

Atelier prédiction génomique : premiers pas

Vincent Segura (INRAE)

d'après ['premiers-pas.Rmd'](#) de Timothée Flutre (INRAE)

Lundi 17 février 2020

Introduction

R, rmarkdown, RStudio

- Cette présentation a été générée à partir d'un fichier texte au format Rmd utilisé par le logiciel libre [R](#)
- La fonction `render` du package [rmarkdown](#) permet de générer le fichier html à partir du fichier Rmd

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

- Il est généralement plus simple pour faire ça d'utiliser le logiciel [RStudio](#)
- Le format Rmd permet également d'utiliser le langage LaTeX pour écrire des équations

Packages

- Cette présentation nécessite par ailleurs le chargement des packages [MASS](#) et [stats4](#) qui sont généralement inclus par défaut dans R

```
library(MASS)  
library(stats4)
```

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**
- Lorsqu'on propose un modèle, on commence par expliquer les **notations**
- **Conventions :**
 - lettres grecques pour les paramètres (non-observés), par exemple θ
 - lettres romaines pour les données observées, y
 - lettres romaines surmontées d'un tilde pour les données prédites, \tilde{y}
 - les ensembles de données ou de paramètres sont généralement notés en majuscule, $\mathcal{D} = \{y_1, y_2, y_3\}$ ou $\Theta = \{\theta_1, \theta_2\}$
 - s'il y a plusieurs paramètres ou données, ils se retrouvent mathématiquement dans des vecteurs, en gras, $\boldsymbol{\theta}$ et \boldsymbol{y}
 - les vecteurs sont en colonne

La notion de vraisemblance

- Une fois les notations établies, on écrit la **vraisemblance** (*likelihood*), souvent présentée comme étant la “probabilité des données sachant les paramètres”
- Si les données sont des **variables continues**, c’est la densité de probabilité des données sachant les paramètres, notée $p(y|\theta)$, si les données sont des **variables discrètes**, c’est la fonction de masse, notée $P(y|\theta)$
- La vraisemblance est une fonction des **paramètres**, d’où le fait qu’on la note $\mathcal{L}(\theta)$ ou $\mathcal{L}(\theta|y)$
- la méthode du **maximum de vraisemblance** cherche à identifier la valeur du paramètre, notée $\hat{\theta}$ par convention, qui maximise la vraisemblance

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathcal{L} \Leftrightarrow \frac{\partial \mathcal{L}}{\partial \theta}(\hat{\theta}) = 0$$

L'incertitude

- Certains décrivent les statistiques comme étant la “science de l'incertitude” ([Lindley, 2000](#))
- 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**).

Comprendre la vraisemblance

- 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
- On prend 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 on dénote par y_1 , y_2 et y_3 les trois observations, vues comme des réalisations de cette variable aléatoire :

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

- Etant donné les caractéristiques du phénomène, il est raisonnable de supposer que la variable Y suit une **loi Normale** (c.f. le théorème central limite)

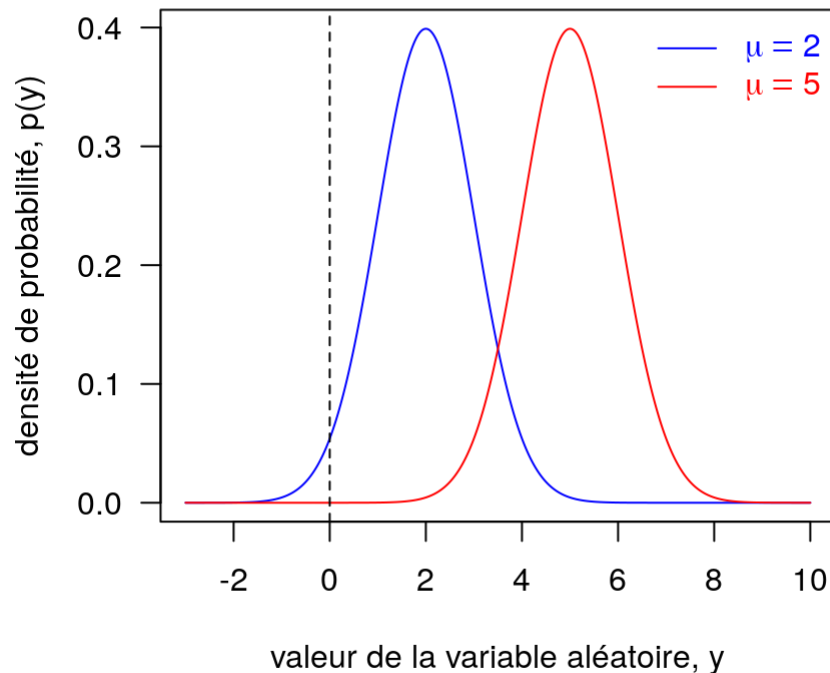
- Cette distribution de probabilité est caractérisée par deux paramètres, sa **moyenne** que l'on note généralement μ , et sa **variance** que l'on note généralement σ^2 (σ étant l'écart-type)
- En terme de notation, on écrit $Y \sim \mathcal{N}(\mu, \sigma^2)$, et la densité de probabilité de la réalisation y de Y s'écrit :

