Package: ICESat2VegR (via r-universe)

October 23, 2024

Type Package

Title NASA's Ice, Cloud, and Elevation Satellite (ICESat-2) Data Analysis for Land and Vegetation Applications

Version 0.0.1

Description Set of tools for downloading, reading, visualizing, processing and exporting NASA's ICESat-2 ATL03 (Global Geolocated Photon Data) and ATL08 (Land and Vegetation Height) products for Land and Vegetation Applications.

License GPL (>= 3)

Imports curl, data.table, fs, getPass, jsonlite, hdf5r, httr2, magrittr, methods, Rcpp, Rdpack, R6, randomForest, reticulate, terra, xml2

Encoding UTF-8

URL `https://github.com/carlos-alberto-silva/ICESat2VegR`

BugReports `https://github.com/carlos-alberto-silva/ICESat2VegR/issues`

LazyData true

NeedsCompilation yes

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)

RdMacros Rdpack

Collate 'ANNIndex.R' 'lazy_applier.R' 'ATL03_ATL08_compute_seg_attributes_dt_segStat.R' 'utmTools.R' 'ATL03_ATL08_photons_attributes_dt_LAS.R' 'ATL03_ATL08_photons_attributes_dt_clipBox.R' 'ATL03_ATL08_photons_attributes_dt_clipGeometry.R' 'ATL03_ATL08_photons_attributes_dt_gridStat.R' 'ATL03_ATL08_photons_attributes_dt_join.R' 'ATL03_ATL08_photons_attributes_dt_polyStat.R' 'ATL03_ATL08_photons_seg_dt_fitground.R' 'ATL03_ATL08_photons_seg_dt_height_normalize.R' 'ATL03_ATL08_seg_cover_dt_compute.R'

'ATL03_ATL08_segment_create.R' 'ATL03_h5_clip.R' 'clipTools.R'
'ATL03_h5_clipBox.R' 'ATL03_h5_clipGeometry.R'
'ATL03_photons_attributes_dt.R' 'lasTools.R'
'ATL03_photons_attributes_dt_LAS.R'
'ATL03_photons_attributes_dt_clipBox.R'
'ATL03_photons_attributes_dt_clipGeometry.R'
'ATL03_photons_dt_classify.R' 'ATL03_photons_plot.R'
'class_tools.R' 'class.icesat2.R' 'zzz.R' 'ATL03_read.R'
'ATL03_seg_attributes_dt.R' 'ATL08_read.R' 'ATL08_h5_clip.R'
'ATL08_h5_clipBox.R' 'ATL08_photons_attributes_dt.R'
'ATL08_photons_attributes_dt_LAS.R' 'ATL08_seg_attributes_dt.R'
'ATL08_seg_attributes_dt_LAS.R'
'ATL08_seg_attributes_dt_clipBox.R'
'ATL08_seg_attributes_dt_clipGeometry.R'
'ATL08_seg_attributes_dt_gridStat.R'
'ATL08_seg_attributes_dt_polyStat.R'
'ATL08_seg_attributes_h5_gridStat.R' 'ATLAS_dataDownload.R'
'ATLAS_dataFinder.R' 'earthaccess.R' 'ATLAS_dataFinder_cloud.R'
'ATLAS_dataFinder_direct.R' 'ICESat2VegR_configure.R'
'gee-base.R' 'addEEImage.R' 'argParse.R'
'class.icesat2.h5ds_cloud.R' 'class.icesat2.h5_cloud.R'
'class.icesat2.h5ds_local.R' 'class.icesat2.h5_local.R'
'clip.R' 'extract.R' 'gdalBindings.R' 'gee-base-feature.R'
'gee-base-list.R' 'gee-base-number.R' 'gee-search.R'
'map_create.R' 'model_fit.R' 'model_tools.R' 'predict.R'
'predict_h5.R' 'rasterize_h5.R' 'rgt_extract.R' 'sample.R'
'stats_model.R' 'to_vect.R' 'var_select.R'

Suggests chromote, devtools, e1071, ggplot2, grid, gridExtra, htmlwidgets, knitr, leaflet, leafsync, lidR, lwgeom, mapview, nnet, png, rmarkdown, rstudioapi, signal, stars, testthat (>= 3.0.0), webshot, viridis, yaImpute

Config/testthat/edition 3

LinkingTo Rcpp

VignetteBuilder knitr

Repository https://carlos-alberto-silva.r-universe.dev

RemoteUrl https://github.com/carlos-alberto-silva/ICESat2VegR

RemoteRef HEAD

RemoteSha 9216b578e28d010bca4b0343a78bf03e44139d84

Contents

`addEEImage` *Adds Earth Engine Image class to leaflet*

Description

Adds Earth Engine Image class to leaflet

Usage

```
addEEImage(
  map,
  x,
  bands,
  min_value = 0,
  max_value = 1,
  palette = c("red", "green"),
  ...
)
```

Arguments

`map`	`leaflet::leaflet`. A leaflet class map.
`x`	Earth Engine Image open with `ee$Image`.
`bands`	define the bands that should be used for visualization, default NULL, will use the available bands from the Image.
`min_value`	The minimum value to represent for visualization purposes
`max_value`	The maximum value to represent for visualization purposes
`palette`	A `character` describing a list of colors in either hexadecimal or valid CSS color names.
`...`	Other parameters to be passed to `leaflet::addTiles()` function.

Value

A `leaflet::leaflet/htmlwidget` with the Earth Engine image added.

Examples

```
## Not run:
collection_id <- "NASA/HLS/HLSL30/v002"

ee_collection <- ee$ImageCollection(collection_id)

cloudMask <-
  2^1 +
  2^2 +
  2^3

hlsMask <- function(image) {
  image$updateMask(
    !(image[["Fmask"]] & cloudMask)
  )
}

image <- c(
  ee_collection$filterDate("2020-04-01", "2020-07-31")$map(hlsMask),
  ee_collection$filterDate("2021-04-01", "2021-07-31")$map(hlsMask)
)$filter("CLOUD_COVERAGE < 30")$median()

if (require("leaflet")) {
  leaflet() %>%
    addEEImage(
      image,
      bands = list(3, 2, 1),
      min_value = 0.001,
      max_value = 0.2
    ) %>%
    setView(lng = -82.2345, lat = 29.6552, zoom = 10)
  }

## End(Not run)
```

`addEEImage,leaflet,ee.image.Image-method`
Adds Earth Engine Image class to leaflet map

Description

This method handles adding an Earth Engine Image to a leaflet map.

Usage

```
## S4 method for signature 'leaflet,ee.image.Image'
addEEImage(
  map,
  x,
  bands = NULL,
  min_value = 0,
  max_value = 1,
  palette = defaultPallete,
  ...
)
```

Arguments

`map`	A `leaflet::leaflet` object. A leaflet map to which the Earth Engine image will be added.
`x`	An Earth Engine Image object created with `ee$Image`.
`bands`	A vector defining the bands to be used for visualization. Default is NULL, which uses the available bands from the image.
`min_value`	The minimum value for visualization purposes. Default is 0.
`max_value`	The maximum value for visualization purposes. Default is 1.
`palette`	A character vector describing a list of colors in either hexadecimal or valid CSS color names. Default is a predefined palette.
`...`	Other parameters to be passed to the `leaflet::addTiles()` function.

Value

A `leaflet::leaflet/htmlwidget` object with the Earth Engine image added.

`ANNIndex` *ANNIndex Class*

Description

This class uses Approximate Neareast Neighbor Index for finding points that are within a specified range.

Public fields

`tree` The C++ pointer for the built tree

Methods

Public methods:

- `ANNIndex$new()`
- `ANNIndex$searchFixedRadius()`

Method `new()`: Creates a new instance of the `ANNIndex` class.

Usage:

```
ANNIndex$new(x, y)
```

Arguments:

`x` NumericVector of x/longitude

`y` NumericVector of y/latitude

`radius` the minimum radius between the points

Method `searchFixedRadius()`: Given a x, y point get the indexes that are within a specified radius.

Usage:

```
ANNIndex$searchFixedRadius(x, y, radius)
```

Arguments:

`x` NumericVector of x/longitude

`y` NumericVector of y/latitude

`radius` the minimum radius between the points

See Also

Mount, D. M.; Arya, S. ANN: A Library for Approximate Nearest Neighbor Searching, available in https://www.cs.umd.edu/~mount/ANN/

`aspect` *Calculates aspect in degrees from a terrain DEM.*

Description

The local gradient is computed using the 4-connected neighbors of each pixel, so missing values will occur around the edges of an image.

Usage

```
aspect(x)
```

Arguments

`x`	The `ee.Image` on which to apply the aspect function.

Value

An `ee.Image` with a single band named "Aspect".

`atan2.ee.ee_number.Number` *Calculates the angle formed by the 2D vector x, y.*

Description

Calculates the angle formed by the 2D vector x, y.

Usage

```
atan2.ee.ee_number.Number(x)
```

Arguments

`x`	The number to calculate the `atan2` on.

Value

The resulting ee.Number with the results from `atan2`.

`ATL03_ATL08_compute_seg_attributes_dt_segStat` *Statistics of ATL03 and ATL08 labeled photons at the segment level*

Description

Computes a series of statistics from ATL03 and ATL08 labeled photons within a given segment length.

Usage

```
ATL03_ATL08_compute_seg_attributes_dt_segStat(
  atl03_atl08_seg_dt,
  list_expr,
  ph_class = c(0, 1, 2, 3),
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  quality_ph = NULL,
  night_flag = NULL
)
```

Arguments

atl03_atl08_seg_dt	An S4 object of class icesat2.atl03_atl08_seg_dt containing ATL03 and ATL08 data (output of ATL03_ATL08_photons_attributes_dt_join() function).
list_expr	The function to be applied for computing the defined statistics
ph_class	Character vector indicating photons to process based on the classification (1=ground, 2=canopy, 3=top canopy), Default is c(2,3)
beam	Character vector indicating beams to process. Default is c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
quality_ph	Indicates the quality of the associated photon. 0 = nominal, 1 = possible_afterpulse, 2 = possible_impulse_response_ effect, 3=possible_tep. Default is 0
night_flag	Flag indicating the data were acquired in night conditions: 0=day, 1=night. Default is 1

Value

Returns an S4 object of class icesat2.atl08_dt Containing Statistics of ATL03 and ATL08 labeled photons

Examples

```
# Specifying ATL03 and ATL08 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL03 and ATL08 labeled photons
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)

# Computing the max canopy height at 30 m segments
atl03_atl08_dt_seg <- ATL03_ATL08_segment_create(atl03_atl08_dt, segment_length = 30)

max_canopy <- ATL03_ATL08_compute_seg_attributes_dt_segStat(atl03_atl08_dt_seg,
  list_expr = max(ph_h),
  ph_class = c(2, 3),
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  quality_ph = 0,
  night_flag = 0
)

head(max_canopy)

# Computing a series of canopy height statistics from customized list expressions
canopy_metrics <- ATL03_ATL08_compute_seg_attributes_dt_segStat(atl03_atl08_dt_seg,
  list_expr = list(
    max_ph_elevation = max(h_ph),
    h_canopy = quantile(ph_h, 0.98),
    n_canopy = sum(classed_pc_flag == 2),
    n_top_canopy = sum(classed_pc_flag == 3)
  ),
  ph_class = c(2, 3),
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  quality_ph = 0,
  night_flag = 0 # there are no night photons in this dataset
)

head(canopy_metrics)

close(atl03_h5)
close(atl08_h5)
```

`ATL03_ATL08_photons_attributes_dt_clipBox`
Clip joined ATL03 and ATL08 photons by Bounding Box

Description

This function clips joined ATL03 and ATL08 photon attributes within a given Bounding Box

Usage

```
ATL03_ATL08_photons_attributes_dt_clipBox(
  atl03_atl08_dt,
  lower_left_lon,
  upper_right_lon,
  upper_right_lat,
  lower_left_lat
)
```

Arguments

`atl03_atl08_dt`	An S4 object of class `icesat2.atl03atl08_dt` containing ATL03 and ATL08 data (output of `ATL03_ATL08_photons_attributes_dt_join()` function).
`lower_left_lon`	Numeric. West longitude (x) coordinate of bounding rectangle, in decimal degrees.

`upper_right_lon`	Numeric. East longitude (x) coordinate of bounding rectangle, in decimal degrees.
`upper_right_lat`	Numeric. North latitude (y) coordinate of bounding rectangle, in decimal degrees.
`lower_left_lat`	Numeric. South latitude (y) coordinate of bounding rectangle, in decimal degrees.

Value

Returns an S4 object of class `icesat2.atl03atl08_dt` containing a subset of the ATL03 and ATL08 photon attributes.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL03 and ATL08 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Joining ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)
head(atl03_atl08_dt)

# Bounding rectangle coordinates
lower_left_lon <- -103.7604
lower_left_lat <- 59.4672
upper_right_lon <- -103.7600
upper_right_lat <- 59.4680

# Clipping ATL08-derived canopy metrics by boundary box extent
```

```
atl03_atl08_dt_clip <- ATL03_ATL08_photons_attributes_dt_clipBox(
  atl03_atl08_dt,
  lower_left_lon,
  upper_right_lon,
  upper_right_lat,
  lower_left_lat
)
head(atl03_atl08_dt_clip)

close(atl03_h5)
close(atl08_h5)
```

ATL03_ATL08_photons_attributes_dt_clipGeometry
Clip Joined ATL03 and ATL08 by Geometry

Description

This function clips joined ATL03 and ATL08 photon attributes within a given geometry.

Usage

```
ATL03_ATL08_photons_attributes_dt_clipGeometry(
  atl03_atl08_dt,
  polygon,
  split_by = NULL
)
```

Arguments

`atl03_atl08_dt`	An S4 object of class `icesat2.atl03atl08_dt` containing ATL03 and ATL08 data (output of `ATL03_ATL08_photons_attributes_dt_join()` function).
`polygon`	An object of class `terra::SpatVector`, which can be loaded as an ESRI shapefile using terra::vect function.
`split_by`	Optional. Polygon ID. If defined, the data will be clipped by each polygon using the polygon ID from the attribute table.

Value

Returns an S4 object of class `icesat2.atl03atl08_dt` containing the clipped ATL08 attributes.

Examples

```
# Specifying the path to ATL03 and ATL08 files
atl03_path <- system.file("extdata", "atl03_clip.h5", package = "ICESat2VegR")
atl08_path <- system.file("extdata", "atl08_clip.h5", package = "ICESat2VegR")

# Reading ATL03 and ATL08 data (h5 files)
```

```
atl03_h5 <- ATL03_read(atl03_path)
atl08_h5 <- ATL08_read(atl08_path)

# Joining ATL03 and ATL08 photon attributes
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)
head(atl03_atl08_dt)

# Specifying the path to the shapefile
polygon_filepath <- system.file("extdata", "clip_geom.shp", package = "ICESat2VegR")

# Reading shapefile as a SpatVector object
polygon <- terra::vect(polygon_filepath)

# Clipping ATL08 terrain attributes by geometry
atl03_atl08_dt_clip <- ATL03_ATL08_photons_attributes_dt_clipGeometry(
  atl03_atl08_dt,
  polygon,
  split_by = "id"
)
head(atl03_atl08_dt_clip)

close(atl03_h5)
close(atl08_h5)
```

ATL03_ATL08_photons_attributes_dt_gridStat

Statistics of ATL03 and ATL08 photon attributes

Description

This function computes a series of user defined descriptive statistics within each given grid cell for ATL03 and ATL08 photon attributes

Usage

```
ATL03_ATL08_photons_attributes_dt_gridStat(
  atl03_atl08_dt,
  func,
  res = 0.5,
  ph_class = c(2, 3),
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  quality_ph = 0,
  night_flag = 1
)
```

Arguments

`atl03_atl08_dt`	An S4 object of class `icesat2.atl03atl08_dt` containing ATL03 and ATL08 attributes (output of the `ATL03_ATL08_photons_attributes_dt_join()` function).

`func`	The function to be applied for computing the defined statistics
`res`	Spatial resolution in decimal degrees for the output SpatRast raster layer. Default is 0.5.
`ph_class`	Character vector indicating photons to process based on the classification (1=ground, 2=canopy, 3=top canopy), Default is c(2,3)
`beam`	Character vector indicating beams to process. Default is c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
`quality_ph`	Indicates the quality of the associated photon. 0=nominal, 1=possible_afterpulse, 2=possible_impulse_response_effect, 3=possible_tep. Default is 0
`night_flag`	Flag indicating the data were acquired in night conditions: 0=day, 1=night. Default is 1

Value

Return a SpatRast raster layer(s) of selected ATL03 and ATL08 photon attribute(s)

Examples

```
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)
# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# # Extracting ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)

# Computing the mean of ph_h attribute at 0.0002 degree grid cell
mean_ph_h <- ATL03_ATL08_photons_attributes_dt_gridStat(atl03_atl08_dt,
  func = mean(ph_h),
  res = 0.0002
)

plot(mean_ph_h)

# Define your own function
mySetOfMetrics <- function(x) {
  metrics <- list(
    min = min(x), # Min of x
    max = max(x), # Max of x
    mean = mean(x), # Mean of x
    sd = sd(x) # Sd of x
  )
  return(metrics)
}

# Computing a series of ph_h stats at 0.0002 degree grid cell from customized function
ph_h_metrics <- ATL03_ATL08_photons_attributes_dt_gridStat(atl03_atl08_dt,
  func = mySetOfMetrics(ph_h), res = 0.0002
)

plot(ph_h_metrics)

close(atl03_h5)
close(atl08_h5)
```

`ATL03_ATL08_photons_attributes_dt_join`
Join ATL03 and ATL08 photons attributes

Description

This function joins ATL03 and ATL08 computed photons attributes

Usage

