Atelier sur la biostatistique
Aide-mémoire pour la programmation en R

Janie Coulombe

17 juin 2022

Première étape: Installer R Studio

L'utilisation de R Studio, qui est un interface pour la programmation avec R, se fait en deux étapes. D'abord, allez au https://cran.r-project.org/bin/windows/base/ (https://cran.r-project.org/bin/windows/base/) pour installer R (version 3.0.1 ou plus, par exemple vous pouvez présentement télécharger la version 4.2.0).

Ensuite, aller au https://www.rstudio.com/products/rstudio/download/ (https://www.rstudio.com/products/rstudio/download/) pour installer R Studio.

Commandes de base

Une fois que vous avez installé R et R Studio, cliquez sur l'icône R Studio pour ouvrir le logiciel.

Dans R Studio, vous pouvez écrire directement dans la fenêtre *console* vos commandes et peser sur *enter* (entrée) pour exécuter le code que vous venez d'écrire.

Une autre option qui vous est offerte est d'ouvrir un script en R pour y écrire vos lignes de code. Vous pouvez ensuite enregistrer ce script pour une utilisation future. Vous pouvez copier-coller le contenu du script dans la console pour le faire exécuter, ou encore vous placer au-dessus d'une ligne de code du script et peser de façon simultanée sur ctrl+entrée.

Pour ouvrir un nouveau script, cliquez de façon consécutive sur *file* dans le menu en haut, *new file*, puis *R script*. Pour ensuite enregistrer le script, cliquez sur la petite disquette bleue en haut du script.

Pour commenter à l'intérieur d'un script R, vous pouvez utiliser le double dièse (##) en début du texte à mettre en commentaire.

Exemples de quelques commandes de base:

```
2+2
```

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

```
vecteur<- c(1,2,3,4)
vecteur+2
```

```
## [1] 3 4 5 6
```

```
a<- 4
b<- 10
a*b
```

```
## [1] 40
```

```
f<- function(a){a*4}
f(8)
```

```
## [1] 32
```

Lire un fichier de données enregistré sous le format texte (.txt)

Si vous souhaitez faire une analyse de données à partir d'un jeu de données existant, vous pouvez importer le jeu de données dans R Studio. Par exemple, soit le jeu de données *inference.txt* qui se trouve dans vos documents sur votre ordinateur. Le jeu peut être importé sur R Studio grâce aux commandes suivantes (en remplaçant par l'endroit sur votre ordinateur où se trouve le jeu de données inference.txt):

```
data<- read.table('C:\\Users\\Janie\\Documents\\inference.txt', sep=c('\t',' '))
```

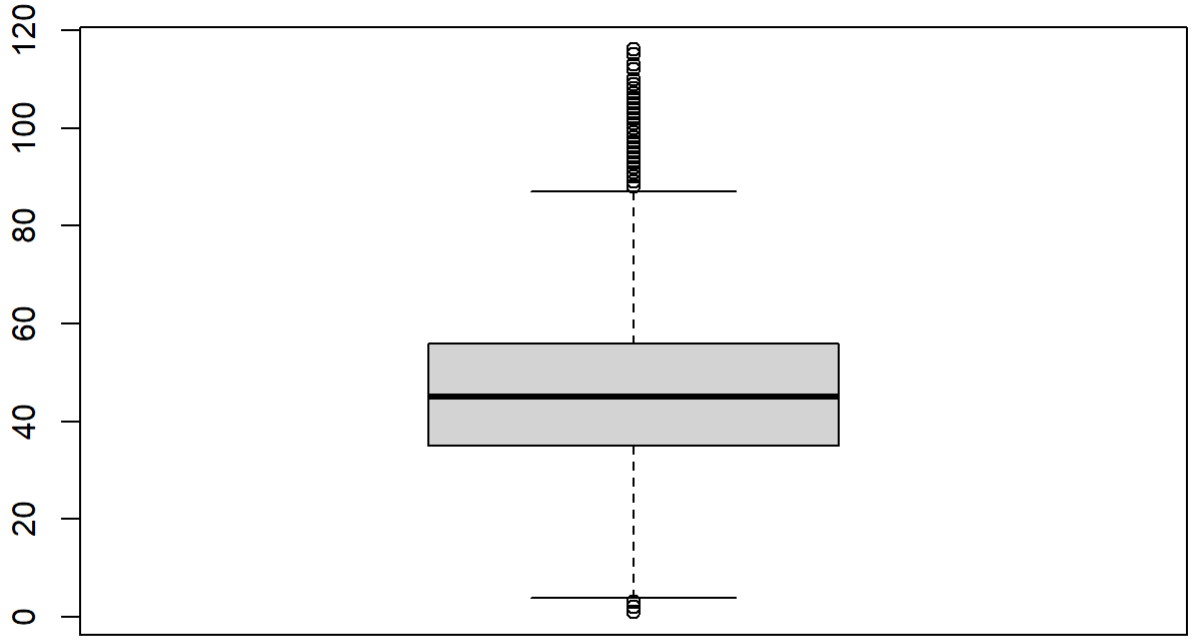
La commande `sep=c('\t',' ')` ci-haut permet de s'assurer que les entrées dans le jeu de données qui sont séparées par des espaces ou des espaces tab sont considéres comme des entrées différentes. Vous pourriez aussi y mettre `sep=c(',')` ou encore `sep=c('.')` si les entrées dans votre jeu de données sont séparées par des virgules ou encore des points.