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

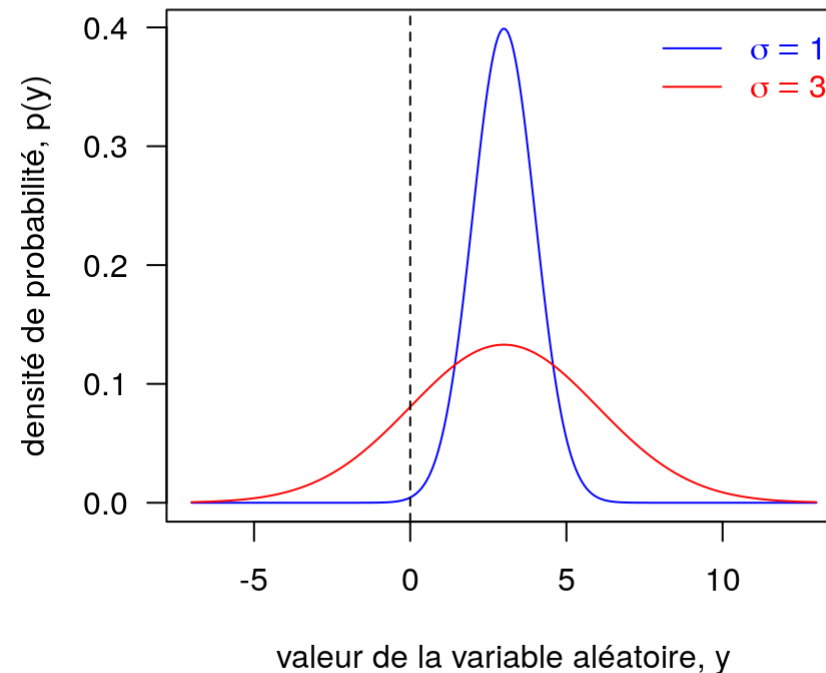
- L'intérêt de ce modèle paramétrique est de pouvoir "résumer" les données, par exemple un million de mesures, par seulement deux valeurs, les paramètres

- 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
- La loi Normale peut être assez différente selon les valeurs de ces paramètres

