Package ‘rLiDAR’

July 12, 2017

Type Package

Title LiDAR Data Processing and Visualization

Version 0.1.1

Author Carlos A. Silva, Nicholas L. Crookston,
Andrew T. Hudak, Lee A. Vierling, Carine Klauberg, Adri<c3><a1>n Cardil

Maintainer Carlos A. Silva <carlos_engflorestal@outlook.com>

Depends R (>= 3.3.2)

Imports spatstat,sp,deldir,plyr,raster,geometry,rgl,bitops, methods

Description Set of tools for reading, processing and visualizing small set of
LiDAR (Light Detection and Ranging) data for forest inventory applications.

License GPL (>= 2)

RoxygenNote 6.0.1

Repository CRAN

Repository/R-Forge/Project rlidar

Repository/R-Forge/Revision 119

Repository/R-Forge/DateTimeStamp 2017-07-12 09:31:23

Date/Publication 2017-07-12 10:39:07 UTC

NeedsCompilation no

R topics documented:

---

chm *LiDAR-derived Canopy Height Model - (CHM)*

Description

LiDAR-derived Canopy Height Model - (CHM)

Usage

```
data(chm)
```

Format

The format is:'RasterLayer'

Details

The LiDAR-derived-CHM provided as an example dataset is from a longleaf pine forest at Eglin AFB, USA.

Source

The CHM was generated using Lastools software. The LiDAR data collection was funded by a grant (11-2-1-11) from the Joint Fire Science Program: Data set for fuels, fire behavior, smoke, and fire effects model development and evaluation-the RxCADRE project. R. Ottmar, PI; multiple Co-Is."

References

USDA Forest Service. Rocky Mountain Research Station - RMRS - Moscow, Idaho, USA.

Examples

```
data(chm)
## plot(chm)
```

CHMsmoothing *LiDAR-derived Canopy Height Model (CHM) smoothing*

Description

LiDAR-derived Canopy Height Model (CHM) smoothing is used to eliminate spurious local maxima caused by tree branches.

Usage

```
CHMsmoothing(chm, filter, ws, sigma)
```

Arguments

chm	A LiDAR-derived Canopy Height Model (CHM) RasterLayer or SpatialGridDataFrame file.
filter	Filter type: mean, median, maximum or Gaussian. Default is mean.
ws	The dimension of a window size, e.g. 3,5, 7 and so on. Default is 5.
sigma	Used only when filter parameter is equal to Gaussian, e.g. 0.5, 1.0, 1.5 and so on. Default is 0.67.

Value

Returns a CHM-smoothed raster.

Author(s)

Carlos Alberto Silva.

See Also

focal in the *raster* package.

Examples

```
#=======================================================================#
# Importing the LiDAR-derived CHM file
data(chm) # or set a CHM. e.g. chm<-raster("CHM_stand.asc")

#=======================================================================#
# Example 01: Smoothing the CHM using a Gaussian filter
#=======================================================================#
# Set the ws:
ws<-3 # dimension 3x3

# Set the filter type
filter<-"Gaussian"
```

```
# Set the sigma value
sigma<-0.6

# Smoothing CHM
sCHM<-CHMsmoothing(chm, filter, ws, sigma)

#=======================================================================#
# Example 02: Smoothing the CHM using a mean filter
#=======================================================================#
# Set the ws:
ws<-5 # dimension 5x5

# Set the filter type
filter<-"mean"

# Smoothing and plotting LiDAR-derived CHM
sCHM<-CHMsmoothing(chm, filter, ws)
```

---

chullLiDAR2D — *2D Convex hull of individual tree LiDAR-derived point cloud*

Description

Compute and plot the 2D convex hull of individual tree LiDAR-derived point cloud

Usage

```
chullLiDAR2D(xyid)
```

Arguments

`xyid`	A 3-column matrix with the x, y coordinates and points id of the LiDAR point cloud.

Value

Returns A list with components "chullPolygon" and "chullArea", giving the polygon and area of the convex hull.

Author(s)

Carlos Alberto Silva

References

grDevices package,see `chull`.

