Premiers pas

Timothée Flutre (INRA)

26/01/2018

Abstract

Ce document a pour but d'introduire concrètement les étudiants à l'un des aspects de la modélisation statistique, la simulation. Il prend comme exemple la régression linéaire simple, historiquement développée par Galton (1886) pour étudier l'hérédité de la taille dans l'espèce humaine.

Contents

1 Préambule

Ce document a été généré à partir d'un fichier texte au format Rmd utilisé avec le logiciel libre R. Pour exporter un tel fichier vers les formats HTML et PDF, installez le paquet rmarkdown (il va vraisemblablement vous être demandé d'installer d'autres paquets), puis ouvrez R et entrez:

```
library(rmarkdown)
render("premiers-pas.Rmd", "all")
```

Il est généralement plus simple d'utiliser le logiciel libre RStudio, mais ce n'est pas obligatoire. Pour plus de détails, lisez cette page.

Le format Rmd permet également d'utiliser le language LaTeX pour écrire des équations. Pour en savoir plus, reportez-vous au livre en ligne.

De plus, ce document nécessite de charger des paquets additionnels (ceux-ci doivent être installés au préalable sur votre machine, via `install.packages("pkg")`):

```
suppressPackageStartupMessages(library(MASS))
```

Il est également utile de savoir combien de temps est nécessaire pour exécuter tout le code R de ce document (voir l'annexe):

```
t0 <- proc.time()
```

2 Contexte

Ce document fait partie de l'atelier "Prédiction Génomique" organisé et animé par Jacques David et Timothée Flutre depuis 2015, avec l'aide de Julie Fiévet et Philippe Brabant, à Montpellier SupAgro dans le cadre de l'option APIMET (Amélioration des Plantes et Ingénierie végétale Méditerranéennes et Tropicales) couplée à la spécialité SEPMET (Semences Et Plants Méditerranéens Et Tropicaux) du Master 3A (Agronomie et Agroalimentaire), et de la spécialisation PIST du Cursus Ingénieur d'AgroparisTech.

Les versions du contenu sont gérées avec le logiciel git, et le dépôt central est hébergé sur GitHub.

3 Introduction

3.1 A propos de modélisation statistique

"Essentially, all models are wrong, but some are useful." (Box, 1987). Cette célèbre citation illustre parfaitement le fait qu'un modèle est une simplification du phénomène étudié, mais qu'après tout, si cette simplification nous apporte des enseignements et nous permet de prendre de bonnes décisions, cela importe tout autant.

Il est donc important de rappeler que la première question à se poser, en tant que modélisateur, concerne la validité du modèle. Bien que cela paraisse évident, ceci consiste avant tout à vérifier que les données à analyser correspondent bien à la question à laquelle on veut répondre (Gelman & Hill, 2006).

Il est également utile, pour mieux comprendre le processus de modélisation statistique, de distinguer le "monde réel", dans lequel vivent les données, du "monde théorique", dans lequel vivent les modèles: "When we use a statistical model to make a statistical inference, we implicitly assert that the variation exhibited by data is captured reasonably well by the statistical model, so that the theoretical world corresponds reasonably well to the real world." (Kass, 2011).

En particulier, il ne faut pas confondre les données avec des variables aléatoires, même si on fait souvent le raccourci: "In both approaches [frequentist and Bayesian], a statistical model is introduced and we may say that the inference is based on what *would* happen if the data *were* to be random variables distributed

according to the statistical model. This modeling assumption would be reasonable if the model *were* to describe accurately the variation in the data." (Kass, 2011).

3.2 Notation et vocabulaire

L'inférence avec un modèle statistique consiste généralement à **estimer les paramètres**, puis à s'en servir pour **prédire de nouvelles données**.

Lorsque l'on propose un modèle pour répondre à une question donnée, on commence donc par expliquer les notations. Suivant les conventions, nous utilisons des lettres grecques pour dénoter les paramètres (non-observés), par exemple θ, des lettres romaines pour les données observées, par exemple y, et des lettres romaines surmontées d'un tilde pour les données prédites, $\tilde{y}$. L'ensemble des données est généralement noté en majuscule, par exemple $\mathcal{D} = \{y_1, y_2, y_3\}$. De même, l'ensemble des paramètres est généralement noté en majuscule, par exemple $\Theta = \{\theta_1, \theta_2\}$. De plus, s'il y a plusieurs paramètres ou données, ils se retrouvent mathématiquement dans des vecteurs, en gras, ce qui donne $\boldsymbol{\theta}$ et $\boldsymbol{y}$. Par convention, les vecteurs sont en colonne.

Une fois les notations établies, on écrit la **vraisemblance** (*likelihood*), souvent présentée comme étant la "probabilité des données sachant les paramètres". En fait, si les données sont des variables continues, c'est la densité de probabilité des données sachant les paramètres, notée $p(y|\theta)$, et si les données sont des variables discrètes, c'est la fonction de masse, notée $P(y|\theta)$. Mais le plus important est de réaliser que, dans la vraisemblance, ce ne sont pas les données qui varient, mais les paramètres: la vraisemblance est une fonction des paramètres, d'où le fait qu'on la note $\mathcal{L}(\theta)$ ou $\mathcal{L}(\theta|y)$.

Tout naturellement, la méthode du **maximum de vraisemblance** cherche à identifier la valeur du paramètre, notée $\hat{\theta}$ par convention, qui maximise la vraisemblance:

$$\hat{\theta} = \text{argmax}_\theta \, \mathcal{L} \quad \Leftrightarrow \quad \frac{\partial \mathcal{L}}{\partial \theta}(\hat{\theta}) = 0 \tag{1}$$

Mais ceci n'est que le tout début de la démarche! En effet, certains décrivent les statistiques comme étant la "science de l'incertitude" (Lindley, 2000). Or pour l'instant, nous n'avons parlé que de la façon d'obtenir *une* valeur par paramètre, celle qui maximise la vraisemblance. Il est donc primordial ensuite de **quantifier l'incertitude** que nous avons quant à cette valeur. C'est sur ce point que différentes approches sont possibles (**fréquentiste**, **bayésienne**). Mais quelle que soit l'approche, si les échantillons sont "représentatifs" et le modèle n'est pas trop mauvais, on peut généralement s'attendre au fait que plus le nombre d'échantillons augmente, plus l'incertitude sur la valeur du paramètre diminue.

3.3 Comprendre la vraisemblance

Le vocabulaire esquissé au paragraphe précédent n'est pas forcément très intuitif. Par exemple, supposons que l'on étudie une quantité physique dont la valeur résulte de la somme d'une très grande quantité de facteurs indépendants, chacun ayant un faible impact sur la valeur finale. Prenons trois mesures de cette quantité d'intérêt. Comme il y a de la variation, on choisit d'introduire une **variable aléatoire** Y correspondant à la quantité d'intérêt, et dénotons par y_1, y_2 et y_3 les trois observations, vues comme des réalisations de cette variable aléatoire.