Notez que ces commandes sont montrées à titre informatif mais nous n'utiliserons pas de jeu de données dans cet atelier. Nous simulerons plutôt des données à partir de fonctions en R qui nous permettent de simuler des variables aléatoires.

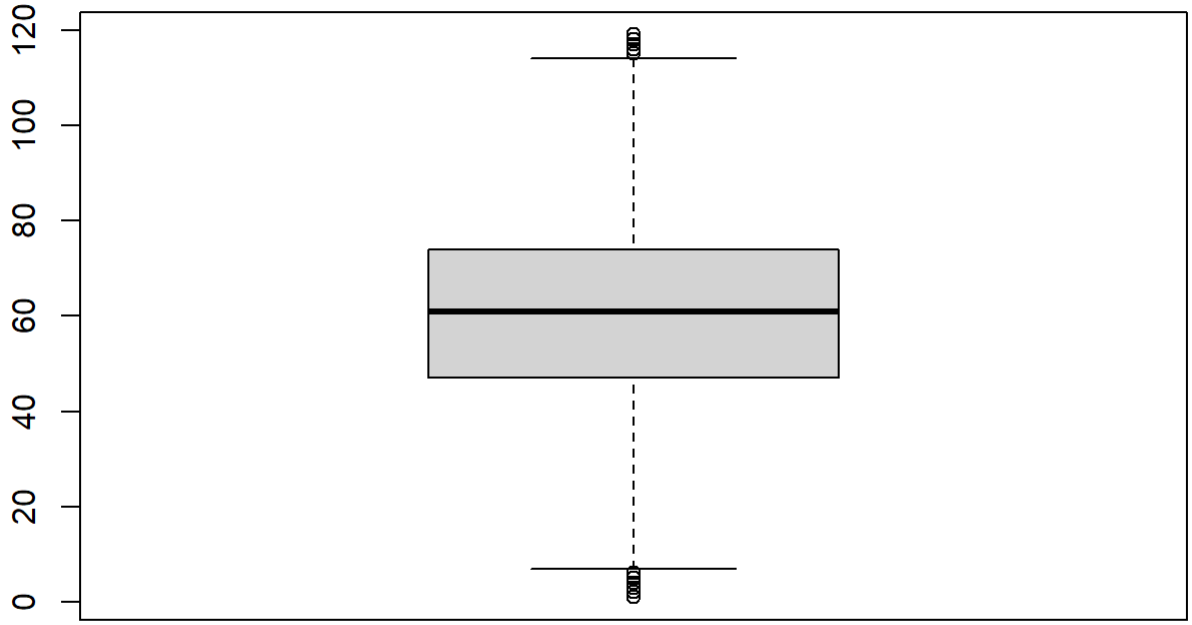
Exercice 1: Paradoxe de Simpson

Tout d'abord, j'ai simulé des données sur l'âge et la mortalité due à la COVID19 dans deux pays distincts. La distribution de l'âge était différente entre les deux pays, et l'âge affectait aussi la mortalité due à la COVID19. Commençons d'abord avec la simulation de l'âge:

