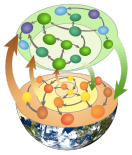


Séance 5: La visualisation des données

BIO 500 - Méthodes en écologie computationnelle

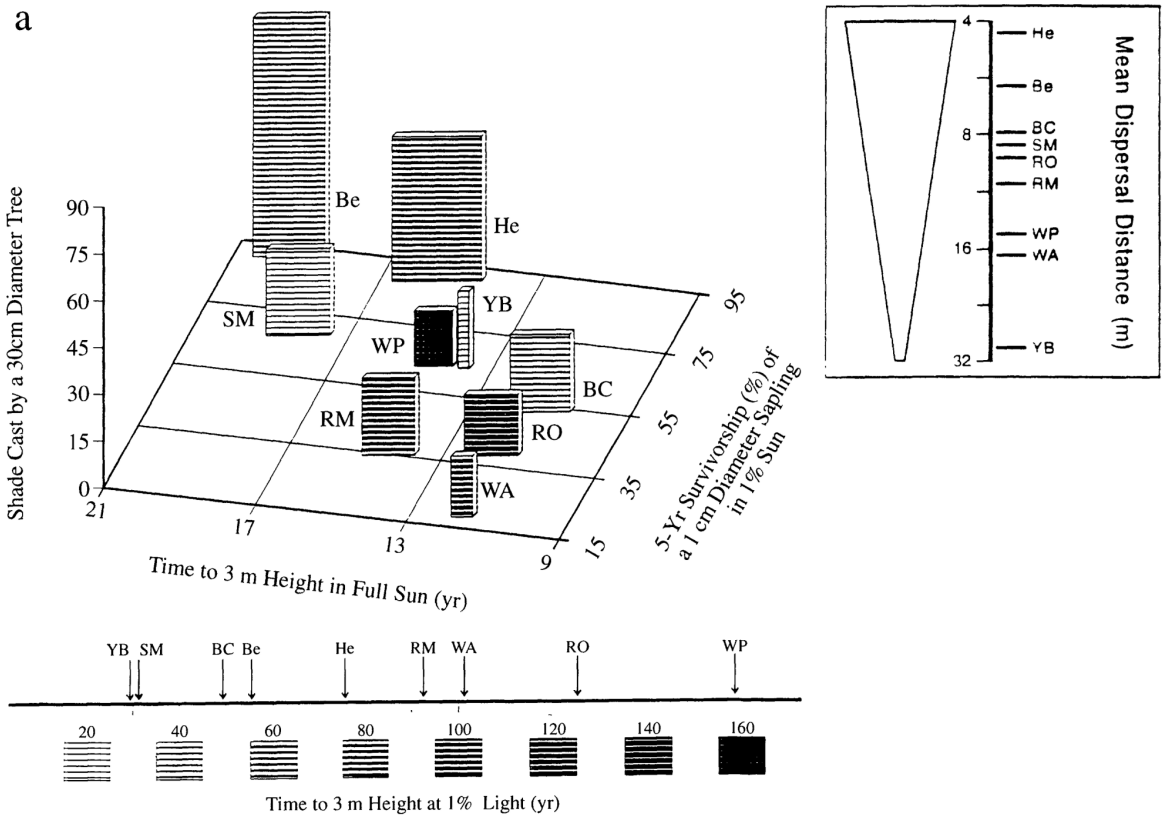
Dominique Gravel & Steve Vissault
Laboratoire d'écologie intégrative

Séance 5

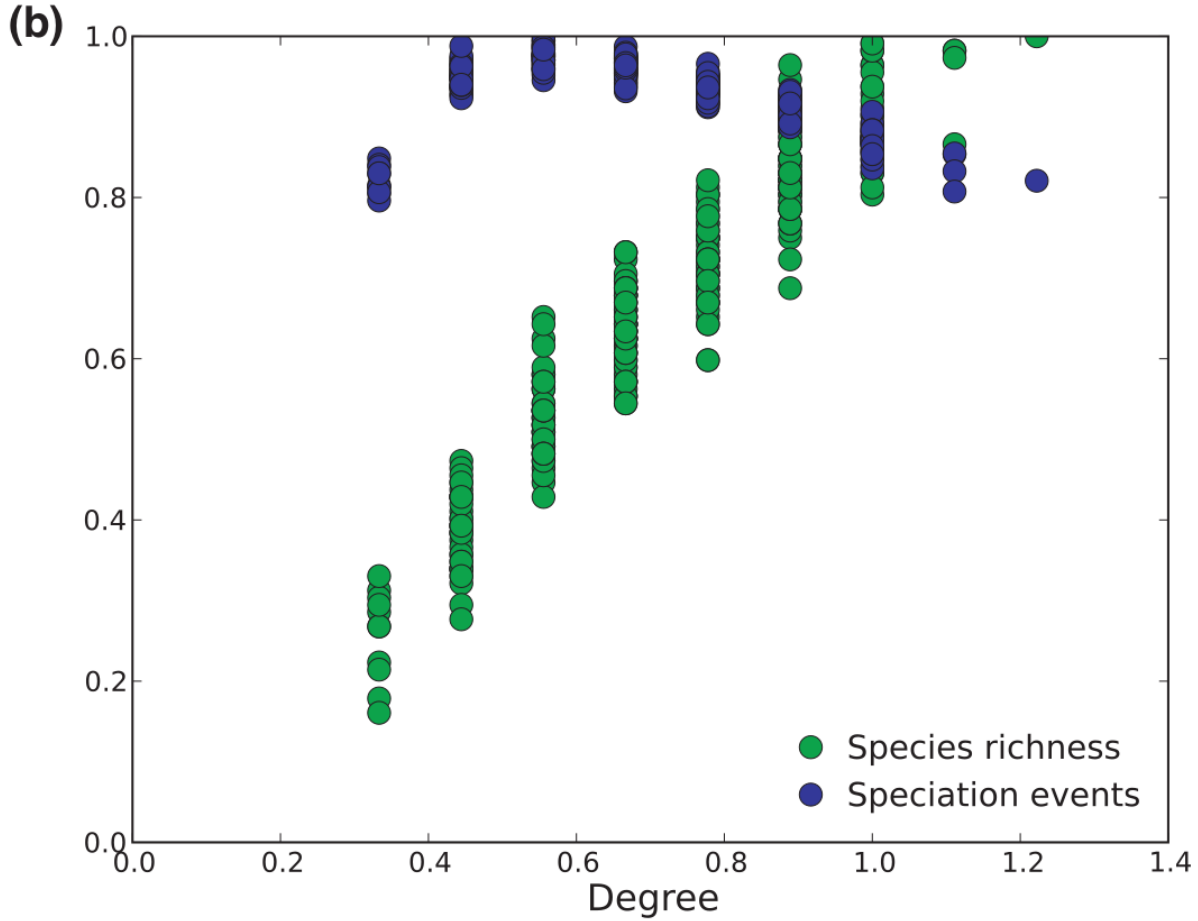
- ✓ Ces diapositives sont disponibles en **version web** et en **PDF**.
- ✓ L'ensemble du matériel de cours est disponible sur la page du portail **moodle**.
- ✓ Vous trouverez du matériel supplémentaire dans le **cours** de **Kevin Cazelles** et **Nicolas Casajus** lors d'un atelier de communication visuelle du CSBQ.
- ✓ Certaines diapositives sont également extraites de la présentation de **David Taylor**

Qu'est-ce qui fait une bonne figure ?