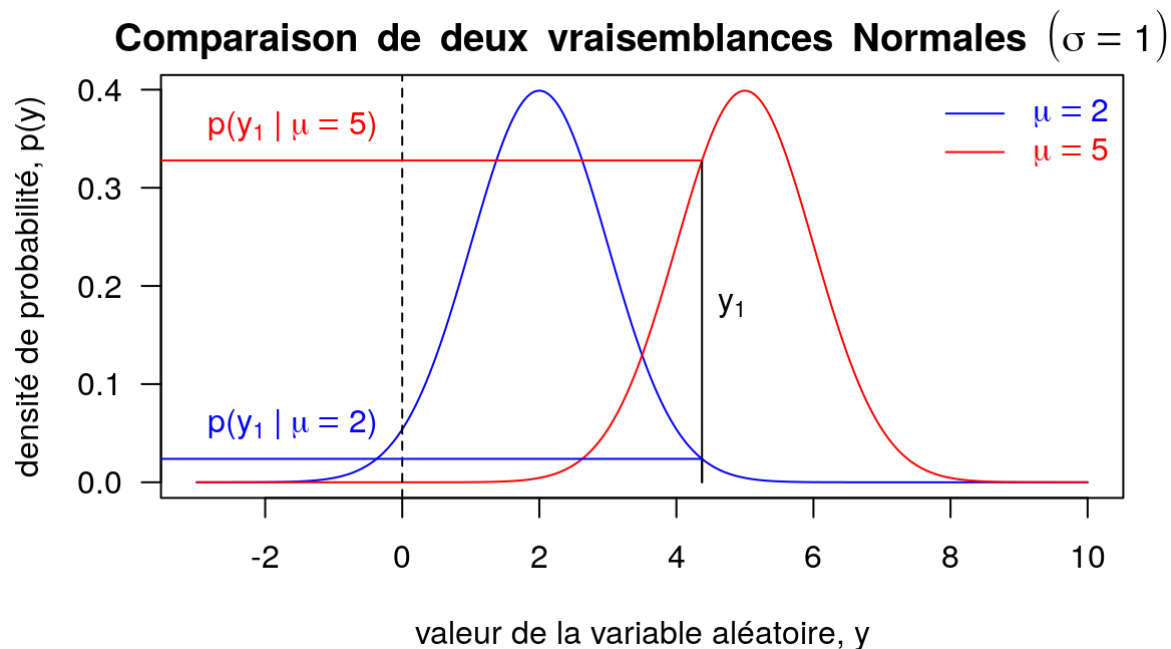
Comparaison de deux lois Normales ($\sigma = 1$)



Comparaison de deux lois Normales ($\mu = 3$)



- Revenons à nos trois mesures : 4.374, 5.184, 4.164
- Parmi toutes les valeurs possibles des paramètres, quelles sont celles pour lesquelles la loi Normale est une bonne description du mécanisme qui a généré ces données ?
- Pour simplifier, supposons que l'on connaisse déjà la variance : $\sigma^2 = 1$, il ne nous reste plus qu'à trouver la moyenne : μ .
- Pour la première observation, $y_1 = 4.374$:



- D'après le graphique précédent :
 $p(y_1 \mid \mu = 5, \sigma = 1) > p(y_1 \mid \mu = 2, \sigma = 1)$
- Cela se vérifie si l'on fait le calcul avec la formule :
 - $p(y_1 \mid \mu = 5, \sigma = 1) = 0.328$
 - $p(y_1 \mid \mu = 2, \sigma = 1) = 0.024$
- Comme les deux densités ont la même valeur pour σ , la différence vient bien du terme $(y - \mu)^2$ dans l'exponentielle, terme qui représente l'écart à la moyenne
- Au final, nous pouvons conclure pour la première observation, que 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 :

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

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

$$\begin{aligned}\mathcal{L}(\mu, \sigma) &= p(y_1 \mid \mu, \sigma) \times p(y_2 \mid \mu, \sigma) \times p(y_3 \mid \mu, \sigma) \\ &= \prod_{i=1}^3 p(y_i \mid \mu, \sigma)\end{aligned}$$

- 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) :

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

- 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
```


Ecrire le modèle

Notations

- n : nombre d'individus (diploïdes, supposés non-apparentés)
- i : indice indiquant le i -ème individu, $i \in \{1, \dots, n\}$
- 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
- x_i : génotype de l'individu i à ce SNP, codé comme le nombre de copie(s) de l'allèle minoritaire, $\forall i \ x_i \in \{0, 1, 2\}$
- 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

Notations (suite)

- Données : $\mathcal{D} = \{(y_1 | x_1), \dots, (y_n | x_n)\}$
- Paramètres : $\Theta = \{\mu, \beta, \sigma\}$

Vraisemblance

- On suppose que le génotype au SNP d'intérêt a un effet additif sur la moyenne du phénotype, ce qui s'écrit généralement :

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

- Une autre façon équivalente de l'écrire :

$$\forall i \quad y_i \mid x_i, \mu, \beta, \sigma \stackrel{\text{i.i.d}}{\sim} \mathcal{N}(\mu + \beta x_i, \sigma^2)$$

Simuler des données

- Initialisation :

On utilise un générateur de nombres pseudo-aléatoires qui peut être initialisé avec une graine (seed), ce qui est très utile pour la reproductibilité des analyses

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

- Nombre d'individus :

```
n <- 200
```

- Moyenne générale :

