```
# Appendix 3   R function for the calculation of the functional imbalance
measures

# Arguments of function FunImbalance:
# comm = matrix or data frame with communities as rows,
# species as columns and abundance data as entries;
# dis = an object of class dist that provides the
# functional dissimilarities between species;
# method = a character string: one or a vector of
# "CorB", "SESB", and "QB". The strings indicate the name
# of the index of functional imbalance that must be used;
# nrep = the number of repetitions (permutations) to use
# in the calculation of SESB and QB;
# tol = a tolerance number: a value in [-tol, tol] is
# considered as zero.

FunImbalance <-
function (comm, dis, method = c("CorB", "SESB", "QB"), nrep = 10000, tol
= 1e-16)
{
    if (!inherits(comm, "data.frame") & !inherits(comm, "matrix"))
        stop("Non convenient comm")
    comm <- t(comm)
    if (any(comm < 0))
        stop("Negative value in comm")
    if(any(!method%in%c("CorB", "SESB", "QB"))) stop("method can be CorB,
SESB or QB")
    method <- unique(method)
    if(is.null(dis))  stop("Object dis is undefined")
    if (!inherits(dis, "dist"))
    stop("Object of class 'dist' expected for argument dis")
    D <- as.matrix(dis)
    if (nrow(comm) != nrow(D))
        stop("Non convenient comm")
    FUNIMB <- as.data.frame(matrix(0, ncol(comm), length(method)))
    names(FUNIMB) <- method
    rownames(FUNIMB) <- colnames(comm)

    corB <- function(VEC) {

        if (sum(VEC) < tol){
            warning("empty communities")
            return(NA)
        }
        else{
            Q <- (t(VEC) %*% D %*% VEC)/(sum(VEC)^2)
            Simpson <- 1-sum((VEC/sum(VEC))^2)
            meanD <- function(x) mean(as.dist(D[x>tol, x>tol]))
            mD <- meanD(VEC)
            B <- Q - Simpson*mD
            S <- length(VEC[VEC > 0])
            C <- (VEC%*%t(VEC))/(sum(VEC))^2
            varW <- (2/(S*(S-1)))*sum(as.dist(C[VEC>0, VEC>0]-
Simpson/(S*(S-1)))^2)
            varD <- (2/(S*(S-1)))*sum(as.dist(D[VEC>0, VEC>0]-mD)^2)
            CORB <- B / ((S*(S-1))*sqrt(varW* varD))
        }
    }
```

```
    quadiv <- function(VECT) {
        quad <- (t(VECT) %*% D %*% VECT)/(sum(VECT)^2)
        return(quad)
    }

    simquadiv <- function(VECT) {
        simVECT <- sample(VECT)
        simQ <- (t(simVECT) %*% D %*% simVECT)/(sum(simVECT)^2)
        return(simQ)
    }

    simQ <- function(VEC) {
        if (sum(VEC) < tol){
            warning("empty communities")
            return(rep(NA, nrep))
        }
        return(sapply(1:nrep, function(zz) simquadiv(VEC)))
    }

    if (any(method %in% c("SESB", "QB"))){
        obsQ <- apply(comm, 2, quadiv)
        tabsimQ <- apply(comm, 2, simQ)
    }


    for(m in method){
        if(m == "CorB") FUNIMB[, "CorB"] <- apply(comm, 2, corB)
        if(m == "SESB") FUNIMB[, "SESB"]  <- (obsQ - apply(tabsimQ, 2,
mean)) / apply(tabsimQ, 2, sd)
        if(m == "QB") FUNIMB[, "QB"]  <- (obsQ - apply(tabsimQ, 2, min))
/ (apply(tabsimQ, 2, max) - apply(tabsimQ, 2, min))
    }

    return(FUNIMB)
}
```