Package 'ForestTools'

October 12, 2022

Type Package

Title Analyzing Remotely Sensed Forest Data

Version 0.2.5

Date 2021-09-09

Author Andrew Plowright, Jean-Romain Roussel

Maintainer Andrew Plowright <andrew.plowright@alumni.ubc.ca>

Description Provides tools for analyzing remotely sensed forest data, including functions for detecting treetops from canopy models, outlining tree crowns, calculating textural metrics and generating spatial statistics.

Depends R (>= 4.0)

License GPL (>= 3)

Encoding UTF-8

LazyData true

LinkingTo Rcpp

Imports APfun, doParallel, foreach, imager, methods, parallel, plyr, progress, raster, Rcpp, rgeos, sp

Suggests testthat, knitr, rmarkdown

RoxygenNote 7.1.1

URL https://github.com/andrew-plowright/ForestTools

BugReports https://github.com/andrew-plowright/ForestTools/issues

VignetteBuilder knitr

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-09-11 04:50:02 UTC

R topics documented:

`.calcGLCM` *Calculate GLCM*

Description

Some notes about this function: 1. Input should be a matrix 2. Shouldn't receive negative values 3. Shouldn't receive all NA values 4. Shouldn't be an empty matrix (i.e.: nrow = 0, ncol = 0) 5. 'n_grey' shouldn't be larger than the number of unique values

Usage

```
.calcGLCM(data, n_grey, angle, d = 1, normalize = TRUE)
```

Arguments

`data`	matrix. Input image
`n_grey`	integer. Number of grey levels used to discretize image
`angle`	integer. Angle at which GLCM will be calculated. Valid inputs are 0, 45, 90, or 135
`d`	numeric. Distance for calculating GLCM
`normalize`	boolean. Normalize output if TRUE

.GLCMstats *Calculate stats for GLCM*

Description

Calculate stats for GLCM

Usage

```
.GLCMstats(data)
```

Arguments

data	matrix. GLCM computed using '.calcGLCM'

CHMdemo *Canopy height model demo*

Description

A small section of a canopy height model of a forest in British Columbia, Canada.

Usage

```
CHMdemo
```

Format

A RasterLayer

Cell values are equal to canopy height above ground (in meters)

ForestTools *Forest Tools: A package for analyzing geospatial forest data*

Description

Forest Tools provides functions for analyzing remotely sensed forest data. Functions like vwf and mcws are applied to rasterized canopy height models, and can detect treetops and outline their respective crowns. sp_summarise can summarize tree counts and attributes within particular areas of interest or within continuous spatial grids.

glcm *Grey-Level Co-Occurrence Matrix*

Description

Generate textural metrics for a segmented raster using Grey-Level Co-Occurrence Matrices (GLCM). It will return a series of GLCM statistics for each segment (`segs`) based on an underlying single-band raster image (`image`) in the form of a data.frame.

Usage

```
glcm(
  segs,
  image,
  n_grey = 32,
  angle = 0,
  clusters = 1,
  showprog = FALSE,
  roundCoords = 4
)
```

Arguments

`segs`	RasterLayer. A segmented raster. Cell values should be equal to segment numbers
`image`	RasterLayer. A single-band raster layer from which texture is measured
`n_grey`	integer. Number of grey levels the image should be quantized into
`angle`	integer. Angle at which GLCM will be calculated. Valid inputs are 0, 45, 90, or 135
`clusters`	integer. Number of clusters to use during parallel processing
`showprog`	logical. Display progress in terminal
`roundCoords`	integer. Errors in coordinate precision can trigger errors in this function. Internally, the coordinates are rounded to this decimal place. Default value of 4 decimals.

Details

The underlying C++ code for computing GLCMs and their statistics was originally written by Joel Carlson for the defunct [radiomics](https://github.com/cran/radiomics) library. It has been reused here with permission from the author.

