Introduction au logiciel R et applications

Marc J. Mazerolle
Centre d'étude de la forêt-UQAT

cef
Centre d'étude de la forêt

Université du Québec en Abitibi-Témiscamingue

Qu'est-ce que le projet R?

R est un langage de programmation et statistique

- développé au début des années 1990 par 2 statisticiens et programmeurs de Nouvelle-Zélande (R. Ihaka et R. Gentleman)
- dialecte du langage S (qui lui n'est pas gratuit, intégré dans S-PLUS)
- R provient du besoin de flexibilité à effectuer des analyses et des graphiques sans avoir à débourser des sommes astronomiques (i.e., rendre accessible à tous)

Qu'est-ce que le projet R?

Initiative de statisticiens et programmeurs qui rendent disponible gratuitement le logiciel et le code source.

Ce que ça implique:

- le logiciel source est ouvert à tous et peut être modifié selon les besoins de chacun (pour utilisateurs avertis...)
- de nouvelles fonctions peuvent être créées par les utilisateurs et rendues disponibles à tous
- les mises à niveau du logiciel sont fréquentes et gratuites (nouvelle version améliorée ~ tous les 6 mois)
- on ne verse pas des coûts de licence annuelle à une compagnie qui impose ses standards (SAS...)
- une grande communauté d'utilisateurs à travers le monde

Utilisations de R

Langage de programmation:
découle du langage S et C

Applications mathématiques (une grosse calculatrice):
calcul matriciel, différentiel, intégral

Applications statistiques et optimisation:
éventail varié de possibilités

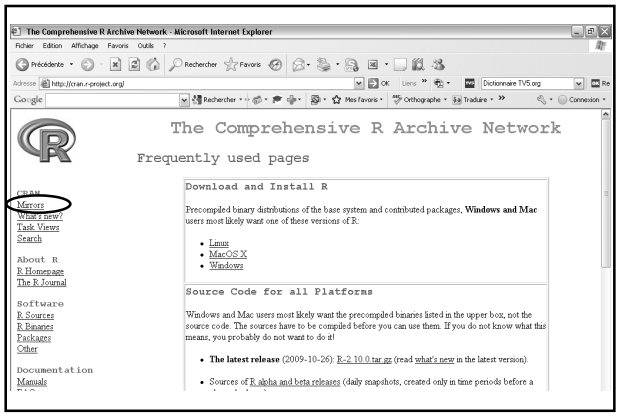
Excellente capacité graphique:
vaste éventail de graphiques

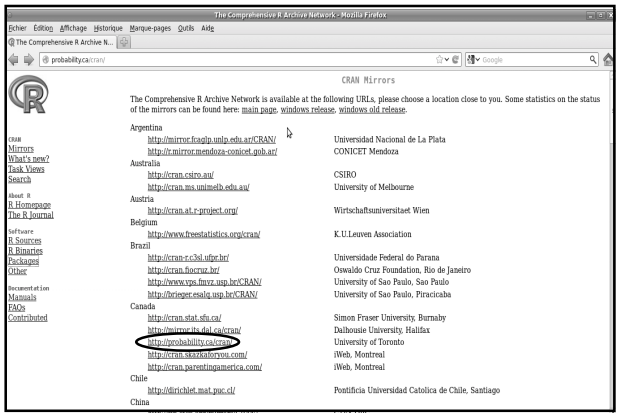
Flexibilité:
facile de créer des nouvelles fonctions,
utilisation de boucles

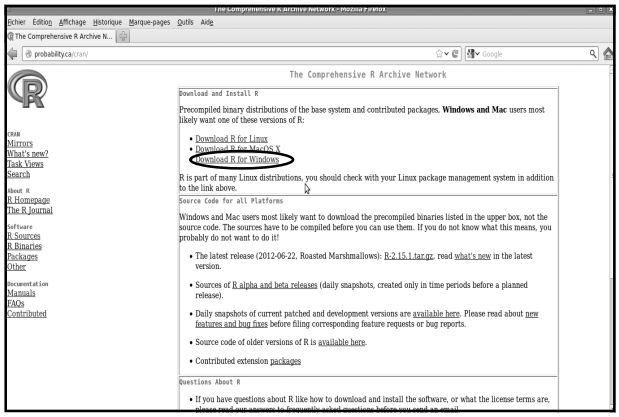
Où se procurer R?

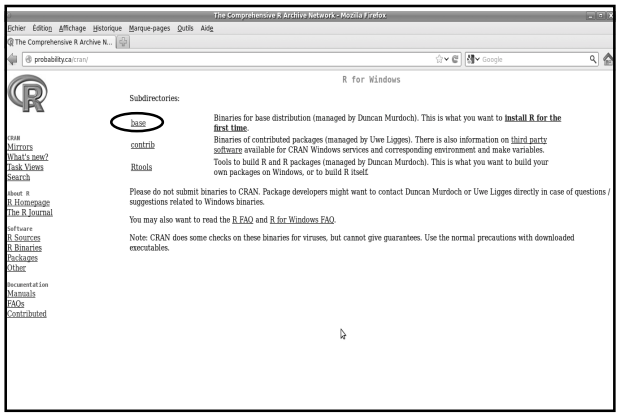
http://cran.r-project.org/

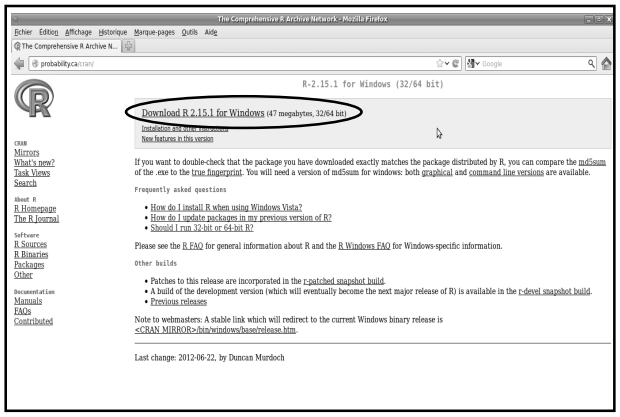
et ses nombreux sites miroirs à travers le monde
(préférable de choisir le site le plus près)




