Fiche TD avec le logiciel R : onls

Modélisation de l'auto-corrélation temporelle des glandées par régression orthogonale non-linéaire

Pr Jean R. Lobry

À partir du suivi pendant 13 ans de la production de glands par 10 chênes sessiles dans la forêt de Seillon on montre comment on peut utiliser la régression orthogonale non-linéaire pour résumer l'auto-corrélation temporelle. On préfèrera au final par travailler avec des résidus parallèles à la première bissectrice.

Table des matières

1 Les données illustratives

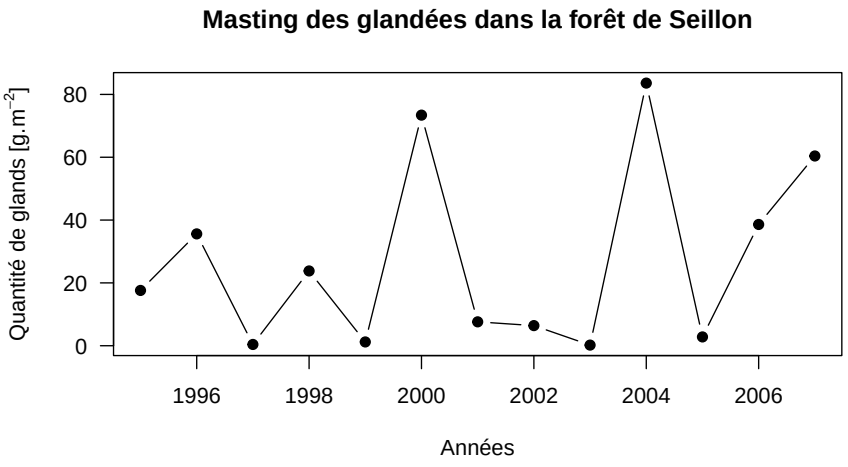
LES données sont extraites de la base de données MASTREE+ [2]. La version est celle utilisée dans la fiche descriptive des variables[1]. On en extrait les données sur les glandées des chênes sessiles (*Quercus petrae*) de la forêt de Seillon (département de l'Ain en France) fournies par le réseau RENECOFOR que nous remercions au passage pour les avoir mises à disposition.

```
load(url("https://pbil.univ-lyon1.fr/R/donnees/mastree.Rda"))
fds <- subset(mastree, Site == "CHS 01") # Forêt de Seillon
fds[ , c("Species", "Year", "Value", "Unit", "No_indivs")]
```

```
              Species Year Value Unit No_indivs
39468 Quercus petraea 1995  17.6 g/m2        10
39469 Quercus petraea 1996  35.6 g/m2        10
39470 Quercus petraea 1997   0.4 g/m2        10
39471 Quercus petraea 1998  23.8 g/m2        10
39472 Quercus petraea 1999   1.2 g/m2        10
39473 Quercus petraea 2000  73.4 g/m2        10
39474 Quercus petraea 2001   7.6 g/m2        10
39475 Quercus petraea 2002   6.4 g/m2        10
39476 Quercus petraea 2003   0.2 g/m2        10
39477 Quercus petraea 2004  83.6 g/m2        10
39478 Quercus petraea 2005   2.8 g/m2        10
39479 Quercus petraea 2006  38.6 g/m2        10
39480 Quercus petraea 2007  60.4 g/m2        10
```

LA représentation graphique des données illustre le phénomène de *masting*, à savoir une production épisodique et synchrone des glands par les chênes :

```
x <- fds$Year ; y <- fds$Value
par(mar = c(5, 5, 4, 2) + 0.1)
plot(x, y, pch = 19, type = "b", las = 1,
     main = "Masting des glandées dans la forêt de Seillon", xlab = "Années",
     ylab = expression(paste("Quantité de glands [g.", m^-2, "]")))
```



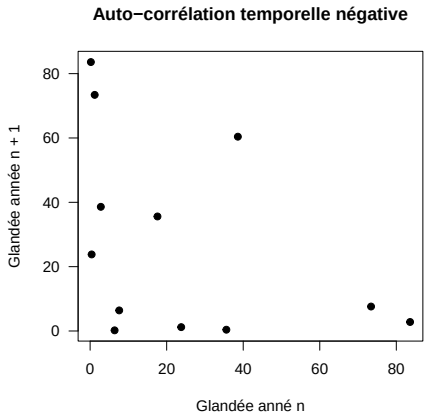
2 Auto-corrélation temporelle

CHEZ le chêne on observe une auto-corrélation temporelle négative entre les glandées dans le sens où une forte glandée l'année n implique une faible

[1] Voir `https://pbil.univ-lyon1.fr/R/pdf/MASTREE.pdf`

glandée l'année $n+1$, mais une faible glandée l'année n n'implique pas automatiquement une forte glandée l'année $n+1$. Ceci conduit à un nuage de points en forme de « L » dans le plan (y_n, y_{n+1}) :

```
n <- length(y)
yn <- y[-n]   # Année n
ynpu <- y[-1] # Année n + 1
plot(yn, ynpu, pch = 19, xlab = "Glandée anné n", ylab = "Glandée année n + 1",
     las = 1, main = "Auto-corrélation temporelle négative")
```



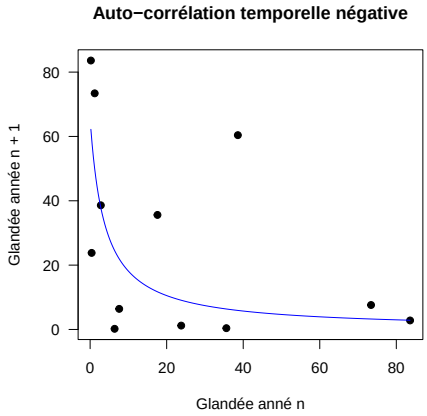
3 Régression non-linéaire classique

POUR résumer la forme du nuage de points dans le plan (y_n, y_{n+1}) on décide d'utiliser une branche d'hyperbole équilatère d'équation :

$$y = f(x) = \frac{\lambda^2}{\beta + x} \tag{1}$$

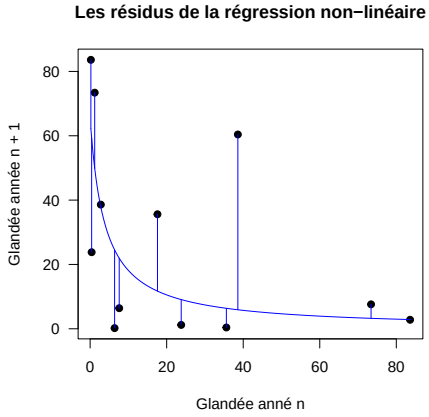
La fonction standard `nls()` permet facilement d'ajuster ce modèle :

```
df <- list(yn = yn, ynpu = ynpu)
resnls <- nls(ynpu ~ lambda^2/(beta + yn), data = df,
              start = list(lambda = 100, beta = 10))
plot(yn, ynpu, pch = 19, xlab = "Glandée anné n", ylab = "Glandée année n + 1",
     las = 1, main = "Auto-corrélation temporelle négative")
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, predict(resnls, list(yn = yy)), type = "l", col = "blue")
```



