Package 'FORTLS'

January 19, 2026

Title Automatic Processing of Terrestrial-Based Technologies Point
Cloud Data for Forestry Purposes

Version 1.6.2

Date 2026-1-19

Author Juan Alberto Molina-Valero [aut, cph, cre],
Adela Martínez-Calvo [aut, com],
Juan Gabriel Álvarez-González [aut, ths],
Fernando Montes [aut],
César Pérez-Cruzado [aut, ths]

Maintainer Juan Alberto Molina-Valero <jamolinavalero@gmail.com>

Description Process automation of point cloud data derived from terrestrial-based technologies such as Terrestrial Laser Scanner (TLS) or Mobile Laser Scanner. 'FORTLS' enables (i) detection of trees and estimation of tree-level attributes (e.g. diameters and heights), (ii) estimation of stand-level variables (e.g. density, basal area, mean and dominant height), (iii) computation of metrics related to important forest attributes estimated in Forest Inventories at stand-level, and (iv) optimization of plot design for combining TLS data and field measured data. Documentation about 'FORTLS' is described in Molina-Valero et al. (2022, <doi:10.1016/j.envsoft.2022.105337>).

URL https://molina-valero.github.io/FORTLS/,

https://github.com/Molina-Valero/FORTLS

BugReports https://github.com/Molina-Valero/FORTLS/issues

License GPL-3

Depends R (>= 4.1.0)

Imports data.table, dbscan, Distance, htmlwidgets, lidR, methods,
moments, parallel, plotly, progress, raster, Rcpp(>= 1.0.10),
RCSF, reticulate, scales, sf, tidyr, VoxR, vroom

Suggests devtools, kableExtra, knitr, rmarkdown, systemfonts, testthat

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

StagedInstall yes

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-01-19 10:10:02 UTC

Contents

FORTLS-package   *FORTLS: Automatic Processing of Terrestrial-Based Technologies Point Cloud Data for Forestry Purposes*

Description

Process automation of point cloud data derived from terrestrial-based technologies such as Terrestrial Laser Scanner (TLS) or Mobile Laser Scanner. 'FORTLS' enables (i) detection of trees and estimation of tree-level attributes (e.g. diameters and heights), (ii) estimation of stand-level variables (e.g. density, basal area, mean and dominant height), (iii) computation of metrics related to important forest attributes estimated in Forest Inventories at stand-level, and (iv) optimization of plot design for combining TLS data and field measured data. Documentation about 'FORTLS' is described in Molina-Valero et al. (2022, <doi:10.1016/j.envsoft.2022.105337>).

Details

Usage of **FORTLS** includes the following functionalities:

- Tree detection: this is the first and necessary step for the other functionalities of **FORTLS**. This can be achieved using the following functions:

1. `normalize`: mandatory first step for obtaining the relative coordinates of a TLS point cloud.
2. `tree.detection.single.scan`: detects as many trees as possible from a normalized TLS single-scan point clouds.
3. `tree.detection.multi.scan`: detects as many trees as possible from a normalized TLS multi-scan, SLAM, or similar terrestrial-based technologies point clouds.
4. `tree.detection.several.plots`: includes the two previous functions for a better workflow when there are several plots to be sequentially analyzed.

- Estimation of variables when no field data are available: this is the main functionality of **FORTLS** and can be achieved using the following functions:

1. `distance.sampling`: optional function which can be used for considering methodologies for correcting occlusion effects in estimating variables.
2. `estimation.plot.size`: enables the best plot design to be determined on the basis of TLS data only.
3. `metrics.variables`: is used for estimating metrics and variables potentially related to forest attributes at stand level.

- Estimation of variables when field data are available: this is the main and most desirable functionality of **FORTLS** and can be achieved using the following functions:

1. `distance.sampling`: as before.
2. `simulations`: computes simulations of TLS and field data for different plot designs. This is a prior step to the next functions.
3. `relative.bias`: uses `simulations` output to assess the accuracy of direct estimations of variables according to homologous TLS and field data.
4. `correlations`: uses `simulations` output to assess correlations among metrics and variables obtained from TLS data, and variables of interest estimated from field data.
5. `optimize.plot.design`: using `correlations` output, represents the best correlations for variables of interest according to the plot design. It is thus possible to select the best plot design for estimating forest attributes from TLS data.
6. `metrics.variables`: as before, but in this case plot parameters will be choosen on the basis of field data and better estimates will therefore be obtained.

Author(s)

Maintainer: Juan Alberto Molina-Valero <juanalberto.molina.valero@usc.es> [copyright holder]

Authors:

- María José Ginzo Villamayor [contributor]
- Manuel Antonio Novo Pérez [contributor]
- Adela Martínez-Calvo [contributor]
- Juan Gabriel Álvarez-González [contributor]
- Fernando Montes [contributor]
- César Pérez-Cruzado [contributor]

References

Molina-Valero J. A., Ginzo-Villamayor M. J., Novo Pérez M. A., Martínez-Calvo A., Álvarez-González J. G., Montes F., & Pérez-Cruzado C. (2019). FORTLS: an R package for processing TLS data and estimating stand variables in forest inventories. *The 1st International Electronic Conference on Forests — Forests for a Better Future: Sustainability, Innovation, Interdisciplinarity*. doi:10.3390/IECF202008066

---

correlations	*Correlation Between Field Estimations and TLS Metrics*

Description

Computes correlations between variables estimates from field data and metrics derived from TLS data. Field estimates and TLS metrics for a common set of plots are required in order to compute correlations. These data must be obtained from any of the three different plot designs currently available (fixed area, k-tree and angle-count), and correspond to plots with incremental values for the plot design parameter (radius, k and BAF, respectively). Two correlation measures are implemented: Pearson's correlation coefficient and Spearman's *rho*. In addition to estimating these measures, tests for association are also executed, and interactive line charts graphically representing correlations are generated.

Usage

```
correlations(simulations,
             variables = c("N", "G", "d", "dg", "d.0", "h", "h.0"),
             method = c("pearson", "spearman"), save.result = TRUE,
             dir.result = NULL)
```

Arguments

simulations   List including estimated variables based on field data and metrics derived from
                   TLS data. The structure and format must be analogous to output returned by the
                   simulations function. Specifically, it must have at least one of the following
                   named elements:

- fixed.area.plot: data frame with field estimates and TLS metrics under
  a circular fixed area plot design. Each row corresponds to a (plot, radius)
  pair, and all or any of the following columns are included:
  Plot identification and radius:
  - id, radius: same description and format as indicated for same named
    columns of fixed.area.plot in simulations 'Value'.

  Variables estimated on the basis of simulated field plots:
  - N, G, V, d, dg, dgeom, dharm, h, hg, hgeom, hharm, d.0, dg.0, dgeom.0,
    dharm.0, h.0, hg.0, hgeom.0, hharm.0: same description and for-
    mat as indicated for same named columns of fixed.area.plot in
    simulations 'Value'.

  TLS metrics derived from simulated TLS plots:

- N.tls, N.hn, N.hr, N.hn.cov, N.hr.cov, N.sh, num.points, num.points.est, num.points.hom, num.points.hom.est, G.tls, G.hn, G.hr, G.hn.cov, G.hr.cov, G.sh, V.tls, V.hn, V.hr, V.hn.cov, V.hr.cov, V.sh, d.tls, dg.tls, dgeom.tls, dharm.tls, h.tls, hg.tls, hgeom.tls, hharm.tls, d.0.tls, dg.0.tls, dgeom.0.tls, dharm.0.tls, h.0.tls, hg.0.tls, hgeom.0.tls, hharm.0.tls, P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99: same description and format as indicated for same named columns of fixed.area.plot in simulations 'Value'.

If the fixed.area.plot element is included in simulations argument, it must contain at least id and radius columns, one of the field estimates columns and one of the TLS metrics columns.

- k.tree.plot: data frame with field estimates and TLS metrics under the k-tree plot design. Each row corresponds to a (plot, k) pair, and all or any of the following columns are included:
  Plot identification and k:

  - id, k: same description and format as indicated for same named columns of k.tree.plot in simulations 'Value'.

  Variables estimates on the basis of simulated field plots:

  - N, G, V, d, dg, dgeom, dharm, h, hg, hgeom, hharm, d.0, dg.0, dgeom.0, dharm.0, h.0, hg.0, hgeom.0, hharm.0: same description and format as indicated for same named columns of k.tree.plot in simulations 'Value'.

  TLS metrics derived from simulated TLS plots:

  - N.tls, N.hn, N.hr, N.hn.cov, N.hr.cov, N.sh, num.points, num.points.est, num.points.hom, num.points.hom.est, G.tls, G.hn, G.hr, G.hn.cov, G.hr.cov, G.sh, V.tls, V.hn, V.hr, V.hn.cov, V.hr.cov, V.sh, d.tls, dg.tls, dgeom.tls, dharm.tls, h.tls, hg.tls, hgeom.tls, hharm.tls, d.0.tls, dg.0.tls, dgeom.0.tls, dharm.0.tls, h.0.tls, hg.0.tls, hgeom.0.tls, hharm.0.tls, P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99: same description and format as indicated for same named columns of k.tree.plot in simulations 'Value'.

If a k.tree.plot element is included in the simulations argument, it must include at least id and k columns, one of the field estimate columns, and one of the TLS metrics columns.

- angle.count.plot: data frame with field estimates and TLS metrics under the angle-count plot design. Each row corresponds to a (plot, BAF) pair, and all or one of the following columns are included:
  Plot identification and BAF:

  - id, BAF: same description and format as indicated for same named columns of angle.count.plot in simulations 'Value'.

  Variables estimated on the basis of simulated field plots:

  - N, G, V, d, dg, dgeom, dharm, h, hg, hgeom, hharm, d.0, dg.0, dgeom.0, dharm.0, h.0, hg.0, hgeom.0, hharm.0: same description and format as indicated for same named columns of angle.count.plot in simulations 'Value'.

TLS metrics derived from simulated TLS plots:

- `N.tls, N.pam, num.points, num.points.est, num.points.hom, num.points.hom.est, G.tls, G.pam, V.tls, V.pam, d.tls, dg.tls, dgeom.tls, dharm.tls, h.tls, hg.tls, hgeom.tls, hharm.tls, d.0.tls, dg.0.tls, dgeom.0.tls, dharm.0.tls, h.0.tls, hg.0.tls, hgeom.0.tls, hharm.0.tls, P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99`: same description and format as indicated for same named columns of `angle.count.plot` in [`simulations`](#) 'Value'.

  If the `angle.count.plot` element is included in the `simulations` argument, it must contain at least `id` and `BAF` columns, one of the field estimates columns and one of the TLS metrics columns.

variables   Optional character vector naming field estimates for which correlations between these and all the available TLS metrics will be computed. If this argument is specified by the user, it must include at least one of the following character strings: "N", "G", "V", "d", "dg", "dgeom", "dharm", "d.0", "dg.0", "dgeom.0", "dharm.0", "h", "hg", "hgeom", "hharm", "h.0", "hg.0", "hgeom.0", or "hharm.0". If this argument is not specified by the user, it will be set to
`c("N", "G", "V", "d", "dg", "d.0", "h", "h.0")` by default. In both cases all the elements in the `simulations` argument must include at least the columns corresponding to the field estimates specified in the `variables` argument.

method   Optional character vector naming which correlation measurements will be used. If this argument is specified by the user, it must include at least one of the following character strings: "pearson" or "spearman". If this argument is not specified by the user, it will be set to `c("pearson", "spearman")` by default.

save.result   Optional logical indicating wheter or not the output files described in 'Output Files' section must be saved in `dir.result` or not. If this argument is not specified by the user, it will be set to `TRUE` by default and, as a consequence, the output files will be saved.

dir.result   Optional character string naming the absolute path of an existing directory where files described in the 'Output Files' section will be saved. `.Platform$file.sep` must be used as the path separator in `dir.result`. If this argument is not specified by the user, and `save.result` is `TRUE`, it will be set to `NULL` by default and, as consequence, the current working directory of the R process will be assigned to `dir.result` during execution.

Details

For each radius, k or BAF value (according to the currently available plot designs: circular fixed area, k-tree and angle-count), this function computes correlations between each variable estimated from field data specified in the `variables` argument and all the metrics derived from TLS data existing in the data frames included in the `simulations` argument.

Two correlation measures are implemented at present: Pearson's correlation coefficient and Spearman's *rho*. For each method, in addition to the estimated measure, the p-value of a test for association is also returned. The [`cor.test`](#) function from the **utils** package is used to compute both the estimated correlations and the p-values of the associated tests; more details about these measures and their tests for association can be found in the corresponding documentation. There cannot be

missing data for three or more plots, and there cannot be zero standard deviation, in order to prevent missing correlation values for each (field estimation, TLS metric) pair and plot design parameter (radius, k or BAF).

Apart from estimated correlations and their corresponding p-values, for ecah method, the function also returns the plot design parameter and field estimates, the value of the optimal correlation (i.e. the maximum of the absolute value of available correlations) and the TLS metric to which it corresponds.

Value

correlations   A list including the estimated correlations for each measure specified in the `method` argument will be generated. This will include all or any of the following named elements:

- pearson: if "pearson" is not included in `method` parameter, missing; otherwise, the list will include the estimated Pearson's correlations for each plot design specified in the `simulations` argument. In the latter case, the list will include all or any of the following named elements:
  - fixed.area.plot: if the `simulations` argument does not have an element named "fixed.area.plot", missing; otherwise, the matrix will include the estimated Pearson's correlations for a circular fixed area plot design. Each row will correspond to a radius value, and the following columns will be included:
    * radius: radius (m) of the simulated plots used for computing the estimated correlations.
    * Column(s) '<x>.<y>': numeric column(s) containing estimated Pearson's correlations between '<x>', a field estimate, and '<y>', a TLS metric.
  - k.tree.plot: if the `simulations` argument does not include an element named "k.tree.plot", missing; otherwise, the matrix will include the estimated Pearson's correlations for a k-tree plot design. Each row will correspond to a k value, and the following columns will be included:
    * k: number of trees (trees) of the simulated plots used for computing the estimated correlations.
    * Column(s) '<x>.<y>': same description and format as indicated in correlations$pearson$fixed.area.plot element.
  - angle.count.plot: if the `simulations` argument does not have any element named "angle.count", missing; otherwise, the matrix will include the estimated Pearson's correlations for the angle-count plot design. Each row will correspond to a BAF value, and the following columns will be included:
    * BAF: BAF (m^2/ha) of the simulated plots used for computing the estimated correlations.
    * Column(s) '<x>.<y>': same description and format as indicated in correlations$pearson$fixed.area.plot element.
- spearman: if "spearman" is not included in `method` parameter, missing; otherwise, the list will include the estimated Spearman's correlations for

each plot design specified in `simulations` argument. In the latter case, the structure and format will be analogous to that indicated for the previous element but estimated Pearson's correlations will be replaced by Spearman's correlations.

correlations.pval

List containing the p-value of the test for association corresponding to each measure specified in `method` argument. The structure and format will be the same as indicated for the previous element but estimated correlations will be replaced by p-values for their corresponding tests for association.

opt.correlations

List containing the optimal correlations, and the names of the TLS metrics to which they correspond, for each measure specified in `method` argument. The list will will include all or any of the following named elements:

- `pearson`: if "pearson" is not included in `method` parameter, missing; otherwise, the list will include the optimal Pearson's correlations, and the names of the TLS metrics to which they correspond, for each plot design specified in `simulations` argument. In the latter case, it will include all or any of the following named elements:
  - `fixed.area.plot`: if `simulations` argument does not have any element named "`fixed.area.plot`", missing; otherwise, the data frame will include the optimal Pearson's correlations, and the names of the TLS metrics to which they correspond, for a circular fixed area plot design. Each row will correspond to a radius value, and the following columns will be included:
    * `radius`: radius (m) of the simulated plots used for computing the estimated correlations.
    * Columns '`<x>.cor`' and '`<x>.metric`': the former, numeric column(s) including optimal Pearson's correlations between '`<x>`', a field estimate, and all the available TLS metrics; and the latter, character column(s) will include names of the TLS metrics to which they correspond.
  - `k.tree.plot`: if the `simulations` argument does not have any element named "`k.tree.plot`", missing; otherwise, the data frame will include the optimal Pearson's correlations and the names of the TLS metrics to which they correspond for the k-tree plot design. Each row will correspond to a k value, and the following columns will be included:
    * `k`: number of trees (trees) of the simulated plots used for computing the estimated correlations.
    * Columns '`<x>.cor`' and '`<x>.metric`': same description and format as indicated in `opt.correlations$pearson$fixed.area.plot` element.
  - `angle.count.plot`: if the `simulations` argument does not have any element named "`angle.count`", missing; otherwise, the data frame will include the optimal Pearson's correlations, and the names of the TLS metrics to which they correspond for the angle-count plot design. Each row will correspond to a BAF value, and the following columns will be included:

* BAF: BAF (m^2/ha) of the simulated plots used for computing the estimated correlations.

