Package 'aRchi'

September 3, 2022

Type Package

Title Quantitative Structural Model ('QSM') Treatment for Tree Architecture

Version 2.1.3

Author Olivier Martin and Bastien Lecigne

Maintainer Olivier Martin <oli.martin@ntymail.com>

Description Provides a set of tools to make quantitative structural model of trees (i.e the so-called 'QSM') from LiDAR point cloud, to manipulate and visualize the QSMs as well as to compute metrics from them. It can be used in various context of forest ecology (i.e biomass estimation) and tree architecture (i.e architectural metrics), see Martin-Ducup et al. (2020) <doi:10.1111/1365-2435.13678>. The package is based on a new S4 class called 'aRchi'.

License CeCILL

Encoding UTF-8

URL https://github.com/umr-amap/aRchi

RoxygenNote 7.1.1

Imports data.table, dplyr, plyr, rgl, methods, lidR, FNN, DiceKriging, stringr, utils, stats, progress, gtools, Rfast, VoxR, fastcluster, pracma, pkgcond, R.matlab, svMisc, circular

Collate 'nullOrnumeric.R' 'nullOrlist.R' 'nullOrLASOrDatatable.R' 'nullOrDatatable.R' 'aRchiClass.R' 'BranchAngle.R' 'Compute_A0.R' 'Compute_Mf.R' 'DAI.R' 'ForkRate.R' 'LeonardoRatio.R' 'Make_Node.R' 'Make_Path.R' 'PathFraction.R' 'QSM2mesh.R' 'SelectinQSM_3d.R' 'TreeBiomass.R' 'TreeVolume.R' 'Truncate_QSM.R' 'WBEparameters.R' 'WoodSurface.R' 'aRchi.R' 'aRchi2treeQSM.R' 'add_diameter.R' 'add_leaves.R' 'add_non_reconstructed.R' 'add_physiological_ages.R' 'add_pointcloud.R' 'angle3d.R' 'build_aRchi.R' 'clean_QSM.R' 'clean_point_cloud.R' 'cross_prod_3d.R' 'dist2line.R' 'get_QSM.R' 'get_nodes.R' 'get_operations.R' 'get_paths.R' 'get_pointcloud.R' 'plot.R' 'read_QSM.R' 'read_aRchi.R' 'segment_annual_shoots.R' 'simplify_skeleton.R' 'skeletonize.R' 'smooth_skeleton.R' 'write_aRchi.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2022-09-03 09:00:02 UTC

R topics documented:

`add_leaves` *Generate foliage to a QSM or skeleton with segmented annual shoots.*

Description

Generate foliage to a QSM or skeleton with segmented annual shoots.

Usage

```
add_leaves(
  aRchi,
  aArea = 73.027,
  bArea = 0.826,
  aNl = 2.563,
  bNl = 0.4402,
  aNin = 2.1337,
  bNin = 0.3818,
  elev = c(17, 45),
  elev_error = 5,
  az_err = 20,
  phyllo = "opposite_decussate",
  simple = T,
  petiole = 0.5
)

## S4 method for signature 'aRchi'
add_leaves(
  aRchi,
  aArea = 73.027,
  bArea = 0.826,
  aNl = 2.563,
  bNl = 0.4402,
  aNin = 2.1337,
  bNin = 0.3818,
  elev = c(17, 45),
  elev_error = 5,
  az_err = 20,
  phyllo = "opposite_decussate",
  simple = T,
  petiole = 0.5
)
```

Arguments

aRchi	a file of class aRchi.
aArea	numeric. Allometric coefficient (a) for leaf area per annual shoot.
bArea	numeric. Allometric coefficient (b) for leaf area per annual shoot.
aNl	numeric. Allometric coefficient (a) for the number of leaves per annual shoot.
bNl	numeric. Allometric coefficient (b) for the number of leaves per annual shoot.
aNin	numeric. Allometric coefficient (a) for the number of internodes per annual shoot.
bNin	numeric. Allometric coefficient (b) for the number of internodes per annual shoot.
elev	numeric. A single value or a vector of length 2 giving the leaves elevation angles.
elev_error	numeric. A random error added to leaves elevation angle.
az_err	numeric. A random error added to leaves azimuth.
phyllo	character. The phyllotaxy used to insert the leaves along the annual shoot. Accepted values: "alternate", "spiral", "opposite", "opposite_decussate".
simple	logical. Bypass the allometry for the number of leaves per annual shoot to insert the leaves on each internode.
petiole	numeric. The petiole length given as a multiplier of the leaf length.

Details

Allometries for the leaf area, number of leaves and number of internodes are computed at the annual shoot level and are of the form $X = a*AS\_length\hat{}b$. The leaves elevation angle is constant if a single value is provided for `elev` or is linearly interpolated based of the leaf relative height within the tree crown if two values are provided.

Value

The aRchi file now including the reconstructed foliage.

References

Lecigne, B., Delagrange, S., Lauri, P. É., & Messier, C. (2022). Trimming influences tree light interception and space exploration: contrasted responses of two cultivars of Fraxinus pennsylvanica at various scales of their architecture. Trees, 1-17. https://doi.org/10.1007/s00468-022-02273-5

Examples

```
# import aRchi file
aRchi=system.file("extdata","Tree_2.aRchi",package = "aRchi")
aRchi = aRchi::read_aRchi(aRchi)

# smooth skeleton
aRchi = smooth_skeleton(aRchi)
```

```
# segment annual shoots
aRchi = aRchi::segment_annual_shoots(aRchi,tree_age = 13)

# add physiological ages
aRchi = aRchi::add_physiological_ages(aRchi)

# add leaves
aRchi = aRchi::add_leaves(aRchi)

plot(aRchi,leaves=TRUE,bg="white",color="chocolate4")
```

`add_non_reconstructed` *Detect and add the axes that were not reconstructed by the QSM method*

Description

Detect and add the axes that were not reconstructed by the QSM method

Usage

```
add_non_reconstructed(
  aRchi,
  max_dist = 99999,
  sec_length = 0.2,
  method = "statistical",
  th = 2.5,
  d_clust = 0.02
)

## S4 method for signature 'aRchi'
add_non_reconstructed(
  aRchi,
  max_dist = 99999,
  sec_length = 0.2,
  method = "statistical",
  th = 2.5,
  d_clust = 0.02
)
```

Arguments

`aRchi`	a file of class aRchi.
`max_dist`	numeric. The maximum distance of a point to the skeleton to be considered in the computation.
`sec_length`	numeric. The length of an axis section to compute the average distance used in the computation of the point relative distance to skeleton.

method	character. Defines the method to use to identify the non reconstructed portions of the point cloud, see details.
th	numeric. The threshold to use to identify the non reconstructed portions of the point cloud. See details.
d_clust	numeric. The distance to use for axes clustering from the identifyed non reconstructed portions of the point cloud.