```
mu <- 50
```

- Génotypes (on suppose 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))
```

```
head(genotypes)
```

```
## [1] 2 0 1 0 1 1
```

```
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
```


- Effet du génotype sur le phénotype, β :

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

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

- Erreurs, ϵ (par simplicité, on fixe σ à 1) :

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

- Nous avons maintenant tout ce qu'il faut pour simuler les phénotypes, y , via l'équation précédente : $y_i = \mu + \beta x_i + \epsilon_i$

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

- Il est habituel dans R d'organiser les données dans un tableau

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

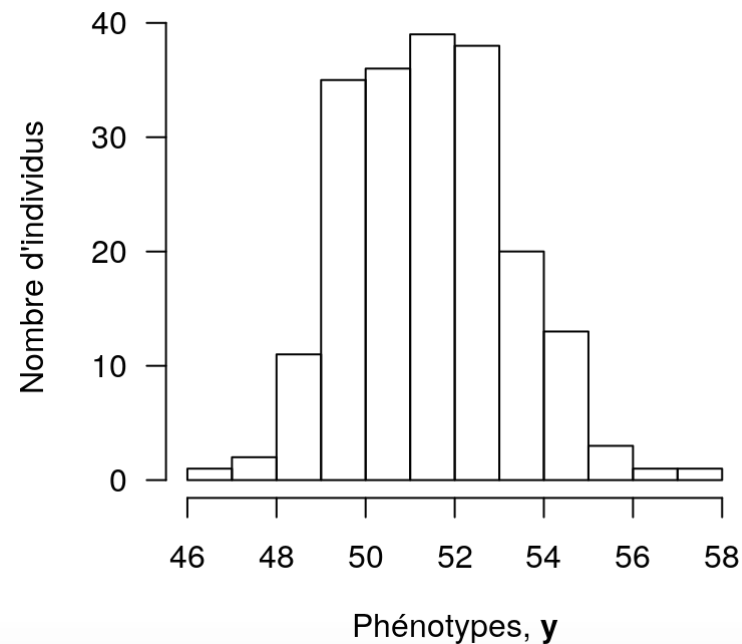
```
##    x    y  
## 1 2 52.7  
## 2 0 50.2  
## 3 1 53.6  
## 4 0 49.1  
## 5 1 51.7  
## 6 1 52.9
```

Réaliser l'inférence

Visualisation graphique

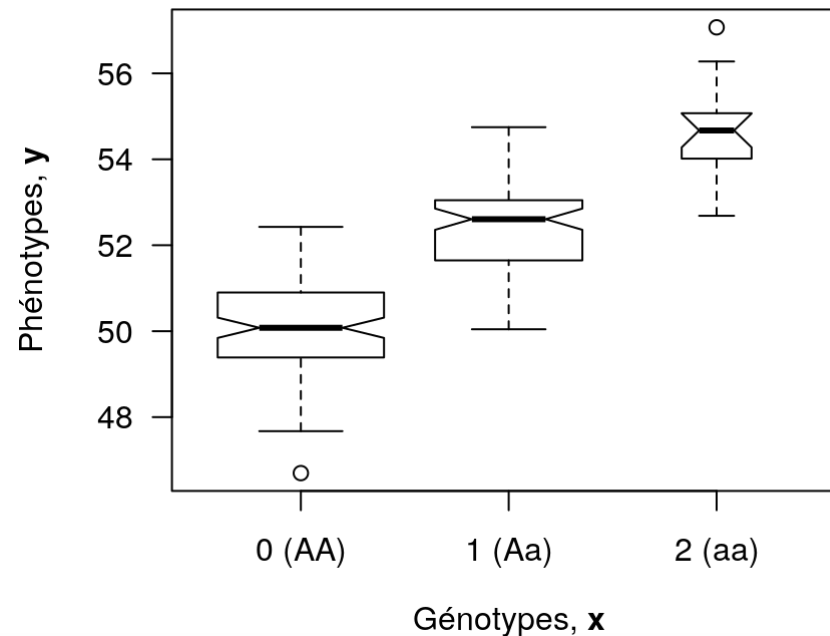
- Distribution du phénotype

```
par(mar = c(4, 4, 1, 1))  
hist(phenotypes, las = 1, main = "",  
     xlab = expression(paste("Phénotypes, ", bold(y))),  
     ylab = "Nombre d'individus")
```



