Introduction au logiciel R

et à la pratique des statistiques multivariées

Christophe Ambroise

Université d'Évry Val d'Essonne

9-10 juillet 2012

`http://stat.genopole.cnrs.fr/~jchiquet/fr/initiation_R`

Première partie I

Entrées / Sorties

Plan
Entrées / Sorties

commande `scan`

Une utilisation élémentaire de `scan` permet une saisie plus agréable que la saisie directe des éléments d'un vecteur.

```
> x<-scan()
1: 1
2: 2
3: 3
4: 4
5: 5
6:
Read 5 items
>
> x
[1] 1 2 3 4 5
```

$\rightsquigarrow$ valable pour les jeux de données d'au plus quelques dizaines d'éléments...

Éditer des données

commande `edit`

Permet d'éditer des données existantes à l'aide d'un mini-tableur. Utile pour faire de petites modifications.

```
> new.data <- edit(old.data)
```

FIGURE: Éditeur Mac OS 10.6 / R 2.10

Fichiers binaires

commandes `save` et `load`

La commande `save` permet de sauvegarder un sous ensemble des données de l'espace de travail dans un fichier binaire ; `load` permet de les recharger.

```
> x <- rnorm(125)
> y <- 1 + x + x^2
> save(file = "mes_simus", x, y)
> rm(list = ls())
> objects()

character(0)

> load(file = "mes_simus")
> objects()

[1] "x" "y"
```

Jeux de données prédéfinies

commande `data`

R dispose d'une collection de données prédéfinies directement utilisables. La commande `data()` permet de les lister puis de les charger.

```
> data(iris3)
> head(iris3)

[1] 5.1 4.9 4.7 4.6 5.0 5.4
```

- La description d'un jeu de données est accessible dans l'aide.
- L'installation d'un nouveau package rend souvent disponibles de nouveaux jeux de données accessibles par `data`.

D'abord, un bon éditeur . . . il vous permet de constater le formatage d'un fichier texte et comment en « attaquer » l'importation.

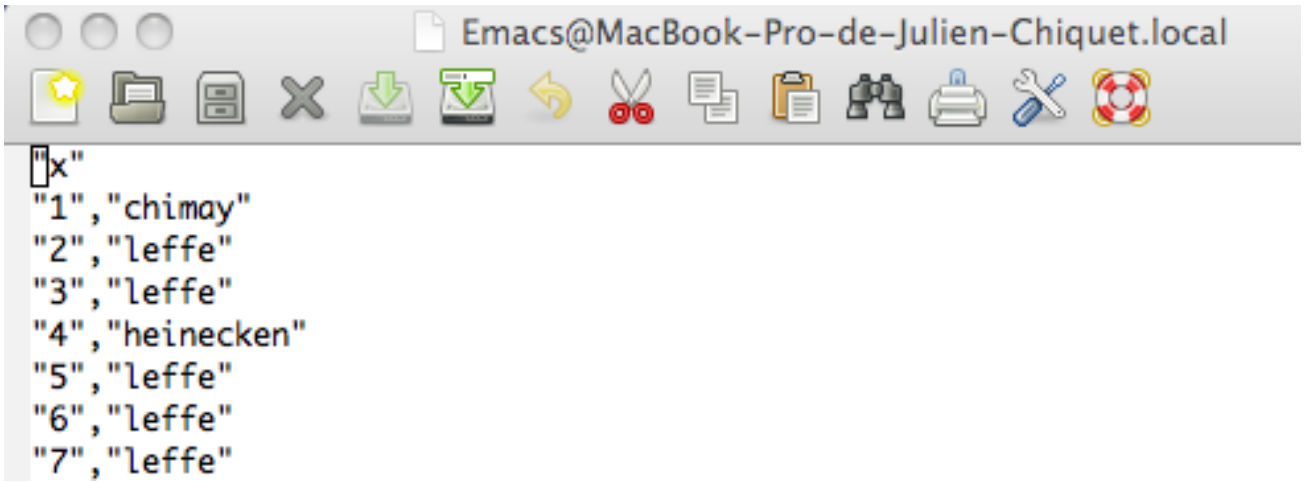


FIGURE: Fichier au formatage "csv"

commande `read.table`

Elle permet de lire un fichier formaté sous forme de table.

`read.table` stocke les données sous forme d'objet `data.frame`.

```
> mes_donnees <- read.table("mesures_baie_raisin_2008-2009.txt",
+     header = TRUE, sep = "\t")
> head(mes_donnees)
  Population variete nbre.pepin.baie.2008 poids.pulpe.baie..g..2008
1         CE    1784                   1.0                      0.89
2         CE     124                   1.0                      1.14
3         CE     210                   1.2                      1.26
4         CE    1805                   1.2                      0.66
5         CE    1303                   1.2                      0.83
6         CE     284                   1.3                      0.54
  volume.baie..cm3..2008 nbre.pepin.baie.2009 poids.pulpe.baie..g..2009
1                   7.70                    NA                       NA
2                   8.82                    NA                       NA
3                  10.20                    NA                       NA
4                     NA                    NA                       NA
5                     NA                    NA                       NA
6                   4.61                    NA                       NA
```

Lecture de fichiers III

`read.csv,read.delim`

commandes `read.csv` et `read.delim`

Ce sont des raccourcis pour la fonction `read.table`, spécialisés dans l'importation des données « .csv » (*comma-separated value*) ou tabulées (le séparateur est la tabulation).

commandes `write.table`, `write.csv` et `write.delim`

La fonction `write.table` permet d'imprimer les données issues d'une `data.frame` dans un fichier texte externe. `write.csv` et `write.delim` sont des raccourcis pour les données `csv` ou tabulée.

Pour aller plus loin. . .

Beaucoup de choses sur l'importation des données dans

R Data Import /Export.
`http://cran.r-project.org/doc/manuals/R-data.pdf`

- Exemples avancés avec `read.table`,
- communication avec les bases de données (SQL),
- importation de données Excel,
- . . .

Paramètres récurrents

Forme générique

La plupart des fonctions graphique s'utilisent par un appel du type

1. `nom.function(object, options)`,
2. `nom.function(x, y , options)`.

Parmi les options les plus courantes, on trouve :

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

Paramètres récurrents

Forme générique

La plupart des fonctions graphique s'utilisent par un appel du type

1. `nom.function(object, options)`,
2. `nom.function(x, y , options)`.

Parmi les options les plus courantes, on trouve :

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

Tracer une fonction numérique I

commande `plot`

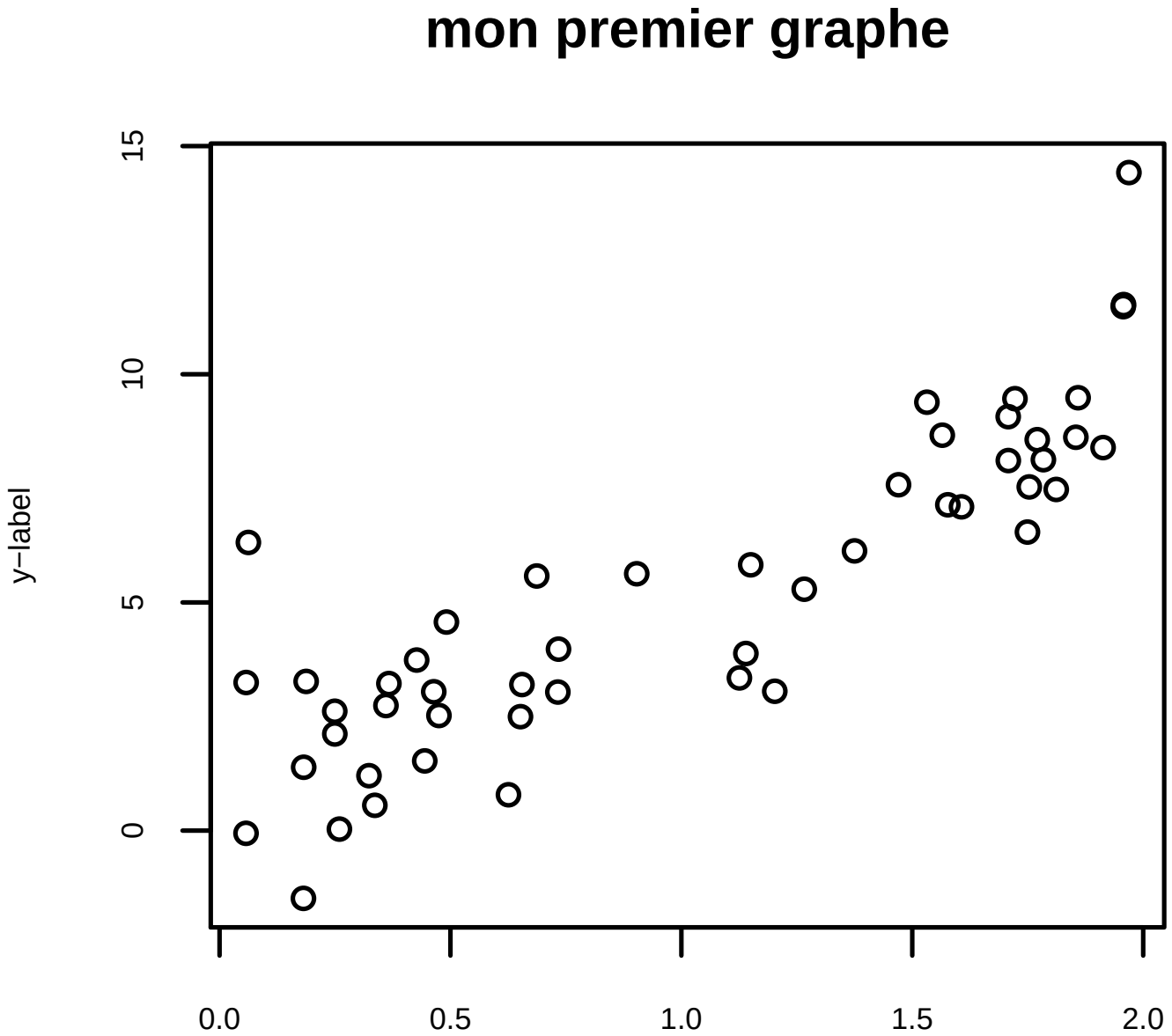
Fonction élémentaire de réprésentation graphique.

- `plot(vect)` représente le graphe des valeurs de `vect` sur l'axe des y.
- `plot(vect1,vect1)` représente le graphe des valeurs de `vect2` en fonction de `vect1`.

Par exemple, avec deux vecteurs :

```
> x <- runif(50, 0, 2)
> y <- 3 * x + 2 * x^2 + 1 + rnorm(50, sd = 1.5)
> plot(x, y, xlab = "x-label", ylab = "y-label", main = "mon premier graphe")
```

Tracer une fonction numérique II



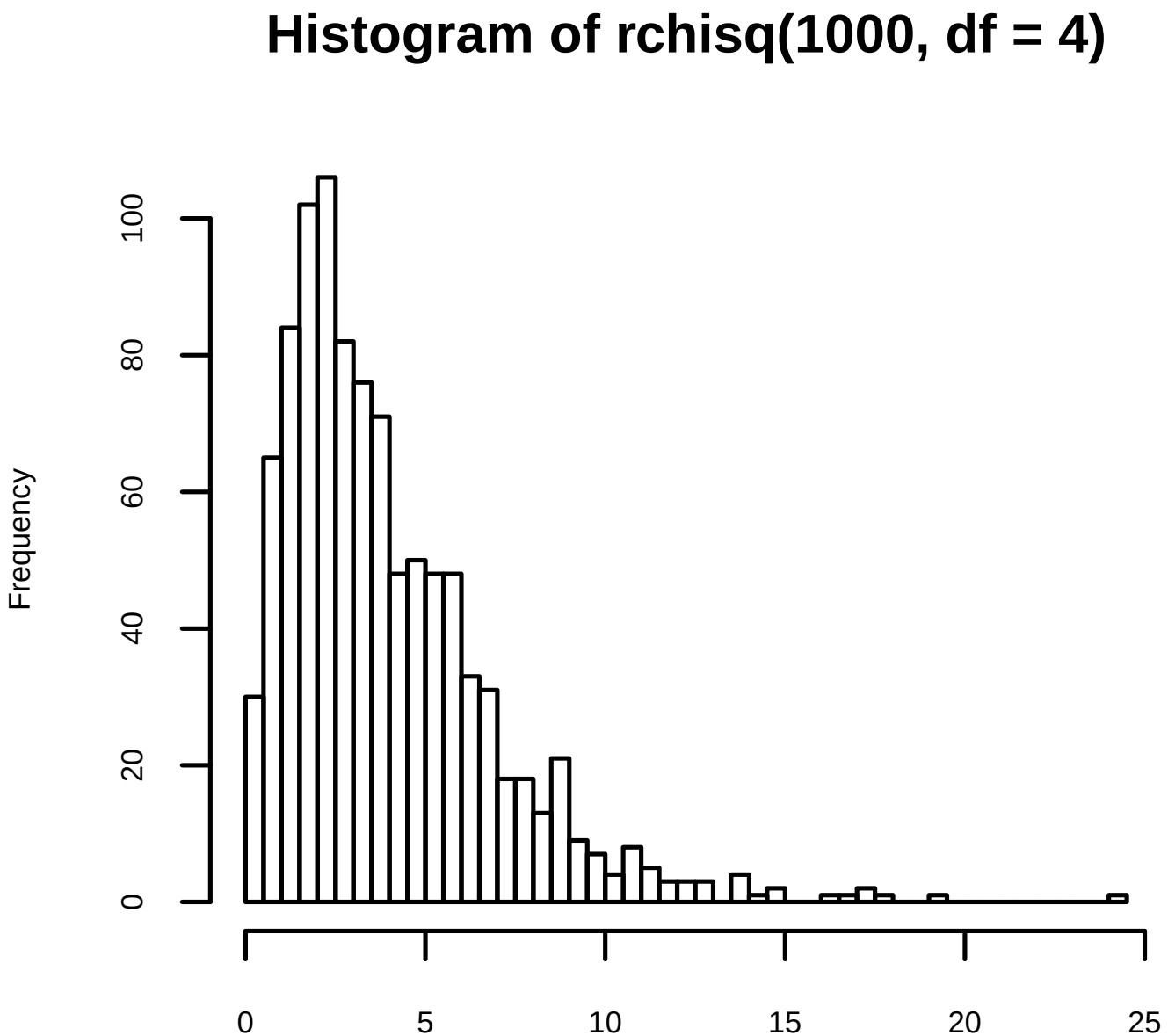
Tracer un objet R I
Toujours avec la commande plot

