Formation R - Tronc commun

Sylvain Mareschal

Janvier 2015

Centre Henri Becquerel
INSERM U918

Formation R - Tronc commun

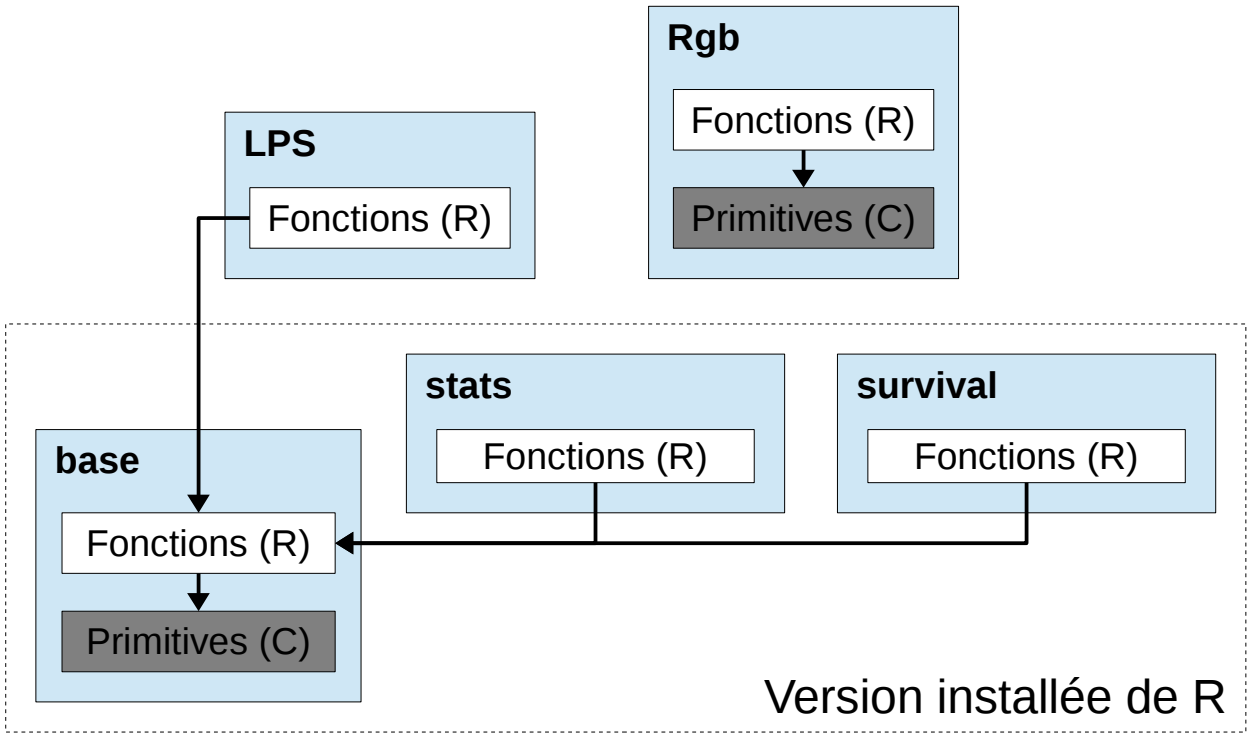
Partie A - Fondamentaux

Dérivé du langage « S »
Implémentation originale par Bell labs (1976)
« Turn ideas into software, quickly and faithfully » (John Chambers)
Réimplémenté par Ross Ihaka & Robert Gentleman (1993)

Implémentation libre
GNU General Public License (GPL)
Gratuit, sources accessibles et modifiables
Sous « copyleft »
≠ SAS©, STATA©, MATLAB© ...

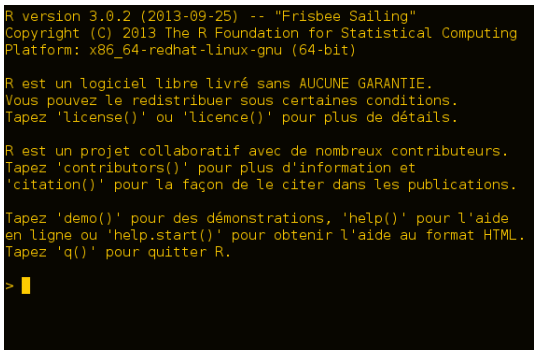
Langage de programmation
Définit une syntaxe
résultat <- fonction(argument)
Définit un vocabulaire
mean, median, sd, +, % ...
Une infinité de combinaisons

Logiciel modulaire
14 packages de base
6204 packages au CRAN
934 packages dans Bioconductor
Interfaces C, C++, Fortran, Java ...
Interfaces web, base de données ...
Interfaces binaires ...

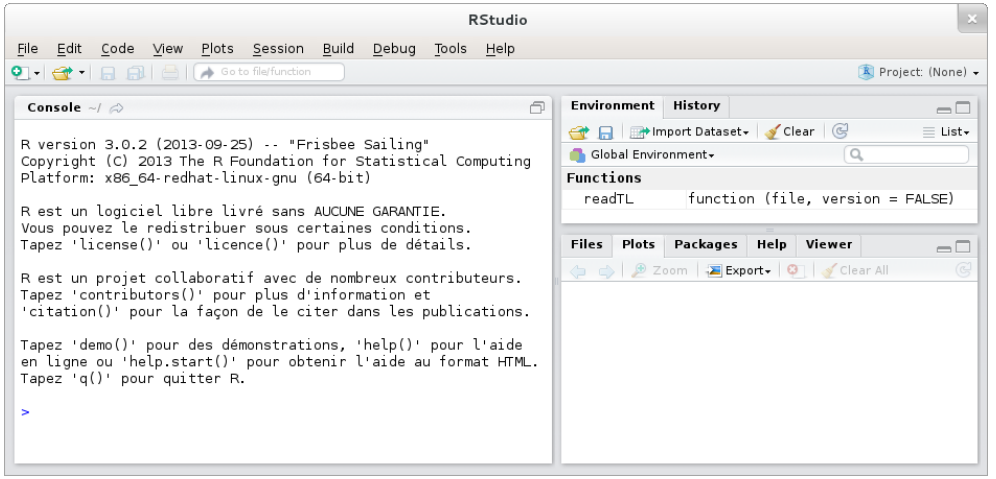


***Command Line Interface* (CLI)**
Écriture et envoi des lignes de commande
Retour direct des résultats au format texte

```
> log(1+8)
[1] 2.197225
```

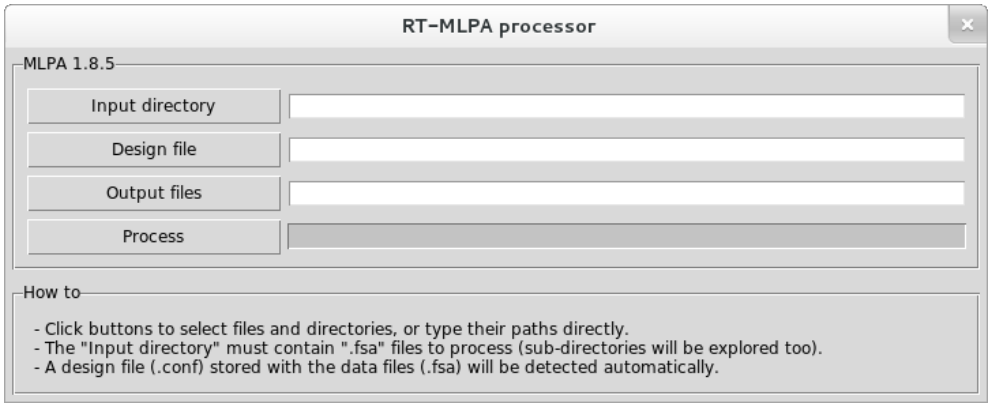


***Graphical User Interface* (GUI)**
Logiciel d'habillage (Rgui, RStudio ...)
Facilite l'utilisation :
- Auto-complétion
- Gestion des graphiques produits
- Liste des variables déclarées
- ...



Scripts (.r)
Fichier texte avec une commande par ligne
Exécution séquentielle par R
Permet de reproduire une analyse complète
Possibilité de gérer les arguments

Programmes interfacés
Possibilité de construire une interface
Repose sur un package (tcltk)
Boutons, menus déroulants ...
Hors programme



***Graphical User Interface* (GUI)**
Gratuit, open-source (licence AGPL)
Multi-plateforme (Windows, MacOS, Linux)
Licence commerciale optionnelle (995$ / an)
Version Desktop

Pratique - Installation de R
The Comprehensive R Archive Network (CRAN)
http://cran.r-project.org

