

# Package ‘VoxR’

January 20, 2025

**Title** Trees Geometry and Morphology from Unstructured TLS Data

**Version** 1.0.0

**Description** Tools for 3D point cloud voxelisation, projection, geometrical and morphological description of trees (DBH, height, volume, crown diameter), analyses of temporal changes between different measurement times, distance based clustering and visualisation of 3D voxel clouds and 2D projection. Most analyses and algorithms provided in the package are based on the concept of space exploration and are described in Lecigne et al. (2018, <[doi:10.1093/aob/mcx095](https://doi.org/10.1093/aob/mcx095)>).

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** data.table, FNN, Rfast, circular, dplyr, fastcluster, geometry, raster, rgl, grDevices

**URL** <https://github.com/Blecigne/VoxR>

**BugReports** <https://github.com/Blecigne/VoxR/issues>

**RoxygenNote** 7.1.0

**NeedsCompilation** no

**Author** Bastien Lecigne [aut, cre] (<<https://orcid.org/0000-0002-1496-202X>>)

**Maintainer** Bastien Lecigne <[lecignebastien@gmail.com](mailto:lecignebastien@gmail.com)>

**Repository** CRAN

**Date/Publication** 2020-09-30 08:50:08 UTC

## Contents

axis.angle . . . . .	2
axis.distance . . . . .	3
axis_angle . . . . .	3
axis_distance . . . . .	4
box_counting . . . . .	5
distance_clustering . . . . .	6

filled_voxel_cloud . . . . .	8
filter_noise . . . . .	9
filter_point_density . . . . .	10
level . . . . .	11
obj.rec . . . . .	11
plot_projection . . . . .	12
plot_voxels . . . . .	13
plot_voxels_full_grid . . . . .	14
point.distance . . . . .	15
point_distance . . . . .	15
project . . . . .	16
project_voxels . . . . .	16
raster.proj . . . . .	17
sub.obj . . . . .	18
subtract_point_clouds . . . . .	18
surface . . . . .	19
tree_metrics . . . . .	20
vox . . . . .	21
<b>Index</b>	<b>23</b>

---

axis.angle	<i>Deprecated function</i>
------------	----------------------------

---

### Description

Deprecated function

### Usage

```
axis.angle(...)
```

### Arguments

... parameters

---

axis.distance	<i>Deprecated function</i>
---------------	----------------------------

---

**Description**

Deprecated function

**Usage**

```
axis.distance(...)
```

**Arguments**

... parameters

---

axis_angle	<i>Computes points angle with an axis (X, Y or Z) and the origin of the 3D cartesian coordinates system.</i>
------------	--

---

**Description**

Computes points angle with an axis (X, Y or Z) and the origin of the 3D cartesian coordinates system.

**Usage**

```
axis_angle(data, axis, project, message)
```

**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud or voxel cloud.
axis	character. Specifying the reference axis to compute the angles: "X", "Y" or "Z".
project	character. If specified the point cloud is projected into a 2D plan before computing the angles. Can be "xy", "yz" or "xz". Default is without projection.
message	logical. If FALSE, messages are disabled. Default = TRUE. @references Lecigne, B., Delagrang, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. <i>Annals of botany</i> , 121(4), 589-601.

**Value**

A vector containing the angle values of the points.

**Examples**

```

#- import t1s data
t1s=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- compute angle with the Z axis
t1s[,angle_z:=VoxR::axis_angle(t1s,axis = "Z")]

#- compute angle with the X axis with projection in the xy plan
t1s[,angle_x:=VoxR::axis_angle(t1s,axis = "X",project = "xy")]

#- round angle values for visualization
t1s[,angle_z:=round(angle_z)]
t1s[,angle_x:=round(angle_x)]

#- plot the angle with Z axis
cols=rev(rainbow(max(t1s$angle_z)+1,end=4/6)) # color scale
rgl::open3d()
rgl::plot3d(t1s,col=cols[t1s$angle_z+1],add=TRUE)

#- plot the angle with X axis
cols=rev(rainbow(max(t1s$angle_x)+1,end=4/6)) # color scale
rgl::open3d()
rgl::plot3d(t1s,col=cols[t1s$angle_x+1],add=TRUE)

```

---

axis_distance	<i>Computes points distance to an axis of the cartesian coordinates system.</i>
---------------	---

---

**Description**

Computes points distance to an axis of the cartesian coordinates system.

