Cas estudi: Revisiting Dinosaur's extinction (Shannon entropy and Boostrap Confidence Interval)

Jordi Ocaña, Sergi Civit,

7 de maig de 2018

1 Cas Estudi

Nombre d'espècies de dinosaures per família observades durant tres perìodes del Cretaci a Dakota del Nord i Montana (Sheehan, P.M., Fastovsky, D.E., Hoffmann, R.G., Berghaus, C.B., Gabriel, D.L. (1991). Sudden extinction of the dinosaurs: Latest Cretaceous, Upper Great Plains, U.S.A. Science, 254, 835-839)

```
> family <- factor(c("Ceratops","Hadro","Pachycephalo","Tyranno","Ornithomo",
+                    "Sauronith"),
+                  levels=c("Ceratops","Hadro","Pachycephalo","Tyranno",
+                           "Ornithomo","Sauronith"))
> dinos <- data.frame(
+   upper = c(50, 29, 3, 3, 4, 1),
+   middle= c(53, 51, 2, 3, 8, 6),
+   lower = c(19,  7, 1, 2, 1, 3)
+ )
> rownames(dinos) <- family
> dinos
```

```
             upper middle lower
Ceratops        50     53    19
Hadro           29     51     7
Pachycephalo     3      2     1
Tyranno          3      3     2
Ornithomo        4      8     1
Sauronith        1      6     3
```

Si la biodiversitat (en termes de nombre d'espècies per família) havia anat decreixent al llarg dels tres períodes, possiblement es podria considerar un indici que els dinosaures estaven immersos en una çrisi"que ja venia de lluny. La confrontació d'aquesta possibilitat, davant **la hipòtesi que la biodiversitat s'havia mantingut estable**, requeriria mesurar la diversitat mitjançant un índex objectiu, com ara **l'índex de Shannon - Margalef**.

Proposat per Shannon en el context de la teoria de la informació, Margalef va proposar d'utilitzar-lo en Ecologia com a mesura de biodiversitat.

2 Revisiting the Dinosaur extinction. Authors MNPR 17-18

A progressive decrease of biodiversity across the three periods would agree with an hypothesis of dinosaur progressive decline, while biodiversity stability would be more consistent with the possibility of a catastrophic event as the cause of dinosaur extinction.

Contrasting these possibilities requires the use of an objective biodiversity index, like the Shannon information measure, whose use in a biodiversity context was firstly proposed by Margalef.

Function evaluating the Shannon index from an absolute frequencies vector. These absolute frequencies are internally converted into relative frequencies. In fact, any vector of non-negative values (e.g. surface covered by distinct species) is admissible to compute the index.

```
> #
> shannon <- function(freq.abs)
+ {
+         freq.rels = freq.abs / sum(freq.abs)
+         - sum(ifelse(freq.rels > 0, freq.rels * log2(freq.rels), 0))
+ }
```

Examples of its use:

```
> # Diversity over a given period:
> shannon(dinos[,"middle"])

[1] 1.746269

> shannon(dinos[,2])

[1] 1.746269

> # Diversity for all periods:
> apply(dinos, 2, shannon)

   upper   middle    lower
1.596475 1.746269 1.798425

> vapply(dinos, shannon, FUN.VALUE = 0.0)
```

```
   upper   middle    lower 
1.596475 1.746269 1.798425 
```

Biodiversity seems to be stable, even to increase...

In the before mentioned paper, the authors take the approach of testing for differences between the biodiversities in all the periods (that is to say, H0: all diversities are equal vs H1: almost one pair of diversities differ

3 Inference on a single diversity measure

Let us make some inference over these data. First, we will compute confidence intervals for the biodiversity.