Details

This function detects the non reconstructed axes by analyzing the point cloud distance to the skeleton. In a first step, the distance of each point to the skeleton is computed. At this step, the points that are too far from the skeleton can be removed with the parameter `max_dist`. The relative distance of the points is then computed by dividing their respective distance by the average distance of all the points within axes sections of a given length (`sec_length`). The points that are far from the skeleton (in terms of relative distance) are then considered as being part of a non reconstructed axis (NR). To do so two parameters are available. First, the `method` parameter defines which method should be used to identify the NR points and `th` sets the threshold to use. If `method = "absolute"` then `th` is a cut-off distance for the relative distance (1 being the average distance). If `method = "statistical"` then `th` is a multiplier of the standard deviation so that the points further than th*sd (sd = the standard deviation of the distribution of relative distances) are considered as NR points. NR axes are then segmented by clustering the NR points based on distance. To do so, the `d_clust` parameter sets the minimal distance between two points to be considered as part of two different axes. The NR axes are then reconstructed as a single segment.

Value

the aRchi file with reconstructed axes added to the skeleton with a field "reconstructed".

Note

The tree topology is fully recomputed. Therefore any existing topology will be lost. It is thus recommended to use this function at the beginning of the processing pipeline.

References

Lecigne, B., Delagrange, S., Lauri, P. É., & Messier, C. (2022). Trimming influences tree light interception and space exploration: contrasted responses of two cultivars of Fraxinus pennsylvanica at various scales of their architecture. Trees, 1-17. https://doi.org/10.1007/s00468-022-02273-5

Examples

```
# import aRchi file
aRchi=system.file("extdata","Tree_2.aRchi",package = "aRchi")
aRchi = aRchi::read_aRchi(aRchi)

# smooth the skeleton
aRchi = aRchi::smooth_skeleton(aRchi,th = 0.01)

# clean point cloud
```

```
aRchi = aRchi::clean_point_cloud(aRchi)

# add missing axes
aRchi = aRchi::add_non_reconstructed(aRchi,th = 3)

# plot the skeleton with reconstructed axes in red
plot(aRchi,color="reconstructed",show_point_cloud = TRUE)
```

`add_physiological_ages` *Add the physiological age of an annual shoots based on their length*

Description

Add the physiological age of an annual shoots based on their length

Usage

```
add_physiological_ages(aRchi, th = c(0.1, 0.2, 0.5), correct_PA = TRUE)

## S4 method for signature 'aRchi'
add_physiological_ages(aRchi, th = c(0.1, 0.2, 0.5), correct_PA = TRUE)
```

Arguments

`aRchi`	an object of class aRchi containing at least a QSM.
`th`	numeric. The length thresholds used to segment annual shoots into physiological ages.
`correct_PA`	logical. Should a correction of the physiological age be performed ? This correction is based on the assumption that the child annual shoot can not be of lower order than its parent annual shoot. The parent annual shoot physiological age is modified accordingly.

Value

The aRchi file with a physiological age field added to the QSM slot

References

Lecigne, B., Delagrange, S., & Taugourdeau, O. (2021). Annual Shoot Segmentation and Physiological Age Classification from TLS Data in Trees with Acrotonic Growth. Forests, 12(4), 391. https://doi.org/10.3390/f12040391

Examples

```
# import aRchi file
aRchi=system.file("extdata","Tree_2.aRchi",package = "aRchi")
aRchi = aRchi::read_aRchi(aRchi)

# smooth skeleton
aRchi = smooth_skeleton(aRchi)

# segment annual shoots
aRchi = aRchi::segment_annual_shoots(aRchi,tree_age = 13)

# add physiological ages
aRchi = aRchi::add_physiological_ages(aRchi)

plot(aRchi,color="physiological_age",bg = "white")
```

---

add_pointcloud *Add a point cloud to an object of class aRchi*

Description

Add a point cloud to an object of class aRchi

Usage

```
add_pointcloud(aRchi, point_cloud)

## S4 method for signature 'aRchi'
add_pointcloud(aRchi, point_cloud)
```

Arguments

aRchi	The object of class aRchi
point_cloud	The point cloud. Either a las or a data.frame

---

add_radius *Add the radius to a skeleton based on point distance to the skeleton*

Description

Add the radius to a skeleton based on point distance to the skeleton

Usage

```
add_radius(aRchi, sec_length = 0.5, by_axis = TRUE, method = "median")

## S4 method for signature 'aRchi'
add_radius(aRchi, sec_length = 0.5, by_axis = TRUE, method = "median")
```

Arguments

aRchi	an object of class aRchi containing at least a point cloud and a QSM.
sec_length	numeric. The length of the section to compute the radius. See details.
by_axis	logical. If TRUE the radius is calculated for each section within each axis, if FALSE the radius is calculated within section for all axes simultaneously.
method	character. The axis radius can be either the mean (method = "mean") or the median (method = "median") distance to the skeleton.

Details

The point distance to skeleton is likely to vary considerably over short distances (e.g. at a branching point). Therefore, the radius is computed by averaging the point distance to the skeleton over sections of user defined length.

Value

The aRchi file with an updated radius in the QSM slot.

Examples

```
# import a point cloud
tls=system.file("extdata","Tree_2_point_cloud.las",package = "aRchi")
tls = lidR::readLAS(tls)

aRchi = aRchi::build_aRchi()
aRchi = aRchi::add_pointcloud(aRchi,point_cloud = tls)

# build a skeleton from the point cloud
aRchi = aRchi::skeletonize_pc(aRchi)

# smooth the skeleton
aRchi = aRchi::smooth_skeleton(aRchi)

# add the diameter to the skeleton
aRchi = aRchi::add_radius(aRchi)

# plot the QSM
plot(aRchi,skeleton = FALSE,show_point_cloud = FALSE)
```

angle3d *Calculate the zenith angle from xyz coordinates*

Description

Calculate a zenith angle between two segments from 3D (i.e X,Y,Z) coordinates

Usage

```
angle3d(o, a, b)
```

Arguments

o	3D coordinates of the common point of the two segments
a	3D coordinates of the other point of segment a
b	3D coordinates of the other point of segment b

Value

The angle in degree

Examples

```
origin=c(0,0,0)
a=c(0,0,1)
b=c(1,0,0)

angle3d(o=origin,a=a,b=b)
```

aRchi-class *aRchi*

Description

Class containing files to compute and display in three dimensions tree architectural metrics at different level of organization

Value

An object of class aRchi

Fields

`QSM` a data.table containing QSM information according to `read_QSM` function format

`pointcloud` a data.table containing the point cloud used to generate the QSM

`Paths` a data.table of Paths according to `Make_Path` function

`Nodes` Metrics computed at the node scale (see `Make_Node`)

`operations` Record all the operations performed on the object.