* Columns '`<x>.cor`' and '`<x>.metric`': same description and format as indicated in `opt.correlations$pearson$fixed.area.plot` element.

- spearman: if "`spearman`" is not included in `method` parameter, missing; otherwise, the list will include the optimal Spearman's correlations, and the names of the TLS metrics to which they correspond, for each plot design specified in `simulations` argument. In the latter case, the structure and format will be analogous to that indicated for the previous element, but optimal Pearson's correlations will be replaced by Spearman's correlations.

Output Files

During the execution, if the `save.result` argument is `TRUE`, the function prints to files the matrices and data frames included in `correlations` and `opt.correlations` elements described in 'Value'. Both are written without row names in `dir.result` directory by using the [write.csv](#) function in the **utils** package. The patterns used for naming these files are '`correlations.<plot design>.<method>.csv`' and '`opt.correlations.<plot design>.plot.<method>.csv`' for correlation matrices and optimal correlation data frames, respectively, where '`<plot design>`' is equal to "`fixed.area.plot`", "`k.tree.plot`" or "`angle.count.plot`" according to plot design, and '`<method>`' equals "`pearson`" or "`spearman`" according to the correlation measure.

Furthermore, if the `save.result` argument is `TRUE`, interactive line charts graphically representing correlations will also be created and saved in the `dir.result` directory by means of [saveWidget](#) function in the **[htmlwidgets](#)** package. Generated widgets enable users to consult correlation data directly on the plots, select/deselect different sets of traces, to zoom and scroll, etc. The pattern used for naming these files is '`correlations.<x>.<plot design>.<method>.html`', where both '`<plot design>`' and '`<method>`' are as indicated for the previous described files, and '`<x>`' is equal to any of elements specified in the `variables` argument.

Note

This function is particularly useful for further steps related to model-based and model-assisted approaches, as correlations measure the strength of a relationship between two variables (linear for Pearson's correlation, monotonic for Spearman's correlation).

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

See Also

[simulations](#), [optimize.plot.design](#).

[cor.test](#) in **utils** package.

Examples

```
# Load field estimates and TLS metrics corresponding to Rioja data set
```

```
data("Rioja.simulations")


# Establish directory where correlation results corresponding to the Rioja example
# will be saved. For instance, current working directory

# dir.result <- tempdir()


# Compute correlations between field estimates and TLS metrics corresponding
# to Rioja example
# Pearson's and Spearman's correlations for variables by default

# corr <- correlations(simulations = Rioja.simulations, dir.result = dir.result)


# Pearson's and Spearman's correlations for variable 'N'

# corr <- correlations(simulations = Rioja.simulations, variables = "N",
#                      dir.result = dir.result)


# Only Pearson's correlations for variables by default

# corr <- correlations(simulations = Rioja.simulations, method = "pearson",
#                      dir.result = dir.result)


# Pearson's and Spearman's correlations corresponding to angle-count design for
# all available variables

# corr <- correlations(simulations = Rioja.simulations["angle.count"],
#                      variables <- c("N", "G", "V", "d", "dg", "dgeom", "dharm",
#                                      "d.0", "dg.0", "dgeom.0", "dharm.0", "h",
#                                      "hg", "hgeom", "hharm", "h.0", "hg.0",
#                                      "hgeom.0", "hharm.0"),
#                      dir.result = dir.result)
```

---

distance.sampling   *Distance Sampling Methods for Correcting Occlusions Effects*

Description

Calculation of the probability of detection of every tree by using distance sampling methodologies (more specifically point transects methods), by means of fitting detection functions to the histogram

of tree distribution according to their distance to TLS. Use both half normal and hazard rate functions, without and with *dbh* as covariate. These probabilities are used for correcting estimation bias caused by lack of detection of trees due to occlusion.

Usage

```
distance.sampling(tree.tls,
                  id.plots = NULL,
                  strata.attributes = NULL)
```

Arguments

tree.tls   Data frame with a list of trees detected and their *dbh* and horizontal distances from TLS with the same structure and format as `tree.detection.single.scan` and `tree.detection.several.plots` 'Value'.

id.plots   Optional vector with plot identification encoded as character string or numeric for the plots considered. In this case, `tree.tls` argument must include a common column named 'id'. If this argument is not specified by the user, it will be set to NULL by default, and as a consequence, all plots will be considered.

strata.attributes

Optional data frame inluding plot radius considered at strata level. It must contain a column named 'stratum' (numeric) with encoding coinciding with that used in previous functions (`normalize`, `tree.detection.single.scan` and `tree.detection.several.plots`) for identifying strata. Therefore, strata must heve been included previously in 'tree.tls'. Another column named 'plot.radius' (numeric) will be required to set maximum horizontal distance (m) considered for fitting detection probability functions. If this argument is not specified by the user, it will be set to NULL by default, and as a consequence, all trees will be included.

Details

All internal functions related to distance sampling methodologies are fitted with the ds function included in the **Distance** package.

Detection functions are left-truncated at 1 m, according to Astrup et al., (2014).

Same warning messages as ds function are provided when fits do not converge or another warnings occur.

For further details on these point transects methods and similar sampling methodologies, as well as their application with R, see Buckland et al., (2001); Marques & Buckland, (2003); Miller & Thomas, (2015) and Clark (2016). Examples of distance sampling analyses, as well as lectures, are available at https://distancesampling.org/resources/vignettes.html.

Value

List containing the following elements:

tree   Data frame with detection probabilities for every tree and method.

- stratum: stratum identification (coincident with strata of `tree.tls`). If there are not strata, it will be set as a single stratum encoded as 1 (numeric).
- id: plot identification (coincident with id of `tree.tls`).
- tree: tree numbering (coincident with tree of `tree.tls`).
- P.hn: tree detection probability according to half normal function.
- P.hn.cov: tree detection probability according to half normal function with *dbh* as covariate.
- P.hr: tree detection probability according to half rate function.
- P.hr.cov: tree detection probability according to half rate function with *dbh* as covariate.

parameters   Data frame with parameters estimated for detection functions (see references for understanding their meaning).

- P.hn.scale: scale parameter for half normal function (sigma).
- P.hn.cov.scale.intercept: alpha.0 parameter of scale parameter for half normal function with *dbh* as covariate.
- P.hn.cov.dbh: alpha.1 parameter of scale parameter for half normal function with *dbh* as covariate.
- P.hr.scale: scale parameter for half rate function (sigma).
- P.hr.shape: shape parameter for half rate function (b).
- P.hr.cov.scale.intercept: alpha.0 parameter of scale parameter for half normal function with *dbh* as covariate.
- P.hr.cov.dbh: alpha.1 parameter of scale parameter for half normal function with *dbh* as covariate.
- P.hr.cov.shape: shape parameter for half rate function with *dbh* as covariate (b).

AIC   Data frame with Akaike information criterions (AIC) of every detection function fit.

- P.hn: AIC of half normal function fit.
- P.hn.cov: AIC of half normal function with *dbh* as covariate fit.
- P.hr: AIC of half rate function fit.
- P.hr.cov: AIC of half rate function with *dbh* as covariate fit.

Note

Although this step is optional for other functionalities of **FORTLS**, such as obtaining metrics and assessing the best plot designs (implemented in `metrics.variables`, `correlations`, `relative.bias` and `optimize.plot.design`), its inclusion is highly recommended, especially with high rates of occlusions.

Note that this function could be more useful after assessing the best possible plot design with `estimation.plot.size`, `correlations`, `relative.bias` or `optimize.plot.design` functions.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

References

Astrup, R., Ducey, M. J., Granhus, A., Ritter, T., & von Lüpke, N. (2014). Approaches for estimating stand-level volume using terrestrial laser scanning in a single-scan mode. *Canadian Journal of Forest Research*, **44(6)**, 666-676. doi:10.1139/cjfr20130535.

Buckland, S. T., Anderson, D. R., Burnham, K. P., Laake, J. L., Borchers, D. L., & Thomas, L. (2001). *Introduction to distance sampling: estimating abundance of biological populations*, Oxford, United Kindown, Oxford University Press.

Clark, R. G. (2016). Statistical efficiency in distance sampling. *PloS one*, **11(3)**, e0149298. doi:10.1371/journal.pone.0149298.

Marques, F. F., & Buckland, S. T. (2003). Incorporating covariates into standard line transect analyses. *Biometrics*, **59(4)**, 924-935. doi:10.1111/j.0006341X.2003.00107.x.

Miller, D. L., & Thomas, L. (2015). Mixture models for distance sampling detection functions. *PloS one*, **10(3)**, e0118726. doi:10.1371/journal.pone.0118726.

See Also

tree.detection.single.scan, tree.detection.several.plots, metrics.variables, simulations. ds in **Distance** package.

Examples

```
# Loading example data

data(Rioja.data)

tree.tls <- Rioja.data$tree.tls

# Whithout considering maximum distance

ds <- distance.sampling(tree.tls)
```

---

estimation.plot.size   *Assess Consistency of Metrics for Simulated TLS Plots*

Description

Plots empirical linear charts of density (*N*, trees/ha) and basal area (*G*, m^2/ha) estimates (derived from simulated TLS plots) as a function of plot size (estimation-size charts) for different plot designs (circular fixed area, k-tree and angle-count), through continuous size increments (radius, k and BAF respectively). Size increments are set at 0.1 m, 1 tree and 0.1 m^2/ha for fixed area, k-tree

and angle-count plot designs, respectively. These size-estimation line charts represent the consistency in predicting the stand variables across different values of radius, k and BAF. Size-estimation charts can be drawn for individual sample plots (including all plots together in the same charts) or for mean values (global mean computed for all the sample plots, or for group means if different strata are considered). Finally, different plot designs can be compared if specified in the arguments, producing one size-estimation chart per variable (*N* and *G*).

Usage

```
estimation.plot.size(tree.tls,
                     plot.parameters = data.frame(radius.max = 25,
                                                  k.max = 50,
                                                  BAF.max = 4),
                     dbh.min = 4,
                     average = FALSE, all.plot.designs = FALSE)
```

Arguments

tree.tls   Data frame with information of trees detected from TLS point cloud data in the same format as `tree.detection.single.scan` and `tree.detection.multi.scan` 'Value'.

plot.parameters

Optional data frame containing parameters for circular fixed area, k-tree and angle-count plot designs. The parameters are as follows:

- radius.max: maximum plot radius (m) considered for circular fixed area plots. If the radius.max specified is larger than the farthest tree from the plot centre, the horizontal distance from the farthest tree will be considered the maximum radius. By default, the radius.max will be 25 m.

- k.tree.max: maximum number of trees considered for k-tree plots. If k.tree.max specified is larger than the maximum number of trees of the densest plot, this number of trees will be considered the maximum k.tree.max. By default, k.tree.max is 50.

- BAF.max: maximum basal area factor (m^2/ha) considered for angle-count plots. By default, BAF.max is 4.

dbh.min   Optional minimum *dbh* (cm) considered for detecting trees. By default it will be set at 4 cm.

average   Logical; if TRUE, plot means values and standard deviation of estimations will be represented. By default, it will be set as FALSE.

all.plot.designs

Logical; if TRUE, charts for each plot design are drawn together. By default, it will be set as FALSE.

Details

If there are strata in the tree.tls argument, they will be differentiated in charts with different colours. Strata must be specified in a numeric column named stratum.

The `all.plot.designs` argument only works for single strata, and therefore if there are additional strata in the `tree.tls` argument, they will be considered equal.

The outputs of this function are inspired by Fig. 3 of Brunner and Gizachew (2014).

Value

`Invisible NULL`

Note

Mean values are relevant when plots are representing homogenous strata.

Note that this is an option for choosing the best plot design when field data are not available. Otherwise, using `correlations`, `relative.bias` and `optimize.plot.design` will be more desirable for obtaining the best possible plot design.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

References

Brunner, A., & Gizachew, B. (2014). Rapid detection of stand density, tree positions, and tree diameter with a 2D terrestrial laser scanner. *European Journal of Forest Research*, **133(5)**, 819-831.

See Also

`tree.detection.single.scan`, `tree.detection.multi.scan`, `tree.detection.several.plots`

Examples

```
# Loading dataset with trees detected from TLS single-scans

data("Rioja.data")

tree.tls <- Rioja.data$tree.tls


# Without strata and plot parameters by default

estimation.plot.size(tree.tls)

estimation.plot.size(tree.tls, average = TRUE)

estimation.plot.size(tree.tls, all.plot.designs = TRUE)
```

---

install_fortls_python_deps

Install Python dependencies required by FORTLS

Description

Install Python dependencies required by FORTLS

Usage

```
install_fortls_python_deps(envname = "fortls_env", method = "auto")
```

Arguments

envname	Name of the virtual environment or conda environment.
method	Either "virtualenv", "conda", or "auto".

---

metrics.variables   *Compute Metrics and Variables for Terrestrial-Based Technologies Point Clouds*

Description

This function computes a set of metrics and variables from the Terrestrial-Based Technologies point cloud data, which have a high potential to be related or used as direct estimates (in the case of variables) of forest attributes at plot level. These can be obtained for different plot designs (circular fixed area, k-tree and angle-count plots). This function also includes methodologies for correcting occlusions generated in TLS single-scan point clouds.

Usage

```
metrics.variables(tree.tls, tree.ds = NULL, tree.field = NULL,
                  plot.design = c("fixed.area", "k.tree", "angle.count"),
                  plot.parameters = data.frame(radius = 25, k = 10, BAF = 2),
                  scan.approach = "single",
                  var.metr = list(tls = NULL, field = NULL),
                  v.calc = "parab", dbh.min = 4, h.min = 1.3,
                  max.dist = Inf, dir.data = NULL,
                  save.result = TRUE, dir.result = NULL)
```

Arguments

tree.tls	Data frame with information about trees detected from terrestrial-based technologies point clouds data in the same format as [tree.detection.single.scan](#) or [tree.detection.multi.scan](#) 'Values'.
tree.ds	Optional list containing detection probabilities of trees with distance sampling methods. The format must be the same as the 'Value' obtained with [distance.sampling](#). If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, TLS metrics using distance sampling based correction will not be calculated for circular fixed area and k-tree plot designs.
tree.field	Data frame with information about trees measured in the field plots. Each row must correspond to a (plot, tree) pair, and it must include at least the following columns:

- id: plot identification encoded as character string or numeric. Plot identifications must coincide with those included in id column of the tree.tls argument.
- tree: trees numbering.
- h.dist: horizontal distance (m) from plot centre to tree centre. Centres of the field plots must coincide with centres of their corresponding Terrestrial-based technologies plots.
- dbh: tree diameter (cm) at breast height (1.3 m).
- h: tree total height (m).
- h.blc: height based live crown (m) (optional).
- h.bdc: height based dead crown (m) (optional).
- v.user: tree volume (m^3) (optional).
- w.user: tree biomass (Mg) (optional).
- dead: integer value indicating for each tree if it is dead (1) or not (NA).

plot.design	Vector containing the plot designs considered. By default, all plot designs will be considered (circular fixed area, k-tree and angle-count plots).
plot.parameters	

Data frame containing parameters for circular fixed area, k-tree and angle-count plot designs. If there is a stratum column in the tree.list.tls argument, it must have the same number of rows as strata values and they must be named using strata encoding. If plot parameters are not specified, the corresponding plot designs will not be considered in the function. If no parameter is specified, the function will stop giving an error message! The parameters are as follows:

- radius: plot radius (m) considered for circular fixed area plots. Absence of this argument rules out this plot design.
- k.tree: number of trees (trees) considered for k-tree plots. Absence of this argument rules out this plot design.
- BAF: basal area factor (m^2/ha) considered for angle-count plots. Absence of this argument rules out this plot design.
- num.trees: number of dominant trees per ha (tree/ha), i.e. those with largest *dbh*, considered for calculating dominant diameters and heights. In the absence of this argument, the number will be assumed to be 100 trees/ha.

scan.approach   Character parameter indicating TLS single-scan ('single') or TLS multi-scan
                    approach or SLAM point clouds ('multi') approaches. If this argument is not
                    specified by the user, it will be set to 'multi' approach.

var.metr   Optional vector containing all the metrics and variables of interest. By default
                    it will be set as NULL and thus, all the metrics and variables available will be
                    generated.

v.calc   Optional parameter to calculate volume when is not included in tree.tls input
                    data.

dbh.min   Optional minimum *dbh* (cm) considered for detecting trees. By default it will
                    be set at 4 cm.

h.min   Optional minimum *h* (m) considered for detecting trees. By default it will be set
                    at 1.3 m.

max.dist   Optional argument to specify the maximum horizontal distance considered in
                    which trees will be included.

dir.data   Optional character string naming the absolute path of the directory where TXT
                    files containing TLS point clouds are located. .Platform$file.sep must be
                    used as the path separator in dir.data, and TXT files in the directory must
                    have the same description and format as indicated for TXT files in normalize
                    'Output Files'. If this argument is not specified by the user, it will be set to NULL
                    by default and, as consequence, the current working directory of the R process
                    will be assigned to dir.dat during the execution.

save.result   Optional logical which indicates whether or not the output files described in
                    'Output Files' section must be saved in dir.result. If this argument is not
                    specified by the user, it will be set to TRUE by default and, as consequence, the
                    output files will be saved.

dir.result   Optional character string naming the absolute path of an existing directory where
                    files described in 'Output Files' section will be saved. .Platform$file.sep
                    must be used as the path separator in dir.result. If this argument is not spec-
                    ified by the user, and save.result is TRUE, it will be set to NULL by default
                    and, as a consequence, the current working directory of the R process will be
                    assigned to dir.result during the execution.