Examples

```
# Importing LAS file:
LASfile <- system.file("extdata", "LASexample1.las", package="rLiDAR")

# Reading LAS file
LAS<-readLAS(LASfile,short=TRUE)

# Height subsetting the data
xyz<-subset(LAS[,1:3],LAS[,3] >= 1.37)

# Getting LiDAR clusters
set.seed(1)
clLAS<-kmeans(xyz, 32)

# Set the points id
id<-as.factor(clLAS$cluster)

# Set the xyid input
xyid<-cbind(xyz[,1:2],id)

# Compute the LiDAR convex hull of the clusters
chullTrees<-chullLiDAR2D(xyid)

# Plotting the LiDAR convex hull
library(sp)
plot(SpatialPoints(xyid[,1:2]),cex=0.5,col=xyid[,3])
plot(chullTrees$chullPolygon,add=TRUE, border='green')

# Get the ground-projected area of LiDAR convex hull
chullList<-chullTrees$chullArea
summary(chullList) # summary
```

`chullLiDAR3D` *3D convex hull of the individual tree LiDAR-derived point cloud*

Description

Compute and plot the 3D convex hull (and its surface area and volume) of the individual tree LiDAR-derived point cloud.

Usage

```
chullLiDAR3D(xyzid,plotit=TRUE,col="forestgreen",alpha=0.8)
```

Arguments

`xyzid`	A matrix with four columns (xyz coordinates and tree id).
`plotit`	Logical. If FALSE, returns only volume and surface area.

`col`	A vector or a caracter of the convex hull color.
`alpha`	A vector or a caracter of the convex hull transparency (0-1).

Value

A list with components 'crownvolume' and 'crownsurface', giving the volume and surface of the convex hull.

Author(s)

Carlos Alberto Silva. Uses code by Remko Duursma (*YplantQMC* package,see "crownhull").

References

www.qhull.org and *geometry* package (see convhulln).

Examples

```
# Importing LAS file:
LASfile <- system.file("extdata", "LASexample1.las", package="rLiDAR")

# Reading LAS file
LAS<-readLAS(LASfile,short=TRUE)

# Setring the xyz coordinates and subsetting the data
xyz<-subset(LAS[,1:3],LAS[,3] >= 1.37)

# Finding clusters
clLAS<-kmeans(xyz, 32)

# Set the id vector
id<-as.factor(clLAS$cluster)

#=================================================#
# Example 01
#=================================================#
# Set the alpha
alpha<-0.6

# Set the plotCAS parameter
plotit=TRUE

# Set the convex hull color
col="forestgreen"

# Combining xyz and id
xyzid<-cbind(xyz,id)

# Get the volume and surface area
library(rgl)
open3d()
```

```
volumeList<-chullLiDAR3D(xyzid=xyzid, plotit=plotit, col=col,alpha=alpha)
summary(volumeList) # summary

plot3d(xyzid[,1:3], add=TRUE)   # add the 3D point cloud
axes3d(c("x+", "y-", "z-"))                 # axes
grid3d(side=c('x+', 'y-', 'z'), col="gray") # grid
title3d(xlab = "UTM Easthing", ylab = "UTM Northing",zlab = "Height", col="red")
aspect3d(1,1,0.7) # scale

#=================================================#
# Example 02
#=================================================#
# Set the alpha
alpha<-0.85

# Set the plotCAS parameter
plotit=TRUE

# Set the convex hull color
col=levels(factor(id))

# Combining xyz and id
xyzid<-cbind(xyz,id)

# Get the volume and surface area
open3d()
volumeList<-chullLiDAR3D(xyzid=xyzid, plotit=plotit, col=col,alpha=alpha)
summary(volumeList)

# Add other plot parameters
plot3d(xyzid[,1:3], col=xyzid[,4], add=TRUE)   # add the 3D point cloud
axes3d(c("x+", "y-", "z-"))                 # axes
grid3d(side=c('x+', 'y-', 'z'), col="gray") # grid
title3d(xlab = "UTM Easthing", ylab = "UTM Northing",zlab = "Height", col="red")
aspect3d(1,1,0.7) # scale
```

`CrownMetrics` *LiDAR-derived individual tree crown metrics*

Description

Compute individual tree crown metrics from lidar data

Usage

```
CrownMetrics(xyziId)
```

Arguments

`xyziId`	A 5-column matrix with the x, y, z coordinates, intensity and the tree id classification for the LiDAR point cloud.

