Package 'forestr'

July 22, 2025

Type Package

Version 2.0.2

Title Ecosystem and Canopy Structural Complexity Metrics from LiDAR

Author Jeff Atkins [aut, cre],
Gil Bohrer [aut],
Robert Fahey [aut],
Brady Hardiman [aut],
Chrisopher Gough [aut],
Timothy Morin [aut],
Atticus Stovall [aut],
Naupaka Zimmerman [ctb, aut],
Chris Black [ctb]

URL https://github.com/atkinsjeff/forestr

Maintainer Jeff Atkins <jwatkins6@vcu.edu>

Description Provides a toolkit for calculating forest and canopy structural complexity metrics from terrestrial LiDAR (light detection and ranging). References: Atkins et al. 2018 <doi:10.1111/2041-210X.13061>; Hardiman et al. 2013 <doi:10.3390/f4030537>; Parker et al. 2004 <doi:10.1111/j.0021-8901.2004.00925.x>.

Depends R (>= 3.1.2)

Imports ggplot2, plyr, dplyr, stats, tools, viridis, tidyr, moments, tibble

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Suggests knitr, rmarkdown

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-14 19:20:05 UTC

Contents

adjust_by_user *Adjust by user height*

Description

adjust_by_user adjusts data based on the user height to acccount for the laser's distance from the ground.

Usage

```
adjust_by_user(df, user_height)
```

Arguments

df	the data frame of raw pcl data
user_height	the height of the laser off the ground as mounted on the user in meters

Details

The function adjust_by_user simply adds the height of the user to the return distances in the data frame to estimate true height.

Value

a data frame adjusted by height

Examples

```
# Adust raw data to account for user height as PCL is user-mounted and correction
# gives actual distance from ground.

pcl_adjusted <- adjust_by_user(pcl_coded, user_height = 1.05)
```

calc_enl *Calculate rugosity and other higher level complexity metrics*

Description

calc_enl calculates the effective number of layers in a canopy.

Usage

```
calc_enl(m)
```

Arguments

m	a data frame of VAI for x, z bins from

Value

the effective number of layers

Examples

```
# Calculates the effective number of layers
calc_enl(pcl_vai)
```

---

calc_gap_fraction *Calculate gap fraction*

Description

`calc_gap_fraction` produces clumping index based on gap fraction through the canopy.

Usage

```
calc_gap_fraction(m)
```

Arguments

m	the matrix of bin hits calculated as density of LiDAR returns for each x column.

Details

This is a specific function that works using the adjusted matrix to calculate gap fraction through the canopy. This function also returns clumping index.

Examples

```
calc_gap_fraction(pcl_vai)
```

calc_intensity *Intensity Statistics*

Description

`calc_intensity` calcualtes statisitcs from the intensity column of the PCL data

Usage

```
calc_intensity(df, filename)
```

Arguments

`df`	data frame of uncorrected PCL data
`filename`	name of file currently being processed

Details

The `calc_intensity` function calculates statistics about the intensity data in the PCL data, including min, max, sd, mean, median.

Value

statisics on the intensity data

Examples

```
intensity_stats <- calc_intensity(pcl_adjusted, filename = "UVA")
```

calc_rugosity *Calculate rugosity and other higher level complexity metrics*

Description

`calc_rugosity` calculates canopy structural complexity metrics from PCL data and prints them to the screen.

Usage

```
calc_rugosity(df, m, filename)
```

Arguments

`df`	is a LiDAR summary matrix data frame
`m`	matrix of light adjusted vai values.
`filename`	the name of the file currently being processed.

Details

This is a specific function calculates canopy rugosity and other metrics, including rumple, height metrics, etc.

Value

a series of metrics that describe canopy and ecosystem height, density, openness, cover, etc.

Examples

```
# Calculates metrics of canopy structural complexity.
calc_rugosity(pcl_summary, pcl_vai, filename = "")
```

---

calc_rumple *Calculates rumple*

Description

`calc_rumple` calculates canopy rumple.

