ASSISTANCE PUBLIQUE HÔPITAUX DE PARIS
Direction de la Recherche Clinique et de Innovation

Center for Methodology and Modeling
Malaria – Public Health – Infectious Diseases.
West-Africa
C2M

Biostatistiques

Partie 4 : Modèles multivariés et fonctions « avancées »

DU EPIDEMIOLOGIE DE TERRAIN - IFRISSE

Kankoé SALLAH MD, PhD

kankoe.sallah@univ-amu.fr
kankoe@skml.fr

Apr 2020

Les modèles multivariés

OBJECTIF MAJEUR:

Lire les principaux résultats d'un test multivarié dans un document scientifique et pouvoir se servir de R pour mettre en œuvre des fonctions et modèles avancés.

Les modèles multivariés

Résumé des test univariés

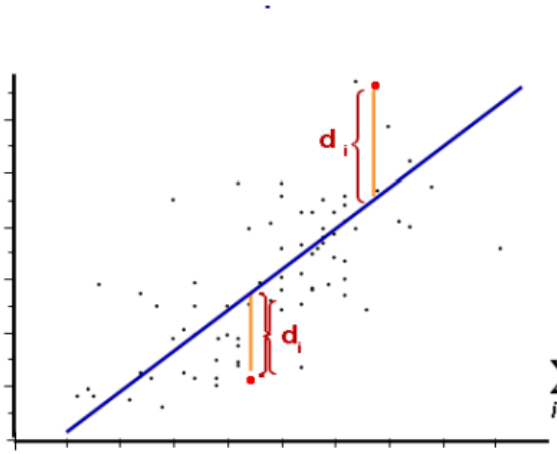
Types des données	Test paramétrique	Test non paramétrique	Exemple
Quanti-Quanti corrélation	Coeff. Cor. Pearson *cor.test(method="pearson")*	Coeff. Cor. Spearman *cor.test(method=« spearman")*	âge ~ TAS
Quanti-Quali Comparaison de moyennes	Student, ANOVA *t.test()* *aov()*	Wilcoxon, Kruskall-Wallis *wilcox.test()* *kruskal.test()*	TAS ~ Statut fumeur(oui/non) Notes ~ Spécialités Master
Quali-Quali Comparaison de pourcentages	Chi2 *chisq.test()*	Test exact de Fisher *fisher.test()*	Statut cancer ~ Statut fumeur Sexe ~couleur des yeux
Une distribution statistique		Test de Kolmogorov-Smirnov Test de Shapiro-Wilk (norm) *ks.test() | shapiro.test()*	

Les modèles multivariés

β1 et β0 sont estimés par la méthode des moindres carrés qui cherche à minimiser la somme ci-dessous

$$\sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 = \sum_{i=1}^{n} \left(Y_i - \left[\hat{\beta}_0 + \hat{\beta}_1 X_i\right]\right)^2$$

Somme des carrés des erreurs.
Sum of squares errors



Estimations obtenues

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^{n} (X_i - \bar{X})^2}$$

$$\hat{\beta}_0 = \bar{Y} - \hat{\beta}_1 \bar{X}$$

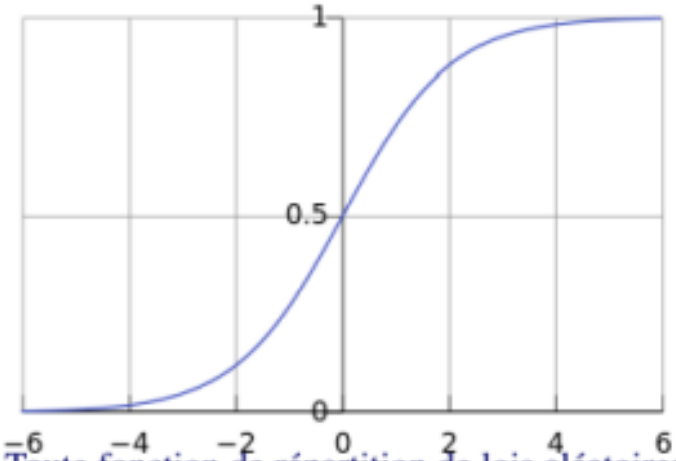
```
fit1 <-lm(data$hb0 ~ data$age,data=dataset)
summary(fit1)
```

