R: présentation éclair

Ensimag 1A
C. Dutang
2024

1 Logiciel R: le langage, l'environment de travail

1.1 Historique

- R se base sur le langage S dévéveloppé dans les années 70 chez Bell laboratories
- Malgré une version commerciale de S appelée S+ dès les années 80, deux néo-zélandais de l'université d'Auckland ont décidé de créer une version gratuite.
- Ross Ihaka et Robert Gentleman créent R en 1996 en version GPL (http://www.gnu.org/licenses/gpl-3.0.fr.html (http://www.gnu.org/licenses/gpl-3.0.fr.html)) sur une idée de Martin Maechler (ETH Zurich) : voir https://cran.r-project.org/doc/html/interface98-paper/paper.html (https://cran.r-project.org/doc/html/interface98-paper/paper.html).
- R est complètement gratuit et ouvert sous licence GPL disponible à http://www.r-project.org (http://www.r-project.org)
- R est mis régulièrement à jour : R 1.0.0 en 2000, R 2.0.0 en 2004, R 3.0.0 en 2013, R 4.0.0 en 2020, actuellement R 4.3.2.
- Les personnes réalisant les améliorations, la maintenance et le développement sont regroupés dans l'équipe R core team : https://www.r-project.org/contributors.html (https://www.r-project.org/contributors.html).
- 800 000 lignes de code composent les sources de R : 45% de C, 20% de R, 17% de Fortran et 18% autre (shell script…)
- R utilise notamment la bibliothèque Lapack (https://www.netlib.org/lapack/faq.html#_what_and_where_are_the_lapack_vendors_implementations) comme beaucoup d'autres langages

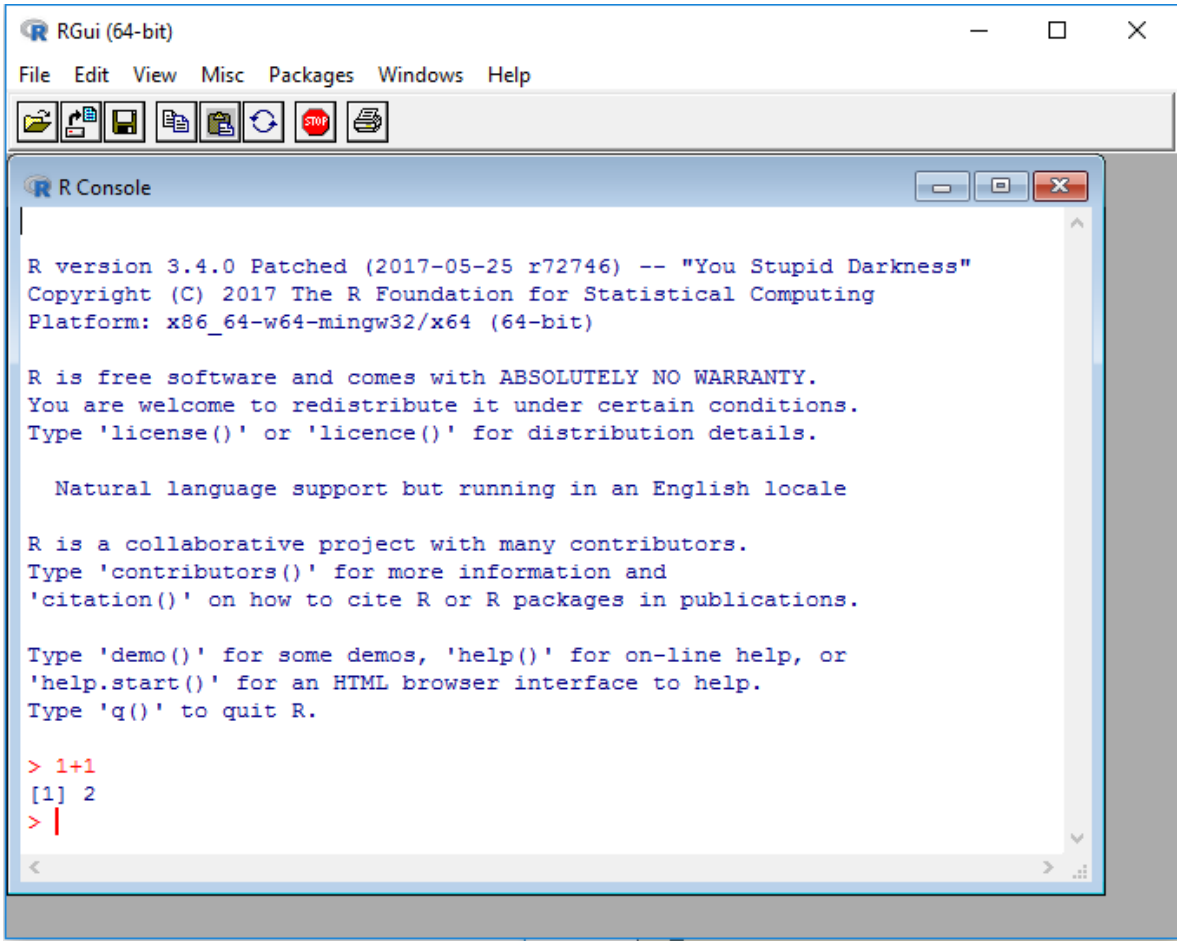
1.2 Niveau d'utilisation par les universités et les entreprises

- Donateurs publics sont des universités, des laboratoires, … tels que Agrocampus Ouest, Université de Nantes voir https://www.r-project.org/foundation/donors.html (https://www.r-project.org/foundation/donors.html).
- Les donateurs privés sont petits ou grands tels que AT&T Research, Greenwich Statistics…

- Chaque année, une conférence useR est organisée regroupant plus de 1200 participants voir http://www.user2019.fr/ (http://www.user2019.fr/)
- Les sponsors traditionnels sont Google, HP, Oracle, DataRobot,…
- R est massivement utilisée par les organismes publics et les entreprises privées, pour preuve les conférences suivantes
    - R in Insurance : Milliman, Deloitte, AXA, SCOR,… https://rininsurance17.sciencesconf.org/ (https://rininsurance17.sciencesconf.org/)
    - Insurance data science conference : https://insurancedatascience.org/ (https://insurancedatascience.org/)
    - R in Finance : Avant, Interactive Brokers,…
    - une liste non exhaustive : http://www.revolutionanalytics.com/companies-using-r (http://www.revolutionanalytics.com/companies-using-r)

