Package ‘irg’

February 15, 2019

Title  Instantaneous Rate of Green Up
Version  0.1.1
Description  Fits a double logistic function to NDVI time series and calculates instantaneous rate of green (IRG) according to methods described in Bischoff et al. (2012) <doi:10.1086/667590>.
License  GPL-3
Encoding  UTF-8
LazyData  true
RoxygenNote  6.1.1
Suggests  testthat, knitr, rmarkdown, DiagrammeR, ggplot2
Depends  R (>= 2.10)
Imports  data.table, RcppRoll, stats
VignetteBuilder  knitr
URL  http://irg.robitalec.ca/
BugReports  https://gitlab.com/robit.a/irg/issues
NeedsCompilation  no
Author  Alec L. Robitaille [aut, cre] (<https://orcid.org/0000-0002-4706-1762>)
Maintainer  Alec L. Robitaille <robit.alec@gmail.com>
Repository  CRAN
Date/Publication  2019-02-15 15:20:06 UTC

R topics documented:

  calc_irg ................................................................. 2
  filter_ndvi ............................................................ 3
  filter_qa ............................................................... 4
  filter_roll ............................................................ 5
  filter_top ............................................................. 6
  filter_winter .......................................................... 7
Description

Calculate the instantaneous rate of green-up.

Usage

calc_irg(DT, id = "id", year = "yr", scaled = TRUE)

Arguments

DT data.table of model parameters (output from model_params).
id id column. default is 'id'. See details.
year year column name. default is 'yr'.
scaled boolean indicating if irg should be rescaled between 0-1 within id and year. If TRUE, provide id and year. Default is TRUE.

Details

The DT argument expects a data.table of model estimated parameters for double logistic function of NDVI for each year and individual. Since it is the rate of green-up, model parameters required are only xmidS and scalS.

The scaled argument is used to optionally rescale the IRG result to 0-1, for each year and individual. The id argument is used to split between sampling units. This may be a point id, polygon id, pixel id, etc. depending on your analysis. This should match the id provided to filtering functions. The formula used is described in Bischoff et al. (2012):

\[
IRG = \frac{\exp((t+xmidS)/scalS)}{(2*scalS*(\exp(1)^{(t+xmidS)/scalS})+(scalS*(\exp(1)^{(2*t)/scalS}))+(scalS*exp(1))}
\]

(See the "Getting started with irg vignette" for a better formatted formula.)

Value

Extended data.table 'irg' column of instantaneous rate of green-up calculated for each day of the year, for each individual and year.
filter_ndvi

See Also

Other irg: irg

Examples

```r
# Load data.table
library(data.table)

# Read in example data
dvi <- fread(system.file("extdata", "dvi.csv", package = "irg"))

# Filter and scale NDVI time series
filter_ndvi(dvi)
scale_doy(dvi)
scale_ndvi(dvi)

# Guess starting parameters
model_start(dvi)

# Double logistic model parameters given starting parameters for nls
mods <- model_params(
    ndvi,
    return = 'models',
    xmidS = 'xmidS_start',
    xmidA = 'xmidA_start',
    scalS = 0.05,
    scalA = 0.01
)

# Fit double logistic curve to NDVI time series
fit <- model_ndvi(mods, observed = FALSE)

# Calculate IRG for each day of the year
calc_irg(fit)
```

Description

Meta function, calling all filtering steps, in order. Only defaults.

Usage

`filter_ndvi(DT)`

Arguments

- `DT`: data.table of NDVI time series
filter_qa

Value

filtered NDVI time series.

See Also

Other filter: filter_qa, filter_roll, filter_top, filter_winter

Examples

# Load data.table
library(data.table)

# Read example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

# Use filter_ndvi to apply all filtering steps (with defaults)
filter_ndvi(ndvi)

---

filter_qa  
*Filter with QA Band*

Description

Using MODIS QA band information, filter the NDVI time series.

Usage

`filter_qa(DT, qa = "SummaryQA", good = c(0, 1))`

Arguments

- **DT**: data.table of NDVI time series
- **qa**: QA column. default is 'SummaryQA'.
- **good**: values which correspond to quality pixels. default is 0 and 1.

Value

filtered data.table with appended 'filtered' column of "quality" NDVI.