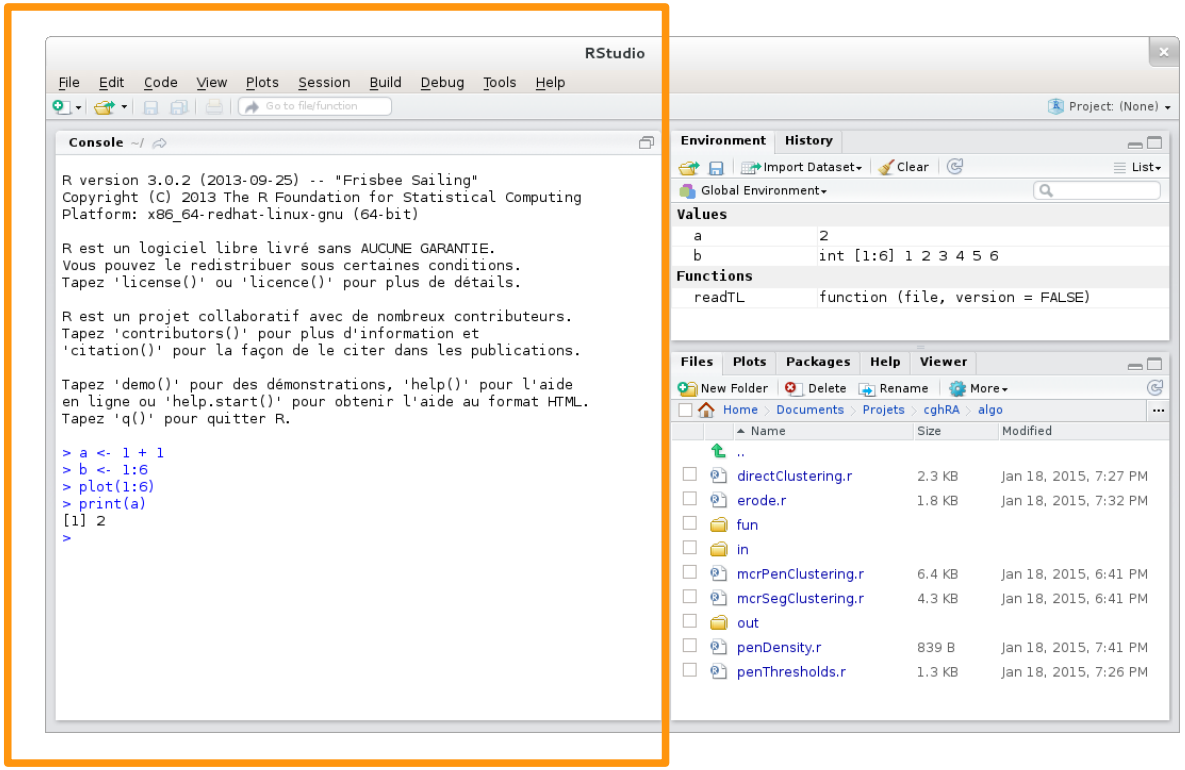
Pratique - Installation de R Studio
http://www.rstudio.com

Pratique - Découverte de l'interface

Pratique - Découverte de l'interface

Console

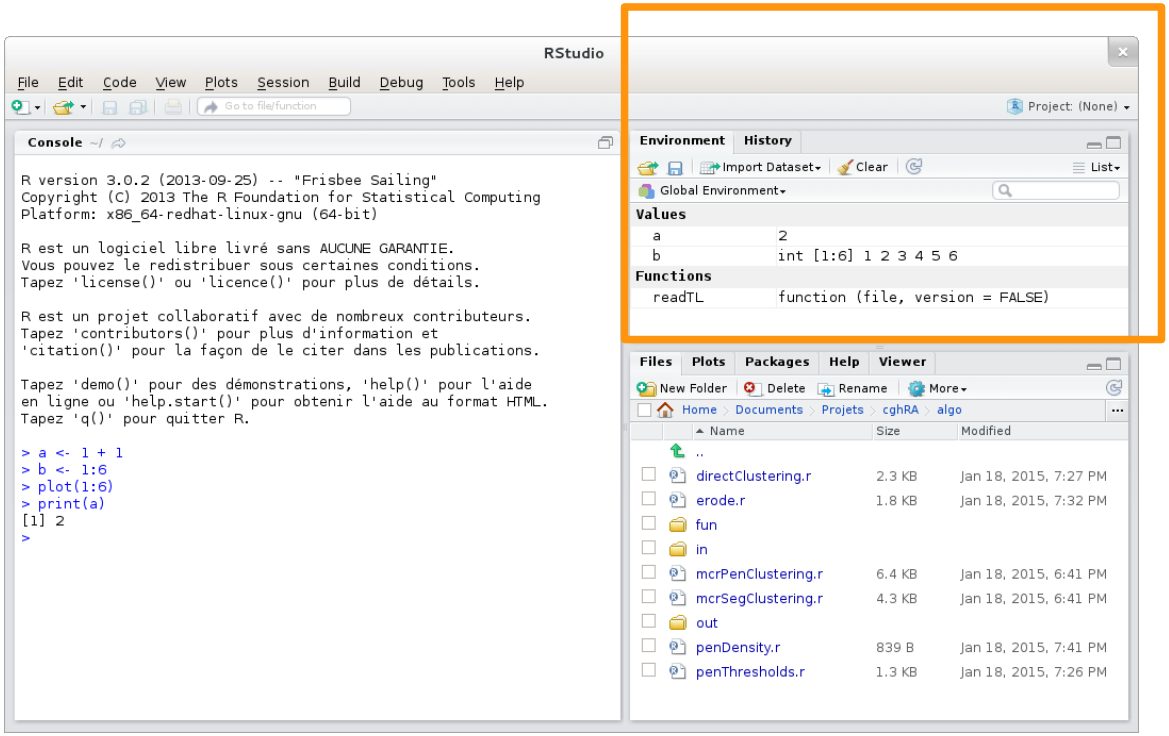
R proprement dit
Commandes en bleu
Résultats en noir



Pratique - Découverte de l'interface

Environment

Liste les éléments définis jusqu'à présent
Variables, fonctions, tableaux ...