The Comprehensive R Archive Network - Mozilla Firefox
CRAN Mirrors
The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.
Argentina
http://mirror.fcaglp.unlp.edu.ar/CRAN/ Universidad Nacional de La Plata
http://r.mirror.mendoza-conicet.gob.ar/ CONICET Mendoza
Australia
http://cran.csiro.au/ CSIRO
http://cran.ms.unimelb.edu.au/ University of Melbourne
Austria
http://cran.at.r-project.org/ Wirtschaftsuniversitaet Wien
Belgium
http://www.freestatistics.org/cran/ K.U.Leuven Association
Brazil
http://cran-r.c3sl.ufpr.br/ Universidade Federal do Parana
http://cran.fiocruz.br/ Oswaldo Cruz Foundation, Rio de Janeiro
http://www.vps.fmvz.usp.br/CRAN/ University of Sao Paulo, Sao Paulo
http://brieger.esalq.usp.br/CRAN/ University of Sao Paulo, Piracicaba
Canada
http://cran.stat.sfu.ca/ Simon Fraser University, Burnaby
http://mirror.its.dal.ca/cran/ Dalhousie University, Halifax
http://probability.ca/cran/ University of Toronto
http://cran.skazkaforyou.com/ iWeb, Montreal
http://cran.parentingamerica.com/ iWeb, Montreal
Chile
http://dirichlet.mat.puc.cl/ Pontificia Universidad Catolica de Chile, Santiago
China
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed


The Comprehensive R Archive Network - Mozilla Firefox
The Comprehensive R Archive Network
Download and Install R
Precompiled binary distributions of the base system and contributed packages, Windows and Mac users most likely want one of these versions of R:
Download R for Linux
Download R for MacOS X
Download R for Windows
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.
Source Code for all Platforms
Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!
The latest release (2012-06-22, Roasted Marshmallows): R-2.15.1.tar.gz, read what's new in the latest version.
Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
Source code of older versions of R is available here.
Contributed extension packages
Questions About R
If you have questions about R like how to download and install the software, or what the license terms are,
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed


The Comprehensive R Archive Network - Mozilla Firefox
R for Windows
Subdirectories:
base
Binaries for base distribution (managed by Duncan Murdoch). This is what you want to install R for the first time.
contrib
Binaries of contributed packages (managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
Rtools
Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.
Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.
You may also want to read the R FAQ and R for Windows FAQ.
Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed

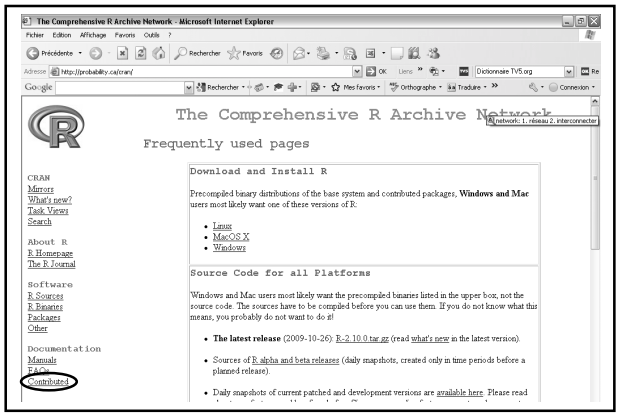


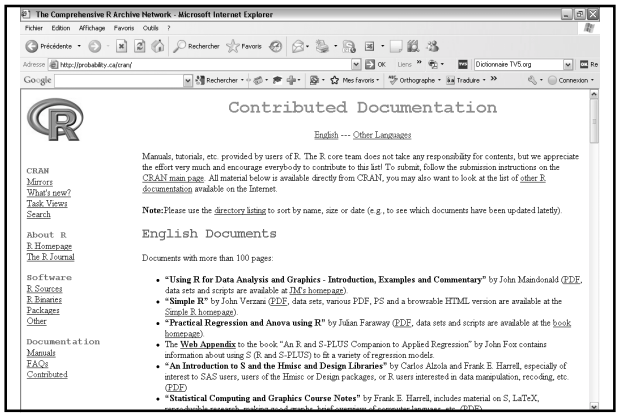
Installation

Exécuter fichier d'installation:
`R-2.15.2-win.exe` (avec options par défaut)

La distribution inclut des manuels d'utilisateurs
(manuels un peu arides...)