- Relation génotypes - phénotypes

```
par(mar = c(4, 4, 1, 1))  
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)"))
```



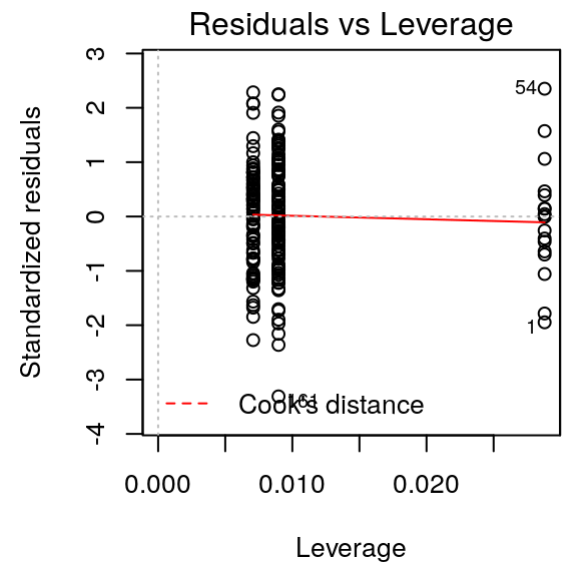
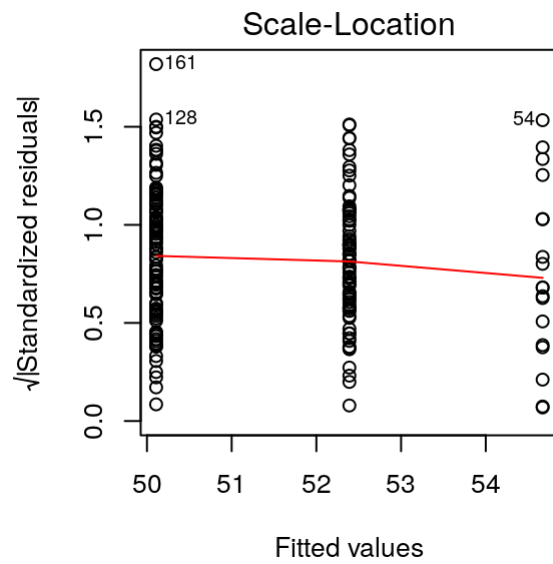
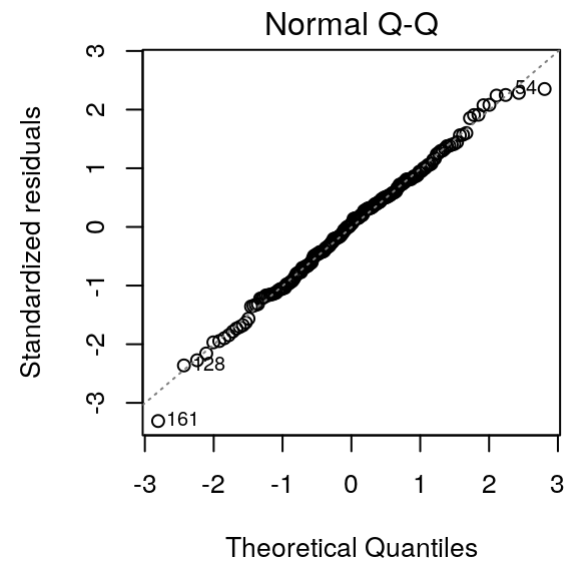
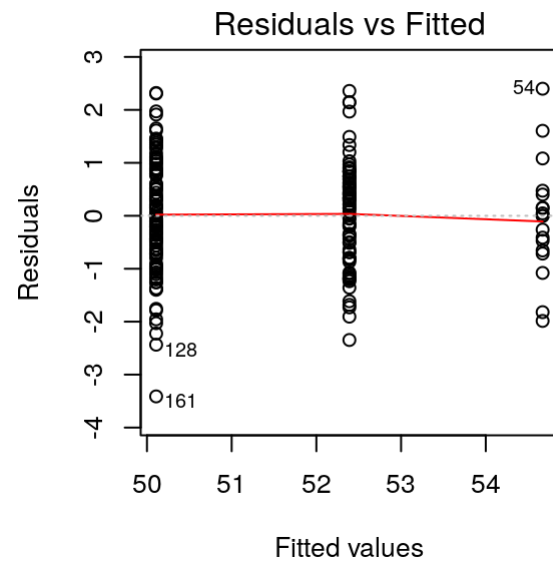
Implémentation (facile)

