Package 'VoxR'

February 19, 2015

Type Package

Title Metrics extraction of trees from T-LiDAR data

Version 0.5.1

Date 2013-05-28

Author Bastien Lecigne, Sylvain Delagrange and Christian Messier

Maintainer Bastien Lecigne <lecignebastien@gmail.com>

Depends R (>= 2.10)

Suggests rgl, raster, sp

Description Tools for tree crown structure description based on T-LiDAR data voxelisation

License GPL (>= 2.0)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-01-29 19:30:02

R topics documented:

VoxR-package *Metrics extraction of trees from T-LiDAR data*

Description

Tools for a geometric description of tree crown, tree growth and spatially differentiated objects recognition based on point cloud voxelisation.

Details

VoxR offers tools for 3d point cloud voxelisation (see vox), 3d point cloud projection and visualization of projections (see project, raster.proj, level and surface), geometrical description of 3d point cloud (see point.distance, axis.distance and axis.angle), tree growth analysis based on voxelised LiDAR point clouds comparison (see sub.obj) and spatially differentiated objects recognition (see obj.rec).

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

Maintainer : Bastien Lecigne <lecignebastien@gmail.com>

References

This package uses previous work from the following packages :

- sp : Roger S. Bivand, Edzer J. Pebesma, Virgilio Gomez-Rubio, 2008. Applied spatial data analysis with R. Springer, NY. http://www.asdar-book.org/
- raster : Robert J. Hijmans & Jacob van Etten (2013). raster: Geographic data analysis and modeling. R package version 2.1-25. http://CRAN.R-project.org/package=raster

These packages must be intalled to use the function raster.proj.

The examples of 3d functions are plotted using the plot3d function from the rgl package : Daniel Adler and Duncan Murdoch (2013). rgl: 3D visualization device system (OpenGL). R package version 0.93.932. http://CRAN.R-project.org/package=rgl

Examples

```
data(treecloud)

#-voxelisation using the vox function

treecloud_vox <- vox(treecloud,res=0.02)

#-visualisation

require(rgl)
library(rgl)
```

```
open3d()
plot3d(treecloud_vox,size=0.1)
```

`axis.angle` *Computing angle of points with an axis*

Description

Computing angles of points with an axis (x, y or z) and the origin of the 3d Cartesian coordinate system.

Usage

```
axis.angle(data, axis, projected, plan)
```

Arguments

`data`	a data frame containing the x, y, z, ... coordinates of a point cloud
`axis`	charecter string specifying the reference axis to compute the angles : "X", "Y" or "Z"
`projected`	logical : if TRUE the original point cloud will be projected in a 2d coordinate system
`plan`	character string specifying the plan of projection (only if projected = TRUE), see details for more information.

Details

Complementary information for the plan parameter : if axis = "X", projected should be "xz" or "xy" ; if axis = "Y", projected should be "xy" or "xz" ; if axis = "Z", projected should be "xz" or "yz".

Default : projected = FALSE.

Value

A vector containing the angle values of the points.

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

See Also

point.distance and axis.distance

Examples

```
data(treecloud_vox)

###################
#- using projection
#- computing angles

dist <- axis.angle(treecloud_vox,axis="X",projected=TRUE,plan="xy")
treecloud_vox[,4] <- dist

#- density plot

plot(density(dist,na.rm=TRUE))

#- visualisation

z <- c(sort(unique(round(treecloud_vox[,4],digits=0)),decreasing=TRUE))
col <- rainbow(n=length(z),start=0,end=2/6)
library(rgl)
open3d()
for(i in 1:length(z)){
  a <- subset(treecloud_vox,round(treecloud_vox[,4],digits=0)==z[i])
  plot3d(a,col=col[i],add=TRUE)}

####################
#- without projection
#- computing angles

dist <- axis.angle(treecloud_vox,axis="X",projected=FALSE)
treecloud_vox[,4] <- dist

#- density plot

plot(density(dist,na.rm=TRUE))

#- visualisation

z <- c(sort(unique(round(treecloud_vox[,4],digits=0)),decreasing=TRUE))
col <- rainbow(n=length(z),start=0,end=2/6)
open3d()
for(i in 1:length(z)){
  a <- subset(treecloud_vox,round(treecloud_vox[,4],digits=0)==z[i])
  plot3d(a,col=col[i],add=TRUE)
}
```

`axis.distance` *Computing distance of points from an axis*

Description

Computing distance of points with an axis (x, y or z) and the origin of the 3d Cartesian coordinate system.