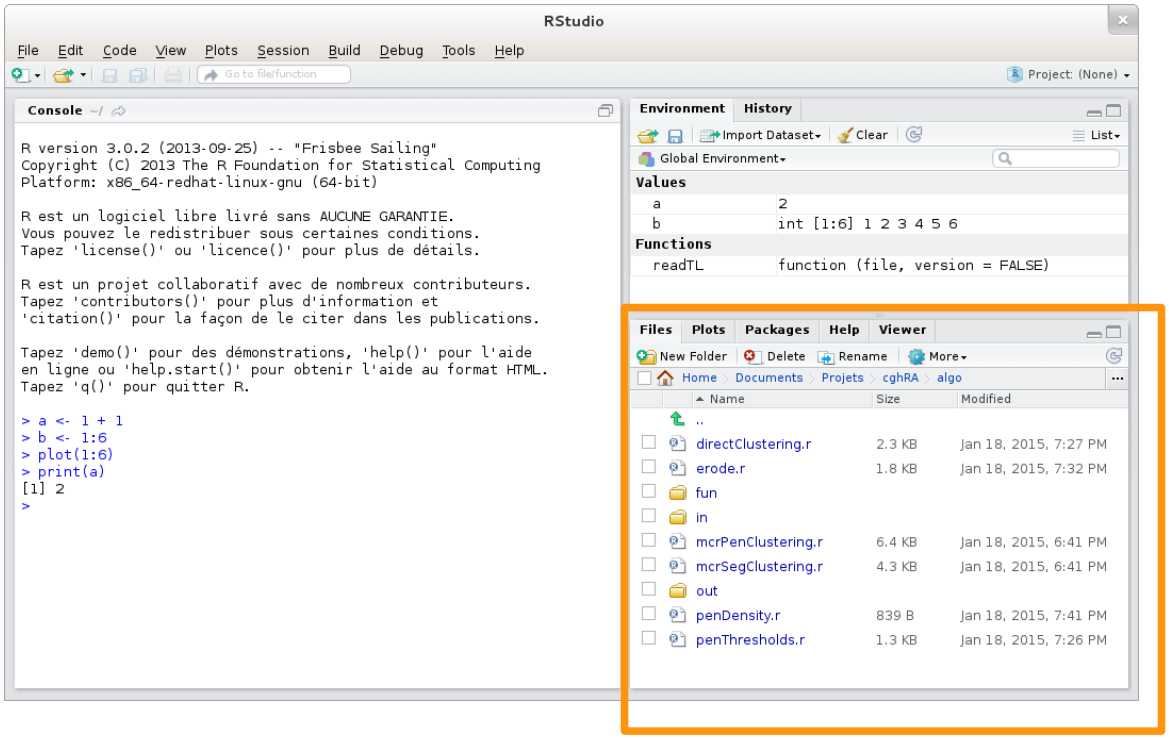


Pratique - Découverte de l'interface

Files

Explorateur de fichiers

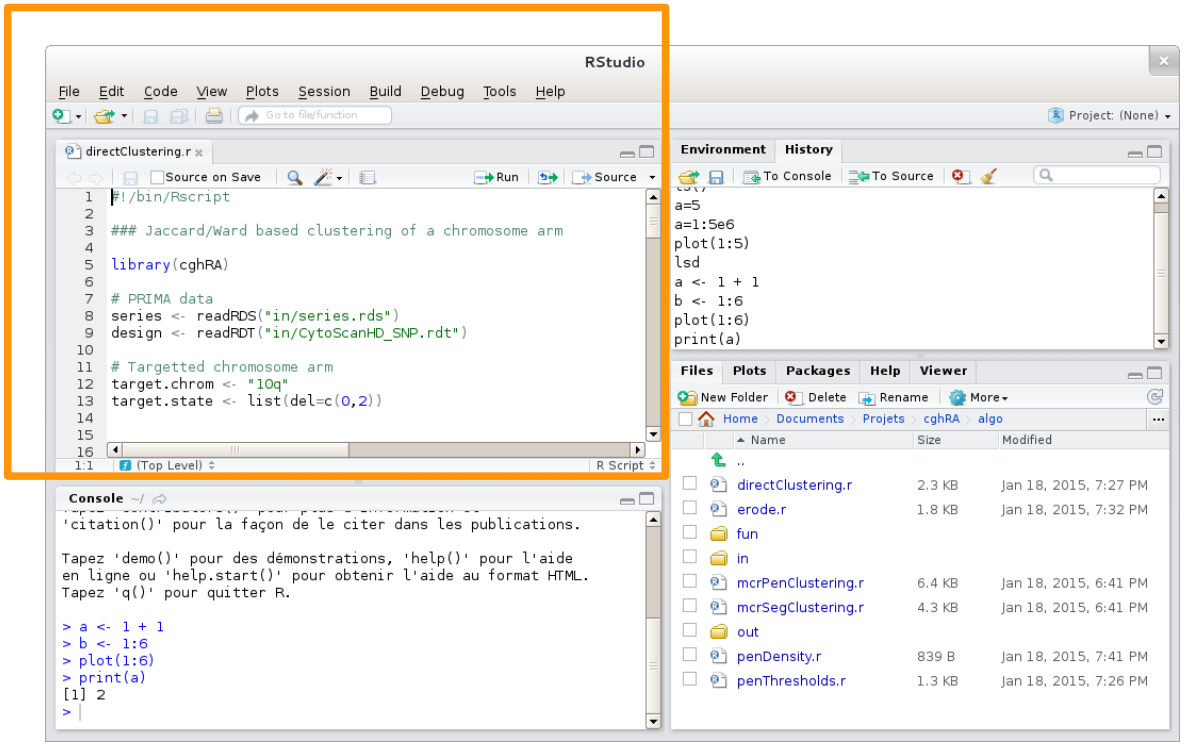
Permet de retrouver ses scripts R



Pratique - Découverte de l'interface

Éditeur de fichiers

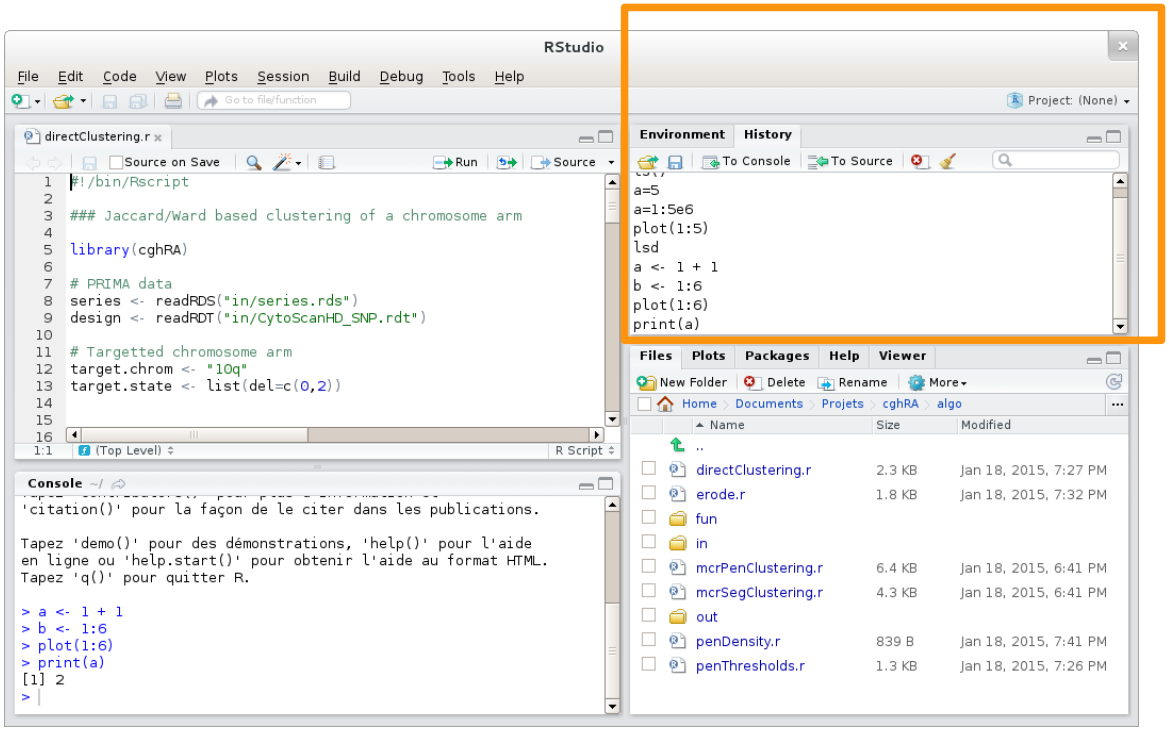
Éditeur de texte
Coloration syntaxique
Possibilité d'exécuter une ligne / un bloc / tout le script



Pratique - Découverte de l'interface

History

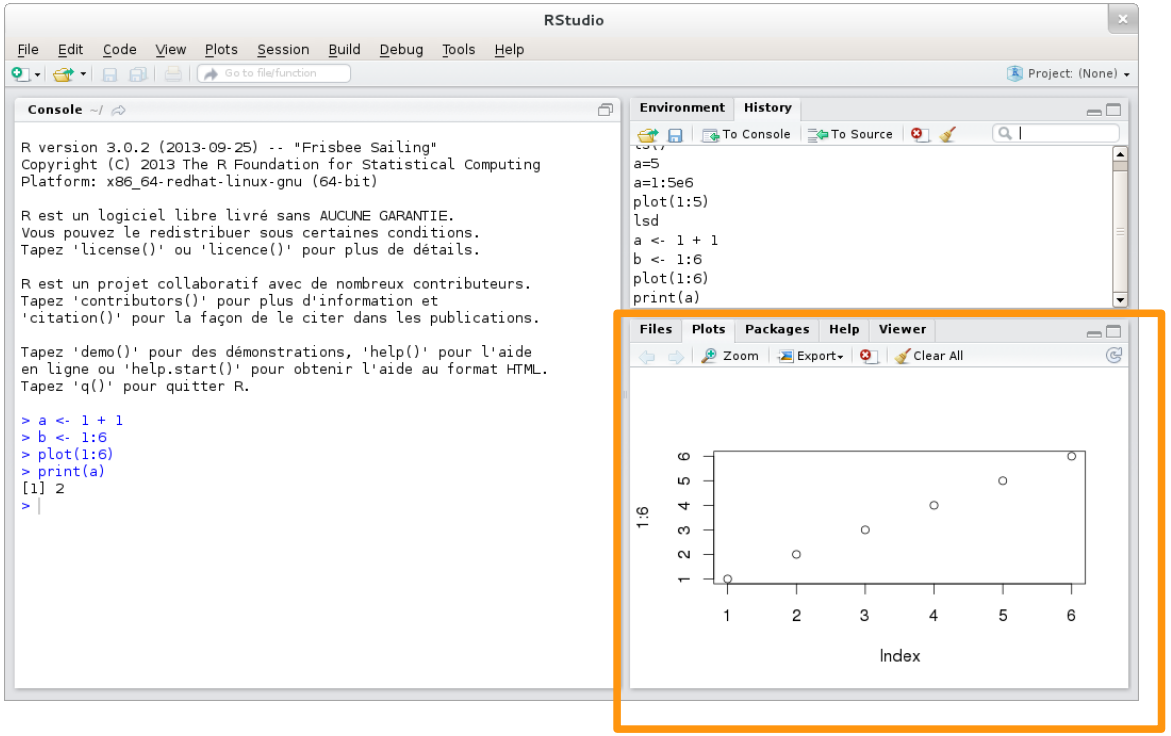
Historique des commandes exécutées
Possibilité de charger / enregistrer l'historique
Possibilité de copier une ligne vers le script ou la console



Pratique - Découverte de l'interface

Plots

Historique des graphiques générés
Possibilité d'exporter en PDF, JPG, PNG, BMP, TIFF, EPS, SVG ...
Possibilité de voir dans une fenêtre plus grande (Zoom)