We include, as an option, the computation of the standard error of the Shannon index in function 'shannon':

```
> shannon <- function(freq.abs, compute.se = TRUE)
+ {
+   n <- sum(freq.abs)
+   freq.rels = freq.abs / n
+         log2freq.rels <- ifelse(freq.rels > 0, log2(freq.rels), 0)
+         result <- - sum(freq.rels * log2freq.rels)
+         if (compute.se) {
+     attr(result, "se") <- sqrt((sum(freq.rels * log2freq.rels^2) - result^2) / n)
+         }
+         return(result)
+ }
> biodivs <- lapply(dinos, shannon)
> biodivs

$upper
[1] 1.596475
attr(,"se")
[1] 0.1366722

$middle
[1] 1.746269
attr(,"se")
[1] 0.1091939

$lower
[1] 1.798425
attr(,"se")
[1] 0.234451

> unlist(biodivs)
```

```
   upper   middle    lower 
1.596475 1.746269 1.798425 

> sapply(biodivs, attr, "se")

    upper    middle     lower 
0.1366722 0.1091939 0.2344510 

> biodivs <- sapply(dinos, shannon, compute.se = FALSE)
> biodivs

   upper   middle    lower 
1.596475 1.746269 1.798425 

> sapply(biodivs, attr, "se")

$upper
NULL

$middle
NULL

$lower
NULL
```

We assume that each column in 'dinos' is a multinomial sample from unknown "true"probabilities and 'size' parameter = sum(frequencies). Thus, the relative frequencies are an estimate of these probabilities and each (parametric?) bootstrap resample may be generated from a multinomial

Mn(size, relative frequencies)

Caution: in the R function 'rmultinom' the first argument, 'n', stands for the number of multinomial samples to be generated (here, in a bootstrap context, this will correspond to the number of bootstrap resamples, say B) while the second argument, 'size' corresponds to the "n"multinomial parameter

E.g. for the ̈upper"period:

```
> freq.abs <- dinos[,"upper"]
> freq.abs

[1] 50 29  3  3  4  1

> shannon.sample <- shannon(freq.abs)
> shannon.sample

[1] 1.596475
attr(,"se")
[1] 0.1366722
```

```
> n <- sum(freq.abs)
> n

[1] 90

> freq.rels <- freq.abs / n
> freq.rels

[1] 0.55555556 0.32222222 0.03333333 0.03333333 0.04444444 0.01111111

> # One bootstrap multinomial resample:
>
> set.seed(12345)
> freq.boot <- rmultinom(1, size = n, prob = freq.rels)
> freq.boot

     [,1]
[1,]   56
[2,]   26
[3,]    2
[4,]    3
[5,]    1
[6,]    2

> # Caution: it is a 'k x one-column matrix' with k = length(freq.rels)
```

Next session

4 Bootstrap-t Confidence Interval

```
> # studentized statistic over the preceding bootstrap sample:
> shannon.boot <- shannon(freq.boot)
> (shannon.boot - shannon.sample) / attr(shannon.boot, "se")

[1] -1.223689
attr(,"se")
[1] 0.1415952

> # Function computing the studentized statistic:
> tShannon <- function(freq.abs, shannon.value) {
+   shannon.sampl <- shannon(freq.abs)
+   (shannon.sampl - shannon.value) / attr(shannon.sampl, "se")
+ }
> tShannon(freq.boot[,1], shannon.sample)

[1] -1.223689
attr(,"se")
[1] 0.1415952
```

```
> nboot <- 10
> # 'nboot' bootstrap resamples:
> set.seed(12345)
> rmultinom(nboot, size = n, prob = freq.rels)

     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   56   51   46   50   46   42   54   46   46    54
[2,]   26   27   35   27   29   37   21   32   37    23
[3,]    2    0    5    0    6    2    6    2    2     4
[4,]    3    4    1    3    4    1    3    5    4     0
[5,]    1    7    3    7    5    8    6    5    1     9
[6,]    2    1    0    3    0    0    0    0    0     0

> # Studentized statistic over each resample:
> set.seed(12345)
> apply(rmultinom(nboot, size = n, prob = freq.rels), 2, tShannon, shannon.value = shannon.s

 [1] -1.22368895 -0.43462940 -0.89915318  0.07590594  0.97634418 -0.47183546
 [7]  0.15158507  0.11941429 -1.57301290 -1.06391798

> alpha <- 0.05
> nboot <- 10000
> set.seed(12345)
> tBoots <- apply(rmultinom(nboot, size = n, prob = freq.rels), 2, tShannon, shannon.value =
```