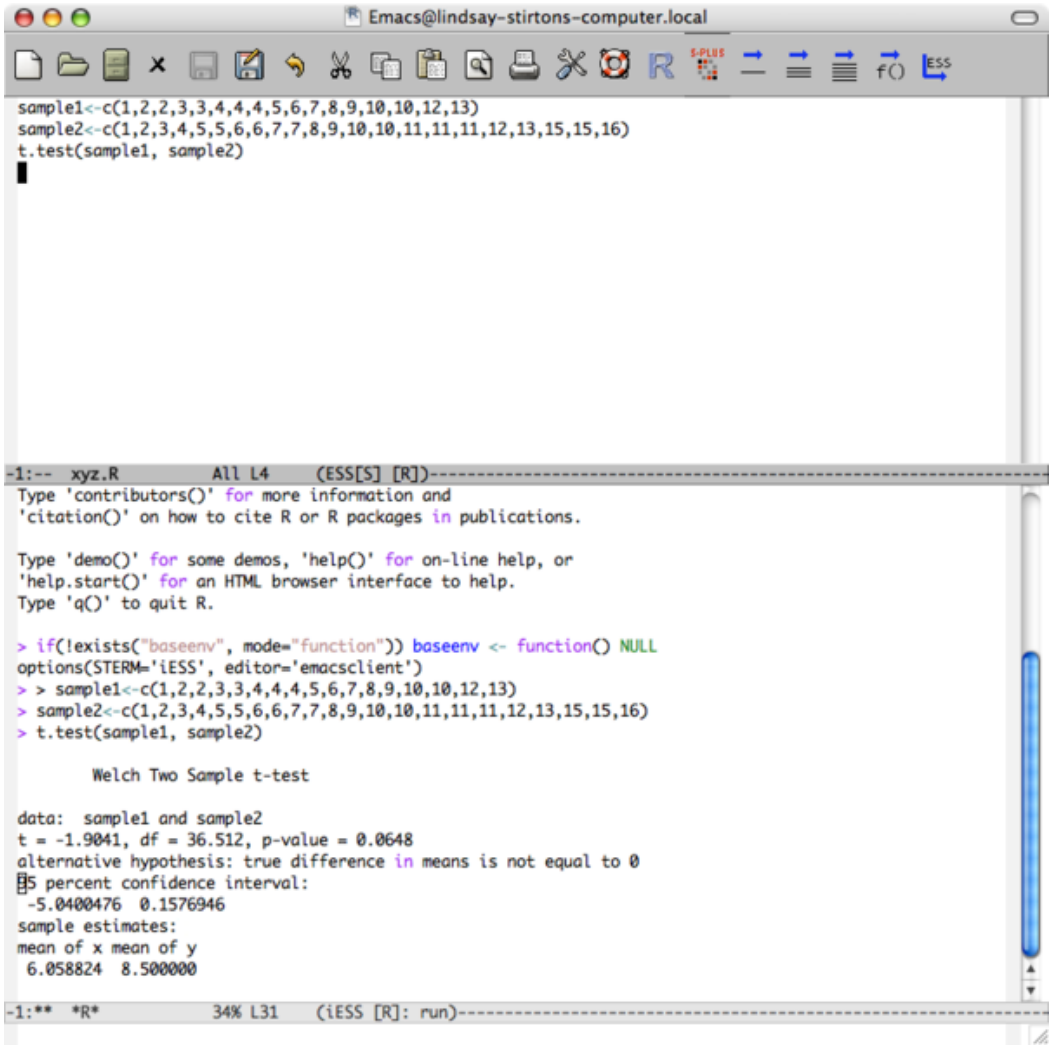
1.3 Installation, utilisation

- Il est téléchargeable à http://www.r-project.org/CRANmirrors (http://www.r-project.org/CRANmirrors)
- R est disponible sur windows, mac os et linux. La console ressemble