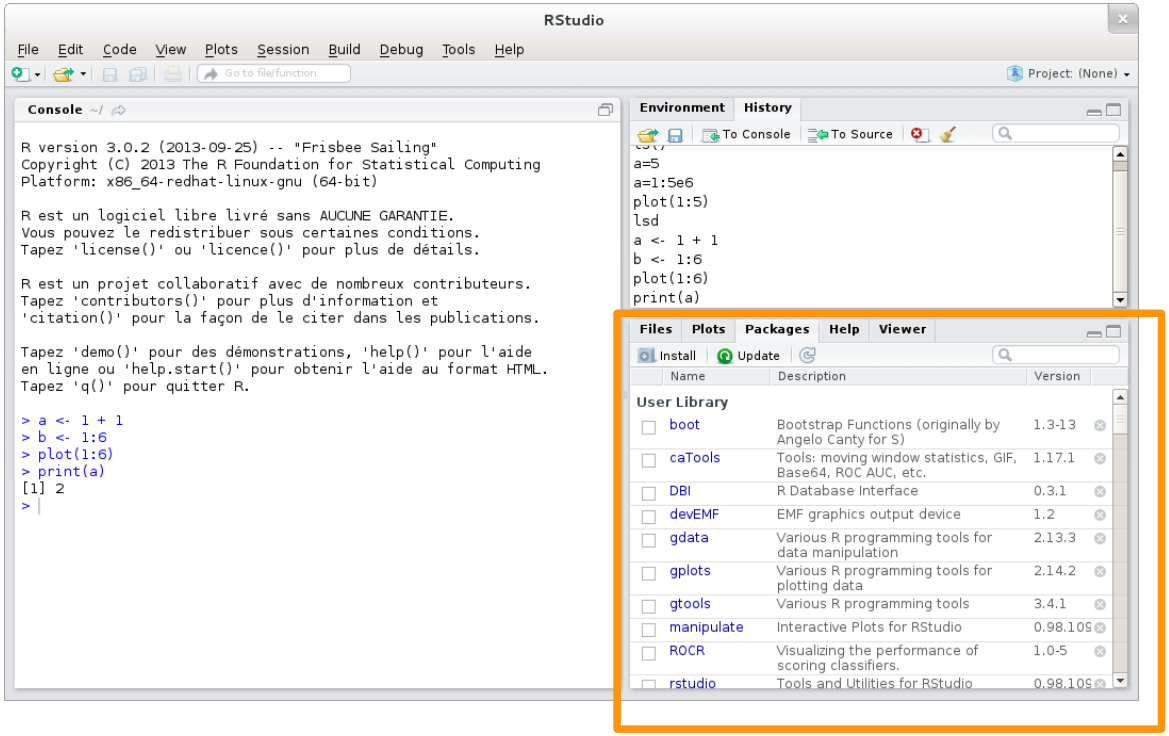


Pratique - Découverte de l'interface

Packages

Liste des packages installés

Accès internet au CRAN pour en installer des nouveaux



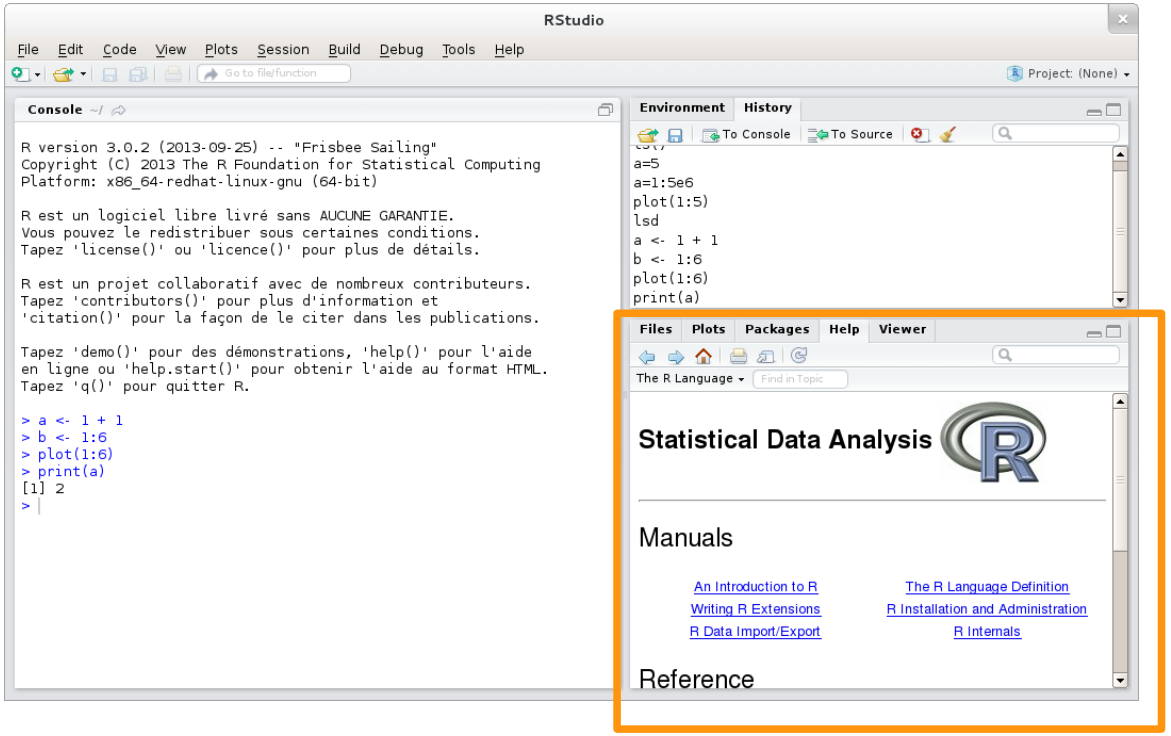
Pratique - Découverte de l'interface

Help

Aide interactive de R (format web, données locales)
Affichage des pages d'aides demandées en console
Possibilité de voir dans une fenêtre plus grande

Viewer

Navigateur web local
Pas utile dans cette formation



Une calculatrice améliorée
R comprend tout ce qu'une calculatrice comprend :

Addition

```
1 + 1
```

Soustraction

```
8 - 2
```

Multiplication

```
4 * 6
```

Division

```
12 / 4
```

Décimales

```
1.2 + 1.8
```

Parenthèses

```
(1+2+3) / 3
((1+2+3)/2)+((4+7+8)/12)
```

Puissances

```
2^3
```

Une calculatrice améliorée
R comprend aussi ce qu'une calculatrice scientifique comprend :

Racine carrée (« Squared root »)

```
sqrt(9)
```

Logarithme

```
log(1)
```

Trigonométrie

```
cos(0)
sin(1)
tan(1)
```

Syntaxe des appels à fonctions

nom_de_la_fonction(arguments)

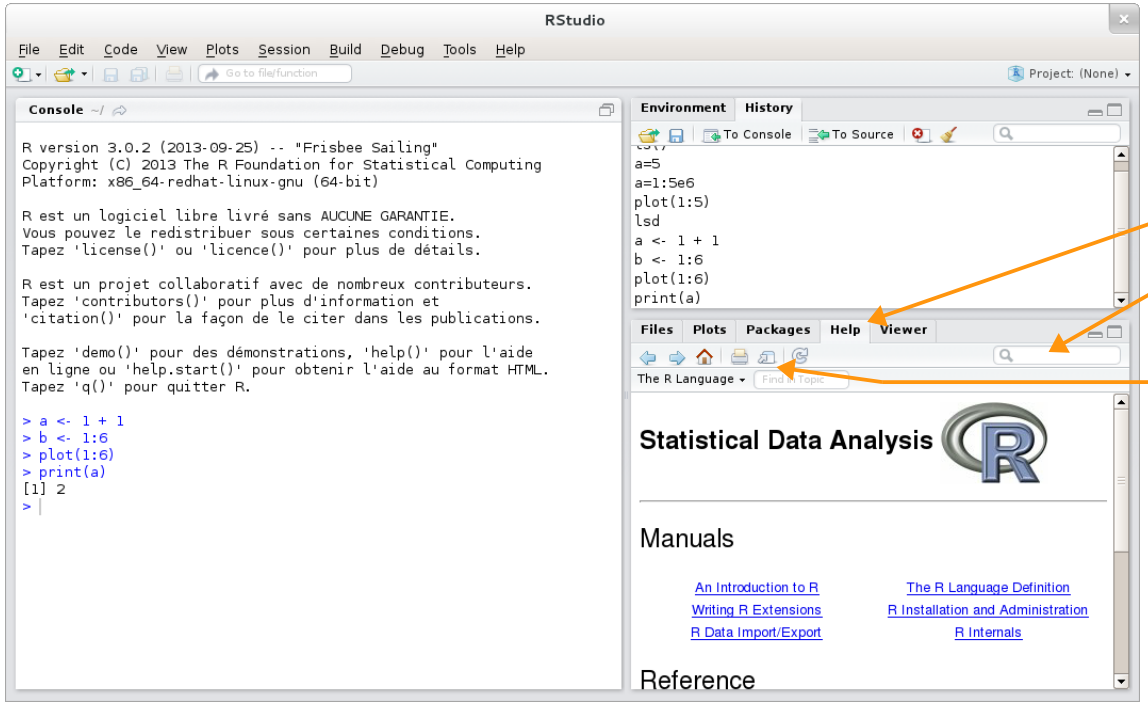
Un ou plusieurs arguments

Logarithme à base 10

```
log(1000, 10)
```

Problèmes soulevés
Comment s'appelle la fonction dont j'ai besoin ?
Quels arguments et dans quel ordre ?
⟶ utiliser l'aide de R !

Pratique - Demander de l'aide sur la fonction « log »



En utilisant R Studio

1. Ouvrir le panneau « Help »
2. Entrer le nom de la fonction
3. Appuyer sur « Entrée »
4. *Cliquer sur le bouton « Show in a new window »*

En ligne de commandes

```
help(log)
?log
```

Pratique - Demander de l'aide sur la fonction « log »

log {base} R Documentation

Logarithms and Exponentials

Description

log computes logarithms, by default natural logarithms, log10 computes common (i.e., base 10) logarithms, and log2 computes binary (i.e., base 2) logarithms. The general form log(x, base) computes logarithms with base base.

log1p(x) computes *log(1+x)* accurately also for *|x| << 1* (and less accurately when *x is approximately -1*).

exp computes the exponential function.

expm1(x) computes *exp(x) - 1* accurately also for *|x| << 1*.

Usage

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)

log1p(x)

exp(x)
expm1(x)
```

Arguments

x	a numeric or complex vector.
base	a positive or complex number: the base with respect to which logarithms are computed. Defaults to *e*=exp(1).

Details

All except logb are generic functions: methods can be defined for them individually or via the Math group generic.

log10 and log2 are only convenience wrappers, but logs to bases 10 and 2 (whether computed *via* log or the wrappers) will be computed more efficiently and accurately where supported by the OS. Methods can be set for them individually (and otherwise methods for log will be used).

logb is a wrapper for log for compatibility with S. If (S3 or S4) methods are set for log they will be dispatched. Do not set S4 methods on logb itself.

All except log are primitive functions.

Value

A vector of the same length as x containing the transformed values. log(0) gives -Inf, and log(x) for negative values of x is NaN. exp(-Inf) is 0.

For complex inputs to the log functions, the value is a complex number with imaginary part in the range *[-pi, pi]*: which end of the range is used might be platform-specific.

S4 methods

exp, expm1, log, log10, log2 and log1p are S4 generic and are members of the Math group generic.

Note that this means that the S4 generic for log has a signature with only one argument, x, but that base can be passed to methods (but will not be used for method selection). On the other hand, if you only set a method for the Math group generic then base argument of log will be ignored for your class.

Source

log1p and expm1 may be taken from the operating system, but if not available there are based on the Fortran subroutine dlnrel by W. Fullerton of Los Alamos Scientific Laboratory (see http://www.netlib.org/slatec/fnlib/dlnrel.f and (for small x) a single Newton step for the solution of log1p(y) = x respectively.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole. (for log, log10 and exp.)

Chambers, J. M. (1998) *Programming with Data. A Guide to the S Language*. Springer. (for logb.)

See Also

Trig, sqrt, Arithmetic.

Examples

