Package 'caliver'

February 19, 2021

Type Package

Title Calibration and Verification of Gridded Model Outputs

Version 2.0.0

Maintainer Claudia Vitolo <claudia.vitolo@ecmwf.int>

URL https://ecmwf.github.io/caliver/, https://github.com/ecmwf/caliver

BugReports https://github.com/ecmwf/caliver/issues

Description Utility functions for the post-processing, calibration and validation of grid model outputs. Initial test cases include the outputs of the following forest fire models: GEFF and RISICO. The package is described in Vitolo et al. (2018) ``Caliver: An R package for CALIbration and VERification of forest fire gridded model outputs'' <doi:10.1371/journal.pone.0189419>.

Depends R (>= 3.5)

Imports ncdf4, ggplot2, lubridate, raster, rworldmap, graphics, stats

Suggests testthat, knitr, rmarkdown, covr, lintr

VignetteBuilder knitr

Encoding UTF-8

SystemRequirements GDAL, PROJ, netcdf4, openssl

License Apache License 2.0

Language en-GB

RoxygenNote 7.1.1

NeedsCompilation no

Author Claudia Vitolo [aut, cre] (<https://orcid.org/0000-0002-4252-1176>), Francesca Di Giuseppe [ctb], Mirko D'Andrea [aut, ctb]

Repository CRAN

Date/Publication 2021-02-19 11:00:03 UTC

R topics documented:

`anomaly` *anomaly*

Description

This function calculates the anomaly (number of standard deviations from the mean climatology) of a forecast layer.

Usage

```
anomaly(r, b)
```

Arguments

`r`	is the RasterLayer to compare to the climatology.
`b`	RasterBrick/Stack containing the historical observations or a proxy (typically a reanalysis) that is used to derive the climatological information.

Details

The objects `r` and `b` should be comparable: same resolution and extent. More information on anomaly is available here: https://bit.ly/2Qvekz4. To estimate fire climatology one can use hindcast or reanalysis data. Examples of the latter are available from Zenodo: https://zenodo.org/communities/wildfire.

Value

The function returns a RasterLayer with extent, resolution and land-sea mask matching those of `r`. Values are the number standard deviations from the historical mean values.

Examples

```
## Not run:
  # Generate dummy RasterLayer
  r <- raster(nrows = 1, ncols = 1,
              xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 0.3)
  names(r) <- as.Date("2018-01-01")
  # Generate dummy RasterBrick
  b <- raster::brick(lapply(1:(365 * 3),
                  function(i) raster::setValues(r, runif(raster::ncell(r)))))
  names(b) <- seq.Date(from = as.Date("1993-01-01"),
                       to = as.Date("1995-12-31"),
                       by = "day")
  # Compute anomaly
  x <- anomaly(r, b)

  # This plots nicely using rasterVis::levelplot(), see example on GWIS
  # (\url{https://gwis.jrc.ec.europa.eu}
  rasterVis::levelplot(x, col.regions = colorRamps::matlab.like(n = 11))

## End(Not run)
```

`caliver` *caliver: CALIbration and VERification of gridded model outputs*

Description

Utility functions for the post-processing, calibration and validation of gridded model outputs. Initial test cases include the outputs of the following forest fire models: GEFF and RISICO.

`classify_index` *classify_index*

Description

This function classifies a fire danger index based on user defined danger thresholds. If `index` is one of GEFF's indices (fwi, ffmc, dmc, dc, isi, bui), the user does not need to specify thresholds and labels. Default thresholds and labels are those defined by JRC for Europe: https://effis.jrc.ec.europa.eu/about-effis/technical-background/fire-danger-forecast/ For instance if index = "fwi", default thresholds and labels are: very Low = 0-5.2, low = 5.2-11.2, moderate = 11.2-21.3, high = 21.3-38.0, very high = 38-50.0, extreme = > 50. However, if threshold != NULL, the default values will be overwritten by the custom values.