Aller voir dans Documentation et Contributed sur la page de R et télécharger autres documents pratiques manuels

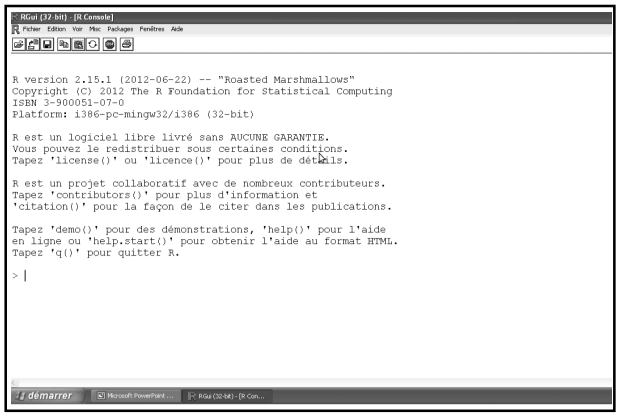




Pour débuter

Interface graphique minimale pour R dans Windows où on fait clic, clic, clic...

(même chose pour Mac et aucune pour Linux)



Pour débuter

Pas d'interface Windows pour R

Il faut communiquer par le biais de commandes

Pour faciliter la tâche, on peut préparer le code dans l'éditeur de notre choix (`Word`, `Bloc-notes`, `Emacs`…)

Dépend de la plateforme (OS)

Éditeurs

Plusieurs éditeurs intelligents disponibles pour R (`Emacs, Tinn-R, Jedit`)

Un éditeur intelligent:

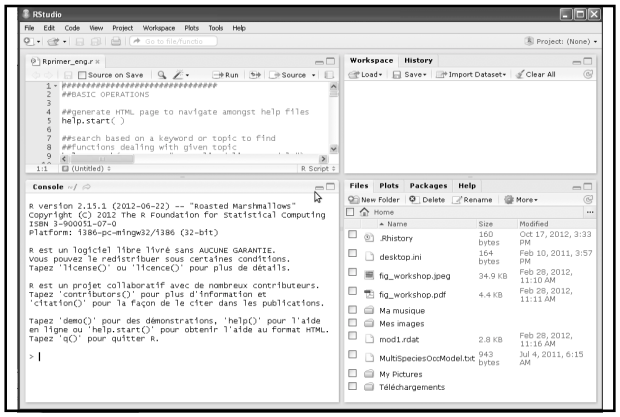
- reconnaît le langage R
- utilise des couleurs pour distinguer différents éléments
- apparie les parenthèses
- envoie les commandes directement à R
- peut aussi numéroter les lignes, compléter les commandes

Éditeurs

Éditeur suggéré:

`RStudio`

- disponible au `www.rstudio.org`
- fonctionne sur plusieurs plate-formes (Windows, Mac, Linux)
- plusieurs options intéressantes
- organise le tout en 4 panneaux (éditeur, console R, environnement de travail, graphiques/aide)

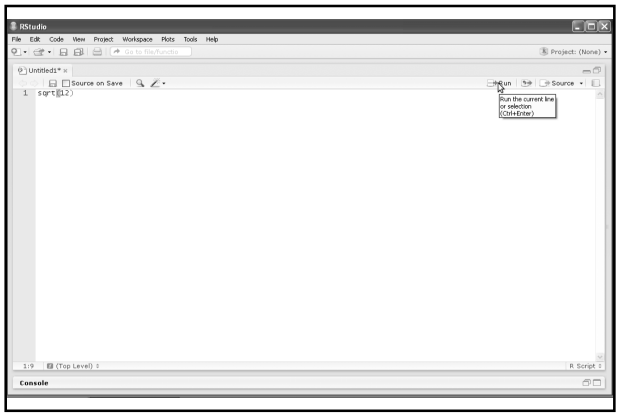


RStudio comme éditeur

Panneaux peuvent être maximisés en cliquant les icônes pour maximiser et minimiser

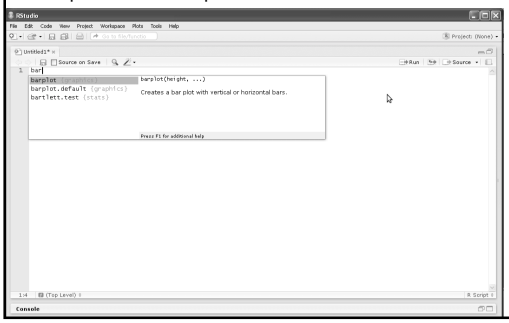
On modifie les préférences à l'aide du menu `Tools -> Options..`
(numérotation des lignes, indentation, insertion de paires de parenthèses, Organisation des panneaux)

On écrit le code dans l'éditeur (dans le panneau en haut à gauche) et on l'envoie à R via l'icône « Run »



RStudio comme éditeur

Complétion automatique des commandes avec la touche `tab` du clavier

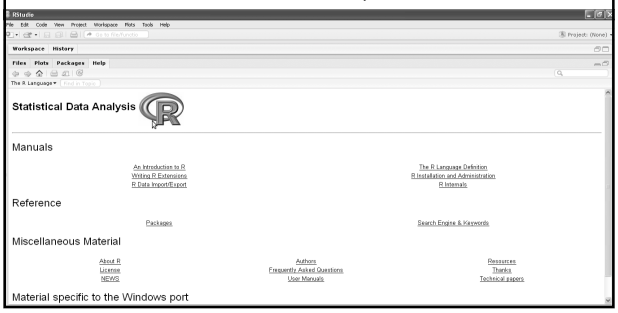


on débute le nom d'une fonction `bar`

on obtient des suggestions en appuyant sur `tab`

RStudio comme éditeur

On accède au menu d'aide de R dans le panneaux d'aide



Conseil pratique

Utilisez un éditeur afin d'écrire et de sauvegarder vos programmes

Sauvegarder le fichier avec l'extension `.R` ou `.r`

dès qu'on ouvre le fichier `.R` ou `.r` avec un éditeur intelligent, il sait que l'on travaille en R

Le symbole # est réservé aux commentaires
(dès qu'il est rencontré, R saute à la prochaine ligne)

Utilisez les commentaires abondamment!!!

