Introduction au machine learning

Premiers pas sur R

Denis Oblin
Denis.oblin@memorandum.pro **06 71 62 74 92**

Romain Jouin
Romain.jouin@memorandum.pro **06 52 86 87 30**

Pourquoi R ?

Un logiciel de développement scientifique spécialisé dans le calcul et l'analyse statistique

- un langage,
- un environnement,
- un projet open source (projet GNU),
- un logiciel multiplateforme (Linux, Mac, Windows),

Fonctionnalités

- Gestionnaire de données : Lecture, manipulation, stockage.
- Algèbre linéaire : Opérations classiques sur vecteurs, tableaux et matrices
- Statistiques et analyse de données : Dispose d'un grand nombre de méthodes d'analyse de données (des plus anciennes et aux plus récentes)
- Moteur de sorties graphiques : Sorties écran ou fichier
- Système de modules
- Alimenté par la communauté (+ de 2000 extensions !)
- Interface « facile » avec C/C++, Fortran,. .

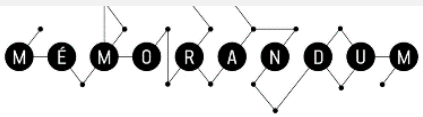
Un peu d'histoire

- 1970s développ. de S au Bell labs.
- 1980s développ. de S-PLUS au AT&T. Lab
- 1993 développ. de R sur le modèle de S par Robert Gentleman et Ross Ihaka.
- 1995 dépôts des codes sources sous licence GNU/GPL
- 1997 élargissement du groupe
- 2002 la fondation R dépose ses statuts sous la présidence de Gentleman et Ihaka

Un développement entièrement bénévole

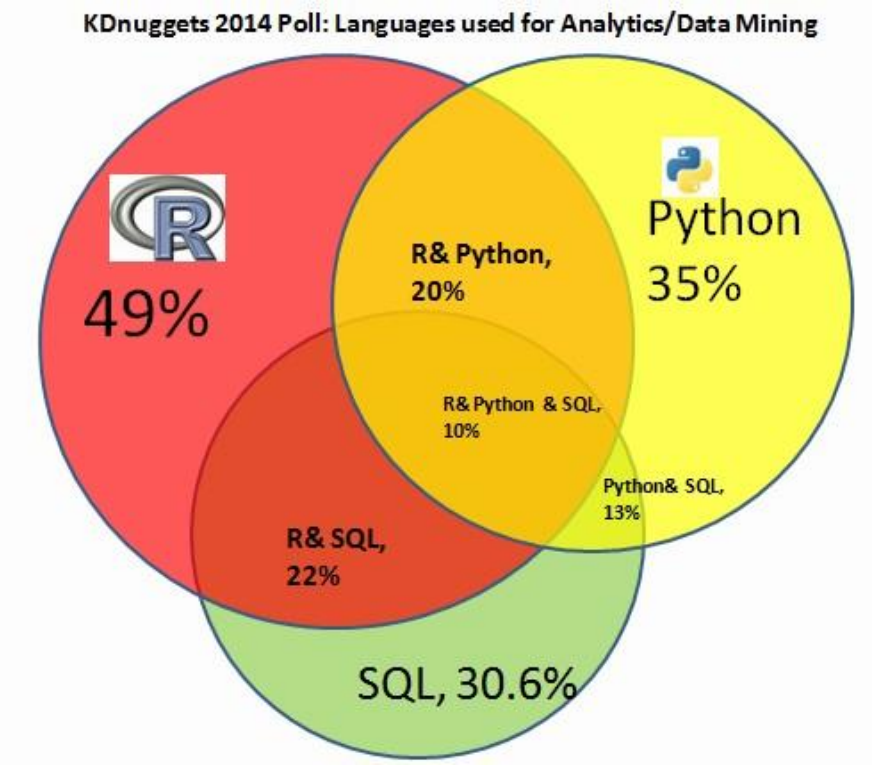
- Participation de nombreux chercheurs (>2000 packages)

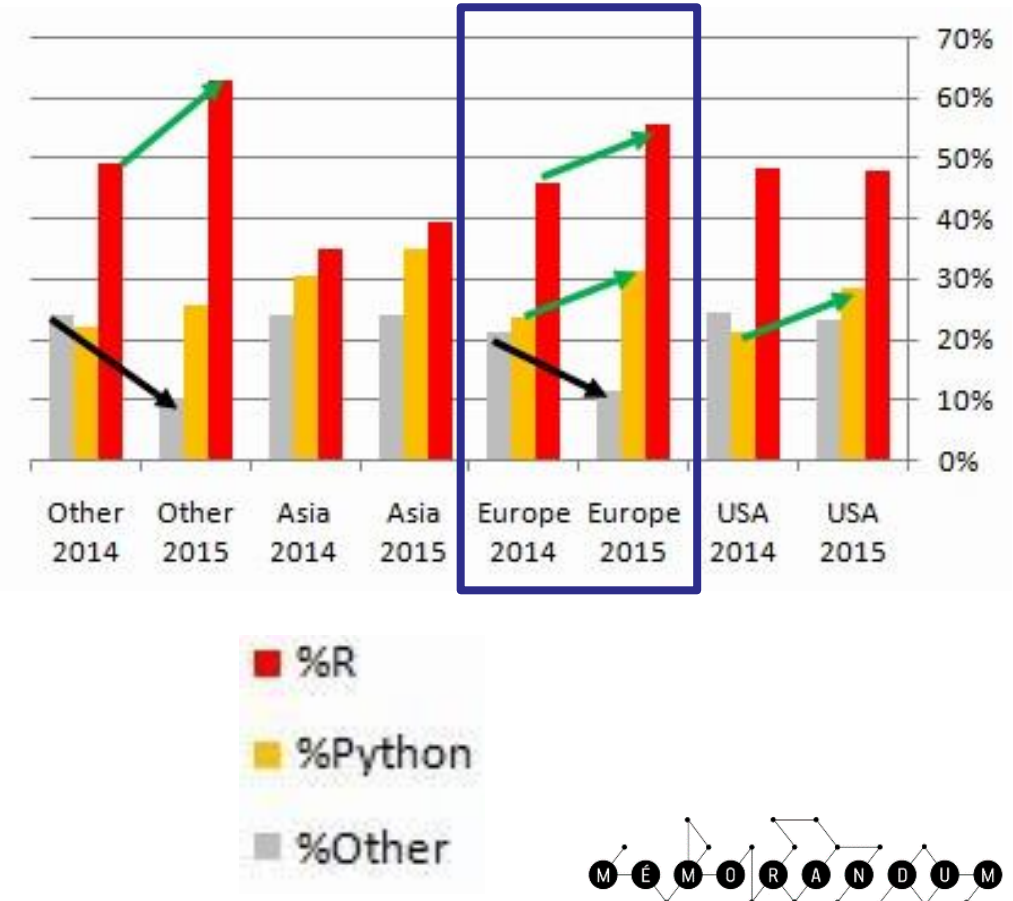
- Le projet : http://www.r-project.org/
- Documentation package (CRAN) : http://cran.r-project.org/
- Articles : http://journal.r-project.org/, https://www.r-bloggers.com/



Pourquoi R ?

