Package 'fgeo.analyze'

December 5, 2020

Title Analyze ForestGEO Data

Version 1.1.14

Description To help you access, transform, analyze, and visualize ForestGEO data, we developed a collection of R packages (<https://forestgeo.github.io/fgeo/>). This package, in particular, helps you to implement analyses of plot species distributions, topography, demography, and biomass. It also includes a torus translation test to determine habitat associations of tree species as described by Zuleta et al. (2018) <doi:10.1007/s11104-018-3878-0>. To learn more about ForestGEO visit <https://forestgeo.si.edu/>.

License GPL-3

URL https://github.com/forestgeo/fgeo.analyze

BugReports https://github.com/forestgeo/fgeo.analyze/issues

Depends R (>= 3.2)

Imports dplyr (>= 0.8.0.1), fgeo.tool (>= 1.2.4), glue (>= 1.3.1), graphics, lubridate (>= 1.7.4), magrittr (>= 1.5), MASS, purrr (>= 0.3.2), rlang (>= 0.3.4), stats, tibble (>= 2.1.1), tidyr (>= 0.8.3), withr (>= 2.1.2)

Suggests covr (>= 3.2.1), fgeo.plot (>= 1.1.8), fgeo.x (>= 1.1.3), ggplot2 (>= 3.1.1), knitr (>= 1.22), measurements (>= 1.3.0), readr (>= 1.3.1), rmarkdown (>= 1.12), spelling (>= 2.1), testthat (>= 2.1.1)

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Mauro Lepore [aut, ctr, cre] (<https://orcid.org/0000-0002-1986-7988>), Gabriel Arellano [aut, rev], Richard Condit [aut], Matteo Detto [aut],

Kyle Harms [aut],
Suzanne Lao [aut, rev],
KangMin Ngo [rev],
Haley Overstreet [rev],
Sabrina Russo [aut, rev],
Daniel Zuleta [aut, rev],
CTFS-ForestGEO [cph]

Maintainer Mauro Lepore <maurolepore@gmail.com>

Repository CRAN

Date/Publication 2020-12-05 05:40:12 UTC

R topics documented:

---

abundance *Abundance and basal area, optionally by groups.*

Description

- `abundance()` counts the number of rows in a dataset, optionally by groups created with `dplyr::group_by()` (similar to `dplyr::n()`). It warns if it detects duplicated values of treeid.
- `basal_area()` sums the basal area of all stems in a dataset, optionally by groups created with `group_by()`. It warns if it detects duplicated values of stemid. It does not convert units (but see examples).

Both `abundance()` and `basal_area()` warn if they detect multiple censusid and multiple plots.

Usage

```
abundance(data)

basal_area(data)
```

Arguments

`data`	A dataframe. `basal_area()` requires a column named dbh (case insensitive).

Details

You may want to calculate the abundance or basal area for a specific subset of data (e.g. "alive" stems or stems which dbh is within some range). Subsetting data is not the job of these functions. Instead see `base::subset()`, `dplyr::filter()`, or `[`.

See Also

`dplyr::n()`, `dplyr::group_by()`.