```
log(exp(3))
log10(1e7) # = 7

x <- 10^-(1+2*1:9)
cbind(x, log(1+x), log1p(x), exp(x)-1, expm1(x))
```

[Package *base* version 3.0.2 Index]

Description
Présentation rapide de la fonction

Usage
Présentation des arguments et leurs valeurs par défaut

Arguments
Explication des différents arguments

Details
Présentation approfondie de la fonction

Value
Qu'est ce que cette fonction affiche comme résultat ?

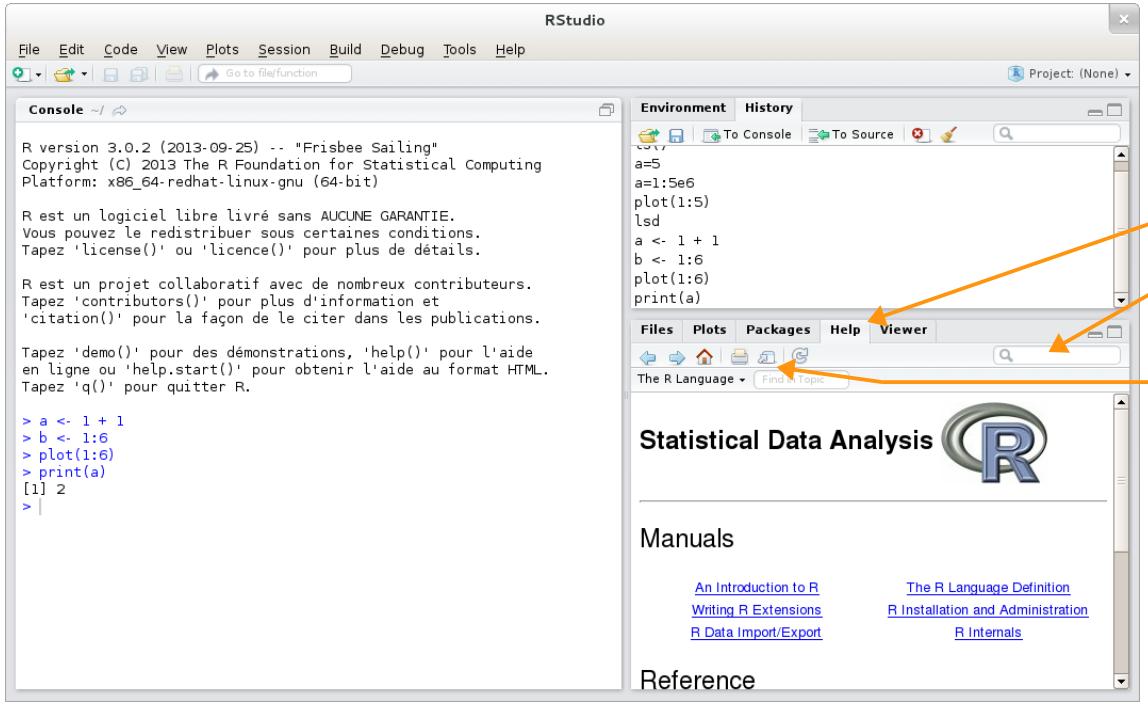
Paragraphes optionnels

References
Livres et articles décrivant le calcul implémenté

See Also
Liens vers l'aide d'autres fonctions similaires

Examples
Exemples de code R utilisant cette fonction

Pratique - Retrouver la fonction affichant l'heure actuelle



En utilisant R Studio

1. Ouvrir le panneau « Help »
2. Entrer le mot clé recherché
3. Appuyer sur « Entrée »
4. *Cliquer sur le bouton « Show in a new window »*

En ligne de commandes

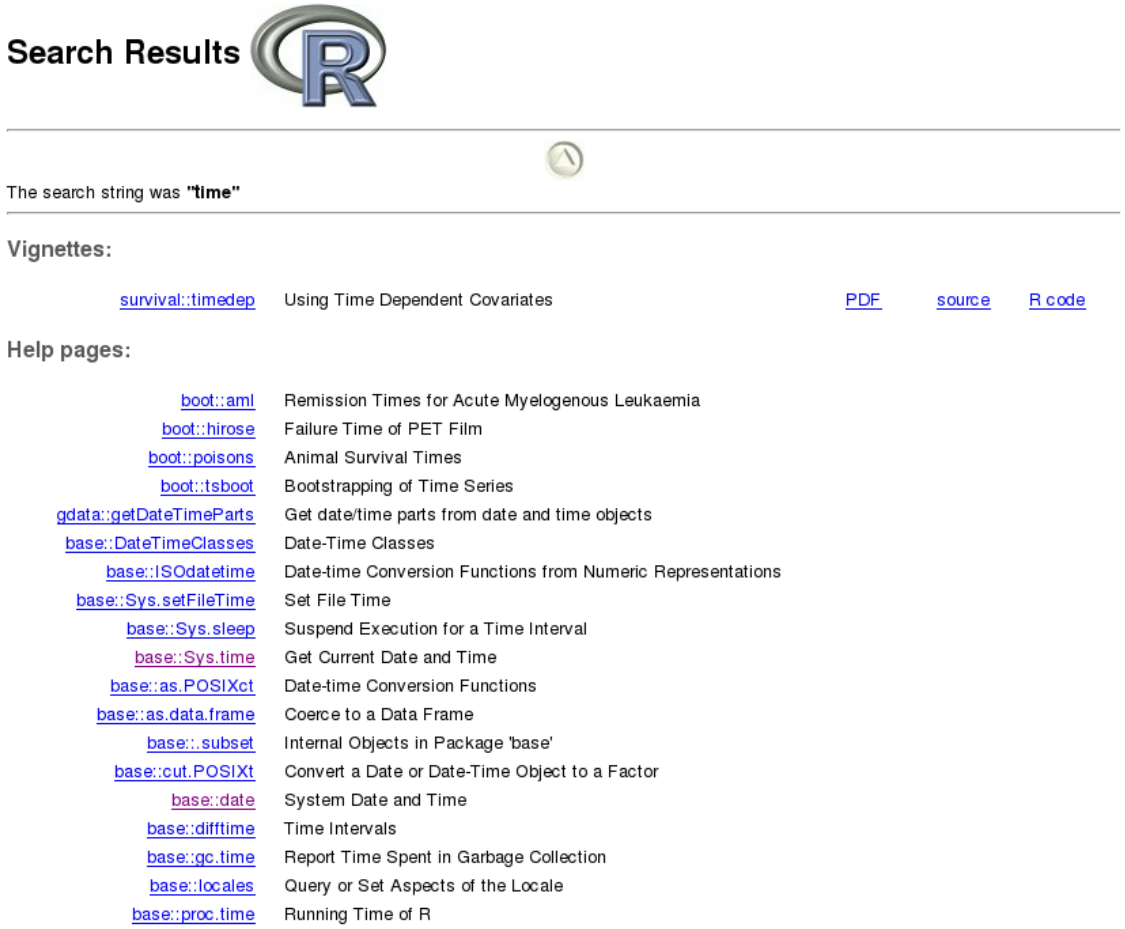
```
help.search(time)
??time
```

En utilisant Google

"R" current time

Pratique - Retrouver la fonction affichant l'heure actuelle

package::fonction Description



Search Results

The search string was **"time"**

Vignettes:

survival::timedep	Using Time Dependent Covariates	PDF	source	R code

Help pages:

boot::aml	Remission Times for Acute Myelogenous Leukaemia
boot::hirose	Failure Time of PET Film
boot::poisons	Animal Survival Times
boot::tsboot	Bootstrapping of Time Series
gdata::getDateTimeParts	Get date/time parts from date and time objects
base::DateTimeClasses	Date-Time Classes
base::ISOdatetime	Date-time Conversion Functions from Numeric Representations
base::Sys.setFileTime	Set File Time
base::Sys.sleep	Suspend Execution for a Time Interval
base::Sys.time	Get Current Date and Time
base::as.POSIXct	Date-time Conversion Functions
base::as.data.frame	Coerce to a Data Frame
base::.subset	Internal Objects in Package 'base'
base::cut.POSIXt	Convert a Date or Date-Time Object to a Factor
base::date	System Date and Time
base::difftime	Time Intervals
base::gc.time	Report Time Spent in Garbage Collection
base::locales	Query or Set Aspects of the Locale
base::proc.time	Running Time of R

Arguments obligatoires
Messages d'erreur plutôt clairs :

```
log()
print()
```

Comment faire la différence ?
Page d'aide, paragraphe « Usage »

Usage

```
log(x, base = exp(1))
```

Valeur par défaut pour base : « exp(1) »
Pas de valeur par défaut pour x

Arguments à donner dans l'ordre …

```
log(1000, 10)
```

x base

… ou à nommer (préférable)

```
log(x=1000, base=10)
log(base=10, x=1000)
```

Pratique :

$$\frac{1}{2^{\log_6\left(\frac{7}{4}\times 8\right)+\sqrt{2{,}123}}}$$

```
1 / 2^(log((7/4)*8, base=6) + sqrt(2.123))
```

Langage <u>procédural</u>
Une fonction suffit rarement à résoudre le problème
Nécessité de découper le problème en étapes
Nécessité de stocker les résultats intermédiaires

Variable
Emplacement en mémoire
Contient une seule valeur à la fois
Respecte un type (nombre, texte ...)

Syntaxe du stockage de valeur

nom <- valeur