Quel langage de programmation utilisez vous pour vos travaux analytics / data mining / data science





Installation et première découverte

Installer

- R, version 3.2.5 : **https://www.r-project.org/**
- Rstudio : https://www.rstudio.com/products/rstudio/download/

Exemple de ivres en ligne R

- Logiciel R et programmation, ewen Gallic
- Introduction à R – Max Brruciamcchi
- Notes de cours sur R – Anne Philippe
- Introduction à la programmation en R - Vincent Goulet
- R et espace – Hadrien Commenge
- Principales commandes R et python –

Python : distribution anaconda : https://www.continuum.io/downloads

Répertoire de travail :
- L'identifier : getwd(), le changer : setwd(« chemin ») :
  - sous Windows changer les « \ » du chemin par des « / »

Affectation, assignation de variable : « <- » ou « = »
- nomVariable <- ValeurVariable ; nomVariable =ValeurVariable

Obtenir de l'aide
- Champ de recherche de la zone en bas à droite de R studio
- Champ de commande : e l'aide : ?*mafonction*,

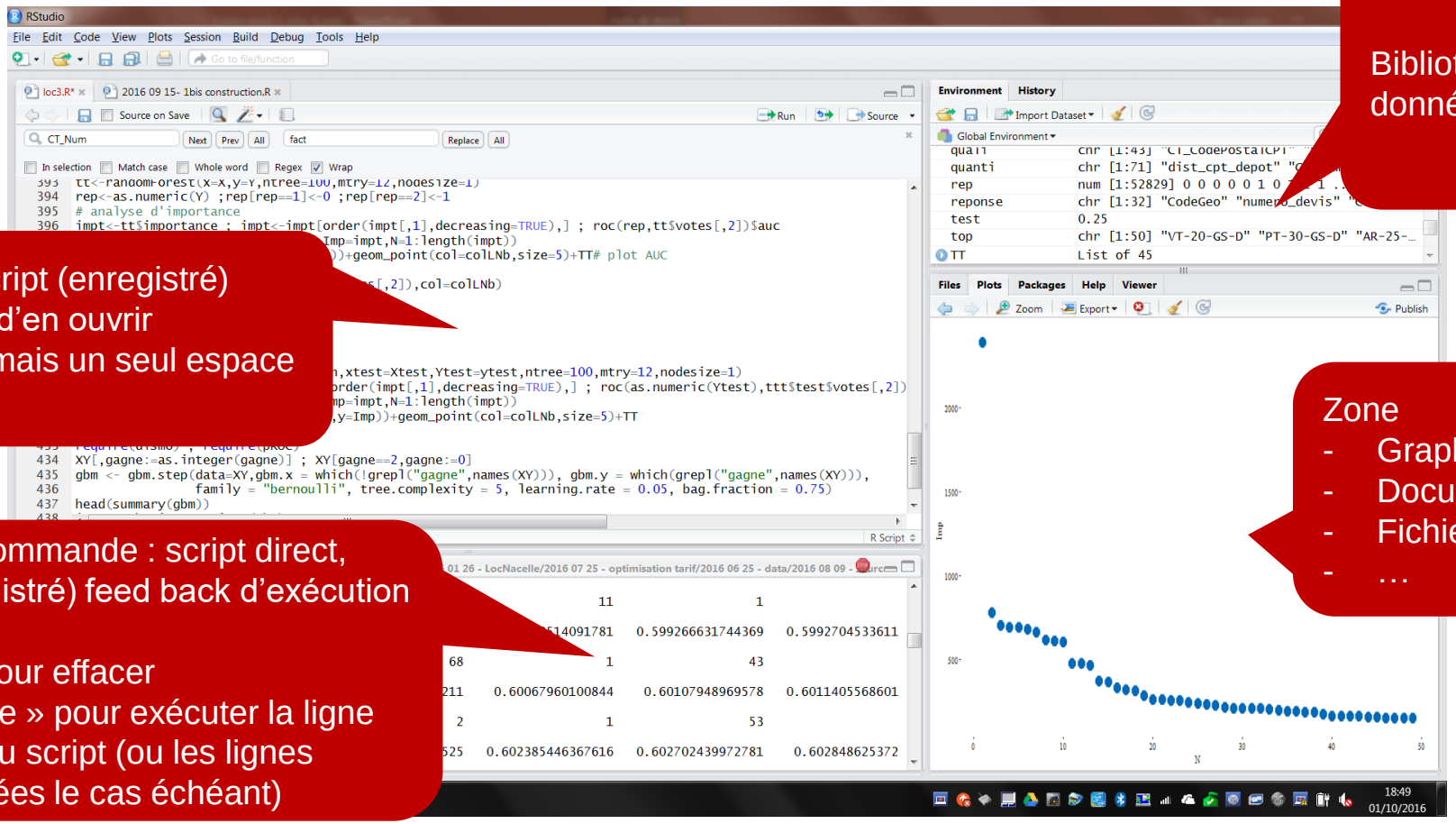
« ; » séparer des instructions sur une même ligne

Commentaire
- # en début de ligne
- « ctrl shift c » pour passer tout un bloc sélectionné en commentaire

Paragraphes (possibilité de condenser des paragraphes de code)
- ### en fin de ligne pour définir des paragraphes
- { }

Attention : sensible à la casse (majuscule minuscule)

Installation et première découverte



Bibliothèque des données

Zone de script (enregistré)
Possibilité d'en ouvrir plusieurs, mais un seul espace de travail

Zone
- Graphique
- Documentation
- Fichiers
- …

Zone de commande : script direct, (non enregistré) feed back d'exécution

« ctrl L » pour effacer
« ctrl entrée » pour exécuter la ligne courante du script (ou les lignes sélectionnées le cas échéant)