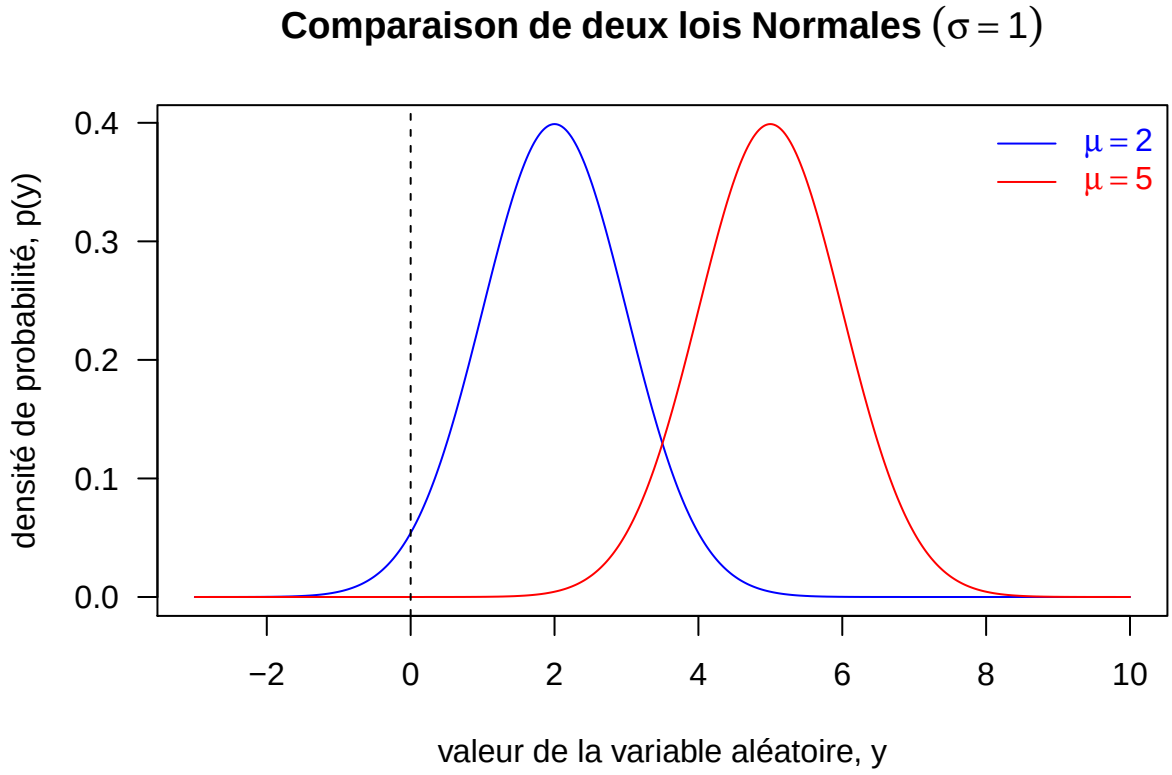
Par exemple, supposons les valeurs suivantes:

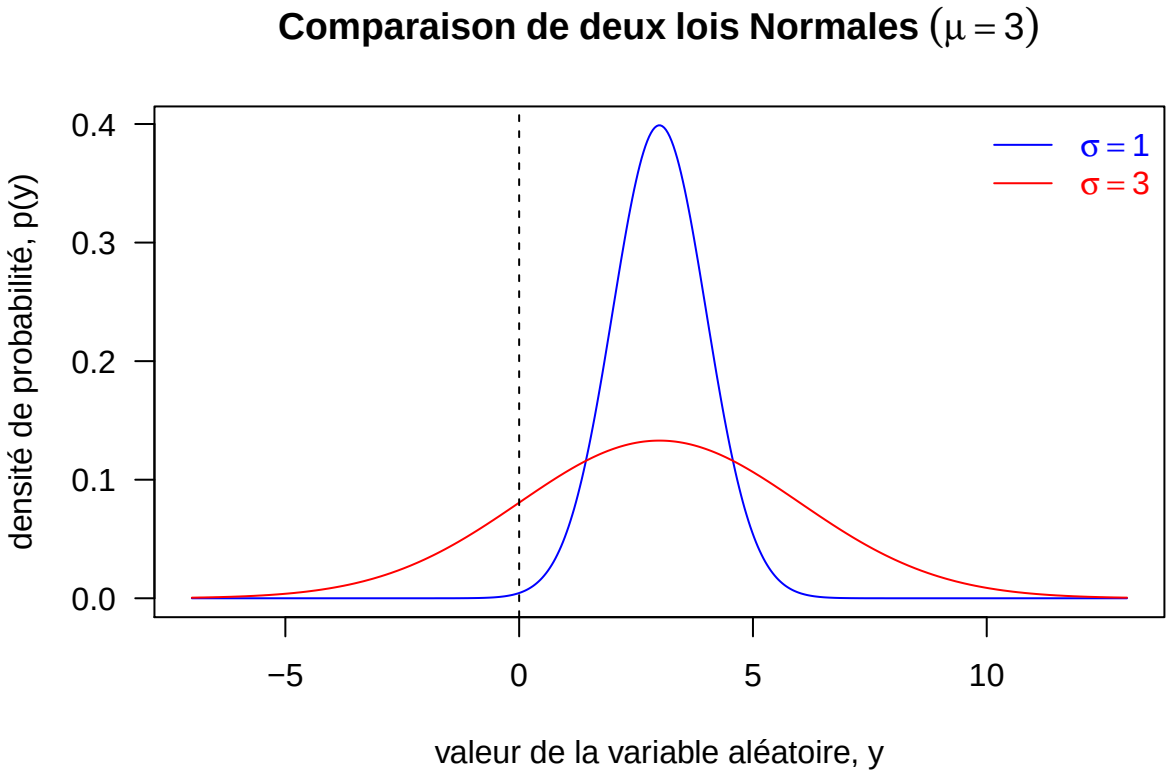
- $y_1 = 4.374$
- $y_2 = 5.184$
- $y_3 = 4.164$

Etant donné les caractéristiques du phénomène ("somme d'une très grande quantité de facteurs indépendants, chacun ayant un faible impact"), il est raisonnable de supposer que la variable Y suit une **loi Normale** (c.f. le théorème central limite). Cette distribution de probabilité est caractérisée par deux paramètres, sa **moyenne** que l'on note généralement μ, et sa **variance** que l'on note généralement σ^2 (σ étant l'écart-type). En terme de notation, on écrit $Y \sim \mathcal{N}(\mu, \sigma^2)$, et la densité de probabilité de la réalisation y de Y s'écrit:

$$Y \sim \mathcal{N}(\mu, \sigma^2) \quad \Leftrightarrow \quad p(Y = y \,|\, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) \tag{2}$$