- Sous R, la fonction `lm` implémente l'estimation par maximum de vraisemblance

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

- Vérification des hypothèses du modèle (homoscédasticité, normalité, indépendance)

```
par(mfrow=c(2, 2), mar = c(4, 4, 2, 1))  
plot(fit)
```

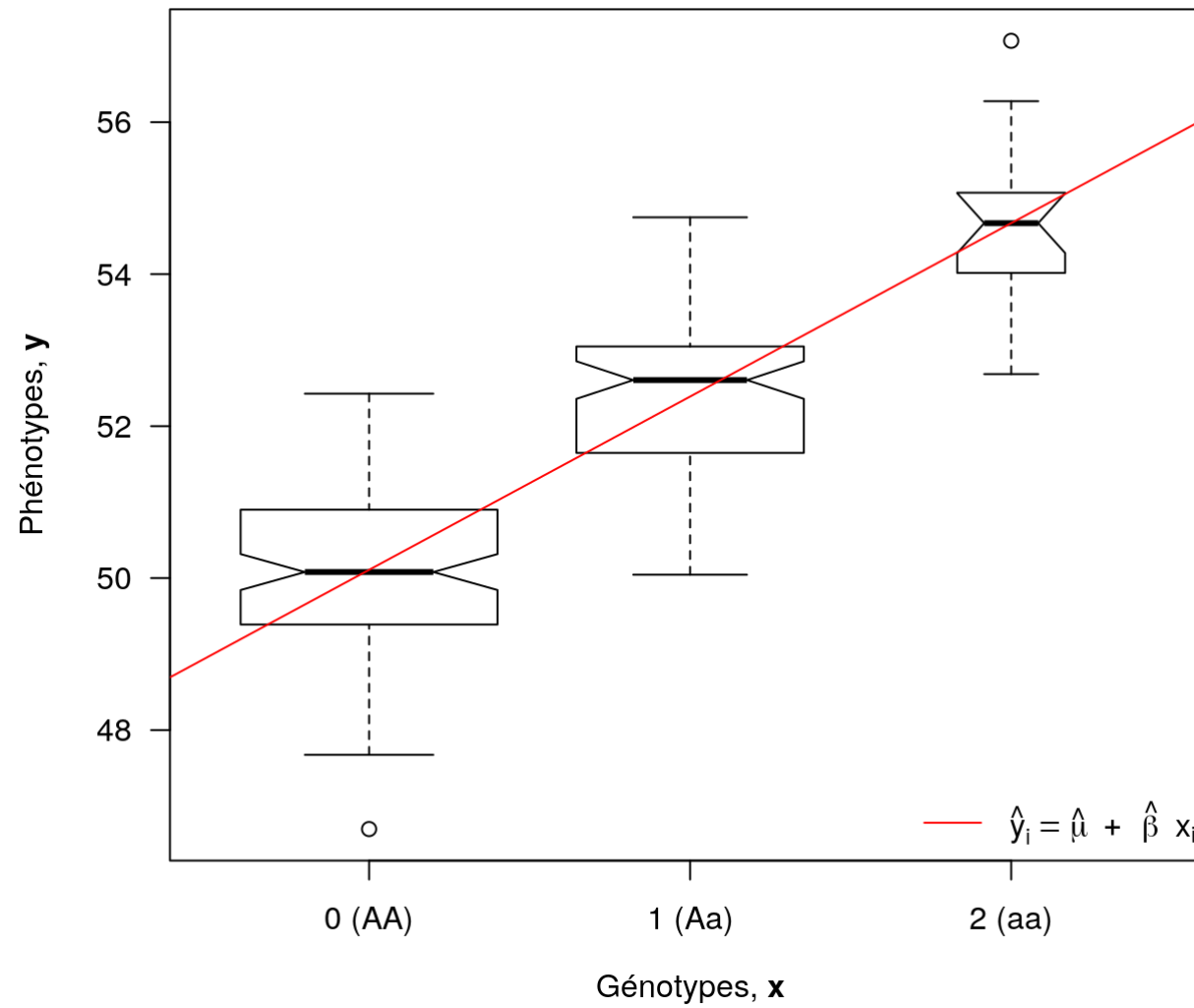


```
summary(fit)
```

```
##
## 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
```