```
ATL03_ATL08_photons_attributes_dt_join(
  atl03_h5,
  atl08_h5,
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
)
```

Arguments

`atl03_h5`	A ICESat-2 ATL03 object (output of `ATL03_read()` function). An S4 object of class `icesat2.atl03_dt`.
`atl08_h5`	A ICESat-2 ATL08 object (output of `ATL08_read()` function). An S4 object of class `icesat2.atl08_dt`.
`beam`	Character vector indicating beams to process (e.g. "gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")

Details

These are the photons attributes extracted by default:

- `ph_segment_id`: Georeferenced segment id (20-m) associated with each photon.

- lon_ph: Longitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point.
- lat_ph: Latitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point.
- h_ph: Height of each received photon, relative to the WGS-84 ellipsoid including the geophysical corrections noted in section 6.0. Please note that neither the geoid, ocean tide nor the dynamic atmospheric corrections (DAC) are applied to the ellipsoidal heights.
- quality_ph: Indicates the quality of the associated photon. 0=nominal, 1=possible_afterpulse, 2=possible_impulse_response_effect, 3=possible_tep. Use this flag in conjunction with signal_conf_ph to identify those photons that are likely noise or likely signal.
- solar_elevation: Elevation of the sun above the horizon at the photon bounce point.
- dist_ph_along: Along-track distance of the photon from the beginning of the segment.
- dist_ph_across: Across-track distance of the photon from the center of the segment.
- night_flag: Flag indicating the data were acquired in night conditions: 0=day, 1=night. Night flag is set when solar elevation is below 0.0 degrees.
- classed_pc_indx: Indices of photons tracking back to ATL03 that surface finding software identified and used within the creation of the data products.
- classed_pc_flag: The L2B algorithm is run if this flag is set to 1 indicating data have sufficient waveform fidelity for L2B to run.
- ph_h: Height of photon above interpolated ground surface.
- d_flag: Flag indicating whether DRAGANN labeled the photon as noise or signal.
- delta_time: Mid-segment GPS time in seconds past an epoch. The epoch is provided in the metadata at the file level.
- orbit_number: Orbit number identifier to identify data from different orbits.
- beam: Beam identifier.
- strong_beam: Logical indicating if the beam is a strong beam.

Value

Returns an S4 object of class icesat2.atl03atl08_dt containing the ATL08 computed photons attributes.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)
```

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# # Extracting ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)
head(atl03_atl08_dt)

close(atl03_h5)
close(atl08_h5)
```

ATL03_ATL08_photons_attributes_dt_LAS

Converts ATL03/ATL08 classified photon cloud to LAS

Description

Converts ATL03/ATL08 classified photon cloud to LAS

Usage

```
ATL03_ATL08_photons_attributes_dt_LAS(
  atl03_atl08_dt,
  output,
  normalized = TRUE
)
```

Arguments

atl03_atl08_dt	An S4 object of class icesat2.atl08_dt containing ATL03 and ATL08 data (output of ATL03_ATL08_photons_attributes_dt_join() function).
output	character. The output path of for the LAS(Z) file(s). The function will create one LAS file per UTM Zone in WGS84 datum.
normalized	logical, default TRUE. Whether the output should be normalized LAS or raw altitude.

Value

Nothing, it just saves outputs as LAS file in disk

Examples

```
outdir <- tempdir()

# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)

if (require("lidR")) {
  ATL03_ATL08_photons_attributes_dt_LAS(
    atl03_atl08_dt,
    output = file.path(outdir, "output.laz"),
    normalized = TRUE
  )
}

close(atl03_h5)
close(atl08_h5)
```

`ATL03_ATL08_photons_attributes_dt_polyStat` *Statistics of ATL03 and ATL08 joined photons attributes within a given area*

Description

Computes a series of statistics ATL03 and ATL08 joined photons attributes within area defined by a polygon

Usage

```
ATL03_ATL08_photons_attributes_dt_polyStat(
  atl03_atl08_dt,
  func,
  poly_id = NULL
)
```

Arguments

`atl03_atl08_dt`	An S4 object of class `icesat2.atl08_dt` containing ATL03 and ATL08 data (output of `ATL03_ATL08_photons_attributes_dt_join()` function).
`func`	The function to be applied for computing the defined statistics
`poly_id`	Polygon id. If defined, statistics will be computed for each polygon

Value

Returns an S4 object of class `icesat2.atl08_dt` Containing Statistics of ATL08 classified canopy photons

Examples

```
# Specifying the path to ATL03 and ATL08 files
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)
head(atl03_atl08_dt)

# Specifying the path to shapefile
polygon_filepath <- system.file("extdata", "clip_geom.shp", package = "ICESat2VegR")

# Reading shapefile as sf object
polygon <- terra::vect(polygon_filepath)

# Clipping ATL08 terrain attributes by Geometry
atl03_atl08_dt_clip <- ATL03_ATL08_photons_attributes_dt_clipGeometry(atl03_atl08_dt,
  polygon, split_by = "id")
```

```
# Computing the maximum ph_h by polygon id
max_ph_h <- ATL03_ATL08_photons_attributes_dt_polyStat(atl03_atl08_dt_clip,
  func = max(ph_h), poly_id = "poly_id")
head(max_ph_h)

# Define your own function
mySetOfMetrics <- function(x) {
  metrics <- list(
    min = min(x), # Min of x
    max = max(x), # Max of x
    mean = mean(x), # Mean of x
    sd = sd(x) # Sd of x
  )
  return(metrics)
}

# Computing a series of ph_h statistics from customized function
ph_h_metrics <- ATL03_ATL08_photons_attributes_dt_polyStat(
  atl03_atl08_dt_clip,
  func = mySetOfMetrics(ph_h),
  poly_id = "poly_id"
)

head(ph_h_metrics)

close(atl03_h5)
close(atl08_h5)
```

ATL03_ATL08_photons_dt_height_normalize

Fit and estimate ground elevation for photons or arbitrary distances from the track beginning.

Description

Function to estimate ground elevation using smoothing and interpolation functions

Usage

```
ATL03_ATL08_photons_dt_height_normalize(
  atl03_atl08_seg_dt,
  smoothing_window = NA,
  smoothing_func = median,
  interpolation_func = NA,
  xout_parameter_name = "xout",
  ...
)
```

Arguments

`atl03_atl08_seg_dt`
: An S4 object of class `icesat2.atl03_atl08_seg_dt` containing ATL03 and ATL08 data (output of `ATL03_ATL08_photons_attributes_dt_join()` function).

`smoothing_window`
: numeric. The smoothing window size in meters for smoothing the photon cloud. Default is NA, see details for more information.

`smoothing_func`
: function. The smoothing function to be applied on the smoothing window.

`interpolation_func`
: function. The interpolation function to estimate the ground elevation. Default `stats::approx()`.

`xout_parameter_name`
: character. The parameter name used for the `interpolation_func` to which will be used to predict, default "xout".

`...`
: parameters to be passed forward to `ATL03_ATL08_photons_seg_dt_fitground()`.

Details

The function for calculating the ground will first pass a smoothing window with `smoothing_window` size, applying the `smoothing_func` to aggregate the ground photons.

Then it will use an interpolation function between those aggregated photons to calculate a smooth surface.

The `smoothing_func` signature will depend on the function used. It is assumed that the first two arguments are vectors of `x` (independent variable) and `y` (the prediction to be interpolated). The remaining arguments are passed through `...`.

The interpolation functions need a third parameter which is the `x` vector to be interpolated. Functions from `stats` base package `stats::approx()` and `stats::spline()` name this argument as `xout`, so you can use:

```
ATL03_ATL08_photons_fitground_seg_dt(
  dt,
  interpolation_func = approx,
  xout = 1:30
)
```

For example, to interpolate the values for the 1:30 vector. But other functions may name the parameter differently, such as `signal::pchip()`, which name the parameter as `xi` instead of `xout`. `signal::pchip()` is the algorithm used by ATL08 ATBD.

The `smoothing_window` can be left NA, which will use the ATBD algoritm for calculating the window size:

$Sspan = ceil[5 + 46 * (1 - e^{-a*length})]$, where *length* is the number of photons within segment.

$$a \approx 21x10^{-6}$$

$$window_size = \frac{2}{3}Sspan$$

This is not the same algorithm as used in ATL08 but is an adapted version that uses the ATL08 pre-classification

---

ATL03_ATL08_photons_seg_dt_fitground

Fit and estimate ground elevation for photons or arbitrary distances from the track beginning.

Description

Function to estimate ground elevation using smoothing and interpolation functions

Usage

```
ATL03_ATL08_photons_seg_dt_fitground(
  atl03_atl08_seg_dt,
  smoothing_window = NA,
  smoothing_func = median,
  interpolation_func = NA,
  xout_parameter_name = "xout",
  ...
)
```

Arguments

`atl03_atl08_seg_dt`	An S4 object of class `icesat2.atl03_atl08_seg_dt` containing ATL03 and ATL08 data (output of `ATL03_ATL08_photons_attributes_dt_join()` function).
`smoothing_window`	numeric. The smoothing window size in meters for smoothing the photon cloud. Default is NA, see details for more information.
`smoothing_func`	function. The smoothing function to be applied on the smoothing window.
`interpolation_func`	function. The interpolation function to estimate the ground elevation.
`xout_parameter_name`	character. Optional, can be used to inform the parameter name that the interpolation_func uses for passing the prediction vector and already use the photons for prediction. Default NA will use the ...
`...`	Optional parameters to pass to the interpolation_func, see details for more information.

Details

The function for calculating the ground will first pass a smoothing window with `smoothing_window` size, applying the `smoothing_func` to aggregate the ground photons.

Then it will use an interpolation function between those aggregated photons to calculate a smooth surface.

The `smoothing_func` signature will depend on the function used. It is assumed that the first two arguments are vectors of `x` (independent variable) and `y` (the prediction to be interpolated). The remaining arguments are passed through `...`.

The interpolation functions need a third parameter which is the `x` vector to be interpolated. Functions from `stats` base package `stats::approx()` and `stats::spline()` name this argument as `xout`, so you can use:

```
ATL03_ATL08_photons_fitground_seg_dt(
  dt,
  interpolation_func = approx,
  xout = 1:30
)
```

For example, to interpolate the values for the 1:30 vector. But other functions may name the parameter differently, such as `signal::pchip()`, which name the parameter as `xi` instead of `xout`. `signal::pchip()` is the algorithm used by ATL08 ATBD.

The `smoothing_window` can be left NA, which will use the ATBD algoritm for calculating the window size:

$Sspan = ceil[5 + 46 * (1 - e^{-a*length})]$, where *length* is the number of photons within segment.

$$a \approx 21x10^{-6}$$

$$window_size = \frac{2}{3}Sspan$$

This is not the same algorithm as used in ATL08 but is an adapted version that uses the ATL08 pre-classification

---

`ATL03_ATL08_segment_create`
Compute segments id for a given segment length

Description

This function reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) as h5 file.

Usage

```
ATL03_ATL08_segment_create(
  atl03_atl08_dt,
  segment_length,
  centroid = "mean",
  output = NA,
  overwrite = FALSE
)
```

Arguments

`atl03_atl08_dt`	`icesat2.atl03atl08_dt`. The output of the `ATL03_ATL08_photons_attributes_dt_join()`.
`segment_length`	`numeric`. The desired segment length to split the photons.
`centroid`	character. Method used to calculate the segment centroid, either "mean" or "mid-point", see details. Default 'mean'.
`output`	Character vector. The output vector file. The GDAL vector format will be inferred by the file extension using `terra::writeVector()`
`overwrite`	logical input to control if the output vector file should be overwritten. Default FALSE.

Details

The centroid will be computed using either the photons centroid or the approximate segment centroid.

- "mean": calculated using the average coordinates from all photons within the segment. This approach will better represent the mean statistics location.
- "mid-point": the minimum and maximum coordinates will be averaged to calculate a midpoint within the segment. This will give a better representation of the segment true mid-point

Value

Returns an S4 object of class `icesat2.atl03atl08_dt` containing ICESat-2 ATL08 data.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# Specifying the path to ICESat-2 ATL03 and ATL08 data
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)

atl03_atl08_dt_seg <- ATL03_ATL08_segment_create(atl03_atl08_dt,
  segment_length = 30,
  centroid = "mean",
  output = NA,
  overwrite = FALSE
)

head(atl03_atl08_dt_seg)

close(atl03_h5)
close(atl08_h5)
```

ATL03_ATL08_seg_cover_dt_compute

Compute segments id for a given segment length

Description

This function reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) as h5 file.

Usage

```
ATL03_ATL08_seg_cover_dt_compute(atl03_atl08_dt, reflectance_ratio = 1)
```

Arguments

`atl03_atl08_dt`	`icesat2.atl03atl08_dt`. The output of the `ATL03_ATL08_photons_attributes_dt_join()`.
`reflectance_ratio`	Numeric. The reflectance ratio to use to compute the coverage metric. ρ_v/ρ_g, where ρ_v is the vegetation reflectance and ρ_v is the ground reflectance. Default is 1 (same reflectance).

Details

Coverage is calculated with the formula

$$\frac{1}{1+\frac{\rho_v N_g}{\rho_g N_v}}$$

Value

Returns an S4 object of class `data.table::data.table` containing ICESat-2 ATL08 data.

See Also

`https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf`

Examples

```
# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)

close(atl08)
```

`ATL03_h5_clipBox` *Clips ICESat-2 ATL03 H5 data*

Description

This function clips ATL03 HDF5 file within beam groups, but keeps metada and ancillary data the same.

Usage

```
ATL03_h5_clipBox(
  atl03,
  output,
  bbox,
  beam = c("gt1r", "gt2r", "gt3r", "gt1l", "gt2l", "gt3l"),
  additional_groups = c("orbit_info")
)
```

Arguments

`atl03`	`icesat2.atl03_h5` object, obtained through `ATL03_read()` for clipping
`output`	character. Path to the output h5 file.
`bbox`	`numeric` or `terra::SpatExtent` for clipping, the order of the bbox is the default from NASA's ICESat-2 CMS searching: (ul_lat, ul_lon, lr_lat, lr_lon).
`beam`	`character`. The vector of beams to include, default all c("gt1l", "gt2l", "gt3l", "gt1r", "gt2r", "gt3r")
`additional_groups`	`character`. Other addional groups that should be included, default c("orbit_info")

Value

Returns the clipped S4 object of class `icesat2.atl03_h5`

Examples

```
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Bounding rectangle coordinates
xmin <- -106.5723
xmax <- -106.5693
ymin <- 41.533
ymax <- 41.537

# Clipping ATL03 photons by boundary box extent
output <- tempfile(fileext = ".h5")
atl03_photons_dt_clip <- ATL03_h5_clipBox(atl03_h5, output, c(ymax, xmin, ymin, xmax))

close(atl03_h5)
```

`ATL03_h5_clipGeometry` *Clips ICESat-2 ATL03 H5 data*

Description

This function clips ATL03 HDF5 file within beam groups, but keeps metada and ancillary data the same.

Usage

```
ATL03_h5_clipGeometry(
  atl03,
  output,
  vect,
  polygon_id = NULL,
  beam = c("gt1r", "gt2r", "gt3r", "gt1l", "gt2l", "gt3l"),
  additional_groups = c("orbit_info")
)
```

Arguments

`atl03`	`icesat2.atl03_h5` object, obtained through `ATL03_read()` for clipping
`output`	character. Path to the output h5 file, the attribute for polygons will be appended to the file name.
`vect`	`terra::SpatVector` for clipping
`polygon_id`	`character`. The attribute name used for identifying the different polygons. Default is "id"
`beam`	`character`. The vector of beams to include, default all c("gt1l", "gt2l", "gt3l", "gt1r", "gt2r", "gt3r")
`additional_groups`	`character`. Other addional groups that should be included, default c("orbit_info")

Value

Returns a list of clipped S4 object of class `icesat2.atl03_h5`

Examples

```
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

output <- tempfile(fileext = ".h5")

vect_path <- system.file("extdata",
  "polygons.shp",
  package = "ICESat2VegR"
)

vect <- terra::vect()

# Clipping ATL03 photons  by boundary box extent
atl03_photons_dt_clip <- ATL03_h5_clipGeometry(
  atl03_h5,
  output,
  vect,
  polygon_id = "id"
)

close(atl03_h5)
```

`ATL03_photons_attributes_dt`

ATL03 photons attributes

Description

This function extracts photons attributes from ICESat-2 ATL03 data

Usage

```
ATL03_photons_attributes_dt(
  atl03_h5,
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
)
```

Arguments

`atl03_h5`	A ICESat-2 ATL03 object (output of `ATL03_read()` function). An S4 object of class `icesat2.atl03_dt`.
`beam`	Character vector indicating beams to process (e.g. "gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")

Details

These are the photons attributes extracted:

- *lon_ph* Longitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point.
- *lat_ph* Latitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point.
- *lat_ph* Latitude of each received photon. Computed from the ECEF Cartesian coordinates of the bounce point. Height of each received photon, relative to the WGS-84 ellipsoid including the geophysical corrections noted in section 6.0. Please note that neither the geoid, ocean tide nor the dynamic atmospheric corrections (DAC) are applied to the ellipsoidal heights.
- *quality_ph* Indicates the quality of the associated photon. 0=nominal, 1=possible_afterpulse, 2=possible_impulse_response_ effect, 3=possible_tep. Use this flag in conjunction with signal_conf_ph to identify those photons that are likely noise or likely signal
- *night_flag* Flag indicating the data were acquired in night conditions: 0=day, 1=night. Night flag is set when solar elevation is below 0.0 degrees.