Details

This function also works for several plots. In this case, a column named "id" to identify plots
(string character or numeric) must be included in the tree.list.tls database argument. This
must coincide with the corresponding "id" assigned in normalize to TXT files saved in dir.data
(for more details see normalize). In addition, if there are several strata, they can be processed
separately according to plot.parameters values (where each row will represent one stratum). If
tree.list.tls does not include a specific "stratum" column, it will be assumed to have only one
stratum, which will be encoded according to rownames(plot.parameters)[1].

Using TLS data, this function computes metrics and estimations of variables (see 'Value') for plot
design specified in the plot.parameters argument. The notation used for variables is based on
IUFRO (1959).

At this stage, three plot designs are available:

- Circular fixed area plots, simulated only if a `radius` value is specified in the `plot.parameters` argument.

- k-tree plots, simulated only if a `k.tree` value is specified in the `plot.parameters` argument.

- Angle-count plots, simulated only if a `BAF` value is specified in the `plot.parameters` argument.

Volume is estimated modelling stem profile as a paraboloid and calculating the volumes of revolution; where trees *dbh* are estimated in `tree.detection.single.scan` and `tree.detection.multi.scan`, and total heights are estimated as percentile 99 of z coordinate of points delimited by Voronoi polygons.

Regarding occlusion corrections for estimating variables, apart from distance sampling methods considered in `distance.sampling`, another occlusion corection based on correcting the shadowing effect (Seidel & Ammer, 2014) has been included to estimate some variables in circular fixed area and k-tree plots. In the case of angle-count plots, occlusion corrections are based on gap probability attenuation with distance to TLS depending on Poisson model (Strahler et al., 2008; Lovell et al., 2011).

Value

List including TLS-based metrics and variables computed for plot designs considered. The list will contain one element per plot design considered (fixed.area.plot, k.tree.plot and angle.count.plot):

fixed.area.plot

If no value for `plot.radius` is specified in the `plot.parameters` argument, NULL; otherwise, the data frame will include TLS metrics and variables estimated in a circular fixed area plot design (rows represent simulations). The data frame will have the following columns:

Stratum, plot identification and radius:

- `stratum`: stratum identification encoded as a character string or numeric. It must coincide with those included in the `stratum` column of `tree.list.tls`.

- `id`: plot identification encoded as character string or numeric. It will coincide with those included in the `id` column of `tree.list.tls` and, if applicable, the `distance.sampling` argument.

- `radius`: radius (m) of the simulated plot.

Terrestrial-Based Technologies variables:

- `N.tls`: stand density (trees/ha) without occlusion corrections.

- `N.hn`, `N.hr`, `N.hn.cov`, `N.hr.cov`: stand density (trees/ha) with occlusion corrections based on distance sampling methodologies (see `distance.sampling`). These columns will not appear if the `distance.sampling` argument is NULL.

- `N.sh`: stand density (trees/ha) with correction of the shadowing effect. respectively).

- `G.tls`: stand basal area (m^2/ha) without occlusion corrections.

- `G.hn`, `G.hr`, `G.hn.cov`, `G.hr.cov`: stand basal area (m^2/ha) with occlusion corrections based on distance sampling methodologies (see `distance.sampling`). These columns are missing if `distance.sampling` argument is NULL.

- `G.sh`: stand basal area (m^2/ha) with correction of the shadowing effect.
- `V.tls`: stand volume (m^3/ha) without occlusion corrections.
- `V.hn`, `V.hr`, `V.hn.cov`, `V.hr.cov`: stand volume (m^3/ha) with occlusion corrections based on distance sampling methodologies (see distance.sampling). These columns will be missing if the distance.sampling argument is NULL.
- `V.sh`: stand volume (m^3/ha) with correction of the shadowing effect.
- `d.tls`, `dg.tls`, `dgeom.tls`, `dharm.tls`: mean tree diameters (cm) at breast height (1.3 m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `h.tls`, `hg.tls`, `hgeom.tls`, `hharm.tls`: mean tree heights (m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `d.0.tls`, `dg.0.tls`, `dgeom.0.tls`, `dharm.0.tls`: dominant mean tree diameters (cm) at breast height (1.3 m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `h.0.tls`, `hg.0.tls`, `hgeom.0.tls`, `hharm.0.tls`: dominant mean tree heights (m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.

Terrestrial-Based Technologies metrics:

- `n.pts`, `n.pts.est`, `n.pts.red`, `n.pts.red.est`: number of points and estimated number of points (points) belonging to trees normal sections (+/- 5 cm) in the original point cloud (`n.pts` and `n.pts.est`, respectively); and number of points and estimated number of points (points) belonging to trees normal sections (+/- 5 cm) in the reduced point cloud (`n.pts.red` and `n.pts.red.est`, respectively).
- `P01`, `P05`, `P10`, `P20`, `P25`, `P30`, `P40`, `P50`, `P60`, `P70`, `P75`, `P80`, `P90`, `P95`, `P99`: height percentiles derived from z coordinates of TLS point clouds relative to ground level (m).
- `mean.arit.z/rho/phi`, `mean.qua.z/rho/phi`, `mean.geom.z/rho/phi`, `mean.harm.z/rho/phi`, `median.z/rho/phi`, `mode.z/rho/phi`: central tendency statistics for z, rho (horizontal distance) and r (euclidean distance) coordinates (arithmetic, quadratic, geometrical and harmonic means, median and mode, respectively).
- `var.z/rho/phi`, `sd.z/rho/phi`, `CV.z/rho/phi`, `D.z/rho/phi`, `ID.z/rho/phi`, `max.z/rho/phi`, `min.z/rho/phi`: dispersion tendency statistics for z, rho and r coordinates (variance, standar deviation, coefficient of variation, range, interquartile range, maximum and minimum, respectively).
- `kurt.z/rho/phi`, `skw.z/rho/phi`: curtosis and skewness, respectively.
- `L1.z/rho/phi`, `L2.z/rho/phi`, `L3.z/rho/phi`, `L4.z/rho/phi`, `L-CV.z/rho/phi`, `MAD-median.z/rho/phi`, `MAD-mode.z/rho/phi`: L-moments of order 2, 3 and 4, median of the absolute deviations from the overall median and mode of the absolute deviations from the overall mode, respectively.
- `L3.mu.z/rho/phi`, `L4.mu.z/rho/phi`: third and fourth central moments, respectively.

- `PA.2m`, `PA.mean.z`, `PA.mode.z`: percentage of laser returns above 2 m, mean and mode, respectively.
- `PB.2m`, `PB.mean.z`, `PB.median.z`, `PB.mode.z`: percentage of laser returns below 2 m, mean and mode, respectively.
- `weibull.b.z/rho/phi`, `weibull.c.z/rho/phi`: scale and shape parameters, respectively, for Weibull distribution fitted for z coordinates of TLS point clouds relative to ground level.

k.tree.plot
If no value for `k.tree` is specified in the `plot.parameters` argument, NULL; otherwise, the data frame will include TLS metrics and variables estimated in the k-tree plot design (rows represent simulations). The data frame will include the following columns (same description and format as indicated in `fixed.area.plot` element):

Stratum, plot identification and k:

- `stratum`: stratum identification encoded as character string or numeric. It must coincide with those included in the `stratum` column of `tree.list.tls`.
- `id`: plot identification encoded as character string or numeric value. It will coincide with those included in `id` column of `tree.list.tls` or, if applicable, the `distance.sampling` argument.
- `k`: number of trees in the simulated plot.

Terrestrial-Based Technologies variables:

- `N.tls`, `N.hn`, `N.hr`, `N.hn.cov`, `N.hr.cov`, `N.sh`,
  `G.tls`, `G.hn`, `G.hr`, `G.hn.cov`, `G.hr.cov`, `G.sh`,
  `V.tls`, `V.hn`, `V.hr`, `V.hn.cov`, `V.hr.cov`, `V.sh`,
  `d.tls`, `dg.tls`, `dgeom.tls`, `dharm.tls`,
  `h.tls`, `hg.tls`, `hgeom.tls`, `hharm.tls`,
  `d.0.tls`, `dg.0.tls`, `dgeom.0.tls`, `dharm.0.tls`,
  `h.0.tls`, `hg.0.tls`, `hgeom.0.tls`, `hharm.0.tls`

Terrestrial-Based Technologies metrics:

- `num.points`, `num.points.est`, `num.points.hom`, `num.points.hom.est`,
  `P01`,`P05`, `P10`, `P20`, `P25`, `P30`, `P40`, `P50`, `P60`, `P70`, `P75`, `P80`, `P90`, `P95`, `P99`:
  same description and format as indicated in fixed.area.plot element.

angle.count.plot
If no value for `BAF` is specified in the `plot.parameters` argument, NULL; otherwise, the data frame will include TLS metrics and variables estimated in the angle-count plot design (rows represent simulations). The data frame will include the following columns:

Stratum, plot identification and BAF:

- `stratum`: stratum identification encoded as character string or numeric. It must coincide with those included in `stratum` column of `tree.list.tls`.
- `id`: plot identification encoded as character string or numeric. It will coincide with those included in the `id` column of `tree.list.tls`.
- `BAF`: BAF (m^2/ha) of the simulated plot.

Terrestrial-Based Technologies variables:

- `N.tls`: same description and format as indicated in the `fixed.area.plot` element.

- N.pam: stand density (trees/ha) with occlusion correction based on gap probability attenuation with distance to TLS. fixed.area.plot element.
- G.tls: same description and format as indicated for the fixed.area.plot element.
- G.pam: stand basal area (m^2/ha) with occlusion correction based on gap probability attenuation with distance from TLS.
- V.tls: same description and format as indicated for the fixed.area.plot element.
- V.pam: stand volume (m^3/ha) with occlusion correction based on gap probability attenuation with distance from TLS.
- d.tls, dg.tls, dgeom.tls, dharm.tls,
  h.tls, hg.tls, hgeom.tls, hharm.tls,
  d.0.tls, dg.0.tls, dgeom.0.tls, dharm.0.tls,
  h.0.tls, hg.0.tls, hgeom.0.tls, hharm.0.tls

Terrestrial-Based Technologies metrics:

- num.points, num.points.est, num.points.hom, num.points.hom.est, P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99: same description and format as indicated for the fixed.area.plot element.

Output Files

After computing metrics and variables, if the save.result argument is TRUE, the function will save the elements in the list described in 'Value' (fixed.area.plot, k.tree.plot and angle.count.plot), which are different from NULL as CSV files. Data frames are written without row names in the dir.result directory by using write.csv function from the **utils** package. The pattern used for naming these files is 'metrics.variables.<plot design>.csv', being '<plot design>' equal to "fixed.area.plot", "k.tree.plot" or "angle.count.plot" according to the plot design.

Note

In order to optimize plot designs and, therefore, for better use of metrics.variables, other functions such as correlations, relative.bias and estimation.plot.size should be used.

This function will be updated as new metrics are developed.

Author(s)

Juan Alberto Molina-Valero, Adela Martínez-Calvo.

References

IUFRO (1959). Standarization of symbols in forest mensuration. Vienna, Austria, IUFRO.

Lovell, J. L., Jupp, D. L. B., Newnham, G. J., & Culvenor, D. S. (2011). Measuring tree stem diameters using intensity profiles from ground-based scanning lidar from a fixed viewpoint. *ISPRS Journal of Photogrammetry and Remote Sensing*, **66(1)**, 46-55. doi:10.1016/j.isprsjprs.2010.08.006

Seidel, D., & Ammer, C. (2014). Efficient measurements of basal area in short rotation forests based on terrestrial laser scanning under special consideration of shadowing. *iForest-Biogeosciences and Forestry*, **7(4)**, 227. doi:10.3832/ifor1084007

Strahler, A. H., Jupp, D. L. B., Woodcock, C. E., Schaaf, C. B., Yao, T., Zhao, F., Yang, X., Lovell, J., Culvenor, D., Newnham, G., Ni-Miester, W., & Boykin-Morris, W. (2008). Retrieval of forest structural parameters using a ground-based lidar instrument (Echidna®). *Canadian Journal of Remote Sensing*, **34(sup2)**, S426-S440. doi:10.5589/m08046

See Also

tree.detection.single.scan, tree.detection.multi.scan, tree.detection.several.plots, distance.sampling, normalize.

Examples

```
# Establishment of working directories (optional)
# By default here we propose the current working directory of the R process

dir.data <- tempdir()
dir.result <- tempdir()


# Loading example data included in FORTLS

data("Rioja.data")
tree.tls <- Rioja.data$tree.tls
tree.tls <- tree.tls[tree.tls$id == "1", ]

# Download example of TXT file corresponding to plot 1 from Rioja data set

download.file(url = "https://www.dropbox.com/s/w4fgcyezr2olj9m/Rioja_1.txt?dl=1",
              destfile = file.path(dir.data, "1.txt"), mode = "wb")


# Considering distance sampling methods (only for single-scan approaches)

# ds <- distance.sampling(tree.tls)

met.var.TLS <- metrics.variables(tree.tls = tree.tls,
                                 # tree.ds = ds,
                            plot.parameters = data.frame(radius = 10, k = 10, BAF = 2),
                                 dir.data = dir.data, dir.result = dir.result)
```

| normalize | *Relative Coordinates and Density Reduction for Terrestrial-Based Technologies Point Clouds* |

Description

This function obtains coordinates relative to the plot centre speciefied for Terrestrial Laser Scanner (TLS) and Mobile Laser Scanner (MLS) point clouds (supplied as LAS or LAZ files). Point clouds obtained from other devices/approaches (e.g. photogrammetry) can be also used, but the guarantee of good performance is likely to be lower. In addition, the point cropping process developed by Molina-Valero et al., (2019) is applied as a criterion for reducing point density homogeneously in space and proportionally to object size when TLS single-scans are provided.

Usage

```
normalize(las, normalized = NULL,
          x.center = NULL, y.center = NULL,
          x.side = NULL, y.side = NULL,
          xpoly = NULL, ypoly = NULL,
          max.dist = NULL, min.height = NULL, max.height = 50,
          algorithm.dtm = "knnidw", res.dtm = 0.2,
          csf = list(cloth_resolution = 0.5),
          intensity = NULL, RGB = NULL,
          scan.approach = "single",
          voxel_size = NULL,
          threads = 1,
          id = NULL, file = NULL, plot = TRUE,
          dir.data = NULL, save.result = TRUE, dir.result = NULL,
          save.las = NULL)
```

Arguments

las	Character string containing the name of LAS/LAZ file belonging to point cloud, including .las/.laz extensions (see 'Examples'). Planimetric coordinates of point cloud data must be in local, representing TLS scan point the origin with Cartesian coordinates x and y as (0, 0).
normalized	Optional argument to establish as TRUE when point cloud is already normalized.
x.center	Planimetric x center coordinate of point cloud data.
y.center	Planimetric y center coordinate of point cloud data.
x.side	x-side (m) of the plot when the plot is square or rectangular.
y.side	y-side (m) of the plot when the plot is square or rectangular.
xpoly	Vector containing the x-coordinates of the vertices of the clipping polygon.
ypoly	Vector containing the y-coordinates of the vertices of the clipping polygon.
max.dist	Optional maximum horizontal distance (m) considered from the plot centre. All points farther than max.dist will be discarded after the normalization process. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, all points will be used in processing, with max.dist representing the farthest point.
min.height	Optional minimum height (m) considered from ground level. All points below min.height will be discarded after the normalization process. If this argument

	is not specified by the user, it will be set to NULL by default and, as a consequence, all points will be used in processing, with `min.height` representing the lowest point.
max.height	Optional maximum height (m) considered from ground level. All points above `max.height` will be discarded after the normalization process. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, all points will be used in processing, with `max.height` representing the highest point.
algorithm.dtm	Algorithm used to generate the digital terrain model (DTM) from the TLS point cloud. There are two posible options based on spatial interpolation: 'tin' and 'knnidw' (see 'Details'). If this argument is not specified by the user, it will be set to 'knnidw' algorithm.
res.dtm	Numeric parameter. Resolution of the DTM generated to normalize point cloud (see 'Details'). If this argument is not specified by the user, it will be set to 0.2 m.
csf	List containing parameters of CSF algorithm:

  • `cloth_resolution`: by default 0.5.

intensity	Logical parameter useful when point clouds have LiDAR intesinty values.
RGB	Logical parameter useful when point clouds are colorized, thus including values of RGB colors. It is based on the Green Leaf Algorithm (GLA) (see 'Details').
scan.approach	Character parameter indicating TLS single-scan ('single') or TLS multi-scan approach or SLAM point clouds ('multi') approaches. If this argument is not specified by the user, it will be set to 'single' approach.
voxel_size	Defines the size of the 3D grid cells used for downsampling.
threads	Number of threads.
id	Optional plot identification encoded as character string or numeric. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the plot will be encoded as 1.
file	Optional file name identification encoded as character string or numeric value. If it is null, file will be encoded as `id` by default.
plot	Optional logical which indicates whether or not the normalized point cloud will be plot. If this argument is not specified by the user, it will be set to TRUE by default and, as consequence, the normalized point cloud will be plot.
dir.data	Optional character string naming the absolute path of the directory where LAS files containing TLS point clouds are located. `.Platform$file.sep` must be used as the path separator in `dir.data`. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to `dir.data` during the execution.
save.result	Optional logical which indicates whether or not the output files described in 'Output Files' section must be saved in `dir.result`. If this argument is not specified by the user, it will be set to TRUE by default and, as consequence, the output files will be saved.

dir.result	Optional character string naming the absolute path of an existing directory where files described in 'Output Files' section will be saved. .Platform$file.sep must be used as the path separator in dir.result. If this argument is not specified by the user, and save.result is TRUE, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to dir.result during the execution.
save.las	Optional logical which indicates whether or not the imput point cloud must be saved in dir.result as LAZ file.