```
set.seed(2455)            ## J'ai mis un germe pour que l'on
                         ##obtienne tous les mêmes résultats
age_pays1<- ceiling( rnorm( n=1500000, mean=45, sd= 15)) ## Simulation âge
age_pays1[age_pays1<1]<-1 ## enlever âges <1
boxplot(age_pays1)        ## Boîte à moustache âge pays 1
```

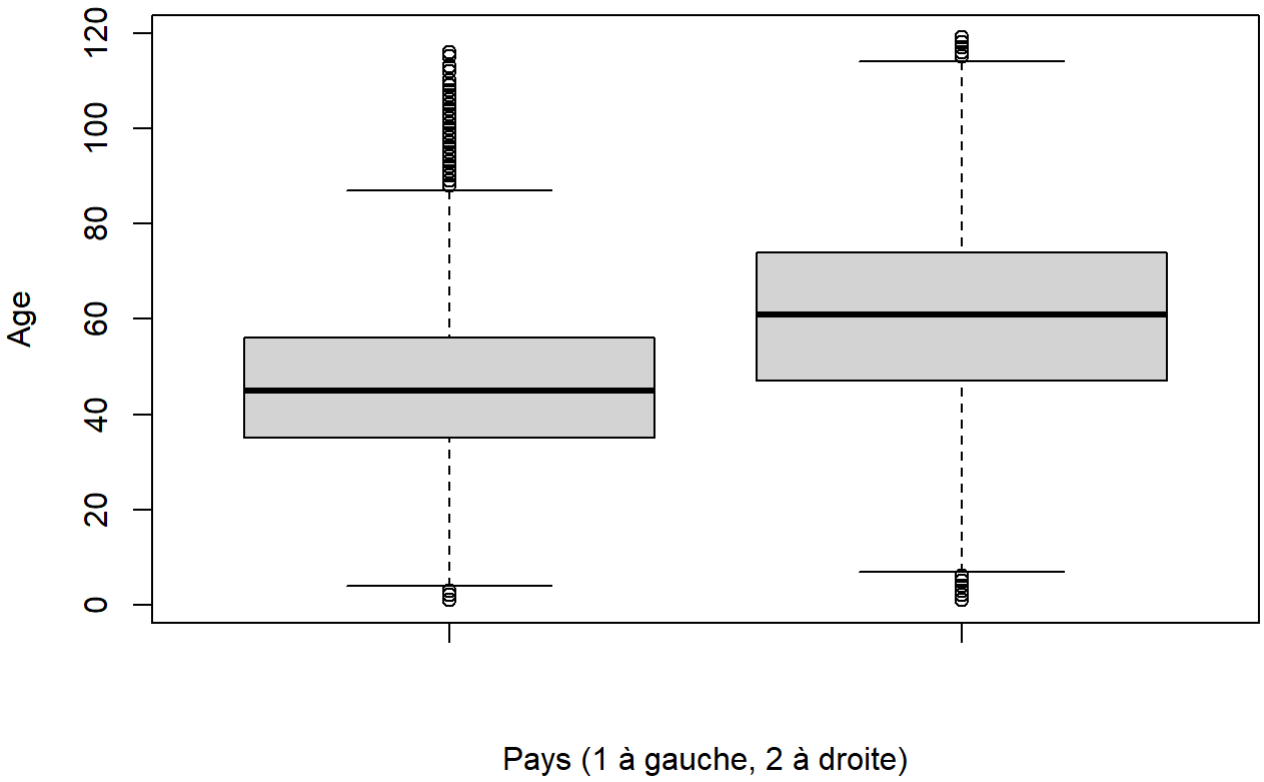



```
age_pays2<- ceiling( rnorm( n=1500000, mean=60, sd= 20)) ## Âge pays 2
age_pays2[age_pays2<1]<-1 ## enlever âges <1
age_pays2[age_pays2>119]<-119 ## enlever âges >119
boxplot(age_pays2)
```