- Représentation graphique du modèle

```
par(mar = c(4, 4, 1, 1))
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")
```



Implémentation (plus difficile)

- 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)

```
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
```

Evaluer les résultats

Sélection de modèles

- Evaluation de l'ajustement du modèle aux données
- Dans notre cas de régression linéaire simple, on peut utiliser le coefficient de détermination R^2

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

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

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

$$R^2 = \frac{\hat{\beta}^2 \text{Var}(\mathbf{x})}{\hat{\beta}^2 \text{Var}(\mathbf{x}) + \hat{\sigma}^2} ,$$

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

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

Estimation des paramètres

- Prenons l'exemple de β , comme nous avons simulé les données, nous connaissons sa vraie valeur

```
beta
```

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

- Après avoir ajusté le modèle avec la fonction `lm`, nous pouvons récupérer l'estimation de ce paramètre ($\hat{\beta}$)

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

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


- Pour comparer les deux, on définit une fonction de perte (*loss function*) reliant le paramètre (β) à son estimation ($\hat{\beta}$)
- On utilise une fonction quadratique, dont on prend l'espérance, ce qui donne l'erreur quadratique moyenne (*mean squared error*)

$$MSE = E \left((\hat{\beta} - \beta)^2 \right)$$

- On calcule sa racine carrée pour que le résultat soit dans la même unité que le paramètre

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

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

Prédiction de données

- On peut 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
```

- On peut aussi utiliser la fonction `predict`

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

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

- Le vecteur *errors* correspond aux résidus du modèle

```
head(errors)
```

```
##          1          2          3          4          5          6
## -1.9859  0.0643  1.2021 -0.9601 -0.6865  0.5462
```

```
head(resid(fit))
```

```
##          1          2          3          4          5          6
## -1.9859  0.0643  1.2021 -0.9601 -0.6865  0.5462
```

```
(rmse.y <- sqrt(mean(resid(fit)^2)))
```

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

- De façon plus intéressante, on souhaiterait évaluer les prédictions phénotypiques sur n_{new} nouveaux individus
- Pour cela, on commence par simuler de nouvelles données, $\mathcal{D}_{\text{new}} = \{(y_{i,\text{new}} \mid x_{i,\text{new}})\}$, 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 on utilise les estimations des paramètres obtenues précédemment pour prédire les nouveaux phénotypes à partir des nouveaux génotypes,

$$\tilde{\mathcal{D}}_{\text{new}} = \{(\tilde{y}_{i,\text{new}} = \hat{\mu} + \hat{\beta} x_{i,\text{new}})\} \text{ (out-of-sample predictions)}$$

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

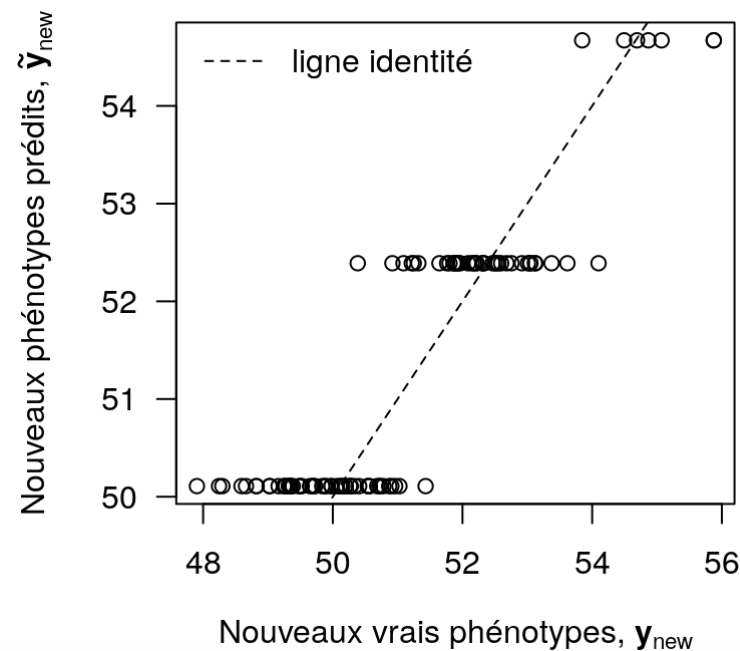
- Enfin, on calcule l'erreur quadratique moyenne de prédiction

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

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

Graphiquement :