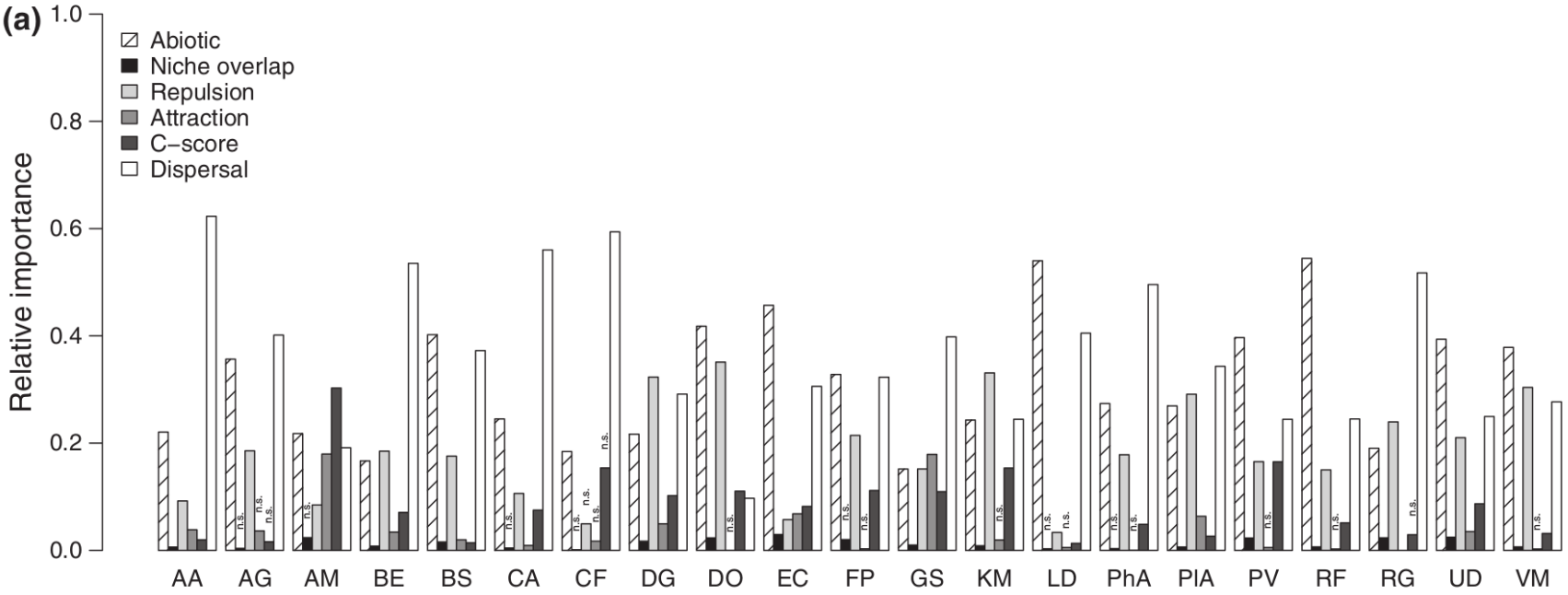
Trop d'information



Non respect des normes graphiques



Abus de symboles et de couleurs



L'art graphique

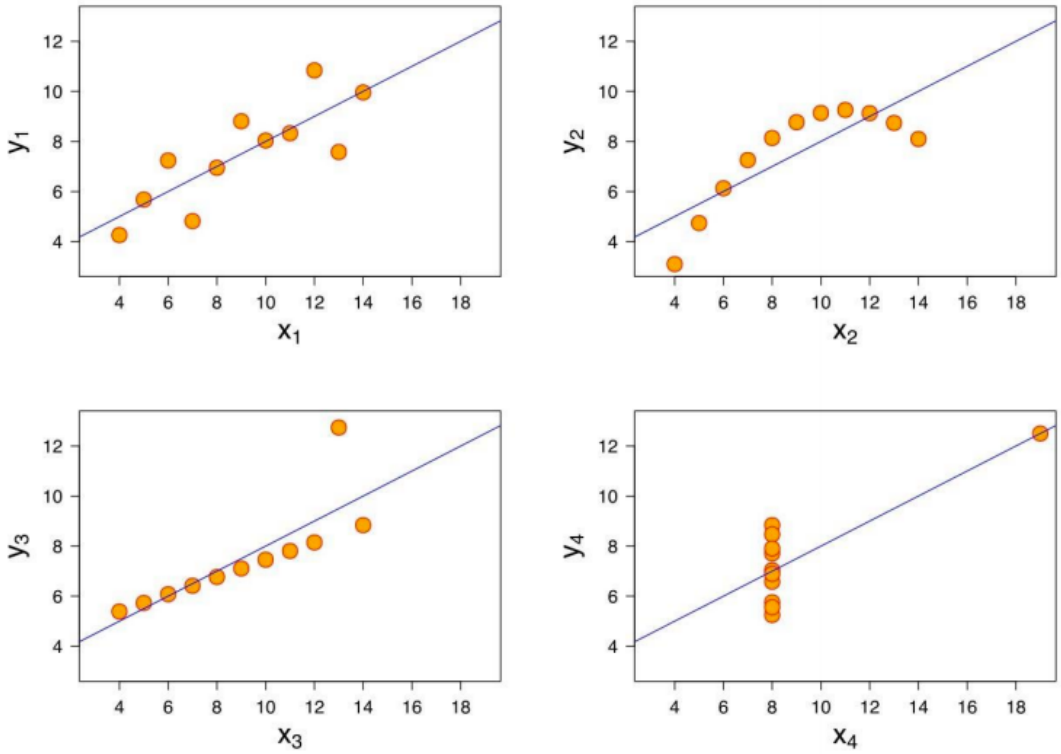
L'importance des graphiques

- ✓ Synthétiser l'information.
- ✓ Communiquer plus efficacement qu'un tableau.
- ✓ Explorer nos données par la visualisation.
- ✓ Présenter nos résultats et convaincre.

Explorer nos données par la visualisation

	Set A		Set B		Set C		Set D	
	X	Y	X	Y	X	Y	X	Y
0	10	8.04	10	9.14	10	7.46	8	6.58
1	8	6.95	8	8.14	8	6.77	8	5.76
2	13	7.58	13	8.74	13	12.74	8	7.71
3	9	8.81	9	8.77	9	7.11	8	8.84
4	11	8.33	11	9.26	11	7.81	8	8.47
5	14	9.96	14	8.10	14	8.84	8	7.04
6	6	7.24	6	6.13	6	6.08	8	5.25
7	4	4.26	4	3.10	4	5.39	19	12.50
8	12	10.84	12	9.13	12	8.15	8	5.56
9	7	4.82	7	7.26	7	6.42	8	7.91
10	5	5.68	5	4.74	5	5.73	8	6.89
mean	9.00	7.50	9.00	7.50	9.00	7.50	9.00	7.50
std	3.32	2.03	3.32	2.03	3.32	2.03	3.32	2.03
corr	0.82		0.82		0.82		0.82	
lin. reg.	$y = 3.00 + 0.500x$		$y = 3.00 + 0.500x$		$y = 3.00 + 0.500x$		$y = 3.00 + 0.500x$	

Explorer nos données par la visualisation

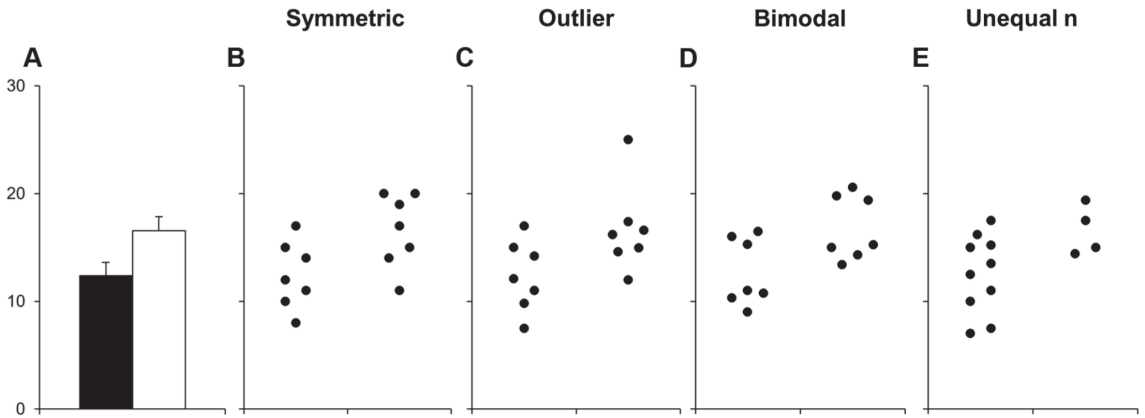


Communiquer par les graphiques

- ✓ Les graphiques sont généralement **plus efficaces à communiquer** un message/un résultat qu'un tableau.
- ✓ **Problème:** La représentation graphique peut parfois nous conduire à une **fausse interprétation**. L'idée est de transmettre une idée sans biaiser le lecteur.

Communiquer par les graphiques

Communiquer par les graphiques



Test			p value	
T-test: Equal var.	0.035	0.050	0.026	0.063
T-test: Unequal var.	0.035	0.050	0.026	0.035
Wilcoxon	0.054	0.073	0.128	0.103

Fig 1. Many different datasets can lead to the same bar graph. The full data may suggest different conclusions from the summary statistics. The means and SEs for the four example datasets shown in Panels B–E are all within 0.5 units of the means and SEs shown in the bar graph (Panel A). *p*-values were calculated in R (version 3.0.3) using an unpaired t-test, an unpaired t-test with Welch's correction for unequal variances, or a Wilcoxon rank sum test. In Panel B, the distribution in both groups appears symmetric. Although the data suggest a small difference between groups, there is substantial overlap between groups. In Panel C, the apparent difference between groups is driven by an outlier. Panel D suggests a possible bimodal distribution. Additional data are needed to confirm that the distribution is bimodal and to determine whether this effect is explained by a covariate. In Panel E, the smaller range of values in group two may simply be due to the fact that there are only three observations. Additional data for group two would be needed to determine whether the groups are actually different.

Communiquer par les graphiques

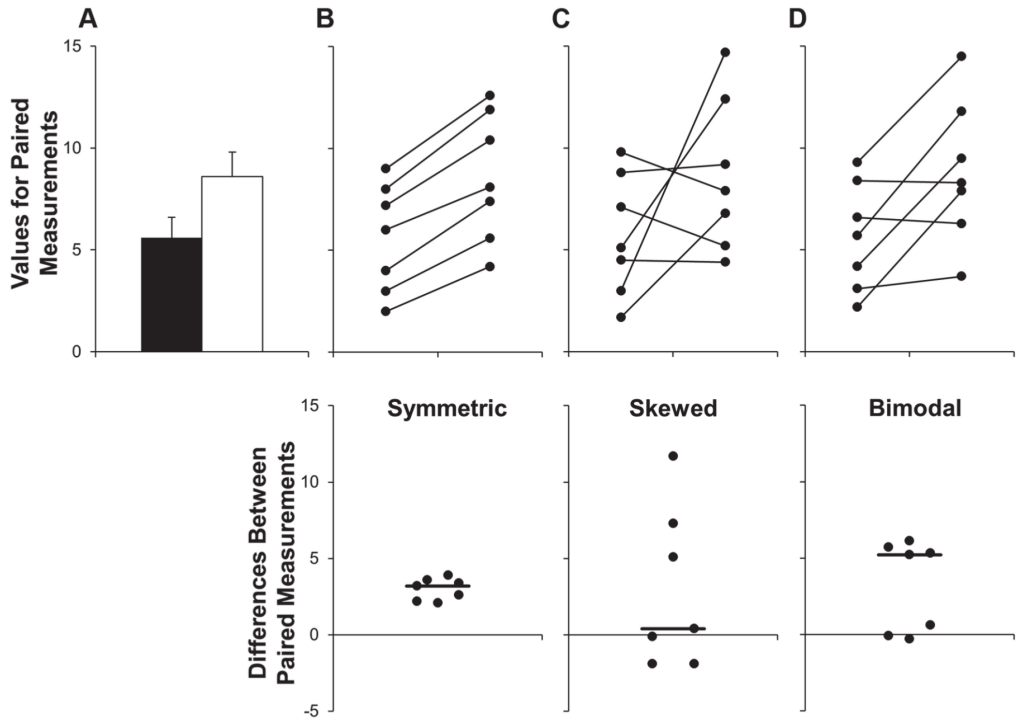


Fig 2. Additional problems with using bar graphs to show paired data. The bar graph (mean ± SE) suggests that the groups are independent and provides no information about whether changes are consistent across individuals (Panel A). The scatterplots shown in the Panels B–D clearly demonstrate that the data are paired. Each scatterplot reveals very different patterns of change, even though the means and SEs differ by less than 0.3 units. The lower scatterplots showing the differences between measurements allow readers to quickly assess the direction, magnitude, and distribution of the changes. The solid lines show the median difference. In Panel B, values for every subject are higher in the second condition. In Panel C, there are no consistent differences between the two conditions. Panel D suggests that there may be distinct subgroups of "responders" and "nonresponders."

Règles et composantes graphiques

Les composantes graphiques

- ✓ Les axes et échelles.
- ✓ Le titre de la figure.
- ✓ La légende
- ✓ Le **type de représentation des données**.

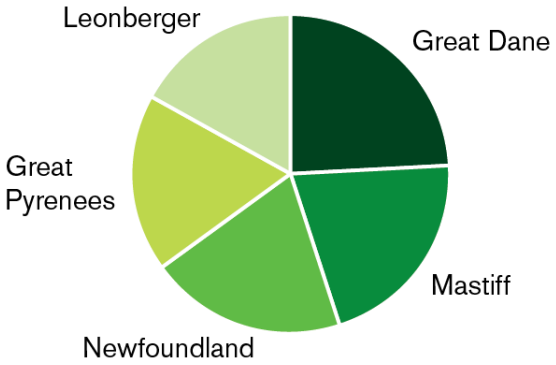
Les règles graphiques

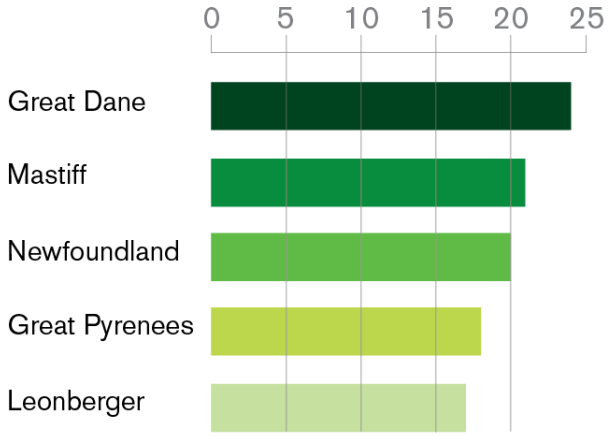
- ✓ Une figure doit renvoyer un seul message/résultat.
- ✓ Chaque élément d'une figure doit **aider à comprendre** ce message.
- ✓ **Choisir le bon type de représentation** permet de mettre en valeur plus facilement ce qui doit être illustré.
- ✓ **Attention aux normes graphiques**: Choix des couleurs, taille des caractères, épaisseur de la ligne, disposition des marges, cadrage etc.