Details

Relative coordinates are obtained by means of a normalization process, generating a digital terrain model (DTM) from the TLS point cloud, with the ground height set at 0 m. The DTM is generated by spatial interpolation of ground points classified with the CSF algorithm (Zhang et al., (2016)). Two algorithms are available for that purpose: (i) spatial interpolation based on a Delaunay triangulation, which performs a linear interpolation within each triangle ('tin'); (ii) spatial interpolation using a k-nearest neighbour (KNN) approach with inverse-distance weighting (IDW) ('knnidw'). Note that normalization process is based on **lidR** package functions: classify_ground, grid_terrain and normalize_height.

The point cropping process reduces the point cloud density proportionally to the likelihood that objects will receive points according to their distance from TLS and their size, which is determined by angle aperture (the farther they are, the lower the density is). The result is an approximately homogeneous point cloud in three-dimensional space (for more details see Molina-Valero et al., (2019)).

The Green Leaf Algorithm (GLA) is calculated according to Louhaichi et al., (2001)as follows:

Those points with values below 0 are clasified as woody parts, thus retained for tree detection in further functions.

Value

Data frame of normalized point cloud including the following columns (each row corresponds to one point):

id	Plot identification encoded as a character string or numeric in the argument id.
file	File name identification encoded as character string or numeric, corresponding to the normalized and reduced point clouds saved. This coincides with the TXT file in the absolute path specified in dir.result (if save.result is set to TRUE).
Coordinates	Cartesian (according to https://en.wikipedia.org/wiki/Cartesian_coordinate_system notation):

- x: x axis distance (m).
- y: y axis distance (m).
- z: height (m).

Cylindrical (according to https://en.wikipedia.org/wiki/Cylindrical_coordinate_system notation):

- rho: horizontal distance (m).
- phi: angular coordinate (rad).

- z: height (m).

Spherical (according to https://en.wikipedia.org/wiki/Spherical_coordinate_system notation):

- r: radial distance (m).
- theta: polar angle (rad).
- phi: azimuthal angle (rad)

slope	Slope of the terrain (rad).
intensity	Intensity (only if point cloud has intensity values and specified in arguments).
R	Red (only if point cloud is colorized and specified in arguments).
G	Green (only if point cloud is colorized and specified in arguments).
B	Blue (only if point cloud is colorized and specified in arguments).
GLA	Green Leaf Algorithm (only if point cloud is colorized and specified in arguments).
prob	selection probability assigned in point cropping process (0-1]. Only the farthest will have probability of 1.
prob.select	final selection probability assigned in point cropping process. Selected (1) and discarded point (0).

Output Files

At the end of the normalization process, if the save.result argument is TRUE, the function will save the reduced point cloud as TXT file and encoded according to file argument. The format is the same as data frame described in 'Value', except for a prob column, which is removed because all points selected after the point cropping process have a final selection probability of 1. The data frame is written without row names in dir.result directory using the vroom_write function in the **vroom** package.

Note

Note that max.dist, min.height and max.height arguments may be useful for optimizing computing time as well as for removing unnecessary and/or outlier points. These values may be selected more appropriately when inventory data are already available, or the user has some knowledge about autoecology of scanned tree species.

Note also that the linear interpolation algorithm ('tin' in this package) showed the highest accuracy in Liang et al., (2018) in DTM generation with single-scans. In this work a DTM resolution of 0.2 m was also considered adequately for square plots of 32 x 32 m.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

References

Liang, X., Hyyppä, J., Kaartinen, H., Lehtomäki, M., Pyörälä, J., Pfeifer, N., ... & Wang, Y. (2018). International benchmarking of terrestrial laser scanning approaches for forest inventories. *ISPRS journal of photogrammetry and remote sensing*, **144**, 137-179. doi:10.1016/j.isprsjprs.2018.06.021

Louhaichi, M., Borman, M. M., & Johnson, D. E. (2001). Spatially located platform and aerial photography for documentation of grazing impacts on wheat. *Geocarto International*, **16(1)**, 65-70. doi:10.1080/10106040108542184

Molina-Valero J. A., Ginzo-Villamayor M. J., Novo Pérez M. A., Álvarez-González J. G., & Pérez-Cruzado C. (2019). Estimación del área basimétrica en masas maduras de Pinus sylvestris en base a una única medición del escáner laser terrestre (TLS). *Cuadernos de la Sociedad Espanola de Ciencias Forestales*, **45(3)**, 97-116. doi:10.31167/csecfv0i45.19887.

Zhang, W., Qi, J., Wan, P., Wang, H., Xie, D., Wang, X., & Yan, G. (2016). An easy-to-use airborne LiDAR data filtering method based on cloth simulation. *Remote Sensing*, **8(6)**, 501. doi:10.3390/rs8060501.

See Also

tree.detection.single.scan, tree.detection.multi.scan, tree.detection.several.plots.

Examples

```
# Establishment of working directories (optional)
# By default here we propose the current working directory of the R process

dir.data <- tempdir()
dir.result <- tempdir()



# TLS SINGLE-SCAN APPROACH

# Loading example TLS data (LAZ file) to dir.data

download.file("https://www.dropbox.com/s/17yl25pbrapat52/PinusRadiata.laz?dl=1",
              destfile = file.path(dir.data, "PinusRadiata.laz"),
              mode = "wb")



# Normalizing the whole point cloud data without considering arguments

# Note that in the case of TLS single-scans, the coordinates of the plot center
# must be specified by the x.center and y.center arguments, which are
# x.center = 0 and y.center = 0 in this case

pcd <- normalize(las = "PinusRadiata.laz",

                 id = "PinusRadiata",

                 x.center = 0, y.center = 0,
```

```
                            dir.data = dir.data, dir.result = dir.result)


    # MLS OR TLS MULTI-SCAN APPROACHES

    # Loading example MLS data (LAZ file) to dir.data

    download.file(
    "www.dropbox.com/scl/fi/es5pfj87wj0g6y8414dpo/PiceaAbies.laz?rlkey=ayt21mbndc6i6fyiz2e7z6oap&dl=1",
                    destfile = file.path(dir.data, "PiceaAbies.laz"),
                    mode = "wb")

    # Normalizing the whole point cloud data without considering arguments

    pcd <- normalize(las = "PiceaAbies.laz",

                        id = "PiceaAbies",

                        scan.approach = "multi",

                        threads = 2,

                        dir.data = dir.data, dir.result = dir.result)
```

---

optimize.plot.design   *Optimize Plot Design Based on Optimal Correlations*

Description

Generation of interactive heatmaps graphically represent the optimal correlations between variables estimated from field data, and metrics derived from TLS data. These data must be derived from any of the three different plot designs currently available (circular fixed area, k-tree and angle-count) and correspond to plots with incremental values for the plot design parameter (radius, k and BAF, respectively). In addition, correlation measures that are currently admissible are Pearson's correlation coefficient and/or Spearman's *rho*.

Usage

```
optimize.plot.design(correlations,
                        variables = c("N", "G", "d", "dg", "d.0", "h", "h.0"),
                        dir.result = NULL)
```

Arguments

correlations   List including the optimal correlations between field estimations and TLS met-
                 rics. The structure and format must be analogous to the opt.correlations

element in the output returned by the [correlations](#) function. In particular, the list must include at least one of the following named elements:

- pearson: list containing the optimal Pearson's correlations, and the names of the TLS metrics to which they correspond. It must include at least one of the following named elements:
  - fixed.area.plot: data frame containing the optimal Pearson's correlations, and the names of the TLS metrics to which they correspond, under circular fixed area plot design. Each row will correspond to a radius value, and the data frame will include the following columns:
    * radius: radius (m) of the simulated plots used for computing the estimated correlations.
    * Columns '<x>.cor' and '<x>.metric': the former, numeric column(s) containing optimal Pearson's correlations between '<x>', a field estimate, and the available TLS metrics when [correlations](#) function was executed; and the latter, character column(s) containing names of the TLS metrics to which they correspond.

    If fixed.area.plot is included in pearson element, it must contain at least the radius column and a ('<x>.cor', '<x>.metric') pair of columns corresponding to the same field estimation.
  - k.tree.plot: data frame including the optimal Pearson's correlations and the names of the TLS metrics to which they correspond, under the k-tree plot design. Each row will correspond to a k value, and the following columns will be included:
    * k: number of trees (trees) of the simulated plots used for computing the estimated correlations.
    * Columns '<x>.cor' and '<x>.metric': same description and format as indicated in correlations$pearson$fixed.area.plot element.

    If k.tree.plot is included in pearson element, it must contain at least k column, and a ('<x>.cor', '<x>.metric') pair of columns corresponding to the same field estimation.
  - angle.count.plot: data frame including the optimal Pearson's correlations and the names of the TLS metrics to which they correspond, for the angle-count plot design. Each row will correspond to a BAF value, and the data frame will include the following columns:
    * BAF: BAF (m^2/ha) of the simulated plots used to compute the estimated correlations.
    * Columns '<x>.cor' and '<x>.metric': same description and format as indicated in correlations$pearson$fixed.area.plot element.

    If the angle.count.plot is included in the pearson element, it must include at least the BAF column and a ('<x>.cor', '<x>.metric') pair of columns corresponding to the same field estimation.
- spearman: list containing the optimal Spearman's correlations, and the names of the TLS metrics to which they correspond. The structure and format will be analogous to that indicated for the previous element but with optimal Pearson's correlations replaced by Spearman's correlations.

variables   Optional character vector naming field estimations whose optimal correlations will be represented graphically in the heatmaps generated during the execution. If this argument is specified by the user, it must include at least one of the following character strings: "N", "G", "V", "d", "dg", "dgeom", "dharm", "d.0", "dg.0", "dgeom.0", "dharm.0", "h", "hg", "hgeom", "hharm", "h.0", "hg.0", "hgeom.0", or "hharm.0". If this argument is not specified by the user, it will be set to c("N", "G", "V", "d", "dg", "d.0", "h", "h.0") by default. In both cases, all data frames in the correlations argument must have at least the ('<x>.cor', '<x>.metric') pairs corresponding to the field estimations specified in the variables argument.

dir.result   Optional character string naming the absolute path of an existing directory where files described in 'Output Files' section will be saved. .Platform$file.sep must be used as the path separator in dir.result. If this argument is not specified by the user, it will be set to NULL by default and, as consequence, the current working directory of the R process will be assigned to dir.result during the execution.

Details

This function represents graphically, by means of interactive heatmaps, the strongest correlations (positive or negative) for each plot design and size simulated, between the estimated variables based on field data specified in the variables argument, and metrics derived from TLS data, under circular fixed area, k-tree and/or angle-count plot designs.

Two correlation measures are implemented at present: Pearson's correlation coefficient and Spearman's rho. Hence, only optimal correlations based on correlations arguments will be taken into account during the execution.

For each correlation measure and plot design, at least one no missing value for optimal correlations must be represented; otherwise, execution will be stopped, and an error message will appear. In addition, at least two different no missing values for optimal correlations are required to ensure that the colour palette is correctly applied when the heatmap is generated.

Value

Invisible NULL.

Output Files

During the execution, interactive heatmaps graphically representing optimal correlations values between field estimations and TLS metrics are created and saved in dir.result directory by means of the saveWidget function in the **htmlwidgets** package. The widgets generated allow users to consult optimal correlations values and TLS metrics to which they correspond directly on the plots, to zoom and scroll, and so on. The pattern used for naming these files is 'opt.correlations.<plot design>.<method>.html', where '<plot design>' equals "fixed.area.plot", "k.tree.plot" or "angle.count.plot" according to plot design, and '<method>' equals "pearson" or "spearman" according to correlation measure.

Note

This function is key to choosing the best possible plot design (in terms of correlation measures) considering all variables of interest before establishing definitive sampling design.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

See Also

correlations.

Examples

```
# Load field estimations and TLS metrics corresponding to Rioja data set

data("Rioja.simulations")


# Compute correlations between field estimations and TLS metrics corresponding
# to Rioja example, and select optimal correlations results

corr <- correlations(simulations = Rioja.simulations,
                     variables = c("N", "G", "d", "dg", "dgeom","dharm",
                                   "d.0", "dg.0", "dgeom.0", "dharm.0", "h",
                                   "hg", "hgeom", "hharm", "h.0", "hg.0",
                                   "hgeom.0", "hharm.0"),
                     save.result = FALSE)

opt.corr <- corr$opt.correlations


# Establish directory where optimal correlations heatmaps corresponding to Rioja
# example will be saved. For instance, current working directory

dir.result <- tempdir()


# Generate heatmaps for optimal correlations between field estimations and TLS
# metrics corresponding to Rioja example
# Optimal Pearson's and Spearman's correlations for variables by default

# optimize.plot.design(correlations = opt.corr, dir.result = dir.result)


# Optimal Pearson's and Spearman's correlations for variables 'N' and 'G'

optimize.plot.design(correlations = opt.corr, variables = c("N", "G"),
                     dir.result = dir.result)
```

Description

Computes relative bias between variables estimated from field data and their TLS counterparts derived from TLS data. Field estimates and TLS metrics for a common set of plots are required in order to compute relative bias. These data must come from any of the three different plot designs currently available (circular fixed area, k-tree and angle-count) and correspond to plots with incremental values for the plot design parameter (radius, k and BAF, respectively). In addition to computing relative bias, interactive line charts graphically representing the values obtained between each field estimate and its related TLS metrics are also generated.

Usage

```
relative.bias(simulations,
              variables = c("N", "G", "d", "dg", "d.0", "h", "h.0"),
              save.result = TRUE, dir.result = NULL)
```

Arguments

simulations   List containing variables estimated from field data and metrics derived from
               TLS data. The structure and format must be analogous to output returned by
               the [simulations](#) function. In particular, teh list must include at least one of the
               following named elements:

- fixed.area.plot: same description and format as indicated for same named
  element in simulations argument of [correlations](#) function. The only
  difference is the columns required when it is included in the argument: in
  adittion to id, radius and at least one of the field estimate columns, it must
  include at least one TLS counterpart for each field estimate condidered.
- k.tree.plot: same description and format as indicated for same named
  element in simulations argument of [correlations](#) function. The only
  difference is the columns required when it is included in the argument: in
  adittion to id, k and at least one of the field estimate columns, it must
  include at least one TLS counterpart for each field estimate considered.
- angle.count.plot: same description and format as indicated for the same
  named element in the simulations argument of [correlations](#) function.
  The only difference is the columns that are required when this element is
  included in the argument: in adittion to id, BAF and at least one of the field
  estimate columns, it must contain at least one TLS counterpart for each field
  estimate considered.

variables   Optional character vector naming field estimates for which the relative bias be-
               tween them and all their available TLS counterparts will be computed. If this
               argument is specified by the user, it must contain at least one of the following

character strings: "N", "G", "V", "d", "dg", "dgeom", "dharm", "d.0", "dg.0",
"dgeom.0", "dharm.0", "h", "hg", "hgeom", "hharm", "h.0", "hg.0",
"hgeom.0", or "hharm.0". If this argument is not specified by the user, it will
be set to c("N", "G", "V", "d", "dg", "d.0", "h", "h.0") by default. In
both cases, all the elements in simulations argument must include at least
the columns corresponding to the field estimations specified in the variables
argument.

save.result   Optional logical which indicates whether or not the output files described in
'Output Files' section must be saved in dir.result. If this argument is not
specified by the user, it will be set to TRUE by default and, as a consequence, the
output files will be saved.

dir.result   Optional character string naming the absolute path of an existing directory where
files described in 'Output Files' section will be saved. .Platform$file.sep
must be used as the path separator in dir.result. If this argument is not spec-
ified by the user, and save.result is TRUE, it will be set to NULL by default
and, as a consequence, the current working directory of the R process will be
assigned to dir.result during the execution.