Le problème est qu'en régression non-linéaire classique on cherche à minimiser la somme des carrés des résidus entre les valeurs observées, y_i, et les valeurs prédites par le modèle, $f(x_i)$, soit graphiquement :

```
plot(yn, ynpu, pch = 19, xlab = "Glandée anné n", ylab = "Glandée année n + 1",
     las = 1, main = "Les résidus de la régression non-linéaire")
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, predict(resnls, list(yn = yy)), type = "l", col = "blue")
segments(yn, ynpu, yn, predict(resnls, list(yn = yn)), col = "blue")
```



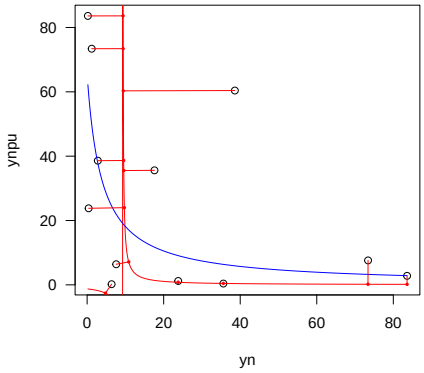
Cela signifie concrètement que la variable en x est censée être une variable de contrôle, ou tout du moins connue avec une très grande précision, et que les erreurs ne sont que sur la variable en y. C'est particulièrement idiot dans notre cas puisque nous avons la *même* variable en x et en y, simplement translatée dans le temps.

4 Régression orthogonale non-linéaire

En régression orthogonale l'erreur est aussi bien en x qu'en y. J'ai utilisé le paquet `onls` [3], il n'est plus maintenu (depuis le 2022-05-06) sur le CRAN à la date de rédaction de ce document, je suis parti de la dernière archive disponible (2015-09-09). Il a besoin du paquet `minpack.lm` [1]. La fonction `onls()`

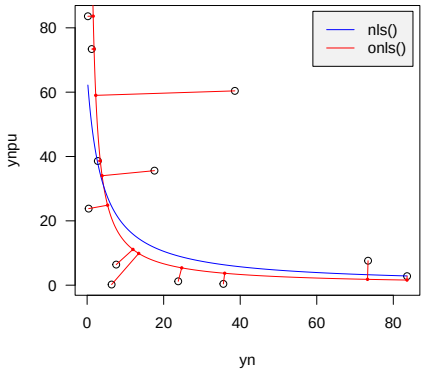
fonctionne de façon très similaire à `nls()`. Ma première tentative a été un échec parce que je convergeais vers une solution avec une asymptote verticale d'abscisse positive :

```
library(onls)
resonls <- onls(ynpu ~ lambda^2/(beta + yn), data = df,
                start = list(lambda = 100, beta = 10), verbose = FALSE)
plot(resonls, las = 1)
```



J'ai donc forcé la positivité du paramètre β :

```
resonls <- onls(ynpu ~ lambda^2/(abs(beta) + yn), data = df,
                start = list(lambda = 100, beta = 10), verbose = FALSE)
plot(resonls, las = 1)
legend("topright", inset = 0.02, legend = c("nls()", "onls()"), lty = 1,
       col = c("blue", "red"), bg = grey(0.95))
```



La fonction générique `print()` donne des résultats importants :

```
print(resonls)
```

```
Nonlinear orthogonal regression model
  model: ynpu ~ lambda^2/(abs(beta) + yn)
   data: df
    lambda       beta
1.154e+01 -8.482e-08
vertical residual sum-of-squares: 534719
orthogonal residual sum-of-squares: 1790
PASSED: 12 out of 12 fitted points are orthogonal.
Number of iterations to convergence: 16
Achieved convergence tolerance: 1.49e-08
```

Ce qui est important ici est que la solution obtenue est telle que tous les points soient orthogonaux à la courbe. On voit aussi que la somme des carrés des résidus a considérablement été améliorée par rapport à la régression classique. Si un point n'est pas orthogonal on pourra le repérer avec la composante `ortho` :

```
resonls$ortho
```
```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

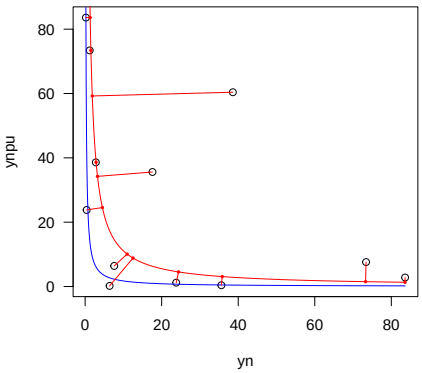
La fonction générique `summary()` donne des information sur la valeur des paramètres :

```
summary(resonls)
```
```
Formula: ynpu ~ lambda^2/(abs(beta) + yn)
Parameters:
         Estimate Std. Error t value Pr(>|t|)
lambda  1.154e+01  6.647e+00   1.736    0.113
beta   -8.482e-08  2.473e-01   0.000    1.000

Residual standard error of vertical distances: 231.2 on 10 degrees of freedom
Residual standard error of orthogonal distances: 13.38 on 10 degrees of freedom

Number of iterations to convergence: 16
Achieved convergence tolerance: 1.49e-08
```

On constate ici que le paramètre β est epsilonesque, on a donc tout intérêt à simplifier le modèle en posant $\beta = 0$, ce qui nous évitera au passage d'avoir à forcer sa positivité de façon un peu brutale. Notre modèle simplifié est donc :

$$y = g(x) = \frac{\lambda^2}{x} \tag{2}$$

```
mod2 <- onls(ynpu ~ lambda^2/yn, data = df,
             start = list(lambda = 10), verbose = FALSE)
plot(mod2, las = 1)
```



Cela ne change pas grand chose et est bien plus simple. Vérifions que tout va bien :