Syntaxe R

Toutes les commandes R suivent la même stratégie:

un nom suivi de parenthèses

e.g., `mean( )` fonction pour calculer moyenne arithmétique

options specifiées entre parenthèses

e.g., `mean(DBH)` calcule moyenne arithmétique de la variable DBH

L'aide dans R

`help.start( )`

Génère une page `html` à partir de la quelle on peut naviguer vers différents thèmes via des hyperliens

`help.search( )`

Recherche de sujet spécifique dans les fichiers d'aide de R

l'argument `apropos` permet de spécifier le sujet de recherche

```
help.search(apropos = "generalized linear models")
```

retourne l'ensemble des fonctions qui traite du sujet parmi les fonctions de R

L'aide dans R

? suivi du nom d'une fonction

Pour obtenir de l'aide sur les fonctions (syntaxe, arguments, usage)

Détails généralement suivis d'exemples illustrant l'application des fonctions (très pratique)

Aide en format `html` – si non branché à internet, on peut voir l'aide en texte `options(help_type = "text")`

`RSiteSearch(" ")`

Pour chercher les archives de R sur le web

Ouvre une fenêtre du « browser » pour accéder au site

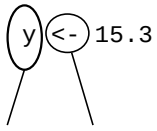
Langage orienté-objet (object oriented)

Un objet peut prendre la forme d'une variable (vecteur), d'un jeu de données, d'une matrice, d'une fonction, d'une liste, ou d'un résultat.

On assigne simplement un nom à un objet et il demeure en mémoire durant toute la session de travail.

Vecteurs et Opérations de base

Créer une variable (vecteur)



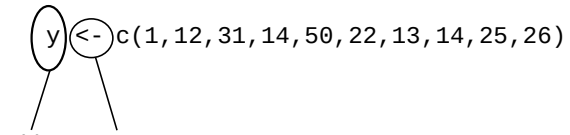
Objet — Opérateur indiquant d'associer la valeur 15.3 à l'objet « y ».

le nom de l'objet doit commencer par une lettre et peut être combinaison de chiffres ou lettres (bonne pratique de les garder courts et informatifs)

le nom peut contenir « _ » ou « . » mais pas d'espace

Ex. pH_sol, pH.sol Pas pH sol

Créer une variable (vecteur)



Objet

Opérateur indiquant d'associer les valeurs à l'objet « y ».

`c( )` indique de combiner toutes les valeurs

Créer une variable (vecteur)

Attention aux majuscules et minuscules

il faut utiliser le nom exact de l'objet

```
> Y
Erreur : objet "Y" non trouvé

> y
 [1]  1 12 31 14 50 22 13 14 25 26
```

Sélectionner valeurs

`y[2]`

sélectionne 2ième valeur de y

`y[1:5]`

sélectionne les 5ière valeurs de y

Création de vecteur

Pour répéter une valeur

`rep( )`

```
> new_y<-rep(x=5,times=20)
> new_y
 [1] 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
```

fonctionne avec valeurs numériques ou lettres

```
> new_z<-rep(x=c("new", "old"), times=3)
> new_z
[1] "new" "old" "new" "old" "new" "old"
```

Créer un vecteur

Pour créer une séquence (série) avec un pas spécifique

`seq( )`

```
> y2<-seq(from=1, to=10, by=1)
> y2
 [1]  1  2  3  4  5  6  7  8  9 10
```

on peut aussi générer une séquence qui décline

```
y3<-seq(from=10, to=1, by=-2)
> y3
[1] 10  8  6  4  2
```

Vérifier les caractéristiques du vecteur

On peut vérifier la nature d'un objet avec `class( )` ou des tests logiques

```
> class(y3)
[1] "numeric"
> is.numeric(y3)
[1] TRUE
```

```
> new_z
[1] "new" "old" "new" "old" "new" "old"
> class(new_z)
[1] "character"
> is.numeric(new_z)
[1] FALSE
> is.character(new_z)
[1] TRUE
```

Déterminer la longueur d'un vecteur

`length( )` longueur de l'objet

```
> y
 [1]  1 12 31 14 50 22 13 14 25 26

> length(y)
[1] 10
```

Opérateurs mathématiques

Opérateurs mathématiques

Opérateurs de base:

```
+     -     *     /
```

```
> 30+10-2*3
34
```

respecte les convention de priorité de * et / avant + ou -

```
> y2
 [1]  1  2  3  4  5  6  7  8  9 10
> y
 [1]  1 12 31 14 50 22 13 14 25 26
> y+y2
 [1]  2 14 34 18 55 28 20 22 34 36

> 2-y2
 [1]  1  0 -1 -2 -3 -4 -5 -6 -7 -8
```

la valeur 2 est recyclée pour effectuer le calcul

Opérateurs mathématiques

Autres fonctions de base:

`sum( )` somme de tous les éléments

```
> sum(y)
[1] 208
```

`prod( )` produit de tous les éléments

```
> prod(y)
[1] 6.77717e+11
```

`^( )` mettre à une puissance donnée

```
> 2^3
[1] 8
```

`round( )` arrondir à *x* chiffres décimaux

```
> round(123.3453, digits=2)
[1] 123.35
```

```
> round(123.3453, digits=0)
[1] 123
```