Value

Returns an S4 object of class `data.table::data.table` containing the ATL03 photons attributes.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

Examples

```
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting ATL03 photons attributes
atl03_photons_dt <- ATL03_photons_attributes_dt(atl03_h5 = atl03_h5)

head(atl03_photons_dt)
close(atl03_h5)
```

`ATL03_photons_attributes_dt_clipBox`
Clip ATL03 photons by Coordinates

Description

This function clips ATL03 photons attributes within a given bounding coordinates

Usage

```
ATL03_photons_attributes_dt_clipBox(
  atl03_photons_dt,
  lower_left_lon,
  upper_right_lon,
  lower_left_lat,
  upper_right_lat
)
```

Arguments

`atl03_photons_dt`	A atl03_photons_dt object (output of `ATL03_photons_attributes_dt()` function). An S4 object of class `icesat2.atl03_dt`
`lower_left_lon`	Numeric. West longitude (x) coordinate of bounding rectangle, in decimal degrees.

`upper_right_lon`	Numeric. East longitude (x) coordinate of bounding rectangle, in decimal degrees.
`lower_left_lat`	Numeric. South latitude (y) coordinate of bounding rectangle, in decimal degrees.
`upper_right_lat`	Numeric. North latitude (y) coordinate of bounding rectangle, in decimal degrees.

Value

Returns an S4 object of class `icesat2.atl03_dt` containing the ATL03 photons attributes.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting ATL03 photons attributes
atl03_photons_dt <- ATL03_photons_attributes_dt(atl03_h5 = atl03_h5)

# Bounding rectangle coordinates
lower_left_lon <- -106.57
lower_left_lat <- 41.53
upper_right_lon <- -106.5698
upper_right_lat <- 41.54

# Clipping ATL08-derived canopy metrics by boundary box extent
atl03_photons_dt_clip <- ATL03_photons_attributes_dt_clipBox(
  atl03_photons_dt,
  lower_left_lon,
  upper_right_lon,
  lower_left_lat,
  upper_right_lat
)

head(atl03_photons_dt_clip)

close(atl03_h5)
```

ATL03_photons_attributes_dt_clipGeometry
Clip ATL03 photons by Coordinates

Description

This function clips ATL03 photon attributes within given bounding coordinates.

Usage

```
ATL03_photons_attributes_dt_clipGeometry(
  atl03_photons_dt,
  sppoly,
  split_by = "id"
)
```

Arguments

`atl03_photons_dt`	An ATL03 photon data table. An S4 object of class `icesat2.atl03_dt`.
`sppoly`	Spatial Polygon. An object of class `terra::SpatVector`, which can be loaded as an ESRI shapefile using the terra::vect function in the *sf* package.
`split_by`	Polygon id. If defined, GEDI data will be clipped by each polygon using the polygon id from the attribute table defined by the user.

Value

Returns an S4 object of class `icesat2.atl03_dt` containing the ATL03 photon attributes.

See Also

`https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf`

Examples

```
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting ATL03 photon attributes
atl03_photons_dt <- ATL03_photons_attributes_dt(atl03_h5 = atl03_h5)
```

```
# Specifying the path to shapefile
polygon_filepath <-
  system.file(
    "extdata",
    "clip_geom.shp",
    package = "ICESat2VegR"
  )

# Reading shapefile as sf object
sppoly <- terra::vect(polygon_filepath)

# Clipping ATL03 photon attributes by Geometry
atl03_photons_dt_clip <-
  ATL03_photons_attributes_dt_clipGeometry(atl03_photons_dt, sppoly, split_by = "id")

head(atl03_photons_dt_clip)

close(atl03_h5)
```

ATL03_photons_attributes_dt_LAS

Converts ATL03 photon cloud to LAS

Description

Converts ATL03 photon cloud to LAS

Usage

```
ATL03_photons_attributes_dt_LAS(atl03_dt, output)
```

Arguments

`atl03_dt`	An S4 object of class `icesat2.atl03_dt` (output of `ATL03_photons_attributes_dt()` function).
`output`	character. The output path of for the LAS(Z) file(s) The function will create one LAS file per UTM Zone in WGS84 datum.

Details

As the las format expects a metric coordinate reference system (CRS) we use helper functions to define UTM zones to which the original ICESat-2 data will be converted.

The function credits go to Chuck Gantz- chuck.gantz@globalstar.com.

Value

Nothing, it just saves outputs as LAS file in disk

See Also

https://oceancolor.gsfc.nasa.gov/docs/ocssw/LatLong-UTMconversion_8cpp_source.html

Examples

```
# ATL03 file path
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting ATL03 and ATL08 photons and heights
atl03_dt <- ATL03_photons_attributes_dt(atl03_h5, beam = "gt1r")

outdir <- tempdir()
ATL03_photons_attributes_dt_LAS(
  atl03_dt,
  file.path(outdir, "output.laz")
)

close(atl03_h5)
```

`ATL03_photons_plot` *Read ICESat-2 ATL08 data*

Description

This function reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) as h5 file.

Usage

```
ATL03_photons_plot(atl08_path)
```

Arguments

`atl08_path`	File path pointing to ICESat-2 ATL08 data. Data in HDF5 Hierarchical Data Format (.h5).

Value

Returns an S4 object of class `icesat2.atl08_dt` containing ICESat-2 ATL08 data.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)

close(atl08)
```

`ATL03_read` *Read ICESat-2 ATL03 data*

Description

This function reads the ICESat-2 Global Geolocated Photons (ATL03) Product (ATL03) as h5 file.

Usage

```
ATL03_read(atl03_path)

## S4 method for signature 'ANY'
ATL03_read(atl03_path)

## S4 method for signature 'icesat2.granule_cloud'
ATL03_read(atl03_path)
```

Arguments

`atl03_path`	An object of class `icesat2.granule_cloud` pointing to the ICESat-2 ATL03 data in the cloud.

Value

Returns an S4 object of class `icesat2.atl03_dt` containing ICESat-2 ATL03 data.

An S4 object of class `icesat2.atl03_h5` containing ICESat-2 ATL03 data.

An S4 object of class `icesat2.atl03_h5` containing ICESat-2 ATL03 data.

Functions

- `ATL03_read(ANY)`: Method for reading ICESat-2 ATL03 data from a local h5 file.
- `ATL03_read(icesat2.granule_cloud)`: Method for reading ICESat-2 ATL03 data from a cloud granule.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL03 data (h5 file)
ATL03 <- ATL03_read(atl03_path = atl03_path)
close(ATL03)
```

ATL03_seg_attributes_dt

ATL03 segments attributes

Description

This function extracts segment attributes from ICESat-2 ATL03 data

Usage

```
ATL03_seg_attributes_dt(
  atl03_h5,
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  attributes = c("h_ph", "altitude_sc", "bounce_time_offset", "delta_time",
   "full_sat_fract", "near_sat_fract", "neutat_delay_derivative", "neutat_delay_total",
     "neutat_ht", "ph_index_beg", "pitch", "podppd_flag", "range_bias_corr",
   "ref_azimuth", "ref_elev", "reference_photon_index", "roll", "segment_dist_x",
   "segment_id", "segment_length", "segment_ph_cnt", "sigma_across", "sigma_along",
   "sigma_h", "sigma_lat", "sigma_lon", "solar_azimuth", "solar_elevation", "surf_type",
     "tx_pulse_energy", "tx_pulse_skew_est",
      "tx_pulse_width_lower",
     "tx_pulse_width_upper", "yaw")
)
```

Arguments

`atl03_h5`	A ICESat-2 ATL03 object (output of `ATL03_read()` function). An S4 object of class `icesat2.atl03_dt`.
`beam`	Character vector indicating beams to process (e.g. "gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
`attributes`	Character vector indicating the attrivutes

Details

These are the available variables for extraction:

- `h_ph`: Height of the reference photon above the WGS84 ellipsoid.
- `altitude_sc`: Height of the spacecraft above the WGS84 ellipsoid.
- `bounce_time_offset`: The difference between the transmit time and the ground bounce time of the reference photons.
- `delta_time`: Transmit time of the reference photon, measured in seconds from the atlas_sdp_gps_epoch.
- `full_sat_fract`: The fraction of pulses within the segment determined to be fully saturated.
- `near_sat_fract`: The fraction of pulses within the segment determined to be nearly saturated.
- `neutat_delay_derivative`: Change in neutral atmospheric delay per height change.
- `neutat_delay_total`: Total neutral atmosphere delay correction (wet+dry).
- `neutat_ht`: Reference height of the neutral atmosphere range correction.
- `ph_index_beg`: Index (1-based) within the photon-rate data of the first photon within this segment.
- `pitch`: Spacecraft pitch, computed using a 3, 2, 1 Euler angle sequence, with units in degrees.
- `podppd_flag`: A composite flag indicating the quality of input geolocation products for the specific ATL03 segment.
- `range_bias_corr`: The estimated range bias from geolocation analysis.
- `ref_azimuth`: Azimuth of the unit pointing vector for the reference photon in the local ENU frame, in radians.
- `ref_elev`: Elevation of the unit pointing vector for the reference photon in the local ENU frame, in radians.
- `reference_photon_index`: Index of the reference photon within the set of photons grouped within a segment.
- `reference_photon_lat`: Latitude of each reference photon.
- `reference_photon_lon`: Longitude of each reference photon.
- `roll`: Spacecraft roll, computed using a 3, 2, 1 Euler angle sequence, with units in degrees.
- `segment_dist_x`: Along-track distance from the equator crossing to the start of the 20 meter geolocation segment.
- `segment_id`: A 7-digit number identifying the along-track geolocation segment number.
- `segment_length`: The along-track length of the along-track segment, typically 20 meters.
- `segment_ph_cnt`: Number of photons in a given along-track segment.
- `sigma_across`: Estimated Cartesian across-track uncertainty (1-sigma) for the reference photon.
- `sigma_along`: Estimated cartesian along-track uncertainty (1-sigma) for the reference photon.
- `sigma_h`: Estimated height uncertainty (1-sigma) for the reference photon bounce point.
- `sigma_lat`: Estimated geodetic Latitude uncertainty (1-sigma) for the reference photon bounce point.

- `sigma_lon`: Estimated geodetic Longitude uncertainty (1-sigma) for the reference photon bounce point.
- `solar_azimuth`: Azimuth of the sun position vector from the reference photon bounce point position in the local ENU frame, in degrees east.
- `solar_elevation`: Elevation of the sun position vector from the reference photon bounce point position in the local ENU frame, in degrees.
- `surf_type`: Flags describing which surface types an interval is associated with (e.g., land, ocean, sea ice, land ice, inland water).
- `tx_pulse_energy`: Average transmit pulse energy, measured by the internal laser energy monitor, split into per-beam measurements.
- `tx_pulse_skew_est`: Difference between the averages of the lower and upper threshold crossing times, estimating the transmit pulse skew.
- `tx_pulse_width_lower`: Average distance between the lower threshold crossing times measured by the Start Pulse Detector.
- `tx_pulse_width_upper`: Average distance between the upper threshold crossing times measured by the Start Pulse Detector.
- `velocity_sc`: Spacecraft velocity components (east component, north component, up component) an observer on the ground would measure.
- `yaw`: Spacecraft yaw, computed using a 3, 2, 1 Euler angle sequence, with units in degrees.

Value

Returns an S4 object of class data.table::data.table containing the ATL03 segment attributes.

See Also

`https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf`

Examples

```
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting ATL03 segment attributes
atl03_segment_dt <- ATL03_seg_attributes_dt(atl03_h5 = atl03_h5)

head(atl03_segment_dt)
close(atl03_h5)
```

ATL08_attributes_dt_LAS

Read ICESat-2 ATL08 data

Description

This function reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) as h5 file.

Usage

```
ATL08_attributes_dt_LAS(atl08_path)
```

Arguments

atl08_path	File path pointing to ICESat-2 ATL08 data. Data in HDF5 Hierarchical Data Format (.h5).

Value

Returns an S4 object of class icesat2.atl08_dt containing ICESat-2 ATL08 data.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)

close(atl08)
```

`ATL08_h5_clipBox` *Clips ICESat-2 ATL08 data*

Description

This function clips the ATl08 HDF5 file. This function will only clip the beam groups within hdf5, it won't change metadata or ancillary data.

Usage

```
ATL08_h5_clipBox(
  atl08,
  output,
  bbox,
  beam = c("gt1r", "gt2r", "gt3r", "gt1l", "gt2l", "gt3l"),
  additional_groups = c("orbit_info")
)
```

Arguments

`atl08`	`icesat2.atl08_h5` object, obtained through `ATL08_read()` for clipping
`output`	character. Path to the output h5 file.
`bbox`	`numeric` or `terra::SpatExtent` for clipping, the order of the bbox is the default from NASA's ICESat-2 CMS searching: (ul_lat, ul_lon, lr_lat, lr_lon).
`beam`	`character`. The vector of beams to include, default all c("gt1l", "gt2l", "gt3l", "gt1r", "gt2r", "gt3r")
`additional_groups`	`character`. Other addional groups that should be included, default c("orbit_info")

Value

Returns the clipped S4 object of class `icesat2.atl08_h5`

Examples

```
# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Bounding rectangle coordinates
ul_lon <- -106.5723
lr_lon <- -106.5693
```

```
lr_lat <- 41.533
ul_lat <- 41.537

# Clipping ATL08 terrain and canopy attributes by boundary box
atl08_seg_att_dt_clip <- ATL08_h5_clipBox(
  atl08_h5,
  output = tempfile(fileext = ".h5"),
  c(ul_lat, ul_lon, lr_lat, lr_lon)
)

close(atl08_h5)
close(atl08_seg_att_dt_clip)
```

ATL08_h5_clipGeometry *Clips ICESat-2 ATL08 data*

Description

This function clips ATL08 HDF5 file within beam groups, but keeps metada and ancillary data the same.

Usage

```
ATL08_h5_clipGeometry(
  atl08,
  output,
  vect,
  polygon_id = "id",
  beam = c("gt1r", "gt2r", "gt3r", "gt1l", "gt2l", "gt3l"),
  additional_groups = c("orbit_info")
)
```

Arguments

atl08	icesat2.atl08_h5 object, obtained through ATL08_read() for clipping
output	character. Path to the output h5 file.
vect	terra::SpatVector for clipping
polygon_id	character. The attribute name used for identifying the different polygons. Default is "id"
beam	character. The vector of beams to include, default all c("gt1l", "gt2l", "gt3l", "gt1r", "gt2r", "gt3r")
additional_groups	character. Other addional groups that should be included, default c("orbit_info")

Value

Returns a list of clipped S4 object of class icesat2.atl08_h5

Examples

```
# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

output <- tempfile(fileext = ".h5")

vect_path <- system.file("extdata",
  "clip_geom.shp",
  package = "ICESat2VegR"
)

vect <- terra::vect(vect_path)

# Clipping ATL08 photons by boundary box extent
atl08_photons_dt_clip <- ATL08_h5_clipGeometry(
  atl08_h5,
  output,
  vect,
  polygon_id = "id"
)

close(atl08_h5)
```

`ATL08_photons_attributes_dt` *ATL08 computed photons attributes*

Description

This function extracts computed photons attributes from ICESat-2 ATL08 data

Usage

```
ATL08_photons_attributes_dt(
  atl08_h5,
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
 photon_attributes = c("ph_segment_id", "classed_pc_indx", "classed_pc_flag", "ph_h",
    "d_flag", "delta_time")
)
```

Arguments

`atl08_h5`	A ICESat-2 ATL08 object (output of `ATL08_read()` function). An S4 object of class `icesat2.atl08_dt`.
`beam`	Character vector indicating beams to process (e.g. "gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
`photon_attributes`	character vector indicating the attributes to extract from the ATL08 photons. Default all `c("ph_segment_id", "classed_pc_indx", "classed_pc_flag", "ph_h", "d_flag", "delta_time")`.

Details

These are the photons attributes extracted by default:

- **ph_segment_id**: Georeferenced bin number (20-m) associated with each photon
- **classed_pc_indx**: Indices of photons tracking back to ATL03 that surface finding software identified and used within the creation of the data products.
- **classed_pc_flag**: The L2B algorithm is run if this flag is set to 1 indicating data have sufficient waveform fidelity for L2B to run
- **ph_h**: Height of photon above interpolated ground surface
- **d_flag**: Flag indicating whether DRAGANN labeled the photon as noise or signal
- **delta_time**: Mid-segment GPS time in seconds past an epoch. The epoch is provided in the metadata at the file level

Value

Returns an S4 object of class data.table::data.table containing the ATL08 computed photons attributes.

See Also

`https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf`

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path)

# Extracting ATL08 classified photons and heights
atl08_photons <- ATL08_photons_attributes_dt(atl08_h5 = atl08_h5)
```

```
head(atl08_photons)
close(atl08_h5)
```

ATL08_read *Read ICESat-2 ATL08 data*

Description

This function reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) as h5 file.

Usage

```
ATL08_read(atl08_path)
```

Arguments

`atl08_path`	File path pointing to ICESat-2 ATL08 data. Data in HDF5 Hierarchical Data Format (.h5).

Value

Returns an S4 object of class `icesat2.atl08_h5` containing ICESat-2 ATL08 data.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)
close(atl08)
```

ATL08_read,character-method
Read ICESat-2 ATL08 data from local HDF5 file

Description

This method reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) from a local HDF5 file.

