```
################################################
#
# Deriving tree size distributions of tropical forests from lidar
# Franziska Taubert , Rico Fischer , Nikolai Knapp  and Andreas Huth
# correspondence: franziska.taubert@ufz.de; Tel.: +49-341-235-1896
#
# The supplementary file contains R-code to calculate
# the stem diameter distribution of a forest from lidar data.
#
################################################

require(spatstat)
require(hydroGOF)

################################################

# functions for plot discretization and selection
patchorderx <- function(arrayE,gridsizePx){
 # return the patch-number, in which tree is located
 # grid begins in the lower left corner with (1,1)
 # trees at boundaries/interfaces of patches always fall within the lower left patch

 # arrayE: tree measurements
 # gridsizePx: width (m) in x-direction of the forest patch

 xi <- as.numeric(arrayE[4]);

 n1 <- floor(xi/gridsizePx)+1;
 return (n1);
}

patchorderLx <- function(arrayE,gridsizePx){
 # return the patch-number, in which lidar return is located
 # grid begins in the lower left corner with (1,1)
 # lidar points at boundaries/interfaces of patches always fall within the lower left patch

 # arrayE: lidar point measurements
 # gridsizePx: width (m) in x-direction of the forest patch

 xi <- as.numeric(arrayE[1]);

 n1 <- floor(xi/gridsizePx)+1;
 return (n1);
}

patchordery <- function(arrayE,gridsizePy){
 # return the patch-number, in which tree is located
 # grid begins in the lower left corner with (1,1)
 # trees at boundaries/interfaces of patches always fall within the lower left patch

 # arrayE: tree measurements
 # gridsizePy: width (m) in y-direction of the forest patch
```

```
  yi <- as.numeric(arrayE[5]);
  n2 <- floor(yi/gridsizePy)+1;

  return (n2);
}

patchorderLy <- function(arrayE,gridsizePy){
  # return the patch-number, in which lidar return is located
  # grid begins in the lower left corner with (1,1)
  # lidar points at boundaries/interfaces of patches always fall within the lower left patch

  # arrayE: lidar point measurements
  # gridsizePx: width (m) in y-direction of the forest patch

  xi <- as.numeric(arrayE[2]);

  n1 <- floor(xi/gridsizePy)+1;
  return (n1);
}

DataListPatch <- function(posD,cellD,dataD){
  # return tree measurements which are within the patch of number cellD
  # posD: patch-numbers of tree measurements
  # cellD: specific patch
  # dataD: tree measurements

  inD <- which((posD[,1]==cellD[1]) & (posD[,2]==cellD[2]));

  if(length(inD)==0){
    subL <- array(0,c(0,0));
  }else{
    subL <- as.matrix(dataD[inD,]);
  }

  return (subL);
}

#############################################################################

getLeafInventory = function(cellE, datenarray, posvec, gridsizePx,gridsizePy,
                            Fmat, HeightLayer){
  # return vertical leaf area profile based on field inventory and leaf-tree matrix
  # only trees which are within the patch of number cellE are used
  # datenarray: includes tree measurements
  # cellE: specific patch
  # posvec: tree positions within the patch
  # gridsizePx,gridsizePy: x- and y-dimension of the patch
  # Fmat: leaf-tree matrix
  # HeightLayer: height layer discretization

  subD <- DataListPatch(posvec,cellE,datenarray);

  if(dim(subD)[1]!=0){
```

```
   plotsize = gridsizePx*gridsizePy # m²
   dbhs = as.numeric(subD[,11])/10.0 # convert to cm

   if(hall == "power"){
     heights = hd1p*(dbhs^hd2p)
   }else if(hall == "asymp"){
     heights = (hd1a*(dbhs/100.0))/(hd2a+(dbhs/100.0))
   }