Quelques conseils

✓ Ne pas **JAMAIS** utiliser de diagramme en pointe de tarte

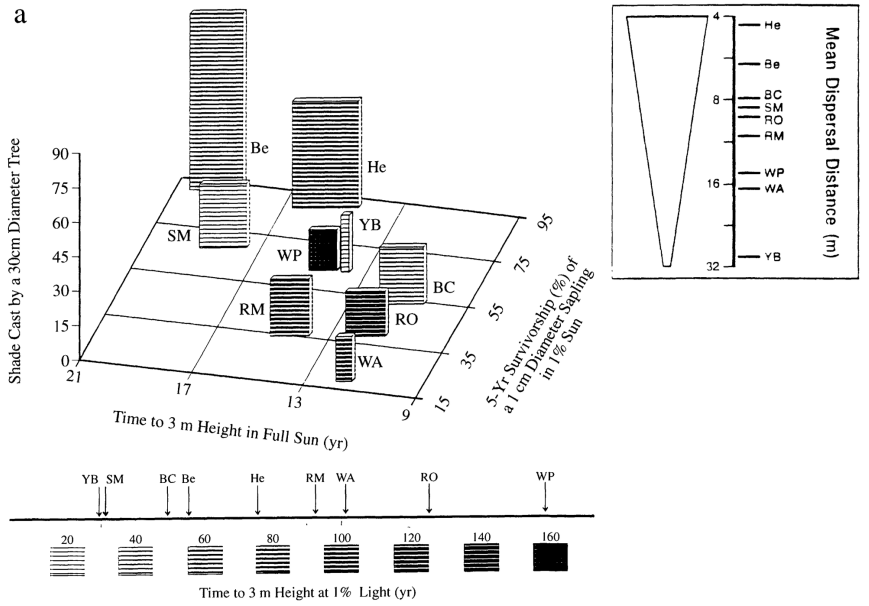
Pie vs. bar chart





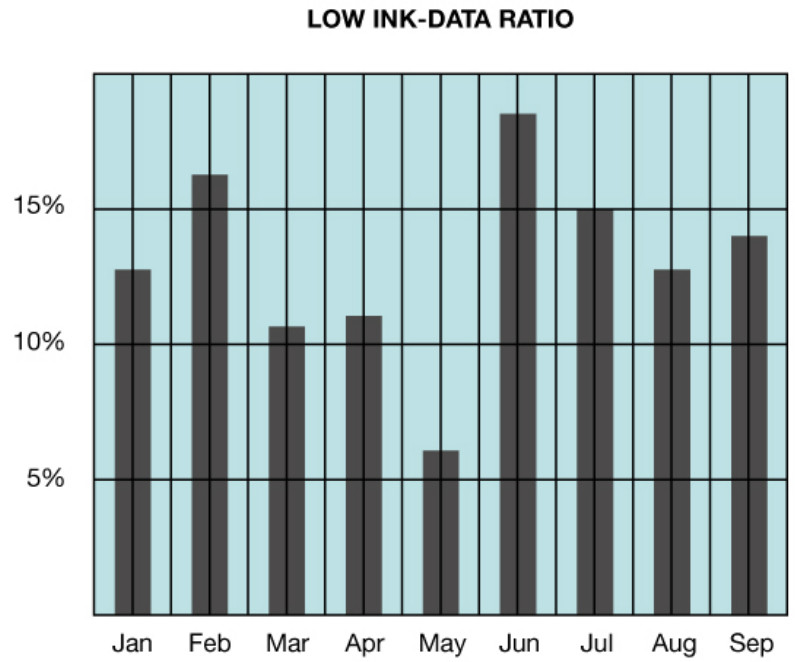
Quelques conseils

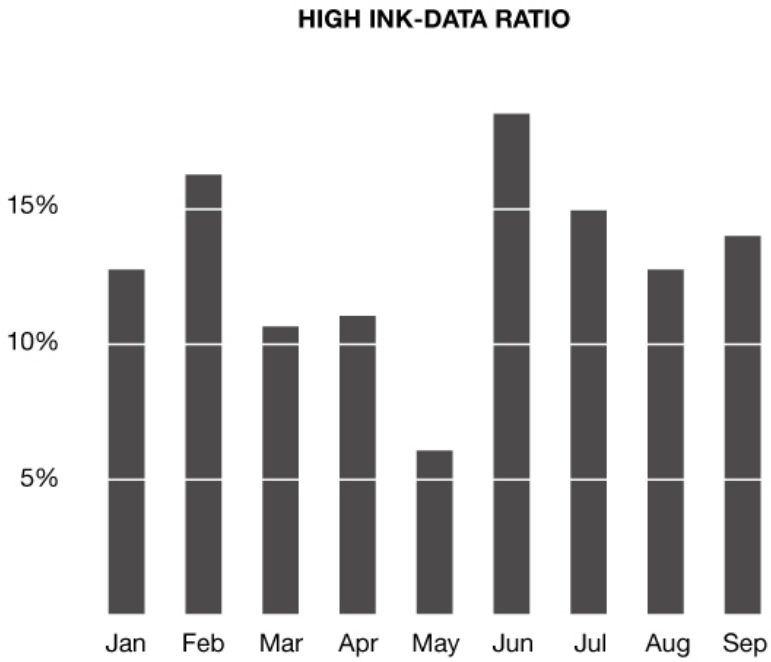
- ✓ Éviter les figures 3D.
- ✓ Limiter le nombre de dimensions (3 ou 4 dimensions max).
- ✓ La multi-dimensionnalité peut être gérée en:
  - Modifiant la forme et la la taille des points
  - Ajoutant des couleurs



Quelques conseils

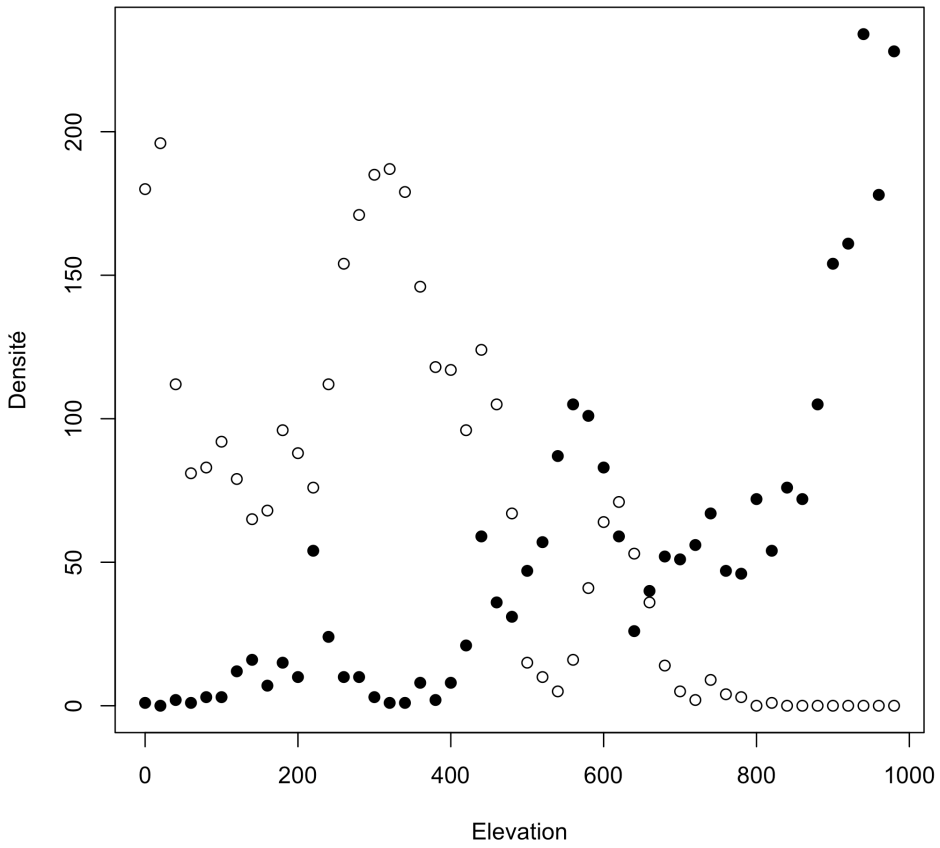
✓ Limiter le ratio encre/données afin de faciliter la lecture.