Details

List of the individual tree crown metrics:

- TotalReturns: Total number of returns
- ETOP - UTM Easting coordinate of the tree top
- NTOP - UTM Northing coordinate of the tree top
- EMIN - Minimum UTM Easting coordinate
- NMIN - Minimum UTM Northing coordinate
- EMAX - Maximum UTM Easting coordinate
- NMAX - Maxmium UTM Northing coordinate
- EWIDTH - Tree crown width 01
- NWIDTH - Tree crown width 02
- HMAX - Maximum Height
- HMEAN - Mean height
- HSD - Standard deviation of height
- HCV - Coefficient of variation of height
- HMOD - Mode of height
- H5TH - 5th percentile of height
- H10TH - 10th percentile of height
- H20TH - 20th percentile of height
- H25TH - 25th percentile of height
- H30TH - 30th percentile of height
- H40TH - 40th percentile of height
- H50TH - 50th percentile of height
- H60TH - 60th percentile of height
- H70TH - 70th percentile of height
- H75TH - 75th percentile of height
- H80TH - 80th percentile of height
- H90TH - 90th percentile of height
- H95TH - 95th percentile of height
- H99TH - 99th percentile of height
- IMAX - Maximum intensity
- IMEAN - Mean intensity
- ISD - Standard deviation of intensity

- ICV - Coefficient of variation of intensity
- IMOD - Mode of intensity
- I5TH - 5th percentile of intensity
- I10TH - 10th percentile of intensity
- I20TH - 20th percentile of intensity
- I25TH - 25th percentile of intensity
- I30TH - 30th percentile of intensity
- I40TH - 40th percentile of intensity
- I50TH - 50th percentile of intensity
- I60TH - 60th percentile of intensity
- I70TH - 70th percentile of intensity
- I75TH - 75th percentile of intensity
- I80TH - 80th percentile of intensity
- I90TH - 90th percentile of intensity
- I95TH - 95th percentile of intensity
- I99TH - 99th percentile of intensity

Value

Returns A matrix of the LiDAR-based metrics for the individual tree detected.

Author(s)

Carlos Alberto Silva

Examples

```
#=======================================================================#
# Individual tree detection using K-means cluster
#=======================================================================#
# Importing LAS file:
LASfile <- system.file("extdata", "LASexample1.las", package="rLiDAR")

# Reading LAS file
LAS<-readLAS(LASfile,short=TRUE)

# Setting the xyz coordinates and subsetting the data
xyzi<-subset(LAS[,1:4],LAS[,3] >= 1.37)

# Finding clusters (trees)
clLAS<-kmeans(xyzi[,1:2], 32)

# Set the tree id vector
Id<-as.factor(clLAS$cluster)
```

```
# Combining xyzi and tree id
xyziId<-cbind(xyzi,Id)

#=======================================================================#
#  Computing individual tree LiDAR metrics
#=======================================================================#

TreesMetrics<-CrownMetrics(xyziId)
head(TreesMetrics)
```

`FindTreesCHM` *Individual tree detection whitin the LiDAR-derived Canopy Height Model (CHM)*

Description

Detects and computes the location and height of individual trees within the LiDAR-derived Canopy Height Model (CHM). The algorithm implemented in this function is local maximum with a fixed window size.

Usage

```
FindTreesCHM(chm,fws,minht)
```

Arguments

chm	A LiDAR-derived Canopy Height Model (CHM) raster file.
fws	A single dimension (in raster grid cell units) of fixed square window size, e.g. 3, 5, 7 and so on. Default is 5.
minht	Height threshold. Detect individual trees above specified height threshold, e.g. 1.37, 2.0, 3.5 m and so on. Default is 1.37 m.

Value

Returns A matrix with four columns (tree id, xy coordinates, and height).