**Usage**

```
axis_distance(data, axis, message)
```

**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud or voxel cloud.
axis	character. Specifying the reference axis to compute the distance: "X", "Y" or "Z".
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Value**

A vector containing the distance values of the points

## References

Lecigne, B., Delagrangé, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

## Examples

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- compute distance to the Z axis
tls[,dist:=VoxR::axis_distance(tls,"Z")]

#- round distance values for visualization
tls[,dist:=round(dist*100)]

#- plot the distance to the Z axis
cols=rev(rainbow(max(tls$dist)+1,end=4/6)) # color scale
rgl::open3d()
rgl::plot3d(tls,col=cols[tls$dist+1],add=TRUE)
```

---

box\_counting

*Computes fractal dimension using the box counting method.*

---

## Description

Computes fractal dimension using the box counting method.

## Usage

```
box_counting(data, min_vox_size, store_fit, message)
```

## Arguments

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud.
min_vox_size	numeric. The minimum size of a voxel. Default = 0.01.
store_fit	logical. If TRUE, the parameters linear model's fit are returned. Default = FALSE.
message	logical. If FALSE, messages are disabled. Default = TRUE.

## Value

If `store_fit = FALSE` only the fractal dimension is returned. If `store_fit = TRUE` the parameters of the linear model used to estimate the fractal dimension and a table containing the number of boxes (i.e. voxels) at any resolution are returned in a list in addition to the fractal dimension.

## References

Seidel, D. (2018). A holistic approach to determine tree structural complexity based on laser scanning data and fractal analysis. *Ecology and evolution*, 8(1), 128-134.

## Examples

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- box counting
FD = VoxR::box_counting(tls,store_fit = TRUE)

#- fractal dimension
FD$fractal_dim
#- linear model fit
FD$fit_summary
#- plot fit
plot(log(FD$fit_table$N)~log(1/FD$fit_table$res))
abline(FD$fit_summary)
```

---

distance\_clustering     *Clustering of non connected objects in a point cloud.*

---

## Description

Clustering objects with non common points: two points located within a user defined distance from each other are considered as the parts of a unique object. This function is well suited to be applied to the outputs of the [subtract\\_point\\_clouds](#) function.

Clustering objects with non common points: two points located within a user defined distance from each other are considered as the parts of a unique object. This function is well suited to be applied to the outputs of the [subtract\\_point\\_clouds](#) function.

## Usage

```
distance_clustering(data, d_clust, method, C_size, message)
```

```
distance_clustering(data, d_clust, method, C_size, message)
```

## Arguments

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud or voxel cloud.
d_clust	numeric. The distance required to consider two points as being part of two different clusters. Default = 0.02.
method	character. The algorithm to use for clustering. Can be either "D_mat" or "Iter", see details. Default = "D_mat".

C_size	(optional) numeric. If method = "Iter", sets the maximal size of a cluster (in distance unit).
message	logical. If FALSE, messages are disabled. Default = TRUE.

### Details

If method == "D\_mat" the clustering process is based on building a matrix distance. This is time efficient but use a lot of memory. If method == "Iter" a slower but memory efficient iterative process is used. In some cases, D\_clust can help to speed up the process.

### Value

The input data with an additionnal field containing the cluster ID.

The input data with an additionnal field containing the cluster ID.