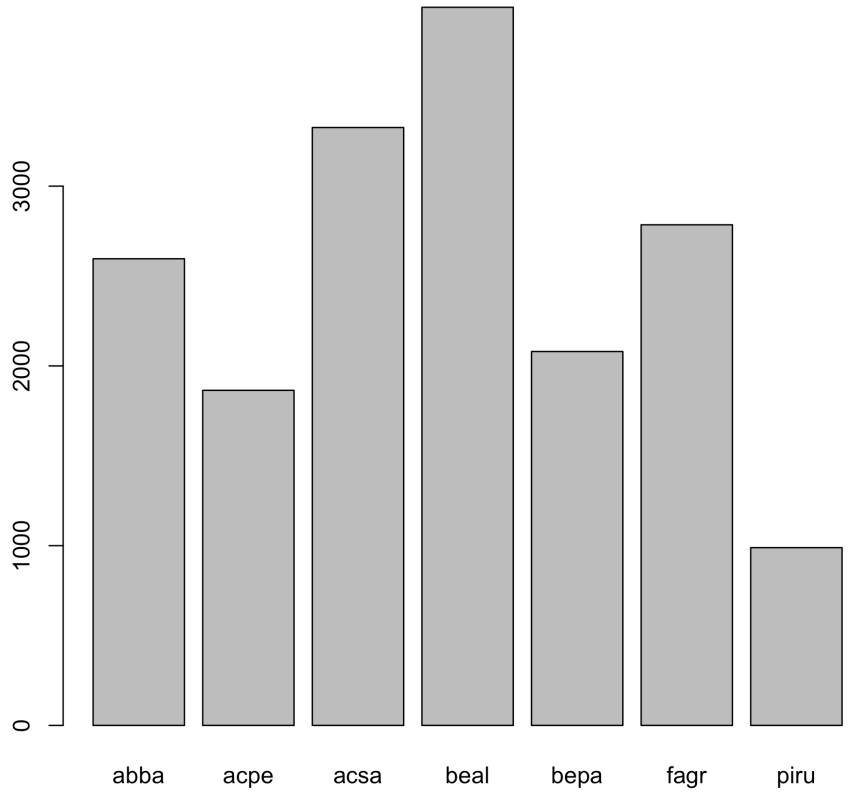


Types de figures

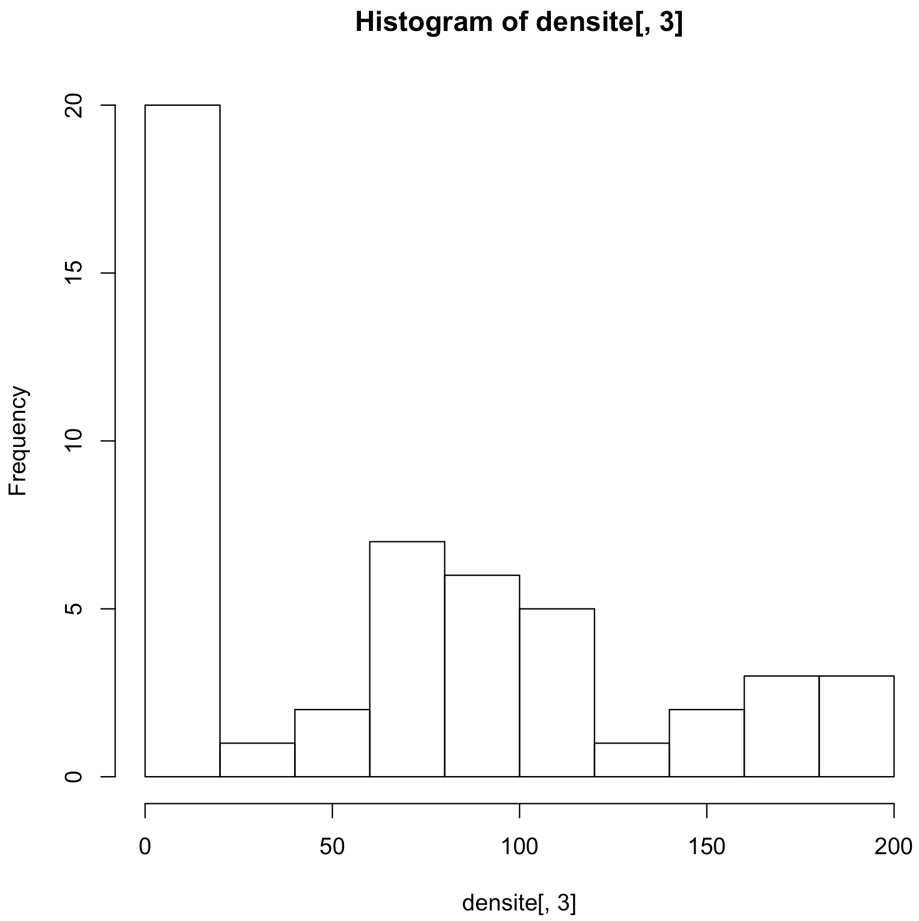
Diagramme de dispersion (Scatter plot)



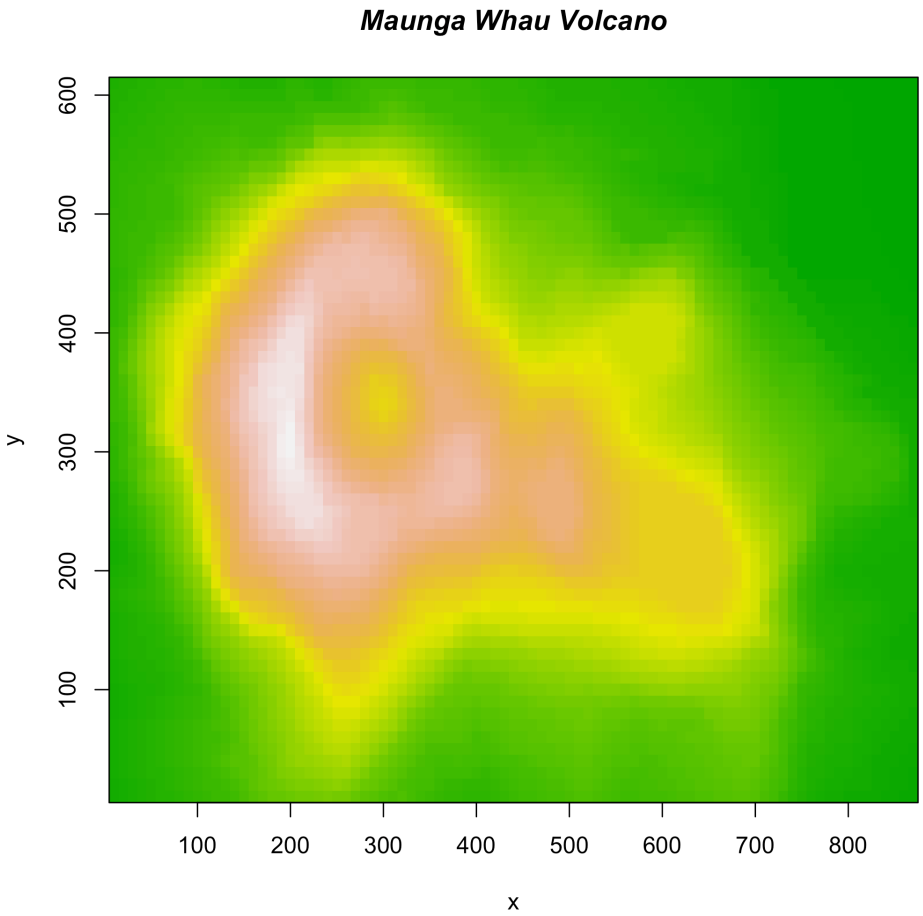
Diagrammes à bâtons (Bar plot)



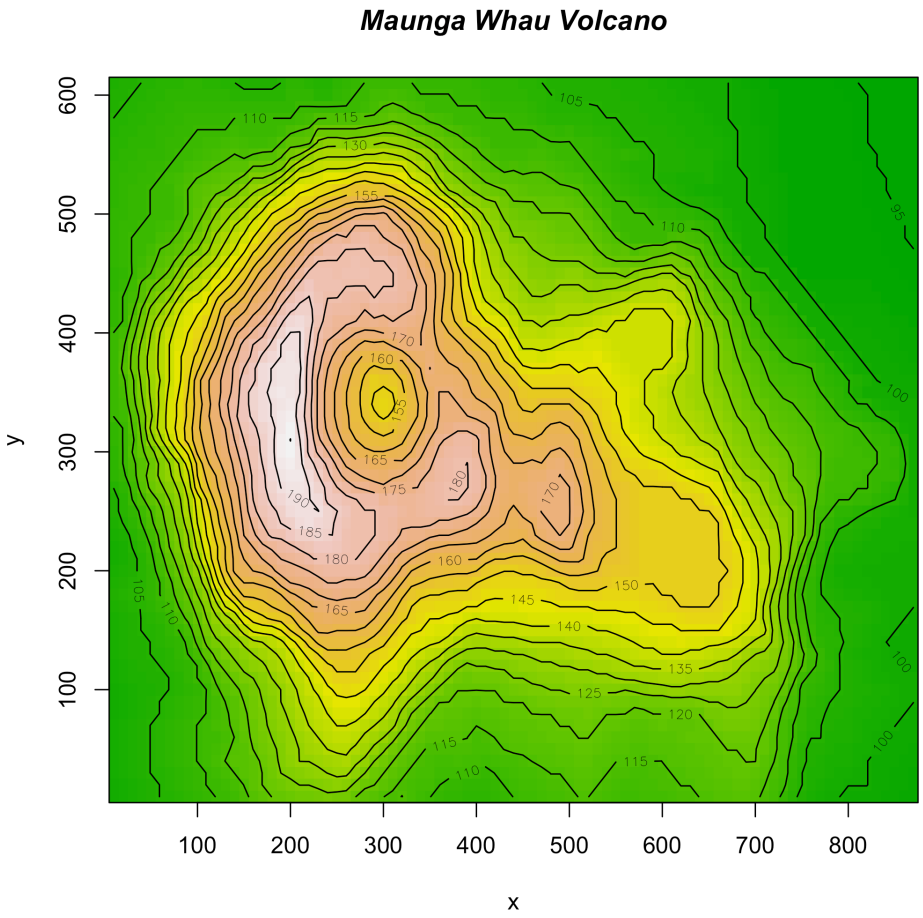
Histogrammes



Représentation 3-D



Lignes de contour



Faire une figure étape par étape avec R

Prépares les données adéquatement

- ✓ Habituellement un `data.frame` ou `une matrice`
- ✓ Une observation par ligne (format long)

Ouvrir une fenêtre graphique

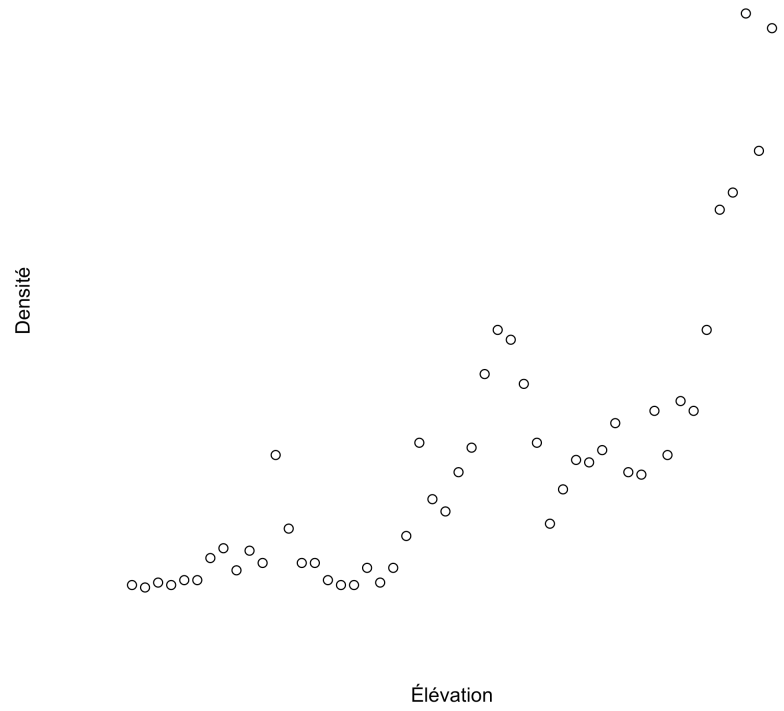
```
dev.new(width = 10, height = 7)
```

Fixer certains paramètres

```
# Fixer la largeur et la hauteur des marges
par(mar = c(5,6,2,1))

# Fixer le nombre de figures en colonnes et rangées
par(mfrow = c(1,1))
```

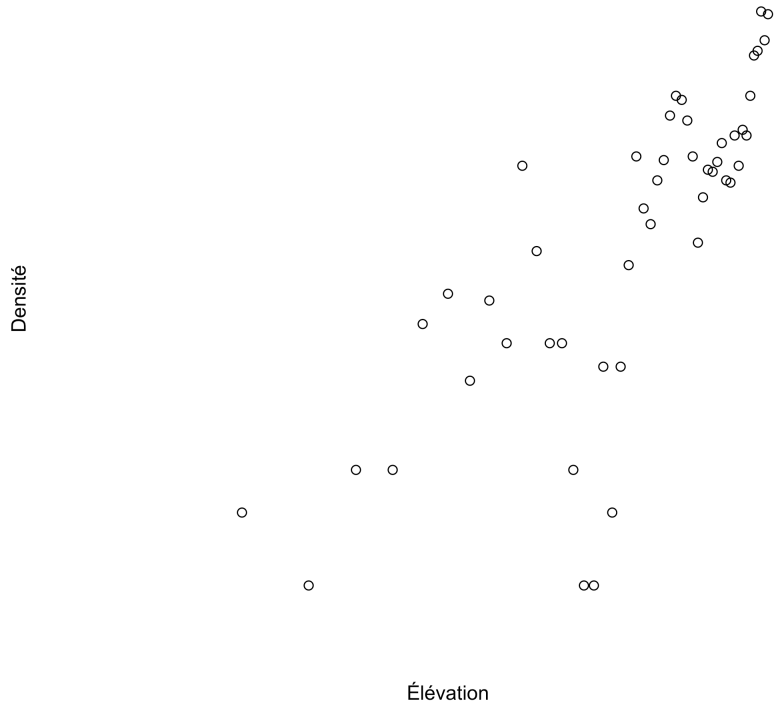
Démarrer une figure avec `plot()`

```
arbres <- read.csv2("donnees/arbres.csv")
densite <- table(arbres[,c(3,5)])
elevation <- as.numeric(row.names(densite))
plot(elevation, densite[,1], axes = FALSE,
      xlab = "Élévation", ylab = "Densité")
```