See Also

Other filter: filter_ndvi, filter_roll, filter_top, filter_winter
Examples

```r
# Load data.table
library(data.table)

# Read example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

filter_qa(ndvi, qa = 'SummaryQA', good = c(0, 1))
```

**filter_roll**  
*Filter with rolling median*

Description

Using a rolling median, filter the NDVI time series for each id.

Usage

```r
filter_roll(DT, window = 3L, id = "id", method = "median")
```

Arguments

- `DT`: data.table of NDVI time series
- `window`: window size. default is 3.
- `id`: id column. default is 'id'. See details.
- `method`: median. no other options yet. let me know if you are looking for something else.

Details

The id argument is used to split between sampling units. This may be a point id, polygon id, pixel id, etc. depending on your analysis.

Value

filtered data.table with appended 'rolled' column of each id’s rolling median, filtered NDVI time series.

See Also

Other filter: `filter_ndvi, filter_qa, filter_top, filter_winter`
Examples

```r
# Load data.table
library(data.table)

# Read example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

filter_qa(ndvi, qa = 'SummaryQA', good = c(0, 1))
filter_winter(ndvi, probs = 0.025, limits = c(60L, 300L), doy = 'DayOfYear', id = 'id')
filter_roll(ndvi, window = 3L, id = 'id')
```

---

**filter_top**  
*Filter top NDVI*

Description

Using upper quantile (default = 0.925) of multi-year MODIS data, determine the top NDVI for each id.

Usage

```r
filter_top(DT, probs = 0.925, id = "id")
```

Arguments

- `DT`  
data.table of NDVI time series
- `probs`  
quantile probability to determine top. default is 0.925.
- `id`  
id column. default is 'id'. See details.

Details

The id argument is used to split between sampling units. This may be a point id, polygon id, pixel id, etc. depending on your analysis.

Value

filtered data.table with appended 'top' column of each id’s top (quantile) NDVI value.

See Also

Other filter: `filter_ndvi, filter_qa, filter_roll, filter_winter`
**filter_winter**

**Examples**

```r
# Load data.table
library(data.table)

# Read example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

filter_qa(ndvi, qa = 'SummaryQA', good = c(0, 1))
filter_winter(ndvi, probs = 0.025, limits = c(60L, 300L), doy = 'DayOfYear', id = 'id')
filter_roll(ndvi, window = 3L, id = 'id')
filter_top(ndvi, probs = 0.925, id = 'id')
```

---

**Description**

Using lower quantile (default = 0.025) of multi-year MODIS data, determine the "winterNDVI" for each id.

**Usage**

```r
filter_winter(DT, probs = 0.025, limits = c(60L, 300L),
              doy = "DayOfYear", id = "id")
```

**Arguments**

- **DT**: data.table of NDVI time series
- **probs**: quantile probability to determine "winterNDVI". default is 0.025.
- **limits**: integer vector indicating limit days of absolute winter (snow cover, etc.). default = 60 days after Jan 1 and 65 days before Jan 1.
- **doy**: julian day column. default is 'DayOfYear'. integer type.
- **id**: id column. default is 'id'. See details.

**Details**

The id argument is used to split between sampling units. This may be a point id, polygon id, pixel id, etc. depending on your analysis.

**Value**

filtered data.table with appended 'winter' column of each id's "winterNDVI" baseline value.

**See Also**

Other filter: `filter_ndvi`, `filter_qa`, `filter_roll`, `filter_top`
Examples

```r
# Load data.table
library(data.table)

# Read example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))
filter_qa(ndvi, qa = 'SummaryQA', good = c(0, 1))
filter_winter(ndvi, probs = 0.025, limits = c(60, 300), doy = 'DayOfYear', id = 'id')
```

---

### Description

Wrapper function for one step IRG calculation. Only defaults.

### Usage

```r
irg(DT)
```

### Arguments

- **DT**
  - data.table of NDVI time series

### Details

data.table must have columns:

- 'id' - individual identifier
- 'yr' - year of observation
- 'NDVI' - NDVI value
- 'DayOfYear' - day of year/julian day of observation
- 'SummaryQA' - summary quality value for each sample (provided by MODIS)

### Value

Extended data.table 'irg' column of instantaneous rate of green-up calculated for each day of the year, for each individual and year.

