ou les colonnes

Notations :

- i indice des lignes et j indice des colonnes
- k_{ij} la valeur non-nulle en position (i, j)
- nz nombre de valeurs non-nulles

Idée :

- Réduire l'information stockée aux valeurs non-nulles de la matrice

Principales méthodes de stockage :

- La plus intuitive / simple :
  - un ensemble de triplets (ligne, colonne, valeur)
- Plus performante / moins gourmande :
  - en utilisant un système de stockage compressé sur les lignes ou les colonnes

Notations :

- i indice des lignes et j indice des colonnes
- k_{ij} la valeur non-nulle en position (i, j)
- nz nombre de valeurs non-nulles

Idée :

- Réduire l'information stockée aux valeurs non-nulles de la matrice

Principales méthodes de stockage :

- La plus intuitive / simple :
  - un ensemble de triplets (ligne, colonne, valeur)
- Plus performante / moins gourmande :
  - en utilisant un système de stockage compressé sur les lignes ou les colonnes

Notations :

- i indice des lignes et j indice des colonnes
- k_{ij} la valeur non-nulle en position (i, j)
- nz nombre de valeurs non-nulles

Triplet :

- stockage en triplets (i, j, k_{ij}) via 3 vecteurs :
  - $I \in \mathbb{R}^{nz}$: indices des lignes
  - $J \in \mathbb{R}^{nz}$: indices des colonnes
  - $K \in \mathbb{R}^{nz}$: valeurs non-nulles
- packages **Matrix** et **slam**

Compression des colonnes (resp. des lignes) :

- stockage de l'information via 3 vecteurs :
  - $I(resp.J) \in \mathbb{R}^{nz}$:les indices i (resp. j) des lignes (resp. des colonnes)
  - $P \in \mathbb{R}^{np}, np \leq nz$: les indices des pointeurs p des colonnes (resp. des lignes)
  - $K \in \mathbb{R}^{nz}$: les valeurs non-nulles
- packages **Matrix** (colonnes ou lignes) et **spam** (lignes)

Triplet :

- stockage en triplets (i, j, k_{ij}) via 3 vecteurs :
  - $I \in \mathbb{R}^{nz}$: indices des lignes
  - $J \in \mathbb{R}^{nz}$: indices des colonnes
  - $K \in \mathbb{R}^{nz}$: valeurs non-nulles
- packages **Matrix** et **slam**

Compression des colonnes (resp. des lignes) :

- stockage de l'information via 3 vecteurs :
  - $I(resp.J) \in \mathbb{R}^{nz}$:les indices i (resp. j) des lignes (resp. des colonnes)
  - $P \in \mathbb{R}^{np}, np \leq nz$: les indices des pointeurs p des colonnes (resp. des lignes)
  - $K \in \mathbb{R}^{nz}$: les valeurs non-nulles
- packages **Matrix** (colonnes ou lignes) et **spam** (lignes)