Beaucoup d'objet R accepte la commande plot ! En particulier, les histogrammes :

```
> mon_histo <- hist(rchisq(1000, df = 4), nclass = 75)
> plot(mon_histo, main = "distribution empirique du Khi-2")
```

Tracer un objet R II

Toujours avec la commande `plot`



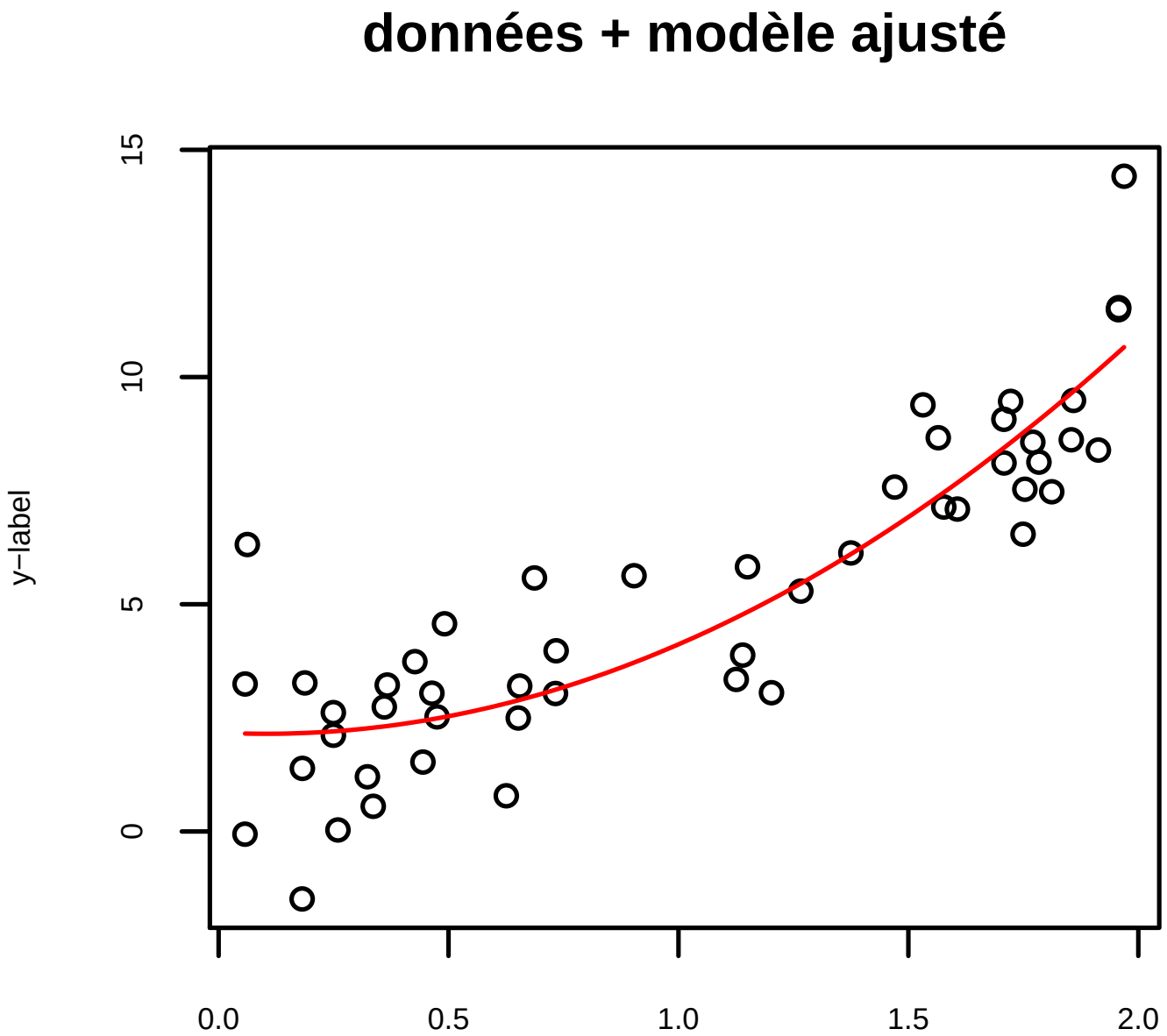
Tracer une fonction symbolique I

commande `curve`

Elle permet de tracer une fonction définie par une expression de `x`.

```
> plot(x, y, main = "données + modèle ajusté", xlab = "x-label",
+     ylab = "y-label")
> a <- coefficients(lm(y ~ 1 + x + I(x^2)))
> curve(a[1] + a[2] * x + a[3] * x^2, add = TRUE, col = "red")
```

Tracer une fonction symbolique II



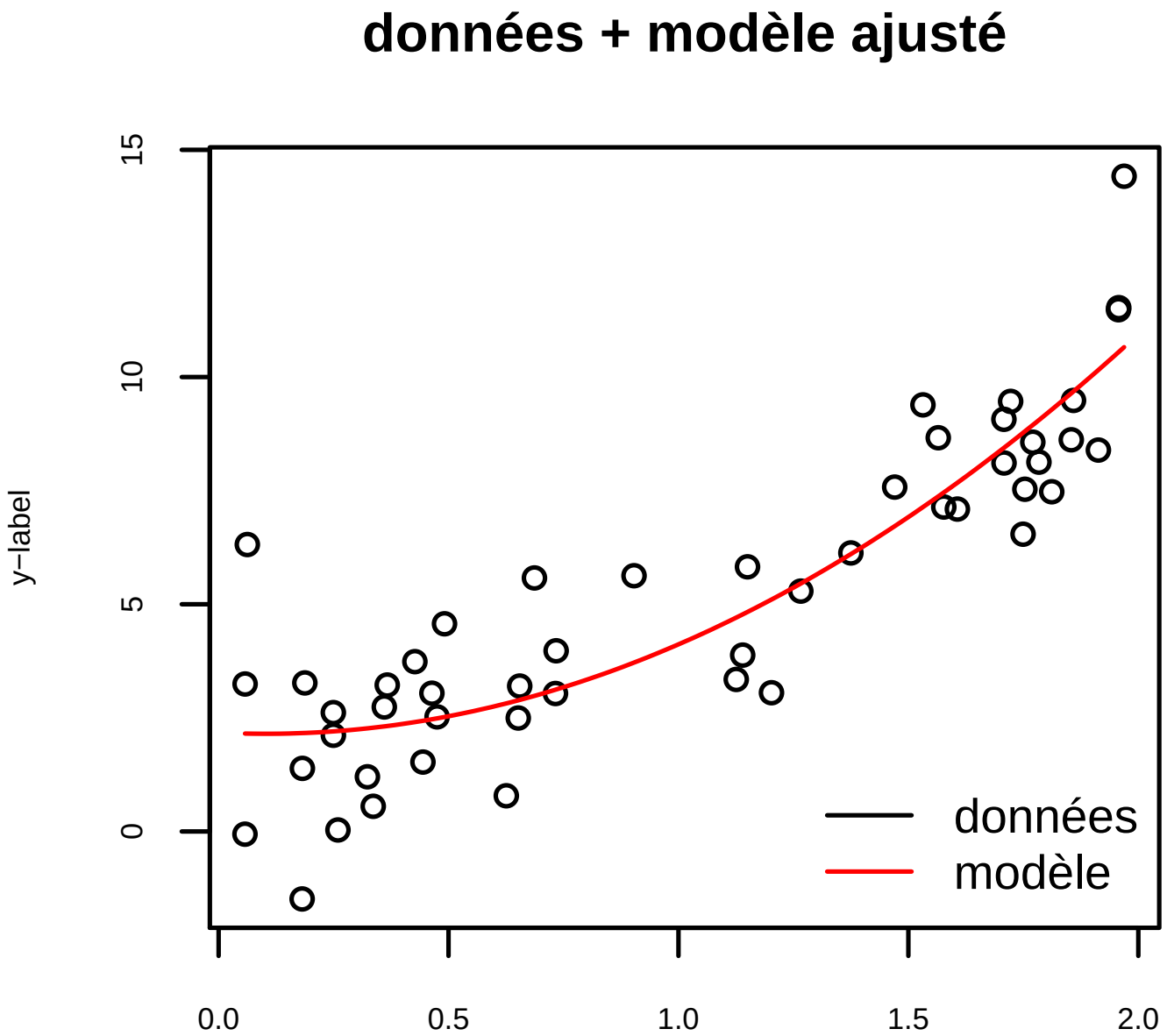
Ajouter une légende I

commande `legend`

Pour ajouter une légende. Attention aux options, assez nombreuses !

```
> plot(x, y, main = "données + modèle ajusté", xlab = "x-label",
+     ylab = "y-label")
> a <- coefficients(lm(y ~ 1 + x + I(x^2)))
> curve(a[1] + a[2] * x + a[3] * x^2, add = TRUE, col = "red")
> legend("bottomright", c("données", "modèle"), lty = c(1, 1),
+     col = c("black", "red"), bty = "n")
```

Ajouter une légende II



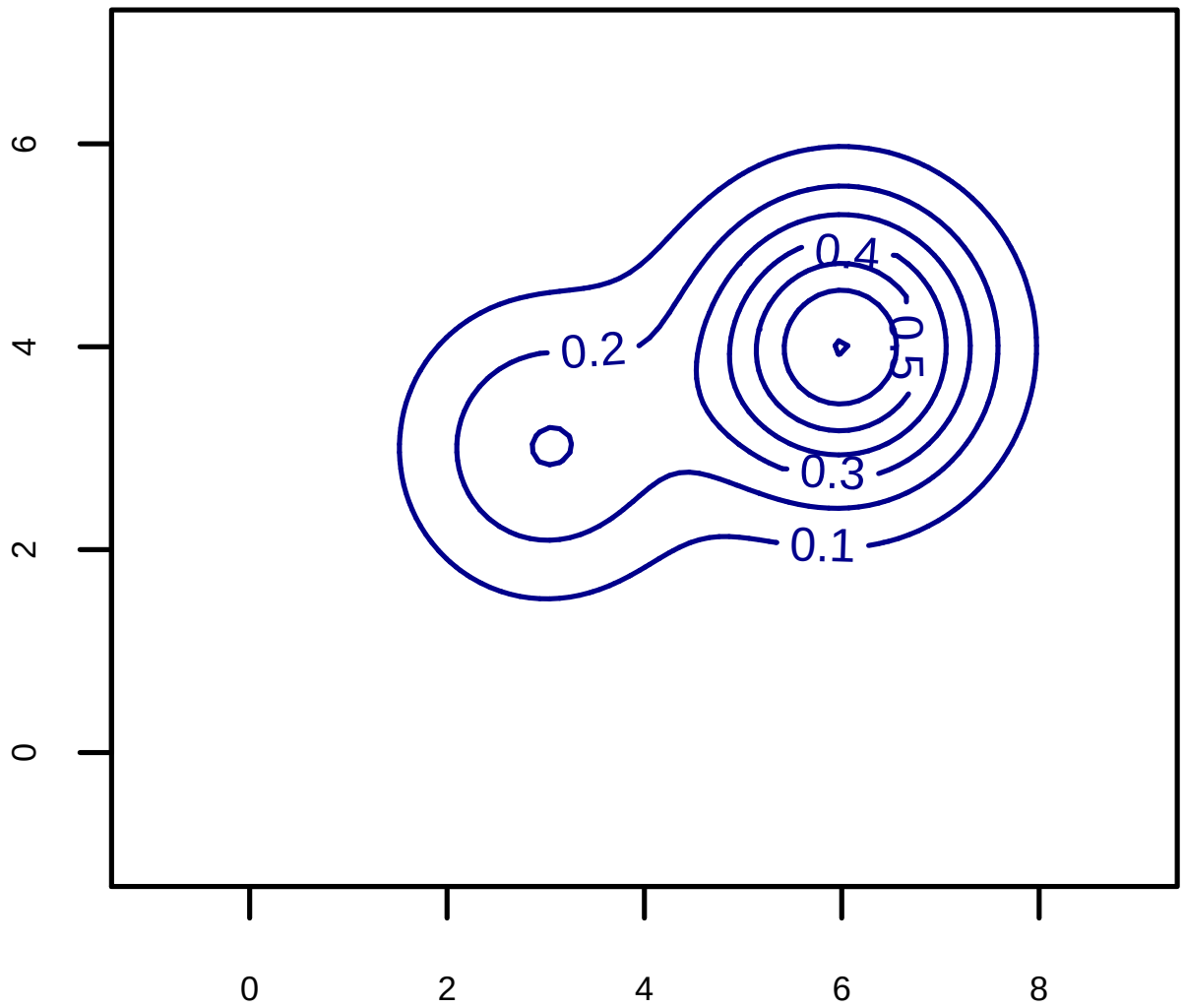
Représentation 3D (courbe de niveaux) I

commande `contour`

`contour(x,y,z)` permet de tracer des courbes de niveaux : `x` et `y` sont des vecteurs et `z` une matrice telle que les dimensions de `z` soient `length(x),length(y)`.

```
> x <- seq(-1, 9, length = 100)
> y <- seq(-1, 7, length = 100)
> z <- outer(x, y, function(x, y) 0.3 * exp(-0.5 * ((x - 3)^2 +
+     (y - 3)^2)) + 0.7 * exp(-0.5 * ((x - 6)^2 + (y - 4)^2)))
> contour(x, y, z, col = "blue4")
```

Représentation 3D (courbe de niveaux) II



Ajout de droites I

commande `abline`

`abline` permet d'ajouter à un graphe courant

- des droites de décalage `a` et de coefficient directeur `b` avec `abline(a,b)`,
- des droites verticales avec `abline(v=)`,
- des droites horizontales avec `abline(h=)`.

