Package ‘fgeo.tool’

March 23, 2020

Title Import and Manipulate ForestGEO Data

Version 1.2.6

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 easily import, filter, and modify ForestGEO data. To learn more about ForestGEO visit <http://www.forestgeo.si.edu/>.

License GPL-3

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

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

Depends R (>= 3.2)

Imports dplyr (>= 0.8.0.1), glue (>= 1.3.1), magrittr (>= 1.5), purrr (>= 0.3.2), readr (>= 1.3.1), rlang (>= 0.3.4), tibble (>= 2.1.1), tidyselect (>= 0.2.5)

Suggests covr (>= 3.2.1), fgeo.x (>= 1.1.3), knitr (>= 1.22), pingr (>= 1.1.2), roxygen2 (>= 6.1.1), spelling (>= 2.1), stringr (>= 1.4.0), testthat (>= 2.1.1), tidyr (>= 0.8.3)

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Mauro Lepore [aut, ctr, cre] (<https://orcid.org/0000-0002-1986-7988>),
Richard Condit [aut],
Suzanne Lao [aut],
Anudeep Singh [aut],
CTFS-ForestGEO [cph, fnd]

Maintainer Mauro Lepore <maurolepore@gmail.com>

Repository CRAN

Date/Publication 2020-03-23 15:00:05 UTC
**R topics documented:**

- `add_status_tree` ................................................................. 2
- `add_subquad` ................................................................. 3
- `add_var` ................................................................. 4
- `fgeo_elevation` ................................................................. 7
- `pick_drop` ................................................................. 8
- `pick_main_stem` ................................................................. 10
- `read_vft` ................................................................. 12
- `sanitize_vft` ................................................................. 13

**Index**

<table>
<thead>
<tr>
<th>add_status_tree</th>
<th>Add column status_tree based on the status of all stems of each tree.</th>
</tr>
</thead>
</table>

**Description**

Add column `status_tree` based on the status of all stems of each tree.

**Usage**

```r
add_status_tree(data, status_a = "A", status_d = "D")
```

**Arguments**

- `data` A ForestGEO-like dataframe: A `ViewFullTable`, `tree` or `stem` table.
- `status_a`, `status_d` Sting to match alive and dead stems; it corresponds to the values of the variable `status` (in census tables) or `Status` (with capital "S" in `ViewFull` tables).

**Value**

The input data set with the additional variable `status_tree`.

**See Also**

Other functions to add columns to dataframes: `add_subquad()`, `add_var()`
Other functions for ForestGEO data: `add_subquad()`, `add_var()`
Other functions for fgeo census: `add_var()`, `guess_plotdim()`, `pick_drop`
Other functions for fgeo vft: `add_subquad()`, `add_var()`, `guess_plotdim()`, `pick_drop`
add_subquad

Examples

stem <- tribble(~CensusID, ~treeID, ~stemID, ~status,
1, 1, 1, "A",
1, 1, 2, "D",
1, 2, 3, "D",
1, 2, 4, "D",
2, 1, 1, "A",
2, 1, 2, "G",
2, 2, 3, "D",
2, 2, 4, "G"
)

add_status_tree(stem)

add_subquad Add column subquadrat based on QX and QY coordinates.

Description

Add column subquadrat based on QX and QY coordinates.

Usage

add_subquad(data, x_q, y_q = x_q, x_sq, y_sq = x_sq, subquad_offset = NULL)

Arguments

data A dataframe with quadrat coordinates QX and QY (e.g. a ViewFullTable).
x_q, y_q Size in meters of a quadrat’s side. For ForestGEO sites, a common value is 20.
x_sq, y_sq Size in meters of a subquadrat’s side. For ForestGEO sites, a common value is 5.
subquad_offset Either -1 or 1, to rest or add one unit to the digit of each subquadrat that rep-resents the column number.

<table>
<thead>
<tr>
<th>First column is 0</th>
<th>First column is 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>04 14 24 34</td>
<td>14 24 34 44</td>
</tr>
<tr>
<td>03 13 23 33</td>
<td>13 23 33 43</td>
</tr>
<tr>
<td>02 12 22 32</td>
<td>12 22 32 42</td>
</tr>
<tr>
<td>01 11 21 31</td>
<td>11 21 31 41</td>
</tr>
</tbody>
</table>
Value

Returns data with the additional variable subquadrat.

Author(s)

Anudeep Singh and Mauro Lepore.