4.1 Boostrap-t Confidence Interval(equiprobability tails)

```
> # Bootstrap-t (equals tails) confidence interval:
> halfAlpha <- alpha / 2
> tLimits <- quantile(tBoots, probs = c(1 - halfAlpha, halfAlpha))
> tLimits

    97.5%      2.5%
 1.718856 -2.731609

> ci <- shannon.sample - tLimits * attr(shannon.sample, "se")
> names(ci) = NULL
> attr(ci, "conf.level") = 1 - alpha
> ci

[1] 1.361555 1.969810
attr(,"conf.level")
[1] 0.95
```

4.2 Symmetrized Boostrap-t Confidence Interval

```
> # Symmetrized bootstrap-t confidence interval:
> tLimit <- quantile(abs(tBoots), probs = 1 - alpha)
> ci <- shannon.sample - c(tLimit, -tLimit) * attr(shannon.sample, "se")
> names(ci) = NULL
> attr(ci, "conf.level") = 1 - alpha
> ci
```

```
[1] 1.271795 1.921155
attr(,"conf.level")
[1] 0.95
```

4.3 Boostrap-t Confidence Interval: Own R Function

```
> # All encapsulated in a function:
> shannonBoot.t <- function(freq.abs, alpha = 0.05, symmetrized = FALSE, nboot = 10000)
+ {
+   shannon.sample <- shannon(freq.abs)
+   n <- sum(freq.abs)
+   tBoots <- apply(
+               rmultinom(nboot, size = n, prob = freq.abs / n), 2, tShannon,
+               shannon.value = shannon.sample
+   )
+   if (symmetrized) {
+     tLimit <- quantile(abs(tBoots), probs = 1 - alpha)
+     ci <- shannon.sample - c(tLimit, -tLimit) * attr(shannon.sample, "se")
+   } else {
+     halfAlpha <- alpha / 2
+     tLimits <- quantile(tBoots, probs = c(1 - halfAlpha, halfAlpha))
+     ci <- shannon.sample - tLimits * attr(shannon.sample, "se")
+   }
+   names(ci) = NULL
+   attr(ci, "conf.level") = 1 - alpha
+   return(ci)
+ }
> shannonBoot.t(dinos[,"upper"])
```

```
[1] 1.363588 1.971527
attr(,"conf.level")
[1] 0.95
```

```
> sapply(dinos, shannonBoot.t)
```

```
        upper   middle    lower
[1,] 1.355335 1.555341 1.383789
[2,] 1.972916 2.015863 2.541113
```

```
> shannonBoot.t(dinos[,"upper"], symmetrized = TRUE)
```

```
[1] 1.266284 1.926665
attr(,"conf.level")
[1] 0.95
```

```
> sapply(dinos, shannonBoot.t, symmetrized = TRUE)

        upper   middle    lower
[1,] 1.271392 1.507066 1.176602
[2,] 1.921558 1.985473 2.420248
```

5 Bootstrap-p Confidence Interval

```
> # PERCENTILE BOOTSTRAP INTERVAL:
> # This is the simplest to compute bootstrap confidence interval, it only requires
> # the generation of a sample of bootstrap parameter estimations, say, here Shannon's
> # biodiversities
>
> nboot = 10000
> set.seed(12345)
> shann.boot = replicate(nboot, shannon(rmultinom(1, size = n, prob = freq.rels), compute.se
> ci = quantile(shann.boot, probs = c(alpha/2, 1 - alpha/2))
> names(ci) = NULL
> attr(ci, "conf.level") = 1 - alpha
> ci

[1] 1.270338 1.815184
attr(,"conf.level")
[1] 0.95
```

5.1 Boostrap-p Confidence Interval: Own R Function

```
> # Percentile CI as a function:
> shannonBoot.perc <- function(freq.abs, alpha = 0.05, nboot = 10000)
+ {
+   n <- sum(freq.abs)
+   shannBoots <- apply(rmultinom(nboot, size = n, prob = freq.abs / n), 2, shannon, compute
+   ci = quantile(shannBoots, probs = c(alpha/2, 1- alpha/2))
+   names(ci) = NULL
+   attr(ci, "conf.level") = 1 - alpha
+   return(ci)
+ }
> # Some examples of its use:
> p = c(0.6, 0.2, 0.1, 0.05, 0.025, 0.025)
> n = 100
> shann.real = shannon(p, compute.se = FALSE)
> shann.real

[1] 1.720951
```