```
print(mod2)
```

```
Nonlinear orthogonal regression model
  model: ynpu ~ lambda^2/yn
   data: df
lambda
 10.53
 vertical residual sum-of-squares: 369323
 orthogonal residual sum-of-squares: 1773
 PASSED: 12 out of 12 fitted points are orthogonal.
Number of iterations to convergence: 21
Achieved convergence tolerance: 1.49e-08
```

```
summary(mod2)
```

```
Formula: ynpu ~ lambda^2/yn
Parameters:
       Estimate Std. Error t value Pr(>|t|)
lambda   10.527      1.536   6.855 2.75e-05 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error of vertical distances: 183.2 on 11 degrees of freedom
Residual standard error of orthogonal distances: 12.7 on 11 degrees of freedom

Number of iterations to convergence: 21
Achieved convergence tolerance: 1.49e-08
```

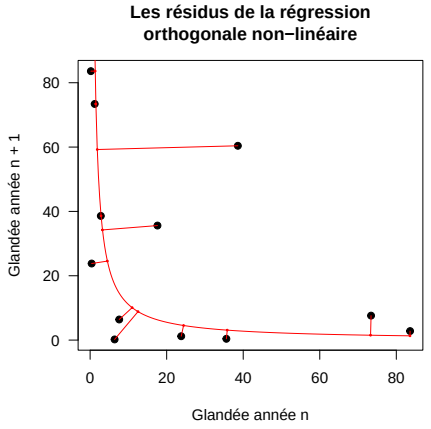
Il faut utiliser la fonction `residuals_o()` pour calculer les résidus orthogonaux :

```
residuals_o(mod2)
```

```
 [1]  1.12542301  4.18174014  0.30970541  0.07071625 10.59706930  5.00598707
 [7] 14.42775828  3.39505994  2.70266364 36.74766525  6.08893673  1.47425017
attr(,"label")
[1] "Orthogonal residuals from orthogonal model"
```

La fonction `plot.onls()` impose un certain nombre de paramètres graphiques, ce qui n'est pas forcément très pratique, par exemple si on veut changer la légende des axes. Voyons si nous pouvons nous pouvons faire la même chose à la main.

```
plot(yn, ynpu, pch = 19, xlab = "Glandée année n", ylab = "Glandée année n + 1",
     las = 1, main = "Les résidus de la régression\northogonale non-linéaire")
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, predict(mod2, list(yn = yy)), type = "l", col = "red")
points(mod2$x0, mod2$y0, pch = 19, cex = 0.25, col = "red")
segments(mod2$x0, mod2$y0, mod2$pred, mod2$resp, col = "red")
```



5 Données avec des zéros

Pour illustrer le problème des données comportant des valeurs nulles on force une valeur à zéro dans notre jeu de données. On utilise la fonction `try()` pour intercepter l'erreur :

```
y[2] <- 0
yn <- y[-n]   # Année n
ynpu <- y[-1] # Année n + 1
df <- list(yn = yn, ynpu = ynpu)
mod3 <- try(onls(ynpu ~ lambda/yn, data = df,
                 start = list(lambda = 10), verbose = FALSE), silent = TRUE)
class(mod3)
```

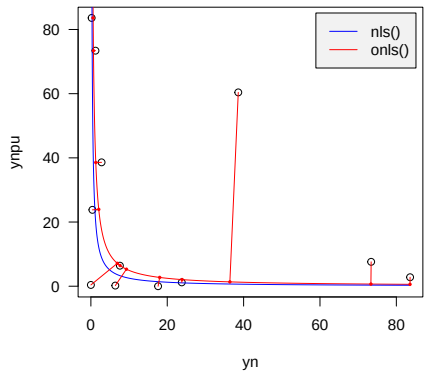
```
[1] "try-error"
```

```
attr(mod3, "condition")
```

```
<simpleError in numericDeriv(formula[[3L]], pnames, env): Valeur manquante ou infinie obtenue au cours du calcu
```

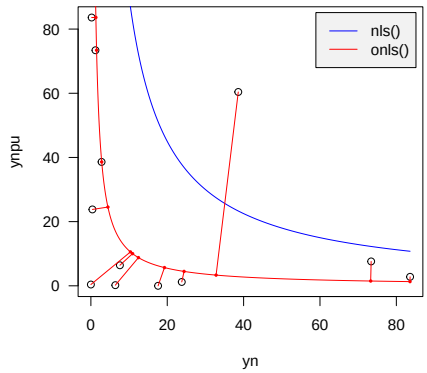
L'utilisation de `try()` permet de continuer l'exécution quand on a de nombreuses séries à ajuster. Le problème vient de ce que quand une valeur est nulle le gradient n'est plus défini. Mais il est dommage de ne pas pouvoir profiter de l'ajustement quand même. Pour ce faire j'ai écrit une fonction `newonls()` qui dérive directement de `onls()` simplement en mettant en commentaire tout ce qui a trait au calcul du gradient :

```
source("https://pbil.univ-lyon1.fr/R/donnees/onls/newonls.R")
mod4 <- try(newonls(ynpu ~ lambda^2/yn, data = df,
                 start = list(lambda = 5), verbose = FALSE),
                 silent = TRUE)
plot(mod4, las = 1)
legend("topright", inset = 0.02, legend = c("nls()", "onls()"), lty = 1,
       col = c("blue", "red"), bg = grey(0.95))
```



Notez que `onls()` neutralise en dur tous les messages d'avis avec `options(warn = -1)`, c'est pourquoi on ne voit pas ici que `nls()` a échoué, en fait il renvoie l'estimation initiale de λ :

```
mod5 <- try(newonls(ynpu ~ lambda^2/yn, data = df,
                 start = list(lambda = 30), verbose = FALSE),
                 silent = TRUE)
plot(mod5, las = 1)
legend("topright", inset = 0.02, legend = c("nls()", "onls()"), lty = 1,
       col = c("blue", "red"), bg = grey(0.95))
```



6 Signification et significativité de λ

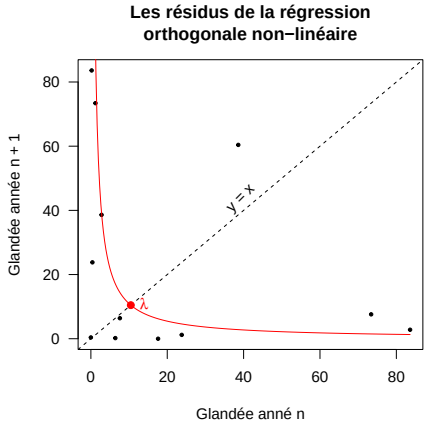
6.1 Signification de λ

Si on envisage la courbe ajustée comme un modèle déterministe en temps discret, λ est un point fixe du modèle puisque :

$$g(\lambda) = \frac{\lambda^2}{\lambda} = \lambda \tag{3}$$

Graphiquement :