Usage

```
## S4 method for signature 'character'
ATL08_read(atl08_path)
```

Arguments

`atl08_path`	Character. File path pointing to ICESat-2 ATL08 data in HDF5 format (.h5).

Value

Returns an S4 object of class `icesat2.atl08_h5` containing ICESat-2 ATL08 data.

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)
close(atl08)
```

ATL08_read,icesat2.granules_cloud-method
Read ICESat-2 ATL08 data from multiple granules

Description

This method stops the process when multiple granules are provided.

Usage

```
## S4 method for signature 'icesat2.granules_cloud'
ATL08_read(atl08_path)
```

Arguments

`atl08_path`	Object of class `icesat2.granules_cloud`. This parameter represents multiple granules of ICESat-2 ATL08 data.

Value

This method stops execution with an error message indicating that the package works with only one granule at a time.

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)
close(atl08)
```

`ATL08_read,icesat2.granule_cloud-method`
Read ICESat-2 ATL08 data from a single granule in the cloud

Description

This method reads the ICESat-2 Land and Vegetation Along-Track Products (ATL08) from a single granule in the cloud.

Usage

```
## S4 method for signature 'icesat2.granule_cloud'
ATL08_read(atl08_path)
```

Arguments

`atl08_path`	Object of class `icesat2.granule_cloud`. This parameter represents a single granule of ICESat-2 ATL08 data.

Value

Returns an S4 object of class `icesat2.atl08_h5` containing ICESat-2 ATL08 data.

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ICESat-2 ATL08 data (h5 file)
atl08 <- ATL08_read(atl08_path = atl08_path)
close(atl08)
```

ATL08_seg_attributes_dt

ATL08 Terrain and Canopy Attributes

Description

This function extracts terrain and canopy attributes by segments from ICESat-2 ATL08 data

Usage

```
ATL08_seg_attributes_dt(
  atl08_h5,
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  attributes = c("delta_time", "h_canopy", "canopy_openness", "h_te_mean",
    "terrain_slope")
)
```

Arguments

`atl08_h5`	A ICESat-2 ATL08 object (output of `ATL08_read()` function). An S4 object of class `icesat2.atl08_dt`.
`beam`	Character vector indicating beams to process (e.g. "gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
`attributes`	A character vector containing the list of terrain and canopy attributes to be extracted. Default is attribute = c("h_canopy","canopy_h_metrics","canopy_openness","h_te_mean","h_te_

Details

ATL08 canopy attributes:

- *"h_canopy"*
- *"canopy_rh_conf"*
- *"h_median_canopy_abs"*
- *"h_min_canopy"*
- *"h_mean_canopy_abs"*

- *"h_median_canopy"*
- *"h_canopy_abs"*
- *"toc_roughness"*
- *"h_min_canopy_abs"*
- *"h_dif_canopy"*
- *"h_canopy_quad"*
- *"h_canopy_20m"*
- *"n_ca_photons"*
- *"photon_rate_can"*
- *"centroid_height"*
- *"canopy_h_metrics_abs"*
- *"h_mean_canopy"*
- *"subset_can_flag"*
- *"canopy_h_metrics"*
- *"n_toc_photons"*
- *"h_max_canopy_abs"*
- *"h_canopy_uncertainty"*
- *"canopy_openness"*
- *"h_max_canopy"*
- *"segment_cover"*

ATL08 terrain attributes:

- *"h_te_best_fit"*
- *"h_te_best_fit_20m"*
- *"h_te_interp"*
- *"h_te_max"*
- *"h_te_mean"*
- *"h_te_median"*
- *"h_te_mode"*
- *"h_te_rh25"*
- *"h_te_skew"*
- *"h_te_std"*
- *"h_te_uncertainty"*
- *"n_te_photons"*
- *"photon_rate_te"*
- *"subset_te_flag"*
- *"terrain_slope"*

Value

Returns an S4 object of class icesat2.atl08_dt containing the ATL08-derived terrain and canopy attributes by segments.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL08-derived terrain and canopy attributes
atl08_seg_att_dt <- ATL08_seg_attributes_dt(atl08_h5 = atl08_h5)
head(atl08_seg_att_dt)

close(atl08_h5)
```

`ATL08_seg_attributes_dt_clipBox`

Clip ATL08 Terrain and Canopy Attributes by Bounding Box

Description

This function clips ATL08 Terrain and Canopy Attributes within a given bounding box coordinates

Usage

```
ATL08_seg_attributes_dt_clipBox(
  atl08_seg_att_dt,
  lower_left_lon,
  upper_right_lon,
  lower_left_lat,
  upper_right_lat
)
```

Arguments

`atl08_seg_att_dt`	A atl08_seg_att_dt object (output of `ATL08_seg_attributes_dt()` function). An S4 object of class `icesat2.atl08_dt`
`lower_left_lon`	Numeric. West longitude (x) coordinate of bounding rectangle, in decimal degrees.
`upper_right_lon`	Numeric. East longitude (x) coordinate of bounding rectangle, in decimal degrees.
`lower_left_lat`	Numeric. South latitude (y) coordinate of bounding rectangle, in decimal degrees.
`upper_right_lat`	Numeric. North latitude (y) coordinate of bounding rectangle, in decimal degrees.

Value

Returns an S4 object of class `icesat2.atl08_dt` containing the clipped ATL08 terrain and canopy attributes.

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path)

# Extracting ATL08-derived Canopy Metrics
atl08_seg_att_dt <- ATL08_seg_attributes_dt(atl08_h5 = atl08_h5)

# Bounding rectangle coordinates
lower_left_lon <- -103.7604
lower_left_lat <- 59.4672
upper_right_lon <- -103.7600
upper_right_lat <- 59.4680

# Clipping ATL08 terrain and canopy attributes by boundary box
atl08_seg_att_dt_clip <- ATL08_seg_attributes_dt_clipBox(
  atl08_seg_att_dt,
  lower_left_lon,
  upper_right_lon,
  lower_left_lat,
  upper_right_lat
)

close(atl08_h5)
```

ATL08_seg_attributes_dt_clipGeometry
Clip ATL08 Terrain and Canopy Attributes by Geometry

Description

This function clips ATL08 Terrain and Canopy Attributes within a given geometry

Usage

```
ATL08_seg_attributes_dt_clipGeometry(
  atl08_seg_att_dt,
  polygon,
  split_by = NULL
)
```

Arguments

`atl08_seg_att_dt`	A atl08_seg_att_dt object (output of `ATL08_seg_attributes_dt()` function). An S4 object of class `icesat2.atl08_dt`
`polygon`	Polygon. An object of class `terra::SpatVector`, which can be loaded as an ESRI shapefile using terra::vect function in the *sf* package.
`split_by`	Polygon id. If defined, ATL08 data will be clipped by each polygon using the polygon id from table of attribute defined by the user

Value

Returns an S4 object of class `icesat2.atl08_dt` containing the clipped ATL08 Terrain and Canopy Attributes.

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL08-derived terrain and canopy attributes
atl08_seg_att_dt <- ATL08_seg_attributes_dt(atl08_h5 = atl08_h5)

polygon_path <- system.file("extdata",
  "clip_geom.shp",
  package = "ICESat2VegR"
)

if (require(terra)) {
  polygon <- terra::vect(polygon_path)

  head(atl08_seg_att_dt)
  # Clipping ATL08 Terrain and Canopy Attributes by Geometry
  atl08_seg_att_dt_clip <- ATL08_seg_attributes_dt_clipGeometry(atl08_seg_att_dt,
      polygon, split_by = "id")

  hasLeaflet <- require(leaflet)

  if (hasLeaflet) {
    leaflet() %>%
      addCircleMarkers(atl08_seg_att_dt_clip$longitude,
        atl08_seg_att_dt_clip$latitude,
        radius = 1,
        opacity = 1,
        color = "red"
      ) %>%
      addScaleBar(options = list(imperial = FALSE)) %>%
      addPolygons(
        data = polygon, weight = 1, col = "white",
        opacity = 1, fillOpacity = 0
      ) %>%
      addProviderTiles(providers$Esri.WorldImagery,
        options = providerTileOptions(minZoom = 3, maxZoom = 17)
      )
  }
}
close(atl08_h5)
```

`ATL08_seg_attributes_dt_gridStat` *Statistics of ATL08 Terrain and Canopy Attributes at grid level*

Description

This function computes a series of user defined descriptive statistics within each grid cell for ATL08 Terrain and Canopy Attributes

Usage

```
ATL08_seg_attributes_dt_gridStat(atl08_seg_att_dt, func, res = 0.5)
```

Arguments

`atl08_seg_att_dt`	An S4 object of class `icesat2.atl08_dt` containing ATL08 data (output of `ATL08_seg_attributes_dt()` functions).

func	The function to be applied for computing the defined statistics
res	Spatial resolution in decimal degrees for the output SpatRast raster layer. Default is 0.5.

Value

Return a SpatRast raster layer(s) of selected ATL08 terrain and canopy attribute(s)

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL08-derived terrain and canopy attributes
atl08_seg_att_dt <- ATL08_seg_attributes_dt(atl08_h5 = atl08_h5)

# Computing the top h_canopy at 0.05 degree grid cell
res <- 0.0001
mean_h_canopy <- ATL08_seg_attributes_dt_gridStat(
  atl08_seg_att_dt,
  func = mean(h_canopy),
  res = res
)

plot(mean_h_canopy)

# Define your own function
mySetOfMetrics <- function(x) {
  metrics <- list(
    min = min(x, na.rm = TRUE), # Min of x
    max = max(x, na.rm = TRUE), # Max of x
    mean = mean(x, na.rm = TRUE), # Mean of x
    sd = sd(x, na.rm = TRUE) # Sd of x
  )
  return(metrics)
}

res <- 0.05
# Computing h_canopy statistics at 0.05 degree grid cell from user-defined function
h_canopy_metrics <- ATL08_seg_attributes_dt_gridStat(
  atl08_seg_att_dt,
  func = mySetOfMetrics(h_canopy),
  res = res
)

plot(h_canopy_metrics)
```

```
close(atl08_h5)
```

`ATL08_seg_attributes_dt_LAS`
Converts ATL08 segments to LAS

Description

Converts ATL08 segments to LAS

Usage

```
ATL08_seg_attributes_dt_LAS(atl08_dt, output)
```

Arguments

`atl08_dt`	An S4 object of class `icesat2.atl08_dt` containing ATL03 and ATL08 data (output of `ATL03_ATL08_photons_attributes_dt_join()` function).
`output`	character. The output path of for the LAS(Z) file(s) The function will create one LAS file per UTM Zone in WGS84 datum.

Details

As the las format expects a metric coordinate reference system (CRS) we use helper functions to define UTM zones to which the original ICESat-2 data will be converted.

The function credits go to Chuck Gantz- chuck.gantz@globalstar.com.

Value

Nothing, it just saves outputs as LAS file in disk

See Also

https://oceancolor.gsfc.nasa.gov/docs/ocssw/LatLong-UTMconversion_8cpp_source.html

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# # Extracting ATL03 and ATL08 photons and heights
```

```
atl08_dt <- ATL08_seg_attributes_dt(atl08_h5)

outputLaz <- tempfile(fileext = ".laz")
ATL08_seg_attributes_dt_LAS(
  atl08_dt,
  outputLaz
)

close(atl08_h5)
```

ATL08_seg_attributes_dt_polyStat

Statistics of ATL08 Terrain and Canopy Attributes by Geometry

---

Description

Computes a series of statistics of ATL08 terrain and canopy attributes within area defined by a polygon

Usage

```
ATL08_seg_attributes_dt_polyStat(atl08_seg_att_dt, func, poly_id = NULL)
```

Arguments

`atl08_seg_att_dt`	An S4 object of class `icesat2.atl08_dt` containing ATL08 terrain and canopy attributes (output of `ATL08_seg_attributes_dt()` function).
`func`	The function to be applied for computing the defined statistics
`poly_id`	Polygon id. If defined, statistics will be computed for each polygon

Value

Returns an S4 object of class `icesat2.atl08_dt` Containing Statistics of ATL08 terrain and canopy attributes

Examples

```
# Specifying the path to ATL08 file
atl08_path <- system.file(
  "extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL08 terrain and canopy attributes
```

```
atl08_seg_att_dt <- ATL08_seg_attributes_dt(atl08_h5 = atl08_h5)

# Specifying the path to shapefile
polygon_filepath <- system.file(
  "extdata",
  "clip_geom.shp",
  package = "ICESat2VegR"
)

# Reading shapefile as sf object
polygon <- terra::vect(polygon_filepath)

# Clipping ATL08 terrain and canopy attributes by Geometry
atl08_seg_att_dt_clip <- ATL08_seg_attributes_dt_clipGeometry(
  atl08_seg_att_dt,
  polygon,
  split_by = "id"
)

# Computing the max h_canopy by polygon id
max_h_canopy <- ATL08_seg_attributes_dt_polyStat(
  atl08_seg_att_dt_clip,
  func = max(h_canopy),
  poly_id = "poly_id"
)
head(max_h_canopy)

# Define your own function
mySetOfMetrics <- function(x) {
  metrics <- list(
    min = min(x), # Min of x
    max = max(x), # Max of x
    mean = mean(x), # Mean of x
    sd = sd(x) # Sd of x
  )
  return(metrics)
}

# Computing a series of canopy statistics from customized function
h_canopy_metrics <- ATL08_seg_attributes_dt_polyStat(
  atl08_seg_att_dt_clip,
  func = mySetOfMetrics(h_canopy),
  poly_id = "poly_id"
)

head(h_canopy_metrics)

close(atl08_h5)
```

ATL08_seg_attributes_h5_gridStat

Rasterize ATL08 canopy attributes from h5 files at large scale

Description

This function will read multiple ATL08 H5 files and create a stack of raster layers: count, and 1st, 2nd, 3rd and 4th moments (count, m1, m2, m3 and m4) for each metric selected, from which we can calculate statistics such as Mean, SD, Skewness and Kurtosis.

Usage

```
ATL08_seg_attributes_h5_gridStat(
  atl08_dir,
 metrics = c("h_canopy", "canopy_rh_conf", "h_median_canopy_abs", "h_min_canopy",
    "h_mean_canopy_abs", "h_median_canopy", "h_canopy_abs", "toc_roughness",
   "h_min_canopy_abs", "h_dif_canopy", "h_canopy_quad", "h_canopy_20m", "n_ca_photons",
   "photon_rate_can", "centroid_height", "canopy_h_metrics_abs", "h_mean_canopy",
    "subset_can_flag", "canopy_h_metrics", "n_toc_photons", "h_max_canopy_abs",
   "h_canopy_uncertainty", "canopy_openness", "h_max_canopy", "segment_cover"),
  beam = c("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"),
  out_root,
  ul_lat,
  ul_lon,
  lr_lat,
  lr_lon,
  res,
  creation_options = def_co,
  agg_function = default_agg_function,
  agg_join = default_agg_join,
  finalizer = default_finalizer
)
```

Arguments

`atl08_dir`	CharacterVector. The directory paths where the ATL08 H5 files are stored;
`metrics`	CharacterVector. A vector of canopy attributes available from ATL08 product (e.g. "h_canopy")
`beam`	Character vector indicating beams to process (e.g. "gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r")
`out_root`	Character. The root name for the raster output files, the pattern is {out_root}*{metric}*{count/m1/m2}.tif. This should include the full path for the file.
`ul_lat`	Numeric. Upper left latitude for the bounding box
`ul_lon`	Numeric. Upper left longitude for the bounding box
`lr_lat`	Numeric. Lower right latitude for the bounding box
`lr_lon`	Numeric. Lower right longitude for the bounding box

`res`	NumericVector. Resolution lon lat for the output raster in coordinates decimal degrees
`creation_options`	CharacterVector. The GDAL creation options for the tif file. Default c("COMPRESS=PACKBITS", "BIGTIFF=IF_SAFER", "TILED=YES", "BLOCKXSIZE=512", "BLOCKYSIZE=512") will create BIGTIFF if needed, with DEFLATE compression and tiled by 512x512 pixels.
`agg_function`	Formula function-like. An aggregate function which should return a data.table with the aggregate statistics
`agg_join`	Function. A function to merge two different agg objects.
`finalizer`	List<name, formula>. A list with the final raster names and the formula which uses the base statistics.

Details

This function will create five different aggregate statistics (n, mean, variance, min, max). One can calculate mean and standard deviation with the following formulas according to Terriberry (2007) and Joanes and Gill (1998):

The `agg_function` is a formula which return a data.table with the aggregate function to perform over the data. The default is:

```
~data.table(
    n = length(x),
    mean = mean(x,na.rm = TRUE),
    var = var(x) * (length(x) - 1),
    min = min(x, na.rm=T),
    max = max(x, na.rm=T)
  )
```

The `agg_join` is a function to merge two data.table aggregates from the `agg_function`. Since the h5 files will be aggregated one by one, the statistics from the different h5 files should have a function to merge them. The default function is:

```
function(x1, x2) {
    combined = data.table()
    x1$n[is.na(x1$n)] = 0
    x1$mean[is.na(x1$mean)] = 0
    x1$variance[is.na(x1$variance)] = 0
    x1$max[is.na(x1$max)] = -Inf
    x1$min[is.na(x1$min)] = Inf

    combined$n = x1$n + x2$n

    delta = x2$mean - x1$mean
    delta2 = delta * delta

    combined$mean = (x1$n * x1$mean + x2$n * x2$mean) / combined$n
    combined$variance = x1$variance + x2$variance +
      delta2 * x1$n * x2$n / combined$n

    combined$min = pmin(x1$min, x2$min, na.rm=F)
    combined$max = pmax(x1$max, x2$max, na.rm=F)
    return(combined)
}
```

The `finalizer` is a list of formulas to generate the final rasters based on the intermediate statistics from the previous functions. The default `finalizer` will calculate the `sd`, `skewness` and `kurtosis` based on the `variance`, `M3`, `M4` and `n` values. It is defined as:

```
list(
  sd = ~sqrt(variance/(n - 1)),
)
```

Value

Nothing. It outputs multiple raster tif files to the out_root specified path.