Value

data.frame

References

Parmar, C., Velazquez, E.R., Leijenaar, R., Jermoumi, M., Carvalho, S., Mak, R.H., Mitra, S., Shankar, B.U., Kikinis, R., Haibe-Kains, B. and Lambin, P. (2014). *Robust radiomics feature quantification using semiautomatic volumetric segmentation. PloS one, 9*(7)

Examples

```
## Not run:
# Generate raster segments
segs <- mcws(kootenayTrees, kootenayCHM, minHeight = 0.2, format = "raster")

# Get textural metrics for ortho's red band
tex <- glcm(segs, kootenayOrtho[[1]])

## End(Not run)
```

`glcm0` *Create a 0 degree GLCM*

Description

Used internally by glcm()

Usage

```
glcm0(x, n_grey, d)
```

Arguments

`x`	A Numeric matrix, integer values only
`n_grey`	Number of grey levels
`d`	distance from reference pixel to neighbour pixel

`glcm135` *Create a 135 degree GLCM*

Description

Used internally by glcm()

Usage

```
glcm135(x, n_grey, d)
```

Arguments

x	A Numeric matrix, integer values only
n_grey	Number of grey levels
d	distance from reference pixel to neighbour pixel

---

glcm45 *Create a 45 degree GLCM*

Description

Used internally by glcm()

Usage

```
glcm45(x, n_grey, d)
```

Arguments

x	A Numeric matrix, integer values only
n_grey	Number of grey levels
d	distance from reference pixel to neighbour pixel

---

glcm90 *Create a 90 degree GLCM*

Description

Used internally by glcm()

Usage

```
glcm90(x, n_grey, d)
```

Arguments

x	A Numeric matrix, integer values only
n_grey	Number of grey levels
d	distance from reference pixel to neighbour pixel

glcm_features *GLCM Features*

Description

GLCM Features

Usage

```
glcm_mean(glcm)

glcm_variance(glcm)

glcm_autoCorrelation(glcm)

glcm_cProminence(glcm)

glcm_cShade(glcm)

glcm_cTendency(glcm)

glcm_contrast(glcm)

glcm_correlation(glcm)

glcm_differenceEntropy(glcm, base = 2)

glcm_dissimilarity(glcm)

glcm_energy(glcm)

glcm_entropy(glcm, base = 2)

glcm_homogeneity1(glcm)

glcm_homogeneity2(glcm)

glcm_IDMN(glcm)

glcm_IDN(glcm)

glcm_inverseVariance(glcm)

glcm_maxProb(glcm)

glcm_sumAverage(glcm)
```

```
glcm_sumEntropy(glcm, base = 2)

glcm_sumVariance(glcm)
```

Arguments

glcm	A matrix of class "glcm" produced by glcm.
base	Base of the logarithm in differenceEntropy.

Functions

- glcm_mean: Mean
- glcm_variance: Variance
- glcm_autoCorrelation: Autocorrelation
- glcm_cProminence: Cluster Prominence
- glcm_cShade: Cluster Shade
- glcm_cTendency: Cluster Tendency
- glcm_contrast: Contrast
- glcm_correlation: Correlation
- glcm_differenceEntropy: Difference Entropy
- glcm_dissimilarity: Dissimilarity
- glcm_energy: Energy
- glcm_entropy: Entropy
- glcm_homogeneity1: Homogeneity
- glcm_homogeneity2: Homogeneity 2
- glcm_IDMN: Inverse Difference Moment (Normalized)
- glcm_IDN: Inverse Difference (Normalized)
- glcm_inverseVariance: Inverse Variance
- glcm_maxProb: Maximum Probability
- glcm_sumAverage: Sum Average
- glcm_sumEntropy: Sum Entropy
- glcm_sumVariance: Sum Variance

References

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0102107

glcm_img *Get GLCM statistics for a single unsegmented image*

Description

Get GLCM statistics for a single unsegmented image

Usage

```
glcm_img(img, n_grey = 32, angle = 0, d = 1, normalize = TRUE)
```

Arguments

img	matrix or raster. Input image
n_grey	integer. Number of grey levels used to discretize image
angle	integer. Angle at which GLCM will be calculated. Valid inputs are 0, 45, 90, or 135
d	numeric. Distance for calculating GLCM
normalize	boolean. Normalize output image before calculating statistics

kootenayBlocks *Kootenay forest - Cut blocks*

Description

Boundaries of cut blocks within a 1.5 hectare section of forest in the Kootenay mountains, in British Columbia, Canada. Each block contains trees of different levels of maturity. Overlaps with kootenayTrees, kootenayCrowns, kootenayOrtho and kootenayCHM.

Usage

```
kootenayBlocks
```

Format

A SpatialPolygonsDataFrame with the following attributes:

BlockID numerical identifier for each block

Shape_Leng length of polygon on meters

Shape_Area area of polygon in square meters

See Also

kootenayTrees kootenayCHM kootenayCrowns kootenayOrtho

kootenayCHM *Kootenay forest - Canopy height model*

Description

A canopy height model of a 1.5 hectare section of forest in the Kootenay mountains, in British Columbia, Canada.