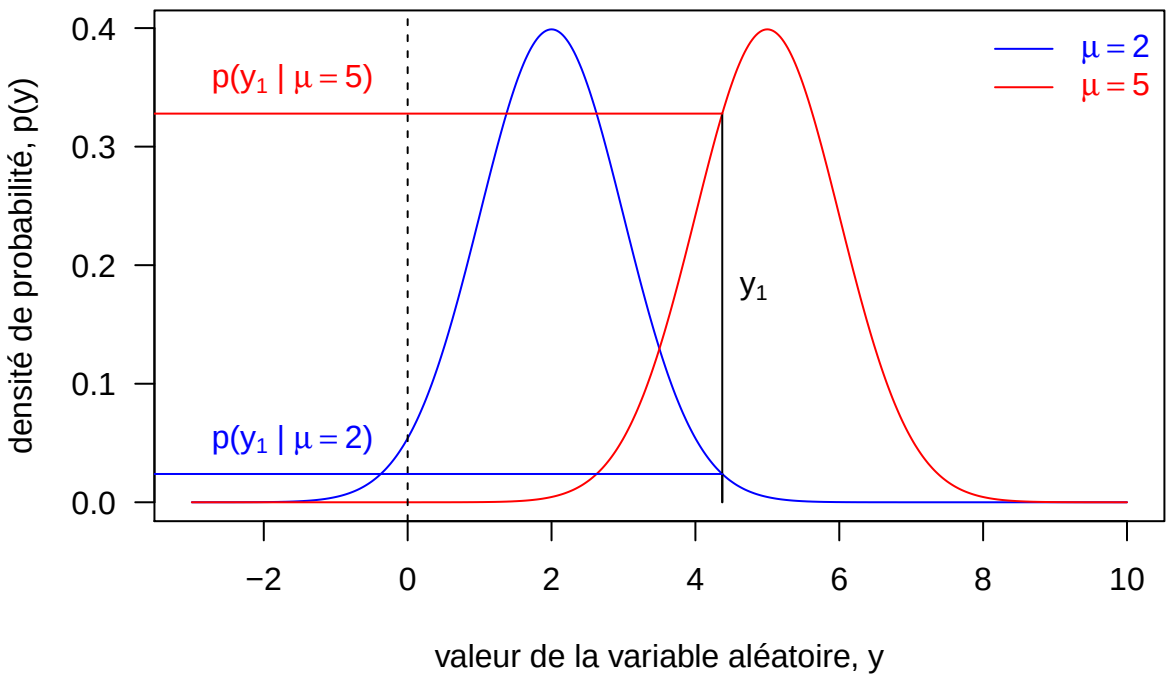
L'intérêt de ce modèle paramétrique est de pouvoir "résumer" les données, par exemple un million de mesures, par seulement deux valeurs, les paramètres. Mais bien entendu, nous ne connaissons pas les valeurs de paramètres! La moyenne μ peut prendre toutes les valeurs entre $-\infty$ et $+\infty$, et la variance σ^2 n'a pour seule restriction que d'être positive. Or comme le montrent les graphiques ci-dessous, la loi Normale peut être assez différente selon les valeurs de ces paramètres:





Revenons à nos trois mesures: 4.374, 5.184, 4.164. Parmi toutes les valeurs possibles des paramètres, le défi consiste donc à trouver celles pour lesquelles la loi Normale est une bonne description du mécanisme qui a généré ces données. On parle donc de trouver les valeurs des paramètres de telle sorte que la vraisemblance d'obtenir ces données soit la plus élevée possible pour ces valeurs des paramètres.

Pour rendre les choses plus facile, supposons que l'on connaisse déjà la variance: $\sigma^2 = 1$. Il ne nous reste plus qu'à trouver la moyenne, μ. Regardons cela de plus près pour la première observation, $y_1 = 4.374$:



D'après le graphique:

$$p(y_1 \,|\, \mu = 5, \sigma = 1) \; > \; p(y_1 \,|\, \mu = 2, \sigma = 1)$$

Cela se vérifie si l'on fait le calcul avec la formule (2):

- $p(y_1 \,|\, \mu = 5, \sigma = 1) = 0.328$;
- $p(y_1 \,|\, \mu = 2, \sigma = 1) = 0.024$.

Comme les deux densités ont la même valeur pour σ, la différence vient bien du terme $(y - \mu)^2$ dans l'exponentielle de (2), terme qui représente l'écart à la moyenne. Au final, nous pouvons donc dire qu'en ce qui concerne la première observation, la vraisemblance $\mathcal{L}(\mu = 5, \sigma = 1)$ est plus grande que $\mathcal{L}(\mu = 2, \sigma = 1)$.

Comme on dispose de plusieurs observations, $\{y_1, y_2, y_3\}$, et qu'on suppose qu'elles sont toutes des réalisations de la même variable aléatoire, Y, il est pertinent de calculer la vraisemblance de toutes ces observations conjointement plutôt que séparément:

$$\mathcal{L}(\mu, \sigma) = p(y_1, y_2, y_3 \,|\, \mu, \sigma) \tag{3}$$

Si l'on fait aussi l'hypothèse que ces observations sont indépendantes, cela se simplifie en:

$$\mathcal{L}(\mu, \sigma) = p(y_1 \,|\, \mu, \sigma) \times p(y_2 \,|\, \mu, \sigma) \times p(y_3 \,|\, \mu, \sigma) \tag{4}$$

$$= \prod_{i=1}^{3} p(y_i \,|\, \mu, \sigma) \tag{5}$$

Il n'est pas très pratique de maximiser la vraisemblance directement, on préfère passer au log (qui est monotone, donc le maximum de l'un est aussi le maximum de l'autre):

$$
\begin{aligned}
l(\mu, \sigma) &= \log \mathcal{L}(\mu, \sigma) && (6)\\
&= \sum_{i=1}^{3} \log p(y_i \mid \mu, \sigma) && (7)\\
&= \sum_{i=1}^{3} \log \left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - \mu)^2}{2\sigma^2} \right) \right] && (8)\\
&= -3 \log \sigma \; - \frac{3}{2} \log(2\pi) \; - \frac{1}{2\sigma^2} \sum_{i=1}^{3} (y_i - \mu)^2 && (9)
\end{aligned}
$$

Lorsque le nombre d'observations augmente, un simple examen graphique n'est pas très pratique ni suffisant. D'où le fait, qu'en pratique, (i) on écrit une fonction qui calcule la log-vraisemblance, et (ii) on cherche le maximum de cette fonction:

```
compute.log.likelihood <- function(parameters, data){
  mu <- parameters[1]
  sigma <- parameters[2]
  y <- data
  n <- length(y)
  log.lik <- - n * log(sigma) - (n/2) * log(2 * pi) - sum(((y - mu)^2) / (2 * sigma^2))
  return(log.lik)
}

compute.log.likelihood(c(5,1), y)
```

```
## [1] -3.32
```

```
compute.log.likelihood(c(2,1), y)
```

```
## [1] -13
```

Dans le cas de la loi Normale, il existe déjà dans R des fonctions implémentant la densité de probabilité, ce qui nous permet de vérifier que nous n'avons pas fait d'erreur:

```
sum(dnorm(x=y, mean=5, sd=1, log=TRUE))
```

```
## [1] -3.32
```

```
sum(dnorm(x=y, mean=2, sd=1, log=TRUE))
```

```
## [1] -13
```

Cette section devrait vous avoir donné les bases du raisonnement ainsi que des techniques que nous allons utiliser dans la suite de l'atelier, commençant dès maintenant avec la régression linéaire simple.

4 Ecrire le modèle

4.1 Notations

- n: nombre d'individus (diploïdes, supposés non-apparentés)
- i: indice indiquant le i-ème individu, donc $i \in \{1, \ldots, n\}$
- y_i: phénotype de l'individu i pour la caractère d'intérêt

- μ: moyenne globale du phénotype des n individus
- f: fréquence de l'allèle minoritaire au marqueur SNP d'intérêt (situé sur un autosome)
- x_i: génotype de l'individu i à ce SNP, codé comme le nombre de copie(s) de l'allèle minoritaire à ce SNP chez cet individu ($\forall i \; x_i \in \{0, 1, 2\}$)
- β: effet additif de chaque copie de l'allèle minoritaire en unité du phénotype
- ϵ_i: erreur pour l'individu i
- σ^2: variance des erreurs

Données: $\mathcal{D} = \{(y_1 \,|\, x_1), \ldots, (y_n \,|\, x_n)\}$

Paramètres: $\Theta = \{\mu, \beta, \sigma\}$

4.2 Vraisemblance

Dans notre cas, nous supposons que le génotype au SNP d'intérêt a un effet additif sur la moyenne du phénotype, ce qui s'écrit généralement:

$$\forall i \;\; y_i = \mu + \beta x_i + \epsilon_i \text{ avec } \epsilon_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, \sigma^2) \tag{10}$$

Une autre façon, mais équivalente, de l'écrire est:

$$\forall i \;\; y_i \,|\, x_i, \mu, \beta, \sigma \overset{\text{i.i.d}}{\sim} \mathcal{N}(\mu + \beta x_i, \sigma^2) \tag{11}$$

Cette deuxième formulation montre bien que, de manière générale, si l'on connaît les valeurs des variables explicatives ainsi que celles des paramètres, alors on est capable de simuler des valeurs de réponse.

