Fiche TD avec le logiciel R : StatsMasting

Quelques propriétés des statistiques décrivant le phénomène dit de masting

Pr Jean R. Lobry

On s'intéresse ici à quelques statistiques, tant au niveau individuel que populationnel, susceptibles d'éclairer le phénomène dit de masting. Elles sont illustrées via un jeu de données individuelles de 25 chênes pédonculés suivis pendant 12 ans et par 1433 séries temporelles quantitatives de plus de 12 ans au niveau populationnel issues de la base de données MASTREE+. La mise en œuvre du calcul de ces statistiques est détaillée, en expliquant comment définir des fonctions qui puissent à la fois s'utiliser directement ou comme arguments des fonctions de ré-échantillonnage intensif (bootstrap) tout en gérant correctement les valeurs manquantes. La distribution bootstrap est utilisée quand c'est possible pour construire des intervalles de confiance et détecter les cas où les séries temporelles sont peu informatives. On trouvera également des indications sur la façon de lancer les calculs en arrière-plan et de paralléliser par les données.

Contents

1 Introduction **3**
1.1 Le masting : définition 3
1.2 Le masting : illustration avec données individuelles 4
1.3 Le masting : illustration avec données populationnelles 10
1.4 Fonctions utilitaires 10

2 Notation des statistiques **12**
2.1 Valeurs des séries temporelles 12
2.2 Statistiques individuelles (par arbre) 12
2.3 Statistiques individuelles moyennes (par site) 13
2.4 Statistiques populationnelles (par site) 13

3 Statistiques de la variabilité inter-annuelle **14**
3.1 CV : coefficient de variation 14
3.2 PV : variabilité proportionnelle 29
3.3 CVp *vs.* PVp 42
3.4 SDL : écart type des logs 44
3.5 SDR : écart-type des rangs 46

4 Statistiques de synchronisation **49**
4.1 Statistiques de corrélation 49
4.2 Synchronistaion des arbres pour les glandées 53
4.3 Synchronisation fine des arbres 55

5 Statistiques d'auto-corrélation temporelle **59**
5.1 Lien avec le *masting* 59
5.2 Auto-corrélation avec le r de PEARSON 61
5.3 Auto-corrélation avec le ρ de SPEARMAN 65
5.4 Auto-corrélation avec le τ de KENDALL 69
5.5 Désordre consécutif D 73
5.6 PVD 74
5.7 Hell statistic 78

6 Remerciements **83**

Références **83**

Figure 1: Exemple de glandée observée le 12 octobre 2021 au niveau d'un chêne pédonculé (*Quercus robur* L., 1753) au bord du lac de la Pounte à Labastide-Villefranche (64270) dans les Pyrénées en France. Notez que le sol est littéralement jonché de glands. Pour un stade plus avancé voir la figure 2 page 5. Crédit photo : Thomas Caignard.

1 Introduction

1.1 Le masting : définition

Le terme de *mast seeding*, contracté par la suite en *masting*, a été introduit [27] en 1976 par Daniel H. Janzen pour caractériser le comportement de certaines espèces de bambous qui ne fleurissent que très rarement[1] mais en faisant explicitement référence aux chênes et aux hêtres[2]. Le *masting* est la production épisodique et synchrone de grandes quatités de graines par des populations de plantes. Comme le souligne Walter D. Koenig [32] dans sa « brève histoire du *masting* » cette définition soulève immédiatement plusieurs questions[3] :

Populations : de quelle taille sont-elles ? Les trois plants de tomate qui poussent en pot sur mon balcon sont-ils éligibles ? Pour les 5973 séries

[1] Par exemple *Phyllostachys bambusoides* environ tous les 120 ans.

[2] *Mast seeding* is the synchronized production of seed at long intervals by a population of plants. The term derives from oak mast, beech mast, etc, as traditionally used to describe the large amount of acorns, beech seeds, etc on the ground beneath midlatitude forests in a mast year (cf [25, 26]).

[3] How synchronized? How long and regular an interval between mast events? How large a population? And how should a « large amount » of seed be defined? [22, 28].

temporelles compilées dans la base données MASTREE+[4] [17], les surfaces des zones étudiées vont de moins de 10^2 à plus de 10^6 hectares. Notons simplement que l'unité de base est l'hectare, soit la surface d'un terrain de football, et que mon balcon n'est pas éligible.

Synchronisme : à quel point les individus de la population doivent-ils être synchrones ?

Grande quantité : à partir de quel seuil décide-t-on qu'il y a une glandée[5] (cf. figure 1 page 3), c'est à dire une année présentant un pic de production de glands ?

Épisodisme : quelle durée et quelle irrégularité doit-il y avoir entre les pics de production ? Une production cyclique, facilement anticipable par les prédateurs, est-elle recevable, ou bien requiert-on un comportement plus chaotique ?

Les statistiques du *masting* visent donc, à partir de séries temporelles de suivi de production de graines par des plantes, à quantifier ces différents aspects. On trouvera souvent dans la littérature relative au *masting* le terme impropre de « *masting metric* » qui porte à confusion parce qu'au moins une des statistiques du *masting* (cf. section 3.2.1 page 29), est basée sur la définition d'une distance, c'est à dire une métrique au sens propre du terme. Nous n'emploierons donc ici le terme de métrique qu'en relation avec des notions de distance.

1.2 Le masting : illustration avec données individuelles

Le jeu de données illustratif est extrait[6] d'une étude [6] sur l'intensité de l'effort de reproduction chez le chêne pédonculé, *Quercus robur* L., sur le campus de *Silwood Park* en Angleterre, suivi au niveau individuel sur une trentaine d'arbres pendant 12 ans.

```
download.file("https://pbil.univ-lyon1.fr/R/donnees/StatsMasting/asp.Rda",
              destfile = "asp.Rda")
```

```
load("asp.Rda")
head(aps[ , 1:7])
```

```
       T_1001 T_1002 T_1003 T_1004 T_1005 T_1006 T_1010
Y_1982  0.970  0.640  0.550  0.930  0.170  0.150  0.062
Y_1983  0.505  0.000  0.075  0.025  0.225  0.000  0.250
Y_1984  0.325  0.075  0.650  0.425  0.100  0.000  0.025
Y_1985  0.700  0.300  0.400  0.625  0.325  0.150  0.175
Y_1986  0.040  0.040  0.040  0.000  0.020  0.000  0.080
Y_1987  0.575  0.125  0.400  0.525  0.150  0.175  0.000
```

[4] Version du 2022-02-03.

[5] Le terme s'entend aussi dans un sens moins restrictif comme dans le titre « [l]e droit à la glandée » de l'article de Cédric Michalski [38] sur l'histoire d'icelui. On y trouvera la trace documentée d'expressions poétiques telle que le « panage d'oiseaux », de l'allemand *vogel-mast*, pour signifier la restriction des droits des porchers à exploiter la ressource en cas d'insuffisance de production de glands.

[6] Les 5 arbres suivants ont été censurés ici : n° 1007 frappé par la foudre en 1995, n° 1008 détruit par la tempête centennale de 1987, n° 1009 très affecté par la sécheresse de 1989, n° 1011 qui a perdu de nombreuses branches principales et n° 1013 qui n'a jamais produit de fleurs femelles pendant les 12 années de suivi.

Figure 2: Photo prise le 29 mai 2008 dans la forêt domaniale de Trois Fontaines dans la Marne en France. Après une glandée le sol est recouvert au printemps de jeunes pousses de chênes, ce que les forestiers appellent de façon imagée « la salade ». Crédit photo : François Débias.

Les données se présentent sous la forme d'une table avec les séries temporelles de chaque arbre en colonne. Le choix fait ici de disposer les séries en colonne ne doit rien au hasard. Le logiciel R suit la convention universelle des logiciels de statistique de placer les individus, au sens statistique du terme, en ligne et les variables en colonne. Ceci ne nous aide pas beaucoup parce que l'on peut aussi bien considérer que nous avons des individus-arbres observés sur des variables-années que des individus-années observés sur des variables-arbres. Le choix découle de considérations techniques : il est beaucoup plus rapide d'accéder aux données d'une colonne qu'à celle d'une ligne : les données d'une colonne sont stockées de façon contiguë en mémoire et peuvent être extraites d'un bloc alors que celles d'une ligne sont stockées de façon fragmentée et leur invocation nécessite des calculs pour les localiser. Ce que nous allons très majoritairement manipuler par la suite ce sont les séries temporelles, il est donc plus efficace de les placer en colonne :

		Arbre n° 1	…	Arbre n° j	…	Arbre n° 25
		1	…	j	…	p
1982	1	x_{11}	…	x_{1j}	…	x_{1p}
⋮	⋮	⋮		⋮		⋮
⋮	i	x_{i1}	…	x_{ij}	…	x_{ip}
⋮	⋮	⋮		⋮		⋮
1993	n	x_{n1}	…	x_{nj}	…	x_{np}

Les n lignes correspondent aux n années d'observation, ici $n = 12$. Les p colonnes correspondent aux p arbres suivis, ici $p = 25$. La valeur x_{ij} représente l'observation faite l'année i sur l'arbre j. Le vecteur $\boldsymbol{x}_j \in \mathbb{R}_+^n$ est la série temporelle associée à l'arbre j.

Pour illustrer le fait que les colonnes sont stockées de façon contiguë en mémoire, on utilise une matrice ayant la suite des quatre premiers entiers dans toutes ses colonnes, on la sauve au format binaire XDR [50] puis on la visualise avec le paquet `hexView` [39]. On constate que les quatre premiers entiers sont bien rangés de façon consécutive en mémoire :

```
dump <- matrix(c(1L, 2L, 3L, 4L), nrow = 4, ncol = 5)
dump
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    2    2    2    2    2
[3,]    3    3    3    3    3
[4,]    4    4    4    4    4
```

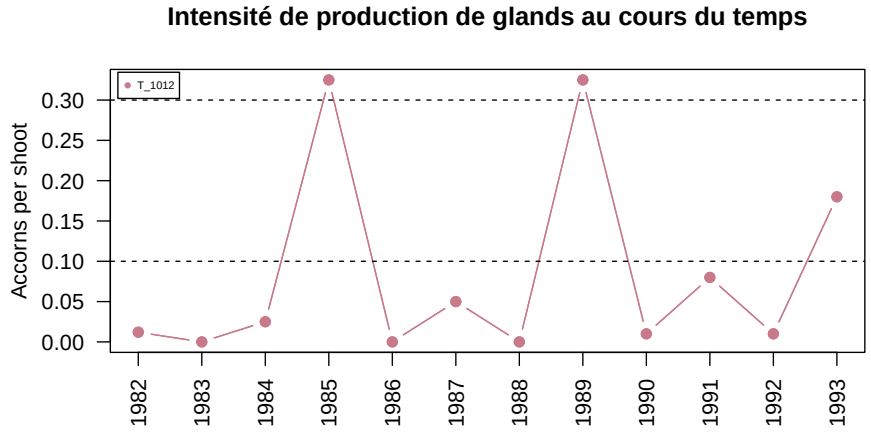
```
saveRDS(dump, file = "dump.rds", compress = FALSE)
library(hexView)
readRaw("dump.rds", human = "int", endian = "big", offset = 7)
```

```
  7  :  04 01 01 00 03 05 00 00 00 00 05 55  |     67174656    50659328        1365
 19  :  54 46 2d 38 00 00 02 0d 00 00 00 14  |   1413885240        525          20
 31  :  00 00 00 01 00 00 00 02 00 00 00 03  |           1          2           3
 43  :  00 00 00 04 00 00 00 01 00 00 00 02  |           4          1           2
 55  :  00 00 00 03 00 00 00 04 00 00 00 01  |           3          4           1
 67  :  00 00 00 02 00 00 00 03 00 00 00 04  |           2          3           4
 79  :  00 00 00 01 00 00 00 02 00 00 00 03  |           1          2           3
 91  :  00 00 00 04 00 00 00 01 00 00 00 02  |           4          1           2
103  :  00 00 00 03 00 00 00 04 00 00 04 02  |           3          4        1026
115  :  00 00 00 01 00 04 00 09 00 00 00 03  |           1     262153           3
127  :  64 69 6d 00 00 00 0d 00 00 00 02 00  |  1684630784       3328         512
139  :  00 00 04 00 00 00 05 00 00 00 fe     |        1024       1280
```

Nous utilisons la fonction `plot.tree()`, qui sera détaillée par la suite dans la section 1.4.1 page 10, pour représenter les données de l'arbre n° 1012 :

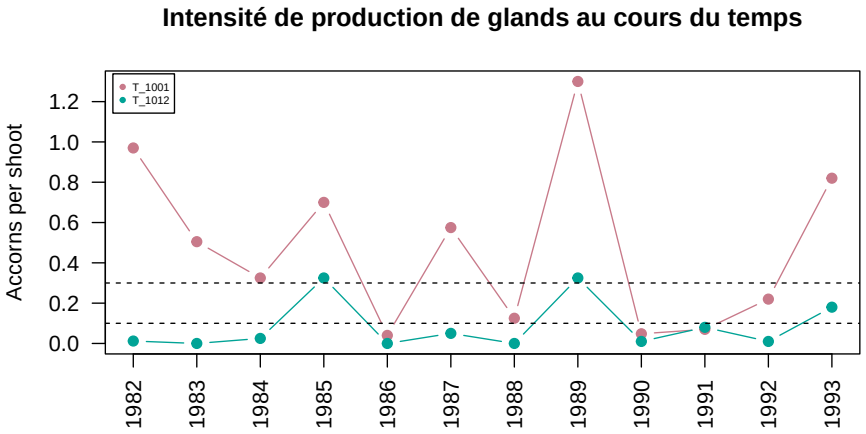
```
download.file("https://pbil.univ-lyon1.fr/R/donnees/StatsMasting/plottree.Rda",
              destfile = "plottree.Rda")
```

```
load("plottree.Rda")
plot.tree(aps, "T_1012")
abline(h = c(0.1, 0.3), lty = 2)
```



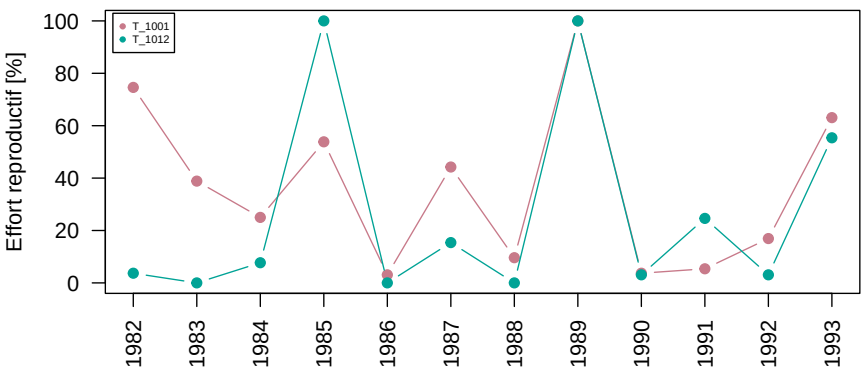
On aurait donc envie de dire qu'il y a eu pour cette arbre glandée en 1985 et 1989 (>0.3) et pas de glandée pour les années de production inférieure à 0.1. Mais que dire de l'année 1993 avec sa production très intermédiaire ? Ajoutons maintenant l'arbre n° 1001 :

```
plot.tree(aps, c("T_1012", "T_1001"))
abline(h = c(0.1, 0.3), lty = 2)
```



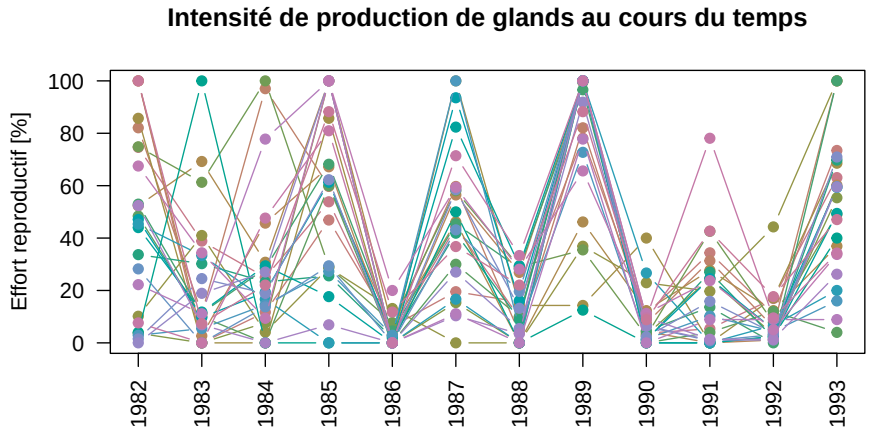
La première chose qui saute aux yeux, c'est le grand synchronisme entre ces deux arbres, une caractéristique essentielle du *masting*. Mais on remarque que, en moyenne, l'arbre n° 1001 est un bien plus gros producteur que l'arbre n° 1012. Si on dispose d'une série temporelle suffisamment longue, on peut considérer que la valeur maximale observée représente le maximum du potentiel reproducteur de l'arbre, et alors exprimer les données en pourcentage de cette valeur :

```
apsr <- apply(aps, 2, \(x) 100*x/max(x))
save(apsr, file = "apsr.Rda")
plot.tree(apsr, c("T_1012", "T_1001"), ylab = "Effort reproductif [%]")
```



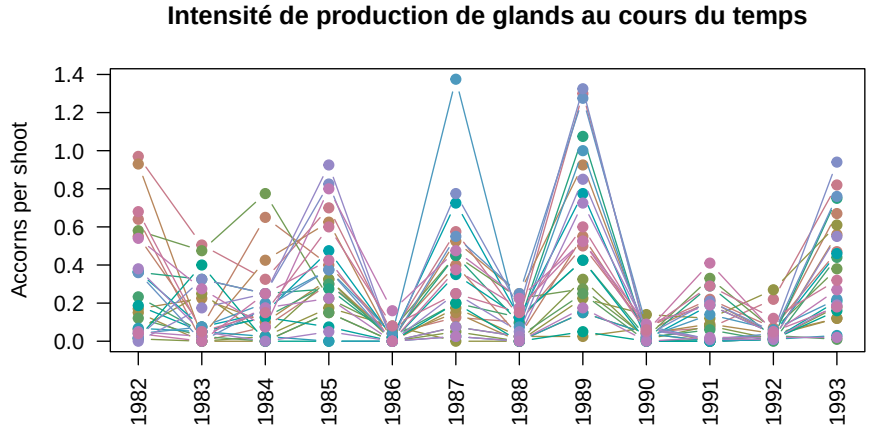
Pour ces deux arbres il y a certainement glandée en 1989 puisque les deux sont au maximum de leur potentiel reproducteur. Si on représente maintenant tous les arbres de la population :

```
plot.tree(apsr, colnames(apsr), ylab = "Effort reproductif [%]", leg = FALSE)
```



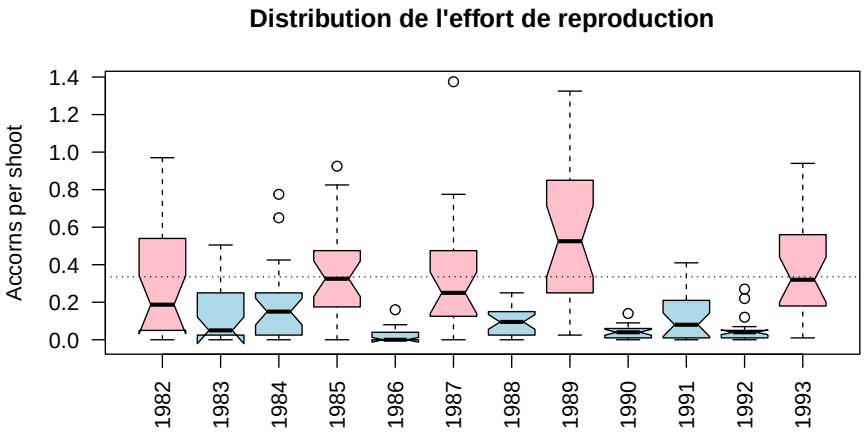
On a très envie de déclarer qu'il y a eu glandée en 1989 puisque 20 arbre sur 25 sont à plus de 60 % de leur potentiel reproducteur. Cette représentation peut cependant être trompeuse : si on imagine que ces 20 arbres sont en moyenne de très faibles producteurs et que les 5 autres sont des hyper-producteurs, on n'observera pas de glandée au niveau populationnel. Il faut revenir au données brutes pour vérifier qu'il n'y a pas de problème :

```
plot.tree(aps, colnames(apsr), leg = FALSE)
```



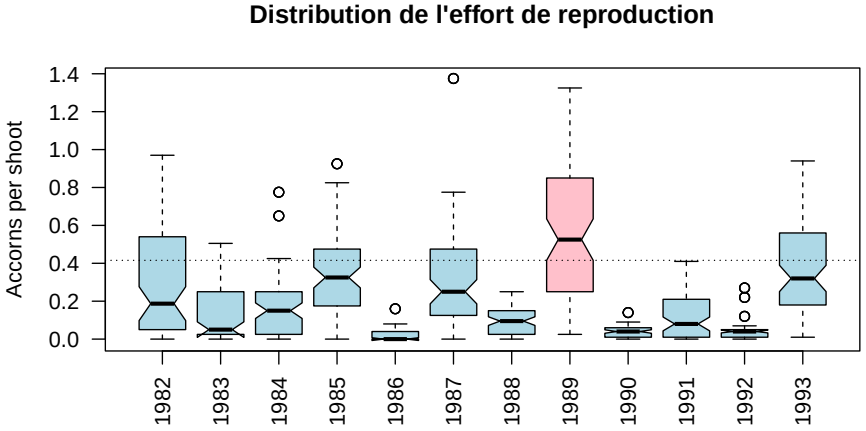
On voit qu'il y a bien eu glandée en 1989. Peut-on préciser les choses ? On utilise une représentation avec des boîtes-à-moustache avec l'option qui indique avec des encoches l'intervalle de confiance ($\alpha = 0.05$) pour la médiane de la distribution :

```
rbx <- boxplot(t(aps), notch = TRUE, plot = FALSE)
col <- ifelse(rbx$conf[2 ,] > rbx$conf[1, 8], "pink", "lightblue")
boxplot(t(aps), notch = TRUE, xaxt = "n", las = 1, ylab = "Accorns per shoot",
  main = "Distribution de l'effort de reproduction", col = col)
axis(1, at = 1:12, labels = 1982:1993, las = 3)
abline(h = rbx$conf[1, 8], lty = 3)
```



L'année 1989 est bien caractérisée par une production anormalement élevée, mais on ne peut pas affirmer que la médiane de cette production était plus élevée qu'en 1982, 1985, 1987 et 1993. On voit ici que la définition d'une « grande quantité de glands » est loin d'être évidente. On pourrait être tenté au vu de la représentation ci-dessus de déclarer qu'il y a glandée toutes les années où la médiane ne diffère pas significativement de celle du pic de production. Le problème est que ce critère est très lié à la taille de la population suivie, qui est généralement faible vu le coût expérimental. Si on triple artificiellement la taille de l'échantillon, il n'y aurait plus qu'une glandée en 1989 :

```
trpl <- rbind(t(aps), t(aps), t(aps))
rbx <- boxplot(trpl, notch = TRUE, plot = FALSE)
col <- ifelse(rbx$conf[2 ,] > rbx$conf[1, 8], "pink", "lightblue")
boxplot(trpl, notch = TRUE, xaxt = "n", las = 1, ylab = "Accorns per shoot",
  main = "Distribution de l'effort de reproduction", col = col)
axis(1, at = 1:12, labels = 1982:1993, las = 3)
abline(h = rbx$conf[1, 8], lty = 3)
```



1.3 Le masting : illustration avec données populationnelles

Les séries individuelles sont rares, par exemple l'article de Carlos M. Herrera ne comporte que 16 études, dont la moitié non publiées, chez 11 espèces. On ne dispose en général de données qu'au niveau populationnel. Pour illustrer cet aspect on utilisera les 1433 séries temporelles quantitatives documentées pour au moins 12 ans de la base de données MASTREE+ [17] dont la version et les variables sont décrites par ailleurs[7].

```
download.file("https://pbil.univ-lyon1.fr/R/donnees/StatsMasting/mastree.Rda",
              destfile = "mastree.Rda")
```

```
load("mastree.Rda")
ms <- mastree[mastree$VarType == "C" & mastree$Length >= 12, ]
rm(mastree)
length(unique(ms$ID))
```

```
[1] 1433
```

```
save(ms, file = "ms.Rda")
```

1.4 Fonctions utilitaires

1.4.1 `plot.tree()`

Cette fonction sert à représenter une ou plusieurs séries temporelles du jeu de données individuelles et à déjà été illustrée précédemment (cf. section 1.2 page 4). Le premier argument `df` doit avoir les années pour nom des lignes sous la forme `Y_NNNN` et les arbres pour nom des colonnes. Le vecteur `list_tree` donne la liste des arbres à représenter. Le booléen `leg` contrôle l'affichage de la légende.

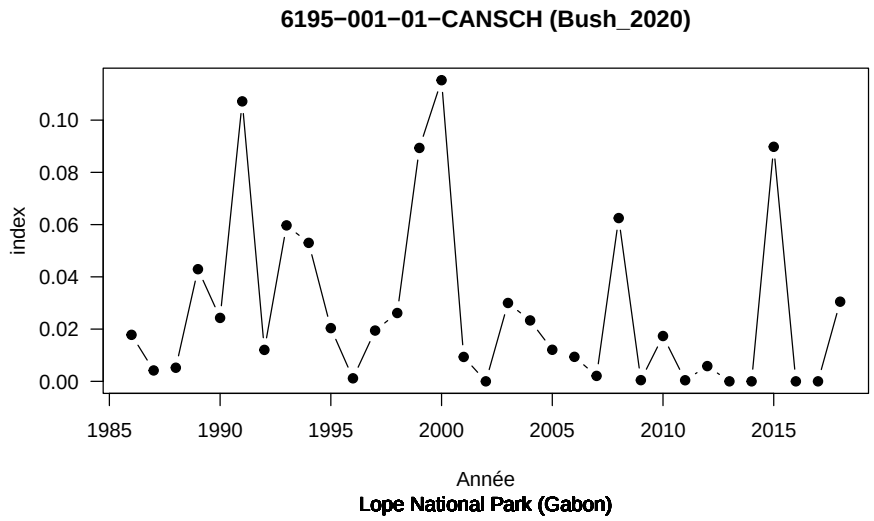
[7] Voir la fiche dédiée à `https://pbil.univ-lyon1.fr/R/pdf/MASTREE.pdf`

```
plot.tree <- function(df, list_tree, type = "b", pch = 19,
  ylab = "Accorns per shoot", leg = TRUE, posleg = "topleft",
  main = "Intensité de production de glands au cours du temps", ...){
  Years <- sapply(rownames(df),
    \(x) as.integer(unlist(strsplit(x, split = "_"))[2]))
  jcol <- which(colnames(df) %in% list_tree)
  ylim <- c(0, max(df[ , jcol]))
  nt <- length(jcol)
  col <- hcl.colors(nt, "Dark 2")
  plot(Years, df[ , jcol[1]], type = type, pch = pch, xaxt = "n", las = 1,
     xlab = "", ylab = ylab, col = col[1], ylim = ylim, main = main, ...)
  axis(1, at = Years, las = 3)
  for(j in 2:nt)
    points(Years, df[ , jcol[j]], type = type, pch = pch, col = col[j])
  if(leg) legend(posleg, inset = 0.01, pch = pch, col = col,
         legend = colnames(df)[jcol], cex = 0.5)
}
save(plot.tree, file = "plottree.Rda")
```

1.4.2 `plotID()`

Cette fonction sert à représenter une série temporelle de la base de données MASTREE+ [17] à partir de sa clef d'identification. Elle est illustrée ici par un suivi[8] de 33 ans de *Canarium schweinfurthii* dans le parc national de Lopé au Gabon.

```
plotID <- function(the_ID, las = 1, type = "b", pch = 19, ...){
  df <- ms[ms$ID == the_ID, ]
  x <- df$Year
  y <- df$Value
  main <- paste(the_ID, " (", unique(df$Reference), ")", sep = "")
  plot(x, y, main = main, xlab = "Année", ylab = unique(df$Unit),
       las = las, type = type, pch = pch, ...)
  title(sub = paste(df$Site, " (", df$Country, ")", sep = ""))
}
save(plotID, file = "plotID.Rda")
plotID("6195-001-01-CANSCH")
```



[8] Tutin, CEG; Abernethy, K; White, L; Dimoto, E; Dikangadissi, JT; Jeffery, KJ; Momont, L; Ukizintambara, T; Bush, ER (2029): Lopé Tree Phenology Dataset. Version 1.2. University of Stirling. Faculty of Natural Sciences. Dataset. http://hdl.handle.net/11667/152

1.4.3 myarrows()

Cette fonction dérive directement de la fonction de base `arrows()` avec simplement un jeu de paramètres permettent de tracer les intervalles de confiance. Elle sera utilisée par exemple dans la section 3.1.6 page 27.

```
myarrows <- function(x0, y0, x1, y1, length = 0.05, angle = 90, code = 3,
  col = grey(0.5), ...){
  arrows(x0, y0, x1, y1, length, angle, code, col, ...)
}
save(myarrows, file = "myarrows.Rda")
```

1.4.4 plotboot()

Cette fonction est utilisée pour visualiser la distribution bootstrap d'une statistique. La ligne verticale bleu donne la valeur de la statistique, les lignes verticales rouges l'intervalle de confiance. Elle sera utilisée par exemple dans la section 3.1.3 page 17.

```
plotboot <- function(boot.out, rug = FALSE, ticksize = 0.03, ...){
  plot(density(boot.out$t), ...)
  abline(v = boot.out$t0, col = "blue")
  text(boot.out$t0, par("usr")[4]/2, signif(boot.out$t0, 3), pos = 4, col = "blue", xpd = NA)
  ci <- boot.ci(boot.out, type = "bca")
  abline(v = ci$bca[4:5], col = "red")
  text(ci$bca[4], par("usr")[4]/4, signif(ci$bca[4], 3), pos = 4, col = "red", xpd = NA)
  text(ci$bca[5], 3*par("usr")[4]/4, signif(ci$bca[5], 3), pos = 4, col = "red", xpd = NA)
  if(rug) rug(unique(boot.out$t), ticksize, col = "blue")
}
save(plotboot, file = "plotboot.Rda")
```

2 Notation des statistiques

On reprend ici des notations classiquement utilisées dans la littérature ayant trait au *masting*. Pour qu'il n'y ait pas d'ambiguïté on précise comment elles sont calculées avec la structure du jeu de données utilisé comme exemple (cf. section 1.2 page 4).

2.1 Valeurs des séries temporelles

Toutes les séries temporelles sont constituées de valeurs positives ou nulles :

$$\forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, p\} : x_{ij} \in \mathbb{R}_+$$

2.2 Statistiques individuelles (par arbre)

Les statistiques individuelles sont notées S_i, par exemple CV_i. Pour un arbre j donné, la statistique $S_i(\boldsymbol{x}_j)$ est calculée en colonne dans le jeu de données :

		Arbre n° 1	...	Arbre n° j	...	Arbre n° 25
		1	...	j	...	p
1982	1	x_{11}	...	x_{1j}	...	x_{1p}
⋮	⋮	⋮		⋮		⋮
⋮	i	x_{i1}	...	x_{ij}	...	x_{ip}
⋮	⋮	⋮		⋮		⋮
1993	n	x_{n1}	...	x_{nj}	...	x_{np}
		↓	...	↓	...	↓
		$\mathrm{S_i}(\boldsymbol{x}_1)$	...	$\mathrm{S_i}(\boldsymbol{x}_j)$	...	$\mathrm{S_i}(\boldsymbol{x}_p)$

2.3 Statistiques individuelles moyennes (par site)

Les statistiques individuelles moyennes par site sont notées $\overline{\mathrm{S_i}}$, par exemple $\overline{\mathrm{CV_i}}$, et calculées comme étant la moyenne arithmétique des statistiques individuelles :

$$\overline{\mathrm{S_i}} = \frac{1}{p}\sum_{j=1}^{p} \mathrm{S_i}(\boldsymbol{x}_j)$$

Bien qu'elle soit basée sur des statistiques individuelles, il n'y a qu'une seule statistique $\overline{\mathrm{S_i}}$ pour un site donné :

		Arbre n° 1	...	Arbre n° j	...	Arbre n° 25	
		1	...	j	...	p	
1982	1	x_{11}	...	x_{1j}	...	x_{1p}	
⋮	⋮	⋮		⋮		⋮	
⋮	i	x_{i1}	...	x_{ij}	...	x_{ip}	
⋮	⋮	⋮		⋮		⋮	
1993	n	x_{n1}	...	x_{nj}	...	x_{np}	
		$\mathrm{S_i}(\boldsymbol{x}_1)$	...	$\mathrm{S_i}(\boldsymbol{x}_j)$	...	$\mathrm{S_i}(\boldsymbol{x}_p)$	$\rightarrow \overline{\mathrm{S_i}}$

2.4 Statistiques populationnelles (par site)

Les statistiques populationnelles sont notées $\mathrm{S_p}$, par exemple $\mathrm{CV_p}$. C'est ce que l'on aurait observé si les statistiques individuelles n'étaient pas disponibles : pour une année donnée, i, nous n'avons accès que à la moyenne $\overline{x_i}$. En notant $\bar{\boldsymbol{x}}$ la série temporelle des moyennes, la statistique populationnelle se calcule comme $\mathrm{S_p} = \mathrm{S}(\bar{\boldsymbol{x}})$:

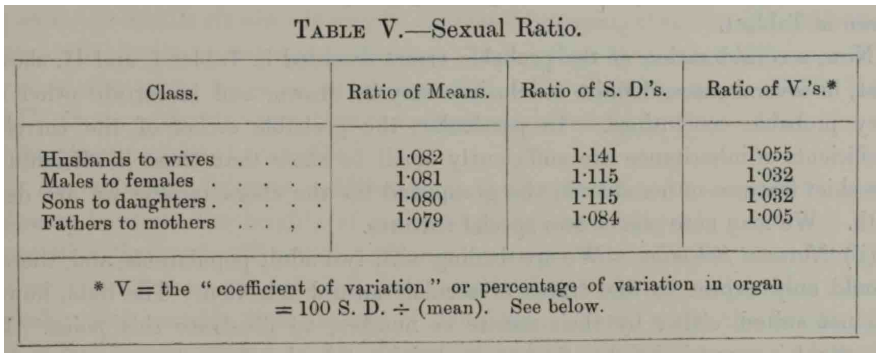


TABLE V.—Sexual Ratio.