Usage

```
calc_rumple(df)
```

Arguments

`df`	LiDAR summary matrix data frame

Details

This function uses the summary matrix created by the function `make_summary_matrix` to calculate canopy rumple, the relationship between outer canopy surface and the ground area.

Value

rumple for the canopy based on 2-D transect

Examples

```
calc_rumple(pcl_summary)
```

calc_tls_csc *Calculates rumple*

Description

calc_tls_csc calculates canopy structural complexity metrics from the tls vai matrix

Usage

```
calc_tls_csc(m, filename)
```

Arguments

m	matrix of vai data with mean leaf height column
filename	the name of the file being process0

Details

This is a specific function to calculate canopy structural complexity or CSC metrics from the VAI matrix imported in.

Value

csc metrics

Examples

```
## Not run:
calc_tls_csc(m)

## End(Not run)
```

calc_tls_mean_leaf_ht *Process single PCL transects.*

Description

calc_tls_mean_leaf_ht used in process_tls to calculate mean leaf height from tls slife

Usage

```
calc_tls_mean_leaf_ht(m)
```

Arguments

m	the vai matrix

Details

This function derives mean leaf height from x, z vai from TLS data.

Value

adds columns to the matrix of height.bin

Examples

```
# with designated file
## Not run: process_pcl("pcl_data.csv", marker.spacing = 10, user_height = 1.05, max.vai = 8)
```

`calc_vai` *Calculate vegetation area index (VAI) from normalized PCL data matrix*

Description

`calc_vai` calculates vegetation area index (VAI) from a normalized matrix of LiDAR data.

Usage

```
calc_vai(df, max.vai)
```

Arguments

`df`	data frame of pcl data that has been corrected for light extinction using the `normalize_pcl` function.
`max.vai`	the maximum value of column VAI. The default is 8. Should be a max value, not a mean.

Value

a matrix of vai by x, z in the canopy

Examples

```
pcl_vai <- calc_vai(pcl_norm, max.vai = 8)
```

code_hits *Code hits*

Description

code_hits classifies data values as canopy returns, sky returns, or data markers.

Usage

```
code_hits(df)
```

Arguments

df	a raw set of pcl data

Details

The function code_hits accounts for the NAs that are in the return distance column which are actually the sky hits (i.e. when the lidar does not record a canopy hit).

Examples

```
# classify data values that have been imported using read_pcl
pcl_coded <- code_hits(pcl_data)
```

csc_metrics *Cover and sky fraction estimates*

Description

csc_metrics creates first-order canopy structural metrics that do not require normalization

Usage

```
csc_metrics(df, filename, transect.length)
```

Arguments

df	data frame of uncorrected PCL data
filename	name of file currently being processed
transect.length	the length of the transect

Details

The csc_metrics function processes uncorrected PCL data to generate canopy structural complexity (CSC) metrics that do not require normalization (i.e. correction for light saturation based on Beer-Lambert Law). These metrics include: mean return height of raw data, sd of raw canopy height returns, maximum measured canopy height, scan density (the average no. of LiDAR returns per linear meter), and both openness and cover fraction which are used for gap fraction calcuations.

Value

slew of cover and sky fraction metrics

Examples

```
csc.metrics <- csc_metrics(pcl_adjusted, filename = "UVA", transect.length = 10)
```

---

get_transect_length *Get transect length of PCL transect (in meters)*

Description

get_transect_length acquires the length of a transect based on a known marker spacing of the data markers stored in pcl data.

Usage

```
get_transect_length(df, marker.spacing)
```

Arguments

df	data frame of unprocessed PCL data
marker.spacing	distance between transect markers, typically 5 or 10 m

Details

Returns the transect length of a given PCL file given a known marker spacing.

Value

length of transect

Examples

```
# Get the length of the transect given a known spacing between data markers
transect.length <- get_transect_length(pcl_data, marker.spacing = 10)
```

make_matrix *Make PCL matrix for higher level complexity measures*

Description

make_matrix produces a matrix of, x, z values in coordinate space with the number and type of each LiDAR return in each x, z bin combination

Usage

```
make_matrix(df)
```

Arguments

df	data frame of PCL data that has been processed with split_transect_from_pcl

Details

The make_matrix function munges data in to a data frame of x, z bins with the number of canopy hits located in each bin.

