MESS-ing with joint Hill numbers

Andrew Rominger

4/22/2019

We're wondering if we can use joint Hill numbers to capture shared information found across multiple axes, such as abundances and π values. To explore this I use Chao's definition of a Hill number for trait differences and apply it to differences in (abundance, π) space. Hill numbers are governed by a parameter q to changes the contribution of rare versus common species (small q mean rare species have more weight). Chao's functional trait Hill number is

$$qH = \left(\sum_i^S \sum_j^S d_{ij} \left(\frac{p_i p_j}{Q} \right)^q \right)^{1/(1-q)}$$

where d_{ij} is the matrix of distances between trait values of specie i and species j; p_k is the relative abundance of species k (i.e. n_k/J) and Q is a normalization term defined as $\sum_i^S \sum_j^S d_{ij} p_i p_j$. This Hill number can be put in units of "effective species" like this:

$$qD = \sqrt{\frac{qF}{Q}}.$$

Now the trick is that instead of defining d_{ij} in terms of traits, we can simply calculate distances between the abundances and π values of different species, i.e. let $d_{ij} = \sqrt{(n_i - n_j)^2 + (\pi_i - \pi_j)^2}$. Call Hill numbers from this type of distance "joint Hill numbers."

Below I code up the Hill functions

```r
library(mvtnorm)
```

```
## Warning: package 'mvtnorm' was built under R version 3.4.3
```

```r
library(pika)
library(parallel)

# function to return the Hill number for a given q
qH <- function(v, a, q, Q) {
    # browser()
    if(q == 1) {
        return(exp(- sum(v * a / Q * log(a / Q))))
    } else {
        return(sum(v * (a / Q)^q)^(1 / (1 - q)))
    }
}

# function to return the generalized Hill numbers (i.e. effective spp) for given q's
qD <- function(v, a, q) {
    Q <- sum(v * a)

    hh <- sapply(q, function(qi) {
        qH(v, a, qi, Q)
    })

    return(sqrt(hh / Q))
}
```

And now run an experiment calculating the joint Hill numbers for two arbitrary attributes. I simulate scenarios where these attributes have a 0.75 covariance (look for `xcor` in the code below) and where they have 0 covariance (look for `xnoc`).

```r
# explore distribution of generalized Hill numbers for correlated and uncorrelated values
diffCors <- mclapply(1:200, mc.cores = 10, FUN = function(i) {
    S <- 1000
    xcor <- rmvnorm(S, rep(0, 2), matrix(c(1, 0.75, 0.75, 1), nrow = 2))
    xnoc <- rmvnorm(S, rep(0, 2), matrix(c(1, 0, 0, 1), nrow = 2))

    n <- rfish(S, 0.05)
    pij <- outer(n, n, '*') / (sum(n)^2)

    dijCor <- as.matrix(dist(xcor))
    dijNoc <- as.matrix(dist(xnoc))

    o <- c(qD(dijCor, pij, 1:4), qD(dijNoc, pij, 1:4))
    names(o) <- paste(rep(c('cor', 'noc'), each = 4), 1:4, sep = '_')

    return(o)
})

diffCors <- do.call(rbind, diffCors)
```

Now we look at generalized joint Hill numbers (across multiple *q*) for correlated and uncorrelated attributes. You can see that, sadly, the Hill numbers completely overlap for both the correlated and uncorrelated cases.

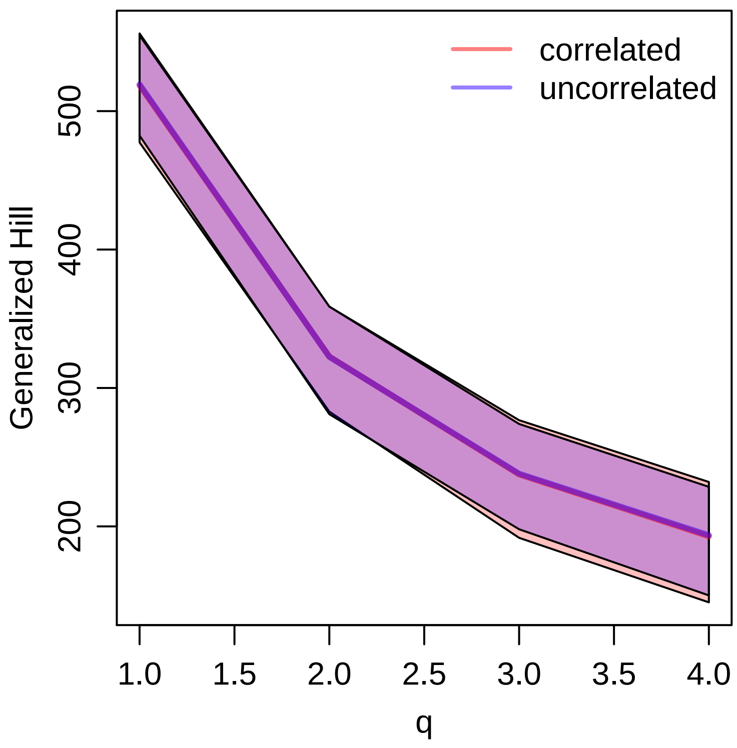
```r
diffSum <- apply(diffCors, 2, quantile, probs = c(0.025, 0.5, 0.975))

par(mar = c(3, 3, 0, 0) + 0.5, mgp = c(2, 0.75, 0))

plot(diffSum[2, 1:4], ylim = range(diffSum), type = 'l', lwd = 3,
     col = hsv(0, alpha = 0.5),
     panel.first = {
         polygon(c(1:4, 4:1), c(diffSum[1, 1:4], diffSum[3, 4:1]),
                 col = hsv(0, alpha = 0.25))
         polygon(c(1:4, 4:1), c(diffSum[1, 5:8], diffSum[3, 8:5]),
                 col = hsv(0.7, alpha = 0.25))
     },
     xlab = 'q', ylab = 'Generalized Hill')

points(diffSum[2, 5:8], ylim = range(diffSum), type = 'l', lwd = 3,
       col = hsv(0.7, alpha = 0.5))

legend('topright', legend = c('correlated', 'uncorrelated'), lty = 1, lwd = 2,
       col = hsv(c(0, 0.7), alpha = 0.5), bty = 'n')
```



correlated
uncorrelated
Generalized Hill
200
300
400
500
1.0
1.5
2.0
2.5
3.0
3.5
4.0
q