Supplementary Text 6: R code

```
abond.expo <- function(x,a=NULL,plot=TRUE,interact=FALSE)
{
# Computes the number of trees in each diameter class according to the
# exponential model for tree size distribution
# INPUT
# x: matrix with n rows and p columns corresponding to n forest plots where
#    each row gives the observed number of trees in p diameter classes
# a: vector giving the lower bounds of the p diameter classes
# OUTPUT
# Matrix with the same dimensions as x giving the predicted number of trees
# in the n plots and p diameter classes according to the exponential model

if(is.null(a)) a <- as.numeric(firstword(names(x),"\\.",2))
D0 <- a[1]
Di <- a+diff(a[1:2])/2
b <- c(a[-1],Inf)
m <- colMeans(x)
s <- apply(x,2,sd)
xs <- scale(x)
abond.cond <- function(N,mu) N*(exp(-mu*(a-D0))-exp(-mu*(b-D0)))
p <- matrix(nrow=nrow(x),ncol=ncol(x))
for(i in 1:nrow(x)) {
        cat(N <- sum(x[i,]),"-> ")
        mu <- 1/(sum(Di*x[i,])/sum(x[i,])-D0)
        sqd <- function(Nmu) sum((xs[i,]-(abond.cond(Nmu[1],Nmu[2])-m)/s)^2)
        Nmu <- optim(c(N,mu),sqd,method="L-BFGS-B",lower=c(0,0.0001))$par
        cat(Nmu[1],"\n"); flush.console()
        p[i,] <- abond.cond(Nmu[1],Nmu[2])
        if(plot) {
                points(c(barplot(unlist(x[i,]))),p[i,],pch=16,col="red",xpd=NA)
                if(interact) locator(n=1) }}
p <- as.data.frame(p)
row.names(p) <- row.names(x)
names(p) <- names(x)
p
}
abond.power <- function(x,a=NULL,plot=TRUE,interact=FALSE)
{
# Computes the number of trees in each diameter class according to the
# power model for tree size distribution
# INPUT
# x: matrix with n rows and p columns corresponding to n forest plots where
```

```
#    each row gives the observed number of trees in p diameter classes
# a: vector giving the lower bounds of the p diameter classes
# OUTPUT
# Matrix with the same dimensions as x giving the predicted number of trees
# in the n plots and p diameter classes according to the power model

if(is.null(a)) a <- as.numeric(firstword(names(x),"\\.",2))
D0 <- a[1]
Di <- a+diff(a[1:2])/2
b <- c(a[-1],Inf)
m <- colMeans(x)
s <- apply(x,2,sd)
xs <- scale(x)
abond.cond <- function(N,z) N*((a/D0)^(1-z)-(b/D0)^(1-z))
p <- matrix(nrow=nrow(x),ncol=ncol(x))
for(i in 1:nrow(x)) {
        cat(N <- sum(x[i,]),"->␣")
        z <- 2
        sqd <- function(Nz) sum((xs[i,]-(abond.cond(Nz[1],Nz[2])-m)/s)^2)
        Nz <- optim(c(N,z),sqd,method="L-BFGS-B",lower=c(0,1))$par
        cat(Nz[1],"\n"); flush.console()
        p[i,] <- abond.cond(Nz[1],Nz[2])
        if(plot) {
                points(c(barplot(unlist(x[i,]))),p[i,],pch=16,col="red",xpd=NA)
                if(interact) locator(n=1) }}
p <- as.data.frame(p)
row.names(p) <- row.names(x)
names(p) <- names(x)
p
}
correl.test <-
function(Ap,A,B=200,plot=TRUE)
{
# Correlation between two structural attributes of forest plots. A
# randomization test is performed on the difference between the observed
# and predicted attributes
# INPUT
# Ap: matrix with n rows and 2 columns giving the two predicted structural
#     attributes of the n plots
# A: matrix with n rows and 2 columns giving the two observed structural
#    attributes of the n plots
# B: number of randomizations for the randomization test

E <- A-Ap
obs <- cor(A[,1],A[,2])
```

```
mod <- cor(Ap[,1],Ap[,2])
rho <- cor(E[,1],E[,2])
rho.null <- rep(NA,B)
for(b in 1:B) {
        if((b%%100)==0) { cat("."); if((b%%7600)==0) cat("\n") }
        rho.null[b] <- cor(sample(E[,1]),sample(E[,2])) }
cat("\n")
pval <- sum(rho.null >= rho)/B
pval <- min(pval,1-pval)*2
cat("correlation␣between␣observed␣variables:",obs,"\n")
cat("correlation␣between␣modeled␣variables:",mod,"\n")
cat("correlation␣for␣residual␣matrix␣E:",rho,"\n")
cat("p-value:",pval,"\n")
if(plot) {
        hist(rho.null,xlim=c(-1,1))
        points(rho,0,pch=16,col="red",xpd=NA) }
invisible(c(obs=obs,mod=mod,res=rho))