Value

sorted matrix of LiDAR returns for each x, z position

Examples

```
pcl_matrix <- make_matrix(pcl_split)
```

make_matrix_part_one *Make PCL matrix part one*

Description

make_matrix_part_one produces a matrix of, x, z values in coordinate space with the number and type of each LiDAR return in each x, z bin combination

Usage

```
make_matrix_part_one(df)
```

Arguments

df	data frame of PCL data that has been processed with

Value

sorted matrix of LiDAR returns for each x, z position

`make_matrix_part_two` *Make PCL matrix part two*

Description

`make_matrix_part_two` produces a matrix of, x, z values in coordinate space with the number and type of each LiDAR return in each x, z bin combination

Usage

```
make_matrix_part_two(df)
```

Arguments

`df`	data frame of PCL data that has been processed with

Value

sorted matrix of LiDAR returns for each x, z position

`make_summary_matrix` *Creates summary matrix*

Description

`make_summary_matrix` creates a summary matrix of data through data wrangling the VAI data frame.

Usage

```
make_summary_matrix(df, m)
```

Arguments

`df`	sorted data frame of processed PCL data
`m`	matrix of PCL hit density with x and z coordinates

Details

This makes a dataframe that is as long as a transect is. If the transect is 40 m, this data frame has 40 rows. As input, `make_summary_matrix` requires a data frame of values from `split_transects_from_pcl` first, and second, the data frame of VAI from the function `calc_vai`.

#' This function allows you to express your love of cats.

Value

a matrix of summary stats by each x and z coordinate position

Examples

```
pcl_summary <- make_summary_matrix(pcl_split, pcl_vai)
```

`normalize_pcl` *Normalize PCL data based on light saturation and attenuation*

Description

`normalize_pcl` normalizes a PCL matrix for occlusion.

Usage

```
normalize_pcl(df)
```

Arguments

`df`	data frame of pcl hit density processed from `make_matrix`

Details

This function corrects saturated columns of LiDAR data for occlusion based on assumptions from the Beer-Lambert Law.

Value

a data frame of PCL hit density corrected for light saturation and attentuation based on Beer's Law

Examples

```
pcl_norm <- normalize_pcl(pcl_matrix)
```

osbs — *PCL transect from Ordway-Swisher Biological Station, Florida, US.*

Description

A dataset that consists of one 40 m transect taken in a longleaf pine-oak savanna in North-central Florida. Data collected April, 2016 by J. Atkins and R. Fahey.

Usage

```
osbs
```

Format

A data frame with 10506 rows:

index index of raw data–position along transect

return_distance raw, uncorrected LiDAR return distances from laser

intensity intensity values as recorded by LiDAR system

Source

http://atkinsjeff.github.io

pcl_adjusted — *a data frame LiDAR returns that have been split to x and z position and coded and adjusted for user height*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_adjusted
```

Format

A data frame with 14576 rows:

index index of raw data–position along transect

return_distance raw, uncorrected LiDAR return distances from laser

intensity intensity values as recorded by LiDAR system

sky_hit lidar return that does not hit the canopy

can_hit lidar return that hits the canopy

marker negative value that indicates marker

@source http://atkinsjeff.github.io

`pcl_coded` *a data frame LiDAR returns that have been split to x and z position and coded*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_coded
```

Format

A data frame with 14576 rows:

index index of raw data–position along transect

return_distance raw, uncorrected LiDAR return distances from laser

intensity intensity values as recorded by LiDAR system

sky_hit lidar return that does not hit the canopy

can_hit lidar return that hits the canopy

marker negative value that indicates marker

@source http://atkinsjeff.github.io

`pcl_data` *PCL transect from the University of Virginia*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_data
```

Format

An object of class `data.frame` with 14576 rows and 3 columns.

Details

#' @format A data frame with 14576rows:

index index of raw data–position along transect

return_distance raw, uncorrected LiDAR return distances from laser

intensity intensity values as recorded by LiDAR system

Source

http://atkinsjeff.github.io

`pcl_diagnostic_plot` *PCL diagnostic plot*

Description

`pcl_diagnostic_plot` this function provides a diagnostic view of raw PCL data

Usage