Voici une comparaison de l'âge entre les deux pays:

```
boxplot(age_pays1, age_pays2, xlab='Pays (1 à gauche, 2 à droite)', ylab='Age')
```



Simulons maintenant la mortalité à partir d'une variable de loi Bernoulli:

```
pays<- c(rep(1,1500000), rep(2,1500000))
age<- c(age_pays1, age_pays2)
gg<- exp( -6.5 + 0.04*age)/(1+exp(-6.5 + 0.04*age))
mortalite<- rbinom( n=3000000, size=1, prob= gg)
table(mortalite, pays)
```

```
##          pays
## mortalite       1       2
##         0 1483658 1466397
##         1   16342   33603
```

On retrouve donc des taux de mortalité respectifs de:

```
16342/(16342+1483658)*100
```

```
## [1] 1.089467
```

```
33603/(33603+1466397)*100
```

```
## [1] 2.2402
```

Donc, 1.1% versus 2.2%. Le pays 2 a un taux deux fois plus élevé.

Voyons maintenant ce qui se produit si l'on compare les taux par tranche d'âge (en faisant de fines tranches d'âges). Tout d'abord, je crée une variable *age_group* qui stratifie l'âge en fines catégories de 5 ans. Cette nouvelle variable est catégorielle et possède 21 catégories:

```
age_group<- ifelse(age<5, 1,
             ifelse(age<10, 2,
             ifelse(age<15, 3,
             ifelse(age<20, 4,
             ifelse(age<25, 5,
             ifelse(age<30, 6,
             ifelse(age<35, 7,
               ifelse(age<40, 8,
             ifelse(age<45, 9,
             ifelse(age<50, 10,
             ifelse(age<55, 11,
             ifelse(age<60, 12,
             ifelse(age<65, 13,
             ifelse(age<70, 14,
             ifelse(age<75, 15,
             ifelse(age<80, 16,
             ifelse(age<85, 17,
             ifelse(age<90, 18,
             ifelse(age<95, 19,
             ifelse(age<100, 20, 21 ))))))))))))))))))))

table(age_group, pays) ## distribution des groupes d'âge entre les pays
```

```
##          pays
## age_group      1      2
##        1    4725   3805
##        2    7476   4201
##        3   16847   8004
##        4   33318  14106
##        5   58652  23607
##        6   93662  36913
##        7  133408  54550
##        8  169832  75301
##        9  193311  97202
##        10 197396 118092
##        11 180329 136804
##        12 148330 147006
##        13 109135 149639
##        14  71654 140785
##        15  42516 127185
##        16  21953 106474
##        17  10482  84369
##        18   4443  62179
##        19   1705  43288
##        20    580  28316
##        21    246  38174
```

Calculons maintenant les taux de mortalité des deux pays à travers chaque tranche d'âge possible. Ici, j'utilise une boucle pour passer à travers chacune des tranches d'âge possibles (21 tranches):

```
dat<- data.frame( mortalite, pays, age_group) ## Créer jeu de données combiné


for(i in 1:5){  ## boucle qui passe sur chaque catégorie d'âge
   ## vous pouvez changer 5 pour 21***
 dat1<- dat[dat$age_group ==i,]

 print(c('Tranche Age',i))

 print('pays 1 taux')
 print( table( dat1$mortalite, dat1$pays)[2]/ ( table( dat1$mortalite,   dat1$pays)[2] +
 table( dat1$mortalite, dat1$pays)[1])*100    )

 print('pays 2 taux')
 print( table( dat1$mortalite, dat1$pays)[4]/ ( table( dat1$mortalite, dat1$pays)[4] + ta
ble( dat1$mortalite, dat1$pays)[3])*100    )
 }
```

```
## [1] "Tranche Age" "1"
## [1] "pays 1 taux"
## [1] 0.1269841
## [1] "pays 2 taux"
## [1] 0.1839685
## [1] "Tranche Age" "2"
## [1] "pays 1 taux"
## [1] 0.1738898
## [1] "pays 2 taux"
## [1] 0.1428231
## [1] "Tranche Age" "3"
## [1] "pays 1 taux"
## [1] 0.2314952
## [1] "pays 2 taux"
## [1] 0.2748626
## [1] "Tranche Age" "4"
## [1] "pays 1 taux"
## [1] 0.2731256
## [1] "pays 2 taux"
## [1] 0.2764781
## [1] "Tranche Age" "5"
## [1] "pays 1 taux"
## [1] 0.4023733
## [1] "pays 2 taux"
## [1] 0.3600627
```