Plusieurs consoles R studio peuvent être ouvertes et exécutés simultanément (les espaces de travail sont indépendant)

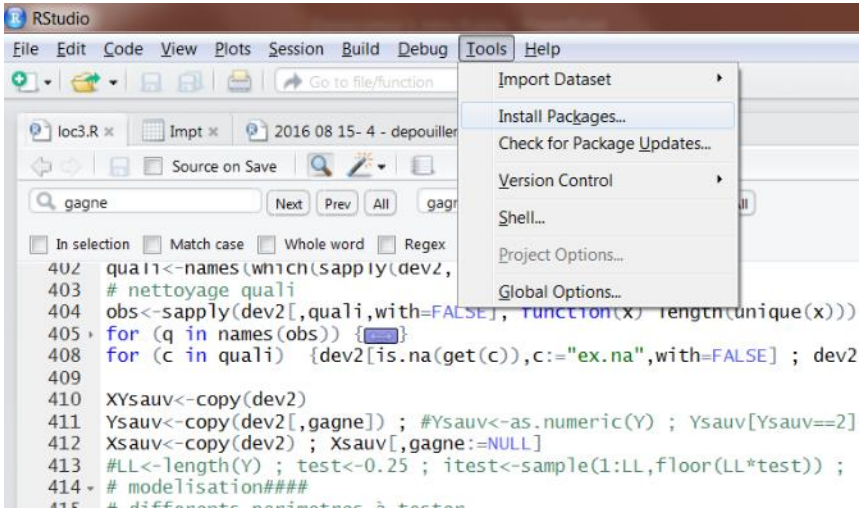
Package

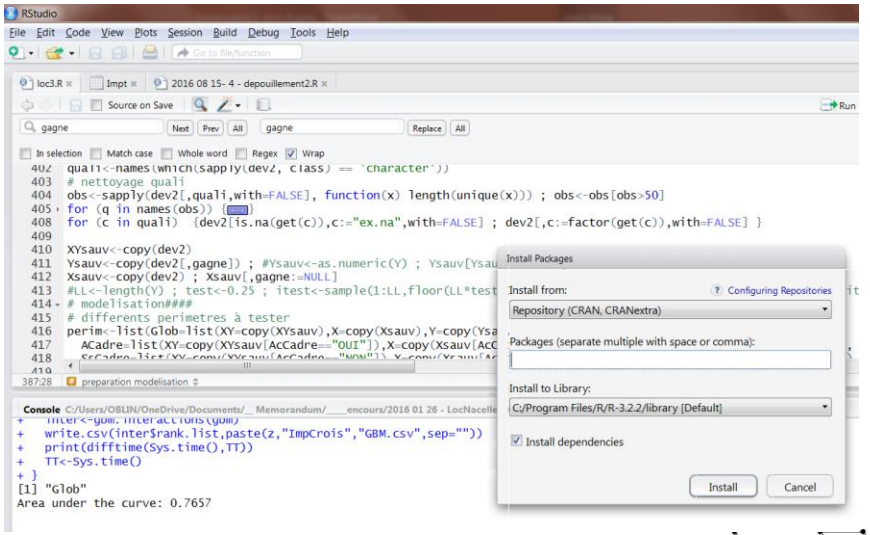
Des fonctions qui enrichissent l'environnement de base de R

– Des packages pour chaque algo, des fonctions graphiques, des manipulations de données …

A la première utilisation

– Installer le package sur votre PC →
  • À faire une seule fois
– Au début de votre programme (à chaque fois) : appeler le package pour activer les fonctions
  • require(« monpackage »)
  • Ou library(« monpackage »)





Les types de données

Character
- Entre guillemet ‘’ chaine de caractères ‘’

Integer

numeric

Logical : TRUE, FALSE
- Possibilité de faire des additions
  - Sum(is.na(vecteur)) : compte les valeurs manquantes

Transformation : as.numeric, as.character, as.logical …(crée des NA si impossible)
- As.logical(c(1,0,5)) : TRUE, FALSE, NA

- Données particulières
  - « NA » : valeur manquante : teste : is.na(*Valeur*)
  - « NaN » : résultat numérique aberrant ,
  - « Inf » et « -Inf »
  - « NULL » objet nul

- Exemple fonctions sur les numeric
  - Floor(), ceiling() → partie entière, arrondis enter au dessus
  - Round(2/3,3) arrondis

- Exemple fonctions sur les logical
  - Et : & Ou : | xor : terme à terme dans un vecteur

- Fonctions sur les caractères (voir aussi package « stringr »)
  - phrase = "je manipule les caracteres"
  - nchar(phrase)
  - substr(phrase,start=1,stop=8)
  - strsplit(phrase,"p")
  - mots = strsplit(phrase," ")
  - mots = unlist(mots)
  - lettres = strsplit(phrase,NULL)
  - length(lettres[[1]])
  - Mettre en majuscule : toupper(phrase) ;
  - Mettre en miniscule : tolower("AAA")
  - Trouver une chaine de caractere : grep("le",mots) ;
    - grepl : trouver toutes les occurrences
  - sub("je","il",phrase)

Les types de données
→ Facteurs

Un facteur est un vecteur de variables catégorielles. Ce format est requis pour plusieurs algorithme pour la manipulation de variables qualitatives

– Ce format fige la liste des valeurs que peut prendre une valeur qualitative

Lors de l'acquisition en mode dataframe (read.csv, read.table ..) R transforme automatiquement les qualitatifs en facteur

– Pour garder un format 'character' ouvert ajouter le paramètre stringsAsFactors
– Exemple read.csv(« fichier.csv », **stringsAsFactors=FALSE**)