Details

For each radius, k or BAF value (according to the currently available plot designs: circular fixed
area, k-tree and angle-count), this function computes the relative bias between each variable es-
timated from field data, and specified in the variables argument, and their counterparts derived
from TLS data, and existing in the data frames included in the simulations argument. TLS met-
rics considered *counterparts* for each field estimate are detailed below (see [simulations](#) 'Value'
function for details about used notation):

- TLS counterparts for N are N.tls, N.hn, N.hr, N.hn.cov, N.hr.cov and N.sh in the fixed
  area and k-tree plot designs; and N.tls and N.pam in the angle-count plot design.

- TLS counterparts for G are G.tls, G.hn, G.hr, G.hn.cov, G.hr.cov and G.sh in the fixed
  area and k-tree plot designs; and G.tls and G.pam in the angle-count plot design.

- TLS counterparts for V are V.tls, V.hn, V.hr, V.hn.cov, V.hr.cov and V.sh in the fixed
  area and k-tree plot designs; and V.tls and V.pam in the angle-count plot design.

- TLS counterparts for d, dg, dgeom, dharm, d.0, dg.0, dgeom.0, and dharm.0 are, respectively:
  d.tls, dg.tls, dgeom.tls, dharm.tls, d.0.tls, dg.0.tls, dgeom.0.tls,
  and dharm.0.tls in any of the three available plot designs.

- TLS counterparts for h, hg, hgeom, hharm, h.0, hg.0, hgeom.0, and hharm.0 are, respectively
  h.tls, hg.tls, hgeom.tls, hharm.tls, h.0.tls, hg.0.tls, hgeom.0.tls, and
  hharm.0.tls in any of the three available plot designs. In adittion, P99 is also taken into
  account as a counterpart for all these field estimates.

The relative bias between a field estimation and any of its TLS counterparts is estimated as follows

$$\frac{\frac{1}{n}\sum_{i=1}^{n} y_i - \frac{1}{n}\sum_{i=1}^{n} x_i}{\frac{1}{n}\sum_{i=1}^{n} x_i} * 100$$

where x_i and y_i are the values of the field estimate and its TLS counterpart, respectively, corre-
sponding to plot i for $i = 1, \ldots, n$.

Value

fixed.area.plot

If no "fixed.area.plot" element exists in simulations argument, missing; otherwise, the matrix will include estimates of the the relative bias under circular fixed area plot design between each field estimation specified in variables argument and its TLS counterpart(s) existing in the "fixed.area.plot" element in the simulations argument. Each row will correspond to a radius value, and the following columns will be included:

- radius: radius (m) of the simulated plots used for computing the estimated relative bias.
- Column(s) '<x>.<y>': numeric column(s) containing estimated relative bias between '<x>', a field estimation, and '<y>', a TLS counterpart.

k.tree.plot   If no "k.tree.plot" element exists in the simulations argument, missing; otherwise, the matrix will include the relative bias estimated in the k-tree plot design between each field estimation specified in variables argument and its TLS counterpart(s) existing in "k.tree.plot" element in simulations argument. Each row will correspond to a k value, and the following columns will be included:

- k: number of trees (trees) of the simulated plots used for computing the estimated relative bias.
- Column(s) '<x>.<y>': numeric column(s) containing estimated relative bias between '<x>', a field estimation, and '<y>', a TLS counterpart.

angle.count.plot

If no "angle.count" element exists in simulations argument, missing; otherwise, the matrix will contain estimated relative bias under angle-count plot design between each field estimation specified in the variables argument and its TLS counterpart(s) existing in the "angle.count.plot" element in the simulations argument. Each row will correspond to a BAF value, and the following columns will be included:

- BAF: BAF (m^2/ha) of the simulated plots used for computing the estimated relative bias.
- Column(s) '<x>.<y>': numeric column(s) containing estimated relative bias between '<x>', a field estimation, and '<y>', a TLS counterpart.

Output Files

During the execution, if the save.result argument is TRUE, the function will print the matrices described in the 'Value' section to files. These are written without row names in dir.result directory using write.csv function from the **utils** package. The pattern used for naming these files is 'RB.<plot design>.csv', where '<plot design>' is equal to "fixed.area.plot", "k.tree.plot" or "angle.count.plot" is according to the plot design.

Furthermore, if the save.result argument is TRUE, interactive line charts graphically representing relative bias values will also be created and saved in the dir.result directory by means of the saveWidget function in the **htmlwidgets** package. Generated widgets allow users to consult relative bias data directly on the plots, select/deselect different sets of traces, to zoom and scroll, and so on. The pattern used for naming these files is 'RB.<x>.<plot design>.html', where '<plot design>'

is indicated for the previously described files, and '<x>' equals N, G, V, d and/or h according to the
`variables` argument. All relative biases related to diameters are plotted in the same chart (files
named as 'RB.d.<plot design>.html'), and the same applies to those related to heights (files
named as 'RB.h.<plot design>.html').

Note

The results obtained using this function are merely descriptive, and they do not guarantee any type
of statistical accuracy in using TLS metrics instead of field estimations in order to estimate forest
attributes of interest.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

See Also

simulations, correlations.

Examples

```
# Load variables estimated from field data, and TLS metrics
# corresponding to Rioja data set

data("Rioja.simulations")


# Establish directory where relative bias results corresponding to Rioja example
# will be saved. For instance, current working directory

dir.result <- tempdir()


# Compute relative bias between field-based estimates of TLS metrics
# corresponding to Rioja example
# Relative bias for variables by default

rb <- relative.bias(simulations = Rioja.simulations, dir.result = dir.result)


# Relative bias for variable 'N'

rb <- relative.bias(simulations = Rioja.simulations, variables = "N",
                    dir.result = dir.result)


# Relative bias corresponding to angle-count design for all available variables

rb <- relative.bias(simulations = Rioja.simulations["angle.count"],
                    variables <- c("N", "G", "V", "d", "dg", "dgeom", "dharm",
                                   "d.0", "dg.0", "dgeom.0", "dharm.0", "h",
```

```
                                "hg", "hgeom", "hharm", "h.0", "hg.0",
                                "hgeom.0", "hharm.0"),
                  dir.result = dir.result)
```

Rioja.data	*Inventoried Plots Data for a Stand Case Study in La Rioja*

Description

This list includes trees detected with TLS for 16 single scans corresponding to plots located in La Rioja, a region of Spain, in the north of the Iberian Peninsula (first element), as well as those inventoried in the field for these 16 plots (second element). Plot attributes related to stand stratum are also included (third element).

The elements of the list are as follows:

1. `tree.tls`: data frame that includes the list of trees detected with `tree.detection.single.scan` for 16 TLS single-scan sampling points. The following variables are provided for each tree (see `tree.detection.single.scan` 'Value' for more details):

[,1]	id	character/numeric
[,2]	file	character
[,3]	tree	numeric
[,4]	x	numeric
[,5]	y	numeric
[,6:8]	phi, phi.left, phi.right	numeric
[,9]	h.dist	numeric
[,10]	dbh	numeric
[,11]	h	numeric
[,12]	v	numeric
[,13:16]	n.pts, n.pts.red, n.pts.est, n.pts.red.est	numeric
[,15]	partial.occlusion	numeric

2. `tree.list.field`: data frame that includes the list of trees measured in 16 circular fixed area plots of radius 20 m, whose centres coincide with TLS single-scans points. The following variables are provided for each tree:

[,1]	id	numeric	plot identification (coincident to TLS scans)
[,2]	tree	numeric	trees numbering
[,3]	Sp	numeric	specie code according to NFI ()
[,4]	x	numeric	x cartesian coordinate
[,5]	y	numeric	7 cartesian coordinate
[,6]	h.dist	numeric	horizontal distance (m) from plot center to tree center
[,7]	dbh	numeric	tree diameter (cm) at breast height (1.3 m)
[,8]	h	numeric	tree total height (m)
[,9]	dead	numeric	dead (1) or not (NA)

[,10]	v.user	numeric	stem volume (m^3) estimated with allometric equations
[,11]	w.user	numeric	stem biomass (Mg) estimated with allometric equations

Usage

```
data(Rioja.data)
```

Format

List with 2 data frames containing 604 observations and 17 variables (tree.tls) and 659 observations and 11 variables (tree.field).

Rioja.simulations   *Simulated Metrics and Variables for a Stand Case Study in La Rioja*

Description

This list contains metrics and variables estimated from field data and TLS data from `Rioja.data`.

The elements on this list correspond to `simulations` 'Value', as follows:

1. `fixed.area`: data frame with TLS metrics and variables estimated on the basis of simulated plots in a fixed area plot design with radius increment of 0.1 m (from smallest possible radius to 20 m). The following variables are provided for each pair (plot, radius) (see `simulations` 'Value' for more details):

[,1]	id	character/numeric
[,2]	radius	numeric
[,3:5]	N, G, V	numeric
[,6:9]	d, dg, dgeom, dharm	numeric
[,10:13]	h, hg, hgeom, hharm	numeric
[,14:17]	d.0, dg.0, dgeom.0, dharm.0	numeric
[,18:21]	h.0, hg.0, hgeom.0, hharm.0	numeric
[,22:27]	N.tls, N.hn.tls, N.hr.tls, N.hn.cov.tls, N.hr.cov.tls, N.sh.tls	numeric
[,28:31]	num.points, num.points.est, num.points.hom, num.points.hom.est	numeric
[,32:37]	G.tls, G.hn.tls, G.hr.tls, G.hn.cov.tls, G.hr.cov.tls, G.sh.tls	numeric
[,38:43]	V.tls, V.hn.tls, V.hr.tls, V.hn.cov.tls, V.hr.cov.tls, V.sh.tls	numeric
[,44:47]	d.tls, dg.tls, dgeom.tls, dharm.tls	numeric
[,48:51]	h.tls, hg.tls, hgeom.tls, hharm.tls	numeric
[,52:55]	d.0, dg.0, dgeom.0, dharm.0	numeric
[,56:59]	h.0, hg.0, hgeom.0, hharm.0	numeric
[,60:74]	P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99	numeric

2. `k.tree`: data frame with TLS metrics and variables estimated on the basis of simulated plots under k-tree plot design for incremental values of 1 tree (from 1 to largest number of trees

in one plot). The following variables are provided for each pair (plot, k) (see simulations 'Value' for more details):

[,1]	id	character/numeric
[,2]	k	numeric
[,3:5]	N, G, V	numeric
[,6:9]	d, dg, dgeom, dharm	numeric
[,10:13]	h, hg, hgeom, hharm	numeric
[,14:17]	d.0, dg.0, dgeom.0, dharm.0	numeric
[,18:21]	h.0, hg.0, hgeom.0, hharm.0	numeric
[,22:27]	N.tls, N.hn.tls, N.hr.tls, N.hn.cov.tls, N.hr.cov.tls, N.sh.tls	numeric
[,28:31]	num.points, num.points.est, num.points.hom, num.points.hom.est	numeric
[,32:37]	G.tls, G.hn.tls, G.hr.tls, G.hn.cov.tls, G.hr.cov.tls, G.sh.tls	numeric
[,38:43]	V.tls, V.hn.tls, V.hr.tls, V.hn.cov.tls, V.hr.cov.tls, V.sh.tls	numeric
[,44:47]	d.tls, dg.tls, dgeom.tls, dharm.tls	numeric
[,48:51]	h.tls, hg.tls, hgeom.tls, hharm.tls	numeric
[,52:55]	d.0, dg.0, dgeom.0, dharm.0	numeric
[,56:59]	h.0, hg.0, hgeom.0, hharm.0	numeric
[,60:74]	P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99	numeric

3. angle.count: data frame with TLS metrics and variables estimated on the basis of simulated plots in an angle-count plot design. They plots are simulated for correlative angle-count plots and incremental values of 0.1 m^2/ha for BAF. The following variables are provided for each pair (plot, BAF) (see simulations 'Value' for more details):

[,1]	id	character/numeric
[,2]	BAF	numeric
[,3:5]	N, G, V	numeric
[,6:9]	d, dg, dgeom, dharm	numeric
[,10:13]	h, hg, hgeom, hharm	numeric
[,14:17]	d.0, dg.0, dgeom.0, dharm.0	numeric
[,18:21]	h.0, hg.0, hgeom.0, hharm.0	numeric
[,22:23]	N.tls, N.pam.tls	numeric
[,24:27]	num.points, num.points.est, num.points.hom, num.points.hom.est	numeric
[,28:29]	G.tls, G.pam.tls	numeric
[,30:31]	V.tls, V.pam.tls	numeric
[,32:35]	d.tls, dg.tls, dgeom.tls, dharm.tls	numeric
[,48:51]	h.tls, hg.tls, hgeom.tls, hharm.tls	numeric
[,36:39]	d.0, dg.0, dgeom.0, dharm.0	numeric
[,40:43]	h.0, hg.0, hgeom.0, hharm.0	numeric
[,44:62]	P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99	numeric

Usage

```
data(Rioja.simulations)
```

Format

List with 3 data frames containing 2224 observations and 74 variables (simulations.fixed.area.plot), 272 observations and 74 variables (simulations.k.tree.plot), and 576 observations and 62 variables (simulations.angle.count.plot).

---

simulations   *Compute Metrics and Variables for Simulated TLS and Field Plots*

Description

Computes TLS metrics derived from simulated TLS plots and variables estimated on the basis of simulated field plots. Real TLS and field data from the same set of plots are required in order to build simulated plots. Three different plot designs are currently available: circular fixed area, k-tree and angle-count. During the simulation process, plots with incremental values for radius, k and BAF are simulated for circular fixed area, k-tree and angle-count designs, respectively, according to the parameters specified in the `plot.parameters` argument. For TLS metrics, different method are included for correcting occlusions generated in TLS point clouds.

Usage

```
simulations(tree.tls, tree.ds = NULL, tree.field,
            plot.design = c("fixed.area", "k.tree", "angle.count"),
          plot.parameters = data.frame(radius.max = 25, k.max = 50, BAF.max = 4),
            scan.approach = "single", var.metr = list(tls = NULL, field = NULL),
            v.calc = "parab", dbh.min = 4, h.min = 1.3, max.dist = Inf,
            dir.data = NULL, save.result = TRUE, dir.result = NULL)
```

Arguments

tree.tls   Data frame with information about trees detected from TLS point cloud data. The structure and format must be analogous to output returned by [tree.detection.single.scan](#) and [tree.detection.multi.scan](#) (or [tree.detection.several.plots](#)) functions. In particular, each row must correspond to a (plot, tree) pair, and it must include at least the following columns:

- `id`, `file`, `tree`, `x`, `y`, `phi.left`, `phi.right`, `horizontal.distance`, `dbh`, `num.points`, `num.points.hom`, `num.points.est`, `num.points.hom.est`, `partial.occlusion`: same description and format as indicated for the same named columns in [tree.detection.single.scan](#) 'Value'.

tree.ds   An optional list containing results arises from the application of distance sampling methodologies. The structure and format must be analogous to output returned by [distance.sampling](#) function. In particular, it must include at least the following elements:

- `tree`: data frame with detection probabilities for each tree using distance sampling methodologies. Each row must correspond to a (plot, tree) pair, and it must include at least the following columns:

– id, tree, P.hn, P.hn.cov, P.hr, P.hr.cov: same description and format as indicated for same named columns of tree in `distance.sampling` 'Value'. In addition, plot identification and tree numbering included in id and tree columns must coincide with those included in the same named columns of tree.tls argument.

