

Séance 5: La visualisation des données

BIO 500 - Méthodes en écologie computationnelle

Dominique Gravel & Steve Vissault
Laboratoire d'écologie intégrative

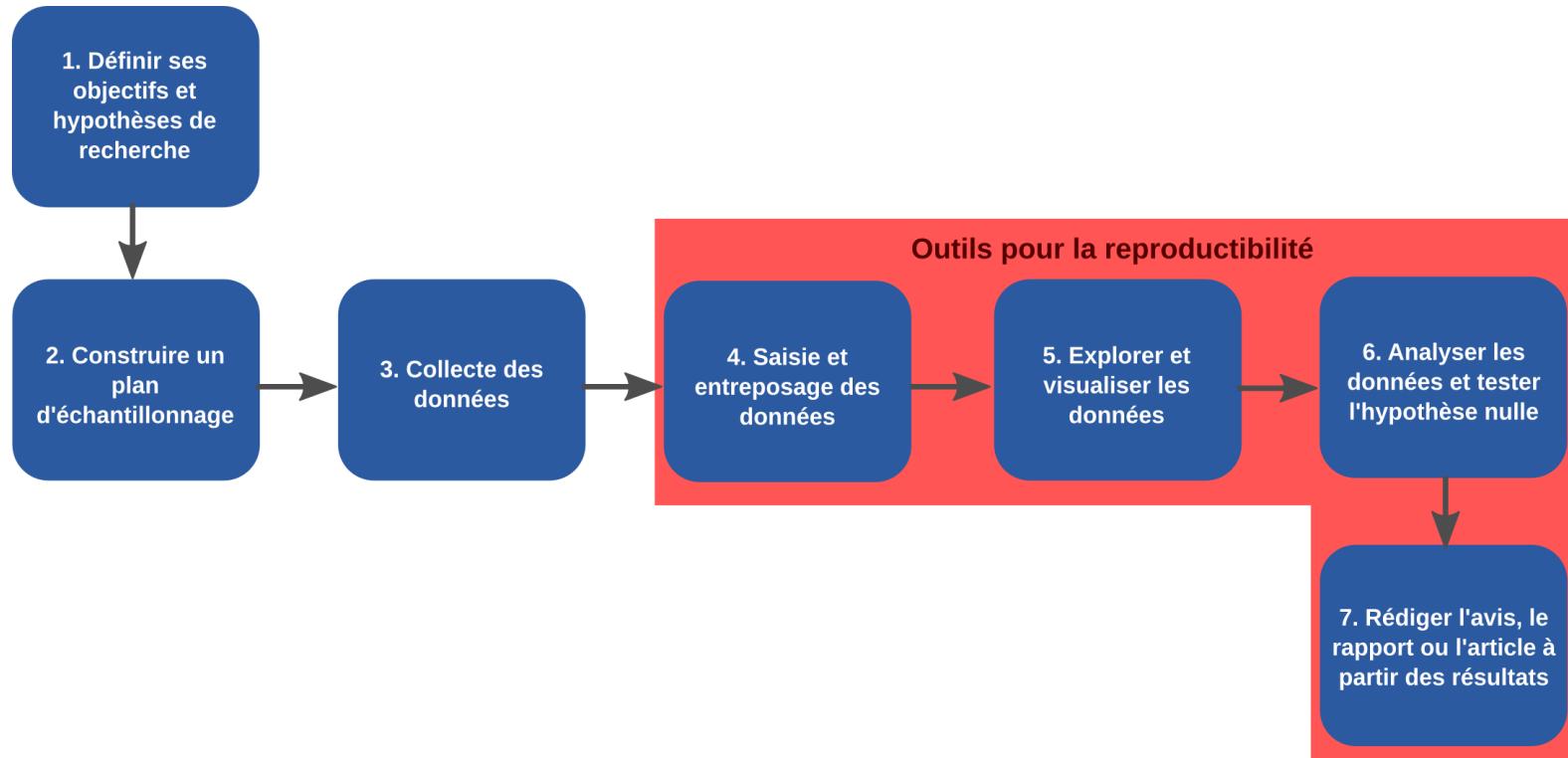


Séance 5

- ✓ 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**

Le makefile

Les étapes du travail d'un biologiste



Qu'est-ce que le makefile ?

Définition: le makefile est un fichier qui contient un ensemble de directives qui sont exécutées par l'ordinateur. Les instructions et leurs dépendances sont spécifiées dans le makefile.

À quoi il sert ?

Le makefile est un logiciel permettant d'exécuter une série d'instructions, lorsqu'elles sont nécessaires. Les dépendances sont vérifiées et seulement les instructions qui requièrent une mise à jour sont exécutées.

Nous utiliserons le makefile pour assurer la reproductibilité de la démarche entreprise dans le cours. L'ensemble des instructions nécessaires à la production du rapport, de la création de la base de données à la compilation du document écrit, seront contenues dans le makefile.

Objectifs de la leçon

- ✓ Reconnaître les parties importantes d'un makefile, les règles, les cibles, les dépendances et les actions
- ✓ Écrire un makefile simple
- ✓ Exécuter un makefile à partir du terminal
- ✓ Préparer le makefile pour le projet de session

Anatomie du makefile

```
<target>: <dependencies...>
  <commands>
```

target

La cible est habituellement le nom d'un fichier généré par la commande.

dependencies . . .

Une dépendance (également appelée "*prerequisite*") est un fichier qui est utilisé pour créer un autre fichier appelée cible ("**target**").

La *target* peut contenir plusieurs dépendances.

Il est néanmoins possible d'avoir un fichier cible qui ne requiert pas de dépendances.

commands

La commande est l'action à réaliser. Dans notre cas, nous utiliserons une commande pour exécuter un script R comme :

```
Rscript script.R
```

Nous verrons plus tard dans la session le langage de mise en forme LaTeX. Dans ce cas, la commande serait:

```
pdflatex manuscrit.tex
```

Un exemple

```
# Première étape, on génère des données
data.txt :
  Rscript script1.R

# Seconde étape, on fait un modèle statistique à partir
# de ces données
model.Rdata: data.txt
  Rscript script2.R

# Troisième étape, on produit une figure à partir du modèle
# et des données
figure.pdf: model.Rdata data.txt
  Rscript script3.R
```

Ce fichier s'appelle **makefile** (sans extension) et il est exécuté en inscrivant simplement la commande **make nomDeLaCible**.

Étape par étape

Rscript est un programme permettant d'exécuter du code R sans passer par la console R. On peut utiliser le terminal pour appeler le programme **Rscript** et lui donner comme argument un script R: **Rscript script1.R**

Les commandes sont espacées par une *tabulation* ou 8 espaces. Assurez vous que votre tabulation corresponds bien à 8 espaces.

Ensemble, la cible, les dépendances et les actions constituent une règle. Cet exemple a donc 3 règles.

Comment créer un premier makefile

- ✓ Ouvrez un nouveau document au moyen de atom
- ✓ Copiez les commandes de l'exemple
- ✓ Sauvegardez le fichier, avec pour nom makefile
- ✓ Assurez vous de trouver les 3 scripts dans le dépôt git

Script 1

```
set.seed(1)
X <- runif(25, 0, 100)
Y <- rnorm(25, mean = X*2 + 10, sd = 25)
write.table(cbind(X,Y), file = "data.txt")
```

Script 2

```
data <- read.table("data.txt", header = T)
model <- lm(data$Y ~ data$X)
save(model, file = "model.Rdata")
```

Script 3

```
data <- read.table("data.txt", header = T)
load("model.Rdata")

pdf("resultat.pdf", 7 5)
plot(data$X, data$Y, xlab = "X", ylab = "Y")
abline(model)
dev.off()
```

Exécuter un makefile

Il n'est pas nécessaire de nommer le makefile *makefile*. On peut toujours spécifier un autre nom et l'exécuter ainsi :

```
make -f MonMakefile
```

Les dépendances

- ✓ Exécutez le makefile une première fois.
- ✓ Ensuite, modifiez une valeur dans le fichier data.txt.
- ✓ Exécutez le makefile à nouveau pour voir ce qui se produit.

Le makefile comme outil de reproductibilité

La rédaction du makefile nous force à spécifier les différentes étapes de notre démarche, à identifier les entrées et les sorties de différentes instructions à l'ordinateur. De cette façon, le makefile permet de documenter rigoureusement la démarche réalisée.

Il s'agit aussi d'un aide-mémoire qui permet de se rappeler des étapes.

Exercice

Conceptualisez les différentes étapes de la démarche du travail de session jusqu'à présent.

Identifiez les entrées, les sorties et les actions.