Usage

```
classify_index(r, index = NULL, thresholds = NULL, labels = NULL)
```

Arguments

r	is the Raster* object to classify.
index	this is the code of the fire danger index.
thresholds	numeric vector containing 5 thresholds.
labels	string of characters to be used as labels

Value

The function returns a Raster* object of the same dimensions of r but the values are categorical from 1 to 6 (corresponding to very low, low, moderate, high, very high and extreme danger, respectively). Some cells may contain NAs, this happens typically over the sea.

Examples

```
## Not run:
  r <- readRDS(system.file("extdata", "RISICO_raster.rds", package = "caliver"))
  x <- classify_index(r, index = "fwi")

  # This can be plotted using rasterVis::levelplot()
  rasterVis::levelplot(x)

## End(Not run)
```

---

daily_clima *daily_clima*

Description

This function generates daily climatological maps.

Usage

```
daily_clima(b, dates = NULL)
```

Arguments

b	RasterBrick/Stack containing the historical observations or a proxy (typically a reanalysis) that is used to derive the climatological information. This needs to contain daily layers for multiple years. names(b) should contain dates for comparison (e.g. X2017.01.01).
dates	Dates for which we need to calculate daily climatology. By default, this is a leap year.

Value

The function returns a RasterBrick (if dates contains one date) or a list of RasterBricks (if dates contains more than one date). Extent, resolution and land-sea mask match those of b. Values are the subset of b related to the given dates.

Examples

```
## Not run:
  # Generate dummy RasterBrick
  set.seed(0)
  r <- raster(nrows = 2, ncols = 2,
              xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 30)
  b <- raster::brick(lapply(1:(365 * 3),
       function(i) raster::setValues(r, runif(n = raster::ncell(r),
                                              min = 0, max = 100))))
  names(b) <- seq.Date(from = as.Date("1993-01-01"),
                       to = as.Date("1995-12-31"),
                       by = "day")
  daily_clima(b, as.Date("1996-01-01"))

## End(Not run)
```

---

get_fire_danger_levels

get_fire_danger_levels

Description

This function calculates the danger levels (VeryLow-Low-Moderate-High-VeryHigh-Extreme) for a given country.

Usage

```
get_fire_danger_levels(fire_index, ndays = 4)
```

Arguments

fire_index	RasterBrick containing the fire index to calculate the thresholds for. Please note that names(fire_index) should contain dates.
ndays	Number of days per year in which a fire is expected to occur. By default this is 4 days.

Value

A numeric vector listing the thresholds.

Examples

```
## Not run:

  # Generate dummy brick
  set.seed(0)
  r <- raster(nrows = 2, ncols = 2,
              xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 30)
  # Simulate a long reanalysis record
  b <- raster::brick(lapply(1:(365 * 3),
       function(i) raster::setValues(r, runif(n = raster::ncell(r),
                                              min = 0, max = 100))))
  names(b) <- seq.Date(from = as.Date("1993-01-01"),
                       to = as.Date("1995-12-31"),
                       by = "day")
  # Generate danger levels
  get_fire_danger_levels(fire_index = b)


## End(Not run)
```

`get_fire_season` *get_fire_season*

Description

Get the season for each date in a vector

Usage

```
get_fire_season(dates, fss = NULL, fse = NULL, zone = NULL)
```

Arguments

`dates`	vector of dates
`fss`	Fire Season Start (date in the format Y-m-d)
`fse`	Fire Season End (date in the format Y-m-d)
`zone`	this can either: "north", "south" or "tropics"

Value

A logical vector, where TRUE corresponds to a date in the fire season and FALSE correspond to a date not in the fire season.

Examples

```
## Not run:
  # Modify default seasons
  seasons <- get_fire_season(dates,
                             fss = as.Date("2012-04-01", format = "%Y-%m-%d"),
                             fse = as.Date("2012-10-31", format = "%Y-%m-%d"))

## End(Not run)
```

`get_percentile_map` *get_percentile_map*

Description

This function calculates percentile(s) at each grid point. Wrappers raster::calc.

