Estimateur de Chao généralisé

Contents

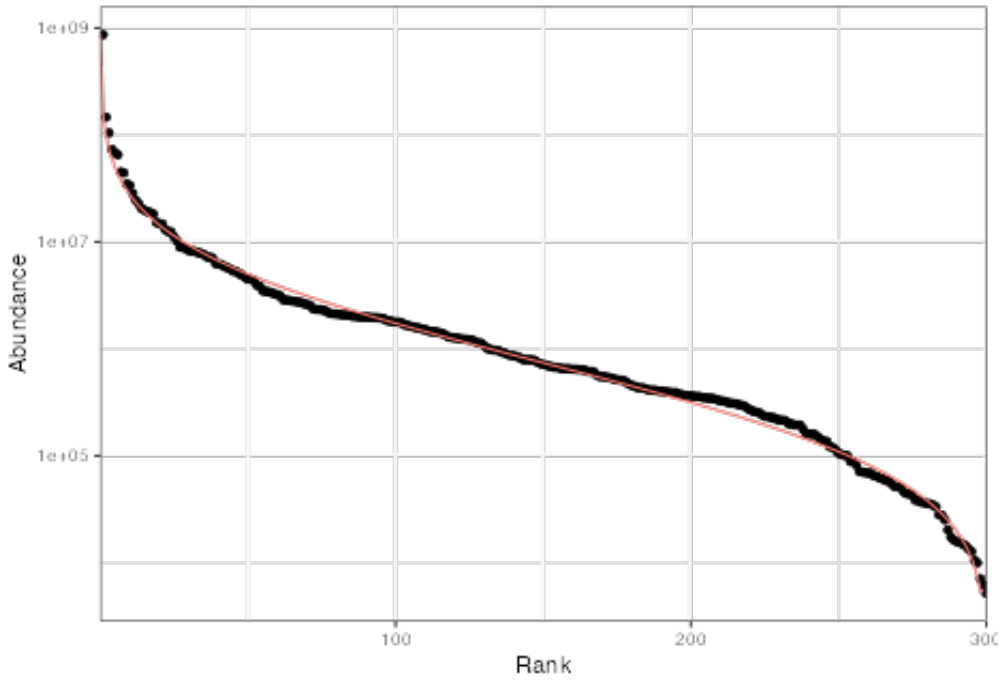
Simulation d'une communauté

Une communauté théorique correspondant à une distribution connue est simulée. Pour éviter les problèmes de sous-échantillonage, la communauté est de la taille maximale supportée par R.

```
library("entropart")
richness <- 300
Community <- rCommunity(
  1, # Nombre de communautés simulées
  size = .Machine$integer.max, # Taille
  S = richness, # Nb d'espèces
  Distribution = "lnorm", # Log-normale
  sd = 2 # Ecart-type d'une forêt tropicale
)
```

La courbe rang-abondance permet de visualiser la communauté.

```
autoplot(Community, Distribution = "lnorm")
```



Toutes les espèces doivent avoir un effectif très supérieur à 2 (sinon, la communauté serait visiblement un échantillon partiel d'une communauté plus grande).

```
min(Community)
```

```
## [1] 5170
```

Autres données

Pas utilisées ici. Dans ces inventaires forestiers, tous les arbres de plus de 10 cm de diamètre à hauteur de poitrine (1.3 m) sont inventoriés.

BCI

50 ha de forêt tropicale à Barro Colorado Island, Panama. Aide avec `?BCI`.

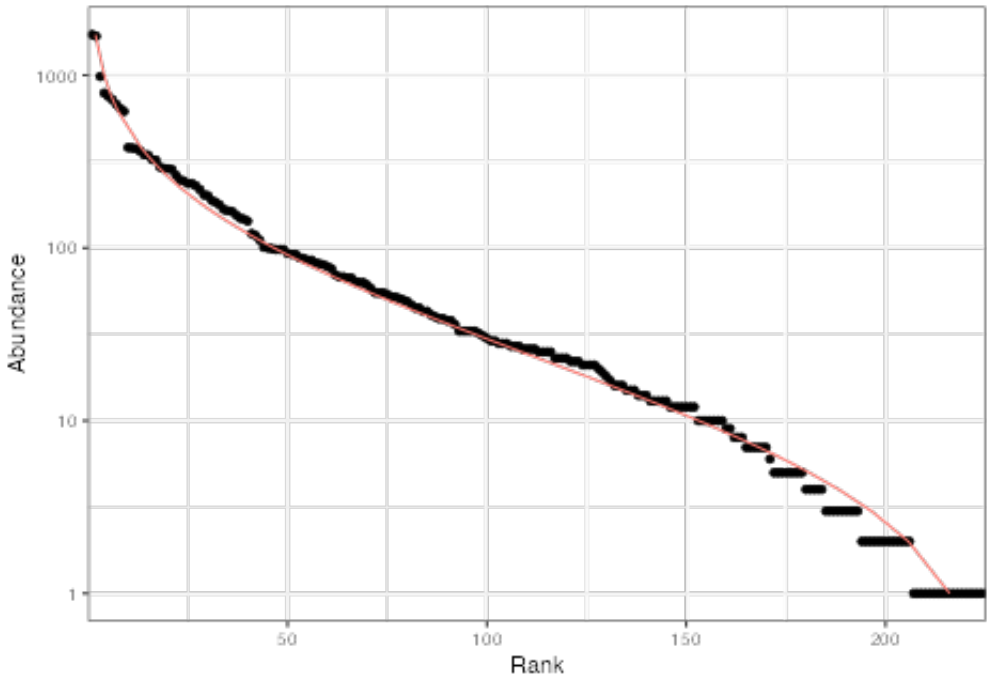
```
library("vegan")
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.5-7
```

```
data(BCI)
# Somme des 50 inventaires d'un hectare
N_BCI <- as.AbdVector(colSums(BCI))
autoplot(N_BCI, Distribution = "lnorm")
```