}
expo <- function(x)
{
# Computes the structural attributes of different plots according to the
# exponential model for tree size distribution
# INPUT
# x: matrix with n rows and p colums corresponding to n forest plots where
#    each row gives the observed structural attributes for this plot. The
#    structural attributes considered here are those returned by the
#    function 'expo.cond'
# OUTPUT
# Matrix with the same dimensions as x giving the predicted structural
# attributes of the n plots

m <- colMeans(x)
s <- apply(x,2,sd)
xs <- scale(x)
NG <- matrix(NA,nrow(x),2)
for(i in 1:nrow(x)) {
        cat(x$N[i],x$G[i],"->␣")
        sqd <- function(NG) sum((xs[i,]-(expo.cond(NG[1],NG[2])-m)/s)^2)
        NG[i,] <- optim(c(x$N[i],x$G[i]),sqd,method="L-BFGS-B",lower=c(0,0))$par
        cat(NG[i,],"\n"); flush.console() }
p <- expo.cond(NG[,1],NG[,2])
row.names(p) <- row.names(x)
p
}
expo.cond <-
```

```
function(N=457,G=32,x0=10)
{
# Given stand density N and stand basal area G, computes the other structural
# attributes of the forest plot according to the exponential model for tree
# size distribution
# INPUT
# N: density of trees in the plot (per ha)
# G: plot basal area (in m2/ha)
# x0: minimum dbh for inventory (in cm)
# OUTPUT
# Vector with 10 structural attributes: 1) density of trees, 2) basal area,
# 3) mean diameter, 4) equivalent diameter, 5) density of trees in the diameter
# class [10,30[ cm, 6) density of trees in the diameter class [30,60[ cm,
# 7) density of trees in the diameter class >= 60 cm, 8) aboveground biomass,
# 9) proportion of biomass in the diameter class => 60 cm, 10) wood density

alpha <- G*1e4/(N*pi*x0^2/4) # G in m2 whereas x0 in cm
mu <- (sqrt(2*alpha-1)+1)/(alpha-1)/x0
Dm <- 1/mu+x0
De <- sqrt(4e4*G/(pi*N))
a <- c(10,30,60)
b <- c(a[-1],Inf)
Ni <- as.data.frame(N*(exp(-outer(mu,a-x0,"*"))-exp(-outer(mu,b-x0,"*"))))
names(Ni) <- paste("N",1:length(a),sep="")
rho <- 0.6
f <- function(x) rho*exp(-1.499+2.148*log(x)+0.207*(log(x)^2)-0.0281*(log(x)^3))
B <- p <- rep(NA,length(N))
for(i in seq_along(N)) {
        B[i] <- N[i]*mu[i]*integrate(function(x) f(x)*exp(-mu[i]*(x-x0)),x0,
                Inf)$value/1e3
        p[i] <- N[i]*mu[i]*integrate(function(x) f(x)*exp(-mu[i]*(x-x0)),60,
                Inf)$value/1e3/B[i] }
cbind(N=N,G=G,Dm=Dm,De=De,Ni,B=B,p=p,R=rho)
}
firstword <-
function(x=c("first␣second","premier␣second","primo␣secundo"),sep="␣",n=1)
{
# extracts the nth word from sentences
# INPUT
# x: vector of sentences
# sep: character separating words in sentences
# n: value of n
# OUTPUT
# Vector of the nth words of each sentence
```

```
unlist(lapply(strsplit(as.character(x),sep),"[",n))
}
pca.test <-
function(Ap,A,B=200,trans="I",plot=TRUE,...)
{
# Principal component analysis of the structural attributes of plots, whether
# these attributes are observed, predicted, or are the differences between
# observations and predictions. A randomization test is performed using the
# sum of the two first eigenvalues of the PCA as test statistic
# INPUT
# Ap: matrix with n rows and p columns corresponding to n forest plots where
#    each row gives the predicted structural attributes of each plot
# A: matrix with n rows and p columns corresponding to n forest plots where
#    each row gives the observed structural attributes of each plot
# B: number of randomizations for the randomization test
# trans: transformation of data prior to PCA (default 'I' means no transformation)
# ...: additional arguments for the 'dudi.pca' function of the ade4 package

library(ade4)
library(BiodiversityR)
trf <- function(X,meth) {
        if(meth=="I") return(X)
        disttransform(X,method=meth) }
trE <- trf(A-Ap,trans)
obs <- dudi.pca(trf(A,trans),nf=2,scannf=FALSE,...)
mod <- dudi.pca(trf(Ap,trans),nf=2,scannf=FALSE,...)
res <- dudi.pca(trE,nf=2,scannf=FALSE,...)