Etc. Ici vous pouvez changer 5 pour 21, pour voir les 21 comparaisons comme dans l'atelier. J'ai arrêté à 5 pour utiliser moins d'espace dans le document d'instructions pour l'atelier.

Exercice 2: Paradoxe de l'amitié

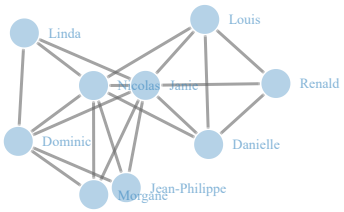
Première étape: créer un jeu de données maison à partir de paires de noms et de poids assignés à chaque relation. Par exemple, je mets un poids très fort pour la paire Janie-Nicolas puisque Nicolas est mon conjoint.

```
rm(list=ls(all=TRUE))
library(networkD3)



dataset <- cbind(  ## la fonction cbind colle les colonnes ensemble une à côté de l'autre

  ## Personne 1 de chaque paire (rep crée une répétition)
  c( rep( 'Janie', 8) , rep( 'Nicolas', 6), rep( 'Dominic' , 3 ), rep( 'Danielle' , 2 ),
 rep('Louis', 1)) ,

  ## Personne 2 de chaque paire
  c( 'Nicolas', 'Danielle','Louis', 'Renald' , 'Linda', 'Jean-Philippe', 'Dominic', 'Morg
ane', 'Danielle','Louis',  'Linda', 'Jean-Philippe', 'Dominic', 'Morgane', 'Linda', 'Jean
-Philippe', 'Morgane', 'Louis','Renald', 'Renald'),

  ## Poids de chaque relation
  c( 0.95, 0.8, 0.4, 0.5, 0.2, 0.1, 0.1, 0.05, 0.5,  0.1, 0.8, 0.9, 0.85, 0.4, 0.75, 0.8,
0.95, 0.85, 0.65, 0.4 )
  )


colnames(dataset)<- c('Paire_1', 'Paire_2' ,'Poids') ## assigner des noms à
                ## chaque variable dans le jeu de données
dataset2<- data.frame(dataset) ## mettre en data.frame, sinon la
                    ##fonction simpleNetwork ne fonctionne pas
head(dataset2)
```

```
##   Paire_1       Paire_2 Poids
## 1   Janie       Nicolas  0.95
## 2   Janie      Danielle   0.8
## 3   Janie         Louis   0.4
## 4   Janie        Renald   0.5
## 5   Janie         Linda   0.2
## 6   Janie Jean-Philippe   0.1
```

Deuxième étape: Visualiser le réseau (en ligne) à partir du code R suivant:

```
simpleNetwork(dataset2[,1:2])                    ## voir le réseau
```



On peut calculer le nombre d’amis moyen pour chaque individu, ainsi que pour les amis de chaque individu, comme suit:

```
## considérer toutes les paires dans M
M=(rbind(as.matrix(dataset2[,1:2]),as.matrix(dataset2[,2:1])))

## Garder les paires distinctes
nodes=unique(M[,1])

## Fonction friends trouve toutes les paires contenant une personne donnée
friends = function(x) as.character(M[which(M[,1]==x),2])

## Fonction nb_friends compte le nombre d'amis de
## chaque ami d'une personne, e.g. nb_friends(friends('Janie')) montre
## le nombre d'amis de chaque ami de Janie

nb_friends = Vectorize(function(x) length(friends(x)))

## Fonction friends_of_friends et nb_friends_of_friends permettent de calculer nombre d'a
mis des amis

friends_of_friends = function(y) (Vectorize(function(x) length(friends(x)))(friends(y)))
nb_friends_of_friends = Vectorize(function(x) mean(friends_of_friends(x)))

## Ex. avec Janie:

nb_friends('Janie')
```

```
## Janie
##     8
```

```
friends_of_friends('Janie')
```

```
##       Nicolas      Danielle         Louis        Renald         Linda
##             7             4             4             3             3
## Jean-Philippe       Dominic       Morgane
##             3             5             3
```

```
nb_friends_of_friends('Janie')
```

```
## Janie
##     4
```

```
## Comparaison de ces nombres pour le graph complet (tout le monde):

Nb  = nb_friends(nodes)
Nb2 = nb_friends_of_friends(nodes)
mean(Nb)
```