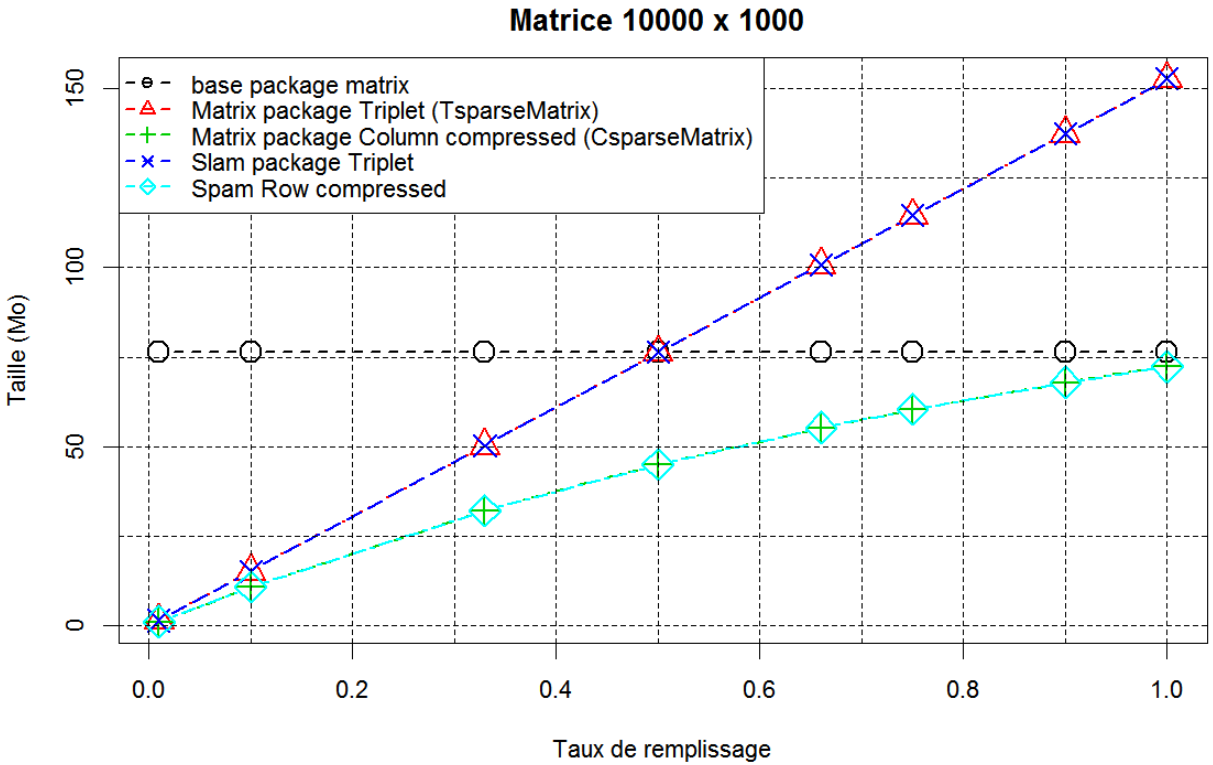
```
> # données
> k <- c(4,5,7,1,3,2,6)
> i <- c(1,1,5,4,2,1,3)
> j <- c(3,4,4,1,2,2,4)

> my.matrix <- matrix(ncol=4, nrow = 5, 0)
> s <- sapply(1:7, function(ii){
+   my.matrix[i[ii], j[ii]] <<- k[ii]
+ })
> my.matrix

     [,1] [,2] [,3] [,4]
[1,]    0    2    4    5
[2,]    0    3    0    0
[3,]    0    0    0    6
[4,]    1    0    0    0
[5,]    0    0    0    7
```

```
> require(Matrix)
> Matrix.Tsparse<-sparseMatrix(i = i, j = j, x = k, giveCsparse = FALSE)
> dim(Matrix.Tsparse)

[1] 5 4

> str(Matrix.Tsparse)

Formal class 'dgTMatrix' [package "Matrix"] with 6 slots
  ..@ i       : int [1:7] 0 0 4 3 1 0 2
  ..@ j       : int [1:7] 2 3 3 0 1 1 3
  ..@ Dim     : int [1:2] 5 4
  ..@ Dimnames:List of 2
  .. ..$ : NULL
  .. ..$ : NULL
  ..@ x       : num [1:7] 4 5 7 1 3 2 6
  ..@ factors : list()

> Matrix.Tsparse

5 x 4 sparse Matrix of class "dgTMatrix"

[1,] . 2 4 5
[2,] . 3 . .
[3,] . . . 6
[4,] 1 . . .
[5,] . . . 7
```

```
> require(slam)
> slam.matrix <- simple_triplet_matrix(i = i, j = j, v = k)
> # (=) as.simple_triplet_matrix(my.matrix)
> dim(slam.matrix)

[1] 5 4

> str(slam.matrix)

List of 6
 $ i       : int [1:7] 1 1 5 4 2 1 3
 $ j       : int [1:7] 3 4 4 1 2 2 4
 $ v       : num [1:7] 4 5 7 1 3 2 6
 $ nrow    : int 5
 $ ncol    : int 4
 $ dimnames: NULL
 - attr(*, "class")= chr "simple_triplet_matrix"

> slam.matrix

A 5x4 simple triplet matrix.
```

```
> Matrix.Csparse<-sparseMatrix(i = i, j = j, x = k, giveCsparse = TRUE)
> # (=) Matrix(my.matrix, sparse = TRUE)
> dim(Matrix.Csparse)

[1] 5 4

> str(Matrix.Csparse)

Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
  ..@ i       : int [1:7] 3 0 1 0 0 2 4
  ..@ p       : int [1:5] 0 1 3 4 7
  ..@ Dim     : int [1:2] 5 4
  ..@ Dimnames:List of 2
  .. ..$ : NULL
  .. ..$ : NULL
  ..@ x       : num [1:7] 1 2 3 4 5 6 7
  ..@ factors : list()

> Matrix.Csparse

5 x 4 sparse Matrix of class "dgCMatrix"

[1,] . 2 4 5
[2,] . 3 . .
[3,] . . . 6
[4,] 1 . . .
[5,] . . . 7
```

```
> require(spam)
> spam.matrix <- spam(x = as.numeric(my.matrix), nrow = 5, ncol = 4)
> dim(spam.matrix)

[1] 5 4

> str(spam.matrix)

Formal class 'spam' [package "spam"] with 4 slots
  ..@ entries    : num [1:7] 2 4 5 3 6 1 7
  ..@ colindices : int [1:7] 2 3 4 2 4 1 4
  ..@ rowpointers: int [1:6] 1 4 5 6 7 8
  ..@ dimension  : int [1:2] 5 4

> spam.matrix

     [,1] [,2] [,3] [,4]
[1,]    0    2    4    5
[2,]    0    3    0    0
[3,]    0    0    0    6
[4,]    1    0    0    0
[5,]    0    0    0    7
Class 'spam'
```



