Annexe 2

Jean-Yves Barnagaud & Olivier Gimenez

juin 2023

ATTENTION : MODIFIER LE CHEMIN D’ACCES VERS VOTRE REPERTOIRE COURANT AVANT DE LANCER CE SCRIPT

# packages utiles et options du script

## packages

# génériques  
library(tidyverse)

> ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
> ✔ dplyr 1.1.1 ✔ readr 2.1.4  
> ✔ forcats 1.0.0 ✔ stringr 1.5.0  
> ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
> ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
> ✔ purrr 1.0.1   
> ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
> ✖ dplyr::filter() masks stats::filter()  
> ✖ dplyr::lag() masks stats::lag()  
> ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(formatR)  
library(patchwork)   
library(viridis)

> Le chargement a nécessité le package : viridisLite

library(lmtest)

> Le chargement a nécessité le package : zoo  
>   
> Attachement du package : 'zoo'  
>   
> Les objets suivants sont masqués depuis 'package:base':  
>   
> as.Date, as.Date.numeric

## répertoire courant

Attention : le répertoire de travail est changé dans l’en-tête du script Markdown.

setwd('A:/Biotope\_analyse\_donnees/annexes')

## options graphiques

Thème par défaut des graphiques

theme\_set(theme\_light(base\_size = 16))

Attention : les figures sont exportées dans des chunks de code à part qui sont invisibles dans le document généré par ce script et non exécutés. Pour exécuter ces chunks de code, aller dans le script Markdown et passer ces chunks de code en *eval = T*.

## Reproductibilité des simulations

set.seed(2020)

# Comment s’en sortir?

Les données

bufo <- read.csv2("donnees/pheno\_bufobufo.csv",dec=".")

La régression du chapitre 7

mod.bufo <- lm(date\_resc ~ TmaxFev.Mars, data = bufo)  
summary(mod.bufo)

>   
> Call:  
> lm(formula = date\_resc ~ TmaxFev.Mars, data = bufo)  
>   
> Residuals:  
> Min 1Q Median 3Q Max   
> -42.158 -3.528 0.948 7.246 20.368   
>   
> Coefficients:  
> Estimate Std. Error t value Pr(>|t|)   
> (Intercept) 136.548 20.567 6.639 2.37e-07 \*\*\*  
> TmaxFev.Mars -5.873 1.461 -4.020 0.000361 \*\*\*  
> ---  
> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
>   
> Residual standard error: 12.53 on 30 degrees of freedom  
> Multiple R-squared: 0.3501, Adjusted R-squared: 0.3285   
> F-statistic: 16.16 on 1 and 30 DF, p-value: 0.0003612

les paramètres à conserver :

pente <- -6   
ordonnee\_origine <- 140   
sd\_residuelle <- 10

L’effectif :

taille\_echantillon <- 10

On simule des températures dans une gamme plausible, en l’occurrence entre le minimum et le maximum observés dans les données sur le crapaud commun :

variable\_explicative <- seq(min(bufo$TmaxFev.Mars),  
 max(bufo$TmaxFev.Mars),  
 length = taille\_echantillon)  
variable\_explicative

> [1] 11.66250 12.30972 12.95694 13.60417 14.25139 14.89861 15.54583 16.19306  
> [9] 16.84028 17.48750

On simule la variable de réponse :

moyenne\_reponse <- ordonnee\_origine + pente \* variable\_explicative

On ajoute la variabilité :

variable\_reponse <- moyenne\_reponse + rnorm(n = taille\_echantillon,   
 mean = 0,   
 sd = sd\_residuelle)  
variable\_reponse

> [1] 73.79472 69.15715 51.27810 47.07094 26.52632 57.81407 56.11621 40.54789  
> [9] 56.54965 36.24867

On essaie maintenant de retrouver les paramètres :

reg <- lm(variable\_reponse ~ variable\_explicative)   
summary(reg)