References

Joanes DN, Gill CA (1998). "Comparing measures of sample skewness and kurtosis." *Journal of the Royal Statistical Society: Series D (The Statistician)*, **47**(1), 183–189. doi:10.1111/1467-9884.00122.

Terriberry, Timothy B. (2007), Computing Higher-Order Moments Online, archived from the original on 23 April 2014, retrieved 5 May 2008

Examples

```
# Specifying the path to GEDI leveatl08_canopy_dt data (zip file)
library(ICESat2VegR)
library(data.table)

# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Bounding rectangle coordinates
ul_lat <- 41.5386848449707031
ul_lon <- -106.5708541870117188
lr_lat <- 41.5314979553222656
lr_lon <- -106.5699081420898438

res <- 100 # meters
```

```
lat_to_met_factor <- 1 / 110540
lon_to_met_factor <- 1 / 111320
xres <- lon_to_met_factor * res
yres <- lat_to_met_factor * res

agg_function <- ~ data.table(
  min = min(x),
  max = max(x),
  sum = sum(x),
  n = length(x)
)

agg_join <- function(agg1, agg2) {
  agg1[is.na(agg1)] <- 0
  data.table(
    min = pmin(agg1$min, agg2$min),
    max = pmax(agg1$max, agg2$max),
    sum = agg1$sum + agg2$sum,
    n = agg1$n + agg2$n
  )
}

finalizer <- list(
  mean = "sum/n",
  range = "max-min"
)

outdir <- tempdir()


gc()
file.remove(list.files(outdir, "*.tif"))
close(atl08_h5)
```

ATLAS_dataDownload *Download CESat-2 ATL03 and ALT08 data*

Description

Download ICESat-2 ATL03 and ALT08 data from LP DAAC Data Pool. Users will need to enter their Earth Explore login Information for downloading the data.

Usage

```
ATLAS_dataDownload(
  url,
  outdir = NULL,
  overwrite = FALSE,
  buffer_size = 512,
  timeout = 10
)
```

Arguments

`url`	character vector object; url to ATL03 or ALT08 data, or both (output of `ATLAS_dataFinder()`)
`outdir`	Vector object, output directory for downloading GEDI data, default `tempdir()`
`overwrite`	logical; overwrite file if they already exists in destination, default FALSE
`buffer_size`	integer; the size of download chunk in KB to hold in memory before writing to file, default 512.
`timeout`	integer; connection timeout in seconds.

Value

No return value on success, on failure it will `stop()`

References

Credits to Cole Krehbiel. Code adapted from https://git.earthdata.nasa.gov/projects/LPDUR/repos/daac_data_download_r/browse/DAACDataDownload.R

Examples

```
## Not run:
# Set path to ICESat-2 data
# herein we will only download xml metedata
base_path <- "https://data.nsidc.earthdatacloud.nasa.gov/nsidc-cumulus-prod-protected"
url <- c(
 paste0(base_path, "/ATLAS/ATL08/006/2019/07/13/ATL08_20190713210015_02410406_006_02.h5"),
 paste0(base_path, "/ATLAS/ATL08/006/2019/07/15/ATL08_20190715084425_02640402_006_02.h5")
)

# Set dir to download files to
outdir <- tempdir()

# Create .netrc file
netrc <- file.path(outdir, ".netrc")
netrc_conn <- file(netrc)

# Assuming login data is saved in the
# environmental variables EARTHDATA_USERNAME and EARTHDATA_PASSWORD
writeLines(c(
  "machine urs.earthdata.nasa.gov",
  sprintf("login %s", Sys.getenv("EARTHDATA_USERNAME")),
  sprintf("password %s", Sys.getenv("EARTHDATA_PASSWORD"))
), netrc_conn)

close(netrc_conn)

#' Downloading ICEsat-2 data
```

```
ATLAS_dataDownload(url, outdir)

## End(Not run)
```

`ATLAS_dataFinder` *ICESat-2 ATL03 and ATL08 data finder for either direct download or cloud computing*

Description

This function finds the exact granule(s) that contain ICESat-2 ATLAS data for a given region of interest and date range

Usage

```
ATLAS_dataFinder(
  short_name,
  lower_left_lon,
  lower_left_lat,
  upper_right_lon,
  upper_right_lat,
  version = "006",
  daterange = NULL,
  persist = TRUE,
  cloud_hosted = TRUE,
  cloud_computing = FALSE
)
```

Arguments

`short_name`	ICESat-2 ATLAS data level short_name; Options: "ATL03", "ATL08",
`lower_left_lon`	Numeric. Minimum longitude in (decimal degrees) for the bounding box of the area of interest.
`lower_left_lat`	Numeric. Minimum latitude in (decimal degrees) for the bounding box of the area of interest.
`upper_right_lon`	Numeric. Maximum longitude in lon (decimal degrees) for the bounding box of the area of interest.
`upper_right_lat`	Numeric. Maximum latitude in (decimal degrees) for the bounding box of the area of interest.
`version`	Character. The version of the ICESat-2 ATLAS product files to be returned (only V005 or V006). Default "006".
`daterange`	Vector. Date range. Specify your start and end dates using ISO 8601 [YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]Z. Ex.: c("2019-07-01T00:00:00Z","2020-05-22T23:59:59Z"). If NULL (default), the date range filter will be not applied.

`persist`	Logical. If TRUE, it will create the .netrc
`cloud_hosted`	Logical. Flag to indicate use of cloud hosted collections.To be used when the cloud computing parameter is FALSE.
`cloud_computing`	Logical. If TRUE, it will return granules for cloud computing directly, otherwise it will return links to be passed in the function ATLAS_dataDownload for data download.

Value

Return either a vector object pointing out the path to ICESat-2 ATLAS data found within the boundary box coordinates provided for data download or a vector object containing the granules hosted on the cloud for cloud computing directly.

See Also

bbox: Defined by the upper left and lower right corner coordinates, in lat,lon ordering, for the bounding box of the area of interest (e.g. lower_left_lon,lower_left_lat,upper_right_lon,upper_right_lat)

This function relies on the existing CMR tool: `https://cmr.earthdata.nasa.gov/search/site/docs/search/api.html`

Examples

```
# ICESat-2 data finder is a web service provided by NASA
# usually the request takes more than 5 seconds

# Specifying bounding box coordinates
lower_left_lon <- -96.0
lower_left_lat <- 40.0
upper_right_lon <- -96.5
upper_right_lat <- 40.5

# Specifying the date range
daterange <- c("2022-05-01", "2022-05-02")

# Extracting the path to ICESat-2 ATLAS data for the specified boundary box coordinates
# for data download

# Shouldn't be tested because it relies on a web service which might be down

ATLAS02b_list <- ATLAS_dataFinder(
  short_name = "ATL08",
  lower_left_lon,
  lower_left_lat,
  upper_right_lon,
  upper_right_lat,
  version = "006",
  daterange = daterange
)
```

build_ee_forest — *Given an R `randomForest::randomForest()` model, transform to a Google Earth Engine randomForest model*

Description

Given an R `randomForest::randomForest()` model, transform to a Google Earth Engine randomForest model

Usage

```
build_ee_forest(rf)
```

Arguments

`rf`	the `randomForest::randomForest()` model object.

Value

The Google Earth Engine classifier

clip — *Clip ICESat-2 Data*

Description

This function clips ATL03 and ATL08 HDF5 file within beam groups, but keeps metada and ancillary data the same.

Usage

```
clip(x, output, clip_obj, ...)

## S4 method for signature 'icesat2.atl03_h5,character,SpatExtent'
clip(x, output, clip_obj, ...)

## S4 method for signature 'icesat2.atl03_h5,character,SpatVector'
clip(x, output, clip_obj, polygon_id = "id", ...)

## S4 method for signature 'icesat2.atl03_h5,character,numeric'
clip(x, output, clip_obj, ...)

## S4 method for signature 'icesat2.atl08_h5,character,SpatExtent'
clip(x, output, clip_obj, ...)

## S4 method for signature 'icesat2.atl08_h5,character,numeric'
```

```
clip(x, output, clip_obj, ...)

## S4 method for signature 'icesat2.atl08_h5,character,SpatVector'
clip(x, output, clip_obj, polygon_id, ...)
```

Arguments

`x`	An `icesat2.atl08_h5` object, obtained through `ATL08_read()`.
`output`	A character string specifying the path to the output H5 file.
`clip_obj`	A `terra::SpatVector` object for clipping. The bounding box order is the default from NASA's ICESat-2 data searching: (ul_lat, ul_lon, lr_lat, lr_lon).
`...`	Additional arguments passed to specific method implementation.
`polygon_id`	A character string specifying the ID of the polygon for clipping.

Details

Generic function for clipping ICESat-2 ATL03 and ATL08 H5 data.

Value

Returns the clipped S4 object of the same class as the input file

Returns the clipped `icesat2.atl03_h5` object.

Returns the clipped `icesat2.atl03_h5` object.

Returns the clipped `icesat2.atl08_h5` object.

Returns the clipped `icesat2.atl08_h5` object.

Returns the clipped `icesat2.atl08_h5` object.

Functions

- `clip(x = icesat2.atl03_h5, output = character, clip_obj = SpatExtent)`: Clips ATL03 data using a `SpatExtent`.
- `clip(x = icesat2.atl03_h5, output = character, clip_obj = SpatVector)`: Clips ATL03 data using a `SpatVector`.

  This method clips ATL03 HDF5 data within beam groups using a `terra::SpatVector` object, but keeps metadata and ancillary data the same.
- `clip(x = icesat2.atl03_h5, output = character, clip_obj = numeric)`: Clips ATL03 data using numeric coordinates.

  This method clips ATL03 HDF5 data within beam groups using numeric coordinates, but keeps metadata and ancillary data the same.
- `clip(x = icesat2.atl08_h5, output = character, clip_obj = SpatExtent)`: Clips ATL08 data using a `SpatExtent`.

  This method clips ATL08 HDF5 data within beam groups using a `terra::SpatExtent` object, but keeps metadata and ancillary data the same.

- `clip(x = icesat2.atl08_h5, output = character, clip_obj = numeric)`: Clips ATL08 data using numeric coordinates.

  This method clips ATL08 HDF5 data within beam groups using numeric coordinates, but keeps metadata and ancillary data the same.
- `clip(x = icesat2.atl08_h5, output = character, clip_obj = SpatVector)`: Clips ATL08 data using a `SpatVector`.

  This method clips ATL08 HDF5 data within beam groups using a `terra::SpatVector` object, but keeps metadata and ancillary data the same.

See Also

`ATL03_h5_clipBox()`, `ATL03_h5_clipGeometry()`, `ATL08_h5_clipBox()`, `ATL08_h5_clipGeometry()`

Examples

```
# ATL08 file path
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

output <- tempfile(fileext = ".h5")

vect_path <- system.file("extdata",
  "clip_geom.shp",
  package = "ICESat2VegR"
)

vect <- terra::vect(vect_path)
ext <- terra::ext(vect)

# Clipping ATL08 photons by boundary box extent
atl08_clip <- clip(
  atl08_h5,
  output,
  ext
)

close(atl08_clip)

# Clipping ATL08 photons by geometry
atl08_clip_geom <- clip(
  atl08_h5,
  output,
  vect,
  polygon_id = "id"
)

close(atl08_clip_geom, close)
```

clip,ANY,ANY,ANY-method

Default clip method

Description

Default clip method

Usage

```
## S4 method for signature 'ANY,ANY,ANY'
clip(x, output, clip_obj, ...)
```

Arguments

x	Maps to x1 from graphics::clip()
output	Maps to x2 from graphics::clip()
clip_obj	Maps to y1 from graphics::clip()
...	Maps first element of list to y2 of graphics::clip()

Functions

- `clip(x = ANY, output = ANY, clip_obj = ANY)`: clip will dispatch to `graphics::clip()`

close,icesat2.h5-method

Safely closes the ICESat2.h5 base classes

Description

Closing files will avoid locking HDF5 ATL03 files.

Usage

```
## S4 method for signature 'icesat2.h5'
close(con, ...)
```

Arguments

con	An object of class ICESat2.h5
...	Inherited from base

`close.GDALDataset` *Method to close GDALDataset*

Description

Closing ensures the data is actually flushed (saved) to the dataset also it releases the lock from the file.

Usage

```
## S3 method for class 'GDALDataset'
close(con, ...)
```

Arguments

`con`	GDALDataset to be closed.
`...`	not used, inherited from generic close function.

`close.icesat2.predict_h5` *Closes the HDF5 file pointer to release resources*

Description

Closes the HDF5 file pointer to release resources

Usage

```
## S3 method for class 'icesat2.predict_h5'
close(con, ...)
```

Arguments

`con`	The HDF5 file pointer of class `icesat2.predict_h5`
`...`	Additional parameters inherited from generic

Value

Nothing, just closes the HDF5 file pointer

createDataset *Creates a new GDALDataset*

Description

Create a new raster file based on specified data. It will output a *.tif file.

Usage

```
createDataset(
  raster_path,
  nbands,
  datatype,
  projstring,
  lr_lat,
  ul_lat,
  ul_lon,
  lr_lon,
  res,
  nodata,
  co = c("TILED=YES", "BLOCKXSIZE=512", "BLOCKYSIZE=512", "COMPRESSION=LZW")
)
```

Arguments

raster_path	Character. The output path for the raster data
nbands	Integer. Number of bands. Default 1.
datatype	GDALDataType. The GDALDataType to use for the raster, use (GDALDataType$) to find the options. Default GDALDataType$GDT_Float64
projstring	The projection string, either proj or WKT is accepted.
lr_lat	Numeric. The lower right latitude.
ul_lat	Numeric. The upper left latitude.
ul_lon	Numeric. The upper left longitude.
lr_lon	Numeric. The lower right longitude.
res	Numeric. The resolution of the output raster
nodata	Numeric. The no data value for the raster. Default 0.
co	CharacterVector. A CharacterVector of creation options for GDAL. Default NULL

Value

An object from GDALDataset R6 class.

Examples

```
# Parameters
raster_path <- tempfile(fileext = ".tif")
ul_lat <- -15
ul_lon <- -45
lr_lat <- -25
lr_lon <- -35
res <- c(0.01, -0.01)
datatype <- GDALDataType$GDT_Int32
nbands <- 1
projstring <- "EPSG:4326"
nodata <- -1
co <- c("TILED=YES", "BLOCKXSIZE=512", "BLOCKYSIZE=512", "COMPRESSION=LZW")

# Create a new raster dataset
ds <- createDataset(
  raster_path = raster_path,
  nbands = nbands,
  datatype = datatype,
  projstring = projstring,
  lr_lat = lr_lat,
  ul_lat = ul_lat,
  ul_lon = ul_lon,
  lr_lon = lr_lon,
  res = res,
  nodata = nodata,
  co = co
)

# Get the GDALRasterBand for ds
band <- ds[[1]]

# Set some dummy values
band[[0, 0]] <- 1:(512 * 512)

ds$Close()
```

`earthaccess` *The pointer to the* `earthaccess` *python reticulate module*

Description

The pointer to the `earthaccess` python reticulate module

Usage

```
earthaccess
```

Format

An object of class `python.builtin.module` (inherits from `python.builtin.object`) of length 1.

`earthaccess_login` *Try logging in on earthaccess*

Description

Try logging in on earthaccess

Usage

```
earthaccess_login(persist = TRUE)
```

Arguments

`persist`	Logical. If TRUE, it will persist the login credentials in .netrc file.

Value

Nothing, just try to login in earthaccess

Examples

```
# Try to login in NASA earthaccess

# Shouldn't be tested because it relies on a web services requiring authentication.
earthaccess_login()
```

`ee` *The pointer to the* `earth-engine-api` *python reticulate module*

Description

The pointer to the `earth-engine-api` python reticulate module

Usage

```
ee
```

Format

An object of class `python.builtin.module` (inherits from `python.builtin.object`) of length 1.

See Also

https://developers.google.com/earth-engine/apidocs

`ee_initialize` *Initializes the Google Earth Engine API*

Description

Initializes the Google Earth Engine API

Usage

```
ee_initialize()
```

Value

Nothing, it just initializes the Google Earth Engine API

`ee_number` *Creates an* `Earth Engine` *server number*

Description

Creates an `Earth Engine` server number

Usage

```
ee_number(x)
```

Arguments

`x`	the number to convert to Earth Engine's number.

Value

The Earth Engine number

See Also

https://developers.google.com/earth-engine/apidocs/ee-number

`extract` *Given a geometry with point samples and images from Earth Engine retrieve the point geometry with values for the images*

Description

Given a geometry with point samples and images from Earth Engine retrieve the point geometry with values for the images

Usage

```
extract(stack, geom, scale)
```

Arguments

`stack`	A single image or a vector/list of images from Earth Engine.
`geom`	A geometry from `terra::SpatVector` read with `terra::vect()`.
`scale`	The scale in meters for the extraction (image resolution).