De plus, on peut écrire la vraisemblance sous forme plus explicite comme une fonction des paramètres $\Theta = \{\mu, \beta, \sigma\}$:

$$\begin{aligned}
\mathcal{L}(\Theta \,|\, \mathcal{D}) &= p(\boldsymbol{y} \,|\, \boldsymbol{x}, \Theta) && (12)\\
&= \prod_{i=1}^{n} p(y_i \,|\, x_i, \mu, \beta, \sigma) && (13)\\
&= \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y_i - (\mu + \beta x_i))^2}{2\sigma^2}\right) && (14)
\end{aligned}$$

5 Simuler des données

Afin de simuler des données, nous allons utiliser un générateur de nombres pseudo-aléatoires. Le mot "pseudo" est là pour rappeler que les générateurs informatiques sont déterministes et peuvent donc être initialisés avec une graine (*seed*), très utile pour la reproductibilité des analyses:

```
set.seed(1866) # année de parution de l'article de Mendel fondant la génétique
```

Commençons par fixer le nombre d'individus, n:

```
n <- 200
```

Puis la moyenne globale, μ (de manière arbitraire, ce n'est pas très important car on peut toujours centrer les phénotypes en début d'analyse):

```
mu <- 50
```

Pour simuler les génotypes, $\boldsymbol{x}$, nous allons supposer que la population est à l'équilibre d'Hardy-Weinberg:

```
##' Genotype frequencies
##'
##' Calculate the genotype frequencies at a locus assuming the Hardy-Weinberg equilibrium
##' (https://en.wikipedia.org/wiki/Hardy%E2%80%93Weinberg_principle).
##' @param maf frequency of the minor allele, a
##' @return vector of genotype frequencies
##' @author Timothee Flutre
calcGenoFreq <- function(maf){
  stopifnot(is.numeric(maf), length(maf) == 1, maf >= 0, maf <= 0.5)
  geno.freq <- c((1 - maf)^2,
                 2 * (1 - maf) * maf,
                 maf^2)
  names(geno.freq) <- c("AA", "Aa", "aa")
  return(geno.freq)
}

f <- 0.3
genotypes <- sample(x=c(0,1,2), size=n, replace=TRUE, prob=calcGenoFreq(f))
```

Le morceau de code ci-dessus vous montre aussi les bonnes pratiques de programmation:

- choisir des noms de fonctions et variables clairs et explicites;
- documenter le code;
- citer des référence si nécessaire;
- vérifier la validité des arguments au début d'une fonction.

Regardons à quoi ressemblent les génotypes que nous venons de simuler:

```
table(genotypes)
```

```
## genotypes
##   0   1   2
## 102  80  18
```

```
sum(genotypes) / (2 * n) # estimate of the MAF
```

```
## [1] 0.29
```

```
var(genotypes) # important for the estimate of beta
```

```
## [1] 0.426
```

Tirons une valeur pour l'effet du génotype sur le phénotype, β:

```
(beta <- rnorm(n=1, mean=2, sd=1))
```

```
## [1] 2.45
```

Simulons les erreurs, $\boldsymbol{\epsilon}$ (par simplicité, fixons σ à 1):

```r
sigma <- 1
errors <- rnorm(n=n, mean=0, sd=sigma)
```

Nous avons maintenant tout ce qu'il faut pour simuler les phénotypes, $\boldsymbol{y}$, via (10):

```r
phenotypes <- mu + beta * genotypes + errors
```

Pour la suite, il est habituel dans R d'organiser les données dans un tableau:

```r
dat <- data.frame(x=genotypes, y=phenotypes)
summary(dat)
```

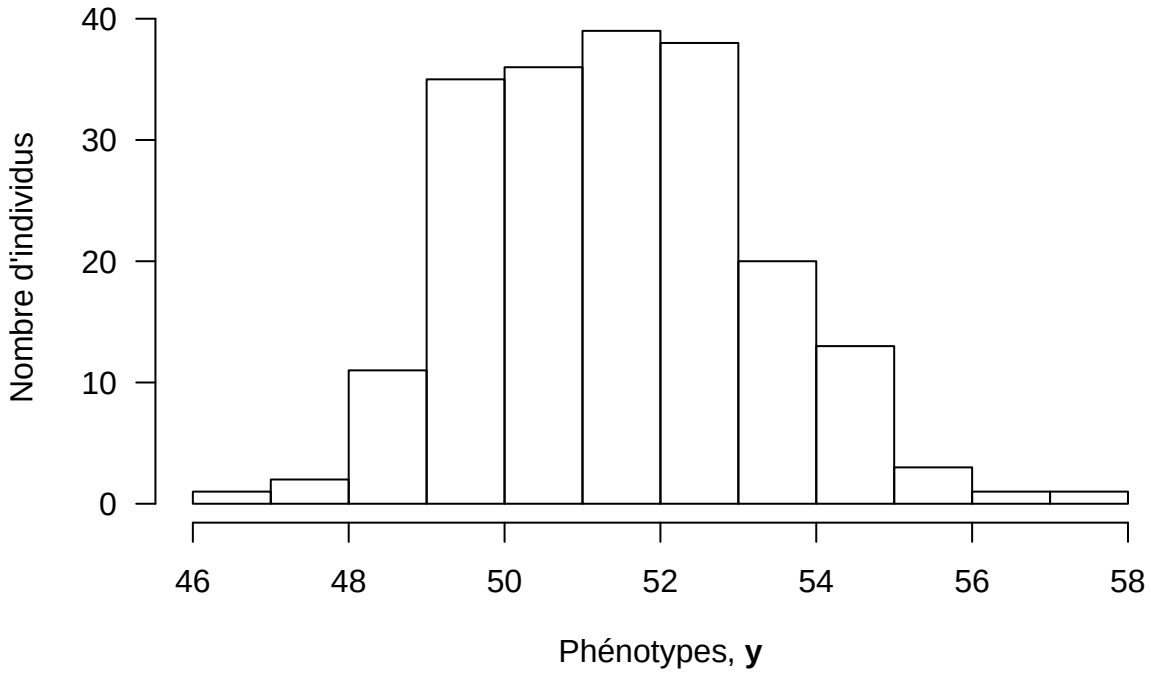
```
##        x              y        
##  Min.   :0.00   Min.   :46.7  
##  1st Qu.:0.00   1st Qu.:50.0  
##  Median :0.00   Median :51.3  
##  Mean   :0.58   Mean   :51.4  
##  3rd Qu.:1.00   3rd Qu.:52.7  
##  Max.   :2.00   Max.   :57.1
```