Usage

```
axis.distance(data, axis)
```

Arguments

data	a data frame containing the x, y, z, ... coordinates of a point cloud
axis	character string specifying the reference axis to compute the distances : "X", "Y" or "Z"

Details

Default : proportion = FALSE

Value

A vector countaining the distance values of the points.

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

See Also

point.distance and axis.angle

Examples

```
data(treecloud_vox)

#- computing distances

dist <- axis.distance(treecloud_vox,axis="Z")
treecloud_vox[,4] <- dist

#-density plot

plot(density(dist,na.rm=TRUE))

#-visualisation

z <- c(sort(unique(round(treecloud_vox[,4],digits=2)),decreasing=TRUE))
col <- rainbow(n=length(z),start=0,end=2/6)
require(rgl)
library(rgl)
open3d()
for(i in 1:length(z)){
```

```
a <- subset(treecloud_vox,round(treecloud_vox[,4],digits=2)==z[i])
plot3d(a,col=col[i],add=TRUE)}
```

---

data1 *Cubic point cloud*

Description

Cubic point cloud used as an example for the sub.obj function complementary toward data2.

Usage

```
data(data1)
```

Format

A data frame with 9261 observations on the following 4 variables.

`X` a numeric vector

`Y` a numeric vector

`Z` a numeric vector

`NBpts` a numeric vector

Examples

```
data(data1)
require(rgl)
library(rgl)
plot3d(data1)
```

---

data2 *Cubic point cloud*

Description

Cubic point cloud used as an example for the sub.obj function complementary toward data1.

Usage

```
data(data2)
```

Format

A data frame with 7140 observations on the following 4 variables.

`X` a numeric vector

`Y` a numeric vector

`Z` a numeric vector

`NBpts` a numeric vector

Examples

```
data(data2)
require(rgl)
library(rgl)
plot3d(data2)
```

`data_part` *Geometrics objects spatially differentiated*

Description

Point cloud presenting spatially differentiated geometric objects used as an example for the obj.rec function.

Usage

```
data(data_part)
```

Format

A data frame with 2382 observations on the following 4 variables.

`X` a numeric vector

`Y` a numeric vector

`Z` a numeric vector

`NBpts` a numeric vector

Examples

```
data(data_part)
require(rgl)
library(rgl)
plot3d(data_part)
```

`level` *Density levels definition*

Description

This function creates density levels from point clouds projected using the function project. It can be used as a sub function of the raster.proj function to discriminate voxels, point or ratio npts/nvox density.

Usage

```
level(datas, by, levels)
```

Arguments

`datas`	a vector containing the values of a variable
`by`	character string specifying which kind of discrimination will be used to define the density levels : "quantiles" or "percents".
`levels`	a vector containing the values of discretisation (see details for more information).

Details

Details on levels parameter : if by = "quantiles" indicate the proportion of the variable contaned in each level (if level=0.25 the quantiles will be returned, if level=0.2 the quintiles will be returned). If by = "percents" the levels are defined as a percentage of the varialable (if level=c(0.25,0.5,0.75) the discretisation values will be respectively 25, 50, 75 and 100% of the maximum value of the variable).

Defaults : by = "quantiles" and if by = "quantiles", level = 0.25 ; if by = "percents", levels = c(25, 50, 75)

Value

A vector containing the values of the discretisation levels

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

See Also

surface

Examples

```
#- projection

data(treecloud_vox)
proj <- project(treecloud_vox,dim="xy")
nvox <- c(proj[,3])
npts <- c(proj[,4])
ratio <- c(proj[,5])

#- computing discretisation level

  #- number of voxels by quantiles
lev_vox <- level(nvox,by="quantiles",levels=c(0.2))
  #-number of points by percents
lev_pts <- level(npts,by="percents",levels=c(0.2,0.4,0.6,0.8))
  #-ration npts/nvox by quantiles
lev_ratio <- level(ratio,by="quantiles",levels=c(0.25))

#- to see the levels

lev_vox
lev_pts
lev_ratio
```

---

obj.rec *Spatially differentiated objects recognition*

Description

Recognition of spatially differentiated objects within a point cloud. Two points located under within distance of reaserch from each other are considered as the parts of a unique object.