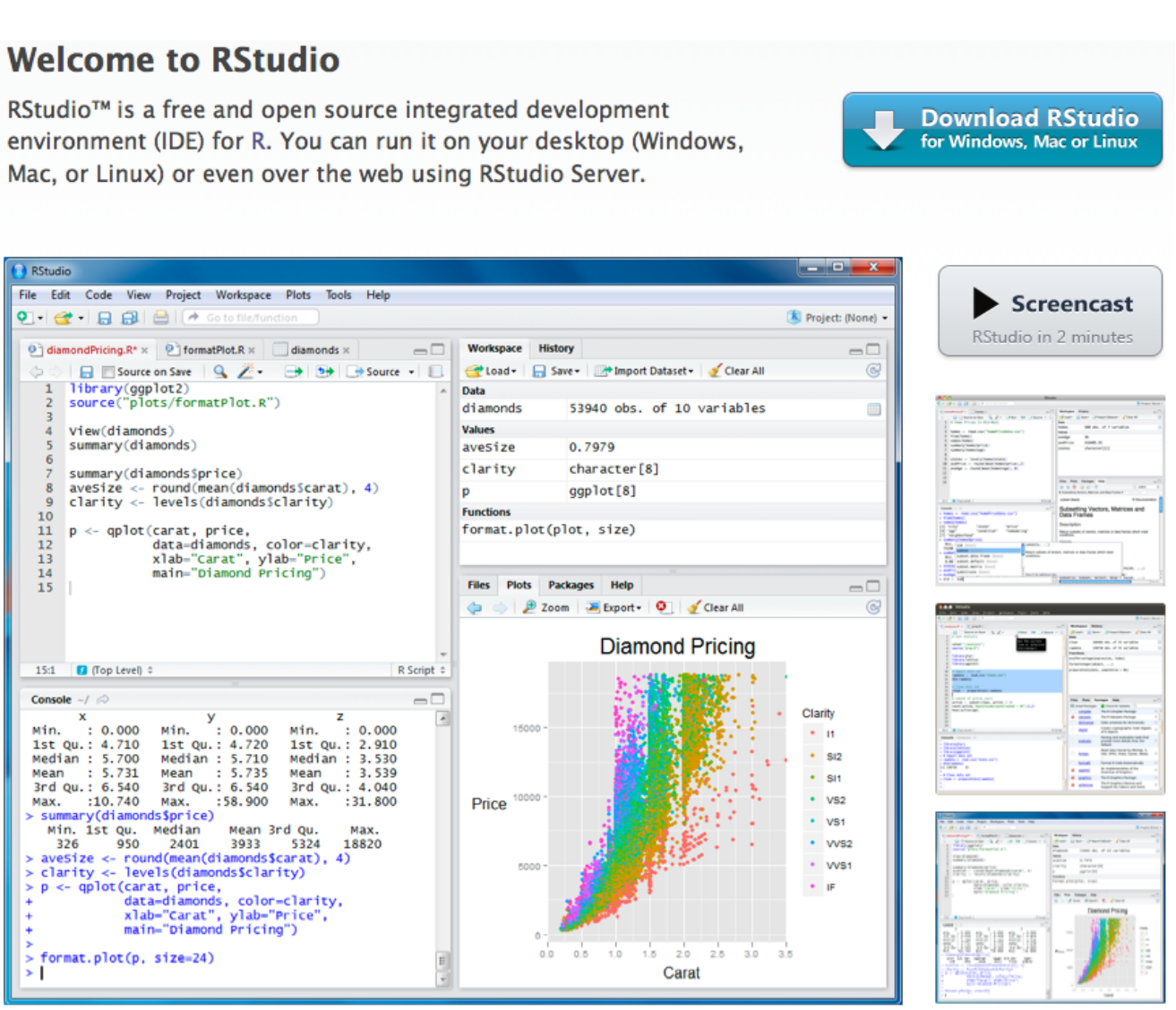


1.4 Editeurs possibles

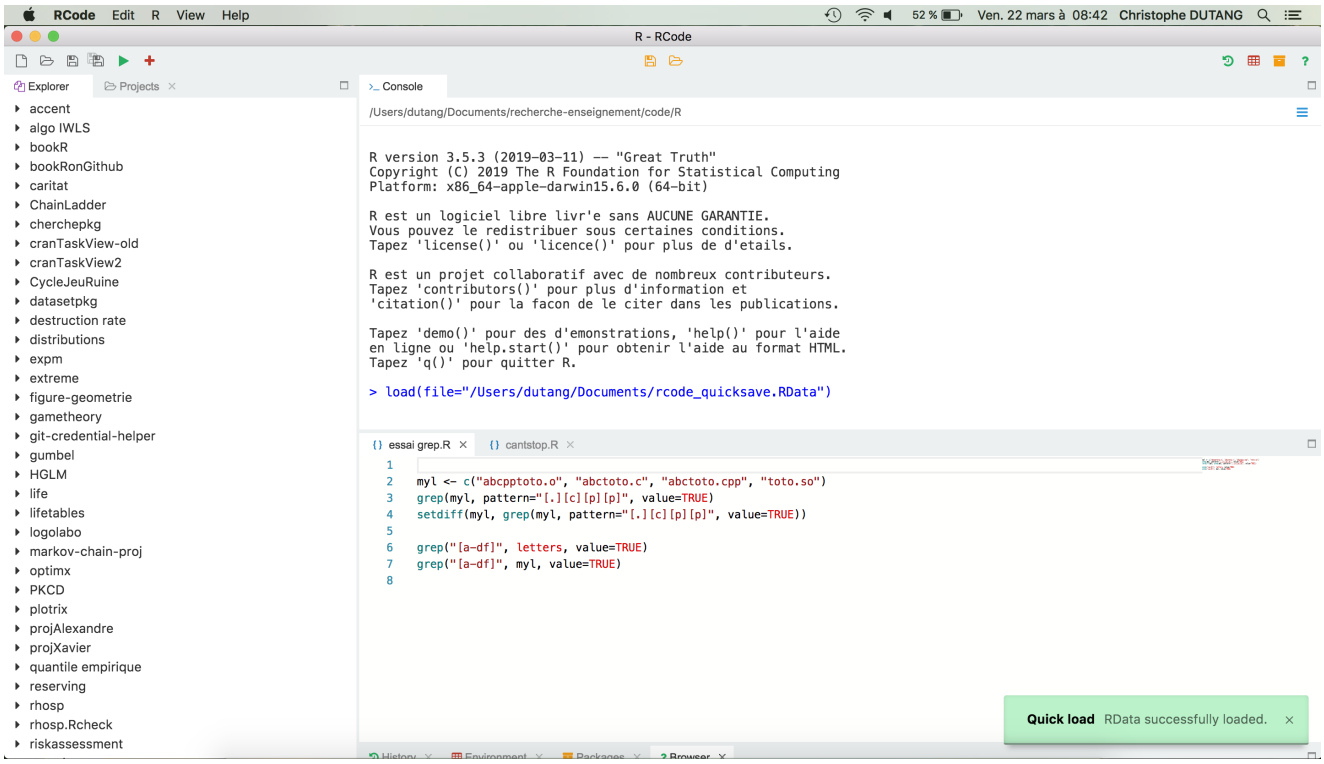
- Editeur de base fourni avec R : R GUI
- Sur toutes les plateformes, Emacs



- Sur toutes les plateformes, Rstudio http://rstudio.org/ (http://rstudio.org/)



- Sur toutes les plateformes, Rcode



2 Architecture de R

- R est un ensemble de fonctionnalités regroupées en module appelés paquetage ou package en anglais.

- Il y a 3 types de packages: standard, recommandés et externes, cf. https://cran.r-project.org/doc/FAQ/R-FAQ.html (https://cran.r-project.org/doc/FAQ/R-FAQ.html).