Opérateurs mathématiques

Autres fonctions de base:

`sqrt( )` racine carrée

`exp( )` fonction exponentielle de x = (e^x)

`log( )` log à base *e*

`log2( )` log à base 2

`log10( )` log à base 10

`abs( )` valeur absolue

Opérateurs mathématiques

Autres fonctions de base:

`min( )` retourne la valeur minimale

`max( )` retourne la valeur maximale

`range( )` retourne la valeur minimale et maximale

```
> range(y)
[1]  1 50
```

Fonctions trigonométriques:

`sin( ), cos( ), asin( ), tan( ), etc...`

Opérateurs mathématiques

Calcul différentiel symbolique `D( )`

```
> D(expression(x^2),"x")
2 * x
```

Dérivée de x^2 par rapport à x

```
> D(expression(x+(x^2)+(2*z^3)), "z")
2 * (3 * z^2)
```

Dérivée par rapport à z

Calcul intégral `integrate( )`

```
> integrate(dnorm, -1.96, 1.96)
0.9500042 with absolute error < 1.0e-11
```

L'aire sous la courbe d'une distribution normale centrée-réduite entre -1.96 et 1.96.

Les matrices et les calculs matriciels

Un autre type d'objet: la matrice

Créer une matrice avec la fonction `matrix( )`

```
> mat<-matrix(data=c(1:9), nrow=3, ncol=3)
> mat
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Par défaut, les éléments sont entrés par colonnes.

```
> mat<-matrix(data=c(1:9), nrow=3, ncol=3, byrow=TRUE)
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

`byrow=TRUE` permet de les entrer par rangées

Vérifier les caractéristiques de la matrice

On peut vérifier la nature d'un objet avec `class( )` ou des tests logiques

```
> class(mat)
[1] "matrix"
> is.numeric(mat)
[1] TRUE
> is.vector(mat)
[1] FALSE
```

On détermine le nombre d'éléments dans la matrice avec `length( )`

```
> length(mat)
[1] 9
```

On détermine les dimensions de la matrice avec `dim( )`

```
> dim(mat)
[1] 3 3
```

Sélectionner éléments d'une matrice

Avec des vecteurs on utilise `[ ]`, mais pour les matrices il faut spécifier les deux dimensions `[ , ]` (rangée, colonne).

```
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9

> mat[1,2]          > mat[3,1]
[1] 2               [1] 7
```

Sélectionner éléments d'une matrice

Avec des vecteurs on utilise `[ ]`, mais pour les matrices il faut spécifier les deux dimensions `[ , ]` (rangée, colonne).

```
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

`> mat[,1]` sélectionne la colonne 1

```
[1] 1 4 7
```

`> mat[2,]` sélectionne la rangée 2

```
[1] 4 5 6
```

Sélectionner éléments d'une matrice

Avec des vecteurs on utilise `[ ]`, mais pour les matrices il faut spécifier les deux dimensions `[ , ]` (rangée, colonne).

```
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

`> mat[1:2,1]` sélectionne les éléments de la 1ière colonne pour rangées 1 et 2

```
[1] 1 4
```

`> mat[c(1,3),3]` éléments de la 3ième colonne pour les 1ière et 3ième rangées

```
[1] 3 9
```

Calculs matriciels

Obtenir diagonale de la matrice `diag( )`

```
> mat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> diag(mat)
[1] 1 5 9
```

Transposer matrice `t( )`

```
> t(mat)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Calculs matriciels

Autres fonctions

`colSums( )` calcule somme de chaque colonne

```
> colSums(mat)
[1] 12 15 18
```

`colMeans( )` calcule moyenne de chaque colonne

`rowSums( )` calcule somme de chaque rangée

`rowMeans( )` calcule moyenne de chaque rangées

`solve( )` pour obtenir l'inverse de la matrice (si elle peut être inversée)

Calculs matriciels

Multiplication impliquant des matrices `%*%`

```
> vecteur<-c(2,4,5)
> mat%*%vecteur
     [,1]
[1,]   25
[2,]   58
[3,]   91
```

Les jeux de données

Un autre type d'objet: les jeux de données

Créer un jeu de donnée avec la fonction `data.frame( )`

```
> Temps<-c(1.2, 3.4, 2.1, 5.5)
> Masse<-c(2.5, 4.2, 5.6, 3.4)
> Sexe<-c("mâle", "femelle", "mâle", "femelle")

> jeu<-data.frame(Temps, Masse, Sexe)
> jeu
  Temps Masse    Sexe
1   1.2   2.5    mâle
2   3.4   4.2 femelle
3   2.1   5.6    mâle
4   5.5   3.4 femelle
```

Vérifier les caractéristiques du jeu de données

```
> class(jeu)
[1] "data.frame"
```

```
> length(jeu)
[1] 3
```

avec un jeu de données, `length( )` donne le nombre de variables du jeu de données

```
> dim(jeu)
[1] 4 3
```

avec un jeu de données, `dim( )` donne le nombre de variables du jeu de données et le nombre d'observations pour chacune

Vérifier les caractéristiques du jeu de données

Pour la structure du jeu de données

```
> str(jeu)
'data.frame':   4 obs. of  3 variables:
 $ Temps: num  1.2 3.4 2.1 5.5
 $ Masse: num  2.5 4.2 5.6 3.4
 $ Sexe : Factor w/ 2 levels "femelle","mâle": 2 1 2 1
```

Avec un jeu de données, `summary( )` donne des stats descriptives de base pour chaque variable

