Package 'eva3dm'

April 22, 2026

Type Package

Title Evaluation of 3D Meteorological and Air Quality Models

Version 1.20

Date 2026-04-21

Description Provides tools for post-process, evaluate and visualize results from 3d Meteorological and Air Quality models against point observations (i.e. surface stations) and grid (i.e. satellite) observations.

Maintainer Daniel Schuch <underschuch@gmail.com>

License MIT + file LICENSE

Imports terra, ncdf4, utils

Suggests knitr, riem, rmarkdown, testthat (>= 3.0.0)

Encoding UTF-8

BugReports https://github.com/Schuch666/eva3dm/issues/

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

URL https://schuch666.github.io/eva3dm/

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Daniel Schuch [aut, cre] (ORCID: <https://orcid.org/0000-0001-5977-4519>)

Repository CRAN

Date/Publication 2026-04-22 03:30:02 UTC

Contents

`atr` *Read and write attributes on a NetCDF file*

Description

Read and write metadata information of a NetCDF files

Usage

```
atr(file = NA, var = "?", att = NA, action = "get", value = NA, verbose = TRUE)
```

Arguments

file	file name
var	variable name, 0 to global and "?" to show options
att	attribute names (NA for get all attnames)
action	"get" (default), "write" or "print" (return the value) of an attribute
value	value to write
verbose	display additional information

Value

string with the NetCDF attribute value

Examples

```
nc <- paste0(system.file("extdata",package="eva3dm"),'/wrfinput_d01')
atr(nc,0)
atr(nc,'Times')
atr(nc,'XLAT')
atr(nc,'XLONG')

atr(nc,'XLONG','MemoryOrder')
atr(nc,'XLONG','description')
atr(nc,'XLONG','units')
atr(nc,'XLONG','stagger')
atr(nc,'XLONG','FieldType')
```

---

calculate_column *Calculate column concentration of trace gases form WRF-Chem*

Description

Read output from WRF model and calculate the column of trace gases. The column concentration C_{column} is computed as:

$$C_{\text{column}} = \frac{N_A}{R} \int_{\text{surface}}^{\text{model\_top}} C \frac{P}{T}\, dz$$

where C is the pollutant concentration, N_A is Avogadro's number, R is the universal gas constant, P is the pressure [Pa], T is the temperature [K], and dz is the layer thickness [m].

Usage

```
calculate_column(
  file = file.choose(),
  name,
  met,
  DU,
  flip_v = FALSE,
  flip_h = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

file	WRF output file, see notes
name	trace gas name to be integrated
met	(optional) WRF output file for meteorological variables, see notes
DU	true to change the output units from 'molecules cm-1' to 'DU'
flip_v	passed to wrf_rast, see notes
flip_h	passed to wrf_rast, see notes
verbose	display additional information
...	extra arguments passed to eva3dm::wrf_rast or eva3dm::wrf_sds

Value

SpatRaster object (from terra package).

Note

files in file should contain Times, XLAT, XLONG variables in addition to the concentration to be integrated, the variable should include at least 3 dimensions including vertical.

met is a optional file, it should have containing PHB, PH, PB, P, and T variables with the same dimension of the concentration integrated.

post processing can affect the orientation of the variables, the arguments flip_v and flip_h and other arguments from eva3dm::wrf_rast and eva3dm::wrf_sds can be used to take effect into account.

Examples

```
file <- paste0(system.file("extdata",package="eva3dm"),"/wrf_column_o3_Boston.nc")
O3_column <- calculate_column(file,'o3', verbose = TRUE)
```

cate *Calculate categorical statistics in related to a threshold*

Description

Calculate traditional statistics related to a threshold

Usage

```
cate(
  model,
  observation,
  threshold,
  cutoff = NA,
  nobs = 8,
  rname,
  to.plot = FALSE,
  col = "#4444bb",
  pch = 19,
  lty = 3,
  lcol = "#333333",
  lim,
  verbose = TRUE,
  ...
)
```

Arguments

model	numeric vector with paired model data
observation	numeric vector with paired observation data
threshold	reference value
cutoff	(optionally the maximum) valid value for observation
nobs	minimum number of observations
rname	row name
to.plot	TRUE to plot a scatter-plot
col	color for points
pch	pch of points
lty	lty of threshold lines
lcol	col of threshold lines
lim	limit for x and y
verbose	display additional information
...	arguments passed to plot

Value

a data.frame including: Accuracy (A); Critical Success Index (CSI); Probability of Detection (POD); Bias(B); False Alarm Ratio (FAR); Heidke Skill Score (HSS); Pearce skill Score (PSS) in

References

Gandin, L. S., & Murphy, A. H. (1992). Equitable skill scores for categorical forecasts. Monthly weather review, 120(2), 361-370.

Examples

```
data <- 0.02 * 1:100
set.seed(666)
model  <- abs(rnorm(100,0.01))

oldpar <- par(pty="s")
cate(model = model, observation = data, threshold = 1,
     to.plot = TRUE, rname = 'example')
par(oldpar)
```

---

daily *Calculate daily mean, min or max*

Description

function to calculate daily mean, min or max of a data.frame

Usage

```
daily(
  data,
  time = "date",
  stat = mean,
  min_offset = 0,
  hour_offset = 0,
  numerical = TRUE,
  verbose = TRUE
)
```

Arguments

data	data.frame with time column and variable columns to be processed
time	name of the time column (default is date) in POSIXct
stat	function of the statistics to calculate (default is mean)
min_offset	minutes of observation from previous hour (default is 0)
hour_offset	hours of observation from previous day (default is 0)

numerical	TRUE (default) include only numerical columns
verbose	display additional information

Value

data.frame with time and the daily mean, min or max

Examples

```
# in case there is connection issue
load_data <- function(cond) {
  message(paste("conection issue, loading pre-downloaded data"))
  DATA <- readRDS(paste0(system.file("extdata",package="eva3dm"),
                         "/riem_OAKB_jan_2012.Rds"))
  return(DATA)
}

sites <- c("OAKB")
for(site in sites){
  cat('Trying to download METAR from:',site,'...\n')
  DATA <- tryCatch(riem::riem_measures(station    = sites,
                                       date_start = "2012-01-01",
                                       date_end   = "2012-02-01"),
                   error = load_data)
}
data_daily_mean <- daily(DATA,time = 'valid')
data_daily_min  <- daily(DATA[1:7],time = 'valid',stat = min)
data_daily_max  <- daily(DATA[1:7],time = 'valid',stat = max)
```

---

eva *Model statistical evaluation*

Description

Statistical (or categorical) evaluation from 2 data.frames. The input data.frames (model and observation) must contain a time column (containing POSIXlt). The function perform pre-conditioning of the data, data pairing (using the time column) of the observations and model and calculate the metrics for statistical or categorical evaluation.

Usage

```
eva(
  mo,
  ob,
  rname = site,
  table = NULL,
  site = "ALL",
  wd = FALSE,
```