- "triplet" avec **Matrix** et **slam** similaires. Intéressants à partir de 50% de zéros
- stockage Column compressed via **Matrix** ou Row compressed via **spam** avec des performances identiques

Package	Sparse objet	Description
Matrix	Matrix object	Stockage, manipulation, statistiques descriptives et calculs algèbriques (colSums, colMeans, rowSums, rowMeans, rBind, cBind, chol, hilbert, ...)
slam	Slam object	Lecture/écriture, stockage, manipulation, statistiques descriptives (rowapply_simple_triplet_matrix, crossapply_simple_triplet_matrix, abind, read/write_stm_CLUTO, read/write_stl_MC, rollup, row_sums, col_sums, ...)
spam	Spam object	Lecture / conversion, stockage, manipulation, statistiques descriptives et calculs algèbriques (read.HB (Harqwell-Boeing), read.MM (MatrixMarket), as.dgCMatrix.spam, apply, cbind, rbind, permutation, rowSums, rowMeans, diag, chol, ...)
glmnet	Matrix object	Extremely efficient procedures for fitting the entire lasso or elastic-net regularization path for linear regression, logistic and multinomial regression models, poisson regression and the Cox
irlba	Matrix object	A fast and memory-efficient method for computing a few approximate singular values and singular vectors of large matrices

Base de données :

- données du prix Netflix
- 100 millions de notes de 480 119 membres sur 17 770 films

	Film 1	Film 2	...	Film 17770
User 1		2		5
User 2				
...	3			
User 480119	1			4

Taux de remplissage :

- 98.2% de zéros

- une matrice ? : 65 Go nécessaires...
- en triplet : 1.5 Go
- en Column (ou Row) Compressed : 1.1 Go

- Les fonctions proposées prennent en compte les zéros
- On aimerait la moyenne des notes, quand il y en a, par film
- col_sums et apply ?

```
> netflix.slam <- readRDS("D:/netflix/netflix/netflixSlam.rds")
> system.time(res <- col_sums(netflix.slam) / colapply_simple_triplet_matrix(
+  netflix.slam, FUN = function(x) length(which(x != 0))))

   user  system elapsed
  71.54   25.07   96.61

> res[1:10]

 [1] 3.749543 3.558621 3.641153 2.739437 3.919298 3.084396 2.129032 3.189805
 [9] 2.621053 3.180723
```