   ###################################################
   ## Calculate leaf area profile of inventory using the leaf-tree matrix F

   histh = hist(heights,breaks=c(0,HeightLayer),plot=F)
   invleaf = (Fmat %*% histh$counts)/plotsize
  }else{
   invleaf = rep(0,length(HeightLayer))
  }
  return (c(cellE,invleaf))
}

#########################################################

getLeafLidar = function(cellE, gridsizePx,gridsizePy,HeightLayer,poslid,lidararray){
  # return vertical leaf area profile based on lidar data
  # only signals which are within the patch of number cellE are used
  # lidararray: includes lidar points
  # cellE: specific patch
  # poslid: positions of lidar returns within the patch
  # gridsizePx,gridsizePy: x- and y-dimension of the patch
  # HeightLayer: height layer discretization

  subLi <- DataListPatch(poslid,cellE,lidararray);

  if(dim(subLi)[1]!=0){

   plotsize = gridsizePx*gridsizePy # m²

   ###################################################
   ## Calculate leaf area profile from lidar by counting points

     hl = hist(as.numeric(subLi[,3]),breaks=c(0,HeightLayer),plot=F)
     pd = hl$counts/plotsize

     leafi = 0
     for(i in 1:(length(HeightLayer))){
       leafi = c(leafi,lparam*pd[length(HeightLayer)+1-i]*exp(kparam*deltah*sum(leafi)))
     }

     lidarleaf = rev(leafi)

  }else{
   lidarleaf = rep(0,length(HeightLayer))
```

```
 }
 return (c(cellE,lidarleaf))
}

#######################################################

getTreesInventory = function(cellE, datenarray, posvec, gridsizePx,gridsizePy,HeightLayer){
 # return tree size distribution based on field inventory
 # only trees which are within the patch of number cellE are used
 # datenarray: includes tree measurements
 # cellE: specific patch
 # posvec: tree positions within the patch
 # gridsizePx,gridsizePy: x- and y-dimension of the patch
 # HeightLayer: height layer discretization

 subD <- DataListPatch(posvec,cellE,datenarray);

 if(dim(subD)[1]!=0){

  plotsize = gridsizePx*gridsizePy # m²
  dbhs = as.numeric(subD[,11])/10.0 # convert to cm

  if(hall == "power"){
   DBHLayer <- (HeightLayer/hd1p)^(1.0/hd2p)
  }else if(hall == "asymp"){
   HeightLayer <- seq(1,maxCan,deltah)
   dx = (100:100000)/10000.0
   heightsq = (hd1a*dx)/(hd2a+dx)
   DBHLayer = vector("numeric",length=0)
   for(iyx in 1:length(HeightLayer)){
    dif = abs(heightsq - HeightLayer[iyx])
    indx = which(dif == min(dif))
    DBHLayer = c(DBHLayer,100*dx[indx])
   }
  }

  histd = hist(dbhs,breaks = c(0,DBHLayer),plot=F)
  invtrees = histd$counts/plotsize

 }else{
  invtrees = rep(0,length(HeightLayer))
 }
 return (c(cellE,invtrees))
}

#######################################################

getTreesLidar = function(cellE, gridsizePx,gridsizePy,
                         Fmat, HeightLayer,poslid,lidararray){
 # return tree size distribution based on lidar data
 # only lidar points which are within the patch of number cellE are used
 # lidararray: includes lidar measurements
 # cellE: specific patch
```

```
# poslid: positions of lidar signals within the patch
# gridsizePx,gridsizePy: x- and y-dimension of the patch
# Fmat: leaf-tree matrix
# HeightLayer: height layer discretization

subLi <- DataListPatch(poslid,cellE,lidararray);

if(dim(subLi)[1]!=0){

 plotsize = gridsizePx*gridsizePy # m²

 ################################################
 ## Calculate leaf area profile from lidar by counting points

  hl = hist(as.numeric(subLi[,3]),breaks=c(0,HeightLayer),plot=F)
  pd = hl$counts/plotsize

  leafi = 0
  for(i in 1:(length(HeightLayer))){