Les messages de make

Il est possible de forcer make à réaliser certaines actions. Par exemple, si on exécute à nouveau make

```
make
```

On obtient le message :

```
make : 'data.txt' is up to date
```

Les messages de make

On peut forcer une étape, par exemple le calcul du modèle, ainsi :

```
make model.Rdata
```

Dans ce cas-ci, on obtient le message :

```
make: Nothing to be done for 'script2.R'
```

Les messages de make

up to date signifie que le makefile a une règle dont la cible est le nom du fichier et qu'il est à jour

Nothing to be done indique que le fichier existe, mais que

- ✓ il n'y a pas de règle pour ce fichier
- ✓ il y a une règle, mais aucune action à réaliser

Nettoyage du dépôt en fin de script

Il arrive que certains scripts génèrent des fichiers temporaires qui ne doivent pas être conservés inutilement.

Dans l'exemple précédent, on pourrait vouloir éliminer le modèle, qui n'est finalement utilisé que pour réaliser la figure. On peut ainsi inscrire à la fin du makefile :

```
clean :  
    rm -f model.Rdata
```

`rm -f model.Rdata` va éliminer à la toute fin l'objet R contenant le modèle intitulé **model.Rdata**. Notez ici qu'il n'y a pas de dépendance, ce qui indique que cette opération sera systématiquement exécutée.

Dépendances

L'ordre de présentation des dépendances est arbitraire. Elles ne seront pas nécessairement vérifiées dans l'ordre présenté.

Un truc est de schématiser les dépendances (des noeuds) et les actions (des flèches)

Autres ressources disponibles en ligne :

<https://swcarpentry.github.io/make-novice/02-makefiles/>

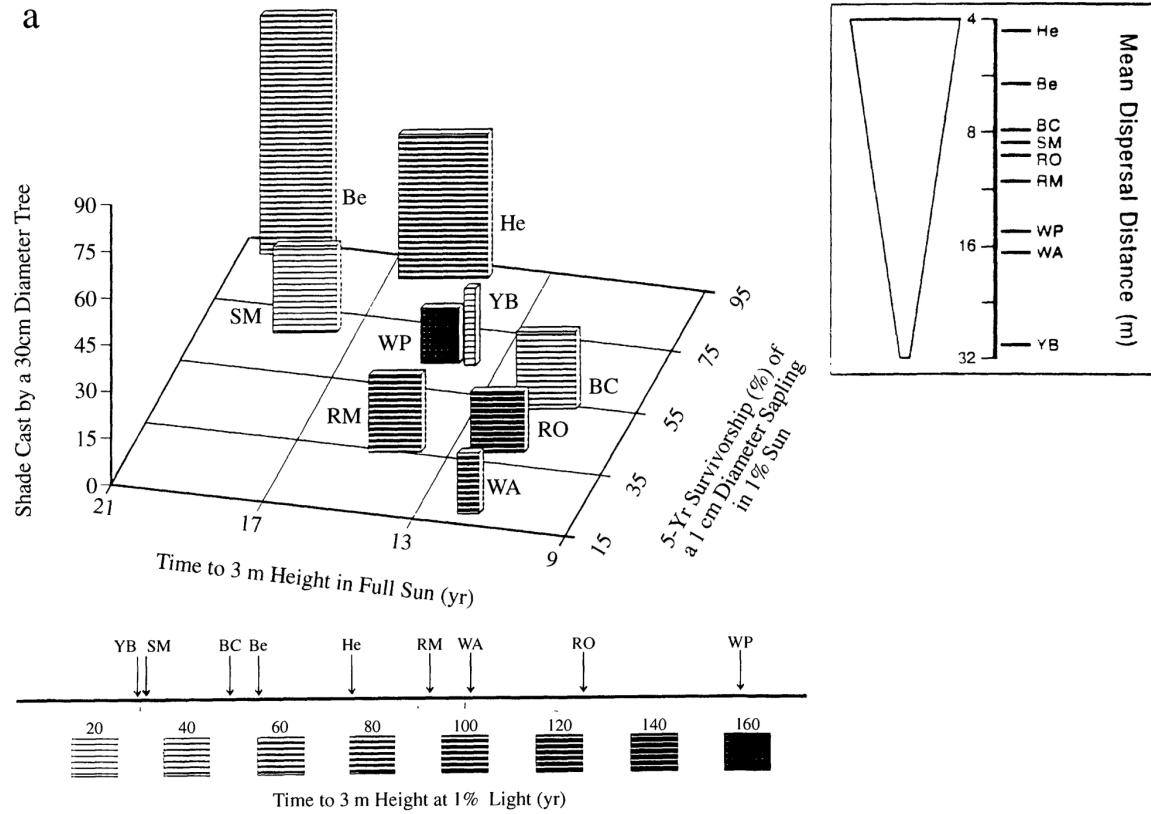
<https://gist.github.com/isaacs/62a2d1825d04437c6f08>

<http://gl.developpez.com/tutoriel/outil/makefile/>

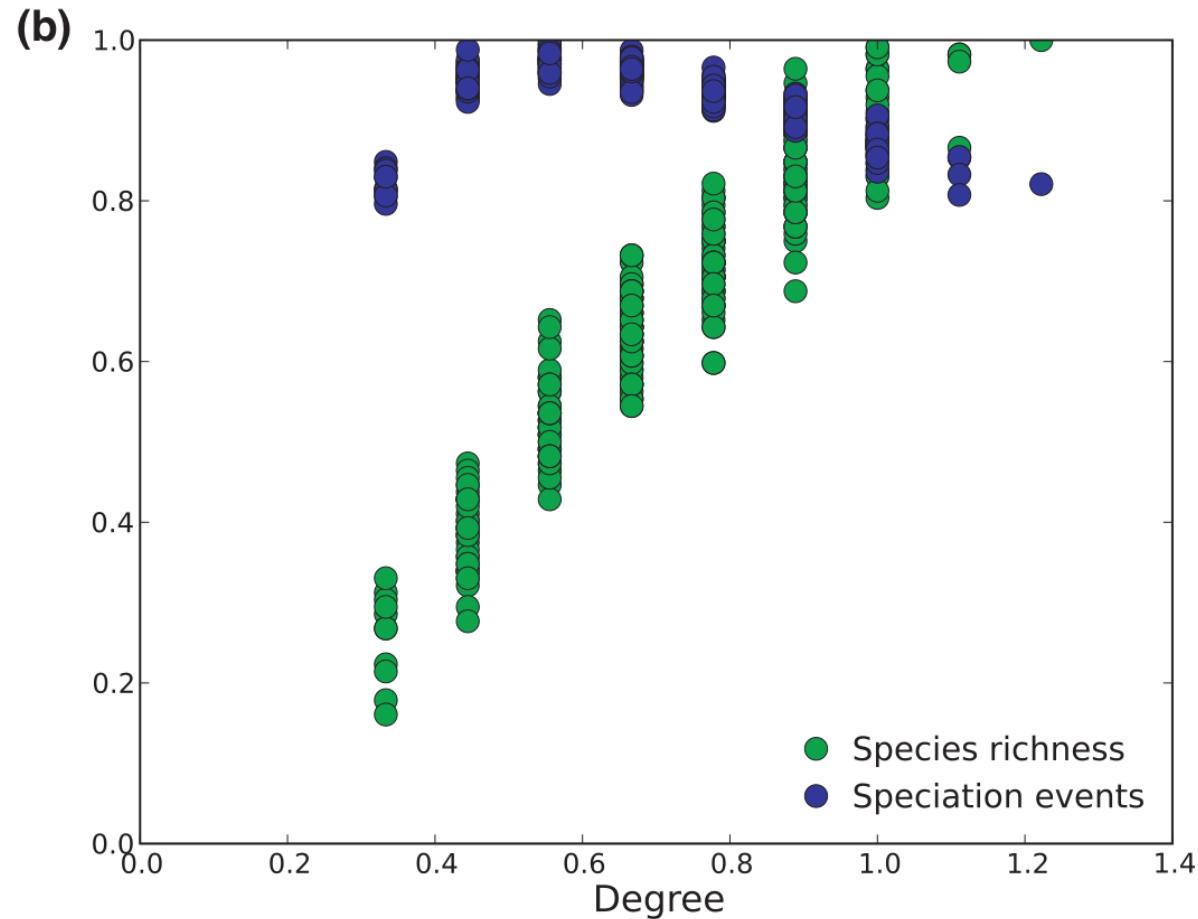
http://icps.u-strasbg.fr/people/loechner/public_html/enseignement/GL/make.pdf

Qu'est-ce qui fait une bonne figure ?