### See Also

Other irg: `calc_irg`
model_ndvi

Examples

# Load data.table
library(data.table)

# Read in example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

# Calculate IRG for each day of the year and individual
out <- irg(ndvi)

model_ndvi

Model NDVI time series

Description

Fit double logistic model to NDVI time series given parameters estimated with model_params.

Usage

model_ndvi(DT, observed = TRUE)

Arguments

DT data.table of model parameters (output from model_params).

observed boolean indicating if a full year of fitted values should be returned (observed = FALSE) or if only observed values will be fit (observed = TRUE)

Value

Model parameter data.table appended with 'fitted' column of double logistic model of NDVI for a full year. Calculated at the daily scale with the following formula from Bischoff et al. (2012).

\[
fitted = \frac{1}{1 + \exp \frac{2 \text{midS} - t}{2 \text{scals}}} - \frac{1}{1 + \exp \frac{2 \text{midA} - t}{2 \text{scala}}}
\]

(See the "Getting started with irg vignette" for a better formatted formula.)

References

https://www.journals.uchicago.edu/doi/abs/10.1086/667590

See Also

Other model: model_params, model_start
Examples

```r
# Load data.table
library(data.table)

# Read in example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

# Filter and scale NDVI time series
filter_ndvi(ndvi)
scale_doy(ndvi)
scale_ndvi(ndvi)

# Guess starting parameters for xmidS and xmidA
model_start(ndvi)

## Two options: fit to full year or observed data
## Option 1 - returns = 'models'

## Double logistic model parameters
## given global starting parameters for scaleS, scaleA
## and output of model_start for xmidS, xmidA
mods <- model_params(
  ndvi,
  returns = 'models',
  xmidS = 'xmidS_start',
  xmidA = 'xmidA_start',
  scaleS = 0.05,
  scaleA = 0.01
)

# Fit to the whole year (requires assignment)
fit <- model_ndvi(mods, observed = FALSE)

## Option 2 - returns = 'columns'
model_params(
  ndvi,
  returns = 'columns',
  xmidS = 'xmidS_start',
  xmidA = 'xmidA_start',
  scaleS = 0.05,
  scaleA = 0.01
)

# Fit double logistic curve to NDVI time series for the observed days
model_ndvi(ndvi, observed = TRUE)
```

---

**model_params**

Estimate model parameters
model_params

Description

Model estimated parameters for fitting double logistic curve.

Usage

model_params(DT, returns = NULL, id = "id", year = "yr", 
               xmidS = NULL, xmidA = NULL, scalS = NULL, scalA = NULL)

Arguments

  DT      data.table of NDVI time series. Also optionally starting estimates. See Details.
  returns either 'models' or 'columns'. 'models' will return a data.table of model outcomes by id and year. 'columns' will append model estimate parameters to the input DT.
  id      id column. default is 'id'. See details.
  year    year column name. default is 'yr'.
  xmidS   starting estimates. see Details. - "spring inflection point"
  xmidA   starting estimates. see Details. - "fall inflection point"
  scalS   starting estimates. see Details. - "scale parameter for spring green-up portion of the NDVI curve"
  scalA   starting estimates. see Details. - "scale parameter for fall dry-down portion of the NDVI curve"

Details

Arguments xmidS, xmidA, scalS, scalA allow users to provide either group level or global starting estimates to be used for all models.

Either: a character indicating the column name which stores a group level starting parameter (possibly created by model_start OR a numeric value used as a global value for all models. See nls for more details on starting parameters.

Default value for the year column is 'yr'. If you only have one year of data, set to NULL.

The id argument is used to split between sampling units. This may be a point id, polygon id, pixel id, etc. depending on your analysis. This should match the id provided to filtering functions.

Formula and arguments xmidS, xmidA, scalS, scalA following this from Bischoff et al. (2012).

\[
\text{fitted} = \frac{1}{1 + \exp \frac{\text{xmidS}}{\text{scalS}} - \frac{1}{1 + \exp \frac{\text{xmidA}}{\text{scalA}}}}
\]

Value

data.table of model estimated parameters for double logistic model. If any rows are NULL, nls could not fit a model given starting parameters to the data provided.