   leafi = c(leafi,lparam*pd[length(HeightLayer)+1-i]*exp(kparam*deltah*sum(leafi)))
  }

  lidarleaf = rev(leafi)


 if(calc == "direct"){
  ## direct calculation of stem numbers using the inverse of the leaf-tree matrix
  ntrees = solve(Fmat) %*% (lidarleaf[2:length(lidarleaf)] * plotsize)
 }

 if(calc == "backward"){
  ## iterative numeric backward calculation
  L = lidarleaf[2:length(lidarleaf)] * plotsize
  nL = length(L)
  Lunexplained = L
  ntrees = vector("numeric",length=nL)
  tol = 0.05

  for(inx in seq(nL,1,-1)){

   if(Lunexplained[inx] > (Fmat[inx,inx] * tol)){
    kL = floor(Lunexplained[inx]/Fmat[inx,inx])
    if((Lunexplained[inx] - (kL * Fmat[inx,inx])) > tol*L[inx]){
     kL = kL+1
    }
    ntrees[inx] = kL
   }

   Lunexplained = Lunexplained - (ntrees[inx] * Fmat[,inx])

  }
 }
```

```
   lidartrees = ntrees/plotsize

  }else{
   lidartrees = rep(0,length(HeightLayer))
  }
  return (c(cellE,lidartrees))
}

#########################################################

getStats = function(cellE, datenarray, posvec, gridsizePx,gridsizePy,
                Fmat, HeightLayer,poslid,lidararray){
  # return statistics for the comparison of tree size distribution derived from lidar and inventory
  # only lidar points and trees which are within the patch of number cellE are used
  # lidararray: includes lidar measurements
  # lidararray: includes tree measurements
  # cellE: specific patch
  # poslid: positions of lidar signals within the patch
  # posvec: positions of trees within the patch
  # gridsizePx,gridsizePy: x- and y-dimension of the patch
  # Fmat: leaf-tree matrix
  # HeightLayer: height layer discretization

  subD <- DataListPatch(posvec,cellE,datenarray);
  subLi <- DataListPatch(poslid,cellE,lidararray);
  plotha = (gridsizePx * gridsizePy) / 10000;

  if(dim(subD)[1]!=0 && dim(subLi)[1]!=0){

    plotsize = gridsizePx*gridsizePy # m²
    dbhs = as.numeric(subD[,11])/10.0 # convert to cm
    if(hall == "power"){
     heights = hd1p*(dbhs^hd2p)
    }else if(hall == "asymp"){
     heights = (hd1a*(dbhs/100.0))/(hd2a+(dbhs/100.0))
    }

    ###################################################
    ## Calculate leaf area profile from lidar by counting points
     hl = hist(as.numeric(subLi[,3]),breaks=c(0,HeightLayer),plot=F)
     pd = hl$counts/plotsize

     leafi = 0
     for(i in 1:(length(HeightLayer))){
      leafi = c(leafi,lparam*pd[length(HeightLayer)+1-i]*exp(kparam*deltah*sum(leafi)))
     }

     lidarleaf = rev(leafi)

    ## inventory-based leaf area profile
    invleaf = getLeafInventory(cellE, datenarray, posvec, gridsizePx,gridsizePy,
                          Fmat, HeightLayer)
```

```
## Calculate tree size distribution from lidar profile
if(calc == "direct"){
  ## direct calculation of stem numbers
  ntrees = solve(Fmat) %*% (lidarleaf[2:length(lidarleaf)] * plotsize)
}

if(calc == "backward"){
  ## iterative numerical backward calculation
  L = lidarleaf[2:length(lidarleaf)] * plotsize
  nL = length(L)
  Lunexplained = L
  ntrees = vector("numeric",length=nL)
  tol = 0.05

  for(inx in seq(nL,1,-1)){

    if(Lunexplained[inx] > (Fmat[inx,inx] * tol)){
      kL = floor(Lunexplained[inx]/Fmat[inx,inx])
      if((Lunexplained[inx] - (kL * Fmat[inx,inx])) > tol*L[inx]){
        kL = kL+1
      }
      ntrees[inx] = kL
    }

    Lunexplained = Lunexplained - (ntrees[inx] * Fmat[,inx])

  }
}

lidartrees = ntrees

## calculate tree size distribution from inventory
if(hall == "power"){
  DBHLayer <- (HeightLayer/hd1p)^(1.0/hd2p)
}else if(hall == "asymp"){
  HeightLayer <- seq(1,maxCan,deltah)