```
plot(yn, ynpu, pch = 19, xlab = "Glandée anné n", ylab = "Glandée année n + 1",
     las = 1, main = "Les résidus de la régression\northogonale non-linéaire",
     cex = 0.5)
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, predict(mod5, list(yn = yy)), type = "l", col = "red")
abline(c(0, 1), lty = 2)
lest <- mod5$parONLS[[1]]
text(40, 40, "y = x", srt = 45, pos = 3)
points(lest, lest, pch = 19, col = "red")
text(lest, lest, expression(lambda), pos = 4, col = "red")
```

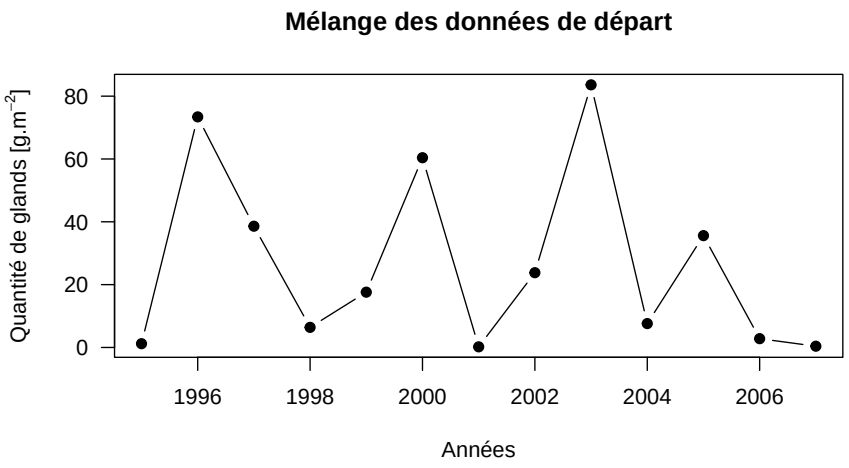


On peut voir dans λ un indice de tendance centrale peu sensible aux valeurs extrêmes. Plus le nuage de points aura la forme d'un « L », plus λ sera petit. Ce qui nous intéresse c'est donc de savoir si λ est anormalement petit.

6.2 Significativité de λ

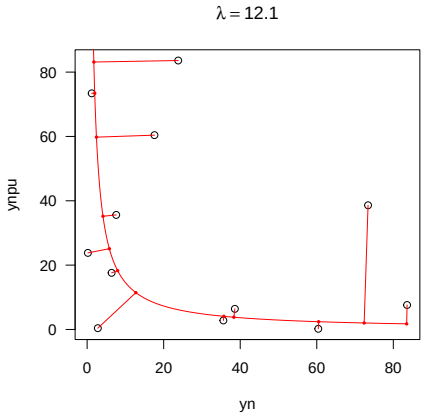
Que se passe-t-il si on mélange les données de départ avec la fonction `sample()` de façon à détruire l'auto-corrélation temporelle ?

```
set.seed(2)
x <- fds$Year ; y <- sample(fds$Value)
par(mar = c(5, 5, 4, 2) + 0.1)
plot(x, y, pch = 19, type = "b", las = 1,
     main = "Mélange des données de départ", xlab = "Années",
     ylab = expression(paste("Quantité de glands [g.", m^-2, "]")))
```



On voit que par hasard on peut avoir quelque chose qui ressemble à du *masting* à cause de la distribution bimodale des valeurs dans notre jeu de données. Estimons le paramètre λ sur nos données mélangées :

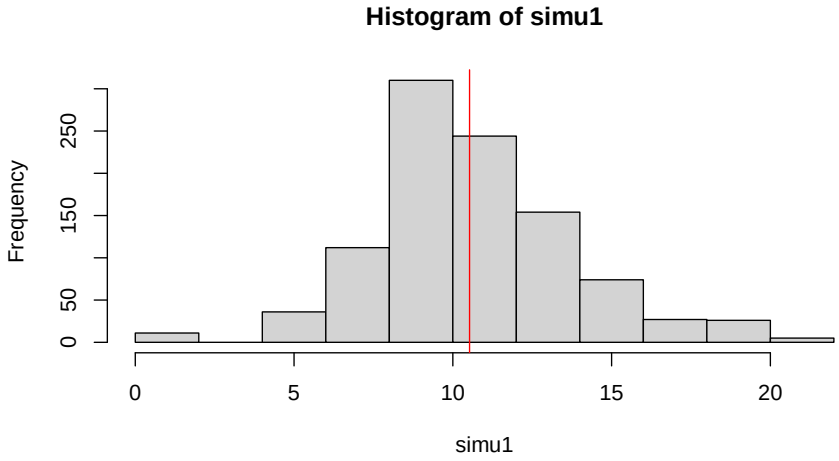
```
n <- length(y)
yn <- y[-n]   # Année n
ynpu <- y[-1] # Année n + 1
df <- list(yn = yn, ynpu = ynpu)
mod6 <- newonls(ynpu ~ lambda^2/yn, data = df,
                start = list(lambda = 30), verbose = FALSE)
lambda <- signif(mod6$parONLS[[1]], 3)
main <- bquote(lambda == .(lambda))
plot(mod6, las = 1, fitted.nls = FALSE, main = main)
```



Il est donc tout à fait possible d'obtenir par hasard une faible valeur du paramètre λ. Pour apprécier le caractère exceptionnel on non de la valeur observée de λ on va répéter cette manipulation un grand nombre de fois.

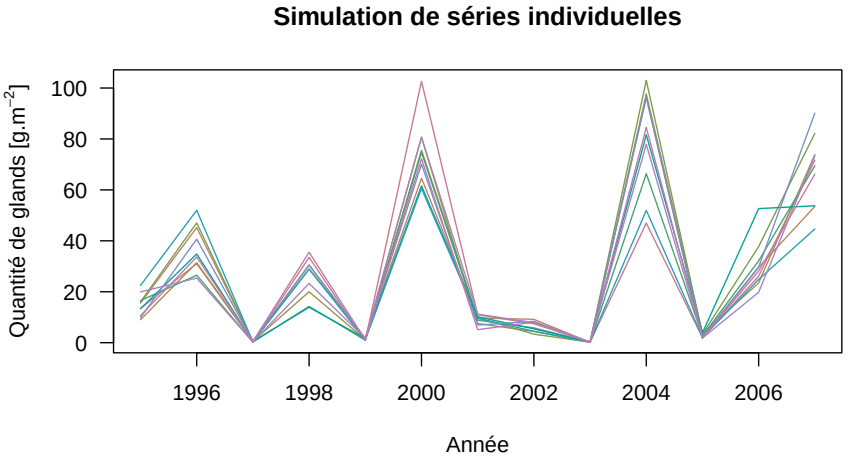
```
manip <- function(){
  y <- sample(fds$Value)
  n <- length(y)
  yn <- y[-n]   # Année n
  ynpu <- y[-1] # Année n + 1
  df <- list(yn = yn, ynpu = ynpu)
  res <- newonls(ynpu ~ lambda^2/yn, data = df,
                 start = list(lambda = 30), verbose = FALSE)
  return(res$parONLS[[1]])
}
simu1 <- replicate(999, manip())
save(simu1, file = "simu1.Rda")
```