Les modèles multivariés

Le modèle logistique : assomptions

- L'espérance π est modélisée en fonction de X: $\pi(x)$.
- En particulier, π dépend d'une combinaison linéaire (prédicteur linéaire) des variables indépendantes par le biais de la fonction logistique

$$\pi(x) = E(Y \mid x) = g^{-1}(\beta_0 + \beta_1 x) = \frac{e^{\beta_0+\beta_1 x}}{e^{\beta_0+\beta_1 x}+1} = \frac{1}{e^{-(\beta_0+\beta_1 x)}+1}$$



(NB: la fonction logistique est bornée entre 0 et 1).

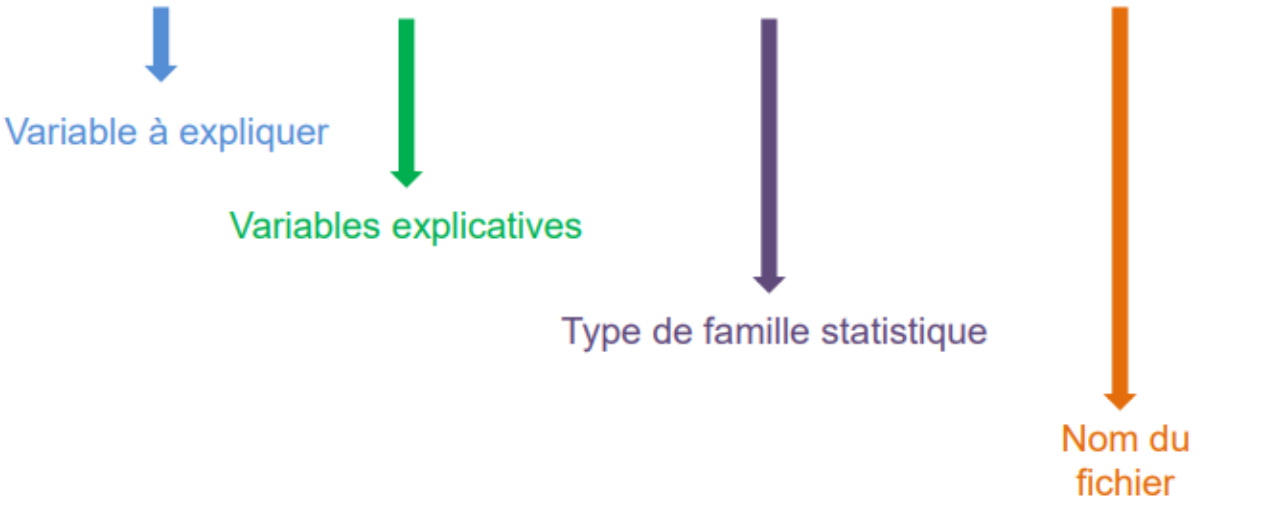
- Toute fonction de répartition de lois aléatoires continues avec la droite réelle comme support ont la même propriété. mais la fonction logistique a l'avantage de fournir de coefficients interprétables en tant que (log de) odds ratios

Les modèles multivariés

Régression logistique avec R

- Fonction « glm »

```
glm (cancer ~ tab + inh, family = binomial, data = tabac)
```



Les modèles multivariés

Modèle de Cox

$$\frac{\lambda(t, j_2)}{\lambda(t, j_1)} = \frac{\lambda_0(t)\, exp\left(\beta_1 X_1' + \cdots + \beta_{k-1} X_{k-1}' + \beta_k \times 1 + \beta_{k+1} X_{k+1}' + \cdots + \beta_n X_n'\right)}{\lambda_0(t)\, exp\left(\beta_1 X_1' + \cdots + \beta_{k-1} X_{k-1}' + \beta_k \times 0 + \beta_{k+1} X_{k+1}' + \cdots + \beta_n X_n'\right)} = exp(\beta_k)$$

Hypothèse des risques proportionnels

Le rapport est donc indépendant du temps, autrement dit, quel que soit le temps t, l'individu *j2* a un risque instantané de mourir ***exp(beta)*** fois celui de l'individu *j1.*

```
library(survival)
coxph()
```