See Also

Other functions to add columns to dataframes: `add_status_tree()`, `add_var()`
Other functions for ForestGEO data: `add_status_tree()`, `add_var()`
Other functions for fgeo vft: `add_status_tree()`, `add_var()`, `guess_plotdim()`, `pick_drop`

Examples

```r
vft <- tribble(
  ~QX, ~QY,
  17.9, 0,
  4.1, 15,
  6.1, 17.3,
  3.8, 5.9,
  4.5, 12.4,
  4.9, 9.3,
  9.8, 3.2,
  18.6, 1.1,
  17.3, 4.1,
  1.5, 16.3
)

add_subquad(vft, 20, 20, 5, 5)

add_subquad(vft, 20, 20, 5, 5, subquad_offset = -1)
```

Description

These functions add columns to position trees in a forest plot. They work with `ViewFullTable`, `tree` and `stem` tables. From the input table, most functions use only the `gx` and `gy` columns (or equivalent columns). The exception is the function `add_gxgy()` which inputs quadrat information. If your data lacks some important column, an error message will inform you which column is missing.
add_var

Usage

add_lxly(data, gridsize = 20, plotdim = NULL)
add_qxqy(data, gridsize = 20, plotdim = NULL)
add_index(data, gridsize = 20, plotdim = NULL)
add_col_row(data, gridsize = 20, plotdim = NULL)
add_hectindex(data, gridsize = 20, plotdim = NULL)
add_quad(data, gridsize = 20, plotdim = NULL, start = NULL, width = 2)
add_gxgy(data, gridsize = 20, start = 0)

Arguments

data      A ForestGEO-like dataframe: A ViewFullTable, tree or stem table.
gridsize  The gridsize of the census plot (commonly 20 m).
plotdim   The global dimensions of the census plot (i.e. the maximum possible values of
gx and gy).
start     Defaults to label the first quadrat as "0101". Use 0 to label it as "0000" instead.
width     Number; width to pad the labels of plot-columns and -rows.

Details

These functions are adapted from the CTFS R Package.

Value

For any given var, a function add_var() returns a modified version of the input dataframe, with
the additional variable(s) var.

See Also

Other functions to add columns to dataframes: add_status_tree(), add_subquad()
Other functions for ForestGEO data: add_status_tree(), add_subquad()
Other functions for fgeo census: add_status_tree(), guess_plotdim(), pick_drop
Other functions for fgeo vft: add_status_tree(), add_subquad(), guess_plotdim(), pick_drop

Examples

x <- tribble(
  ~gx, ~gy,
  0, 0,
  50, 25,
  999.9, 499.95,
```

1000, 500
)

# 'gridsize' has a common default; 'plotdim' is guessed from the data
add_lxly(x)

gridsize <- 20
plotdim <- c(1000, 500)

add_qxqy(x, gridsize, plotdim)

add_index(x, gridsize, plotdim)

add_hectindex(x, gridsize, plotdim)

add_quad(x, gridsize, plotdim)

add_quad(x, gridsize, plotdim, start = 0)

# 'width' gives the number of digits to pad the label of plot-rows and
# plot-columns, e.g. 3 pads plot-rows with three zeros and plot-columns with
# an extra three zeros, resulting in a total of 6 zeros.
add_quad(x, gridsize, plotdim, start = 0, width = 3)

add_col_row(x, gridsize, plotdim)

# From 'quadrat' or 'QuadratName'--------------------------------------
x <- tribble(~QuadratName,  
             "0001",  
             "0011",  
             "0101",  
             "1001"
)

# Output 'gx' and 'gy' ------------------------------------
add_gxgy(x)

assert_is_installed("fgeo.x")
# Warning: The data may already have 'gx' and 'gx' columns
gxgy <- add_gxgy(fgeo.x::tree5)
select(gxgy, matches("gx|gy"))

# Output 'col' and 'row' -----------------------------------

# Create columns 'col' and 'row' from 'QuadratName' with 'tidyr::separate()'  
# The argument 'sep' lets you separate 'QuadratName' at any position
## Not run:
tidyr_is_installed <- requireNamespace("tidyr", quietly = TRUE)
stringr_is_installed <- requireNamespace("stringr", quietly = TRUE)
```
if (tidyr_is_installed && stringr_is_installed) {
    library(tidyr)
    library(stringr)

    vft <- tibble(QuadratName = c("0001", "0011"))
    vft

    separate(
        vft,
        QuadratName,
        into = c("col", "row"),
        sep = 2
    )

    census <- select(fgeo.x::tree5, quadrat)
    census

    census$quadrat <- str_pad(census$quadrat, width = 4, pad = 0)

    separate(
        census,
        quadrat,
        into = c("col", "row"),
        sep = 2,
        remove = FALSE
    )
}

## End(Not run)

---

**fgeo_elevation**

Create elevation data.

**Description**

This function constructs an object of class "fgeo_elevation". It standardizes the structure of elevation data to always output a dataframe with names gx, gy and elev.

**Usage**

fgeo_elevation(elev)

**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).
Value

A dataframe with names x/gx, y/gy and elev.