`aRchi2treeQSM` *Generate a QSM in treeQSM format*

Description

a data.table with treeQSM format (v2.3.3)

Usage

```
aRchi2treeQSM(aRchi)

## S4 method for signature 'aRchi'
aRchi2treeQSM(aRchi)
```

Arguments

`aRchi`	the object of class aRchi to write

`BranchAngle` *Estimation of the tree branch angle from an aRchi file*

Description

Estimate the branch angle of a QSM. Two methods are possible (see method argument)

Usage

```
BranchAngle(aRchi, method = NULL, A0 = FALSE, level = "Tree")

## S4 method for signature 'aRchi'
BranchAngle(aRchi, method = NULL, A0 = FALSE, level = "Tree")
```

Arguments

aRchi	an object of class aRchi with at least a QSM and a path table
method	character. SegmentAngle or King98
A0	logical (default = FALSE). If TRUE the main axis to remove from the calculation is re-estimated using the Compute_A0 function. If false the default branch order 0 is kept.
level	character. The level at which the branch angle is computed. Tree for tree level; branching_order for branch order level; Axis one angle value per axis.

Details

The method "SegmentAngle" compute the angle by considering the first and the last cylinder or each segment, mean is then used for the level of organization selected.

The method "King98" compute the angle by considering the first and the last cylinder of each axis mean is then used for the level of organization selected.

The main axis is always removed.

Value

a numeric or data.table. The branch angle in degree at the selected level. with 0 a perfectly vertical branch angle, 90 a perfectly horizontal branch angle and > 90 a downward branch angle

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Compute the branch angle at various level
BranchAngle(Tree1_aRchi,method="SegmentAngle")
BranchAngle(Tree1_aRchi,level="branching_order",method="SegmentAngle",A0=TRUE)
```

---

build_aRchi *Build a an object of class aRchi*

Description

Build an object of class aRchi

Usage

```
build_aRchi(QSM, point_cloud, keep_original = FALSE)
```

Arguments

QSM	A data.table obtained from read_QSM function
point_cloud	A point cloud. Either a LAS or a data.table with at least three columns with 3d coordinates (i.e X,Y,Z)
keep_original	logical (Default = FALSE). Should the original branching order and axis be kept ? Otherwise, it is re-estimated.

See Also

aRchi; write_aRchi; read_aRchi

Examples

```
file_QSM=system.file("extdata","Tree_1_TreeQSM.txt",package = "aRchi")
file_pc=system.file("extdata","Tree_1_point_cloud.las",package = "aRchi")
QSM=read_QSM(file_QSM,model="treeQSM")
pc=lidR::readLAS(file_pc)
# Make an object of class aRchi
Tree1_aRchi=build_aRchi(QSM=QSM,point_cloud=pc)
```

---

clean_point_cloud *Filter noise from a point cloud*

Description

Uses the filter_noise function to filter noise from the point cloud.

Usage

```
clean_point_cloud(aRchi, k = 5, sigma = 1.5)

## S4 method for signature 'aRchi'
clean_point_cloud(aRchi, k = 5, sigma = 1.5)
```

Arguments

aRchi	An aRchi object containing a point cloud
k	numeric. The number of nearest neighbours to use.
sigma	numeric. The multiplier of standard deviation to consider a point as noise.

Value

The aRchi file with a clean point cloud.

Examples

```
# import aRchi file
aRchi=system.file("extdata","Tree_2.aRchi",package = "aRchi")
aRchi = aRchi::read_aRchi(aRchi)

# clean point cloud
aRchi = aRchi::clean_point_cloud(aRchi)
```

Clean_QSM *Cleans a QSM*

Description

Cleans the QSM in an object of class aRchi by removing branches that have a disproportionate lower radius than their siblings.

Usage

```
Clean_QSM(aRchi, threshold = NULL, plotresult = FALSE)

## S4 method for signature 'aRchi'
Clean_QSM(aRchi, threshold = NULL, plotresult = FALSE)
```

Arguments

aRchi	an object of class aRchi with at least a QSM and a Paths table.
threshold	numeric. The proportion of the largest daughter diameter (between 0 and 1) under which a branch is removed.
plotresult	logical (default = FALSE). Show the results in a 3d plot if TRUE

Details

This cleaning is done by browsing the tree QSM from the base to the top. Each time a ramification point is encountered a daughter branch is removed if its radius is lower than a selected (i.e threshold) proportion of radius of the largest daughter. This allows removing small branches on large branches that can be for example traumatic or epicormic shoots or false branches due to noise in QSM. In ForkRate function the same approach is used with a threshold of 75% (i.e 0.75) to count the number of fork and compute the fork rate.

Value

An object of class aRchi with the cleaned QSM.

See Also

ForkRate to compute the fork rate; Truncate_QSM to truncate a QSM at a specific diameter threshold

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Clean the QSM: threshold of 0.5
Cleaned_Tree1_aRchi=Clean_QSM(Tree1_aRchi,threshold = 0.5,plotresult = TRUE)
# show the cleaned QSM data.table
get_QSM(Cleaned_Tree1_aRchi)
```

Compute_A0 *Find the principal axis of a tree*

Description

Find the principal axis of a tree (i.e A0) and add a column to the QSM of an aRchi object. This is alternative method to the default branch order proposed in a QSM for the principal axis only.

Usage

```
Compute_A0(aRchi, plotresult = FALSE)

## S4 method for signature 'aRchi'
Compute_A0(aRchi, plotresult = FALSE)
```

Arguments

aRchi	an object of class aRchi with at least a QSM and the Paths table
plotresult	logical (default = FALSE). Show the results in a 3d plot if TRUE

Details

The method used to find the principal axis consist in finding the highest vertical path with consecutive segments of similar diameters and orientations. An index called A0 of the probability of being the principal axis is thus computed for each path of the tree and the path with the highest value is considered as the principal axis (see Martin-Ducup et al. 2020 for more information).

A0 ranges between 0 and 4 with 0 indicating a path with a low probability of being the principal axis and 4 indicating a high probability of being the principal axis, i.e. the highest vertical path with consecutive segments of similar diameters and orientations.The path with the maximum A0 value was selected as the principal axis.

The new column A0 of the QSM slot take the value 2 if the cylinder is part of the principal axis or 1 if not.

Value

The aRchi object with the QSM having a new column A0.

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

See Also

DAI to compute the dominance of a principal axis index that uses the A0 index.

Examples

```
# Read an aRchi file with at least the QSM and the paths table
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

Tree1_aRchi=Compute_A0(Tree1_aRchi,plotresult=TRUE)
```

Compute_Mf *Compute Moment of force*

Description

Compute the moment of gravity force Mf and the moment of gravity force relative to cylinder radius Mf_r from an object of class aRchi.