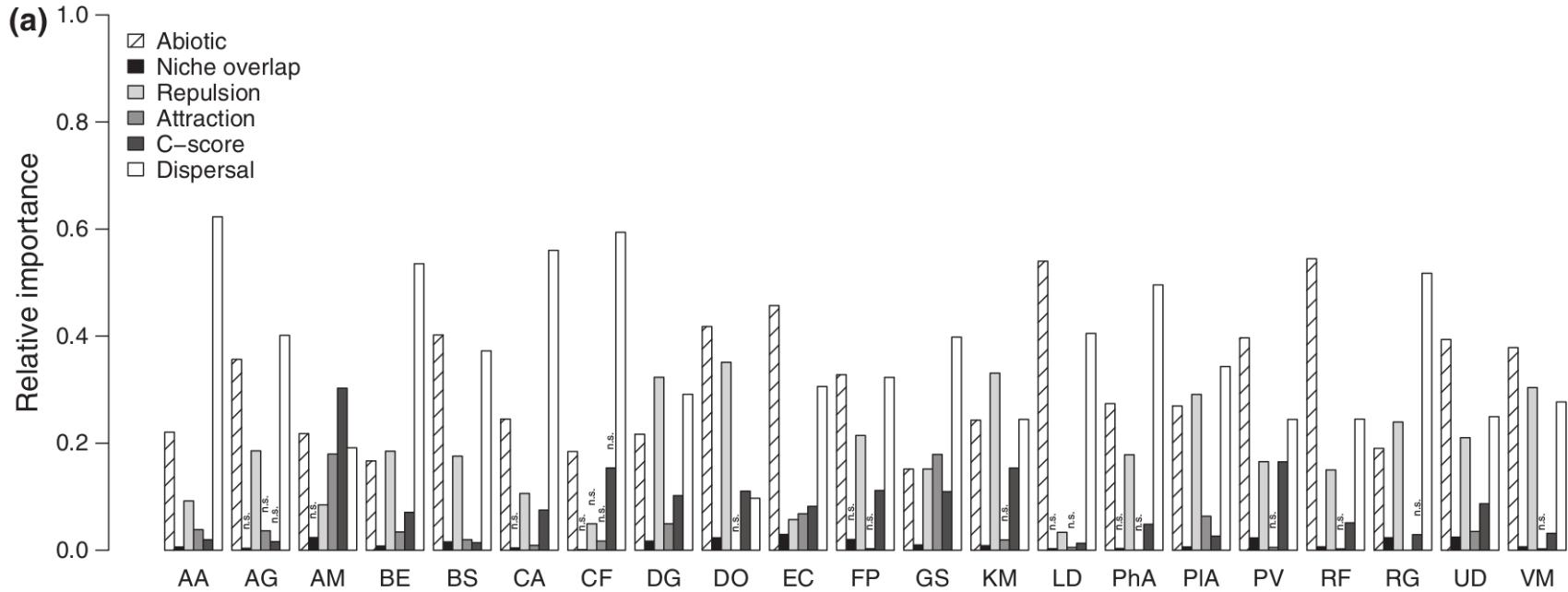
Trop d'information



Non respect des normes graphiques



Abus de symboles et de couleurs



L'art graphique

L'importance des graphiques

La représentation visuelle des données permet de:

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

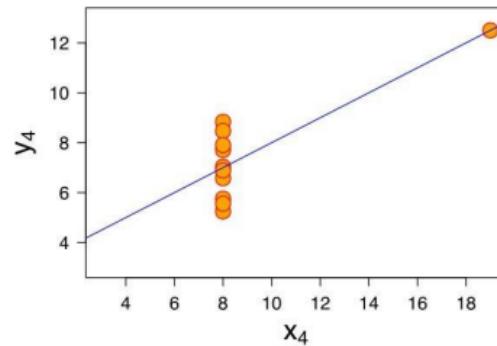
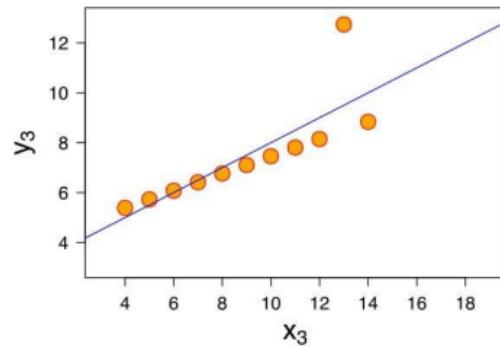
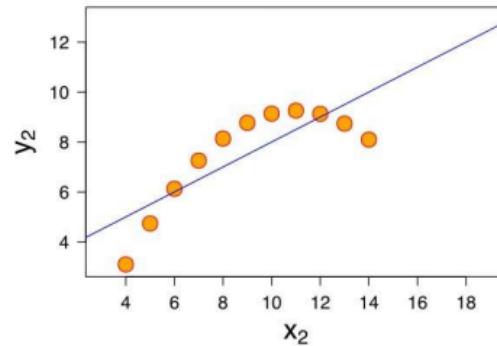
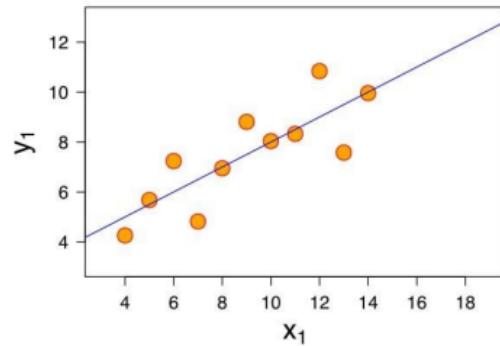
Explorer nos données par la visualisation

Voici un exemple illustrant l'importance de visualiser ses données:

	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

Voici un exemple illustrant l'importance de visualiser ses données:

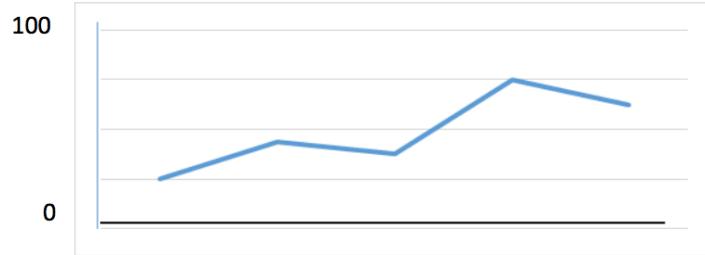


Communiquer par les graphiques

- ✓ 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

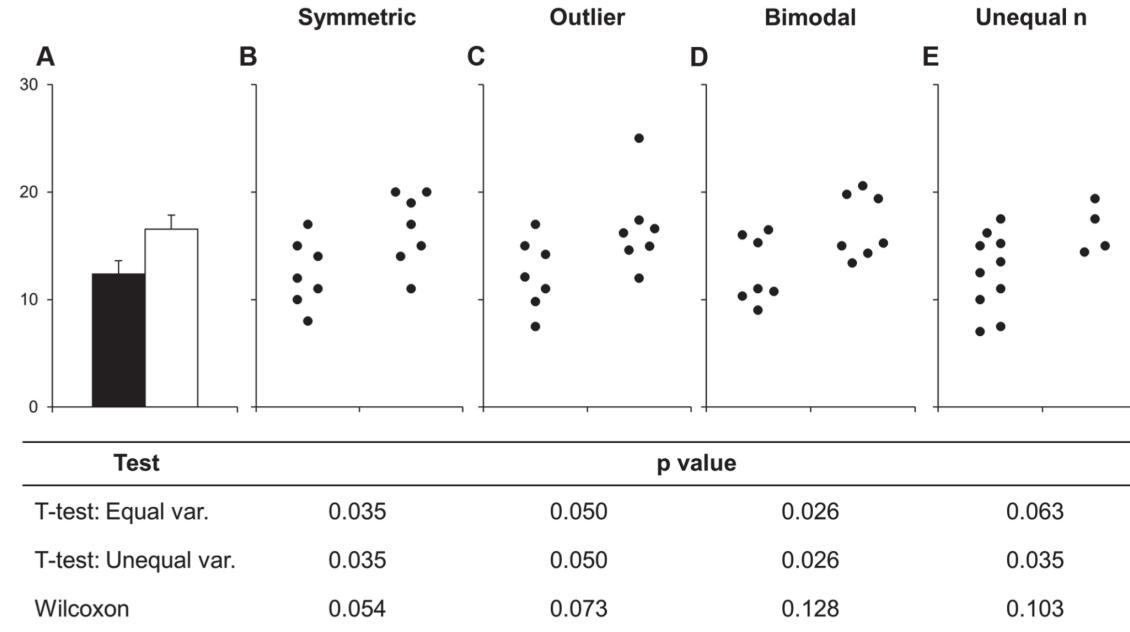


Fig 1. Many different datasets can lead to the same bar graph. The full data may suggest different conclusions from the summary statistics. The means and SEs for the four example datasets shown in Panels B–E are all within 0.5 units of the means and SEs shown in the bar graph (Panel A). *p*-values were 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