```
load('simu1.Rda')
hist(simu1)
abline(v = mod2$parONLS[[1]], col = "red")
```



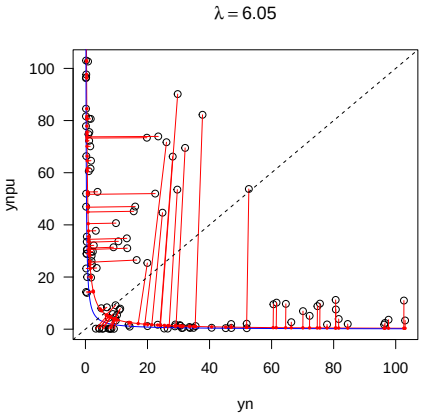
On voit ici que la valeur observée de λ n'a absolument rien d'exceptionnel. Ceci illustre la faible puissance du test quand on dispose d'une série populationnelle courte. Mais qu'en est-il si on a la chance de disposer de séries individuelles ? Simulons un jeu de données avec 10 arbres :

```
set.seed(1)
ny <- length(fds$Value) ; na <- 10
ind <- matrix(NA, nrow = na, ncol = ny)
for(i in seq_len(na)){
  ind[i , ] <- fds$Value*runif(n = ny, min = 0.5, max = 1.5)
}


par(mar = c(5, 5, 4, 2) + 0.1)
plot.new() ; plot.window(xlim = range(fds$Year), ylim = range(ind))
axis(1) ; axis(2, las = 1) ; box()
title(main = "Simulation de séries individuelles")
title(xlab = "Année")
title(ylab = expression(paste("Quantité de glands [g.", m^-2, "]")))
col <- hcl.colors(na, "Dark 2")
for(i in seq_len(na)){
  points(fds$Year, ind[i ,], type = "l", col = col[i])
}
```



POUR pouvoir ajuster le modèle j'ai modifié les paramètres de `onls()`. J'ai utilisé `extend = c(0, 0)` pour restreindre l'observation strictement au domaine des valeurs observées, sans quoi on a des valeurs négatives qui font projeter certains points sur la branche négative de l'hyperbole. J'ai exploré les valeurs possibles du paramètre `window` entre 12 et 60, ce n'est qu'avec une valeur de 60 que j'avais tous les points orthogonaux.

```
n <- ncol(ind)
yn <- ind[ , -n] ; dim(yn) <- NULL     # Année n
ynpu <- ind[ , -1] ; dim(ynpu) <- NULL # Année n + 1
df <- list(yn = yn, ynpu = ynpu)
modi <- newonls(ynpu ~ lambda^2/yn, data = df,
                start = list(lambda = 30), extend = c(0, 0),
                window = 60, verbose = FALSE)
lambda <- signif(modi$parONLS[[1]], 3)
main <- bquote(lambda == .(lambda))
plot(modi, las = 1, fitted.nls = TRUE, main = main)
abline(c(0,1), lty = 2)
```



On voit ici une limitation de `onls()` : le critère d'orthogonalité est bien respecté, mais la somme des carrés des résidus n'est pas minimale puisque certains points sont projetés verticalement alors qu'il aurait été plus optimal de les projeter horizontalement. Il va falloir ruser un peu.

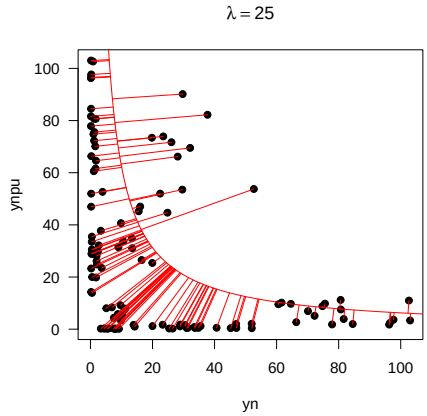
7 Rusons un peu

Pour un λ donné et un point de coordonnées (x_i, y_i) on définit une fonction `d2()` qui calcule le carré de la distance entre le point et la courbe à l'abscisse x (et donc à l'ordonnée $\frac{\lambda^2}{x}$).

```
d2 <- function(x, lambda, xi, yi){
  return((x - xi)^2 + (lambda^2/x - yi)^2)
}
```

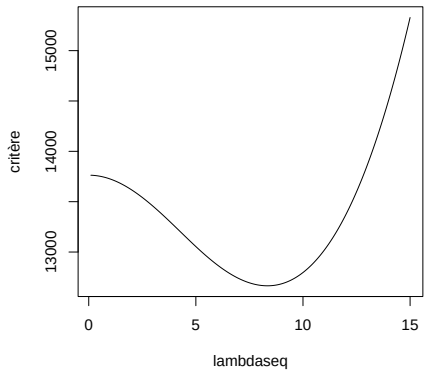
On définit maintenant la fonction `d2min()` qui retourne la valeur de x qui minimise la distance à la courbe. Pour $x_i \neq y_i$ il y a deux minimums locaux selon que l'on projette verticalement ou horizontalement. L'astuce va consister à utiliser une condition initiale pour x différente selon que le point (x_i, y_i) est au-dessus ou au-dessous de la première bissectrice.