Échelles logarithmiques

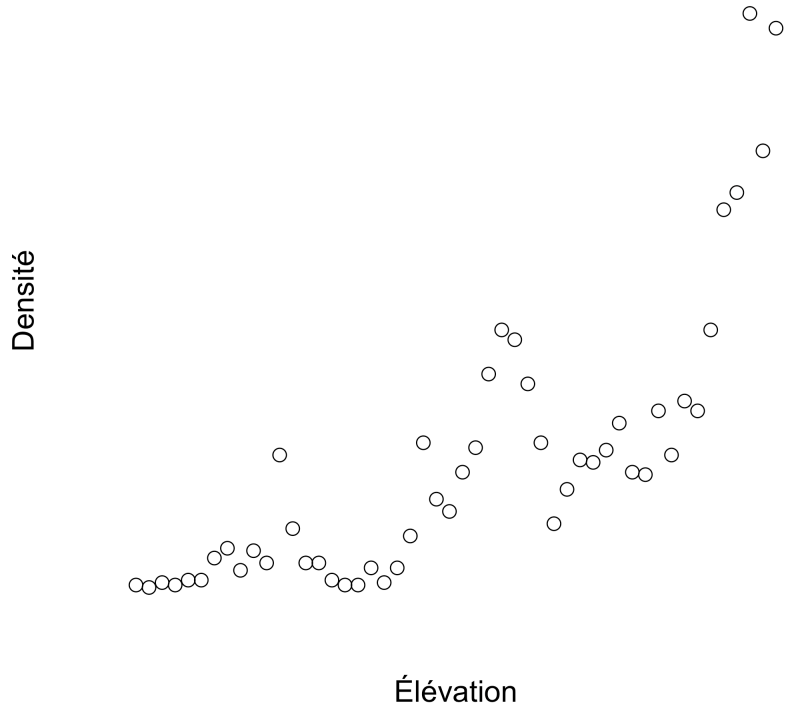
```
plot(elevation, densite[,1], axes = FALSE,
      xlab = "Élévation", ylab = "Densité",
      log = "xy")
```



Ajuster les tailles de caractères

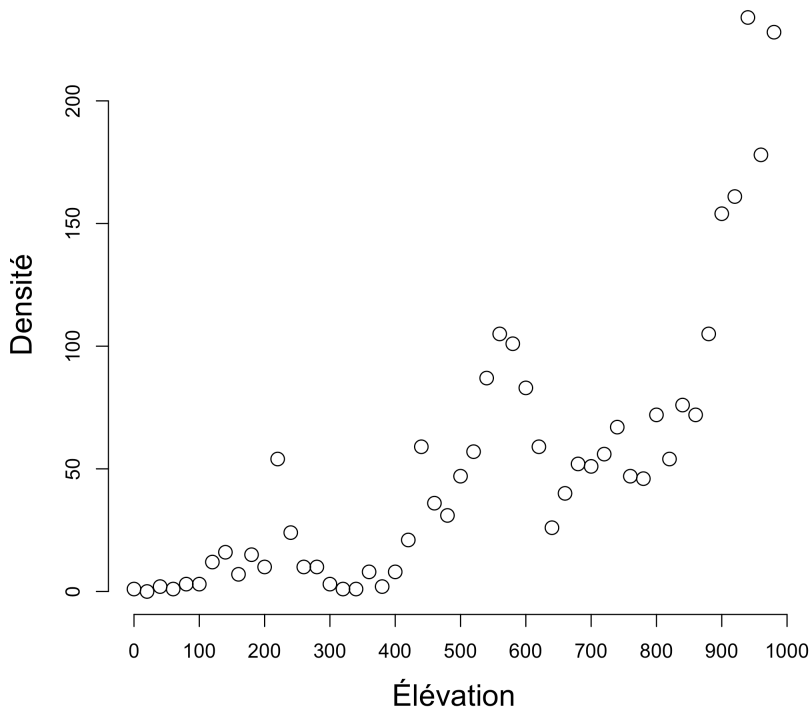
`cex` `cex.lab` `cex.axis`

```
plot(elevation, densite[,1], axes = FALSE,
      xlab = "Élévation", ylab = "Densité",
      cex.lab = 1.5, cex.axis = 1.25, cex = 1.5)
```



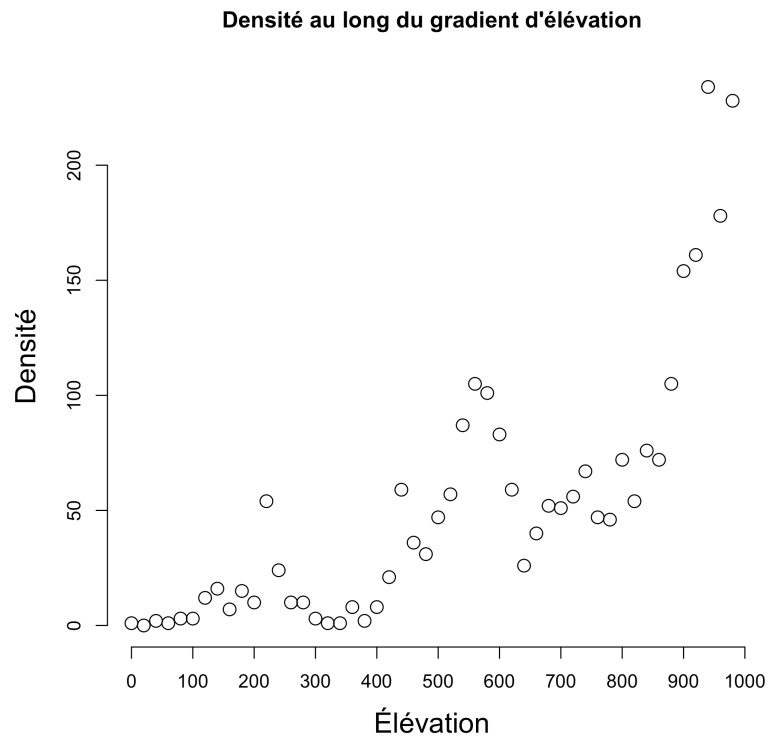
Modifier les axes

```
axis(1, seq(0,1000,100))
axis(2)
```



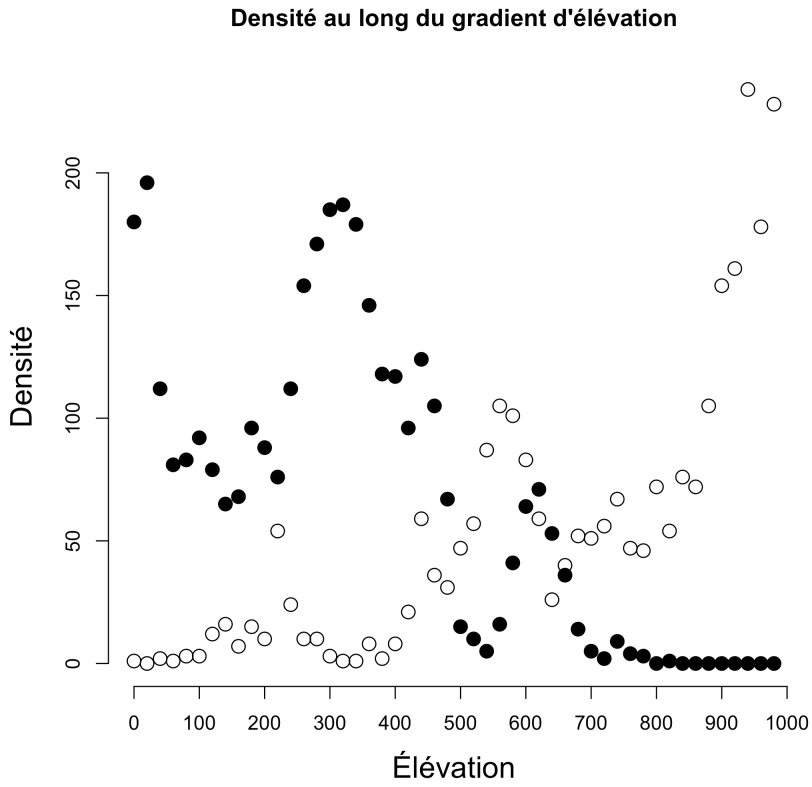
Ajouter un titre

```
title(main = "Densité au long du gradient d'élévation")
```



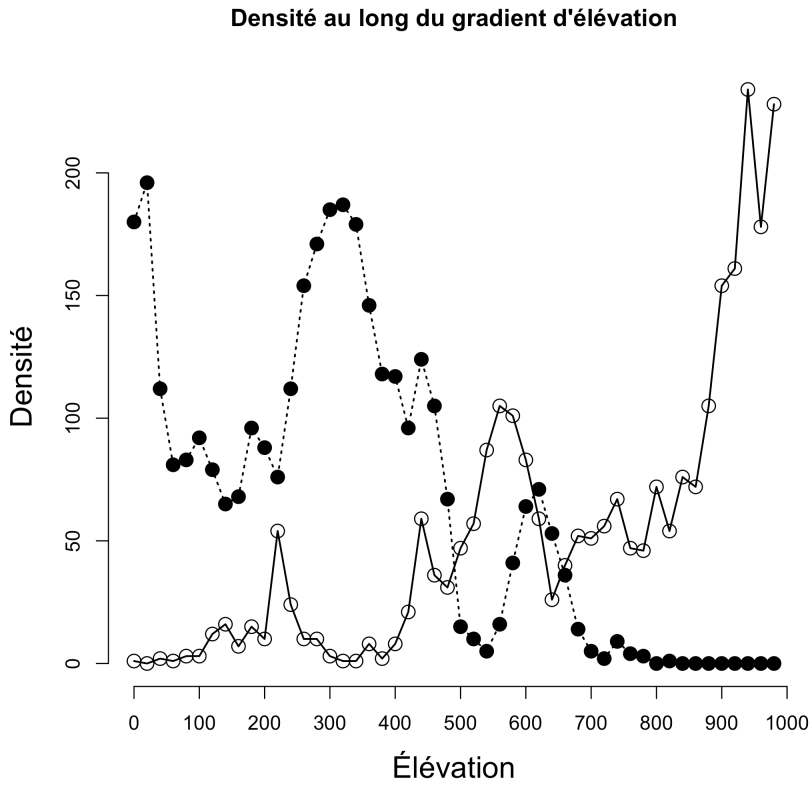
Superposer des points d'une autre série de données

```
points(elevation, densite[,3], pch = 19, cex = 1.5)
```