```
par(mar = c(4, 4.5, 1, 1))  
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")
```



Perspectives

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 ?

Ré-écrire le modèle

- Modèle sous forme *multivariée*

$$\mathbf{y} \mid \mathbf{x}, \mu, \beta, \sigma \sim \mathcal{N}_n(\mathbf{1}\mu + \mathbf{x}\beta, \sigma^2 I_n)$$

- avec :
 - \mathbf{y} : vecteur de dimension n contenant les phénotypes,
 - \mathbf{x} : vecteur de dimension n contenant les génotypes,
 - $\mathbf{1}$: vecteur de dimension n ne contenant que des 1,
 - I_n : matrice identité de dimension $n \times n$

- Exemple d'un vecteur aléatoire de longueur 2, $\boldsymbol{\theta}$, distribué selon une loi Normale bivariée $\mathcal{N}_2(\boldsymbol{\mu}, \Sigma)$:

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

- On fixe 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)
```

- On simule des vecteurs aléatoires avec la fonction `mvrnorm` du package [MASS](#)

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

```
## [1] 100  2
```

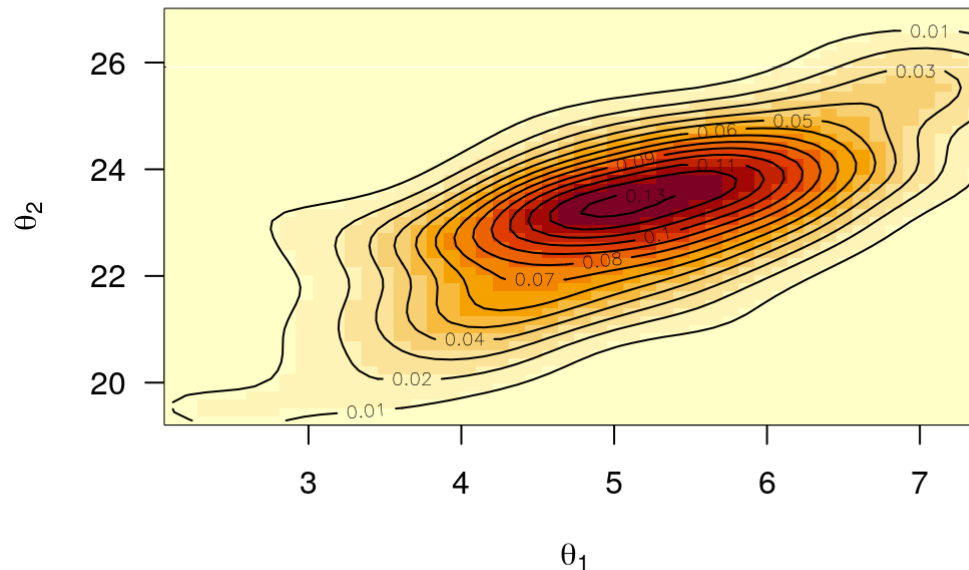
```
head(theta)
```

```
##      [,1] [,2]
## [1,] 4.00 22.7
## [2,] 6.25 23.0
## [3,] 5.35 23.8
## [4,] 5.67 22.4
## [5,] 6.57 24.1
## [6,] 4.48 23.3
```

- Représentation graphique

```
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)
```

Echantillons d'une Normal bivariée avec $\rho = 0.8$



Annexe

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

```
## R version 3.6.2 (2019-12-12)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 19.10
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.8.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.8.0
##
## attached base packages:
## [1] stats4      stats      graphics  grDevices  utils      datasets  methods
## [8] base
##
## other attached packages:
## [1] MASS_7.3-51.5
##
## loaded via a namespace (and not attached):
## [1] compiler_3.6.2  magrittr_1.5    tools_3.6.2     htmltools_0.4.0
## [5] yaml_2.2.1      Rcpp_1.0.3      stringi_1.4.5   rmarkdown_2.1
## [9] knitr_1.28      stringr_1.4.0   xfun_0.12       digest_0.6.24
## [13] rlang_0.4.4     evaluate_0.14
```