beta.div.r

```
beta.div <- function(Y, method="hellinger", sqrt.D=FALSE, samp=TRUE,
nperm=999, save.D=FALSE, clock=FALSE)
#
# Compute estimates of total beta diversity as the total variance in Y,
# for 20 dissimilarity coefficients or analysis of raw data (not
recommended).
# LCBD indices are tested by permutation within columns of Y.
# This version includes direct calculation of the Jaccard, Sorensen and
Ochiai
# coefficients for presence-absence data.
#
# Arguments --
#
# Y : community composition data matrix.
# method : name of one of the 20 dissimilarity coefficients, or "none"
for
#          direct calculation on Y (also the case with
method="euclidean").
# sqrt.D : If sqrt.D=TRUE, the distances in matrix D are square-rooted
before
#          computation of SStotal, BDtotal and LCBD.
# samp : If samp=TRUE, the abundance-based distances (ab.jaccard,
ab.sorensen,
#        ab.ochiai, ab.simpson) are computed for sample data. If
samp=FALSE,
#        they are computed for true population data.
# nperm : Number of permutations for test of LCBD.
# save.D : If save.D=TRUE, the distance matrix will appear in the
output list.
# clock : If clock=TRUE, the computation time is printed in the R
console.
#
# License: GPL-2
# Author:: Pierre Legendre, December 2012, April-May 2013
{
### Internal functions
centre <- function(D,n)
    # Centre a square matrix D by matrix algebra
    # mat.cen = (I - 11'/n) D (I - 11'/n)
    {    One <- matrix(1,n,n)
        mat <- diag(n) - One/n
        mat.cen <- mat %*% D %*% mat
    }
###
BD.group1 <- function(Y, method, save.D, per)
    {
    if(method=="profiles") Y = decostand(Y, "total")
    if(method=="hellinger") Y = decostand(Y, "hellinger")
    if(method=="chord") Y = decostand(Y, "norm")
```

```
    if(method=="chisquare") Y = decostand(Y, "chi.square")
    #
    s <- scale(Y, center=TRUE, scale=FALSE)^2   # eq. 1
    SStotal <- sum(s)           # eq. 2
    BDtotal <- SStotal/(n-1)    # eq. 3
    if(!per) { SCBD<-apply(s,2,sum)/SStotal }else{ SCBD<-NA }  # eqs.
4a and 4b
    LCBD <- apply(s, 1, sum)/SStotal  # eqs. 5a and 5b
    #
    D <- NA
    if(!per & save.D)   D <- dist(Y)
    #
    out <- list(SStotal_BDtotal=c(SStotal,BDtotal), SCBD=SCBD,
LCBD=LCBD,
    method=method, D=D)
    }
###
BD.group2 <- function(Y, method, sqrt.D)
    {
    if(method == "divergence") {
        D = D11(Y)

    } else if(any(method ==
      c("jaccard","sorensen","ochiai")))
        {
        if(method=="jaccard") D = dist.binary(Y, method=1) # ade4 takes
sqrt(D)
        if(method=="sorensen")  D = dist.binary(Y, method=5) #ade4
takes sqrt(D)
        if(method=="ochiai") D = dist.binary(Y, method=7) # ade4 takes
sqrt(D)

    } else if(any(method ==
c("manhattan","canberra","whittaker","percentagedifference","wishart"))
)
        {
        if(method=="manhattan") D = vegdist(Y, "manhattan")
        if(method=="canberra")  D = vegdist(Y, "canberra")
        if(method=="whittaker") D =
vegdist(decostand(Y,"total"),"manhattan")/2
        if(method=="percentagedifference") D = vegdist(Y, "bray")
        if(method=="wishart")   D = WishartD(Y)
        } else {
        if(method=="modmeanchardiff") D = D19(Y)
        if(method=="kulczynski")  D = vegdist(Y, "kulczynski")
        if(method=="ab.jaccard")  D = chao(Y, coeff="Jaccard",
samp=samp)
        if(method=="ab.sorensen") D = chao(Y, coeff="Sorensen",
samp=samp)
```

```
        if(method=="ab.ochiai")   D = chao(Y, coeff="Ochiai",
samp=samp)
        if(method=="ab.simpson")  D = chao(Y, coeff="Simpson",
samp=samp)
        }
    #