  dx = (100:100000)/10000.0
  heightsq = (hd1a*dx)/(hd2a+dx)
  DBHLayer = vector("numeric",length=0)
  for(iyx in 1:length(HeightLayer)){
    dif = abs(heightsq - HeightLayer[iyx])
    indx = which(dif == min(dif))
    DBHLayer = c(DBHLayer,100*dx[indx])
  }
}

histd = hist(dbhs,breaks = c(0,DBHLayer),plot=F)
invtrees = histd$counts

# remove classes in which both (inventory & lidar) have no stems
rem0 = max(which(lidartrees > 0))
rem1 = max(which(invtrees > 0))
dr = max(rem0,rem1)+1
```

```
invtrees = invtrees[-(dr:length(lidartrees))]
dclassupper = DBHLayer[-(dr:length(lidartrees))]
lidartrees = lidartrees[-(dr:length(lidartrees))]

## statistics
## tree sizes
# remove first two height layers (< 3 m)
xo = invtrees[-c(1,2)]
yo = lidartrees[-c(1,2)]
dclasslower = c(0,dclassupper[-c(length(dclassupper))])
dclassupper = dclassupper
dclassmid = (dclasslower+dclassupper)/2.0
dclasslower = dclasslower[-c(1,2)]
dclassupper = dclassupper[-c(1,2)]
dclassmid = dclassmid[-c(1,2)]

## linear regression
x1a = log(xo[which(xo>0 & yo>0)]/plotha)
y1a = log(yo[which(xo>0 & yo>0)]/plotha)
if(length(x1a)>0){
  regx=summary(lm(y1a ~ x1a))
}else{
  regx=summary(lm(0 ~ 0))
}

## calculation of RMSE and nRMSE
hdl =hist(dclassupper,breaks=seq(0,max(dclassupper,na.rm = T)+9.99,10),plot=F)
indices = c(0,cumsum(hdl$counts))
xon = vector("numeric",length=0)
yon = vector("numeric",length=0)
for(inb in 1:length(hdl$counts)){
  xon = c(xon,sum(xo[(indices[inb]+1):(indices[inb+1])]))
  yon = c(yon,sum(yo[(indices[inb]+1):(indices[inb+1])]))
}
# only diameter classes >= 10 cm
xon = xon[-1]/plotha
yon = yon[-1]/plotha
if(length(xon)>0){
  ermy1 = rmse(xon,yon)
  ermy2 = ermy1/(max(xon,na.rm = T)-min(xon,na.rm = T))
}else{
  ermy1 = NA
  ermy2 = NA
}

## classified in size categories
small = which(dclassupper<10)
mid = which(dclassupper>10 & dclassupper<50)
large = which(dclassupper>50)
x0a = (xo[small])/plotha
y0a = (yo[small])/plotha
x0b = (xo[mid])/plotha
y0b = (yo[mid])/plotha
```

```
 x0c = (xo[large])/plotha
 y0c = (yo[large])/plotha

 if(length(x0a)>0){
  ermz11 = rmse(x0a,y0a)
 }else{
  ermz11 = NA
 }

 if(length(x0b)>0){
  ermz21 = rmse(x0b,y0b)
 }else{
  ermz21 = NA
 }

 if(length(x0c)>0){
  ermz31 = rmse(x0c,y0c)
 }else{
  ermz31 = NA
 }

 ### properties of lidar leaf profile
 ns=100*length(yo[which(yo<0)])/length(yo)
 remx = c(1,2,length(lidarleaf))
 leafl = lidarleaf[-remx]
 posvalspos = which(leafl > 0)
 posvals = leafl[posvalspos]

 ### forest stand structure indices from inventory
 tdens10 = length(dbhs[which(dbhs>10)])/plotha
 ba10 = sum((pi/4)*((dbhs[which(dbhs>10)]/100.0)^2),na.rm=T)/plotha
 th = heights

 ## forest structure comparison (based on lidar estimate)
 sel = which(dclassupper>10)+1
 tdenslid10 = sum(yo[sel],na.rm=T)/plotha

 dbcl = dclassmid/(100.0)
 balidper = (((pi/4)*(dbcl^2))*yo)
 balid10 = sum(balidper[sel],na.rm=T)/plotha #m² per ha

 stats = c(regx$coefficients[2],regx$r.squared,
        ermy1,ermy2,ermz11,ermz21,ermz31,
        ns, median(posvalspos),max(posvalspos),var(posvalspos),
        median(posvals),max(posvals),var(posvals),
        weighted.median(posvalspos,posvals),weighted.var(posvalspos,posvals),
        length(as.numeric(subLi[,3])),sd(th),median(th),
        ba10,tdens10,
        balid10,tdenslid10)

}else{
 stats = c(rep(0,23))
}
```

```
  return (c(cellE,stats))
}


#########################################################
#########################################################
##### read-in data and preparation ###############
# set your working directory
setwd("...")