```
d2min <- function(lambda, xi, yi){
  if(yi > xi){ # au dessus de la première bissectrice
    xstart <- lambda^2/yi
  } else {
    xstart <- xi
  }
  return(nlm(d2, xstart, lambda = lambda, xi = xi, yi = yi)$estimate)
}
# Vérifions que cela nous convienne
lambda <- 25
main <- bquote(lambda == .(lambda))
plot(yn, ynpu, pch = 19, las = 1, main = main)
xseq <- seq(0.1, 110, le = 256)
points(xseq, lambda^2/xseq, type = "l", col = "red")
for(i in seq_len(length(yn))){
  x <- d2min(lambda, yn[i], ynpu[i])
  segments(x, lambda^2/x, yn[i], ynpu[i], col = "red")
}
```



C'est bien la projection des points sur la courbe que nous voulons. Il ne nous reste plus qu'à trouver la valeur de λ qui minimise la somme des carrés des distances. On définit une fonction `sce()` qui nous retourne cette valeur pour étudier son comportement.

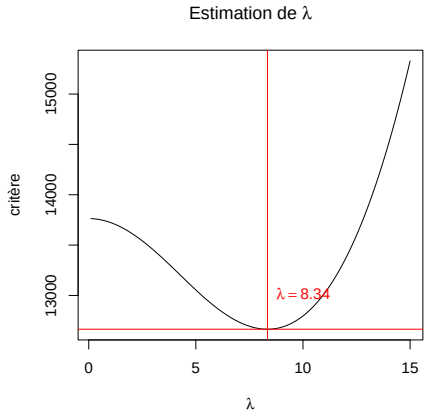
```
sce <- function(lambda, x, y){
  somme <- 0
  for(i in seq_len(length(x))){
    xopt <- d2min(lambda, x[i], y[i])
    somme <- somme + d2(xopt, lambda, x[i], y[i])
  }
  return(somme)
}
lambdaseq <- seq(0.1, 15, le = 256)
critère <- numeric(length(lambdaseq))
for(i in seq_len(length(lambdaseq))){
  critère[i] <- sce(lambdaseq[i], yn, ynpu)
}
plot(lambdaseq, critère, type = "l")
```



Le comportement semble sympathique sans minimum local, ni mesa, sur ce cas particulier. On voit que l'on a un optimum pour $\lambda \approx 8$. On définit la fonction `hellfit()` pour trouver la valeur de λ qui minimise la somme des carrés des écarts.

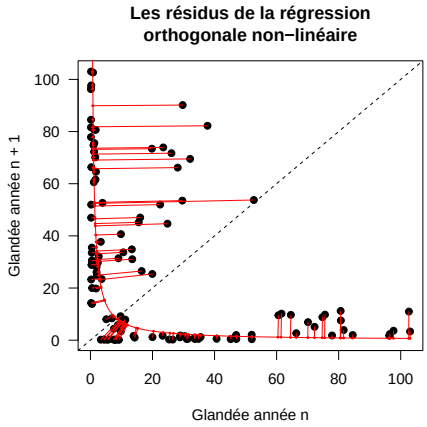
```
hellfit <- function(x, y){
  lambdastart <- max(x)/2
```

```
  n <- length(x)
  estimate <- nlm(sce, lambdastart, x = x, y = y)$estimate
  estimate <- abs(estimate)
  scemin <- sce(estimate, x, y)
  x0 <- numeric(n)
  for(i in seq_len(n)) x0[i] <- d2min(estimate, x[i], y[i])
  y0 <- estimate^2/x0
  resid_o <- numeric(n)
  for(i in seq_len(n)) resid_o[i] <- sqrt((x[i] - x0[i])^2 + (y[i] - y0[i])^2)
  return(list(estimate = estimate, scemin = scemin, resid_o = resid_o,
              x0 = x0, y0 = y0))
}
sol <- hellfit(yn, ynpu)
main <- expression(paste("Estimation de ", lambda))
plot(lambdaseq, critère, type = "l", main = main,
     xlab = expression(lambda))
abline(v = sol$estimate, col = "red")
abline(h = sol$scemin, col = "red")
lambda <- signif(sol$estimate, 3)
text(sol$estimate, 13000, bquote(lambda == .(lambda)), col = "red", pos = 4)
```



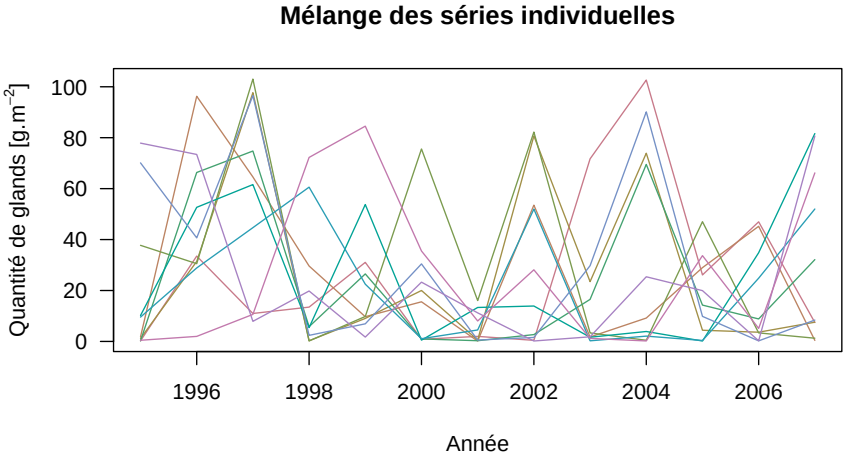
Avec la fonction `onls()` nous avions une somme des carrés des résidus orthogonaux de 35722 alors qu'avec la fonction `hellfit()` nous tombons à 12665, nous avons donc considérablement amélioré les choses. Voyons l'ajustement.