6 Réaliser l'inférence

6.1 Visualisation graphique

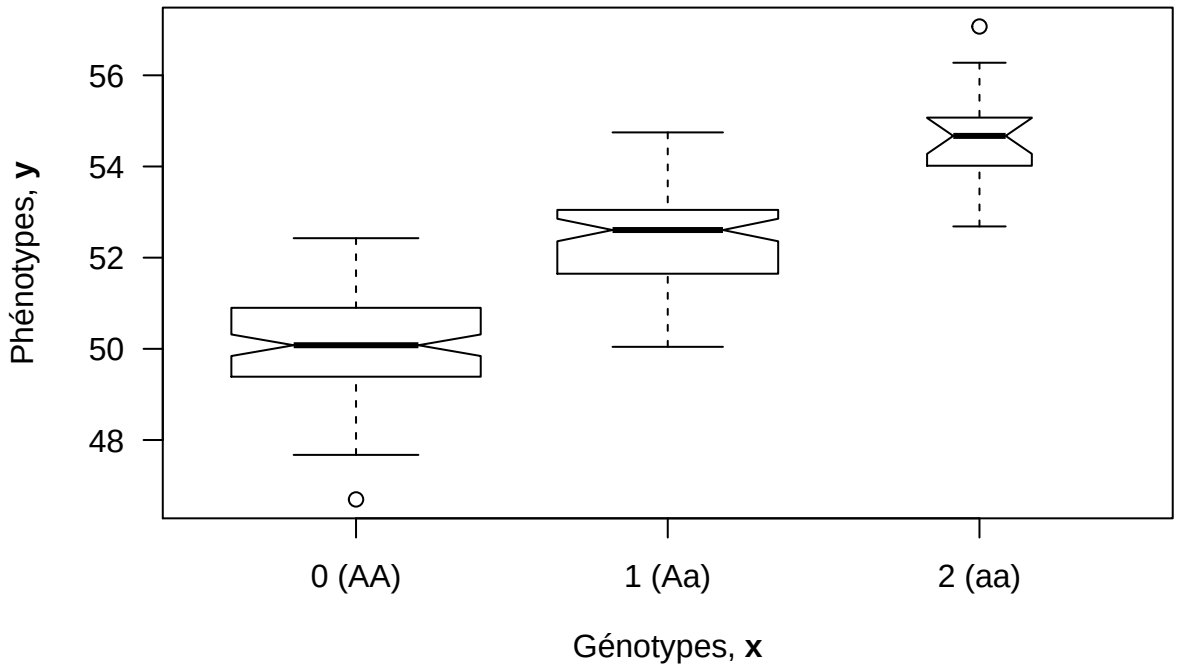
Avant toute autre chose, regardons à quoi ressemblent les données:

```r
hist(phenotypes, breaks="FD", las=1, main="",
     xlab=expression(paste("Phénotypes, ", bold(y))),
     ylab="Nombre d'individus")
```



Comme ce qui nous intéresse ici, ce ne sont pas uniquement les phénotypes, mais bien la relation entre les génotypes et les phénotypes, un autre type de graphique semble plus approprié:

```r
boxplot(phenotypes ~ genotypes,
        xlab=expression(paste("Génotypes, ", bold(x))),
        ylab=expression(paste("Phénotypes, ", bold(y))),
        varwidth=TRUE, notch=TRUE, las=1, xaxt="n", at=0:2)
axis(side=1, at=0:2, labels=c("0 (AA)", "1 (Aa)", "2 (aa)"))
```



Remarquez l'importance de faire des graphiques les plus clairs et intelligibles possible. Ce ne sont souvent que les graphiques que l'on montre à la place des tableaux de résultats, ceux-ci étant connus pour être plus difficiles à lire (Gelman *et al.*, 2002).

6.2 Dérivation mathématique

Pour trouver les valeurs des paramètres qui maximisent la vraisemblance (14), on travaille généralement avec la log-vraisemblance, $l(\Theta) = \log \mathcal{L}(\Theta)$, puis on utilise les règles d'analyse:

- $\frac{\partial l}{\partial \beta}(\hat{\beta}) = 0 \;\Leftrightarrow\; \hat{\beta} = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$
- $\frac{\partial l}{\partial \mu}(\hat{\mu}) = 0 \;\Leftrightarrow\; \hat{\mu} = \bar{y} - \hat{\beta}\bar{x}$
- $\frac{\partial l}{\partial \sigma}(\hat{\sigma}) = 0 \;\Leftrightarrow\; \hat{\sigma} = \frac{1}{n}\sum_i (y_i - (\hat{\mu} + \hat{\beta} x_i))^2$

où $\bar{x} = \frac{1}{n}\sum_i x_i$ et $\bar{y} = \frac{1}{n}\sum_i y_i$.

Une fois ces estimations obtenues, on veut généralement faire deux choses. La première est d'évaluer la qualité du modèle, c'est-à-dire son ajustement aux données. Dans le cas de la régression linéaire, ceci peut se faire à l'aide du coefficient de détermination, interprété comme la proportion de la variance phénotypique expliquée par les prédicteurs:

$$R^2 = \frac{Var(\hat{\mu} + \hat{\beta}\boldsymbol{x})}{Var(\boldsymbol{y})} = \frac{\hat{\beta}^2\, Var(\boldsymbol{x})}{\hat{\beta}^2\, Var(\boldsymbol{x}) + \hat{\sigma}^2} \quad \Leftrightarrow \quad R^2 = \frac{\sum_i ((\hat{\mu} + \hat{\beta} x_i) - \bar{y})^2}{\sum_i (y_i - \bar{y})^2} \tag{15}$$

La deuxième est de quantifier l'incertitude autour de ces estimations. Dans la démarche fréquentiste, on considère que les estimations sont des réalisations de variables aléatoires appelées "estimateurs". Dans le cas de la régression linéaire, ces estimateurs, M pour μ, B pour β et S pour σ, sont des combinaisons linéaires des $\{Y_i\}$, variables indépendantes et suivant une loi Normale, donc les estimateurs suivent eux aussi une loi Normale. Par exemple, pour l'estimateur B du paramètre β:

$$B = \frac{\sum_i (x_i - \bar{x})(Y_i - \bar{Y})}{\sum_i (x_i - \bar{x})^2} \quad \text{et} \quad B \sim \mathcal{N}(E_B, V_B) \tag{16}$$

Il reste alors à calculer l'espérance et la variance de la loi Normale pour chaque estimateur, et c'est notamment à partir de la variance de l'estimateur que l'on obtient l'erreur standard de l'estimation, celle-ci permettant de quantifier l'incertitude autour de l'estimation.

Pour en savoir plus, reportez-vous par exemple au livre "Statistique inférentielle" de Daudin, Robin et Vuillet (Presses Universitaires de Rennes, 2001). Du matériel pédagogique sur le modèle linéaire est aussi disponible en accès libre sur le site internet de la Société Française de Statistique (ici).