If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the TLS metrics using distance sampling based correction will not be calculated for a circular fixed area or k-tree plot designs.

tree.field   Data frame with information about trees measured in the field plots. Each row must correspond to a (plot, tree) pair, and it must include at least the following columns:

- id: plot identification encoded as character string or numeric. Plot identifications must coincide with those included in id column of the tree.tls argument.
- tree: trees numbering.
- horizontal.distance: horizontal distance (m) from plot centre to tree centre. Centres of the field plots must coincide with centres of their corresponding TLS plots.
- dbh: tree diameter (cm) at breast height (1.3 m).
- total.height: tree total height (m).
- dead: integer value indicating for each tree if it is dead (1) or not (NA).

plot.design   Vector containing the plot designs considered. By default, all plot designs will be considered (circular fixed area, k-tree and angle-count plots).

plot.parameters

Optional list containing parameters for circular fixed area, k-tree and angle-count plot designs. User can set all or any of the following parameters specifying them as named elements of the list:

- radius.max: maximum radius (m) allowed for the increasing sequence of radius values to be used in the simulation process of TLS and field plots under circular fixed area plot design. If this element is not included in the argument, circular fixed area plots will not be simulated.
- radius.increment: positive increment (m) for the increasing sequence of radius values to be used in the simulation process of TLS and field plots in a circular fixed area plot design. If this element is not included in the argument, it is set to 0.1 m by default.
- k.tree.max: maximum number of trees (trees) allowed for the increasing sequence of k values to be used in the simulation process of TLS and field plots under k-tree plot design. If this element is not included in the argument, k-tree plots will not be simulated.
- BAF.max: maximum BAF (m^2/ha) allowed for the increasing sequence of BAF values to be used in the simulation process of TLS and field plots in an angle-count plot design. If this element is not included in the argument, angle-count plots will not be simulated.
- BAF.increment: positive increment (m^2/ha) for the increasing sequence of BAF values to be used in the simulation process of TLS and field plots

under angle-count plot design. If this element is not included in the argument, it is set to 0.1 m^2/ha by default.

- num.trees: number of dominant trees per ha (tree/ha) to be used for calculating dominant diameters and heights during the simulation process of TLS and field plots under all the available plot designs. Dominant trees are those with the largest diameter at breast height. If this element is not included in plot.parameters argument, it is set to 100 trees/ha by default.

If this argument is specified by the user, it must include at least one of the following elements: radius.max, k.tree.max or BAF.max. If this argument is not specified by the user, it is set to list(radius.max = 25, k.tree.max = 50, BAF.max = 4) by default and, as a consequence, the three available plot designs will be simulated.

scan.approach   Character parameter indicating TLS single-scan ('single') or TLS multi-scan approach or SLAM point clouds ('multi') approaches. If this argument is not specified by the user, it will be set to 'multi' approach.

var.metr   Optional vector containing all the metrics and variables of interest. By default it will be set as NULL and thus, all the metrics and variables available will be generated.

v.calc   Optional parameter to calculate volume when is not included in tree.tls input data.

dbh.min   Optional minimum *dbh* (cm) considered for detecting trees. By default it will be set at 4 cm.

h.min   Optional minimum *h* (m) considered for detecting trees. By default it will be set at 1.3 m.

max.dist   Optional argument to specify the maximum horizontal distance considered in which trees will be included.

dir.data   Optional character string naming the absolute path of the directory where TXT files containing TLS point clouds are located. .Platform$file.sep must be used as the path separator in dir.dat, and TXT files in the directory must have the same description and format as indicated for TXT files in [normalize](#) 'Output Files'. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to dir.dat during the execution.

save.result   Optional logical which indicates wheter or not the output files described in 'Output Files' section must be saved in dir.result. If this argument is not specified by the user, it will be set to TRUE by default and, as a consequence, the output files are saved.

dir.result   Optional character string naming the absolute path of an existing directory where files described in 'Output Files' section will be saved. .Platform$file.sep must be used as the path separator in dir.result. If this argument is not specified by the user, and save.result is TRUE, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to dir.result during the execution.

Details

Using real TLS and field data from the same set of plots, this function enables construction of simulated plots under different plot designs and computation of the corresponding TLS metrics and estimated variables. The notation used for variables is based on IUFRO (1959).

At this stage, three plot designs are available:

- Circular fixed area plots, simulated only if a `radius.max` value is specified in the `plot.parameters` argument.
- k-tree plots, simulated only if a `k.tree.max` value is specified in the `plot.parameters` argument.
- Angle-count plots, simulated only if a `BAF.max` value is specified in the `plot.parameters` argument.

For each real plot, a simulation process is run under each of the plot designs specified by means of elements of the `plot.parameters` argument. Although there are some minor differences depending on the plot design, the rough outline of the simulation process is similar for all, and it consists of the following main steps:

1. Define an increasing sequence of the plot design parameter (radius, k or BAF) according to the maximum value and, if applicable, the positive increment set in `plot.parameters` argument.
2. Build simulated plots for each parameter value in the previous sequence based on either TLS or field data.
3. Compute either TLS metrics or variables estimated on the basis of simulated plots for each parameter value (see 'Value' section for details). For the simulated TLS plots, note that in addition to the counterparts of variables computed for the simulated field plots, the function also computes the following:
   - Metrics related to the number of points belonging to normal tree sections.
   - Metrics with occlusion corrections based on the following:
     - Distance sampling methodologies (Astrup et al., 2014) for circular fixed area and k-tree plot designs, if the `distance.sampling` argument is not `NULL`.
     - Correction of the shadowing effect (Seidel & Ammer, 2014) for circular fixed area and k-tree plot designs.
     - Gap probability attenuation with distance to TLS (Strahler et al., 2008; Lovell et al., 2011) for angle-count plot design.
   - Height percentiles derived from z coordinates of TLS point clouds relative to ground level.

Value

List with field estimates and TLS metrics for plot designs considered. It will contain one element per plot design considered (fixed.area.plot, k.tree.plot and angle.count.plot)

`fixed.area.plot`

> If no value for `radius.max` is specified in the `plot.parameters` argument, `NULL`; otherwise, data frame with TLS metrics and variables will be estiamted on the basis of simulated plots in a circular fixed area plot design. Each row will correspond to a (plot, radius) pair, and the following columns will be included: Plot identification and radius:

- `id`: plot identification encoded as a character string or numeric. It will coincide with those included in the `id` column of `tree.tls`, `tree.field` or, if applicable, `distance.sampling` arguments.
- `radius`: radius (m) of the simulated plot.

Variables estimated on the basis of simulated field plots:

- `N`: stand density (trees/ha).
- `G`: stand basal area (m^2/ha).
- `V`: stand volume (m^3/ha).
- `d`, `dg`, `dgeom`, `dharm`: mean tree diameters (cm) at breast height (1.3 m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `h`, `hg`, `hgeom`, `hharm`: mean tree heights (m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `d.0`, `dg.0`, `dgeom.0`, `dharm.0`: dominant mean tree diameters (cm) at breast height (1.3 m), calcualted from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `h.0`, `hg.0`, `hgeom.0`, `hharm.0`: dominant mean tree heights (m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.

TLS variables derived from simulated TLS plots:

- `N.tls`: stand density (trees/ha) without occlusion corrections.
- `N.hn`, `N.hr`, `N.hn.cov`, `N.hr.cov`: stand density (trees/ha) with occlusion corrections based on distance sampling methodologies. These columns will be missing if the `distance.sampling` argument is NULL.
- `N.sh`: stand density (trees/ha) with correction of the shadowing effect.
- `G.tls`: stand basal area (m^2/ha) without occlusion corrections.
- `G.hn`, `G.hr`, `G.hn.cov`, `G.hr.cov`: stand basal area (m^2/ha) with occlusion corrections based on distance sampling methodologies. These columns will be missing if the `distance.sampling` argument is NULL.
- `G.sh`: stand basal area (m^2/ha) with correction of the shadowing effect.
- `V.tls`: stand volume (m^3/ha) without occlusion corrections.
- `V.hn`, `V.hr`, `V.hn.cov`, `V.hr.cov`: stand volume (m^3/ha) with occlusion corrections based on distance sampling methodologies. These columns will be missing if the `distance.sampling` argument is NULL.
- `V.sh`: stand volume (m^3/ha) with correction of the shadowing effect.
- `d.tls`, `dg.tls`, `dgeom.tls`, `dharm.tls`: mean tree diameters (cm) at breast height (1.3 m), calculated from the arithmetic mean, quadratic mean geometric mean, and harmonic mean, respectively.
- `h.tls`, `hg.tls`, `hgeom.tls`, `hharm.tls`: mean tree heights (m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.
- `d.0.tls`, `dg.0.tls`, `dgeom.0.tls`, `dharm.0.tls`: dominant mean tree diameters (cm) at breast height (1.3 m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.

- `h.0.tls, hg.0.tls, hgeom.0.tls, hharm.0.tls`: dominant mean tree heights (m), calculated from the arithmetic mean, quadratic mean, geometric mean and harmonic mean, respectively.

TLS metrics derived from simulated TLS plots:

- `num.points, num.points.est, num.points.hom, num.points.hom.est`: number of points and estimated number of points (points) belonging to trees with normal sections (+/- 5 cm) in the original point cloud (`num.points` and `num.points.est`, respectively); and number of points and estimated number of points (points) belonging to trees normal sections (+/- 5 cm) in the reduced point cloud (`num.points.hom` and `num.points.hom.est`, respectively).

- `P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99`: height percentiles derived from z coordinates of TLS point clouds relative to ground level.

`k.tree.plot`   If no value for `k.tree.max` is specified in the `plot.parameters` argument, `NULL`; otherwise the data frame with TLS metrics and estimations of variables will be based on simulated plots in the k-tree plot design. Each of row will correspond to a (plot, k) pair, and the following columns will be included:

Plot identification and k:

- `id`: plot identification encoded as character string or numeric. The `id` will coincide with those included in the `id` column of `tree.tls`, `tree.field` or, if applicable, `distance.sampling` arguments.

- `k`: number of trees (trees) in the simulated plot.

Estimated variables based on simulated field plots:

- `N, G, V, d, dg, dgeom, dharm, h, hg, hgeom, hharm, d.0, dg.0, dgeom.0, dharm.0, h.0, hg.0, hgeom.0, hharm.0`: same description and format as indicated in the `fixed.area.plot` element.

TLS variables derived from simulated TLS plots:

- `N.tls, N.hn, N.hr, N.hn.cov, N.hr.cov, N.sh,`
  `G.tls, G.hn, G.hr, G.hn.cov, G.hr.cov, G.sh,`
  `V.tls, V.hn, V.hr, V.hn.cov, V.hr.cov, V.sh,`
  `d.tls, dg.tls, dgeom.tls, dharm.tls,`
  `h.tls, hg.tls, hgeom.tls, hharm.tls,`
  `d.0.tls, dg.0.tls, dgeom.0.tls, dharm.0.tls,`
  `h.0.tls, hg.0.tls, hgeom.0.tls, hharm.0.tls`

TLS metrics derived from simulated TLS plots:

- `num.points, num.points.est, num.points.hom, num.points.hom.est, P01, P05, P10, P20, P25, P30, P40, P50, P60, P70, P75, P80, P90, P95, P99`: same description and format as indicated in `fixed.area.plot` element.

`angle.count.plot`

If no value for `BAF.max` is specified in the `plot.parameters` argument, `NULL`; otherwise the data frame will include TLS metrics and estimated variables based on simulated plots in the angle-count plot design. Each row will correspond to a (plot, BAF) pair, and the following columns will be included:

Plot identification and BAF:

- `id`: plot identification encoded as character string or numeric. The `id` will coincide with those included in the `id` column of `tree.tls` and `tree.field`.
- `BAF`: BAF (m^2/ha) of the simulated plot.

Estimated variables based on simulated field plots:

- `N`, `G`, `V`, `d`, `dg`, `dgeom`, `dharm`, `h`, `hg`, `hgeom`, `hharm`, `d.0`, `dg.0`, `dgeom.0`, `dharm.0`, `h.0`, `hg.0`, `hgeom.0`, `hharm.0`: same description and format as indicated in the `fixed.area.plot` element.

TLS variables derived from simulated TLS plots:

- `N.tls`: same description and format as indicated in the `fixed.area.plot` element.
- `N.pam`: stand density (trees/ha) with occlusion correction based on gap probability attenuation with distance to TLS.
- `G.tls`: same description and format as indicated in `fixed.area.plot` element.
- `G.pam`: stand basal area (m^2/ha) with occlusion correction based on gap probability attenuation with distance to TLS.
- `V.tls`: same description and format as indicated in `fixed.area.plot` element.
- `V.pam`: stand volume (m^3/ha) with occlusion correction based on gap probability attenuation with distance to TLS.
- `d.tls`, `dg.tls`, `dgeom.tls`, `dharm.tls`,
  `h.tls`, `hg.tls`, `hgeom.tls`, `hharm.tls`,
  `d.0.tls`, `dg.0.tls`, `dgeom.0.tls`, `dharm.0.tls`,
  `h.0.tls`, `hg.0.tls`, `hgeom.0.tls`, `hharm.0.tls`

TLS metrics derived from simulated TLS plots:

- `num.points`, `num.points.est`, `num.points.hom`, `num.points.hom.est`, `P01`, `P05`, `P10`, `P20`, `P25`, `P30`, `P40`, `P50`, `P60`, `P70`, `P75`, `P80`, `P90`, `P95`, `P99`: same description and format as indicated in `fixed.area.plot` element.

Output Files

At the end of the simulation process, if the `save.result` argument is `TRUE`, the function will print all the elements described in 'Value' section and which are different from `NULL` to files. Data frames are written without row names in `dir.result` directory using the `write.csv` function from the **utils** package. The pattern used for naming these files is 'simulations.<plot design>.csv', where '<plot design>' is equal to "fixed.area.plot", "k.tree.plot" or "angle.count.plot" according to plot design.

Note

The simulation process implemented in this function is computationally intensive. Although the function currently uses the vroom function from the **vroom** package for reading large files and contains fast implementations of several critical calculations (C++ via **Rcpp** package), long computation times may be required when a large number of plots are considered, number of points in TLS point clouds are very high, or the radius, k or BAF sequences used in the simulation process are very long.

Using reduced point clouds (according to point cropping process implemented in the [normalize](#) function), rather than original ones, may be recommended in order to cut down on computing time. Another possibility would be to specify large increments for radius and BAF, and/or low maximum values for radius, number of trees and BAF in the `plot.parameters` argument. This would make the function more efficient, though there may be a notable loss of detail in the results generated.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

References

Astrup, R., Ducey, M. J., Granhus, A., Ritter, T., & von Lüpke, N. (2014). Approaches for estimating stand level volume using terrestrial laser scanning in a single-scan mode. *Canadian Journal of Forest Research*, **44(6)**, 666-676. doi:10.1139/cjfr20130535

IUFRO (1959). *Standarization of symbols in forest mensuration*. IUFRO, Wien, 32 pp.

Lovell, J. L., Jupp, D. L. B., Newnham, G. J., & Culvenor, D. S. (2011). Measuring tree stem diameters using intensity profiles from ground-based scanning lidar from a fixed viewpoint. *ISPRS Journal of Photogrammetry and Remote Sensing*, **66(1)**, 46-55. doi:10.1016/j.isprsjprs.2010.08.006

Seidel, D., & Ammer, C. (2014). Efficient measurements of basal area in short rotation forests based on terrestrial laser scanning under special consideration of shadowing. *iForest-Biogeosciences and Forestry*, **7(4)**, 227. doi:10.3832/ifor1084007

Strahler, A. H., Jupp, D. L. B., Woodcock, C. E., Schaaf, C. B., Yao, T., Zhao, F., Yang, X., Lovell, J., Culvenor, D., Newnham, G., Ni-Miester, W., & Boykin-Morris, W. (2008). Retrieval of forest structural parameters using a ground-based lidar instrument (Echidna®). *Canadian Journal of Remote Sensing*, **34(sup2)**, S426-S440. doi:10.5589/m08046

See Also

[tree.detection.single.scan](#), [tree.detection.multi.scan](#), [tree.detection.several.plots](#), [distance.sampling](#), [normalize](#).

Examples

```
# Load information of trees detected from TLS point clouds data corresponding to
# plots 1 and 2 from Rioja data set

data("Rioja.data")
example.tls <- subset(Rioja.data$tree.tls, id < 3)

# Compute detection probabilities using distance sampling methods

example.ds <- distance.sampling(example.tls)

# Load information of trees measured in field plots corresponding to plot 1 and 2
# from Rioja data set

example.field <- subset(Rioja.data$tree.field, id < 3)
```

```
# Establish directory where TXT file containing TLS point cloud corresponding to
# plot 1 from Rioja data set is located. For instance, current working directory

dir.data <- tempdir()

# Download example of TXT file corresponding to plots 1 and 2 from Rioja data set

download.file(url = "https://www.dropbox.com/s/w4fgcyezr2olj9m/Rioja_1.txt?dl=1",
              destfile = file.path(dir.data, "1.txt"), mode = "wb")

download.file(url = "https://www.dropbox.com/s/sghmw3zud424s11/Rioja_2.txt?dl=1",
              destfile = file.path(dir.data, "2.txt"), mode = "wb")

# Establish directory where simulation results corresponding to plots 1 and 2
# from Rioja data set will be saved. For instance, current working directory

dir.result <- tempdir()

# Compute metrics and variables for simulated TLS and field plots corresponding
# to plots 1 and 2 from Rioja data set
# Without occlusion correction based on distance sampling methods

sim <- simulations(tree.tls = example.tls, tree.field = example.field,
                   plot.parameters = data.frame(radius.max = 10, k.max = 20,
                                                 BAF.max = 2),
                   dir.data = dir.data, dir.result = dir.result)
```

---

tree.detection.multi.scan

Tree-Level Variables Estimation

Description

Detects trees from point clouds corresponding to TLS multi-scan approaches and SLAM devices. For each tree detected, the function calculates the central coordinates and estimates the diameter at 1.3 m above ground level (which is known as *dbh*, diameter at breast height) and classifies it as fully visible or partially occluded. Finally, the function obtains the number of points belonging to normal sections of trees (those corresponding to *dbh* +/- 5 cm) and estimates them for both original and reduced (with random selection process) point clouds.