```
plot(yn, ynpu, pch = 19, xlab = "Glandée année n", ylab = "Glandée année n + 1",
     las = 1, main = "Les résidus de la régression\northogonale non-linéaire")
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, sol$estimate^2/yy, type = "l", col = "red")
points(sol$x0, sol$y0, pch = 19, cex = 0.25, col = "red")
segments(sol$x0, sol$y0, yn, ynpu, col = "red")
abline(c(0, 1), lty = 2)
```



C'est bien plus satisfaisant que ce que nous avions obtenu avec `onls()`. Revenons maintenant à notre question de la significativité de λ. On commence par mélanger les données pour chaque arbre.

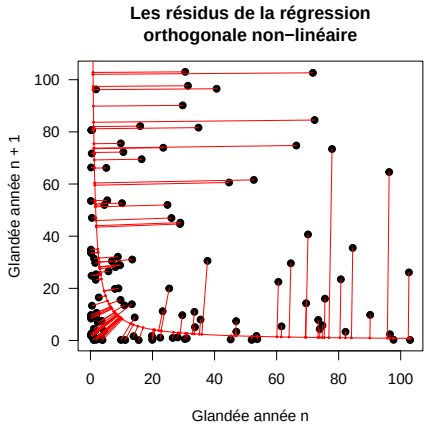
```
melind <- ind
set.seed(1)
for(i in seq_len(na)) melind[i, ] <- sample(melind[i, ])
par(mar = c(5, 5, 4, 2) + 0.1)
plot.new() ; plot.window(xlim = range(fds$Year), ylim = range(melind))
axis(1) ; axis(2, las = 1) ; box()
title(main = "Mélange des séries individuelles")
title(xlab = "Année")
title(ylab = expression(paste("Quantité de glands [g.", m^-2, "]")))
col <- hcl.colors(na, "Dark 2")
for(i in seq_len(na)){
  points(fds$Year, melind[i ,], type = "l", col = col[i])
}
```



Cette fois on sent que l'on a bien cassé la structure d'auto-corrélation temporelle, voyons l'ajustement au modèle.

```
n <- ncol(melind)
yn <- melind[ , -n] ; dim(yn) <- NULL     # Année n
ynpu <- melind[ , -1] ; dim(ynpu) <- NULL # Année n + 1
solmel <- hellfit(yn, ynpu)
plot(yn, ynpu, pch = 19, xlab = "Glandée année n", ylab = "Glandée année n + 1",
```

```
     las = 1, main = "Les résidus de la régression\northogonale non-linéaire")
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, solmel$estimate^2/yy, type = "l", col = "red")
points(solmel$x0, solmel$y0, pch = 19, cex = 0.25, col = "red")
segments(solmel$x0, solmel$y0, yn, ynpu, col = "red")
```



Ce n'est pas très bon tout ça, finalement avec la régression orthogonale les points extrêmes n'arrivent pas à faire bouger la courbe puisqu'ils sont peu sensibles aux variations de λ. Ce n'est pas la courbe que j'aurais tracée à la main pour résumer le nuage de points.

8 Régression avec des résidus selon $y = x$

On cherche la courbe qui minimise la somme des carrés des résidus parallèles à la première bissectrice. C'est justifié dans notre cas puisque les variables en x et en y sont à la même échelle, c'est en fait la même variable décalée d'un an. Pour une utilisation plus générale il faudrait commencer par mettre à la même échelle les valeurs en abscisse et en ordonnée. La fonction `d2()` est inchangée et pour `d2min45()` on a une solution analytique, ce qui va accélérer les calculs puisque l'on n'a plus de boucle interne d'optimisation.

```
d2min45 <- function(lambda, xi, yi){
  delta <- (xi - yi)^2 + 4*lambda^2
  return(0.5*(xi - yi + sqrt(delta)))
}
```

La fonction qui calcule la somme des carrés des écarts est inchangée sauf qu'elle fait appel à `d2min45()` au lieu de `d2min()` pour projeter un point sur la courbe.

```
sce45 <- function(lambda, x, y){
  somme <- 0
  for(i in seq_len(length(x))){
    xopt <- d2min45(lambda, x[i], y[i])
    somme <- somme + d2(xopt, lambda, x[i], y[i])
  }
  return(somme)
}
```

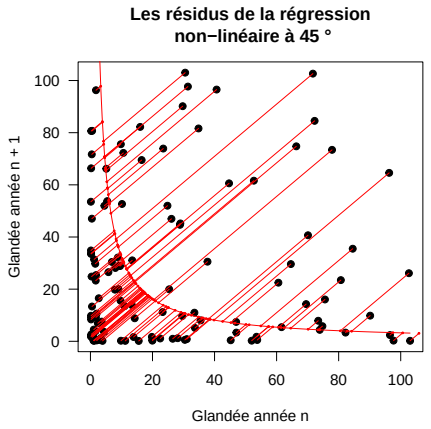
La fonction `hellfit45()` prend comme argument les coordonnées x et y du nuage de points(`x` et `y`). Elle retourne la valeur estimée de λ (`estimate`),

la valeur minimale de la somme des carrés des écarts (`scemin`), les résidus à 45ř (`resid_45`) c'est à dire la distance à la courbe, les coordonnées de la projection des points sur la courbe (`x0` et `y0`). Si on a besoin des résidus signés il suffit de multiplier par `sign(x - x0)`.

```
hellfit45 <- function(x, y){
  lambdastart <- max(x)/2
  n <- length(x)
  estimate <- nlm(sce45, lambdastart, x = x, y = y)$estimate
  estimate <- abs(estimate)
  scemin <- sce45(estimate, x, y)
  x0 <- numeric(n)
  for(i in seq_len(n)) x0[i] <- d2min45(estimate, x[i], y[i])
  y0 <- estimate^2/x0
  resid_45 <- numeric(n)
  for(i in seq_len(n)) resid_45[i] <- sqrt((x[i] - x0[i])^2 + (y[i] - y0[i])^2)
  return(list(estimate = estimate, scemin = scemin, resid_45 = resid_45,
              x0 = x0, y0 = y0))
}
```

On peut maintenant représenter l'ajustement de la courbe avec notre jeu de données individuelles simulées et mélangées.

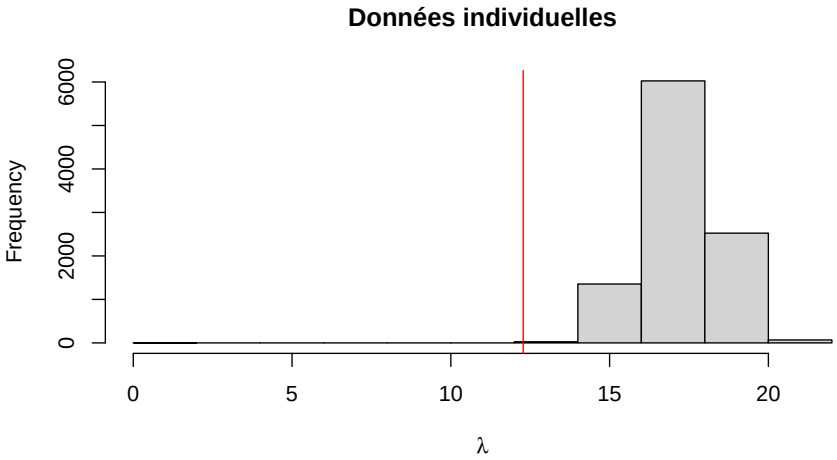
```
solmel45 <- hellfit45(yn, ynpu)
plot(yn, ynpu, pch = 19, xlab = "Glandée année n", ylab = "Glandée année n + 1",
     las = 1, main = "Les résidus de la régression\n non-linéaire à 45 ř")
yy <- seq(min(yn), max(yn), length.out = 256)
points(yy, solmel45$estimate^2/yy, type = "l", col = "red")
points(solmel45$x0, solmel45$y0, pch = 19, cex = 0.25, col = "red")
segments(solmel45$x0, solmel45$y0, yn, ynpu, col = "red")
```



Le test de permutation donne maintenant une valeur anormalement faible de la statistique observée, on gagne en puissance avec des données individuelles.