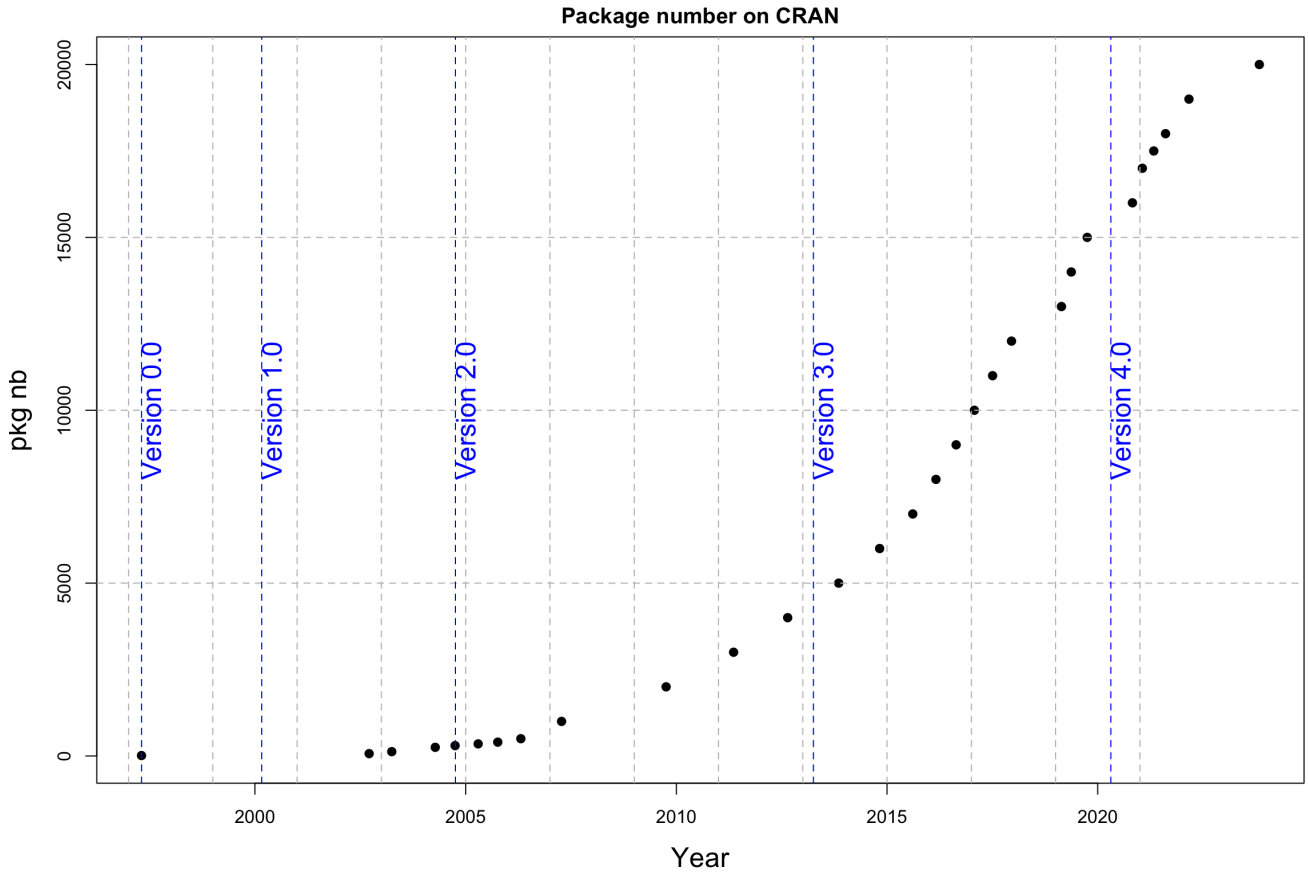
2.1 Packages livrés avec le logiciel R

- Les packages standard (livrés et chargés en mémoire) sont
  - base : Base R functions and datasets (before R 2.0.0).
  - compiler : R byte code compiler (added in R 2.13.0).
  - datasets : Base R datasets (added in R 2.0.0).
  - grDevices : Graphics devices for base and grid graphics (added in R 2.0.0).
  - graphics : R functions for base graphics.
  - grid : A rewrite of the graphics layout capabilities, plus some support for interaction.
  - methods : Formally defined methods and classes for R objects, plus other programming tools.
  - parallel : Support for parallel computation, including by forking and by sockets, and random-number generation (added in R 2.14.0).
  - splines : Regression spline functions and classes.
  - stats : R statistical functions.
  - stats4 : Statistical functions using S4 classes.
  - tcltk : Interface and language bindings to Tcl/Tk GUI elements.
  - tools : Tools for package development and administration.
  - utils : R utility functions.
- Les packages recommendés (livrés mais pas chargés en mémoire) sont
  - KernSmooth : Functions for kernel smoothing (and density estimation).
  - MASS : Functions and datasets from the main package of Venables and Ripley.
  - Matrix: A Matrix package. (Recommended for R 2.9.0 or later.)
  - boot : Functions and datasets for bootstrapping.
  - class : Functions for classification (k-nearest neighbor and LVQ).
  - cluster : Functions for cluster analysis.
  - codetools : Code analysis tools. (Recommended for R 2.5.0 or later.)
  - foreign : Functions for reading and writing data stored by statistical software like Minitab, S, SAS, SPSS, Stata, Systat, etc.
  - lattice : Lattice graphics, an implementation of Trellis Graphics functions.
  - mgcv : Routines for GAMs and other generalized ridge regression problems with multiple smoothing parameter selection by GCV or UBRE.
  - nlme : Fit and compare Gaussian linear and nonlinear mixed-effects models.
  - nnet : Software for single hidden layer perceptrons (“feed-forward neural networks”), and for multinomial log-linear models.
  - rpart : Recursive PARTitioning and regression trees.
  - spatial : Functions for kriging and point pattern analysis.
  - survival : Functions for survival analysis, including penalized likelihood.

2.2 Les packages externes

- Plus les 20 294 autres packages externes. Leur nombre évolue avec plus ou moins de régularité selon les critères de dépôt sur CRAN
  - 2023-11-02 20000 pkgs (+1.6/day over 610 days)
  - 2022-03-02 19000 pkgs (+4.9/day over 203 days)
  - 2021-08-11 18000 pkgs (+5/day over 201 days)
  - 2021-01-22 17000 pkgs (+11.6/day over 86 days)
  - 2020-10-28 16000 pkgs (+2.6/day over 392 days)
  - 2019-10-02 15000 pkgs (+7.2/day over 138 days)
  - 2019-05-17 14000 pkgs (+11.8/day over 85 days)
  - 2019-02-21 13000 pkgs (+2.3/day over 433 days)
  - 2017-12-15 12000 pkgs (+6.1/day over 164 days) 6910 mnts (+3.2/day)
  - 2017-07-04 11000 pkgs (+6.3/day over 158 days) 6377 mnts (+3.3/day)
  - 2017-01-27 10000 pkgs (+6.3/day over 158 days) 5845 mnts (+3.5/day)
  - 2016-08-22 9000 pkgs (+5.7/day over 175 days) 5289 mnts (+5.8/day)
  - 2016-02-29 8000 pkgs (+5/day over 201 days) 4279 mnts (+0.7/day)
  - 2015-08-12 7000 pkgs (+3.5/day over 287 days) 4130 mnts (+2.4/day)
  - 2014-10-29 6000 pkgs (+2.8/day over 355 days) 3444 mnts (+1.6/day)

- 2013-11-08 5000 pkgs (+2.3/day over 442 days) 2900 mnts (+1.2/day)
- 2012-08-23 4000 pkgs (+2.1/day over 469 days) 2350 mnts
- 2011-05-12 3000 pkgs (+1.7/day over 585 days)
- 2009-10-04 2000 pkgs (+1.1/day over 906 days)
- 2007-04-12 1000 pkgs
- 2004-10-01 500 pkgs
- 2003-04-01 250 pkgs
- 2002-09-17 68 pkgs
- 1997-04-23 12 pkgs

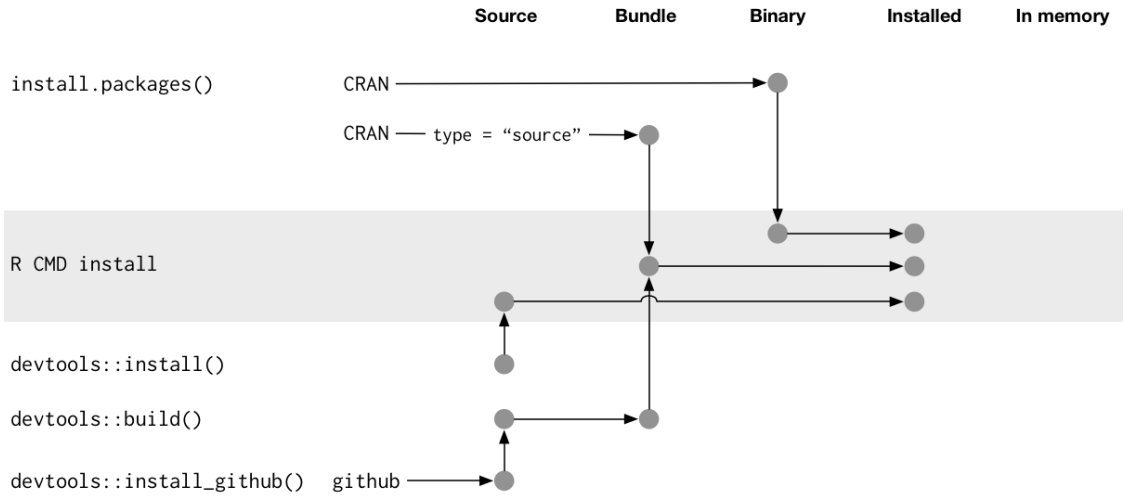


- Le plus simple pour trouver un package est de consulter les 44 pages thématiques, voir https://cran.r-project.org/web/views/ (https://cran.r-project.org/web/views/)

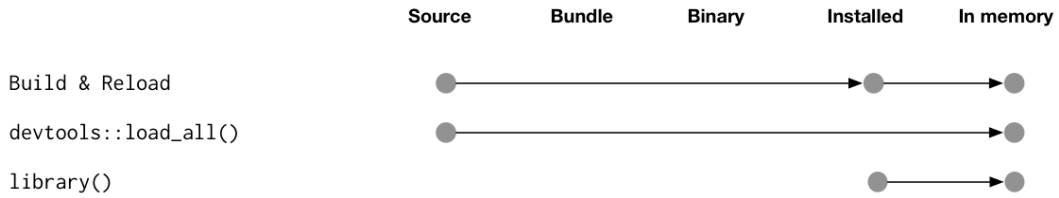
Bayesian	Bayesian Inference
ChemPhys	Chemometrics and Computational Physics
ClinicalTrials	Clinical Trial Design, Monitoring, and Analysis
Cluster	Cluster Analysis & Finite Mixture Models
DifferentialEquations	Differential Equations
Distributions	Probability Distributions
Econometrics	Computational Econometrics
Environmetrics	Analysis of Ecological and Environmental Data
ExperimentalDesign	Design of Experiments (DoE) & Analysis of Experimental Data
Finance	Empirical Finance
Genetics	Statistical Genetics
Graphics	Graphic Displays & Dynamic Graphics & Graphic Devices & Visualization
HighPerformanceComputing	High-Performance and Parallel Computing with R
MachineLearning	Machine Learning & Statistical Learning
MedicalImaging	Medical Image Analysis
Multivariate	Multivariate Statistics
NaturalLanguageProcessing	Natural Language Processing
OfficialStatistics	Official Statistics & Survey Methodology
Optimization	Optimization and Mathematical Programming
Pharmacokinetics	Analysis of Pharmacokinetic Data
Phylogenetics	Phylogenetics, Especially Comparative Methods
Psychometrics	Psychometric Models and Methods
ReproducibleResearch	Reproducible Research
Robust	Robust Statistical Methods
SocialSciences	Statistics for the Social Sciences
Spatial	Analysis of Spatial Data
Survival	Survival Analysis
TimeSeries	Time Series Analysis
gR	gRaphical Models in R