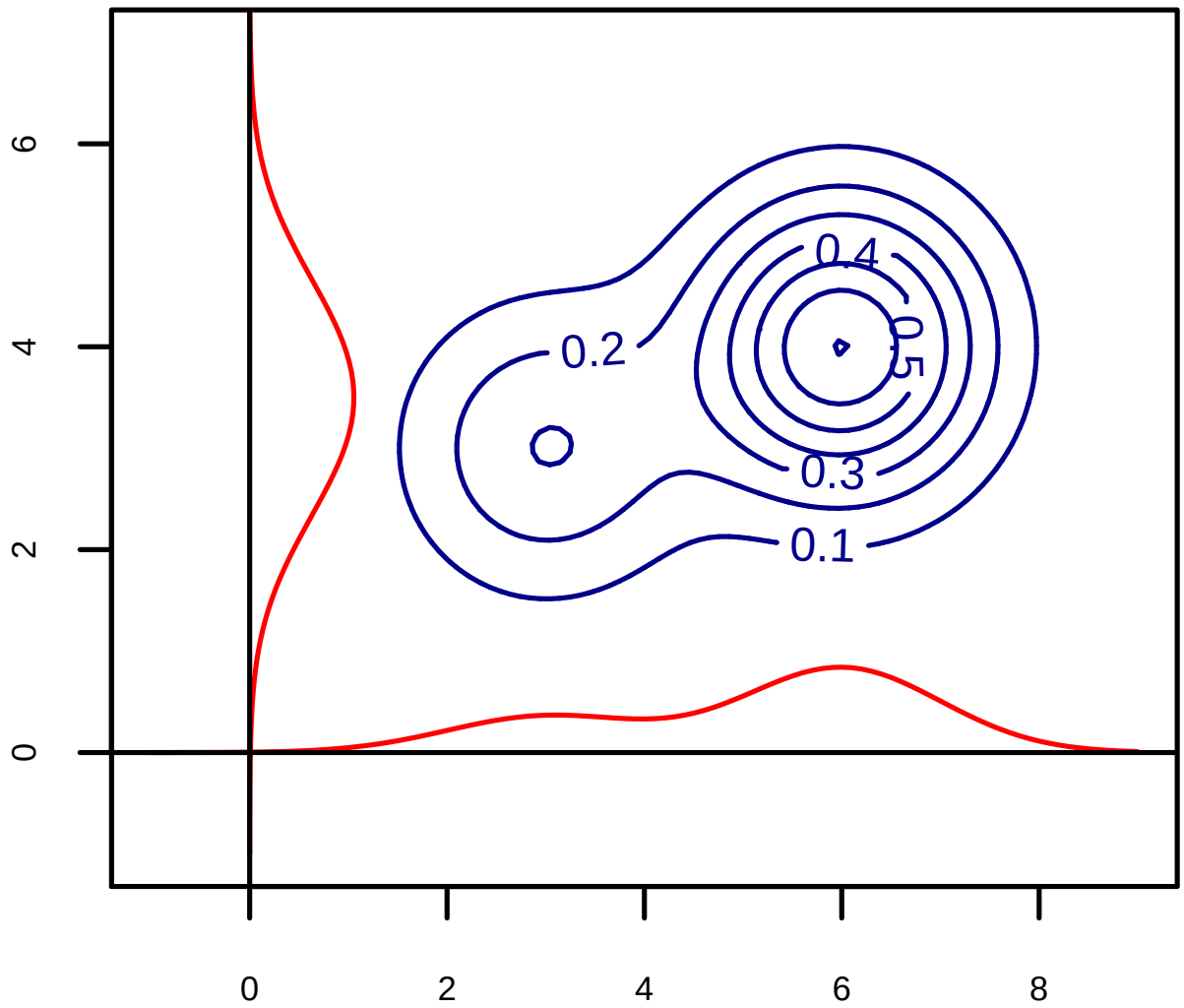
commandes `lines` et `points`

Pour ajouter une courbe ou des points : s'utilisent de manière similaire à `plot`.

Ajout de droites II

```
> contour(x, y, z, col = "blue4")
> curve((0.3 * dnorm(x, mean = 3) + 0.7 * dnorm(x, mean = 6)) *
+     3, -1, 9, col = "red", ylim = c(-1, 7), add = T)
> x <- seq(-1, 9, length = 100)
> lines((0.5 * dnorm(x, mean = 3) + 0.5 * dnorm(x, mean = 4)) *
+     3, x, col = "red")
> abline(h = 0)
> abline(v = 0)
```

Ajout de droites III



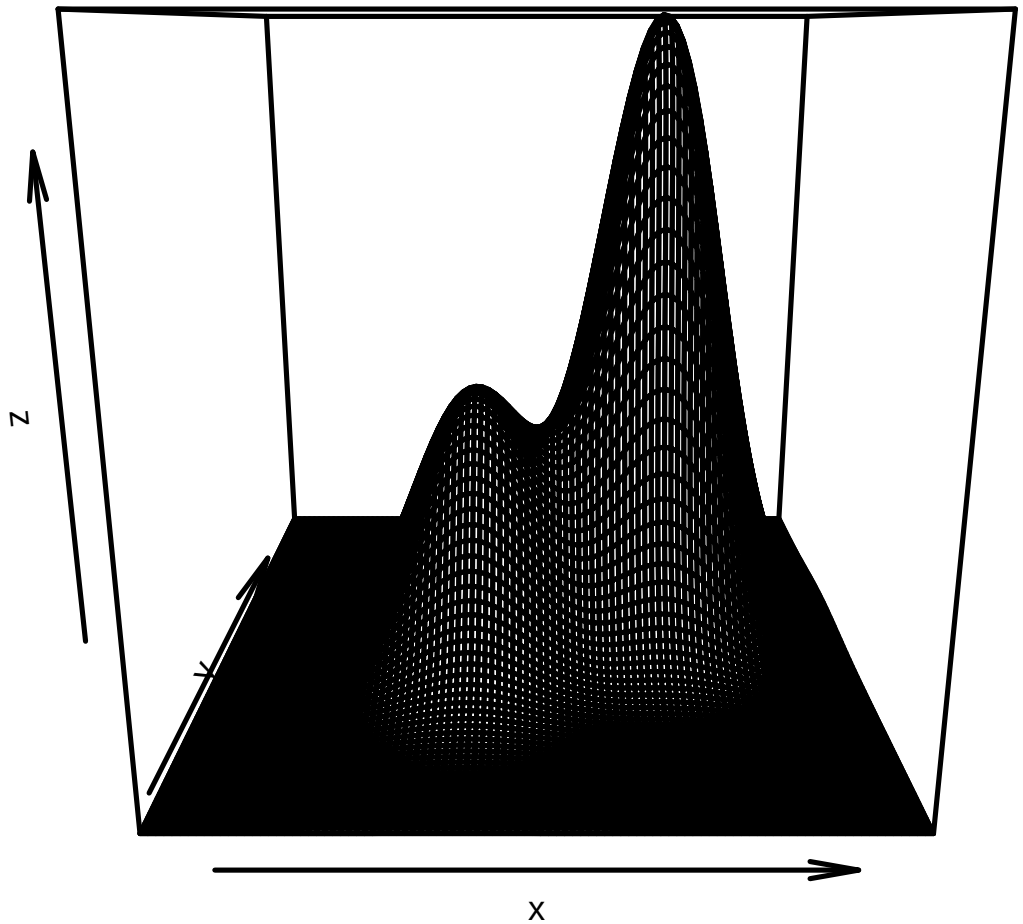
Graphe en 3D I

commande `persp`

Fonctionne comme la fonction `contour` en proposant une représentation en perpective.

```
> persp(x, y, z)
```

Graphe en 3D II



Rediriger la sortie graphique

Par défaut, R envoie les graphiques sur la sortie *écran*. De nombreuses

Exportation de graphes

Se réalise en encadrant les fonctions graphiques par les commandes `format_export(file="nom_fichier")` et `dev.off()`, où `format_fichier` peut prendre les valeurs `pdf`,`postscrip`,`png`,....

```
pdf(file="ma\_sortie.pdf")
plot(runif(20),runif(20))
dev.off()
```

Graphes multiples I

Ouverture d'une nouvelle fenêtre graphique

Se fait, selon les plateformes, avec les commandes

- `x11()` pour Linux,
- `quartz()` ou `x11()` pour Mac OS,
- `windows()`.

Découpage d'une fenêtre

Plusieurs possibilités :

- `layout(mat,width=,height=)`, qui s'utilise en découpant l'écran via la matrice `mat`.
- `par(mfrow=vect)` ou `par(mfcol=vect)` qui découpent en n lignes et m colonne spécifiées par le vecteur `vect`. Le remplissage se fait par ligne ou par colonne selon la fonction choisie.

Pour aller plus loin

- D'autres fonctions de haut niveau dans la partie dédiée aux statistiques
- Utiliser la liste des commandes usuelles pour les options et fonctions secondaires,
- La commande `par` gère les options graphiques,
- Consulter le package `lattice`, *extrêmement* puissant.

Lattice : Multivariate Data Visualization with R
Deepayan Sarkar
http://lmdvr.r-forge.r-project.org/

$\rightsquigarrow$ Cette page web propose toutes les figures et tous les codes R correspondant à leur génération !

Deuxième partie II

Statistiques et outils connexes sous R

Plan
Statistiques et outils connexes sous R

Plan
Statistiques et outils connexes sous R

Les distributions disponibles

Distribution	R	Paramètres
beta	`beta`	
binomiale	`binom`	size, prob
binomiale négative	`nbinom`	
Cauchy	`cauchy`	
Chi-deux	`chiqsq`	df
Exponentielle	`exp`	rate
Fisher	`f`	df1, df2
Gamma	`gamma`	
géométrique	`geom`	
hypergéométrique	`hyper`	
log-normal	`lnorm`	
logistique	`logis`	
normale	`norm`	mean, sd
Poisson	`pois`	
Student	`t`	df
uniforme	`unif`	min, max
Weibull	`weibull`	
Wilcoxon	`wilcox`	

TABLE: Principales distributions

Les distributions disponibles

Distribution	R	Paramètres
beta	`beta`	
binomiale	`binom`	size, prob
binomiale négative	`nbinom`	
Cauchy	`cauchy`	
Chi-deux	`chiqsq`	df
Exponentielle	`exp`	rate
Fisher	`f`	df1, df2
Gamma	`gamma`	
géométrique	`geom`	
hypergéométrique	`hyper`	
log-normal	`lnorm`	
logistique	`logis`	
normale	`norm`	mean, sd
Poisson	`pois`	
Student	`t`	df
uniforme	`unif`	min, max
Weibull	`weibull`	
Wilcoxon	`wilcox`	

TABLE: Principales distributions

Les distributions disponibles

Distribution	R	Paramètres
beta	`beta`	
binomiale	`binom`	size, prob
binomiale négative	`nbinom`	
Cauchy	`cauchy`	
Chi-deux	`chiqsq`	df
Exponentielle	`exp`	rate
Fisher	`f`	df1, df2
Gamma	`gamma`	
géométrique	`geom`	
hypergéométrique	`hyper`	
log-normal	`lnorm`	
logistique	`logis`	
normale	`norm`	mean, sd
Poisson	`pois`	
Student	`t`	df
uniforme	`unif`	min, max
Weibull	`weibull`	
Wilcoxon	`wilcox`	

TABLE: Principales distributions

Tirage aléatoire

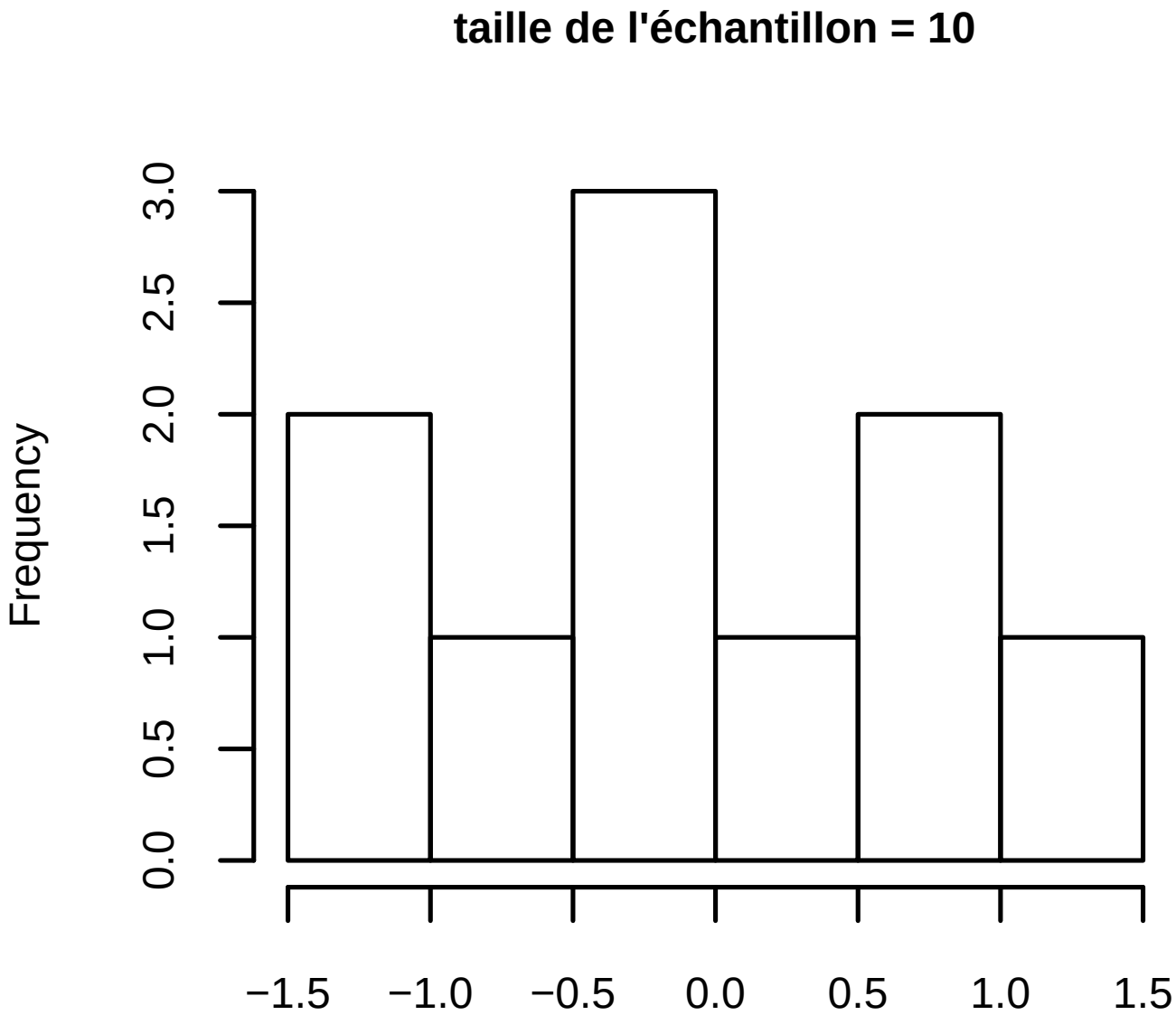
Forme générique : `r+distrib(n,...)`

`r` pour « random » : `n` donne la taille de l'échantillon et `...` sont les paramètres requis selon la forme de `distrib`.

```
> rexp(10, rate = 1/5)

 [1]  1.1054148  0.8953368 12.1682716  1.9114737  7.1211419  4.2790374
 [7]  2.9347596  1.6264013 15.7787362 10.8417846

> rchisq(10, df = 5)

 [1]  2.728322  4.078512  2.175310  9.056470 15.755881  4.161039  3.226331
 [8]  2.567603  2.092264  4.350377

> runif(10, min = -2, max = 2)

 [1] -1.880313645 -1.137548671 -1.884651253  0.008914834 -0.250922194
 [6] -1.903616990  1.653592046 -0.472785781 -0.267910359  1.780345962

> mean(rbinom(1000, 10, prob = 1/2))

[1] 4.986

> var(rnorm(1000, mean = 5, sd = 2))

[1] 4.005591
```