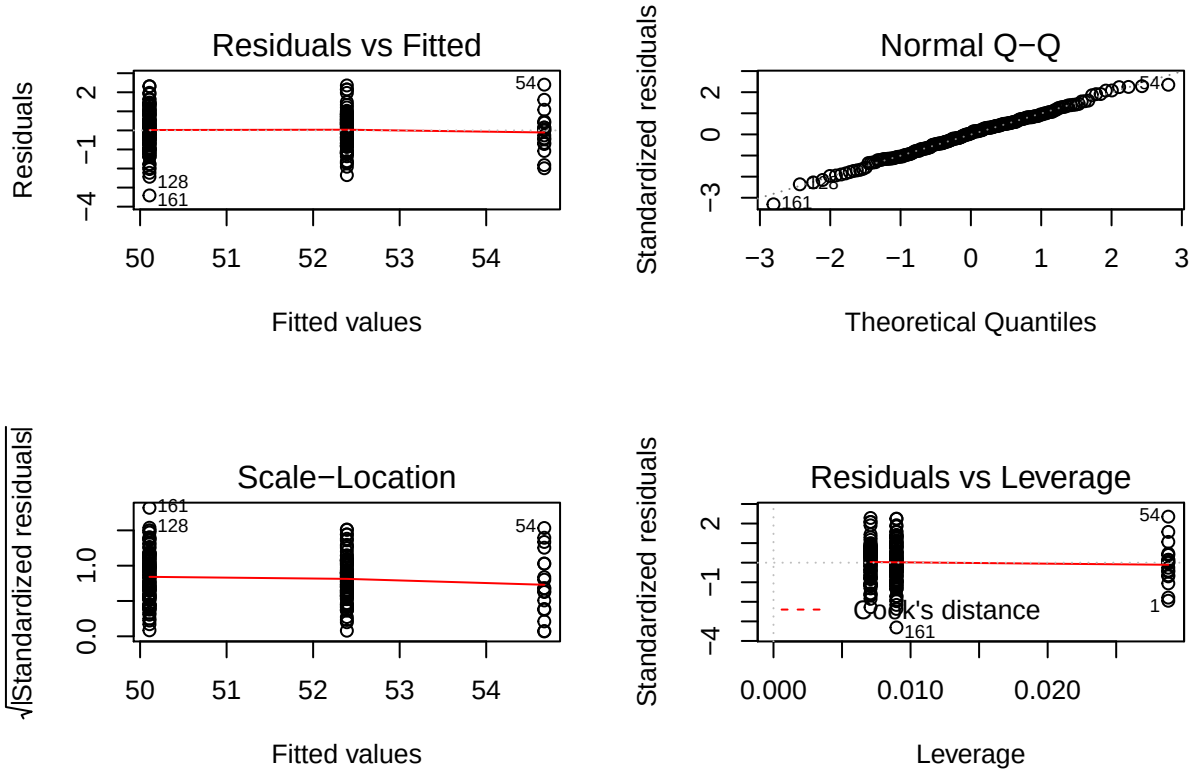
6.3 Implémentation (facile)

Bien entendu, R a nativement une fonction implémentant l'estimation par maximum de vraisemblance d'un modèle linéaire, lm:

```
fit <- lm(y ~ x, data=dat)
```

Avant toute autre chose, il nous faut vérifier que les hypothèses du modèles sont vérifiées (homoscédasticité, Normalité, indépendance):

```
par(mfrow=c(2,2))
plot(fit)
```



Ceci semble être bien le cas (évidemment puisque nous avons simulé nous-même les données !).

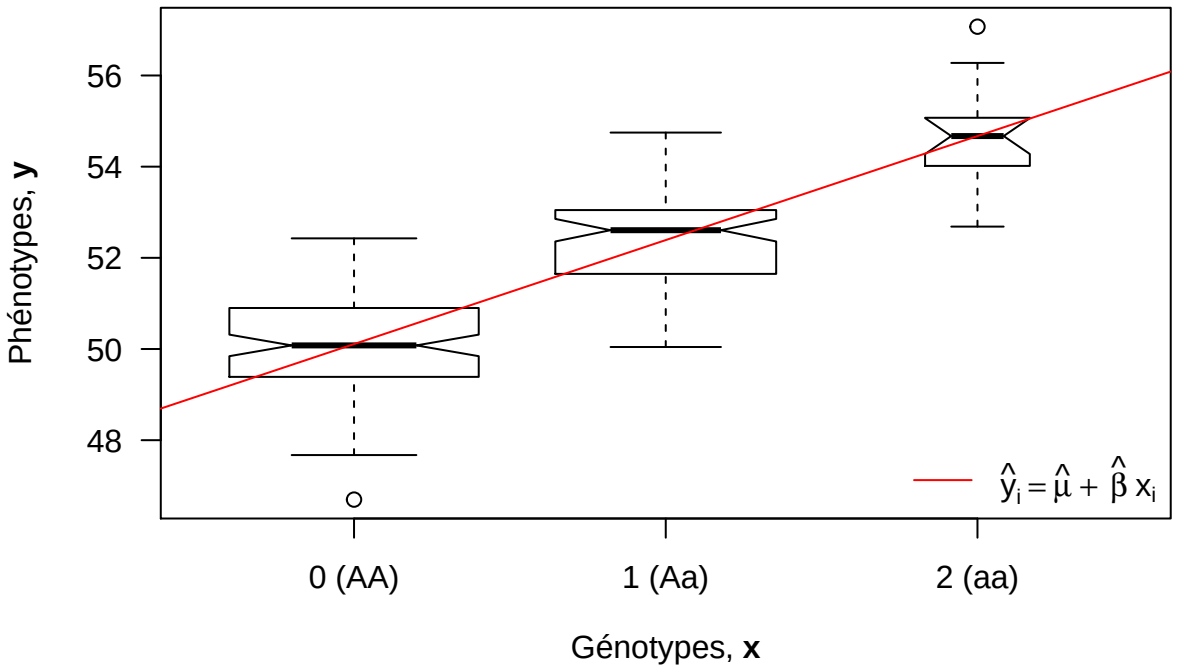
Regardons donc les résultats:

```
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x, data = dat)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.411 -0.701  0.055  0.688  2.399
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  50.1085     0.0981   511.0   <2e-16 ***
## x             2.2814     0.1125    20.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.04 on 198 degrees of freedom
## Multiple R-squared:  0.675,  Adjusted R-squared:  0.673
## F-statistic:  411 on 1 and 198 DF,  p-value: <2e-16
```

Rajoutons la droite de régression au graphique des données:

```
boxplot(phenotypes ~ genotypes,
        xlab=expression(paste("Génotypes, ", bold(x))),
        ylab=expression(paste("Phénotypes, ", bold(y))),
        varwidth=TRUE, notch=TRUE, las=1, xaxt="n", at=0:2)
axis(side=1, at=0:2, labels=c("0 (AA)", "1 (Aa)", "2 (aa)"))
abline(a=coefficients(fit)[1], b=coefficients(fit)[2], col="red")
legend("bottomright",
       legend=expression(hat(y)[i]==hat(mu)~+~hat(beta)~x[i]),
       col="red", lty=1, bty="n")
```



6.4 Implémentation (plus dure)

Il peut être intéressant, surtout dans ce cas simple, d'implémenter cette méthode par soi-même pour mieux la comprendre. Pour cela, on peut utiliser la fonction mle du paquet stats4.

Il faut d'abord écrire une fonction calculant l'opposé de la log-vraisemblance:

```
negLogLik <- function(mu, beta, sigma){
  - sum(dnorm(x=dat$y, mean=mu + beta * dat$x, sd=sigma, log=TRUE))
}
```

Puis demander à la fonction `mle` de la maximiser (en spécifiant que le paramètre σ ne peut pas être négatif ou nul):

```
library(stats4)
fit2 <- mle(negLogLik, start=list(mu=mean(dat$y), beta=0, sigma=1),
            method="L-BFGS-B", nobs=nrow(dat),
            lower=c(-Inf,-Inf,10^(-6)), upper=c(+Inf,+Inf,+Inf))
summary(fit2)
```

```
## Maximum likelihood estimation
##
## Call:
## mle(minuslogl = negLogLik, start = list(mu = mean(dat$y), beta = 0,
##     sigma = 1), method = "L-BFGS-B", nobs = nrow(dat), lower = c(-Inf,
##     -Inf, 10^(-6)), upper = c(+Inf, +Inf, +Inf))
##
## Coefficients:
##       Estimate Std. Error
## mu       50.11     0.0976
## beta      2.28     0.1119
## sigma     1.03     0.0515
##
```

```
## -2 log L: 579
```

7 Evaluer les résultats

7.1 Sélection de modèles

Cette première étape a à voir avec l'ajustement du modèle aux données. Dans notre cas de régression linéaire simple, on peut utiliser le coefficient de détermination:

```
summary(fit)$r.squared
```

```
## [1] 0.675
```

On peut facilement vérifier que cette valeur renvoyée par la fonction `lm` correspond à la formule (15):

```
(coefficients(fit)[2]^2 * var(dat$x)) /
  (coefficients(fit)[2]^2 * var(dat$x) + summary(fit)$sigma^2)
```

```
##     x
## 0.674
```

Dans le cas où plusieurs modèles sont intéressants, on peut essayer de vouloir en sélectionner un s'il paraît "meilleur" que les autres. Cette étape ne sera pas discutée plus avant ici, mais pour en savoir plus, lisez ce que vous trouverez sur internet en cherchant les termes "AIC", "BIC" et "validation croisée" (*cross validation*).