Class.	Ratio of Means.	Ratio of S. D.'s.	Ratio of V.'s.*
Husbands to wives	1·082	1·141	1·055
Males to females	1·081	1·115	1·032
Sons to daughters	1·080	1·115	1·032
Fathers to mothers	1·079	1·084	1·005

* V = the "coefficient of variation" or percentage of variation in organ = 100 S. D. ÷ (mean). See below.

Figure 3: Première mention du coefficient de variation dans la table V de l'article de PEARSON [42]. Il divise l'écart-type de chaque groupe par la moyenne pour obtenir une mesure sans dimensions et simplifier les calculs. Son objectif ici n'était pas de définir une statistique de la variabilité.

		Arbre n° 1	...	Arbre n° j	...	Arbre n° 25	
		1	...	j	...	p	$\bar{\boldsymbol{x}}$
1982	1	x_{11}	...	x_{1j}	...	x_{1p}	$\overline{x_1}$
⋮	⋮	⋮		⋮		⋮	
⋮	i	x_{i1}	...	x_{ij}	...	x_{ip}	$\overline{x_i}$
⋮	⋮	⋮		⋮		⋮	
1993	n	x_{n1}	...	x_{nj}	...	x_{np}	$\overline{x_n}$
							↓
							S_p

3 Statistiques de la variabilité inter-annuelle

3.1 CV : coefficient de variation

3.1.1 Définition et propriétés

Le coefficient de variation, CV [42], est l'écart-type de l'échantillon, s_x, divisé par la moyenne arithmétique de l'échantillon, $\bar{x}$, et exprimé en pourcentage (cf. figure 3 page 14). Par convention, il est forcé à 0 quand la moyenne est nulle[9]. C'est une statistique très utilisée pour quantifier la variabilité inter-annuelle dans les études relatives au *masting* [47, 53, 28, 21, 33, 32].

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$s_x^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2$$

[9]Les séries temporelles relatives au *masting* sont souvent riches en zéros, par ré-échantillonnage intensif il n'est pas du tout improbable de tomber sur le vecteur nul. De plus, il existe des séries documentées constamment égales à zéro (cf. section 3.1.5 page 20).

$$\mathrm{CV}(\boldsymbol{x}) = 100 \times \begin{cases} 0 & \text{si} \quad \bar{x} = 0 \\ \frac{s_x}{\bar{x}} & \text{si} \quad \bar{x} \neq 0 \end{cases}$$

UNE propriété importante du CV, qui explique sans doute en partie sa popularité dans les études relatives au *masting*, est son invariance d'échelle : on peut multiplier une série temporelle $\boldsymbol{x}_j$ par un scalaire λ strictement positif sans changer la valeur de la statistique.

$$\forall \lambda \in \mathbb{R}_+^*, \forall j \in \{1, 2, \ldots, p\}, \forall \boldsymbol{x}_j \in \mathbb{R}_+^n : \mathrm{CV}(\lambda \boldsymbol{x}_j) = \mathrm{CV}(\boldsymbol{x}_j)$$

L'INTÉRÊT de l'invariance d'échelle est qu'elle correspond à une notion souvent utilisée en biologie de « variabilité relative » ou de « variabilité intrinsèque ». Ce que cela signifie est expliqué ainsi par SIMPSON, ROE et LEWONTIN ([48] cité par [34]) : « le fait que les éléphants aient un écart type de 50 mm pour une variable morphométrique linéaire et que les musaraignes aient un écart type de 0.5 mm pour cette même variable ne signifie pas nécessairement que les éléphants soient plus variables, d'un point de vue zoologique, que les musaraignes. La taille des éléphants est cent fois celle des musaraignes, nous nous attendons donc à ce que les variations dans l'absolu soient cent fois plus importantes sans qu'il n'y ait de différence essentielle de variabilité fonctionnelle ». Il en va de même pour les études relatives au *masting* : si nous observons pour l'arbre n° 1 la série temporelle $\boldsymbol{x}_1 = (1, 5, 0, 2)$ et pour l'arbre n° 2 la série temporelle $\boldsymbol{x}_2 = (10, 50, 0, 20)$ nous voyons deux arbres qui présentent intrinsèquement la même variabilité, simplement l'arbre n° 2 est un bien meilleur producteur. Enfin, les séries temporelles relatives au *masting* ayant un « zéro absolu[10] », l'invariance d'échelle fait que l'on peut comparer les CV de séries exprimées dans des unité différentes.

ATTENTION, l'invariance d'échelle ne signifie pas qu'il y a invariance par translation. L'exemple suivant est adapté de l'article de EISENHAUER [12] à partir de données réelles [31].

```
CV <- function(x) 100*sqrt((length(x) - 1)/length(x))*sd(x)/mean(x)
x1 <- c(279.85, 279.85, 280.95, 280.05, 286.35, 287.85, 291.45,
        290.15, 288.25, 285.45, 280.35, 278.65)
CV(x1)
[1] 1.557691
x2 <- c(44.06, 44.06, 46.04, 44.42, 55.76, 58.46, 64.94, 62.6,
        59.18, 54.14, 44.96, 41.9)
CV(x2)
[1] 15.4046
x3 <- c(6.7, 6.7, 7.8, 6.9, 13.2, 14.7, 18.3, 17.0, 15.1, 12.3,
        7.2, 5.5)
CV(x3)
[1] 40.41461
```

NOUS avons donc trois séries temporelles avec des CV bien différents, de 1.6 % à 40.4 % en passant par 15.4 %. Il faudrait bien se garder d'en conclure quoi que ce soit parce qu'il s'agit en fait de la *même* série temporelle de températures mais exprimée dans des unités différentes :

[10] Quand une année il n'y a pas de glands, par exemple à l'automne 2021 dans la forêt de Seillon, eh bien il n'y a pas de glands ! Quelle que soit la façon de mesurer les choses cela fera toujours zéro. On peut bien entendu avoir un faux zéro si l'effort d'échantillonnage est insuffisant ou un problème de saturation du signal pour les valeurs élevées, mais en l'absence de biais dans la mesure le CV permet bien de comparer la variabilité de séries exprimées dans des unités différentes.

```
x3            # degrés Celcius
[1]  6.7  6.7  7.8  6.9 13.2 14.7 18.3 17.0 15.1 12.3  7.2  5.5
x1 - 273.15   # Kelvin -> degrés Celcius
[1]  6.7  6.7  7.8  6.9 13.2 14.7 18.3 17.0 15.1 12.3  7.2  5.5
(x2 - 32)/1.8 # degrés Fahrenheit -> degrés Celcius
[1]  6.7  6.7  7.8  6.9 13.2 14.7 18.3 17.0 15.1 12.3  7.2  5.5
```

3.1.2 Implémentation

Pour calculer le coefficient de variation nous avons besoin de l'écart-type de l'échantillon. La fonction `sd()` de R calcule $\hat{\sigma}$, l'estimation non-biaisée de l'écart-type de la population :

$$\hat{\sigma} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} = s_x\sqrt{\frac{n}{n-1}}$$

Nous allons nous baser sur `sd()` pour définir une fonction `sdn()` pour calculer l'écart-type de l'échantillon. Ce n'est pas aussi simple que cela en à l'air car il faut penser à gérer correctement les valeurs manquantes et que nous voulons pouvoir l'utiliser directement ou bien comme argument de la fonction `boot()` du paquet standard éponyme [5, 8]. Les fonctions passées en argument à `boot()` doivent avoir comme premier argument le vecteur `x` des observations et comme deuxième argument le vecteur `i` des indices des valeurs ré-échantillonnées. Notez que la fonction `sdn()` fonctionnera même si l'argument `i` est manquant puisque `x[ ]` renvoie `x`.

```
sdn <- function(x, i, ...){
  n <- sum(!is.na(x)) # nombre de valeurs renseignées
  return(sqrt((n - 1)/n)*sd(x[i], ...))
}
# Utilisation directe
obs <- 1:10 # données bidon
sdn(obs)
[1] 2.872281
sdn(c(NA, obs))
[1] NA
sdn(c(NA, obs), na.rm = TRUE)
[1] 2.872281
# Passage en argument à boot()
library(boot)
boot.out <- boot(obs, sdn, R = 9999)
boot.ci(boot.out, type = "bca")$bca[4:5]
[1] 2.154066 3.822610
```

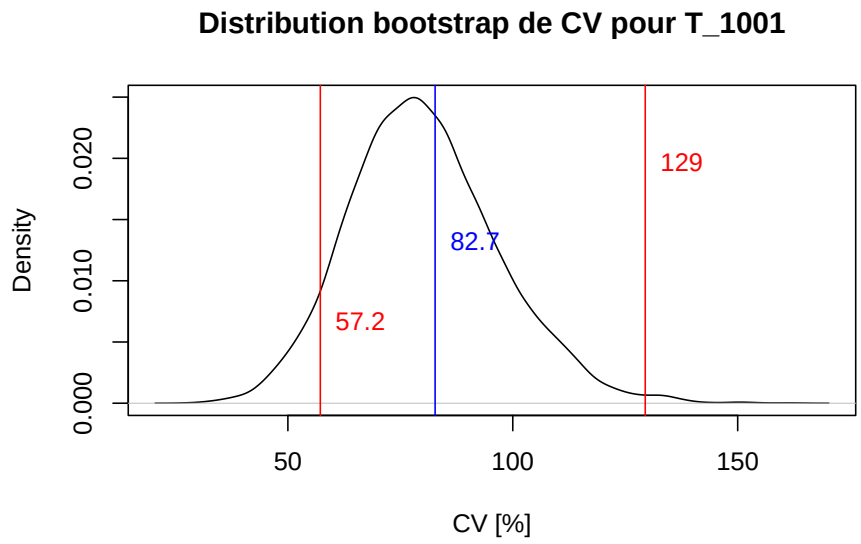
Pour définir la fonction `CV()` nous utilisons la fonction standard de R `all.equal()` qui permet de tester si la valeur de la moyenne de la série est nulle en tenant compte des imprécisions numériques de calcul.

```
CV <- function(x, i, ...){
  barx <- mean(x[i], ...)
  if(isTRUE(all.equal(barx, 0))) return(0)
  return(100*sdn(x[i], ...)/barx)
}
CV(0)
[1] 0
all.equal(CV(obs), CV(c(NA, obs), na.rm = TRUE))
[1] TRUE
```

3.1.3 CV individuels

On prend comme exemple les données de l'arbre n° 1001. On définit une petite fonction pour représenter la distribution bootstrap de CV. Elle est obtenue en tirant de nombreuses fois *avec remise* dans les valeurs observées et en calculant à chaque fois la valeur de CV. Pour le calcul de l'intervalle de confiance on choisit le type `bca` recommandé par le « pape » du bootstrap [11].

```
set.seed(1)
boot.CV <- boot(aps[ , 1], CV, R = 9999)
plotboot(boot.CV, main = "Distribution bootstrap de CV pour T_1001", xlab = "CV [%]")
```



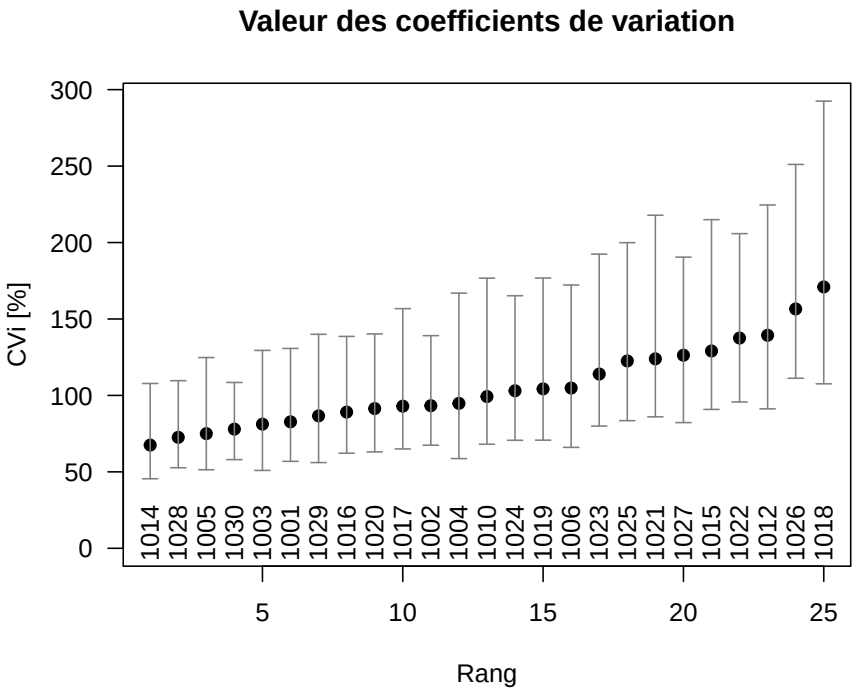
Comme on peut le constater la valeur de CV pour cet arbre est de 82.7 %, mais avec une grande imprécision puisque l'intervalle de confiance va de 57.2 à 129 %, soit une amplitude relative de 87 %. On calcule maintenant le CV et les intervalles de confiance pour tous les arbres :

```
CVi <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(CVi) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  CVi[i, "Tree_name"] <- colnames(aps)[i]
  boot.out <- boot(aps[ , i], CV, R = 9999)
  CVi[i, "est"] <- boot.out$t0
  ci <- boot.ci(boot.out, type = "bca")
  CVi[i, "inf"] <- ci$bca[4]
  CVi[i, "sup"] <- ci$bca[5]
}
save(CVi, file = "CVi.Rda")
```

Dans mon document source ce fragment de code n'est pas exécuté pour produire le PDF final (`«eval=FALSE»=`). Je l'ai exécuté manuellement une fois et sauvegardé le résultat dans le fichier `CVi.Rda`, je n'ai donc plus qu'à le recharger en mémoire avec `load("CVi.Rda")` pour pouvoir produire rapidement mon document final sans avoir à ré-exécuter des calculs potentiellement longs :

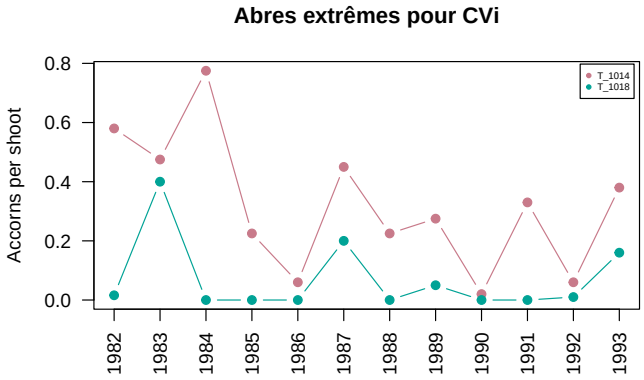
```
load("CVi.Rda")
CVi <- CVi[order(CVi$est), ]
n <- nrow(CVi)
```

```
plot(1:n, CVi$est, pch = 19, ylim = c(0, max(CVi$sup)),
     las = 1, ylab = "CVi [%]", xlab = "Rang",
     main = "Valeur des coefficients de variation")
with(CVi, myarrows(1:n, inf, 1:n, sup))
text(1:n, rep(10, n), substr(CVi$Tree_name, 3, 7), srt = 90, xpd = NA)
```



Les intervalles de confiance sont tellement larges que l'on ne peut même pas affirmer qu'il y a une différence significative entre l'arbre le moins variable (n° 1014) et l'arbre le plus variable (n° 1018). Pourtant, quand on regarde les données, on aurait bien envie de dire que ces deux arbres ont un comportement différent : n° 1014 produit toujours, bon an, mal an, quelques glands alors que n° 1018 est beaucoup plus erratique et n'a produit des glands en moyenne qu'une année sur quatre :

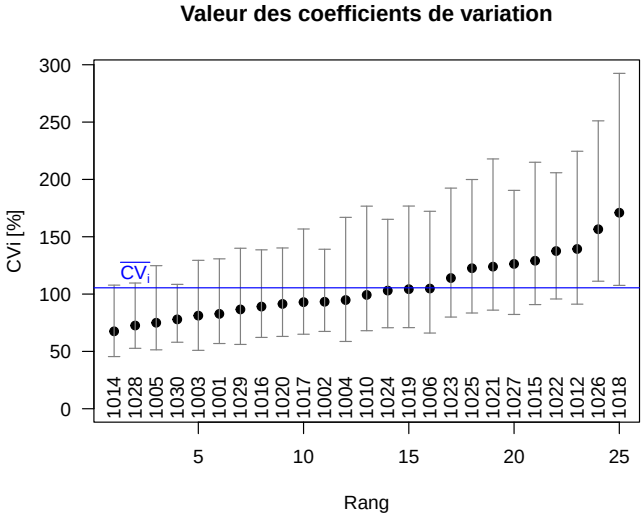
```
plot.tree(aps, c("T_1014", "T_1018"), main = "Abres extrêmes pour CVi",
          posleg = "topright")
```



3.1.4 CV individuels moyen

C'est la moyenne arithmétique des CV_i (cf. section 2.3 page 13). Graphiquement on peut voir que quasiment aucun arbre n'a un CV_i différant significativement du $\overline{CV_i}$:

```
plot(1:n, CVi$est, pch = 19, ylim = c(0, max(CVi$sup)),
     las = 1, ylab = "CVi [%]", xlab = "Rang",
     main = "Valeur des coefficients de variation")
with(CVi, myarrows(1:n, inf, 1:n, sup))
text(1:n, rep(10, n), substr(CVi$Tree_name, 3, 7), srt = 90, xpd = NA)
CVm <- mean(CVi$est)
abline(h = CVm, col = "blue")
text(2, 100, expression(bar(CV[i])), pos = 3, col = "blue")
```



Cela ne présente pas forcément un grand intérêt ici, mais pour être homogène avec ce qui précède on peut facilement calculer un intervalle de confiance par bootstrap :

```
moy <- function(x, i, ...) mean(x[i], ...)
(ci.CVm <- boot.ci(boot(CVi$est, moy, R = 9999), type = "bca")$bca[4:5])
```

```
[1]  95.9250 116.9485
```

3.1.5 CV populationnel

C'est le CV calculé sur la moyenne arithmétique des valeurs chaque année (cf. section 2.4 page 13). La moyenne étant égale à une constante près à la somme des valeurs, tout se passe bien comme si on avait observé l'effort de reproduction à l'échelle de la population. Pour calculer la moyenne on utilise la fonction de base `rowMeans()` parce qu'optimisée pour son temps d'exécution.

```
xmoy <- rowMeans(aps)
boot.out <- boot(xmoy, CV, R = 9999)
(CVp <- boot.out$t0)
```

```
[1] 75.7088
```

```
(ci.CVp <- boot.ci(boot.out, type = "bca")$bca[4:5])
```

```
[1]  53.54241 114.11090
```

Comme le calcul de statistiques pour de nombreuses séries peut être un peu long, nous allons décrire ici en détail comment lancer les calculs en arrière plan. La première étape est de créer un sous-dossier `batch` pour ranger proprement les résultats et d'y sauvegarder tout[11] son environnement de travail en fin de session interactive (ou en fin du fichier `*.Rnw` ou `*.Rmd`) avec la fonction `save.image()` :

```
save.image(file = "batch/myws.Rda")
```

La deuxième étape consiste à créer dans ce même dossier un fichier contenant le script R à exécuter :

batch/CV.R

```
load("myws.Rda")
library(boot)
IDs <- unique(ms$ID) ; n <- length(IDs)
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 4))
colnames(tabres) <- c("ID", "est", "inf", "sup")

for(i in seq_len(n)){
  the_ID <- IDs[i]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.boot <- try(boot::boot(x, CV, R = 9999))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.boot$t0
  try.ci <- try(boot::boot.ci(try.boot, type = "bca"))
  if(!inherits(try.ci, "try-error")) tabres[i, c("inf", "sup")] <- try.ci$bca[4:5]
  CVp <- tabres ; save(CVp, file = "CVp.Rda")
}
```

La première ligne consiste à restaurer l'intégralité de notre espace de travail. L'instruction `library(boot)` sert à pouvoir utiliser directement les fonctions `boot()` et `boot.ci()`, une alternative aurait été de donner le chemin d'accès complet avec `boot::boot()` et `boot::boot.ci()`. Les clefs d'identification des `n` séries temporelles sont stockées dans le vecteur `IDs`. Le tableau des résultats `tabres` comporte le clef d'identification de la série `ID`, la valeur de la statistique `est` et les bornes de l'intervalle de confiance `inf` et `sup`. Toutes les valeurs du tableau des résultats sont initialisées à `NA`, ce qui nous permettra de détecter s'il y a eu une erreur lors du calcul de l'une d'entre elles.

[11] Cela peut sembler un peu excessif, mais qui peut le plus peut le moins, et ainsi on est certain de ne rien oublier.

La suite est une simple boucle pour parcourir toutes les séries temporelles disponibles. Quand on exécute un code en arrière-plan il est intéressant d'être robuste pour qu'une erreur sur une série temporelle n'arrête pas tous les calculs. Ceci est obtenu facilement en testant la valeur retournée par la fonction `try()`. Notez que la table des résultats est sauvegardée à chaque itération, cela permet aux impatients de suivre la progression des calculs.

La dernière étape consiste à lancer les calculs en arrière-plan avec la commande ci-après, à lancer dans un terminal. C'est le dernier caractère « `&` » qui indique de lancer l'exécution en arrière-plan. La première directive `nohup` sert à ne pas interrompre le processus même si l'on se déconnecte de sa session. Voir la documentation (?BATCH) pour les options disponibles pour `R CMD BATCH infile`.

```
unix$ nohup R CMD BATCH CV.R &
```

Si vous êtes d'une nature impatiente, et que vous avez la chance de travailler sur un ordinateur multi-processeurs ou multi-cœurs, ce qui est de plus en plus la norme, il y a moyen d'accélérer considérablement les choses en faisant ce que l'on appelle de la parallélisation par les données. L'idée est simple : comme tous les calculs sont indépendants les uns des autres pour chaque série, on va segmenter le tableau des résultats et confier le calcul de chaque segment à un processus différent. Nous avons donc un processus « maître » qui s'occupe de répartir le travail :

batch/repartiteurCV.R

```
load("myws.Rda")

nproc <- 10 # Nombre de processeurs désiré
R <- 9999 # Taille échantillon bootstrap
IDs <- unique(ms$ID) ; n <- length(IDs)
# Découpage du travail en segments de iinf à isup
ii <- floor(seq(from = 1, to = n, length.out  = nproc + 1))
iinf <- ii[-length(ii)]
isup <- ii[-1] - 1
isup[length(isup)] <- n # Pour ne pas rater le dernier

genericLines <- readLines("genericCV.R")

for(i in seq_len(nproc)){
  # Création du fichier de script R
  fname <- paste0("genericCV-", i, ".R")
  codeR <- c(paste0("iproc <- ", i),
             paste0("iseq <- ", iinf[i], ":", isup[i]),
             paste0("Stat <- 'CV'"),
             paste0("R <- ", R),
             genericLines)
  writeLines(codeR, fname)
  # Création commande de lancement des scripts
  cmd <- paste0("nohup R CMD BATCH ", fname, "&")
  system(cmd)
}
```

Il n'y a pas vraiment besoin de lancer ce processus en arrière-plan puisque c'est lui qui s'occupe de lancer les `nproc` processus en arrière-plan. On peut donc se contenter d'un quasi-instantané :

```
unix$ R CMD BATCH repartiteurCV.R
```

Ce script va lancer en parallèle autant de processus que le nombre de processeurs demandés (variable `nproc`), par exemple pour le n° 3 le script correspondant est :

batch/genericCV-3.R

```
iproc <- 3
iseq <- 287:429
Stat <- 'CV'
R <- 9999
# Début de la partie générique
load("myws.Rda")
IDs <- unique(ms$ID) ; n <- length(iseq)
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 4))
colnames(tabres) <- c("ID", "est", "inf", "sup")
S <- get(Stat)
tabname <- paste0("tab-", iproc, Stat)

for(i in seq_len(n)){
  the_ID <- IDs[iseq[i]]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.boot <- try(boot::boot(x, S, R = R))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.boot$t0
  try.ci <- try(boot::boot.ci(try.boot, type = "bca"))
  if(!inherits(try.ci, "try-error")) tabres[i, c("inf", "sup")] <- try.ci$bca[4:5]
  assign(tabname, tabres)
  save(list = tabname, file = paste0(tabname, ".Rda"))
}
```

Le numéro du processus `iproc` va nous permettre de construire des noms différents pour les tableaux de chaque segment (`tab-1CV.Rda`, `tab-2CV.Rda`, `tab-3CV.Rda`, etc.) de façon à éviter des collisions lors de la sauvegarde des résultats. La séquence `iseq` donne les indices des séries à traiter par le processus, le n° 3 est donc en charge du segment allant des séries 287 à 429. Une fois les calculs effectués[12], il ne nous reste plus qu'à recoller les morceaux :

```
(fics <- dir(path = "batch", pattern = glob2rx("tab-*CV.Rda"), full.names = TRUE))
```

```
 [1] "batch/tab-10CV.Rda" "batch/tab-1CV.Rda"  "batch/tab-2CV.Rda"
 [4] "batch/tab-3CV.Rda"  "batch/tab-4CV.Rda"  "batch/tab-5CV.Rda"
 [7] "batch/tab-6CV.Rda"  "batch/tab-7CV.Rda"  "batch/tab-8CV.Rda"
[10] "batch/tab-9CV.Rda"
```

```
for(i in seq_len(length(fics))) load(fics[i])
(tabs <- ls(pattern = glob2rx("tab-*CV")))
```

```
 [1] "tab-10CV" "tab-1CV"  "tab-2CV"  "tab-3CV"  "tab-4CV"  "tab-5CV"  "tab-6CV"
 [8] "tab-7CV"  "tab-8CV"  "tab-9CV"
```

```
CVpbis <- do.call("rbind", lapply(tabs, get))
save(CVpbis, file = "batch/CVpbis.Rda")
```

L'inspection du tableau des résultats nous montre que le calcul de CV a toujours été possible puisqu'il n'y a aucune valeur manquante, mais que le calcul de l'intervalle de confiance n'a pas été possible pour trois séries :

```
load("batch/CVpbis.Rda")
CVpbis[is.na(CVpbis$est), "ID"]
```

```
character(0)
```

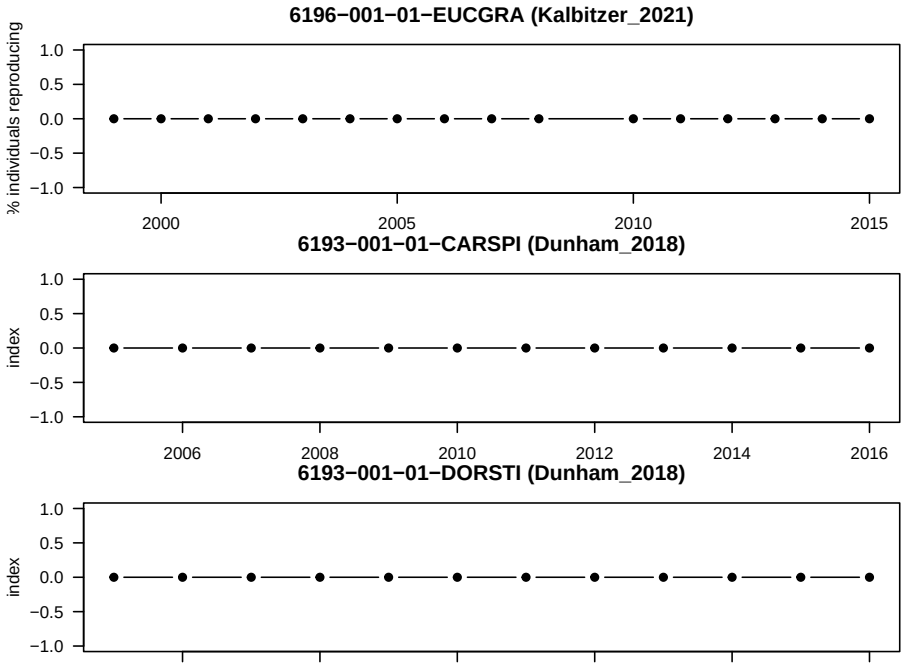
```
CVpbis[is.na(CVpbis$inf), "ID"]
```

```
[1] "6196-001-01-EUCGRA" "6193-001-01-CARSPI" "6193-001-01-DORSTI"
```

Dans les trois cas il s'agit de séries constamment égales à zéro. Comme il n'y a aucune variabilité tous les échantillons bootstrap sont identiques, la valeur de leur statistique est toujours égale à zéro, et il n'est pas possible de calculer un intervalle de confiance.

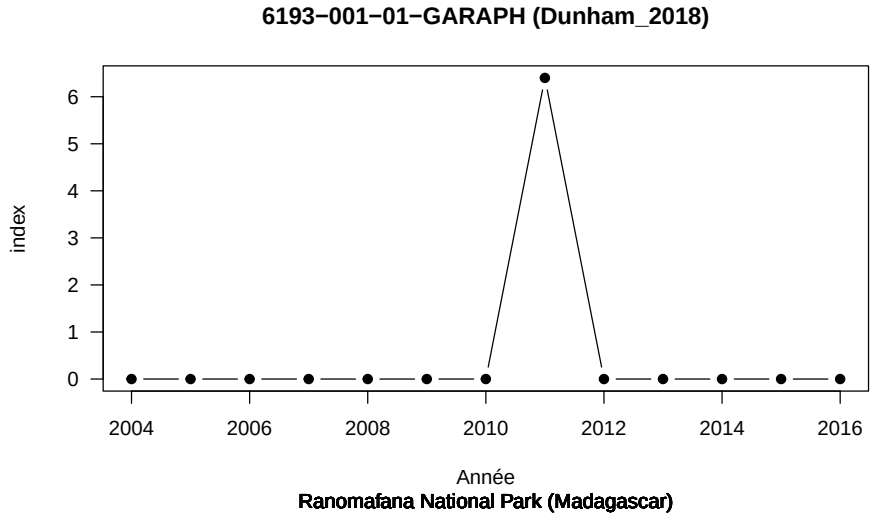
[12] Pour donner un ordre de grandeur, le calcul non parallélisé prend environ 20 minutes sur mon portable et tombe à 3 minutes et demi dans la version parallélisée.

```
par(mfrow = c(3, 1), mar = c(1, 4, 3, 1))
plotID("6196-001-01-EUCGRA")
plotID("6193-001-01-CARSPI")
plotID("6193-001-01-DORSTI")
```



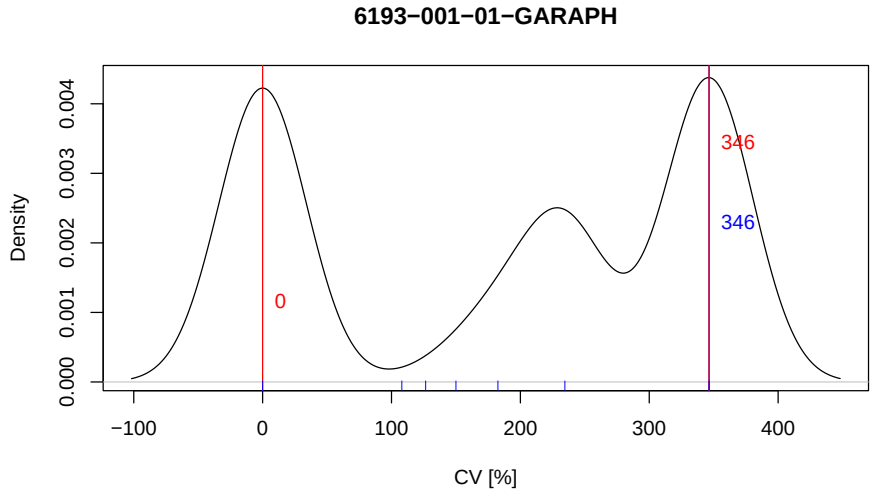
Les séries de code `Alpha_num` 6193 sont issues d'une étude de Amy E. Dunham et collaborateurs [10] pour *Carissa spinarum* et *Doratoxylon stipulatum* suivies dans le parc national de Ranomafana à Madagascar. La série de code 6196 sont des données non publiées partagées par Urs Kalbitzer, co-auteur de [17], pour *Eucalyptus grandis* suivi dans la parc national de Kibale en Ouganda. Il est donc parfaitement possible d'observer de longues séries temporelle constamment nulles dans les études ayant trait au *masting*.

