Introduction aux statistiques bayésiennes – code R

Le présent document a pour objectif d'introduire l'analyse de régression simple bayésienne. Cette analyse sera réalisée sur une base de données anonymisée portant sur une étude en psychologie sociale. Nous utiliserons l'identification sociale envers un groupe, une variable continue, pour prédire l'attitude envers le groupe, une seconde variable continue. L'objectif est de tester la relation linéaire entre l'identification et les attitudes favorables envers un groupe social.

Pour mener vos analyses, assurez-vous d'avoir les logiciels suivants installés sur votre ordinateur :

R: https://cran.r-project.org/
Rstudio: https://www.rstudio.com/products/rstudio/download/#download
JAGS: https://sourceforge.net/projects/mcmc-jags/files/

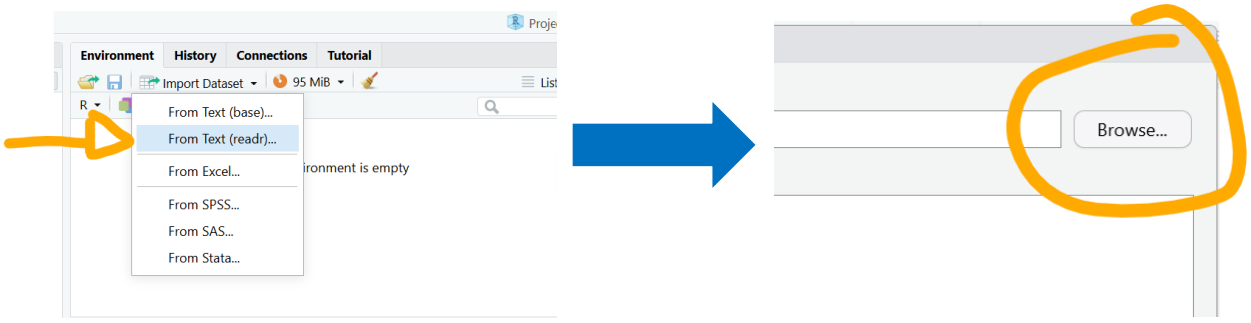
Étape 1 – installer et activer les packages nécessaires pour faire les analyses

Il est nécessaire d'installer certains *packages* dans notre environnement R afin d'exercer nos opérations. Les packages nécessaires sont R2jags (permet de faire les analyses bayésiennes), coda (permet d'analyser les MCMC) et readr (permet d'importer les données dans l'environnement R). Dans un premier temps, il faut installer les packages, avec la fonction *install.packages()* et ensuite les activer avec la fonction *library()*.

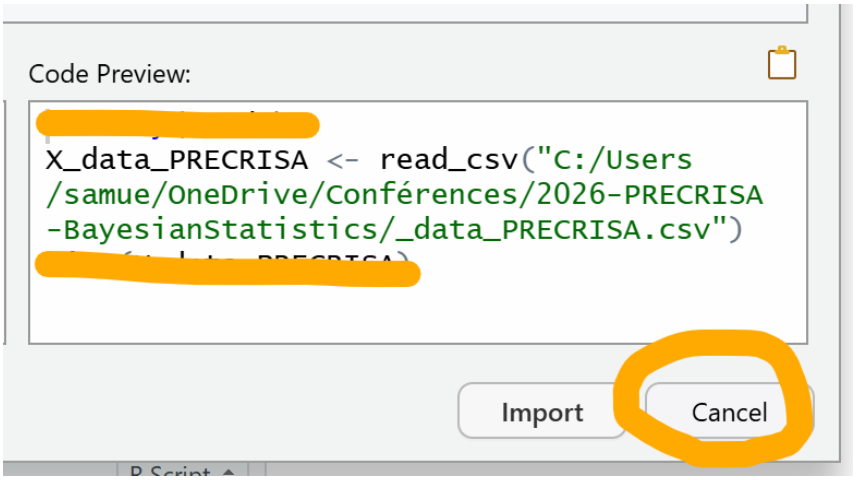
```
# ---- Packages ----
install.packages("R2jags")
install.packages("coda")}
install.packages("readr")

library(R2jags)
library(coda)
library(readr)
```

Étape 2 – importer les données

Importer la base de données dans l'environnement R à l'aide du menu déroulant *Import Dataset*, sélectionnez l'option *from Text (readr)* et allez chercher la base de données à l'emplacement de votre enregistrement.



Ensuite, assurez-vous de sélectionner le code permettant d'importer les données. Faites copier et coller le code dans votre script. De cette façon, vous pourrez plus facilement importer les données dans le logiciel R.