Superposer des lignes

```
lines(elevation, densite[,1],lty = 1, lwd = 1.5)
lines(elevation, densite[,3], lty  = 3, lwd = 1.5)
```



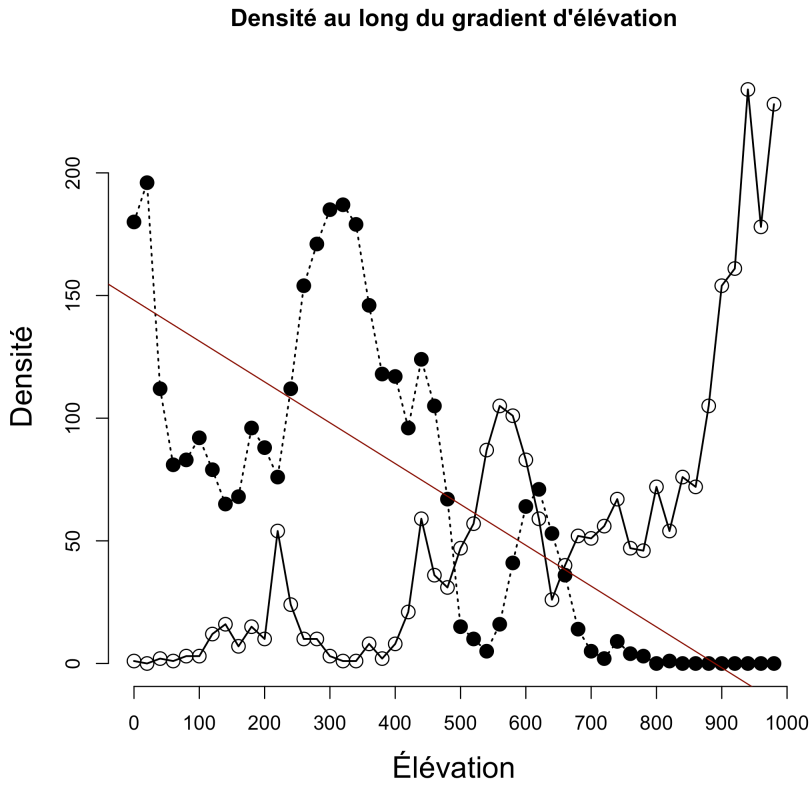
Ajouter une ligne de tendance

```
model = lm(densite[,3]~elevation)
summary(model)
abline(model, col = "darkred")
```

```
##
## Call:
## lm(formula = densite[, 3] ~ elevation)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -59.796 -26.743  -3.565  24.050  92.175
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 148.10588   11.23433  13.183   <2e-16 ***
## elevation    -0.16650    0.01976  -8.428    5e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 40.32 on 48 degrees of freedom
## Multiple R-squared:  0.5968, Adjusted R-squared:  0.5884
## F-statistic: 71.04 on 1 and 48 DF,  p-value: 4.999e-11
```

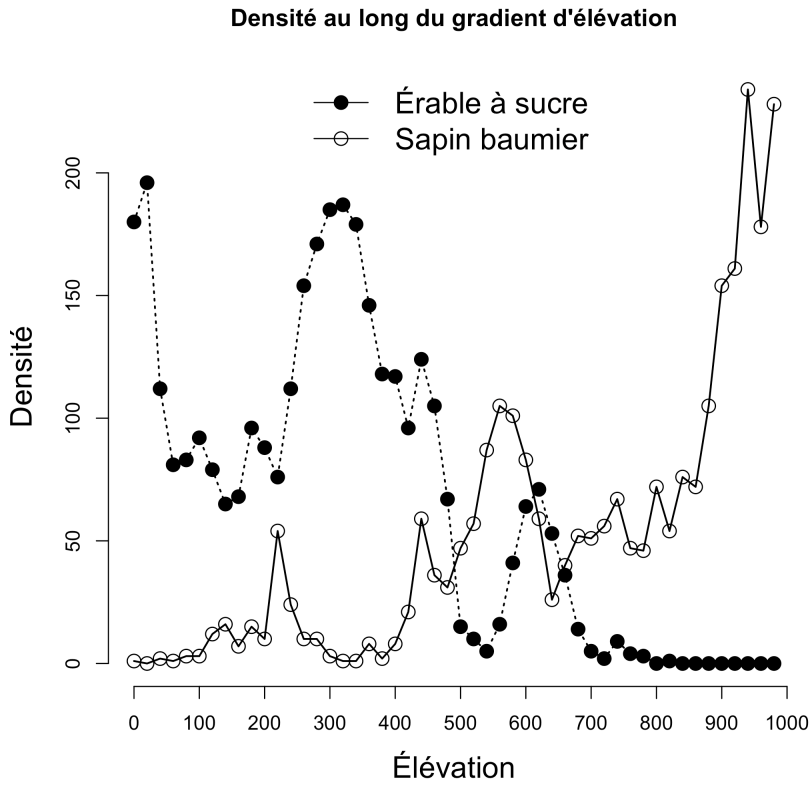
Ajouter une ligne de tendance

```
model = lm(densite[,3]~elevation)
abline(model, col = "darkred")
```



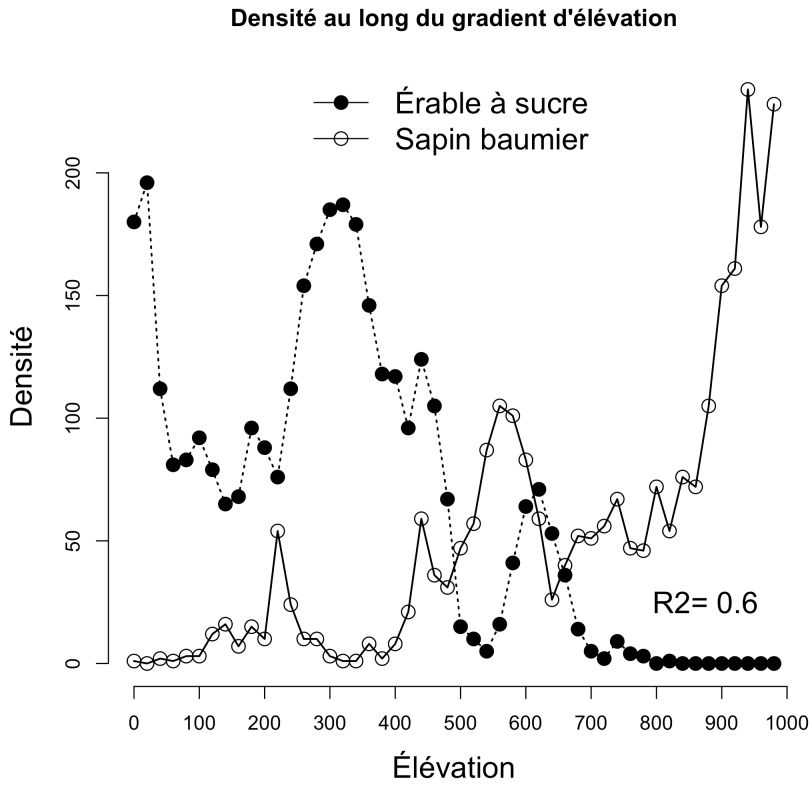
Ajouter une légende

```
legend("top", bty = "n", pch = c(19,1), lty = 1,
    legend = c("Érable à sucre", "Sapin baumier"),
    cex = 1.5)
```



Ajouter du texte

```
r2 <- round(summary(model)$r.squared, 2)
text(x = 850, y = 25, paste("R2=",r2),
    cex = 21.5)
```



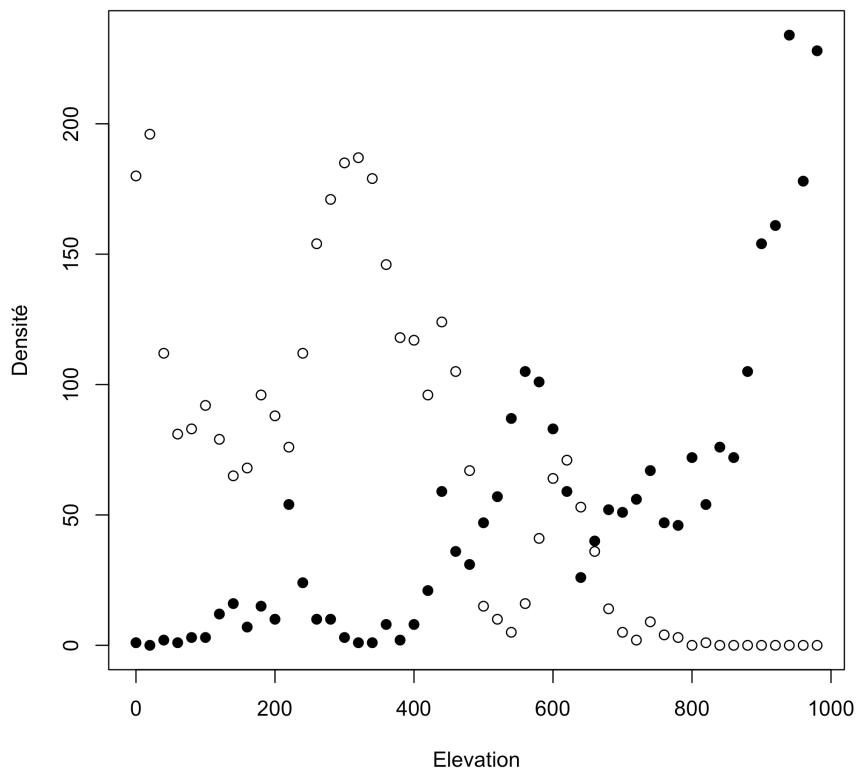
Pour plus d'information

- ✓ `?plot`
- ✓ `?par`
- ✓ `?axis`
- ✓ `?mtext`

Créer d'autres types de figure

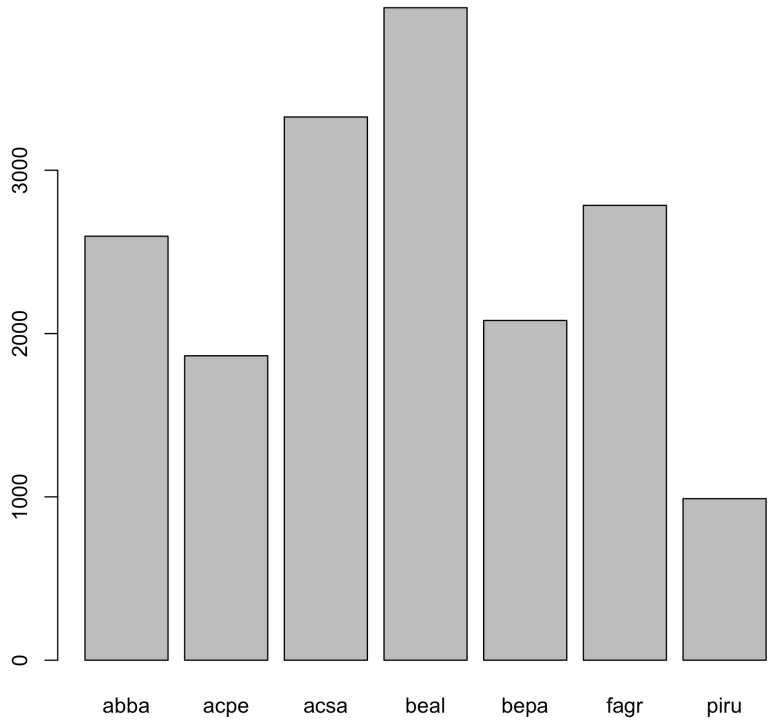
Diagramme de dispersion (Scatter plot)

```
arbres  <- read.csv2("donnees/arbres.csv")
densite <- table(arbres[,c(3,5)])
elevation <- as.numeric(row.names(densite))
plot(elevation, densite[,1], pch = 19,
  xlab = "Elevation", ylab = "Densité")
points(elevation, densite[,3])
```