Presque toutes les espèces sont inventoriées.

```
# Nombre d'espèces observées
Richness(N_BCI, Correction = "None")
```

```
## None
##  225
```

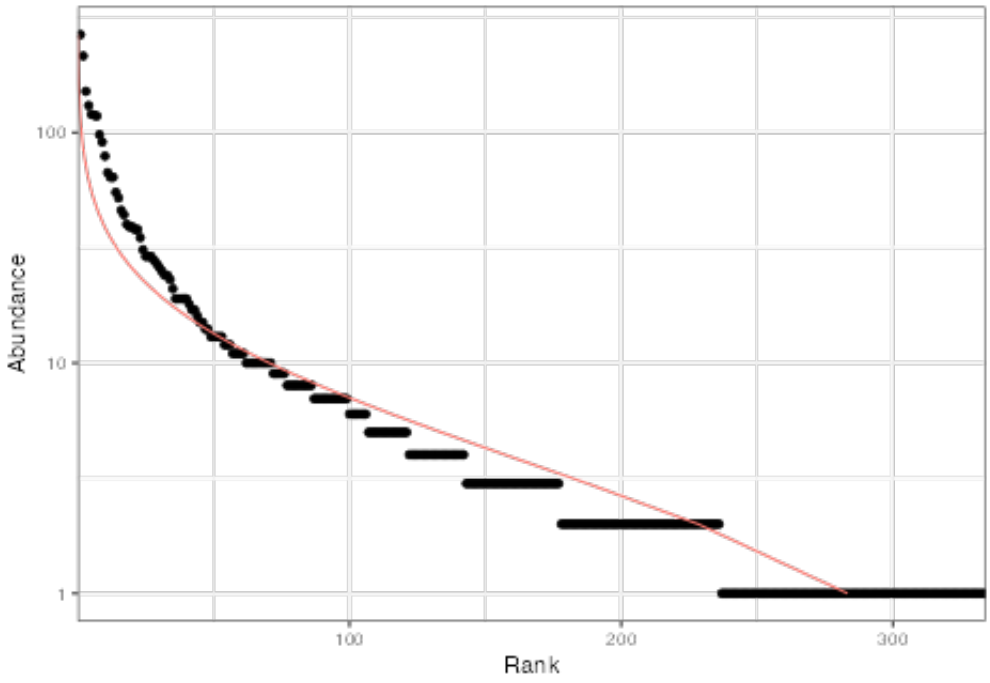
```
# Estimateur de Chao
Richness(N_BCI, Correction = "Chao1")
```

```
##   Chao1
## 238.884
```

Paracou

Parcelle 6 de Paracou : 6.25 ha, inventaire de 2016. Les arbres sont localisés.

```
load(file = "data/Paracou6.rda")
# Comptage des individus par espèce
N_Paracou <- as.AbdVector(tapply(Paracou6$x, Paracou6$marks$PointType,
    length))
autoplot(N_Paracou, Distribution = "lnorm")
```



L'ajustement du modèle log-normal n'est pas bon parce que les données sont un échantillon de la communauté dont de nombreuses espèces manquent. Meilleure estimation de la richesse avec le Jackknife d'ordre 2.

```
Richness(N_Paracou)
```

```
## Jackknife 2
##         471
```

L'échantillonnage est trop insuffisant pour que l'estimateur de Chao soit utilisable.

```
Richness(N_Paracou, Correction = "Chao1")
```

```
##    Chao1
## 415.3668
```