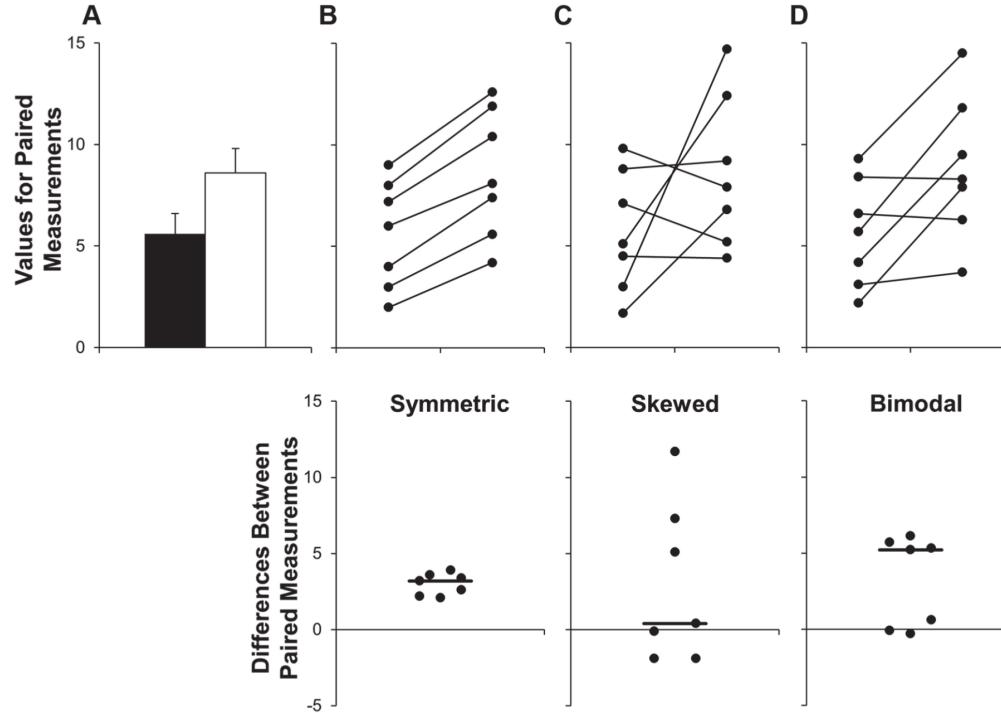


Fig 2. Additional problems with using bar graphs to show paired data. The bar graph (mean \pm SE) suggests that the groups are independent and provides no information about whether changes are consistent across individuals (Panel A). The scatterplots shown in the Panels B–D clearly demonstrate that the 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

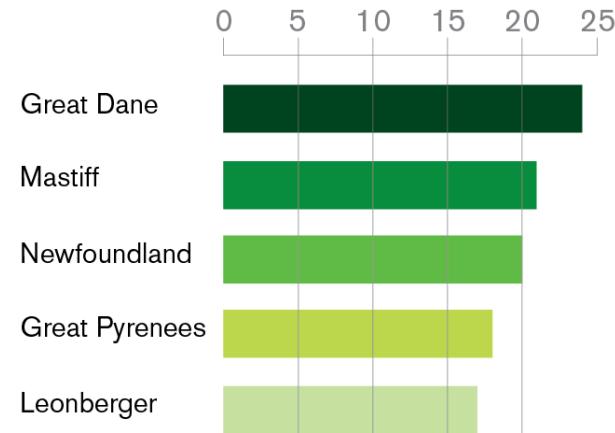
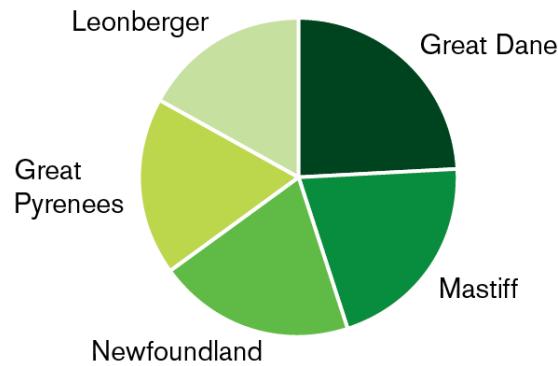
- ✓ Une figure doit renvoyer un seul message/résultat.
- ✓ Chaque élément d'une figure doit **aider à comprendre** ce message.
- ✓ **Choisir le bon type de représentation** permet de mettre en valeur plus facilement ce qui doit être illustré.
- ✓ **Attention aux normes graphiques:** Choix des couleurs,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



Quelques conseils

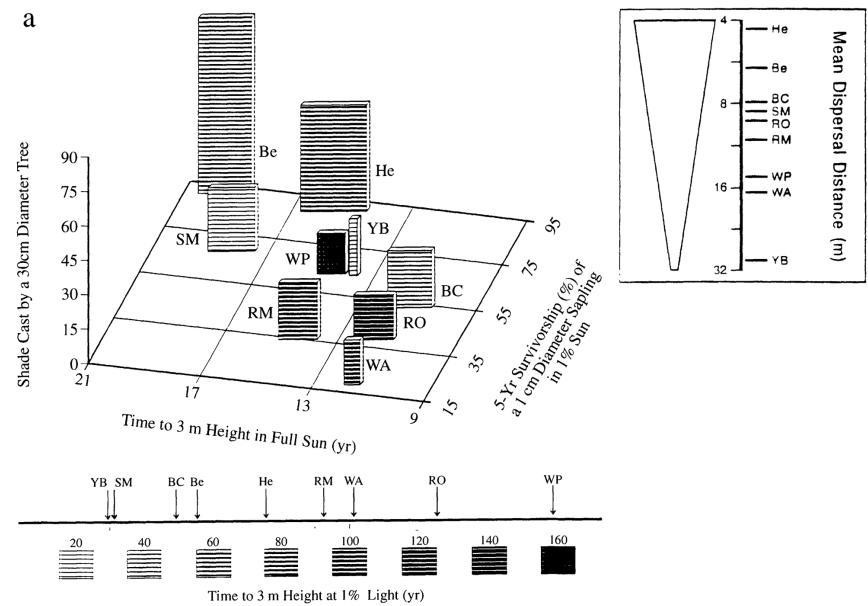
- ✓ Ne pas **JAMAIS** utiliser de diagramme en pointe de tarte

Pie vs. bar chart



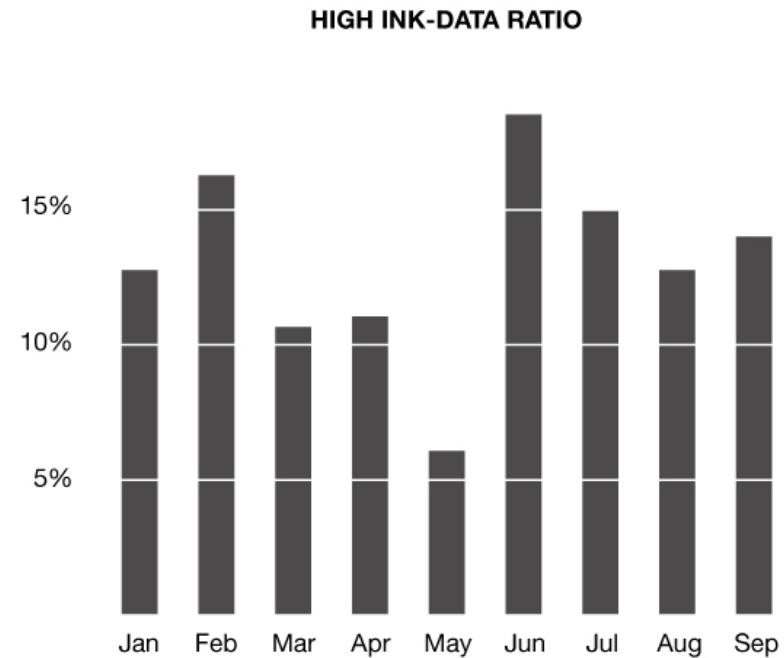
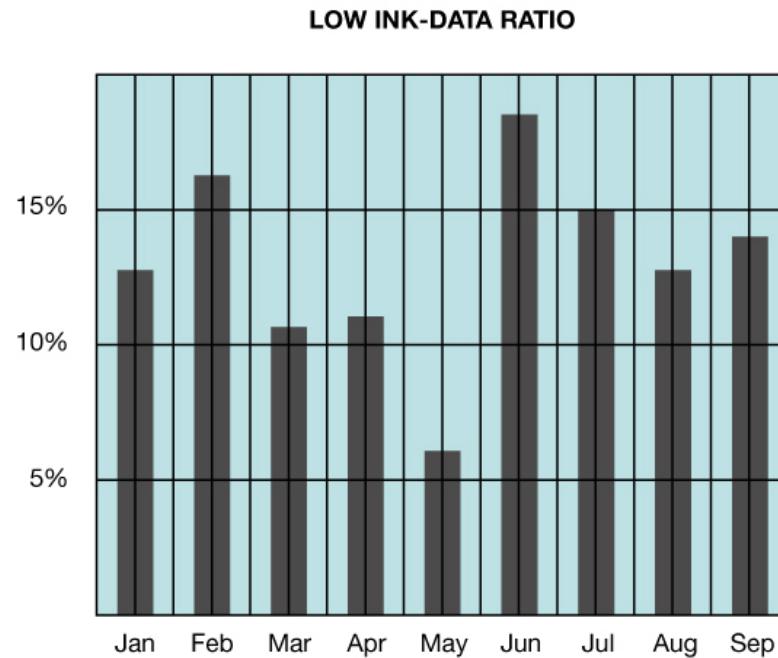
Quelques conseils