Usage

```
get_percentile_map(r, probs)
```

Arguments

`r`	Raster* object (either RasterStack or RasterBrick). This could be the full record of daily indices or daily climatology.
`probs`	numeric vector of probabilities with values in the range [0, 1] listing which percentiles should be calculated.

Value

list containing all the generated percentile maps

Examples

```
## Not run:
 # Generate dummy RasterLayer
 r <- raster(nrows = 2, ncols = 2, xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 30)
 # Generate dummy RasterBrick
 b <- raster::brick(lapply(1:(365 * 3),
                    function(i) raster::setValues(r,
                      runif(n = raster::ncell(r), min = 0, max = 100))))
 # Get percentile maps
 get_percentile_map(b, probs = c(0.50, 0.75, 0.90, 0.99))

## End(Not run)
```

get_perc_risk_index *get_perc_risk_index*

Description

Generates the mean of the values over a certain percentile threshold for the portion of the Raster* that intersects a polygon

Usage

```
get_perc_risk_index(b, poly, perc_val = 75, mod = "gt")
```

Arguments

b	RasterLayer/Brick/Stack containing the historical observations or a proxy (typically a reanalysis).
poly	is the spatial polygon on which to aggregate the values
perc_val	is the percentile value used as a threshold
mod	defines if the values considered for the mean are above (gt) or below (lt) the threshold

Value

The function returns a numeric value (for each layer in b), corresponding to the mean of the values of b above/below a given percentile of the historical observations.

Examples

```
## Not run:
  # Read RISICO test data
  r_risico <- readRDS(system.file("extdata", "RISICO_raster.rds",
                                  package = "caliver"))
  # Set missing crs
  raster::crs(r_risico) <- "+proj=longlat +datum=WGS84 +no_defs"

  # Read dummy polygon
  shape <- as(raster::extent(6, 18, 35, 47), "SpatialPolygons")
  # Set missing crs
  raster::crs(shape) <- "+proj=longlat +datum=WGS84 +no_defs"

  get_perc_risk_index(b = r_risico, poly = shape, perc_val = 75, mod = "gt")


## End(Not run)
```

mask_crop_subset *mask_crop_subset*

Description

mask and/or crop a Raster* based on a Polygon.

Usage

```
mask_crop_subset(r, p, idx = NULL, ...)
```

Arguments

r	Raster* object
p	SpatialPolygon* object
idx	vector of strings indicating the layer indices to subset.
...	additional arguments as in writeRaster (e.g. progress = "text")

Details

Please note that cells along the border with centroids falling outside the polygon p will not be returned. If cells along the border are needed, we suggest to identify cells covering the polygon and set all remaining pixels to NA, as described in this post: https://goo.gl/22LwJt.

Value

A Raster* object with resolution and land-sea mask matching those of r and extent matching p.

Examples

```
## Not run:

  # Define dummy polygon
  shape <- as(raster::extent(7, 18, 37, 40), "SpatialPolygons")

  # Read RISICO test data
  r_risico <- readRDS(system.file("extdata", "RISICO_raster.rds",
                                  package = "caliver"))

  mask_crop_subset(r = r_risico, p = shape)


## End(Not run)
```

plot_pdf *plot_pdf*

Description

Plot PDF of fire index

Usage

```
plot_pdf(fire_index, thresholds, upper_limit = NULL, v_lines = NULL)
```

Arguments

fire_index	RasterBrick containing the fire index
thresholds	thresholds calculated using the function get_fire_danger_levels()
upper_limit	FWI upper limit to visualise (the default is the maximum FWI)
v_lines	named vector of values to plot as vertical lines (this can be quantiles for comparison)

Examples

```
## Not run:
  r <- readRDS(system.file("extdata", "RISICO_raster.rds", package = "caliver"))
  plot_pdf(r, thresholds = c(5.2, 11.2, 21.3, 38, 50))

## End(Not run)
```

plot_percentile_map *plot_percentile_map*

Description

This function plots the maps of percentiles

Usage

```
plot_percentile_map(
  maps,
  region = "GLOB",
  add_background = FALSE,
  col = NULL,
  ...
)
```