Author(s)

Carlos Alberto Silva

Examples

```
# Importing the LiDAR-derived CHM raster file
data(chm) # or set a CHM. e.g. chm<-raster("CHM_stand.asc")

# Smoothing CHM
schm<-CHMsmoothing(chm, "mean", 5)
```

```
# Setting the fws:
fws<-5 # dimention 5x5

# Setting the specified height above ground for detectionbreak
minht<-8.0

# Getting the individual tree detection list
treeList<-FindTreesCHM(schm, fws, minht)
summary(treeList)

# Plotting the individual tree location on the CHM
library(raster)
plot(chm, main="LiDAR-derived CHM")
library(sp)
XY<-SpatialPoints(treeList[,1:2]) # Spatial points
plot(XY, add=TRUE, col="red")       # plotthing tree location
```

ForestCAS *Individual trees crown deliniation from LiDAR-derived Canopy Height Model (CHM)*

Description

Delineate and compute ground-projected area of individual tree crowns detected from LiDAR-derived CHM

Usage

```
ForestCAS(chm, loc, maxcrown, exclusion)
```

Arguments

`chm`	A LiDAR-derived Canopy Height Model (CHM) RasterLayer or SpatialGridDataFrame file.
`loc`	A matrix or dataframe with three columns (tree xy coordinates and height).
`maxcrown`	A single value of the maximum individual tree crown radius expected. Default 10.0 m.
`exclusion`	A single value from 0 to 1 that represents the

Value

Returns a list that contains the individual tree canopy boundary polygons and the 4-column matrix with the tree xy coordinates, heights and ground-projected canopy area (with units of square meters).

Author(s)

Carlos Alberto Silva

Examples

```
## Not run:

# Import the LiDAR-derived CHM file
data(chm) # or set a CHM. e.g. chm<-raster("CHM_stand.asc")

# Set the loc parameter
sCHM<-CHMsmoothing(chm, filter="mean", ws=5) # smoothing CHM
loc<-FindTreesCHM(sCHM, fws=5, minht=8)      # or import a tree list

# Set the maxcrown parameter
maxcrown=10.0

# Set the exclusion parameter
exclusion=0.3 # 30

# Compute individual tree detection canopy area
canopy<-ForestCAS(chm, loc, maxcrown, exclusion)

#=================================================================================#
# Retrieving the boundary for individual tree detection and canopy area calculation
#=================================================================================#
boundaryTrees<-canopy[[1]]
# Plotting the individual tree canopy boundary over the CHM
plot(chm, main="LiDAR-derived CHM")
plot(boundaryTrees, add=T, border='red', bg='transparent') # adding tree canopy boundary

#============================================================================#
# Retrieving the list of individual trees detected for canopy area calculation
#============================================================================#
canopyList<-canopy[[2]] # list of ground-projected areas of individual tree canopies
summary(canopyList)     # summary

# Spatial location of the trees
library(sp)
XY<-SpatialPoints(canopyList[,1:2])    # Spatial points
plot(XY, col="black", add=T, pch="*")  # adding tree location to the plot

## End(Not run)
```

`LASmetrics` *LiDAR-derived metrics*

Description

Compute LiDAR metrics that describe statistically the Lidar dataset

Usage

```
LASmetrics(LASfile, minht, above)
```

Arguments

`LASfile`	A LAS standard LiDAR data file
`minht`	Use only returns above specified height break, e.g. 1.30 m. Default is 1.37 m.
`above`	Compute covers metrics using specified height break, e.g. 2.5 m. Default is 2 m.

Value

Returns A matrix with the LiDAR-derived vegetation height and canopy cover metrics (see *cloudmetrics*, in McGaughey, 2014)

Author(s)

Carlos Alberto Silva

See Also

McGaughey, R. 2014. FUSION/LDV: Software for lidar data analysis and visualization. Version 3.41. Seattle, WA: U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station.

List of the LiDAR-derived metrics:

- Total all return count
- Total first return count
- Total all return count above *minht*
- Return 1 count above *minht*
- Return 2 count above *minht*
- Return 3 count above *minht*
- Return 5 count above *minht*
- Return 6 count above *minht*
- Return 7 count above *minht*
- Return 8 count above *minht*
- Return 9 count above *minht*
- HMIN - Maximum Height
- HMAX - Maximum Height
- HMEAN - Mean height
- HMOD - Modal height
- HMEDIAN - Median height
- HSD - Standard deviation of heights