```
  fair = NULL,
  cutoff = NA,
  cutoff_NME = NA,
  no_tz = FALSE,
  nobs = 8,
  eval_function = stat,
  select_time,
  time = "date",
  remove_ch = FALSE,
  clean = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

mo	data.frame with model data
ob	data.frame with observation data
rname	row name of the output (default is site argument)
table	data.frame to append the results
site	name of the station or "ALL" (default) or "complete", see notes
wd	default is FALSE, see notes
fair	model data.frame (or list of names) to perform a fair comparison, see notes
cutoff	minimum (optionally the maximum) valid value for observation
cutoff_NME	minimum (optionally the maximum) valid value for observation for NME
no_tz	ignore tz from input (force GMT)
nobs	minimum number of valid observations, default is 8
eval_function	evaluation function (default is stat)
select_time	select the observation (ob) using time from model (mo) data.frame
time	name of the time column (containing time in POSIXct)
remove_ch	remove special characters on column names
clean	remove rows when number of observations < nobs, for site="complete"
verbose	display additional information
...	arguments to be passing to stats and plot

Value

data.frame with statistical metric

Note

Fair argument can be a data.frame or a character string to be used for the analysis, alternatively the function

If wd = TRUE, used for wind direction, a rotation of 360 (or -360) is applied to minimize the wind direction difference.

If site == 'ALL' (default) all the columns from observations are combined in one column (same for observation) and all the columns are evaluated together.

If site == 'complete' a internal loop, calls recursively eva() to evaluate all sites in the first argument (model) and using all sites (see "ALL").

Special thanks to Kiarash and Libo to help to test the wind direction option.

See Also

stat for additional information about the statistical metrics and cate for categorical metrics, and check the example with the custom evaluation function (inclusion of p.value from stats::cor.test()).

Examples

```
model <- readRDS(paste0(system.file("extdata",package="eva3dm"),
                        "/model.Rds"))
obs   <- readRDS(paste0(system.file("extdata",package="eva3dm"),
                        "/obs.Rds"))

# if there is no observed data
# the function return an empty row
table <- eva(mo = model, ob = obs, site = "VVIbes")
print(table)

# if the site are not in the input data frame a message is displayed
# and the function return an empty row
table <- eva(mo = model, ob = obs, site = "Ibirapuera")
print(table)

# calculating statistical with a few observed values
table <- eva(mo = model, ob = obs, site = "Americana")
print(table)

# calculating categorical (using 2 for threshold) with a few observed values
table <- eva(mo = model, ob = obs, site = "Americana",
             eval_function = cate, threshold = 2)
print(table)

# calculating categorical (using 10 for threshold) with a few observed values
table <- eva(mo = model, ob = obs, site = "Americana",
             eval_function = cate, threshold = 10)
print(table)

# customizing the evaluation function: inclusion of p.value from stats::cor.test()
stat_p <- function(x, y, ...){
  table         <- eva3dm::stat(x, y, ...)
```

```
  cor.result    <- stats::cor.test(x, y, ... )
  table$p.value <- cor.result$p.value
  table         <- table[,c(1:4,12,5:11)]
  return(table)
}

table <- eva(mo = model, ob = obs, site = "Americana",eval_function = stat_p)
print(table)
```

`extract_max_8h` *Create a NetCDF file with the surface maximum of O3*

Description

Read the values from o3 and T2, convert o3 to ug m-3 and calculate the maximum of 8-hour moving avarage from a list of files.

Usage

```
extract_max_8h(
  filelist,
  variable = "o3",
  field = "4d",
  prefix = "max_8h",
  units = "ug m-3",
  meta = TRUE,
  filename,
  verbose = TRUE
)
```

Arguments

`filelist`	list of files to be read
`variable`	variable name
`field`	'4d' (default), '3d', '2d' or '2dz' see notes
`prefix`	to output file, default is serie
`units`	units on netcdf file (default is ug m-3), change to skip unit conversion
`meta`	use Times, XLONG and XLAT data (only works with 2d variable for file)
`filename`	name for the file, in this case prefix is not used
`verbose`	display additional information

Value

No return value

Note

The field argument '4d' / '2dz' is used to read a 4d/3d variable droping the 3rd dimention (z).

Examples

```
dir.create(file.path(tempdir(), "MDA8"))
folder <- system.file("extdata",package="eva3dm")
wrf_file <- paste0(folder,"/test_small_o3.nc")
extract_max_8h(filelist = wrf_file,
               prefix = paste0(file.path(tempdir(),"MDA8"),'/mean'),
               field = '3d')
```

`extract_mean` *Create a NetCDF file with the surface mean*

Description

Read and calculate the mean value of a variable from a list of wrf output files.

Usage

```
extract_mean(
  filelist,
  variable = "o3",
  field = "4d",
  prefix = "mean",
  units = "ppmv",
  meta = TRUE,
  filename,
  verbose = TRUE
)
```

Arguments

`filelist`	list of files to be read
`variable`	variable name
`field`	'4d' (default), '3d', '2d' or '2dz' see notes
`prefix`	to output file, default is serie
`units`	units on netcdf file (default is ppmv)
`meta`	use Times, XLONG and XLAT data (only works with 2d variable for file)
`filename`	name for the file, in this case prefix is not used
`verbose`	display additional information

Value

No return value

Note

The field argument '4d' / '2dz' is used to read a 4d/3d variable droping the 3rd dimention (z).

Examples

```
dir.create(file.path(tempdir(), "MEAN"))
folder <- system.file("extdata",package="eva3dm")
wrf_file <- paste0(folder,"/wrf.day1.o3.nc")
extract_mean(filelist = wrf_file,prefix = paste0(file.path(tempdir(),"MEAN"),'/mean'))
```

---

extract_serie *Extract time series of wrf file list of lat/lon*

Description

Read and extract data from a list of wrf output files and a table of lat/lon points based on the distance of the points and the center of model grid points, points outside the domain (and points on domain boundary) are not extracted.

Usage

```
extract_serie(
  filelist,
  point,
  variable = "o3",
  field = "4d",
  level = 1,
  prefix = "serie",
  new = "check",
  return.nearest = FALSE,
  fast = FALSE,
  use_ij = FALSE,
  model = "WRF",
  id = "id",
  remove_ch = FALSE,
  verbose = TRUE
)
```

Arguments

filelist	list of files to be read
point	data.frame with lat/lon
variable	variable name
field	dimension of the variable, '4d' (default), '2dz', '2dz', '2d' or 'xyzt', 'xyz', 'xyz', and 'xy' see notes

Field	Dimensions	Example
3dt	xyzt	WRF dimensions for 3D array with multiple times
2dt	xyt	WRF dimensions for 2D array with multiple times
2dz	xyz	WRF dimensions for 3D array with single time
2d	xy	WRF dimensions for 2D array with single time

level	model level (index) to be extracted
prefix	to output file, default is serie
new	TRUE, FALSE of 'check' see notes
return.nearest	return the data.frame of nearest points instead of extract the serie
fast	faster calculation of grid distances but less precise
use_ij	logical, use i and j from input instead of calculate
model	"WRF" (default), "CAMx" (CAMx, CMAQ and smoke files), and "WACCM" (WACCM and CAM-Chem)
id	name of the column with station names, if point is a SpatVector (points) from terra package
remove_ch	remove special characters on row.names
verbose	display additional information

Value

No return value

Note

The field argument '4d' or '2dz' is used to read a 4d/3d variable droping the 3rd dimension (z), this should be based how ncdf4 R-package reads the model output.

new = TRUE create a new file, new = FALSE append the data in a old file, and new = 'check' check if the file exist and append if the file exist and create if the file doesnt exist

The option field '3d' was removed, a new option should be used instead (2dt or 2dz).