Usage

```
Compute_Mf(aRchi, WoodDensity)

## S4 method for signature 'aRchi'
Compute_Mf(aRchi, WoodDensity = NULL)
```

Arguments

aRchi	an object of class aRchi with at least the QSM and the paths table.
WoodDensity	a numeric or a data.table. A single wood density value for the whole tree or one value per cylinder in kg/m3. If wood density is given for each cylinder a data.table with two column (i.e cyl_ID and WoodDensity) must be given.

Details

The moment of gravity force (i.e `Mf`) is calculated at each cylinder position. `Mf` can be seen as a proxy of the mechanical loading history due to gravity at a given position of a tree. This quantity is defined by the following the equation: Mf=R*F where R is the lever arm, which is the norm of the horizontal vector between the position where `Mf` is measured (i.e a cylinder) and the position where the force is applied (i.e., the center of mass, G, of the whole structure upstream a cylinder: a subtree).

The mass of the cylinders are needed to calculate the center of mass and are estimated using their volume and the wood density provided in argument `WoodDensity`. Finally, F is the weight of the subtree: F=g*M with g the standard acceleration due to gravity (9.81 $m.s^{-2}$).

The moment of gravity force relative (i.e `Mf_r`) to cylinder radius r is also computed following the formula: Mf_r = Mf/r^3

Value

The aRchi file with the QSM slot having three new columns: the biomass upstream the cylinder `sub_tree_biomass`, the moment of gravity force `Mf` and the moment of gravity force relative to cylinder radius `Mf_r`.

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

# Compute the moment of force for each cylinder
Tree1_aRchi=Compute_Mf(Tree1_aRchi,WoodDensity=550)

# show the QSM data.table with the three new columns sub_tree_biomass, MF and Mf_r)
get_QSM(Tree1_aRchi)
```

DAI *Estimate the index of dominance of the principal axis: DAI.*

Description

Estimate the index of dominance of the principal axis (DAI) from an aRchi object.

Usage

```
DAI(aRchi)

## S4 method for signature 'aRchi'
DAI(aRchi)
```

Arguments

aRchi	an object of class aRchi with at least the QSM and the Paths table

Details

The idea of DAI is to disentangle architectures of trees with a strong apical dominance, i.e. with a central main stem growing more strongly than other side axes (such as in most conifers, for instance), from those having a spread out branching pattern with similar axes and no obvious main stem.

The higher the index the more dominant the principal axis. DAI is computed based on the indices A0 which is an index of the probability of being the principal axis for each path of the tree (see function Compute_A0). More information are given in Martin-Ducup et al 2020

Value

Numeric. The value of DAI.

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

See Also

Compute_A0 to identify the principal axis (i.e max(A0)).

Examples

```
# Read an aRchi file with at least the QSM and the paths table
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

DAI(Tree1_aRchi)
```

ForkRate *Compute the fork rate of a tree*

Description

Compute the fork rate from an aRchi object

Usage

```
ForkRate(aRchi)

## S4 method for signature 'aRchi'
ForkRate(aRchi)
```

Arguments

aRchi	a file of class aRchi with at least a QSM and a Path table.

Details

The fork rate is the mean number of forks per meter of tree height. This metric is computed by browsing tree QSM from the base to the top. Each time a ramification point is encountered it is evaluated as a fork if at least one daughter had a radius not less than 75% of the diameter of the largest daughter. This threshold was chosen in order to exclude non-perennial structures, such as traumatic or epicormic shoots, or branches that will not last on the tree. If the ramification point is a fork, all retained daughter branches are browsed through until the next ramification point and further until the path end. If a daughter is rejected, it is removed as well as all the paths passing through it.

Value

a vector with two numeric value. The number of Fork and the fork rate.

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

See Also

Clean_QSM to clean a QSM based on a threshold of percentage of largest daughter's diameter.

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Compute the fork rate of Tree1
ForkRate(Tree1_aRchi)
```

---

get_nodes *Get the nodes from an object of class aRchi*

Description

Get the nodes from an object of class aRchi

Usage

```
get_nodes(aRchi)

## S4 method for signature 'aRchi'
get_nodes(aRchi)
```

Arguments

aRchi	The object of class aRchi

See Also

get_QSM; get_pointcloud; get_paths

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

# get the nodes (a list of two data.table)
get_nodes(Tree1_aRchi)
```

---

get_operations *Get the nodes from an object of class aRchi*

Description

Show a list with all the operations (and their parameters) that have been performed on an object of class aRchi

Usage

```
get_operations(aRchi)

## S4 method for signature 'aRchi'
get_operations(aRchi)
```

Arguments

aRchi	The object of class aRchi

See Also

get_QSM; get_pointcloud; get_paths

Examples

```
## Not run:
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

# Making some operations
Tree1_aRchi<-Make_Path(Tree1_aRchi)
```

```
Tree1_aRchi=Compute_A0(Tree1_aRchi)
Tree1_aRchi=Compute_Mf(Tree1_aRchi,WoodDensity = 550)
Tree1_aRchi=Clean_QSM(Tree1_aRchi,threshold = 0.5)

# Show the oprations and their parameters
get_operations(Tree1_aRchi)

## End(Not run)
```

---

get_paths *Get the paths from an object of class aRchi*

Description

Get the paths from an object of class aRchi

Usage

```
get_paths(aRchi)

## S4 method for signature 'aRchi'
get_paths(aRchi)
```

Arguments

aRchi	The object of class aRchi

See Also

get_QSM; get_pointcloud; get_nodes;

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

# get the paths (a data.table)
get_paths(Tree1_aRchi)
```

get_pointcloud *Get the pointcloud from an object of class aRchi*

Description

Get the pointcloud from an object of class aRchi

Usage

```
get_pointcloud(aRchi)

## S4 method for signature 'aRchi'
get_pointcloud(aRchi)
```

Arguments

aRchi	The object of class aRchi

See Also

get_QSM; get_paths; get_nodes

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

# get the pointcloud
get_pointcloud(Tree1_aRchi)
```

get_QSM *Get the QSM from an object of class aRchi*

Description

Get the QSM from an object of class aRchi

Usage

```
get_QSM(aRchi)

## S4 method for signature 'aRchi'
get_QSM(aRchi)
```

Arguments

aRchi	The object of class aRchi

See Also

get_pointcloud; get_paths; get_nodes

Examples

```
# Read an aRchi file with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

# show the QSM data.table
get_QSM(Tree1_aRchi)
```

LeonardoRatio *Compute Leonardo's ratio*

Description

Compute from an object of class aRchi the Leonardo's ratio (i.e R_ratio) at node, axis, branch order or tree level.

Usage

```
LeonardoRatio(aRchi, level = "Tree", position = 10)

