Package: CrownScorchTLS (via r-universe)

June 9, 2026

Type Package

Title Estimate Crown Scorch from Terrestrial LiDAR Scans

Version 0.1.1

Description Estimates tree crown scorch from terrestrial lidar scans collected with a RIEGL vz400i. The methods follow those described in Cannon et al. (2025, Fire Ecology 21:71, <doi:10.1186/s42408-025-00420-0>).

License GPL-3

Encoding UTF-8

URL https://github.com/jbcannon/CrownScorchTLS

BugReports https://github.com/jbcannon/CrownScorchTLS/issues

RoxygenNote 7.3.3

Imports lidR, randomForest, tidyr, Rcpp

LinkingTo Rcpp, RcppArmadillo,RcppEigen, BH

Suggests knitr, rmarkdown, dplyr, Boruta

VignetteBuilder knitr

NeedsCompilation yes

Author Jeffery Cannon [aut, cre], Andrew Whelan [ctb]

Maintainer Jeffery Cannon <Jeffery.Cannon@jonesctr.org>

Config/pak/sysreqs libabsl-dev cmake libfreetype6-dev libgdal-dev gdal-bin libgeos-dev libglu1-mesa-dev make texlive libicu-dev libpng-dev libuv1-dev libgl1-mesa-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Repository https://cranhaven.r-universe.dev

Date/Publication 2026-06-09 07:02:00 UTC

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/CrownScorchTLS

RemoteSha 8ea9a454529541de058a9997d6c20d1867e3f17e

RemoteSubdir CrownScorchTLS

Contents

`add_reflectance` *Add Reflectance column to LAS if it is missing for RIEGL vz400i*

Description

Function to provide relative Reflectance for RIEGL vz400i. Lidar prediction of crown scorch is based on relative range-corrected reflectance relative to a white reference object orthonormal to scanner. Raw range-corrected amplitudes from RIEGL vz400i are linearly correlated to relative intensity which usually ranges from -20 dB to 0 dB

Usage

```
add_reflectance(las)
```

Arguments

`las`	'LAS' object from 'lidR' package representing an individually segmented tree containing an 'Intensity' column representing 16-bit range- corrected amplitude from RIEGL vz400i Terrestrial Lidar Scanner

Value

modified LAS object with Reflectance column

Examples

```
library(lidR)
library(CrownScorchTLS)

 #download external data from github repo
url <- paste0(
  "https://raw.githubusercontent.com/jbcannon/CrownScorchTLS-data/main/data/manual-clip-trees/",
  "M-04-15549_post.laz")
 las_file = tempfile(fileext = paste0(".", tools::file_ext(url)))
 download.file(url, las_file, mode = "wb", quiet = TRUE)
 las <- readLAS(las_file)

 # or load your own data
 #las <- readLAS('C:/path/to/your/file.laz')
```

```
las = add_reflectance(las)
colnames(las@data)
```

`get_histogram` *Generate histogram of Reflectance for prediction with random forests*

Description

Generates a histogram of Reflectance intensities for prediction with Random Forests. Histogram breaks can be defined.

Usage

```
get_histogram(las, breaks = seq(-20, 0, by = 0.2))
```

Arguments

`las`	'LAS' object from 'lidR' package representing an individually segmented tree containing a 'Reflectance' column representing relative reflectance from from RIEGL vz400i Terrestrial Lidar Scanner. See 'add_reflectance()'
`breaks`	sequence of breaks for histograms, default from Cannon et al. 2025.

Value

data.frame with columns intensity and density

Examples

```
library(lidR)
library(CrownScorchTLS)
 #download external data from github repo
url <- paste0(
 "https://raw.githubusercontent.com/jbcannon/CrownScorchTLS-data/main/data/manual-clip-trees/",
  "M-04-15549_post.laz")
 las_file = tempfile(fileext = paste0(".", tools::file_ext(url)))
 download.file(url, las_file, mode = "wb", quiet = TRUE)
 las <- readLAS(las_file)

 # or load your own data
 #las <- readLAS('C:/path/to/your/file.laz')

las = add_reflectance(las)
histogram = get_histogram(las)
plot(density ~ intensity, data=histogram, xlab='Reflectance (dB)', type='l')
```

predict_scorch *Predict canopy scorch from 'LAS' tree object following Cannon et al. 2025*

Description

This function follows methods in Cannon et al. 2025 to predict crown scorch of a 'LAS' object representing an individual tree collected using a RIEGL vz400i Terrestrial Lidar system. The function uses the 'relative reflectance' (in decibels) and predicts crown scorch using 'randomForests' following Cannon et al. 2025

Usage

```
predict_scorch(las, model = NULL, plot = FALSE)
```

Arguments

las	'LAS' object from 'lidR' package representing an individually segmented tree collected from RIEGL vz400i Terrestrial Lidar Scanner
model	'randomForests' model object containing histogram data generated from 'get_histogram' function. if 'model' is 'NULL', then default model from Cannon et al. 2025 is used. But custom model may be generated.
plot	Boolean indicating whether reflectance histogram should be plotted

Value

predicted scorch as numeric vector

Examples

```
library(lidR)
library(CrownScorchTLS)

#download external data from github repo
url <- paste0(
 "https://raw.githubusercontent.com/jbcannon/CrownScorchTLS-data/main/data/manual-clip-trees/",
  "M-04-15549_post.laz")
las_file = tempfile(fileext = paste0(".", tools::file_ext(url)))
download.file(url, las_file, mode = "wb", quiet = TRUE)
las <- readLAS(las_file)

# or load your own data
#las <- readLAS('C:/path/to/your/file.laz')

predict_scorch(las) #using default model from Cannon et al. 2025
```

remove_stem *Remove tree bole from 'LAS'*

Description

This function identifies and removes tree boles using the 'TreeLS' package available at https://github.com/tiagodc/TreeLS