>   
> Call:  
> lm(formula = variable\_reponse ~ variable\_explicative)  
>   
> Residuals:  
> Min 1Q Median 3Q Max   
> -26.261 -6.107 1.906 8.640 13.978   
>   
> Coefficients:  
> Estimate Std. Error t value Pr(>|t|)   
> (Intercept) 109.023 32.508 3.354 0.010 \*  
> variable\_explicative -3.946 2.212 -1.784 0.112   
> ---  
> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
>   
> Residual standard error: 13.01 on 8 degrees of freedom  
> Multiple R-squared: 0.2845, Adjusted R-squared: 0.1951   
> F-statistic: 3.181 on 1 and 8 DF, p-value: 0.1123

On extrait la p-value :

p\_value <- coef(summary(reg))["variable\_explicative", "Pr(>|t|)"]  
  
p\_value

> [1] 0.1123447

## Déterminer la taille d’échantillon nécessaire

Nombre de simulations :

nombre\_simulations <- 400

On fait varier la taille d’échantillon de 10 à 100, par pas de 5

taille\_echantillon <- seq(5, 100, by = 5)

on prépare un vecteur pour stocker les p-values :

p\_value <- rep(NA, nombre\_simulations)

et pour stocker l’indicateur de puissance :

puissance <- rep(NA, length(taille\_echantillon))

La simulation : on boucle sur les tailles d’échantillon qu’on veut tester, et pour chaque taille d’échantillon on fait 400 tirages.

# première boucle : on répète toutes les commandes sur toutes les tailles d'échantillon  
  
for (j in 1:length(taille\_echantillon)) {  
   
 variable\_explicative <- seq(min(bufo$TmaxFev.Mars),  
 max(bufo$TmaxFev.Mars),  
 length = taille\_echantillon[j])  
  
# deuxième boucle, imbriquée dans la précédente : on répète 400 fois la simulation   
   
 for (i in 1:nombre\_simulations) {  
   
 moyenne\_reponse <- ordonnee\_origine + pente \* variable\_explicative  
   
 variable\_reponse <- rnorm(taille\_echantillon[j],  
 mean = moyenne\_reponse,  
 sd = sd\_residuelle)  
   
 reg <- lm(variable\_reponse ~ variable\_explicative)  
   
 p\_value[i] <-  
 coef(summary(reg))["variable\_explicative", "Pr(>|t|)"]  
   
 }  
   
 puissance[j] <- sum(p\_value < 0.05) / nombre\_simulations  
}  
  
# stocker les tailles d'échantillons et les puissances dans un tableau  
  
df <- data.frame(taille\_echantillon, puissance)

Résultats :

round(df, 2)

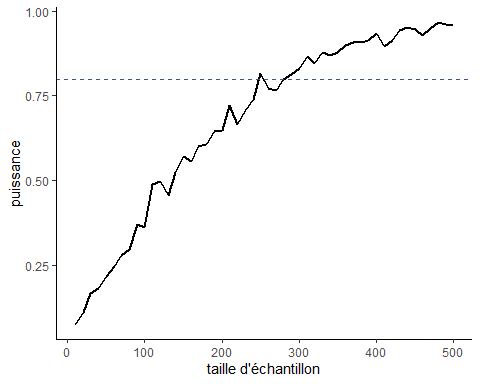
> taille\_echantillon puissance  
> 1 5 0.47  
> 2 10 0.87  
> 3 15 0.97  
> 4 20 0.99  
> 5 25 1.00  
> 6 30 1.00  
> 7 35 1.00  
> 8 40 1.00  
> 9 45 1.00  
> 10 50 1.00  
> 11 55 1.00  
> 12 60 1.00  
> 13 65 1.00  
> 14 70 1.00  
> 15 75 1.00  
> 16 80 1.00  
> 17 85 1.00  
> 18 90 1.00  
> 19 95 1.00  
> 20 100 1.00

# La relation taille d’échantillon – puissance pour un effet faible

La simulation : on garde tous les paramètres identiques à la simulation précédente, sauf la pente