- ✓ 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 taille des points
 - Ajoutant des couleurs



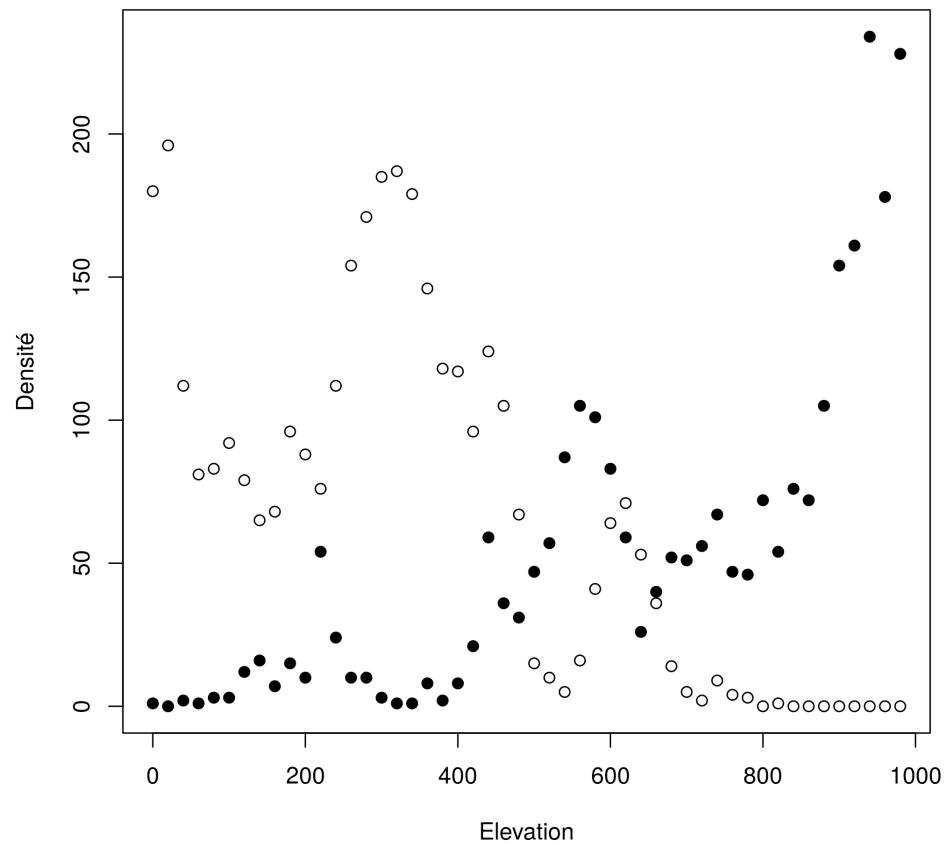
Quelques conseils

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

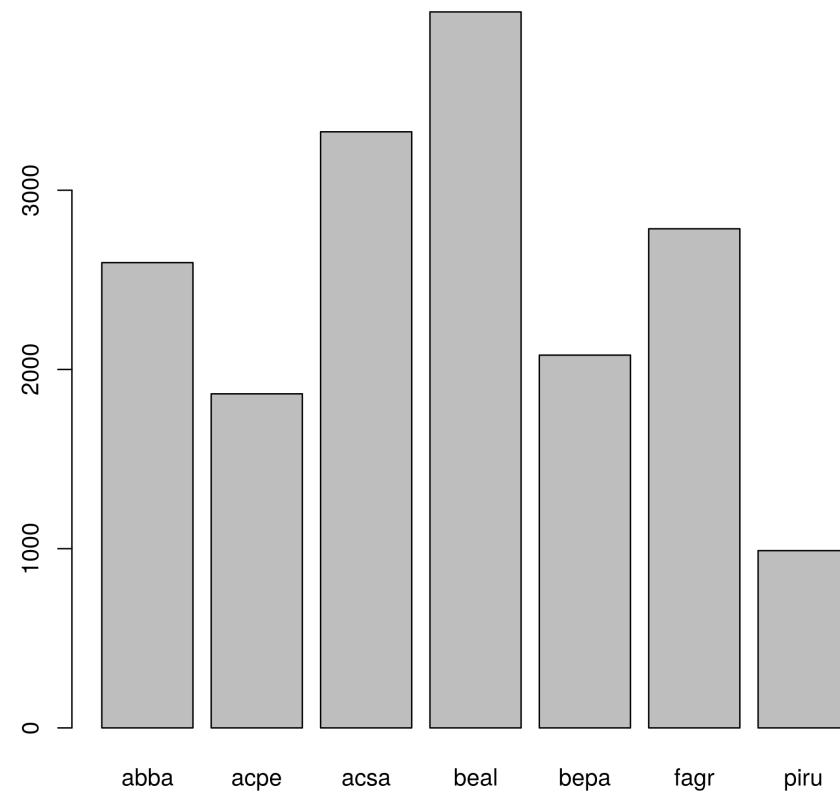


Types de figures

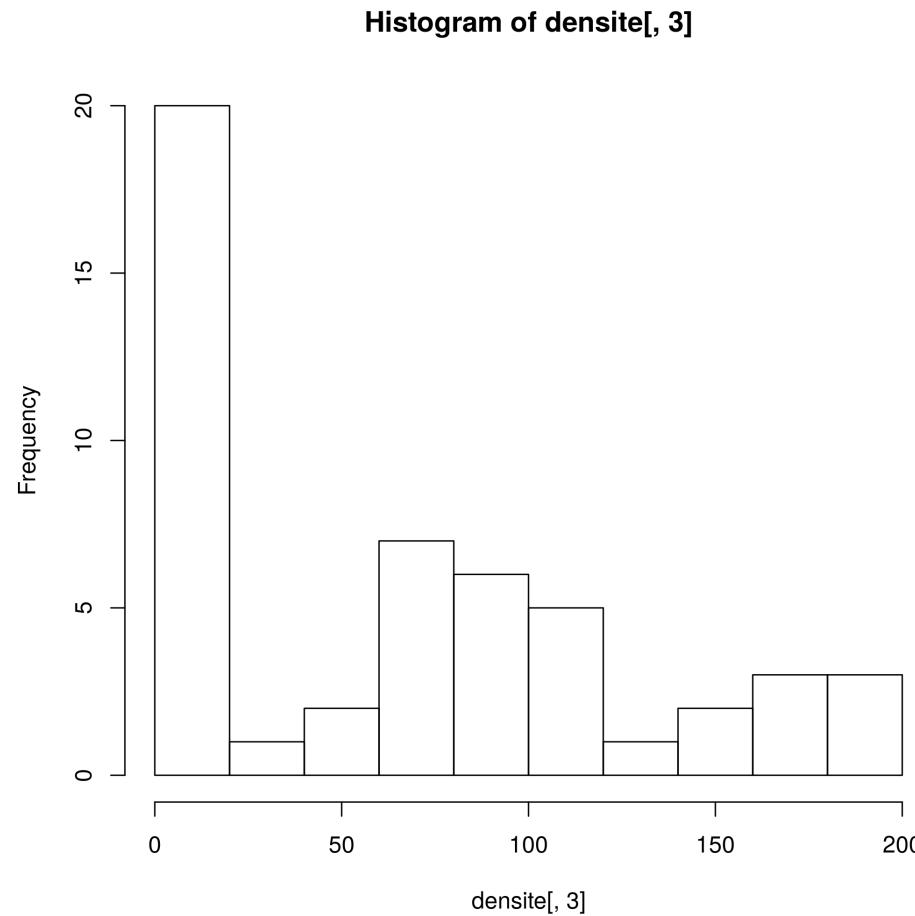
Diagramme de dispersion (Scatter plot)



