Annexe 2 : Script de la boucle de régression linéaire pour trouver la meilleur combinaison de tableaux de notation

```
####Etapes
#1.Construction des multiples tableaux de note par indicateur
#2.fonction qui donne toutes les combinaisons possibles de tableaux de note 1 par 1
        #2.1.fonction qui donne la note finale de tous les relevés grâce à la combinaison
        #obtenue avant
        #2.2.fonction qui renvoie les résultats de la régression linaire entre les notes
        #finales et l'avis d'expert
#3.fonction qui selectionne les N (10 par exemple) meilleurs R².

####nettoyage de la mémoire
rm(list = ls())
#########################################################################################################
####
# 1. construction des listes de tableaux de notes (1 liste par indicateur)
#########################################################################################################
####
####préparation indicateurs à 2 classes

ind.2.classes.1=data.frame(classes=c(1,2), note=c(0,-30))
ind.2.classes.2=data.frame(classes=c(1,2), note=c(0,-60))

ind.2.classes=list(ind.2.classes.1, ind.2.classes.2)

####préparation indicateurs à 3 classes

ind.3.classes.1=data.frame(classes=c(1,2,3), note=c(0,-5,-10))
ind.3.classes.2=data.frame(classes=c(1,2,3), note=c(0,-10,-20))
ind.3.classes.3=data.frame(classes=c(1,2,3), note=c(0,-15,-30))
ind.3.classes.4=data.frame(classes=c(1,2,3), note=c(0,-20,-40))

ind.3.classes=list(ind.3.classes.1, ind.3.classes.2, ind.3.classes.3, ind.3.classes.4)

        fct.hydro=ind.2.classes
        surface=ind.3.classes
        macro.temp=ind.3.classes
        evee=ind.3.classes
        eaee=ind.3.classes
        berges=ind.3.classes

#########################################################################################################
####
#Boucle de selection de la (des) combinaison(s) qui donne(nt) la(les) meilleure(s)
#regression(s) linéaire(s) avec l'avis d'expert.
#########################################################################################################
####

data = read.csv("data_temp_sans_graus.csv",
        header=TRUE, sep=";")
        #data est un tableau comprenant les pièces d'eau en 1ere colonne, les indicateurs
        #de la col. 2 à la col.8, l'avis d'expert à la col.9. Les colonnes d'indicateurs
        #sont remplis par les classes de modalités.
        #Attention : impératif de remplacer les absences de données par NA, sinon le
        #calcul foire !

source("note_totale_med_temp_sans_graus.r")
source("script_fct_reglin.r")
source("script_fct_findNbestR_med_temp_sans_graus.r")
        #cf détail de ces fonctions dans leurs fichiers.

N=3                                     #Nombre de combinaisons de tableaux qui présentent les
                                        #meilleurs R² qu'on veut avoir.
bestr = rep(0,N)                #objet qui contiendra à la fin de la boucle, les N
                                        #meilleurs R²
```

```
best_notation = list()                          #contiendra les tableaux de note qui ont donné ce R²
best_LM = list()

for (i in c(1:length(fct.hydro))){
    for (k in c(1:length(surface))){
        for (m in c(1:length(evee))){
            for (n in c(1:length(eaee))){
                                    for (q in c(1:length(berges))){
                                            for (r in c(1:length(macro.temp))){
notation=c(fct.hydro[i], surface[k], evee[m], eaee[n], berges[q], macro.temp[r])
#donne une combinaison de tableaux de notation

                    tabletotest = notetotale(data, notation)
                            #fonction qui permet de donner l'ensemble des notes totales des stations
                            #grâce à la combinaison de tableaux de notes créée avant.

                    LM = reglin(data, tabletotest)
                            #fait une regression linéaire entre l'avis d'expert et les notes totales
                            #calculées par la fonction précédente.

                    results=findNbestR(bestr, LM, notation, best_notation, best_LM)
                    bestr=results$bestr
                    best_notation = results$best_notation
                    best_LM = results$best_LM
                            #fonction qui compare le R² de la reglin avec ceux enregistrés dans bestr,
                            #si le R²>min(bestr) alors il remplace le min(bestr) par le nouveau R²,
                            #sinon il fait rien.
                            #Si le R² est remplacé, la fct remplace aussi le tableau de notation qui va
                            #avec dans l'objet best_notation.