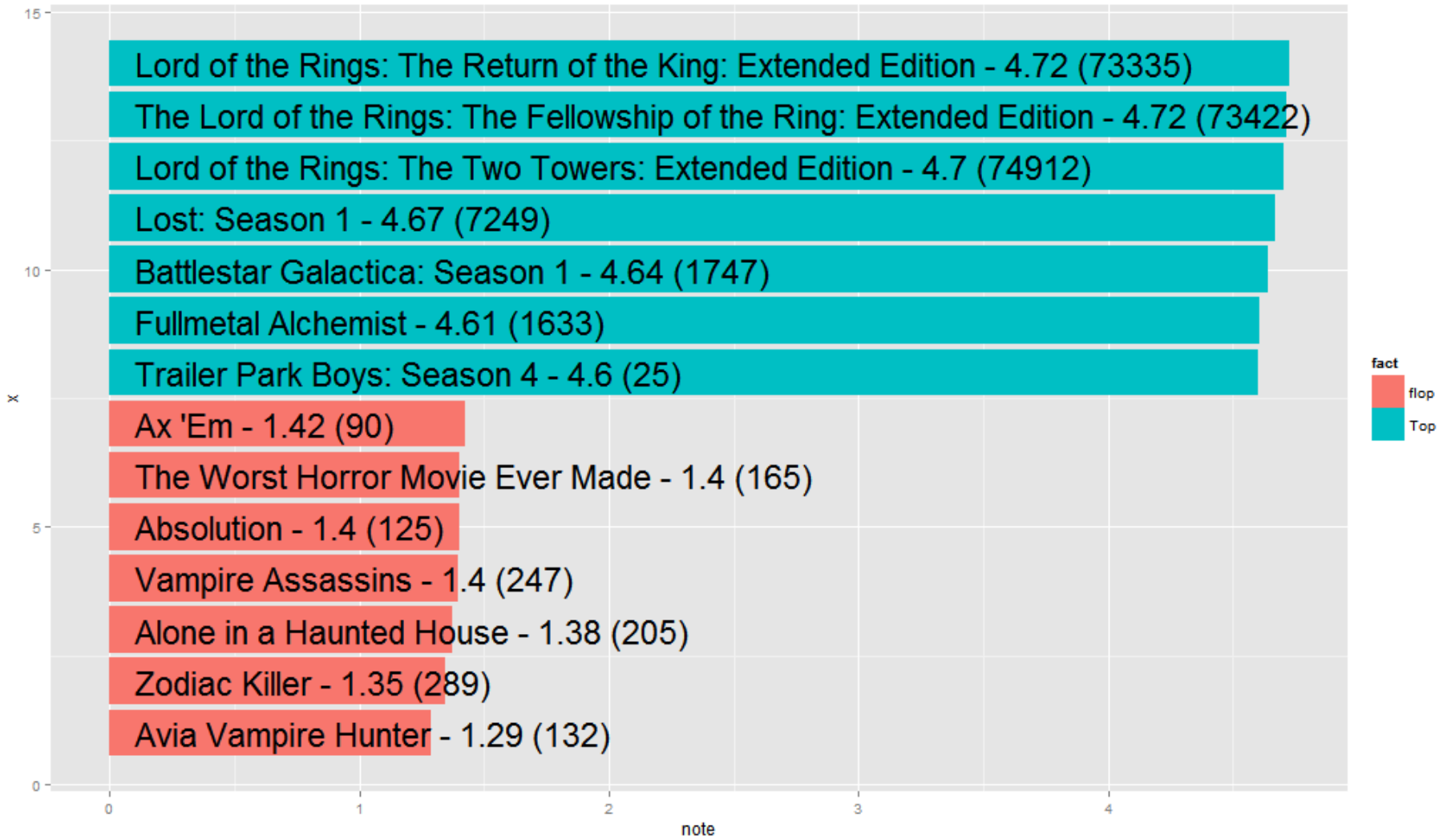
Un peu long...