Exploration

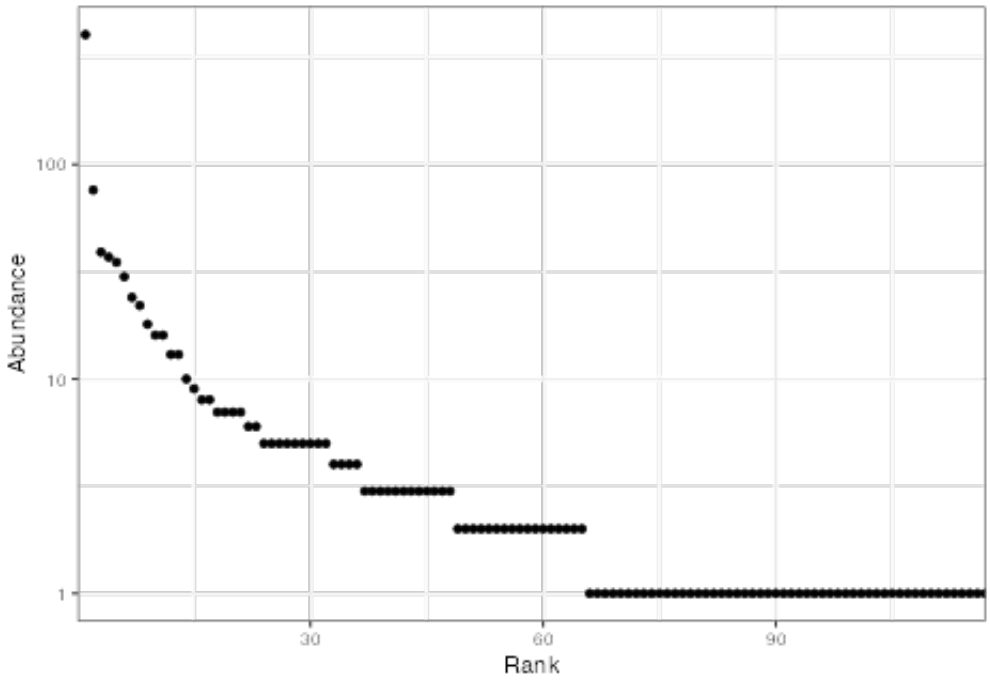
Tirage d'un échantillon

Un inventaire de taille choisie est simulé par un tirage dans une loi multinomiale.

```
# Taille de l'échantillon
size <- 1000
Sample <- rmultinom(1, size, Community)
```

L'échantillon obtenu contient beaucoup moins d'espèces et beaucoup de singletons et doubletons.

```
autoplot(as.AbdVector(Sample))
```



Application de l'estimateur

Estimation du nombre d'espèces non observées.

```
GenChaoS0 <- function(f, k, kprime) {
    # Abondances représentées
    abd <- f[, 1]
    # Abondance totale
    n <- sum(f[, 1] * f[, 2])
    # Estimateur des espèces non observées en log pour
    # éviter les dépassements de capacité
    log_S0 <- k/(k - kprime) * (log(f[which(abd ==
        kprime), 2]) - lchoose(n, kprime)) + kprime/(k -
        kprime) * (lchoose(n, k) - log(f[which(abd ==
        k), 2]))
    # Résultat
    return(exp(log_S0))
}
```

Estimation pour $(k; k') = (2; 1)$.

```
# Fréquence des abondances : tableau des f.
f <- AbdFreqCount(Sample)
# Estimation de Chao1
GenChaoS0(f, k = 2, kprime = 1) + sum(f[, 2])
```

```
## NbSpecies
##  196.4499
```

```
# Vérification (fonction Richness du package
# entropart)
Richness(Sample, Correction = "Chao1")
```

```
##    Chao1
## 196.4499
```

Test de toutes les combinaisons de k et k'

On fixe $k > k'$ sans perte de généralité. Toutes les paires de k et k' sont créées.

```
# Toutes les paires k, k'
kkprime <- expand.grid(f[, 1], f[, 1])
# Seulement celles telles que k > k'
kkprime <- kkprime[kkprime[, 1] > kkprime[, 2], ]
colnames(kkprime) <- c("k", "kprime")
head(kkprime)
```

```
##   k kprime
## 2 2      1
## 3 3      1
## 4 4      1
## 5 5      1
## 6 6      1
## 7 7      1
```

L'estimateur est appliqué avec toutes les paires. La plus grande valeur est affichée.