7.2 Estimation de paramètres

Prenons l'exemple du paramètre β, de son estimateur du maximum de vraisemblance B et de son estimation $\hat{\beta}$. Comme nous avons simulé les données, nous connaissons la vraie valeur du paramètre:

```
beta
```

```
## [1] 2.45
```

Après avoir ajusté le modèle avec la fonction `lm()`, nous pouvons récupérer l'estimation de ce paramètre:

```
(beta.hat <- coefficients(fit)[2])
```

```
##    x
## 2.28
```

Pour comparer les deux, on définit une fonction de perte (*loss function*) reliant le paramètre, β, et son estimateur, B. On utilise communément une fonction quadratique, dont on prend l'espérance, ce qui donne l'erreur quadratique moyenne (*mean squared error*):

$$MSE = E\left((B-\beta)^2\right) \tag{17}$$

Calculons sa racine carrée afin que le résultat soit dans la même unité que le paramètre:

```
(rmse.beta <- sqrt((beta.hat - beta)^2))
```

```
##     x
## 0.172
```

7.3 Prédiction de données

Nous pouvons aussi calculer l'erreur quadratique moyenne avec les phénotypes déjà observés (on parle de *in-sample predictions*):

```
y <- phenotypes
y.hat <- (coefficients(fit)[1] + coefficients(fit)[2] * genotypes)
errors <- y - y.hat
(rmse.y <- sqrt(mean(errors^2)))
```

```
## [1] 1.03
```

Plus facilement, on peut aussi utiliser la fonction `predict`:

```
errors <- phenotypes - predict(fit)
(rmse.y <- sqrt(mean(errors^2)))
```

```
## [1] 1.03
```

Mais, de façon plus intéressante, nous voudrions évaluer les prédictions phénotypiques sur n_{new} nouveaux individus. Pour cela, commençons par simuler de nouvelles données, $\mathcal{D}_{\text{new}} = \{(y_{i,\text{new}} \,|\, x_{i,\text{new}})\}$, toujours avec les *mêmes* "vraies" valeurs des paramètres, $\Theta = \{\mu, \beta, \sigma\}$:

```
set.seed(1944) # année de découverte de l'ADN comme support des gènes
n.new <- 100
x.new <- sample(x=c(0,1,2), size=n.new, replace=TRUE, prob=calcGenoFreq(f))
y.new <- mu + beta * x.new + rnorm(n=n.new, mean=0, sd=sigma)
```

Puis utilisons les estimations des paramètres obtenues précédemment pour prédire les nouveaux phénotypes à partir des nouveaux génotypes, $\tilde{\mathcal{D}}_{\text{new}} = \{(\tilde{y}_{i,\text{new}} = \hat{\mu} + \hat{\beta}\, x_{i,\text{new}})\}$ (*out-of-sample predictions*):

```
y.new.tilde <- (coefficients(fit)[1] + coefficients(fit)[2] * x.new)
```

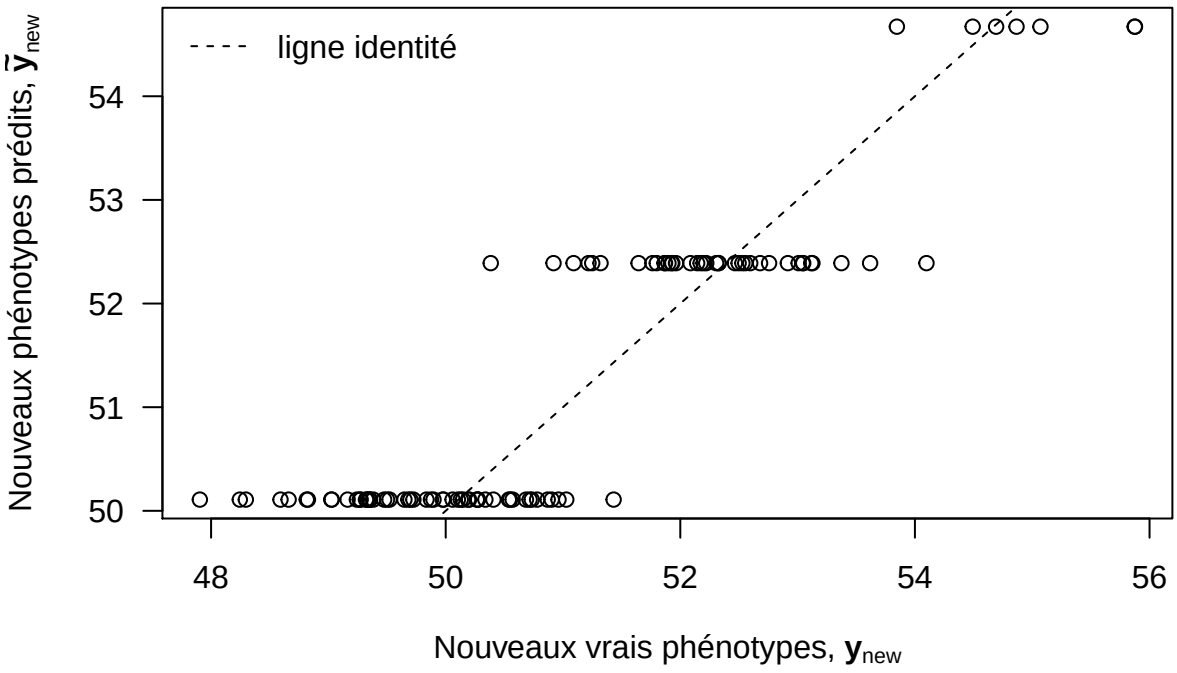
Enfin, calculons l'erreur quadratique de prédiction:

```
errors.tilde <- y.new - y.new.tilde
(rmspe <- sqrt(mean(errors.tilde^2)))
```

```
## [1] 0.8
```

Regardons visuellement ce que ça donne:

```
plot(x=y.new, y=y.new.tilde, las=1,
     xlab=expression(paste("Nouveaux vrais phénotypes, ", bold(y)[new])),
     ylab=expression(paste("Nouveaux phénotypes prédits, ", bold(tilde(y))[new])))
abline(a=0, b=1, lty=2)
legend("topleft", legend="ligne identité", lty=2, bty="n")
```



8 Explorer les simulations possibles

La simulation est un outil particulièrement utile pour explorer comment un modèle répond à des changements dans les données et les paramètres.

On pourrait par exemple avoir envie de savoir ce qui se passe si la taille de l'échantillon, n, varie. Idem, que se passe-t-il si, à n et σ fixés, on modifie β ?

C'est à vous !