### References

Lecigne, B., Delagrangé, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

### Examples

```
#- import datasets
t0=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))
t1=data.table::fread(system.file("extdata", "Tree_t1.asc", package="VoxR"))

#- keep only the tree crown
t0 = t0[z>=0,]
t1 = t1[z>=0,]

#- subtract t0 to t1 with the hull method
diff = VoxR::subtract_point_clouds(t0 = t0,t1 = t1, method = "hull")

#- clustering the difference between t0 and t1
clust = VoxR::distance_clustering(diff,d_clust = 0.03)

#- plot the result (NOTE that colors are redundant)
rgl::open3d()
rgl::plot3d(clust,col=clust$cluster,add=TRUE)
#- import datasets
t0=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))
t1=data.table::fread(system.file("extdata", "Tree_t1.asc", package="VoxR"))

#- keep only the tree crown
t0 = t0[z>=0,]
t1 = t1[z>=0,]

#- subtract t0 to t1 with the hull method
diff = VoxR::subtract_point_clouds(t0 = t0,t1 = t1, method = "hull")
```

```

#- clustering the difference between t0 and t1 with the matrix distance based method
clust = VoxR::distance_clustering(diff,d_clust = 0.03)

#- plot the result (NOTE that colors are redundant)
rgl::open3d()
rgl::plot3d(clust,col=clust$cluster,add=TRUE)

#- clustering the difference between t0 and t1 with the iterative method
clust = VoxR::distance_clustering(diff,d_clust = 0.03,method = "Iter")

#- plot the result (NOTE that colors are redundant)
rgl::open3d()
rgl::plot3d(clust,col=clust$cluster,add=TRUE)

#- clustering the difference between t0 and t1 with the iterative method with maximum object size
clust = VoxR::distance_clustering(diff,d_clust = 0.03,method = "Iter",C_size = 1)

#- plot the result (NOTE that colors are redundant)
rgl::open3d()
rgl::plot3d(clust,col=clust$cluster,add=TRUE)

```

---

filled\_voxel\_cloud      *Produces a filled voxel cloud.*

---

## Description

This function produces a filled voxel cloud of a tree, i.e. a voxels cloud within which empty objects (e.g. trunk and large branches) are filled. The algorithm was inspired from the one described by Vonderach et al. (2012) with some modifications. First, the point cloud is voxelized with a given (*res*) voxel resolution. The voxel cloud is then sliced into one voxel tick layers. Within a single layer different objects are then clustered based on their distance to each other (see the [distance\\_clustering](#) function for more details). Each cluster is then filled by adding voxels along the range of *Y* for each *X* value of the cluster and reversly along the range of *X* for each *Y* of the cluster. All unique voxels are then returned.

## Usage

```
filled_voxel_cloud(data, res, d_clust, estimate_volume, message)
```

## Arguments

<i>data</i>	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud.
<i>res</i>	numeric. Resolution of a voxel.
<i>d_clust</i>	numeric. The distance to use for clustering, see the <a href="#">distance_clustering</a> for more details.
<i>estimate_volume</i>	logical. If TRUE the tree volume is computed as done in Vonderach et al. (2012).
<i>message</i>	logical. If FALSE, messages are disabled. Default = TRUE.



**Value**

If `estimate_volume = FALSE` a `data.frame` or `data.table` containing the voxels coordinates is returned. If `estimate_volume = TRUE` a list containing the voxels coordinates and the estimated tree volume is returned.

**References**

Vonderach, C., Voegtle, T., & Adler, P. (2012). Voxel-based approach for estimating urban tree volume from terrestrial laser scanning data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 39, 451-456.

**Examples**

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t1.asc", package="VoxR"))

#- keep the tree trunk
tls=tls[z<=0]

#- run filled voxel voxelisation
filled = VoxR::filled_voxel_cloud(tls,0.02)

#- run usual voxelisation
voxels = VoxR::vox(tls,0.02)

#- compare filled voxel cloud to empty voxel cloud
VoxR::plot_voxels(filled,res = 0.02)
VoxR::plot_voxels(voxels,res = 0.02)

#- compare the volume estimate from Vonderach et al. 2012 to estimate based on voxel volume
#- run filled voxel voxelisation with volume estimation
filled = VoxR::filled_voxel_cloud(tls,0.01,estimate_volume = TRUE)