- Construisons notre fonction via *Matrix*, *Rcpp* et *RcppEigen*

```
library(inline); library(Rcpp); library(RcppEigen)

sparse_mean <- cxxfunction(signature(CMat = "numeric", TRANSPOSE = "integer"),
  'using Eigen::MappedSparseMatrix; using Eigen::SparseMatrix;

  Eigen::MappedSparseMatrix<double> MapM(as<MappedSparseMatrix<double> >(CMat));
  SparseMatrix<double> SpM;
  NumericVector cont_transpose(TRANSPOSE);

  std::vector<double> mean;
  double cpt, val;

  if(cont_transpose[0]==0){
    SpM=SparseMatrix<double>(MapM);
  }else{
    SpM=SparseMatrix<double>(MapM.transpose());
  }

  for (int i=0; i<SSpM.outerSize(); ++i){
    cpt=0; val=0;
    for (SparseMatrix<double>::InnerIterator it(SpM,i); it; ++it)    {
        val+=it.value(); cpt+=1;
    }
    mean.push_back(val/cpt);
  }
  return Rcpp::wrap(mean);', plugin = "RcppEigen")
```

sparse_mean.R

```
> system.time(netflix <- readRDS("D:/netflix/netflix/netflixCsparse.rds"))

   user  system elapsed
   6.21    0.33    8.37

> class(netflix)

[1] "dgCMatrix"
attr(,"package")
[1] "Matrix"

> dim(netflix)

[1] 480189  17770

> source("D:/netflix/sparse_mean.R")

cygwin warning:
  MS-DOS style path detected: C:/PROGRA~1/R/R-31~1.0/etc/x64/Makeconf
  Preferred POSIX equivalent is: /cygdrive/c/PROGRA~1/R/R-31~1.0/etc/x64/Makeco
  CYGWIN environment variable option "nodosfilewarning" turns off this warning.
  Consult the user's guide for more details about POSIX paths:
    http://cygwin.com/cygwin-ug-net/using.html#using-pathnames

> system.time(mean.rate.film <-sparse_mean(netflix, TRANSPOSE = FALSE))

   user  system elapsed
   1.89    2.79    6.30
```


Lord of the Rings: The Return of the King: Extended Edition - 4.72 (73335)
The Lord of the Rings: The Fellowship of the Ring: Extended Edition - 4.72 (73422)
Lord of the Rings: The Two Towers: Extended Edition - 4.7 (74912)
Lost: Season 1 - 4.67 (7249)
Battlestar Galactica: Season 1 - 4.64 (1747)
Fullmetal Alchemist - 4.61 (1633)
Trailer Park Boys: Season 4 - 4.6 (25)
Ax 'Em - 1.42 (90)
The Worst Horror Movie Ever Made - 1.4 (165)
Absolution - 1.4 (125)
Vampire Assassins - 1.4 (247)
Alone in a Haunted House - 1.38 (205)
Zodiac Killer - 1.35 (289)
Avia Vampire Hunter - 1.29 (132)
15
10
5
0
x
0
1
2
3
4
note
fact
flop
Top

- SVD partielle, *irlba*

```
> system.time(psvd <- irlba(netflix))

   user  system elapsed
  22.56    0.51   30.35
```

- suivie d'un kmeans, *stats*

```
> system.time(k <- kmeans(psvd$v, centers = 200))

   user  system elapsed
   0.75    0.00    0.88
```

- Somme pondérée des lignes

```
> poids.film <- runif(17770)
> system.time(row_sums_poids <- netflix%*%poids.film)

   user  system elapsed
   0.53    0.00    0.53
```

- Intel(R) Core(TM) i7-2670QM 2.20 GHz, 8 Go RAM - Windows 7

Travailler avec des matrices creuses sous R

3èmes Rencontres R - Montpellier

B. Thieurmel - bt@datak.fr

27 Juin 2014

DATAKnowledge