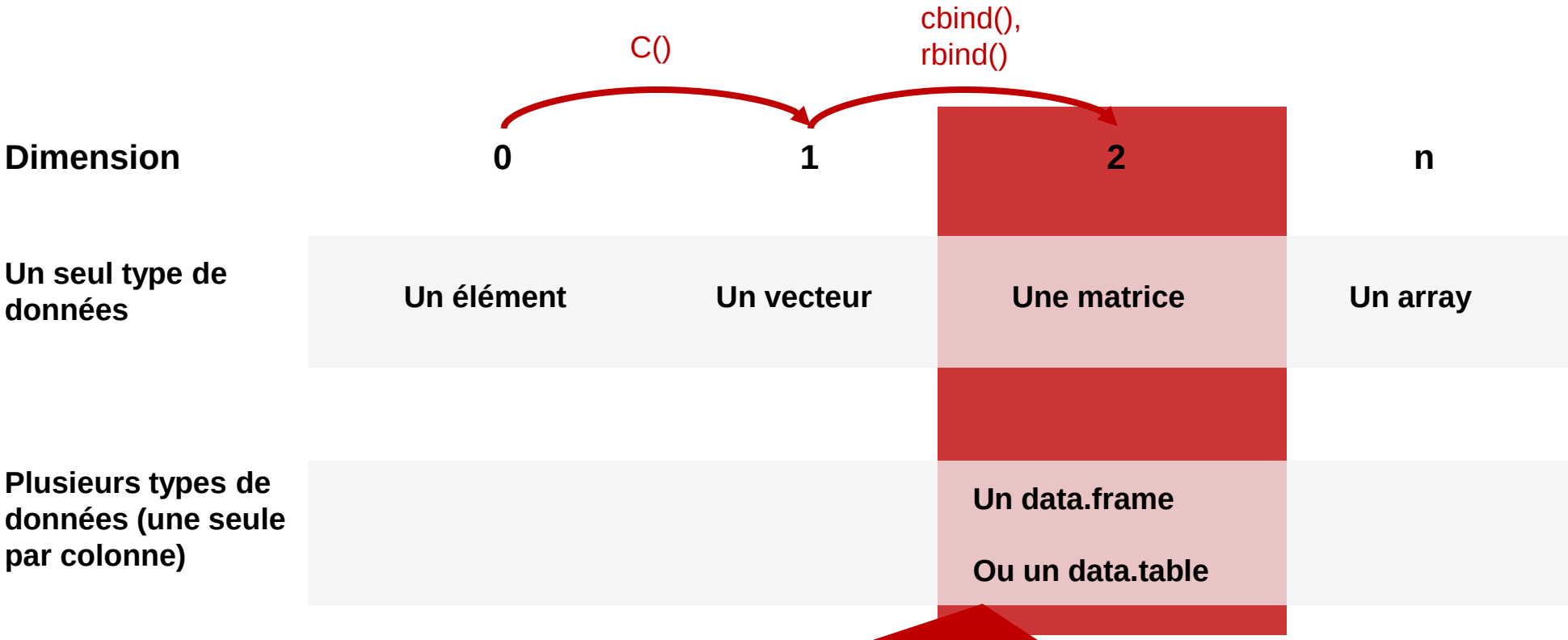
Les niveaux du facteur peuvent être ordonnés ou pas.

– x <- factor(sample(c("thesard", "CR", "MdC"), 15, replace = TRUE)) : non ordonné
– x <- ordered(sample(c("thesard", "CR", "MdC"), 15, replace = TRUE)) : ordonné

Les niveaux peuvent être définis manuellement

x <- factor(sample(c("thesard", "CR", "MdC"), 15, replace = TRUE),**levels=**c("CR", "MdC", "thesard",''autre'')

Les structures de données



Associé au package « data.table »

Chargement rapide avec fread()
Grande souplesse de manipulation de données
Attention : une assignation ne créée pas de nouvel objet : utiliser copy() pour créer un doublon indépendant

Les structures de données
→ Vecteurs

Propriétés
- objet le plus élémentaire sous R,
- collection d'entités de même nature,
- mode (ou type) défini par la nature des entités qui le composent. (homogène)

Opération entre vecteurs : terme à terme par défaut
- Vecteur 1 + vecteur 2, Vecteur 1 - vecteur 2
- Vecteur 1 * vecteur 2, Vecteur 1 / vecteur 2
- Vecteur 1 > vecteur 2, Vecteur 1 > vecteur 2
  - > < >= <= == !=

Ils doivent avoir la même taille ! R recycle : pratique mais dangereux !

- Les générer :
  - C() : opérateur de base
  - 4:9
    → 4,5,6,7,8,9
    - Attention : 1:6/2 diffèrent de 1:(6/2)
  - seq(from,to,by=) ou seq(2, 10, length.out = 6)
    - 1:3:10 équivalent à seq(1,10,by=3)
  - rep(1, 3)
    → 1 1 1
  - rep(c("A", "B", "C"), c(3, 2, 4))
    → "A" "A" "A" "B" "B" "C" "C" "C" "C"
  - rep(c(TRUE, FALSE), each = 2)
    → TRUE FALSE

Opération sur un vecteur
- prod, sum, max, min, range, which.min, which.max, length, table,unique
- Analyse en cumul : cumsum, cumprod, cummin, cummax

Comparaison de 2 vecteurs (génere un nouveau vecteur)
- Intersect(vecA,vecB) ,union(vecA,vecB),setdiff(vecA,vecB)

Un élément est il dans un vecteur x %in% vecteur (résultat logical)

Les structures de donnée
→Matrice

Un vecteur réparti en colonne ! (cf mode de construction ci contre). Conséquence : un seul type de données toléré !

Opération entre matrice : terme à terme par défaut

- + - * / : terme à terme
- %*% : produit matriciel
- crosprod() : produits scalaire
- t() : transpose une matrice
- diag() : diagonale (donc un vecteur)
- cbind, rbind →
  - Marche aussi pour data.frame, attention aux factor !)
- which(x, arr.ind = FALSE): restitue les indices lignes colonne répondant à une condition (x matrice logique)

- Les générer :
  - matrix(c(2,0,1,3), nrow=2,ncol=2, byrow=TRUE)
  - A <- matrix(c(4, 2, 8, -3), 2, 2)

```
> a <- matrix(1, 2, 3)
> b <- matrix(2, 2, 3)
> c(a, b)

 [1] 1 1 1 1 1 1 2 2 2 2 2 2
```

```
> cbind(a, b)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    1    2    2    2
[2,]    1    1    1    2    2    2

> rbind(a, b)

     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
[3,]    2    2    2
[4,]    2    2    2
```

b <- c(2, 3) ; solve(A, b)

det : calcule le déterminant d'une matrice ;

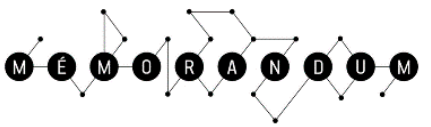
eigen : calcule valeurs propres et vecteurs propres d'une matrice ;

svd : calcule la décomposition en valeurs singulières.