#- compare volumes
filled$estimated_volume # Vonderach
nrow(filled$filled_voxels)*0.01^3 # voxel based
```

---

 filter\_noise

*Statistical filtering of a point cloud.*


---

**Description**

Implements the Statistical Outliers Removal (SOR) filter available in [CloudCompare](#). Computes the distance of each point to its `k` nearest neighbours and considers a point as noise if it is further than the average distance (for the entire point cloud) plus `sigma` times the standard deviation away from other points.

**Usage**

```
filter_noise(data, k, sigma, store_noise, message)
```

**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud.
k	numeric. The number of nearest neighbours to use. Default = 5.
sigma	numeric. The multiplier of standard deviation to consider a point as noise. Default = 1.5.
store_noise	logical. Should the noisy points be retained ? Default = FALSE.
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Value**

If store\_noise = TRUE the input data is returned with an additional field ("Noise") where points that are classified as noise points are labeled with 2 and the points not classified as noise are labeled as 1. If store\_noise = FALSE only the points that were not classified as noise are returned.

**Examples**

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- run noise filter
clean=VoxR::filter_noise(tls,store_noise = TRUE)

#- plot the result (noise in red)
rgl::open3d()
rgl::plot3d(clean,col=clean$Noise,add=TRUE)
```

---

filter\_point\_density *Retains one point of the original point cloud within a voxel of given size.*

---

**Description**

Retains one point of the original point cloud within a voxel of given size.

**Usage**

```
filter_point_density(data, res, message)
```

**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud.
res	numeric. The voxel resolution.
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Value**

a data.frame or data.table with reduced point sensivity.

**Examples**

```

#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- keep one point in 2cm voxels
filtered=VoxR::filter_point_density(tls,0.02)
rgl::open3d()
rgl::plot3d(filtered,add=TRUE)

#- keep one point in 10cm voxels
filtered=VoxR::filter_point_density(tls,0.1)
rgl::open3d()
rgl::plot3d(filtered,add=TRUE)

```

---

level	<i>Deprecated function</i>
-------	----------------------------

---

**Description**

Deprecated function

**Usage**

```
level(...)
```

**Arguments**

... parameters

---

obj.rec	<i>Deprecated function</i>
---------	----------------------------

---

**Description**

Deprecated function

**Usage**

```
obj.rec(...)
```

**Arguments**

... parameters

---

plot_projection	<i>Visualization of a projected voxel cloud.</i>
-----------------	--

---

### Description

Visualization of a projected voxel cloud.

### Usage

```
plot_projection(data, var, th, palette)
```

### Arguments

data	a data.frame or data.table containing the output of the <a href="#">project_voxels</a> function: x, y, number of voxels, number of points and ratio of a projected voxel cloud.
var	character. The variable to plot: "nvox" for the number of voxels per pixel, "npts" for the number of points or "ratio" for the ratio npts/nvox. Default is "nvox".
th	numeric between 0 and 1. A quantile threshold that defines the maximum value of var to be plotted. Values > th are replaced by the value of th. Disabled by default.
palette	a color palette to use for plotting.

### References

Lecigne, B., Delagrangé, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

### Examples

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- voxelisation
voxels = VoxR::vox(tls,0.05)

#- project into the xy plan
project = VoxR::project_voxels(voxels,"xy")

#- plot the number of voxels
VoxR::plot_projection(project,var = "nvox")

#- plot the number of points
VoxR::plot_projection(project,var = "npts")

#- plot the ratio npts/nvox
VoxR::plot_projection(project,var = "ratio")
```

```
#- plot the number of voxels with different color palette
VoxR::plot_projection(project,palette = terrain.colors)

#- plot the number of voxels with a 95% percentile threshold
VoxR::plot_projection(project,th = 0.95)
```

---

plot\_voxels

*Voxel cloud visualization.*


---

## Description

Voxel cloud visualization.

## Usage