2.3 Installation / utilisation d'un paquetage

- Processus d'installation d'un paquetage



- Processus de chargement d'un paquetage



2.4 Où trouver de l'aide?

- Documents de références http://cran.r-project.org/manuals.html (http://cran.r-project.org/manuals.html)
  - Introduction à R http://cran.r-project.org/doc/manuals/r-release/R-intro.html (http://cran.r-project.org/doc/manuals/r-release/R-intro.html)
  - Import-export http://cran.r-project.org/doc/manuals/r-release/R-data.html (http://cran.r-project.org/doc/manuals/r-release/R-data.html)
- Dans R,
  - Sur une fonction particulière à l'aide de la fonction `help()` ou `?`
  - Par mot clé à l'aide la fonction `help.search()`
  - Utiliser `RSiteSearch()` à http://search.r-project.org/ (http://search.r-project.org/)
  - Package `RWSearch` , voir l'aide https://cran.r-project.org/web/packages/RWsearch/vignettes/RWsearch-1-Introduction.html (https://cran.r-project.org/web/packages/RWsearch/vignettes/RWsearch-1-Introduction.html)
  - Package `sos` , voir l'aide https://cran.r-project.org/web/packages/sos/vignettes/sos.pdf (https://cran.r-project.org/web/packages/sos/vignettes/sos.pdf)
- Livres http://cran.r-project.org/other-docs.html (http://cran.r-project.org/other-docs.html) : les deux plus utiles
  - Introduction à la programmation en R, Vincent Goulet
  - R pour les débutants, Emmanuel Paradis
- e-books gratuits
  - https://www.bigbookofr.com/ (https://www.bigbookofr.com/)
  - https://csgillespie.github.io/efficientR/ (https://csgillespie.github.io/efficientR/)
  - et plein d'autres sur https://bookdown.org/ (https://bookdown.org/)
- Journaux scientifiques (gratuit)
  - JSS : http://www.jstatsoft.org/ (http://www.jstatsoft.org/)
  - R journal : http://journal.r-project.org/ (http://journal.r-project.org/)

3 Opérations et types de base

3.1 Objet et types

- Tout dans le langage R est un objet. Le mode de l'objet définit son utilisation: `numeric` , `character` , `logical` , `complex` , `function` , `list` , `Date` (principalement).
- Plusieurs versions de programmation orienté objet cohabite
  - le système `S3` propose sur les types de base des fonctions génériques. Un objet est une liste possédant des champs particuliers.
  - le système `S4` propose une formalisation plus propre des objets.
  - le système `R6` propose une formalisation semblable aux autres langages.
- L'affectation est réalisé par `<-` ou `=` (déconseillé), voir `?"<-"` .

Description	Mode	Classe	Longueur	Opérations	Extraction	Val. spéciales
réels	`"numeric"`	`"numeric"` , `"matrix"` , `"array"`	nb éléments	`+` , `-` , `*` , `/` , `^` , `%%` , `%/%` , `%*%` , `==` , `!=` , `>` , `<` , `<=` , `>=` , `%in%`	`[]` , `[,]` , `[,,,]`	`Inf` , `NaN`
complexes	`"complex"`	`"complex"` , `"matrix"` , `"array"`	nb éléments	`+` , `-` , `*` , `/` , `^` , `%*%` , `==` , `!=` , `>` , `<` , `<=` , `>=` , `%in%`	`[]` , `[,]` , `[,,,]`	`Inf` , `NaN`

Description	Mode	Classe	Longueur	Opérations	Extraction	Val. spéciales
booléens	`"logical"`	`"logical"`, `"matrix"`, `"array"`	nb éléments	`&`, `|`, `!`, `&&`, `||`, `%in%`	`[]`, `[,]`, `[,,,]`	`NA`
chaines caractères	`"character"`	`"character"`, `"matrix"`, `"array"`	nb éléments	`==`, `!=`, `>`, `<`, `<=`, `>=`, `%in%`	`[]`, `[,]`, `[,,,]`	
types énumérés ou ordonnés	`"numeric"`	`"factor"`, `"ordered"`	nb éléments	`==`, `!=`, `%in%`	`[]`	
liste ou liste de données	`"list"`	`"list"`, `"data.frame"`	nb éléments, nb colonnes		`[[]]`, `[]`, `$`	
date	`"numeric"`	`"Date"`	nb éléments		`[]`	
fonction	`"function"`	`"function"`	1			
vide `NULL`	`"NULL"`	`"NULL"`	0			

3.2 Fonctions centrales

- booléen, voir `methods(class="logical")`
  - `all()` calcule le **et** logique sur toutes les composantes
  - `any()` calcule le **ou** logique sur toutes les composantes
- opération de base sur toutes les composantes, voir `methods(class="numeric")`
  - `sum()` calcule la somme
  - `prod()` calcule le produit
  - `min()` calcule le min
  - `max()` calcule le max
- résume l'objet
  - `range()` calcule la portée
  - `summary()` résume l'objet
  - `str()` affiche la structure
- manipule l'objet
  - `head()` renvoie les premiers éléments
  - `tail()` renvoie les derniers éléments
  - `sort()` trie
- calcule des statistiques
  - `quantile` calcule les quantiles
  - `ecdf` calcule la fonction de répartition empirique
  - `table()` calcule un tableau croisé
- suite `apply`
  - `apply` applique sur une fonction sur un tableau multidimensionnel
  - `sapply`, `lapply` applique sur une fonction sur une liste ou un vecteur
  - `tapply` applique une fonction après un regroupement par catégorie

3.3 Fonctions mathématiques

- nombre: `abs()`, `sign()`, `sqrt()`, `floor()`, `ceiling()`, `trunc()`, `round()`, `signif()`
- fonctions usuelles: `exp()`, `log()`, `expm1()`, `log1p()`, `cos()`, `sin()`, `tan()`, `cospi()`, `sinpi()`, `tanpi()`, `acos()`, `asin()`, `atan()`
- fonctions hyperboliques: `cosh()`, `sinh()`, `tanh()`, `acosh()`, `asinh()`, `atanh()`
- fonctions spéciales: `lgamma()`, `gamma()`, `digamma()`, `trigamma()`
- fonctions cumulées: `cumsum()`, `cumprod()`, `cummax()`, `cummin()`
- fonctions de complexe: `Arg()`, `Cong()`, `Im()`, `Mod()`, `Re()`

3.4 Création de vecteurs

Mode	par mode	par concaténation	par séquence
`"numeric"`	`numeric(10)`	`c(1, 345, 3.14, pi)`	`1:10`
`"complex"`	`complex(2)`	`c(1+2i, 1-1i, 3-3i)`	`1:10+1i`
`"logical"`	`logical(3)`	`c(TRUE, FALSE, FALSE)`	`1:10 > 5`
`"character"`	`character(7)`	`c("abc", "def", "ghi", "jkl")`	

- l'extraction se fait par `[]`.

