```
distorefpool <- function(comm, dis, fundis, ...){

    if (!(inherits(comm, "data.frame") | inherits(comm, "matrix")))
        stop("comm must be a data.frame or a matrix")
    if(nrow(comm) < 2) stop("comm must have at least two rows")
    if (!(inherits(dis, "dist") | inherits(dis, "matrix")))
        stop("dis must be an object of class dist or a matrix")
    dis <- as.dist(dis)
    if (!inherits(fundis, "character"))
        stop("fundis must be a character string")
    if (!exists(fundis))
        stop("the function specified in argument fundis does not exist")
    thefundis <- match.fun(fundis)
    ncom <- nrow(comm)
    funi <- function(i){
        commrel <- sweep(comm, 1, rowSums(comm), "/")
        Veta <- colSums(commrel[-i, ])/(ncom-1)
        comi <- cbind.data.frame(unlist(comm[i, ]/sum(comm[i, ])), Veta)
        comi <- as.data.frame(t(comi))
        colnames(comi) <- colnames(comm)
        rownames(comi) <- c("k","eta")
        return(as.vector(thefundis(comi, dis, ...)))
    }
    RES <- sapply(1:ncom, funi)
    names(RES) <- rownames(comm)
    return(RES)

}

rtestsampledis <- function(comm, dis, fac, fundis, option = 1:2, nrep =
999, ...){

    if (!(inherits(comm, "data.frame") | inherits(comm, "matrix")))
        stop("comm must be a data.frame or a matrix")
    if (!(inherits(dis, "dist") | inherits(dis, "matrix")))
        stop("dis must be an object of class dist or a matrix")
    dis <- as.dist(dis)
    if(ncol(comm)!=attributes(dis)$Size) stop("Species in comm must be
similar and in the same order as species in dis")
    if(!inherits(fac, "factor") | length(fac)!=nrow(comm))
stop("Incorrect definition of argument fac")
    ngroups <- length(levels(fac))
    if(min(table(fac))<2) stop("Each group in fac must contain at least
two communities")
    ncom <- nrow(comm)
    if (!inherits(fundis, "character"))
        stop("fundis must be a character string")
    if (!exists(fundis))
        stop("the function specified in argument fundis does not exist")
    thefundis <- match.fun(fundis)
    if(!option[1]%in%(1:2)) stop("Incorrect definition of argument
option")
    if(option[1] == 1){
        Dhk <- as.matrix(thefundis(comm, dis, ...))
        LDhk <- list()
        for(i in 1:ngroups) {
           LDhk[[i]] <- Dhk[fac==levels(fac)[i], fac==levels(fac)[i]]
        }
```

```
        LDk <- lapply(LDhk, function(x) sapply(1:nrow(x), function(j)
mean(as.matrix(x)[j, -j])))
    }
    else {
        LDk <- list()
        for(i in 1:ngroups) {
            LDk[[i]] <- distorefpool(comm[fac==levels(fac)[i], ], dis,
fundis, ...)