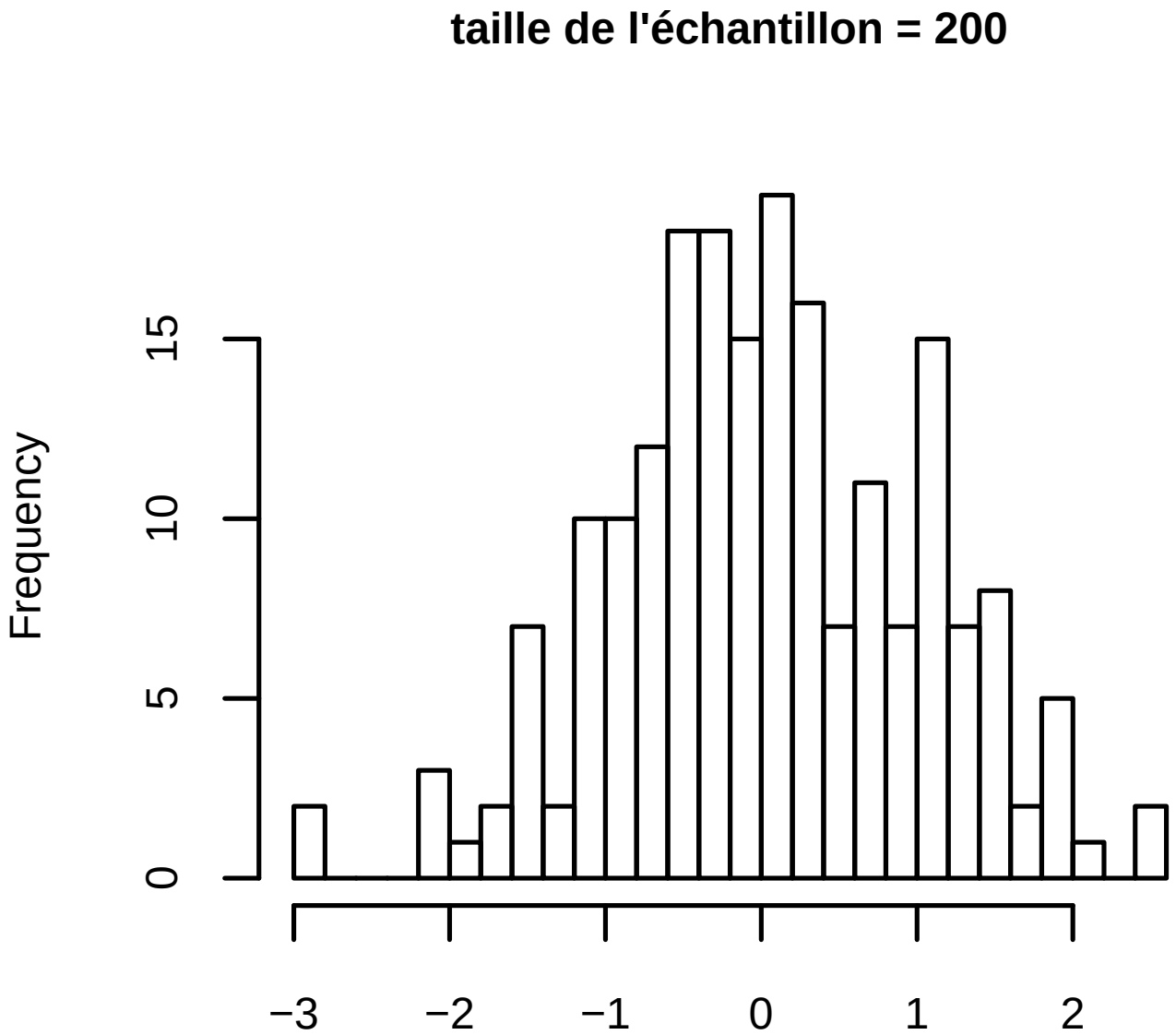
Exemple avec la loi normale : histogramme

Avec $n = 10$



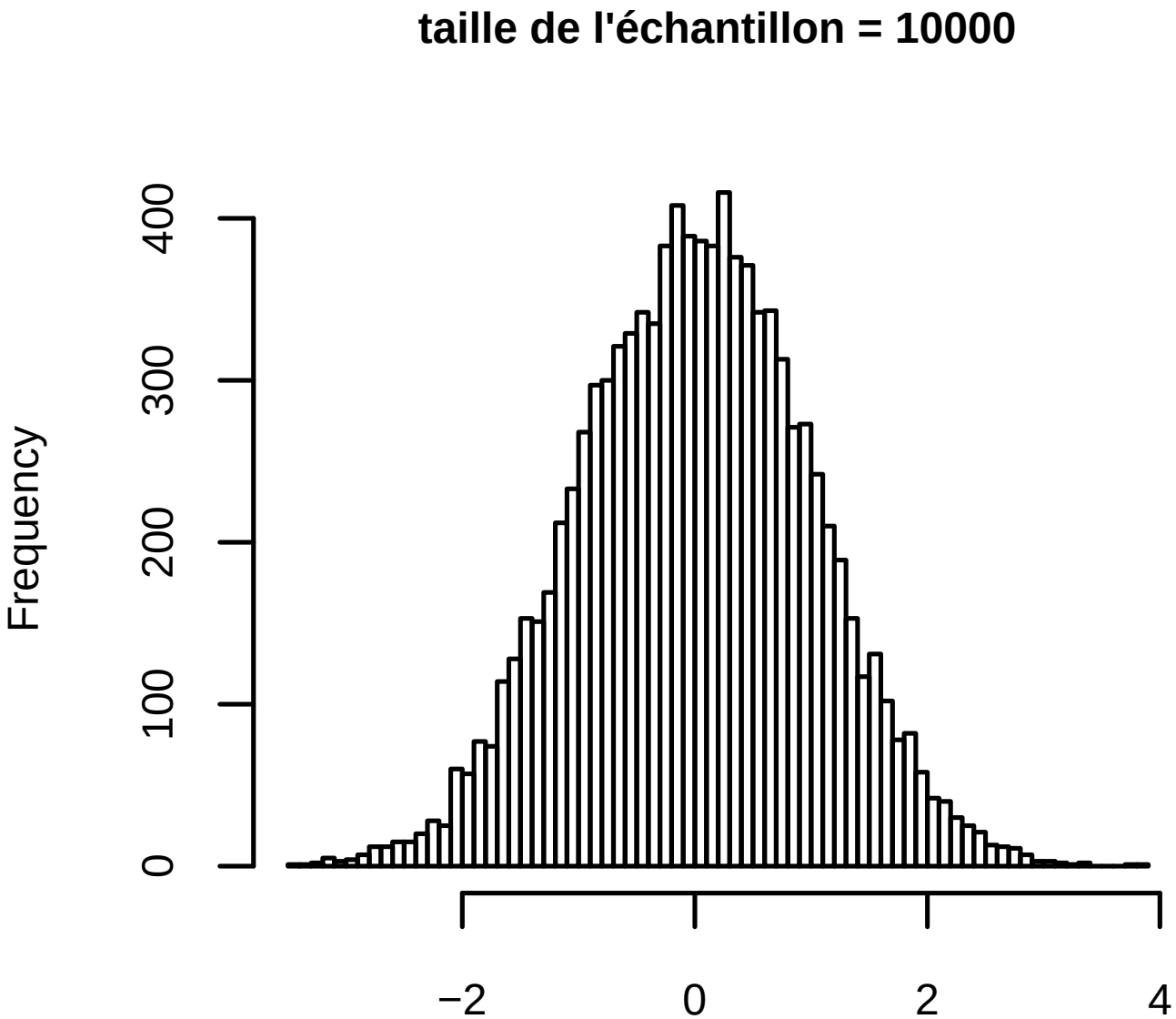
Exemple avec la loi normale : histogramme