Other functions for abundance and basal area: `abundance_byyr()`

Examples

```
library(fgeo.tool)

# abundance() ------------------------------------------------------------

abundance(data.frame(1))

# One stem per tree
tree <- tribble(
  ~TreeID, ~StemID, ~DBH,
  "1", "1.1", 11,
  "2", "2.1", 21
)

abundance(tree)

# One tree with multiple stems
stem <- tribble(
  ~TreeID, ~StemID, ~DBH,
  "1", "1.1", 11,
  "1", "1.2", 12
)

abundance(stem)

# Skip R CMD check for speed

# Similar but more realistic
assert_is_installed("fgeo.x")
stem <- fgeo.x::download_data("luquillo_stem5_random")

abundance(stem)

abundance(pick_main_stem(stem))


vft <- tribble(
  ~PlotName, ~CensusID, ~TreeID, ~StemID, ~DBH,
  "p", 1, "1", "1.1", 10,
  "q", 2, "1", "1.1", 10
```

```
)

# * Warns if it detects multiple values of censusid or plotname
# * Also warns if it detects duplicated values of treeid
abundance(vft)

# If trees have buttressess, the data may have multiple stems per treeid or
# multiple measures per stemid.
vft2 <- tribble(
  ~CensusID, ~TreeID, ~StemID, ~DBH, ~HOM,
  1, "1", "1.1", 88, 130,
  1, "1", "1.1", 10, 160,
  1, "2", "2.1", 20, 130,
  1, "2", "2.2", 30, 130,
)

# You should count only the main stem of each tree
(main_stem <- pick_main_stem(vft2))

abundance(main_stem)

vft3 <- tribble(
  ~CensusID, ~TreeID, ~StemID, ~DBH, ~HOM,
  1, "1", "1.1", 20, 130,
  1, "1", "1.2", 10, 160, # Main stem
  2, "1", "1.1", 12, 130,
  2, "1", "1.2", 22, 130 # Main stem
)

# You can compute by groups
by_census <- group_by(vft3, CensusID)
(main_stems_by_census <- pick_main_stem(by_census))

abundance(main_stems_by_census)

# basal_area() ------------------------------------------------------------

# Data must have a column named dbh (case insensitive)
basal_area(data.frame(dbh = 1))

# * Warns if it detects multiple values of censusid or plotname
# * Also warns if it detects duplicated values of stemid
basal_area(vft)

# First you may pick the main stemid of each stem
(main_stemids <- pick_main_stemid(vft2))

basal_area(main_stemids)

# You can compute by groups
basal_area(by_census)

# Skip R CMD check for speed
```

```
measurements_is_installed <- requireNamespace("measurements", quietly = TRUE)
if (measurements_is_installed) {
  library(measurements)

  # Convert units
  ba <- basal_area(by_census)
  ba$basal_area_he <- conv_unit(
    ba$basal_area,
    from = "mm2",
    to = "hectare"
  )

  ba
}
```

---

abundance_byyr *Create tables of abundance and basal area by year.*

Description

- abundance_byyr() first picks the main stem of each tree (see ?fgeo.tool::pick_main_stem()). Then, for each species and each round-mean-year of measurement, it counts the number of trees. The result includes **main stems** within a given dbh range.
- basal_area_byyr() first sums the basal basal area of all stems of each tree. Then, for each species and each round-mean-year of measurement, it sums the basal area of all trees. The result includes all stems within a given dbh range (notice the difference with abundance_byyr()).

Usage

```
abundance_byyr(vft, ...)

basal_area_byyr(vft, ...)
```

Arguments

vft	A ForestGEO-like dataframe; particularly a ViewFullTable. As such, it should contain columns PlotName, CensusID, TreeID, StemID, Status, DBH, Genus, SpeciesName, ExactDate, PlotCensusNumber, Family, Tag, and HOM. ExactDate should contain dates from 1980-01-01 to the present day in the format yyyy-mm-dd.
...	Expressions to pick main stems of a specific dbh range (e.g. DBH >= 10 or DBH >= 10, DBH < 20, or DBH >= 10 & DBH < 20).

Details

You don't need to pick stems by status before feeding data to these functions. Doing so may make your code more readable but it should not affect the result. This is because the expressions passed to ... pick data by dbh and exclude the missing dbh values associated to non-alive stems, including dead, missing, and gone stems.

Value

A dataframe.

See Also

`fgeo.tool::pick_main_stem()`.