```
> summary(jeu)
 Temps          Masse            Sexe
 Min.   :1.200   Min.   :2.500   femelle:2
 1st Qu.:1.875   1st Qu.:3.175   mâle   :2
 Median :2.750   Median :3.800
 Mean   :3.050   Mean   :3.925
 3rd Qu.:3.925   3rd Qu.:4.550
 Max.   :5.500   Max.   :5.600
```

Vérifier les caractéristiques du jeu de données

Pour identifier les variables du jeu de données

```
> names(jeu)
[1] "Temps" "Masse" "Sexe"
```

Pour voir les premières observations de chaque variable

```
> head(jeu)
```

Pour voir les dernières observations de chaque variable

```
> tail(jeu)
```

Les jeux de données

On peut aussi créer un jeu de données en une seule étape

```
> data_set<-data.frame(Vitesse=c(25, 40, 70, 100),
Type=c("auto", "camion", "auto", "camion"))
> data_set
  Vitesse   Type
1      25   auto
2      40 camion
3      70   auto
4     100 camion
> Vitesse
Erreur : objet "Vitesse" non trouvé
```

Comment accéder aux variables???

Accéder à une ou plusieurs variables

On a quelques options afin de sélectionner des variables

Option 1: utiliser l'opérateur $

```
> data_set$Type
[1] auto   camion auto   camion
Levels: auto camion
```

Option 2: utiliser l'opérateur `[ , ]`

```
> data_set[,2]
[1] auto   camion auto   camion
Levels: auto camion
```

Type correspond à la colonne 2

Importation de fichiers

Très peu pratique d'entrer les données à la main.

Heureusement, on peut facilement importer les jeux de données (en format `.txt`) dans R.

Structure des données

Les données récoltées lors d'une expérience:

Témoin	Engrais A	Engrais B	Engrais C
6.1	6.3	7.1	8.1
5.9	6.2	8.2	8.5
5.8	5.8	7.3	7.6
5.4	6.3	6.9	7.8

Structure correcte pour lecture et importation de fichier:

Réponse	Traitement
6.1	Témoin
5.9	Témoin
5.8	Témoin
5.4	Témoin
6.3	Engrais_A
6.2	Engrais_A
...	...

Une variable = 1 colonne

Préparation à l'importation dans R

Lorsque le fichier est dans le bon format (1 colonne = 1 variable), on peut sauvegarder en `.txt` (séparateur tabulations; tab delimited) à partir de `MS Excel` ou `Calc de OpenOffice`.

Points très importants:

La première ligne de chaque colonne est réservée au nom de la colonne

Aucun espace entre les noms des colonnes ou dans les données

Ex.
hauteur moy INCORRECT
hauteur_moy ou hauteur.moy CORRECT

Avec des données manquantes, il faut utiliser `NA` pour indiquer "Not Available" (aucune cellule ne peut être vide)

Importation dans R

Pour importer:

la fonction `read.table( )`

```
worms<-read.table("c:\\chemin_sur_votre_ordi\\worms.txt",  header=TRUE)
```

(`header = TRUE` indique que le nom des variables se trouve à la 1ière ligne)

Points importants:

Spécifier le bon chemin (sensibles aux majuscules et minuscules)

Utiliser soit \\ ou / dans le chemin, mais pas \ comme dans Windows

Mettre le chemin et le nom du fichier entre " "

Importation dans R

Trois stratégies équivalentes:

1) on spécifie le chemin complet sur l'ordi où se trouve le fichier

```
worms<-read.table(file="c:\\chemin_complet\\worms.txt", header=TRUE)
```

2) on utilise la fonction `file.choose()` et on sélectionne le fichier nous-même à l'aide d'une fenêtre

```
worms<-read.table(file=file.choose(), header=TRUE)
```

3) on assigne un répertoire de travail au tout début de la session où se trouvent les fichiers d'intérêt et on va les chercher par la suite

```
setwd(dir="c:\\chemin_complet\\")
worms<-read.table(file= "worms.txt", header=TRUE)
```

Manipulations de base dans R

```
> worms[1:10,]
          Field.Name Area Slope Vegetation Soil.pH  Damp Worm.density
1        Nashs.Field  3.6    11  Grassland     4.1 FALSE            4
2     Silwood.Bottom  5.1     2     Arable     5.2 FALSE            7
3      Nursery.Field  2.8     3  Grassland     4.3 FALSE            2
4        Rush.Meadow  2.4     5     Meadow     4.9  TRUE            5
5    Gunness.Thicket  3.8     0      Scrub     4.2 FALSE            6
6           Oak.Mead  3.1     2  Grassland     3.9 FALSE            2
7       Church.Field  3.5     3  Grassland     4.2 FALSE            3
8            Ashurst  2.1     0     Arable     4.8 FALSE            4
9        The.Orchard  1.9     0    Orchard     5.7 FALSE            9
10     Rookery.Slope  1.5     4  Grassland     5.0  TRUE            7
```

Manipulations de base dans R

Pour vérifier le noms des variables du jeu de données

Utiliser fonction `names ( )`