Value

An ee.FeatureCollection with the properties extracted from the stack of images from ee.

`formulaCalculate` *Calculate raster values based on a formula*

Description

Calculate raster values based on a formula

Usage

```
formulaCalculate(formula, data, updateBand)
```

Arguments

`formula`	Formula. A formula to apply to the RasterBands from `data`
`data`	List. A named list with the used variables in the textual formula
`updateBand`	GDALRasterBand. The GDALRasterBand which will be updated with the calculated values.

Value

Nothing, it just updates the band of interest.

Examples

```
# Parameters
raster_path <- file.path(tempdir(), "output.tif")
ul_lat <- -15
ul_lon <- -45
lr_lat <- -25
lr_lon <- -35
res <- c(0.01, -0.01)
datatype <- GDALDataType$GDT_Int32
nbands <- 2
projstring <- "EPSG:4326"
nodata <- -1
co <- c("TILED=YES", "BLOCKXSIZE=512", "BLOCKYSIZE=512", "COMPRESSION=LZW")

# Create a new raster dataset
ds <- createDataset(
  raster_path = raster_path,
  nbands = nbands,
  datatype = datatype,
  projstring = projstring,
  lr_lat = lr_lat,
  ul_lat = ul_lat,
  ul_lon = ul_lon,
  lr_lon = lr_lon,
  res = res,
  nodata = nodata,
  co = co
)

# Get the GDALRasterBand for ds
band <- ds[[1]]

# The updateBand can be the same
# using a different one just for testing
updateBand <- ds[[2]]

# Set some dummy values
band[[0, 0]] <- 1:(512 * 512)

# Calculate the double - 10
formulaCalculate(
  formula = ~ x * 2 - 10,
  data = list(x = band),
  updateBand = updateBand
)

ds$Close()
```

GDALDataset *R6 Class GDALDataset wrapping*

Description

Wrapping class for GDALDataset C++ API exporting GetRasterBand, GetRasterXSize, GetRasterY-Size

Methods

Public methods:

- `GDALDataset$new()`
- `GDALDataset$GetRasterBand()`
- `GDALDataset$GetRasterXSize()`
- `GDALDataset$GetRasterYSize()`
- `GDALDataset$Close()`
- `GDALDataset$clone()`

Method `new()`**:** Create a new raster file based on specified data. It will output a *.tif file.

Usage:

```
GDALDataset$new(ds)
```

Arguments:

`ds` GDALDatasetR pointer. Should not be used

`datatype` GDALDataType. The GDALDataType to use for the raster, use (GDALDataType$) to find the options. Default GDALDataType$GDT_Float64

Returns: An object from GDALDataset R6 class.

Method `GetRasterBand()`**:** Function to retrieve the GDALRasterBand R6 Object.

Usage:

```
GDALDataset$GetRasterBand(x)
```

Arguments:

`x` Integer. The band index, starting from 1 to number of bands.

Returns: An object of GDALRasterBand R6 class.

Method `GetRasterXSize()`**:** Get the width for the raster

Usage:

```
GDALDataset$GetRasterXSize()
```

Returns: An integer indicating the raster width

Method `GetRasterYSize()`**:** Get the height for the raster

Usage:

```
GDALDataset$GetRasterYSize()
```

Returns: An integer indicating the raster height

Method `Close()`**:** Closes the GDALDataset

Usage:

```
GDALDataset$Close()
```

Returns: An integer indicating the raster width

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
GDALDataset$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`GDALDataType` *List of datatypes supported by the GDALDataset R6 class*

Description

List of datatypes supported by the GDALDataset R6 class

Usage

```
GDALDataType
```

Format

An object of class `list` of length 7.

`GDALOpen` *Open GDAL raster*

Description

Function to open GDAL Dataset

Usage

```
GDALOpen(filename, readonly = TRUE)
```

Arguments

`filename`	Character. The path to a GDAL dataset.
`readonly`	Logical. Flag to open a read only GDALDataset with GA_ReadOnly or GA_Update. Default TRUE.

Value

An R6 object of GDALDataset class.

Examples

```
ds_path <- system.file("extdata", "example.tif", package = "ICESat2VegR")

ds <- GDALOpen(ds_path)
ds$Close()
```

GDALRasterBand *R6 Class GDALRasterBand wrapping*

Description

Wrapping class for GDALRasterBand C++ API exporting GetBlockXSize, GetBlockYSize, ReadBlock, WriteBlock for better IO speed.

Methods

Public methods:

- `GDALRasterBand$new()`
- `GDALRasterBand$ReadBlock()`
- `GDALRasterBand$WriteBlock()`
- `GDALRasterBand$GetBlockXSize()`
- `GDALRasterBand$GetBlockYSize()`
- `GDALRasterBand$GetXSize()`
- `GDALRasterBand$CalculateStatistics()`
- `GDALRasterBand$GetYSize()`
- `GDALRasterBand$GetNoDataValue()`
- `GDALRasterBand$GetRasterDataType()`
- `GDALRasterBand$clone()`

Method `new()`: Creates a new GDALRasterBand

Usage:

`GDALRasterBand$new(band)`

Arguments:

`band` The C++ pointer to the GDALRasterBandR object.

`datatype` The GDALDataType for this band

Returns: An object of GDALRasterBand R6 class

Method `ReadBlock()`: Efficiently reads a raster block

Usage:

`GDALRasterBand$ReadBlock(iXBlock, iYBlock)`

Arguments:

`iXBlock` Integer. The i-th column block to access. The `iXBlock` will be offset $BLOCKXSIZE \times iXBlock$ from the origin.

`iYBlock` Integer. The i-th row block to access. The `iYBlock` will be offset $BLOCKYSIZE \times iYBlock$ from the origin.

Details: The returned Vector will be single dimensional with the length $BLOCKXSIZE \times BLOCKYSIZE$. If you use matrix(, ncol=BLOCKXSIZE) the matrix returned will actually be transposed. You should either transpose it or you can calculate the indices using $y \cdot xsize + x$

Returns: RawVector for GDALDataType$GDT_Byte, IntegerVector for int types and NumericVector for floating point types.

Method `WriteBlock()`**:** Efficiently writes a raster block

Usage:

```
GDALRasterBand$WriteBlock(iXBlock, iYBlock, buffer)
```

Arguments:

`iXBlock` Integer. The i-th column block to write. The `iXBlock` will be offset $BLOCKXSIZE \times iXBlock$ from the origin.

`iYBlock` Integer. The i-th row block to write. The `iYBlock` will be offset $BLOCKYSIZE \times iYBlock$ from the origin.

`buffer` RawVector/IntegerVector/NumericVector depending on the GDALDataType. This should be a 1D vector with size equal to raster $BLOCKXSIZE \times BLOCKYSIZE$.

Details: The returned Vector will be single dimensional with the length $BLOCKXSIZE \times BLOCKYSIZE$. If you use matrix(, ncol=BLOCKXSIZE) the matrix returned will actually be transposed. You should either transpose it or you can calculate the indices using $y \cdot xsize + x$.

Returns: Nothing

Method `GetBlockXSize()`**:** Get the block width

Usage:

```
GDALRasterBand$GetBlockXSize()
```

Returns: An integer indicating block width

Method `GetBlockYSize()`**:** Get the block height

Usage:

```
GDALRasterBand$GetBlockYSize()
```

Returns: An integer indicating block height

Method `GetXSize()`**:** Get the band width

Usage:

```
GDALRasterBand$GetXSize()
```

Returns: An integer indicating band width

Method `CalculateStatistics()`**:** Calculate statistics for the `GDALRasterBand`

Usage:

```
GDALRasterBand$CalculateStatistics()
```

Returns: nothing

Method `GetYSize()`**:** Get the band height

Usage:

`GDALRasterBand$GetYSize()`

Returns: An integer indicating band height

Method `GetNoDataValue()`**:** Get band no data value

Usage:

`GDALRasterBand$GetNoDataValue()`

Returns: Numeric indicating no data value

Method `GetRasterDataType()`**:** Get band datatype

Usage:

`GDALRasterBand$GetRasterDataType()`

Returns: Numeric indicating the datatype

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

`GDALRasterBand$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

Note

This constructor must not be called at all, this is automatically called from GDALDataset$GetRasterBand function.

`geomSampling` *Get observations given a minimum radius distance between samples*

Description

Get observations given a minimum radius distance between samples

Usage

```
geomSampling(size, geom, split_id = NULL, chainSampling = NULL)
```

Arguments

`size`	the sample size. Either an integer of absolute number of samples or if it is between (0, 1) it will sample a percentage relative to the number of available observations within the group.
`geom`	a `terra::SpatVector` object opened with `terra::vect()`
`split_id`	character. The variable name of the geometry to use as split factor for sampling
`chainSampling`	chains different methods of sampling by providing the result of another samplingMethod `randomSampling()`, `spacedSampling()`, `stratifiedSampling()`.

Value

A `icesat2_sampling_method` for defining the method used in `sample()`

`getTileUrl` *Retrieve Google Earth Engine's tile url for an Image or ImageCollection*

Description

Retrieve Google Earth Engine's tile url for an Image or ImageCollection

Usage

```
getTileUrl(img)
```

Arguments

`img`	The `Image` or `ImageCollection` to retrieve the URL

Value

The url for the tile service.

`get_catalog_id` *Retrieve the Google Earth Engine image catalog id*

Description

Retrieve the Google Earth Engine image catalog id

Usage

```
get_catalog_id(id)
```

Arguments

`id`	character. The id retrieved from the data.table resulting from `search_datasets()`.

Value

The catalog id to open within Google Earth Engine.

glcmTexture *Maps to* `ee.Image.glcmTexture`

Description

Computes texture metrics from the Gray Level Co-occurrence Matrix around each pixel of every band. The GLCM is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in an image. It counts the number of times a pixel of value X lies next to a pixel of value Y, in a particular direction and distance. and then derives statistics from this tabulation.

Usage

```
glcmTexture(x, size = 1, kernel = NULL, average = TRUE)
```

Arguments

`x`	The input image to calculate the texture on.
`size`	integer, default 1. The size of the neighborhood to include in each GLCM.
`kernel`	default NULL. A kernel specifying the x and y offsets over which to compute the GLCMs. A GLCM is computed for each pixel in the kernel that is non-zero, except the center pixel and as long as a GLCM hasn't already been computed for the same direction and distance. For example, if either or both of the east and west pixels are set, only 1 (horizontal) GLCM is computed. Kernels are scanned from left to right and top to bottom. The default is a 3x3 square, resulting in 4 GLCMs with the offsets (-1, -1), (0, -1), (1, -1) and (-1, 0).
`average`	logical, default TRUE. If true, the directional bands for each metric are averaged.

Details

This implementation computes the 14 GLCM metrics proposed by Haralick, and 4 additional metrics from Conners. Inputs are required to be integer valued.

The output consists of 18 bands per input band if directional averaging is on and 18 bands per directional pair in the kernel, if not:

1. ASM: f1, Angular Second Moment; measures the number of repeated pairs
2. CONTRAST: f2, Contrast; measures the local contrast of an image
3. CORR: f3, Correlation; measures the correlation between pairs of pixels
4. VAR: f4, Variance; measures how spread out the distribution of gray-levels is
5. IDM: f5, Inverse Difference Moment; measures the homogeneity
6. SAVG: f6, Sum Average
7. SVAR: f7, Sum Variance
8. SENT: f8, Sum Entropy
9. ENT: f9, Entropy. Measures the randomness of a gray-level distribution

10. DVAR: f10, Difference variance
11. DENT: f11, Difference entropy
12. IMCORR1: f12, Information Measure of Corr. 1
13. IMCORR2: f13, Information Measure of Corr. 2
14. MAXCORR: f14, Max Corr. Coefficient. (not computed)
15. DISS: Dissimilarity
16. INERTIA: Inertia
17. SHADE: Cluster Shade
18. PROM: Cluster prominence

Value

Another Earth Engine image with bands described on details section.

See Also

More information can be found in the two papers: Haralick et. al, 'Textural Features for Image Classification', http://doi.org/10.1109/TSMC.1973.4309314 and Conners, et al, Segmentation of a high-resolution urban scene using texture operators', http://doi.org/10.1016/0734-189X(84)90197-X.

https://developers.google.com/earth-engine/apidocs/ee-image-glcmtexture

---

`gridSampling` *Get samples stratified by grid cells of specified size*

Description

Get samples stratified by grid cells of specified size

Usage

```
gridSampling(size, grid_size, chainSampling = NULL)
```

Arguments

`size`	the sample size. Either an integer of absolute number of samples or if it is between (0, 1) it will sample a percentage relative to the number of available observations within the group.
`grid_size`	generic params. The `gridSampling()` will take a `grid_size` parameter defining the grid size for the sampling
`chainSampling`	chains different methods of sampling by providing the result of another samplingMethod `randomSampling()`, `spacedSampling()`, `stratifiedSampling()`.

Value

A `icesat2_sampling_method` for defining the method used in `sample()`

icesat2.atl03atl08_dt-class

Class for joined ATL03 and ATL08 attributes

Description

Class for joined ATL03 and ATL08 attributes

See Also

data.table in the data.table package and https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

icesat2.atl03_atl08_seg_dt-class

Class that represent custom segments created from ATL03 and ATL08 joined data

Description

Class that represent custom segments created from ATL03 and ATL08 joined data

Details

This class is actually just a wrap around the data.table, but it indicates the output from ATL03_ATL08_segment_create(), which means the dataset will contain the needed structure for computing value for the computing the stats with ATL03_ATL08_compute_seg_attributes_dt_segStat()

icesat2.atl03_dt-class

Class for ATL03 attributes

Description

Class for ATL03 attributes

See Also

data.table in the data.table package and https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

`icesat2.atl03_h5-class` *Class for ICESat-2 ATL03*

Description

Class for ICESat-2 ATL03

Slots

`h5` Object of class `H5File` from `hdf5r` package containing the ICESat-2 Global Geolocated Photon Data (ATL03)

See Also

`H5File` in the `hdf5r` package and `https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf`

`icesat2.atl03_seg_dt-class` *Class for ATL03 segment attributes*

Description

Class for ATL03 segment attributes

See Also

`data.table` in the `data.table` package and `https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf`

`icesat2.atl08_dt-class` *Class for ATL08 attributes*

Description

Class for ATL08 attributes

See Also

`data.table` in the `data.table` package and `https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf`

`icesat2.atl08_h5-class` *Class for ICESat-2 ATL08*

Description

Class for ICESat-2 ATL08

Slots

`h5` Object of class `H5File` from `hdf5r` package containing the ICESat-2 Land and Vegetation Along-Track Products (ATL08)

See Also

`H5File` in the `hdf5r` package and `https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL08_ATBD_r006.pdf`

`icesat2.h5-class` *Base class for all ICESat2VegR package's H5 files for generic functions that can be run on any H5*

Description

Base class for all ICESat2VegR package's H5 files for generic functions that can be run on any H5

`ICESat2.h5ds_cloud` *Class representing dataset opened from the cloud using h5py*

Description

Class representing dataset opened from the cloud using h5py

Class representing dataset opened from the cloud using h5py

Details

This class should not be instanced by the user, instead it is automatically handled when the user opens a dataset using `[[]]` operator from a h5 file.

Public fields

`ds` Dataset pointer

`dims` Dimensions of the dataset

`chunk_dims` Chunk dimensions for the dataset

Methods

Public methods:

- `ICESat2.h5ds_cloud$new()`
- `ICESat2.h5ds_cloud$get_type()`
- `ICESat2.h5ds_cloud$get_fill_value()`
- `ICESat2.h5ds_cloud$get_creation_property_list()`
- `ICESat2.h5ds_cloud$clone()`

Method `new()`**:** Constructor using a dataset pointer

Usage:

```
ICESat2.h5ds_cloud$new(ds)
```

Arguments:

`ds` Dataset pointer

Method `get_type()`**:** Get the type of data for this dataset

Usage:

```
ICESat2.h5ds_cloud$get_type()
```

Method `get_fill_value()`**:** Get the fill_value used by the dataset

Usage:

```
ICESat2.h5ds_cloud$get_fill_value()
```

Method `get_creation_property_list()`**:** Get creation property list for the dataset (plist)

Usage:

```
ICESat2.h5ds_cloud$get_creation_property_list()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ICESat2.h5ds_cloud$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ICESat2.h5ds_local` *Class representing dataset opened from locally using hdf5r*

Description

Class representing dataset opened from locally using hdf5r

Class representing dataset opened from locally using hdf5r

Details

This class should not be instanced by the user, instead it is automatically handled when the user opens a dataset using `[[]]` operator from a h5 file.

Public fields

`ds` Dataset pointer

`dims` Dimensions of the dataset

`chunk_dims` Chunk dimensions for the dataset

Methods

Public methods:

- `ICESat2.h5ds_local$new()`
- `ICESat2.h5ds_local$get_type()`
- `ICESat2.h5ds_local$get_fill_value()`
- `ICESat2.h5ds_local$get_creation_property_list()`
- `ICESat2.h5ds_local$clone()`

Method `new()`**:** Constructor using a dataset pointer

Usage:

```
ICESat2.h5ds_local$new(ds)
```

Arguments:

`ds` Dataset pointer

Method `get_type()`**:** Get the type of data for this dataset

Usage:

```
ICESat2.h5ds_local$get_type()
```

Method `get_fill_value()`**:** Get the fill_value used by the dataset

Usage:

```
ICESat2.h5ds_local$get_fill_value()
```

Method `get_creation_property_list()`**:** Get creation property list for the dataset (plist)

Usage:

```
ICESat2.h5ds_local$get_creation_property_list()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ICESat2.h5ds_local$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ICESat2.h5_cloud`	*The class representing the h5 file opened from the cloud for cloud computing.*

Description

The class representing the h5 file opened from the cloud for cloud computing.