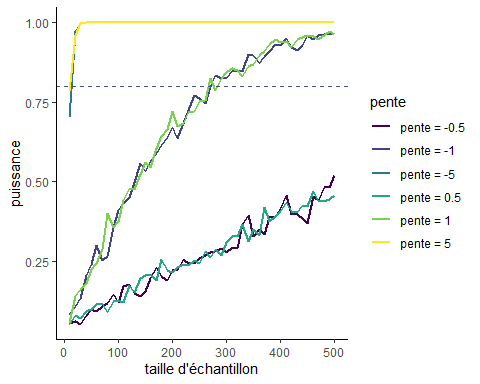
pente <- -1  
ordonnee\_origine <- 140  
sd\_residuelle <- 10  
  
nombre\_simulations <- 400  
  
taille\_echantillon <- seq(10, 500, by = 10)  
  
p\_valeur <- rep(NA, nombre\_simulations)  
puissance <- rep(NA, length(taille\_echantillon))  
  
for (j in 1:length(taille\_echantillon)) {  
 variable\_explicative <- seq(min(bufo$TmaxFev.Mars),  
 max(bufo$TmaxFev.Mars),  
 length = taille\_echantillon[j])  
   
 for (i in 1:nombre\_simulations) {  
 moyenne\_reponse <- ordonnee\_origine + pente \* variable\_explicative  
   
 variable\_reponse <- rnorm(taille\_echantillon[j],  
 mean = moyenne\_reponse,  
 sd = sd\_residuelle)  
   
 reg <- lm(variable\_reponse ~ variable\_explicative)  
   
 p\_valeur[i] <-  
 coef(summary(reg))["variable\_explicative", "Pr(>|t|)"]  
 }  
 puissance[j] <- sum(p\_valeur < 0.05) / nombre\_simulations  
}

Résultat :



# Autres paramètres à étudier

pente <- c(-5, -1, -0.5, 0.5, 1, 5)  
ordonnee\_origine <- 140  
sd\_residuelle <- 10  
  
nombre\_simulations <- 400  
  
taille\_echantillon <- seq(10, 500, by = 10)  
  
p\_valeur <- rep(NA, nombre\_simulations)  
puissance0 <- rep(NA, length(taille\_echantillon))  
puissance <-  
 matrix(NA, nrow = length(puissance0), ncol = length(pente))  
  
for (k in 1:length(pente)) {  
 for (j in 1:length(taille\_echantillon)) {  
 variable\_explicative <- seq(min(bufo$TmaxFev.Mars),  
 max(bufo$TmaxFev.Mars),  
 length = taille\_echantillon[j])  
   
 for (i in 1:nombre\_simulations) {  
 moyenne\_reponse <-  
 ordonnee\_origine + pente[k] \* variable\_explicative  
   
 variable\_reponse <- rnorm(taille\_echantillon[j],  
 mean = moyenne\_reponse,  
 sd = sd\_residuelle)  
   
 reg <- lm(variable\_reponse ~ variable\_explicative)  
   
 p\_valeur[i] <-  
 coef(summary(reg))["variable\_explicative", "Pr(>|t|)"]  
 }  
 puissance0[j] <- sum(p\_valeur < 0.05) / nombre\_simulations  
 }  
 puissance[, k] <- puissance0  
}  
  
df.k <- as.data.frame(puissance)  
colnames(df.k) <- paste("pente =", pente, sep = " ")  
df.k.w <- as.data.frame(pivot\_longer(df.k, cols = 1:length(pente)))  
df.k.w$sample.size <- rep(taille\_echantillon, each = length(pente))  
  
ggplot(df.k.w) +  
 aes(x = sample.size, y = value, col = name) +  
 geom\_line(linewidth = 1) +  
 geom\_hline(yintercept = 0.8,  
 color = "#3b528b",  
 lty = 2) +  
 labs(x = "taille d'échantillon", y = "puissance",col="pente") +  
 theme\_classic()+  
 scale\_color\_viridis\_d()



ggsave("outputs/A2F2.png", width = 8, height = 5)

# Test de puissance pour un GLM

lomolino <- read.csv2("donnees/lomolino.csv", dec = ".")  
lomolino$log.surface <- log(lomolino$surface)