```
> names(worms)
[1] "Field.Name"    "Area"          "Slope"         "Vegetation"
"Soil.pH"       "Damp"          "Worm.density"
```

Manipulations de base dans R

Pour obtenir un bref résumé des variables (min, max, quartiles)

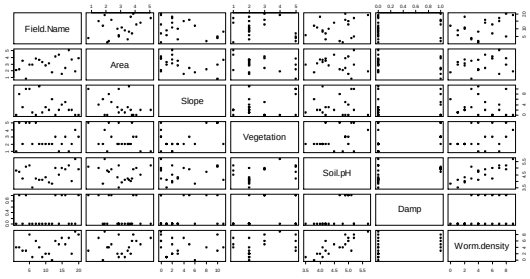
Utiliser fonction `summary ( )`

```
> summary(worms)
       Field.Name      Area           Slope           Vegetation
 Ashurst     : 1   Min.   :0.800   Min.   : 0.00   Arable   :3
 Cheapside   : 1   1st Qu.:2.175   1st Qu.: 0.75   Grassland:9
 Church.Field: 1   Median :3.000   Median : 2.00   Meadow   :3
 Farm.Wood   : 1   Mean   :2.990   Mean   : 3.50   Orchard  :1
 Garden.Wood : 1   3rd Qu.:3.725   3rd Qu.: 5.25   Scrub    :4
 Gravel.Pit  : 1   Max.   :5.100   Max.   :11.00
 (Other)     :14
    Soil.pH         Damp          Worm.density
 Min.   :3.500   Mode :logical   Min.   :0.00
 1st Qu.:4.100   FALSE:14        1st Qu.:2.00
 Median :4.600   TRUE :6         Median :4.00
 Mean   :4.555                   Mean   :4.35
 3rd Qu.:5.000                   3rd Qu.:6.25
 Max.   :5.700                   Max.   :9.00
```

Manipulations de base dans R

Pour obtenir une matrice de graphique de toutes les variables

Utiliser fonction `plot ( )`



Manipulations de base dans R

Pour sélectionner une partie du jeu de données

Utiliser indices et `[rangée, colonnes]`

- `> worms[1,]` sélectionne la 1ière ligne du jeu de données
- `> worms[1:15,]` sélectionne les 15ières lignes du jeu de données
- `> worms[,3]` sélectionne la 3ième colonne du jeu de données
- `> worms[,3:5]` sélectionne les colonnes 3-5 du jeu de données
- `> worms[1,3:5]` sélectionne la 1ière ligne des colonnes 3-5

Manipulations de base dans R

Les tests logiques

`>, <, >=, <=, ==, !=`

```
> worms$Slope
 [1] 11  2  3  5  0  2  3  0  0  4 10  1  2  6  0  0  8  2  1 10
```

`> worms$Slope==0` (strictement égal à 0)

```
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE
FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE
```

```
> worms$Slope>0
[1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE
TRUE  TRUE TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

`> worms$Slope!=0` (différent de 0)

```
 [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE
TRUE  TRUE TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

Manipulations de base dans R

Les tests logiques

```
> worms$Vegetation
 [1] Grassland Arable    Grassland Meadow    Scrub     Grassland
Grassland     Arable    Orchard   Grassland Scrub     Grassland
Grassland Grassland Meadow    Meadow    Scrub     Arable
Grassland Scrub
Levels: Arable Grassland Meadow Orchard Scrub
```

Avec des facteurs (caractères), il faut mettre valeur entre " "

```
> worms$Vegetation=="Grassland"
 [1]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
FALSE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
```

Manipulations de base dans R

Les tests logiques

union de deux conditions (ou inclusif): |

```
> worms$Vegetation
 [1] Grassland Arable    Grassland Meadow    Scrub     Grassland Grassland
       Arable    Orchard   Grassland Scrub     Grassland Grassland
Grassland      Meadow    Meadow    Scrub     Arable    Grassland Scrub
```

```
> worms$Slope
 [1] 11  2  3  5  0  2  3  0  0  4 10  1  2  6  0  0  8  2  1 10
```

```
> worms$Vegetation == "Grassland" | worms$Slope > 0
 [1]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
[13]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
```

intersection de deux conditions (et exclusif): &

```
> worms$Vegetation == "Grassland" & worms$Slope > 0
 [1]  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE  TRUE
[13]  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
```

Manipulations de base dans R

Les tests logiques

`ifelse( )` test conditionnel sur un vecteur

On peut créer une variable binaire à partir d'une autre variable

```
> worms$Grassland<-ifelse(worms$Vegetation=="Grassland", 1, 0)
> worms$Grassland
 [1] 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 0 0 1 0
```

On peut imbriquer les `ifelse( )` pour créer plusieurs catégories

```
> worms$Slope_cat<-ifelse(worms$Slope >= 0 & worms$Slope < 3, "Low",
ifelse(worms$Slope >= 3 & worms$Slope < 6, "Med",
ifelse(worms$Slope >= 6 & worms$Slope < 9, "High", "Very High")))
> worms$Slope_cat
 [1] "Very High" "Low"       "Med"       "Med"       "Low"  "Low"
 [7] "Med"       "Low"       "Low"       "Med"       "Very High“ “Low"
[13] "Low"       "High"      "Low"       "Low"       "High"      "Low"
[19] "Low"       "Very High"
```

Manipulations de base dans R

Les tests logiques

`which( )` pour determiner quels éléments correspondent à la condition

```
> which(worms$Slope==0)
[1]  5  8  9 15 16

> which(worms$Vegetation=="Grassland")
[1]  1  3  6  7 10 12 13 14 19
```

Manipulations de base dans R

Utiliser tests logiques pour sélectionner un sous-ensemble