Usage

```
obj.rec(data, fac)
```

Arguments

data	a data frame containing the x, y, z, ... coordinates of a voxel cloud
fac	numeric specifying the distance of research (in the scale of the original coordinate system)

Value

A data frame containing the x, y, z coordinates and object ID of the input data

Note

This function can be time consuming if used on big data sets

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

See Also

sub.obj

Examples

```
data(data_part)

#- voxelisation
data_vox <- vox(data_part,res=1)

#- objects reconuition

datasep <- obj.rec(data_vox,fac=2)


#-visualisation

require(rgl)
library(rgl)
open3d()
b <- subset(datasep,datasep[,4]==1)
plot3d(b,col="green",add=TRUE)
c <- subset(datasep,datasep[,4]==2)
plot3d(c,col="red",add=TRUE)
e <- subset(datasep,datasep[,4]==3)
plot3d(e,col="blue",add=TRUE)
f <- subset(datasep,datasep[,4]==4)
plot3d(f,col="purple",add=TRUE)
```

`point.distance` *Computing distance of points from a unique point*

Description

Computing distance of points from a unique point within a point cloud.

Usage

```
point.distance(data, point)
```

Arguments

`data`	a data frame containing the x, y, z, ... coordinates of a point cloud
`point`	a vector containing the x, y, z coordinates of the reference point

Value

A vector containing the distance values of the points

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

See Also

axis.angle and axis.distance

Examples

```
data(treecloud_vox)

#- computing distances

dist <- point.distance(treecloud_vox,point=c(0,0,0))
treecloud_vox[,4] <- dist

#-density plot

plot(density(dist,na.rm=TRUE))

#-3d visualisation

z <- c(sort(unique(round(treecloud_vox[,4],digits=2)),decreasing=TRUE))
col <- rainbow(n=length(z),start=0,end=2/6)
require(rgl)
library(rgl)
open3d()
for(i in 1:length(z)){
   a <- subset(treecloud_vox,round(treecloud_vox[,4],digits=2)==z[i])
   plot3d(a,col=col[i],add=TRUE)
 }
```

---

project *Projection of voxels within a plan*

Description

Projection of voxels within a 2d coordinate system formed by two axes of the original coordinate system.

Usage

```
project(data, dim)
```

Arguments

data	a data frame containing the x, y, z, ... coordinates of a point cloud
dim	a character string specifying the projection plan : "xy", "xz" or "yz"

Details

Default : dim = "xy"

Value

A data frame of a 2D point cloud containing : x, y coordinates of the pixels and the number of voxels (nvox), number of points (npts), ratio npts/nvox contained in each pixel.

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

Examples

```
data(treecloud_vox)

#-projection in "xy" plan

proj <- project(treecloud_vox,dim="xy")

#-creating vectors with interests variables

npts <- c(proj[,4])
nvox <- c(proj[,3])
ratio <- c(proj[,5])

#-discretisation level

 # by quantiles (20% quantiles or 25% quantiles)
lev_vox <- level(nvox,by="quantiles",levels=c(0.2))
lev_ratio <- level(ratio,by="quantiles",levels=c(0.25))
# by percents
lev_pts <- level(npts,by="percents",levels=c(0.2,0.4,0.6,0.8))


#- computing surfaces

# surface in proportion
surf_nvox <- surface(proj,method="nvox",levels=lev_vox,res=0.02,proportion=TRUE)
# surface in absolute
surf_ratio <- surface(proj,method="ratio",levels=lev_ratio,res=1,proportion=FALSE)
# surface in absolute
surf_npts <- surface(proj,method="npts",levels=lev_pts,res=0.02,proportion=FALSE)

#- ploting raster images
```

```
par(mfrow=c(1,1),mai=c(1,0,1,0),omi=c(0,1,0,0))

raster.proj(proj,title="nvox",res=0.02,method="nvox",levels=lev_vox,
colors=c("lightblue","green","yellow","red","purple"),
            contour=TRUE,surf=surf_nvox,dim="xy")

raster.proj(proj,title="npts",res=0.02,method="npts",levels=lev_pts,
colors=c("lightblue","green","yellow","red","purple"),
            contour=TRUE,surf=surf_npts,dim="xy")

raster.proj(proj,title="ratio",res=0.02,method="ratio",levels=lev_ratio,
colors=c("lightblue","green","yellow","red"),
            contour=TRUE,surf=surf_ratio,dim="xy")

################################################################################
#>>>>> Easy way : define level using the 99 percentile help to stabilize the results

npts_perc <- quantile(npts,probs=seq(0,1,0.01))[100]
npts_norm_perc  <- subset(npts,npts<npts_perc)
lev_perc <- level(npts_norm_perc,by="quantiles",levels=c(0.2))
lev_perc[5] <- lev_pts[5]
surf_npts_perc <- surface(proj,method="npts",levels=lev_perc,res=0.02,proportion=FALSE)

raster.proj(proj,title="npts 99perc",res=0.02,method="npts",levels=lev_perc,
colors=c("lightblue","green","yellow","red","purple"),
contour=TRUE,surf=surf_npts_perc,dim="xy")
```