6 Boostrap-t and Boostrap-p: Naive simulation study

```
> # A very naive simulation (that would need a lot of improvement)
> # comparing the true coverage of bootstrap-t and percentile bootstrap intervals
> # for a given configuration of hypothetic multinomial parameters:
> p = c(0.6, 0.2, 0.1, 0.05, 0.025, 0.025)
> n = 100
> set.seed(12345)
> nsim = 100
> res = replicate(nsim,
+   {
+     mostra = rmultinom(1, size = n, prob = p)
+     ic.perc = shannonBoot.perc(mostra, nboot = 1000)
+     recob.perc = (ic.perc[1] <= shann.real) & (shann.real <= ic.perc[2])
+     ic.tSym = shannonBoot.t(mostra, symmetrized = TRUE, nboot = 1000)
+     recob.tSym = (ic.tSym[1] <= shann.real) & (shann.real <= ic.tSym[2])
+     c(recob.perc, recob.tSym)
+   }
+ )
> rownames(res) = c("percentile", "symmetrized bootstrap-t")
> rowMeans(res)
```

```
            percentile symmetrized bootstrap-t 
                  0.88                    0.96 
```

7 Boostrap-p BCa Confidence Interval

```
> # BCa INTERVAL:
> 
> # BCa bootstrap interval for the Shannon biodiversity:
> 
> # Obtaining the Shannon's index jaccknife values
> # E.g. for the "upper" period:
> freq.abs <- dinos[,"upper"]
> freq.abs
```

```
[1] 50 29  3  3  4  1
```

```
> # Family "Ceratops" has a frequency of 50: there are 50 opportunities to remove
> # one data of this family, but all 'shannon' values will be equal, they will be
> # based on the frequencies: 50-1 29  3  3  4  1. Similarly, there will be
> # 29 repeated values from the frequencies 50 29-1  3  3  4  1, etc
> 
> k <- length(freq.abs)
> matrix(freq.abs, ncol = k, nrow = k)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   50   50   50   50   50   50
[2,]   29   29   29   29   29   29
[3,]    3    3    3    3    3    3
[4,]    3    3    3    3    3    3
[5,]    4    4    4    4    4    4
[6,]    1    1    1    1    1    1

> diag(ncol = k, nrow = k)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    0    0    0    0    0
[2,]    0    1    0    0    0    0
[3,]    0    0    1    0    0    0
[4,]    0    0    0    1    0    0
[5,]    0    0    0    0    1    0
[6,]    0    0    0    0    0    1

> matrix(freq.abs, ncol = k, nrow = k) - diag(ncol = k, nrow = k)

     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   49   50   50   50   50   50
[2,]   29   28   29   29   29   29
[3,]    3    3    2    3    3    3
[4,]    3    3    3    2    3    3
[5,]    4    4    4    4    3    4
[6,]    1    1    1    1    1    0

> shannon.jack = apply(matrix(freq.abs, ncol = k, nrow = k) - diag(ncol = k, nrow = k),
+   2, shannon, compute.se = FALSE)
> shannon.jack

[1] 1.604812 1.595862 1.556305 1.556305 1.561813 1.525351

> # Mean of the jackknife values:
> shannon.jack. = weighted.mean(shannon.jack, w = freq.abs)
> a <- sum((shannon.jack. - shannon.jack)^3 * freq.abs) /
+   (6 * sum((shannon.jack. - shannon.jack)^2 * freq.abs)^1.5)
> a

[1] 0.04044708

> shannon.sample <- shannon(freq.abs, compute.se = FALSE)
> z0 <- qnorm(sum(shann.boot <= shannon.sample) / nboot)
> z0

[1] 0.2931598
```

```
> zAlpha <- - qnorm(alpha/2)
> zAlpha

[1] 1.959964

> p1 <- pnorm(z0 + (z0 - zAlpha) / (1 - a * (z0 - zAlpha)))
> p2 <- pnorm(z0 + (z0 + zAlpha) / (1 - a * (z0 + zAlpha)))
> ci = quantile(shann.boot, probs = c(p1, p2))
> names(ci) = NULL
> attr(ci, "conf.level") = 1 - alpha
> ci

[1] 1.373911 1.927913
attr(,"conf.level")
[1] 0.95
```