The site-list of two global data-sets (METAR/ISD and AERONET) are provided on examples and site-list for stations on Brazil (INMET and Air Quality stations). Site-lists for other regions (US, Canada, Europa, etc) are provided as additional examples.

Examples

```
cat('Example 1: METAR site list\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_METAR.Rds"))

cat('Example 2: Integrated Surface Dataset (ISD) site list\n')
# row.names are a combination of State Code,County Code and Site Number (2,3 and 4 digits)
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_ISD.Rds"))

cat('Example 4: AERONET site list\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_AERONET.Rds"))

cat('Example 5: list of INMET stations in Brazil\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_INMET.Rds"))

cat('Example 6: list of Air Quality stations in Brazil\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_AQ_BR.Rds"))

cat('Example 7: list of AQM stations in US\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_AQS.Rds"))

cat('Example 8: list of CASTNET stations in rural areas of US/Canada\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_CASTNET.Rds"))

cat('Example 9: list of Longterm European Program (EMEP) stations\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_EMEP.Rds"))

cat('Example 10: list of FLUXNET stations\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_FLUXNET.Rds"))

cat('Example 11: list of IMPROVE stations\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_IMPROVE.Rds"))

cat('Example 12: list of TOAR stations\n')
sites <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_TOAR.Rds"))

files    <- dir(path = system.file("extdata",package="eva3dm"),
               pattern = 'wrf.day',
               full.names = TRUE)
dir.create(file.path(tempdir(),"SERIE"))
folder <- file.path(tempdir(),"SERIE")

# extract data for 3 locations
extract_serie(filelist = files, point = sites[1:3,],prefix = paste0(folder,'/serie'))
```

`extract_surgical` *Extract time series from a list model files using time UTC, latitude, longitude, and altitude (meters).*

Description

Read output from a list of model (WRF or UFS) and calculate the column of trace gases.

Usage

```
extract_surgical(
  filelist,
  point,
  vars,
  model = "WRF",
  tol = 60,
  boundary = 5,
  include_indices = TRUE,
  include_model = TRUE,
  include_distances = TRUE,
  filename,
  verbose = TRUE
)
```

Arguments

filelist	list of files to be read
point	data.frame with time (POSIXct in UTC), lat (numeric in degrees), lon (numeric in degrees), alt (numeric in meters)
vars	variable names
model	'WRF' (default) or 'UFS'
tol	tolerance in seconds
boundary	number of grid points that are considered
include_indices	to include t,i,j,k indices
include_model	to include model time, latitude, longitude and altitude
include_distances	include the distance from the center of the grid to the observations
filename	name of a file to save the dataframe
verbose	display additional information

Value

data.frame

get_distances *Get the distance in kilometers between two points*

Description

Get the distance in kilometers between two points

Usage

```
get_distances(lat1, long1, lat2, long2, R = 6371)
```

Arguments

lat1	Latitude in decimals
long1	Longitude in decimals
lat2	Latitude in decimals
long2	Longitude in decimals
R	Radius of the earth in kmdescription (R=6371)

Value

A numeric vector with the distance in kilometers.

#' source: https://github.com/gustavobio/brclimate/blob/master/R/get_distances.R

hourly *Calculate hourly mean, min or max*

Description

function to calculate hourly mean, min or max of a data.frame

Usage

```
hourly(
  data,
  time = "date",
  stat = mean,
  min_offset = 30,
  numerical = TRUE,
  verbose = TRUE
)
```

Arguments

data	data.frame with time column and variable columns to be processed
time	name of the time column (default is date) in POSIXct
stat	function of the statistics to calculate (default is mean)
min_offset	minutes of observation from previous hour (default is 30)
numerical	TRUE (default) includes only numerical columns
verbose	display additional information

Value

data.frame including only numerical columns

data.frame with time and the hourly mean, min or max

Examples

```
# in case there is connection issue
load_data <- function(cond) {
  message(paste("conection issue, loading pre-downloaded data"))
  DATA <- readRDS(paste0(system.file("extdata",package="eva3dm"),
                         "/riem_OAHR_jan_2012.Rds"))
  return(DATA)
}

sites <- c("OAHR")
for(site in sites){
  cat('Trying to download METAR from:',site,'...\n')
  DATA <- tryCatch(riem::riem_measures(station    = sites,
                                       date_start = "2012-01-01",
                                       date_end   = "2012-02-01"),
                   error = load_data)
}
data_hourly_mean <- hourly(DATA,time = 'valid')
data_hourly_min  <- hourly(DATA[1:7],time = 'valid',stat = min)
data_hourly_max  <- hourly(DATA[1:7],time = 'valid',stat = max)
```

---

interp *Interpolation (project and resample)*

Description

function to project and interpolate rast

Usage

```
interp(x, y, method = "bilinear", mask, verbose = FALSE)
```

Arguments

x	rast to be interpolated
y	target rast of the interpolation
method	passed to terra::resample
mask	optional SpatVector to mask a region of the data
verbose	display additional information (not used)

Value

SpatRaster (terra package)

Examples

```
model_o3 <- terra::rast(paste0(system.file("extdata",package="eva3dm"),
                               "/camx_no2.Rds"))
omi_o3   <- terra::rast(paste0(system.file("extdata",package="eva3dm"),
                               "/omi_no2.Rds"))

# interpolate omi O3 column to model grid
omi_o3c_interp_model <- interp(omi_o3,model_o3)

# interpolate model o3 column to omi grid
model_o3c_interp_omi <- interp(omi_o3,model_o3)
```

---

legend_range *Plot a legend with the range of values*

Description

Plot a legend with the range of values

Usage

```
legend_range(
  x,
  y,
  text.width = NULL,
  dig = c(2, 2, 2),
  xjust = 0.005,
  yjust = 0.95,
  horiz = TRUE,
  y.intersp = 0.5,
  x.intersp = 0.5,
  show.mean = TRUE,
  unit = "",
```

```
  label_mean = "ALL:",
  ...
)
```

Arguments

x	rast or array
y	rast or array to mean (x is used only for the range in this case)
text.width	Longitude in decimals
dig	vector with number of digits for plot
xjust	passed to legend
yjust	passed to legend
horiz	passed to legend
y.intersp	passed to legend
x.intersp	passed to legend
show.mean	set TRUE to hide mean value
unit	a string for units
label_mean	label in case y is provided
...	extra arguments passed to legend