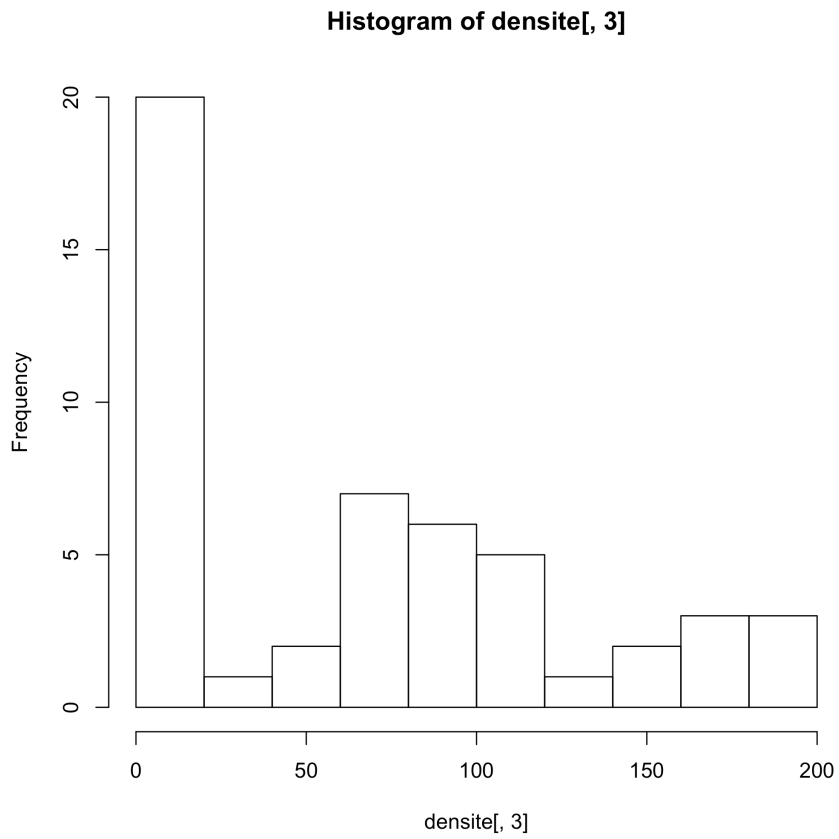
Diagrammes à bâtons (Bar plot)

```
arbres  <- read.csv2("donnees/arbres.csv")
n_tot <- table(arbres$esp)
barplot(n_tot)
```



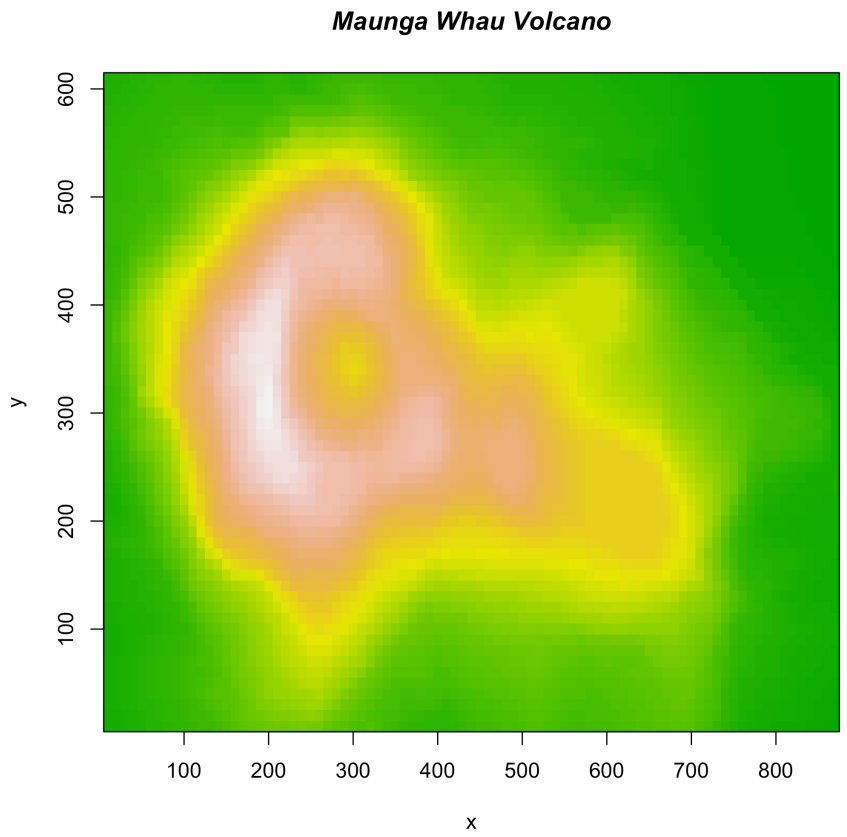
Histogrammes

```
hist(densite[,3])
```



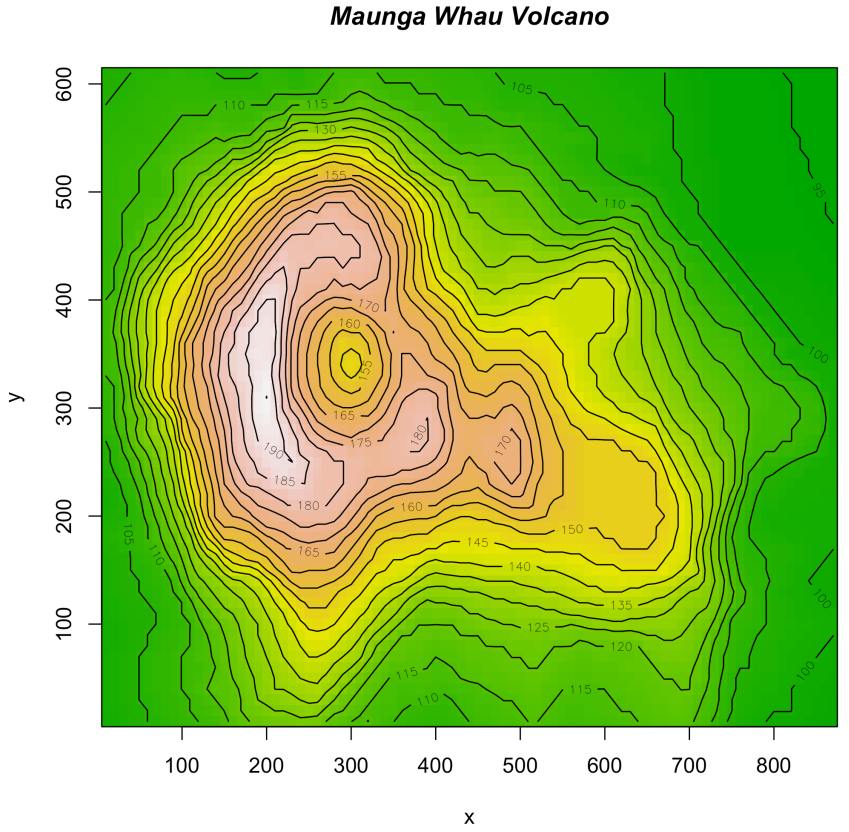
Représentation 3-D

```
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))

image(x, y, volcano,
  col = terrain.colors(100), axes = FALSE)

axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()

title(main = "Maunga Whau Volcano", font.main = 4)
```



Lignes de contour

```
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))

image(x, y, volcano,
  col = terrain.colors(100), axes = FALSE)

axis(1, at = seq(100, 800, by = 100))
axis(2, at = seq(100, 600, by = 100))
box()

title(main = "Maunga Whau Volcano", font.main = 4)

contour(x, y, volcano,
  levels = seq(90, 200, by = 5),
  add = TRUE, col = "black")
```



Enregistrer une figure

```
dev.copy2pdf(file = "test.pdf")
dev.copy2png(file = "test.png")
dev.copy2eps(file = "test.eps")
```

Exercice: faire une représentation visuelle de la distribution de degrés

La visualisation de réseau avec igraph

Installation

```
install.packages("igraph")
```

```
## Installing package into '/usr/local/lib/R/3.3/site-library'
## (as 'lib' is unspecified)
```

```
## Error in contrib.url(repos, type): trying to use CRAN without setting a mirror
```

```
library(igraph)
```

```
## Loading required package: methods
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

Transformer une matrice d'adjacence en objet `igraph`

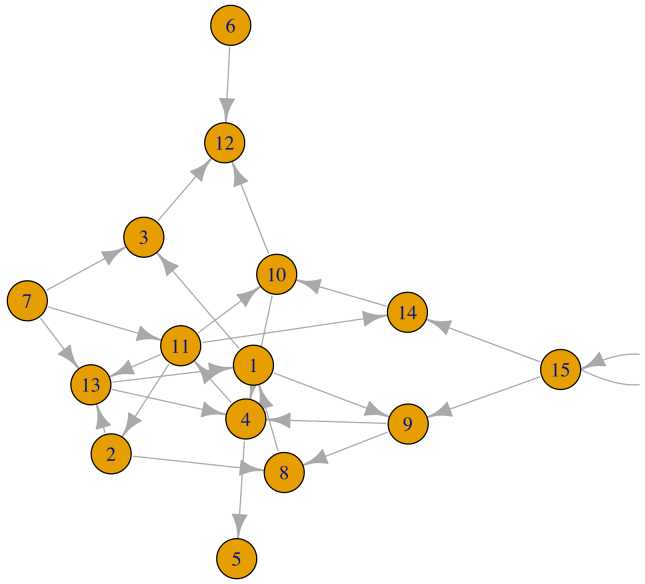
```
library(igraph)
C <- 0.1
S <- 15
L <- matrix(0, nr = S, nc = S)
L[runif(S*S) < C] = 1
sum(L)
```

```
## [1] 26
```

```
g <- graph.adjacency(L)
```

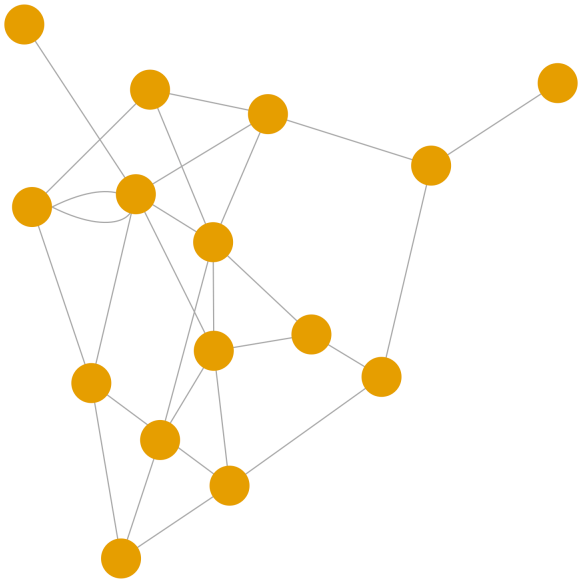
Utiliser la fonction `plot` pour faire une représentation d'un réseau