Matrices et array

La commande *matrix* permet de créer une matrice

```
M = matrix(z,2,3)
```

ce qui peut également être fait en concaténant des vecteurs en ligne (*rbind*) ou en colonne (*cbind*) :

```
M = rbind(x,y)
M = cbind(x,y)
```

Les tableaux à plus de 2 dimensions (*array*) sont également utilisables :

```
T = array(0,dim=c(2,3,4))
```

Listes

Une liste est une combinaison de structures de données de natures potentiellement différentes :

```
L=list(elt1=c(1,2,3),elt2=matrix(rnorm(9),3,3),
       elt3='tutu',elt4=seq(1,4,by=0.5))
```

Les éléments de la liste sont alors accessible par un '$', et les noms des éléments par la commande *names* :

```
L$elt4
```

```
## [1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0
```

```
names(L)
```

```
## [1] "elt1" "elt2" "elt3" "elt4"
```

Introduction à R via Rstudio

Les fonctions

R dispose d'un grand nombre de fonctions prédéfinies (outres les fonction d'analyses statistiques...) :

```
mean(x)
```

```
## [1] 8
```

```
rnorm(5)
```

```
## [1]  0.2928666 -0.2844265  0.3180094  0.9239928 -1.3615745
```

```
rnorm(5,mean=1,sd=2)
```

```
## [1] -1.817238 -1.205668  3.839729  4.700798  2.181697
```

Astuce: lorsque vous commencez à taper le nom de la fonction, la touche *tabulation* permet de voir les différentes complétion possibles. Lorsque le nom de la fonction est totalement saisi, la tabulation permet de voir les arguments attendus par la fonction.

Introduction à R via Rstudio

Les scripts

Vous n'êtes pas obligé de taper toutes les commandes R dans la fenêtre de commande. Il est possible de créer des scripts R (dans la fenêtre d'édition), en les enregistrant avec une extension '.r' et de les exécuter à l'aide de la commande source :

```
source('myscript.r')
```

Les icônes *source* et *run* permettent d'exécuter tout ou partie du script R affiché dans la fenêtre d'édition.

Introduction à R via Rstudio

Importer des données depuis Excel

La fonction **read.xls** (ou **read.xls**) permet d'importer des données directement depuis Excel.

```
library(gdata)
DataVoitures = read.xls("DataVoitures2010.xlsx")
```

```
str(DataVoitures)
```

```
## 'data.frame':    29 obs. of  13 variables:
##  $ Type         : Factor w/ 5 levels "4 x 4","Citadine",..: 2 3 3 5 3 2 2 3 5 3 ...
##  $ Marque       : Factor w/ 16 levels "Audi","Bmw","Citroen",..: 13 11 11 11 13 11 13 3 13 13 ...
##  $ Modele       : Factor w/ 29 levels "207","5008","528",..: 12 22 2 4 19 1 26 7 14 18 ...
##  $ Tarif        : int  10940 17250 24400 35500 24250 14600 12050 26350 33750 17650 ...
##  $ Cylindree    : int  1461 1560 1560 1997 1461 1398 1461 1560 1995 1461 ...
##  $ Puissance    : int  65 90 110 136 110 70 65 110 130 85 ...
##  $ Consommation: num  5.4 5.7 6.5 7.1 5 4.4 4.3 5.6 7.4 5.3 ...
##  $ CO2          : int  115 150 140 179 133 117 113 149 190 140 ...
##  $ Vitesse      : int  165 150 183 190 187 166 164 191 184 158 ...
##  $ Coffre       : int  255 675 679 830 508 270 230 439 650 660 ...
##  $ Poids        : int  1015 1407 1472 1600 1386 1194 980 1503 1757 1389 ...
##  $ Longueur     : num  3.82 4.38 4.53 4.73 4.8 4.05 3.6 4.78 4.66 4.21 ...
##  $ Hauteur      : num  1.42 1.87 1.64 1.75 1.45 1.47 1.47 1.46 1.73 1.8 ...
```

Attention : la paramétrisation de cette fonction est assez spécifique à chaque machine (système d'exploitation, version de Java, . . .). Quelques tests seront nécessaires. . .