Acknowledgments

This function was inspired by David Kenfack.

Examples

assert_is_installed("fgeo.x")

# Input: Elevation dataframe
elevation_df <- fgeo.x::elevation$col
fgeo_elevation(elevation_df)

class(elevation_df)
class(fgeo_elevation(elevation_df))

names(elevation_df)
names(fgeo_elevation(elevation_df))

# Input: Elevation list
elevation_ls <- fgeo.x::elevation
fgeo_elevation(elevation_ls)

class(elevation_ls)
class(fgeo_elevation(elevation_ls))

names(elevation_ls)
names(fgeo_elevation(elevation_ls))

Description

These functions provide an expressive and convenient way to pick specific rows from ForestGEO datasets. They allow you to remove missing values (with na.rm = TRUE) but conservatively default to preserving them. This behavior is similar to base::subset() and unlike dplyr::filter(). This conservative default is important because you want want to include missing trees in your analysis.

Usage

```r
pick_dbh_min(data, value, na.rm = FALSE)
pick_dbh_max(data, value, na.rm = FALSE)
pick_dbh_under(data, value, na.rm = FALSE)
```
pick_drop

pick_dbh_over(data, value, na.rm = FALSE)

pick_status(data, value, na.rm = FALSE)

drop_status(data, value, na.rm = FALSE)

Arguments

data     A ForestGEO-like dataframe: A ViewFullTable, tree or stem table.
value    An atomic vector; a single value against to compare each value of the variable
         encoded in the function’s name.
na.rm    Set to TRUE if you want to remove missing values from the variable encoded in
         the function’s name.

Value

A dataframe similar to .data but including only the rows with matching conditions.

See Also

dplyr::filter(), Extract().
Other functions for fgeo census and vft: guess_plotdim()
Other functions for fgeo census: add_status_tree(), add_var(), guess_plotdim()
Other functions for fgeo vft: add_status_tree(), add_subquad(), add_var(), guess_plotdim()
Other functions to pick or drop rows of a ForestGEO dataframe: pick_main_stem()

Examples

census <- tribble(
  ~dbh, ~status,
    0,   "A",
    50,  "A",
   100,  "A",
   150,  "A",
      NA, "M",
      NA, "D",
      NA, NA
)

# <=
pick_dbh_max(census, 100)
pick_dbh_max(census, 100, na.rm = TRUE)

# >=
pick_dbh_min(census, 100)
pick_dbh_min(census, 100, na.rm = TRUE)

# <
### pick_main_stem

Pick the main stem or main stemid(s) of each tree in each census.

**Description**

- `pick_main_stem()` picks a unique row for each treeID per census.
pick_main_stem

- pick_main_stemid() picks a unique row for each stemID per census. It is only useful when a single stem was measured twice in the same census, which sometimes happens to correct for the effect of large buttresses.

Usage

pick_main_stem(data)

pick_main_stemid(data)

Arguments

data A ForestGEO-like dataframe: A ViewFullTable, tree or stem table.

Details

- pick_main_stem() picks the main stem of each tree in each census. It collapses data of multi-stem trees by picking a single stem per treeid per censusid. From this group, it picks the stem at the top of a list sorted first by descending order of hom and then by descending order of dbh. This this corrects the effect of buttresses and picks the main stem. It ignores groups of grouped data and rejects data with multiple plots.

- pick_main_stemid() does one step less than pick_main_stem(). It only picks the main stemid(s) of each tree in each census and keeps all stems per treeid. This is useful when calculating the total basal area of a tree, because you need to sum the basal area of each individual stem as well as sum only one of the potentially multiple measurements of each buttressed stem per census.

Value

A dataframe with a single plotname, and one row per per treeid per censusid.

Warning

These functions may be considerably slow. They are fastest if the data already has a single stem per treeid. They are slower with data containing multiple stems per treeid (per censusid), which is the main reason for using this function. The slowest scenario is when data also contains duplicated values of stemid per treeid (per censusid). This may happen if trees have buttresses, in which case these functions check every stem for potential duplicates and pick the one with the largest hom value.

For example, in a windows computer with 32 GB of RAM, a dataset with 2 million rows with multiple stems and buttresses took about 3 minutes to run. And a dataset with 2 million rows made up entirely of main stems took about ten seconds to run.