da1 = read.delim("PlotsDataReport7.txt", header=T,sep="\t")
li1 = read.delim("BCI_50ha_FieldLidar_thinned_4pts_sqm.txt",header=T,sep=";")

# remove trees with DBH = NA, DBH < 1 cm or dead/lost
rem1 = which(is.na(da1$DBH)==T)
if(length(rem1)>0){data = da1[-rem1,]}
rem2 = which(data$DBH < 10)
if(length(rem2)>0){data = data[-rem2,]}
sel = which(data$Status == "alive")
data = data[sel,]

# set lidar ground returns to 0
rem1 = which(li1$Z <= 0)
li1$Z[rem1] = 0

####################
## Settings

## Tree allometry
# Allometric height-stem diameter function
hall = "asymp" # "asymp" or "power"
# power = power-law -> h (m) = hd1*(d(cm) ^ hd2)
# asymp = asymptotic function -> h (m) = (hd1a * d(m))/(hd2a + d(m))

# Power-law parameter
hd2p = 0.6
hd1p = 2.74 #equal to: 43.4*((1/100)^hd2p)

## Asymptotic function parameter
hd1a = 57.4
hd2a = 0.43

# Allometric crown radius-stem diameter function: cr (m) = cr1 * (d(cm) ^ cr2)
cr2 = 0.68
cr1 = (9.08*((1/100)^cr2))

# Allometric crown length-stem diameter function: cl (m) = cl1 * h(m)
cl1 = 0.4

# maximum canopy height
if(hall == "power"){
  maxCan = 80
}else{
  maxCan = 55
```

```
}

## Matrix and layer discretization
rho = 0.44 # m²/m³ leaf density of one tree crown
shape = "ellipsoid" # tree crown shape: ellipsoid, sphere, cylinder
deltah = 1 # in m, width of height layer

#####################################################

if(hall == "power"){
  HeightLayer <- seq(1,maxCan,deltah)
  DBHLayer <- (HeightLayer/hd1p)^(1.0/hd2p)
}else if(hall == "asymp"){
  HeightLayer <- seq(1,maxCan,deltah)
  dx = (100:100000)/10000.0
  heightsq = (hd1a*dx)/(hd2a+dx)
  DBHLayer = vector("numeric",length=0)
  for(iyx in 1:length(HeightLayer)){
    dif = abs(heightsq - HeightLayer[iyx])
    indx = which(dif == min(dif))
    DBHLayer = c(DBHLayer,100*dx[indx])
  }
}

CRLayer = cr1*(DBHLayer^cr2)
CLLayer = cl1*HeightLayer

# leaf-tree matrix F
n = length(HeightLayer)
Fmatrix = array(0,c(n,n))

for(j in 1:(length(DBHLayer))){

  V_sum = 0
  db = DBHLayer[j]
  rsph = CRLayer[j]
  hsph = HeightLayer[j]
  lsph = CLLayer[j]

  if(shape == "sphere"){

    for(i in seq(floor(hsph/deltah),1,-1)){
      # height of sphere segment
      hsphere = min(2*rsph, (hsph - ((i-1)*deltah))) # ensure that crown length is not larger than height
      # volume of sphere segement
      V = (pi/3) *(hsphere^2) * (3*rsph - hsphere)
      # volume of sphere section
      v = V - V_sum;
      V_sum = V_sum + v;
      # matrix entry
      Fmatrix[i, j] = (rho * v) / deltah
    }
  }
```

```
 if(shape == "cylinder"){
  V = lsph * (pi*(rsph^2))
  for(i in 1:(length(HeightLayer))){
   if (((hsph-lsph) <= (deltah*i)) && (hsph >= (deltah*(i-1)))){
    Fmatrix[i, j] = (min(deltah*i,hsph) - max(deltah*(i-1),hsph-lsph))/lsph*V*(rho/deltah)
   }
  }
 }

 if(shape == "ellipsoid"){
  for(i in seq(floor(hsph/deltah),1,-1)){
   l = lsph/2
   hsphere = min(2*l, hsph-(i-1)*deltah)
   V = pi*(rsph^2)/(l^2)*(((l^3)-(l^3)/3)-((l^2)*(l-hsphere)-(l-hsphere)^3/3))
   v = V - V_sum;
   V_sum = V_sum + v;
   Fmatrix[i, j] = (rho * v) / deltah;
  }
 }
}

#########################################################
##### plot selection
b.areax <- 1000.