`raster.proj` *Create a raster image from project, level and surface*

Description

Easy tool for raster image creation from projections created by the project function using level function for density discretisation with the possibility of integrating an automatic caption.

Usage

```
raster.proj(data, title, res, method, levels, colors, contour, classlegend, surf, dim)
```

Arguments

`data`	2d point cloud x, y, npts, nvox, npts/nvox
`title`	the raster's title
`res`	pixel resolution in the scale of the original orthonormed system
`method`	a character string specifying the variable of interest from the projection ("nvox", "npts" ou "ratio")
`levels`	a vector containing the density levels of discretisation as created by level
`colors`	a vector containing the colors of the density classes

`contour`	logical : if TRUE the contours of the density classes are drawn
`classlegend`	a vector of characters string containing the names of the density classes for the caption
`surf`	a vector containing the surface of each density class for the caption as created by the surface function
`dim`	a character string specifying the dimension of projection ("xy", "yz", "xz")

Details

Defaults :

- method = "nvox"
- res = 1
- levels = quantiles 0.25
- title = ""
- colors = c(grey, green, yellow, red)
- contour = TRUE
- classlegend = ""
- surf = ""
- dim = "xy"

Value

Draw the raster image in the default R graphic device

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

References

This function use previous work from the following packages :

- sp : Roger S. Bivand, Edzer J. Pebesma, Virgilio Gomez-Rubio, 2008. Applied spatial data analysis with R. Springer, NY. http://www.asdar-book.org/
- raster : Robert J. Hijmans & Jacob van Etten (2013). raster: Geographic data analysis and modeling. R package version 2.1-25. http://CRAN.R-project.org/package=raster

Examples

```
data(treecloud_vox)

#- projection in xy plan

proj <- project(treecloud_vox,dim="xy")

#-computing parameters for levels and surf
```

```
nvox <- c(proj[,3])
lev_vox <- level(nvox,by="quantiles",levels=c(0.2))
surf_nvox <- surface(proj,method="nvox",levels=lev_vox,res=0.02,proportion=TRUE)

#- drawing raster

par(mfrow=c(1,1),mai=c(1,0,1,0),omi=c(0,1,0,0))

raster.proj(proj,title="nvox",res=0.02,method="nvox",levels=lev_vox,
colors=c("lightblue","green","yellow","red","purple"),
contour=TRUE,surf=surf_nvox,dim="xy")
```

`sub.obj` *Substraction of two voxel coulds*

Description

Isolation of unique voxels to one of two voxel clouds based on the detection of isolated voxels within a voxels of bigger dimmensions.