```
pcl_diagnostic_plot(df, filename)
```

Arguments

`df`	data frame of unprocessed PCL data
`filename`	name of file currently being processed

Details

This function provides a graphic view of raw PCL data to check for equal data spacing and marker spacing

Value

a plot of PCL data showing marker spacing

Examples

```
# using the Ordway-Swisher Data set
pcl_diagnostic_plot(osbs)
```

pcl_matrix *a LiDAR hit density by x, z position*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_matrix
```

Format

A data frame with 1120 rows:

xbin x-bin position

zbin z-bin position

bin.hits number of LiDAR returns at each x- and z- bin

sky.hits total numer of sky hits per x column

can.hits total numer of canopy hits per x column

lidar.pulses no. of lidar pulses emitted per column

Freq no idea

@source http://atkinsjeff.github.io

pcl_norm *a data frame of normalized LiDAR return density*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_norm
```

Format

A data frame with 1120 rows:

.id column numbering

xbin x-bin position

zbin z-bin position

bin.hits number of LiDAR returns at each x- and z- bin

sky.hits total numer of sky hits per x column

can.hits total numer of canopy hits per x column

lidar.pulses no. of lidar pulses emitted per column

Freq no idea

hit.count total number of hits distributed through canopy

phi percent of saturation

dee percent of returns distributed

x.counter counting variable

sum.dee distributed proportion

fee coefficent

Source

http://atkinsjeff.github.io

---

pcl_split *a data frame LiDAR returns that have been split to x and z position*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_split
```

Format

A data frame with 13982 rows:

index index of raw data–position along transect

return_distance raw, uncorrected LiDAR return distances from laser

intensity intensity values as recorded by LiDAR system

sky_hit lidar return that does not hit the canopy

can_hit lidar return that hits the canopy

marker negative value that indicates marker

seg_num intermediate to get x position

chunk_num intermediate to get x position

xbin position along horizontal axis

zbin position along vertical axis

Source

http://atkinsjeff.github.io

`pcl_summary` *summary matrix*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_summary
```

Format

A data frame with 40 rows:

xbin x-bin position

mean.ht mean height

sd.ht standard deviation of mean leaf height

max.ht max measured height

max.vai highest measured max VAI

sum.vai total VAI for the column

sd.vai standard deviation of VAI

vai.z.sum density adjuste height

max.vai.z height of peak VAI

height.bin mean leaf height

Source

http://atkinsjeff.github.io

pcl_vai *a data frame of vegetation area index (VAI)*

Description

Derived from data collected at the University of Virginia Data collected August, 2016 by J. Atkins. Dervied from the calc_vai function

Usage

```
pcl_vai
```

Format

A data frame with 1120 rows:

.id column numbering

xbin x-bin position

zbin z-bin position

bin.hits number of LiDAR returns at each x- and z- bin

sky.hits total numer of sky hits per x column

can.hits total numer of canopy hits per x column

lidar.pulses no. of lidar pulses emitted per column

Freq no idea

hit.count total number of hits distributed through canopy

phi percent of saturation

dee percent of returns distributed

x.counter counting variable

sum.dee distributed proportion

fee coefficent

cvr cover proportion

olai max LAI or VAI number

vai calculated VAI

Source

http://atkinsjeff.github.io

plot_hit_grid *Plots LiDAR hit grids of VAI*

Description

plot_hit_grid produces a LiDAR hit grid plot

Usage

```
plot_hit_grid(m, filename, transect.length, max.ht, max.vai)
```

Arguments

m	matrix of light adjusted vai values.
filename	the name of the file currently being processed.
transect.length	the length of the transect used to create the x-axis
max.ht	the maximum measured height used to create the y-axis
max.vai	the maximum density of VAI, defaul = 8

Value

a hit gride of VAI

Examples

```
# Calculates metrics of canopy structural complexity.
plot_hit_grid(pcl_vai, filename = "UVA LiDAR data", transect.length = 40,
 max.ht = 30, max.vai = 8)
```

plot_pavd *Graphs Plant Area Volume Density Profiles*

Description

plot_pavd produces a PAVD plot from matrix data

Usage