Value

No return value

Note

for use with rast use before any change of projection

text.width can vary depending on map dimensions

Examples

```
x <- 1:10 + rnorm(10,sd = .4)
plot(x,ty='l')
legend_range(x)
```

ma8h *Calculate 8-hour moving average*

Description

function to calculate Ozone 8-hour moving average for a data.frame

Usage

```
ma8h(data, time = "date", var, n = 8, verbose = TRUE, ...)
```

Arguments

data	data.frame with time column and variable columns to be processed
time	name of the time column (default is date) in POSIXct
var	name of the columns to be calculated
n	custom time window (n = 8 for 8-hour average)
verbose	display additional information
...	parameters passed to hourly

Value

data.frame with time and the 8-hour moving average

See Also

mda8 for Maximum Daily 8-hour moving average

Examples

```
model_file <- paste(system.file("extdata", package = "eva3dm"),
                                "/model_o3_ugm3_36km.Rds", sep="")
model      <- readRDS(model_file)
model_8h   <- ma8h(model)
plot(model$date,model$Campinas, pch = 19,
     main = expression(O[3]~~'['*mu*g*m^-3*']'))
points(model_8h$date,model_8h$Campinas, col = 'blue', pch = 19)
legend('topleft',bty = 'n',
       pch = 19,
       legend = c('hourly','8h-mov average'),
       col = c('black','blue'))
```

mda8 *Maximum Daily 8-hr Average*

Description

function to calculate Ozone Maximum Daily 8-hr Average or 8-hr moving Average for a data.frame

Usage

```
mda8(data, time = "date", var, verbose = TRUE)
```

Arguments

data	data.frame with time column and variable columns to be processed
time	name of the time column (default is date) in POSIXct
var	name of the columns to be calculated
verbose	display additional information

Value

data.frame with time and the maximum daily 8-hr average

See Also

ma8h for 8-hour Moving Average

Examples

```
model_file <- paste(system.file("extdata", package = "eva3dm"),
                    "/model_o3_ugm3_36km.Rds", sep="")
model      <- readRDS(model_file)
model_mda8 <- mda8(model)
model_8h   <- ma8h(model)
plot(model$date,model$Campinas, pch = 19,
     main = expression(O[3]~~'['*mu*g*m^-3*']'))
points(model_8h$date,model_8h$Campinas, col = 'blue', pch = 19)
points(model_mda8$date + 17*60*60,model_mda8$Campinas,
       col = 'red', pch = 4, cex = 2)
legend('topleft',bty = 'n',
       pch = c(19,19,4),
       legend = c('hourly','8h-mov average','MD8A'),
       col = c('black','blue','red'))
```

monthly *Calculate monthly mean, min or max*

Description

function to calculate monthly mean, min or max of a data.frame

Usage

```
monthly(
  data,
  time = "date",
  stat = mean,
  min_offset = 0,
  hour_offset = 0,
  days_offset = 0,
  numerical = TRUE,
  verbose = TRUE
)
```

Arguments

data	data.frame with time column and variable columns to be processed
time	name of the time column (default is date) in POSIXct
stat	function of the statistics to calculate (default is mean)
min_offset	minutes of observation from previous month (default is 0)
hour_offset	hours of observation from previous month (default is 0)
days_offset	day of observation from previous month (default is 0)
numerical	TRUE (default) include only numerical columns
verbose	display additional information

Value

data.frame with time and the monthly mean, min or max

Examples

```
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-12-31',tz = 'UTC'),
             by = 'hour')

DATA <- data.frame(date = times,
                   var1 = rnorm(n = length(times), mean = 1,sd = 1),
                   var2 = rnorm(n = length(times), mean = 2,sd = 0.5),
                   var3 = rnorm(n = length(times), mean = 3,sd = 0.25))
```

```
data_month_mean <- monthly(DATA)
data_month_min  <- monthly(DATA,stat = min)
data_month_max  <- monthly(DATA,stat = max)
```

---

ncdump *Print a 'ncdump -h' command*

Description

Read a NetCDF and print the medatada

Usage

```
ncdump(file = file.choose())
```

Arguments

file	file name

Value

No return value, only display information

Examples

```
ncdump(file = paste0(system.file("extdata",package="eva3dm"),
                     '/wrfinput_d01'))
```

---

overlay *Plot or add points using a color scale*

Description

Custon plot for SpatRaster (terra R-package) object based on terra package

Usage

```
overlay(
  p,
  z,
  col,
  col2,
  lim = range(z, na.rm = TRUE),
  symmetry = TRUE,
  pch = 19,
```

```
  pch2 = NA,
  cex = 1,
  cex2 = 1.2 * cex,
  outside = TRUE,
  add = FALSE,
  plg = list(tic = "none", shrink = 1),
  pax = list(),
  unit,
  expand = 1.15,
  ...
)
```

Arguments

p	SpatVector points
z	column name or a vector of values to plot
col	color for the point
col2	color for the outline
lim	range of values for scale
symmetry	calculate symmetrical scale
pch	type of point
pch2	type of point for contour
cex	character expansion for the points
cex2	character expansion for the contour
outside	to include values outside range
add	add to existing plot
plg	list of parameters passed to terra::add_legend
pax	list of parameters passed to graphics::axis
unit	used title in terra::add_legend
expand	to expand the plot region
...	arguments to be passing to terra::plot

Value

No return value

Examples

```
sp<- terra::vect(paste0(system.file("extdata",package="eva3dm"),"/masp.shp"))
BR<- terra::vect(paste0(system.file("extdata",package="eva3dm"),"/BR.shp"))

p    <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/sites_AQ_BR.Rds"))
p$id      <- row.names(p)
point     <- terra::vect(p)
point$NMB <- 1:45 - 20 # some values to plot
```

```
terra::plot(BR, main = 'add points',xlim = c(-52,-37),ylim = c(-25,-18))
terra::lines(BR)
terra::lines(sp, col = 'gray')
overlay(point,point$NMB,cex = 1.4, add = TRUE)

overlay(point,point$NMB,cex = 1.4, add = FALSE, main = 'new plot')
terra::lines(BR)
terra::lines(sp, col = 'gray')
```

`plot_diff` *Plot the difference from two SpatRaster objects*

Description

Custom difference (x - y) plots for SpatRaster object (based on terra package)

Usage

```
plot_diff(
  x,
  y,
  col,
  absolute = FALSE,
  relative = FALSE,
  lim_a = NA,
  lim_r = NA,
  scale,
  unit = c(units(x), expression("%")),
  fill = FALSE,
  ...
)
```

Arguments

`x`	SpatVector points
`y`	values to plot
`col`	color
`absolute`	to plot absolute difference
`relative`	to plot relative difference
`lim_a`	range of values for absolute scale
`lim_r`	range of values for relative scale
`scale`	variable multiplier for absolute difference
`unit`	annotation for units
`fill`	filling NAs
`...`	arguments to be passing to plot_raster