7.1 Boostrap-p BCa Confidence Interval. Own R function

```
> # The above calculations may be encapsulated in a function:
>
> # Ad hoc utility function returning the centered jackknife values:
> shannon.BCa.jackVals <- function(freq.abs) {
+   k <- length(freq.abs)
+   jack.vals <- apply(
+     matrix(freq.abs, ncol = k, nrow = k) - diag(ncol = k, nrow = k),
+     2, shannon, compute.se = FALSE
+   )
+   return(weighted.mean(jack.vals, w = freq.abs) - jack.vals)
+ }
> # BCa confidence interval:
> shannonBoot.BCa <- function(freq.abs, alpha = 0.05, nboot = 10000) {
+   shannon.sample <- shannon(freq.abs, compute.se = FALSE)
+   n <- sum(freq.abs)
+   freq.rels = freq.abs / n
+   shannonBoots <- apply(
+     rmultinom(nboot, size = n, prob = freq.rels), 2, shannon,
+     compute.se = FALSE
+   )
+   shannon.jack <- shannon.BCa.jackVals(freq.abs)
+   shannon.jack2 <- shannon.jack * shannon.jack * freq.abs
+   a <- sum(shannon.jack * shannon.jack2) / (6 * sum(shannon.jack2)^1.5)
+   z0 <- qnorm(sum(shannonBoots <= shannon.sample) / nboot)
+   zAlpha <- - qnorm(alpha/2)
+   p1 <- pnorm(z0 + (z0 - zAlpha) / (1 - a * (z0 - zAlpha)))
+   p2 <- pnorm(z0 + (z0 + zAlpha) / (1 - a * (z0 + zAlpha)))
+   ci = quantile(shannonBoots, probs = c(p1, p2))
+   names(ci) = NULL
+   attr(ci, "conf.level") = 1 - alpha
+   attr(ci, "a") = a
+   attr(ci, "z0") = z0
+   return(ci)
+ }
> set.seed(12345)
> shannonBoot.BCa(freq.abs)
```

```
[1] 1.373911 1.927913
attr(,"conf.level")
[1] 0.95
attr(,"a")
[1] 0.04044708
attr(,"z0")
[1] 0.2931598
```

8 Boostrap-p BCa and symmetrized Boostrap-t: Naive simulation study

```
> # A very naive simulation (that would need a lot of improvement)
> # comparing the true coverage of the percentile, BCa and symmetrized bootstrap-t
> # intervals for a given configuration of hypothetic multinomial parameters:
>
> nsim = 1000
> res = replicate(nsim,
+ {
+   mostra = rmultinom(1, size = n, prob = p)
+   ic.perc = shannonBoot.perc(mostra, nboot = 1000)
+   recob.perc = (ic.perc[1] <= shann.real) & (shann.real <= ic.perc[2])
+   ic.BCa = shannonBoot.BCa(mostra, nboot = 1000)
+   recob.BCa = (ic.BCa[1] <= shann.real) & (shann.real <= ic.BCa[2])
+   ic.t = shannonBoot.t(mostra, symmetrized = FALSE, nboot = 1000)
+   recob.t = (ic.t[1] <= shann.real) & (shann.real <= ic.t[2])
+   ic.tSym = shannonBoot.t(mostra, symmetrized = TRUE, nboot = 1000)
+   recob.tSym = (ic.tSym[1] <= shann.real) & (shann.real <= ic.tSym[2])
+   c(recob.perc, recob.BCa, recob.t, recob.tSym)
+ }
+ )
> rownames(res) = c("percentile", "percentile BCa", "bootstrap-t", "symmetrized bootstrap-t'
> rowMeans(res)
```

```
             percentile          percentile BCa             bootstrap-t
                  0.905                   0.938                   0.960
symmetrized bootstrap-t
                  0.957
```