Usage

```
kootenayCHM
```

Format

A RasterLayer

Cell values are equal to canopy height above ground (in meters)

Source

Data acquired from a photogrammetric drone survey performed by Spire Aerobotics on June 16th, 2016.

See Also

kootenayTrees kootenayBlocks kootenayCrowns kootenayOrtho

kootenayCrowns *Kootenay forest - Tree crowns*

Description

Outlines of tree crowns corresponding to the kootenayTrees treetops. Generated using mcws.

Usage

```
kootenayCrowns
```

Format

A SpatialPolygonsDataFrame with the following attributes:

height height of the tree's apex, in meters above ground. Inherited from kootenayTrees.

winRadius radius of the moving window at the treetop's location. Inherited from kootenayTrees.

crownArea area of crown outline in square meters

See Also

kootenayTrees kootenayCHM kootenayBlocks kootenayOrtho

kootenayOrtho *Kootenay forest - Orthomosaic*

Description

An orthomosaic of a 1.5 hectare section of forest in the Kootenay mountains, in British Columbia, Canada.

Usage

```
kootenayOrtho
```

Format

A RasterLayer

Cell values are equal to canopy height above ground (in meters)

Source

Data acquired from a photogrammetric drone survey performed by Spire Aerobotics on June 16th, 2016.

See Also

kootenayTrees kootenayBlocks kootenayCrowns kootenayCHM

kootenayTrees *Kootenay forest - Dominant trees over 2 m*

Description

Dominant trees from a 1.5 hectare section of forest in the Kootenay mountains, in British Columbia, Canada. Trees were detected by applying the vwf function to the kootenayCHM raster dataset. Only trees over 2 m above ground were detected.

Usage

```
kootenayTrees
```

Format

A SpatialPointsDataFrame with the following attributes:

height height of the tree's apex, in meters above ground

winRadius radius of the moving window (see vwf) at the treetop's location

See Also

kootenayCHM kootenayBlocks kootenayCrowns kootenayOrtho

`mcws` *Marker-Controlled Watershed Segmentation*

Description

Implements the watershed function to segment (i.e.: outline) crowns from a canopy height model. Segmentation is guided by the point locations of treetops, typically detected using the vwf function. See Meyer & Beucher (1990) for details on watershed segmentation.

Usage

```
mcws(
  treetops,
  CHM,
  minHeight = 0,
  format = "raster",
  OSGeoPath = NULL,
  verbose = FALSE
)
```

Arguments

`treetops`	SpatialPointsDataFrame. The point locations of treetops. The function will generally produce a number of crown segments equal to the number of treetops.
`CHM`	Canopy height model in raster format. Should be the same that was used to create the input for `treetops`.
`minHeight`	numeric. The minimum height value for a `CHM` pixel to be considered as part of a crown segment. All `CHM` pixels beneath this value will be masked out. Note that this value should be lower than the minimum height of `treetops`.
`format`	string. Format of the function's output. Can be set to either 'raster' or 'polygons'.
`OSGeoPath`	character. Optional path to the OSGeo4W installation directory. If both OSGeo4W and Python are installed, this will enable the function to use a faster algorithm for producing polygonal crown outlines (see Details below).
`verbose`	logical. Print processing progress to console.

Details

This function can return a crown map as either a raster or a SpatialPolygonsDataFrame, as defined using the `format` argument. For most analytical purposes, it is preferable to have crown outlines as polygons. However, polygonal crown maps take up significantly more disk space, and take longer to process. It is advisable to run this function using a raster output first, in order to check its results and adjust parameters.

Using the polygons provides the added benefit of transferring treetop attributes (such as *height*) to the newly created polygons. The area of each crown will also automatically be calculated and added to the polygons' data under the *crownArea* field. Furthermore, "orphaned" segments (i.e.: outlines without an associated treetop) will be removed when `format` is set to 'polygons'.

By default, polygonal crown outlines are produced internally using the the `rasterToPolygons` function from the raster package. This function is problematic due to it being 1) very slow and 2) leaking memory when applied to multiple datasets. An alternative is provided for users who've installed OSGeo4W and Python. By setting the `OSGeoPath` path to the OSGeo4W installation directory (usually 'C:\OSGeo4W64'), the function will use the *gdal_polygonize.py* GDAL utility to generate polygonal crown outlines instead.