Value

No return value

Examples

```
folder <- system.file("extdata",package="eva3dm")
wrf    <- paste0(folder,"/wrfinput_d01")
A      <- wrf_rast(wrf,'XLAT')
terra::units(A) <- 'degrees'
B      <- wrf_rast(wrf,'XLONG')
plot_diff(A,B,int = 2)
```

---

plot_rast *Plot rast (SpatRaster) object*

Description

Custon plot for SpatRaster (terra R-package) object based on terra package

Usage

```
plot_rast(
  r,
  color,
  ncolor = 21,
  proj = FALSE,
  plg = list(tic = "none", shrink = 1),
  pax = list(),
  latitude = TRUE,
  longitude = TRUE,
  int = 10,
  grid = FALSE,
  grid_int = int,
  grid_col = "#666666",
  grid_lwd = 1.2,
  add_range = FALSE,
  show.mean = TRUE,
  ndig = 2,
  log = FALSE,
  range,
  scale,
  min = -3,
  max,
  unit,
  mask,
  fill = FALSE,
  ...
)
```

Arguments

r	raster
color	color scale, or name of a custom color scale (see notes)
ncolor	number of colors
proj	TRUE to project the raster to lat-lon
plg	list of parameters passed to terra::add_legend
pax	list of parameters passed to graphics::axis
latitude	add a latitude axis
longitude	add a longitude axis
int	interval of latitude and longitude lines
grid	add grid (graticule style)
grid_int	interval of grid lines
grid_col	color for grid lines
grid_lwd	lwd for the grid lines
add_range	add legend with max, average and min r values
show.mean	show the average on legend, default TRUE
ndig	number of digits for legend_range
log	TRUE to plot in log-scale
range	range of original values to plot
scale	variable multiplier (not affect min/max/range)
min	minimum log value for log scale (default is -3)
max	maximum log value for log scale
unit	title for color bar
mask	optional SpatVector to mask the plot
fill	filling NAs
...	arguments to be passing to terra::plot

Value

No return value

Note

color scales including: 'eva3', 'eva4', 'blues', 'diff', 'rain', 'pur', 'blackpur', 'acid', and 'regime'. Also reverse version with addition of a r ('eva3r' is the default).

Examples

```
wrf <- paste(system.file("extdata", package = "eva3dm"),
                          "/wrfinput_d01", sep="")

r <- wrf_rast(file=wrf, name='XLAT')

plot_rast(r)
```

q2rh *Convert absolute humidity to relative humidity*

Description

function to convert absolute humidity to relative humidity.

Usage

```
q2rh(q, t = 15, p = 101325)
```

Arguments

q	vector (or data.frame) of absolute humidity (in g/Kg)
t	vector (or data.frame) of temperature (in Celcius)
p	vector (or data.frame) of pressure (in Pa)

Value

vector or data.frame with time and the relative humidity, units are

Note

default values are from standard atmosphere (288.15 K (15C) / 101325 Pa)

if rh and temp arguments are data.frame, both need to have the same number of lines and columns, first column (time column) will be ignored.

Examples

```
# for a single value (or same length vectors)
q2rh(q = 0.0002038, t = 29.3, p = 100800)

# using all data.frames
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-01-02',tz = 'UTC'),
             by = 'hour')[1:5]
q2   <- data.frame(time = times, a = rep(0.0002038,5))
temp <- data.frame(time = times, a = rep(     29.3,5))
pres <- data.frame(time = times, a = rep(   100800,5))
q2rh(q = q2, t = temp, p = pres)

# using data.frame for q and t (p is cte.)
q2rh(q = q2, t = temp, p = 100000)

# using data.frame for q and p (t is cte.)
q2rh(q = q2, t = 26, p = pres)

# using data.frame only for q (p and t are cte.)
```

```
q2rh(q = q2, t = 26, p = 100000)
```

`rain` *conversion of model precipitation to hourly precipitation*

Description

function that converts model accumulated precipitation to hourly precipitation.

Usage

```
rain(rainc, rainnc, verbose = TRUE)
```

Arguments

`rainc`	data.frame or SpatRaster with RAINC variable
`rainnc`	data.frame or SpatRaster with RAINNC variable
`verbose`	set TRUE to display additional information

Value

data.frame time and the hourly precipitation or SpatRaster hourly precipitation

Examples

```
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-01-01 04:00:00',tz = 'UTC'),
             by = 'hour')
RNC   <- data.frame(date = times, aa = c(0.149,0.149,0.149,0.149,0.149))
RNNC  <- data.frame(date = times, aa = c(0.919,1.0,1.1,1.1,2.919))
rain(rainc = RNC, rainnc = RNNC)
```

`rast_to_netcdf` *Function to convert/save a SpatRaster array/Netcdf*

Description

Conversion of SpatRaster to array and optionally save on a existing Netcdf File.

Usage

```
rast_to_netcdf(r, file, name, unit = units(r), XY = FALSE, verbose = TRUE)
```

Arguments

r	SpatRaster object
file	Netcdf file name
name	variable name on a Netcdf file
unit	unit of the variable (set to NA to don't change unit)
XY	set to true if MemoryOrder is XY (only if file is missing)
verbose	display additional information

Value

numerical array

Note

eva3dm::wrf_rast support 3d SpatRaster, in case of a 4d variable use other approach to save on file.

Examples

```
folder   <- system.file("extdata",package="eva3dm")
wrf_file <- paste0(folder,"/wrf.day1.o3.nc")

Rast     <- wrf_rast(wrf_file,'o3')
A        <- rast_to_netcdf(Rast)
```

---

read_stat *Function to read stats and evaluation*

Description

Function to read stats and evaluation output

Usage

```
read_stat(file, sep = ";", dec = ".", verbose = FALSE, ...)
```

Arguments

file	model data.frame
sep	the field separator string, passed to read.table function
dec	he string to use for decimal points, passed to read.table function
verbose	display additional information
...	arguments passed to read.table functions

Value

No return value

Examples

```
sample <- read_stat(file   = paste0(system.file("extdata", package = "eva3dm"),"/sample.txt"),
                    verbose = TRUE)

sample <- read_stat(file   = paste0(system.file("extdata", package = "eva3dm"),"/sample.csv"),
                    verbose = TRUE)
```

---

rh2q *Convert relative humidity to absolute humidity*

Description

function to convert humidity to absolute humidity using Tetens formula, assuming standard atmosphere conditions.

Usage

```
rh2q(rh, temp = 15)
```

Arguments

rh	vector (or data.frame) of relative humidity (in percentage)
temp	vector (or data.frame) of temperature (in Celsius)

Value

value of data.frame with time and the absolute humidity, units are g/g

Note

default values are from standard atmosphere (288.15 K / 15 C)

if rh and temp arguments are data.frame, both need to have the same number of lines and columns, first column (time column) will be ignored.