3.5 Création de matrices

Mode	par mode	par concaténation des colonnes	par concaténation des lignes
`"numeric"`	`matrix(numeric(10))`	`cbind(1, 345:347, 3.14, pi)`	`rbind(1, 345:347, 3.14, pi)`
`"complex"`	`matrix(complex(4))`	`cbind(1+2i, 1-1i, 1:3-3i)`	`rbind(1+2i, 1-1i, 1:3-3i)`
`"logical"`	`matrix(logical(4))`	`cbind(TRUE, FALSE, FALSE)`	`rbind(TRUE, FALSE, FALSE)`
`"character"`	`matrix(character(4))`	`cbind(c("a","b","c"), c("d","e","f"))`	`rbind(c("a","b","c"), c("d","e","f"))`

- l'extraction se fait par `[,]`.

3.6 Fonctions graphiques

- le paquet `graphics` propose les graphiques les plus utiles et les plus simples
  - `plot()` est la fonction générique de base appliquable aux objets simples, `?plot.default`, comme au formule `plot.formula`
  - `points()` et `lines` complète un graphique initiée par `plot()``
  - `curve()` permet de gérer facilement les fonctions.
  - `persp()`, `contour()` et `image()` affichent les nuages de points en 3D ou projetés en 2D.
- le paquet externe `ggplot2` propose une philosophie pour réaliser des graphiques.

3.7 Variables et valeurs réservées

- La création de variables se fait par l'affectation `<-`.
- Les noms des objets doivent commencer par une lettre et utiliser des caractères alpha-numériques. Les noms dans `R` sont sensibles aux majuscules et minuscules.
- Les noms suivants sont interdits (voir `?Reserved`):
  - `break`, `else`, `for`, `function`, `if`, `in`, `next`, `repeat`, `return`, `while`,
  - `TRUE`, `FALSE`,
  - `Inf`, `NA`, `NaN`, `NULL`,
  - `NA_integer_`, `NA_real_`, `NA_complex_`, `NA_character_`,
  - `...`, `..1`, `..2`, etc.
- Les noms suivants sont *FORTEMENT* déconseillés:
  - `c`, `q`, `t`, `C`, `D`, `I`, `diff`, `length`, `mean`, `pi`, `range`, `var`

4 Structures de contrôles et boucles

4.1 Structures de contrôle - Exécution conditionnelle

- Exécution conditionnelle est rendue possible par les structures `if`, `ifelse` ou `switch`.
- Une structure `if` est de la forme
  `if(` **condition booléenne** `)`
  {

Suites d'instructions
`}else if(` **condition booléenne** `)`
`{`
Suites d'instructions
`}else`
`{`
Suites d'instructions
`}`

- Les deuxième et troisième parties `else` sont optionnelles.
- Les accolades `{}` sont optionnelles s'il y a une seule instruction.
- On peut générer une série condition à l'aide de l'opérateur `switch` .
  `switch(` **expression retournant un nombre ou une chaine**, *alternative 1*, *alternative 2*, *alternative 3*, … `)`
- La structure `ifelse()` simplifie l'appel et vectorise la structure de contrôle.

4.2 Structures de contrôle - boucle itérative sans condition

- La boucle `for` permet d'itérer parmi une séquence.
- Une structure `for` est de la forme
  `for(` **variable** `in` **sequence** `)`
  `{`
  Suites d'instructions
  `}`

4.3 Structures de contrôle - boucle conditionnelle

- La boucle `while` permet d'itérer tant qu'une condition n'est pas vérifiée.
- Une structure `while` est de la forme
  `while(` **condition booléenne** `)`
  `{`
  Suites d'instructions
  `}`
- Une structure `repeat` - `break` est de la forme
  `repeat`
  `{`
  Suites d'instructions
  `if` (**condition booléenne**)
  `break`
  `}`

5 Langage vectoriel

- En `R` , tout est un vecteur. Contrairement à certains autres langages de programmation, il n'y a pas de notion de scalaire : un scalaire est simplement un vecteur de longueur 1.
- En conséquence, les boucles sont à proscrire autant que faire se peut.
- On préfère les fonctions `apply` , `sapply` , …

5.1 Recyclage des paramètres

- Les opérations élémentaires recyclent les arguments si les longueurs des variables ne correspondent pas par rapport à la longueur la plus grande.
- Exemple naturel bien pensé

$$\begin{pmatrix}1\\2\\3\\4\end{pmatrix} + \begin{pmatrix}1\\2\end{pmatrix} \Leftrightarrow \begin{pmatrix}1\\2\\3\\4\end{pmatrix} + \begin{pmatrix}1\\2\\1\\2\end{pmatrix} = \begin{pmatrix}2\\4\\4\\6\end{pmatrix}$$

```
1:4 + 1:2
```

```
## [1] 2 4 4 6
```

- Exemple mal pensé, noter le `warning` levé par la dernière opération

$$\begin{pmatrix}1\\2\\3\\4\end{pmatrix} + \begin{pmatrix}1\\2\\3\end{pmatrix} \Leftrightarrow \begin{pmatrix}1\\2\\3\\4\end{pmatrix} + \begin{pmatrix}1\\2\\3\\1\end{pmatrix} = \begin{pmatrix}2\\4\\6\\5\end{pmatrix}$$

```
1:4 + 1:3
```

```
## Warning in 1:4 + 1:3: longer object length is not a multiple of shorter object
## length
```

```
## [1] 2 4 6 5
```

5.2 Indices négatifs

- La présence d'indices négatifs lors de l'extraction équivaut à enlever les éléments correspondants.
- Par exemple

```
(1:10)[-3]
```

```
## [1]  1  2  4  5  6  7  8  9 10
```

```
(1:10)[-(3:7)]
```

```
## [1]  1  2  8  9 10
```

```
try((1:10)[-1:1])
```

```
## Error in (1:10)[-1:1] : only 0's may be mixed with negative subscripts
```

6 Fonctions

6.1 Définition d'une fonction

- L'utilisateur a la possibilité de définir ses propres fonctions.
- Une fonction se définit de la manière suivante
  nom `<- function(` **arguments** `)`
  `{`
  Suites d'instructions
  `}`
- La valeur retournée par une fonction est soit la dernière valeur calculée soit celle retournée par `return`.
- Pour débugger les fonctions, il est parfois utile d'utiliser `traceback()` et `print()`.

6.2 Nommage des arguments

- Les arguments d'une fonction ont un nom précis sauf ceux utilisés dans `...`.
- Les arguments peuvent aussi avoir ou non une valeur par défaut.
- Dans un appel de fonction, les arguments n'ont pas besoin d'être nommés si l'ordre des arguments est respecté.

6.3 Variable globale, variable locale et portée

- Toutes les affections `<-` sont locales et temporaires. Il faut utiliser `<<-` ou `assign()` pour effectuer des affectations globales et permanentes.

7 Statistiques et probabilités usuelles

- Les observations sont notées $x_1, \ldots, x_n$ et sont les réalisations de variables aléatoires $X_1, \ldots, X_n$.
- L'hypothèse est que les variables $X_1, \ldots, X_n$ sont indépendemment et identiquement distribués (iid). Notons X la variable générique.