- HVAR - Variance of heights
- HCV - Coefficient of variation of heights
- HKUR - Kurtosis of Heights
- HSKE - Skewness of Heights
- H01TH - 01th percentile of height
- H05TH - 05th percentile of height
- H10TH - 10th percentile of height
- H15TH - 15th percentile of height
- H20TH - 20th percentile of height
- H25TH - 25th percentile of height
- H30TH - 30th percentile of height
- H35TH - 35th percentile of height
- H40TH - 40th percentile of height
- H45TH - 45th percentile of height
- H50TH - 50th percentile of height
- H55TH - 55th percentile of height
- H60TH - 60th percentile of height
- H65TH - 65th percentile of height
- H70TH - 70th percentile of height
- H75TH - 75th percentile of height
- H80TH - 80th percentile of height
- H90TH - 90th percentile of height
- H95TH - 95th percentile of height
- H99TH - 99th percentile of height
- CRR - Canopy relief ratio
- IMIN - Minimum intensity
- IMAX - Maximum intensity
- IMEAN - Mean intensity
- IMOD - Modal intensity
- IMEDIAN - Median intensity
- ISD - Standard deviation of intensities
- IVAR - Variance of heights
- ICV - Coefficient of variation of intensities
- IKUR - Kurtosis of intensities
- ISKE - Skewness of intensities
- I01TH - 1th percentile of intensity
- I05TH - 5th percentile of intensity

- I10TH - 10th percentile of intensity
- I15TH - 15th percentile of intensity
- I20TH - 20th percentile of intensity
- I25TH - 25th percentile of intensity
- I30TH - 30th percentile of intensity
- I35TH - 35th percentile of intensity
- I40TH - 40th percentile of intensity
- I45TH - 45th percentile of intensity
- I50TH - 50th percentile of intensity
- I55TH - 55th percentile of intensity
- I60TH - 60th percentile of intensity
- I65TH - 65th percentile of intensity
- I70TH - 70th percentile of intensity
- I75TH - 75th percentile of intensity
- I80TH - 80th percentile of intensity
- I90TH - 90th percentile of intensity
- I95TH - 95th percentile of intensity
- I99TH - 99th percentile of intensity
- Pentage first returns above *above*
- Percentage all returns above *above*
- (All returns above above / Total first returns)*100
- First returns above *above*
- All returns above *above*
- Percentage first returns above mean
- Percentage first returns above mode
- Percentage all returns above mean
- Percentage all returns above mode
- (All returns above mean / Total first returns)*100
- (All returns above mode / Total first returns)* 100
- First returns above mean"
- First returns above mode
- All returns above mean
- All returns above mode

Examples

```
#=======================================================================#
# Example 01: Computing LiDAR metrics for a single LAS file
#=======================================================================#
# Import the LAS data file
LASfile <- system.file("extdata", "LASexample1.las", package="rLiDAR")

# Set the minht and above parameters
minht<-1.37  # meters or feet
above<-2.00  # meters or feet

# LiDAR metrics computation
LiDARmetrics<-LASmetrics(LASfile, minht, above)

#==========================================================================#
# Example 02: Computing Lidar metrics for multiple LAS files within a folder
#==========================================================================#
# Set folder where LAS source files reside
folder=dirname(LASfile)

# Get list of LAS files residing in the folder
LASlist <- list.files(folder, pattern="*.las", full.names=TRUE)

# Set the "minht" and "above" parameters
minht<-1.37  # meters or feet
above<-2.00  # meters or feet

# Creat an empty dataframe in whic to store the LiDAR metrics
getMetrics<-data.frame()

# Set a loop to compute the LiDAR metrics
for ( i in LASlist) {
 getMetrics<-rbind(getMetrics, LASmetrics(i, minht, above))}

# Table of the Lidar metrics
LiDARmetrics<-cbind(Files=c(basename(LASlist)), getMetrics)
head(LiDARmetrics)
```

LiDARForestStand   *3D stand visualization of LiDAR-derived individual trees*

Description

Draws a 3D scatterplot for individual trees detected from Lidar data.