Examples

```
# for a singfle value
rh2q(rh = 99, temp = 25)

# vector of rh values
rh2q(rh = c(0,seq(1,100, by = 4)), temp = 25)

# vector of values for rh and temp
```

```
rh2q(rh = c(0,seq(1,100, by = 4)), temp = 10:35)

# rh is data.frame and temp is a value
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-01-02',tz = 'UTC'),
             by = 'hour')
rh2q(rh   = data.frame(time = times, a = seq(1,100, by = 4)),temp = 25)

# using both rh and temp are data.frames
rh2q(rh   = data.frame(time = times, a = seq(1,100, by = 4)),
     temp = data.frame(time = times, a = 11:35))
```

`sat` *Functions to model evaluation using satellite*

Description

functions to evaluate the spatial performance using satellite

Usage

```
sat(
  mo,
  ob,
  rname,
  table = NULL,
  n = 6,
  min = NA,
  max = NA,
  scale,
  method = "bilinear",
  eval_function = stat,
  mask,
  skip_interp = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

`mo`	SpatRaster or raster with model
`ob`	SpatRaster or raster with observations
`rname`	passed to stat
`table`	data.frame to append the results
`n`	number of points from the boundary removed, default is 5
`min`	minimum value cutoff

max	maximum value cutoff
scale	multiplier for model and observation (after min/max cutoff)
method	passed to terra::resample
eval_function	evaluation function (default is stat)
mask	optional SpatVector to mask the results
skip_interp	skip the interpolation step
verbose	set TRUE to display additional information
...	other arguments passed to stat

Value

a data.frame

Note

If a YOU DIED error message appears, means you are removing all the valid values using the arguments min or max.

If cate() is used for eval_function, the argument threshold must be included (see example).

Examples

```
model_no2 <- terra::rast(paste0(system.file("extdata",package="eva3dm"),
                                "/camx_no2.Rds"))
omi_no2   <- terra::rast(paste0(system.file("extdata",package="eva3dm"),
                                "/omi_no2.Rds"))

# generate the statistical indexes
sat(mo = model_no2,ob = omi_no2,rname = 'NO2_statistical')

# generate categorical evaluation using 3.0 as threshold
sat(mo = model_no2,ob = omi_no2,rname = 'NO2_categorical',
    eval_function = cate, threshold = 3.0)

# customizing the evaluation function: inclusion of p.value from stats::cor.test()
stat_p <- function(x, y, ...){
  table         <- eva3dm::stat(x, y, ...)
  cor.result    <- stats::cor.test(x, y, ... )
  table$p.value <- cor.result$p.value
  table         <- table[,c(1:4,12,5:11)]
  return(table)
}

sat(mo = model_no2,ob = omi_no2,rname = 'NO2_statistical_with_p',eval_function = stat_p)
```

`select` *Selection from data.frames with time-series*

Description

Utility function to select periods from a data.frame. This function is inspired by openair::selectByDate.

Usage

```
select(
  data,
  year,
  month,
  day,
  hour,
  minutes,
  seconds,
  julian,
  start,
  end,
  range,
  time = "date"
)
```

Arguments

`data`	data.frame with model or observation data
`year`	numeric vector for selection
`month`	numeric vector (1-12) for selection, can be abbreviated to 3 or more letters
`day`	numeric vector (1-31) for selection, weekdays can be abbreviated to 3 or more letters, or weekday/weekend
`hour`	numeric vector (0-23) for selection
`minutes`	numeric vector (0-60) for selection
`seconds`	numeric vector (0-60) for selection
`julian`	Julian day (1-366)
`start`	POSIXct or character (YYYY-MM-DD) with the initial date of selection
`end`	POSIXct or character (YYYY-MM-DD) with the initial date of selection
`range`	pair of start/end or a data.frame with time (default is "date")
`time`	name of the column for time (default is "date")

Value

data.frame

See Also

See %IN% for selection based on position and model domains.

Examples

```
model <- readRDS(paste0(system.file("extdata",package="eva3dm"),
                        "/model.Rds"))
summary(model)
summary(select(data = model, start = '2012-01-09'))
summary(select(data = model, start = '2012-01-05', end  = '2012-01-09'))
summary(select(data = model, day  = 6))
summary(select(data = model, hour = 12))
summary(select(data = model, day = 6, hour = 12))
summary(select(data = model, day  = 'weekday'))
summary(select(data = model, day  = 'weekend'))
summary(select(data = model, day  = 'tue'))
summary(select(data = model, day  = 'jan'))
```

---

stat *Calculate evaluation statistics from numerical vectors*

Description

Calculate statistical indexes (Number of pairs, observation average, model average, correlation, Index Of Agreement, Factor of 2, Root Mean Square Error, Mean Bias, Mean error, Normalized Mean Bias, and Normalized Mean Bias) for model evaluation

Usage

```
stat(
  model,
  observation,
  wd = FALSE,
  cutoff = NA,
  cutoff_NME = NA,
  nobs = 8,
  rname,
  verbose = TRUE
)
```

Arguments

model	numeric vector with paired model data
observation	numeric vector with paired observation data
wd	logical, set true to apply a rotation on wind direction, see notes
cutoff	(optionally the maximum) valid value for observation
cutoff_NME	(optionally the maximum) valid value for observation for NME, MFB and MFE

nobs	minimum number of observations
rname	row name
verbose	display additional information

Value

data.frame with calculated Number of pairs, observation average, model average, correlation, Index Of Agreement, Factor of 2, Root Mean Square Error, Mean Bias, Mean error, Normalized Mean Bias, and Normalized Mean Bias

Note

the option wd = TRUE applies a rotation of 360 on model wind direction to minimize the angular difference.

References

Emery, C. and Tai., E. 2001. Enhanced Meteorological Modeling and Performance Evaluation for Two Texas Ozone Episodes.

Monk, K. et al. 2019. Evaluation of Regional Air Quality Models over Sydney and Australia: Part 1—Meteorological Model Comparison. Atmosphere 10(7), p. 374. doi: 10.3390/atmos10070374.

Ramboll. 2018. PacWest Newport Meteorological Performance Evaluation.

Zhang, Y. et al. 2019. Multiscale Applications of Two Online-Coupled Meteorology-Chemistry Models during Recent Field Campaigns in Australia, Part I: Model Description and WRF/Chem-ROMS Evaluation Using Surface and Satellite Data and Sensitivity to Spatial Grid Resolutions. Atmosphere 10(4), p. 189. doi: 10.3390/atmos10040189.

Emery, C., Liu, Z., Russell, A.G., Odman, M.T., Yarwood, G. and Kumar, N. 2017. Recommendations on statistics and benchmarks to assess photochemical model performance. Journal of the Air & Waste Management Association 67(5), pp. 582–598. doi: 10.1080/10962247.2016.1265027.

Zhai, H., Huang, L., Emery, C., Zhang, X., Wang, Y., Yarwood, G., ... & Li, L. (2024). Recommendations on benchmarks for photochemical air quality model applications in China—NO2, SO2, CO and PM10. Atmospheric Environment, 319, 120290.

Examples