9 Confidence interval for the biodiversity difference

Confidence interval for the biodiversity difference. Difference of 2 sample Shannon diversities with its standard error. It is reasonable to assume independence between periods and then compute the variance of the difference as the sum of variances.

```
> dShannon <- function(freq.abs1, freq.abs2, compute.se = FALSE) {
+   shan1 <- shannon(freq.abs1, compute.se)
+   shan2 <- shannon(freq.abs2, compute.se)
+   diff <- shan1 - shan2
+   if (compute.se)
+     attr(diff, "se") <- sqrt(attr(shan1,"se")^2 + attr(shan2,"se")^2)
+   return(diff)
+ }
> dShannon(dinos[,"upper"], dinos[,"middle"])

[1] -0.1497945

> dShannon(dinos[,"upper"], dinos[,"middle"], compute.se = TRUE)

[1] -0.1497945
attr(,"se")
[1] 0.174936
```

9.1 Confidence interval for the biodiversity difference: Bootstrap -p. Step by Step and own R function

```
> freq.abs1 = dinos[,"upper"]
> freq.abs2 = dinos[,"middle"]
> n1 <- sum(freq.abs1)
> freq.rels1 <- freq.abs1 / n1
> n2 <- sum(freq.abs2)
> freq.rels2 <- freq.abs2 / n2
> # Percentile interval:
> nboot = 10000
> alpha = 0.05
> set.seed(12345)
> diffBoots <- replicate(nboot,
+   dShannon(
+     rmultinom(1, size = n1, prob = freq.rels1),
+     rmultinom(1, size = n2, prob = freq.rels2)
+   )
+ )
> ci = quantile(diffBoots, probs = c(alpha/2, 1 - alpha/2))
> names(ci) = NULL
> attr(ci, "conf.level") = 1 - alpha
> ci

[1] -0.5238042  0.1861986
attr(,"conf.level")
[1] 0.95

> dShannonPerc = function(freq.abs1, freq.abs2, alpha = 0.05, nboot = 10000) {
+   n1 <- sum(freq.abs1)
+   freq.rels1 <- freq.abs1 / n1
+   n2 <- sum(freq.abs2)
+   freq.rels2 <- freq.abs2 / n2
+   diffBoots <- replicate(nboot,
+     dShannon(
+       rmultinom(1, size = n1, prob = freq.rels1),
+       rmultinom(1, size = n2, prob = freq.rels2)
+     )
+   )
+   ci = quantile(diffBoots, probs = c(alpha/2, 1 - alpha/2))
+   names(ci) = NULL
+   attr(ci, "conf.level") = 1 - alpha
+   return(ci)
+ }
> set.seed(12345)
> dShannonPerc(dinos[,"upper"], dinos[,"middle"])

[1] -0.5238042  0.1861986
attr(,"conf.level")
[1] 0.95
```