9 Perspectives

9.1 Ré-écrire le modèle

La loi Normale dans l'équation de vraisemblance (11) est utilisée sous sa forme *univariée*, c'est-à-dire qu'on s'en sert pour "tirer" aléatoirement un seul nombre correspondant à une réalisation de la variable aléatoire d'intérêt.

Or la même loi existe aussi sous forme *multivariée*, ce qui permet, à chaque fois qu'on "tire" dedans, d'obtenir *plusieurs* nombres, ceux-ci étant arrangés dans un vecteur. Sous cette forme, la vraisemblance s'écrit maintenant:

$$\boldsymbol{y} \,|\, \boldsymbol{x}, \mu, \beta, \sigma \;\sim \mathcal{N}_n(\mathbf{1}\mu + \boldsymbol{x}\beta, \sigma^2 I_n) \tag{18}$$

où $\boldsymbol{y}$ est le vecteur de dimension n contenant les phénotypes, $\boldsymbol{x}$ est le vecteur de dimension n contenant les génotypes, $\mathbf{1}$ est le vecteur de dimension n ne contenant que des 1, et I_n est la matrice identité de dimension $n \times n$.

Si vous n'êtes pas très familier de la version multivariée, voici un exemple avec un vecteur aléatoire de longueur 2, $\boldsymbol{\theta}$, distribué selon une loi Normale bivariée $\mathcal{N}_2(\boldsymbol{\mu}, \Sigma)$. Ce vecteur aléatoire a deux éléments, θ_1 et θ_2. Le vecteur $\boldsymbol{\mu}$ a donc également deux éléments, le premier, μ_1, étant l'espérance de θ_1, et le deuxième, μ_2, étant l'espérance de θ_2. La matrice Σ contient les variances σ_1^2 de θ_1 et σ_2^2 de θ_2 sur la diagonale. Mais hors de la diagonale elle contient la covariance σ_{12}^2 entre θ_1 et θ_2. Dans ce cas simple, on retrouve donc la corrélation, ρ, entre θ_1 et θ_2 comme étant $\sigma_{12}^2/\sqrt{\sigma_1^2\sigma_2^2}$. On peut donc écrire:

$$\begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} \sim \mathcal{N}_2\left(\begin{bmatrix}\mu_1 \\ \mu_2\end{bmatrix}, \begin{bmatrix}\sigma_1^2 & \sigma_{12}^2 \\ \sigma_{12}^2 & \sigma_2^2\end{bmatrix}\right) \Leftrightarrow \boldsymbol{\theta} \sim \mathcal{N}_2(\boldsymbol{\mu}, \Sigma) \tag{19}$$

Passons au simulations, en commençant par fixer quelques valeurs:

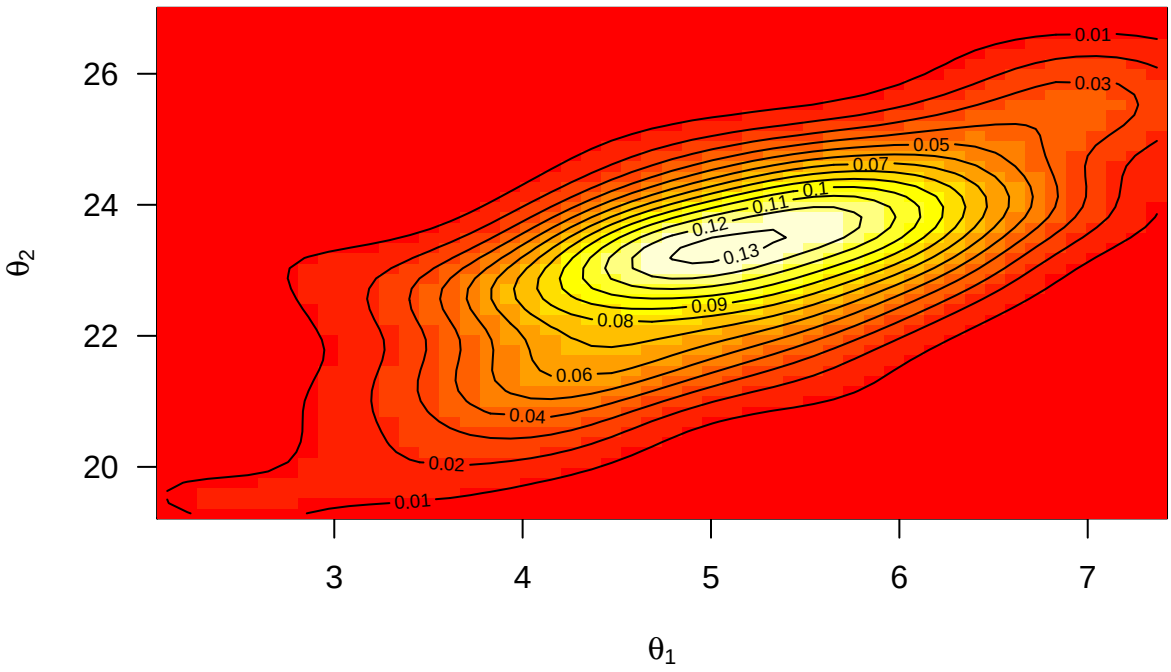
```
mu.1 <- 5
mu.2 <- 23
mu <- c(mu.1, mu.2)
var.1 <- 1
var.2 <- 2
rho <- 0.8
covar.12 <- rho * sqrt(var.1 * var.2)
Sigma <- matrix(c(var.1, covar.12, covar.12, var.2), nrow=2, ncol=2)
```

Voici une fonction permettant de simuler des vecteurs aléatoires, issue du paquet MASS:

```
theta <- mvrnorm(n=100, mu=mu, Sigma=Sigma)
```

Regardons à quoi les échantillons ressemblent:

```
bivn.kde <- kde2d(x=theta[,1], y=theta[,2], n=50)
image(bivn.kde, xlab=expression(theta[1]), ylab=expression(theta[2]), las=1,
      main=bquote(bold(paste("Echantillons d'une Normal bivariée avec ",
                             rho, " = ", .(rho)))))
contour(bivn.kde, add=T)
```



A vue d'oeil, on retrouve bien les valeurs utilisées pour les espérances, ainsi qu'une forte corrélation positive puisqu'on a utilisé $\rho = 0.8$ dans les simulations.

9.2 Améliorer le modèle

Naturellement, l'activité de modélisation statistique ne se limite pas à simuler des données sur ordinateur. Bien au contraire, elle est au coeur de l'activité de recherche en ce qu'elle vise à identifier les caractéristiques saillantes d'un phénomène naturel afin d'en réaliser l'inférence.

Concernant le thème de l'atelier, la prédiction génomique, quelles sont les limites du modèle exploré ci-dessus ? Que proposez-vous pour y remédier ?

10 Annexe

```
t1 <- proc.time(); t1 - t0
```

```
##    user  system elapsed
##    2.93    0.61    3.28
```

```
print(sessionInfo(), locale=FALSE)
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 16.04.3 LTS
##
## Matrix products: default
## BLAS: /usr/lib/openblas-base/libblas.so.3
## LAPACK: /usr/lib/libopenblasp-r0.2.18.so
```

```
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  base
##
## other attached packages:
## [1] MASS_7.3-48   knitr_1.17    rmarkdown_1.8
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.4.3  backports_1.1.1 magrittr_1.5    rprojroot_1.2
##  [5] tools_3.4.3     htmltools_0.3.6 yaml_2.1.14     Rcpp_0.12.14
##  [9] stringi_1.1.6   methods_3.4.3   stringr_1.2.0   digest_0.6.12
## [13] evaluate_0.10.1
```