Usage

```
LiDARForestStand(crownshape = c("cone", "ellipsoid", "halfellipsoid",
                 "paraboloid", "cylinder"), CL = 4, CW = 8, HCB = 10,
                   X = 0, Y = 0, dbh = 0.3, crowncolor = "forestgreen",
                  stemcolor = "chocolate4", resolution="high", mesh=TRUE)
```

Arguments

crownshape	shape of individual tree crown: "cone", "ellipsoid","halfellipsoid", "paraboloid" or "cylinder". Default is "halfellipsoid".
CL	crown length.
CW	crown diameter.
HCB	height at canopy base.
X	x-coordinate.
Y	y-coordinate.
dbh	diameter at breast height (1.73 m).
crowncolor	crown color.
stemcolor	stem color.
resolution	crown resolution: "low", "medium" and "high".
mesh	Logical, if TRUE (default) returns a tree crown mesh model, and if FALSE returns a tree crown line mode.

Value

Returns a 3-D scatterplot of the individual trees as identified automatically from the LiDAR.

Author(s)

Carlos Alberto Silva and Remko Duursma. Uses code by Remko Duursma (*Maeswrap* package,see "Plotstand").

References

http://maespa.github.io/

Examples

```
## Not run:
#=======================================================================#
# EXAMPLE 01: Plotting single trees
#=======================================================================#

# cone crown shape
library(rgl)
open3d()
LiDARForestStand(crownshape = "cone", CL = 10, CW =7,
```

```
           HCB = 5, X =0, Y = 0, dbh = 0.4, crowncolor = "forestgreen",
           stemcolor = "chocolate4", resolution="high", mesh=TRUE)

# ellipsoid crown shape
open3d()
LiDARForestStand(crownshape = "ellipsoid", CL = 10, CW =7,
           HCB = 5, X =0, Y = 0, dbh = 0.4, crowncolor = "forestgreen",
           stemcolor = "chocolate4", resolution="high", mesh=TRUE)

# halfellipsoid crown shape
open3d()
LiDARForestStand(crownshape = "halfellipsoid", CL = 10, CW =7,
           HCB = 5, X =0, Y = 0, dbh = 0.4, crowncolor = "forestgreen",
           stemcolor = "chocolate4", resolution="high", mesh=TRUE)

# paraboloid crown shape
open3d()
LiDARForestStand(crownshape = "paraboloid", CL = 10, CW =7,
           HCB = 5, X =0, Y = 0, dbh = 0.4, crowncolor = "forestgreen",
           stemcolor = "chocolate4", resolution="high", mesh=TRUE)

# cylinder crown shape
open3d()
LiDARForestStand(crownshape = "cylinder", CL = 10, CW =7,
           HCB = 5, X =0, Y = 0, dbh = 0.4, crowncolor = "forestgreen",
           stemcolor = "chocolate4", resolution="high", mesh=TRUE)

# Set the shape=FALSE
open3d()
LiDARForestStand(crownshape = "paraboloid", CL = 10, CW =7,
           HCB = 5, X =0, Y = 0, dbh = 0.4, crowncolor = "forestgreen",
           stemcolor = "chocolate4", resolution="high", mesh=FALSE)

#=======================================================================#
#EXAMPLE 02: Plotting a forest plantation stand in virtual 3-D space
#=======================================================================#

# Set the dimensions of the displayed forest stand
xlength<-30 # x length
ylength<-20 # y length

# Set the space between trees
sx<-3 # x space length
sy<-2 # y space length

# Tree location grid
XYgrid <- expand.grid(x = seq(1,xlength,sx), y = seq(1,ylength,sy))

# Get the number of trees
Ntrees<-nrow(XYgrid)

# Plot a virtual Eucalyptus forest plantation stand using the halfellipsoid tree crown shape
```

```
# Set stand trees parameters
meanHCB<-5  # mean of the height at canopy base
sdHCB<-0.1  # standard deviation of the height at canopy base
HCB<-rnorm(Ntrees, mean=meanHCB, sd=sdHCB) # height at canopy base
CL<-HCB     # tree crown height
CW<-HCB*0.6 # tree crown diameter

open3d()    # open a rgl window

# Plotting the stand
for( i in 1:Ntrees){
 LiDARForestStand(crownshape = "halfellipsoid", CL = CL[i], CW = CW[i],
             HCB = HCB[i], X = XYgrid[i,1], Y = XYgrid[i,2], dbh = 0.4,
             crowncolor = "forestgreen", stemcolor = "chocolate4",
             resolution="high", mesh=TRUE)
}

# Add other plot parameters
axes3d(c("x-", "x-", "y-", "z"), col="gray")       # axes
title3d(xlab = "X Coord", ylab = " Y Coord", zlab = "Height", col="red") # title
planes3d(0, 0, -1, 0.001, col="gray", alpha=0.7)   # set a terrain plane