7.1 Statistiques graphiques

7.1.1 Variables discrètes

- La fréquence absolue de la valeur j est le nombre total n_j d'observations égales à j, c'est à dire $n_j = \sum_{i=1}^{n} 1_{x_i=j}$. La fréquence relative est n_j/n.

```
x <- sample(letters, 50, replace=TRUE)
table(x)
```

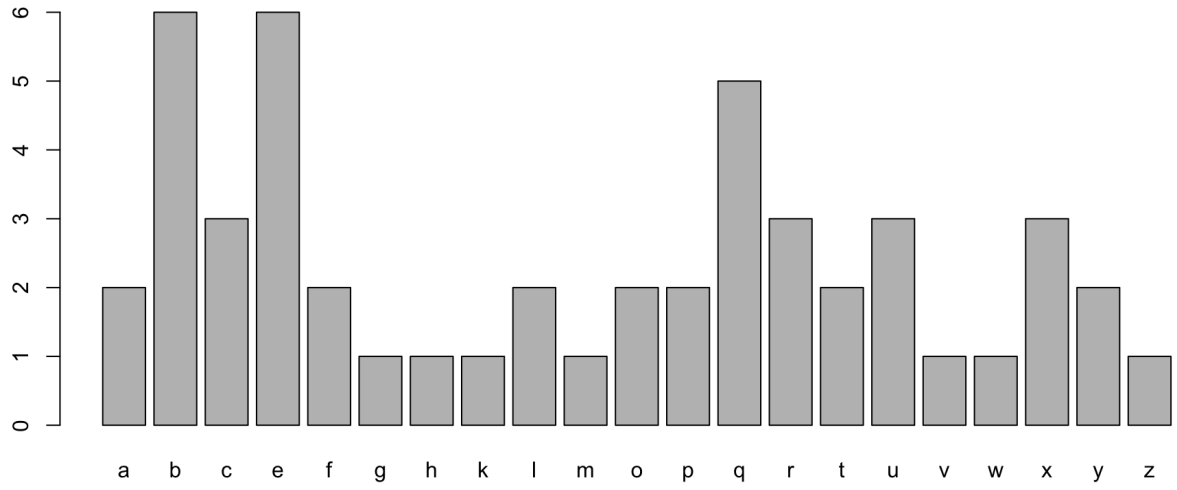
```
## x
## a b c e f g h k l m o p q r t u v w x y z
## 2 6 3 6 2 1 1 1 2 1 2 2 5 3 2 3 1 1 3 2 1
```

```
table(x)/length(x)
```

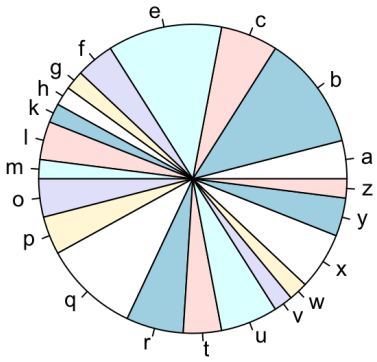
```
## x
##    a    b    c    e    f    g    h    k    l    m    o    p    q    r    t    u
## 0.04 0.12 0.06 0.12 0.04 0.02 0.02 0.02 0.04 0.02 0.04 0.04 0.10 0.06 0.04 0.06
##    v    w    x    y    z
## 0.02 0.02 0.06 0.04 0.02
```

- Deux graphiques classiques sont les diagrammes en baton et en cercle.

```
barplot(table(x))
```




```
pie(table(x))
```

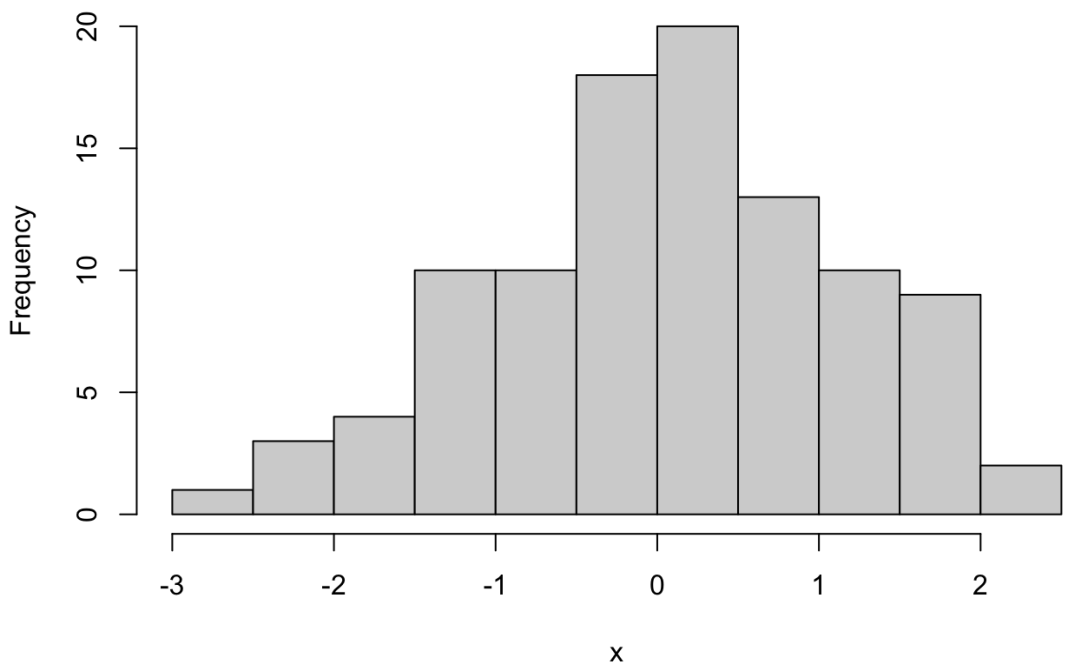


- Pour d'autres graphiques, voir le package `plotrix`

7.1.2 Variables continues

- L'histogramme est la figure constituée des rectangles dont les bases sont des classes (intervalle $]a_{j-1}, a_j]$) et dont les aires sont égales aux fréquences relatives de ces classes. Les classes sont telles que $a_0 < a_1 < \cdots < a_k$, $a_0 < \min_i x_i$ et $a_k > \max_i x_i$. Les hauteurs sont égales à $\frac{n_j}{nh_j}$ avec $h_j = a_j - a_{j-1}$.
  - l'histogramme à pas fixe suppose $a_j - a_{j-1} = h$, donc les hauteurs sont $\frac{n_j}{nh}$.
  - l'histogramme à même effectif suppose que $n_j = \lfloor n/k \rfloor$ est identique, donc les hauteurs sont $\frac{n_j}{nh_j}$.

```
x <- rnorm(100)
hist(x)
```



- La fonction de répartition empirique associé à un échantillon $x_1, \ldots, x_n$ est la fonction en escalier suivante

$$F_n(x) = \frac{1}{n}\sum_{i=1}^{n} 1_{x_i \leq x} = \begin{cases} 0 & \text{si } x < x_1^\star, \\ i/n & \text{si } x_i^\star \leq x < x_{i+1}^\star, \\ 1 & \text{si } x > x_n^\star, \end{cases}$$

  où $x_1^\star, \ldots, x_n^\star$ désigne l'échantillon ordonné.

```
ecdf(x)
```

```
## Empirical CDF
## Call: ecdf(x)
##  x[1:100] = -2.5043, -2.4515, -2.1289,  ..., 2.3053, 2.4709
```