Les structures de données
→ Comparaison matrice, data.frame, data.table

	Matrice	data.frame	Data.table
	Toutes les cellules de même nature (numeric, character, .)	Des vecteurs homogènes mais pas forcément tous de la même nature (colonnes d'entiers, colonnes de numeric, colonne de character…	
Création	Natif dans R MaMatrice <- matrix(c(2,0,1,3), now=2,ncol=2, byrow=TRUE) Données livrées comme un grand vecteur puis réparties en tableau → par défaut par colonne. Attention : R recycle : → matrix(1:3, nrow = 2, ncol = 2)	Natif dans R MonDF <- data.frame (col1=1:5, col2=c("a", "b", "c", "d", "e"), col3 = c(2,8,5,3,0))	Require(data.table) MaDT <- data.table (col1=1:5, col2=c("a", "b", "c", "d", "e"), col3 = c(2,8,5,3,0))
Acquisition	?	Read.table(« fichier ») : read.csv (lire l'aide pour les paramètres) Attention : les characters sont automatiquement enregistrés en factor : utiliser : stringsAsFactors=FALSE)	Fread(« fichier ») (sans argument le plus souvent
transformation	as.matrix(un dataframe par exemple .. Mais attention à l 'homogénéité des données !)	as.data.frame (…)	as.data.table(…)

Créer un array : array(1:8, c(2, 2, 2))



Les structures de données
→ Liste

Une liste est un ensemble de données hétérogènes

Exemple

– MaListe<-list(c(1,2,3), c("robert","johnson"),matrix(rnorm(4),2,2))

```
[[1]]
[1] 1 2 3

[[2]]
[1] "robert"  "johnson"

[[3]]
           [,1]      [,2]
[1,]  0.2913862 0.3303319
[2,] -1.4849716 2.4071609
```

Un nom peut être affecté à chaque élément

– MaListe<- list(numero = c(1,2,3), noms = c("robert","johnson"), mat = matrix(rnorm(4),2,2))

```
$numero
[1] 1 2 3

$noms
[1] "robert"  "johnson"

$mat
          [,1]       [,2]
[1,] 1.4355363 -0.8544773
[2,] 0.6386084  1.9803265
```

Pour rechercher le 2è élément

– MaListe[[2]] → Un vecteur de character(plus une liste)

– Équivalent : MaListe$noms, MaListe[[‘’noms’’]] (notez les guillemets)

Exemple d’opération sur une liste

– lapply(maliste, length) : taille de chaque objet de la liste

– C(liste1,liste2) : concaténation de deux listes (comme pour un vecteur)

Les structures de données
→ Subset de vecteur

Utilisation de [ ] pour spécifier ce que vous voulez accéder

Entre les [ ] vous pouvez indiquer

```
x <- c(3, 6, -2, 9, NA, sin(-pi/6))
```

– Un vecteur logique (de la même taille que le vecteur) : seules les données TRUE seront conservées

```
x[x > 0] ; x[!is.na(x)] ;
x[!is.na(x) & x > 0] ;
x[x <= mean(x, na.rm = TRUE)]
```

– un vecteur numérique qui spécifie les valeurs à inclure ou à exclure (composante négative (indices de colonne)

```
x[-c(1, 5)] ;
x[-(1:5)] ;
x[2]
```

– un vecteur de chaînes de caractères, qui spécifie les noms des éléments de x à conserver.

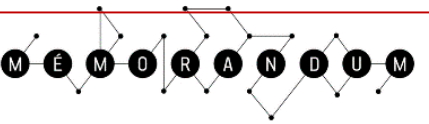
```
names(x) <- c("var1", "var2", "var3",
"var4", "var5", "var6")

x[c("var1", "var3")]
```

Exemple

– sort : renvoie le vecteur classé par ordre croissant ou décroissant (l'affiche trié mais ne change pas le vecteur initial en mémoire)
– Order: renvoie les indices d'ordre des éléments par ordre croissant ou décroissant,
– Which renvoie les indices de x vérifiant une condition ;
– Sample : échantillonne

```
x <- -5:5 ; > y <- sample(x)
sort(y)
order(y)
y[order(y)]
y[order(y, decreasing = TRUE)]
which(sample(x, 4) > 0)
```



Manipulation de données
→ Subset de matrice / data.frame / data.table

matrice	data.frame	data.table
Mamatrice[1:3,c(6,8)]	MonDataframe[1:3,c(6,8)]	MonDataTable[1:3,c(6,8),with=FALSE]
Mamatrice[,c(6,8]	MonDataframe[,c(6,8)]	MonDataTable[,c(6,8),with=FALSE]
Mamatrice[1:3,]	MonDataframe[1:3,]	MonDataTable[1:3]
« , » obligatoire !	« , » obligatoire !	« , » facultatif

MonObjet$*macolonne (→ un vecteur !)*

MonObjet [[''*macolonne*'']] *(→ un vecteur !)*

Le nom de colonne peut être une variable

Mamatrice[,c(''nomcol1'', ''nomcol1'')]	MonDataFrame[,c(''nomcol1'' , ''nomcol1'')]	MonDataTable[,c(''nomcol1'', ''nomcol1''),with=FALSE] MonDataTable[,.(nomcol1, nomcol1)]

Attention
- → MonDataFrame[,2] : un vecteur
- → MonDataFrame[,2,**drop=**FALSE] : dataframe avec une colonne
- → Globalement [[ ]] ou $ perd la structure initiale

Manipulation de données
→ Complément data.table

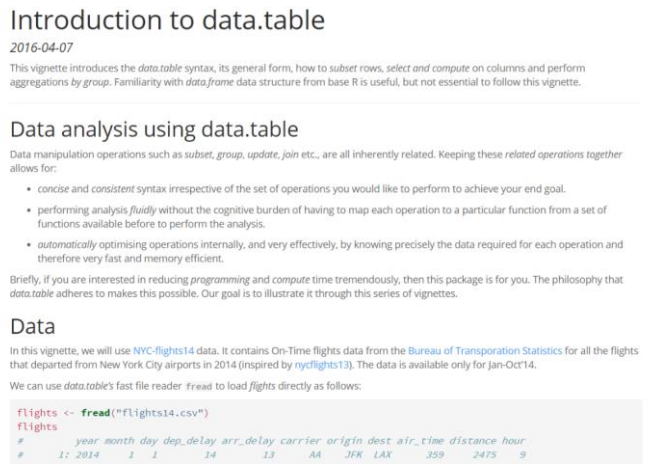
Le format data.table lié au package data.table présente de nombreux avantages