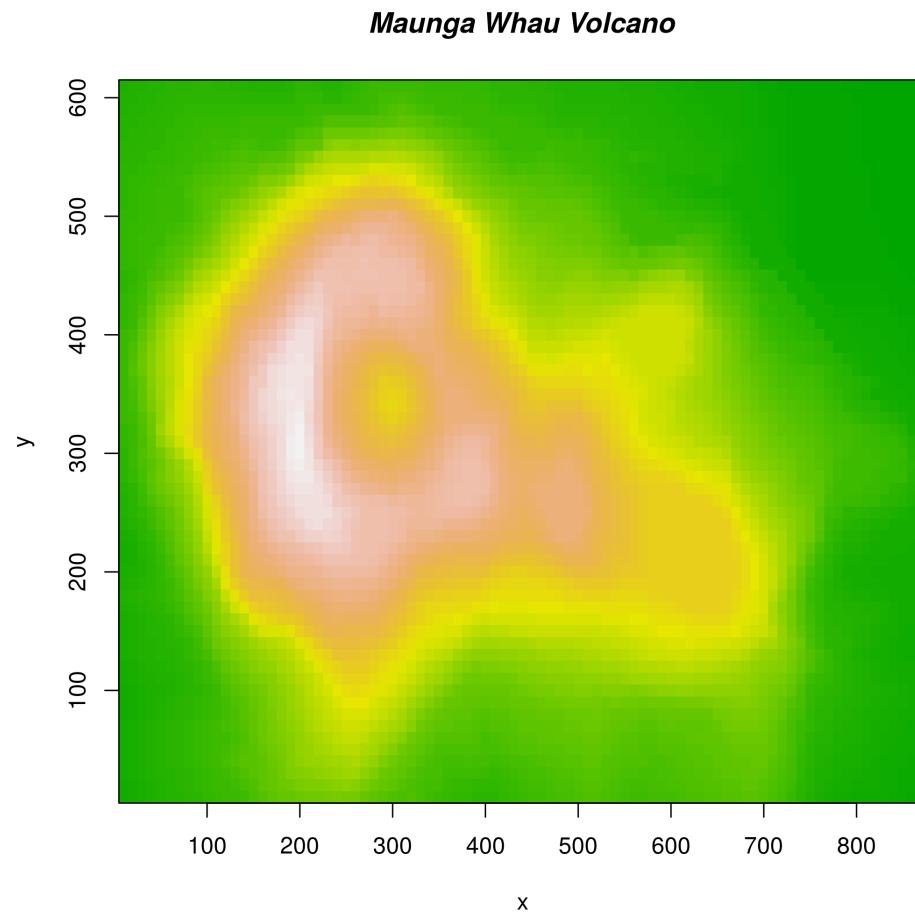
Diagrammes à bâtons (Bar plot)



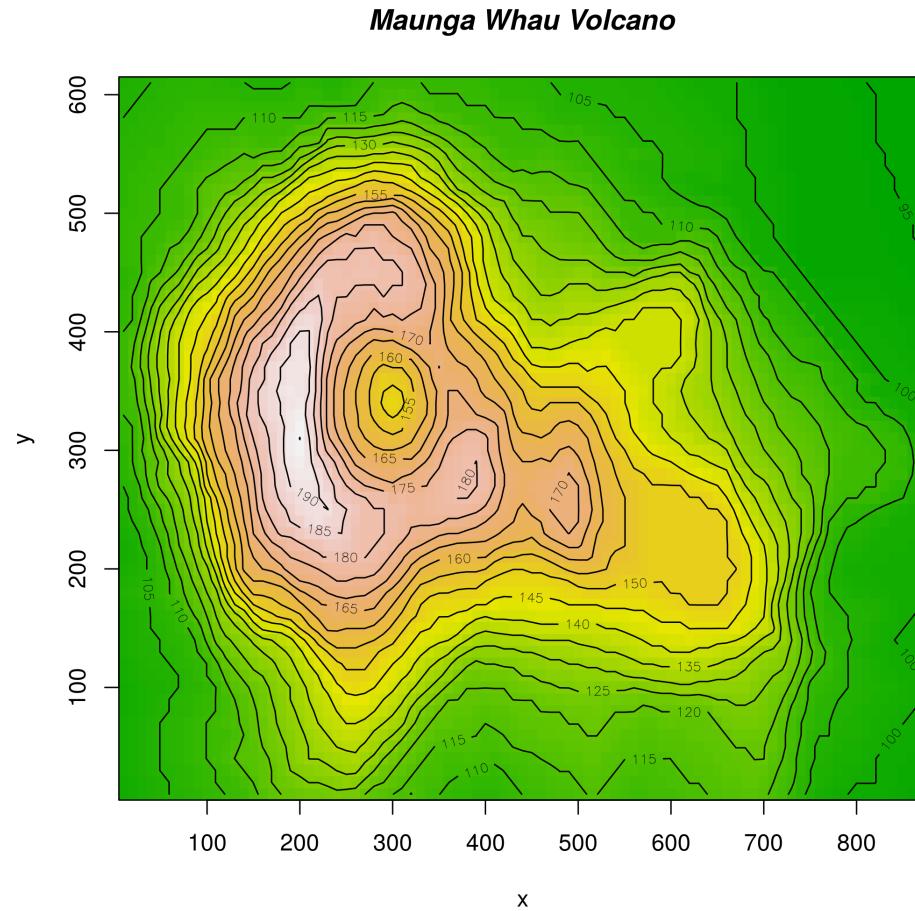
Histogrammes



Représentation 3-D



Lignes de contour



Faire une figure étape par étape avec R

Prépare les données adéquatement

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

Ouvrir une fenêtre graphique

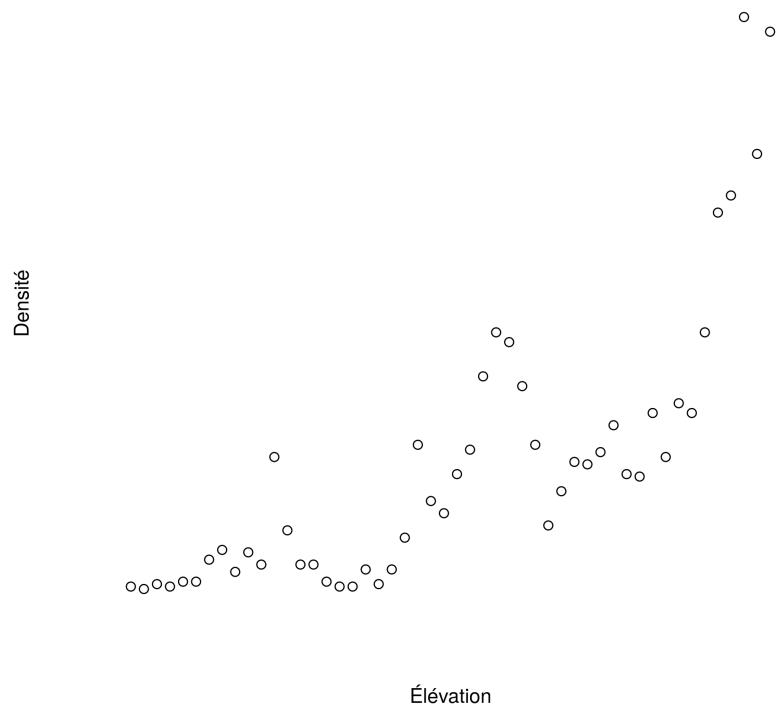
```
dev.new(width = 10, height = 7)
```

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))
```

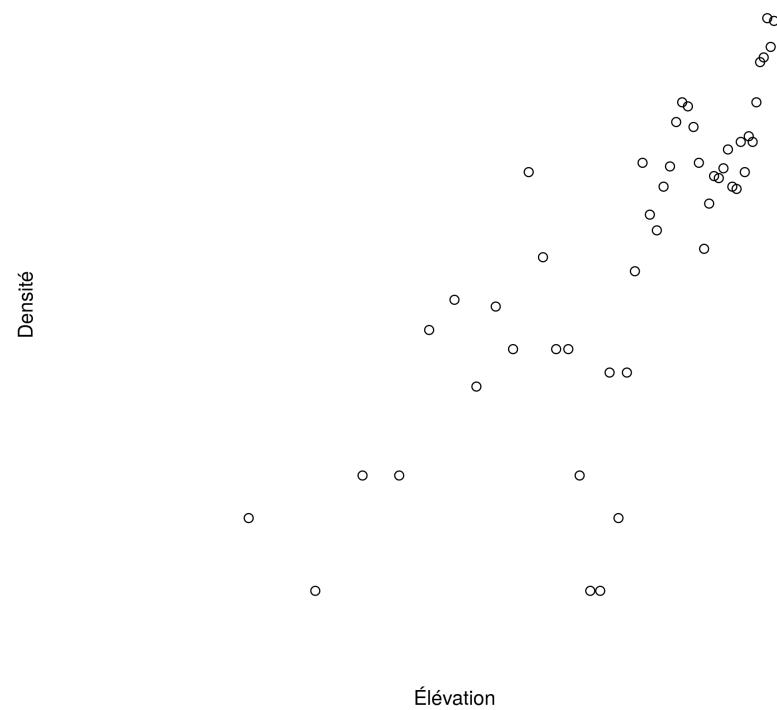
Démarrer une figure avec `plot()`