Avec $n = 200$



Exemple avec la loi normale : histogramme

Avec $n = 10000$



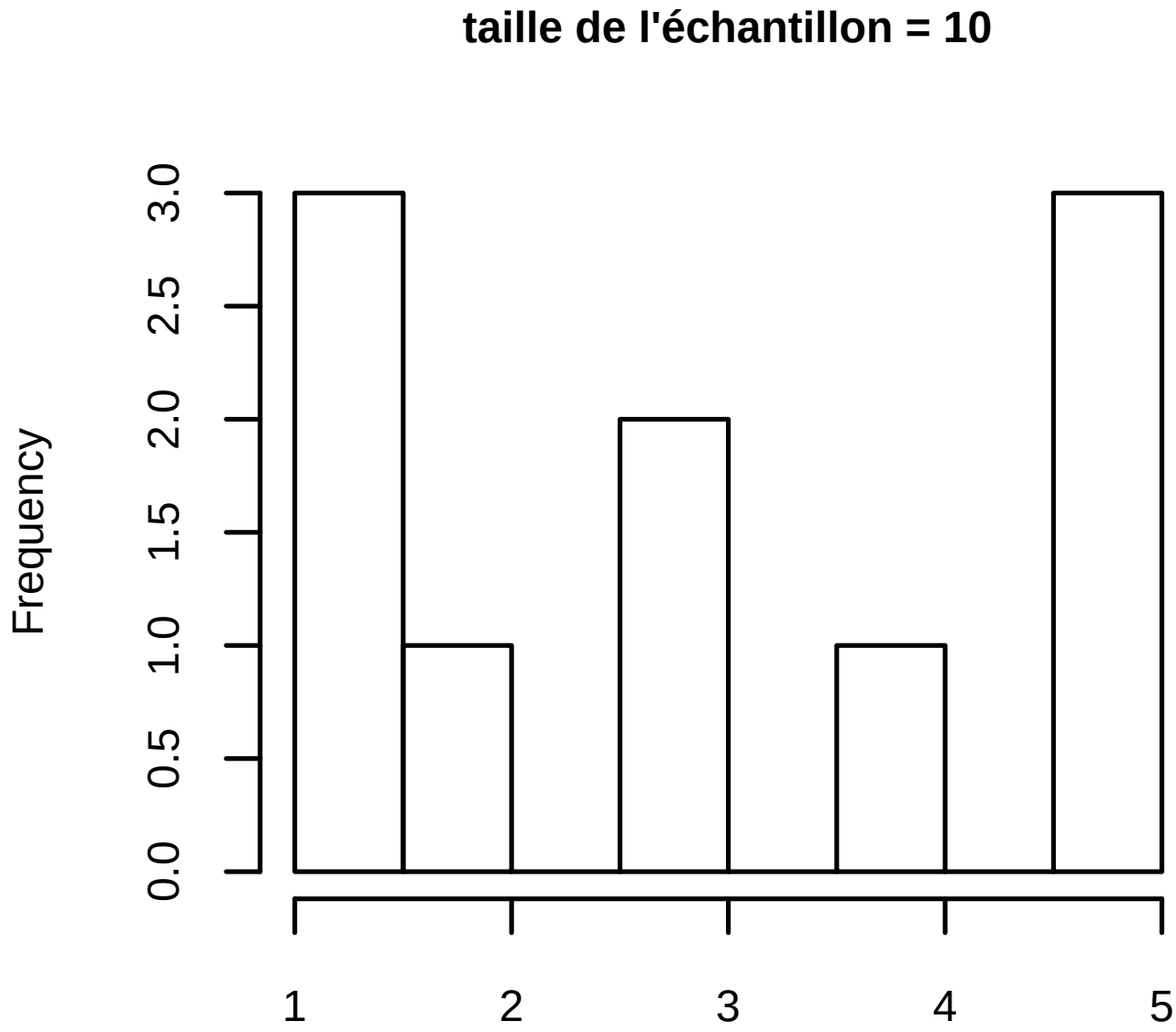
Tirage aléatoire avec `sample`

Définir une distribution discrète

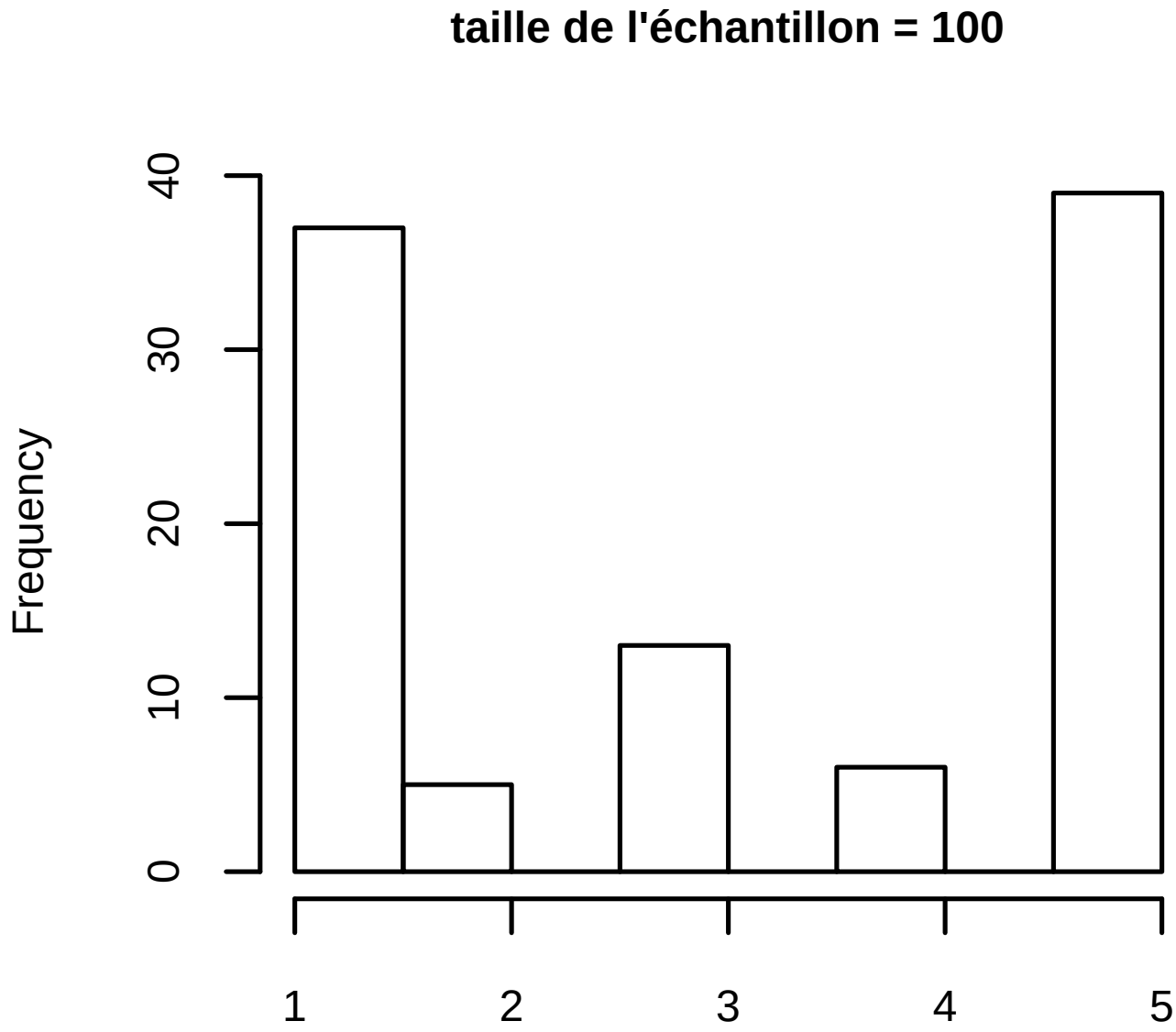
La fonction `sample(x, size, replace=FALSE, prob=NULL)` permet d'échantillonner les éléments de `x` : le tirage est de taille `size`, avec ou sans remise. Si `prob` est vide, chaque élément est équiprobable.

```
> sample(1:5)

[1] 1 5 4 3 2

> sample(1:5, 10, replace = TRUE)

 [1] 4 5 1 5 5 3 2 4 2 1

> sample(1:5, 10, replace = TRUE, prob = c(0.35, 0.1, 0.1, 0.1,
+     0.35))

 [1] 2 1 5 1 1 5 5 1 5 3
```

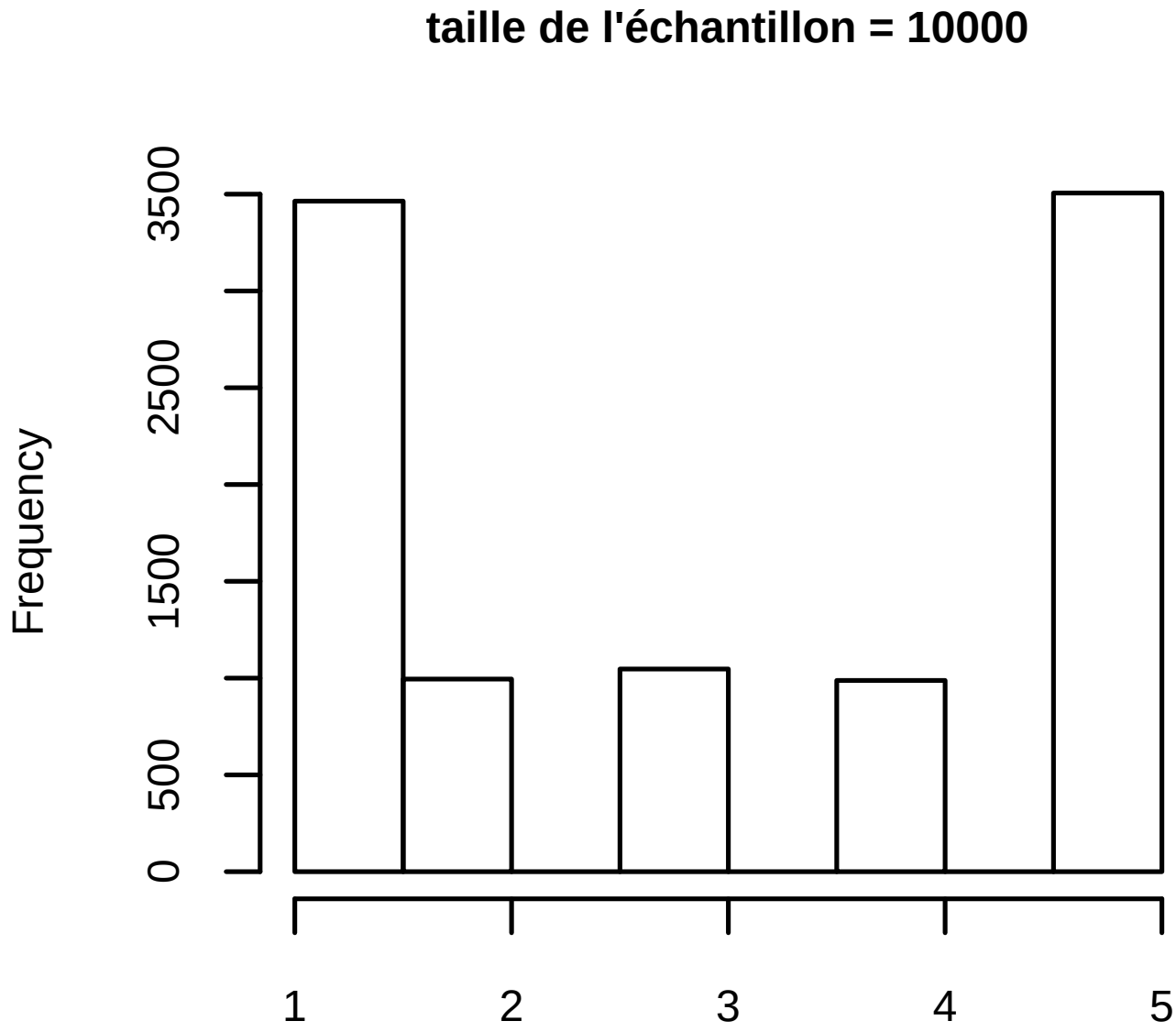
Avec $n = 10$


taille de l'échantillon = 10
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1
2
3
4
5

Avec $n = 100$



Avec $n = 10000$


taille de l'échantillon = 10000
Frequency
0
500
1500
2500
3500
1
2
3
4
5

Fonction de répartition

Forme générique : `p+distrib(x,...)`

`p` pour « probability distribution function » : donne $\mathbb{P}(X \leq \mathtt{x})$, où X est une variable aléatoire de loi `distrib`.

```
> pnorm(0.5)
[1] 0.6914625
> pnorm(0.5, mean = 2, sd = 3)
[1] 0.3085375
> pnorm((0.5 - 2)/3)
[1] 0.3085375
> pbinom(5, 10, 0.25)
[1] 0.9802723
```

Densité

Forme générique : `d+distrib(x,...)`

`d` pour « density » : donne la densité pour une variable aléatoire continue et $\mathbb{P}(X = \mathtt{x})$ pour X une variable aléatoire discrète.

```
> dnorm(0.5)
[1] 0.3520653
> dexp(3, 1/8)
[1] 0.08591116
> dbinom(5, 10, 0.25)
[1] 0.0583992
> dpois(4, 2)
[1] 0.09022352
```

Fractiles

Forme générique : `q+distrib(alpha,...)`

`q` pour « quantile » : donne la valeur de x définie par

$$\mathbb{P}(X \leq x) = \alpha,$$

où X est une variable aléatoire de loi `distrib`.

```
> qnorm(0.95)
[1] 1.644854
> qt(0.4, df = 28)
[1] -0.2557675
> qchisq(0.05, df = 6)
[1] 1.635383
```

Plan
Statistiques et outils connexes sous R

Reprenons l'exemple de la vigne

Renommons les variables et considérons uniquement les données de 2008 pour une manipulation plus agréable. J'enlève également la colonne « variété », car je ne vois pas à quoi elle sert.

```
> vigne <- read.delim("mesures_baie_raisin_2008-2009.txt", header = TRUE)
> vigne <- vigne[-c(2, 6, 7)]
> colnames(vigne) <- c("pop", "pepin.08", "poids.08", "volcm3.08")
> head(vigne)
```

```
  pop pepin.08 poids.08 volcm3.08
1  CE      1.0     0.89      7.70
2  CE      1.0     1.14      8.82
3  CE      1.2     1.26     10.20
4  CE      1.2     0.66        NA
5  CE      1.2     0.83        NA
6  CE      1.3     0.54      4.61
```

```
> attach(vigne)
```

Résumé statistique
Commande `summary`

Le résumé numérique s'adapte selon la nature des variables (univariée, multivariée, factorielle)

```
> summary(pepin.08)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  0.000   1.400   1.800   1.858   2.400   3.200  27.000

> summary(pop)

CE CO TE
84 89 72

> summary(vigne)

 pop        pepin.08         poids.08         volcm3.08
 CE:84   Min.   : 0.000   Min.   : 0.390   Min.   : 3.44
 CO:89   1st Qu.: 1.400   1st Qu.: 0.820   1st Qu.: 7.14
 TE:72   Median : 1.800   Median : 1.060   Median : 8.91
         Mean   : 1.858   Mean   : 1.212   Mean   :10.47
         3rd Qu.: 2.400   3rd Qu.: 1.360   3rd Qu.:11.90
         Max.   : 3.200   Max.   : 3.750   Max.   :33.80
         NA's   :27.000   NA's   :28.000   NA's   :44.00
```

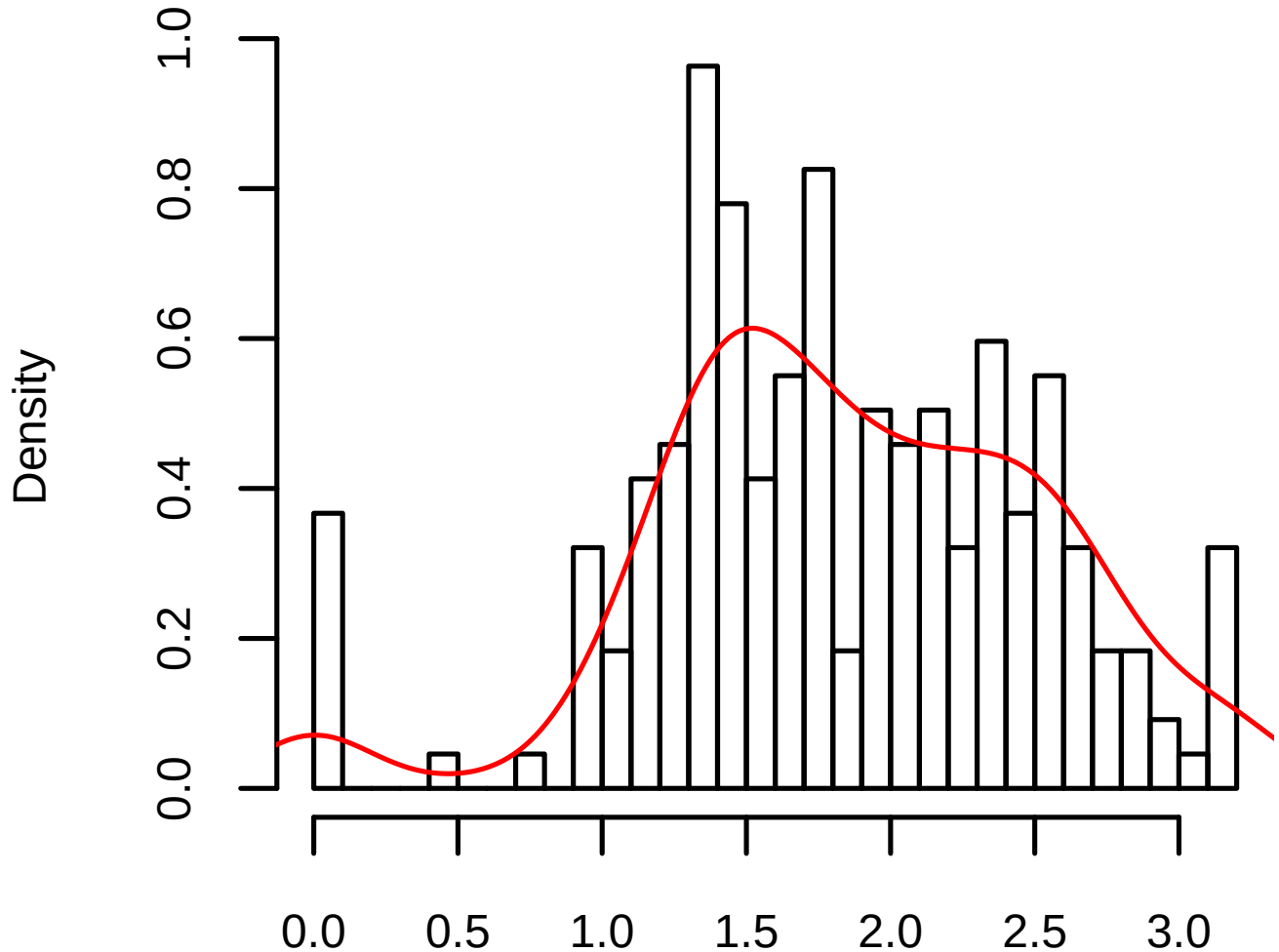
Graphe en tiges et feuilles
Commande `stem`

Un graphe en tige est feuille peut être vu comme un histogramme.

```
> stem(pepin.08)
  The decimal point is 1 digit(s) to the left of the |

   0 | 00000000
   2 |
   4 | 0
   6 |
   8 | 0
  10 | 00000000000
  12 | 00000000000000000
  14 | 0000000000000000000000000000000000000
  16 | 00000000000000000000
  18 | 00000000000000000000
  20 | 00000000000000000000
  22 | 00000000000000000
  24 | 00000000000000000000
  26 | 00000000000000000000
  28 | 00000000
  30 | 000
  32 | 0000000
```