Usage

```
tree.detection.multi.scan(data, single.tree = NULL,
                          dbh.min = 4, dbh.max = 200, h.min = 1.3,
                          geo.dist = 0.1,
```

```
                              tls.precision = NULL,
                 stem.section = c(0.7, 3.5), stem.range = NULL, breaks = NULL,
                         slice = 0.1, understory = NULL, bark.roughness = 1,
                         den.type = 1, d.mer = NULL,
                         segmentation = NULL,
                         plot.attributes = NULL, plot = TRUE,
                         threads = 1,
                      dir.data = NULL, save.result = TRUE, dir.result = NULL)
```

Arguments

data	Data frame with same description and format as indicated for [normalize](#) 'Value'.
single.tree	Optional argument to indicate if there is only one tree.
dbh.min	Optional minimum *dbh* (cm) considered for detecting trees. By default it will be set at 4 cm.
dbh.max	Optional maximum *dbh* (cm) considered for detecting trees. By default it will be set at 200 cm.
h.min	Optional minimum *h* (m) considered for detecting trees. By default it will be set at 1.3 m.
geo.dist	Local surface variation (also known as normal change rate, NCR). By default it will be set as 0.1. For better understanding of this argument see 'Details'.
tls.precision	Average point cloud precision in cm.
stem.section	Section free of noise (shurb, branches, etc.) considered to detect trees. If not specified, an automatic internal algorithm will be applied (see 'Details').
stem.range	Section considered to estimate straightness tree attributes.
breaks	Height above ground level (m) of slices considered for detecting trees. By default it will be considered all possible sections from 0.4 m to maximum height by 0.3 m intervals (+/- 5 cm).
slice	Slice width considered for detecting trees. By default it will be considered as 0.1 m.
understory	Optional argument to indicate if there is dense understory vegetation.
bark.roughness	Bark roughness established in 3 degrees (1 < 2 < 3). By default it will be considered as 1.
den.type	Numeric argument indicating the dendrometic type used to estimate volumen when there are not sections enough to fit a taper equation. Dendrometrics types available are the following: cylinder = 0, paraboloid = 1 (by default), cone = 2 and neiloid = 3.
d.mer	Top stem diameter (cm) considered to estimate commercial timber volume.
segmentation	Tree segmentation.
plot.attributes	
	Data frame with attributes at plot level. It must contain a column named id (character string or numeric value) with encoding coinciding with that used in id argument of [normalize](#) for identifying plots. If there are strata, another column named 'stratum' (numeric) will be required for other functionalities of

	FORTLS (see, for instance, `estimation.plot.size` or `metrics.variables`). If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the function will not add these possible plot attributes.
plot	Optional logical which indicates whether or not the normalized point cloud will be plot. If this argument is not specified by the user, it will be set to `TRUE` by default and, as consequence, the normalized point cloud will be plot.
threads	Number of threads.
dir.data	Optional character string naming the absolute path of the directory where LAS files containing TLS point clouds are located. `.Platform$file.sep` must be used as the path separator in `dir.data`. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to `dir.data` during the execution.
save.result	Optional logical which indicates whether or not the output files described in 'Output Files' section should be saved in `dir.result`. If this argument is not specified by the user, it will be set to `TRUE` by default and, as a consequence, the output files will be saved.
dir.result	Optional character string naming the absolute path of an existing directory where the files described in 'Output Files' section will be saved. `.Platform$file.sep` must be used as the path separator in `dir.result`. If this argument is not specified by the user, and the `save.result` is `TRUE`, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to `dir.result` during the execution.

Details

Slices determined by `breaks` argument are clustered using the DBSCAN algorithm (Ester et al., 1996) on the horizontal plane according to Cartesian coordinates (x, y). Before and after this process, several algorithms are used to remove noisy points and apply classification criteria to select the clusters of trees.

dbh is directly estimated for the section of 1.3 m above ground level, and estimated from other sections using *dbh~breaks* linear regression. Finally, the mean value of all estimates is provided in 'Value' as the *dbh* of the tree section.

Volume is estimated modelling stem profile as a paraboloid and calculating the volumes of revolution; where trees *dbh* are estimated in `tree.detection.single.scan`, and total heights are estimated as percentile 99 of z coordinate of points delimited by Voronoi polygons.

The number of points corresponding to a normal section (+/- 5 cm) is estimated in proportion to *dbh*, using the average number of points per radius unit as reference. In this respect, only tree sections fully visible at 1.3 m above ground level will be considered for estimating the average number of points.

Local surface variation (also known as normal change rate ,NCR), is a quantitative measure of curvature feature (Pauly et al., 2002). This is useful for distinguishing points belonging to fine branches and foliage (e.g. leaves, shrubs) and stem points (e.g. Jin et al., 2016; Zhang et al., 2019). Just as we considered 5 cm as suitable for calculating local surface variation for the stem separation in forests, according to other authors (Ma et al., 2015; Xia et al., 2015), we also established the NCR threshold as 0.1, according to Zhang et al. (2019). However, this argument (`ncr.threshold`) may be modified in order to use more appropriate values.

Value

Data frame with the following columns for every tree detected (each row corresponds to one tree detected):

`id`	Optional plot identification encoded as a character string or numeric. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the plot will be encoded as 1.
`file`	Optional file name identification encoded as character string or numeric. If it is null, the file will be encoded as `id` by default.
`tree`	tree numbering
`Coordinates`	Cartesian (according to https://en.wikipedia.org/wiki/Cartesian_coordinate_system notation):

- x: distance on x axis (m) of tree centre.
- y: distance on y axis (m) of tree centre.

Azimuthal angles:

- phi: angular coordinate (rad) of tree centre.

`h.dist`	horizontal distance (m) from plot centre to tree centre.
\emph{dbh}	estimated tree diameter (cm) at breast height (1.3 m).
\emph{h}	estimated tree total height (m).
\emph{h.com}	estimated commercial tree height (m) according to the top diameter defined in the argument `d.mer`.
\emph{v}	estimated tree stem volume (m^3).
\emph{v.com}	estimated commercial tree stem volume (m^3) according to the top diameter defined in the argument `d.mer`.
\emph{SS.max}	Maximum sagitta (see Prendes et al. (2022)).
\emph{sinuosity}	
	Sinuosity (see Prendes et al. (2022)).
\emph{lean}	Lean (see Prendes et al. (2022)).
`n.pts`	number of points corresponding to a normal section (+/- 5 cm) in the original point cloud.
`n.pts.red`	number of points corresponding to a normal section (+/- 5 cm) in the point cloud reduced by the point cropping process.
`n.pts.est`	number of points estimated for a normal section (+/- 5 cm) in the original point cloud.
`n.pts.red.est`	number of points estimated for a normal section (+/- 5 cm) in the point cloud reduced by the point cropping process.
`partial.occlusion`	
	yes (1) or no (0)

Output Files

At the end of the tree detection process, if the `save.result` argument is `TRUE`, the function will save the data frame described in 'Value' as a CSV file named 'tree.tls.csv'. The data frame will be written without row names in the `dir.result` directory by using `write.csv` function from the **utils** package.

Note

Although `tree.detection.multi.scan` also works with reduced point clouds, thus reducing the computing time, use of the original point cloud is recommended in order to detect more trees. This will also depend on forest conditions, especially those related to visibility. The more distant the trees are, the lower the density of points will be, and using reduced point clouds will therefore complicate detection of the most distant trees.

Note that `dbh.min` and `dbh.max` are important for avoiding outlier values when inventory data are used for reference purposes. Otherwise, knowledge about the autoecology of species could be used for filtering anomalous values of *dbh*.

The argument `breaks = 1.3` could be sufficient for detecting trees visible at *dbh*, involving lower computational cost. However, those trees not detected at *dbh*, may be estimated from lower and/or higher sections. Considering the three default sections in the argument `breaks = c(1.0, 1.3, 1.6)` maintains a good balance in the case study of this package.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

References

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

Jin, S., Tamura, M., & Susaki, J. (2016). A new approach to retrieve leaf normal distribution using terrestrial laser scanners. J. *Journal of Forestry Research*, **27(3)**, 631-638. doi:10.1007/s11676015-0204z

Ma, L., Zheng, G., Eitel, J. U., Moskal, L. M., He, W., & Huang, H. (2015). Improved salient feature-based approach for automatically separating photosynthetic and nonphotosynthetic components within terrestrial lidar point cloud data of forest canopies. *IEEE Transactions Geoscience Remote Sensing*, **54(2)**, 679-696. doi:10.1109/TGRS.2015.2459716

Pauly, M., Gross, M., & Kobbelt, L. P., (2002). Efficient simplification of point-sampled surfaces. In IEEE Conference on Visualization. (pp. 163-170). Boston, USA. doi:10.1109/VISUAL.2002.1183771

Prendes, C., Canga, E., Ordoñez, C., Majada, J., Acuna, M., & Cabo, C. (2022). Automatic assessment of individual stem shape parameters in forest stands from tls point clouds: Application in pinus pinaster. Forests, 13(3), 431. doi:10.3390/f13030431

Xia, S., Wang, C., Pan, F., Xi, X., Zeng, H., & Liu, H. (2015). Detecting stems in dense and homogeneous forest using single-scan TLS. *Forests*. **6(11)**, 3923-3945. doi:10.3390/f6113923

Zhang, W., Wan, P., Wang, T., Cai, S., Chen, Y., Jin, X., & Yan, G. (2019). A novel approach for the detection of standing tree stems from plot-level terrestrial laser scanning data. *Remote Sens*. **11(2)**, 211. doi:10.3390/rs11020211

See Also

normalize, tree.detection.single.scan, tree.detection.several.plots, distance.sampling, estimation.plot.size, simulations, metrics.variables

Examples

```
# Establishment of working directories (optional)
# By default here we propose the current working directory of the R process

dir.data <- getwd()
dir.result <- getwd()


# Loading example data of TLS multi-scan approach point cloud (LAZ file) to dir.data

download.file(
"www.dropbox.com/scl/fi/es5pfj87wj0g6y8414dpo/PiceaAbies.laz?rlkey=ayt21mbndc6i6fyiz2e7z6oap&dl=1",
                destfile = file.path(dir.data, "PiceaAbies.laz"),
                mode = "wb")

# Normalizing the whole point cloud data without considering arguments

pcd <- normalize(las = "PiceaAbies.laz",

                 id = "PiceaAbies",

                 scan.approach = "multi",

                 dir.data = dir.data, dir.result = dir.result)


# Tree detection without considering arguments

tree.tls <- tree.detection.multi.scan(data = pcd,

                                      threads = 2,

                                      dir.result = dir.result)
```

---

tree.detection.several.plots

Tree-Level Variables Estimation for Several Plots

Description

This function integrates both, the [normalize](#) and [tree.detection.single.scan](#) or [tree.detection.multi.scan](#) functions, generating the same 'Output Files' as indicated for these, and it returs the same 'Value' as described for [tree.detection.single.scan](#) or [tree.detection.multi.scan](#) respectively. However, this function is designed for working with several plots, producing a list of all scans considered automatically from LAS files.

Usage

```
tree.detection.several.plots(las.list, id.list = NULL, file = NULL,

                        scan.approach = "single",
                        pcd.red = NULL, normalized = NULL,
                        center.coord = NULL,
                        x.side = NULL, y.side = NULL,
                     max.dist = NULL, min.height = NULL, max.height = 50,
                        algorithm.dtm = "knnidw", res.dtm = 0.2,
                        csf = list(cloth_resolution = 0.5),
                        intensity = NULL, RGB = NULL, voxel_size = NULL,

                        single.tree = NULL,
                        dbh.min = 4, dbh.max = 200, h.min = 1.3,
                        geo.dist = 0.1,
                        tls.resolution = NULL, tls.precision = NULL,
                 stem.section = c(0.7, 3.5), stem.range = NULL, breaks = NULL,
                        slice = 0.1, understory = NULL, bark.roughness = 1,
                        den.type = 1, d.mer = NULL,

                        plot.attributes = NULL, plot = NULL,

                        threads = 1,

                     dir.data = NULL, save.result = TRUE, dir.result = NULL,
                        save.las = NULL)
```

Arguments

`las.list`	Character vector containing the names of all LAS files for analysis and belonging to TLS point cloud, including .las extension (see 'Examples')
`id.list`	Optional vector with plots identification encoded as character string or numeric. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the plots will be encoded with correlative numbers from 1 to n plots.
`file`	Optional vector containing files name identification encoded as character string or numeric value. If it is null, file will be encoded as id by default.
`scan.approach`	Character parameter indicating TLS single-scan ('single') or TLS multi-scan approach or SLAM point clouds ('multi') approaches. If this argument is not specified by the user, it will be set to 'multi' approach.

`pcd.red`	Optional argument to indicate if point cloud density must be reduced to detect trees.
`normalized`	Optional argument to establish as `TRUE` when point cloud is already normalized.
`center.coord`	Planimetric x and y center coordinate of the plots. They has to be introduced as a data frame object with the following columns names: 'id', 'x' and 'y'. They represent plot id, and center coordinates respectively.
`x.side`	x-side (m) of the plot when the plot is square or rectangular.
`y.side`	y-side (m) of the plot when the plot is square or rectangular.
`max.dist`	Optional maximum horizontal distance (m) considered from the plot centre. All points farther than `max.dist` will be discarded after the normalization process. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, all points will be used in processing, with `max.dist` representing the farthest point.
`min.height`	Optional minimum height (m) considered from ground level. All points below `min.height` will be discarded after the normalization process. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, all points will be used in processing, with `min.height` representing the lowest point.
`max.height`	Optional maximum height (m) considered from ground level. All points above `max.height` will be discarded after the normalization process. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, all points will be used in processing, with `max.height` representing the highest point.
`algorithm.dtm`	Algorithm used to generate the digital terrain model (DTM) from the TLS point cloud. There are two posible options based on spatial interpolation: 'tin' and 'knnidw' (see 'Details'). If this argument is not specified by the user, it will be set to 'knnidw' algorithm.
`res.dtm`	Numeric parameter. Resolution of the DTM generated to normalize point cloud (see 'Details'). If this argument is not specified by the user, it will be set to 0.2 m.
`csf`	List containing parameters of CSF algorithm: • `cloth_resolution`: by default 0.5.
`intensity`	Logical parameter useful when point clouds have intensity values. It may be useful in some internal process to filter data.
`RGB`	Logical parameter useful when point clouds are colorized, thus including values of RGB colors. It is based on the Green Leaf Algorithm (GLA) (see 'Details').
`voxel_size`	Defines the size of the 3D grid cells used for downsampling.
`single.tree`	Optional argument to indicate if there is only one tree.
`dbh.min`	Optional minimum *dbh* (cm) considered for detecting trees. By default it will be set at 4 cm.
`dbh.max`	Optional maximum *dbh* (cm) considered for detecting trees. By default it will be set at 200 cm.

h.min	Optional minimum *h* (m) considered for detecting trees. By default it will be set at 1.3 m.
geo.dist	Local surface variation (also known as normal change rate, NCR). By default it will be set as 0.1. For better understanding of this argument see 'Details'.
tls.resolution	List containing parameters of TLS resolution. This can be defined by the angle aperture:

- horizontal.angle: horizontal angle aperture (degrees).
- vertical.angle: vertical angle aperture (degrees).
- point.dist: distance (mm) between two consecutive points.
- tls.dist: distance (m) from TLS at which two consecutive points are separated by point.dist.