Étape 3 – Définition des variables

JAGS nécessite que les variables utilisées dans la base de données ainsi que les paramètres d'intérêt soient définies. Dans le cadre de la présente analyse, nous allons définir notre variable dépendante (attitudes) et indépendante (identification), ainsi que le nombre d'observations de la base de données (n=80) au sein de l'objet *df.data*. Aussi, nous allons spécifier à JAGS les paramètres utilisés dans l'analyse, c'est-à-dire, beta0, beta1 et sigma, et ce au sein de l'objet *df.para*.

```
# ---- Variables ----
y <- df$attitudes
x <- df$identification
N <- nrow(df)

df.data <- list("y", "x", "N")
df.para <- c("beta0", "beta1", "sigma")
```

Étape 4 – spécifier le modèle bayésien

Nous devons spécifier les fonctions a priori et *likelihood* de notre modèle bayésien. Le calcul du *likelihood* est déterminé à partir de la probabilité de chaque donnée, que nous multiplions ensemble. À cet effet, nous utiliserons une boucle pour répéter la commande du calcul des probabilités, et ce pour chacune de nos données. Une boucle permet de répéter des opérations un nombre désiré de fois. Nous utiliserons donc une boucle de 80 itérations puisque nous avons 80 données. Cette boucle est définie par la fonction `for(i in 1:N){}`. La probabilité des données est estimée en fonction d'une loi normale `dnorm()` centrée à la valeur prédite `mu` (c.-à-d., centrée sur la droite de régression). Il faudra considérer que `mu` change d'un.e participant.e à un.e autre. Donc, `mu` sera défini comme `mu[i]` (avec un « i » entre crochets) pour spécifier que les valeurs diffèrent pour chaque participant.e. Pour déterminer la valeur de `mu` de chacun.e des participant.e.s, nous utiliserons la formule de régression `mu[i] <- beta0 + beta1 * x[i]`.

Paramètre de précision (τ): JAGS construit ses distributions avec le paramètre de précision « tau » (et non les paramètres de variance ou d'écart-type). Ainsi, if faut faire la conversion de « sigma » à « tau ». tau est l'inverse de la variance, donc tau = 1 / (sigma * sigma).

Les « priors » sont nos croyances a priori sur nos trois paramètres. Dans le code ci-dessous, l'ordonnée à l'origine reçoit une croyance a priori normale centrée à zéro avec une très faible précision. Notre croyance a priori sur le paramètre de régression est définie selon une distribution normale centrée à .6 et notre croyance sur le paramètre sigma est une distribution uniforme qui s'étend de 0 à 100.

```
Md_1 <- function(){
  for(i in 1:N){

    # likelihood
    y[i] ~ dnorm(mu[i], tau)

    # linear predictor
    mu[i] <- beta0 + beta1 * x[i]
  }

  # priors
  beta0 ~ dnorm(0, .001)
  beta1 ~ dnorm(.6, 1)
  sigma ~ dunif(0, 100)
  tau <- 1 / (sigma * sigma)
```

Étape 5 – initialiser les chaînes de Markov (MCMC)

Les chaînes de Markov ont pour objectif d'explorer l'espace de nos paramètres et d'échantillonner les valeurs les plus probables en fonction des a priori et du *likelihood*. Pour ce faire, il faut spécifier des valeurs de départ de chaque paramètre (beta0, beta1 et sigma) de chacune des chaînes.

```
# Valeurs initiales MCMC
inits1 <- list("beta0"=0, "beta1"=0, "sigma"=0.1)
inits2 <- list("beta0"=1, "beta1"=1, "sigma"=0.2)
init_values <- list(inits1, inits2)
```

Étape 6 – rouler le modèle

Maintenant que nous avons spécifié les variables, les paramètres, le modèle et les chaînes de Markov, nous pouvons rouler le modèle grâce à la fonction jags(). JAGS nécessite la spécification de plusieurs éléments, dont les données (*data*), les valeurs initiales des chaînes MCMC (*inits*), les paramètres à estimer (*parameters.to.save*), le nombre de chaines MCMC (*n.chains*), certaines précisions par rapport aux chaînes MCMC (*n.iter*, *n.burn*, *n.thin*) et le modèle bayésien (*model.file*).

```
fit <- jags(data=df.data,
            inits=init_values,
            parameters.to.save=df.para,
            n.chains=2,
            n.iter=12000,
            n.burn=2000,
            n.thin=10,
            model.file=Md_1)

samples <- as.mcmc(fit)
plot(samples)
```