Histogramme

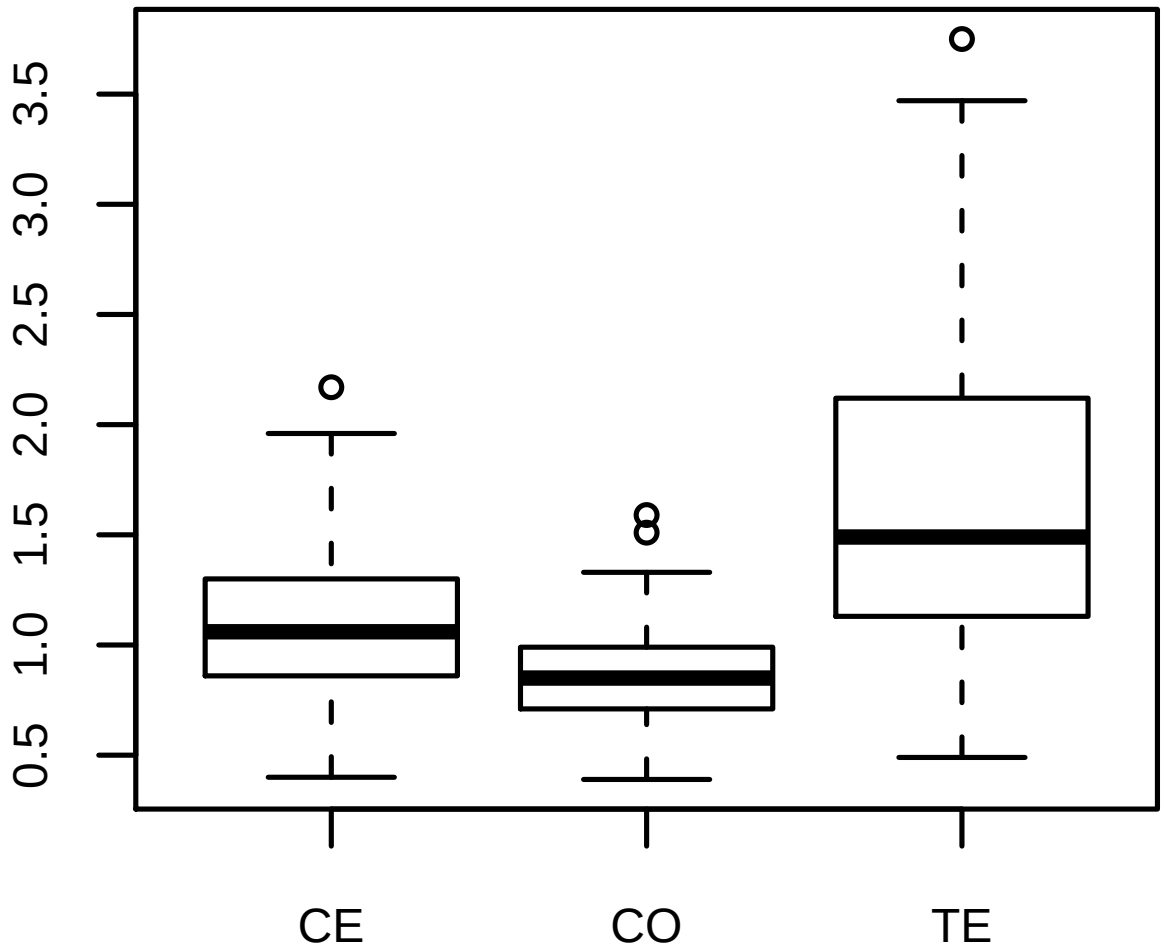
```
> hist(pepin.08, nclass = 25, prob = TRUE)
> lines(density(pepin.08[!is.na(pepin.08)]))
```



Boîtes à moustaches

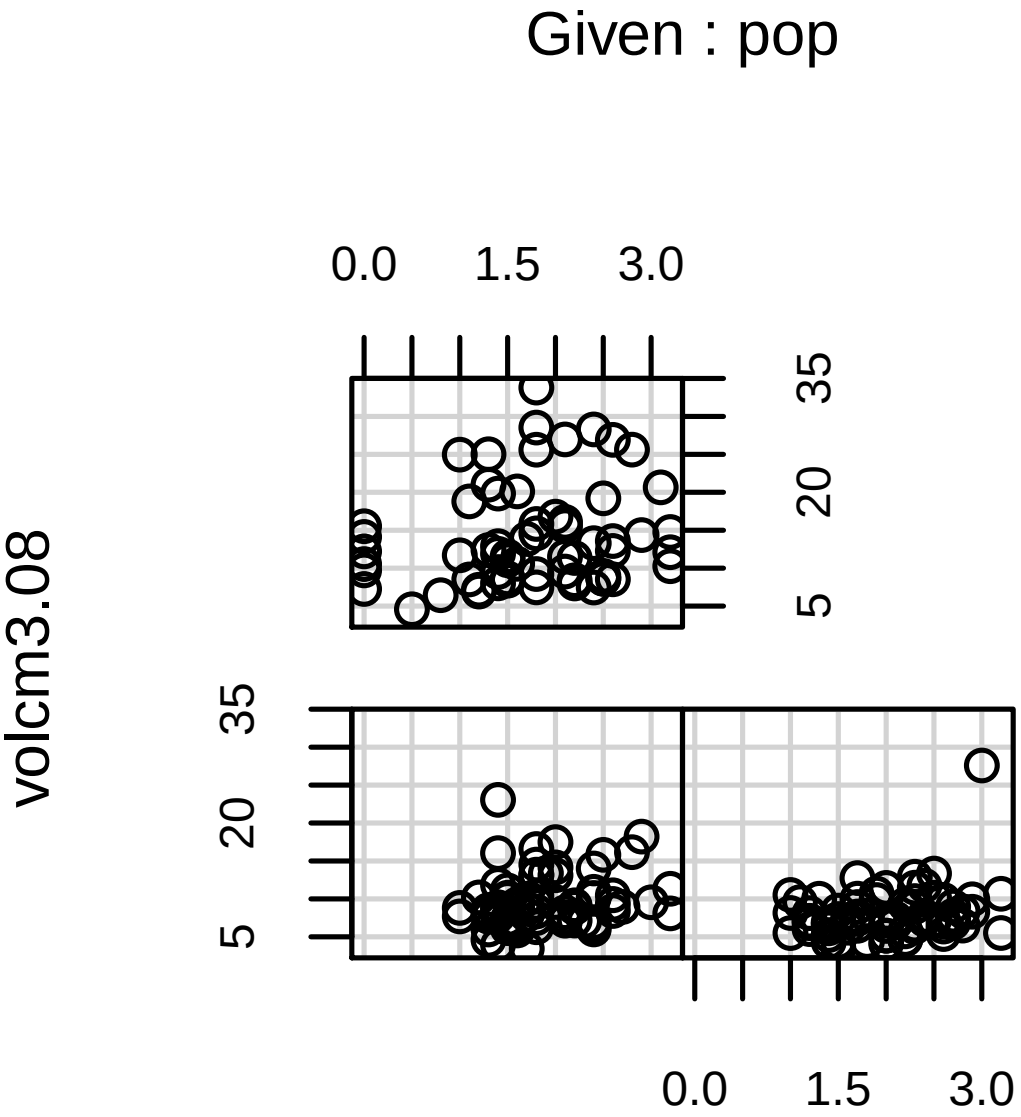
La boîte à moustache permet de visualiser les grands traits caractéristiques d'une distribution.

```
> boxplot(poids.08 ~ pop)
```



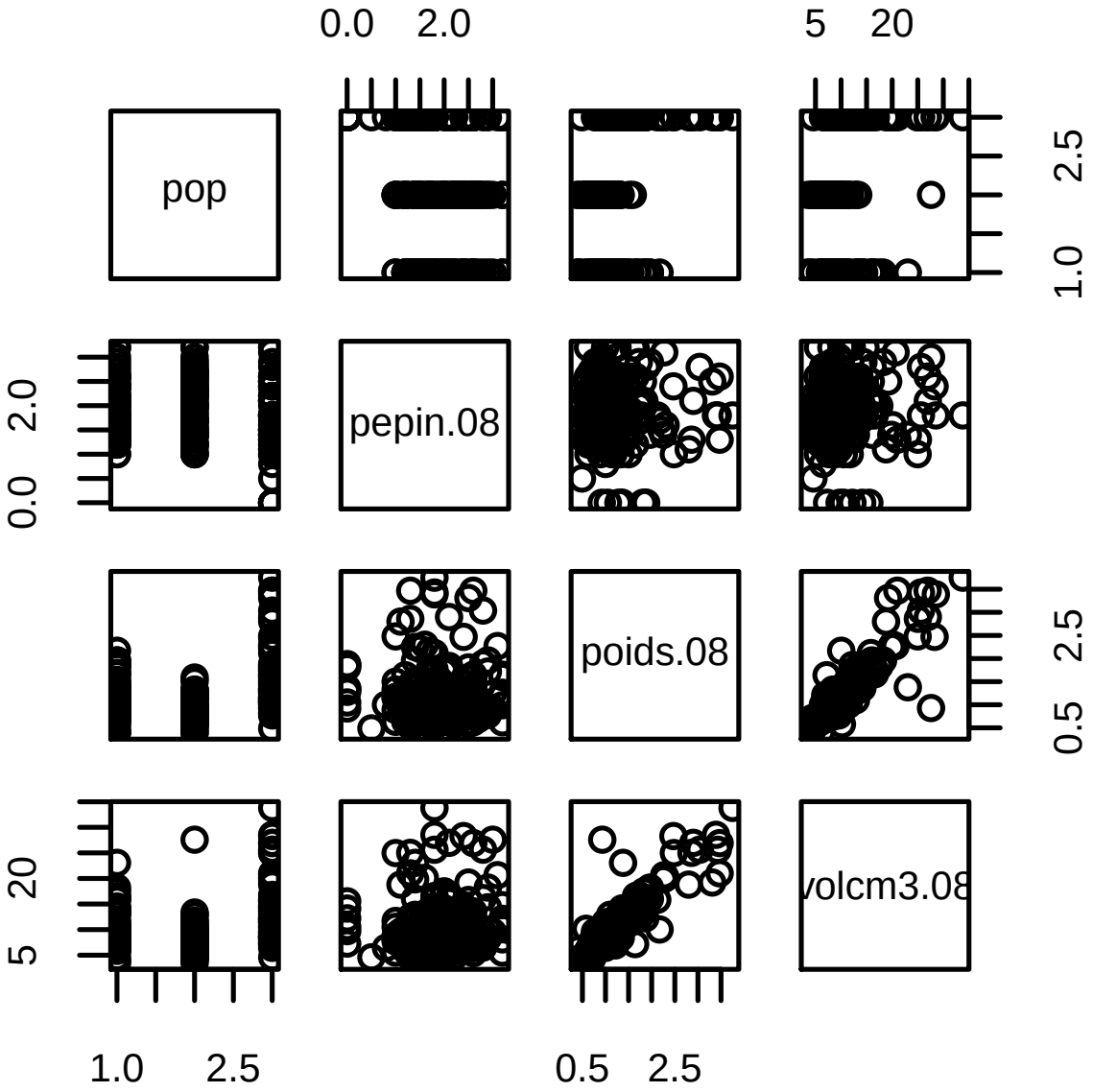
Graphe conditionné par une variable

```
> coplot(volcm3.08 ~ pepin.08 | pop, show.given = FALSE)
```



Graphes pair à pair

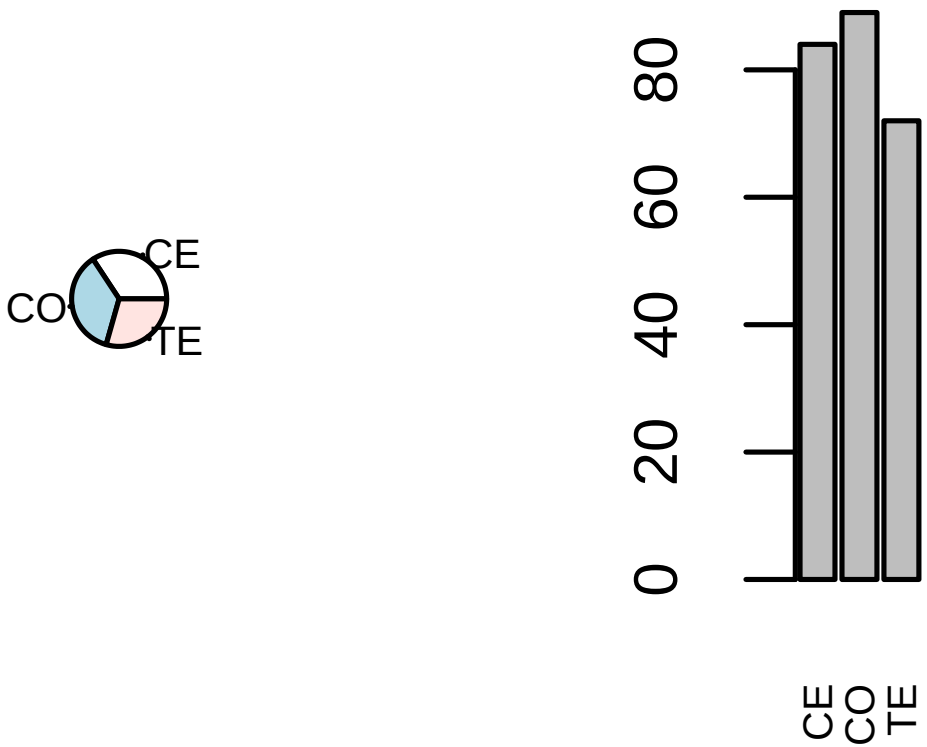
```
> pairs(vigne)
```



Camembert et diagramme en barres

Couplés à la commande `table` Le diagramme en barres et le graphe en camembert permettent de visualiser le découpage d'une population en donnée catégorielle.