```
plot_pavd(m, filename, plot.file.path.pavd, hist = FALSE, save_output = FALSE)
```

Arguments

m	matrix of light adjusted vai values.
filename	the name of the file currently being processed.
plot.file.path.pavd	path of plot file to be written, inherited from process_pcl or process_multi_pcl
hist	logical input to include histogram of VAI, if TRUE it is included, if FALSE, it is not.
save_output	if TRUE it saves the plot, if false it just runs

Details

This function is a nested function inside of process_pcl. It could be run independently using the summary_matrix.csv output files created from running procesS_pcl as well.

Value

plant area volume density plots

See Also

plot_hit_grid

Examples

```
# Calculates metrics of canopy structural complexity.
plot_pavd(pcl_vai, filename = "pcl_test", hist = FALSE, save_output = FALSE)
plot_pavd(pcl_vai, filename = "pcl_test", hist = TRUE, save_output = FALSE)
```

---

process_multi_pcl *Process multiplie PCL transects.*

Description

process_multi_pcl imports and processes mutiple PCL transect.

Usage

```
process_multi_pcl(
  data_dir,
  user_height,
  marker.spacing,
  max.vai,
  pavd = FALSE,
  hist = FALSE,
  save_output = TRUE
)
```

Arguments

data_dir	directory where PCL .csv files are stored
user_height	height of laser from ground based on user in meters
marker.spacing	space between markers in the PCL data, in meters
max.vai	the maximum value of column VAI. The default is 8. Should be a max value, not a mean.
pavd	logical input to include Plant Area Volume Density Plot from [plot_pavd], if TRUE it is included, if FALSE, it is not.
hist	logical input to include histogram of VAI with PAVD plot, if TRUE it is included, if FALSE, it is not.
save_output	needs to be set to true, or else you are just going to get a lot of data on the screen

Details

This is a specific function that works using the input of a data directory of .csv files where the function cycles through the files there and processes multiple files, producing the same output files described in process_pcl

Value

writes the hit matrix, summary matrix, and output variables to csv in an output folder, along with hit grid plot

See Also

process_pcl

Examples

```
# This function works on a directory of raw PCL data
## Not run: data_directory <- "./data/PCL_transects/" #data directory containing PCL transects
process_multi_pcl(data_directory, user_height = 1.05, marker.spacing = 10,
max.vai = 8, pavd = FALSE, hist = FALSE, save_output = FALSE)

process_multi_pcl("./data/PCL_transects/", user_height = 1.05, marker.spacing = 10,
max.vai = 8, pavd = FALSE, hist = FALSE, save_output = FALSE)

## End(Not run)
```

process_pcl *Process single PCL transects.*

Description

process_pcl imports and processes a single PCL transect.

Usage

```
process_pcl(
  f,
  user_height,
  marker.spacing,
  max.vai,
  pavd = FALSE,
  hist = FALSE,
  save_output = TRUE
)
```

Arguments

f	the name of the filename to input <character> or a data frame <data frame>.
user_height	the height of the laser off the ground as mounted on the user in meters. default is 1 m
marker.spacing	distance between markers, defaults is 10 m
max.vai	the maximum value of column VAI. The default is 8. Should be a max value, not a mean.
pavd	logical input to include Plant Area Volume Density Plot from plot_pavd, if TRUE it is included, if FALSE, it is not.
hist	logical input to include histogram of VAI with PAVD plot, if TRUE it is included, if FALSE, it is not.
save_output	the name of the output folder where to write all the output fiels.

Details

This function imports raw pcl data or existing data frames of pcl data and writes all data and analysis to a series of .csv files in an output directory (output) keeping nothing in the workspace.

process_pcl uses a workflow that cuts the data into 1 meter segments with z and x positions in coordinate space where x referes to distance along the ground and z refers to distance above the ground. Data are normalized based on light extinction assumptions from the Beer-Lambert Law to account for light saturation. Data are then summarized and metrics of canopy structure complexity are calculated.

process_pcl will write multiple output files to disk in an output directory that process_pcl creates within the work directing. These files include:

1. an output variables file that contains a list of CSC variables and is written by the subfunction `write_pcl_to_csv` 2. a summary matrix, that includes detailed information on each vertical column of LiDAR data written by the subfunction `write_summary_matrix_to_csv` 3. a hit matrix, which is a matrix of VAI at each x and z position, written by the subfunction `write_hit_matrix_to_pcl` 4. a hit grid, which is a graphical representation of VAI along the x and z coordinate space. 5. optionally, plant area/volume density profiles can be created by including `pavd = TRUE` that include an additional histogram with the optional `hist = TRUE` in the `process_pcl` call.

Value

writes the hit matrix, summary matrix, and output variables to csv in an output folder, along with hit grid plot

See Also

`process_multi_pcl`

Examples