```
manip45 <- function(){
  melind <- ind
  for(i in seq_len(na)) melind[i, ] <- sample(melind[i, ])
  n <- ncol(melind)
  yn <- melind[ , -n] ; dim(yn) <- NULL     # Année n
  ynpu <- melind[ , -1] ; dim(ynpu) <- NULL # Année n + 1
  return(hellfit45(yn, ynpu)$estimate)
}
simu45 <- replicate(9999, manip45())
save(simu45, file = "simu45.Rda")
```

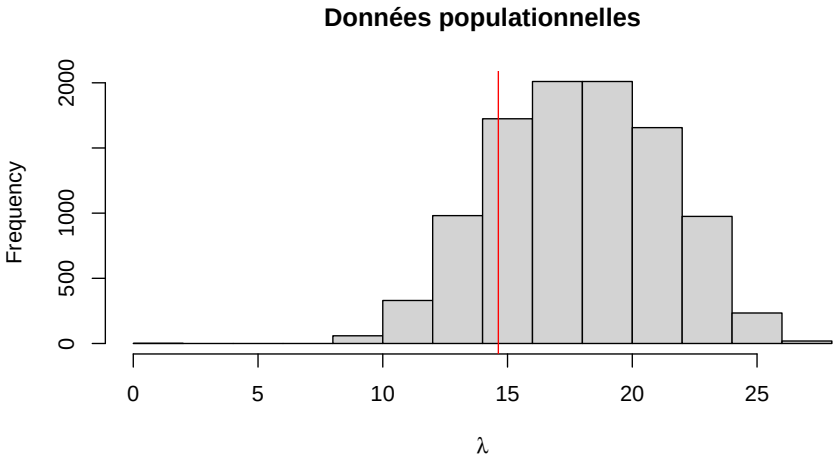
```
load("simu45.Rda")
hist(simu45, xlab = expression(lambda), main = "Données individuelles")
n <- ncol(ind)
yn <- ind[ , -n] ; dim(yn) <- NULL     # Année n
ynpu <- ind[ , -1] ; dim(ynpu) <- NULL # Année n + 1
sol45 <- hellfit45(yn, ynpu)
abline(v = sol45$estimate, col = "red")
```



Pour avoir un élément de comparaison on fait la même manipulation avec des données populationnelles.

```
manip45p <- function(){
  y <- fds$Value ; n <- length(y)
  y <- sample(y)
  yn <- y[-n]   # Année n
  ynpu <- y[-1] # Année n + 1
  return(hellfit45(yn, ynpu)$estimate)
}
simu45p <- replicate(9999, manip45p())
save(simu45p, file = "simu45p.Rda")
```

```
load("simu45p.Rda")
hist(simu45p, xlab = expression(lambda), main = "Données populationnelles")
y <- fds$Value ; n <- length(y)
yn <- y[-n]   # Année n
ynpu <- y[-1] # Année n + 1
sol45p <- hellfit45(yn, ynpu)
abline(v = sol45p$estimate, col = "red")
```



On perd donc de la puissance par rapport au résultat avec des données individuelles puisqu'ici on ne peut pas affirmer que la valeur de λ est anormalement faible. On a cependant amélioré les choses par rapport à quand on travaillait avec des résidus orthogonaux (voir section 6.2).

Références

[1] T.V. Elzhov, K.M. Mullen, A.-N. Spiess, and B. Bolker. *minpack.lm : R Interface to the Levenberg-Marquardt Nonlinear Least-Squares Algorithm Found in MINPACK, Plus Support for Bounds*, 2022. R package version 1.2-2.

[2] A. Hacket-Pain, J.J. Foest, I.S. Pearse, J.M. LaMontagne, W.D. Koenig, G. Vacchiano, M. Bogdziewicz, T. Caignard, P. Celebias, J. van Dormolen, M. Fernández-Martínez, J.V. Moris, C. Palaghianu, M. Pesendorfer, A. Satake, E. Schermer, A.J. Tanentzap, P.A. Thomas, D. Vecchio, A.P. Wion, T. Wohlgemuth, T. Xue, K. Abernethy, M.-C. Aravena A., M.D. Barrera, J.H. Barton, S. Boutin, E.R. Bush, S.D. Calderón, F.S. Carevic, C.V. de Castilho, J.M. Cellini, C.A. Chapman, H. Chapman, F. Chianucci, P. da Costa, L. Croisé, A. Cutini, B. Dantzer, R.J. DeRose, J.-T. Dikangadissi, E. Dimoto, F.L. da Fonseca, L. Gallo, G. Gratzer, D.F. Greene, M.A. Hadad, A.H. Herrera, K.J. Jeffery, J.F. Johnstone, U. Kalbitzer, W. Kantorowicz, C.A. Klimas, J.G.A. Lageard, J. Lane, K. Lapin, M. Ledwo, A.C. Leeper, M.V. Lencinas, A.C. Lira-Guedes, M.C. Lordon, P. Marchelli, S. Marino, H. Schmidt Van Marle, A.G. McAdam, L.R. . Momont, M. Nicolas, L.H. de Oliveira Wadt, P. Panahi, G. Martínez Pastur, T. Patterson, P. Luis Peri, Ł. Piechnik, M. Pourhashemi, C. Espinoza Quezada, F.A. Roig, K. Peña Rojas, Y. Micaela Rosas, S. Schueler, B. Seget, R. Soler, M.A. Steele, M. Toro-Manríquez, C.E.G. Tutin, T. Ukizintambara, L. White, B. Yadok, J.L. Willis, A. Zolles, M. Żywiec, and D. Ascoli. Mastree+ : time-series of plant reproductive effort from six continents. *Global Change Biology*, 00 :1–17, 2022.

[3] A.-N. Spiess. *onls : Orthogonal Nonlinear Least-Squares Regression*, 2015. R package version 0.1-1.