Value

Depending on the argument set with `format`, this function will return a map of outlined crowns as either a RasterLayer (see raster), in which distinct crowns are given a unique cell value, or a SpatialPolygonsDataFrame, in which each crown is represented by a polygon.

References

Meyer, F., & Beucher, S. (1990). Morphological segmentation. *Journal of visual communication and image representation, 1*(1), 21-46.

See Also

`vwf` `sp_summarise` `watershed`

OSGeo4W download page: https://trac.osgeo.org/osgeo4w/

Examples

```
## Not run:
# Use variable window filter to detect treetops in demo canopy height model
ttops <- vwf(CHMdemo, winFun = function(x){x * 0.06 + 0.5}, minHeight = 2)

# Set minimum tree crown height (should be LOWER than minimum treetop height)
minCrwnHgt <- 1

# Use 'mcws' to outline tree crowns
segs <- mcws(ttops, CHMdemo, minCrwnHgt)

## End(Not run)
```

quesnelBlocks *Quesnel forest - Cut blocks*

Description

Boundaries of cut blocks within a 125 hectare section of forest in the Quesnel Timber Supply Area, in British Columbia, Canada. Each block contains trees of different levels of maturity. Overlaps with quesnelTrees and quesnelCHM.

Usage

```
quesnelBlocks
```

Format

A SpatialPolygonsDataFrame with the following attributes:

BlockID numerical identifier for each block

Shape_Leng length of polygon on meters

Shape_Area area of polygon in square meters

See Also

quesnelTrees quesnelCHM

quesnelCHM *Quesnel forest - Canopy height model*

Description

A canopy height model of a 125 hectare section of forest in the Quesnel Timber Supply Area, in British Columbia, Canada.

Usage

```
quesnelCHM
```

Format

A RasterLayer

Cell values are equal to canopy height above ground (in meters)

Source

Data acquired from a photogrammetric drone survey performed by Spire Aerobotics on September 15th, 2016.

See Also

quesnelTrees quesnelBlocks

`quesnelTrees` *Quesnel forest - Dominant trees over 2 m*

Description

Dominant trees from a 125 hectare section of forest in the Quesnel Timber Supply Area, in British Columbia, Canada. Trees were detected by applying the `vwf` function to the quesnelCHM raster dataset. Only trees over 2 m above ground were detected.

Usage

```
quesnelTrees
```

Format

A SpatialPointsDataFrame with the following attributes:

height height of the tree's apex, in meters above ground

winRadius radius of the moving window (see `vwf`) at the treetop's location

See Also

quesnelCHM quesnelBlocks

`sp_summarise` *Spatial Summarization*

Description

Summarization tool for calculating tree counts and statistics within various spatial units.

Usage

```
sp_summarise(
  trees,
  areas = NULL,
  grid = NULL,
  variables = NULL,
  statFuns = NULL
)
```

Arguments

trees	SpatialPointsDataFrame or SpatialPolygonsDataFrame. The locations of a set of trees, typically detected from a canopy height model using vwf. Tree attributes, such as height or crown size, should be stored within this object's @data slot. Tree crowns delineated using mcws can also be used.
areas	SpatialPolygonsDataFrame. An optional set of polygons corresponding to areas of interest. Tree counts and statistics will be returned for each area.
grid	RasterLayer (see raster) or numeric. An alternative to the areas argument. Using grid will compute tree counts and statistics within the cells of a spatial grid. Grid size and placement can be defined by inputting a raster object. A single numeric value can also be used, in which case the function will generate a grid with a cell size equal to this value.
variables	character. The names of tree attribute variables (stored in the trees@data slot). In addition to tree counts, the function will compute statistics for each of these variables. Only numeric variables are accepted.
statFuns	list. A named list of custom functions that are used to compute tree attribute statistics. If none are provided, default statistics are mean, median, standard deviation, minimum and maximum. Note that each element of the list should have a name that describes the statistics generated by the function. See below for details on defining custom functions.

Details

Input trees can either be point locations (SpatialPointsDataFrame) or crown outlines (SpatialPolygonsDataFrame). If crown outlines (or other polygons) are inputted, they will be partitioned between spatial units according to their geographic centroids.

In addition to tree counts, statistics for the trees' attributes can also be generated. These attributes should be defined within the @data slot of the input. Only numeric variables are accepted.