Other functions for abundance and basal area: `abundance()`

Examples

```
library(fgeo.tool)

# Example data
vft <- tibble(
  PlotName = c("luq", "luq", "luq", "luq", "luq", "luq", "luq", "luq"),
  CensusID = c(1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L),
  TreeID = c(1L, 1L, 2L, 2L, 1L, 1L, 2L, 2L),
  StemID = c(1.1, 1.2, 2.1, 2.2, 1.1, 1.2, 2.1, 2.2),
  Status = c(
    "alive", "dead", "alive", "alive", "alive", "gone",
    "dead", "dead"
  ),
  DBH = c(10L, NA, 20L, 30L, 20L, NA, NA, NA),
  Genus = c("Gn", "Gn", "Gn", "Gn", "Gn", "Gn", "Gn", "Gn"),
  SpeciesName = c("spp", "spp", "spp", "spp", "spp", "spp", "spp", "spp"),
  ExactDate = c(
    "2001-01-01", "2001-01-01", "2001-01-01", "2001-01-01",
    "2002-01-01", "2002-01-01", "2002-01-01",
    "2002-01-01"
  ),
  PlotCensusNumber = c(1L, 1L, 1L, 1L, 2L, 2L, 2L, 2L),
  Family = c("f", "f", "f", "f", "f", "f", "f", "f"),
  Tag = c(1L, 1L, 2L, 2L, 1L, 1L, 2L, 2L),
  HOM = c(130L, 130L, 130L, 130L, 130L, 130L, 130L, 130L)
)

vft

abundance_byyr(vft, DBH >= 10, DBH < 20)

abundance_byyr(vft, DBH >= 10)

basal <- basal_area_byyr(vft, DBH >= 10)
basal
```

```
# Skip R CMD check for speed

measurements_is_installed <- requireNamespace("measurements", quietly = TRUE)
if (measurements_is_installed) {
  # Convert units
  years <- c("yr_2001", "yr_2002")
  basal_he <- basal %>%
    purrr::modify_at(
      years,
      ~ measurements::conv_unit(.x, from = "mm2", to = "hectare")
    )
  basal_he

  # Standardize
  number_of_hectares <- 50
  basal_he %>%
    purrr::map_at(years, ~ .x / number_of_hectares)
}
```

`fgeo_habitat` *Create habitat data from measures of topography.*

Description

This function constructs habitat data based on elevation data. It calculates habitats in two steps:

1. It calculates mean elevation, convexity and slope for each quadrat.
2. It calculates habitats based on hierarchical clustering of the topographic metrics from step 1.

Usage

```
fgeo_habitat(elev, gridsize, n, ...)
```

Arguments

`elev`	One of these: • A dataframe containing elevation data, with columns `gx`, `gy`, and `elev`, or `x`, `y`, and `elev` (e.g. `fgeo.x::elevation$col`). • A ForestGEO-like elevation list with elements `xdim` and `ydim` giving plot dimensions, and element `col` containing a dataframe as described in the previous item (e.g. `fgeo.x::elevation`).
`gridsize`	Number giving the size of each quadrat for which a habitat is calculated. Commonly, `gridsize = 20`.
`n`	Integer. Number of cluster-groups to construct (passed to the argument `k` to `stats::cutree()`).
`...`	Arguments passed to `fgeo_topography()`.

Value

A dataframe of subclass fgeo_habitat, with columns gx and gy, rounded with accuracy determined by gridsize, and column habitats, with as many distinct integer values as determined by the argument n.

Author(s)

Richard Condit.

See Also

fgeo.plot::autoplot.fgeo_habitat(), fgeo_topography().

Other habitat functions: fgeo_topography(), tt_test()

Other functions to construct fgeo classes: fgeo_topography()

Examples

```
assert_is_installed("fgeo.x")

# Input a ForestGEO-like elevation list or dataframe
elevation_ls <- fgeo.x::elevation
habitats <- fgeo_habitat(
  elevation_ls,
  gridsize = 20, n = 4
)

str(habitats)

# Habitat data is useful for calculating species-habitat associations
census <- fgeo.x::tree6_3species
as_tibble(
  tt_test(census, habitats)
)
```

---

fgeo_topography *Create topography data: convexity, slope, and mean elevation.*

Description

Create topography data: convexity, slope, and mean elevation.