## S4 method for signature 'aRchi'
LeonardoRatio(aRchi, level = "Tree", position = 10)
```

Arguments

aRchi	an object of class aRchi with at least a QSM and the Nodes table (see function Make_Node)
level	characters. At which level R_ratio has to be estimated. Node for node level, Axis for the axis level, branching_order for branch order level and Tree for tree level (default).
position	At which position from the node R_ratio had to be estimated. Either a numeric or a character. Use a numeric multiple of ten to select the distance from the node in cm where R ratio has to be estimated (e.g 10 for 10cm from the node). Use the % sign after a multiple of ten to select the distance from the node in percentage of the length of the parent and daughters segments (e.g 50% for an estimation at mid-length of the segments). Note that 0 is accepted and correspond to the closest position from the node.

Details

Details for Leonardo Da Vinci's ratio calculation are given in the details part of function Make_Node.

Value

Data.table of the summary of R_ratio for the selected level.

See Also

Make_Node for node metrics estimation; WBEparameters to estimates WBE parameters at different level;

Examples

```
# Read an aRchifile with a QSM and node tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Leonardo'ratio at the branching order level estimated at midlength of the segments
LeonardoRatio(Tree1_aRchi,level="Tree", position="50%")
# Leonardo'ratio at the node level estimated at 30 cm from the node
LeonardoRatio(Tree1_aRchi,level="Node", position=30)
```

Make_Node *Make Node*

Description

Compute several node metrics (i.e Leonardo's ratio and WBE parameters) from an object aRchi at different distance from the node.

Usage

```
Make_Node(aRchi, all_combination = FALSE)

## S4 method for signature 'aRchi'
Make_Node(aRchi, all_combination = FALSE)
```

Arguments

aRchi	a file of class aRchi containing at least a QSM
all_combination	logical (default = FALSE). Should the node metrics be computed at each combination of distance from the node ? (see details).

Details

The Nodes slot contains a list of two data.table (Absolute_positions and Relative_positions). Each data.tables contains for each node several values of Leonardo's rule ratio (R_ratio), radius scaling exponent (alpha), length scaling exponent (beta) and the estimated metabolic rate (theta) which are parameters of the West Brown and Enquist metabolic theory (see Bentley et al. 2013, Lau et al 2019 and Martin-Ducup et al. 2020).

For a given node, each R_ratio and alpha correspond to its value estimated with the branch radius (for alpha) or the cross section area (for R_ratio) at a given position from the node for the parent (pos_parent) and for the daughters (pos_daugthers). The positions (i.e pos_parent and pos_daughters) are the distances in meters (for Absolute_position data.table) or in percentage (for Relative_position data.table) from the node (i.e the ramification point) to the point of radius or cross section area estimation. These positions are given at a 10 cm step and cannot be higher than the length of the shorter segment among the couple daughters/parent for Absolute_position and at a 10% step for the Relative_position.

Kriging models are used to estimate radius (and thus cross section area) along the segment positions. For example pos_parent = 0.2 and pos_daugthers = 0.2 for Absolute_position data.table means that the parameters (i.e R_ratio and alpha) have been estimated at 20cm from the node position for the parent and 20 cm from the node position for the daughters. For the Relative_position data.table, this position is given in proportion to the total length of the segment. For example, pos_parent = 0.5 and pos_daughter = 0.5 means that the parameters are estimated at mid_length of the segments for both the parent and the daughters).

beta values are repeated for each node as it depend on segment length only and not on radius position.

If `all_combination = TRUE` all the possible combination for pos_parent and pos_daughter are computed (e.g pos_parent = 0.3 and pos_daughter = 0.5) but note that this processing might take several minutes and is `FALSE` by default.

Value

The aRchi file with the Nodes slot filled.

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

Lau, A. et al. Estimating architecture-based metabolic scaling exponents of tropical trees using terrestrial LiDAR and 3D modelling. Forest Ecology and Management 439, 132–145 (2019).

Bentley, L. P. et al. An empirical assessment of tree branching networks and implications for plant allometric scaling models. Ecology Letters 16, 1069–1078 (2013).

See Also

`WBEparameters` to estimates WBE parameters at different level; `LeonardoRatio` to estimates Leonardo Da Vinci's ratio at different level.

Examples

```
# Read a QSM file
file=system.file("extdata","Tree_1_TreeQSM.txt",package = "aRchi")
QSM=read_QSM(file,model="treeQSM")
# Build an object of class aRchi
Tree1_aRchi<-build_aRchi(QSM=QSM)
Tree1_aRchi
# Make the node table
Tree1_aRchi<-Make_Node(Tree1_aRchi)
Tree1_aRchi
# Leonardo ratio at the tree level
LeonardoRatio(Tree1_aRchi)
```

Make_Path *Make the path of a QSM in an aRchi object*

Description

Identify and record the paths of a QSM in an object of class aRchi. The path are needed for several tree metrics estimation of the aRchi package.

Usage

```
Make_Path(aRchi)

## S4 method for signature 'aRchi'
Make_Path(aRchi)
```

Arguments

aRchi	a file of class aRchi

Details

A path is a continuous succession of cylinders from a terminal segment (i.e a branch tip) to the trunk base. Thus, there is as many path as terminal segments in a QSM.

This function fill the slot Paths of an object of class aRchi with a data.table of the paths. This data.table contains the same variables as a classic QSM plus an ID_path column. This table is thus larger than the QSM table as each cylinder is repeated as many times as it appears in a path. For example, the first cylinder of the QSM (i.e the beginning of the trunk) is repeated N path times, with N path the number of path of the QSM.

Many function of the aRchi packages request a path table (check see also).

Value

The aRchi object with the table of paths

See Also

BranchAngle;Truncate_QSM; Clean_QSM; ForkRate; PathFraction

Examples

```
# Read a QSM file
file=system.file("extdata","Tree_1_TreeQSM.txt",package = "aRchi")
QSM=read_QSM(file,model="treeQSM")
# Build an object of class aRchi
Tree1_aRchi<-build_aRchi(QSM=QSM)
Tree1_aRchi
# Make the path table
Tree1_aRchi<-Make_Path(Tree1_aRchi)
Tree1_aRchi
PathFraction(Tree1_aRchi)
```

---

nullOrDatatable-class *nullOrDatatable*

Description

Class union

---

nullOrLASOrDatatable-class *nullOrLASOrDatatable*

Description

Class union

---

nullOrlist-class *nullOrlist*

Description

Class union

nullOrnumeric-class *nullOrnumeric*

Description

Class union

PathFraction *Compute the path fraction*

Description

Compute from an object of class aRchi the path fraction

Usage

```
PathFraction(aRchi)

## S4 method for signature 'aRchi'
PathFraction(aRchi)
```

Arguments

aRchi	an object of class aRchi with at least the QSM and the Paths table

Details

The path fraction is the ratio between the mean path length and the maximum path length.

Value

The path fraction

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

Smith, D. D. et al. Deviation from symmetrically self-similar branching in trees predicts altered hydraulics, mechanics, light interception and metabolic scaling. New Phytologist 201, 217–229 (2014).

See Also

Make_Path to compute the paths table.

Examples

```
# Read an aRchi file with at least the QSM and the paths table
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)