Introduction à R via Rstudio

Eléments de programmation

La syntaxe pour la condition **if** est la suivante (la condition *else* peut être omise) :

```
w=2
if (w>3) {res=2} else {res=4}
print(res)
```

```
## [1] 4
```

Une boucle **for** a la syntaxe suivante

```
vec=c()
for (i in 1:10){
  vec=c(vec,i)
  cat('iteration numero: ',i,'\n')
  }
```

Ecrire ses propres fonctions

Un des grands intérêts du logiciel R est qu'il est possible de créer ses propres fonctions.

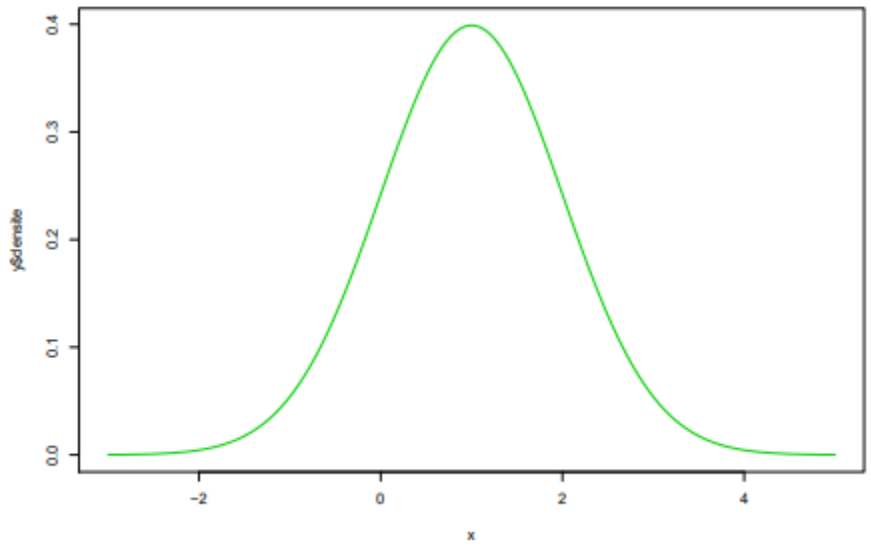
```
mafonction <- function(arg1,arg2=0,arg3=1){
tmp=exp(-1/2*((arg1-arg2)^2/(sqrt(arg3))))/sqrt(2*pi*arg3)
return(res=list(argument1=arg1,densite=tmp))
}
```

- les arguments de la fonction sont donnés après l'instance *function*
- une valeur par défaut à un argument peut être donnée en indiquant cette valeur lorsque les arguments sont définis
- le résultat de la fonction peut être de différente nature (scalaire, vecteur, matrice, liste...)

Introduction à R via Rstudio

Ecrire ses propres fonctions

```
x=seq(-3,5,0.01)
y=mafonction(x,arg2=1)
plot(x,y$densite,type='l',col=3)
```



Introduction à R via Rstudio

Principales fonctions : création de données

- **read.table** : lit un data frame à partir d'un fichier. Arguments: *header=TRUE* si la première ligne correspond aux intitulés des variables; *sep=''* pour indiquer le séparateur de variables dans le fichier; *skip=n* pour ne pas lire les n premières lignes.
- **write.table** : sauvegarde un data frame dans un fichier.
- **c** : concatène des scalaires en un vecteur.
- **rbind, cbind** : concatène en ligne ou en colonne des vecteurs en une matrice.
- **list** : crée une liste.
- **matrix** : crée une matrice à *nrow* lignes et *ncol* colonnes.
- **data.frame** : crée un data frame.
- **array** : crée un tableau dont l'argument *dim* permet de préciser le nombre de dimensions ainsi que la taille de chaque dimension.
- **seq** : créer une séquence d'entiers.
- **rnorm, runif** : simule la génération d'une variable aléatoire normale, uniforme.

Introduction à R via Rstudio

Principales fonctions : manipulation de données

- **x[n]** : n-ème élément du vecteur *x*.
- **x[n:m]** : n-ème au m-ème éléments du vecteur *x*.
- **x[c(k,l,m)]** : k-ème, l-ème et m-ème éléments du vecteur *x*.
- **x[x>m & x<n]** : éléments de *x* compris entre m et n.
- **l$x** ou **l[["x"]]**: élément *x* de la liste *l*.
- **M[i,j]** : élément ligne i et colonne j de la matrice *M*.
- **M[i,]** : i-ème ligne de la matrice *M*.
- **t(M)** : transposée de la matrice *M*.
- **solve(M)** : inverse de la matrice *M*.
- **M%*%N** : produit des matrices *M* et *N*.
- **sort(x)** : tri du vecteur *x*.