The class representing the h5 file opened from the cloud for cloud computing.

Details

Besides representing h5 files, it is also used to represent groups opened with `[[]]`.

The variants `_cloud` and `_local` allows all the other functions to use generic calls using the same interface, with each class implementation is provided accordingly.

The regular usage does not require the user to work with those classes as most other provided functions will actually give access to the most common necessities for working with ICESat-2 data.

Public fields

`h5` A pointer to `h5py` in case the user wants to access features not implemented yet

`beams` The `character` vector of beams available for the granule.

`strong_beams` The `character` vector of strong beams calculated using orbit_info

`weak_beams` The `character` vector of weak beams calculated using orbit_info

Methods

Public methods:

- `ICESat2.h5_cloud$new()`
- `ICESat2.h5_cloud$ls()`
- `ICESat2.h5_cloud$ls_groups()`
- `ICESat2.h5_cloud$ls_attrs()`
- `ICESat2.h5_cloud$dt_datasets()`
- `ICESat2.h5_cloud$exists()`
- `ICESat2.h5_cloud$attr()`
- `ICESat2.h5_cloud$close_all()`
- `ICESat2.h5_cloud$print()`
- `ICESat2.h5_cloud$clone()`

Method `new()`: Direct initialization should not be used, it is handled by `ATL0X_read()`

Usage:

```
ICESat2.h5_cloud$new(h5)
```

Arguments:

`h5` the result of the `ATLAS_dataFinder()` with `cloud_computing = TRUE`

Returns: The class object

Method `ls()`**:** Lists the groups and datasets that are within current group

Usage:

```
ICESat2.h5_cloud$ls()
```

Returns: List the groups and datasets within the current path

Method `ls_groups()`**:** Lists all grouops recursively

Usage:

```
ICESat2.h5_cloud$ls_groups(recursive = FALSE)
```

Arguments:

`recursive` `logical`, default FALSE. If TRUE it will list groups recursively and return the full path.

Returns: The `character` representing

Method `ls_attrs()`**:** Lists the available attributes

Usage:

```
ICESat2.h5_cloud$ls_attrs()
```

Returns: `character` vector of attributes available

Method `dt_datasets()`**:** Get datasets as data.table with columns (name, dataset.dims, rank)

Usage:

```
ICESat2.h5_cloud$dt_datasets(recursive = FALSE)
```

Arguments:

`recursive` `logical`, default FALSE. If TRUE recursively searches and returns the full path.

Returns: A `data.table::data.table` with the columns (name, dataset.dims, rank)

Method `exists()`**:** Checks if a supplied group/dataset exist within the H5

Usage:

```
ICESat2.h5_cloud$exists(path)
```

Arguments:

`path` `character` with the path to test

Returns: `logical` TRUE or FALSE if it exists or not

Method `attr()`**:** Read an attribute from h5

Usage:

```
ICESat2.h5_cloud$attr(attribute)
```

Arguments:

`attribute` `character` the address of the attribute to open

Method `close_all()`**:** Safely closes the h5 file pointer

Usage:

```
ICESat2.h5_cloud$close_all()
```

Returns: Nothing, just closes the file

Method `print()`**:** Prints the data in a friendly manner

Usage:

```
ICESat2.h5_cloud$print(...)
```

Arguments:

`...` Added for compatibility purposes with generic signature

Returns: Outputs information about object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ICESat2.h5_cloud$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ICESat2.h5_local` *The class representing the h5 file opened from local files.*

Description

The class representing the h5 file opened from local files.

The class representing the h5 file opened from local files.

Details

The variants `_cloud` and `_local` allows all the other functions to use generic calls using the same interface, with each class implementation is provided accordingly.

The regular usage does not require the user to work with those classes as most other provided functions will actually give access to the most common necessities for working with ICESat-2 data.

Public fields

`h5` A pointer to `hdf5r::H5File` in case the user wants to access features not implemented yet

`beams` The `character` vector of beams available for the granule.

`strong_beams` The `character` vector of strong beams calculated using orbit_info

`weak_beams` The `character` vector of weak beams calculated using orbit_info

`isOpen` A flag to indicate if the file pointer has already been closed

Methods

Public methods:

- `ICESat2.h5_local$new()`
- `ICESat2.h5_local$ls()`
- `ICESat2.h5_local$ls_groups()`
- `ICESat2.h5_local$ls_attrs()`
- `ICESat2.h5_local$dt_datasets()`
- `ICESat2.h5_local$exists()`
- `ICESat2.h5_local$attr()`
- `ICESat2.h5_local$close_all()`
- `ICESat2.h5_local$print()`
- `ICESat2.h5_local$clone()`

Method `new()`**:** Direct initialization should not be used, it is handled by `ATL0X_read()`

Usage:

```
ICESat2.h5_local$new(h5)
```

Arguments:

`h5` the result of the `ATLAS_dataFinder()`

Returns: The class object

Method `ls()`**:** Lists the groups and datasets that are within current group

Usage:

```
ICESat2.h5_local$ls()
```

Returns: List the groups and datasets within the current path

Method `ls_groups()`**:** Lists all grouops recursively

Usage:

```
ICESat2.h5_local$ls_groups(recursive = FALSE)
```

Arguments:

`recursive` `logical`, default FALSE. If TRUE it will list groups recursively and return the full path.

Returns: The `character` representing

Method `ls_attrs()`**:** Lists the available attributes

Usage:

```
ICESat2.h5_local$ls_attrs()
```

Returns: `character` vector of attributes available

Method `dt_datasets()`**:** Get datasets as data.table with columns (name, dataset.dims, rank)

Usage:

```
ICESat2.h5_local$dt_datasets(recursive = FALSE)
```

Arguments:

`recursive` `logical`, default FALSE. If TRUE recursively searches and returns the full path.

Returns: A `data.table::data.table` with the columns (name, dataset.dims, rank)

Method `exists()`**:** Checks if a supplied group/dataset exist within the H5

Usage:

```
ICESat2.h5_local$exists(path)
```

Arguments:

`path` `character` with the path to test

Returns: `logical` TRUE or FALSE if it exists or not

Method `attr()`**:** Read an attribute from h5

Usage:

```
ICESat2.h5_local$attr(attribute)
```

Arguments:

`attribute` `character` the address of the attribute to open

Method `close_all()`**:** Safely closes the h5 file pointer

Usage:

```
ICESat2.h5_local$close_all(silent = TRUE)
```

Arguments:

`silent` `logical`, default TRUE. Will cast warning messages if silent = FALSE.

Returns: Nothing, just closes the file

Method `print()`**:** Prints the data in a friendly manner

Usage:

```
ICESat2.h5_local$print(...)
```

Arguments:

`...` Added for compatibility purposes with generic signature

Returns: Outputs information about object

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ICESat2.h5_local$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`icesat2.predict_h5-class`

HDF5 prediction class

Description

This is a generic prediction class for HDF5 files to make type checking and method matching with S4 methods.

`ICESat2VegR_configure` *Configure environment for using Google Earth Engine functions and accessing earthaccess cloud data*

Description

Configure environment for using Google Earth Engine functions and accessing earthaccess cloud data

Usage

```
ICESat2VegR_configure()
```

Details

This function will configure the python environment as required by this package for:

- Accessing the h5 files directly from the cloud for cloud computing
- Using earth engine functions wrappers that are made available within this package.

Value

TRUE if everything works as expected, it is just an interactive installer

`icesat2_sampling_method-class` *Class for sampling methods to be passed on for the sample function*

Description

Class for sampling methods to be passed on for the sample function

---

`length,ee.imagecollection.ImageCollection-method`
Returns the number of images in an ImageCollection

Description

Returns the number of images in an ImageCollection

Usage

```
## S4 method for signature 'ee.imagecollection.ImageCollection'
length(x)
```

Arguments

`x`	The `ImageCollection` to get the length of

Value

The number of images in the ImageCollection

`map_create` *Map create*

Description

Applies the fitted model as `randomForest::randomForest` object to the input raster stack (as ee.Image) to produce a wall-to-wall maps.

Usage

```
map_create(model, stack)
```

Arguments

`model`	`randomForest::randomForest`. The classifier (regression) model to use for the prediction.
`stack`	A vector/list of ee.Image created using the ee API.

Value

An `ee.Image` resulting from applying the fitted model.

`model_fit` *Fit different models based on x and y inputs*

Description

Fit different models based on x and y inputs

Usage

```
model_fit(x, y, method = "lm", LOOCV = FALSE, ..., size = 40, linout = TRUE)
```

Arguments

`x`	input data.frame of predictors (independ variables) the data from
`y`	input vector of observed data
`method`	the model to use the data from. The options are "lm" for linear regression, "rf" for random forest, "nnt" for neural network, "svm" for support vector machine and knn.* for k-nearest neighbors (see `yaImpute::yai()` for * methods). The default is "lm".
`LOOCV`	`logical`, default FALSE. Flag do determine if we should compute leave-one-out cross validation.
`...`	Other parameters to pass to the model
`size`	`integer`, default 40, used for neural network number of neurons only
`linout`	`logical`, default TRUE. Used only for neural network as the activation function for the neuron to be linear instead of logistic.

Value

A list containing the method, model and cross validation data if `LOOCV` = `TRUE`.

`plot,icesat2.atl03atl08_dt,character-method` *Plot photons from ATL03 and ATL08 joined products*

Description

This function plots photons along track

This function plots photons along track

This function plots photons along track

This function plots photons along track

Usage

```
## S4 method for signature 'icesat2.atl03atl08_dt,character'
plot(
  x,
  y = "h_ph",
  beam = NULL,
  colors = c("gray", "goldenrod", "forestgreen", "green"),
  legend = "topleft",
  ...
)

## S4 method for signature 'icesat2.atl03atl08_dt,missing'
plot(x, y, ...)

## S4 method for signature 'icesat2.atl08_dt,character'
plot(x, y, beam = "gt1l", col = "gray", xlim = NULL, ylim = NULL, ...)

## S4 method for signature 'icesat2.atl03_dt,character'
plot(x, y, col = "gray", ...)
```

Arguments

`x`	An object of class `icesat2.atl03_dt`
`y`	The attribute name for y axis
`beam`	Character vector indicating only one beam to process ("gt1l", "gt1r", "gt2l", "gt2r", "gt3l", "gt3r"). Default is "gt1r"
`colors`	A vector containing colors for plotting noise, terrain, vegetation and top canopy photons (e.g. c("gray", "#bd8421", "forestgreen", "green")
`legend`	the position of the legend. 'bottomleft', 'bottomright', 'topleft', 'topright' or FALSE to omit
`...`	will be passed to the main plot
`col`	Color for plotting the photons. Default is "gray"
`xlim`	The x limits to use for the plot
`ylim`	the y limits to use for the plot

Value

No return value

No return value

No return value

No return value

Examples

```
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)

plot(
  atl03_atl08_dt, "ph_h",
  colors = c("gray", "#bd8421", "forestgreen", "green"),
  pch = 16, cex = 0.5
)

close(atl03_h5)
close(atl08_h5)
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Specifying the path to ATL08 file
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting ATL03 and ATL08 photons and heights
atl03_atl08_dt <- ATL03_ATL08_photons_attributes_dt_join(atl03_h5, atl08_h5)

plot(
  atl03_atl08_dt,
  colors = c("gray", "#bd8421", "forestgreen", "green"),
  pch = 16, cex = 0.5
)

close(atl03_h5)
close(atl08_h5)
atl08_path <- system.file("extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL08 data (h5 file)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)

# Extracting atl08 and ATL08 photons and heights
atl08_seg_dt <- ATL08_seg_attributes_dt(atl08_h5 = atl08_h5)

plot(
  atl08_seg_dt,
  "h_canopy",
  beam = "gt1r",
  col = "gray"
)

close(atl08_h5)
# Specifying the path to ATL03 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting atl03 and atl03 photons and heights
atl03_photons_dt <- ATL03_seg_attributes_dt(
  atl03_h5 = atl03_h5,
 attributes = c("reference_photon_lon", "reference_photon_lat", "segment_dist_x", "h_ph")
)

plot(
  atl03_photons_dt,
  "h_ph",
  col = "gray",
  pch = 16,
  cex = 0.5
)

close(atl03_h5)
```

`predict.ee.Classifier` `predict` *method for generating predictions for the ee.Classifier.*

Description

`predict` method for generating predictions for the ee.Classifier.

Usage

```
## S3 method for class 'ee.Classifier'
predict(object, data, ...)
```

Arguments

`object`	The ee.Classifier generated either by using the regular ee API functions or using the package's `build_ee_forest()` function.
`data`	`data.frame` or `ee.FeatureCollection` with the features input to be used for the prediction.
`...`	Unused parameter, just to match the generic signature.

Value

A `numeric` with the predicted values

`predict.yai` *Prediction method wrapper for yai*

Description

Prediction method wrapper for yai

Usage

```
## S3 method for class 'yai'
predict(object, ...)
```

Arguments

`object`	the model of class `yai` to predict on
`...`	the first parameter only will be used as the x parameter. If not supplied will be forwarded to defaul yaImpute predict function.

Value

either a numeric or data.frame with the predicted/imputed variables.

predict_h5 *Model prediction over data.tables using HDF5 file as output*

Description

Model prediction over a data.table from ATL03 or ATL08 data containing geolocation data. It can both append results to an existing HDF5 file or create a new file, allowing to incrementally add predictions to the file to avoid memory issues.

Usage

```
predict_h5(model, dt, output)
```

Arguments

`model`	The trained model object
`dt`	The input data.table to run the model
`output`	The output HDF5 file path

Value

An `icesat2.predict_h5`, which is an h5 file with latitude, longitude and prediction datasets.

Examples

```
atl03_path <- system.file(
  "extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

atl03_h5 <- ATL03_read(atl03_path = atl03_path)
atl03_seg_dt <- ATL03_seg_attributes_dt(atl03_h5)

linear_model <- stats::lm(h_ph ~ segment_ph_cnt, data = atl03_seg_dt)
output_h5 <- tempfile(fileext = ".h5")
predicted_h5 <- predict_h5(linear_model, atl03_seg_dt, output_h5)

# List datasets
predicted_h5$ls()$name

# See predicted values
head(predicted_h5[["prediction"]][])

# Close the file
close(predicted_h5)
```

predict_h5,ANY,icesat2.atl03_seg_dt,character-method

S4 method for predicting using HDF5 file as output

Description

This method is used to predict using a trained model and save the results in HDF5 file.

Usage

```
## S4 method for signature 'ANY,icesat2.atl03_seg_dt,character'
predict_h5(model, dt, output)
```

Arguments

model	The trained model object
dt	The input data.table to run the model
output	The output file path

Value

An icesat2.predict_h5, which is an h5 file with latitude, longitude and prediction

Examples

```
atl03_path <- system.file(
  "extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

atl03_h5 <- ATL03_read(atl03_path = atl03_path)
atl03_seg_dt <- ATL03_seg_attributes_dt(atl03_h5)

linear_model <- stats::lm(h_ph ~ segment_ph_cnt, data = atl03_seg_dt)
output_h5 <- tempfile(fileext = ".h5")
predicted_h5 <- predict_h5(linear_model, atl03_seg_dt, output_h5)

# List datasets
predicted_h5$ls()$name

# See predicted values
head(predicted_h5[["prediction"]][])

# Close the file
close(predicted_h5)
```

predict_h5,ANY,icesat2.atl08_dt,character-method

S4 method for predicting using HDF5 file as output

Description

This method is used to predict using a trained model and save the results in HDF5 file.

Usage

```
## S4 method for signature 'ANY,icesat2.atl08_dt,character'
predict_h5(model, dt, output)
```

Arguments

model	The trained model object
dt	The input data.table to run the model
output	The output file path

Details

This method is used to predict using a trained model and save the results in an HDF5 file.

Value

An icesat2.predict_h5, which is an h5 file with latitude, longitude and prediction datasets.

Examples

```
atl08_path <- system.file(
  "extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)
atl08_h5 <- ATL08_read(atl08_path = atl08_path)
atl08_dt <- ATL08_seg_attributes_dt(atl08_h5)
linear_model <- stats::lm(h_canopy ~ canopy_openness, data = atl08_dt)
output_h5 <- tempfile(fileext = ".h5")
predicted_h5 <- predict_h5(linear_model, atl08_dt, output_h5)

# List datasets
predicted_h5$ls()$name

# See predicted values
head(predicted_h5[["prediction"]][])

# Close the file
close(predicted_h5)
```

`prepend_class` *Prepend a class to an object's list of classes*

Description

Prepend a class to an object's list of classes

Usage

```
prepend_class(obj, className)
```

Arguments

`obj`	The object to which prepend the class.
`className`	`character` with the name of the class to prepend.

Value

Nothing, it replaces the class attribute in place

`randomSampling` *Pure random sampling method*

Description

Pure random sampling method

Usage

```
randomSampling(size)
```

Arguments

`size`	the sample size. Either an integer of absolute number of samples or if it is between (0, 1) it will sample a percentage relative to the number of available observations within the group.