    if(sqrt.D) D = sqrt(D)
    SStotal <- sum(D^2)/n      # eq. 8
    BDtotal <- SStotal/(n-1)   # eq. 3
    delta1 <- centre(as.matrix(-0.5*D^2), n)   # eq. 9
    LCBD <- diag(delta1)/SStotal               # eq. 10b
    #
    out <- list(SStotal_BDtotal=c(SStotal,BDtotal), LCBD=LCBD,
    method=method, D=D)
    }
###
###
method <- match.arg(method, c("euclidean", "manhattan",
"modmeanchardiff", "profiles", "hellinger", "chord", "chisquare",
"divergence", "canberra", "whittaker", "percentagedifference",
"wishart", "kulczynski", "ab.jaccard",
"ab.sorensen","ab.ochiai","ab.simpson","jaccard","sorensen","ochiai","n
one"))
#
if(any(method == c("profiles", "hellinger", "chord", "chisquare",
"manhattan", "modmeanchardiff", "divergence", "canberra", "whittaker",
"percentagedifference", "kulczynski"))) require(vegan)
if(any(method == c("jaccard","sorensen","ochiai"))) require(ade4)
#
if(is.table(Y)) Y <- Y[1:nrow(Y),1:ncol(Y)]    # In case class(Y) is
"table"
n <- nrow(Y)
#
aa <- system.time({
if(any(method ==
c("euclidean", "profiles", "hellinger", "chord", "chisquare","none")))
{
    note <- "Info -- This coefficient is Euclidean"
    res <- BD.group1(Y, method, save.D, per=FALSE)
    #
    # Permutation test for LCBD indices, distances group 1
    if(nperm>0) {
        p <- ncol(Y)
        nGE.L = rep(1,n)
        for(iperm in 1:nperm) {
            Y.perm = apply(Y,2,sample)
            res.p <- BD.group1(Y.perm, method, save.D, per=TRUE)
            ge <- which(res.p$LCBD >= res$LCBD)
            nGE.L[ge] <- nGE.L[ge] + 1
            }
        p.LCBD <- nGE.L/(nperm+1)
```

```
        } else { p.LCBD <- NA }
    #
    if(save.D) { D <- res$D } else { D <- NA }
    #
    out <- list(SStotal_BDtotal=res$SStotal_BDtotal, SCBD=res$SCBD,
    LCBD=res$LCBD, p.LCBD=p.LCBD, method=method, note=note, D=D)

} else {
#
    if(method == "divergence") {
        note = "Info -- This coefficient is Euclidean"
    } else if(any(method == c("jaccard","sorensen","ochiai"))) {
        note = c("Info -- This coefficient is Euclidean because
dist.binary ",
        "of ade4 computes it as sqrt(D). Use beta.div with option
sqrt.D=FALSE")
    } else if(any(method ==
c("manhattan","canberra","whittaker","percentagedifference","wishart"))
) {
        if(sqrt.D) {
        note = "Info -- This coefficient, in the form sqrt(D), is
Euclidean"
        } else {
        note = c("Info -- For this coefficient, sqrt(D) would be
Euclidean",
        "Use is.euclid(D) of ade4 to check Euclideanarity of this D
matrix")
        }
    } else {
        note = c("Info -- This coefficient is not Euclidean",
        "Use is.euclid(D) of ade4 to check Euclideanarity of this D
matrix")
    }
#
    res <- BD.group2(Y, method, sqrt.D)
    #
    # Permutation test for LCBD indices, distances group 2
    if(nperm>0) {
        nGE.L = rep(1,n)
        for(iperm in 1:nperm) {
            Y.perm = apply(Y,2,sample)
            res.p <- BD.group2(Y.perm, method, sqrt.D)
            ge <- which(res.p$LCBD >= res$LCBD)
            nGE.L[ge] <- nGE.L[ge] + 1
            }
        p.LCBD <- nGE.L/(nperm+1)
        } else { p.LCBD <- NA }
#
    if(sqrt.D) note.sqrt.D<-"sqrt.D=TRUE"  else  note.sqrt.D<-
```

```
"sqrt.D=FALSE"
    if(save.D) { D <- res$D } else { D <- NA }
    #
    out <- list(SStotal_BDtotal=res$SStotal_BDtotal, LCBD=res$LCBD,
    p.LCBD=p.LCBD, method=c(method,note.sqrt.D), note=note, D=D)
}
#
})
aa[3] <- sprintf("%2f",aa[3])
if(clock) cat("Time for computation =",aa[3]," sec\n")
#
class(out) <- "beta.div"
out
}

D11 <- function(Y, algo=1)