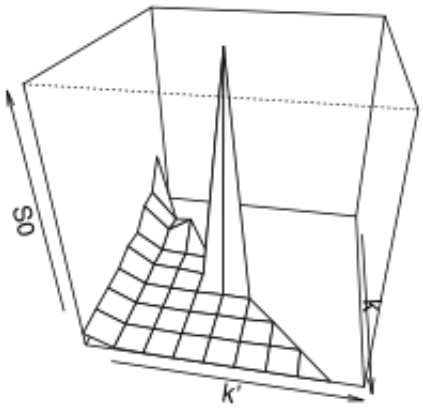
```
# Estimation pour toutes les paires Ajout d'une
# colonne pour recevoir les estimations
kkprime$S0 <- 0
for (i in 1:nrow(kkprime)) {
    kkprime$S0[i] <- GenChaoS0(f, k = kkprime[i, 1],
        kprime = kkprime[i, 2])
}
# Valeur max
maxestkkprime <- kkprime[which.max((kkprime$S0)), ]
# Tableau bien formaté
library(kableExtra)
maxestkkprime %>% as_tibble() %>% kbl()
```

k	kprime	S0
6	5	252.4614

```
# %>% kable_styling()
```

L'estimateur est représenté en fonction de k et k'.

```
# Graphique
k_values <- 10  # nrow(f)
S0_3Dplot <- matrix(nrow = k_values, ncol = k_values)
for (i in 1:k_values) {
    for (j in 1:k_values) {
        found <- (kkprime[, 1] == f[i, 1]) & (kkprime[,
            2] == f[j, 1])
        if (any(found)) {
            S0_3Dplot[i, j] <- kkprime[found, 3]
        } else {
            S0_3Dplot[i, j] <- NA
        }
    }
}
persp(f[1:k_values, 1], f[1:k_values, 1], S0_3Dplot,
    theta = 100, phi = 30, xlab = "k", ylab = "k'",
    zlab = "S0")
```



Manifestement, le biais augmente systématiquement pour toutes les valeurs de $(k; k')$ différentes de celles de l'estimateur de Chao original. De temps à autres, une valeur aberrante apparaît.

Evaluation d'un estimateur

La technique dite de Monte-Carlo consiste à simuler un grand nombre d'échantillons à d'une communauté connue et leur appliquer l'estimateur. Toutes les valeurs estimées fournissent une estimation de la distribution de probabilité des valeurs de l'estimateur. Leur moyenne empirique estime l'espérance de l'estimateur, et la variance empirique de même.

L'estimation pose deux problèmes :

- le biais : la différence entre la valeur réelle et la valeur estimée doit être aussi faible que possible;
- la variance de l'estimateur : l'estimation doit être aussi peu variable que possible.

On montre que l'espérance de l'erreur d'estimation (plus précisément, la racine carré de l'espérance de son carré : RMSE) est la racine carrée de la somme du carré du biais et de la variance. Cette quantité est souvent normalisée par la valeur réelle pour obtenir une erreur relative.

La version complète de la fonction prend comme arguments le vecteur des abondances et les valeurs de k et k". Elle retourne l'estimation du nombre d'espèces (y compris les espèces observées).

```
GenChao <- function(N, k, kprime) {
    # Fréquence des abondances : tableau des f.
    f <- AbdFreqCount(N)
    # Abondances représentées
    abd <- f[, 1]
    # Vérification de l'existence de f_k et f_kprime
    if ((!k %in% abd) | (!kprime %in% abd))
        return(NA)
    # Abondance totale
    n <- sum(N)
    # Estimateur des espèces non observées en log pour
    # éviter les dépassements de capacité
```

```
    log_S0 <- k/(k - kprime) * (log(f[which(abd ==
        kprime), 2]) - lchoose(n, kprime)) + kprime/(k -
        kprime) * (lchoose(n, k) - log(f[which(abd ==
        k), 2]))
    # Résultat, y compris le nombre d'espèces observées
    return(exp(log_S0) + as.numeric(sum(f[, 2])))
}
```

En pratique, il faut simuler un grand nombre de communautés.

```
nSimulations <- 1000
Samples <- rmultinom(nSimulations, size, Community)
# Choix de k et k'
k <- maxestkkprime[1, 1]
kprime <- maxestkkprime[1, 2]
estimates <- apply(Samples, 2, GenChao, k = k, kprime = kprime)
# Biais
(bias <- abs(richness - mean(estimates, na.rm = TRUE)))
```

```
## [1] 11.9693
```

```
variance <- var(estimates, na.rm = TRUE)
(RRMSE <- sqrt(bias^2 + variance))/richness
```

```
## [1] 5.256327
```

```
# identique à
sqrt(mean((richness - estimates)^2, na.rm = TRUE))/richness
```

```
## [1] 5.253597
```

La distribution de l'estimateur peut être représentée graphiquement. Les pointillés correspondent à 95% des estimations.

```
estimates_noNA <- estimates[!is.na(estimates)]
autoplot(as.SimTest(richness, estimates_noNA), main = "Distribution de l'estimateur") +
    xlim(min = min(estimates_noNA), max = max(1.1 *
        richness, min(2 * richness, max(estimates_noNA))))
```