        }
    }
    EffRES <- unlist(lapply(LDk, length)) - 1
    RES <- sum(unlist(lapply(LDk, var)) * EffRES) / (ncom - ngroups)
    EffMOD <- unlist(lapply(LDk, length))
    MOD <- sum(EffMOD*(unlist(lapply(LDk, mean))-mean(unlist(LDk)))^2)
/(ngroups - 1)
    OBS <- MOD / RES
    simul <- function(i) {

        LL <- split(sample(unlist(LDk)), fac)
        EffRES <- unlist(lapply(LL, length)) - 1
        RES <- sum(unlist(lapply(LL, var)) * EffRES) / (ncom - ngroups)
        EffMOD <- unlist(lapply(LL, length))
        MOD <- sum(EffMOD*(unlist(lapply(LL, mean))-mean(unlist(LL)))^2)
/(ngroups - 1)
        return(MOD / RES)

    }
    SIM <- sapply(1:nrep, simul)
    RES <- as.randtest(obs = OBS, sim = SIM, alter = "greater", output =
"full")
    RES$call <- match.call()
    class(RES) <- c(class(RES), "rtestsampledis")
    return(RES)

}

posthoc <- function(Xtest, p.adjust.method = "none", output = c("light",
"full")){

    if(!inherits(Xtest, "rtestsampledis")) stop("X must be of class
rtestsampledis")
    objectsX <- as.list(Xtest$call)
    Comm <- eval.parent(objectsX$comm)
    Dis <- eval.parent(objectsX$dis)
    Fac <- eval.parent(objectsX$fac)
    Fundis <- eval.parent(objectsX$fundis)
    Option <- eval.parent(objectsX$option)
    Nrep <- eval.parent(objectsX$nrep)
    FFundis <- match.fun(Fundis)
    if(length(objectsX)>7) {
        FF <- formals(FFundis)
        objectsXdots <- objectsX[-(1:7)]
        for(i in 1:length(objectsXdots )){
        FF[names(objectsXdots)[i]] <- objectsXdots[[i]]
        }
        formals(FFundis) <- FF
    }
    levFac <- levels(Fac)
```

```
    setFac <- combn(levFac, 2)

    Distorefpool <- function(comm, dis) {
        ncom <- nrow(comm)
        funi <- function(i){
        Veta <- colSums(comm[-i, ])/(ncom-1)
        comi <- cbind.data.frame(unlist(comm[i, ]), Veta)
        comi <- as.data.frame(t(comi))
        colnames(comi) <- colnames(comm)
        rownames(comi) <- c("k","eta")
        return(as.vector(FFundis(comi, dis)))
        }
        RES <- sapply(1:ncom, funi)
        names(RES) <- rownames(comm)
        return(RES)
    }

    Rtestsampledis <- function(comm, dis, fac, option = 1:2, nrep = 999,
...) {

        dis <- as.dist(dis)
        ngroups <- length(levels(fac))
        ncom <- nrow(comm)

        if(option[1] == 1){
        Dhk <- as.matrix(FFundis(comm, dis))
        LDhk <- list()
        for(i in 1:ngroups) {
           LDhk[[i]] <- Dhk[fac==levels(fac)[i], fac==levels(fac)[i]]
        }
        LDk <- lapply(LDhk, function(x) sapply(1:nrow(x), function(j)
mean(as.matrix(x)[j, -j])))
        }
        else {
        LDk <- list()
        for(i in 1:ngroups) {
            LDk[[i]] <- Distorefpool(comm[fac==levels(fac)[i], ], dis)
        }
        }

        EffRES <- unlist(lapply(LDk, length)) - 1
        RES <- sum(unlist(lapply(LDk, var)) * EffRES) / (ncom - ngroups)
        EffMOD <- unlist(lapply(LDk, length))
        MOD <- sum(EffMOD*(unlist(lapply(LDk, mean))-
mean(unlist(LDk)))^2) /(ngroups - 1)
        OBS <- MOD / RES
        simul <- function(i) {

        LL <- split(sample(unlist(LDk)), fac)
        EffRES <- unlist(lapply(LL, length)) - 1
        RES <- sum(unlist(lapply(LL, var)) * EffRES) / (ncom - ngroups)
        EffMOD <- unlist(lapply(LL, length))
        MOD <- sum(EffMOD*(unlist(lapply(LL, mean))-mean(unlist(LL)))^2)
/(ngroups - 1)
        return(MOD / RES)

        }
        SIM <- sapply(1:nrep, simul)
```

```
        return(list(obs=OBS, sim=SIM))

    }

    FUNN <- function(Vi) {
        FAC <- factor(as.character(Fac)[as.character(Fac)%in%Vi])
        COMM <- Comm[as.character(Fac)%in%Vi, ]
        res <- Rtestsampledis(COMM, Dis, FAC, option = Option, nrep =
Nrep)
        return(res)
    }
    RESlist <- apply(setFac, 2, FUNN)
    OBS <- unlist(lapply(RESlist, function(x) x$obs))
    SIM <- cbind.data.frame(lapply(RESlist, function(x) x$sim))
    colnames(SIM) <- apply(setFac, 2, function(x) paste(x, collapse="-"))
    RES <- as.krandtest(sim = SIM, obs = OBS,
        alter = "greater",
        p.adjust.method = p.adjust.method, output = output)
    RES$call <- match.call()
    return(RES)

}
```