PathFraction(Tree1_aRchi)
```

`plot,aRchi,ANY-method` *Plot an object of class aRchi*

Description

Plot an object of class aRchi.

Usage

```
## S4 method for signature 'aRchi,ANY'
plot(
  x,
  y,
  transparency = 1,
  color = "white",
  bg = "black",
  lwd = 3,
  show_point_cloud = FALSE,
  skeleton = TRUE,
  leaves = FALSE,
  lit = TRUE
)
```

Arguments

`x`	An aRchi object
`y`	Unused (inherited from R base)
`transparency`	The transparency of the cylinders
`color`	The color of the cylinders. Can be either a single color or a level of organization: "branching_order" for branching branching_order, "cylinder" to coloryze each cylinder independently, "segment" to coloryze the branch segments, "axis" to coloryze the axis, "A0" to colorize only the main axis from `Compute_A0` function
`bg`	The background color
`lwd`	line width of the skeleton
`show_point_cloud`	logical (Default = `FALSE`). Display the point cloud ?
`skeleton`	logical (Default is `TRUE`). Display the skeleton only (i.e segments). Faster than displaying the whole QSM with the fleshed cylinders.

leaves	logical (Default is FALSE). Display the leaves ?
lit	logical (Default is TRUE). Specify if lighting calculation should take place on the geometry. Only applies if skeleton = FALSE.

Details

Plot an object of class aRchi in a 3d device. The QSM can be plotted according to different level of organization and the point cloud can be displayed if available.

Examples

```
# Read an aRchi file with at least a QSM
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Plot the QSM by coloring the branching order
plot(Tree1_aRchi,color="branching_order")
# Same with the fleshed cylinder and the point cloud
plot(Tree1_aRchi,color="branching_order",skeleton=FALSE,show_point_cloud=TRUE)
```

---

read_aRchi *Read an aRchi file*

Description

Read an aRchi file

Usage

```
read_aRchi(file)

## S4 method for signature 'character'
read_aRchi(file)
```

Arguments

file	The directory to the .aRchi file.

See Also

write_aRchi

read_QSM *Read a QSM*

Description

Read a QSM file generated with treeQSM, simpletree, simpleforest or pypetree.

Usage

```
read_QSM(file, model)
```

Arguments

file	The directory to the QSM file path.
model	treeQSM, simpletree, simpleforest or pypetree depending on the algorithm used to generate the QSM

Details

For treeQSM model, .mat from treeQSM are allowed. the old format (V2.3) as well as the new format (V2.4) are allowed.

Value

a list containing a data.table with the QSM and a character with the model name. This list can be used to build an aRchi object (see function build_aRchi)

See Also

aRchi the aRchi class;build_aRchi to build an object of class aRchi

Examples

```
file=system.file("extdata","Tree_1_TreeQSM.txt",package = "aRchi")
QSM=read_QSM(file,model="treeQSM")
```

`segment_annual_shoots` *Annual shoot segmentation in tree skeleton*

Description

Segment the annual shoots in a tree skeleton based on the detection of the branching patterns created by acrotony.

Usage

```
segment_annual_shoots(aRchi, tree_age, segment_reiterations)

## S4 method for signature 'aRchi'
segment_annual_shoots(aRchi, tree_age, segment_reiterations)
```

Arguments

`aRchi`	an object of class aRchi containing at least a QSM.
`tree_age`	numeric, optional. The tree age. Helps to achieve more robust segmentation.
`segment_reiterations`	list of numeric values. The parameters to segment traumatic reiterations based on their age difference with the bearer annual shoot and their elevation. The list must have the following form: list(age_difference, elevation_angle) where age_difference and elevation_angle are numeric vectors. NOTE the elevation angle is defined relative to the zenith.

Value

The input aRchi file with an additional field in the QSM slot being the segmented annual shoots. NOTE that annual shoot = 1 correspond to the last growing season. If traumatic reiteration segmentation was achieved, an additional field labeling cylinders that belong to a reiteration is added.

References

Lecigne, B., Delagrange, S., & Taugourdeau, O. (2021). Annual Shoot Segmentation and Physiological Age Classification from TLS Data in Trees with Acrotonic Growth. Forests, 12(4), 391. https://doi.org/10.3390/f12040391

Examples

```
# import aRchi file
aRchi=system.file("extdata","Tree_2.aRchi",package = "aRchi")
aRchi = aRchi::read_aRchi(aRchi)

# smooth skeleton
aRchi = smooth_skeleton(aRchi)
```

```
# segment annual shoots
aRchi = aRchi::segment_annual_shoots(aRchi,tree_age = 13)

plot(aRchi,color="annual_shoots",bg = "white")
```

SelectinQSM_3d *SelectinQSM_3d*

Description

Select interactively a sub-part of a QSM (cylinder, segment, node, axis, branch, subtree) in a 3d device and return its characteristics.

Usage

```
SelectinQSM_3d(aRchi, skeleton = TRUE, level = "cylinder")

## S4 method for signature 'aRchi'
SelectinQSM_3d(aRchi, skeleton = TRUE, level = "cylinder")
```

Arguments

aRchi	An object of class aRchi
skeleton	logical. Display the skeleton only. Default is TRUE. Faster than displaying the QSM with the fleshed cylinders.
level	character. cylinder (default), segment, node, axis, branch, subtree.

Details

The selection is performed in two times: i) Identifying the zone of interest in the 3d device and zoom into it if needed. When identified, the user has to hit enter in the R console. At this point, it is impossible to rotate the displayed QSM anymore as the left button of the mouse is used for the selection. However translation are still possible with the right button. ii) Draw a rectangle with the left button of the mouse in the zone of interest.

Some details about the level of organization are given below.

"cylinder": return characteristics for the cylinders selected only

"segment:" return characteristics for the cylinders of the segments selected

"node": return the characteristics for the cylinders of the node selected. For a specific node, select the mother.

"axis": return the characteristics for the cylinders of the axis selected. An axis is a continuous succession of cylinder having a same branching order value.

"branch": return the characteristics for the cylinders of the branch selected. A branch is similar to an axis but regroup also everything that is upstream the axis (i.e all that the axis carries)

"subtree": return the characteristics for the cylinders of the subtree selected. A subtree is similar to a branch but starting from the cylinder selected and not from the point of insertion of the selected axis. In other word, when the user draw a rectangle on a cylinder, the subtree selection return all that the cylinder carries. If several cylinders are selected, the subtree selection return all that the most downstream cylinder carries.

Value

a data.table with the cylinders characteristics at the requested level (i.e sub-part of the original QSM).

Examples