Il y a un autre cas de figure où il est impossible de calculer un itervalle de confiance : c'est quand toutes les valeurs de la série sont nulles sauf une. Il n'y a qu'une série [10] du suivi de *Garcinia aphanophlebia* pendant 13 ans dans le parc national de Ranomafana à Madagascar qui illustre cette situation :



La distribution bootstrat possède trois modes principaux : un mode à zéro parce qu'il est probable ($p = 0.35$) de ne tirer que des valeurs nulles, un pic pour la valeur de la statistique parce qu'il est probable ($p = 0.38$) de ne tirer qu'une valeur non nulle et un pic intermédiaire parce qu'il est probable ($p = 0.19$) de tirer deux valeurs non nulles. Dans ce dernier cas on augmente la valeur de la moyenne et on diminue donc la valeur de CV. On comprend bien qu'aucun échantillon bootstrap ne pourra générer une valeur supérieure à la statistique observée, on est coincé pour le calcul de l'intervalle de confiance.

```
xx <- ms[ms$ID == "6193-001-01-GARAPH", "Value"]
CV.boot <- boot(xx, CV, R = 999)
plotboot(CV.boot, main = "6193-001-01-GARAPH", xlab = "CV [%]", rug = TRUE)
```



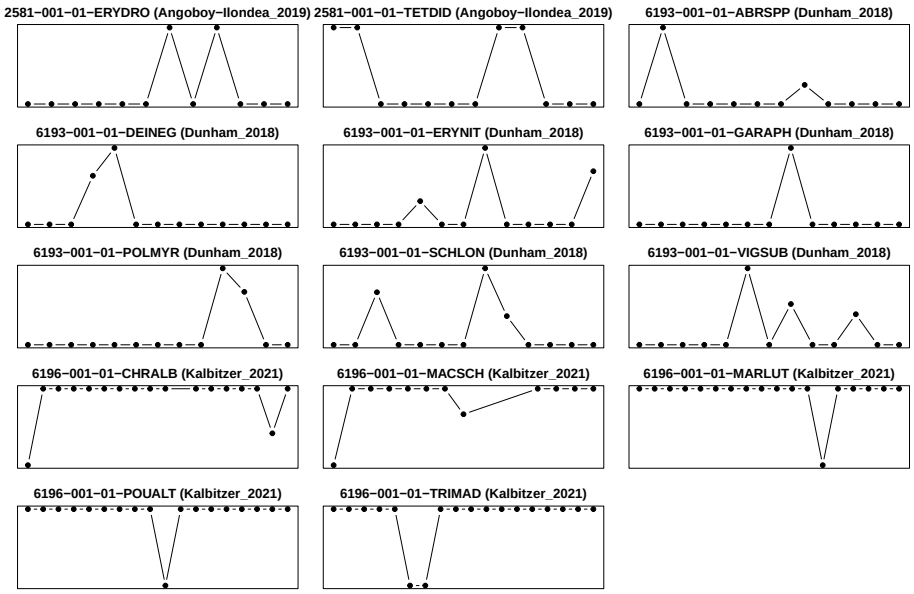
Il y a 14 séries pour lesquelles la borne inférieure de l'intervalle de confiance est nulle. Elles sont issues de trois études. La première de Bhely Angoboy Ilondea et collaborateurs [2] exploite les données historiques (1948-1957) de suivi phénologique de 3 642 arbres (158 espèces) de la forêt du Mayombe dans la

Réserve de Luki en République démocratique du Congo. La deuxième, déjà évoquée *supra*, est un suivi [10] dans le parc national de Ranomafana à Madagascar. La troisième est une étude non publiée partagée par Urs Kalbitzer and Colin Chapman, co-auteurs de [17], dans le parc national Kibale en Ouganda.

```
(zeroinf <- sort(CVpbis[!is.na(CVpbis$inf) & CVpbis$inf <= 0, "ID"]))
```

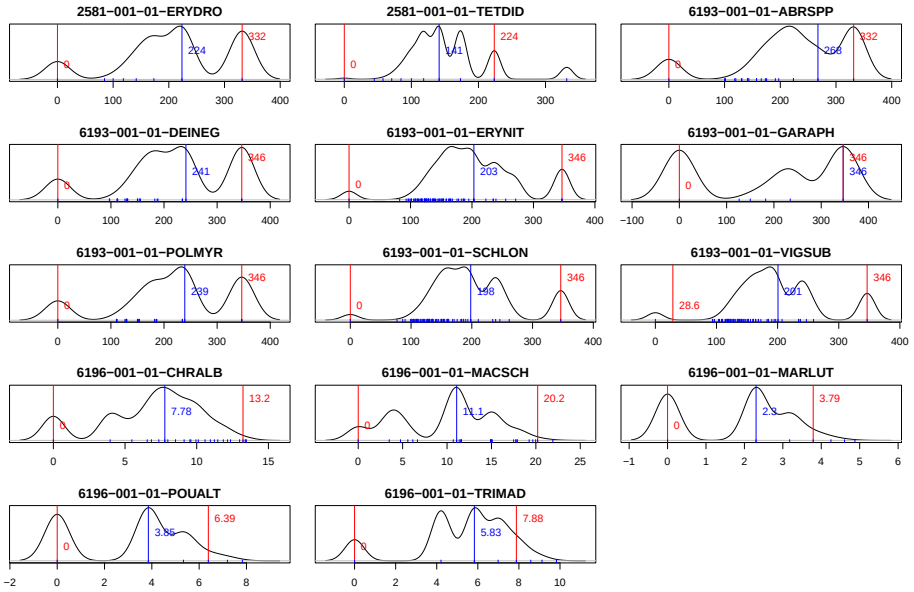
```
 [1] "2581-001-01-ERYDRO" "2581-001-01-TETDID" "6193-001-01-ABRSPP"
 [4] "6193-001-01-DEINEG" "6193-001-01-ERYNIT" "6193-001-01-GARAPH"
 [7] "6193-001-01-POLMYR" "6193-001-01-SCHLON" "6193-001-01-VIGSUB"
[10] "6196-001-01-CHRALB" "6196-001-01-MACSCH" "6196-001-01-MARLUT"
[13] "6196-001-01-POUALT" "6196-001-01-TRIMAD"
```

```
par(mfrow = c(5, 3), mar = c(1, 1, 2, 1), xaxt = "n", yaxt = "n")
for(i in seq_len(length(zeroinf))) plotID(zeroinf[i])
```



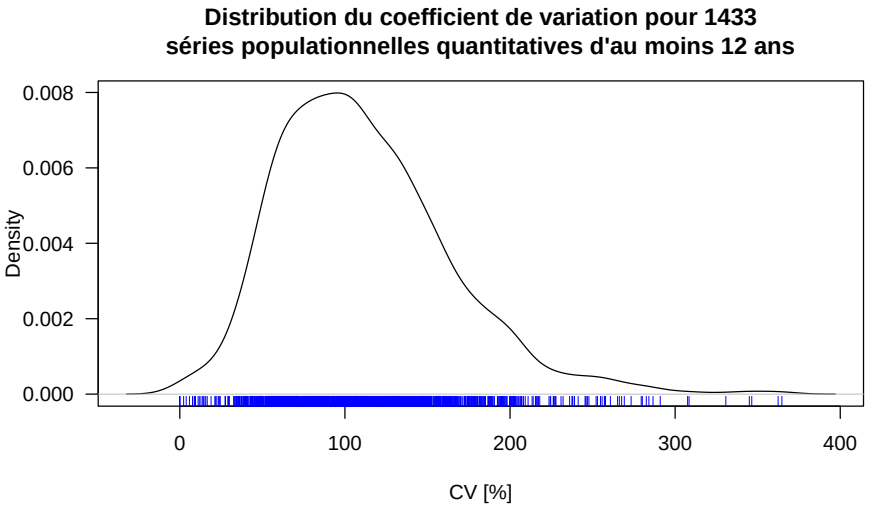
L'examen des distributions bootstrap montre que nos intervalles de confiance sont sujets à caution. Très souvent il n'y a que très peu de valeurs distinctes pour la valeur du coefficient de variation. Ce sont des séries trop peu documentées, avec trop peu de valeurs distinctes, pour que l'on puisse en tirer grand chose.

```
par(mfrow = c(5, 3), mar = c(3, 1, 2, 1), yaxt = "n")
for(i in seq_len(length(zeroinf))){
  xx <- ms[ms$ID == zeroinf[i], "Value"]
  CV.boot <- boot(xx, CV, R = 999)
  plotboot(CV.boot, main = zeroinf[i], xlab = "", rug = TRUE, ticksize = 0.05)
}
```



Les valeurs observées du coefficient de variation sont extrêmement variables puisqu'elles vont de 0 pour les séries constantes à plus de 300 % avec un mode autour de 100 % :

```
x <- CVpbis$est
main <- paste0("Distribution du coefficient de variation pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "CV [%]", main = main, las = 1)
rug(x, col = "blue")
```

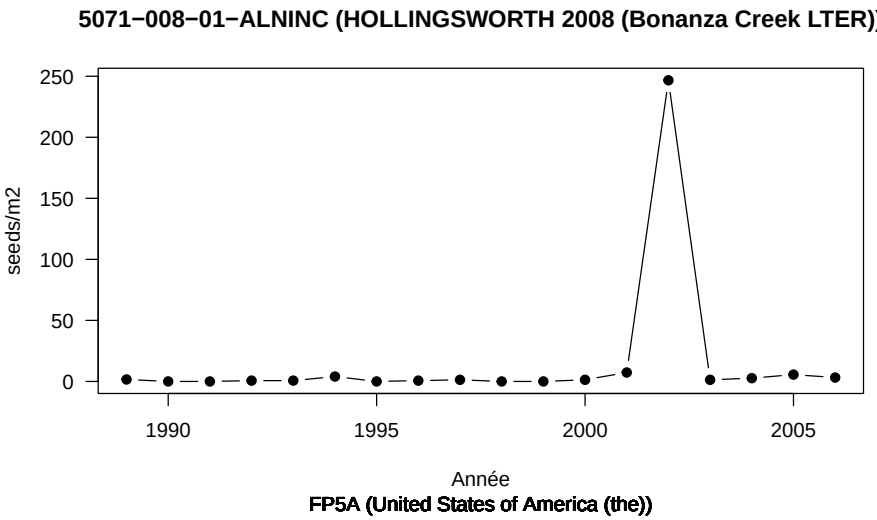


Le record de la série ayant la plus grande valeur du coefficient de variation (365 %) est détenu ici par une étude[13] du suivi de l'aulne blanc (*Alnus*

[13] Van Cleve, Keith; Chapin, F Stuart; Ruess, Roger. 2021. *Bonanza Creek LTER: Yearly Seedfall Summary from 1957 to Present in the Bonanza Creek Experimental Forest near Fairbanks, Alaska*, Bonanza Creek LTER - University of Alaska Fair-

incana) pendant 18 années au cours desquelles les graines n'ont été produite en masse qu'une seule fois :

```
plotID("5071-008-01-ALNINC")
```



3.1.6 Test de la synchronie parfaite avec CV_p et $\overline{CV_i}$

Ce test ne présente pas beaucoup d'intérêt puisqu'en pratique on préfère rejeter l'hypothèse nulle d'absence de synchronie pour justement pouvoir mettre en évidence l'existence d'une synchronisation entre les arbres. Je le mentionne quand même parce qu'il permet, d'une part, de bien comprendre le lien entre CV_p et $\overline{CV_i}$, et, d'autre part, d'introduire une représentation graphique exploitant les intervalles de confiance des statistiques. *Attention*, il ne fonctionne que grâce à l'invariance d'échelle du CV (cf. section 3.1.1 page 14).

Ce que nous entendons ici par synchronie *parfaite* entre les arbres c'est, intuitivement, que les courbes des efforts de reproduction relatifs (cf. section 1.2 page 4) sont toutes confondues. Plus précisément[14], notre hypothèse nulle, H_0, est qu'il existe une série temporelle de base, $\boldsymbol{b} = (b_1, \ldots, b_n)$, telle que les séries temporelles $\boldsymbol{x}_j$ de chaque arbre j s'en déduise par simple multiplication par un scalaire, λ_j, strictement positif propre à chaque arbre :

$$H_0 : \forall j \in \{1, \ldots, p\} \quad \exists \left\{ \boldsymbol{b} \in \mathbb{R}_+^n, \lambda_j \in \mathbb{R}_+^* \right\} \quad \boldsymbol{x}_j = \lambda_j \boldsymbol{b}$$

Sous H_0, que peut-on dire de $\overline{CV_i}$? L'invariance d'échelle du CV fait que tous les CV_i sont égaux entre-eux,

banks. **BNZ:14**, `http://www.lter.uaf.edu/data/data-detail/id/14`. `doi:10.6073/pasta/8ecb7a0eca30148914bcf9cc1472d327`

[14] Il convient d'être précis parce que l'on peut proposer d'autres définitions de la synchronie parfaite. Par exemple, il a été proposé [4] que la synchronie parfaite pouvait être caractérisée par le fait que toutes les séries temporelles dérivent les unes des autres par simple translation ($\boldsymbol{x}_j = \lambda_j + \boldsymbol{b}$). Je ne pense pas que cette définition soit très pertinente dans le cadre des études du *masting* parce que de telles séries dites en synchronie parfaite ne peuvent pas pas passer simultanément toutes par zéro, alors que c'est plutôt ce qui est attendu après une glandée. La définition adoptée ici est celle Carlos M. Herrera [21].

$$\mathrm{CV_i}(\boldsymbol{x}_j) = \mathrm{CV_i}(\lambda_j \boldsymbol{b}) = \mathrm{CV_i}(\boldsymbol{b})$$

et donc leur moyenne, $\overline{\mathrm{CV_i}}$, sera égale à cette valeur :

$$\overline{\mathrm{CV_i}} = \mathrm{CV_i}(\boldsymbol{b})$$

SOUS H_0, que peut-on dire de $\mathrm{CV_p}$? Il nous faut tout d'abord calculer le vecteur $\bar{\boldsymbol{x}}$ des moyennes annuelles (cf. section 2.4 page 13). Il est facile de voir qu'il est proportionnel au vecteur de base $\boldsymbol{b}$:

$$\overline{x_i} = \frac{1}{p}\sum_{j=1}^{p} x_{ij} = \frac{1}{p}\sum_{j=1}^{p} \lambda_j b_i = b_i \frac{1}{p}\sum_{j=1}^{p} \lambda_j = b_i \overline{\lambda} \iff \bar{\boldsymbol{x}} = \bar{\lambda}\boldsymbol{b}$$

De part l'invariance d'échelle du CV :

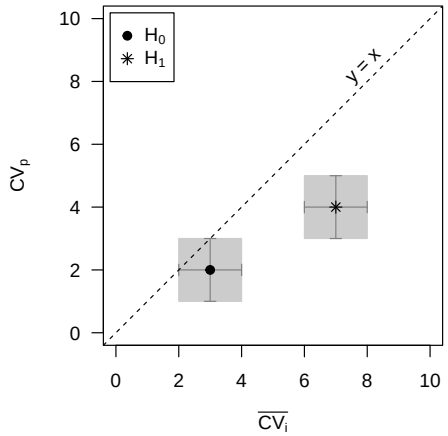
$$\mathrm{CV_p} = \mathrm{CV_i}(\bar{\lambda}\boldsymbol{b}) = \mathrm{CV_i}(\boldsymbol{b})$$

DONC, sous H_0, quand les arbres sont parfaitement synchronisés, il y a égalité entre la moyenne des CV individuels et le CV populationnel :

$$\overline{\mathrm{CV_i}} = \mathrm{CV_p}$$

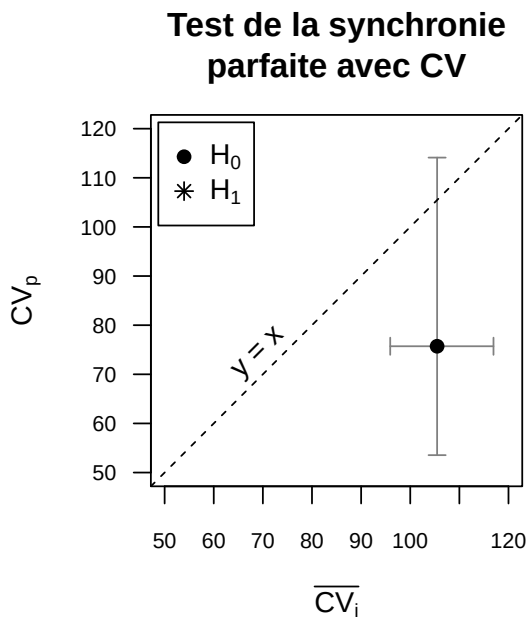
LE test sera conduit en confrontant les intervalles de confiance de $\overline{\mathrm{CV_i}}$ et $\mathrm{CV_p}$: s'ils sont chevauchants on ne pourra pas rejeter H_0. Graphiquement, on rejette l'hypothèse nulle quand le rectangle défini par les intervalles de confiance ne passe pas par la première bissectrice :

```
par(pty="s ", mar = c(5, 4, 0, 0) + 0.1)
plot(0:10, 0:10, type = "n", xlab = expression(bar(CV[i])),
     ylab = expression(CV[p]), las = 1)
gr <- grey(0.8)
rect(6, 3, 8, 5, col = gr, border = gr)
myarrows(6, 4, 8, 4) ; myarrows(7, 3, 7, 5) ; points(7, 4, pch = 8)
rect(2, 1, 4, 3, col = gr, border = gr)
myarrows(2, 2, 4, 2) ; myarrows(3, 1, 3, 3) ; points(3, 2, pch = 19)
abline(c(0, 1), lty = 2) ; text(8, 8, "y = x", srt = 45, pos = 3)
legend("topleft", inset = 0.02, pch = c(19, 8),
       legend = c(expression(H[0]), expression(H[1])))
```



Même si l'on ne pense pas que la statistique soit là pour la pluis d'étoiles[15], on utilisera par convention, moyen mnémotechnique et pour faciliter la lecture des graphiques, le caractère ∗ quand l'hypothèse nulle est rejetée. Voyons maintenant ce qu'il en est pour notre jeu de données :

```
par(pty="s ", mar = c(5, 4, 4, 0) + 0.1)
plot(CVm, CVp, type = "n", xlab = expression(bar(CV[i])),
     ylab = expression(CV[p]), las = 1, ylim = c(50, 120), xlim = c(50, 120),
     main = "Test de la synchronie\nparfaite avec CV", cex.axis = 0.75)
myarrows(ci.CVm[1], CVp, ci.CVm[2], CVp)
myarrows(CVm, ci.CVp[1], CVm, ci.CVp[2])
points(CVm, CVp, pch = 19)
abline(c(0, 1), lty = 2) ; text(70, 70, "y = x", srt = 45, pos = 3)
legend("topleft", inset = 0.02, pch = c(19, 8),
       legend = c(expression(H[0]), expression(H[1])))
```



Avec un risque de première espèce de 5 %, les données expérimentales ne nous permettent pas de rejeter l'hypothèse nulle d'une synchronie parfaite entre les arbres. On accepte donc H_0, faute de mieux, avec un risque de seconde espèce inconnu. Ce que l'on illustre en fait ici c'est la très faible puissance du test à cause de l'imprécision du CV. La simple visualisation des données (cf. section 1.2 page 4) montre que l'hypothèse d'une synchronie parfaite entre les arbres n'est pas tenable.

3.2 PV : variabilité proportionnelle

3.2.1 Définition, propriétés et implémentation

Cette statistique, PV, a été introduite comme étant l'acronyme de *Population Variability* en 2006 [19] puis de *Proportional Variability* en 2013 [20].

[15] Petit clin d'œil à Daniel Chessel: « [s]i vous pensez que la statistique n'est pas là pour la pluie d'étoiles : `options(show.signif.stars = FALSE)` », cf. `https://pbil.univ-lyon1.fr/R/pdf/tdr334.pdf`

Cette dernière appellation porte moins à confusion puisque le PV peut aussi bien être utilisé au niveau individuel (pour caractériser la variabilité inter-annuelle des arbres) qu'au niveau populationnel (pour caractériser la variabilité inter-annuelle des sites). Elle est basée sur une *métrique*, au sens propre du terme, permettant de calculer la distance entre deux valeurs x_{kj} et x_{lj} de la série temporelle $\boldsymbol{x}_j$ de l'arbre j. On peut la définir de plusieurs façons équivalentes :

$$d(x_{kj}, x_{lj}) = \begin{cases} 0 & \text{si} \quad x_{kj} = x_{lj} \\ \frac{|x_{kj} - x_{lj}|}{\max(x_{kj}, x_{lj})} & \text{si} \quad x_{kj} \neq x_{lj} \end{cases}$$

$$\iff d(x_{kj}, x_{lj}) = \begin{cases} 0 & \text{si} \quad x_{kj} = x_{lj} \\ \frac{\max(x_{kj}, x_{lj}) - \min(x_{kj}, x_{lj})}{\max(x_{kj}, x_{lj})} & \text{si} \quad x_{kj} \neq x_{lj} \end{cases}$$

$$\iff d(x_{kj}, x_{lj}) = \begin{cases} 0 & \text{si} \quad x_{kj} = x_{lj} \\ 1 - \frac{\min(x_{kj}, x_{lj})}{\max(x_{kj}, x_{lj})} & \text{si} \quad x_{kj} \neq x_{lj} \end{cases}$$

AVEC la dernière expression on voit bien que les valeurs de d sont comprises dans l'intervalle $[0, 1]$. Notons aussi que $d(0, 0)$ est défini comme un cas particulier de $x_{kj} = x_{lj}$. La statistique PV est la moyenne arithmétique de toutes les $\frac{1}{2}n(n-1)$ paires distinctes possibles (x_{kj}, x_{lj}) :

$$\mathrm{PV}(\boldsymbol{x}_j) = \frac{2}{n(n-1)} \sum_{k=1}^{n-1} \sum_{l=k+1}^{n} d(x_{kj}, x_{lj})$$

LES valeurs de PV sont donc comprises dans l'intervalle $[0, 1]$. Pour faciliter les comparaisons avec CV, je multiplierai ici PV par cent pour avoir des valeurs dans l'intervalle $[0, 100]$. Les valeurs de PV sont bornées alors celles de CV ne le sont pas à droite. Définissons la fonction `d()` qui calcule la distance utilisée par PV :

```
d <- function(x, y){
  if(isTRUE(all.equal(x, y, check.names = FALSE))){
    return(0)
  } else {
    return(abs(x - y)/max(x, y))
  }
}
d(0, 0)
```

```
[1] 0
```

```
d(0, 1)
```

```
[1] 1
```

Définissons la fonction `PV()` :

```
PV <- function(x, i, ...){
  n <- length(x)
  xx <- x[i] # échantillon bootstrap ou x si i manquant
  dists <- numeric(n*(n-1)/2) # distances entre toutes les paires
  ii <- 1
  for(k in seq_len(n - 1)){
    for(l in (k + 1):n){
      dists[ii] <- d(xx[k], xx[l])
      ii <- ii + 1
    }
  }
  return(100*mean(dists, ...))
}
PV(c(0, 0))
```

```
[1] 0
 PV(c(0, 1))
[1] 100
 PV(1:10)
[1] 50
 all.equal(PV(c(1:10)), PV(c(NA, 1:10), na.rm = TRUE)) # TRUE, OK
[1] TRUE
```

Notez la valeur de 50 % obtenue pour les séries des entiers de 1 à n[16]. Si on les divise par n on a une distribution uniforme des valeurs entre 0 et 1. C'est un bon point de repère pour interpréter les valeurs de PV : une valeur inférieure à 50 % correspond à une série moins variable qu'une distribution uniforme entre 0 et 1, une valeur de plus de 50 % à une série plus variable qu'une distribution uniforme entre 0 et 1.

```
 PV((1:10)/10)
[1] 50
 PV(runif(100, min = 0, max = 1))
[1] 46.61763
```

La statistique PV partage avec CV la propriété d'invariance d'échelle, on peut multiplier une série par un scalaire positif sans changer la valeur de la statistique :

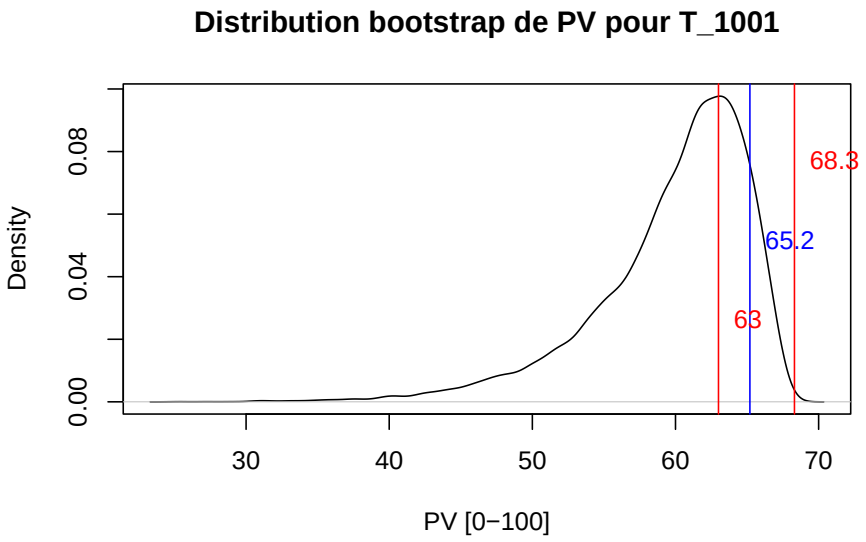
$$\forall \lambda \in \mathbb{R}_+^*, \forall j \in \{1, 2, \ldots, p\}, \forall \boldsymbol{x}_j \in \mathbb{R}_+^n : \mathrm{PV}(\lambda \boldsymbol{x}_j) = \mathrm{PV}(\boldsymbol{x}_j)$$

3.2.2 PV individuels

Nous reprenons ici l'exemple de l'arbre n° 1001 déjà utilisé pour le $\mathrm{CV_i}$ (cf. section 3.1.3 page 17). Le premier prix à payer pour utiliser $\mathrm{PV_i}$ au lieu de $\mathrm{CV_i}$, c'est le temps de calcul. Pour la série temporelle $\boldsymbol{x}_j$ de l'arbre j, le nombre de distances à calculer croît comme le carré du nombre d'années suivies, n. Même si les techniques de ré-échantillonnage intensif nous conduisent à répéter cette opération un grand nombre de fois, je ne pense pas que ce soit un problème en pratique pour les études relevant du *masting*. La plus longue série temporelle quantitative disponible relative au *masting* [18] disponible dans MASTREE+ [17] compte 69 années, soit 2346 paires distinctes à considérer, ce qui reste très raisonnable.

```
 set.seed(1)
 boot.PV <- boot(aps[ , 1], PV, R = 9999)
 save(boot.PV, file = "bootPV.Rda")


 load("bootPV.Rda")
 plotboot(boot.PV, main = "Distribution bootstrap de PV pour T_1001", xlab = "PV [0-100]")
```

[16] Dans [20] il est écrit que c'est pour des séries allant de 0 à n. Je pense que c'est une erreur, le plus court contre-exemple est la série de deux éléments $\boldsymbol{x} = \{0, 1\}$ dont le PV vaut 100.

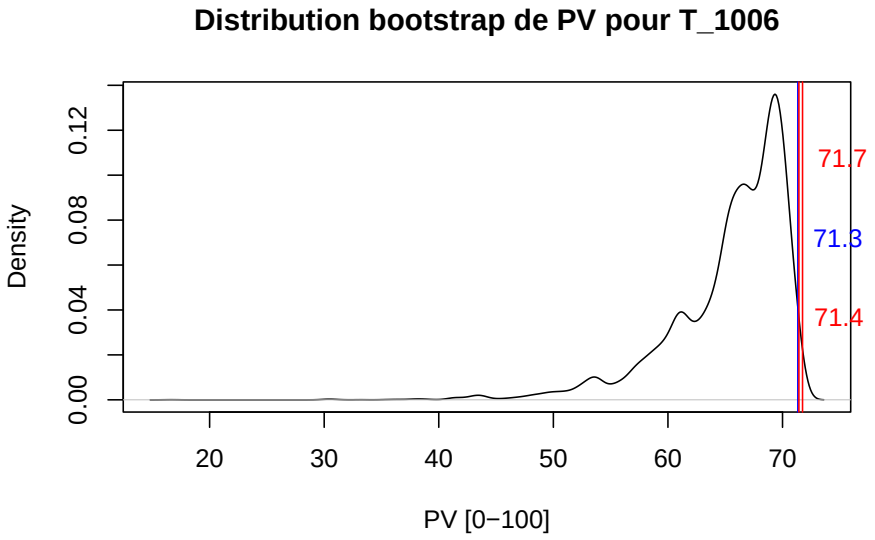


La statistique vaut 65.2 pour l'arbre n° 1001 avec un intervalle de confiance allant de 63 à 68.3, soit une amplitude relative de 8 %, bien plus précis que les 87 % que nous avions avec le CV_i. La distribution de bootstrap est très asymétrique à gauche. La raison de cette asymétrie est que quand on ré-échantillonne avec replacement il n'est pas du tout improbable de tirer plusieurs fois la même valeur x_{kj} et donc générer de zéros, $d(x_{kj}, x_{kj}) = 0$ lors du calcul de PV. Ce n'est généralement pas un problème, même si la fonction `boot.ci()` émet des messages d'avis[17]. L'intérêt du présent jeu de données est qu'il comporte des séries particulièrement toxiques, comme par exemple dans le cas de l'arbre n° 1006 :

```
set.seed(1)
boot.PV1006 <- boot(aps[ , 6], PV, R = 9999)
save(boot.PV1006, file = "bootPV1006.Rda")
```

```
load("bootPV1006.Rda")
plotboot(boot.PV1006, main = "Distribution bootstrap de PV pour T_1006", xlab = "PV [0-100]")
```

[17] Typiquement on aura : « *Warning : BCa Intervals used Extreme Quantiles* », « *Some BCa intervals may be unstable* » et « *extreme order statistics used as endpoints* ».



Comme on peut le constater l'intervalle de confiance est anormalement resserré et la valeur estimée ne tombe même pas à l'intérieur, il y a un sérieux problème. La distribution est tellement asymétrique que dans l'échantillon bootstrap il n'y a que 1 % des valeurs qui sont supérieures à la valeur de la statistique[18] :

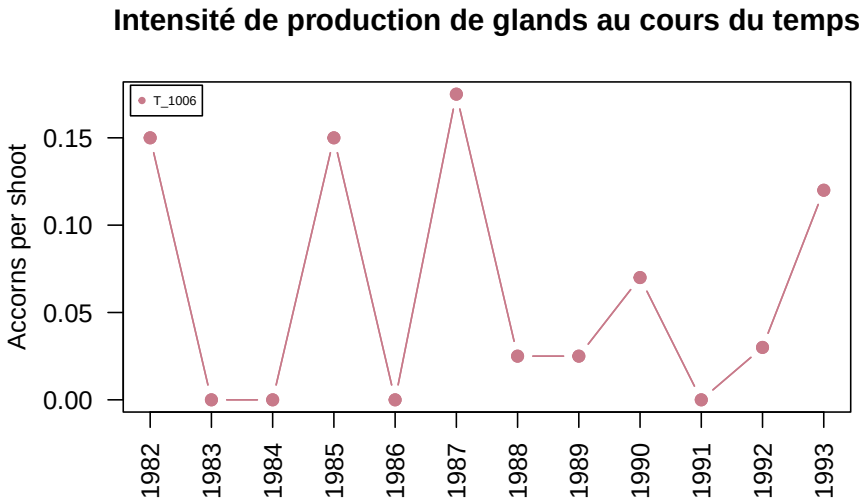
```
100*sum(boot.PV1006$t > boot.PV1006$t0)/nrow(boot.PV1006$t)
[1] 1.060106
```

On comprend bien le problème si on utilise la méthode des quantiles pour construire notre intervalle de confiance bootstrap : avec $\alpha = 0.05$ il faut que j'écarte 2.5 % des valeurs aux deux extrémités de ma distribution et donc ma borne supérieure sera nécessairement inférieure à la valeur de la statistique. Il n'y a rien à y faire, il est impossible de calculer un intervalle de confiance ici[19]. Dans quel cas de figure allons-nous rencontrer ce problème ? La série temporelle de l'arbre n° 1006 est extrêmement fluctuante et comporte pas moins de 4 valeurs nulles, soit les tiers des observations :

```
plot.tree(aps, "T_1006")
```

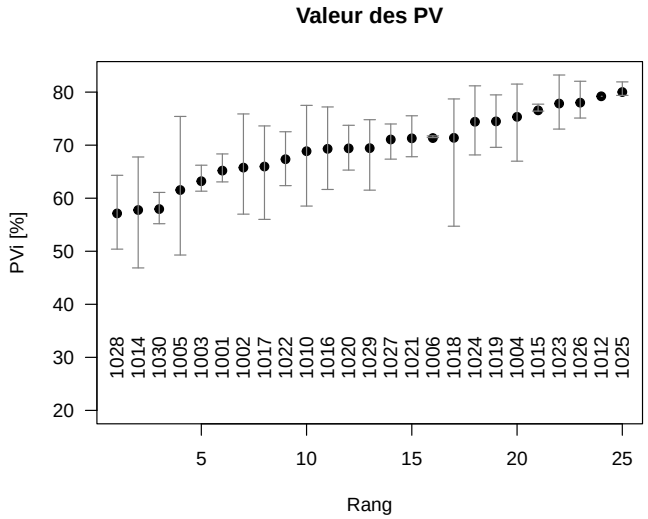
[18] Ne croyez pas pouvoir vous en tirer en augmentant la taille de l'échantillon bootstrap. Pour une série de n valeurs il y a $\binom{n}{2n-1}$ échantillons bootstrap possibles, soit avec $n = 12$: $\binom{12}{23} = 1352078$. J'ai poussé le bouchon jusqu'à un échantillon bootstrap de 10^6 tirages sans que cela ne change rien au fait que seul 1 % des valeurs sont supérieures à celle de la statistique.