```
plot(g)
```



Version moins moche dans les paramètres par défaut

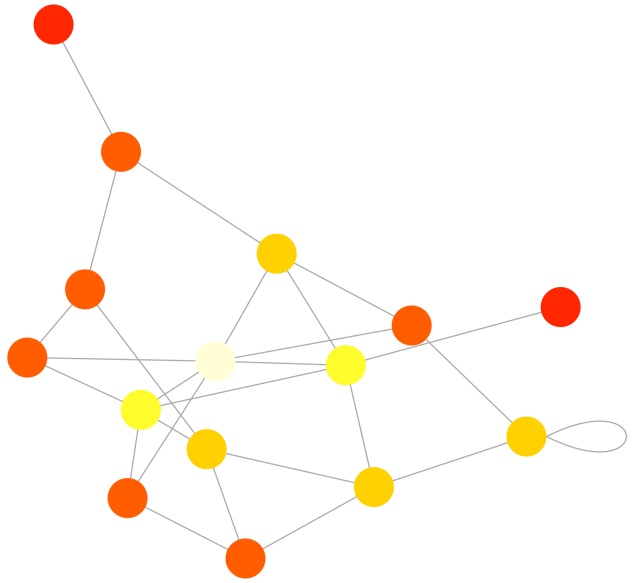
```
plot(g, vertex.label=NA, edge.arrow.mode = 0,
    vertex.frame.color = NA)
```



Exercice : Compiler la matrice d'adjacence et faire une première représentation du réseau avec `igraph`

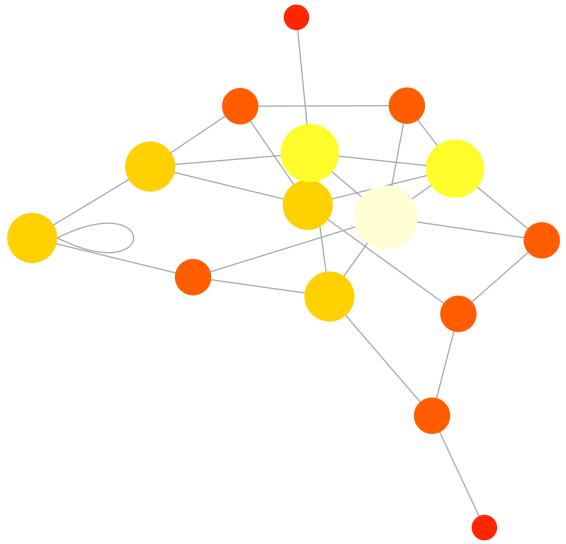
Changer la couleur des noeuds

```
# Calculer le degré
deg <- apply(L, 2, sum) + apply(L, 1, sum)

# Le rang pour chaque noeud
rk <- rank(deg)

# Faire un code de couleur
col.vec <- heat.colors(S)

# Attribuer aux noeuds la couleur
V(g)$color = col.vec[rk]

# Refaire la figure
plot(g, vertex.label=NA, edge.arrow.mode = 0,
    vertex.frame.color = NA)
```



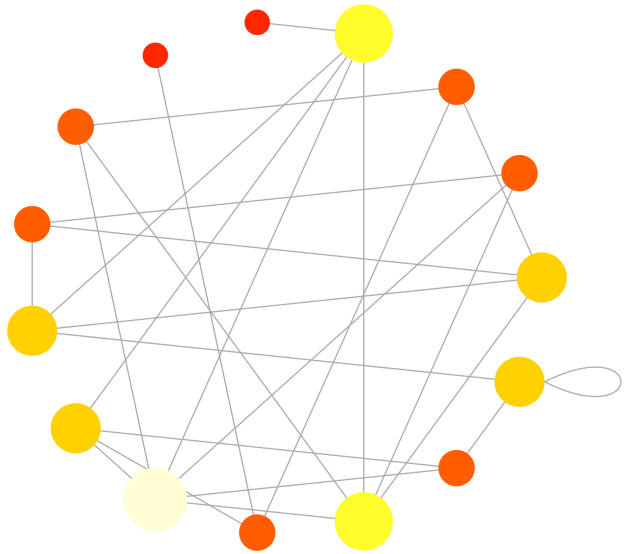
Changer la taille des noeuds

```
# Faire un code de ctaille
col.vec <- seq(10, 25, length.out = S)

# Attribuer aux noeuds la couleur
V(g)$size = col.vec[rk]

# Refaire la figure
plot(g, vertex.label=NA, edge.arrow.mode = 0,
    vertex.frame.color = NA)
```



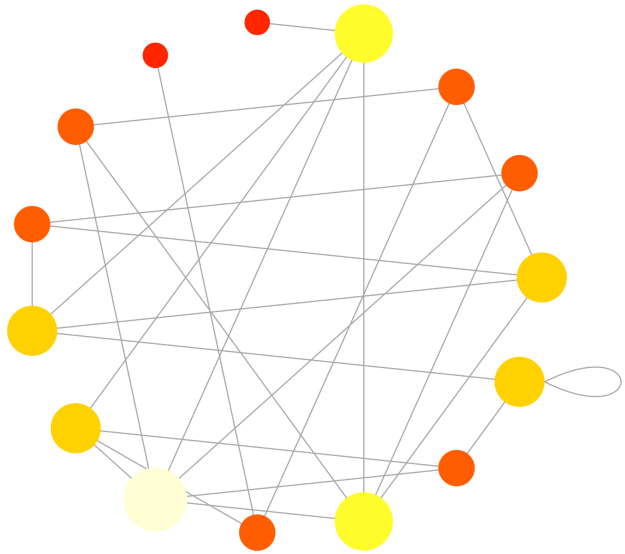
Changer la disposition des noeuds

```
plot(g, vertex.label=NA, edge.arrow.mode = 0,
    vertex.frame.color = NA, layout = layout.reingold.tilford(g))
```



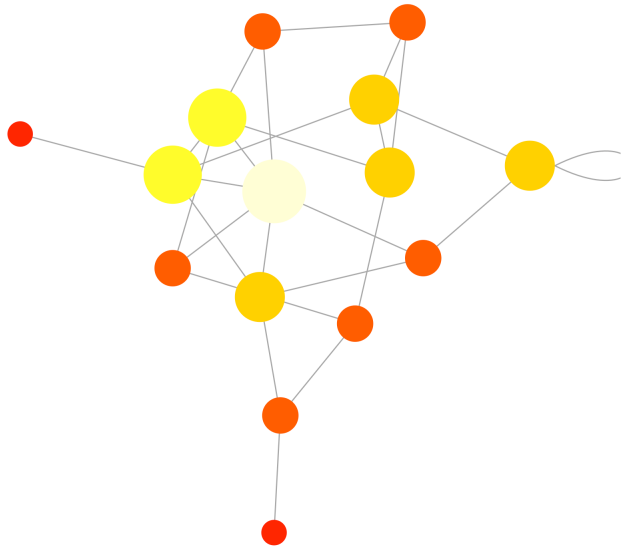
Changer la disposition des noeuds

```
plot(g, vertex.label=NA, edge.arrow.mode = 0,
    vertex.frame.color = NA, layout = layout.circle(g))
```



Changer la disposition des noeuds

```
plot(g, vertex.label=NA, edge.arrow.mode = 0,
    vertex.frame.color = NA, layout = layout.kamada.kawai(g))
```



Calcul de propriétés

```
wtc = walktrap.community(g)
modularity(wtc)
```

```
## [1] 0.2248521
```

Calcul de propriétés

```
distances(g)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]    0    2    1    2    3    3    2    1    1     3     2     2     1
##  [2,]    2    0    3    2    3    4    2    1    2     2     1     3     1
##  [3,]    1    3    0    3    4    2    1    2    2     2     2     1     2
##  [4,]    2    2    3    0    1    3    2    2    1     1     1     2     1
##  [5,]    3    3    4    1    0    4    3    3    2     2     2     3     2
##  [6,]    3    4    2    3    4    0    3    4    4     2     3     1     4
##  [7,]    2    2    1    2    3    3    0    3    3     2     1     2     1
##  [8,]    1    1    2    2    3    4    3    0    1     3     2     3     2
##  [9,]    1    2    2    1    2    4    3    1    0     2     2     3     2
## [10,]    3    2    2    1    2    2    2    3    2     0     1     1     2
## [11,]    2    1    2    1    2    3    1    2    2     1     0     2     1
## [12,]    2    3    1    2    3    1    2    3    3     1     2     0     3
## [13,]    1    1    2    1    2    4    1    2    2     2     1     3     0
## [14,]    3    2    3    2    3    3    2    3    2     1     1     2     2
## [15,]    2    3    3    2    3    4    3    2    1     2     2     3     3
##       [,14] [,15]
##  [1,]     3     2
##  [2,]     2     3
##  [3,]     3     3
##  [4,]     2     2
##  [5,]     3     3
##  [6,]     3     4
```

Calcul de propriétés

```
eigen_centrality(g)$vector
```

```
##  [1] 0.54525807 0.57588016 0.34372408 0.82046697 0.20596409 0.06680901
##  [7] 0.55784521 0.41556710 0.53429148 0.64963376 1.00000000 0.26613662
## [13] 0.87847671 0.50124092 0.34708135
```

Exporter des tableaux

Exporter des tableaux

Exporter des tableaux depuis R vers son document de travail peut être difficile.

1. Enregistrer le `data.frame` dans un fichier avec la fonction `write.table()` ou `write.csv()`
2. Éditer et faire la mise en page dans MS Excel ou MS Word.

Le package `knitr` permet de faciliter cette procédure en exportant le `data.frame` directement dans son document de travail LaTeX.

Exporter des tableaux

Prenons le jeu de données `iris` directement disponible sous R.

```
data(iris)
class(iris)
```

```
## [1] "data.frame"
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Exporter des tableaux

Je souhaite maintenant exporter ce `data.frame` en LateX (un format que nous verrons lors de la prochaine séance):

```
library(knitr)
iris_tex <- kable(iris,format="latex")
writeLines(iris_tex, con = "./donnees/iris.tex", sep = "\n", useBytes = FALSE)
```

Travail # 2

Consignes

- ✓ Identifier clairement vos questions de recherche
- ✓ Illustrer le réseau de collaborations
- ✓ Compléter votre analyse au moyen de 3 figures et 1 tableau

Évaluation

- ✓ Clareté des questions et adéquation des figures et du tableau
- ✓ Efficacité de la présentation
- ✓ Respect de normes graphiques
- ✓ Originalité

Lectures

Sandve et al. 2013. Ten Simple Rules for Reproducible Computational Research. PLoS Computational Biology. 9: e1003285

Silberzahn et Uhlman. 2015. Many hands make tight work. Nature 526 : 189-191.

Barba. 2016. The hard road to reproducibility. Science 354: 142.