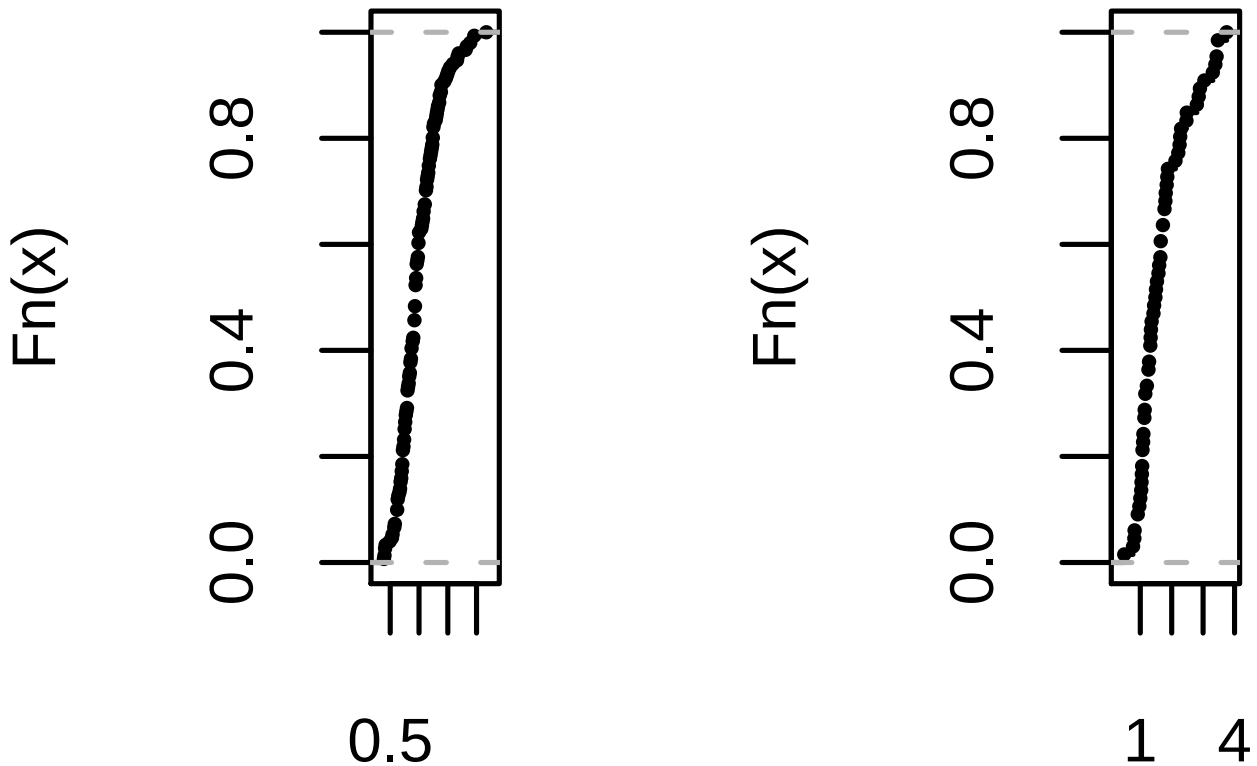
```
> par(mfrow = c(1, 2))
> pie(table(pop))
> barplot(table(pop), las = 3)
```



Fonction de répartition empirique

`ecdf` crée un objet qui peut être tracé avec `plot`.

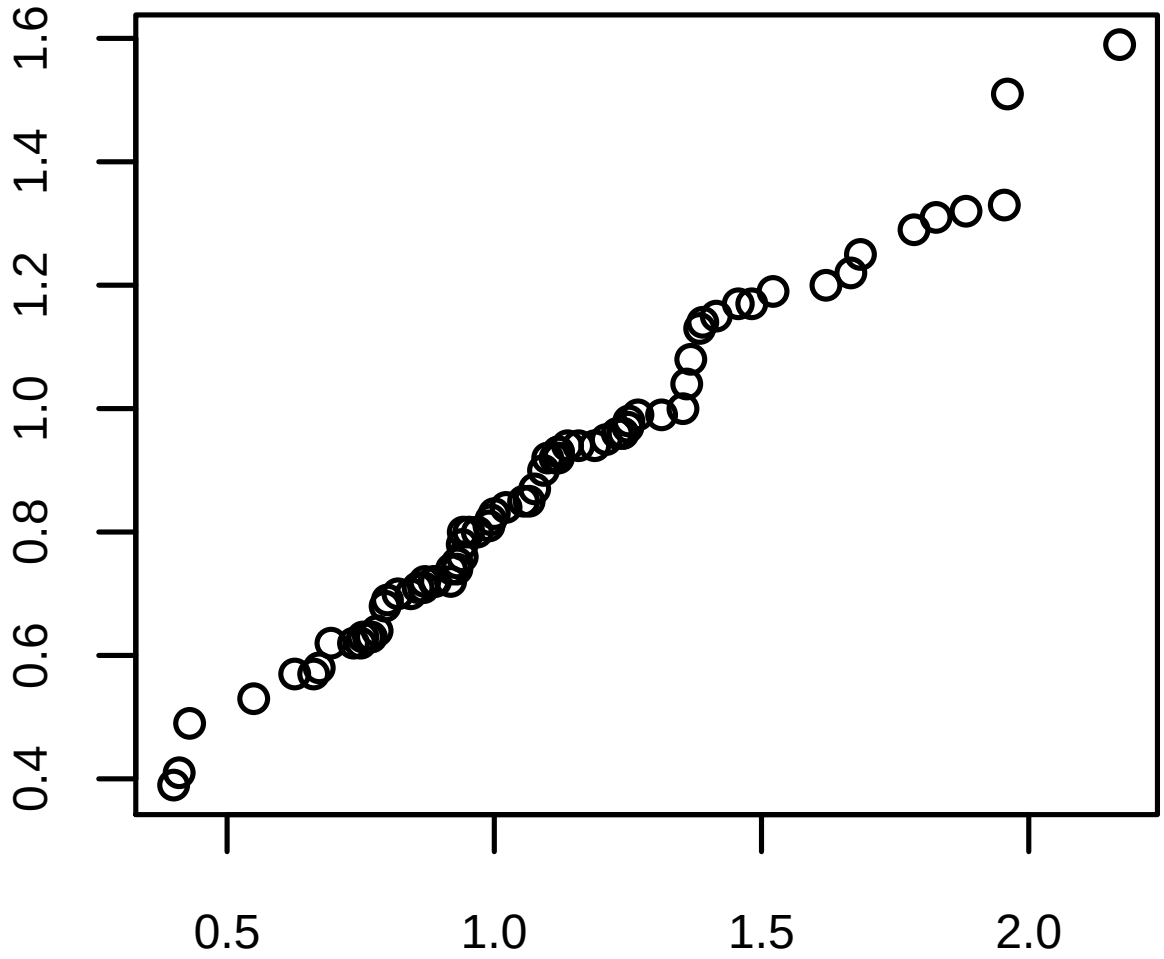
```
> par(mfrow = c(1, 2))
> plot(ecdf(poids.08[pop != "TE"]))
> plot(ecdf(poids.08[pop == "TE"]))
```



Comparaison de distribution

Pour comparer visuellemnt deux distributions, la manière la plus efficace est le graphe quantile/quantile (qui doivent correspondre si les distributions sont proches.)

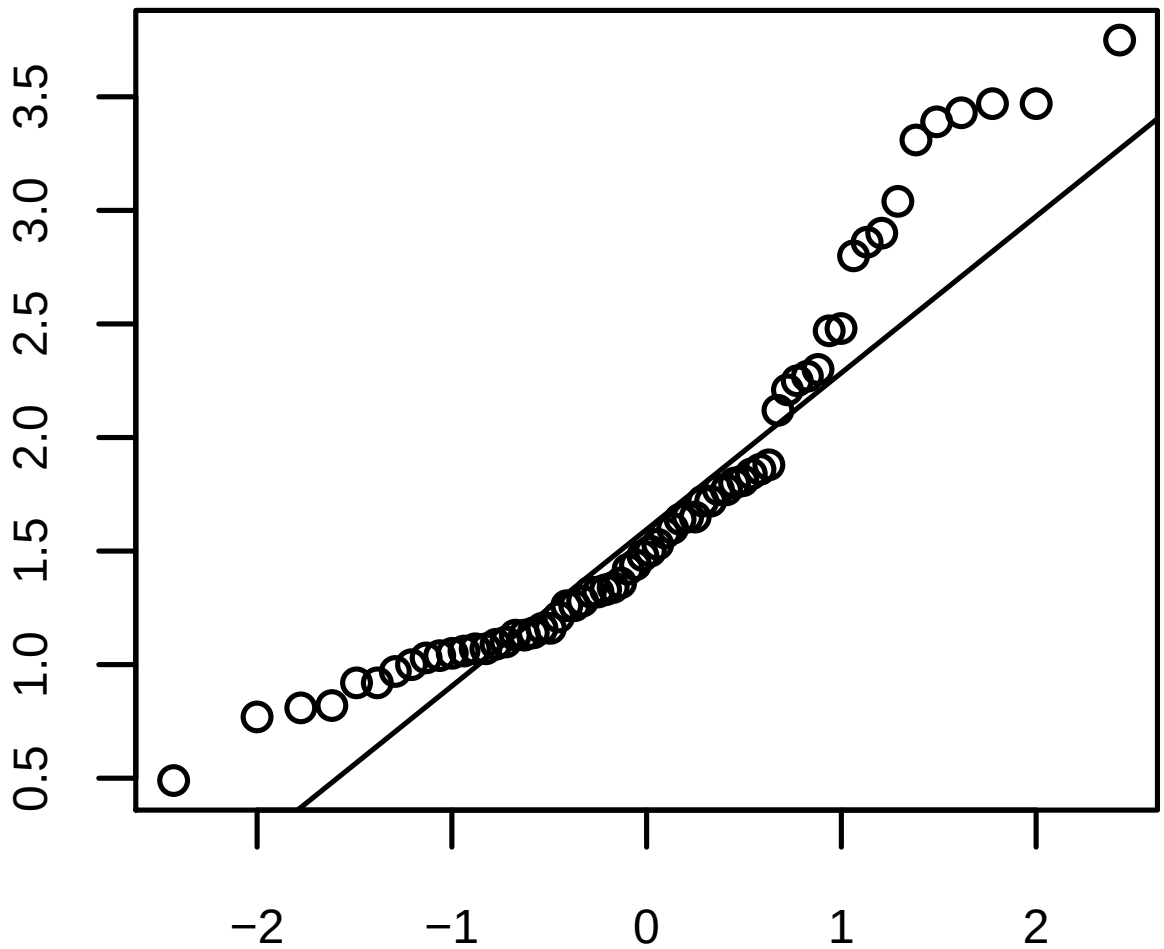
```
> qqplot(poids.08[pop == "CE"], poids.08[pop == "CO"])
```



Normalité d'une distribution

Une distribution est-elle normale ? Avant d'y répondre par un test, les commandes `qqnorm/qqline` donne une indication.

```
> qqnorm(poids.08[pop == "TE"])
> qqline(poids.08[pop == "TE"])
```



La loi des grands nombres

Théorème

Soit $(X_i)_{i=1,\dots,n}$ une suite de variables aléatoires indépendantes de même loi et soit S_n la somme de ces variables. Alors,

$$\frac{S_n}{n} \xrightarrow[p.s]{n\to\infty} \mathbb{E}[X].$$

Conséquence théorique

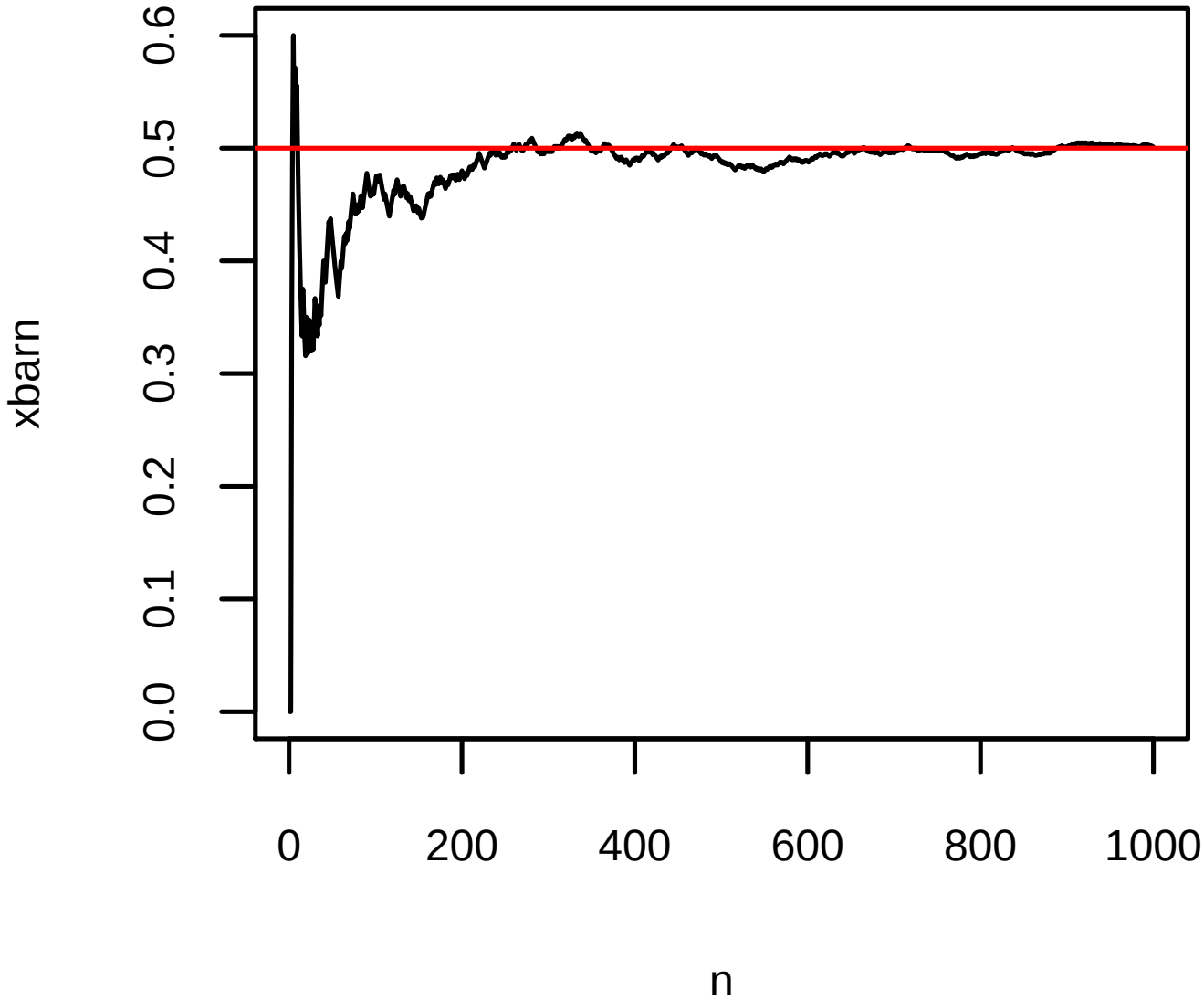
La moyenne empririque $\bar{X} = S_n/n$ est un estimateur fortement convergent de l'espérance d'une loi.