Arguments

maps	is the result of get_percentile_map()
region	string of characters describing the region.
add_background	logical, TRUE (default) to show background map. This only works if longitudes of maps are in the range [-180, +180]
col	custom color palette (default is ‘dput(rev(RColorBrewer::brewer.pal(n = 10, name = "RdYlGn")))‘)
...	additional graphical parameters inherited from plot() in the raster package.

Examples

```
## Not run:
  # Generate dummy RasterLayer
  r <- raster(nrows = 2, ncols = 2, xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 30)
  # Generate dummy RasterBrick
  b <- raster::brick(lapply(1:(365 * 3),
                    function(i) raster::setValues(r,
                    runif(n = raster::ncell(r), min = 0, max = 100))))
  # Get percentile maps
  maps <- get_percentile_map(b, probs = c(0.50, 0.75, 0.90, 0.99))

  # Use default palette
  plot_percentile_map(maps)

## End(Not run)
```

---

ranking *ranking*

Description

The ranking is applied to a forecast map r and provides percentiles of occurrence of the values based on a given climatology (see b).

Usage

```
ranking(r, b)
```

Arguments

r	is the RasterLayer to compare to the climatology.
b	RasterBrick/Stack containing the historical observations or a proxy (typically a reanalysis) that is used to derive the climatological information.

Details

The objects r and b should be comparable: same resolution and extent. More information on ranking is available here: https://bit.ly/2Qvekz4. To estimate fire climatology one can use hindcast or reanalysis data. Examples of the latter are available from Zenodo: https://zenodo.org/communities/wildfire.

Value

The function returns a RasterLayer with extent, resolution and land-sea mask matching those of r. Values are the percentiles of occurrence of the values.

Examples

```
## Not run:
  # Generate dummy RasterLayer
  r <- raster(nrows = 1, ncols = 1,
              xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 0.3)
  names(r) <- as.Date("2018-01-01")
  # Generate dummy RasterBrick
  b <- raster::brick(lapply(1:(365 * 3),
                  function(i) raster::setValues(r, runif(raster::ncell(r)))))
  names(b) <- seq.Date(from = as.Date("1993-01-01"),
                       to = as.Date("1995-12-31"),
                       by = "day")
  # Compute ranking
  x <- ranking(r, b)

  # This plots nicely using rasterVis::levelplot(), see example on GWIS
  # (\url{https://gwis.jrc.ec.europa.eu}
  rasterVis::levelplot(x, col.regions = c("green", "yellow", "salmon",
                                          "orange", "red", "black"))

## End(Not run)
```

---

read_risico_binary *read_risico_binary*

Description

Reads a RISICO output file and returns a raster map

Usage

```
read_risico_binary(filename)
```

Arguments

filename	is the file to read

Value

The function returns a RasterLayer or Brick, depending on whether `filenane` contains one or more layers.

Examples

```
## Not run:
  read_risico_binary(system.file("extdata", "RISICO_binary.bin", package = "caliver"))

## End(Not run)
```

`relative_humidity` *relative_humidity*

Description

Calculate relative humidity from 2m temperature (in Celsius) and 2m dew point temperature (in Celsius).

Usage

```
relative_humidity(t2m, d2m, unit = "Celsius", method = "August-Roche-Magnus")
```

Arguments

`t2m`	2m temperature (in Celsius)
`d2m`	2m dew point temperature (in Celsius)
`unit`	can be "Celsius" (default) or "Kelvin"
`method`	can be "August-Roche-Magnus" (default) or "Clausius-Clapeyron"

Value

The function returns a numeric, with length equal to t2m (and d2m).