b.areay <- 500.

## plot size: choose different sizes (e.g. 1000 and 500 for full plot, or 100 for each for 1-ha) to
calculate small-scale analysis
gridsizex <- 100
gridsizey <- 100

### all patch numbers
b.xcoord <- rep(seq(1,b.areax/gridsizex,1),b.areay/gridsizey);
b.ycfunc <- function(x) rep(x,b.areax/gridsizex);
b.ycoord <- as.vector(apply(array(seq(1,b.areay/gridsizey,1),c(1,b.areay/gridsizey)),2,
                    b.ycfunc));
b.cells <- array(0,c((b.areax/gridsizex)*(b.areay/gridsizey),2))
b.cells[,1] <- b.xcoord
b.cells[,2] <- b.ycoord

### positions of tree measurements
b.posit <- array(0,c(dim(data)[1],2))
b.posit[,1] <- apply(data,1,patchorderx,gridsizePx=gridsizex);
b.posit[,2] <- apply(data,1,patchordery,gridsizePy=gridsizey);

### positions of lidar points
l.posit = array(0,c(dim(li1)[1],2))
l.posit[,1] <- apply(li1,1,patchorderLx,gridsizePx=gridsizex);
l.posit[,2] <- apply(li1,1,patchorderLy,gridsizePy=gridsizey);

#####################################
```

```
lparam = 1.0
kparam = 0.2

#### Calculation of leaf area profiles
resLinv = apply(b.cells,1,getLeafInventory,data,b.posit,gridsizex,gridsizey,Fmatrix,HeightLayer)
resLlid = apply(b.cells,1,getLeafLidar,gridsizex,gridsizey,HeightLayer,l.posit,li1)

#### Calculation of tree size distributions
resNinv = apply(b.cells,1,getTreesInventory,data,b.posit,gridsizex,gridsizey,HeightLayer)

## different calculation methods for solving linear equation system
calc = "backward"
# direct: direct solving (with potential negative solutions)
# backward: numerical backward calulcation

resNlid = apply(b.cells,1,getTreesLidar,gridsizex,gridsizey,Fmatrix,HeightLayer,l.posit,li1)

### Calculate statistics
resStat = apply(b.cells,1,getStats,data,b.posit,gridsizex,gridsizey,Fmatrix,HeightLayer,l.posit,li1)

############ Visualization ##################
plotno = 1
xl = DBHLayer[-c(1,2)]
yl = resNinv[-c(1,2,3,4),plotno]*10000
plot(xl[which(yl>0)],yl[which(yl>0)],type="l",log="xy",
    xlab="Stem diameter (cm)",ylab="Tree number (1/ha)",
    main="",xlim=c(0.5,1000),ylim=c(1E-2,1E4),
    lwd=4,pch=-1,cex=1.5,col=rgb(120,120,120,255,maxColorValue = 255))

yl = resNlid[-c(1,2,3,4),1]*10000
points(xl[which(yl>0)],yl[which(yl>0)],type="p",
     lwd=1,pch=20,cex=2.5,col=rgb(34,150,34,170,maxColorValue = 255))
legend("top",c("Inventory","Lidar-derived"),lty=c(1,-1),pch=c(-1,20),lwd=c(4,1),
     horiz = T,xpd=T,inset = c(0, -0.15),pt.cex=1.,bty="n",
     col=c(rgb(100,100,100,255,maxColorValue = 255),
         rgb(34,150,34,180,maxColorValue = 255)),cex=0.8)
```