```
a <- 8
print(a)
a

b <- 7
a + b

print(b)
b <- b + 1
print(b)

g <- "ABC"
bonjour <- "Bonjour tout le monde !"
```

Pratique - Appliquer la formule suivante à x=1, x=2 et x=3

$$\frac{x}{\log_2(x) + x}$$

```
x <- 1
x / (log(x, 2) + x)
x <- x + 1
x / (log(x, 2) + x)
x <- x + 1
x / (log(x, 2) + x)
```

Pratique - Échanger les valeurs des variables « a » et « b »

```
a <- b
b <- a
```

```
c <- a
a <- b
b <- c
```

Formation R - Tronc commun

Partie B - Application

Point de départ : Excel
RTMLPA.xls

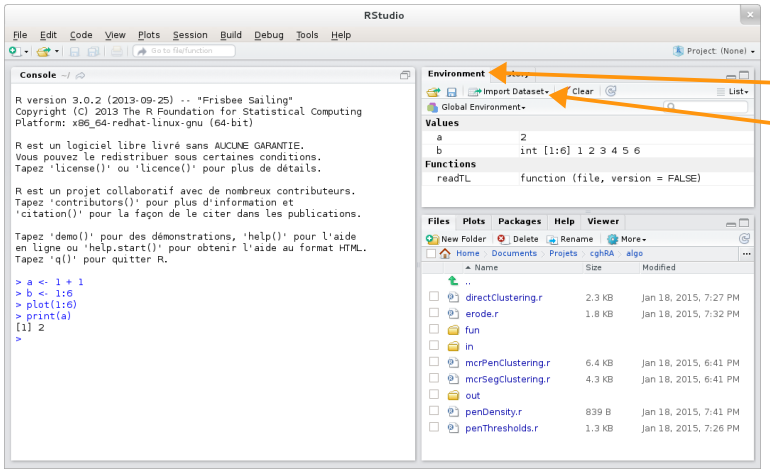
Formats XLS(X) 😡
Formats binaires, propriétaires
Varient d'une version à l'autre

Format CSV 😊
Format « plat », « Comma Separated Values »
Une ligne de texte = une ligne du tableau
Colonnes séparées par des virgules (*comma*)

Fichier > Enregistrer sous > CSV
Attention, Excel en Europe utilise des « ; »

A vérifier avec un éditeur de texte
Clic droit > Ouvrir avec > Bloc Note

```
Série;N° Patient;IHC;MLPA;IRF4;LMO2
training;UPN0964;;GCB;2,168;2,195
training;UPN1010;;other;2,178;0,898
training;UPN1149;;ABC;1,314;0,403
training;UPN0602;;GCB;0,057;0,399
training;UPN1028;;ABC;2,295;0,356
training;UPN1251;;GCB;0,822;1,854
training;UPN0853;;GCB;1,148;0,959
training;UPN1281;;ABC;1,944;0,346
training;UPN1291;;ABC;1,685;0,156
training;UPN1313;;ABC;1,618;0,293
training;UPN0943;;ABC;1,687;0,038
training;UPN1370;;ABC;1,234;0,075
training;UPN1386;;GCB;0,957;1,734
training;UPN0497;;ABC;1,503;0,319
training;UPN0526;;ABC;1,687;0,165
training;UPN0721;;ABC;1,563;0,185
training;UPN0987;;other;1,568;1,158
training;UPN1392;;GCB;1,608;1,477
training;UPN0466;;ABC;1,867;0,255
training;UPN0494;;GCB;1,659;1,489
training;UPN1404;;GCB;0,365;0,807
training;UPN1415;;ABC;2,028;0,051
training;UPN1443;;GCB;0,214;1,263
training;UPN1458;;GCB;1,757;1,259
training;UPN1465;;ABC;1,784;0,119
training;UPN1485;;GCB;0,781;1,182
training;UPN1486;;ABC;2,052;0,066
training;UPN1505;;GCB;1,298;0,886
training;UPN1525;;GCB;1,485;1,936
training;UPN1540;;ABC;1,275;0,051
training;UPN1541;;ABC;2,096;0,147
training;UPN0235;;GCB;0,864;1,065
training;UPN0878;;ABC;1,691;0,081
training;UPN0937;;ABC;1,998;0,301
training;UPN1574;;ABC;1,508;0,446
training;UPN1583;;other;1,103;0,471
...
```

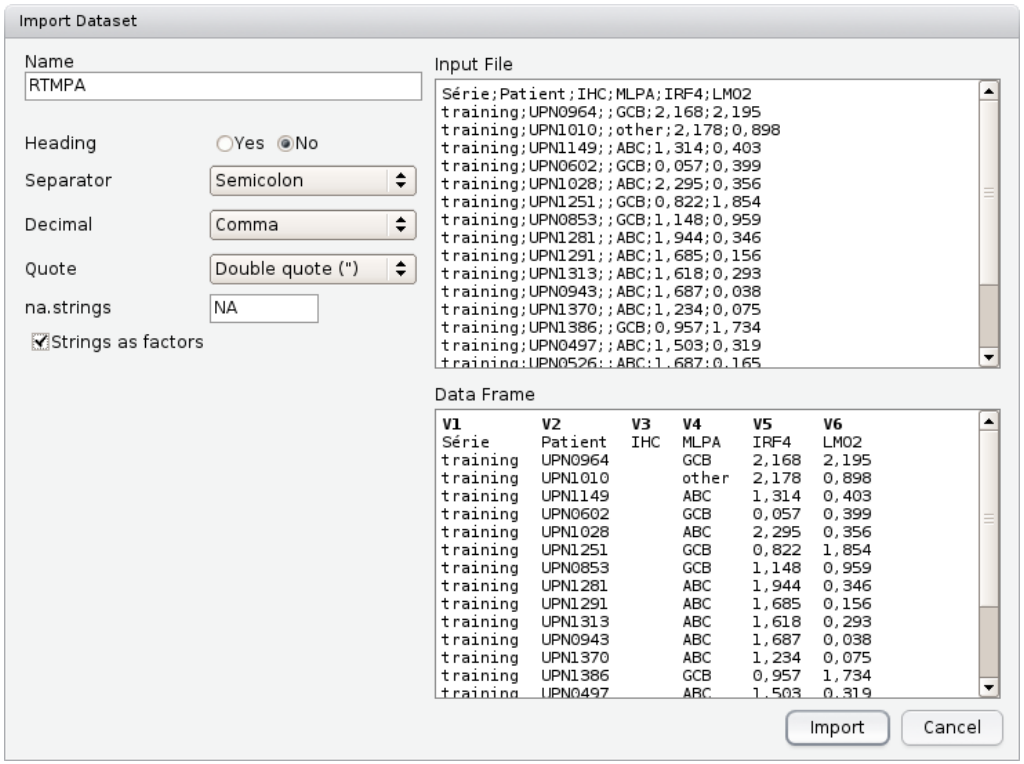


En utilisant R Studio

1. Ouvrir le panneau « Environment »
2. Appuyer sur « Import dataset »
3. « From Text file »
4. Sélectionner le fichier CSV

En ligne de commandes

```
tab <- read.csv2(file=...)
```



Name
Nom de la variable à remplir

Heading
Y a-t-il une ligne avec les noms des colonnes ?

[Column] Separator
Semicolon (;), Comma (,), Tabulation (↹)

Decimal [separator]
Period (.) ou Comma (,)

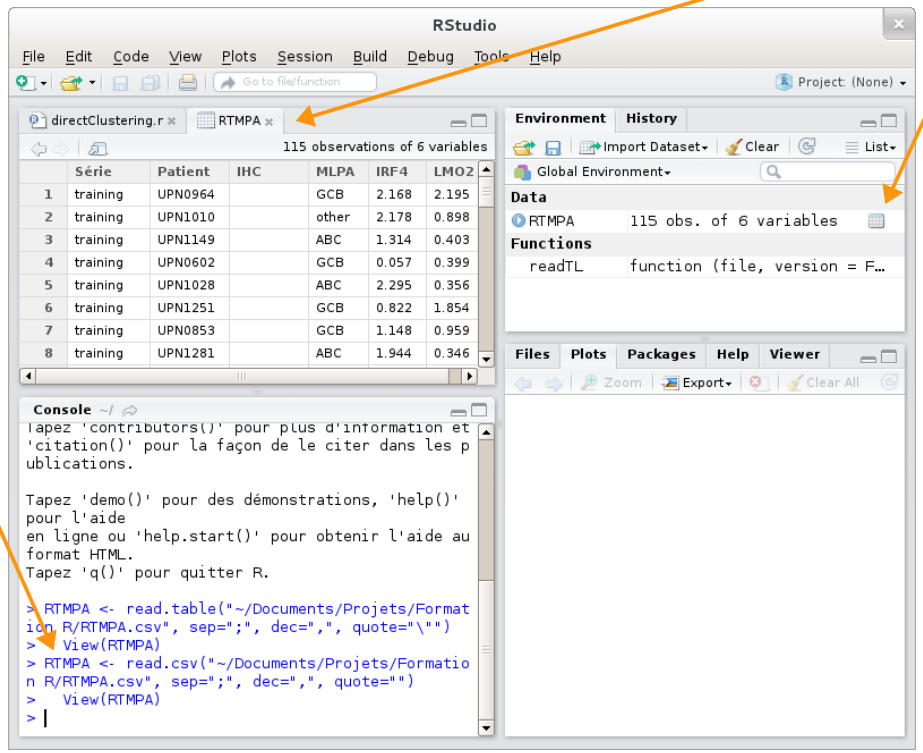
Quote
Protection optionnelle des valeurs

NA strings
Texte à interpréter comme « Not Available »

Strings as factors
Décocher, cf module « Vecteurs & Tables »

Génération de la commande correspondante

Visualisation du contenu



	Série	Patient	IHC	MLPA	IRF4	LMO2
1	training	UPN0964		GCB	2.168	2.195
2	training	UPN1010		other	2.178	0.898
3	training	UPN1149		ABC	1.314	0.403
4	training	UPN0602		GCB	0.057	0.399
5	training	UPN1028		ABC	2.295	0.356
6	training	UPN1251		GCB	0.822	1.854
7	training	UPN0853		GCB	1.148	0.959
8	training	UPN1281		ABC	1.944	0.346

Noms de colonnes

R remplace les caractères spéciaux par des « . »

Privilégier les « _ » aux espaces dans vos données

Les accents sont tolérés, mais à éviter

L'import remplit une variable
Un tableau (« data.frame ») est un type de variable
Chaque colonne du tableau est elle même une variable
Chaque colonne a un type (texte ou nombre) cohérent et indépendant

Vérifier ses données
Utiliser l'interface (diapo précédente)
Afficher directement la variable