# Plotting a virtual single-species forest plantation stand using "cone" tree crown shape

# Set parameters f trees growing within the virtual stand
meanHCB<-3  # mean of the height at canopy base
sdHCB<-0.1  # standard deviation of the height at canopy base
HCB<-rnorm(Ntrees, mean=meanHCB, sd=sdHCB) # height at canopy base
CL<-HCB*2.0 # tree crown height
CW<-HCB*1.3 # tree crown diameter

open3d() # open a rgl window
# Plot stand
for( i in 1:Ntrees){
 LiDARForestStand(crownshape = "cone", CL = CL[i], CW = CW[i],
             HCB = HCB[i], X = XYgrid[i,1], Y = XYgrid[i,2], dbh = 0.4,
             crowncolor = "forestgreen", stemcolor = "chocolate4",
             resolution="high", mesh=TRUE)
}

# Add other plot parameters
axes3d(c("x-", "x-", "y-", "z"), col="gray")       # axes
title3d(xlab = "X Coord", ylab = " Y Coord", zlab = "Height", col="red") # title
planes3d(0, 0, -1, 0.001, col="gray", alpha=0.7)   # set a terrain plane

#=======================================================================#
# EXAMPLE 03: Plotting a virtual mixed forest stand
#=======================================================================#

# 01. Plot different trees species in the stand with different crown shapes

# Set the number of trees
```

```
Ntrees<-80

# Set the trees locations
xcoord<-sample(1:100, Ntrees)  # x coord
ycoord<-sample(1:100, Ntrees)  # y coord

# Set a location grid of trees
XYgrid<-cbind(xcoord,ycoord)

# Plot the location of the trees
plot(XYgrid, main="Tree location")

meanHCB<-7 # mean of the height at canopy base
sdHCB<-3   # standard deviation of height at canopy base
HCB<-rnorm(Ntrees, mean=meanHCB, sd=sdHCB) # height at canopy base
crownshape<-sample(c("cone", "ellipsoid","halfellipsoid",
                  "paraboloid"), Ntrees, replace=TRUE) # tree crown shape
CL<-HCB*1.3 # tree crown height
CW<-HCB     # tree crown diameter

open3d() # open a rgl window
# Plot stand

for( i in 1:Ntrees){
 LiDARForestStand(crownshape = crownshape[i], CL = CL[i], CW = CW[i],
             HCB = HCB[i], X = as.numeric(XYgrid[i,1]), Y = as.numeric(XYgrid[i,2]),
             dbh = 0.4, crowncolor = "forestgreen", stemcolor = "chocolate4",
             resolution="high", mesh=TRUE)
}

# Add other plot parameters
axes3d(c("x-", "x-", "y-", "z"), col="gray")       # axes
title3d(xlab = "X Coord", ylab = " Y Coord", zlab = "Height", col="red") # title
planes3d(0, 0, -1, 0.001, col="gray", alpha=0.7)   # set a terrain plane


# 02. Plot different tree height in the stand using different crown colors

# Set the number of trees
Ntrees<-80

# Set the tree locations
xcoord<-sample(1:100, Ntrees) # x coord
ycoord<-sample(1:100, Ntrees) # y coord

# Set a location grid of trees
XYgrid<-cbind(xcoord,ycoord)

# plot the location of the trees
plot(XYgrid, main="Tree location")

meanHCB<-7 # mean of the height at canopy base
sdHCB<-3   # standard deviation of the height at canopy base
```

```
HCB<-rnorm(Ntrees, mean=meanHCB, sd=sdHCB) # height at canopy base
crownshape<-sample(c("cone", "ellipsoid","halfellipsoid", "paraboloid"),
                  Ntrees, replace=TRUE) # tree crown shape
CL<-HCB*1.3 # tree crown height
CW<-HCB     # tree crown diameter

# Plot tree height based on the HCB quantiles
HCBq<-quantile(HCB) # HCB quantiles
crowncolor<-NA*(1:Ntrees) # set an empty crowncolor vector

# classify trees by HCB quantile
for (i in 1:Ntrees){
 if (HCB[i] <= HCBq[2]) {crowncolor[i]<-"red"}                        # group 1
 if (HCB[i] > HCBq[2] & HCB[i] <= HCBq[3] ) {crowncolor[i]<-"blue"}   # group 2
 if (HCB[i] > HCBq[3] & HCB[i] <= HCBq[4] ) {crowncolor[i]<-"yellow"} # group 3
 if (HCB[i] >= HCBq[4]) {crowncolor[i]<-"dark green"}                 # group 4
}

open3d() # open a rgl window

# Plot stand
for(i in 1:Ntrees){
 LiDARForestStand(crownshape = crownshape[i], CL = CL[i], CW = CW[i],
   HCB = HCB[i], X = as.numeric(XYgrid[i,1]), Y = as.numeric(XYgrid[i,2]),
   dbh = 0.4, crowncolor = crowncolor[i],stemcolor = "chocolate4",
   resolution="high", mesh=TRUE)
}

# Add other plot parameters
axes3d(c("x-", "x-", "y-", "z"), col="gray")       # axes
title3d(xlab = "X Coord", ylab = " Y Coord", zlab = "Height", col="red") # title
planes3d(0, 0, -1, 0.001, col="gray", alpha=0.7)   # set a terrain plane


## End(Not run)
```