```
# Run process complete PCL transect without storing to disk
uva.pcl <- system.file("extdata", "UVAX_A4_01W.csv", package = "forestr")

process_pcl(uva.pcl, marker.spacing = 10, user_height = 1.05,
max.vai = 8, pavd = FALSE, hist = FALSE, save_output = FALSE)

# with data frame
process_pcl(osbs, marker.spacing = 10, user_height = 1.05,
max.vai = 8, pavd = FALSE, hist = FALSE, save_output = FALSE)
```

---

process_tls *Process single PCL transects.*

Description

`process_tls` imports and processes a slice from a voxelated TLS scan.

Usage

```
process_tls(f, slice, pavd = FALSE, hist = FALSE, save_output = TRUE)
```

Arguments

`f`	the name of the filename to input <character> or a data frame <data frame>.
`slice`	the number of the transect to use from xyz tls data
`pavd`	logical input to include Plant Area Volume Density Plot from `plot_pavd`, if TRUE it is included, if FALSE, it is not.

hist	logical input to include histogram of VAI with PAVD plot, if TRUE it is included, if FALSE, it is not.
save_output	needs to be set to true, or else you are just going to get a lot of data on the screen

Details

This function takes as input a four column .CSV file or data frame of x, y, z, and VAI (Vegetation Area Index) derived from 3-D (TLS) LiDAR data. Currently, this function only analyzes a single slice from the inputed TLS data set. VAI is calculated externally by the user using user-determined methodology.

The process_tls function will write multiple output files to disk in an (output) directory that process_tls creates within the work directing. These files include:

1. an output variables file that contains a list of CSC variables and is written by the subfunction write_pcl_to_csv

2. a summary matrix, that includes detailed information on each vertical column of Lidar data written by the subfunction write_summary_matrix_to_csv

3. a hit matrix, which is a matrix of VAI at each x and z position, written by the subfunction write_hit_matrix_to_pcl

4. a hit grid, which is a graphical representation of VAI along the x and z coordinate space. 5. optionally, plant area/volume density profiles can be created by including pavd = TRUE that include an additional histogram with the optional hist = TRUE in the process_pcl call.

Value

writes the hit matrix, summary matrix, and output variables to csv in an output folder, along with hit grid plot

See Also

process_pcl

Examples

```
# with designated file
uva.tls<- system.file("extdata", "UVAX_A4_01_tls.csv", package = "forestr")

process_tls(uva.tls, slice = 5, pavd = FALSE, hist = FALSE, save_output = FALSE)
```

read_pcl *read_pcl imports PCL or portable canopy LiDAR files into the workspace and formats them.*

Description

This function specificially reads in PCL files that are in .csv format, standard format for that data type.

Usage

```
read_pcl(f)
```

Arguments

f	name of file currently being processed

See Also

process_pcl process_multi_pcl

Examples

```
# Link to raw PCL data, in .csv form.
uva_pcl <- system.file("extdata", "UVAX_A4_01W.csv", package = "forestr")

# Import PCL data to the workspace
pcl_data <-read_pcl(uva_pcl)
```

read_pcl_multi *read_pcl_multi imports PCL or portable canopy LiDAR files into the workspace and formats them.*

Description

This function specificially reads in PCL files that are in .csv format, standard format for that data type.

Usage