[19] Non, ce n'est pas du jeu que de modifier la valeur de α en cours de route quand ça nous arrange.



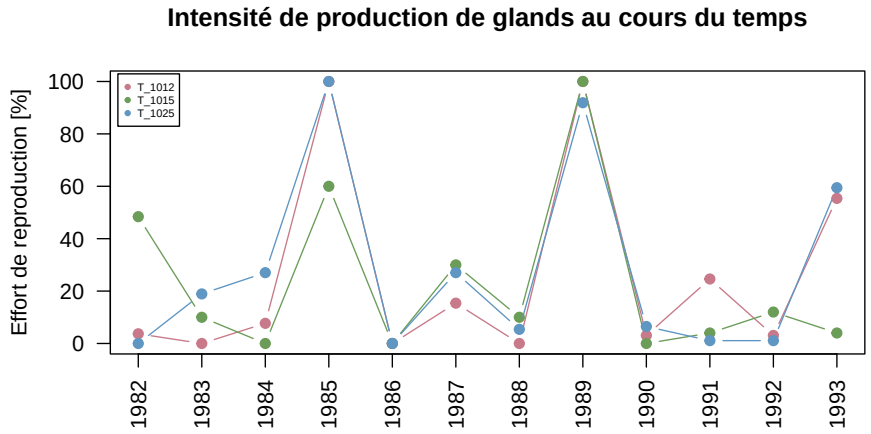
Si on représente les intervalles de confiance pour tous les arbres, on voit que l'on a un problème analogue à celui du n° 1006 pour les arbres n^{os} 1015, 1012 et 1025 :

```
PVi <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(PVi) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  PVi[i, "Tree_name"] <- colnames(aps)[i]
  boot.out <- boot(aps[ , i], PV, R = 9999)
  PVi[i, "est"] <- boot.out$t0
  ci <- boot.ci(boot.out, type = "bca")
  PVi[i, "inf"] <- ci$bca[4]
  PVi[i, "sup"] <- ci$bca[5]
}
save(PVi, file = "PVi.Rda")


load("PVi.Rda")
PVi <- PVi[order(PVi$est), ]
n <- nrow(PVi)
plot(1:n, PVi$est, pch = 19, ylim = c(20, max(PVi$sup)),
     las = 1, ylab = "PVi [%]", xlab = "Rang",
     main = "Valeur des PV")
with(PVi, myarrows(1:n, inf, 1:n, sup))
text(1:n, rep(30, n), substr(PVi$Tree_name, 3, 7), srt = 90, xpd = NA)
```



Ces arbres on un profil ressemblant à celui de l'arbre n° 1006, très fluctuant, et avec beaucoup de valeurs nulles :

```
plot.tree(apsr, c("T_1015", "T_1012", "T_1025"),
          ylab = "Effort de reproduction [%]")
```



Quand on dispose de données individuelles, l'impossibilité de pouvoir calculer un intervalle de confiance pour certains PV_i n'est pas forcément un problème parce que cela nous empêchera pas de calculer un intervalle de confiance pour $\overline{PV_i}$ (cf. *infra*). Quand on ne dispose que de données populationnelles, c'est plus problématique.

3.2.3 PV individuels moyen

C'est la moyenne arithmétique des PV_i (cf. section 2.3 page 13).

```
(PVm <- mean(PVi$est))
```

```
[1] 69.59274
```

```
(ci.PVm <- boot.ci(boot(PVi$est, moy, R = 9999), type = "bca")$bca[4:5])
```

```
[1] 66.86763 72.02562
```

3.2.4 PV populationnel

C'est le PV calculé sur la moyenne arithmétique des valeurs chaque année (cf. section 2.4 page 13).

```
xmoy <- rowMeans(aps)
boot.PVp <- boot(xmoy, PV, R = 9999)
save(boot.PVp, file = "bootPVp.Rda")
```

```
load("bootPVp.Rda")
boot.PVp$t0
```

```
[1] 60.17308
```

```
boot.ci(boot.PVp, type = "bca")$bca[4:5]
```

```
[1] 55.23109 65.55323
```

Pour calculer PV sur les séries populationnelles j'ai utilisé le script ci-après. Le fichier générique `genericPV.R` est en fait le même que celui utilisé pour le calcul de CV puisque le nom de la statistique à utiliser est donné dans la variable `Stat`.

batch/repartiteurPV.R

```
load("myws.Rda")

nproc <- 10 # Nombre de processeurs désiré
R <- 9999 # Taille échantillon bootstrap
IDs <- unique(ms$ID) ; n <- length(IDs)
# Découpage du travail en segments de iinf à isup
ii <- floor(seq(from = 1, to = n, length.out  = nproc + 1))
iinf <- ii[-length(ii)]
isup <- ii[-1] - 1
isup[length(isup)] <- n # Pour ne pas rater le dernier

genericLines <- readLines("genericPV.R")

for(i in seq_len(nproc)){
  # Création du fichier de script R
  fname <- paste0("genericPV-", i, ".R")
  codeR <- c(paste0("iproc <- ", i),
             paste0("iseq <- ", iinf[i], ":", isup[i]),
             paste0("Stat <- 'PV'"),
             paste0("R <- ", R),
             genericLines)
  writeLines(codeR, fname)
  # Création commande de lancement des scripts
  cmd <- paste0("nohup R CMD BATCH ", fname, "&")
  system(cmd)
}
```

Les temps de calculs sont beaucoup plus longs qu'avec CV : 150 heures de calcul cumulées et très variables d'un processus à l'autre, allant de 7 à 28 heures. La répartition équitable de la charge de travail entre les processus est difficile à réaliser parce que le temps de calcul est proportionnel au carré du nombre de points dans chaque série. Après exécution j'ai recollé les morceaux avec :

```
(fics <- dir(path = "batch", pattern = glob2rx("tab-*PV.Rda"), full.names = TRUE))
```

```
 [1] "batch/tab-10PV.Rda" "batch/tab-1PV.Rda"  "batch/tab-2PV.Rda"
 [4] "batch/tab-3PV.Rda"  "batch/tab-4PV.Rda"  "batch/tab-5PV.Rda"
 [7] "batch/tab-6PV.Rda"  "batch/tab-7PV.Rda"  "batch/tab-8PV.Rda"
[10] "batch/tab-9PV.Rda"
```

```
for(i in seq_len(length(fics))) load(fics[i])
(tabs <- ls(pattern = glob2rx("tab-*PV")))
```

```
[1] "tab-10PV" "tab-1PV"  "tab-2PV"  "tab-3PV"  "tab-4PV"  "tab-5PV"  "tab-6PV"
[8] "tab-7PV"  "tab-8PV"  "tab-9PV"
```

```
PVp <- do.call("rbind", lapply(tabs, get))
save(PVp, file = "batch/PVp.Rda")
```

L'inspection du tableau des résultats nous montre que le calcul de PV a toujours été possible puisqu'il n'y a aucune valeur manquante, mais que le calcul de l'intervalle de confiance n'a pas été possible pour 7 séries :

```
load("batch/PVp.Rda")
PVp[is.na(PVp$est), "ID"]
```

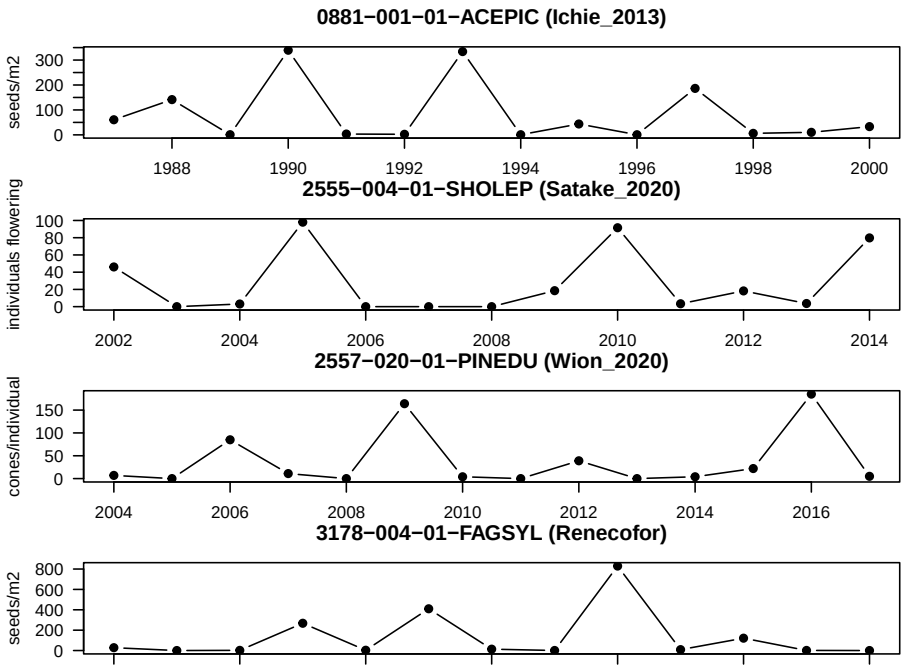
```
character(0)
```

```
PVp[is.na(PVp$inf) | is.na(PVp$sup), "ID"]
```

```
[1] "6196-001-01-EUCGRA" "0881-001-01-ACEPIC" "2555-004-01-SHOLEP"
[4] "2557-020-01-PINEDU" "3178-004-01-FAGSYL" "6193-001-01-CARSPI"
[7] "6193-001-01-DORSTI"
```

On retrouve les 3 séries constamment égales à zéro que l'on avait déjà rencontrées lors du calcul de CV plus les 4 séries suivantes :

```
par(mfrow = c(4, 1), mar = c(1, 4, 3, 1))
plotID("0881-001-01-ACEPIC")
plotID("2555-004-01-SHOLEP")
plotID("2557-020-01-PINEDU")
plotID("3178-004-01-FAGSYL")
```



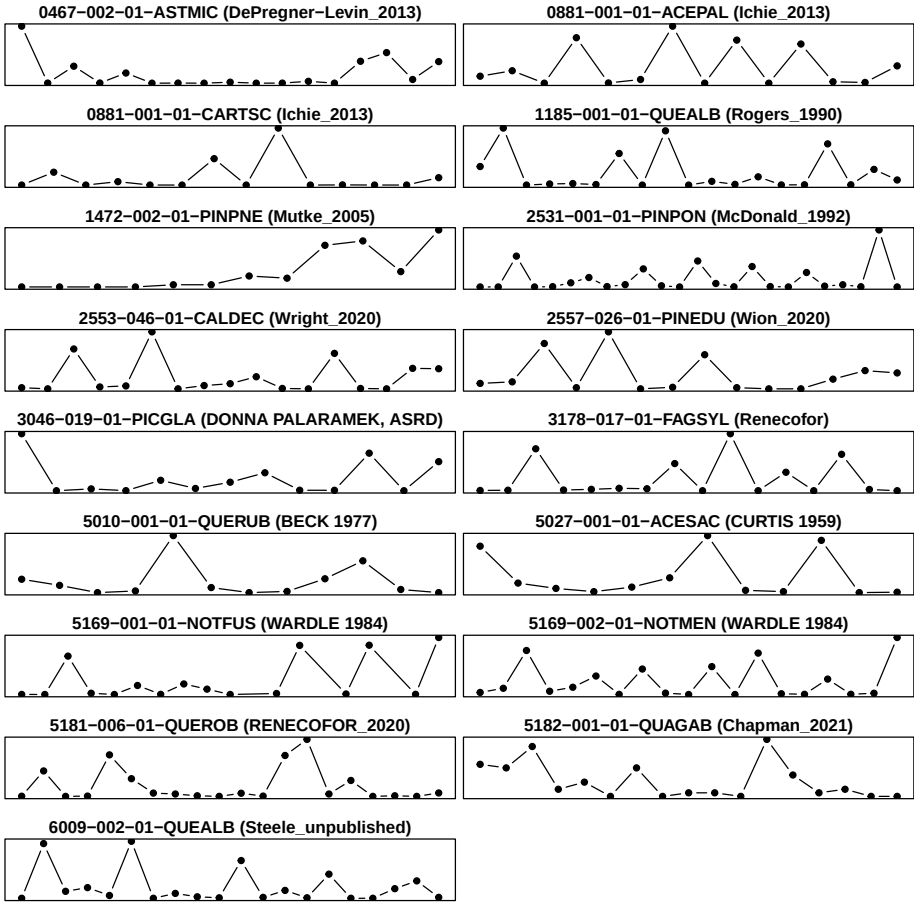
La première série est un suivi [23] pendant 14 ans de *Acer pictum* dans la réserve forestière d'Ogawa au Japon. La seconde est un suivi [46] pendant

13 ans de *Shorea leprosula* dans la réserve forestière de Pasoh en Malaisie. La troisième est un suivi [55] pendant 14 ans de *Pinus edulis* l'Ouest des États Unis entre Grand junction et Silver city. La dernière est un suivi[20] pendant 13 ans du hêtre commun (*Fagus sylvatica*) dans la forêt de Soulan en France. Mais ce n'est pas tout, comme on l'avait vu avec l'arbre n° 1006 pour les séries individuelles, il faut encore s'assurer que la valeur estimée tombe bien dans l'intervalle de confiance, et on trouve 17 séries problématiques :

```
(prbl <- PVp[!is.na(PVp$inf) & !is.na(PVp$sup) &
               !(PVp$est < PVp$sup & PVp$est > PVp$inf), "ID"])
```

```
 [1] "0467-002-01-ASTMIC" "0881-001-01-ACEPAL" "0881-001-01-CARTSC"
 [4] "1185-001-01-QUEALB" "1472-002-01-PINPNE" "2531-001-01-PINPON"
 [7] "2553-046-01-CALDEC" "2557-026-01-PINEDU" "3046-019-01-PICGLA"
[10] "3178-017-01-FAGSYL" "5010-001-01-QUERUB" "5027-001-01-ACESAC"
[13] "5169-001-01-NOTFUS" "5169-002-01-NOTMEN" "5181-006-01-QUEROB"
[16] "5182-001-01-QUAGAB" "6009-002-01-QUEALB"
```

```
par(mfrow = c(9, 2), mar = c(1, 0.5, 1.5, 0))
for(i in seq_len(length(prbl))) plotID(prbl[i], yaxt = "n", yalb = "", xaxt = "n")
```



Ce qui semble caractériser ces séries c'est une forte variabilité associée à la présence de nombreuses valeurs nulles. Si on exclue les trois cas un peu

[20] Site HET09 du suivi RENECOFOR.

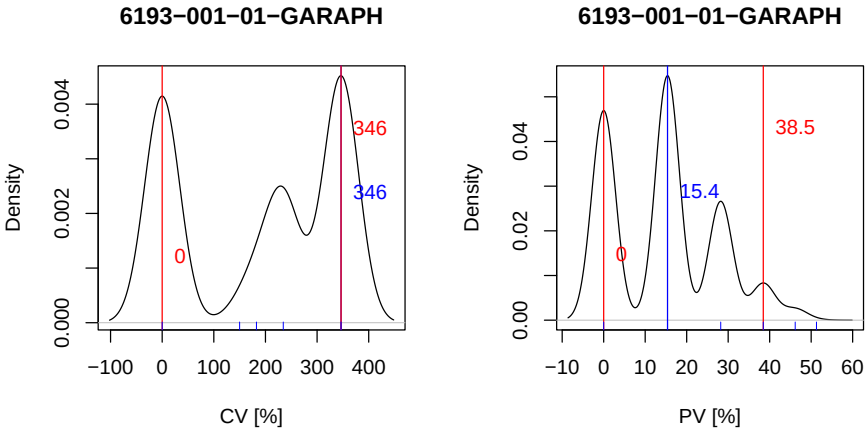
pathologiques des séries constantes[21], cela fait donc 21 séries sur 1430 pour lesquelles l'intervalle de confiance de PVp ne peut être déterminé, soit une proportion de 1.5 %. On ne pourra pas faire mieux qu'une estimation ponctuelle dans ces cas de figure et dire que la « vraie » valeur se trouve quelque part entre cette estimation ponctuelle et 100 %.

Les 17 séries qui posent problème pour le calcul de l'intervalle de confiance pour PV sont les suivantes. Un suivi [9] pendant 17 ans de *Astragalus microcymbus* à *South Beaver Creek Site 15* au Colorado aux USA. Un suivi [23] pendant 14 ans de *Acer palmatum* dans la réserve forestière d'Ogawa au Japon. Un suivi [23] pendant 14 ans de *Carpinus tschonoskii* dans la réserve forestière d'Ogawa au Japon. Un suivi [44] pendant 19 ans de *Quercus alba* dans la forêt expérimentale de Sylamore aux USA. Un suivi [40] pendant 12 ans de *Pinus pinea* à *Quintanilla* en Espagne. Un suivi [37] pendant 24 ans de *Pinus ponderosa* en Californie aux USA. Un suivi [56] pendant 17 ans de *Calocedrus decurrens* en Californie aux USA. Un suivi [55] pendant 14 ans de *Pinus edulis* dans l'Ouest des USA. Un suivi[22] pendant 13 ans de *Picea glauca* dans la *Breeding region E3* au Canada. Un suivi[23] pendant 16 ans de *Fagus sylvatica* à Les Ventes St Rémy en France. Un suivi[24] pendant 12 ans de *Quercus rubra* aux USA. Un suivi [7] pendant 12 ans de *Acer saccharum* dans le Wisconsin aux USA. Un suivi [52] pendant 16 ans de *Nothofagus fusca* en Nouvelle-Zélande. Un suivi [52] pendant 19 ans de *Nothofagus menziesii* en Nouvelle-Zélande. Un suivi[25] pendant 20 ans de *Quercus robur* dans le département du Nord en France. Un suivi[26] pendant 17 ans de *Quassia gabonensis* à *Ngel Nyaki* au Nigeria. Un suivi[27] pendant 20 ans de *Quercus alba* à *Steele* aux USA.

On reprend l'exemple extrême de la série qui ne possède qu'une valeur non nulle dont avait déjà vu qu'il était impossible de calculer un intervalle de confiance pour CV. La distribution de bootstrap pour PV est intéressante : même si elle comporte pas beaucoup de valeurs distinctes, cette fois on est capable de « sortir de la boîte » en ayant des valeurs supérieures à celle de la statistique. C'est un net avantage de PV par rapport à CV :

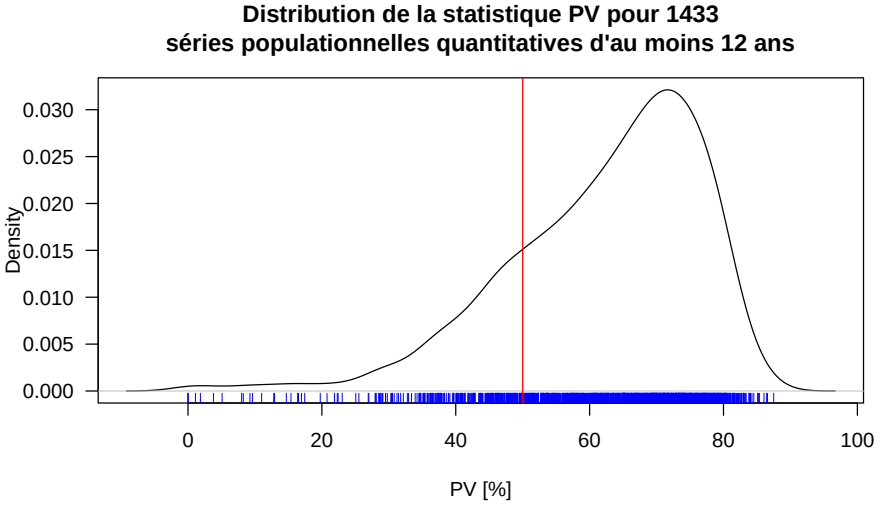
```
xx <- ms[ms$ID == "6193-001-01-GARAPH", "Value"]
par(mfrow = c(1, 2))
CV.boot <- boot(xx, CV, R = 999)
plotboot(CV.boot, main = "6193-001-01-GARAPH", xlab = "CV [%]", rug = TRUE)
PV.boot <- boot(xx, PV, R = 999)
plotboot(PV.boot, main = "6193-001-01-GARAPH", xlab = "PV [%]", rug = TRUE)
```

[21] On ne peut déterminer l'intervalle de confiance d'*aucune* statistique dans ces cas.
[22] Données non publiées référencées « DONNA PALARAMEK, ASRD » dans MASTREE+.
[23] Site HET76 du suivi RENECOFOR.
[24] Données non publiées référencées « BECK 1977 » dans MASTREE+.
[25] Site CHP59 du suivi RENECOFOR.
[26] Données non publiées partagées par Colin A. Chapman, co-auteur de [17].
[27] Données non publiées partagées par Michael A. Steele, co-auteur de [17].



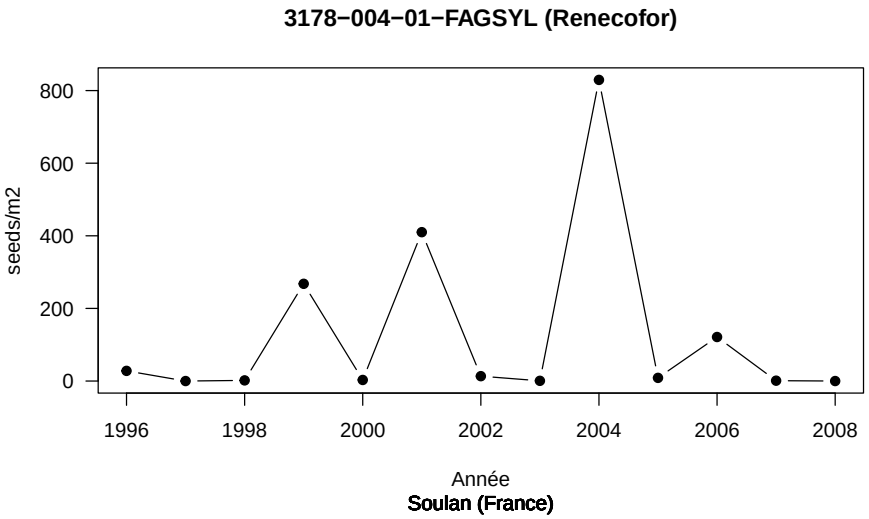
Les valeurs observées de PV vont de 0 pour les séries constantes à 87.5 % avec un mode autour de 75 %. La valeur de 50 % en rouge est la valeur pour une distribution uniforme entre 0 et une constante. Les séries temporelles relatives au *masting* sont donc majoritairement plus variables qu'une telle distribution uniforme :

```
x <- PVp$est
main <- paste0("Distribution de la statistique PV pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "PV [%]", main = main, las = 1)
rug(x, col = "blue")
abline(v = 50, col = "red")
```



Le record de la série ayant la plus grande valeur de PV (87.5 %) est détenu ici par une étude[28] du suivi du hêtre commun (*Fagus sylvatica*) pendant 13 ans dans la forêt communale de Soulan sise dans le département de l'Ariège en France :

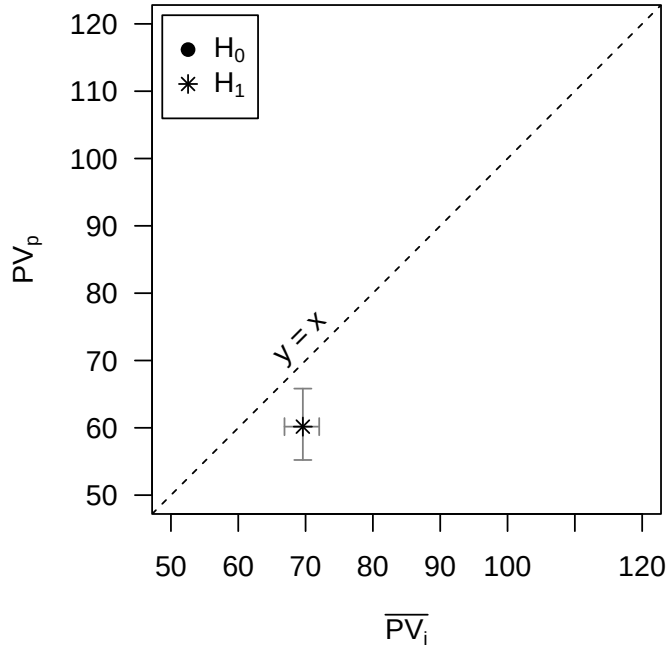
[28] Site HET09 du suivi RENECOFOR.



3.2.5 Test de la synchronie parfaite avec $\mathbf{PV_p}$ et $\overline{\mathbf{PV_i}}$

ON rejette ici l'hypothèse nulle de synchronie parfaite entre les arbres, on a gagné en puissance avec la statistique du PV par rapport à celle du CV.

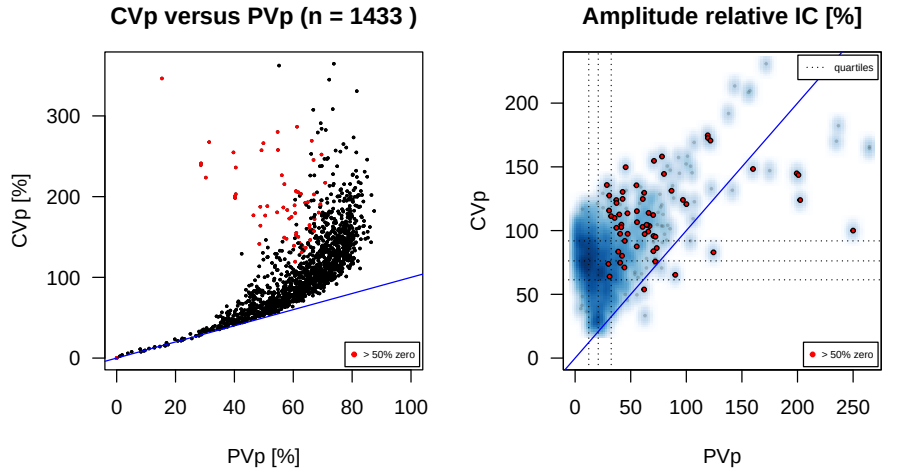
```
load("PVp.Rda")
par(pty="s ", mar = c(5, 4, 0, 0) + 0.1)
plot(PVm, PVp, type = "n", xlab = expression(bar(PV[i])),
     ylab = expression(PV[p]), las = 1, ylim = c(50, 120), xlim = c(50, 120))
myarrows(ci.PVm[1], PVp, ci.PVm[2], PVp)
myarrows(PVm, ci.PVp[1], PVm, ci.PVp[2])
points(PVm, PVp, pch = 8)
abline(c(0, 1), lty = 2) ; text(70, 70, "y = x", srt = 45, pos = 3)
legend("topleft", inset = 0.02, pch = c(19, 8),
       legend = c(expression(H[0]), expression(H[1])))
```



3.3 CVp *vs.* PVp

Pour comparer CVp et PVp on calcule l'amplitude relative des intervalles de confiance, c'est à dire la borne supérieure moins la borne inférieure divisée par la valeur de la statistique et exprimée en pourcentage.

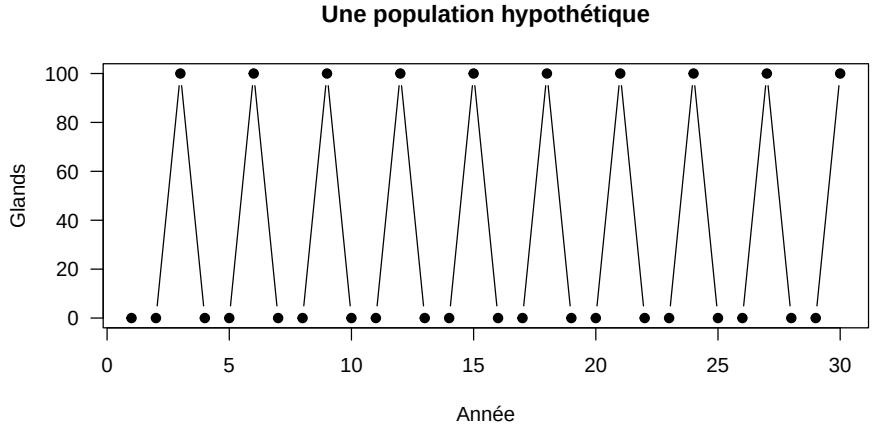
```
load("batch/PVp.Rda") ; load("batch/CVpbis.Rda") ; CVp <- CVpbis
x <- PVp$est ; y <- CVp$est
PVp$col <- "black"
for(i in 1:nrow(PVp)){
  the_ID <- PVp[i, "ID"]
  vals <- ms[ms$ID == the_ID, "Value"]
  pz <- sum(vals <= 0)/length(vals)
  if(!is.na(pz) & pz > 0.5) PVp[i, "col"] <- "red"
}
par(pty = "s", mfrow = c(1, 2), mar = c(4, 4, 3, 1))
plot(x, y, pch = 19, xlim = c(0, 100), cex = 0.25,
     las = 1, ylab = "CVp [%]", xlab = "PVp [%]",
     main = paste("CVp versus PVp (n =", sum(!is.na(x)), ")"))
points(x, y, pch = 19, cex = 0.25, col = PVp$col)
legend("bottomright", inset = 0.01, legend = "> 50% zero", pch = 19, col = "red", cex = 0.5)
abline(c(0, 1), col = "blue")
rcirPV <- 100*(PVp$sup - PVp$inf)/PVp$est
rcirCV <- 100*(CVp$sup - CVp$inf)/CVp$est
smoothScatter(rcirPV, rcirCV, pch = 19, cex = 0.25, col = rgb(0.25, 0.25, 0.25, 0.25),
     main = "Amplitude relative IC [%]", las = 1, xlab = "PVp", ylab = "CVp",
     ylim = c(0, max(rcirCV, na.rm = TRUE)))
ired <- which(PVp$col == "red")
points(rcirPV[ired], rcirCV[ired], pch = 21, bg = "red", cex = 0.5)
abline(c(0, 1), col = "blue")
legend("bottomright", inset = 0.01, legend = "> 50% zero", pch = 19, col = "red", cex = 0.5)
legend("topright", inset = 0.01, legend = "quartiles", lty = 3, cex = 0.5, bg = "white")
abline(v = quantile(rcirPV, c(0.25, 0.5, 0.75), na.rm = TRUE), lty = 3)
abline(h = quantile(rcirCV, c(0.25, 0.5, 0.75), na.rm = TRUE), lty = 3)
```



La surprise ici c'est que pour des valeurs faibles des statistiques, disons moins de 50 %, les valeurs de CVp et de PVp sont très proches. Pour les faibles niveaux de variabilité on peut donc lire PVp comme CVp. Ce n'est que pour des niveaux de variabilité élevés qu'ils diffèrent puisque PVp est plafonné à 100 %. On voit également ici la sensibilité de CVp à la présence de valeurs nulles, avec le cas extrême en haut à gauche de la série qui ne comporte qu'une valeur non nulle, dèjà détaillée précédemment. Du point de vue de l'amplitude relative des intervalles de confiance, il y a un net avantage à utiliser PVp puisque la médiane est à 21 % et la moitié des amplitudes est comprise entre 12 % et 32 % alors qu'avec CVp la médiane est à 76 % et la moitié des amplitudes est comprise entre 61 % et 92 %. Il y a donc un gain de précision non négligeable obtenu quand on passe de CVp à PVp.