```
RTMLPA
print(RTMLPA)
```

Éditer ses données

fix(tableau)

Repose sur une interface type « tableau Excel »
A éviter pour préserver la reproductibilité de l'analyse

Filtrer un tableau : subset

subset(x, subset, select)

« x » est un tableau
« subset » définit une condition que les lignes doivent respecter
« select » liste les colonnes à garder

Appliquer une condition

```
subset(RTMLPA, IHC == "GCB")
```

Appliquer plusieurs conditions

Au plus simple : appels successifs

```
x <- RTMLPA
x <- subset(x, IHC != "")
x <- subset(x, MLPA != "GCB")
```

Au plus court : opérateurs logiques

```
x <- subset(RTMLPA, IHC != "" & MLPA != "GCB")
```

Choisir des colonnes

```
subset(RTMLPA, select="IHC")
subset(RTMLPA, select=c("MLPA", "IHC"))
```

Opérateurs relationnels

==	égalité
!=	différence
<	infériorité stricte
>	supériorité stricte
<=	inférieur ou égal
>=	supérieur ou égal

Opérateurs logiques

&	et
|	ou *(Alt Gr + 6)*
!	inverse

Pratique - Extraire l'expression de LMO2 dans la série de training, chez les GCB d'une part, et les ABC d'autre part (selon la MLPA)

```
abc <- subset(RTMLPA, MLPA=="ABC" & Série=="training", "LMO2")
gcb <- subset(RTMLPA, MLPA=="GCB" & Série=="training", "LMO2")
```

Vecteurs et Scalaires

Une variable peut contenir :
- une unique valeur (c'est un « scalaire »)
- un groupe de valeurs (c'est un « vecteur »)

Les valeurs d'un vecteur sont connectées :
- elles ont toutes le même type (texte, nombre entier, nombre à virgule ...)
- elles ont généralement la même provenance, échelle, unité

Une colonne d'un tableau en R est un vecteur

Attention à subset

subset() retourne toujours un tableau, même à 1 colonne
La plupart des fonctions en R utilisent des vecteurs, pas des tableaux
⟶ utiliser l'argument « drop » de subset()
⟶ bien distinguer les tableaux (« Data » dans R Studio) et vecteurs (« Values »)

Observer la différence dans l'onglet « Environment » :

```
tableau <- subset(RTMLPA, select="LMO2")
vecteur <- subset(RTMLPA, select="LMO2", drop=TRUE)
```

Un élément fondamental en R

La plupart des fonctions et opérateurs sont sérialisés

```
print(vecteur)
print(vecteur + 10)
```

Diminue grandement le nombre d'opérations dans les analyses
Offre pas mal d'occasions de s'arracher les cheveux ...

Le couteau suisse : summary

summary(vecteur)
summary(tableau)

Appliqué à un vecteur, retourne minimum, maximum, quartiles, médiane et moyenne
Appliqué à un tableau, il s'applique indépendamment à chaque colonne

```
summary(RTMLPA)
```

Les fonctions individuelles

sd(vecteur)
mean(vecteur)
median(vecteur)
min(vecteur)
max(vecteur)

S'appliquent toutes à un vecteur

```
LMO2 <- subset(RTMLPA, select="LMO2", drop=TRUE)
mean(LMO2)
```

Pratique - Calculer la moyenne d'expression de LMO2 dans les ABC et les GCB

```
LMO2_ABC <- subset(RTMLPA, MLPA=="ABC", select="LMO2", drop=TRUE)
LMO2_GCB <- subset(RTMLPA, MLPA=="GCB", select="LMO2", drop=TRUE)
mean(LMO2_ABC)
mean(LMO2_GCB)
```

Dessiner un nuage de points : plot

plot(x, y, ...)

« x » et « y » sont des vecteurs numériques

Pratique - LMO2 = f(IRF4)dans la série de training

```
LMO2 <- subset(RTMLPA, Série=="training", select="LMO2", drop=TRUE)
IRF4 <- subset(RTMLPA, Série=="training", select="IRF4", drop=TRUE)
plot(x=IRF4, y=LMO2)
```

Paramètres graphiques

Basique : voir l'aide de « plot »
Avancés : voir l'aide de « par »
Bornes des axes : « xlim » et « ylim »

```
plot(x=IRF4, y=LMO2, xlim=c(0,3), ylim=c(0,3))
```

Pratique - Améliorer le graphique précédent

```
plot(x=IRF4, y=LMO2, xlab="Expression d'IRF4",
    ylab="Expression de LMO2", pch="+", las=1)
```

Paramètres basiques

xlab	nom de l'axe X
ylab	nom de l'axe Y
main	titre du graphique
sub	sous-titre du graphique
type	joindre les points ou non
xlim	limites de l'axe X
ylim	limites de l'axe Y

Paramètres avancés

pch	type de points (voir "?pch")
lty	type de traits
cex	taille des points
lwd	épaisseur des traits
col	couleur des points / traits
las	sens de lecture des axes

Ajouter des points : points

points(x, y, ...)

Similaire à plot()
Utilise un vecteur X et un vecteur Y
Dessine au dessus du graphique existant
Mêmes paramètres graphiques (pch, cex ...)

Pratique - Différencier GCB et ABC sur le graphique précédent

```
ABC <- subset(RTMLPA, Série=="training" & MLPA=="ABC")
GCB <- subset(RTMLPA, Série=="training" & MLPA=="GCB")

ABC_IRF4 <- subset(ABC, select="IRF4", drop=TRUE)
GCB_IRF4 <- subset(GCB, select="IRF4", drop=TRUE)
ABC_LMO2 <- subset(ABC, select="LMO2", drop=TRUE)
GCB_LMO2 <- subset(GCB, select="LMO2", drop=TRUE)

plot(x=ABC_IRF4, y=ABC_LMO2, pch=17)
points(x=GCB_IRF4, y=GCB_LMO2, pch=6)
```

Attention aux bornes des axes !
Si xlim ou ylim manquent, ils sont estimés
Estimés avec les X et Y de plot(), pas de points()

```
plot(x=ABC_IRF4, y=ABC_LMO2, pch=17, xlim=c(0,3), ylim=c(0,3))
points(x=GCB_IRF4, y=GCB_LMO2, pch=6, xlim=c(0,3), ylim=c(0,3))
```

Ajouter une légende : legend

legend(x, pch, legend, ...)

Position de la légende

En définissant « x » et « y »
En utilisant un mot clé : « bottomleft », « topright » ...

Contenu de la légende

Lister les points et leur signification

```
legend(x="topleft", pch=c(17,6), legend=c("ABC","GCB"))
```

Cosmétique

« inset=0.01 » ajoute un espace de 1% avec le bord du graphique
« bty="n" » supprime le cadre de la légende
« bg="grey" » ajoute une couleur de fond

Définir une couleur dans R

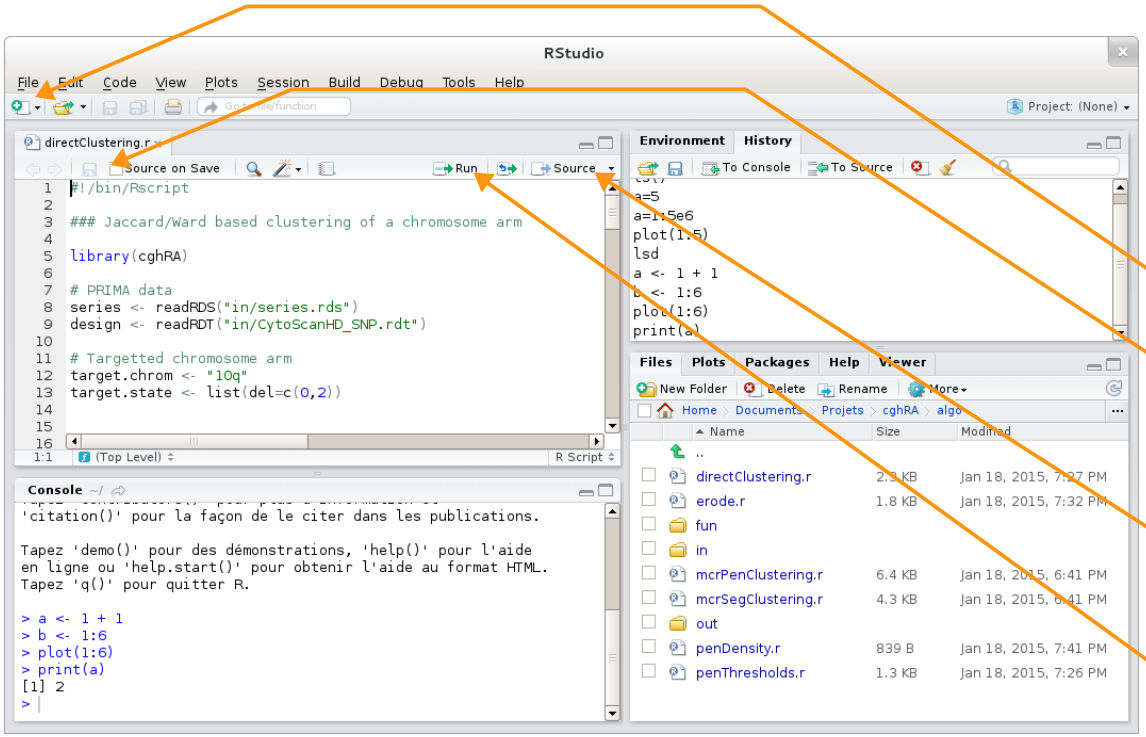
Par son nom : "orange", "green" ...