Usage

```
sub.obj(data1, data2, res, nvox.reaserch)
```

Arguments

`data1`	a data frame containing the x, y, z, ... coordinates of a point cloud from which "data2" will be substracted
`data2`	a data frame containing the x, y, z, ... coordinates of a point cloud to substract to "data1"
`res`	numeric specification of the resolution of the voxels (in the scale of the original coordinate system)
`nvox.reaserch`	numeric specifying as a factor the resolution of the research's voxel (nvox.reasearch * res)

Value

A data frame containing the x, y, z coordinates of the voxels unique to data1

Note

This function can be time consuming if used on big data sets

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

Examples

```
#- importing 2 data dets
data(data1)
data(data2)

#- substraction of data2 to data1

sub <- sub.obj(data1,data2,res=0.5,nvox.reaserch=1)

#-visualisation

require(rgl)
library(rgl)
open3d()
plot3d(sub,col="red",add=TRUE)
plot3d(data2,add=TRUE)
```

`surface` *Compute the surface of density classes of a projection*

Description

Compute the surface of density classes within a projection created by the project function and discretised by level. The surface is calculated using the following relationship : leveln<=surface<leveln+1

Usage

```
surface(data, method, levels, res, proportion)
```

Arguments

`data`	a data frame of a 2D point cloud containing : x, y coordinates of the pixels and the number of voxels (nvox), number of points (npts), ratio npts/nvox contained in each pixel
`method`	character string specifying the variable of interest of the projection : "nvox", "npts" or "ratio"
`levels`	a vector containing the discretisation levels as created by the level function
`res`	numeric definition of the resolution of the pixels. If res = 1, the calculated surface corresponds to the number of pixels.
`proportion`	logical : if TRUE the surfaces are expressed in proportion to the total surface

Details

Defaults :

- levels : quantiles 0.25
- method = "nvox"
- res = 1
- proportion = FALSE

Value

A vector containing the surface of each density class

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

Examples

```
#- projection

data(treecloud_vox)
proj <- project(treecloud_vox,dim="xy")

#- creating vectors with interests variables

npts <- c(proj[,4])
nvox <- c(proj[,3])
ratio <- c(proj[,5])

#-creating level of discretisation

lev_vox <- level(nvox,by="quantiles",levels=c(0.2))
lev_pts <- level(npts,by="percents",levels=c(0.2,0.4,0.6,0.8))
lev_ratio <- c(1,2)# <- level(ratio,by="quantiles",level=c(0.25))

#- computing surfaces

surf_nvox <- surface(proj,method="nvox",levels=lev_vox,res=0.02,proportion=TRUE)
surf_npts <- surface(proj,method="npts",levels=lev_pts,res=0.02,proportion=FALSE)
surf_ratio <- surface(proj,method="ratio",levels=lev_ratio,res=1,proportion=FALSE)
surf_nvox
surf_npts
surf_ratio
```

---

treecloud *LiDAR scene of a tree*

Description

LiDAR scene of a tree digitized using an Ilris 3d (opech) T-LiDAR.

Usage

```
data(treecloud)
```

Format

A data frame with 680710 observations on the following 3 variables.

`X` a numeric vector

`Y` a numeric vector

`Z` a numeric vector

Examples

```
data(treecloud)
require(rgl)
library(rgl)
plot3d(treecloud)
```

`treecloud_vox` *treecloud voxelised*

Description

treecloud LiDAR scene voxelised using vox function with a 0.02m resolution

Usage

```
data(treecloud_vox)
```

Format

A data frame with 373939 observations on the following 4 variables.

`X` a numeric vector

`Y` a numeric vector

`Z` a numeric vector

`NBpts` a numeric vector

Examples

```
data(treecloud_vox)
require(rgl)
library(rgl)
plot3d(treecloud_vox)
```

vox *Voxelisation*

Description

Voxelisation algorythm of 3d point cloud recording the number of points within each voxels

Usage

```
vox(data, res)
```

Arguments

data	a data frame containing the x, y, z, ... coordinates of a point cloud
res	numeric specification of the voxels resolution in the scale of the original coordinate system

Details

Default : res = 1

Value

A data frame containing the x, y, z coordinates and the number of points within each voxel of a voxel cloud.

Author(s)

Bastien Lecigne, Sylvain Delagrange and Christian Messier

Examples

```
#-import data

data(treecloud)

#-voxelisation

treecloud_vox <- vox(treecloud,res=0.02)

#-visualisation

require(rgl)
library(rgl)
open3d()
plot3d(treecloud_vox,size=0.1)
```

Index