```
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

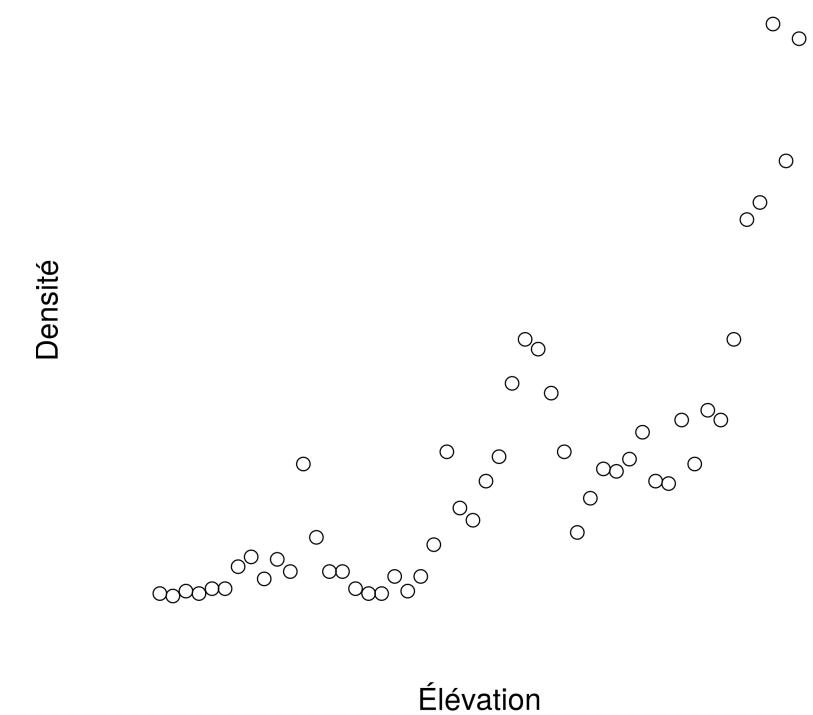
```
plot(elevation, densite[,1], axes = FALSE,  
      xlab = "Élévation", ylab = "Densité",  
      log = "xy")
```



Ajuster les tailles de caractères

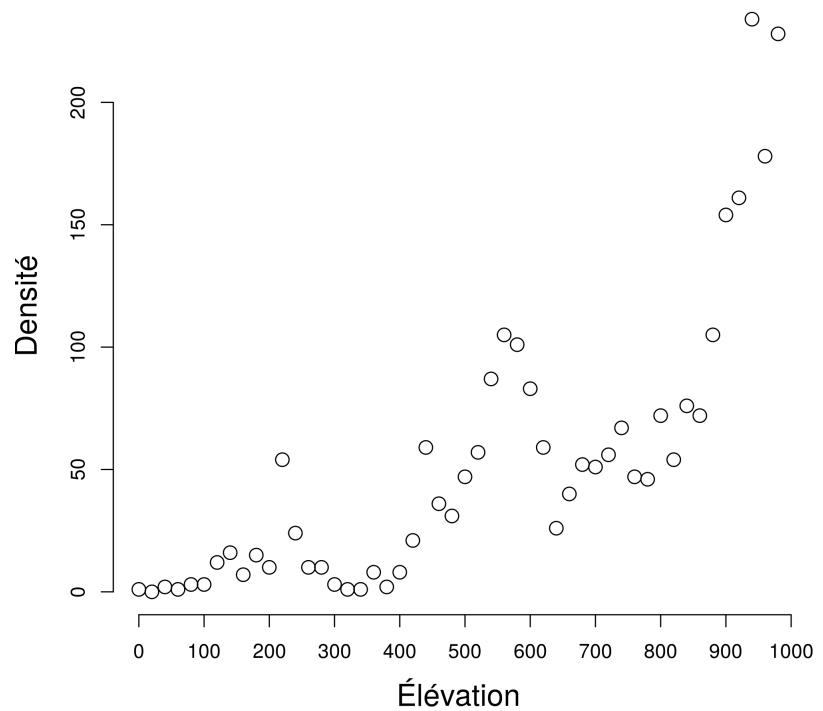
Arguments **cex**, **cex.lab** et **cex.axis**

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



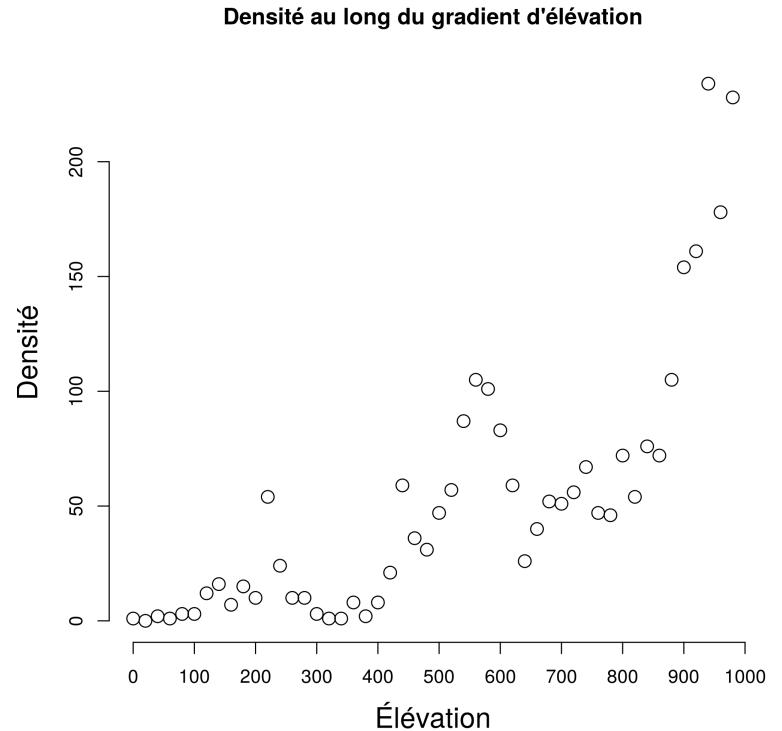
Modifier les axes

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



Ajouter un titre

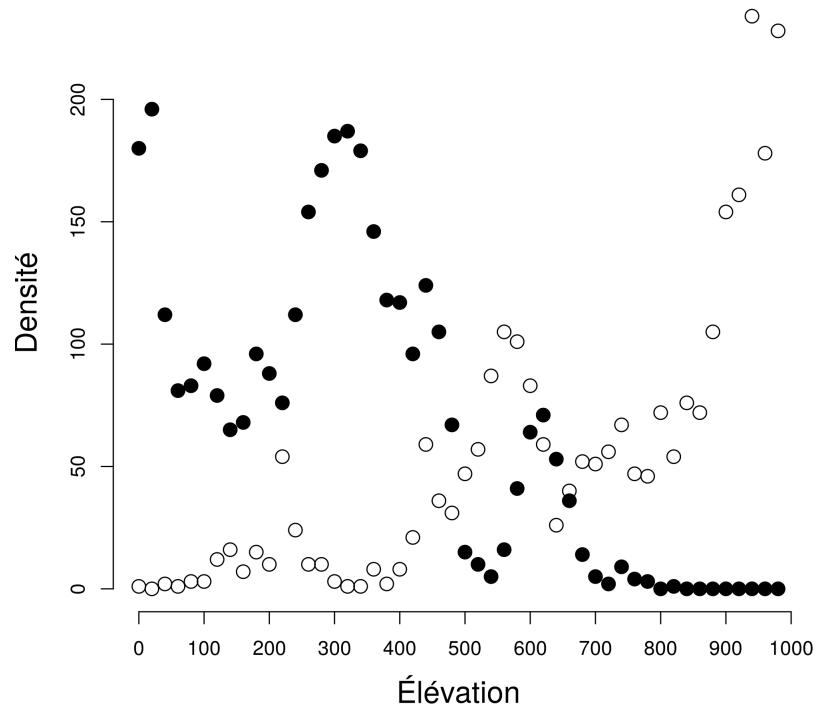
```
title(main = "Densité au long du gradient d'élévation")
```