```
model <- 1:100
data  <- model + rnorm(100,0.2)
stat(model = model, observation = data)
```

`template` *Create templates for model evaluation*

Description

Create templates of code (r-scripts and bash job-submission script) to read, post-process and evaluate model results.

Usage

```
template(
  root,
  template = "WRF",
  case = "case",
  env = "rspatial",
  scheduler = "SBATCH",
  partition = "main",
  project = "PROJECT",
  verbose = TRUE
)
```

Arguments

`root`	directory to create the template
`template`	Character; One of of the following:

argument	TYPE	MODEL/OBS	OBSERVATION
WRF	post-process	WRF	METAR and INMET
WRF-3	post-process	WRF triple nested	METAR
WRF-Chem	post-process	WRF-Chem	METAR, AQS in Brazil and AERONET
EXP	post-process	WRF-Chem	METAR, PBL variables and composition
CAMX	post-process	CAMx	AERONET
CAMX-3	post-process	CAMx triple nested	AERONET
NCO	post-process	WRF	Satellite
terra	post-process	WRF	Satellite
SAT	evaluation	WRF	Satellite (GPCP)
MET	evaluation	WRF	METAR
MET-3	evaluation	WRF triple nested	METAR
AQ	evaluation	WRF or CAMx	O3, Max O3 and PM2.5
AQS_BR	download	observations	AQS in Brazil for SP and RJ
METAR	downlaad	observations	METAR from ASOS
INMET	pre-processing	observations	INMET (automatic and conventional)
merge	pre-processing	observations	merge INMET (automatic and conventional)
ISD	pre-processing	observations	METAR from ISD

`case`	case to be evaluated

env	name of the conda environment
scheduler	job scheduler used (SBATCH or PBS)
partition	partition name
project	project name
verbose	display additional information

Value

no value returned, create folders and other template scripts

Examples

```
temp <- file.path(tempdir(),"POST")
template(root = temp,template = 'WRF', case = 'WRF-only')
```

---

uv2wd *Function to calculate model wind direction*

Description

Function to calculate model wind direction

Usage

```
uv2wd(u, v, verbose = TRUE)
```

Arguments

u	data.frame with model time-series of U10
v	data.frame with model time-series of V10
verbose	display additional information

Value

vector or data.frame with time and the wind direction, units are degree north

Examples

```
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-01-02',tz = 'UTC'),
             by = 'hour')
U10 = data.frame(times = times,
                 test1 = c(3.29,2.07,1.96,2.82,3.73,
                           4.11,4.96,6.33,7.39,7.59,
                           7.51,7.22,6.81,6.43,5.81,
                           4.02,3.03,2.68,2.40,2.20,
```

```
                          2.09,1.95,1.66,1.39,1.4),
                test2 = c(6.29,4.87,6.16,7.12,8.77,
                          10.16,10.85,11.45,11.21,11.04,
                          11.09,10.67,10.48,10.00,8.96,
                          6.36,5.62,5.83,5.83,5.25,
                          4.11,3.08,2.26,1.14,-0.10))
V10 = data.frame(times = times,
                test1 = c(-8.87,-4.23,-2.81,-2.59,-4.58,
                          -4.80,-5.33,-5.86,-6.12,-6.13,
                          -6.11,-5.76,-5.91,-5.60,-5.09,
                          -3.33,-2.50,-2.29,-2.14,-2.07,
                          -1.95,-1.97,-2.04,-2.03,-1.9),
                test2 = c(11.80,5.88,5.74,5.56,6.87,
                          8.39,8.68,8.33,7.90,7.42,
                          6.96,6.87,6.36,5.61,5.16,
                          4.16,4.25,4.59,4.51,3.90,
                          2.97,1.98,1.04,-0.08,-0.44))

uv2wd(u = U10, v = V10)
```

---

uv2ws *Function to calculate model wind speed*

Description

Function to calculate model wind speed

Usage

```
uv2ws(u, v, verbose = TRUE)
```

Arguments

u	data.frame with model time-series of U10
v	data.frame with model time-series of V10
verbose	display additional information

Value

vector or data.frame with time and the wind sped, units are m/s

Examples

```
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-01-02',tz = 'UTC'),
             by = 'hour')

U10 = data.frame(times = times,
```

```
                  test1 = c(3.29,2.07,1.96,2.82,3.73,
                            4.11,4.96,6.33,7.39,7.59,
                            7.51,7.22,6.81,6.43,5.81,
                            4.02,3.03,2.68,2.40,2.20,
                            2.09,1.95,1.66,1.39,1.4),
                  test2 = c(6.29,4.87,6.16,7.12,8.77,
                            10.16,10.85,11.45,11.21,11.04,
                            11.09,10.67,10.48,10.00,8.96,
                            6.36,5.62,5.83,5.83,5.25,
                            4.11,3.08,2.26,1.14,-0.10))
V10 = data.frame(times = times,
                  test1 = c(-8.87,-4.23,-2.81,-2.59,-4.58,
                            -4.80,-5.33,-5.86,-6.12,-6.13,
                            -6.11,-5.76,-5.91,-5.60,-5.09,
                            -3.33,-2.50,-2.29,-2.14,-2.07,
                            -1.95,-1.97,-2.04,-2.03,-1.9),
                  test2 = c(11.80,5.88,5.74,5.56,6.87,
                            8.39,8.68,8.33,7.90,7.42,
                            6.96,6.87,6.36,5.61,5.16,
                            4.16,4.25,4.59,4.51,3.90,
                            2.97,1.98,1.04,-0.08,-0.44))

uv2ws(u = U10, v = V10)
```

---

vars *Function to return variable names*

Description

Return variable names of a NetCDF

Usage

```
vars(file = NA, action = "get", verbose = FALSE)
```

Arguments

file	file name
action	'get' to return variable names or 'print' to print
verbose	display additional information

Value

string

Examples

```
vars(paste0(system.file("extdata",package="eva3dm"),'/wrfinput_d01'))
```

`wrf_rast` *Creates SpatRaster object from wrf file*

Description

Creates a SpatRaster (terra R-package) object from a variable from wrf file (or another compatible NetCDF)

Usage

```
wrf_rast(
  file = file.choose(),
  name = NA,
  map,
  level = 1,
  times,
  latlon = FALSE,
  method = "bilinear",
  as_polygons = FALSE,
  flip_h = FALSE,
  flip_v = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`file`	wrf file
`name`	variable name
`map`	(optional) file with lat-lon variables and grid information
`level`	only for 4d data, numeric, default is 1 for surface (include all times)
`times`	only for 4d data, numeric, set to select time instead of levels (include all levels)
`latlon`	logical (default is FALSE), set TRUE project the output to "+proj=longlat +datum=WGS84 +no_defs"
`method`	method passed to terra::projection, default is bilinear
`as_polygons`	logical, true to return a SpatVector instead of SpatRaster
`flip_h`	horizontal flip (by rows)
`flip_v`	vertical flip (by cols)
`verbose`	display additional information
`...`	extra arguments passed to ncdf4::ncvar_get