- Acquisition rapide avec fread() (au lieu de read.csv
- Laisse les données qualitatives au format character (pas factor)
- Présente une syntaxe de manipulation très riche, très proche de sql

Point d'attention

- Une assignation simple ne crée pas un double mais juste une vue du data.table initial
- Conséquence : une modification du nouvel objet impacte aussi le premier !
- Penser à utiliser la fonction copy() pour rendre les deux objets indépendants
  - monDT2<-monDT1 → objets liés
  - monDT2<-copy(monDT1) __> objets indépendants

A lire pour comprendre la syntaxe de manipulation : → https://rawgit.com/wiki/Rdatatable/data.table/vignettes/datatable-intro.html



Introduction to data.table

2016-04-07

This vignette introduces the *data.table* syntax, its general form, how to *subset* rows, *select and compute* on columns and perform aggregations *by group*. Familiarity with *data.frame* data structure from base R is useful, but not essential to follow this vignette.

Data analysis using data.table

Data manipulation operations such as *subset*, *group*, *update*, *join* etc., are all inherently related. Keeping these *related operations together* allows for:

- *concise* and *consistent* syntax irrespective of the set of operations you would like to perform to achieve your end goal.
- performing analysis *fluidly* without the cognitive burden of having to map each operation to a particular function from a set of functions available before to perform the analysis.
- *automatically* optimising operations internally, and very effectively, by knowing precisely the data required for each operation and therefore very fast and memory efficient.

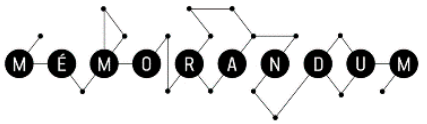
Briefly, if you are interested in reducing *programming* and *compute* time tremendously, then this package is for you. The philosophy that *data.table* adheres to makes this possible. Our goal is to illustrate it through this series of vignettes.

Data

In this vignette, we will use NYC-flights14 data. It contains On-Time flights data from the Bureau of Transporation Statistics for all the flights that departed from New York City airports in 2014 (inspired by nycflights13). The data is available only for Jan-Oct'14.

We can use *data.table*'s fast file reader `fread` to load *flights* directly as follows:

```
flights <- fread("flights14.csv")
flights
#        year month day dep_delay arr_delay carrier origin dest air_time distance hour
#     1: 2014     1   1        14        13      AA    JFK  LAX      359     2475    9
```



Manipulation de données
→ Dcast / melt

Lorsque le données d'une même variables sont étalées sur plusieurs colonnes il est parfois nécessaire de reformater les données. Le cas contraire est aussi fréquent

Exemple

names(airquality) <- tolower(names(airquality))

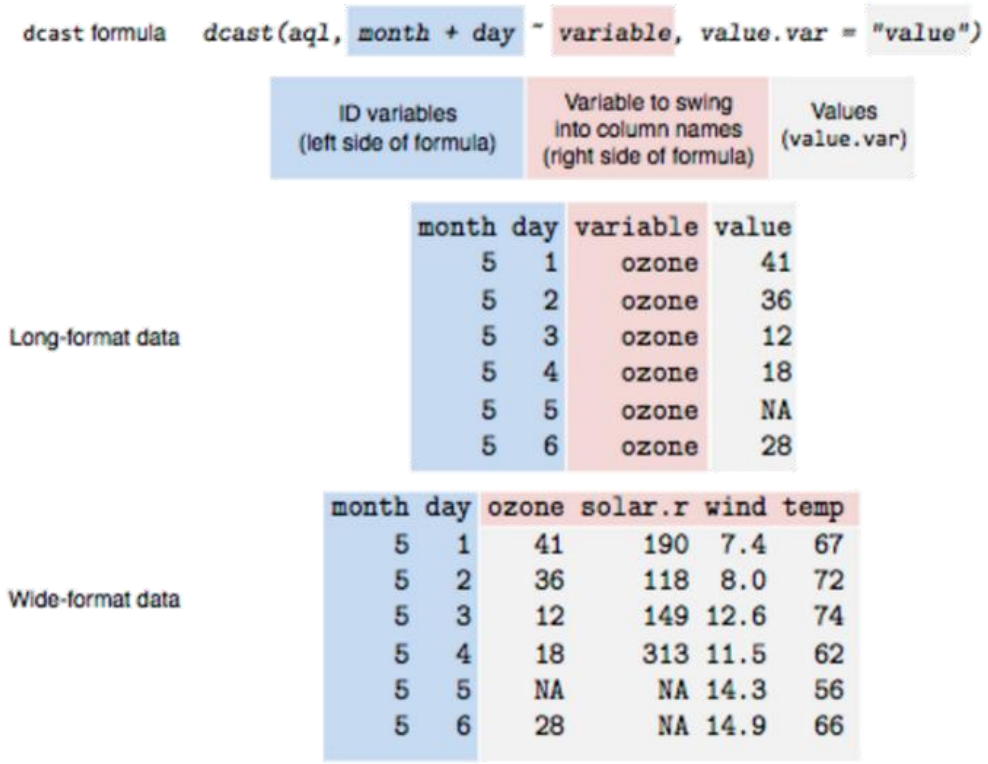
head(airquality); aql <- melt(airquality) ; aql <- melt(airquality, id.vars = c("month", "day")) ;

aql <- melt(airquality, id.vars = c("month", "day"), variable.name = "climate_variable", value.name = "climate_value") ;

aql <- melt(airquality, id.vars = c("month", "day"))

aqw <- dcast(aql, month + day ~ variable)

dcast(aql, month ~ variable, fun.aggregate = mean, na.rm = TRUE)



Chargement et sauvegarde de données

Charger

Une observation préalable du fichier est souvent utile avant d'importer

– Exemple notepad++

mes_donnees <- **read.table**("monfichier.txt",+
header = TRUE, sep = "\t“,
stringsAsFactors=FALSE)

Des formats plus rapide et spécialisés

– mes_donnees <- **read.csv**(“monfichier.csv**"**,+ header = TRUE, “, stringsAsFactors=FALSE)
– Mes_donnees <-**read_excel**(‘’monfichier.xlsx’’)
→ package « **readxl** »

Sauvegarder

Un tableau (matrice, dataframe, ..)