```
plot_voxels(data, res, type, lcol, fcol, lwd, alpha, plot, message)
```

## Arguments

data	a data.frame or data.table containing at least the voxel cloud x, y, z coordinates.
res	numeric. The voxel resolution. If not provided, the function will guess it.
type	character. How to represent a voxel ? If "w" only the voxel hedges are plotted, if "p" plain voxels are plotted, if "b" both hedges and plain voxels are plotted. Default = "b".
lcol	the line color for type = "w" or "b".
fcol	the facets color for type = "p" or "b".
lwd	numeric. The line width for type = "w" or "b".
alpha	numeric. The transparency of the voxel faces for type = "p" or "b".
plot	logical. Plot the voxels ? See return for mesh capture. Default = TRUE.
message	logical. Removes the message from the resolution guessing. Default = FALSE.

## Value

If plot = TRUE, the 3D plot of voxels is plotted. At anytime the mesh object that enables to plot the voxels can be captured and to be plotted using the [shade3d](#) function from rgl. The returned object is a list containing the 3D mesh of the voxel cloud and all additionnal fields of the input data.

## Examples

```
#- import t1s data
t1s=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- voxelisation
voxels = VoxR::vox(t1s,res=0.05)

#- plot the voxels
```

```

VoxR::plot_voxels(voxels)

#- capture the voxels mesh to plot with color scale
###- number of points in the voxel
voxels_mesh = VoxR::plot_voxels(voxels,plot = FALSE) # capture the mesh
colors=rev(rainbow(max(voxels_mesh$additionnal$npts),end=4/6)) # color scale
rgl::open3d()
rgl::shade3d(voxels_mesh$mesh,col=colors[round(voxels_mesh$additionnal$npts)]
             ,lit=FALSE,alpha=0.5) # plot

###- distance from the crow center
# compute distnce
voxels[,distance:=round(VoxR::point_distance(voxels[,1:3],c(mean(x),mean(y),mean(z)))*100)]
voxels_mesh = VoxR::plot_voxels(voxels,plot = FALSE) # capture mesh
cols=rev(rainbow(max(voxels_mesh$additionnal$distance),end=4/6)) # color scale
rgl::open3d()
rgl::shade3d(voxels_mesh$mesh,col=cols[round(voxels_mesh$additionnal$distance)]
             ,lit=FALSE,alpha=0.5) # plot

```

---

plot\_voxels\_full\_grid *Voxel cloud visualization when voxel cloud includes the empty voxels.*

---

## Description

Voxel cloud visualization when voxel cloud includes the empty voxels. Filled voxels are plotted as plain vertices and only the edges of empty voxels are plotted.

## Usage

```
plot_voxels_full_grid(data, res, ecol, fcol, lwd, alpha, plot, message)
```

## Arguments

data	a data.frame or data.table containing at least the voxel cloud x, y, z coordinates.
res	numeric. The voxel resolution. If not provided, the function will guess it.
ecol	color for the edges of empty voxels.
fcol	color for the facets of filled voxels.
lwd	numeric. The line width for the edges of empty voxels.
alpha	numeric. The transparency of the voxel facets for filled voxels.
plot	logical. Plot the voxels ? See return for mesh capture. Default = TRUE.
message	logical. If FALSE removes the message from the resolution guessing.

## Value

At anytime the mesh object that enables to plot the voxels can be captured to plot it using the [shade3d](#) function from rgl. The returned object is a list containing the 3D mesh of filled and empty voxels separately.

**Examples**

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- voxelisation with full.grid option
voxels=VoxR::vox(tls,0.3,full.grid = TRUE)

#- plot the voxels
VoxR::plot_voxels_full_grid(voxels)
```

---

point.distance	<i>Deprecated function</i>
----------------	----------------------------

---

**Description**

Deprecated function

**Usage**

```
point.distance(...)
```

**Arguments**

... parameters

---

point_distance	<i>Computes the distance of a set of points to a user defined point.</i>
----------------	--

---

**Description**

Computes the distance of a set of points to a user defined point.

**Usage**

```
point_distance(data, point, message)
```

**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud or voxel cloud.
point	a vector of length 3 containing the x, y and z coordintes of the reference point.
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Value**

A vector containing the distance values of the points.