Le GLM du chapitre 8 :

m1 <- glm(n\_especes ~ log.surface + distance\_source + latitude,  
 family = "poisson",  
 data = lomolino)

m2 <- glm(n\_especes ~ log.surface,  
 family = "poisson",  
 data = lomolino)

lrtest(m1, m2)

> Likelihood ratio test  
>   
> Model 1: n\_especes ~ log.surface + distance\_source + latitude  
> Model 2: n\_especes ~ log.surface  
> #Df LogLik Df Chisq Pr(>Chisq)   
> 1 4 -52.795   
> 2 2 -57.843 -2 10.095 0.006424 \*\*  
> ---  
> Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

Pour l’analyse de puissance, on reprend les paramètres du m1 :

ordonnee\_origine <- 1   
pente\_surf <- 0.2   
pente\_dsour <- -0.005   
pente\_lati <- 0.005

On tente une dizaine de tailles d’échantillons :

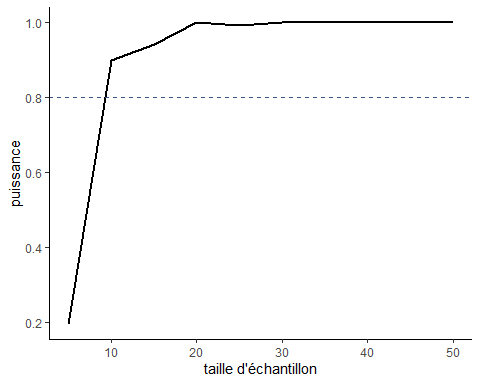
taille\_echantillon <- seq(5, 50, by = 5)

Les variables explicatives :

# stockage des valeurs d'intérêt  
  
p\_value <- rep(NA, nombre\_simulations)  
puissance <- rep(NA, length(taille\_echantillon))  
  
# variables explicatives  
  
for (j in 1:length(taille\_echantillon)) {  
 var\_surf <- runif(  
 min = min(lomolino$surface),  
 max = max(lomolino$surface),  
 n = taille\_echantillon[j]  
 )  
 log.var\_surf <- log(var\_surf)  
   
 var\_dsour <- runif(  
 min = min(lomolino$distance\_source),  
 max = max(lomolino$distance\_source),  
 n = taille\_echantillon[j]  
 )  
   
 var\_lati <- runif(  
 min = min(lomolino$latitude),  
 max = max(lomolino$latitude),  
 n = taille\_echantillon[j]  
 )  
   
 # variable de réponse  
   
 for (i in 1:nombre\_simulations) {  
 log\_lambda <- ordonnee\_origine +  
 pente\_surf \* log.var\_surf +  
 pente\_dsour \* var\_dsour +  
 pente\_lati \* var\_lati  
   
   
 variable\_reponse <- rpois(n = taille\_echantillon[j],  
 lambda = exp(log\_lambda))  
   
 # hypothèse 1  
   
 glm1 <-  
 glm(variable\_reponse ~ var\_surf + var\_dsour + var\_lati, family = "poisson")  
   
 # hypothèse 2  
   
 glm2 <- glm(variable\_reponse ~ var\_surf, family = "poisson")  
   
 # ratio de vraisemblances  
   
 test <- lrtest(glm1, glm2)  
 p\_value[i] <- test$`Pr(>Chisq)`[2]  
   
 }  
   
 puissance[j] <- sum(p\_value < 0.05) / nombre\_simulations  
   
}

La relation effectif - puissance :

df.glm <- data.frame(taille\_echantillon,puissance)  
  
ggplot(df.glm) +  
 aes(x = taille\_echantillon, y = puissance) +  
 geom\_line(linewidth = 1) +  
 geom\_hline(yintercept = 0.8,  
 color = "#3b528b",  
 lty = 2) +  
 labs(x = "taille d'échantillon", y = "puissance") +  
 theme\_classic()+  
 scale\_color\_viridis\_d()



ggsave("outputs/A2F3.png", width = 6, height = 6)