lbd.null <- rep(NA,B)
for(b in 1:B) {
        if((b%%100)==0) { cat("."); if((b%%7600)==0) cat("\n") }
        lbd.null[b] <- sum(dudi.pca(apply(trE,2,sample),nf=2,scannf=FALSE)$eig[
                1:2]) }
cat("\n")
lbd <- sum(res$eig[1:2])
pval <- sum(lbd.null >= lbd)/B
pval <- min(pval,1-pval)*2
cat("sum of first two eigenvalues of PCA of observed A:",sum(obs$eig[1:2]),"\n")
cat("sum of first two eigenvalues of PCA of modeled Ap:",sum(mod$eig[1:2]),"\n")
cat("sum of first two eigenvalues of PCA of residual E:",lbd,"\n")
cat("p-value:",pval,"\n")
if(plot) {
        hist(lbd.null,xlim=range(lbd,lbd.null))
        points(lbd,0,pch=16,col="red",xpd=NA) }
invisible(list(obs=obs,mod=mod,res=res))
}
```

```
power <- function(x,x0=10)
{
# Computes the structural attributes of different plots according to the
# power model for tree size distribution
# INPUT
# x: matrix with n rows and p colums corresponding to n forest plots where
#    each row gives the observed structural attributes for this plot. The
#    structural attributes considered here are those returned by the
#    function 'power.cond'
# x0: minimum dbh for inventory (in cm)
# OUTPUT
# Matrix with the same dimensions as x giving the predicted structural
# attributes of the n plots

m <- colMeans(x)
s <- apply(x,2,sd)
xs <- scale(x)
ND <- matrix(NA,nrow(x),2)
for(i in 1:nrow(x)) {
        cat(x$N[i],x$D[i],"-> ")
        sqd <- function(ND) sum((xs[i,]-(power.cond(ND[1],ND[2],x0)-m)/s)^2)
        o <- try(optim(c(x$N[i],x$D[i]),sqd,method="L-BFGS-B",lower=c(0,x0)),
                silent=TRUE)
        if(class(o)=="try-error") {
                cat("\n\tno fit obtained with L-BFGS-B, trying Nelder-Mead\n\t")
                sqdNM <- function(ND) if((ND[1]<=0)|(ND[2]<=x0)) return(Inf)
                        else return(sqd(ND))
                o <- optim(c(x$N[i],x$D[i]),sqdNM,method="Nelder-Mead",
                        hessian=FALSE) }
        ND[i,] <- o$par
        cat(ND[i,],"\n"); flush.console() }
p <- power.cond(ND[,1],ND[,2],x0)
row.names(p) <- row.names(x)
p
}
power.cond <-
function(N=457,Dm=22,x0=10,x1=200)
{
# Given stand density N and mean diameter Dm, computes the other structural
# attributes of the forest plot according to the exponential model for tree
# size distribution
# INPUT
# N: density of trees in the plot (per ha)
# Dm: mean diameter (in cm)
# x0: minimum dbh for inventory (in cm)
```

```
# x1: maximum dbh for truncation (in cm)
# OUTPUT
# Vector with 10 structural attributes: 1) density of trees, 2) basal area,
# 3) mean diameter, 4) equivalent diameter, 5) density of trees in the diameter
# class [10,30[ cm, 6) density of trees in the diameter class [30,60[ cm,
# 7) density of trees in the diameter class >= 60 cm, 8) aboveground biomass,
# 9) proportion of biomass in the diameter class => 60 cm, 10) wood density

if(any(Dm < x0)) stop("Mean diameter cannot be less than minimum diameter")
DmFUN <- function(z) ifelse(z==1,(x1-x0)/log(x1/x0),ifelse(z==2,x0*x1*log(
        x1/x0)/(x1-x0),(z-1)/(z-2)*(x0^(2-z)-x1^(2-z))/(x0^(1-z)-x1^(1-z))))
findint <- function(Dm) {
        zmn <- 2
        while(DmFUN(zmn)<Dm) zmn <- zmn-1
        zmx <- 3
        while(DmFUN(zmx)>Dm) zmx <- zmx+1
        c(zmn,zmx) }
z <- rep(NA,length(N))
for(i in seq_along(N)) z[i] <- uniroot(function(z) DmFUN(z)-Dm[i],findint(
        Dm[i]))$root
G <- N*(pi/4e4)*(z-1)/(z-3)*(x0^(3-z)-x1^(3-z))/(x0^(1-z)-x1^(1-z))
De <- sqrt(4e4*G/(pi*N))
a <- c(x0,30,60)
b <- c(a[-1],x1)
Ni <- as.data.frame(N*t(outer(a,1-z,"^")-outer(b,1-z,"^"))/(x0^(1-z)-x1^(1-z)))
names(Ni) <- paste("N",1:length(a),sep="")
rho <- 0.6
f <- function(x) rho*exp(-1.499+2.148*log(x)+0.207*(log(x)^2)-0.0281*(log(x)^3))
B <- p <- rep(NA,length(N))
h <- function(x,z) (z-1)*x^(-z)/(x0^(1-z)-x1^(1-z))
for(i in seq_along(N)) {
        B[i] <- N[i]*integrate(function(x,z) f(x)*h(x,z),x0,x1,
                z=z[i])$value/1e3
        p[i] <- N[i]*integrate(function(x,z) f(x)*h(x,z),60,x1,
                z=z[i])$value/1e3/B[i] }
cbind(N=N,G=G,Dm=Dm,De=De,Ni,B=B,p=p,R=rho)
}
```