Pour illustrer l'intérêt de PVp par rapport à CVp, j'ai imaginé une population hypothétique qui ne ferait une glandée que tous les trois ans :

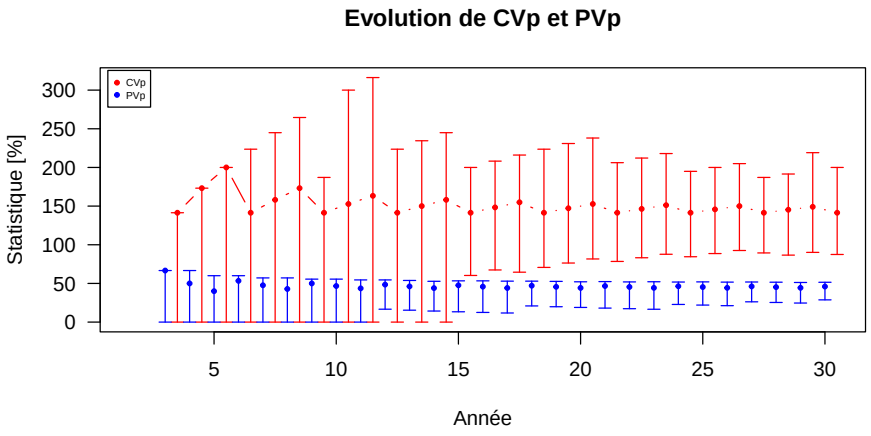
```
ny <- 30
sim <- as.data.frame(cbind(1:ny, rep(c(0, 0, 100), le = ny)))
plot(sim$V1, sim$V2, pch = 19, type = "b", las = 1, xlab = "Année", ylab = "Glands",
     main = "Une population hypothétique")
```



Si on calcule CVp et PVp au fur et à mesure que le temps s'écoule on voit que CVp a des valeurs très fluctuantes au début et qu'il faut attendre 15 ans avant que la borne inférieure de l'intervalle de confiance ne soit plus nulle. Par contraste, PVp est beaucoup plus stable dès le début et la borne inférieure de l'intervalle de confiance n'est plus nulle dès 12 ans.

```
sim$CV <- sim$CVinf <- sim$CVsup <- NA
sim$PV <- sim$PVinf <- sim$PVsup <- NA
for(i in 3:ny){
  x <- sim[1:i, 2]
  sim[i, "CV"] <- CV(x)
  ci <- boot.ci( boot(x, CV, R = 9999), type = "bca")
  sim[i, "CVinf"] <- ci$bca[4] ; sim[i, "CVsup"] <- ci$bca[5]
  sim[i, "PV"] <- PV(x)
  ci <- boot.ci( boot(x, PV, R = 9999), type = "bca")
  sim[i, "PVinf"] <- ci$bca[4] ; sim[i, "PVsup"] <- ci$bca[5]
}
save(sim, file = "sim.Rda")
```

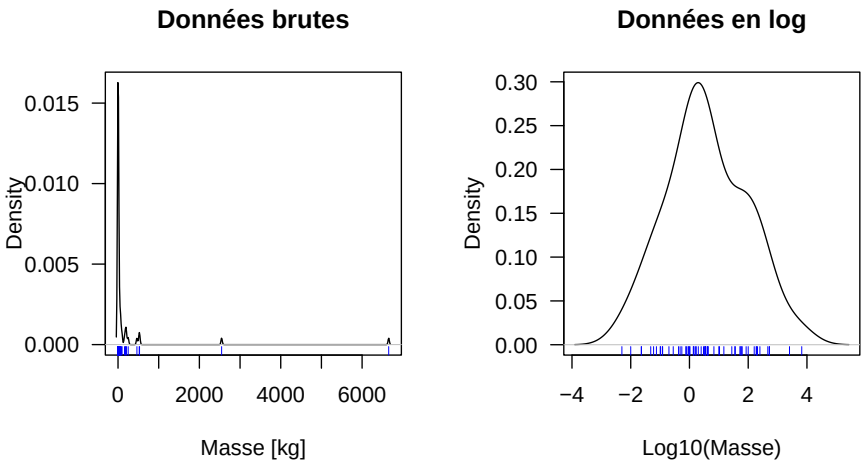
```
load("sim.Rda")
plot(sim$V1 + 0.5, sim$CV, type = "b", ylim = c(0, max(sim$CVsup, na.rm = TRUE)), pch = 19,
     col = "red", cex = 0.5, xlab = "Année", las = 1, ylab = "Statistique [%]",
     main = "Evolution de CVp et PVp")
points(sim$V1, sim$PV, pch = 19, col = "blue", cex = 0.5)
for(i in 3:ny){
  myarrows(i + 0.5, sim$CVinf[i], i + 0.5, sim$CVsup[i], col = "red")
  myarrows(i, sim$PVinf[i], i, sim$PVsup[i], col = "blue")
}
legend("topleft", inset = 0.01, legend = c("CVp", "PVp"), col = c("red", "blue"),
       pch = 19, cex = 0.5)
```



3.4 SDL : écart type des logs

Une méthode très employée pour stabiliser la variance quand on travaille avec des distributions très asymétriques à droite est de travailler avec le logarithme. Par exemple, avec les données sur la masse corporelles de 62 mammifères terrestres [1] disponible dans le jeu de données `mammals` de la bibliothèque standard `MASS` [51], on voit que les valeurs ont une distribution approximativement log-normale sur 6 ordres de grandeur :

```
library(MASS) ; data(mammals)
with(mammals, {
  x <- body
  par(mfrow = c(1, 2))
  plot(density(x), xlab = "Masse [kg]", main = "Données brutes", las = 1)
  rug(x, col = "blue")
  plot(density(log10(x)), xlab = "Log10(Masse)", main = "Données en log", las = 1)
  rug(log10(x), col = "blue")
})
```



Comme le faisait remarquer Lewontin [34], les propriétés du logarithme et de la variance font que la variance (ou l'écart-type) calculé sur les logarithmes possède comme CV et PV la propriété d'invariance d'échelle :

$$\forall \lambda \in \mathbb{R}_+^*, \forall \boldsymbol{x}_j \in (\mathbb{R}_+^*)^n : s^2_{\log \lambda \boldsymbol{x}_j} = s^2_{\log \lambda + \log \boldsymbol{x}_j} = s^2_{\log \lambda} + s^2_{\log \boldsymbol{x}_j} = s^2_{\log \boldsymbol{x}_j}$$

Malheureusement, les séries temporelles du *masting* sont riches en valeurs nulles pour lesquelles la fonction logarithmique n'est pas définie. Un pis-aller consiste à utiliser la transformation $\log(1 + \boldsymbol{x})$ pour balayer discrètement les zéros sous le tapis, mais ce n'est que cautère sur jambe de bois car on perd la propriété d'invariance d'échelle [54, 3, 36, 15]. On se contentera de dire ici que la perte de l'invariance d'échelle fait que l'on ne se penchera pas plus avant sur cette approche[29] :

```
 sdn(log(1:10))
[1] 0.6954075
 sdn(log(pi*(1:10)))
[1] 0.6954075
 sdn(log(1 + 0:10))
[1] 0.7104347
 sdn(log(1 + pi*(0:10)))
[1] 0.989225
```

[29] Mais si par chance vous travaillez avec des variables dont par essence les valeurs sont strictement positives, par exemple des variables morphométriques, c'est une approche qui mérite d'être creusée.

3.5 SDR : écart-type des rangs

Une autre technique bien connue pour stabiliser la variance consiste à transformer les valeurs par leur rang dans le jeu de données. Le calcul des rangs se fait globalement pour ne pas perdre l'information sur les différences d'effort de reproduction entre les arbres[30]. On utilise l'option `"min"` pour l'argument `ties.method` de la fonction `rank()` pour que les valeurs nulles soient toutes de rang 1. On notera $\boldsymbol{r}$ la matrice des rangs déduite ainsi de $\boldsymbol{x}$. La statistique SDR est définie par l'écart type calculé sur les rangs :

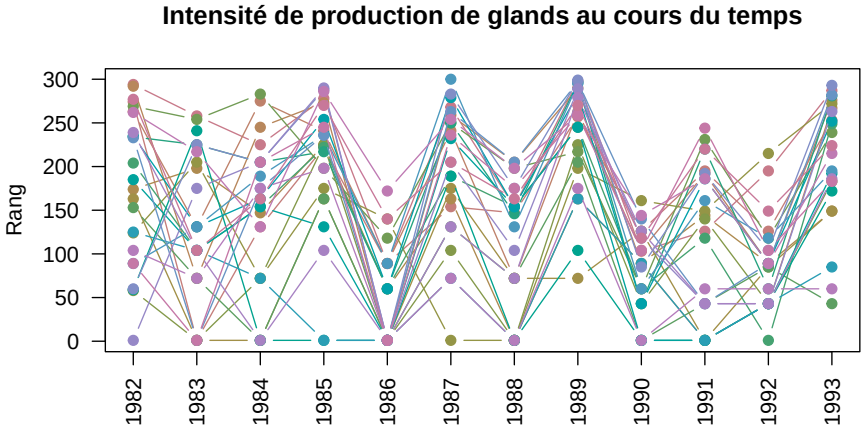
$$\mathrm{SDR}(\boldsymbol{x}_j) = s_{\boldsymbol{r}_j}$$

La statistique SDR partage avec CV et PV la propriété d'invariance d'échelle puisque les rangs sont inchangés après multiplication :

$$\forall \lambda \in \mathbb{R}_+^*, \forall j \in \{1, 2, \ldots, p\}, \forall \boldsymbol{x}_j \in \mathbb{R}_+^n : \mathrm{SDR}(\lambda \boldsymbol{x}_j) = \mathrm{SDR}(\boldsymbol{x}_j)$$

Représentons les données transformées :

```
apsrk <- matrix(rank(as.vector(aps), ties.method = "min"),
                nrow = nrow(aps), ncol = ncol(aps))
apsrk <- as.data.frame(apsrk)
colnames(apsrk) <- colnames(aps)
rownames(apsrk) <- rownames(aps)
plot.tree(apsrk, colnames(apsrk), leg = FALSE,
          ylab = "Rang")
```



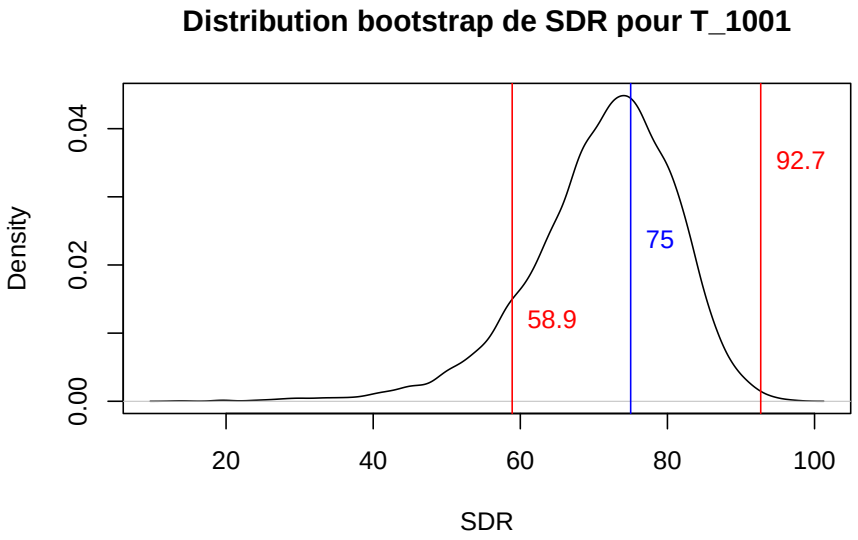
3.5.1 SDR individuels

Nous reprenons ici l'exemple de l'arbre n° 1001 déjà utilisé pour le $\mathrm{CV_i}$ (cf. section 3.1.3 page 17).

[30] De plus, si on calculait les rangs arbre par arbre, l'écart type des rangs serait le même pour tous les arbres. Notez également que cette approche ne serait pas possible, en général, avec des séries disponibles au niveau populationnel : cela n'aurait pas de sens d'inter-classer des valeurs exprimées dans des unités potentiellement différentes.

```
set.seed(1)
boot.SDR <- boot(apsrk[ , 1], sdn, R = 9999)
save(boot.SDR, file = "bootSDR.Rda")
```

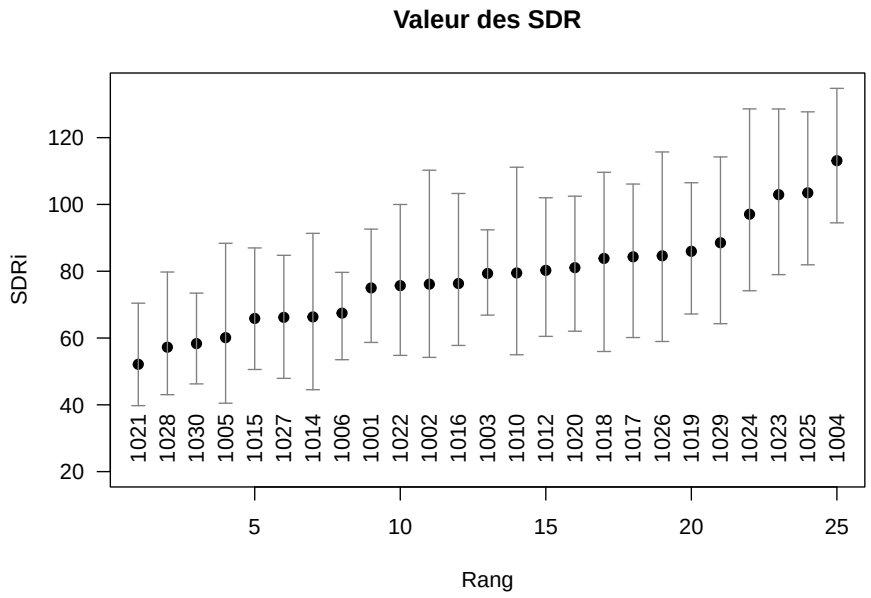
```
load("bootSDR.Rda")
plotboot(boot.SDR, main = "Distribution bootstrap de SDR pour T_1001", xlab = "SDR")
```



On a donc une valeur estimée de 75 pour le SDR_i de cet arbre et un intervalle de confiance qui court de 58.9 à 92.7, soit une amplitude relative de 45 %. C'est intermédiaire entre les 8 % du PV_i et les 87 % du CV_i.

```
SDRi <- as.data.frame(matrix(NA, nrow = ncol(apsrk), ncol = 4))
colnames(SDRi) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(apsrk))){
  SDRi[i, "Tree_name"] <- colnames(apsrk)[i]
  boot.out <- boot(apsrk[ , i], sdn, R = 9999)
  SDRi[i, "est"] <- boot.out$t0
  ci <- boot.ci(boot.out, type = "bca")
  SDRi[i, "inf"] <- ci$bca[4]
  SDRi[i, "sup"] <- ci$bca[5]
}
save(SDRi, file = "SDRi.Rda")
```

```
load("SDRi.Rda")
SDRi <- SDRi[order(SDRi$est), ]
n <- nrow(SDRi)
plot(1:n, SDRi$est, pch = 19, ylim = c(20, max(SDRi$sup)),
     las = 1, ylab = "SDRi", xlab = "Rang",
     main = "Valeur des SDR")
with(SDRi, myarrows(1:n, inf, 1:n, sup))
text(1:n, rep(30, n), substr(SDRi$Tree_name, 3, 7), srt = 90, xpd = NA)
```



3.5.2 SDR individuels moyen

C'est la moyenne arithmétique des SDR_i (cf. section 2.3 page 13).

```
(SDRm <- mean(SDRi$est))
```

```
[1] 78.43089
```

```
(ci.SDRm <- boot.ci(boot(SDRi$est, moy, R = 9999), type = "bca")$bca[4:5])
```

```
[1] 72.99682 84.66543
```

3.5.3 SDR populationnel

C'est le SDR calculé sur la moyenne arithmétique des valeurs chaque année (cf. section 2.4 page 13).

```
xmoy <- rowMeans(apsrk)
boot.SDRp <- boot(xmoy, sdn, R = 9999)
save(boot.SDRp, file = "bootSDRp.Rda")
```

```
load("bootSDRp.Rda")
(SDRp <- boot.SDRp$t0)
```

```
[1] 57.47051
```

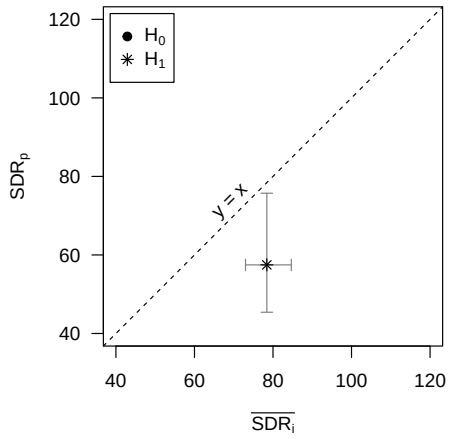
```
(ci.SDRp <- boot.ci(boot.SDRp, type = "bca")$bca[4:5])
```

```
[1] 45.41021 75.71824
```

3.5.4 Test de la synchronie parfaite avec SDR_p et $\overline{SDR_i}$

On rejette ici l'hypothèse nulle de synchronie parfaite entre les arbres, on a gagné en puissance avec la statistique du SDR par rapport à celle du CV, mais on est quand même moins puissant qu'avec PV :

```
par(pty="s ", mar = c(5, 4, 0, 0) + 0.1)
plot(SDRm, SDRp, type = "n", xlab = expression(bar(SDR[i])),
     ylab = expression(SDR[p]), las = 1, ylim = c(40, 120), xlim = c(40, 120))
myarrows(ci.SDRm[1], SDRp, ci.SDRm[2], SDRp)
myarrows(SDRm, ci.SDRp[1], SDRm, ci.SDRp[2])
points(SDRm, SDRp, pch = 8)
abline(c(0, 1), lty = 2) ; text(70, 70, "y = x", srt = 45, pos = 3)
legend("topleft", inset = 0.02, pch = c(19, 8),
       legend = c(expression(H[0]), expression(H[1])))
```

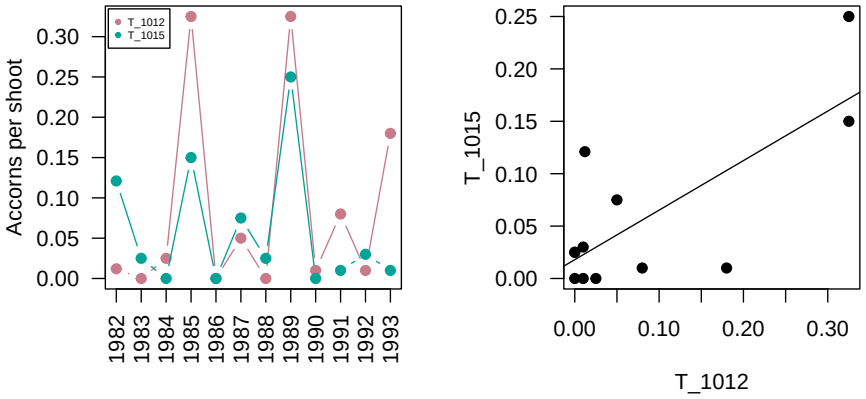


4 Statistiques de synchronisation

4.1 Statistiques de corrélation

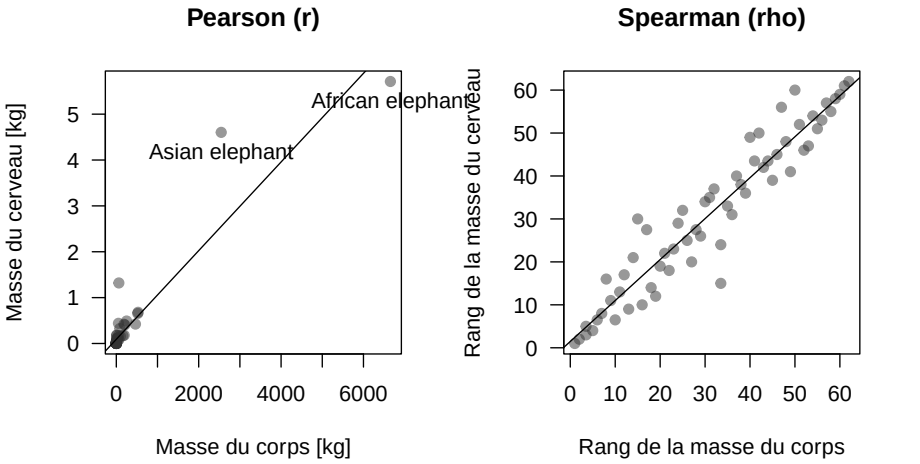
Pour mesurer la synchronie entre deux arbres, on va oublier l'aspect temporel des séries pour se concentrer sur la corrélation des valeurs entre elles. Par exemple, avec les arbres n^{os} 1012 et 1015 :

```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1012", "T_1015"), main = "")
x <- aps[ , which(colnames(aps) == "T_1012")]
y <- aps[ , which(colnames(aps) == "T_1015")]
plot(x, y, pch = 19, las = 1, xlab = "T_1012", ylab = "T_1015")
abline(lm(y~x))
```



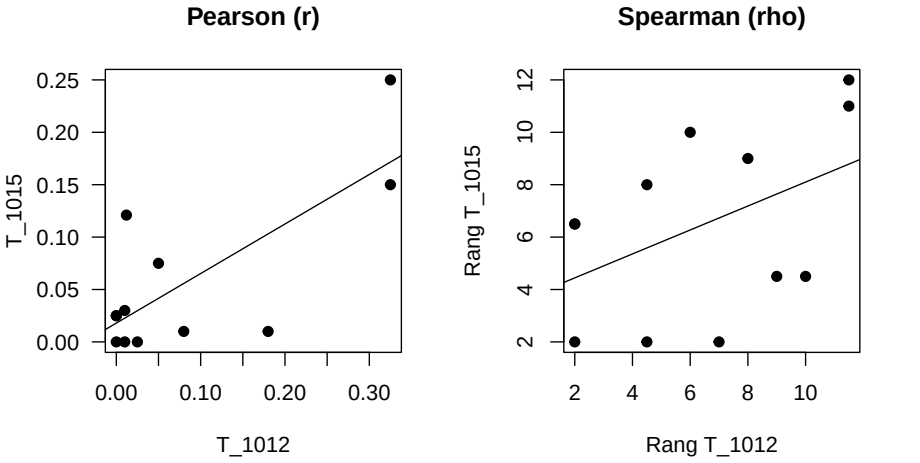
Les deux statistiques les plus courantes pour mesurer le degré d'association entre deux variables quantitatives sont le coefficient de corrélation linéaire r de Pearson [41] et son variant ρ de Spearman [49] qui consiste à travailler avec le rang des données. Illustrons ceci avec la masse du cerveau et du corps de 62 mammifères terrestres [1] disponible dans le jeu de données `mammals` de la bibliothèque standard `MASS` [51] :

```
library(MASS) ; data(mammals)
with(mammals, {
  x <- body ; y <- brain/1000
  col <- rgb(0.2, 0.2, 0.2, 0.5)
  par(mfrow = c(1, 2))
  plot(x, y, pch = 19, xlab = "Masse du corps [kg]",
       ylab = "Masse du cerveau [kg]", main = "Pearson (r)", las = 1, col = col)
  lourd <- grep("elephant", rownames(mammals))
  text(x[lourd], y[lourd], rownames(mammals)[lourd], pos = 1, xpd = NA)
  abline(lm(y~x))
  xr <- rank(x) ; yr <- rank(y)
  plot(xr, yr, pch = 19, xlab = "Rang de la masse du corps", col = col,
       ylab = "Rang de la masse du cerveau", las = 1,
       main = "Spearman (rho)")
  abline(lm(yr~xr))
})
```



Dans les deux cas on observe une corrélation positive entre les deux variables, mais à gauche, avec les données brutes, elle s'explique principalement par l'opposition entre les éléphants et les autres mammifère tandis qu'à droite, avec les données sur les rangs, toutes les espèces participent à la relation. Pour les études relatives au *masting* le coefficient de corrélation linéaire r de Pearson va mettre en avant la synchronisation des glandées tandis que le ρ de Spearman appréciera la synchronie sur l'ensemble des années :

```
x <- aps[ , which(colnames(aps) == "T_1012")]
y <- aps[ , which(colnames(aps) == "T_1015")]
par(mfrow = c(1, 2))
plot(x, y, pch = 19, xlab = "T_1012",
     ylab = "T_1015", main = "Pearson (r)", las = 1)
lourd <- grep("elephant", rownames(mammals))
text(x[lourd], y[lourd], rownames(mammals)[lourd], pos = 1, xpd = NA)
abline(lm(y~x))
xr <- rank(x) ; yr <- rank(y)
plot(xr, yr, pch = 19, xlab = "Rang T_1012",
     ylab = "Rang T_1015",
     main = "Spearman (rho)")
abline(lm(yr~xr))
```



Une autre statistique populaire pour mesurer l'association entre deux variables est le τ de Kendall [29], elle a été employée par exemple par Herrera [21] dans le cadre du *masting*. L'approche ressemble un peu à celle du PV (cf. section 3.2.1 page 29) dans le sens où pour deux séries temporelles $\boldsymbol{x}_k$ et $\boldsymbol{x}_l$ on va envisager tous les couples distincts $(\{x_{ik}, x_{il}\}, \{x_{jk}, x_{jl}\})$ possibles. Un couple est dit *concordant* si $x_{ik} < x_{jk}$ et $x_{il} < x_{jl}$ ou $x_{ik} > x_{jk}$ et $x_{il} > x_{jl}$. Un couple est dit *discordant* si $x_{ik} < x_{jk}$ et $x_{il} > x_{jl}$ ou $x_{ik} > x_{jk}$ et $x_{il} < x_{jl}$. Si $x_{ik} = x_{jk}$ ou $x_{il} = x_{jl}$ le couple est écarté[31]. La statistique de Kendal est définie par :

$$\tau = \frac{\text{(nombre de couples concordants)} - \text{(nombre de couples discordants)}}{\frac{1}{2}n(n-1)}$$

C'est une statistique de rang puisque l'on ne considère que l'ordre des valeurs entre-elles. Le dénominateur étant le nombre total de couples, la valeur de τ est comprise entre -1 et 1. Graphiquement si on reprend l'exemple des arbres n^{os} 1012 et 1015 :

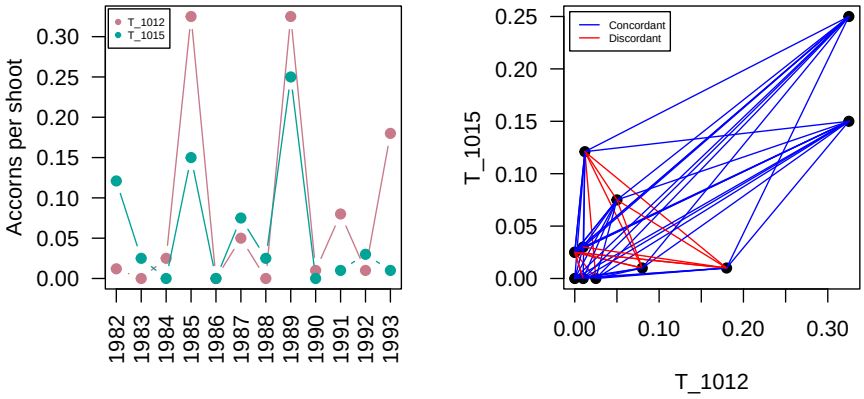
```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1012", "T_1015"), main = "")
xk <- aps[ , which(colnames(aps) == "T_1012")]
xl <- aps[ , which(colnames(aps) == "T_1015")]
plot(xk, xl, pch = 19, xlab = "T_1012",
     ylab = "T_1015", las = 1)
n <- nrow(aps) ; ncon <- 0 ; ndis <- 0
for(i in 1:(n - 1)){
  for(j in (i + 1):n){
    if( ( xk[i] < xk[j] & xl[i] < xl[j]) | (xk[i] > xk[j] & xl[i] > xl[j]) ){
      segments(xk[i], xl[i], xk[j], xl[j], col = "blue")
      ncon <- ncon + 1
    }
    if( ( xk[i] < xk[j] & xl[i] > xl[j]) | (xk[i] < xk[j] & xl[i] > xl[j]) ){
      segments(xk[i], xl[i], xk[j], xl[j], col = "red")
      ndis <- ndis + 1
    }
  }
}
legend("topleft", inset = 0.02, lty = 1, col = c("blue", "red"),
       legend = c("Concordant", "Discordant"), cex = 0.5)
(ncon - ndis)/(0.5*n*(n - 1))
```

```
[1] 0.3939394
```

```
cor(xk, xl, method = "kendall")
```

```
[1] 0.3770492
```

[31] Pas de panique, toute cela sera plus clair avec la représentation graphique ci-après.



On voit qu'il y a un excès de couples concordants par rapport aux couples discordants. La petite différence numérique avec le résultat de la fonction `cor()` vient de ce qu'elle applique une correction pour les ex-æquo [30].