#
# Compute Clark's coefficient of divergence.
# Coefficient D11 in Legendre and Legendre (2012, eq. 7.51).
#
# License: GPL-2
# Author:: Pierre Legendre, April 2011
{
Y <- as.matrix(Y)
n <- nrow(Y)
p <- ncol(Y)
# Prepare to divide by pp = (p-d) = no. species present at both sites
Y.ap <- 1 - decostand(Y, "pa")
d <- Y.ap %*% t(Y.ap)
pp <- p-d   # n. species present at the two compared sites
#
if(algo==1) {   # Faster algorithm
    D <- matrix(0, n, n)
    for(i in 2:n) {
        for(j in 1:(i-1)) {
            num <- (Y[i,]-Y[j,])
            den <- (Y[i,]+Y[j,])
            sel <- which(den > 0)
            D[i,j] = sqrt(sum((num[sel]/den[sel])^2)/pp[i,j])
            }
        }
#
} else {   # Slower algorithm
    D <- matrix(0, n, n)
    for(i in 2:n) {
        for(j in 1:(i-1)) {
            temp = 0
            for(p2 in 1:p) {
                den = Y[i,p2] + Y[j,p2]
                if(den > 0) {
                    temp = temp + ((Y[i,p2] - Y[j,p2])/den)^2
```

```
                    }
                }
            D[i,j] = sqrt(temp/pp[i,j])
            }
        }
#
    }
DD <- as.dist(D)
}

D19 <- function(Y)
#
# Compute the Modified mean character difference.
# Coefficient D19 in Legendre and Legendre (2012, eq. 7.46).
# Division is by pp = number of species present at the two compared
sites
#
# License: GPL-2
# Author:: Pierre Legendre, April 2011
{
Y <- as.matrix(Y)
n <- nrow(Y)
p <- ncol(Y)
# Prepare to divide by pp = (p-d) = n. species present at both sites
Y.ap <- 1 - decostand(Y, "pa")
d <- Y.ap %*% t(Y.ap)
pp <- p-d   # n. species present at the two compared sites
#
D <- vegdist(Y, "manhattan")
DD <- as.dist(as.matrix(D)/pp)
}

WishartD <- function(Y)
#
# Compute dissimilarity - 1 - Wishart similarity ratio (Wishart 1969).
#
# License: GPL-2
# Author:: Pierre Legendre, August 2012
{
CP = crossprod(t(Y))
SS = apply(Y^2,1,sum)
n = nrow(Y)
mat.sq = matrix(0, n, n)
for(i in 2:n) {
    for(j in 1:(n-1)) { mat.sq[i,j] = CP[i,j]/(SS[i] + SS[j] - CP[i,j])
}
    }
mat = 1 - as.dist(mat.sq)
}
```

```
chao <- function(mat, coeff="Jaccard", samp=TRUE)
#
# Compute Chao et al. (2006) abundance-based indices.
#
# Arguments -
# mat = data matrix, species abundances
# coef = "Jaccard" : modified abundance-based Jaccard index
#        "Sorensen": modified abundance-based SÃ¸rensen index
#        "Ochiai"  : modified abundance-based Ochiai index
#        "Simpson" : modified abundance-based Simpson index
# samp=TRUE : Compute dissimilarities for sample data
#     =FALSE: Compute dissimilarities for true population data
#
# Details -
# For coeff="Jaccard", the output values are identical to those
# produced by vegan's function vegdist(mat, "chao").
#
# Help received from A. Chao and T. C. Hsieh in July 2012 for the
computation
# of dissimilarities for true population data is gratefully
acknowledged.