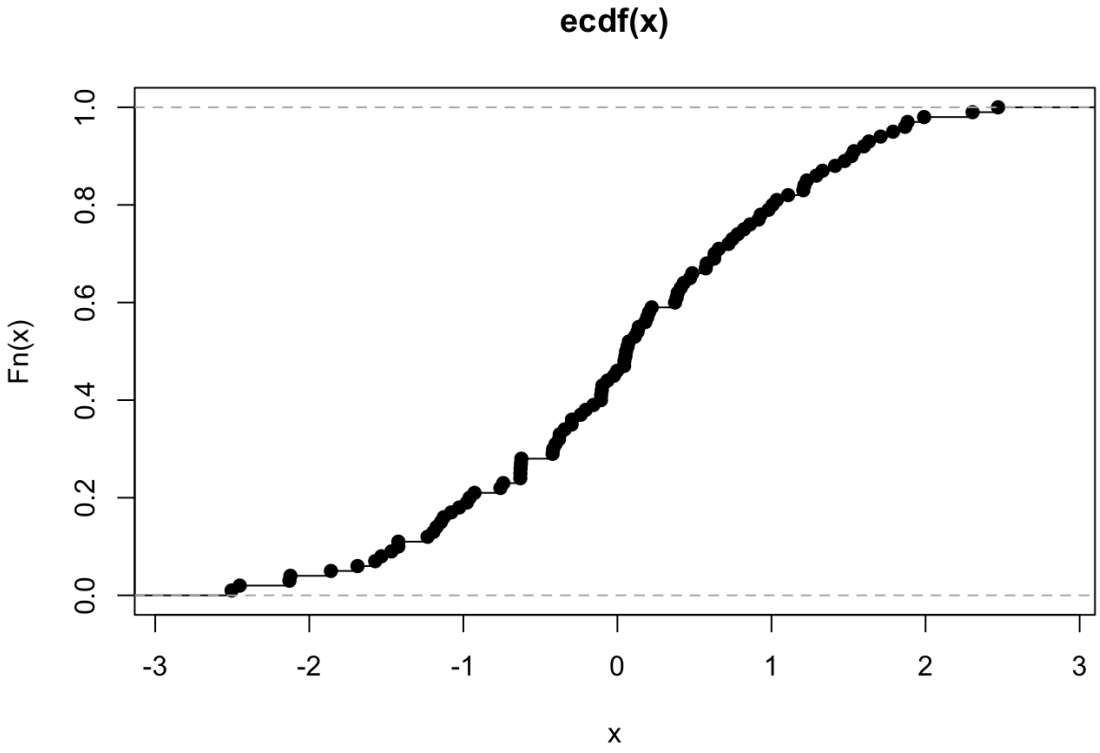
```
summary(ecdf(x))
```

```
## Empirical CDF:     100 unique values with summary
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.50435 -0.62897  0.06123  0.07226  0.83119  2.47088
```

```
class(ecdf(x))
```

```
## [1] "ecdf"     "stepfun"  "function"
```

```
plot(ecdf(x))
```



7.2 Statistiques descriptives

Nom	Indicateur	Quantité d'intérêt	Type	Fonction
moyenne	$\bar{x}_n$	$E(X)$	localisation	`mean()`
médiane	$q_{n,1/2}$	z tel que $P(X \leq z) = 1/2$	localisation	`median()`
variance	s_n^2	$Var(X)$	dispersion	`var()`
écart-type	$\sqrt{s_n^2}$	$\sqrt{Var(X)}$	dispersion	`sd()`
étendue	e_n	support de X	dispersion	`max()-min()`
minimum	$x_1^\star$	$\min(X)$	dispersion, extrême	`min()`
maximum	$x_n^\star$	$\max(X)$	dispersion, extrême	`max()`
quantile	$q_{n,p}$	z tel que $P(X \leq z) = p$	risque, extrême	`quantile()`

7.3 Fonctions d, p, q

En `R`, ces fonctions sont nommés par une lettre précédent le nom de la loi :

- `d` pour la densité f_X ou la fonction de masse de probabilité p_X,
- `p` pour la fonction de répartition F_X,
- `q` pour la fonction quantile q_X.

A chaque lettre, on acolle la racine du nom de la loi sur la base des tableaux suivants. Par exemple, la fonction de répartition de loi exponentielle est `pexp()`, la fonction de densité de loi normale est `dnorm()`.

Loi continue	Racine	Loi continue	Racine
beta	`beta`	logistique	`logis`

Loi continue	Racine	Loi continue	Racine
Cauchy	`cauchy`	lognormale	`lnorm`
chi-2	`chisq`	normale	`norm`
exponentielle	`exp`	Student t	`t`
Fisher F	`f`	uniforme	`unif`
gamma	`gamma`	Weibull	`weibull`

Loi discrète	Racine	Loi discrète	Racine
binomiale	`binom`	binomiale négative	`nbinom`
géométrique	`geom`	hypergéométrique	`hypergeom`
Poisson	`pois`		

7.4 Simulation de lois

Pour générer des nombres aléatoires d'une loi donnée, on utilise un générateur qui se base sur la fonction quantile et un générateur de loi uniforme. En `R`, il suffit d'utiliser le préfixe `r`. Ainsi `rexp()` génère des variables de loi exponentielle,…

7.5 Lois non implémentées dans R (base)

- Voir la thématique Distributions : https://CRAN.R-project.org/view=Distributions (https://CRAN.R-project.org/view=Distributions)
- Parmi la multitude de packages, le package `actuar` implémente un grand nombre des lois les plus utilisées en actuariat.

Feller-Pareto Distributions	Root
Feller-Pareto	`fpareto`
Transformed beta	`trbeta`
Burr	`burr`
Loglogistic	`llogis`
Paralogistic	`paralogis`
Generalized Pareto	`genpareto`
Pareto	`pareto`
Inverse Burr	`invburr`
Inverse Pareto	`invpareto`
Inverse paralogistic	`invparalogis`

Transformed gamma Distributions	Root
Transformed gamma	`trgamma`
Inverse transformed gamma	`invtrgamma`
Inverse gamma	`invgamma`
Inverse Weibull	`invweibull`
Inverse exponential	`invexp`

Other Distributions	Root
Loggamma	`lgamma`
Gumbel	`gumbel`
Inverse Gaussian	`invgauss`
Single parameter Pareto	`pareto1`
Generalized beta	`genbeta`

8 Manipulation de données

8.1 Import de données

- Voir R-data, http://cran.r-project.org/doc/manuals/r-release/R-data.html (http://cran.r-project.org/doc/manuals/r-release/R-data.html)

Type	Fonction	Package
fichier texte	`read.table()`	`utils`
fichier csv	`read.csv()`	`utils`
excel pre 2007	`read.xls()`	`gdata`
excel pre 2007	`read.xls()`	`xlsReadWrite`
excel pre 2007	`read_excel()`	`readxl`
excel post 2007	`read.xlsx()`	`xlsx`
excel post 2007	`read_excel()`	`readxl`
SAS Transport format	`read.xport()`	`foreign`
SAS data file	`read_sas()`	`haven`
depuis internet	`download.file()`	`utils`

8.2 Export de données

- Voir R-data, http://cran.r-project.org/doc/manuals/r-release/R-data.html (http://cran.r-project.org/doc/manuals/r-release/R-data.html)

Type	Fonction	Package
fichier texte	`write.table()`	`utils`
fichier csv	`write.csv()`	`utils`
excel pre 2007	`write.xls()`	`WriteXLS` ,
excel pre 2007	`write.xls()`	`xlsReadWrite`
excel post 2007	`write.xlsx()`	`xlsx`