References

https://www.journals.uchicago.edu/doi/abs/10.1086/667590
See Also

Other model: model_ndvi, model_start

Examples

```r
# Load data.table
library(data.table)

# Read in example data
da <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

# Filter and scale NDVI time series
filter_ndvi(da)
scale_doy(da)
scale_ndvi(da)

# Guess starting parameters for xmidS and xmidA
model_start(da)

# Double logistic model parameters
# given global starting parameters for scala, scalA
# and output of model_start for xmidS, xmidA
mds <- model_params(
  da,
  returns = 'models',
  xmidS = 'xmidS_start',
  xmidA = 'xmidA_start',
  scala = 0.05,
  scalA = 0.01
)
```

---

**model_start**  
*Model starting parameters*

**Description**

Try guessing starting parameters for model_params and model_ndvi.

**Usage**

```
model_start(DT, id = "id", year = "yr")
```

**Arguments**

- **DT**  
  filtered and scaled data.table of NDVI time series. Expects columns 'scaled' and 't' are present.

- **id**  
  id column. default is 'id'. See details.

- **year**  
  year column name. default is 'yr'.
Details

The id argument is used to split between sampling units. This may be a point id, polygon id, pixel id, etc. depending on your analysis. This should match the id provided to filtering functions.

Value

The input DT data.table appended with xmidS_start and xmidA_start columns. Note - we currently do not attempt to guess appropriate starting values for scalaS and scalaA.

See Also

Other model: model_ndvi, model_params

Examples

```R
# Load data.table
library(data.table)

# Read in example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

# Filter and scale NDVI time series
filter_ndvi(ndvi)
scale_doy(ndvi)
scale_ndvi(ndvi)

# Guess starting parameters for xmidS and xmidA
model_start(ndvi)
```

---

**ndvi**

*Raw NDVI Time Series*

---

Description

A data.table containing NDVI samples for ten points over ten years (2002-2012).

Format

A data.table with 2530 rows and 5 variables:

- id - individual identifier
- yr - year of sample
- DayOfYear - julian day/day of year of sample
- NDVI - sampled value
- SummaryQA - Summary quality assessment value

SummaryQA details:
- 0 - Good data, use with confidence
- 1 - Marginal data, useful but look at detailed QA for more information
- 2 - Pixel covered with snow/ice
- 3 - Pixel is cloudy

Examples

```r
# Load data.table
library(data.table)

# Read example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))
```

scale_doy

**Scale DOY**

**Description**

Scale the day of the year to 0-1 (like NDVI).

**Usage**

```r
scale_doy(DT, doy = "DayOfYear")
```

**Arguments**

<table>
<thead>
<tr>
<th>DT</th>
<th>data.table of NDVI time series</th>
</tr>
</thead>
<tbody>
<tr>
<td>doy</td>
<td>julian day column. default is 'DayOfYear'. integer type.</td>
</tr>
</tbody>
</table>

**Value**

data.table with appended 't' column of 0-1 scaled day of year.

**See Also**

Other scale: `scale_ndvi`

**Examples**

```r
# Load data.table
library(data.table)

# Read in example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg"))

# Scale DOY
scale_doy(ndvi)
```
**scale_ndvi**

**Scale NDVI**

---

**Description**

Using filtered NDVI time series, scale it to 0-1.

**Usage**

```r
scale_ndvi(DT)
```

**Arguments**

- `DT`: data.table of NDVI time series

**Details**

This function expects the input `DT` is the output of previous four filtering steps, or `filter_ndvi`.

**Value**

data.table with appended 'scaled' column of 0-1 scaled NDVI.

**See Also**

Other scale: `scale_doy`

**Examples**

```r
# Load data.table
library(data.table)

# Read in example data
ndvi <- fread(system.file("extdata", "ndvi.csv", package = "irg" ))

# Filter and scale NDVI time series
filter_ndvi(ndvi)
scale_ndvi(ndvi)
```
Index

calc_irg. 2, 8
filter_ndvi. 3, 4–7
filter_qa. 4, 4, 5–7
filter_roll. 4, 5, 6, 7
filter_top. 4, 5, 6, 7
filter_winter. 4–6, 7

irg. 3, 8

model_ndvi. 9, 12, 13
model_params. 9, 10, 13
model_start. 9, 11, 12, 12

ndvi. 13
nls. 11

scale_doy. 14, 15
scale_ndvi. 14, 15