See Also

Other functions to pick or drop rows of a ForestGEO dataframe: pick_drop
Examples

```r
# One `treeID` with multiple stems.
# `stemID == 1.1` has two measurements (due to buttresses).
# `stemID == 1.2` has a single measurement.

census <- tribble(
  ~sp, ~treeID, ~stemID, ~hom, ~dbh, ~CensusID,
  "sp1", "1", "1.1", 140, 40, 1, # main stemID (max `hom`) 
  "sp1", "1", "1.1", 130, 60, 1, 
  "sp1", "1", "1.2", 130, 55, 1 # main stemID (only one) 
)

# Picks a unique row per unique `treeID`
pick_main_stem(census)

# Picks a unique row per unique `stemID`
pick_main_stemid(census)
```

---

**Description**

`read_vft()` and `read_taxa()` help you to read ViewFullTable and ViewTaxonomy data from text files delivered by the ForestGEO database. These functions avoid common problems about column separators, missing values, column names, and column types.

**Usage**

```r
read_vft(file, delim = NULL, na = c("", "NA", "NULL"), ...)
read_taxa(file, delim = NULL, na = c("", "NA", "NULL"), ...)
```

**Arguments**

- `file` A path to a file.
- `delim` Single character used to separate fields within a record. The default (`delim = NULL`) is to guess between comma or tab ("," or "\t").
- `na` Character vector of strings to interpret as missing values. Set this option to `character()` to indicate no missing values.
- `...` Other arguments passed to `readr::read_delim()`.

**Value**

A tibble.
sanitize_vft

Acknowledgments

Thanks to Shameema Jafferjee Esufali for inspiring the feature that automatically detects delim (issue 65).

See Also

readr::read_delim(). type_vft(). type_taxa().

Other functions to read text files delivered by ForestGEO’s database: type_vft()

Examples

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

example_path()

file_vft <- example_path("view/vft_4quad.csv")
read_vft(file_vft)

file_taxa <- example_path("view/taxa.csv")
read_taxa(file_taxa)

---

sanitize_vft  Fix common problems in ViewFullTable and ViewTaxonomy data.

Description

These functions fix common problems of ViewFullTable and ViewTaxonomy data:

- Ensure that each column has the correct type.
- Ensure that missing values are represented with NAs – not with the literal string "NULL".

Usage

sanitize_vft(.data, na = c("", "NA", "NULL"), ...)

sanitize_taxa(.data, na = c("", "NA", "NULL"), ...)

Arguments

.data  A dataframe; either a ForestGEO ViewFullTable (sanitize_vft()). or ViewTaxonomy (sanitize_vft()).

na  Character vector of strings to interpret as missing values. Set this option to character() to indicate no missing values.

...  Arguments passed to readr::type_convert().
Value

A dataframe.

Acknowledgments

Thanks to Shameema Jafferjee Esufali for motivating this functions.

See Also

read_vft().

Examples

assert_is_installed("fgeo.x")

vft <- fgeo.x::vft_4quad

# Introduce problems to show how to fix them
# Bad column types
vft[] <- lapply(vft, as.character)
# Bad representation of missing values
vft$PlotName <- "NULL"

# "NULL" should be replaced by `NA` and `DBH` should be numeric
str(vft[c("PlotName", "DBH")])

# Fix
vft_sane <- sanitize_vft(vft)
str(vft_sane[c("PlotName", "DBH")])

taxa <- read.csv(fgeo.x::example_path("taxa.csv"))
# E.g. inserting bad column types
taxa[] <- lapply(taxa, as.character)
# E.g. inserting bad representation of missing values
taxa$SubspeciesID <- "NULL"

# "NULL" should be replaced by `NA` and `ViewID` should be integer
str(taxa[c("SubspeciesID", "ViewID")])

# Fix
taxa_sane <- sanitize_taxa(taxa)
str(taxa_sane[c("SubspeciesID", "ViewID")])
Index

add_col_row(add_var), 4  
add_gxgy(add_var), 4  
add_hectindex(add_var), 4  
add_index(add_var), 4  
add_lxly(add_var), 4  
add_quad(add_var), 4  
add_qxqy(add_var), 4  
add_status_tree, 2, 4, 5, 9  
add_subquad, 2, 3, 5, 9  
add_var, 2, 4, 4, 9  
base::subset(), 8  
drop_status(pick_drop), 8  
fgeo_elevation, 7  
guess_plotdim, 2, 4, 5, 9  
pick_dbh_max(pick_drop), 8  
pick_dbh_min(pick_drop), 8  
pick_dbh_over(pick_drop), 8  
pick_dbh_under(pick_drop), 8  
pick_drop, 2, 4, 5, 8, 11  
pick_main_stem, 9, 10  
pick_main_stem(), 10, 11  
pick_main_stemid(pick_main_stem), 10  
pick_main_stemid(), 11  
pick_status(pick_drop), 8  
read_taxa(read_vft), 12  
read_taxa(), 12  
read_vft(), 12, 14  
readr::read_delim(), 12, 13  
readr::type_convert(), 13  
sanitize_taxa(sanitize_vft), 13  
sanitize_vft, 13  
tibble, 12  
type_taxa(), 13  
type_vft, 13  
type_vft(), 13