```
# Read an aRchi file with at least a QSM
if(interactive()){
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Select a branch
SelectinQSM_3d(Tree1_aRchi,level="branch")
# Same with the fleshed cylinder and keep the branch QSM in an object
My_branch=SelectinQSM_3d(Tree1_aRchi,level="branch",skeleton=FALSE)
My_branch
# Compute the moment of force
Tree1_aRchi=Compute_Mf(Tree1_aRchi,WoodDensity=550)
#Select a cylinder to return the moment of force at his position
SelectinQSM_3d(Tree1_aRchi,skeleton=FALSE)
}
```

---

simplify_skeleton *Simplify a skeleton by removing unnecessary cylinders*

Description

Simplify a skeleton by removing unnecessary cylinders

Usage

```
simplify_skeleton(aRchi, seg_length = 0.1)

## S4 method for signature 'aRchi'
simplify_skeleton(aRchi, seg_length = 0.1)
```

Arguments

aRchi	an object of class aRchi containing at least a QSM.
seg_length	numeric. The target maximal cylinder length. See details.

Details

Simplifies a QSM by merging short cylinders into longer cylinders with a length close to a user defined target length. Note that short cylinders are kept if they support a branching point so that the overall QSM geometry is not affected by the simplification process.

Value

a aRchi file with the simplified QSM.

Examples

```
# import a point cloud
tls=system.file("extdata","Tree_2_point_cloud.las",package = "aRchi")
tls = lidR::readLAS(tls)

aRchi = aRchi::build_aRchi()
aRchi = aRchi::add_pointcloud(aRchi,point_cloud = tls)

# build a skeleton from the point cloud
aRchi = aRchi::skeletonize_pc(aRchi)

# simplyfy the skeleton
aRchi = aRchi::simplify_skeleton(aRchi,seg_length = 0.05)

# plot the simplifyed skeleton
plot(aRchi,show_point_cloud = TRUE)
```

---

skeletonize_pc *Build the skeleton of a tree point cloud*

Description

Build the skeleton of a tree point cloud

Usage

```
skeletonize_pc(
  aRchi,
  D = 0.03,
  progressive = TRUE,
  cl_dist = 0.02,
  max_d = 0.05
)

## S4 method for signature 'aRchi'
skeletonize_pc(
```

```
  aRchi,
  D = 0.03,
  progressive = TRUE,
  cl_dist = 0.02,
  max_d = 0.05
)
```

Arguments

aRchi	an object of class aRchi containing at least a point cloud.
D	numeric. The distance of research for point neighborhood. Sets the layer thickness. See description for details.
progressive	logical. Should the clustering distance be progressive ? See description for details.
cl_dist	numeric. The clustering distance. If pregressive = FALSE sets the clustering distance for all the point cloud. If pregressive = TRUE sets the minimum clustering distance to be used. See description for details.
max_d	The maximum searching distance for skeleton building. See description for details.

Details

The skeletonization algorithm works in four steps. At STEP 1 the point cloud is divided in layers of regular thickness (defined by parameter D). To do so, the tree base is first defined as the first layer. Then, all the points that are within D of any points of the layer *N* belong to the layer *N+1*. This process continues until no more points are found within D. If there are some remaining points (that are further than D to any already classified points), a new layer is defined as the point that is the closest of the already classified point. This continues until no more points remain unclassified. At STEP 2, the layers are divided into clusters based on point distance: two point that are further than a given distance are considered as being part of two different objects. Two possibilities are available to define the clustering distance. First, it remains constant and is defined by cl_dist. Second, the distance is defined as the average distance of the points of the previous layer to the center of their corresponding cluster (this is achieved by setting progressive = TRUE). In this latter case, cl_dist defines the minimal clustering distance that can be used. This option helps to adapt the clustering distance to the size of the actual objects (i.e. branch sections) that have to be to clustered. By default the first layer is assumed as being part of a single cluster. At STEP 3, the cluster centers are used as node of the skeleton and are combined to iteratively build the skeleton. At first iteration, the node with the smallest Z value is used as the root node. At subsequent iterations, the node located within max_d and that is the closest to the root node is integrated into the skeleton and is selected as new root node. This process continues until no node is found within max_d to the root node (either because the cluster is a branch tip or because there is a gap in the point cloud). In this case, the node that belong to the layer that was produced at earliest iteration of STEP 1 and that is the closer to a skeleton node is selected as new root node. This process ends once all nodes are integrated into the skeleton. At STEP 4, a basic tree topology is computed. First, an unique ID is assigned to all segments of the skeleton (i.e. the segment that link two nodes) and its parent segment ID is retrieved. The segments are then classified into axes. An axis being defined as a continuous set of segments that always follow the segment that support the longest portion of the skeleton. Axes are then partitioned into axes segments defined as the portion of an axis located

between two branching point or between a branching point and an extremity of the axis. The axis branching order is then computed as the number of branching point observed between the tree base and the axis first segment + 1.

Value

an aRchi file containing the original point cloud and the corresponding skeleton

Examples

```
#################################################################
# Example with a small high quality point cloud : using default #
# default parameters to detect fine architectural details       #
#################################################################
# import a point cloud
tls=system.file("extdata","Tree_2_point_cloud.las",package = "aRchi")
tls = lidR::readLAS(tls)

# build an empty aRchi file and add the point cloud
aRchi = aRchi::build_aRchi()
aRchi = aRchi::add_pointcloud(aRchi,point_cloud = tls)

# plot the point cloud
plot(aRchi@pointcloud)

# build a skeleton from the point cloud
aRchi = skeletonize_pc(aRchi)

# smooth the skeleton
aRchi = smooth_skeleton(aRchi)

# plot the skeleton
plot(aRchi,show_point_cloud = TRUE)


#################################################################
# Example with a large point cloud with a lot of occlusion : #
# parameters selected for speed                               #
#################################################################
# import a point cloud
tls=system.file("extdata","Tree_1_point_cloud.las",package = "aRchi")
tls = lidR::readLAS(tls)

# build an empty aRchi file and add the point cloud
aRchi = aRchi::build_aRchi()
aRchi = aRchi::add_pointcloud(aRchi,point_cloud = tls)

# plot the point cloud
plot(aRchi@pointcloud)

# build a skeleton from the point cloud
```

```
aRchi = skeletonize_pc(aRchi, D = 0.5, cl_dist = 0.2, max_d = 1)

# smooth the skeleton
aRchi = smooth_skeleton(aRchi)

# plot the skeleton
plot(aRchi,show_point_cloud = TRUE)
```

`smooth_skeleton` *Smooth a tree skeleton*

Description

Smooth a tree skeleton

Usage

```
smooth_skeleton(aRchi, niter = 1, th = 0)

## S4 method for signature 'aRchi'
smooth_skeleton(aRchi, niter = 1, th = 0)
```

Arguments

`aRchi`	a file of class aRchi containing at least a skeleton.
`niter`	integer. Number of iterations to perform.
`th`	numeric. The distance threshold to correct the segments tips position.

Details

Smooth a skeleton or a QSM by computing the distance of a node to the segment formed by its parent and child nodes of the same axis. If the distance is greater than `th`, the node coordinates are moved on the segment. This process is done `niter` times.

Value

a file of class aRchi with a smoothed skeleton

Examples