* http://hansheng.gsm.pku.edu.cn/pdf/2007/prop.pdf

Introduction à R via Rstudio

Principales fonctions : information sur les variables

- **length** : longueur d'un vecteur.
- **ncol**, **nrow** : nombre de colonnes et de lignes d'une matrice.
- **str** : affiche le type d'un objet.
- **as.numeric**, **as.character** : change un objet en un nombre ou une chaine de caractères.
- **is.na** : teste si la variable est de type 'NA' (valeur manquante).

Introduction à R via Rstudio

Principales fonctions : statistiques

- **sum** : somme d'un vecteur.
- **mean** : moyenne d'un vecteur.
- **sd**, **var** : écart-type et variance d'un vecteur (dénominateur $n-1$)
- **rowSums**, **rowMeans**, **colSums** ou **colSums** : somme et moyenne en ligne ou en colonne d'une matrice.
- **max**, **min** : maximum et minimum d'un vecteur.
- **quantile(x,0.1)** : quantile d'ordre 10% du vecteur x.

Introduction à R via Rstudio

Principales fonctions : graphiques

- **plot(x)** : représente une série de points (ordonnée *x* et numéro d'indice en abscisse).
- **plot(x,y)** : représente un nuage de points d'abscisse *x* et d'ordonnée *y*.
- **image(x,y,z)** : représente en niveau de couleur une image où *z* représente l'intensité au point *(x,y)* (*z* est une matrice dont le nombre de ligne est la longueur de *x* et le nombre de colonne celle de *y*).
- **lines, points** : ajoute une ligne ou des points sur un graphique existant.
- **hist** : histogramme.
- **barplot** : graphique en barre.
- **abline** : représente une ligne en précisant la pente *b* et l'ordonnée à l'origine *a*. Une ligne verticale d'abscisse *x* (*v=x*) ou horizontale d'ordonnée *y* (*v=y*)
- **legend** : ajoute une légende en précisant les symboles (*lty* ou *pch*et *col*), le texte (*text*) et l'emplacement (*x='topright'*).

Introduction à R via Rstudio

Principales fonctions : graphiques

- **axis** : ajoute un axe. Argument : *side* (1: bas, 2: gauche, 3: haut, 4: droite).
- **grid** : ajoute un quadrillage.
- **par(mfrow=c(n,p))** : partage la fenêtre graphique en n x p sous graphiques.

De nombreuses fonctions graphiques disposent des paramètres suivants :

- **type** : *'l'* pour ligne et *'p'* pour points.
- **col** : *'black'*, *'red'*, *'green'*, *'blue'* . . . (ou 1, 2, 3, 4. . .)
- **lty** : type de lignes (1: solide, 2: pointillée. . .).
- **pch** : type de points (1: cercle, 2: triangle. . .).
- **main** : titre principale.
- **xlab, ylab** : titre des axes.
- **log** : échelle logarithmique (*'x'* pour l'axe des abscisse, *'y'* pour l'axe des ordonnées, *'xy'* pour les deux axes).

En résumé

Impératifs :

- Savoir passer du tableur (.xls, .xlsx, .ods) au .csv
- Savoir lire son fichier .csv depuis la console R

Exemple :
data <-read.csv2(« G:/Tutorat-2013-14/fichier.csv", header=T)

- Comprendre la notion de fonction. Ex : read.csv2()
- Choisir les bons arguments pour une fonction :

Ex : boxplot(age~sexe, col=c(2,4))

En résumé

- read.csv() ≠ read.csv2()
- MAJUSCULES et minuscules comptent (sensible à la casse)
- Le fichier n’est pas dans le bon répertoire ?
- Attachez si vous voulez mais détachez ensuite detach()
- Ecrivez votre code dans un fichier .R
  - Gain de temps et réutilisation ++
- Utilisez Rstudio, Rkward, Revolution, Jupiter
- Réutilisez vos anciens codes en les adaptant …

Merci