4.2 Synchronistaion des arbres pour les glandées

Quand on utilise le r de Pearson on met l'accent sur la synchronisation des glandées. Calculons la matrice des corrélations entre tous les couples possibles d'arbres :

```
Symat <- function(mat, method = "pearson", use = "pairwise.complete.obs", ...){
  n <- ncol(mat)
  res <- matrix(NA, nrow = n, ncol = n)
  rownames(res) <- colnames(mat)
  diag(res) <- 0 # Pour trouver le max hors diagonale facilement
  for(i in 1:(n-1))
    for(j in (i+1):n){
      res[i, j] <- res[j, i] <- cor(mat[, i], mat[, j], method = method, use = use, ...)
    }
  return(res)
}
Syr <- Symat(aps)
which(Syr == max(Syr), arr.ind = TRUE)
```

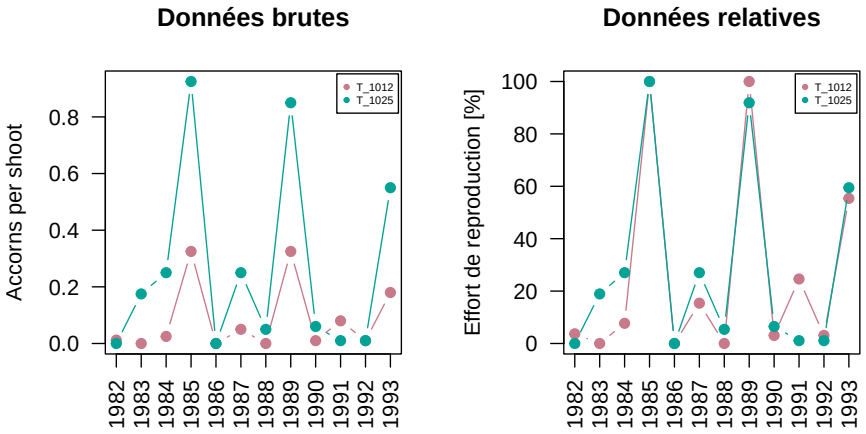
```
       row col
T_1025  20   8
T_1012   8  20
```

```
which(Syr == min(Syr), arr.ind = TRUE)
```

```
       row col
T_1030  25  13
T_1018  13  25
```

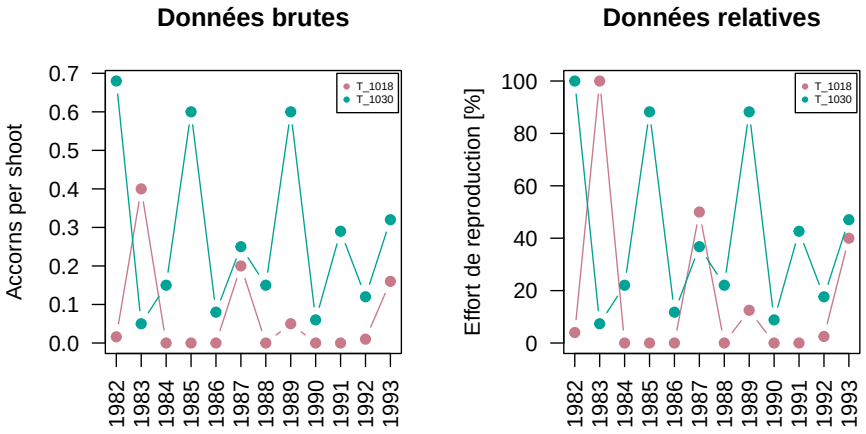
Les deux arbres les mieux synchronisés pour les glandées sont les n° 1012 et 1025. Ils tous les deux un pic de production en 1985 et 1989 et un pic secondaire en 1993 :

```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1012", "T_1025"), main = "Données brutes", posleg = "topright")
plot.tree(apsr, c("T_1012", "T_1025"), main = "Données relatives", posleg = "topright",
          ylab = "Effort de reproduction [%]")
```



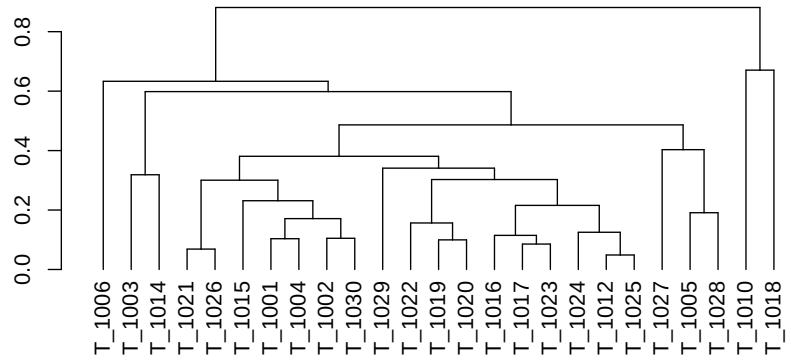
Les deux arbres les moins synchronisés pour les glandées sont les n° 1018 et 1030. Ils sont en quasi opposition de phase en 1982, 1983, 1985 et 1989 :

```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1018", "T_1030"), main = "Données brutes", posleg = "topright")
plot.tree(apsr, c("T_1018", "T_1030"), main = "Données relatives", posleg = "topright",
          ylab = "Effort de reproduction [%]")
```



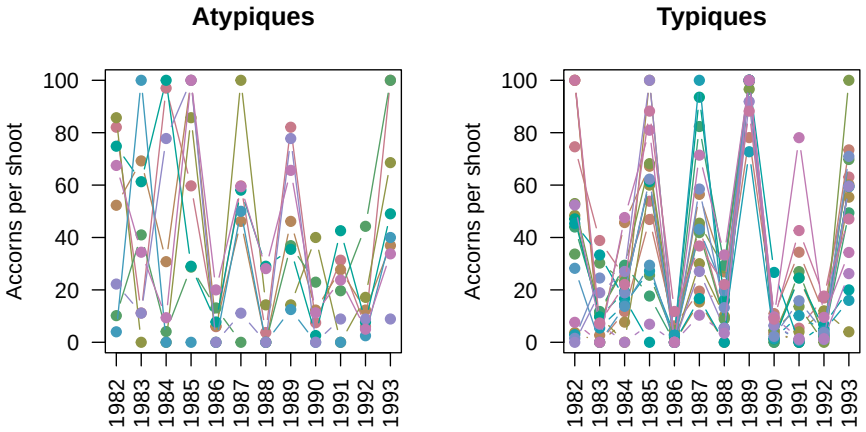
Pour avoir une vue d'ensemble on peut utiliser $1 - r$ comme une mesure de la distance entre deux arbres du point de vue de la synchronie et utiliser une méthode de classification hiérarchique pour représenter le tout :

```
dsr <- Syr
diag(dsr) <- 1
dsr <- 1 - dsr
plot(as.dendrogram(hclust(as.dist(dsr), "average")))
```



On peut distinguer schématiquement deux groupes : un groupe central (de l'arbre n° 1021 à celui n° 1025) qui ont un comportement assez synchrone et le groupe des arbres restants avec un comportement plus atypique, graphiquement :

```
par(mfrow = c(1, 2))
aty <- c("T_1006", "T_1003", "T_1014", "T_1027", "T_1005", "T_1028", "T_1010", "T_1018")
plot.tree(apsr, aty, posleg = "topright", main = "Atypiques", leg = FALSE)
plot.tree(apsr, colnames(apsr)[!colnames(apsr) %in% aty], leg =  FALSE,
  main = "Typiques")
```



4.3 Synchronisation fine des arbres

4.3.1 Avec le ρ de Spearman

```
Sys <- Symat(aps, method = "spearman")
which(Sys == max(Sys), arr.ind = TRUE)
```

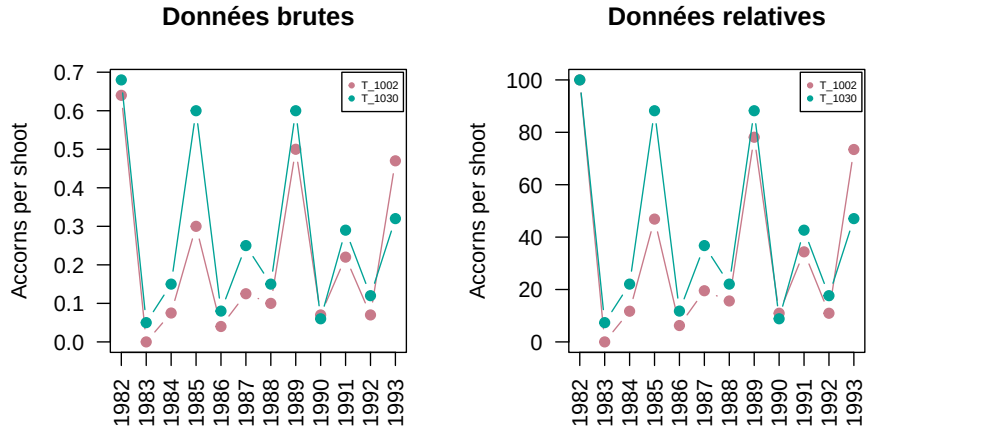
```
      row col
T_1030  25   2
T_1002   2  25
```

```
which(Sys == min(Sys), arr.ind = TRUE)
```

```
      row col
T_1014   9   7
T_1010   7   9
```

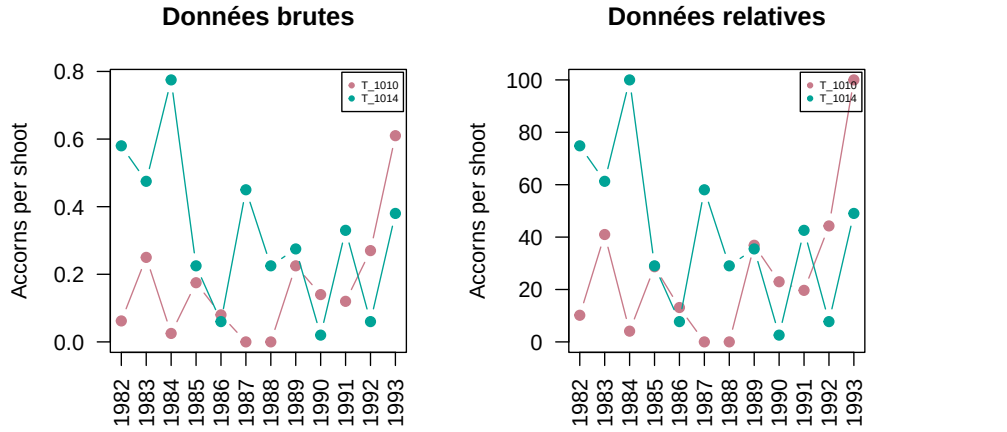
Les deux arbres les mieux synchronisés sont les n^{os} 1002 et 1030, que se soit en données brutes ou bien sur les rangs ils ont effectivement des profils très proches :

```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1002", "T_1030"), main = "Données brutes", posleg = "topright")
plot.tree(apsr, c("T_1002", "T_1030"), main = "Données relatives", posleg = "topright")
```



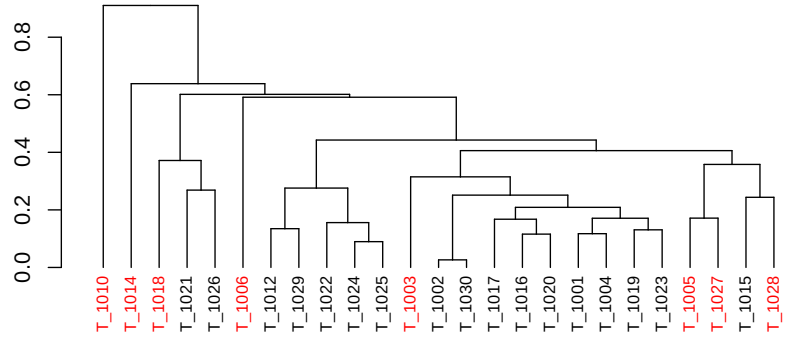
Les deux arbres les moins synchronisés sont les n^{os} 1010 et 1014, et ils sont effectivement souvent en opposition de phase :

```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1010", "T_1014"), main = "Données brutes", posleg = "topright")
plot.tree(apsr, c("T_1010", "T_1014"), main = "Données relatives", posleg = "topright")
```



Pour avoir une vue d'ensemble on peut utiliser $1-\rho$ comme une mesure de la distance entre deux arbres du point de vue de la synchronie et utiliser une méthode de classification hiérarchique pour représenter le tout. En coloriant en rouge les arbres qui étaient atypiques avec $1-r$, on voit que l'on n'obtient pas du tout la même classification des arbres :

```
dss <- Sys
diag(dss) <- 1
dss <- 1 - dss
dhc <- as.dendrogram(hclust(as.dist(dss), "average"))
dL <- dendrapply(dhc, function(n){
  if(is.leaf(n)){
    labelCol <- ifelse(attr(n,"label") %in% aty, "red", "black")
    attr(n, "nodePar") <- list(pch = NA, lab.col = labelCol, lab.cex = 0.75)
  }
  n
})
plot(dL)
```



4.3.2 Avec le τ de Kendall

```
Syk <- Symat(aps, method = "kendall")
which(Syk == max(Syk), arr.ind = TRUE)
```

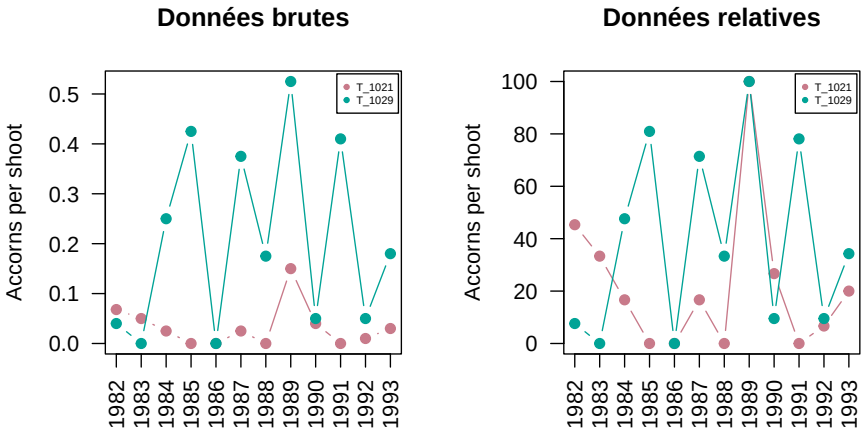
```
       row col
T_1030  25   2
T_1002   2  25
```

```
which(Syk == min(Syk), arr.ind = TRUE)
```

```
       row col
T_1029  24  16
T_1021  16  24
```

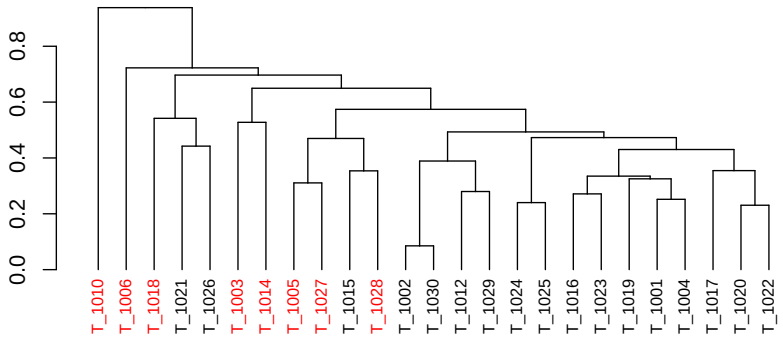
Les deux arbres les mieux synchronisés sont les n^{os} 1002 et 1030, comme avec le ρ de Spearman, mais les deux arbres les moins bien synchronisés sont cette fois les n^{os} 1021 et 1029 :

```
par(mfrow = c(1, 2))
plot.tree(aps, c("T_1021", "T_1029"), main = "Données brutes", posleg = "topright")
plot.tree(apsr, c("T_1021", "T_1029"), main = "Données relatives", posleg = "topright")
```



Pour avoir une vue d'ensemble on peut utiliser $1-\tau$ comme une mesure de la distance entre deux arbres du point de vue de la synchronie et utiliser une méthode de classification hiérarchique pour représenter le tout. En coloriant en rouge les arbres qui étaient atypiques avec $1-r$, on voit que l'on obtient presque la même classification des arbres. Le τ de Kendall est donc ici un peu intermédiaire entre le r de Pearson et le ρ de Spearman :

```
dsk <- Syk
diag(dsk) <- 1
dsk <- 1 - dsk
dhc <- as.dendrogram(hclust(as.dist(dsk), "average"))
dL <- dendrapply(dhc, function(n){
  if(is.leaf(n)){
    labelCol <- ifelse(attr(n,"label") %in% aty, "red", "black")
    attr(n, "nodePar") <- list(pch = NA, lab.col = labelCol, lab.cex = 0.75)
  }
  n
})
plot(dL)
```

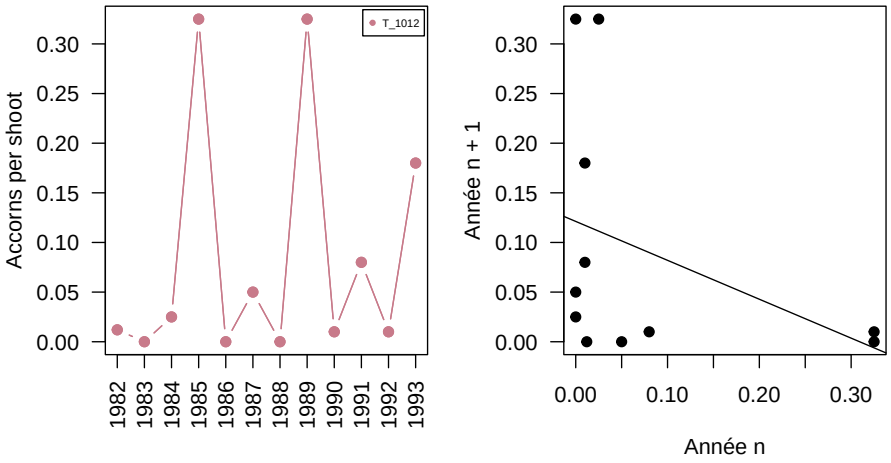


5 Statistiques d'auto-corrélation temporelle

5.1 Lien avec le *masting*

Si l'on revient à la définition du phénomène de *masting* (cf. section 1.1 page 3), on ne voit pas *a priori* le lien avec l'auto-corrélation temporelle. Les études portant sur le *masting* en font pourtant grand usage, avec la mise en évidence classique d'une auto-corrélation temporelle négative d'ordre 1, c'est à dire entre l'année n et l'année $n+1$. Ce phénomène est bien visible ici par exemple sur l'arbre n° 1012 :

```
par(mfrow = c(1, 2), mar = c(4, 4, 0, 1) + 0.1)
plot.tree(aps, "T_1012", main = "", posleg = "topright")
icol <- which(colnames(aps) == "T_1012")
x <- aps[ , icol]
n <- nrow(aps)
xn <-x[-n]    # Année n
xnpu <- x[-1] # Année n + 1
r <- signif(cor(xn, xnpu), 3)
plot(xn, xnpu, pch = 19, xlab = "Année n", ylab = "Année n + 1", las = 1)
abline(lm(xnpu~xn))
```



Les glandées en 1985 et 1989 sont suivies systématiquement par une année de faible investissement dans la reproduction, comme si l'arbre avait épuisé ses ressources. De façon non symétrique, les années de faible investissement dans la reproduction ne sont pas systématiquement suivies par une glandée, comme s'il fallait un certain temps à l'arbre pour reconstituer ses ressources. C'est sur ces hypothèses que se fondent le modèle déterministe en temps discret dit RBM[32] [24, 45]. Ce modèle, avec une paramétrisation correcte, est capable de générer du chaos déterministe et donc un comportement épisodique de l'effort de reproduction, ce qui est très intéressant pour les études ayant trait au *masting*. Mettre en évidence un phénomène d'auto-corrélation négative c'est donc ne pas fermer la porte à un potentiel mécanisme explicatif de l'épisodisme du *masting*.

Comme nous l'avons déjà vu avec la synchronisation (cf. section 4 page 49), les trois statistiques les plus courantes pour mesurer le degré d'association entre deux variables quantitatives sont le coefficient de corrélation linéaire r de Pearson [41], son variant ρ de Spearman [49] qui consiste à travailler avec le rang des données et le τ de Kendall [29].

[32] Pour « Ressource Based Model ».

On ne pourra pas utiliser ici l'approche du bootstrap pour calculer les intervalles de confiance pour ces statistiques puisqu'en ré-échantillonant on détruit la structure temporelle et on se retrouvera avec distributions centrées sur zéro. On pourrait être tenté de ré-échantilloner non pas parmi les x_{ij} mais parmi les couples (x_{ij}, x_{i+1j}) pour préserver la structure d'auto-corrélation temporelle, comme par exemple ici avec le r de Pearson :

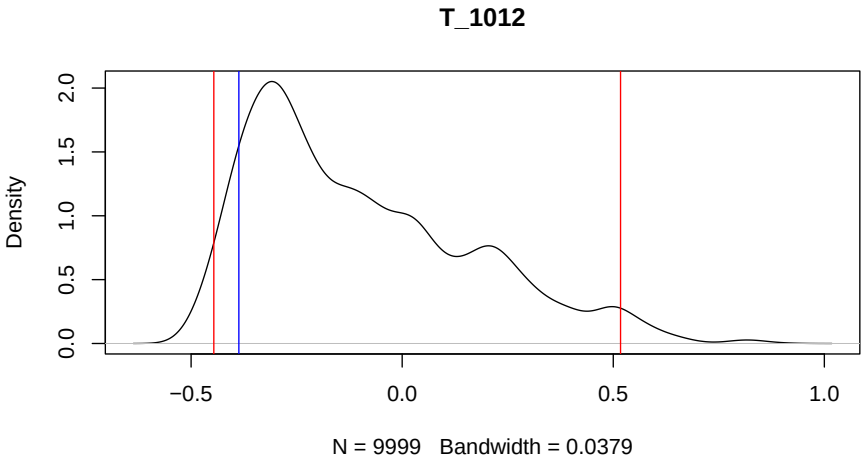
```
library(boot)
essai <- function(x, i = seq_len(length(x)),
                  use = "pairwise.complete.obs", ...){
  i <- i[-length(i)] # indices de n-1 couples
  n <- length(x)
  xn <-x[-n]    # Année n
  xnpu <- x[-1] # Année n + 1
  return(cor(xn[i], xnpu[i], use = use, method = "pearson", ...))
}
icol <- which(colnames(aps) == "T_1012")
boot(aps[, icol], essai, R = 9)$t[ , 1]
```

```
[1] -0.2679555 -0.3241183 -0.5115798 -0.5104179 -0.5861755 -0.3813739 -0.3545068
[8] -0.1919024 -0.5794446
```

On voit que l'on a préservé la structure d'auto-corrélation temporelle puisque l'on obtient des valeurs négatives. Et puis cela pourrait sembler raisonnable, cela revient à pondérer plus ou moins les couples (x_{ij}, x_{i+1j}) pour voir l'impact sur la statistique. Mais il y a un petit problème ici : supposez que par hasard j'ai ré-échantilloné uniquement un seul couple de valeurs, mettons 11 fois le couple $(0.025, 0.325)$. Le souci est qu'il n'y a *aucune* série temporelle qui puisse générer 11 fois le couple $(0.025, 0.325)$. Sauf cas exceptionnel, nos échantillons bootstrap seront physiquement impossible à obtenir à partir d'une série temporelle, ce qui est un peu embêtant comme modèle de la réalité. On ne peut pas non plus utiliser les tests classiques associés à r, τ et ρ puisqu'ils supposent l'hypothèse d'indépendance, clairement violée ici[33]. On peut quand même tester l'existence d'une auto-corrélation avec un test de permutation des rangs, par exemple avec le r de Pearson pour l'arbre n° 1012 :

```
A <- function(x, method = "pearson", plot = FALSE, alpha = 0.05, R = 9999,
                  use = "pairwise.complete.obs", main = "", ...){
  stat <- function(x, method, use){
    n <- length(x)
    xn <-x[-n]    # Année n
    xnpu <- x[-1] # Année n + 1
    return(cor(xn, xnpu, method = method, use = use))
  }
  t0 <- stat(x, method = method, use = use)
  t <- replicate(R, stat(sample(x), method = method, use = use))
  inf <- quantile(t, alpha/2, ...)
  sup <- quantile(t, 1 - alpha/2, ...)
  if(plot){
    plot(density(t, ...), main = main)
    abline(v = c(t0, inf, sup), col = c("blue", "red", "red"))
  }
  invisible(list(t0 = t0, inf = inf, sup = sup))
}
icol <- which(colnames(aps) == "T_1012")
A(aps[ , icol], plot = TRUE, main = "T_1012")
```

[33] Les couples (x_{i-1j}, x_{ij}) et (x_{ij}, x_{i+1j}) sont liés par la valeur partagée x_{ij}.



Avec un risque de première espèce de 5 %, les données expérimentales ne nous permettent pas de rejeter l'hypothèse nulle de l'absence d'auto-corrélation temporelle pour l'arbre n° 1012, on l'accepte donc, faute de mieux, avec un risque de seconde espèce inconnu.

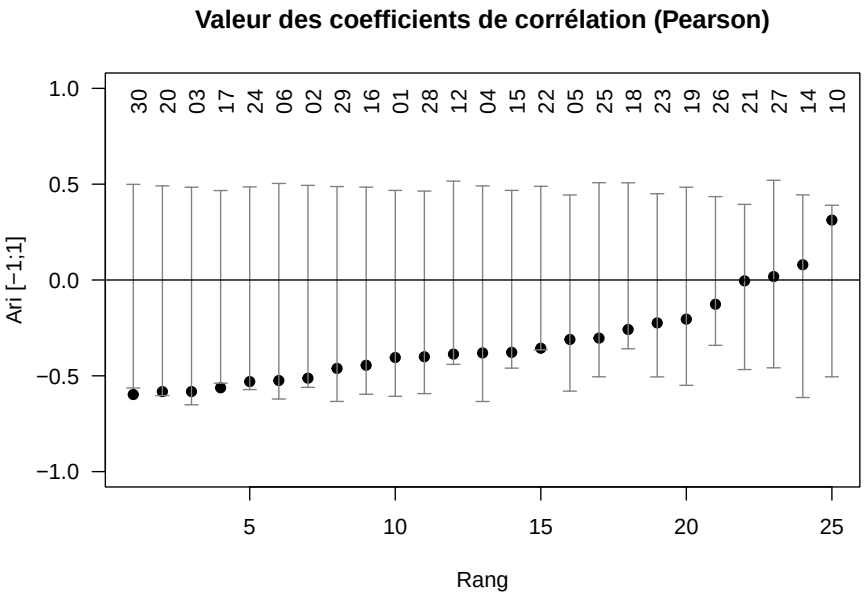
5.2 Auto-corrélation avec le r de Pearson

5.2.1 Ar individuels

On note A_{ri} le coefficient d'auto-corrélation obtenu avec le r de Pearson. L'accent est mis ici sur les valeurs extrêmes, c'est à dire les glandées.

```
Ari <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(Ari) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  Ari[i, "Tree_name"] <- colnames(aps)[i]
  Ares <- A(aps[ , i], method = "pearson")
  Ari[i, "est"] <- Ares$t0
  Ari[i, "inf"] <- Ares$inf
  Ari[i, "sup"] <- Ares$sup
}
save(Ari, file = "Ari.Rda")
```

```
load("Ari.Rda") ; load("myarrows.Rda")
Ari <- Ari[order(Ari$est), ]
n <- nrow(Ari)
plot(1:n, Ari$est, pch = 19, ylim = c(-1, 1),
     las = 1, ylab = "Ari [-1;1]", xlab = "Rang",
     main = "Valeur des coefficients de corrélation (Pearson)")
with(Ari, myarrows(1:n, inf, 1:n, sup))
abline(h = 0)
text(1:n, 1, substring(Ari$Tree_name, 5, 6), srt = 90, pos = 1)
```

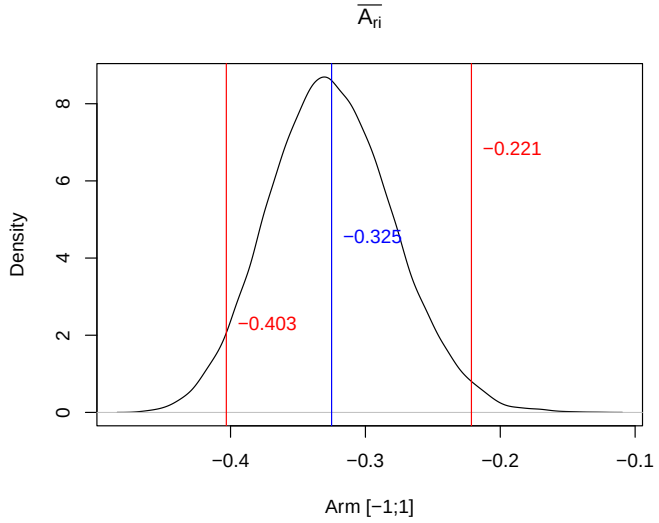


On voit ici que l'on n'arrive pas à rejeter l'hypothèse nulle d'absence d'auto-corrélation pour tous les arbres, à l'exception des nos 1030 et 1017, et encore on est à la limite. On généralise donc à l'ensemble des arbres ce qui avait été observé *supra* pour l'arbre no 1012, à savoir la faible puissance du test. Mais on voit bien qu'il y a une tendance générale, à savoir que pour la très grande majorité des arbres l'auto-corrélation est négative.

5.2.2 Ar individuels moyens

On gagne en puissance on considérant la valeur moyenne de l'auto-corrélation pour tous les arbres, on met clairement en évidence ici une auto-corrélation négative :

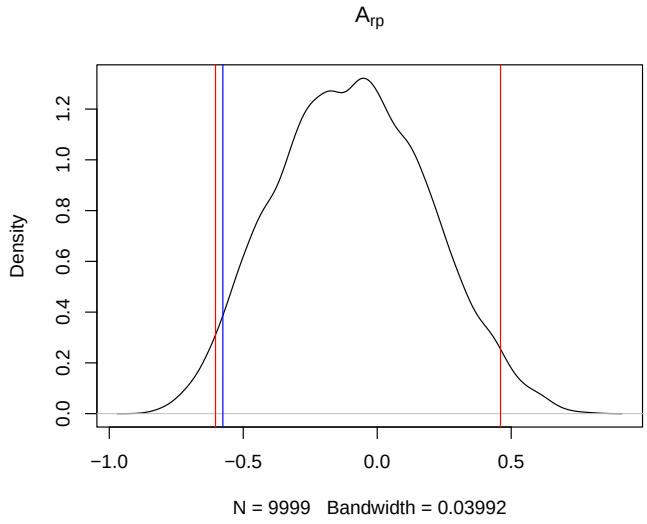
```
moy <- function(x, i, ...) mean(x[i], ...)
boot.Arm <- boot(Ari$est, moy, R = 9999)
plotboot(boot.Arm, xlab = "Arm [-1;1]", main = expression(bar(A[ri])))
```



5.2.3 Ar populationnel

C'est la statistique A_{rp} calculé sur la moyenne arithmétique des valeurs chaque année, comme si on n'avait plus accès aux données individuelles (cf. section 2.4 page 13). Qui dit perte d'information dit perte de puissance, et on est ici incapable de rejeter H_0 :

```
xmoy <- rowMeans(aps)
A(xmoy, method = "pearson", plot = TRUE, main = expression(A[rp]))
```



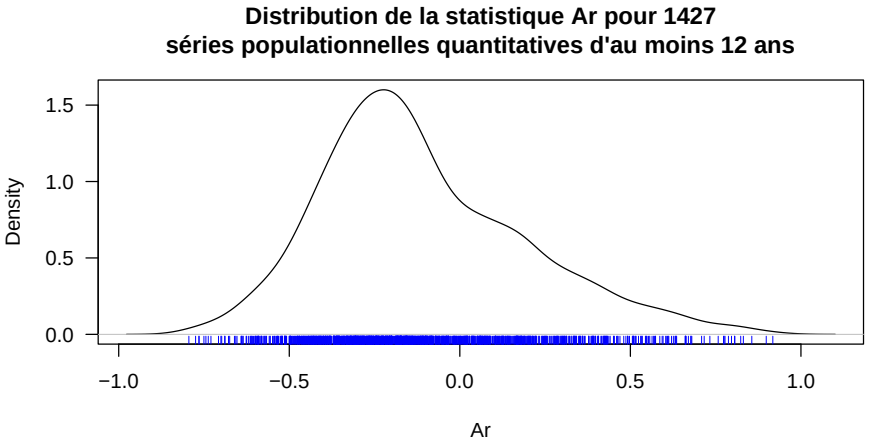
J'ai utilisé le script suivant pour calculer le coefficient d'auto-corrélation sur toutes les séries temporelles :

batch/getAr.R

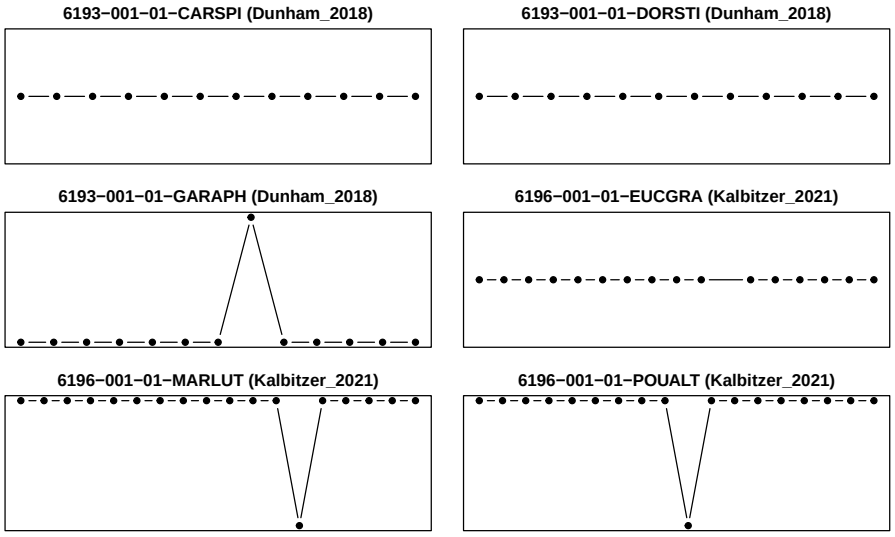
```
load("myws.Rda")
IDs <- unique(ms$ID) ; n <- length(IDs)
```

```
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 2))
colnames(tabres) <- c("ID", "est")

S <- function(x) A(x, method = "pearson")
tabname <- "tabAr"

for(i in seq_len(n)){
  the_ID <- IDs[i]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.res <- try(S(x))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.res
  assign(tabname, tabres)
  save(list = tabname, file = paste0(tabname, ".Rda"))
}
```

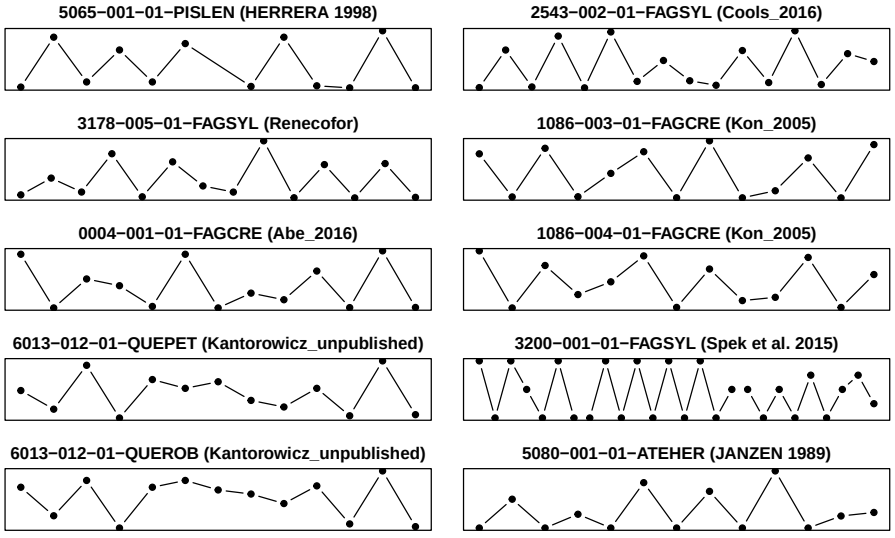
```
load("batch/tabAr.Rda")
x <- tabAr$est ; x <- x[is.finite(x)]
main <- paste0("Distribution de la statistique Ar pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "Ar", main = main, las = 1)
rug(x, col = "blue")
```



On retrouve ici la tendance générale des séries temporelles relatives au *masting* à avoir une auto-corrélation négative avec un mode de la distribution voisin de -0.2. Il y a 6 séries pour lesquelles la statistique n'est pas définie, il s'agit des séries constantes et des séries qui ne présentent que deux valeurs distinctes :



Le top-10 des séries présentant la plus forte auto-corrélation négative est le suivant :



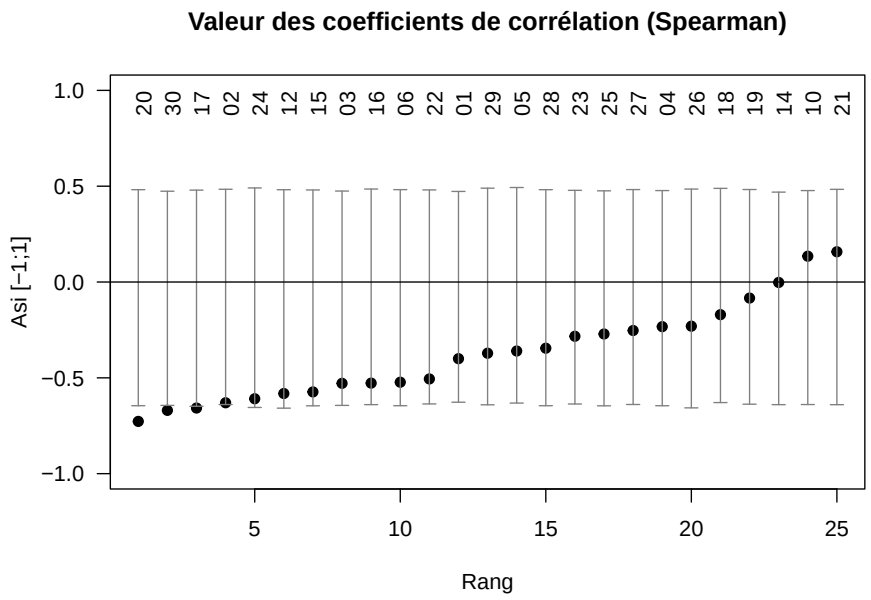
5.3 Auto-corrélation avec le ρ de Spearman

5.3.1 As individuels

On notera A_{si} le coefficient d'auto-corrélation basé sur le ρ de Spearman [49]. Le poids des glandées, c'est à dire les valeurs extrême, est moindre qu'avec le r de Pearson.

```
Asi <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(Asi) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  Asi[i, "Tree_name"] <- colnames(aps)[i]
  Ares <- A(aps[ , i], method = "spearman")
  Asi[i, "est"] <- Ares$t0
  Asi[i, "inf"] <- Ares$inf
```

```
  Asi[i, "sup"] <- Ares$sup
}
save(Asi, file = "Asi.Rda")


load("Asi.Rda")
Asi <- Asi[order(Asi$est), ]
n <- nrow(Asi)
plot(1:n, Asi$est, pch = 19, ylim = c(-1, 1),
     las = 1, ylab = "Asi [-1;1]", xlab = "Rang",
     main = "Valeur des coefficients de corrélation (Spearman)")
with(Asi, myarrows(1:n, inf, 1:n, sup))
abline(h = 0)
text(1:n, 1, substring(Asi$Tree_name, 5, 6), srt = 90, pos = 1)
```

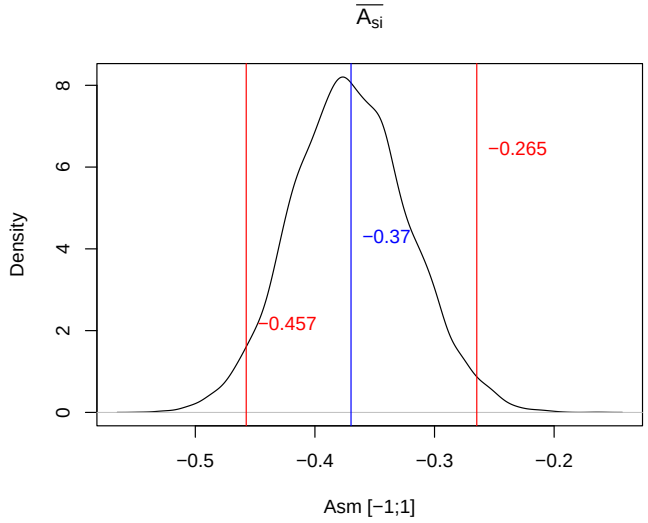


On retrouve ici des résultats analogues à ceux obtenus avec le r de Pearson : on n'arrive pas à rejeter l'hypothèse nulle d'absence d'auto-corrélation pour tous les arbres, à l'exception des n^{os} 1020, 1030 et 1017, et encore on est à la limite. Il y a une tendance générale, à savoir que pour la très grande majorité des arbres l'auto-corrélation est négative.