readLAS *Reading LiDAR data*

Description

This function reads and returns values associated with the LAS file format. The LAS file is a public file format for the interchange of LiDAR 3-dimensional point cloud data (American Society of Photogrammetry and Remote Sensing - ASPRS)

Usage

```
readLAS(LASfile, short=TRUE)
```

Arguments

LASfile	A standard LAS data file (ASPRS)
short	Logical, if TRUE it will return only a 5-column matrix with information on the returned point x, y, z locations, point intensity and the number of return within an individual discrete-return system laser pulse.

Value

Returns a matrix listing the information stored in the LAS file.

Author(s)

Michael Sumner and Carlos Alberto Silva.

Examples

```
#=======================================================================#
# Importing LAS file:
#=======================================================================#
LASfile <- system.file("extdata", "LASexample1.las", package="rLiDAR")

# Reading LAS file
rLAS<-readLAS(LASfile,short=TRUE)

# Summary of the LAS file
summary(rLAS)

#=======================================================================#
# LAS file visualization:
#=======================================================================#

# 01 Set a single color

col<-"forestgreen"

# plot 2D
plot(rLAS[,1],rLAS[,2], col=col,xlab="UTM.Easting", ylab="UTM.Northing", main="Single color")

# plot 3D
library(rgl)
points3d(rLAS[,1:3], col=col, axes=FALSE,xlab="", ylab="", zlab="")
axes3d(c("x+", "y-", "z-"))                 # axes
grid3d(side=c('x+', 'y-', 'z'), col="gray") # grid
title3d(xlab = "UTM.Easting", ylab = "UTM.Northing",zlab = "Height(m)", col="red") # title
planes3d(0, 0, -1, 0.001, col="gray", alpha=0.7)   # terrain


# 02 Set a color by height

# color ramp
```

```
myColorRamp <- function(colors, values) {
v <- (values - min(values))/diff(range(values))
x <- colorRamp(colors)(v)
rgb(x[,1], x[,2], x[,3], maxColorValue = 255)
}

# Color by height
col <- myColorRamp(c("blue","green","yellow","red"),rLAS[,3])

# plot 2D
plot(rLAS[,1], rLAS[,2], col=col, xlab="UTM.Easting", ylab="UTM.Northing", main="Color by height")

# plot 3D
points3d(rLAS[,1:3], col=col, axes=FALSE, xlab="", ylab="", zlab="")
axes3d(c("x+", "y-", "z-"))                     # axes
grid3d(side=c('x+', 'y-', 'z'), col="gray")     # grid
title3d(xlab = "UTM.Easting", ylab = "UTM.Northing",zlab = "Height(m)", col="red") # title
planes3d(0, 0, -1, 0.001, col="gray",alpha=0.7) # terrain
```

Index