  If this argument is not specified by the user, it will be set to NULL by default and, as a consequence the function will stop giving an error message.

tls.precision	Optional argument indicating the average point cloud precision in cm.
stem.section	Section free of noise (shurb, branches, etc.) considered to detect trees. If not specified, an automatic internal algorithm will be applied (see 'Details').
breaks	Height above ground level (m) of slices considered for detecting trees. By default it will be considered all possible sections from 0.1 m to maximum height by 0.3 m intervals (+/- 5 cm).
stem.range	Section considered to estimate straightness tree attributes.
slice	Slice width considered for detecting trees. By default it will be considered as 0.1 m.
understory	Optional argument to indicate if there is dense understory vegetation.
bark.roughness	Bark roughness established in 3 degrees (1 < 2 < 3). By default it will be considered as 2.
den.type	Numeric argument indicating the dendrometic type used to estimate volumen when there are not sections enough to fit a taper equation. Dendrometrics types available are the following: cylinder = 0, paraboloid = 1 (by default), cone = 2 and neiloid = 3.
d.mer	Top stem diameter (cm) considered to estimate commercial timber volume.
plot.attributes	
	Data frame with attributes at plot level. It must contain a column named id (character string or numeric value) with encoding coinciding with that used in id argument of normalize for identifying plots. If there are strata, another column named 'stratum' (numeric) will be required for other functionalities of **FORTLS** (see, for instance, estimation.plot.size or metrics.variables). If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the function will not add these possible plot attributes.
plot	Optional logical which indicates whether or not the normalized point cloud will be plot. If this argument is not specified by the user, it will be set to TRUE by default and, as consequence, the normalized point cloud will be plot.
threads	Number of threads.

dir.data   Optional character string naming the absolute path of the directory where LAS files containing TLS point clouds are located. `.Platform$file.sep` must be used as the path separator in `dir.dat`. If this argument is not specified by the user, it will be set to `NULL` by default and, as consequence, the current working directory of the R process will be assigned to `dir.dat` during the execution.

save.result   Optional logical which indicates whether or not the output files described in 'Output Files' section should be saved in the `dir.result`. If this argument is not specified by the user, it will be set to `TRUE` by default and, as a consequence, the output files will be saved.

dir.result   Optional character string naming the absolute path of an existing directory where files described in 'Output Files' section will be saved. `.Platform$file.sep` must be used as the path separator in `dir.result`. If this argument is not specified by the user, and `save.result` is `TRUE`, it will be set to `NULL` by default and, as a consequence, the current working directory of the R process will be assigned to `dir.result` during the execution.

save.las   Optional logical which indicates whether or not the imput point cloud must be saved in `dir.result` as LAZ file.

Details

See [normalize](#), [tree.detection.single.scan](#) and [tree.detection.multi.scan](#) for further details.

Value

Data frame with the same description and format as [tree.detection.single.scan](#) and [tree.detection.multi.scan](#) 'Values'. In this case, the id of plots will be encoded with correlative numbers from 1 to n, where n is the number of LAS files included in `files` argument, and `file` column will be encoded as id, but including .las extension.

Output Files

At the end of the tree detection process, if the `save.result` argument is `TRUE`, the function will save both, the reduced point clouds as TXT files encoded according to `file` column of 'Value'; and the data frame with the tree list described in 'Value' as CSV file (see [normalize](#) and [tree.detection.single.scan](#) or [tree.detection.multi.scan](#) 'Output files'). All outputs are written without row names in the `dir.result` directory using [vroom_write](#) function from **vroom** package.

Note

This function has been developed for working with several plots, which will be the most common situation in forest inventory approaches. Nevertheless, several LAS files are not provided as examples due to problems with memory capacity.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

See Also

normalize, tree.detection.single.scan, tree.detection.multi.scan, distance.sampling, estimation.plot.size, simulations, metrics.variables.

Examples

```
# Establishment of working directories (optional)
# By default here we propose the current working directory of the R process

dir.data <- getwd()
dir.result <- getwd()

# Loading example data (LAZ files) to dir.data

download.file(
"www.dropbox.com/scl/fi/hzzrt0a39crdy6uvcj9el/PinusSylve1.laz?rlkey=svpwvorkm8889fgbnj14ns1f2&dl=1",
               destfile = file.path(dir.data, "PinusSylvestris1.laz"),
               mode = "wb")

download.file(
"www.dropbox.com/scl/fi/zeszze31jh5m1g4o3ns1o/PinusSylve2.laz?rlkey=wx72bi6ggdc7wedwgzupekp9k&dl=1",
               destfile = file.path(dir.data, "PinusSylvestris2.laz"),
               mode = "wb")


# Tree detection (TLS single-scan aproach)

id <- c("PinusSylvestris1", "PinusSylvestris2")

center.coord <- data.frame(id = id,
                           x = rep(0, length(id)),
                           y = rep(0, length(id)))

tree.tls <- tree.detection.several.plots(las.list = c("PinusSylvestris1.laz",
                                                      "PinusSylvestris2.laz"),

                                 id.list = id,

                                 center.coord = center.coord,

                              tls.resolution = list(point.dist = 7.67, tls.dist = 10),

                                 max.dist = 7.5,

                                 threads = 2)
```

---

tree.detection.single.scan

Tree-Level Variables Estimation for TLS Single-Scan Approach

Description

Detects trees from TLS point clouds corresponding to a single scan. For each tree detected, the function calculates the central coordinates and estimates the diameter at 1.3 m above ground level (which is known as *dbh*, diameter at breast height) and classifies it as fully visible or partially occluded. Finally, the function obtains the number of points belonging to normal sections of trees (those corresponding to *dbh* +/- 5 cm) and estimates them for both original and reduced (with point cropping process) point clouds.

Usage

```
tree.detection.single.scan(data, single.tree = NULL,
                           dbh.min = 4, dbh.max = 200, h.min = 1.3,
                           geo.dist = 0.1,
                           tls.resolution = list(),
                           tls.precision = NULL,
                    stem.section = c(0.7, 3.5), stem.range = NULL, breaks = NULL,
                           slice = 0.1, understory = NULL, bark.roughness = 1,
                           den.type = 1, d.mer = NULL, segmentation = NULL,
                           plot.attributes = NULL, plot = TRUE,
                           threads = 1,
                        dir.data = NULL, save.result = TRUE, dir.result = NULL)
```

Arguments

data	Data frame with same description and format as indicated for [normalize](#) 'Value'.
single.tree	Optional argument to indicate if there is only one tree.
dbh.min	Optional minimum *dbh* (cm) considered for detecting trees. By default it will be set at 4 cm.
dbh.max	Optional maximum *dbh* (cm) considered for detecting trees. By default it will be set at 200 cm.
h.min	Optional minimum *h* (m) considered for detecting trees. By default it will be set at 1.3 m.
geo.dist	Local surface variation (also known as normal change rate, NCR). By default it will be set as 0.1. For better understanding of this argument see 'Details'.
tls.resolution	List containing parameters of TLS resolution. This can be defined by the angle aperture:

- horizontal.angle: horizontal angle aperture (degrees).
- vertical.angle: vertical angle aperture (degrees).
- point.dist: distance (mm) between two consecutive points.

- `tls.dist`: distance (m) from TLS at which two consecutive points are separated by `point.dist`.
  If this argument is not specified by the user, it will be set to NULL by default and, as a consequence the function will stop giving an error message.

`tls.precision`	Average point cloud precision in cm.
`stem.section`	Section free of noise (shurb, branches, etc.) considered to detect trees. If not specified, an automatic internal algorithm will be applied (see 'Details').
`stem.range`	Section considered to estimate straightness tree attributes.
`breaks`	Height above ground level (m) of slices considered for detecting trees. By default it will be considered all possible sections from 0.1 m to maximum height by 0.3 m intervals (+/- 5 cm).
`slice`	Slice width considered for detecting trees. By default it will be considered as 0.1 m.
`understory`	Optional argument to indicate if there is dense understory vegetation.
`bark.roughness`	Bark roughness established in 3 degrees ($1 < 2 < 3$). By default it will be considered as 1.
`den.type`	Numeric argument indicating the dendrometic type used to estimate volumen when there are not sections enough to fit a taper equation. Dendrometrics types available are the following: cylinder = 0, paraboloid = 1 (by default), cone = 2 and neiloid = 3.
`d.mer`	Top stem diameter (cm) considered to estimate commercial timber volume.
`segmentation`	Tree segmentation.
`plot.attributes`	
	Data frame with attributes at plot level. It must contain a column named id (character string or numeric value) with encoding coinciding with that used in id argument of normalize for identifying plots. If there are strata, another column named 'stratum' (numeric) will be required for other functionalities of **FORTLS** (see, for instance, estimation.plot.size or metrics.variables). If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the function will not add these possible plot attributes.
`plot`	Optional logical which indicates whether or not the normalized point cloud will be plot. If this argument is not specified by the user, it will be set to TRUE by default and, as consequence, the normalized point cloud will be plot.
`threads`	Number of threads.
`dir.data`	Optional character string naming the absolute path of the directory where LAS files containing TLS point clouds are located. `.Platform$file.sep` must be used as the path separator in `dir.data`. If this argument is not specified by the user, it will be set to NULL by default and, as a consequence, the current working directory of the R process will be assigned to `dir.data` during the execution.
`save.result`	Optional logical which indicates whether or not the output files described in 'Output Files' section should be saved in `dir.result`. If this argument is not specified by the user, it will be set to TRUE by default and, as a consequence, the output files will be saved.

dir.result   Optional character string naming the absolute path of an existing directory where
                  the files described in 'Output Files' section will be saved. .Platform$file.sep
                  must be used as the path separator in dir.result. If this argument is not spec-
                  ified by the user, and the save.result is TRUE, it will be set to NULL by default
                  and, as a consequence, the current working directory of the R process will be
                  assigned to dir.result during the execution.

Details

Slices determined by breaks argument are clustered using the DBSCAN algorithm (Ester et al.,
1996) on the horizontal plane according to Cartesian coordinates (x, y). Before and after this pro-
cess, several algorithms are used to remove noisy points and apply classification criteria to select
the clusters of trees.

dbh is directly estimated for the section of 1.3 m above ground level, and estimated from other
sections using *dbh~breaks* linear regression. Finally, the mean value of all estimates is provided in
'Value' as the *dbh* of the tree section.

The number of points corresponding to a normal section (+/- 5 cm) is estimated in proportion to *dbh*,
using the average number of points per radius unit as reference. In this respect, only tree sections
fully visible at 1.3 m above ground level will be considered for estimating the average number of
points.

Local surface variation (also known as normal change rate ,NCR), is a quantitative measure of
curvature feature (Pauly et al., 2002). This is useful for distinguishing points belonging to fine
branches and foliage (e.g. leaves, shrubs) and stem points (e.g. Jin et al., 2016; Zhang et al., 2019).
Just as we considered 5 cm as suitable for calculating local surface variation for the stem separation
in forests, according to other authors (Ma et al., 2015; Xia et al., 2015), we also established the
NCR threshold as 0.1, according to Zhang et al. (2019). However, this argument (ncr.threshold)
may be modified in order to use more appropriate values.

Value

Data frame with the following columns for every tree detected (each row corresponds to one tree
detected):

id   Optional plot identification encoded as a character string or numeric. If this
                  argument is not specified by the user, it will be set to NULL by default and, as a
                  consequence, the plot will be encoded as 1.

file   Optional file name identification encoded as character string or numeric. If it is
                  null, the file will be encoded as id by default.

tree   tree numbering

Coordinates   Cartesian (according to https://en.wikipedia.org/wiki/Cartesian_coordinate_
                  system notation):

- x: distance on x axis (m) of tree centre.

- y: distance on y axis (m) of tree centre.

Azimuthal angles:

- phi: angular coordinate (rad) of tree centre.

- `phi.left`: angular coordinate (rad) of left border of tree section.
- `phi.right`: angular coordinate (rad) of right border of tree section

h.dist	horizontal distance (m) from plot centre to tree centre.
\emph{dbh}	estimated tree diameter (cm) at breast height (1.3 m).
\emph{h}	estimated tree total height (m).
\emph{h.com}	estimated commercial tree height (m) according to the top diameter defined in the argument d.mer.
\emph{v}	estimated tree stem volume (m^3).
\emph{v.com}	estimated commercial tree stem volume (m^3) according to the top diameter defined in the argument d.mer.
\emph{SS.max}	Maximum sagitta (see Prendes et al. (2022)).
\emph{sinuosity}	
	Sinuosity (see Prendes et al. (2022)).
\emph{lean}	Lean (see Prendes et al. (2022)).
n.pts	number of points corresponding to a normal section (+/- 5 cm) in the original point cloud.
n.pts.red	number of points corresponding to a normal section (+/- 5 cm) in the point cloud reduced by the point cropping process.
n.pts.est	number of points estimated for a normal section (+/- 5 cm) in the original point cloud.
n.pts.red.est	number of points estimated for a normal section (+/- 5 cm) in the point cloud reduced by the point cropping process.
partial.occlusion	
	yes (1) or no (0)

Output Files

At the end of the tree detection process, if the save.result argument is TRUE, the function will save the data frame described in 'Value' as a CSV file named 'tree.tls.csv'. The data frame will be written without row names in the dir.result directory by using write.csv function from the **utils** package.

Note

Although tree.detection also works with reduced point clouds, thus reducing the computing time, use of the original point cloud is recommended in order to detect more trees. This will also depend on forest conditions, especially those related to visibility. The more distant the trees are, the lower the density of points will be, and using reduced point clouds will therefore complicate detection of the most distant trees.

Note that dbh.min and dbh.max are important for avoiding outlier values when inventory data are used for reference purposes. Otherwise, knowledge about the autoecology of species could be used for filtering anomalous values of *dbh*.

The argument breaks = 1.3 could be sufficient for detecting trees visible at *dbh*, involving lower computational cost. However, those trees not detected at *dbh*, may be estimated from lower and/or higher sections. Considering the three default sections in the argument breaks = c(1.0, 1.3, 1.6) maintains a good balance in the case study of this package.

Author(s)

Juan Alberto Molina-Valero and Adela Martínez-Calvo.

References

Ester, M., Kriegel, H. P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd (Vol. 96, No. 34, pp. 226-231).

Jin, S., Tamura, M., & Susaki, J. (2016). A new approach to retrieve leaf normal distribution using terrestrial laser scanners. J. *Journal of Forestry Research*, **27(3)**, 631-638. doi:10.1007/s11676015-0204z

Ma, L., Zheng, G., Eitel, J. U., Moskal, L. M., He, W., & Huang, H. (2015). Improved salient feature-based approach for automatically separating photosynthetic and nonphotosynthetic components within terrestrial lidar point cloud data of forest canopies. *IEEE Transactions Geoscience Remote Sensing*, **54(2)**, 679-696. doi:10.1109/TGRS.2015.2459716

Pauly, M., Gross, M., & Kobbelt, L. P., (2002). Efficient simplification of point-sampled surfaces. In IEEE Conference on Visualization. (pp. 163-170). Boston, USA. doi:10.1109/VISUAL.2002.1183771

Prendes, C., Canga, E., Ordoñez, C., Majada, J., Acuna, M., & Cabo, C. (2022). Automatic assessment of individual stem shape parameters in forest stands from tls point clouds: Application in pinus pinaster. Forests, 13(3), 431. doi:10.3390/f13030431

Xia, S., Wang, C., Pan, F., Xi, X., Zeng, H., & Liu, H. (2015). Detecting stems in dense and homogeneous forest using single-scan TLS. *Forests*. **6(11)**, 3923-3945. doi:10.3390/f6113923

Zhang, W., Wan, P., Wang, T., Cai, S., Chen, Y., Jin, X., & Yan, G. (2019). A novel approach for the detection of standing tree stems from plot-level terrestrial laser scanning data. *Remote Sens*. **11(2)**, 211. doi:10.3390/rs11020211

See Also

normalize, tree.detection.multi.scan, tree.detection.several.plots, distance.sampling, estimation.plot.size, simulations, metrics.variables

Examples

```
# Establishment of working directories (optional)
# By default here we propose the current working directory of the R process

dir.data <- getwd()
dir.result <- getwd()


# Loading example data (LAZ file) to dir.data

download.file("https://www.dropbox.com/s/17yl25pbrapat52/PinusRadiata.laz?dl=1",
              destfile = file.path(dir.data, "PinusRadiata.laz"),
              mode = "wb")
```

```
# Normalizing the whole point cloud data without considering arguments

# Note that in the case of TLS single-scans, the coordinates of the plot center
# must be specified by the x.center and y.center arguments, which are
# x.center = 0 and y.center = 0 in this case

pcd <- normalize(las = "PinusRadiata.laz",

                 id = "PinusRadiata",

                 x.center = 0, y.center = 0,

                 dir.data = dir.data, dir.result = dir.result)


# Tree detection without considering arguments
# For this case study, TLS resolution was established as:
# point.dist = 6.34 mm and tls.dist = 10 m

tree.tls <- tree.detection.single.scan(data = pcd,

                                       tls.resolution = list(point.dist = 6.34, tls.dist = 10),

                                       threads = 2,

                                       dir.result = dir.result)
```

Index

65