Conséquence pratique

C'est le fondement de la plupart des simulations numériques en statistiques.

Illustration

L'exemple passe-partout (mais parlant) : simuler l'issue d'un tirage de pile ou face, et observer l'évolution.



Le théorème de la limite centrale

Théorème

Soit $(X_i)_{i=1,\dots,n}$ une suite de variable aléatoire indépendante de même loi, d'espérance μ et de variance σ^2. Alors,

$$\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \xrightarrow[\mathcal{L}]{n\to\infty} \mathcal{N}(0,1).$$

Conséquences

- L'écart à la moyenne suit une loi normale lorsque l'on observe beaucoup d'individus, quelque soit la loi de la variable observée.
- C'est un résultat portant sur un indice global de la popualtion pas sur les individus ! ! !

Illustration

Protocole

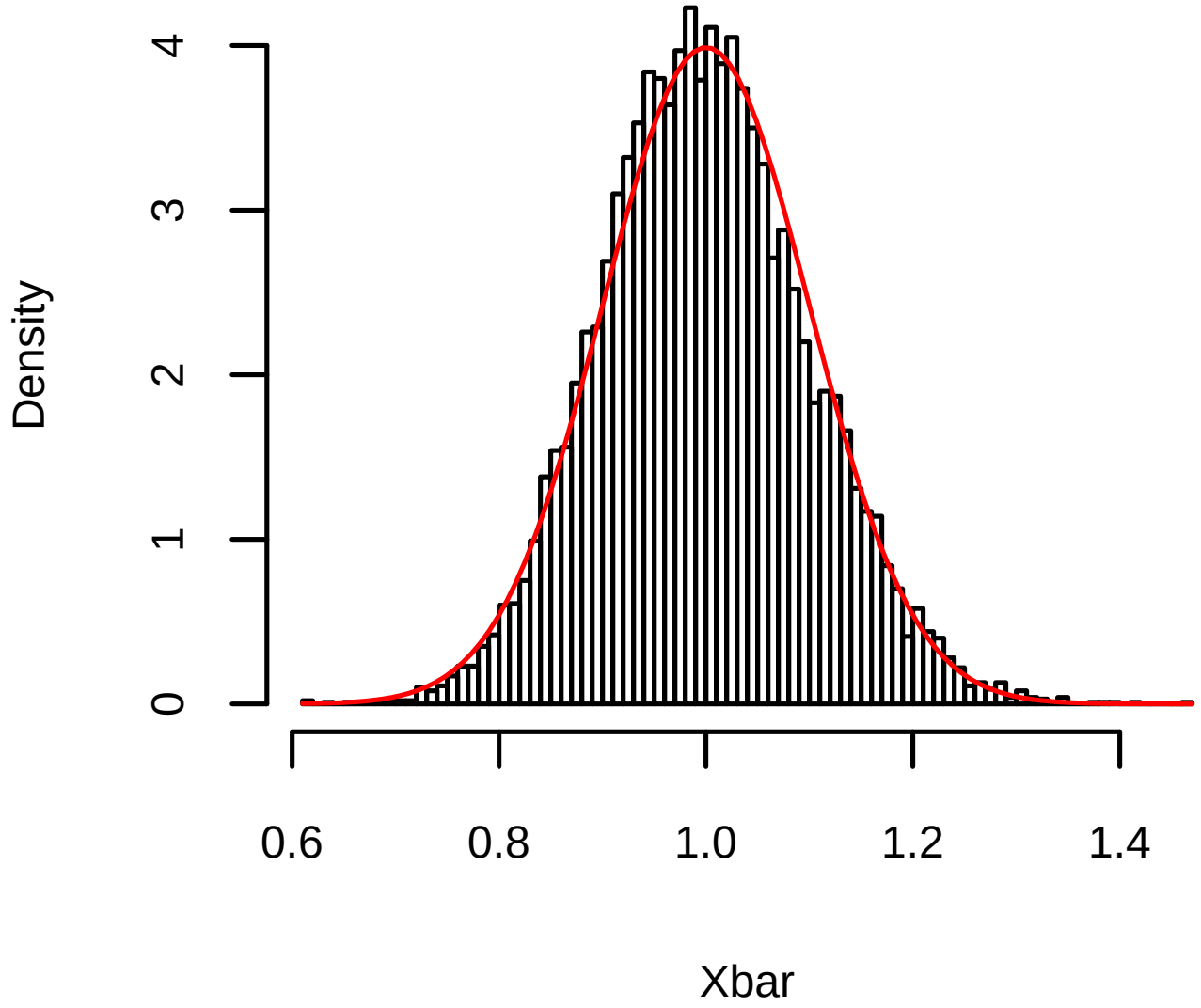
- On simule n réalisations d'une variable exponetielle,
- On calcule la valeur observés de la moyenne empirique,
- On répète m fois cette expérience,
- On trace l'histogramme des m observations de $\bar{X}_n$.

En R (compact !), ca donne :

```
> n <- 100
> m <- 10000
> Xi <- matrix(rexp(n * m), n, m)
> Xbar <- colMeans(Xi)
> hist(Xbar, nclass = 100)
```

Illustration : graphique

```
> hist(Xbar, nclass = 100, prob = TRUE, main = "")
> curve(dnorm(x, mean = 1, sd = 1/sqrt(n)), add = TRUE, col =
```



Le théorème de Cochran

Théorème

Soit $(X_i)_{i=1,\dots,n}$ une série de variables aléatoires indépendantes, de même loi, d'espérance μ et de variance σ^2. Alors, lorsque $n \to \infty$,

$$\sum_{i=1}^{n}(X_i - \bar{X}_n)^2 \sim \sigma^2 \chi^2_{n-1},$$

ce qu'on écrit également

$$n\hat{\sigma}^2 \sim \sigma^2 \chi^2_{n-1},$$

où $\hat{\sigma}^2$ est la variance empirique.

Test de l'espérance d'un caractère (1)

Problème

Soit X un caractère d'intérêt d'une population $\mathcal{P}$ de loi, d'espérance μ et de variance σ^2 inconnue. On propose une valeur μ_0 de l'espérance, et on teste sa validité en observant un échantillon i.i.d $(X_i)_{i=1,\dots,n}$:

$$\begin{cases} H_0 : & \mu = \mu_0, \\ H_1 : & \mu = \mu_1, \text{ avec } \mu_1 \neq \mu_0. \end{cases}$$

Par le théorème central limite et le théorème de Cochran, on montre que

$$\frac{\bar{X}_n - \mu}{\hat{\sigma}/\sqrt{n}} \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{T}(n-1),$$

où $\mathcal{T}(n-1)$ est une loi de Student à $n-1$ degrés de liberté.

Test de l'espérance d'un caractère (2)

Si on teste

$$\begin{cases} H_0 : & \mu = \mu_0, \\ H_1 : & \mu = \mu_1, \text{ avec } \mu_1 > \mu_0, \end{cases}$$

alors, au niveau de confiance α,

$$\text{on rejette } H_0 \ \alpha \text{ si } \bar{x}_n > \mu_0 + \frac{\hat{\sigma}}{\sqrt{n}} u_{1-\alpha}.$$

Définition

La p–valeur ou degré de significativité donne le risque que l'on prend à choisir H_1 plutôt que H_0. Si elle est supérieure à α, on conserve H_0 et on rejette sinon.

Test du paramètre d'espérance avec R

Exemple

(Sirop contre la toux) La notice d'un sirop contre la toux indique comme valeur de référence pour la moyenne m_0 de l'agent actif $40g$/litre. Le contrôleur de la fabrication décidera d'arrêter provisoirement la production si la moyenne m inconnue est strictement inférieure à cette valeur de référence.
Le contrôleur de la fabrication prélève de manière indépendantes 9 bouteilles au hasard dans la production et mesure la quantité d'agent actif.
Conclusion ?

```
> x <- c(38.7, 39.6, 37.9, 40.6, 40.5, 37.7, 41.2, 37.5, 39.1)
> t.test(x, mu = 40, alternative = "less")

        One Sample t-test

data:  x
t = -1.7586, df = 8, p-value = 0.05835
alternative hypothesis: true mean is less than 40
95 percent confidence interval:
     -Inf 40.04593
sample estimates:
mean of x
     39.2
```

Test de Student d'égalité des espérance

Problème

Soit X et Y deux caractères d'intérêt d'une population $\mathcal{P}$ de loi, d'espérance μ_1 et μ_2 et de variance commune σ^2, toutes inconnues. On propose de tester l'égalité des espérances en observant deux échantillons i.i.d $(X_i)_{i=1,\dots,n}$ et $(Y_i)_{i=1,\dots,m}$:

$$\begin{cases} H_0 : & \mu_1 = \mu_2, \\ H_1 : & \mu_1 \neq \mu_2, \end{cases}$$

Par le théorème central limite et le théorème de Cochran, on montre que

$$\frac{\bar{X}_n + \bar{Y}_m - (\mu_1 + \mu_2)}{s^* \sqrt{n^{-1} + m^{-1}}} \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{T}(n+m-2),$$

où $S^{*2} = \frac{(n-1){S_X^*}^2 + (m-1){S_Y^*}^2}{n+m-2}$

Test de Student d'égalité des espérances sous R

Testons l'égalité des poids entre espèces (CE,CO) et TE chez la vigne :

```
> t.test(poids.08[pop != "TE"], poids.08[pop == "TE"])

        Welch Two Sample t-test

data:  poids.08[pop != "TE"] and poids.08[pop == "TE"]
t = -7.1015, df = 75.698, p-value = 5.751e-10
alternative hypothesis: true difference in means is not equal
95 percent confidence interval:
 -0.9142257 -0.5137193
sample estimates:
mean of x mean of y
0.9949669 1.7089394
```

Le test de Fisher

Problème

Soit X et Y deux caractères d'intérêt d'une population $\mathcal{P}$ de lois et de variances σ_1^2 et σ_2^2 inconnues. On propose de tester l'égalité des variances en observant deux échantillons i.i.d $(X_i)_{i=1,\dots,n}$ et $(Y_i)_{i=1,\dots,m}$:

$$\begin{cases} H_0 : & \sigma_1 = \sigma_2, \\ H_1 : & \sigma_1 \neq \sigma_1, \end{cases}$$

Par le théorème central limite et le théorème de Cochran, on montre que

$$\frac{{S_X^*}^2/\sigma_1^2}{{S_Y^*}^2/\sigma_2^2} \xrightarrow[n\to\infty]{\mathcal{L}} \mathcal{F}(n-1, m-1).$$

Le test d'égalité des variances sous R

Testons l'égalité des variances des poids entre espèces (CE,CO) et TE chez la vigne :

```
> var.test(poids.08[pop != "TE"], poids.08[pop == "TE"])

        F test to compare two variances

data:  poids.08[pop != "TE"] and poids.08[pop == "TE"]
F = 0.1846, num df = 150, denom df = 65, p-value < 2.2e-16
alternative hypothesis: true ratio of variances is not equal
95 percent confidence interval:
 0.1198802 0.2746481
sample estimates:
ratio of variances
         0.1845923
```

Formulation générale

$$y = \beta_0 + \sum_{i=1}^{p} X_i \beta_i + \varepsilon,$$

où

- y est la variable à expliquer ou *réponse*,
- $(X_i)_{i=1,\dots,p}$ sont les variable explicatives ou *prédicteurs*,
- β_0 est le biais (à estimer)
- $(\beta_i)_{i=0,\dots,p}$ sont les paramètres à estimer,
- $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ sont les résidus (ce que le modèle n'explique pas)

Quelques formes du modèle linéaire

Remarque

le modèle est linéaire au sens d'une combinaison linéaire des paramètres ! Les modèles suivants sont des modèles linéaires :

1. *Polynomial*

$$y = \beta_0 + X_1\beta_1 + X_2^2\beta_2 + \varepsilon$$

2. *Cobb-Douglas*

$$\ln y = \beta_0 + \beta_1 \ln X_1 + \varepsilon$$

3. *Logistique*

$$\ln \frac{y}{1-y} = \beta_0 + \beta_1 X_1 + \varepsilon$$

$\rightsquigarrow$ Un des modèles les plus puissants et les plus utilisés de la statistique

Le modèle linéaire sous R (1)

La fonction `lm`

Pour « linear model » : s'utilise avec le concept de *formule* :

- 1 variable explicative avec intercept (régression linéaire)

```
lm(y ~ X + 1),
```

- 2 variables explicatives sans intercept

```
lm(y ~ X1 + X2 - 1),
```

- 1 variable factorielle (anova à 1 facteur)

```
lm(y ~ A),
```

- 2 variables factorielles (anova à 2 facteurs sans interactions)

```
lm(y ~ A+B),
```

- 1 variable factorielles (anova à 2 facteurs avec interactions)

```
lm(y ~ A*B).
```

Exemple

On veut expliquer la variable de poids par la population d'origine et le volume de la baie. Plusieurs modèles sont possibles, par exemple

1. sans interactions

```
> m1 <- lm(poids.08 ~ volcm3.08 + pop)
> anova(m1)

Analysis of Variance Table

Response: poids.08
           Df Sum Sq Mean Sq F value    Pr(>F)
volcm3.08   1 62.069  62.069  903.80 < 2.2e-16 ***
pop         2  1.766   0.883   12.86 5.632e-06 ***
Residuals 197 13.529   0.069
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2. avec interactions

```
> m2 <- lm(poids.08 ~ volcm3.08 * pop)
> anova(m2)

Analysis of Variance Table

Response: poids.08
               Df Sum Sq Mean Sq  F value    Pr(>F)
volcm3.08       1 62.069  62.069 1088.990 < 2.2e-16 ***
pop             2  1.766   0.883   15.495 5.684e-07 ***
volcm3.08:pop   2  2.415   1.207   21.183 4.733e-09 ***
Residuals     195 11.114   0.057
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Comparer plusieurs modèles

La fonction `anova(modele1,modele2,...)`.

```
> anova(m1, m2)

Analysis of Variance Table

Model 1: poids.08 ~ volcm3.08 + pop
Model 2: poids.08 ~ volcm3.08 * pop
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    197 13.529
2    195 11.114  2    2.4147 21.183 4.733e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

$\rightsquigarrow$ le modèle avec interactions est plus intéressant

plus de détails avec Bernard Prum.