Value

SpatRaster object (from terra package)

Examples

```
{

wrf <- paste(system.file("extdata", package = "eva3dm"),
                         "/wrfinput_d01", sep="")

r <- wrf_rast(file=wrf, name='XLAT')

plot_rast(r)
}
```

`wrf_sds` *Creates SpatRasterDataset object from wrf file*

Description

Creates a SpatRasterDataset (terra R-package) object from a variable from wrf file (or another compatible NetCDF) for all times and levels

Usage

```
wrf_sds(
  file = file.choose(),
  name = NA,
  map,
  latlon = FALSE,
  method = "bilinear",
  flip_h = FALSE,
  flip_v = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`file`	wrf file
`name`	variable name
`map`	(optional) file with lat-lon variables and grid information
`latlon`	logical (default is FALSE), set TRUE project the output to "+proj=longlat +datum=WGS84 +no_defs"
`method`	method passed to terra::projection, default is bilinear
`flip_h`	horizontal flip (by rows)
`flip_v`	vertical flip (by cols)
`verbose`	display additional information
`...`	extra arguments passed to ncdf4::ncvar_get

Value

SpatRasterDataset object (from terra package), each time-step is considered one subdatasets and each layer is one nlyr.

Note

The convention adopted to select specific times and atmospheric layers on wrf_sds is sds[time,layer] to keep consistence with sds.

Examples

```
file <- paste0(system.file("extdata",package="eva3dm"),"/wrf_4d_o3_Boston.nc")
O34d <- wrf_sds(file,'o3',verbose = TRUE)

# selecting one time, keeping multiple layers
O34d[1,]

# selecting one layer, keeping multiple times
O34d[,1]
```

---

write_stat *Functions to write stats and evaluation*

Description

Functions to write the output from evaluation functions. If the file name ends with .csv the function write.csv is used otherwise the function write.table is used.

Usage

```
write_stat(stat, file, sep = ";", dec = ".", verbose = FALSE, ...)
```

Arguments

stat	observed data.frame
file	model data.frame
sep	the field separator string, passed to write.table function
dec	he string to use for decimal points, passed to write.table function
verbose	display additional information
...	arguments passed to write.table and write.csv functions

Value

No return value

Examples

```
sample <- read_stat(paste0(system.file("extdata", package = "eva3dm"),"/sample.csv"),
                    verbose = TRUE)
dir.create(file.path(tempdir(), "stats"))

write_stat(file    = paste0(file.path(tempdir(), "stats"),'/sample.txt'),
           stat    = sample,
           verbose = TRUE)

write_stat(file    = paste0(file.path(tempdir(), "stats"),'/sample.csv'),
           stat    = sample,
           verbose = TRUE)
```

`yearly` *Calculate yearly mean, min or max*

Description

function to calculate yearly mean, min or max of a data.frame

Usage

```
yearly(
  data,
  time = "date",
  stat = mean,
  min_offset = 0,
  hour_offset = 0,
  days_offset = 0,
  month_offset = 0,
  numerical = TRUE,
  verbose = TRUE
)
```

Arguments

`data`	data.frame with time column and variable columns to be processed
`time`	name of the time column (default is date) in POSIXct
`stat`	function of the statistics to calculate (default is mean)
`min_offset`	minutes of observation from previous year (default is 0)
`hour_offset`	hours of observation from previous year (default is 0)
`days_offset`	day of observation from previous year (default is 0)
`month_offset`	months of observation from previous year (default is 0)
`numerical`	TRUE (default) include only numerical columns
`verbose`	display additional information

Value

data.frame with time and the yearly mean, min or max

Examples

```
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2025-12-31',tz = 'UTC'),
             by = 'day')

DATA <- data.frame(date = times,
                   var1 = rnorm(n = length(times), mean = 1,sd = 1),
                   var2 = rnorm(n = length(times), mean = 2,sd = 0.5),
                   var3 = rnorm(n = length(times), mean = 3,sd = 0.25))

data_year_mean <- yearly(DATA)
data_year_min  <- yearly(DATA,stat = min)
data_year_max  <- yearly(DATA,stat = max)
```

%IN% *Returns the common columns*

Description

results of 'd01 in d02' style syntax

Usage

```
x %IN% y
```

Arguments

x	data.frame
y	data.frame or character string

Value

data.frame with common columns or a cropped SpatRaster

Note

A message is always displayed to keep easy to track and debug issues (with the results and the evaluation process).

Can be used to crop rast objects, such as arguments of sat() function

See Also

See `select` for selection based on time.

Examples

```
times <- seq(as.POSIXct('2024-01-01',tz = 'UTC'),
             as.POSIXct('2024-01-02',tz = 'UTC'),
             by = 'hour')
randon_stuff <- rnorm(25,10)

observation <- data.frame(date   = times,
                          site_1 = randon_stuff,
                          site_3 = randon_stuff,
                          site_4 = randon_stuff,
                          site_5 = randon_stuff,
                          site_6 = randon_stuff,
                          site_7 = randon_stuff)

model_d01 <- data.frame(date  = times,
                        site_1=randon_stuff+1,
                        site_2=randon_stuff+2,
                        site_3=randon_stuff+3,
                        site_4=randon_stuff+4)

model_d02 <- data.frame(date  = times,
                        site_1=randon_stuff-1,
                        site_3=randon_stuff-3)

# multiline
model_d01_in_d02 <- model_d01 %IN% model_d02
eva(mo = model_d01_in_d02, ob = observation, rname = 'd01 in d02')

# or single line
eva(mo = model_d01 %IN% model_d02, ob = observation, rname = 'd01 in d02')
# or
eva(mo = model_d01, ob = observation %IN% model_d02, rname = 'd01 in d02')
```

`%at%` *Combine stats and site list to overlay plot*

Description

combines the stats (from individual station evaluation) and site list in a SpatVector using row.names

Usage

```
stat %at% site
```

Arguments

`stat`	data.frame with stats or other variable (containing row.names and other variables)

site data.frame with site list (containing row.names, lat and lon)

Value

SpatVector (terra package)

Examples

```
sites <- data.frame(lat = c(-22.72500,-23.64300,-20.34350),
                    lon = c(-47.34800,-46.49200,-40.31800),
                    row.names = c('Americana','SAndre','VVIbes'),
                    stringsAsFactors = F)
model<- readRDS(paste0(system.file("extdata",package="eva3dm"),"/model.Rds"))
obs  <- readRDS(paste0(system.file("extdata",package="eva3dm"),"/obs.Rds"))

# evaluation by station
stats <- eva(mo = model, ob = obs, site = "Americana")
stats <- eva(mo = model, ob = obs, site = "SAndre",table = stats)
stats <- eva(mo = model, ob = obs, site = "VVIbes",table = stats)
# evaluation using all stations
stats <- eva(mo = model, ob = obs, site = "ALL", table = stats)

print(stats)

geo_stats <- stats %at% sites

print(geo_stats)
```

Index