5.3.2 As individuels moyens

On gagne en puissance on considérant la valeur moyenne de l'auto-corrélation pour tous les arbres, on met clairement en évidence ici une auto-corrélation négative :

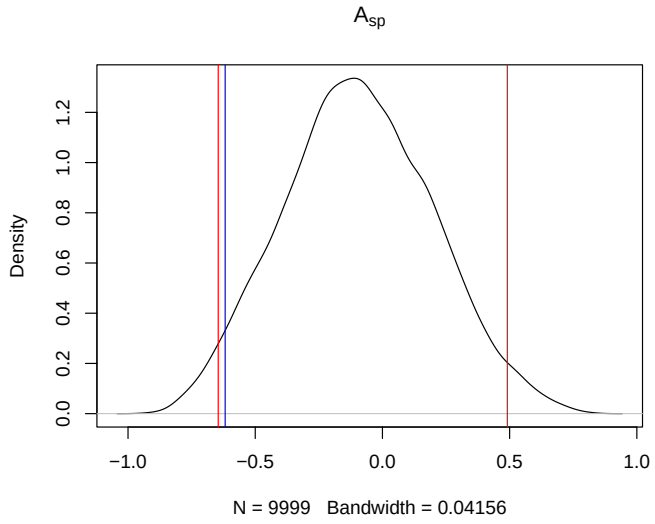
```
boot.Asm <- boot(Asi$est, moy, R = 9999)
plotboot(boot.Asm, xlab = "Asm [-1;1]", main = expression(bar(A[si])))
```



5.3.3 As populationnel

C'est la statistique A_{sp} calculé sur la moyenne arithmétique des valeurs chaque année, comme si on n'avait plus accès aux données individuelles (cf. section 2.4 page 13). Qui dit perte d'information dit perte de puissance, et on est ici incapable de rejeter H_0 :

```
xmoy <- rowMeans(aps)
A(xmoy, method = "spearman", plot = TRUE, main = expression(A[sp]))
```



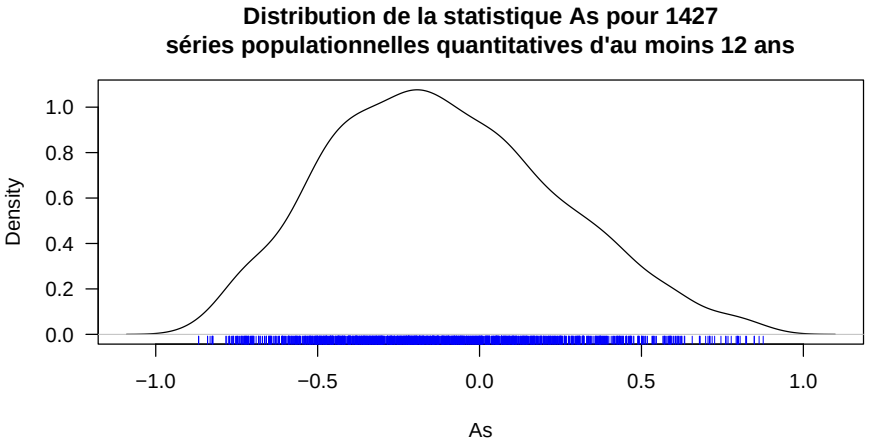
J'ai utilisé le script suivant pour calculer le coefficient d'auto-corrélation sur toutes les séries temporelles :

batch/getAs.R

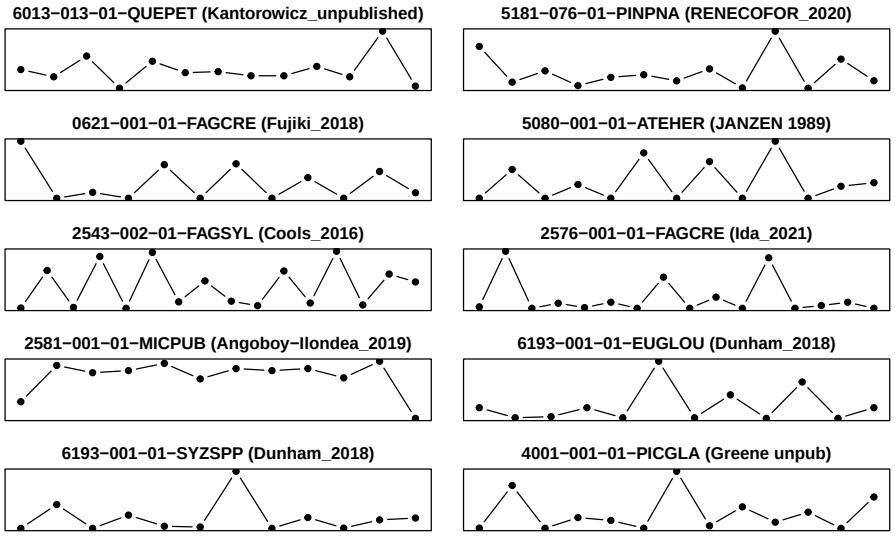
```
load("myws.Rda")
IDs <- unique(ms$ID) ; n <- length(IDs)
```

```
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 2))
colnames(tabres) <- c("ID", "est")

S <- function(x) A(x, method = "spearman")
tabname <- "tabAs"

for(i in seq_len(n)){
  the_ID <- IDs[i]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.res <- try(S(x))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.res
  assign(tabname, tabres)
  save(list = tabname, file = paste0(tabname, ".Rda"))
}
```

```
load("batch/tabAs.Rda")
x <- tabAs$est ; x <- x[is.finite(x)]
main <- paste0("Distribution de la statistique As pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "As", main = main, las = 1)
rug(x, col = "blue")
```



On retrouve ici la tendance générale des séries temporelles relatives au *masting* à avoir une auto-corrélation négative avec un mode de la distribution voisin de -0.2. Il y a 6 séries pour lesquelles la statistique n'est pas définie, il s'agit comme précédemment des séries constantes et des séries qui ne présentent que deux valeurs distinctes. Le top-10 des séries présentant la plus forte auto-corrélation négative est le suivant :

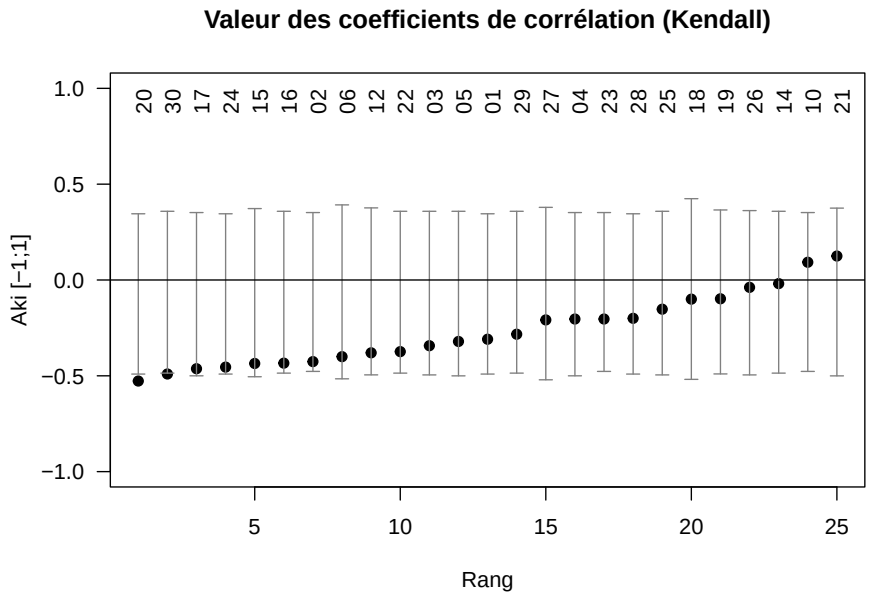


5.4 Auto-corrélation avec le τ de Kendall

5.4.1 Ak individuels

```
Aki <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(Aki) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  Aki[i, "Tree_name"] <- colnames(aps)[i]
  Ares <- A(aps[ , i], method = "kendall")
  Aki[i, "est"] <- Ares$t0
  Aki[i, "inf"] <- Ares$inf
  Aki[i, "sup"] <- Ares$sup
}
save(Aki, file = "Aki.Rda")
```

```
load("Aki.Rda")
Aki <- Aki[order(Aki$est), ]
n <- nrow(Aki)
plot(1:n, Aki$est, pch = 19, ylim = c(-1, 1),
     las = 1, ylab = "Aki [-1;1]", xlab = "Rang",
     main = "Valeur des coefficients de corrélation (Kendall)")
with(Aki, myarrows(1:n, inf, 1:n, sup))
abline(h = 0)
text(1:n, 1, substring(Aki$Tree_name, 5, 6), srt = 90, pos = 1)
```

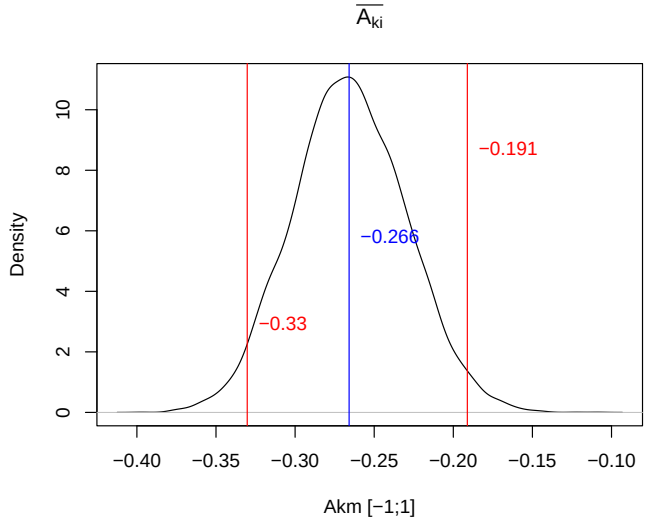


On note A_{ki} le coefficient d'auto-corrélation basé sur le τ de Kendall [29]. On retrouve ici des résultats analogues à ceux obtenus avec le r de Pearson : on n'arrive pas à rejeter l'hypothèse nulle d'absence d'auto-corrélation pour tous les arbres, à l'exception des n^{os} 1020, 1030 et 1017, et encore on est à la limite. Il y a une tendance générale, à savoir que pour la très grande majorité des arbres l'auto-corrélation est négative.

5.4.2 Ak individuels moyens

On gagne en puissance on considérant la valeur moyenne de l'auto-corrélation pour tous les arbres, on met clairement en évidence ici une auto-corrélation négative :

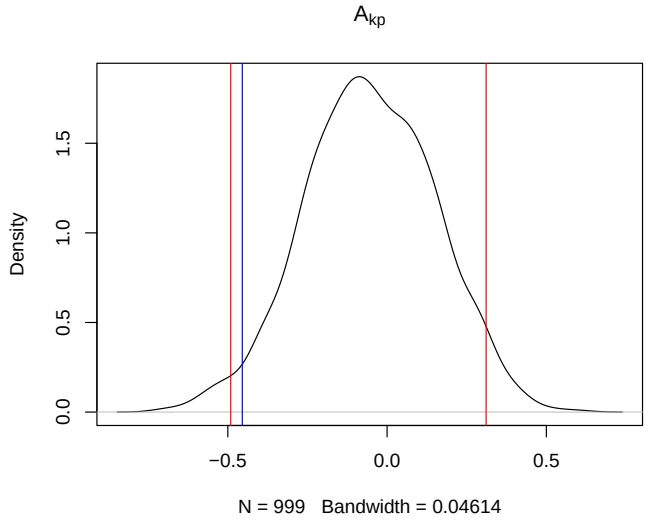
```
boot.Akm <- boot(Aki$est, moy, R = 9999)
plotboot(boot.Akm, xlab = "Akm [-1;1]", main = expression(bar(A[ki])))
```



5.4.3 Ak populationnel

C'est la statistique A_{kp} calculé sur la moyenne arithmétique des valeurs chaque année (cf. section 2.4 page 13). Qui dit perte d'information dit perte de puissance, et comme pour r et τ on est ici incapable de rejeter H_0 :

```
xmoy <- rowMeans(aps) ; set.seed(1)
Ares <- A(xmoy, method = "kendall", plot = TRUE, main = expression(A[kp]), R = 999)
```



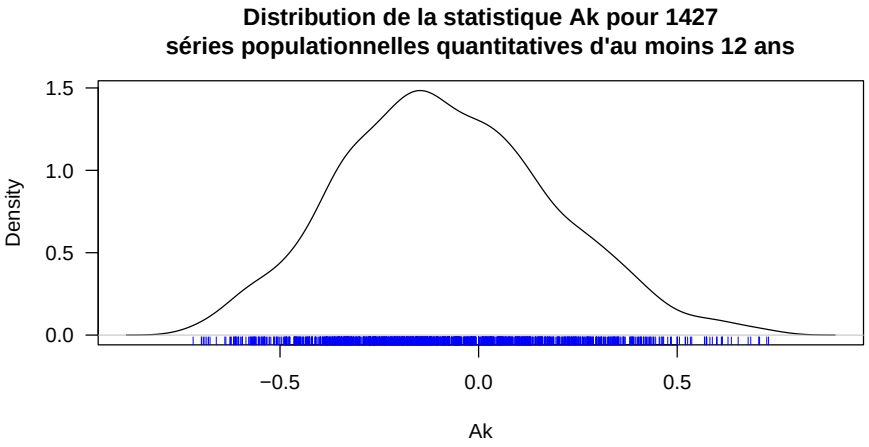
J'ai utilisé le script suivant pour calculer le coefficient d'auto-corrélation sur toutes les séries temporelles :

batch/getAk.R

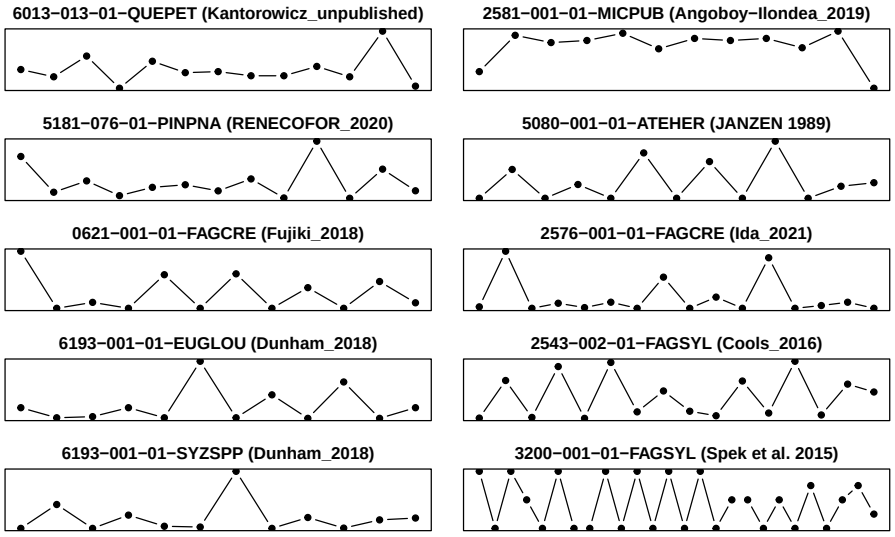
```
load("myws.Rda")
IDs <- unique(ms$ID) ; n <- length(IDs)
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 2))
```

```
colnames(tabres) <- c("ID", "est")

S <- function(x) A(x, method = "kendall")
tabname <- "tabAk"

for(i in seq_len(n)){
  the_ID <- IDs[i]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.res <- try(S(x))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.res
  assign(tabname, tabres)
  save(list = tabname, file = paste0(tabname, ".Rda"))
}
```

```
load("batch/tabAk.Rda")
x <- tabAk$est ; x <- x[is.finite(x)]
main <- paste0("Distribution de la statistique Ak pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "Ak", main = main, las = 1)
rug(x, col = "blue")
```



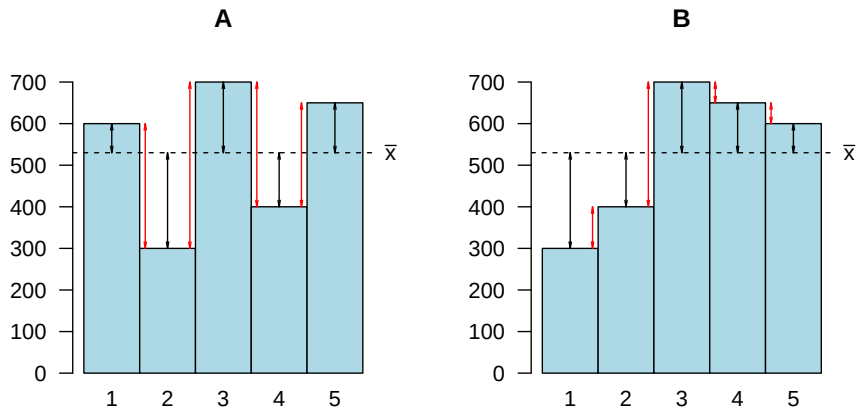
On retrouve ici la tendance générale des séries temporelles relatives au *masting* à avoir une auto-corrélation négative avec un mode de la distribution voisin de -0.1. Il y a 6 séries pour lesquelles la statistique n'est pas définie, il s'agit comme précédemment des séries constantes et des séries qui ne présentent que deux valeurs distinctes. Le top-10 des séries présentant la plus forte auto-corrélation négative est le suivant :



5.5 Désordre consécutif D

La statistique D (pour *desordre*) a été définie par Xavier Martín-Vide [35] pour des séries temporelles de pluviométrie et transposée dans le domaine du *masting* par Marcos Fernández-Martínez et collaborateurs [13, 14] sous le nom de « *consecutive disparity index* ». Je l'ai rangée avec les statistiques d'auto-corrélation puisque l'on considère tous les couples de valeurs consécutives. Reprenons l'exemple didactique donné dans la figure 1 de [35] :

```
XMVA <- c(600, 300, 700, 400, 650) ; XMVB <- c(300, 400, 700, 650, 600)
par(mfrow = c(1, 2), mar = c(4, 4, 4, 2))
myplot <- function(x, ...){
  n <- length(x)
  plot(1:n, XMVA, type = "n", las = 1, ylim = c(0, max(x)), xlim = c(1, n + 1),
       xaxt = "n", bty = "n", xlab = "", ylab = "", ...)
  rect(1:n, 0, (1:n)+1, x, col = "lightblue")
  axis(1, 1:n, at = (1:n) + 0.5, tick = FALSE, line = -1)
  abline(h = mean(x), lty = 2)
  arrows(0.5+(1:n), mean(x), 0.5+(1:n), x, code = 3, length = 0.05, angle = 10)
  x0 <- (2:n) + ifelse(x[-1] > x[-n], -0.1, +0.1)
  arrows(x0, x[-1], x0, x[-n], code = 3, length = 0.05, angle = 10, col = "red")
  text(6.5, mean(x), expression(bar(x)), xpd = NA)
}
myplot(XMVA, main = "A") ; myplot(XMVB, main = "B")
```



Ces deux séries sont formées des mêmes valeurs (300, 400, 600, 650 et 700) mais ordonnées différemment. Elles ont donc la même moyenne, le même écart-type et la même valeur pour les statistiques de la variabilité inter-annuelle (CV, PV et SDR). Pourtant, quand on compare les deux séries, on voit que la série A est beaucoup plus chahutée que la série B, avec des alternances marquées de valeurs élevées puis faibles. L'idée derrière la statistique D est de se concentrer sur les différences consécutives, c'est à dire sur les flèches rouges dans la figure ci-dessus. La statistique proposée est de faire la moyenne des valeurs absolues du logarithme du rapport des valeurs consécutives :

$$D(\boldsymbol{x}_j) = \frac{1}{n-1}\sum_{i=1}^{n-1}\left|\log\frac{x_{i+1j}}{x_{ij}}\right|$$

Malheureusement, les séries temporelles relatives au *masting* sont riches en valeurs nulles pour lesquelles la fonction logarithmique n'est pas définie. Pour pallier ce vilain petit fait, les auteurs [14] proposent d'introduire une constante arbitraire, $k \in \mathbb{R}_+^*$, ainsi :

$$D(\boldsymbol{x}_j) = \frac{1}{n-1}\sum_{i=1}^{n-1}\left|\log\frac{x_{i+1j}+k}{x_{ij}+k}\right|$$

Comme on retrouve les mêmes problèmes qu'avec la transformation logarithmique vue précédemment dans le cas de la statistique SDL (cf. section 3.4 page 44) on ne développera plus avant, et ce d'autant plus qu'il existe une solution simple comme on le verra dans la section suivante.

5.6 PVD

L'idée est de combiner D et PV dans une nouvelle statistique PDV qui consiste simplement de restreindre le calcul de PV aux valeurs consécutives :

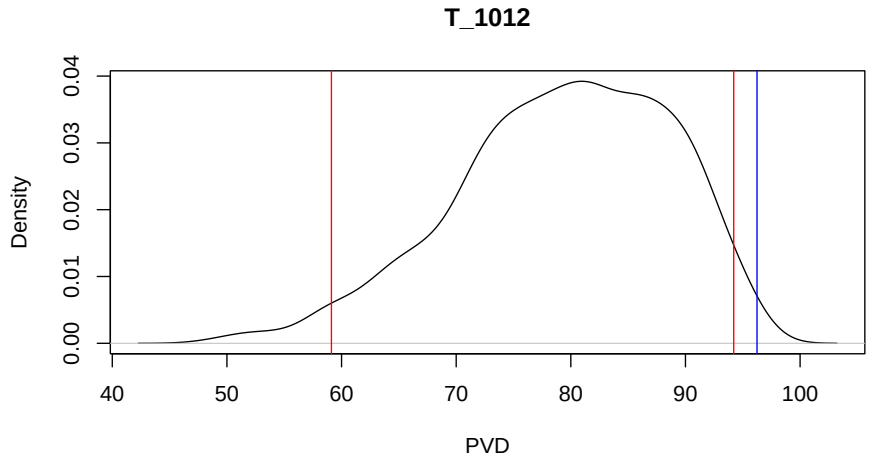
$$\mathrm{PVD}(\boldsymbol{x}_j) = \frac{1}{n-1}\sum_{k=1}^{n-1}\sum_{l=k+1}^{k+1} d(x_{kj}, x_{lj}) = \frac{1}{n-1}\sum_{k=1}^{n-1} d(x_{kj}, x_{k+1j})$$

L'implémentation dérive directement de celle de PV :

```
d <- function(x, y){ # métrique de PV
  if(isTRUE(all.equal(x, y, check.names = FALSE))){
    return(0)
  } else {
    return(abs(x - y)/max(x, y))
  }
}
PVD <- function(x, i, ...){
  n <- length(x)
  xx <- x[i] # échantillon bootstrap ou x si i manquant
  dists <- numeric(n - 1) # distances entre toutes les paires consécutives
  ii <- 1
  for(k in seq_len(n - 1)){
    dists[ii] <- d(xx[k], xx[k + 1])
    ii <- ii + 1
  }
  return(100*mean(dists, ...))
}
PVD(c(0, 0))
[1] 0
PVD(c(0, 1))
[1] 100
PVD(1:10)
[1] 21.43298
all.equal(PVD(c(1:10)), PVD(c(NA, 1:10), na.rm = TRUE)) # TRUE, OK
[1] TRUE
```

Comme pour les statistiques d'auto-corrélation on ne peut pas calculer d'intervalle de confiance pour PVD, mais on peut avec un test de permutation des rangs voir que pour l'arbre n° 1012 la variabilité consécutive est supérieure à celle qui serait attendue par hasard en l'absence de structure temporelle :

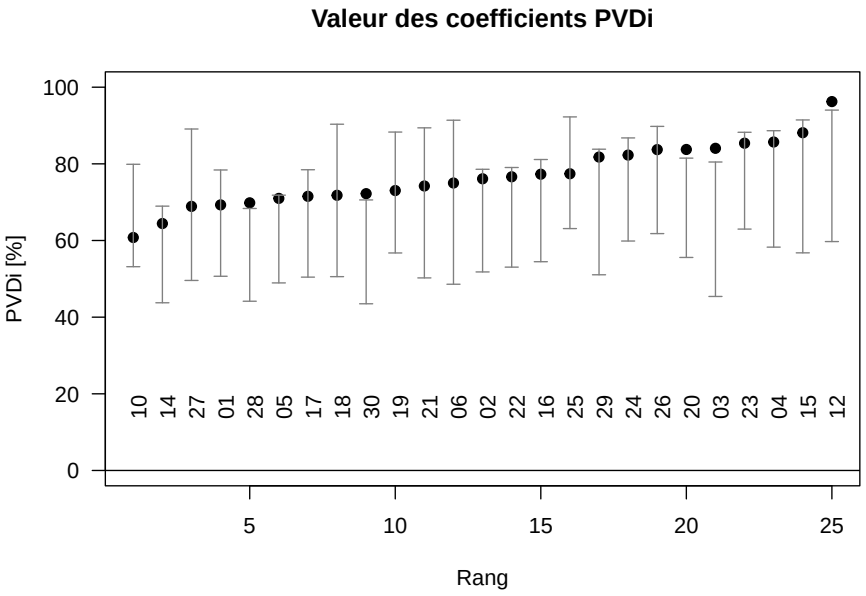
```
plotPVD <- function(x, plot = FALSE, alpha = 0.05, R = 9999, main = "", xlab = "", ...){
  t0 <- PVD(x, ...)
  t <- replicate(R, PVD(sample(x)))
  inf <- quantile(t, alpha/2, ...)
  sup <- quantile(t, 1 - alpha/2, ...)
  if(plot){
    plot(density(t, ...), main = main, xlab = xlab)
    abline(v = c(t0, inf, sup), col = c("blue", "red", "red"))
  }
  invisible(list(t0 = t0, inf = inf, sup = sup))
}
icol <- which(colnames(aps) == "T_1012")
plotPVD(aps[ , icol], plot = TRUE, main = "T_1012", R = 999, xlab = "PVD")
```



```
PVDi <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(PVDi) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  PVDi[i, "Tree_name"] <- colnames(aps)[i]
  Ares <- plotPVD(aps[ , i], plot = FALSE)
  PVDi[i, "est"] <- Ares$t0
  PVDi[i, "inf"] <- Ares$inf
  PVDi[i, "sup"] <- Ares$sup
}
save(PVDi, file = "PVDi.Rda")
```

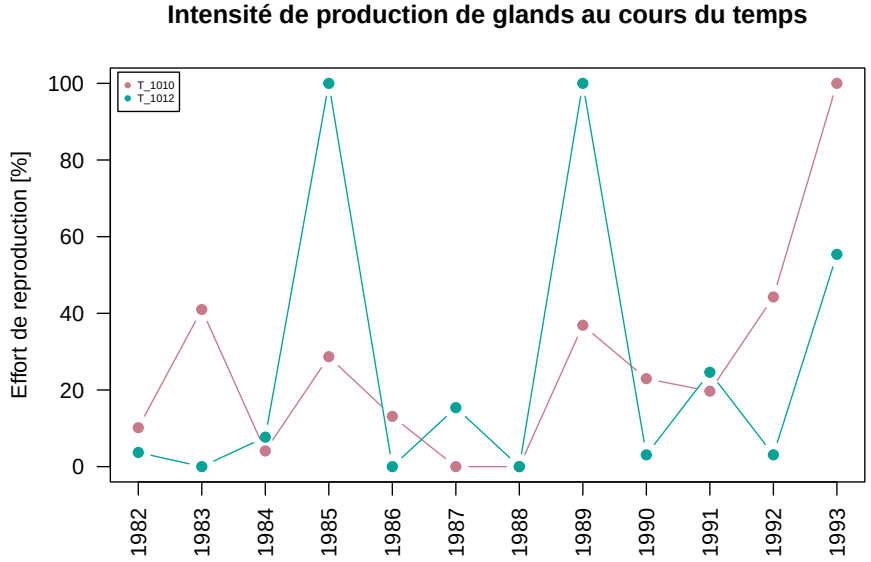
On voit qu'avec notre arbre n° 1012 on était plutôt chanceux puisque dans la majorité des cas on ne rejette pas l'hypothèse nulle.

```
load("PVDi.Rda")
PVDi <- PVDi[order(PVDi$est), ]
n <- nrow(PVDi)
plot(1:n, PVDi$est, pch = 19, ylim = c(0, 100),
     las = 1, ylab = "PVDi [%]", xlab = "Rang",
     main = "Valeur des coefficients PVDi")
with(PVDi, myarrows(1:n, inf, 1:n, sup))
abline(h = 0)
text(1:n, 20, substring(PVDi$Tree_name, 5, 6), srt = 90, pos = 1)
```



Si on représente les deux arbres extrêmes pour la valeur de PVD on constate le n° 1012 a un comportement bien plus chahuté que le n° 1010 :

```
load("apsr.Rda")
plot.tree(apsr, c("T_1012", "T_1010"), ylab = "Effort de reproduction [%]")
```