9.2 Confidence interval for the biodiversity difference: Bootstrap-t Symmetrized. Step by Step and own R function

```
> # Symmetrized interval:
>
> tdShannon <- function(freq.abs1, freq.abs2, diffParam) {
+   diff <- dShannon(freq.abs1, freq.abs2, compute.se = TRUE)
+   return((diff - diffParam) / attr(diff,"se"))
+ }
> diff.sample <- dShannon(freq.abs1, freq.abs2, compute.se = TRUE)
> set.seed(12345)
> tBoots <- replicate(nboot,
+   tdShannon(
+     rmultinom(1, size = n1, prob = freq.rels1),
+     rmultinom(1, size = n2, prob = freq.rels2),
+     diffParam = diff.sample
+   )
+ )
> tLimit <- quantile(abs(tBoots), probs = 1 - alpha)
> ci <- diff.sample - c(tLimit, -tLimit) * attr(diff.sample, "se")
> names(ci) = NULL
> attr(ci, "conf.level") = 1 - alpha
> ci

[1] -0.5287795  0.2291904
attr(,"conf.level")
[1] 0.95

> # Bootstrap-t confidence interval for the difference as a function:
> dShannonBoot.t <- function(freq.abs1, freq.abs2,
+   alpha = 0.05, symmetrized = FALSE, nboot = 10000)
+ {
+   diff.sample <- dShannon(freq.abs1, freq.abs2, compute.se = TRUE)
+   n1 <- sum(freq.abs1)
+   freq.rels1 <- freq.abs1 / n1
+   n2 <- sum(freq.abs2)
+   freq.rels2 <- freq.abs2 / n2
+   tBoots <- replicate(nboot,
+     tdShannon(
+       rmultinom(1, size = n1, prob = freq.rels1),
+       rmultinom(1, size = n2, prob = freq.rels2),
+       diffParam = diff.sample
+     )
+   )
+   if (symmetrized) {
+     tLimit <- quantile(abs(tBoots), probs = 1 - alpha)
+     ci <- diff.sample - c(tLimit, -tLimit) * attr(diff.sample, "se")
+   } else {
+     halfAlpha <- alpha / 2
+     tLimits <- quantile(tBoots, probs = c(1 - halfAlpha, halfAlpha))
+     ci <- diff.sample - tLimits * attr(diff.sample, "se")
+   }
+   names(ci) = NULL
+   attr(ci, "conf.level") = 1 - alpha
+   return(ci)
+ }
> set.seed(12345)
> dShannonBoot.t(dinos[,"upper"], dinos[,"middle"])

[1] -0.5029836  0.2599023
attr(,"conf.level")
[1] 0.95
```

```
> for (i in 1:(ncol(dinos)-1))
+   for (j in (i+1):ncol(dinos)) {
+     cat(names(dinos)[i], " - ", names(dinos)[j], "   \n")
+     print(dShannonBoot.t(dinos[,i], dinos[,j]))
+   }

upper  -  middle
[1] -0.5039581  0.2524403
attr(,"conf.level")
[1] 0.95
upper  -  lower
[1] -0.9343838  0.3400126
attr(,"conf.level")
[1] 0.95
middle  -  lower
[1] -0.7595068  0.4314300
attr(,"conf.level")
[1] 0.95

> dShannonBoot.t(dinos[,"upper"], dinos[,"middle"], symmetrized = TRUE)

[1] -0.5284155  0.2288265
attr(,"conf.level")
[1] 0.95

> for (i in 1:(ncol(dinos)-1))
+   for (j in (i+1):ncol(dinos)) {
+     cat(names(dinos)[i], " - ", names(dinos)[j], "   \n")
+     print(dShannonBoot.t(dinos[,i], dinos[,j], symmetrized = TRUE))
+   }

upper  -  middle
[1] -0.5284619  0.2288728
attr(,"conf.level")
[1] 0.95
upper  -  lower
[1] -0.8561405  0.4522413
attr(,"conf.level")
[1] 0.95
middle  -  lower
[1] -0.6778801  0.5735699
attr(,"conf.level")
[1] 0.95

> for (i in 1:(ncol(dinos)-1))
+   for (j in (i+1):ncol(dinos)) {
+     cat(names(dinos)[i], " - ", names(dinos)[j], "   \n")
+     print(dShannonBoot.t(dinos[,i], dinos[,j], symmetrized = TRUE, alpha = 0.1))
+   }
```

```
upper  -  middle
[1] -0.4592664  0.1596773
attr(,"conf.level")
[1] 0.9
upper  -  lower
[1] -0.7433670  0.3394678
attr(,"conf.level")
[1] 0.9
middle  -  lower
[1] -0.5648157  0.4605055
attr(,"conf.level")
[1] 0.9
```

10 Testing biodiversity differences(p-value). First approach

```
> # Testing, p-vaule...
> t.obs = tdShannon(freq.abs1, freq.abs2, diffParam = 0)
> sum(abs(tBoots) >= abs(t.obs)) / nboot

[1] 0.4288

> # Or:
> (sum(abs(tBoots) >= abs(t.obs)) + 1) / (nboot + 1)

[1] 0.4288571

>
> # These intervals and p-values should need an adjustement for multiplicity,
> # but this is another history...
```