– Write.csv(montableau, ’’masauvegarde.csv’’)

Un objet quelconque

– Save(monObjet, ’’masauvegarde**.rda’’**)
– Le récupérer : load(’’masauvegarde.rda’’)

Exploration de fichier

Head(MonTableau, 5) : 5 premières lignes)

Attach(MonTableau) : rend disponible chaque colonne indépendamment du tableau

str(monTableau)

```
> str(donnees)

'data.frame':        245 obs. of  7 variables:
 $ Population               : Factor w/ 3 levels "CE","CO","TE": 1 1 1 1
 $ variete                  : Factor w/ 245 levels "0Mtp1004","0Mtp1005'
 $ nbre.pepin.baie.2008     : num  1 1 1.2 1.2 1.2 1.3 1.3 1.3 1.3 1.3 .
 $ poids.pulpe.baie..g..2008: num  0.89 1.14 1.26 0.66 0.83 0.54 0.67 0.
 $ volume.baie..cm3..2008   : num  7.7 8.82 10.2 NA NA 4.61 5.97 7.65 8.
 $ nbre.pepin.baie.2009     : num  NA NA NA NA NA NA NA NA NA NA ...
 $ poids.pulpe.baie..g..2009: num  NA NA NA NA NA NA NA NA NA NA ...
```

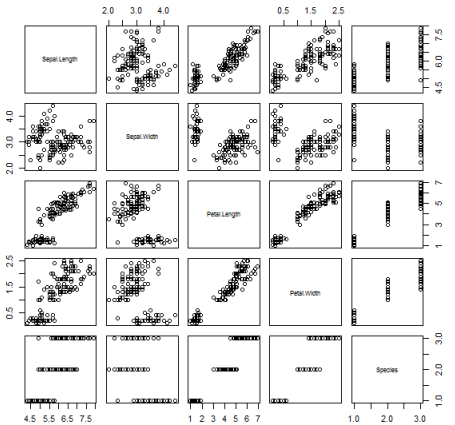
summary(monTableau)

```
> summary(donnees)

 Population     variete    nbre.pepin.baie.2008 poids.pulpe.baie..g..2008
 CE:84      0Mtp1004:  1   Min.   : 0.000       Min.   : 0.390
 CO:89      0Mtp1005:  1   1st Qu.: 1.400       1st Qu.: 0.820
 TE:72      0Mtp1033:  1   Median : 1.800       Median : 1.060
            0Mtp1068:  1   Mean   : 1.858       Mean   : 1.212
            0Mtp1072:  1   3rd Qu.: 2.400       3rd Qu.: 1.360
            0Mtp1073:  1   Max.   : 3.200       Max.   : 3.750
            (Other) :239   NA's   :27.000       NA's   :28.000
 volume.baie..cm3..2008 nbre.pepin.baie.2009 poids.pulpe.baie..g..2009
 Min.   : 3.44          Min.   : 1.000       Min.   : 0.560
 1st Qu.: 7.14          1st Qu.: 1.500       1st Qu.: 0.875
 Median : 8.91          Median : 2.000       Median : 1.230
 Mean   :10.47          Mean   : 2.082       Mean   : 1.513
 3rd Qu.:11.90          3rd Qu.: 2.600       3rd Qu.: 1.607
 Max.   :33.80          Max.   : 3.600       Max.   : 4.340
 NA's   :44.00          NA's   :196.000      NA's   :203.000
```

data(iris)

pairs(iris)



Tableaux croisés

age <- c(25, 35, 32, 27, 32, 40, 26, 25, 26, 28, 30, NA, 36, + 30, 30)
grd <- c("thd", "CR", "MdC", "thd", "thd", "MdC", "MdC", "thd","thd", "MdC", "CR", "MdC", "CR", "thd", "thd")

table(grd)

```
> tapply(age, grd, mean, na.rm = TRUE)
      CR       MdC       thd
33.66667 31.50000 27.85714
>
> by(age, grd, mean, na.rm = TRUE)
grd: CR
[1] 33.66667
------------------------------------------------
grd: MdC
[1] 31.5
------------------------------------------------
grd: thd
[1] 27.85714
>
```

Boucles et structures conditionnelles

if (condition) { action à réaliser si TRUE} **else** { action à réaliser si FALSE}

– Condition : exemple a<b ; Else est facultatif

– Alternative : ifelse(condition, action à réaliser si TRUE, action à réaliser si FALSE}

for (x in vecteur) { action à réaliser }

– x nom d'indice qui parcourra successivement toutes les valeurs de vecteur

While (condition) { action à réaliser}

– Veiller à ce que la valeur de la condition évolue au sein de l'action à réaliser

Repeat {
  action à réaliser
  if (condition) break
}

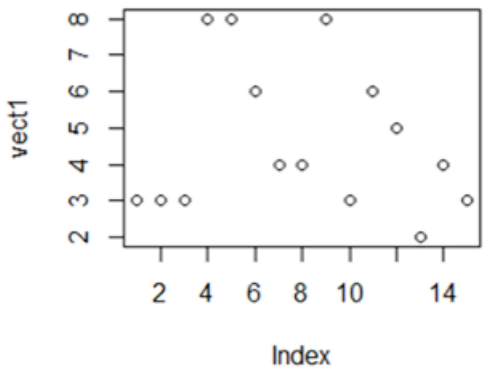
Préférer le plus souvent possible les opérateurs matriciels !
Plus rapide)

Définition de fonctions

```
Mafonction(param1,param2, param3=valeur par défaut) {
        action à réaliser
        return(valeur à retourner)     # → si return est omis, la dernière valeur calculée est renvoyée
                                       # un sul objet peut être retourner. Pour en renvoyer plusieurs, créer une liste
}
Xx<- Mafonction(3,4) # les arguments peuvent être passés dans le désordre s'ils sont nommés
                     # param3 non passé prendra la valeur par défaut)
```

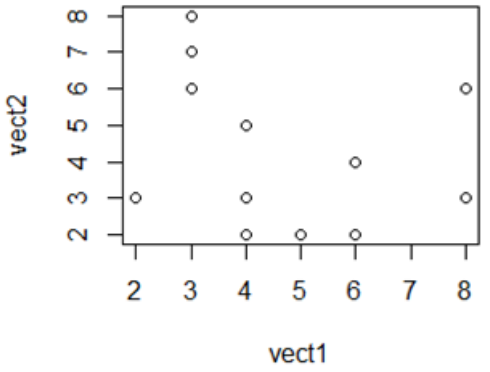
Dessiner (avant de découvrir ggplot2)

plot(vect)