}}}}}}
#vérifications :
bestr
results$best_notation
length(results$best_notation)
#results$best_LM
#length(results$best_LM)
##########################################################################################################
####
#Tracé des graphes de regression linéaire + vérification des hypothèses
##########################################################################################################
####
#si on demande au programme précedent d'extraire aussi les régressions linéaires qui vont avec
#les meilleurs R² ca ne marche pas : dans la fonction effects, l'objet n'est pas reconnu comme
#un argument de type mod. Du coup, il faut refaire la regression linéaire sur la base du
#tableau de notation extrait par le programme, et donc refaire fonctionner aussi la fonction
#notetotale.
tablechosen= notetotale(data, results$best_notation[1:6])

confrontation = data.frame(station = data$station, note_totale = tablechosen$note_totale,
        note_expert = data$note_expert)
LM = lm(note_totale~note_expert, data=confrontation)

require(effects)
#note.eff = effect(term="note_totale", mod=LM)
note.eff = effect(term="note_expert", mod=LM)

plot(confrontation[,2]~confrontation[,3], ylab="note de la méthode", xlab = "note de l'expert",
        main = paste("Meilleure reg.lin note-expert vs note-methode -- R²=", round(summary(LM)$r.squared,3)))
lines(note.eff$x[,1],note.eff$fit)
lines(note.eff$x[,1],note.eff$lower, col="red", lty=2)
lines(note.eff$x[,1],note.eff$upper, col="red", lty=2)

require(RVAideMemoire)
plotresid(LM, shapiro=TRUE)
        #p-value<0.1. condition du test de shapiro-Wilk
        #graphe de gauche : l'hypothèse d'équivariance est acceptée lorsque la dispersion
        #verticale des points est à peu près constante sur toute la longueur de l'axe des
        #abscisses. L'hypothèse d'indépendance est acceptée lorsque l'orientation du nuage de
        #points est horizontale
```

```
	#Graphe de droite : l'hypothèse de normalité est acceptée lorsque les points sont à
	#peu près alignés sur une droite
tablechosen = notetotale(data, notation)
write.table(tablechosen$note_totale)
```

```
##### fonction de calcul de la note

notetotale = function(data, notation)
{
	#data est un tableau comprenant les noms de relevés en premiere colonne, puis les
	#classes et valeurs des indicateurs respectivement qualitatifs et quantitatifs,
	#et enfin l'avis d'expert.
	#notation est la liste comprenant une combinaison de tableaux de notation.
tabletotest = data.frame (station=data$station, fct.hydro=0, surface=0, evee=0, eaee=0, berges=0,
macro.temp=0)

for (i in c(1:nrow(notation[[1]]))){
	elem = data$station[data$fct.hydro == notation[[1]]$classes[i]]
	tabletotest$fct.hydro[is.element(data$station, elem)] = notation[[1]]$note[i]}
for (i in c(1:nrow(notation[[2]]))){
	elem = data$station[data$surface == notation[[2]]$classes[i]]
	tabletotest$surface[is.element(data$station, elem)] = notation[[2]]$note[i]}
for (i in c(1:nrow(notation[[3]]))){
	elem = data$station[data$evee == notation[[3]]$classes[i]]
	tabletotest$evee[is.element(data$station,elem)] = notation[[3]]$note[i]}
for (i in c(1:nrow(notation[[4]]))){
	elem = data$station[data$eaee == notation[[4]]$classes[i]]
	tabletotest$eaee[is.element(data$station,elem)] = notation[[4]]$note[i]}
for (i in c(1:nrow(notation[[5]]))){
	elem = data$station[data$macro.temp == notation[[5]]$classes[i]]
	tabletotest$macro.temp[is.element(data$station,elem)] = notation[[5]]$note[i]}
for (i in c(1:nrow(notation[[6]]))){
	elem = data$station[data$berges == notation[[6]]$classes[i]]
	tabletotest$berges[is.element(data$station,elem)] = notation[[6]]$note[i]}

tabletotest$note_totale = 100 + rowSums(tabletotest[2:ncol(tabletotest)])
return (tabletotest)

}
```

```
#### fonction de calcul du R² avec l'avis d'expert

reglin= function(data, tabletotest)
{
confrontation = data.frame(station = data$station, note_totale = tabletotest$note_totale, note_expert =
data$note_expert)
confrontation.lm = lm(note_totale~note_expert, data=confrontation)
r_squared = round(summary(confrontation.lm)$r.squared,3)

return(confrontation.lm)
}
```

```
findNbestR = function(bestr, LM, notation, best_notation, best_LM)
{
R = summary(LM)$r.squared
if(R > min(bestr)){
	a=which.min(bestr)
	bestr[a]= R
	best_notation[((a-1)*6+1):(a*6)]=notation
	best_LM[((a-1)*13+1):(a*13)]=LM}
results=list(bestr=bestr, best_notation=best_notation, best_LM=best_LM)
return(results)
}
```