```
colors()
```

Par son numéro : 1 à 8

```
palette()
```

Par son code hexadécimal : "#FFCC00" ...



Créer un nouveau script

1. Appuyer sur « Nouveau »
2. Sélectionner « R script »
3. Appuyer sur « Enregistrer »
4. Choisir un nom

Exécuter le script

Avec ou sans l'affichage du code

Exécuter une partie du script

Reproductibilité de l'analyse

Un script complet permet de relancer l'analyse entière

Permet aussi d'en modifier un élément sans tout recommencer

Commentez vos scripts !

Toute ligne commençant par « # » n'est jamais exécutée

Permet d'expliquer à quoi sert une ligne ou un groupe de lignes

```
legend(x="topleft", pch=c(17,6), legend=c("ABC","GCB"))
```

Pratique - Rassembler les lignes générant le graphique précédent dans un script

Pratique - Rassembler les lignes générant le graphique précédent dans un script

```
# Lecture du tableau
RTMLPA <- read.csv("~/Documents/Projets/Formation R/RTMLPA.csv", sep=";", dec=",", quote="", stringsAsFactors=FALSE)

# Tableaux limités aux GCB et ABC
ABC <- subset(RTMLPA, Série=="training" & MLPA=="ABC")
GCB <- subset(RTMLPA, Série=="training" & MLPA=="GCB")

# Vecteurs d'expression de LMO2 et IRF4
ABC_IRF4 <- subset(ABC, select="IRF4", drop=TRUE)
GCB_IRF4 <- subset(GCB, select="IRF4", drop=TRUE)
ABC_LMO2 <- subset(ABC, select="LMO2", drop=TRUE)
GCB_LMO2 <- subset(GCB, select="LMO2", drop=TRUE)

# Création d'un graphique pour les ABC
plot(x=ABC_IRF4, y=ABC_LMO2, pch=17, xlim=c(0,3), ylim=c(0,3))

# Ajout des GCB
points(x=GCB_IRF4, y=GCB_LMO2, pch=6, xlim=c(0,3), ylim=c(0,3))

# Ajout de la légende
legend(x="topleft", pch=c(17,6), legend=c("ABC","GCB"), bty="n")
```

En bonus : coloration syntaxique

Valeurs « texte » en vert

Valeurs numériques ou logiques (TRUE, FALSE) en bleu

Mais aussi

Surlignage de la parenthèse correspondante

Numérotation des lignes

...

« Box » plots : boxplot

boxplot(valeurs~groupes, ...)

« valeurs » est un vecteur numérique
« groupes » est un vecteur textuel

Pratique - Expression de LMO2 en fonction du sous-type MLPA

```
LMO2 <- subset(RTMLPA, select="LMO2", drop=TRUE)
MLPA <- subset(RTMLPA, select="MLPA", drop=TRUE)
boxplot(LMO2~MLPA)
```

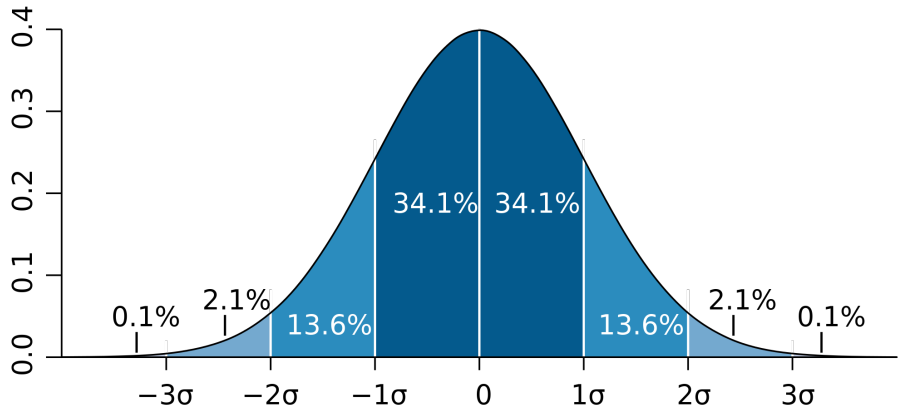
Paramètres graphiques

Gère la plupart des paramètres graphiques gérés par plot()
« varwidth=TRUE » pour que la largeur des boîtes dépende de n
« boxwex=0.5 » pour réduire la largeur des boîtes
« plot=FALSE » pour obtenir les valeurs précises utilisées par le graphique
« horizontal=TRUE » pour pivoter le graphique de 90°

Domaine d'application
Comparaison de moyennes entre deux groupes de valeurs
Valeurs indépendantes, groupes de tailles égales ou non
Non-paramétrique, insensible à l'échelle

Test d'hypothèse
Définit une « hypothèse nulle » (H0) correspondant à l'absence de différence
Définit ce qu'on observerait dans les données si elle était respectée
Calcule la probabilité qu'elle soit vraie (p) en comparant les données à la théorie
La rejette si p < 5%

H0 : basée sur les rangs
Si on ordonne les valeurs des deux groupes
Les moyennes des rangs devraient être comparables entre les groupes
⟶ calcul d'une statistique U
⟶ U suit une loi normale sous H0
⟶ calcul de la probabilité d'observer cette valeur dans cette loi normale



Test de Mann-Whitney : wilcox.test

wilcox.test(x, y, ...)
wilcox.test(valeurs~groupes, ...)

Deux syntaxes possibles
En définissant deux vecteurs numériques (comme pour « plot »)
En donnant un vecteur numérique et un vecteur textuel (comme pour « boxplot »)

Pratique - Expression de LMO2 en fonction du sous-type MLPA

```
LMO2 <- subset(RTMLPA, MLPA != "other", select="LMO2", drop=TRUE)
MLPA <- subset(RTMLPA, MLPA != "other", select="MLPA", drop=TRUE)
wilcox.test(LMO2~MLPA)
```

Ajouter la p-valeur sur un graphique

Récupérer la p-valeur dans une variable

```
p <- wilcox.test(LMO2~MLPA)$p.value
```

Ajouter un texte explicatif avec paste()

```
titre <- paste("Mann-Whitney's p =", p)
```

Passage dans le titre

```
boxplot(LMO2~MLPA, main=titre)
```

Compare deux critères attribuées aux mêmes échantillons
Permet de visualiser des associations

	Critère A	
Critère B	A1	A2
B1	n(A1 et B1)	n(A2 et B1)
B2	n(A1 et B2)	n(A2 et B2)

Tableaux de contingences : table

table(critère1, critère2, ...)

Pratique - Comparer IHC et MLPA

```
IHC <- subset(RTMLPA, select="IHC", drop=TRUE)
MLPA <- subset(RTMLPA, select="MLPA", drop=TRUE)
table(IHC, MLPA)
```

Extensible à n dimensions

```
serie <- subset(RTMLPA, select="Série", drop=TRUE)
table(IHC, MLPA, serie)
```

Domaine d'application
Recherche de lien entre deux conditions binaires
Puissant même sur de petits effectifs (cellules < 10)
Application directe à une table de contingence 2x2

Test d'hypothèse
H0 : il n'existe pas de lien entre les deux variables
Calcul direct de probabilité, sans passer par une loi

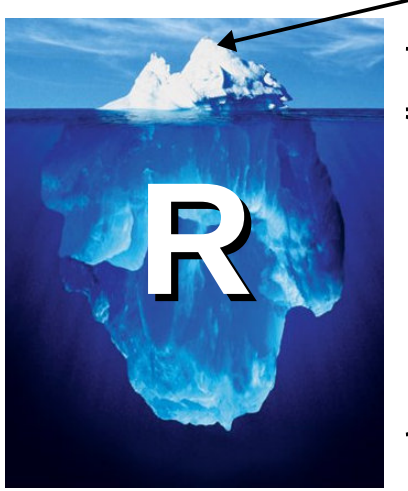
Mise en œuvre : fisher.test

fisher.test(table_de_contingence)

Pratique - Comparer GCB / ABC entre IHC et MLPA

```
data <- subset(RTMLPA, IHC!="" & MLPA!="other")
IHC <- subset(data, select="IHC", drop=TRUE)
MLPA <- subset(data, select="MLPA", drop=TRUE)
tab <- table(IHC, MLPA)
fisher.test(tab)
```

R ne « s'apprend » pas, il se « comprend »



Après cette courte introduction

Après les 22h de formation

Ce qu'il vous restera à découvrir

Comme une langue étrangère :
- Concentrez vous sur sa syntaxe, son fonctionnement
- Le vocabulaire s'enrichit à l'usage
- De nombreuses règles de grammaire ne servent quasiment jamais
- De nombreuses façons d'arriver au même résultat

Généralisez
La plupart des tests fonctionnent de la même façon

Devenez autonome
Utiliser l'aide intégrée de R pour comprendre / découvrir des fonctions
Utiliser Google pour résoudre un problème
⟶ quel que soit votre problème, quelqu'un l'a eut avant vous