#
# Reference --
# Chao, A., R. L. Chazdon, R. K. Colwell and T. J. Shen. 2006.
# Abundance-based similarity indices and their estimation when there
# are unseen species in samples. Biometrics 62: 361â€“371.
#
# License: GPL-2
# Author:: Pierre Legendre, July 2012
{
require(vegan)
nn = nrow(mat)
res = matrix(0,nn,nn)
if(samp) {   # First for sample data
for(k in 2:nn) {
    for(j in 1:(k-1)) {
        #cat("k =",k,"  j =",j,"\n")
        v1 = mat[j,]   # Vector 1
        v2 = mat[k,]   # Vector 2
        v1.pa = decostand(v1,"pa")   # Vector 1 in presence-absence
form
        v2.pa = decostand(v2,"pa")   # Vector 2 in presence-absence
form
        N.j = sum(v1)   # Sum of abundances in vector 1
        N.k = sum(v2)   # Sum of abundances in vector 2
        shared.sp = v1.pa * v2.pa   # Vector of shared species ("pa")
        if(sum(shared.sp) == 0) {
            res[k,j] = 1
        } else {
        C.j = sum(shared.sp * v1)   # Sum of shared sp. abundances in
```

```
v1
        C.k = sum(shared.sp * v2)   # Sum of shared sp. abundances in
v2
        # a1.j = sum(shared.sp * v1.pa)
        # a1.k = sum(shared.sp * v2.pa)
        a1.j = length(which((shared.sp * v2) == 1)) # Singletons in v2
        a1.k = length(which((shared.sp * v1) == 1)) # Singletons in v1
        a2.j = length(which((shared.sp * v2) == 2)) # Doubletons in v2
        if(a2.j == 0) a2.j <- 1
        a2.k = length(which((shared.sp * v1) == 2)) # Doubletons in v1
        if(a2.k == 0) a2.k <- 1
        # S.j = sum(v1[which(v2 == 1)]) # Sum abund. in v1 for
singletons in v2
        # S.k = sum(v2[which(v1 == 1)]) # Sum abund. in v2 for
singletons in v1
        sel2 = which(v2 == 1)
        sel1 = which(v1 == 1)
        if(length(sel2)>0) S.j = sum(v1[sel2]) else S.j = 0
        if(length(sel1)>0) S.k = sum(v2[sel1]) else S.k = 0

        U.j = (C.j/N.j) + ((N.k-1)/N.k) * (a1.j/(2*a2.j)) * (S.j/N.j) #
Eq. 11
        if(U.j > 1) U.j <- 1
        U.k = (C.k/N.k) + ((N.j-1)/N.j) * (a1.k/(2*a2.k)) * (S.k/N.k) #
Eq. 12
        if(U.k > 1) U.k <- 1

        if(coeff == "Jaccard") {                    # "Jaccard"
            res[k,j] = 1 - (U.j*U.k/(U.j + U.k - U.j*U.k))
            } else if(coeff == "Sorensen") {        # "Sorensen"
            res[k,j] = 1 - (2*U.j*U.k/(U.j + U.k))
            } else if(coeff == "Ochiai") {          # "Ochiai"
            res[k,j] = 1 - (sqrt(U.j*U.k))
            } else if(coeff == "Simpson") {
            # Simpson (1943), or Lennon et al. (2001) in Chao et al.
(2006)
            res[k,j] = 1 -
            (U.j*U.k/(U.j*U.k+min((U.j-U.j*U.k),(U.k-U.j*U.k))))
            } else { #
            stop("Incorrect coefficient name")
            }
        }
        }
    }

} else {   # Now for complete population data

for(k in 2:nn) {
    for(j in 1:(k-1)) {
```

```
        v1 = mat[j,]   # Vector 1
        v2 = mat[k,]   # Vector 2
        v1.pa = decostand(v1,"pa")   # Vector 1 in presence-absence
form
        v2.pa = decostand(v2,"pa")   # Vector 2 in presence-absence
form
        shared.sp = v1.pa * v2.pa    # Vector of shared species ("pa")
        if(sum(shared.sp) == 0) {
            res[k,j] = 1
        } else {
        N1 = sum(v1)   # Sum of abundances in vector 1
        N2 = sum(v2)   # Sum of abundances in vector 2
        U = sum(shared.sp * v1)/N1   # Sum of shared sp. abundances in
v1
        V = sum(shared.sp * v2)/N2   # Sum of shared sp. abundances in
v2

        if(coeff == "Jaccard") {                    # "Jaccard"
            res[k,j] = 1 - (U*V/(U + V - U*V))
            } else if(coeff == "Sorensen") {        # "Sorensen"
            res[k,j] = 1 - (2*U*V/(U + V))
            } else if(coeff == "Ochiai") {          # "Ochiai"
            res[k,j] = 1 - (sqrt(U*V))
            } else if(coeff == "Simpson") { # "Simpson"
            res[k,j] = 1 - (U*V/(U*V+min((U-U*V),(V-U*V)))) # Eq. ?
            } else { #
            stop("Incorrect coefficient name")
            }
        }
        }
    }
}
res <- as.dist(res)
}

######## End of beta.div function
```