Usage

```
remove_stem(las)
```

Arguments

las	'LAS' object from 'lidR' package representing an individually segmented tree

Value

LAS object with stem removed

Examples

```
library(lidR)
library(CrownScorchTLS)

#'  #download external data from github repo
url <- paste0(
 "https://raw.githubusercontent.com/jbcannon/CrownScorchTLS-data/main/data/manual-clip-trees/",
  "M-04-15549_post.laz")
 las_file = tempfile(fileext = paste0(".", tools::file_ext(url)))
 download.file(url, las_file, mode = "wb", quiet = TRUE)
 las <- readLAS(las_file)

 # or load your own data
 #las <- readLAS('C:/path/to/your/file.laz')

#plot(las)
crown_only = remove_stem(las)
#plot(crown_only)
```

`stemPoints` *Stem points classification*

Description

Classify stem points of all trees in a **normalized** point cloud. Stem denoising methods are prefixed by `stm`. This file includes code derived from the TreeLS package by Tiago de Conto Original source: https://github.com/tiagodc/TreeLS License: GPL-3 The code below is copied and adapted from TreeLS::stemPoints for the purpose of maintaining CRAN compatibility. All modifications are clearly documented.

Usage

```
stemPoints(las, method = stm.hough())
```

Arguments

`las`	LAS object.
`method`	Function to classify stems. Default: `stm.hough`.

Value

LAS object.

Note

This function includes code derived from TreeLS::stemPoints (GPL-3 license). See source for details. #' @examples library(lidR) library(CrownScorchTLS)

#download external data from github repo url <- paste0("https://raw.githubusercontent.com/jbcannon/CrownScorchTLS-data/main/data/manual-clip-trees/", "M-04-15549_post.laz") las_file = tempfile(fileext = paste0(".", tools::file_ext(url))) download.file(url, las_file, mode = "wb", quiet = TRUE) las <- readLAS(las_file)

or load your own data #las <- readLAS('C:/path/to/your/file.laz')

las$Z = las$Z - min(las$Z) # Normalize las las <- stemPoints(las) # Classify stem points #plot(las, color='Stem')

References

Carvalho, T. (2017). TreeLS: Tools for Terrestrial LiDAR in R. GitHub: https://github.com/tiagodc/TreeLS

`stm.hough` *Stem denoising algorithm: Hough Transform*

Description

This function is meant to be used inside `stemPoints`. It applies an adapted version of the Hough Transform for circle search. Mode details are given in the sections below. This file includes code derived from the TreeLS package by Tiago de Conto Original source: https://github.com/tiagodc/TreeLS License: GPL-3 The code below is copied and adapted from TreeLS::stemPoints for the purpose of maintaining CRAN compatibility. All modifications are clearly documented.

Usage

```
stm.hough(
  h_step = 0.5,
  max_d = 0.5,
  h_base = c(1, 2.5),
  pixel_size = 0.025,
  min_density = 0.1,
  min_votes = 3
)
```

Arguments

`h_step`	`numeric` - height interval to perform point filtering/assignment/classification.
`max_d`	`numeric` - largest tree diameter expected in the point cloud.
`h_base`	`numeric` vector of length 2 - tree base height interval to initiate circle search.
`pixel_size`	`numeric` - pixel side length to discretize the point cloud layers while performing the Hough Transform circle search.
`min_density`	`numeric` - between 0 and 1 - minimum point density within a pixel evaluated on the Hough Transform - i.e. only *dense* point clousters will undergo circle search.
`min_votes`	`integer` - Hough Transform parameter - minimum number of circle intersections over a pixel to assign it as a circle center candidate.

Value

`LAS` object.

`LAS@data` Special Fields

Meaninful new fields in the output:

- `Stem`: `TRUE` for stem points
- `Segment`: stem segment number (from bottom to top and nested with TreeID)
- `Radius`: approximate radius of the point's stem segment estimated by the Hough Transform - always a multiple of the `pixel_size`

- `Votes`: votes received by the stem segment's center through the Hough Transform

#'

Adapted Hough Transform

The Hough Transform circle search algorithm used in TreeLS applies a constrained circle search on discretized point cloud layers. Tree-wise, the circle search is recursive, in which the search for circle parameters of a stem section is constrained to the *feature space* of the stem section underneath it. Initial estimates of the stem's *feature space* are performed on a *baselise* stem segment - i.e. a low height interval where a tree's bole is expected to be clearly visible in the point cloud. The algorithm is described in detail by Conto et al. (2017).

This adapted version of the algorithm is very robust against outliers, but not against forked or leaning stems.

Note

This function includes code derived from TreeLS::stemPoints (GPL-3 license). See source for details.

References

Carvalho, T. (2017). TreeLS: Tools for Terrestrial LiDAR in R. GitHub: https://github.com/tiagodc/TreeLS

Olofsson, K., Holmgren, J. & Olsson, H., 2014. Tree stem and height measurements using terrestrial laser scanning and the RANSAC algorithm. Remote Sensing, 6(5), pp.4323–4344.

Conto, T. et al., 2017. Performance of stem denoising and stem modelling algorithms on single tree point clouds from terrestrial laser scanning. Computers and Electronics in Agriculture, v. 143, p. 165-176.

Index