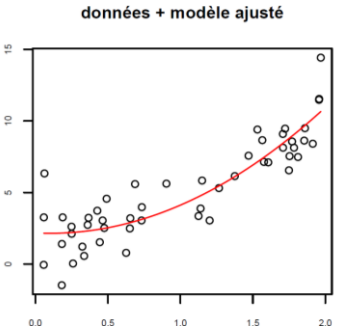


Paramètres plot
type="p" ; spécifie le type de tracé : "p" pour points, "l" pour lignes, "b" pour points liés par des lignes, "o" pour lignes superposées aux points. . .
xlim= ; ylim=, spécifie les limites de axes x et y
xlab= ; ylab=, annotation des axes x et y
main= ; titre du graphe en cours
sub= ; sous-titre du graphe en cours
add=FALSE ; si TRUE superpose le graphe au précédent

plot(vect,vect2)



curve(2 + 3* x + 5* x^2, add = TRUE, col = "red")



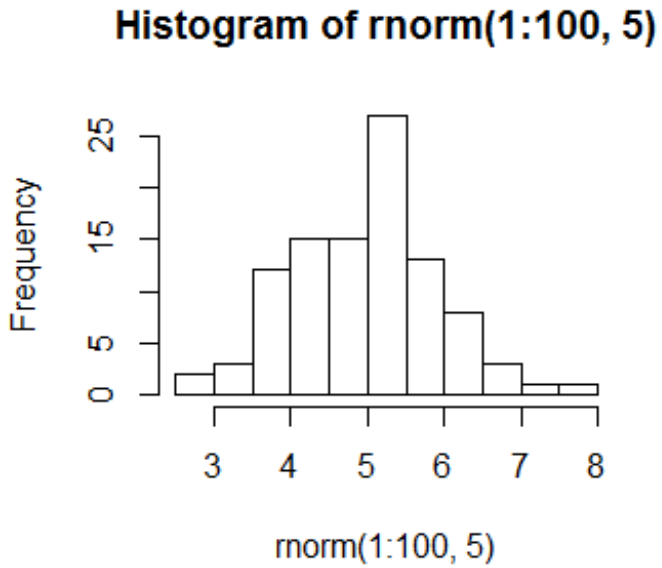
Mais aussi pour ajouter des droites

- abline(a,b) a : ordonnée origine, b : pente
- Abline(v=..) abline(h=.) : droites horizontales et verticales

Dessiner (avant de découvrir ggplot2)

Histogramme

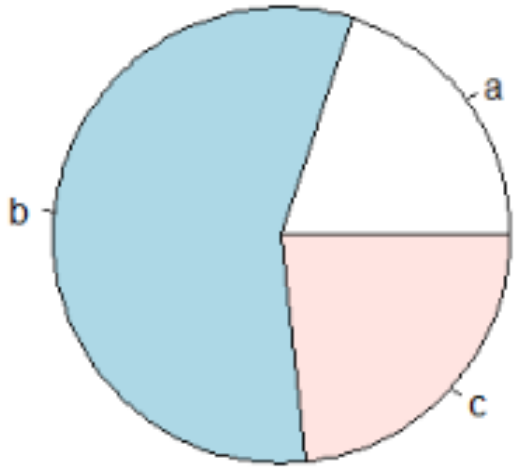
– hist(rnorm(1:100,5))



Tarte …

– x<-sample(c("a","b","c"),30,replace=TRUE)

– Pie(table(x))

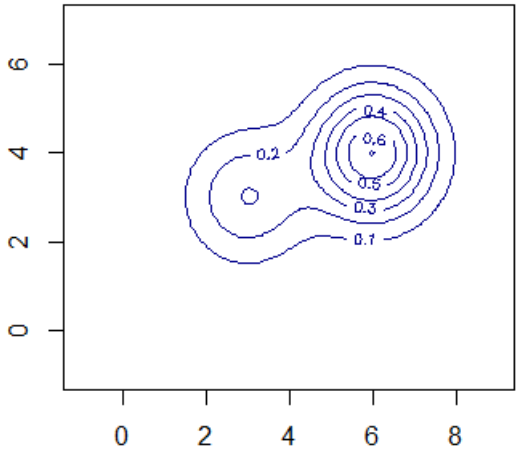


Dessiner (avant de découvrir ggplot2)

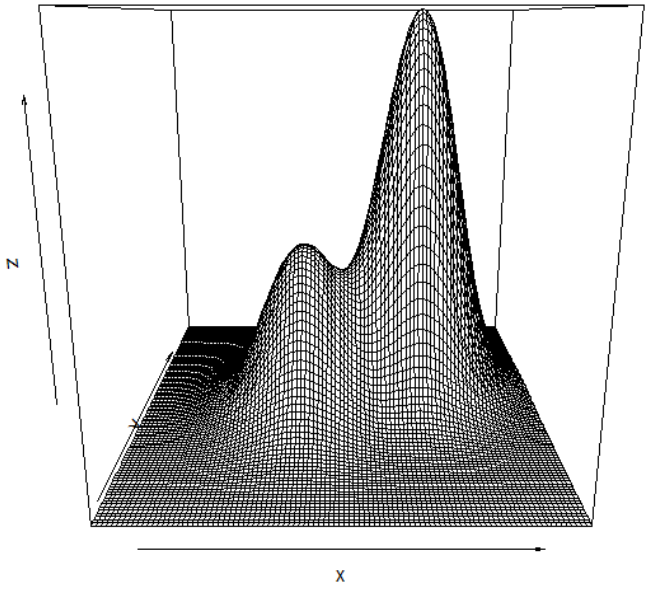
```
x <- seq(-1, 9, length = 100)
y <- seq(-1, 7, length = 100)
z <- outer(x, y, function(x, y) 0.3 * exp(-0.5 * ((x - 3)^2 + (y - 3)^2)) + 0.7 * exp(-0.5 * ((x - 6)^2 + (y - 4)^2)))
```

– Outer : calcule la fonction pour toutes les valeurs de x et y

contour(x, y, z, col = "blue4")



persp(x, y, z)



A consulter : http://zoonek2.free.fr/UNIX/48_R_2004/04.html

Un guide de manipulation de données

https://www.slideshare.net/DerekKane/data-science-iii-eda-model-selection?next_slideshow=1