```
read_pcl_multi(data_directory, filename)
```

Arguments

data_directory	directory where files are stored
filename	name of file to be imported
	Zero-length vectors have sum 0 by definition. See http://en.wikipedia.org/wiki/Empty_sum for more details.

Examples

```
## Not run:
# This function runs internally right now.
read_pcl_multi(data_directory, filename)

## End(Not run)
```

---

red_pine *PCL transect from a red pine plantation in Northern Michigan, US.*

Description

A dataset that consists of one 40 m transect taken in a red pine plantations in Northern Michigan. Data collected July, 2017 by J. Atkins.

Usage

```
red_pine
```

Format

A data frame with 17559 rows:

index index of raw data–position along transect

return_distance raw, uncorrected LiDAR return distances from laser

intensity intensity values as recorded by LiDAR system

Source

http://atkinsjeff.github.io

split_transects_from_pcl *Split transects from PCL*

Description

split_transects_from_pcl places data values into x-bins (x-coordinates and) and z-bins (z-coordinates)

Usage

```
split_transects_from_pcl(
  pcl_data,
  transect.length,
  marker.spacing,
  DEBUG = FALSE,
  data_dir,
  output_file_name
)
```

Arguments

pcl_data	data frame of unprocessed PCL data.
transect.length	total transect length. Default value is 40 meters.
marker.spacing	distance between markers in meters within the PCL data. Default value is 10 m.
DEBUG	check to see order of final output. Default is FALSE.
data_dir	directory where PCL data .csv are stored if value is used.
output_file_name	old code relic that doesn't do much.

Details

Function to add two additional columns to the pcl dataset, one for the segment (which should only be from 1-4) and is designated by a -99999999 value in the return_distance column The only required parameters are the data frame of pcl data, with the length of transect and the marker spacing.

Examples

```
# Function that has the algorithm that splits the raw data into defined, equidistant x-bins.
 pcl_split <- split_transects_from_pcl(pcl_adjusted,
 transect.length = 40, marker.spacing = 10)
```

write_hit_matrix_to_csv *Writes hit matrix to csv for further analysis*

Description

write_hit_matrix_to_csv writes hit matrix to .csv for further analysis

Usage

```
write_hit_matrix_to_csv(m, outputname, output_directory)
```

Arguments

m	matrix of VAI with z and x coordinates
outputname	name of file currently being processed
output_directory	directory where output goes

Details

This is a specific sub-function that writes the output variables to disk in .csv format and runs within the functions process_pcl, process_multi_pcl, and proces_tls.

See Also

process_pcl write_pcl_to_csv write_summary_matrix_to_csv

Examples

```
## Not run:
# This function runs internally.
write_hit_matrix_to_csv(m, outputname, output_directory)

## End(Not run)
```

write_pcl_to_csv *Writes csc metrics and output variables to .csv*

Description

write_pcl_to_csv writes csc metrics and varialbes to .csv format

Usage

```
write_pcl_to_csv(output.variables, outputname, output_directory)
```

Arguments

output.variables	list of concatenated output variables
outputname	name of file currently being processed
output_directory	directory where output goes

Details

This is a specific function that writes the output variables to disk in .csv format and runs within the functions process_pcl, process_multi_pcl, and proces_tls.

See Also

process_pcl write_summary_matrix_to_csv write_hit_matrix_to_csv

Examples

```
## Not run:
write_pcl_to_csv(output_variables, outputname, output_directory)

## End(Not run)
```

---

write_summary_matrix_to_csv

Writes csc metrics and output variables to .csv

Description

write_summary_matrix_to_csv writes summary matrix to .csv format

Usage

```
write_summary_matrix_to_csv(m, outputname, output_directory)
```

Arguments

m	summary matrix
outputname	name of file currently being processed
output_directory	directory where output goes

Details

This is a specific subfunction that writes the summary matrix to disk in .csv format and runs within the functions process_pcl, process_multi_pcl, and proces_tls.

See Also

write_pcl_to_csv write_hit_matrix_to_csv

Examples

```
## Not run:
write_summary_matrix_to_csv()

## End(Not run)
```

Index