Examples

```
## Not run:
  relative_humidity(t2m = 30, d2m = 25,
                    unit = "Celsius",
                    method = "August-Roche-Magnus")

## End(Not run)
```

subset_datacube *subset_datacube*

Description

This function subsets a datacube (RasterStack or RasterBrick) based on dates.

Usage

```
subset_datacube(r, from, to)
```

Arguments

r	is the Raster layer to compare to the climatology.
from	string, starting date (e.g. "2018-12-30").
to	string, ending date (e.g. "2018-12-31").

Details

If the from and to strings are in the format "YYYY-MM-DD", they are automatically converted into a date.

Value

The function returns a subset of r, with layer's date in the range starting with from and ending with to.

Examples

```
## Not run:
  r <- raster(nrows = 2, ncols = 2,
              xmn = 0, xmx = 360, ymn = -90, ymx = 90, vals = 30)
  # Generate dummy RasterBrick
  b <- raster::brick(lapply(1:(365 * 3),
                    function(i) raster::setValues(r,
                    unif(n = raster::ncell(r), min = 0, max = 100))))
  names(b) <- seq.Date(from = as.Date("1993-01-01"),
                       to = as.Date("1995-12-31"),
                       by = "day")
  subset_datacube(r = b, from = "1993-01-01", to = "1993-01-01")

## End(Not run)
```

validate_fire_danger_levels *validate_fire_danger_levels*

Description

This function compares observed and modelled fire data and return a contingency table summarising the hit rates, false alarms, misses and correct negatives. The validation can be done using various thresholds and input data.

Usage

```
validate_fire_danger_levels(
  fire_index,
  observation,
  fire_threshold,
  obs_threshold
)
```

Arguments

fire_index	RasterBrick containing the fire index (only one variable)
observation	RasterBrick containing the observation (only one variable)
fire_threshold	threshold to use to select relevant fire indices
obs_threshold	threshold to use to select relevant observations

Value

A list of two binary vectors: obs (observations) and pred (predictions).

Examples

```
## Not run:
 # Read example data
 r_risico <- readRDS(system.file("extdata", "RISICO_raster.rds",
                                 package = "caliver"))
 # Set missing crs
 raster::crs(r_risico) <- "+proj=longlat +datum=WGS84 +no_defs"

  # Generate obs and pred binary vectors
  validate_fire_danger_levels(fire_index = r_risico,
                              observation = r_risico * 2,
                              fire_threshold = 0.5,
                              obs_threshold = 0.5)


## End(Not run)
```

vdi *vdi*

Description

This function calculates the Vegetation Drought Index (defined by Meteo France), as a combination of Drought Code and Duff Moisture Code

Usage

```
vdi(dc, dmc)
```

Arguments

dc	is the Raster* containing the Drought Code.
dmc	is the Raster* containing the Duff Moisture Code.

Value

The function returns a categorical Raster* object. Values and their descriptions are listed below:

1. 1 = No fire vulnerability, corresponding to an important superficial humidification.
2. 2 = Very limited drying. Small fires possible.
3. 3 = The zone is considered vulnerable, due to strong wind and low humidity.
4. 4 = Important drying; the zone is considered vulnerable. Fires can occur in any conditions, excepts by high air moisture. Very severe Fire Weather Danger by moderate wind, even low wind with foehn effect or very hot and very dry air. Strong drougth rules are applied, the FWI is no longer appropriate.
5. 5 = Extreme drying, the zone is considered extremely vulnerable. Very big fire conditions are gathered. Permanent risk of very big fires on slope zones. Catastrophic fires are possible in any zones, by moderate or strong wind. Strong drought rules are applied, the FWI is no longer appropriate. The IPse works well and models very fast fire propagation speeds (sometimes underestimated).

The function returns a categorical Raster* object with extent, resolution and land-sea mask matching those of dc (or dmc). Values are integers in the range [1, 5].

Examples

```
## Not run:
  dc <- brick("dc.nc")
  dmc <- brick("dmc.nc")
  x <- vdi(dc, dmc)

## End(Not run)
```

Index