```
# import aRchi file
aRchi=system.file("extdata","Tree_2.aRchi",package = "aRchi")
aRchi = aRchi::read_aRchi(aRchi)

# plot original skeleton
plot(aRchi)
```

```
# smooth skeleton
aRchi = aRchi::smooth_skeleton(aRchi)

# plot smoothed skeleton
plot(aRchi)
```

`TreeBiomass` *Tree biomass estimation of the woody part*

Description

Compute the tree wood biomass at different level of organization

Usage

```
TreeBiomass(aRchi, WoodDensity, level = "Tree")

## S4 method for signature 'aRchi'
TreeBiomass(aRchi, WoodDensity = NULL, level = "Tree")
```

Arguments

`aRchi`	an object of class aRchi with at least a QSM
`WoodDensity`	a numeric or a data.table. A single wood density value for the whole tree or one value per cylinder in kg/m3. If wood density is given for each cylinder a data.table with two column (i.e cyl_ID and WoodDensity) must be given.
`level`	character. The level at which the wood biomass is computed. `Tree`, `branching_order` or `Axis`.

Value

a numeric or data.table. The wood biomass in Kg at the requested level

Examples

```
# Read an aRchi file with at least a QSM
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Compute the whole tree wood biomass.
TreeBiomass(Tree1_aRchi,WoodDensity=550)
```

TreeVolume *Tree volume estimation of the woody part*

Description

Compute the tree wood volume at different level of organization

Usage

```
TreeVolume(aRchi, level = "Tree")

## S4 method for signature 'aRchi'
TreeVolume(aRchi, level = "Tree")
```

Arguments

aRchi	an object of class aRchi with at least a QSM
level	character. The level at which the wood volume is computed. Tree, branching_order or Axis.

Value

a numeric or data.table. The wood volume in m3 at the requested level

Examples

```
# Read an aRchi file with at least a QSM
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Compute the whole tree wood biomass.
TreeVolume(Tree1_aRchi)
```

Truncate_QSM *Truncate a QSM*

Description

Truncate a QSM at a radius threshold

Usage

```
Truncate_QSM(
  aRchi,
  threshold = NULL,
  Keepdaughters = FALSE,
  plotresult = FALSE
)

## S4 method for signature 'aRchi'
Truncate_QSM(
  aRchi,
  threshold = NULL,
  Keepdaughters = FALSE,
  plotresult = FALSE
)
```

Arguments

aRchi	an object of class aRchi with at least a QSM and a Paths table.
threshold	numeric. The radius threshold in meter.
Keepdaughters	logical (default = FALSE). Keep the daughters of the last segment retained even if they are lower than the threshold.
plotresult	logical (default = FALSE). Show the results in a 3d plot if TRUE

Details

The threshold is applied to a whole segments. In other word, if a segment has at least one cylinder lower than the threshold it is removed as well as everything upstream (except the direct daughters if Keepdaughters=TRUE).

Value

An aRchi file with the QSM truncated

See Also

Clean_QSM to clean a QSM of an object aRchi.

Examples

```
# Read an aRchifile with a QSM and paths tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Truncate the QSM: 5cm radius threshold
Truncated_Tree1_aRchi=Truncate_QSM(Tree1_aRchi,plotresult = TRUE,threshold = 0.05)
```

`WBEparameters` *Compute WBE parameters*

Description

Compute from an object of class aRchi the West, Brown and Enquist (WBE) scaling exponent (alpha and beta) and the subsequent estimated metabolic rate (theta) at node, branch, branch order or tree level.

Usage

```
WBEparameters(aRchi, level = "Tree", position = 10)

## S4 method for signature 'aRchi'
WBEparameters(aRchi, level = "Tree", position = 10)
```

Arguments

`aRchi`	an object of class aRchi with at least a QSM and the Nodes table (see function `Make_Node`)
`level`	characters. At which level R_ratio has to be estimated. `Node` for node level, `Axis` for axis level, `branching_order` for branch order level and `Tree` for tree level (default).
`position`	At which position from the node WBE parameters have to be estimated. Either a numeric or a character. Use a numeric multiple of ten to select the distance from the node in cm where R ratio has to be estimated (e.g 10 for 10cm from the node). Use the % sign after a multiple of ten to select the distance from the node in percentage of the length of the parent and daughters segments (e.g 50% for an estimation at mid-length of the segments). Note that 0 is accepted and correspond to the closest position from the node.

Details

Details for WBE parameters calculation are given in the details part of function `Make_Node`.

Value

Data.table of the summary (median and mean) of WBE parameters at the selected level. The median should be used according to Bentley et al. 2013.

References

Martin-Ducup, O. et al. Terrestrial laser scanning reveals convergence of tree architecture with increasingly dominant crown canopy position. Functional Ecology (2020).

Lau, A. et al. Estimating architecture-based metabolic scaling exponents of tropical trees using terrestrial LiDAR and 3D modelling. Forest Ecology and Management 439, 132–145 (2019).

Bentley, L. P. et al. An empirical assessment of tree branching networks and implications for plant allometric scaling models. Ecology Letters 16, 1069–1078 (2013).

See Also

Make_Node for node metrics estimation; LeonardoRatio to estimates Leonardo Da Vinci's ratio at different level.

Examples

```
# Read an aRchifile with a QSM and node tables.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# WBE parameters at the branching order level estimated at midlength of the segments
LeonardoRatio(Tree1_aRchi,level="Tree", position="50%")
# WBE parameters at the tree level estimated at 10 cm from the node
LeonardoRatio(Tree1_aRchi)
```

WoodSurface *Estimate the unrolled wood surface from a QSM*

Description

Estimate the unrolled wood surface from a QSM by summing the area of all cylinders at several level of organization.

Usage

```
WoodSurface(aRchi, level = "Tree")

## S4 method for signature 'aRchi'
WoodSurface(aRchi, level = "Tree")
```

Arguments

aRchi	a file of class aRchi
level	text. The level at which the wood surface is computed. Tree, branching_order or Axis.

Value

a numeric or data.table. The wood surface in m2

Examples

```
# Read an aRchi file with at least a QSM.
file=system.file("extdata","Tree_1_aRchi.aRchi",package = "aRchi")
Tree1_aRchi=read_aRchi(file)
# Compute the wood surface at branching order level.
WoodSurface(Tree1_aRchi,level="branching_order")
```

`write_aRchi` *Write an aRchi file*

Description

write an aRchi file

Usage

```
write_aRchi(aRchi, file)

## S4 method for signature 'aRchi,character'
write_aRchi(aRchi, file)
```

Arguments

`aRchi`	the object of class aRchi to write
`file`	The directory where to write the .aRchi file.

See Also

`read_aRchi`

Index