Usage

```
fgeo_topography(elev, ...)

## S3 method for class 'data.frame'
fgeo_topography(
  elev,
```

```
  gridsize,
  xdim = NULL,
  ydim = NULL,
  edgecorrect = TRUE,
  ...
)

## S3 method for class 'list'
fgeo_topography(elev, gridsize, edgecorrect = TRUE, ...)
```

Arguments

`elev`	One of these: • A dataframe containing elevation data, with columns `gx`, `gy`, and `elev`, or `x`, `y`, and `elev` (e.g. `fgeo.x::elevation$col`). • A ForestGEO-like elevation list with elements `xdim` and `ydim` giving plot dimensions, and element `col` containing a dataframe as described in the previous item (e.g. `fgeo.x::elevation`).
`...`	Other arguments passed to methods.
`gridsize`	Number giving the size of each quadrat for which a habitat is calculated. Commonly, `gridsize = 20`.
`xdim, ydim`	(Required if `elev` is a dataframe) `x` and `y` dimensions of the plot.
`edgecorrect`	Correct convexity in edge quadrats?

Value

A dataframe of subclass fgeo_topography.

Acknowledgment

Thanks to Jian Zhang for reporting a bug (issue 59).

Author(s)

This function wraps code by Richard Condit.

See Also

`fgeo_habitat()`.

Other habitat functions: `fgeo_habitat()`, `tt_test()`

Other functions to construct fgeo classes: `fgeo_habitat()`

Examples

```
assert_is_installed("fgeo.x")

elev_list <- fgeo.x::elevation
fgeo_topography(elev_list, gridsize = 20)
```

```
elev_df <- elev_list$col
fgeo_topography(elev_df, gridsize = 20, xdim = 320, ydim = 500)
```

`recruitment_ctfs` *Recruitment, mortality, and growth.*

Description

These functions are adapted from the CTFS-R package. Compared to the original functions, these ones have a similar interface but use more conservative defaults and allow suppressing messages. These functions also feature formal tests, bug fixes, additional assertions, and improved messages.

Usage

```
recruitment_ctfs(
  census1,
  census2,
  mindbh = NULL,
  alivecode = NULL,
  split1 = NULL,
  split2 = NULL,
  quiet = FALSE
)

mortality_ctfs(
  census1,
  census2,
  alivecode = NULL,
  split1 = NULL,
  split2 = NULL,
  quiet = FALSE
)

growth_ctfs(
  census1,
  census2,
  rounddown = FALSE,
  method = "I",
  stdev = FALSE,
  dbhunit = "mm",
  mindbh = NULL,
  growthcol = "dbh",
  err.limit = 1000,
  maxgrow = 1000,
  split1 = NULL,
  split2 = NULL,
```

```
  quiet = FALSE
)
```

Arguments

census1, census2	Two census tables, each being a ForestGEO-like *tree* table (dataframe). A *stem* table won't fail, but you should use a *tree* table because demography analyses make more sense at the scale of trees than at the scale of stems.
mindbh	The minimum diameter above which the counts are done. Trees smaller than mindbh are excluded. By default all living trees of any size are included.
alivecode	Character; valid values of status indicating that a tree is alive. The default, 'A', is the standard CTFS designation for living trees or stems.
split1, split2	Optional vector(s) to aggregate results by. Each vector should be a column of either census1 or census2. The default aggregates the result across the entire census datasets.
quiet	Use TRUE to suppress messages.
rounddown	If TRUE, all dbh < 55 are rounded down to the nearest multiple of 5.
method	Either "I" or "E": • Use "I" to calculate annual dbh increment as (dbh2 -dbh1)/time • Use "E" to calculate the relative growth rate as (log(dbh2) -log(dbh1)) / time
stdev	Logical: • FALSE returns confidence limits. • TRUE returns the SD in growth rate per group.
dbhunit	"cm" or "mm".
growthcol	Either "dbh" or "agb" to define how growth is measured.
err.limit, maxgrow	A number. Numbers such as 10000 are high and will return all measures.