**References**

Lecigne, B., Delagrange, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

**Examples**

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- compute distance to the crown centre
tls[,dist:=VoxR::point_distance(tls,c(mean(x),mean(y),mean(z)))]

#- round distance values for visualization
tls[,dist:=round(dist*100)]

#- plot the distance to crown centre
cols=rev(rainbow(max(tls$dist)+1,end=4/6)) # color scale
rgl::open3d()
rgl::plot3d(tls,col=cols[tls$dist+1],add=TRUE)
```

---

project

*Deprecated function*

---

**Description**

Deprecated function

**Usage**

```
project(...)
```

**Arguments**

...                    parameters

---

project\_voxels

*Project a voxel cloud in a 2D plan formed by two axes of the cartesian coordiantes system.*

---

**Description**

Project a voxel cloud in a 2D plan formed by two axes of the cartesian coordiantes system.

**Usage**

```
project_voxels(data, plan, message)
```



**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a voxel cloud.
plan	character. Defines the projection plan: "xy", "xz" or "yz". Default = "xy".
message	logical. If FALSE, messages are disabled. Default = TRUE.

**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.

**References**

Lecigne, B., Delagrangé, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

**Examples**

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- voxelisation
voxels = VoxR::vox(tls,0.05)

#- project into the xy plan
project_xy = VoxR::project_voxels(voxels,"xy")
VoxR::plot_projection(project_xy) # plot projection

#- project into the xz plan
project_xy = VoxR::project_voxels(voxels,"xz")
VoxR::plot_projection(project_xy) # plot projection
```

---

raster.proj

*Deprecated function*


---

**Description**

Deprecated function

**Usage**

```
raster.proj(...)
```

**Arguments**

... parameters

---

sub.obj *Deprecated function*

---

### Description

Deprecated function

### Usage

sub.obj(...)

### Arguments

... parameters

---

subtract\_point\_clouds

*Point clouds subtraction: identification of changes between two measuring times.*

---

### Description

Identify the point that are unique to one of two point clouds to detect the changes that occurred between two measuring times (e.g. growth, branches losses, branch motion). Two methods are available (see details).

### Usage

subtract\_point\_clouds(t0, t1, method, dist, message)

### Arguments

t0	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud or voxel cloud acquired at time 0.
t1	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud or voxel cloud acquired at time 1.
method	character. The method to use to identify the difference between t0 and t1. Can be either "hull" or "distance", see details.
dist	numeric. The threshold distance to consider a point is unique to t1 if method = "distance".
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Details**

If `method = "hull"`, the convex hull that wraps `t0` is constructed and the difference between `t1` and `t0` are the points outside the convex hull. If `method = "distance"`, the points in `t1` that are distant (i.e. further than `dist`) from the points in `t0` are returned.

**Value**

a `data.frame` or `data.table` containing the `x`, `y`, `z`, ... coordinates of points that are unique to `t1`.

**Note**

`t0` and `t1` must be registered in the same coordinates system.

@references Lecigne, B., Delagrangé, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

**Examples**

```
#- import datasets
t0=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))
t1=data.table::fread(system.file("extdata", "Tree_t1.asc", package="VoxR"))

#- keep only the tree crown
t0 = t0[z>=0,]
t1 = t1[z>=0,]

####- substract t0 to t1 with the hull method
diff = VoxR::substract_point_clouds(t0 = t0,t1 = t1, method = "hull")
#- plot the result (t0 in black, the difference between t1 and t0 in red)
rgl::open3d()
rgl::plot3d(t0,add=TRUE)
rgl::plot3d(diff,col="red",add=TRUE)

####- substract t0 to t1 with the distance based method
diff = substract_point_clouds(t0 = t0,t1 = t1, method = "distance",dist = 0.1)
#- plot the result (t0 in black, the difference between t1 and t0 in red)
rgl::open3d()
rgl::plot3d(t0,add=TRUE)
rgl::plot3d(diff,col="red",add=TRUE)
```

---

surface

*Deprecated function*

---

**Description**

Deprecated function

**Usage**

```
surface(...)
```

**Arguments**

```
...           parameters
```

---

tree_metrics	<i>Estimates a set of morphological parameters from a tls point cloud of a tree.</i>
--------------	--

---

**Description**

Estimates a set of morphological parameters from a tls point cloud of a tree.