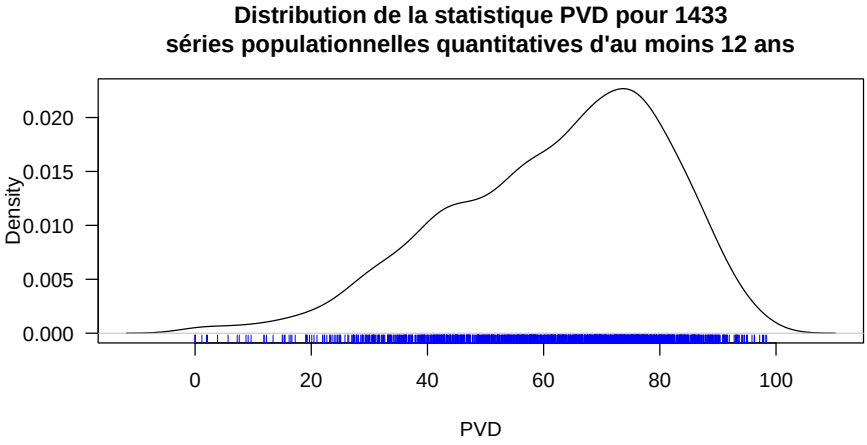


J'ai utilisé le script suivant pour calculer le coefficient d'auto-corrélation sur toutes les séries temporelles :

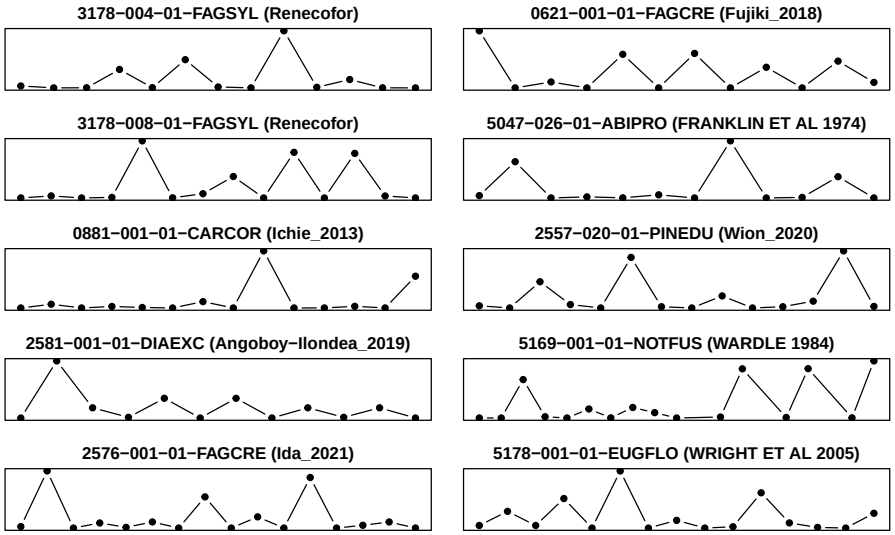
batch/getPVD.R

```
load("myws.Rda")
IDs <- unique(ms$ID) ; n <- length(IDs)
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 2))
colnames(tabres) <- c("ID", "est")

S <- function(x) PVD(x)
tabname <- "tabPVD"

for(i in seq_len(n)){
  the_ID <- IDs[i]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.res <- try(S(x))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.res
  assign(tabname, tabres)
  save(list = tabname, file = paste0(tabname, ".Rda"))
}
```

```
load("batch/tabPVD.Rda")
x <- tabPVD$est ; x <- x[is.finite(x)]
main <- paste0("Distribution de la statistique PVD pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "PVD", main = main, las = 1)
rug(x, col = "blue")
```



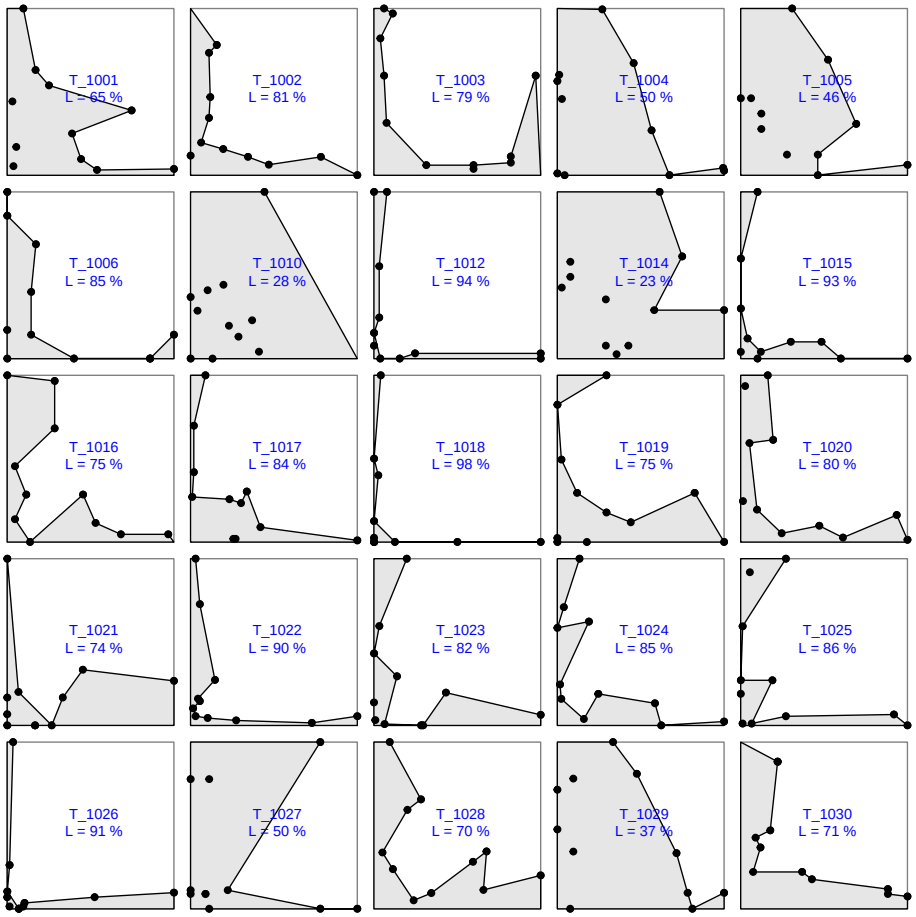
La statistique est toujours définie et avec un mode voisin de 75 %. Le top-10 des séries présentant la plus forte valeur de PVD est le suivant :



5.7 Hell statistic

Il me semble que ce que l'on cherche à caractériser c'est une forme en « L » du nuage de point dans le plan (x_t, x_{t+1}). Pour ce faire j'ai utilisé la fonction `concaveman()` du paquet éponyme [16] pour dessiner un polygone concave autour du « L » afin de pouvoir calculer sa surface. La statistique L est la surface relative de l'aire qui n'est pas occupée par le polygone. Le problème est que la recherche d'un polygone concave, à la différence d'un polygone convexe, n'est pas un problème bien défini puisque tout va dépendre du degré de concavité recherché. J'ai donc employé une méthode pifométrique [43] consistant à ajouter des points légèrement en retrait des deux axes pour forcer la forme en « L » et bidouillé la valeurs des paramètres pour obtenir les polygones que j'aurais aimé avoir si je les avais dessinés à la main.

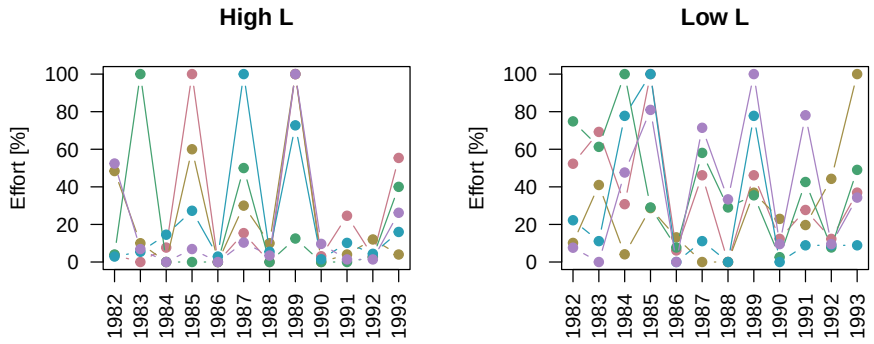
```
library(concaveman)
plotL <- function(x, nb = 64, lxfac = 1000, concavity = 1, plot = FALSE, ...){
  n <- length(x) ; xmax <- max(x)
  xn <-x[-n]    # Année n
  xnpu <- x[-1] # Année n + 1
  if(plot) plot(xn, xnpu, pch = 19, xlab = "Année n", ylab = "Année n + 1", las = 1,
     xlim = c(0, xmax), ylim = c(0, xmax), ...)
  if(plot) rect(0, 0, xmax, xmax, border = grey(0.5))
  lx <- -xmax/lxfac # On ajoute les branches du "L"
  matrix <- cbind(c(0, rep(lx, nb), seq(0, xmax, le = nb), xn),
                  c(0, seq(0, xmax, le = nb), rep(lx, nb), xnpu))
  cch <- concaveman(matrix, concavity = concavity)
  if(plot) polygon(cch, col = grey(0.9))
  if(plot) points(xn, xnpu, pch = 19, ...)
  cch <- cch[-1, ]   # On enleve le point de bouclage
  cch <- cch[nrow(cch):1, ]   # On met dans le sens trigonométrique
  A <- 0   # On calcule l'aire du polygone
  for(i in 1:(nrow(cch) - 1)) A <- A + cch[i, 1]*cch[i + 1, 2] - cch[i + 1, 1]*cch[i, 2]
  A <- A/2
  L <- 100*(xmax*xmax - A)/(xmax*xmax) # Aire relative du non-L
  if(plot) title(sub = paste0("L = ", signif(L, 3), " %"))
  invisible(L)
}
par(mfrow = c(5, 5), mar = c(0, 0, 0, 0) + 0.1)
for(i in 1:25){
  hell <- plotL(aps[, i], plot = TRUE, xaxt = "n", yaxt = "n", bty = "n")
  pu <- par("usr") ; tn <- colnames(aps)[i]
  text(pu[2]/2, pu[4]/2, paste0(tn, "\nL = ", signif(hell, 2), " %"), col = "blue")
}
```



Visuellement on distingue assez bien les arbres qui ont une statistique L

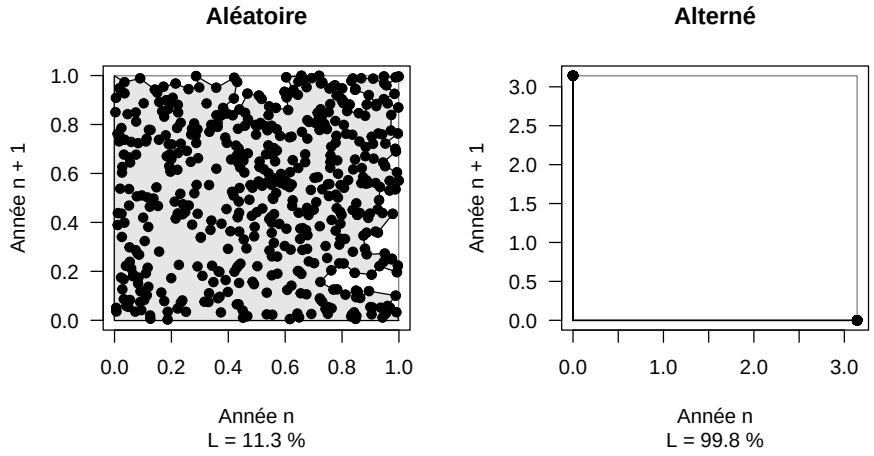
élevée de ceux qui ont une statistique L basse. Si on représente les arbres extrêmes :

```
par(mfrow = c(1, 2))
highL <- paste0("T_10", c(12, 15, 18, 22, 26))
plot.tree(apsr, highL, main = "High L", leg = FALSE, ylab = "Effort [%]")
lowL <- paste0("T_10", c("05", 10, 14, 27, 29))
plot.tree(apsr, lowL, main = "Low L", leg = FALSE, ylab = "Effort [%]")
```



Les valeurs de L sont comprises entre 0 et 100 %. On s'attend à avoir une valeur nulle pour une série aléatoire (sans structuration temporelle) et une valeur maximale pour une série alternant des zéros et une valeur non nulle :

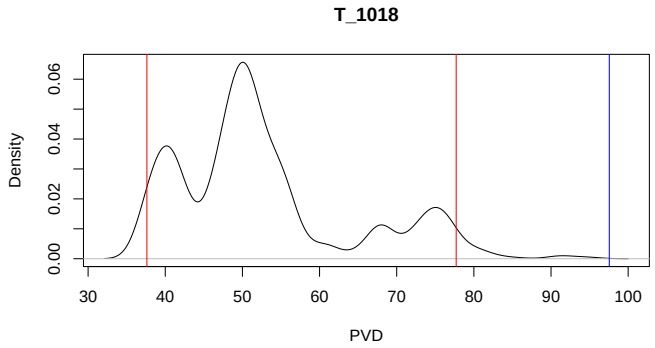
```
par(mfrow = c(1, 2))
plotL(runif(512), plot = TRUE, main = "Aléatoire")
plotL(rep(c(0, pi), 20), plot = TRUE, main = "Alterné")
```



Comme pour les autres statistiques d'auto-corrélation on peut faire un test de permutation des rangs :

```
plotLboot <- function(x, plot = FALSE, alpha = 0.05, R = 9999, main = "", xlab = "", ...){
  t0 <- plotL(x, plot = FALSE, ...)
  t <- replicate(R, plotL(sample(x), plot = FALSE, ...))
  inf <- quantile(t, alpha/2, ...)
  sup <- quantile(t, 1 - alpha/2, ...)
  if(plot){
```

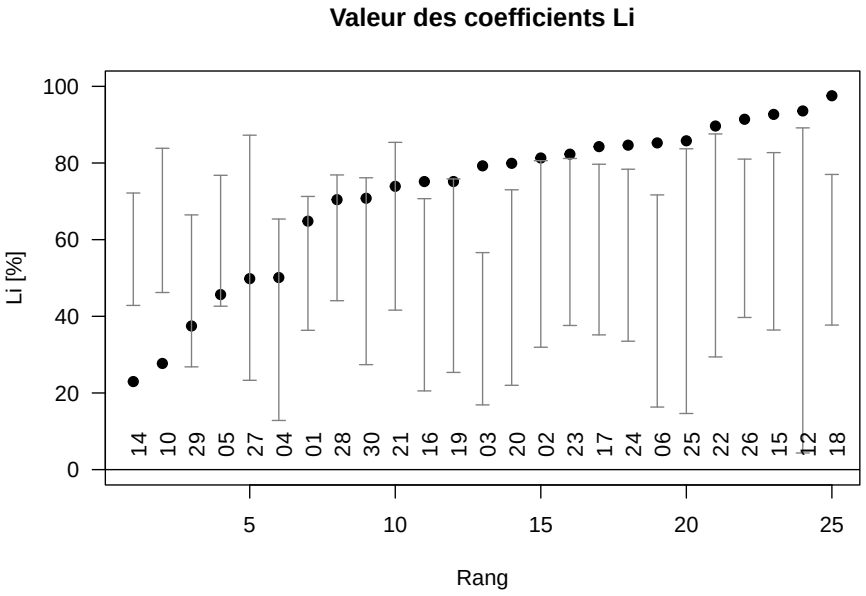
```
    plot(density(t, ...), main = main, xlab = xlab)
    abline(v = c(t0, inf, sup), col = c("blue", "red", "red"))
  }
  invisible(list(t0 = t0, inf = inf, sup = sup))
}
icol <- which(colnames(aps) == "T_1018")
plotLboot(aps[ , icol], plot = TRUE, main = "T_1018", R = 999, xlab = "PVD")
```



Si on répète la même opération pour tous les arbres on voit que l'on a des valeurs significatives en général quand la statistique est au dessus de 75 % :

```
Li <- as.data.frame(matrix(NA, nrow = ncol(aps), ncol = 4))
colnames(Li) <- c("Tree_name", "est", "inf", "sup")
for(i in seq_len(ncol(aps))){
  Li[i, "Tree_name"] <- colnames(aps)[i]
  Ares <- plotLboot(aps[ , i], plot = FALSE)
  Li[i, "est"] <- Ares$t0
  Li[i, "inf"] <- Ares$inf
  Li[i, "sup"] <- Ares$sup
}
save(Li, file = "Li.Rda")
```

```
load("Li.Rda")
Li <- Li[order(Li$est), ]
n <- nrow(Li)
plot(1:n, Li$est, pch = 19, ylim = c(0, 100),
     las = 1, ylab = "Li [%]", xlab = "Rang",
     main = "Valeur des coefficients Li")
with(Li, myarrows(1:n, inf, 1:n, sup))
abline(h = 0)
text(1:n, 10, substring(Li$Tree_name, 5, 6), srt = 90, pos = 1)
```

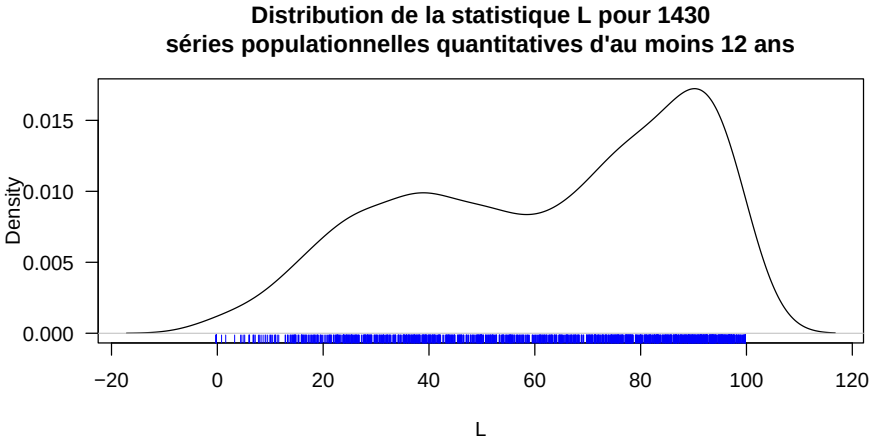


J'ai utilisé le script suivant pour calculer la statistique L sur toutes les séries temporelles :

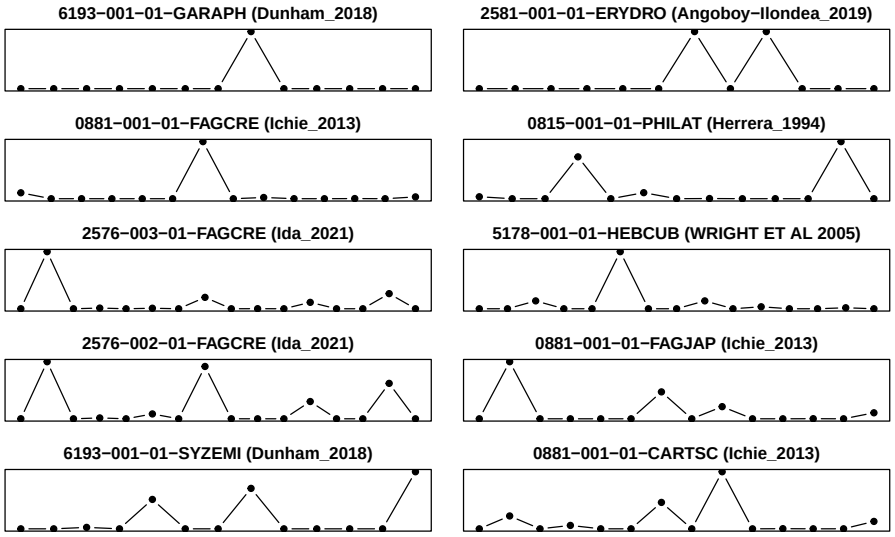
batch/getL.R

```
load("myws.Rda")
library(concaveman)

IDs <- unique(ms$ID) ; n <- length(IDs)
tabres <- as.data.frame(matrix(NA, nrow = n, ncol = 2))
colnames(tabres) <- c("ID", "est")

S <- function(x) plotL(x, plot = FALSE)
tabname <- "tabL"

for(i in seq_len(n)){
  the_ID <- IDs[i]
  tabres[i, "ID"] <- the_ID
  x <- ms[ms$ID == the_ID, "Value"]
  try.res <- try(S(x))
  if(!inherits(try.res, "try-error")) tabres[i, "est"] <- try.res
  assign(tabname, tabres)
  save(list = tabname, file = paste0(tabname, ".Rda"))
}
```

```
load("batch/tabL.Rda")
x <- tabL$est ; x <- x[is.finite(x)]
main <- paste0("Distribution de la statistique L pour ", length(x), "
séries populationnelles quantitatives d'au moins 12 ans")
plot(density(x), xlab = "L", main = main, las = 1)
rug(x, col = "blue")
```



La statistique n'est pas définie pour les trois séries constante. Ce qui est intéressant ici c'est que l'on a une distribution bimodale avec un pic voisin de 40 % et un pic voisin de 100 %. Le top-10 des séries présentant la plus forte statistique L est le suivant :



6 Remerciements

Grand merci à tous les co-auteurs de MASTREE+ [17] d'avoir partagé leurs données. Une mention spéciale à tous les compilateurs pour leur orpaillage la littérature : Andrew Hacket-Pain, Davide Ascoli, Davide Vecchio, Éliane Schermer, Ian Pearse, Jalene LaMontagne, Joep van Dormolen, Jose Moris, Thomas Caignard et Tingting Xue. Merci également au réseau RENECOFOR, à l'université de Stirling et à l'Agence Nationale des Parcs Nationaux (ANPN) du Gabon. Merci à Thomas Caignard et François Débias pour les photos. Ce travail a été financé en partie par le programme ANR-19-CE32-0008.

References

[1] T. Allison and D.V. Cicchetti. Sleep in mammals: ecological and constitutional correlates. *Science*, 194:732–734, 1976.

[2] B. Angoboy Ilondea, H. Beeckman, D.-Y. Ouédraogo, N. Bourland, T. De Mil, J. Van Den Bulcke, J. Van Acker, C. Couralet, C. Ewango, W. Hubau, B. Toirambe, J.-L. Doucet, and A. Fayolle. Une forte saisonnalité du climat et de la phénologie reproductive dans la forêt du mayombe: l'apport des données historiques de la réserve de Luki en République démocratique du Congo. *Bois et Forêts des Tropiques*, 341:39–53, 2019.

[3] F.J. Anscombe. The statistical analysis of insect counts based on the negative binomial distribution. *Biometrics*, 5(2):165–173, 1949.

[4] J.P. Buonaccorsi, J. Elkinton, W. Koenig, R.P. Duncan, D. Kelly, and V. Sork. Measuring mast seeding behavior: relationships among population variation, individual variation and synchrony. *Journal of Theoretical Biology*, 224:107–114, 2003.

[5] A. Canty and B.D. Ripley. *boot: Bootstrap R (S-Plus) Functions*, 2021. R package version 1.3-28.

[6] M.J. Crawley and C.R. Long. Alternate bearing, predator satiation and seedling recruitment in *Quercus robur* L. *Journal of Ecology*, 83:683–696, 1995.

[7] J.T. Curtis. *The vegetation of Wisconsin: an ordination of plant communities*. University of Wisconsin Press, 1930 Monroe Street, Madison, Wisconsin 53711, United States of America, 1959.

[8] D.V. Davison, A.C. Hinkley. *Bootstrap Methods and Their Applications*. Cambridge University Press, Cambridge, 1997. ISBN 0-521-57391-2.

[9] M.E. DePrenger-Levin, J.M. Ramp Neale, T.A. Grant, C. Dawson, and Y.E. Baytok. Life history and demography of *Astragalus microcymbus* Barneby (Fabaceae). *Natural areas journal*, 33:264–275, 2013.

[10] A.E. Dunham, O.H. Razafindratsima, P. Rakotonirina, and P.C. Wright. Fruiting phenology is linked to rainfall variability in a tropical rain forest. *Biotropica*, 50:396–404, 2018.

[11] B. Efron. Better bootstrap confidence intervals. *Journal of the American statistical Association*, 82(397):171–185, 1987.

[12] J.G. Eisenhauer. A measure of relative dispersion. *Teaching Statistics*, 15(2):37–39, 1993.

[13] M. Fernández-Martínez, S. Vicca, I.A. Janssens, J.M. Espelta, and J. Peñuelas. The role of nutrients, productivity and climate in determining tree fruit production in european forests. *New Phytologist*, 213:669–679, 2017.

[14] M. Fernández-Martínez, S. Vicca, I.A. Janssens, J. Carnicer, J. Martín-Vide, and J. Peñuelas. The consecutive disparity index, D: a measure of temporal variability in ecological studies. *Ecosphere*, 9(12):e02527, 2018.

[15] K.J. Gaston and B.H. McArdle. The temporal variability of animal abundances: measures, methods and patterns. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 345(1314):335–358, 1994.

[16] J. Gombin, R. Vaidyanathan, and V. Agafonkin. *concaveman: A Very Fast 2D Concave Hull Algorithm*, 2020. R package version 1.1.0.

[17] A. Hacket-Pain, J.J. Foest, Ian S. Pearse, Jalene M. LaMontagne, Walter D. Koenig, Giorgio Vacchiano, Michał Bogdziewicz, Thomas Caignard, Paulina Celebias, Joep van Dormolen, Marcos Fernández-Martínez, Jose V. Moris, Ciprian Palaghianu, Mario Pesendorfer, Akiko Satake, Eliane Schermer, Andrew J. Tanentzap, Peter A. Thomas, Davide Vecchio, Andreas P. Wion, Thomas Wohlgemuth, Tingting Xue, Katharine Abernethy, Marie-Claire Aravena Acuña, Marcelo Daniel Barrera, Jessica H. Barton, Stan Boutin, Emma R. Bush, Sergio Donoso Calderón, Felipe S. Carevic, Carolina Volkmer de Castilho, Juan Manuel Cellini, Colin A. Chapman, Hazel Chapman, Francesco Chianucci, Patricia da Costa, Luc Croisé, Andrea Cutini, Ben Dantzer, R. Justin DeRose, Jean-Thoussaint Dikangadissi, Edmond Dimoto, Fernanda Lopes da Fonseca, Leonardo Gallo, Georg Gratzer, David F. Greene, Martín A. Hadad, Alejandro Huertas Herrera, Kathryn J. Jeffery, Jill F. Johnstone, Urs Kalbitzer, Władysław Kantorowicz, Christie A. Klimas, Jonathan G. A. Lageard, Jeffrey Lane, Katharina Lapin, Mateusz Ledwoń, Abigail C. Leeper, Maria Vanessa Lencinas, Ana Cláudia Lira-Guedes, Michael C. Lordon, Paula Marchelli, Shealyn Marino, Harald Schmidt Van Marle, Andrew G. McAdam, Ludovic R. W. Momont, Manuel Nicolas, Lúcia Helena de Oliveira Wadt, Parisa Panahi, Guillermo Martínez Pastur, Thomas Patterson, Pablo Luis Peri, Łukasz Piechnik, Mehdi Pourhashemi, Claudia Espinoza Quezada, Fidel A. Roig, Karen Peña Rojas, Yamina Micaela Rosas, Silvio Schueler, Barbara Seget, Rosina Soler, Michael A. Steele, Mónica Toro-Manríquez, Caroline E. G. Tutin, Tharcisse Ukizintambara, Lee White, Biplang Yadok, John L. Willis, Anita Zolles, Magdalena Żywiec, and Davide Ascoli. Mastree+: time-series of plant reproductive effort from six continents. *Global Change Biology*, 00:1–17, 2022.

[18] W. Hase. Die Buchenmast in Schleswig-Holstein und ihre Abhängigkeit von der Witterung. *Mitt. Deutsch. Wetterdienst*, 31:3–45, 1964.

[19] J.P. Heath. Quantifying temporal variability in population abundances. *Oikos*, 115(3):573–581, 2006.

[20] J.P. Heath and P. Borowski. Quantifying proportional variability. *PLoS One*, 8(12):e84074, 2013.

[21] C.M. Herrera. Population-level estimates of interannual variability in seed production: what do they actually tell us? *Oikos*, 82:612–616, 1998.

[22] C.M. Herrera, Jordano P., Guitián J, and A. Traveset. Annual variability in seed production by woody plants and the masting concept: reassessment of principles and relationship to pollination and seed dispersal. *The American Naturalist*, 152:576–594, 1998.

[23] T. Ichie, S. Igarashi, S. Yoshida, T. Kenzo, T. Masaki, and I. Tayasu. Are stored carbohydrates necessary for seed production in temperate deciduous trees? *Journal of Ecology*, 101:525–531, 2013.

[24] Y. Isagi, K. Sugimura, A. Sumida, and H. Ito. How does masting happen and synchronize? *Journal of Theoretical Biology*, 187(2):231–239, 1997.

[25] D.H. Janzen. Seed predation by animals. *Annual review of ecology and systematics*, 2(1):465–492, 1971.

[26] D.H. Janzen. Tropical blackwater rivers, animals, and mast fruiting by the dipterocarpaceae. *Biotropica*, pages 69–103, 1974.

[27] D.H. Janzen. Why bamboos wait so long to flower. *Annual Review of Ecology and systematics*, 7:347–391, 1976.

[28] D. Kelly. The evolutionary ecology of mast seeding. *Trends in ecology & evolution*, 9:465–470, 1994.

[29] M.G. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[30] M.G. Kendall. The treatment of ties in ranking problems. *Biometrika*, 33:239–251, 1945.

[31] A. Kimber. What a scorcher! an analysis of some temperature data. *Teaching Statistics*, 13(2):34–37, 1991.

[32] W.D. Koenig. A brief history of masting research. *Philosophical Transactions of the Royal Society B*, 376(1839):20200423, 2021.

[33] W.D. Koenig, D. Kelly, V.L. Sork, R.P. Duncan, J.S. Elkinton, M.S. Peltonen, and R.D. Westfall. Dissecting components of population-level variation in seed production and the evolution of masting behavior. *Oikos*, 102(3):581–591, 2003.

[34] R.C. Lewontin. On the measurement of relative variability. *Systematic Zoology*, 15(2):141–142, 1966.

[35] X. Martín-Vide. Notes per a la definició d'un índex de «desordre» en pluviometria. *Societat Catalana de Geografia*, 7:89–96, 1986.

[36] B.H. McArdle, K.J. Gaston, and J.H. Lawton. Variation in the size of animal populations: patterns, problems and artefacts. *The Journal of Animal Ecology*, 59:439–454, 1990.

[37] P.M. McDonald. Estimating seed crops of conifer and hardwood species. *Canadian Journal of Forest Research - Revue Canadienne De Recherche Forestiere*, 22:832–838, 1992.

[38] C. Michalski. Le droit à la glandée. *Revue forestière française*, 57:377–391, 2005.

[39] P. Murrell. *hexView: Viewing Binary Files*, 2019. R package version 0.3-4.

[40] S. Mutke, J. Gordo, and L. Gil. Variability of mediterranean stone pine cone production: Yield loss as response to climate change. *Agricultural and Forest Meteorology*, 132:263–272, 2005.

[41] K. Pearson. VII. note on regression and inheritance in the case of two parents. *Proceedings of the royal society of London*, 58:240–242, 1895.

[42] K. Pearson. VII. mathematical contributions to the theory of evolution.—III. regression, heredity, and panmixia. *Philosophical Transactions of the Royal Society of London. Series A*, (187):253–318, 1896.

[43] G. Pif. Sideral glasses to increase the accuracy of bull's-eyes fit method in non-linear regression. *Pif Gadjet*, 1:1–48, 1969.

[44] M.J. Rogers, L.K. Halls, and J.G. Dickson. *Deer habitat in the Ozark forests of Arkansas*, volume Research Paper SO-259. United States Department of Agriculture, Forest Service, Southern Forest Experiment Station, New Orleans, Louisiana, United States of America, 1990.

[45] A. Satake and Y. Iwasa. Pollen coupling of forest trees: forming synchronized and periodic reproduction out of chaos. *Journal of theoretical biology*, 203(2):63–84, 2000.

[46] A. Satake, Leong Y.T., Y. Kosugi, and Y.-Y. Chen. Testing the environmental prediction hypothesis for community-wide mass flowering in southeast asia. *Biotropica*, 53:608–618, 2021.

[47] J.W. Silvertown. The evolutionary ecology of mast seeding in trees. *Biological journal of the Linnean Society*, 14:235–250, 1980.

[48] G.G. Simpson, A. Roe, and R.C. Lewontin. *Quantitative zoology*. Harcourt, Brace and Co., Inc., New York, U.S.A., 1960.

[49] C. Spearman. The proof and measurement of association bteween two things. *The American Journal of Psychology*, 15:72–101, 1904.

[50] Sun Microsystems. XDR: external data representation standard. RFC 1014. Technical report, Network Working Group, 1987.

[51] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002.

[52] J.A. Wardle. *The New Zealand beeches: ecology, utilisation, and management*. White Lion Publishing, 1984.

[53] C.J. Webb and D. Kelly. The reproductive biology of the New Zealand flora. *Trends in Ecology & Evolution*, 8:442–447, 1993.

[54] C.B. Williams. The use of logarithms in the interpretation of certain entomological problems. *Annals of Applied Biology*, 24(2):404–414, 1937.

[55] A.P. Wion, P.J. Weisberg, I.S. Pearse, and M.D. Redmond. Aridity drives spatiotemporal patterns of masting across the latitudinal range of a dryland conifer. *Ecography*, 43:569–580, 2020.

[56] M.C. Wright, P. van Mantgem, N.L. Stephenson, A.J. Das, and J.E. Keeley. Seed production patterns of surviving Sierra Nevada conifers show minimal change following drought. *Forest Ecology and Management*, 480:118598, 2021.