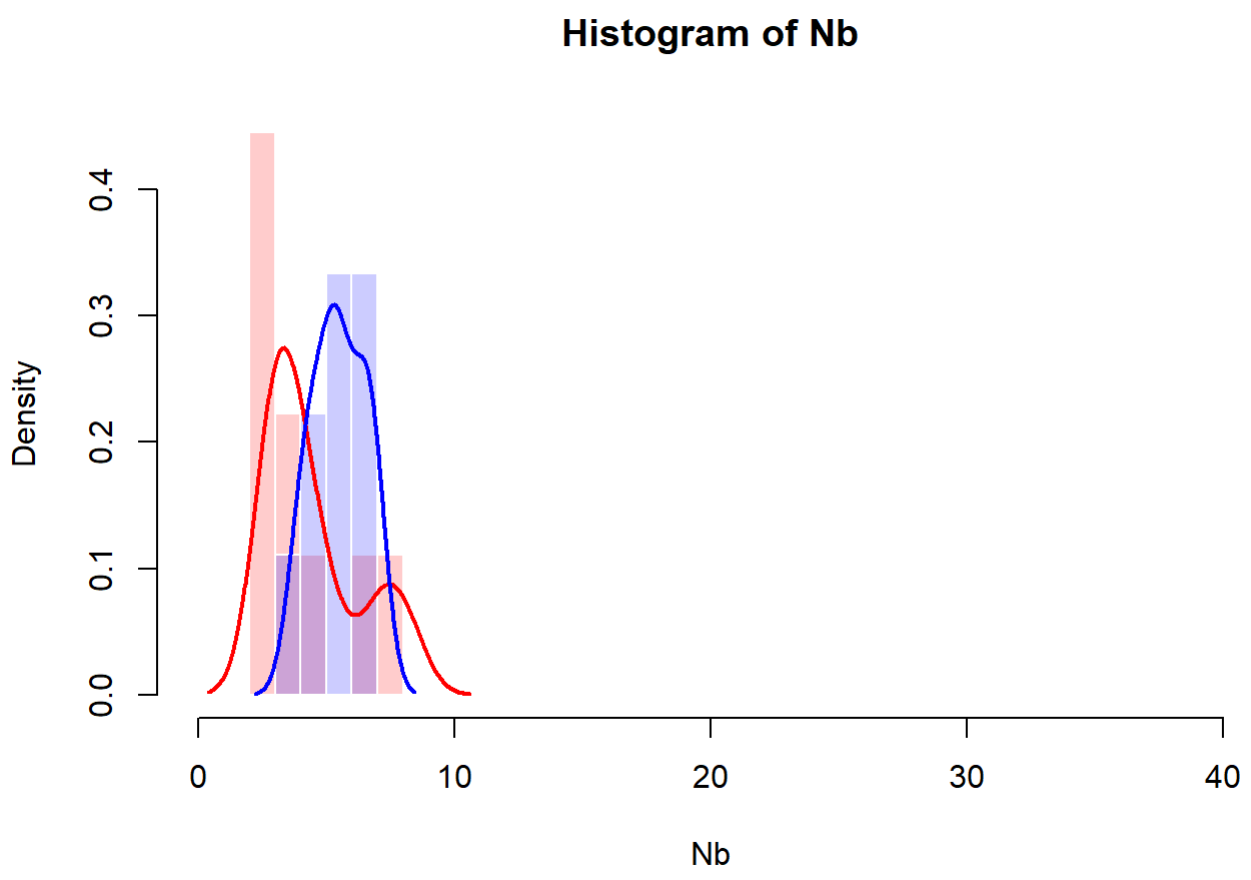
```
## [1] 4.444444
```

```
mean(Nb2)
```

```
## [1] 5.491005
```