**Usage**

```
tree_metrics(
  data,
  dbh,
  height,
  crown_diameter,
  crown_proj_area,
  volume,
  message
)
```

**Arguments**

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud.
dbh	numeric and optional. Estimate tree DBH ?
height	numeric and optional. Estimate tree height ?
crown_diameter	numeric and optional. Estimate tree average crown diameter ?
crown_proj_area	numeric and optional. Estimate tree crown projected area ?
volume	numeric and optional. Estimate tree volume ?
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Details**

**Selecting parameters to compute:** If none of dbh,height,crown\_diameter, crown\_proj\_area and volume are passed, all parameters are computed. However, the user can select a set of parameters by declaring wich parameters should be computed (all other are not).

**Parameters estimates:** The tree DBH is estimated as the diameter of a circle fitted to the point cloud between 1.2m and 1.4m above the ground. The tree height is computed as the elevation difference between the lowest and the highest points of the point cloud. Two values are provided for crown parameter. First a 2D convex hull is used to identify the external points of the crown. Then, a first estimate of the crown diameter ("distant\_points") is computed as the average distance of each point to the further point. A second estimate ("circle\_fitting") correspond to the diameter of a circle fitted to the crown external points. The crown projected area is computed as the area of a 2D convex hull that wraps the projected crown. The volume is computed as the volume of a 3D convex hull that wraps the point cloud.

### Value

a list containing the estimated value for each parameter.

### Examples

```
#- import tls data
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- compute all metrics
VoxR::tree_metrics(tls)

#- compute DBH only
VoxR::tree_metrics(tls,dbh = TRUE)

#- compute DBH and height
VoxR::tree_metrics(tls,dbh = TRUE,height = TRUE)
```

---

vox	<i>Voxelisation of 3D point cloud recording the number of points within each voxels.</i>
-----	--

---

### Description

Voxelisation of 3D point cloud recording the number of points within each voxels.

### Usage

```
vox(data, res, full.grid, message)
```

### Arguments

data	a data.frame or data.table containing the x, y, z, ... coordinates of a point cloud.
res	numeric. Resolution of a voxel.
full.grid	logical. If TRUE empty voxels contained in the tree bounding box are returned. If FALSE, only filled voxels are returned. Default = FALSE.
message	logical. If FALSE, messages are disabled. Default = TRUE.

**Value**

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

**References**

Lecigne, B., Delagrange, S., & Messier, C. (2018). Exploring trees in three dimensions: VoxR, a novel voxel-based R package dedicated to analysing the complex arrangement of tree crowns. *Annals of botany*, 121(4), 589-601.

**Examples**

```
#- import file
tls=data.table::fread(system.file("extdata", "Tree_t0.asc", package="VoxR"))

#- resolution = 0.02m
voxels_002 = VoxR::vox(tls,res=0.02) # voxelisation
VoxR::plot_voxels(voxels_002) # voxels plot

#- resolution = 0.2m
voxels_02 = VoxR::vox(tls,res=0.2) # voxelisation
VoxR::plot_voxels(voxels_02) # voxels plot
```

# Index

`axis.angle`, 2  
`axis.distance`, 3  
`axis_angle`, 3  
`axis_distance`, 4  
  
`box_counting`, 5  
  
`distance_clustering`, 6, 8  
  
`filled_voxel_cloud`, 8  
`filter_noise`, 9  
`filter_point_density`, 10  
  
`level`, 11  
  
`obj.rec`, 11  
  
`plot_projection`, 12  
`plot_voxels`, 13  
`plot_voxels_full_grid`, 14  
`point.distance`, 15  
`point_distance`, 15  
`project`, 16  
`project_voxels`, 12, 16  
  
`raster.proj`, 17  
  
`shade3d`, 13, 14  
`sub.obj`, 18  
`subtract_point_clouds`, 6, 18  
`surface`, 19  
  
`tree_metrics`, 20  
  
`vox`, 21