Details

Survivors are all individuals alive in both censuses, with status == A in the first census, and a diameter greater than mindbh in the first census. The total population in the second census includes all those alive plus any other survivors. Individuals whose status is NA in either census are deleted from all calculations.

Value

Metrics of recruitment: Similar to metrics of mortality.

Metrics of mortality:

- N: the number of individuals alive in the census 1 per category selected.
- D: the number of individuals no longer alive in census 2.

- `rate`: the mean annualized mortality rate constant per category selected, calculated as (log(N)-log(S))/time.
- `upper`: upper confidence limit of mean rate.
- `lower`: lower confidence limit of mean rate.
- `time`: mean time interval in years.
- `date1`: mean date included individuals were measured in census 1, as julian object (R displays as date, but treats as integer).
- `date2`: mean date in census 2.
- `dbhmean`: mean dbh in census 1 of individuals included.

Metrics of growth:

- `rate`, the mean annualized growth rate per category selected, either dbh increment, or relative growth.
- `N`, the number of individuals included in the mean (not counting any excluded).
- `clim` (or sd with `stdev = TRUE`), width of confidence interval; add this number to the mean rate to get upper confidence limit, substract to get lower.
- `dbhmean`, mean dbh in census 1 of individuals included.
- `time`, mean time interval in years.
- `date1`, mean date included individuals were measured in census 1, as julian object (R displays as date, but treats as integer).
- `date2`, mean date in census 2.

Author(s)

Richard Condit, Suzanne Lao.

Examples

```
assert_is_installed("fgeo.x")

census1 <- fgeo.x::tree5
census2 <- fgeo.x::tree6

as_tibble(
  recruitment_ctfs(census1, census2)
)

# Use `interaction(...)` to aggregate by any number of grouping variables
sp_quadrat <- interaction(census1$sp, census1$quadrat)

recruitment <- recruitment_ctfs(
  census1, census2,
  split1 = sp_quadrat,
  quiet = TRUE
)
as_tibble(recruitment)
```

```
mortality <- mortality_ctfs(
  census1, census2,
  split1 = sp_quadrat, quiet = TRUE
)
as_tibble(mortality)

growth <- growth_ctfs(census1, census2, split1 = sp_quadrat, quiet = TRUE)
as_tibble(growth)

# Easy way to separate grouping variables
tidyr_is_installed <- requireNamespace("tidyr", quietly = TRUE)
if (tidyr_is_installed) {
  library(tidyr)

  as_tibble(growth) %>%
    separate(groups, into = c("sp", "quadrat"))
}
```

`summary.tt_df` *Summary of* `tt_test()` *results.*

Description

Summary of `tt_test()` results.

Usage

```
## S3 method for class 'tt_df'
summary(object, ...)

## S3 method for class 'tt_lst'
summary(object, ...)
```

Arguments

`object`	An object of class "tt_df" or "tt_lst".
`...`	Not used (included only for compatibility with `summary`).

Value

A tibble.

Author(s)

Adapted from code contributed by Daniel Zuleta.

See Also

`tt_test()`, `base::summary()`.

Examples

```
assert_is_installed("fgeo.x")

tt_result <- tt_test(fgeo.x::tree6_3species, fgeo.x::habitat)

summary(tt_result)

# Same
summary(as_tibble(tt_result))

# You may want to add the explanation to the result of `tt_test()`

dplyr::left_join(as_tibble(tt_result), summary(tt_result))

# You may prefer a wide matrix
Reduce(rbind, tt_result)

# You may prefer a wide dataframe
tidyr::spread(summary(tt_result), "habitat", "association")
```

`tt_test` *Torus Translation Test to determine habitat associations of tree species.*

Description

Determine habitat-species associations with code developed by Sabrina Russo, Daniel Zuleta, Matteo Detto, and Kyle Harms.