On peut aussi visualiser l'histogramme du nombre d'amis et du nombre d'amis de ses amis, grâce aux commandes suivantes:

```
hist(Nb,breaks=0:40,col=rgb(1,0,0,.2),border="white",probability = TRUE)
hist(Nb2,breaks=0:40,col=rgb(0,0,1,.2),border="white",probability = TRUE,add=TRUE)
lines(density(Nb),col="red",lwd=2)
lines(density(Nb2),col="blue",lwd=2)
```



Exercice 3: Paradoxe de Berkson

Dans cet exercice, nous reproduisons les résultats de l'exercice 3. Nous verrons que l'effet de la maladie HA1P6N sur les risques d'infection à la COVID19 sont différents selon si l'on utilise les données en hôpital ou les données complètes!

Première étape: Simuler des données avec un germe (*set.seed*) précis:

```
set.seed(416667818) # germe de l'atelier pour obtenir les mêmes résultats qu'en classe: 4
16667818
```

Simulation des variables HA1P6N et covid19 (infection à la covid19) pour un échantillon de 100000 personnes:

```
taille<- 100000 ## taille d'échantillon
HA1P6N<- rbinom (n=taille, size=1, prob= 0.05 )

expit<- function(h){exp(h)/(1+exp(h))}
prob_covid<- expit( -5 + 0.05*HA1P6N)
covid19<- rbinom( n=taille, size=1, prob= prob_covid) ## simuler la variable COVID19 selo
n bernoulli
table(covid19) ## voir le nombre de covid19 dans l'échantillon
```

```
## covid19
##     0     1
## 99311   689
```

Simulation de la variable hospitalisation (où les chances d'hospitalisation sont affectées par la covid19 ainsi que la maladie HA1P6N):

```
prob_hospit<- expit( -5  +  1.5*HA1P6N + 0.05*covid19)
hospit<- rbinom( n=taille, size=1, prob=prob_hospit) ## simuler la variable hospitalisati
on
table(hospit) ## voir le nombre d'hospitalisations dans l'échantillon
```

```
## hospit
##     0     1
## 99197   803
```

Regardons la distribution de covid19 à travers ceux qui ont HA1P6N et ne l'ont pas, chez les hospitalisés seulement:

```
table( covid19[hospit==1], HA1P6N[hospit==1])
```

```
##
##       0   1
##   0 643 148
##   1   8   4
```

Calcul du taux:

```
print ((table( covid19[hospit==1], HA1P6N[hospit==1])[4]  / (table( covid19[hospit==1], H
A1P6N[hospit==1])[4] + table( covid19[hospit==1], HA1P6N[hospit==1])[3]))/
(table( covid19[hospit==1], HA1P6N[hospit==1])[2]  / (table( covid19[hospit==1], HA1P6N[h
ospit==1])[2] + table( covid19[hospit==1], HA1P6N[hospit==1])[1])) )
```

```
## [1] 2.141447
```

Maintenant, regardons la distribution de covid19 à travers ceux qui ont HA1P6N et ne l'ont pas, dans la population globale (et non plus chez les hospitalisés seulement):

```
table( covid19, HA1P6N)
```

```
##        HA1P6N
## covid19     0     1
##       0 94268  5043
##       1   659    30
```

Calcul du taux:

```
print( (table( covid19, HA1P6N)[4]  / (table( covid19, HA1P6N)[4] + table( covid19, HA1P6
N)[3]))/
( table( covid19, HA1P6N)[2]  / (table( covid19, HA1P6N)[2] + table( covid19, HA1P6N)[1
])) )
```

```
## [1] 0.8518453
```

L'effet est renversé dans la population globale. Cela est dû au paradoxe de Berkson, où l'hospitalisation agit comme un facteur de collision. Le fait de restreindre l'analyse aux gens hospitalisés crée une distortion de l'effet de la maladie HA1P6N sur l'infection à la covid19.