```
> worms[worms$Vegetation=="Meadow",]
```

sélectionne les lignes pour lesquelles Vegetation = Meadow

```
> worms[worms$Area > 3 & worms$Slope < 3,]
```

sélectionne les lignes qui répondent aux deux conditions

Ou encore, on peut utiliser la fonction `subset( )`

```
> subset(x=worms, subset=worms$Area > 3 & worms$Slope < 3 )
```

Manipulations de base dans R

Pour faire un tri d'une variable

Utiliser `sort( )` pour ordre croissant

```
> worms$Slope
 [1] 11  2  3  5  0  2  3  0  0  4 10  1  2  6  0  0  8  2  1 10
```

`> sort(worms$Slope)` ordonne les valeurs d'une variable

```
 [1]  0  0  0  0  0  1  1  2  2  2  2  3  3  4  5  6  8 10 10 11
```

Utiliser `rev(sort( ))` pour ordre décroissant ou `sort( , decreasing = FALSE)`

```
> rev(sort(worms$Slope))
 [1] 11 10 10  8  6  5  4  3  3  2  2  2  2  1  1  0  0  0  0  0
```

Manipulations de base dans R

Pour faire un tri d'un jeu de données

Utiliser `order( )` pour ordre croissant

```
> worms[order(worms[,1]), 1:6]
```

trier les colonnes 1-6 par les valeurs de la colonne 1 en ordre croissant

Utiliser `rev(order( ))` pour ordre décroissant ou `order( , decreasing = FALSE)`

```
> worms[rev(order(worms[,1])), 1:6]
```

trier les colonnes 1-6 par les valeurs de la colonne 1 en ordre décroissant

Fonctions pratiques

Comment fait-on un tableau croisé dynamique?

`table( )` résume la fréquence de chaque catégorie d'un ou plusieurs facteurs

```
> table(worms$Vegetation)
   Arable Grassland    Meadow   Orchard     Scrub
        3         9         3         1         4

> table(worms$Vegetation, worms$Damp)
            FALSE TRUE
  Arable        3    0
  Grassland     8    1
  Meadow        0    3
  Orchard       1    0
  Scrub         2    2
```

Fonctions pratiques

Pour un meilleur exemple, regardons le fichier `Neural.txt` qui présente effet de traitement vs réduction de douleur chez patients.

```
> neural<-read.table("C:/Neural.txt", header=TRUE)
> neural[1:10,]
   Reduce Treatment Age Gender Duration
1     1         1  76      M       36
2     1         1  52      M       22
3     0         0  80      F       33
4     0         1  77      M       33
5     0         1  73      F       17
6     0         0  82      F       84
7     0         1  71      M       24
8     0         0  78      F       96
9     1         1  83      F       61
10    1         1  75      F       60
```

Fonctions pratiques

```
> tab<-table(neural$Reduce, neural$Treatment, deparse.level=2)
> tab
             neural$Treatment
neural$Reduce 0 1
            0 8 3
            1 0 7
```

`deparse.level=2` extrait les noms de chaque dimension du tableau

`as.data.frame( )` permet de convertir le tableau en jeu de données et l'utiliser dans des analyses subséquentes

```
> tab<-as.data.frame(tab)
> tab
  neural.Reduce neural.Treatment Freq
1             0               0    8
2             1               0    0
3             0               1    3
4             1               1    7
```

Fonctions pratiques

Si on a une colonne associée à la fréquence, on peut résumer les données avec `xtabs( )`

```
> admissions <- as.data.frame(UCBAdmissions)
> admissions
      Admit Gender Dept Freq
1  Admitted   Male    A  512
2  Rejected   Male    A  313
3  Admitted Female    A   89
4  Rejected Female    A   19
...
> xtabs(Freq ~ Gender + Admit, data = admissions)
        Admit
Gender   Admitted Rejected
  Male       1198     1493
  Female      557     1278
```

il y a plusieurs jeux de données déja inclus dans R, on utilise `data( )` pour voir ceux disponibles.

Fonctions pratiques

`reshape( )` permet de réarranger les données en format large ou long

```
> Indometh
   Subject time conc
1        1 0.25 1.50
2        1 0.50 0.94
3        1 0.75 0.78
4        1 1.00 0.48
5        1 1.25 0.37
6        1 2.00 0.19
7        1 3.00 0.12
8        1 4.00 0.11
9        1 5.00 0.08
10       1 6.00 0.07
11       1 8.00 0.05
12       2 0.25 2.03
13       2 0.50 1.63
...
>
Subject conc.0.25 conc.0.5 conc.0.75 conc.1 conc.1.25 conc.2 conc.3 conc.4 conc.5 conc.6 conc.8
      1      1.50     0.94      0.78   0.48      0.37   0.19   0.12   0.11   0.08   0.07   0.05
      2      2.03     1.63      0.71   0.70      0.64   0.36   0.32   0.20   0.25   0.12   0.08
...
```

voir aussi `stack( )` et `unstack( )`

Exportation de tableaux à partir de R

la fonction `write.table( )`

```
write.table(x=nom_du_tableau, file="c:\\chemin_sur_votre_ordi\\worms.txt",
row.names=FALSE, col.names=TRUE, sep="\t")
```

`row.names=F` indique que l'on ne veut pas de nom de rangées
(si désiré, indiquer = `TRUE`)

`col.names=TRUE` indique que l'on veut préserver le nom des colonnes

`sep="\t"` indique que l'on désire des tabulations comme séparateur des valeurs

(`sep=" "` pour espace comme séparateur)