Usage

```
tt_test(tree, habitat, sp = NULL, plotdim = NULL, gridsize = NULL)
```

Arguments

`tree`	A dataframe; a ForestGEO *tree* table (see details).
`habitat`	Object giving the habitat designation for each plot partition defined by `gridsize`. See `fgeo_habitat()`.
`sp`	Character sting giving any number of species-names.
`plotdim, gridsize`	Plot dimensions and gridsize. If `NULL` (default) they will be guessed, and a message will inform you of the chosen values. If the guess is wrong, you should provide the correct values manually (and check that your habitat data is correct).

Details

This test only makes sense at the population level. We are interested in knowing whether or not individuals of a species are aggregated on a habitat. Multiple stems of an individual do not represent population level processes but individual level processes. Thus, you should use data of individual trees – i.e. use a *tree* table, and not a *stem* table with potentially multiple stems per tree.

You should only try to determine the habitat association for sufficiently abundant species. In a 50-ha plot, a minimum abundance of 50 trees/species has been used.

Value

A list of matrices.

Acknowledgments

Nestor Engone Obiang, David Kenfack, Jennifer Baltzer, and Rutuja Chitra-Tarak provided feedback. Daniel Zuleta provided guidance.

Interpretation of Output

- `N.Hab.1`: Count of stems of the focal species in habitat 1.
- `Gr.Hab.1`: Count of instances the observed relative density of the focal species on habitat 1 was greater than the relative density based on the TT habitat map.
- `Ls.Hab.1`: Count of instances the observed relative density of the focal species on habitat 1 was less than the relative density based on the TT habitat map.
- `Eq.Hab.1`: Count of instances the observed relative density of the focal species on habitat 1 was equal to the relative density based on the TT habitat map. The sum of the `Gr.Hab.x`, `Ls.Hab.x`, and `Eq.Hab.x` columns for one habitat equals the number of 20 x20 quads in the plot. The `Rep.Agg.Neut` columns for each habitat indicate whether the species is significantly repelled (-1), aggregated (1), or neutrally distributed (0) on the habitat in question.

The probabilities associated with the test for whether these patterns are statistically significant are in the `Obs.Quantile` columns for each habitat. Note that to calculate the probability for *repelled*, it is *the value given*, but to calculate the probability for *aggregated*, it is *one minus the value given*.

Values of the `Obs.Quantile` < `0.025` means that the species is repelled from that habitat, while values of the `Obs.Quantile` > `0.975` means that the species is aggregated on that habitat.

References

Zuleta, D., Russo, S.E., Barona, A. et al. Plant Soil (2018). https://doi.org/10.1007/s11104-018-3878-0.

Author(s)

Sabrina Russo, Daniel Zuleta, Matteo Detto, and Kyle Harms.

See Also

`summary.tt_lst()`, `summary.tt_df()`, `as_tibble()`, `fgeo_habitat()`.

Other habitat functions: `fgeo_habitat()`, `fgeo_topography()`

Examples

```
library(fgeo.tool)
assert_is_installed("fgeo.x")

# Example data
tree <- fgeo.x::tree6_3species
elevation <- fgeo.x::elevation

# Pick alive trees, of 10 mm or more
census <- filter(tree, status == "A", dbh >= 10)

# Pick sufficiently abundant species
pick <- filter(dplyr::add_count(census, sp), n > 50)

# Use your habitat data or create it from elevation data
habitat <- fgeo_habitat(elevation, gridsize = 20, n = 4)

# Defaults to using all species
as_tibble(
  tt_test(census, habitat)
)

Reduce(rbind, tt_test(census, habitat))

some_species <- c("CASARB", "PREMON")
result <- tt_test(census, habitat, sp = some_species)
summary(result)
```

Index