Superposer des points d'une autre série de données

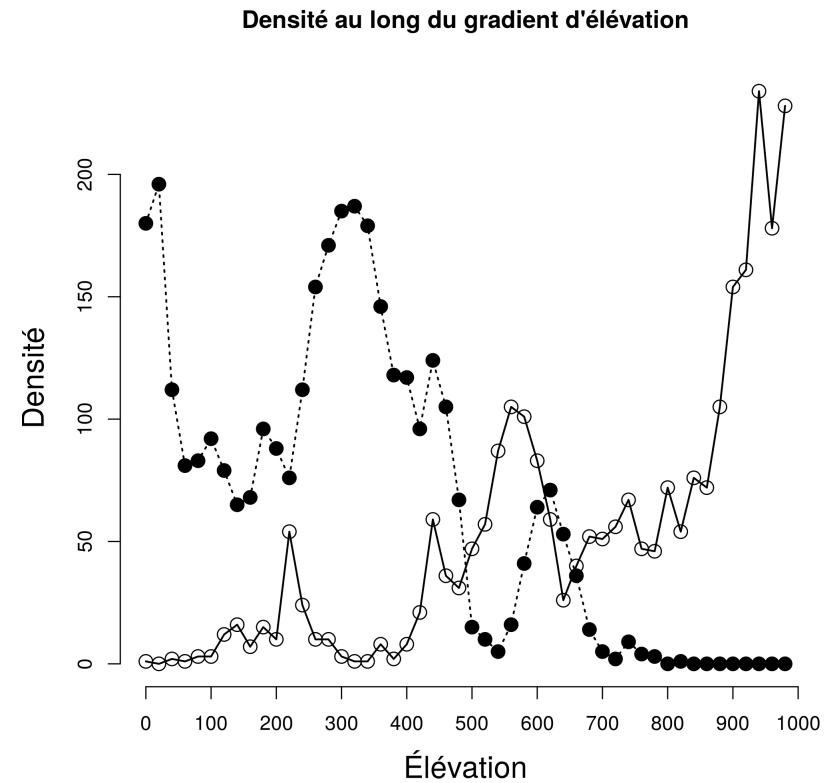
```
points(elevation, densite[,3], pch = 19, cex = 1.5)
```

Densité au long du gradient d'élévation



Superposer des lignes

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



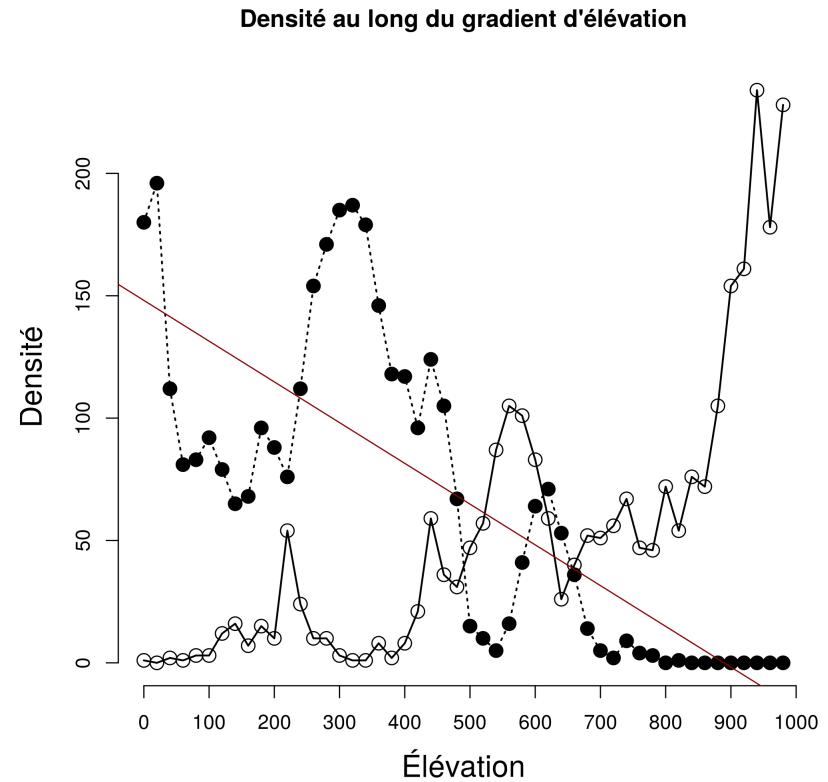
Ajouter une ligne de tendance

```
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
```

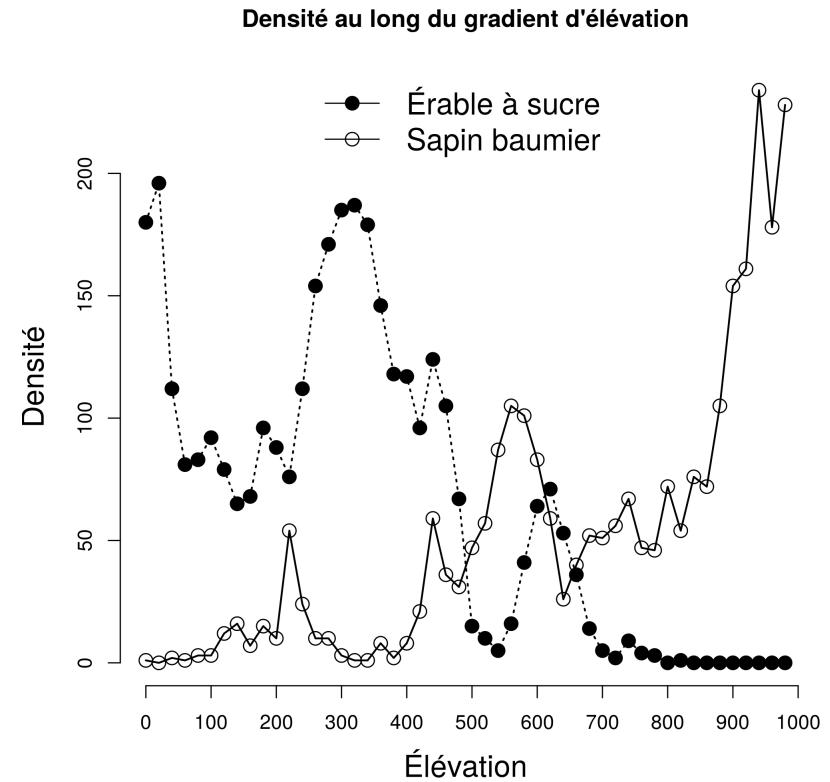
Ajouter une ligne de tendance

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



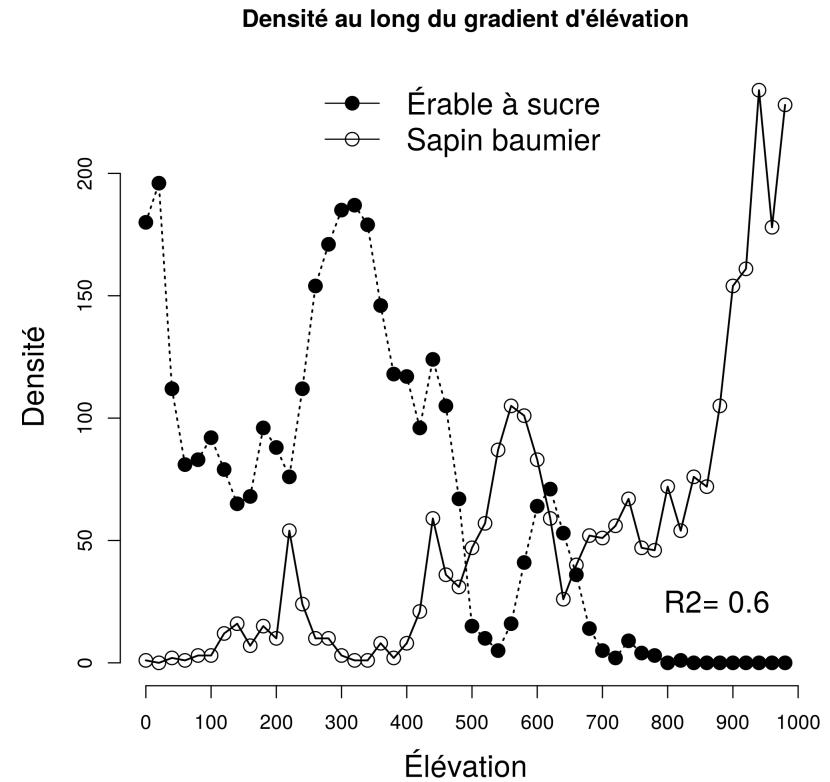
Ajouter une légende

```
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

Poursuite du travail de session

Consignes

- ✓ Identifiez clairement vos questions de recherche
- ✓ Illustrez le réseau de collaborations
- ✓ Complétez 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

Mills et al. 2015. Archiving Primary Data: Solutions for Long-Term Studies. Trends in Ecology and Evolution.

Poisot et al. 2013. Moving toward a sustainable ecological science : don't let data go to waste ! Ideas in Ecology and Evolution 6: 11-19.