By default, the statistics generated for each attribute will be its mean, median, standard deviation, minimum and maximum. However, custom functions can also be used with the statFuns argument. This should be a named list of functions, wherein each list element is given a name to represent the statistic computed by the function.

For example: `list(qunt98 = function(x, ...) quantile(x, c(.98), na.rm = TRUE))`

Furthermore, custom functions should:

- Be able to accept numeric vectors.
- Be able to handle NA values.
- Have an ellipsis (three dots) in their arguments: `function(x, ...)`
- Return a single numeric value.

Value

Tree count and, if any variables are supplied, tree attribute statistics. If no areas or grid is supplied, the tree count and statistics are computed for the entire trees dataset, and returned as a 'data.frame' object. If areas are defined, an identical SpatialPolygonsDataFrame will be returned,

with all computed statistics appended to the object's `@data` slot. If a `grid` is defined, tree count will be returned as a RasterLayer, with cell values equal to the number of trees in each cell. If a `grid` and `variables` are defined, a RasterBrick (see brick) will be returned instead, with tree count and attribute statistics stored as stacked layers.

See Also

`vwf` `mcws`

Examples

```
# Load sample data
library(ForestTools)
library(sp)
data("kootenayTrees", "kootenayBlocks", "kootenayCrowns")

# Get total tree count
sp_summarise(kootenayTrees)

# Get total tree count, tree height and crown area statistics
sp_summarise(kootenayCrowns, variables = c("height", "crownArea"))

# Get tree count, height statistics for specific areas of interest
areaStats <- sp_summarise(kootenayTrees, areas = kootenayBlocks, variables = "height")

# Plot according to tree count
plot(areaStats, col = heat.colors(3)[order(areaStats$TreeCount)])

# Get tree count and height statistics for a 20 x 20 m spatial grid
gridStats <- sp_summarise(kootenayTrees, grid = 20, variables = "height")

# Plot gridded tree count and statistics
plot(gridStats$TreeCount)
plot(gridStats$heightMax)
```

---

vwf *Variable Window Filter*

Description

Implements the variable window filter algorithm (Popescu & Wynne, 2004) for detecting treetops from a canopy height model.

Usage

```
vwf(
  CHM,
  winFun,
```

```
  minHeight = NULL,
  maxWinDiameter = 99,
  minWinNeib = "queen",
  verbose = FALSE
)
```

Arguments

CHM	Canopy height model. Either in raster format, or a path directing to a raster file. A character vector of multiple paths directing to a tiled raster dataset can also be used.
winFun	function. The function that determines the size of the window at any given location on the canopy. It should take the value of a given CHM pixel as its only argument, and return the desired *radius* of the circular search window when centered on that pixel. Size of the window is in map units.
minHeight	numeric. The minimum height value for a CHM pixel to be considered as a potential treetop. All CHM pixels beneath this value will be masked out.
maxWinDiameter	numeric. Sets a cap on the maximum window diameter (in cells). If an improperly calibrated function is set for winFun, it may produce overly large windows that would perform poorly and significantly slow processing time. This setting can be disabled by setting to NULL.
minWinNeib	character. Define whether the smallest possible search window (3x3) should use a queen or a rook neighborhood.
verbose	logical. Print progress to console if set to TRUE.

Details

This function uses the resolution of the raster to figure out how many cells the window needs to cover. This means that the raster value (representing height above ground) and the map unit (represented by the raster's resolution), need to be in the _same unit_. This can cause issues if the raster is in lat/long, whereby its resolution is in decimal degrees.

Value

SpatialPointsDataFrame. The point locations of detected treetops. The object contains two fields in its data table: *height* is the height of the tree, as extracted from the CHM, and *winRadius* is the radius of the search window when the treetop was detected. Note that *winRadius* does not necessarily correspond to the radius of the tree's crown.

References

Popescu, S. C., & Wynne, R. H. (2004). Seeing the trees in the forest. *Photogrammetric Engineering & Remote Sensing, 70*(5), 589-604.

See Also

mcws sp_summarise

Examples

```
# Set function for determining variable window radius
winFunction <- function(x){x * 0.06 + 0.5}

# Set minimum tree height (treetops below this height will not be detected)
minHgt <- 2

# Detect treetops in demo canopy height model
ttops <- vwf(CHMdemo, winFunction, minHgt)
```

Index