Value

A `icesat2_sampling_method` for defining the method used in `sample()`

`rasterize_h5` *Rasterizes the model prediction saved in the HDF5 file*

Description

This is used after running the prediction using `predict_h5()` function to rasterize and aggregate the prediction within raster cells. By default it will calculate (n, mean, variance * (n - 1), min, max, sd) in a single raster file with those 6 bands in that order. You can modify this behavior by changing the `agg_function`, `agg_join` and `finalizer` parameters, see details section.

Usage

```
rasterize_h5(h5_input, output, bbox, res, ...)
```

Arguments

`h5_input`	The input HDF5 file path
`output`	The output raster file path
`bbox`	The bounding box of the raster
`res`	The resolution of the raster
`...`	Additional parameters (see details section)

Details

This function will create five different aggregate statistics (n, mean, variance, min, max).

Within `...` additional parameters we can use:

`agg_function`: is a formula which return a data.table with the aggregate function to perform over the data. The default is:

```
~data.table(
    n = length(x),
    mean = mean(x,na.rm = TRUE),
    variance = var(x) * (length(x) - 1),
    min = min(x, na.rm=T),
    max = max(x, na.rm=T)
  )
```

`agg_join`: is a function to merge two data.table aggregates from the `agg_function`. Since the h5 files will be aggregated in chunks to avoid memory overflow, the statistics from the different chunks should have a function to merge them.

The default function is:

```
function(x1, x2) {
    combined = data.table()
    x1$n[is.na(x1$n)] = 0
    x1$mean[is.na(x1$mean)] = 0
    x1$variance[is.na(x1$variance)] = 0
    x1$max[is.na(x1$max)] = -Inf
    x1$min[is.na(x1$min)] = Inf

    combined$n = x1$n + x2$n

    delta = x2$mean - x1$mean
    delta2 = delta * delta

    combined$mean = (x1$n * x1$mean + x2$n * x2$mean) / combined$n
    combined$variance = x1$variance + x2$variance +
      delta2 * x1$n * x2$n / combined$n

    combined$min = pmin(x1$min, x2$min, na.rm=F)
    combined$max = pmax(x1$max, x2$max, na.rm=F)
    return(combined)
}
```

`finalizer`: is a list of formulas to generate the final rasters based on the intermediate statistics from the previous functions. The default `finalizer` will calculate the `sd`, based on the `variance` and `n` values. It is defined as:

```
list(
  sd = ~sqrt(variance/(n - 1)),
)
```

Examples

```
atl08_path <- system.file(
  "extdata",
  "atl08_clip.h5",
  package = "ICESat2VegR"
)

atl08_h5 <- ATL08_read(atl08_path = atl08_path)
atl08_dt <- ATL08_seg_attributes_dt(atl08_h5)

xmin <- min(atl08_dt$longitude)
xmax <- max(atl08_dt$longitude)
ymin <- min(atl08_dt$latitude)
ymax <- max(atl08_dt$latitude)

linear_model <- stats::lm(h_canopy ~ canopy_openness, data = atl08_dt)
output_h5 <- tempfile(fileext = ".h5")
predicted_h5 <- predict_h5(linear_model, atl08_dt, output_h5)
output_raster <- tempfile(fileext = ".tif")
```

```
rasterize_h5(
  predicted_h5,
  output = output_raster,
  bbox = terra::ext(xmin, xmax, ymin, ymax),
  res = 0.003
)
```

---

rasterize_h5,icesat2.predict_h5,character,SpatExtent,numeric-method

Rasterizes the model prediction saved in the HDF5 file

Description

Rasterizes the model prediction saved in the HDF5 file

Usage

```
## S4 method for signature 'icesat2.predict_h5,character,SpatExtent,numeric'
rasterize_h5(
  h5_input,
  output,
  bbox,
  res,
  chunk_size = 512 * 512,
  agg_function = agg_function_default,
  agg_join = agg_join_default,
  finalizer = finalizer_default
)
```

Arguments

`h5_input`	The input HDF5 file path
`output`	The output raster file path
`bbox`	The bounding box of the raster
`res`	The resolution of the raster
`chunk_size`	The chunk size to read the HDF5 file
`agg_function`	The function to aggregate the data
`agg_join`	The function to join the aggregated data
`finalizer`	The function to finalize the raster

`rasterSampling` *Get observations given a minimum radius distance between samples*

Description

Get observations given a minimum radius distance between samples

Usage

```
rasterSampling(size, raster, chainSampling = NULL)
```

Arguments

`size`	the sample size. Either an integer of absolute number of samples or if it is between (0, 1) it will sample a percentage relative to the number of available observations within the group.
`raster`	a `terra::SpatRaster` object opened with `terra::rast()`
`chainSampling`	chains different methods of sampling by providing the result of another samplingMethod `randomSampling()`, `spacedSampling()`, `stratifiedSampling()`.

Value

A `icesat2_sampling_method` for defining the method used in `sample()`

`rbindlist2` *Wraps around* `data.table::rbindlist()` *function*

Description

Wraps around `data.table::rbindlist()` function

Usage

```
rbindlist2(l, ...)
```

Arguments

`l`	A list containing data.table, data.frame or list objects. ... is the same but you pass the objects by name separately.
`...`	pass directly to `data.table::rbindlist()`

Value

The data.table with the same class as the input

`rgt_extract` *Extract reference ground track from ATL03 segments*

Description

This function extracts reference ground track from ICESat-2 ATL03 data and writes it as a GDAL vector format.

Usage

```
rgt_extract(h5)
```

Arguments

`h5`	A ICESat-2 ATL03 object (output of `ATL03_read()` or `ATL08_read()` function). An S4 object of class `icesat2.atl03_dt` or `icesat2.atl08_dt`.

Details

This function will use the reference photons from the segments as reference for deriving the ground tracks. The begining and end of the lines are interpolated from the information regarding the position of the reference photon within the segment and the segment length.

Value

Returns the ground track boundaries as `terra::SpatVector` extracted from "orbit_info", along with other orbit information.

See Also

https://icesat-2.gsfc.nasa.gov/sites/default/files/page_files/ICESat2_ATL03_ATBD_r006.pdf

Examples

```
# Specifying the path to ATL03 H5 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)

# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)

# Extracting ATL03 photons attributes
rgt <- rgt_extract(h5 = atl03_h5)
head(rgt)

terra::plet(rgt)
```

```
close(atl03_h5)
```

---

sample — *Sample method for applying multiple sampling methods*

Description

Sample method for applying multiple sampling methods

Usage

```
sample(x, ..., method)
```

Arguments

`x`	the generic input data to be sampled
`...`	generic params to pass ahead to the specific sampling function
`method`	the sampling method to use.

Details

It is expected that the user pass a `method` parameter within `...`

See Also

`randomSampling()`, `spacedSampling()`, `gridSampling()`, `stratifiedSampling()`, `geomSampling()`, `rasterSampling()`

---

search_datasets — *Search for Google Earth Engine datasets*

Description

Search for Google Earth Engine datasets

Usage

```
search_datasets(
  ...,
  title = TRUE,
  description = TRUE,
  operator = "and",
  refresh_cache = FALSE
)
```

Arguments

`...`	character arguments to search for within title and/or description
`title`	logical. Whether should search within the title, default TRUE.
`description`	logical. Whether should search within the description of the dataset, default TRUE.
`operator`	character. Should be either "OR" or "AND" to tell if the search needs to include all the queries "AND" or any of the queries "OR". Default "AND".
`refresh_cache`	flag indicating if the results cache should be refreshed, default FALSE.

Value

A `data.table` containing the id, title and description of the datasets that matched the supplied query ordered by relevance.

`seg_gee_ancillary_dt_extract`

Given a stack image raster from GEE retrieve the point geometry with values for the images

---

Description

Given a stack image raster from GEE retrieve the point geometry with values for the images

Usage

```
seg_gee_ancillary_dt_extract(stack, geom, scale = 30, chunk_size = 1000)
```

Arguments

`stack`	A single image or a vector/list of images from Earth Engine.
`geom`	A geometry from `terra::SpatVector` read with `terra::vect`.
`scale`	The scale in meters for the extraction (image resolution).
`chunk_size`	If the number of observations is greater than 1000, it is recommended to chunk the results for not running out of memory within GEE server, default is chunk by 1000.

Value

A data.table::data.table with the properties extracted from the ee images.

slope *Calculates slope in degrees from a terrain DEM.*

Description

The local gradient is computed using the 4-connected neighbors of each pixel, so missing values will occur around the edges of an image.

Usage

```
slope(x)
```

Arguments

`x`	The `ee.Image` on which to apply the aspect function.

Value

An `ee.Image` with a single band named "Slope".

spacedSampling *Get observations given a minimum radius distance between samples*

Description

Get observations given a minimum radius distance between samples

Usage

```
spacedSampling(size, radius, spatialIndex = NULL, chainSampling = NULL)
```

Arguments

`size`	the sample size. Either an integer of absolute number of samples or if it is between (0, 1) it will sample a percentage relative to the number of available observations within the group.
`radius`	the minimum radius between samples.
`spatialIndex`	optional parameter. You can create a spatial index for accelerating the search space with `ANNIndex` and reuse that if needed elsewhere. It will create one automatically everytime if you don't specify one.
`chainSampling`	chains different methods of sampling by providing the result of another samplingMethod `randomSampling()`, `spacedSampling()`, `stratifiedSampling()`.

Value

A `icesat2_sampling_method` for defining the method used in `sample()`

stats_model *Model fit stats*

Description

Computes absolute and relative root-mean-square error (RMSE) and bias, and adjusted coefficient of determination from a linear relationship between predicted and observed data

Usage

```
stats_model(
  observed,
  predicted,
  plotstat = TRUE,
  legend = "topleft",
  unit = " m",
  ...
)
```

Arguments

`observed`	A vector containing observed data
`predicted`	A vector containing predicted data
`plotstat`	if TRUE, a plot showing the 1:1 figure will be displayed. Default is TRUE
`legend`	Character for legend position. Default is "topleft"
`unit`	Character indicating the unit for the observed and predicted data (e.g."Mg/ha")
`...`	Other params to be redirected to the plot.

Value

Returns an data.frame object containing the list of stats

Examples

```
# Observed and predicted aboveground biomass
observed <- c(178, 33, 60, 80, 104, 204, 146)
predicted <- c(184, 28.5, 55, 85, 105, 210, 155)

stats_model(observed, predicted,
  plotstat = TRUE, legend = "topleft", unit = "Mg/ha",
  xlab = "Obserbed AGBD (Mg/ha)",
  ylab = "Predicted AGBD (Mg/ha)", pch = 16
)
```

`stratifiedSampling` *Get samples stratified by a variable binning histogram*

Description

Get samples stratified by a variable binning histogram

Usage

```
stratifiedSampling(size, variable, chainSampling = NULL, ...)
```

Arguments

`size`	the sample size. Either an integer of absolute number of samples or if it is between (0, 1) it will sample a percentage relative to the number of available observations within the group.
`variable`	Variable name used for the stratification
`chainSampling`	chains different methods of sampling by providing the result of another samplingMethod `randomSampling()`, `spacedSampling()`, `stratifiedSampling()`.
`...`	forward to the `graphics::hist()`, where you can manually define the breaks.

Value

A `icesat2_sampling_method` for defining the method used in `sample()`

`to_vect` *Generic function to export icesat2 classes to* `terra::SpatVector`

Description

Generic function to export icesat2 classes to `terra::SpatVector`

Usage

```
to_vect(x, ...)
```

Arguments

`x`	The icesat2 object to convert to `terra::SpatVector`
`...`	other potential parameters needed.

Value

The icesat2 object as the appropriate `terra::SpatVector`

Examples

```
# Get path to ATL03 h5 file
atl03_path <- system.file("extdata",
  "atl03_clip.h5",
  package = "ICESat2VegR"
)
# Reading ATL03 data (h5 file)
atl03_h5 <- ATL03_read(atl03_path = atl03_path)
# Extracting ATL03 segment attributes
atl03_segment_dt <- ATL03_seg_attributes_dt(atl03_h5 = atl03_h5)

# Extract vector
atl03_segment_vect <- to_vect(atl03_segment_dt)

# Terra plot
terra::plot(atl03_segment_vect, col = atl03_segment_vect$segment_ph_cnt)

# Export as temp gpkg
temp_vect <- tempfile(fileext = ".gpkg")
terra::writeVector(atl03_segment_vect, temp_vect)

head(atl03_segment_dt)
close(atl03_h5)
```

`to_vect,icesat2.atl03atl08_dt-method`
Convert ICESat-2 data to terra::SpatVector

Description

This method converts data from various ICESat-2 data types to a `terra::SpatVector-class` object, facilitating geographic operations and visualizations within the terra package.

Usage

```
## S4 method for signature 'icesat2.atl03atl08_dt'
to_vect(x, ...)
```

Arguments

`x`	ICESat-2 data object to be converted.
`...`	Additional parameters for the method.

Value

`terra::SpatVector` object representing the input data.

`to_vect,icesat2.atl03_atl08_seg_dt-method` *Convert ICESat-2 data to terra::SpatVector*

Description

This method converts data from various ICESat-2 data types to a `terra::SpatVector-class` object, facilitating geographic operations and visualizations within the terra package.

Usage

```
## S4 method for signature 'icesat2.atl03_atl08_seg_dt'
to_vect(x, ...)
```

Arguments

`x`	ICESat-2 data object to be converted.
`...`	Additional parameters for the method.

Value

`terra::SpatVector` object representing the input data.

`to_vect,icesat2.atl03_dt-method` *Convert ICESat-2 data to terra::SpatVector*

Description

This method converts data from various ICESat-2 data types to a `terra::SpatVector-class` object, facilitating geographic operations and visualizations within the terra package.

Usage

```
## S4 method for signature 'icesat2.atl03_dt'
to_vect(x, ...)
```

Arguments

`x`	ICESat-2 data object to be converted.
`...`	Additional parameters for the method.

Value

`terra::SpatVector` object representing the input data.

`to_vect,icesat2.atl03_seg_dt-method` *Convert ICESat-2 data to terra::SpatVector*

Description

This method converts data from various ICESat-2 data types to a `terra::SpatVector-class` object, facilitating geographic operations and visualizations within the terra package.

Usage

```
## S4 method for signature 'icesat2.atl03_seg_dt'
to_vect(x, ...)
```

Arguments

`x`	ICESat-2 data object to be converted.
`...`	Additional parameters for the method.

Value

`terra::SpatVector` object representing the input data.

`to_vect,icesat2.atl08_dt-method` *Convert ICESat-2 data to terra::SpatVector*

Description

This method converts data from various ICESat-2 data types to a `terra::SpatVector-class` object, facilitating geographic operations and visualizations within the terra package.

Usage

```
## S4 method for signature 'icesat2.atl08_dt'
to_vect(x, ...)
```

Arguments

`x`	ICESat-2 data object to be converted.
`...`	Additional parameters for the method.

Value

`terra::SpatVector` object representing the input data.

[,icesat2.granules_cloud,ANY,ANY,ANY-method
Subset Granules

Description

This method subsets the granules slot of an icesat2.granules_cloud object.

Usage

```
## S4 method for signature 'icesat2.granules_cloud,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]
```

Arguments

x	An object of class icesat2.granules_cloud.
i	Subset index.
j	Unused, just to match the generic signature.
...	Additional arguments (not used).
drop	Unused, just to match the generic signature.

Value

An object of class icesat2.granule_cloud.

[[,icesat2.granules_cloud,ANY,ANY-method
Extract Granules

Description

This method extracts a single element from the granules slot of an icesat2.granules_cloud object.

Usage

```
## S4 method for signature 'icesat2.granules_cloud,ANY,ANY'
x[[i, j, ...]]
```

Arguments

x	An object of class icesat2.granules_cloud.
i	Extraction index.
j	Unused, just to match the generic signature.
...	Additional arguments (not used).

Value

An object of class `icesat2.granule_cloud`.

Examples

```
## Not run:
granule <- new("icesat2.granules_cloud")
extracted_granule <- granule[[1]]

## End(Not run)
```

`[[.GDALDataset` *GDALDataset [[]] accessor*

Description

This function gives access to the GDALRasterBand using [[i]], where i is the band index to return.

Usage

```
## S3 method for class 'GDALDataset'
x[[slice]]
```

Arguments

`x`	GDALDatset. Automatically obtained from GDALDataset[[]] call.
`slice`	Integer. The index for the band to access.

Value

An object of GDALRasterBand R6 class.

`[[.GDALRasterBand` *GDALRasterBand [[]] getter*

Description

This function gives access to the GDALRasterBand using [[i]], where i is the band index to return.

Usage

```
## S3 method for class 'GDALRasterBand'
x[[blockX, blockY]]
```

Arguments

`x`	GDALRasterBand. Automatically obtained from GDALDataset[[]] call.
`blockX`	Integer. The x index for block to access.
`blockY`	Integer. The y index for block to access.

Value

Nothing, this is a setter

`[[<-.GDALRasterBand` *GDALRasterBand [[]]= setter*

Description

This function gives access to the GDALRasterBand using [[i]], where i is the band index to return.

Usage

```
## S3 replacement method for class 'GDALRasterBand'
x[[blockX, blockY]] <- value
```

Arguments

`x`	GDALRasterBand. Automatically obtained from GDALDataset[[]] call.
`blockX`	Integer. The x index for block to access.
`blockY`	Integer. The y index for block to access.
`value`	Integer. The value buffer to write

Value

Nothing, this is a setter

Index