Package: PEcAn.utils (via r-universe)

November 20, 2024

Type Package

Title PEcAn Functions Used for Ecological Forecasts and Reanalysis

Version 1.8.0.9000

Description The Predictive Ecosystem Carbon Analyzer (PEcAn) is a scientific workflow management tool that is designed to simplify the management of model parameterization, execution, and analysis. The goal of PEcAn is to streamline the interaction between data and models, and to improve the efficacy of scientific investigation.

Imports abind (>= 1.4.5), curl, dplyr, lubridate (>= 1.6.0), magrittr, ncdf4 (>= 1.15), PEcAn.logger, purrr, rlang, stringi, units

Suggests coda (>= 0.18), data.table, ggplot2, MASS, mockery, randtoolbox, rjags, testthat (>= 2.0.0), withr, xtable

License BSD_3_clause + file LICENSE

Copyright Authors

LazyData true

Encoding UTF-8

RoxygenNote 7.3.2

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libicu-dev libnetcdf-dev libssl-dev libudunits2-dev

Repository https://pecanproject.r-universe.dev

RemoteUrl https://github.com/PecanProject/pecan

RemoteRef HEAD

RemoteSha cab30e89b415a888f4d65a0cc6e81d4b81ea217f

Contents

`arrhenius.scaling` *Arrhenius scaling*

Description

Scale temperature dependent trait from measurement temperature to reference temperature

Usage

```
arrhenius.scaling(observed.value, old.temp, new.temp = 25)
```

Arguments

`observed.value`	observed value of temperature dependent trait, e.g. Vcmax, root respiration rate
`old.temp`	temperature at which measurement was taken or previously scaled to
`new.temp`	temperature to be scaled to, default = 25 C

Value

numeric value at reference temperature

Author(s)

unknown

as.sequence *Convert categorical variable into sequential integers*

Description

Turns any categorical variable into a sequential integer. This transformation is required for using data in BUGS/JAGS

Usage

```
as.sequence(x, na.rm = TRUE)
```

Arguments

x	categorical variable as vector
na.rm	logical: return NA's or replace with max(x) + 1

Value

sequence from 1:length(unique(x))

Author(s)

David LeBauer

bibtexify *bibtexify*

Description

Converts author year title to bibtex `author1999abc` format

Usage

```
bibtexify(author, year, title)
```

Arguments

author	name of first author
year	year of publication
title	manuscript title

Value

bibtex citation

Author(s)

unknown

`bugs.rdist` *Sample from an R distribution using JAGS*

Description

Takes a distribution with R parameterization, converts it to a BUGS parameterization, and then samples from the distribution using JAGS

Usage

```
bugs.rdist(
  prior = data.frame(distn = "norm", parama = 0, paramb = 1),
  n.iter = 1e+05,
  n = NULL
)
```

Arguments

`prior`	dataframe with distribution name and parameters
`n.iter`	number of MCMC samples. Output will have n.iter/4 samples
`n`	number of randomly chosen samples to return.

Value

vector of samples

Author(s)

David LeBauer

`capitalize` *Capitalize a string*

Description

Capitalize a string

Usage

```
capitalize(x)
```

Arguments

x	string

Value

x, capitalized

Author(s)

David LeBauer

---

cf2datetime *Convert CF-style date-time to POSIXct date-time*

Description

Convert CF-style date-time to POSIXct date-time

Usage

```
cf2datetime(value, unit, tz = "UTC")
```

Arguments

value	Numeric value of CF date-time
unit	CF style unit (e.g. "days since 2010-01-01")
tz	Time zone of result (default = "UTC")

Value

POSIXct datetime

Author(s)

Alexey Shiklomanov

Examples

```
cf2datetime(5, "days since 1981-01-01")
cf2datetime(27, "minutes since 1963-01-03 12:00:00 -05:00")
# no leap year
cf2datetime(365, "days since 1999-01-01")
# leap year
cf2datetime(365, "days since 2000-01-01 12:00:00 -05:00")
```

`clear.scratch` *Removes previous model run output from worker node local scratch directories on EBI-CLUSTER*

Description

Removes previous model run output from worker node local scratch directories on EBI-CLUSTER

Usage

```
clear.scratch(settings)
```

Arguments

`settings`	list of PEcAn settings. Only `settings$host$name` is used

Value

nothing

Author(s)

Shawn Serbin

Examples

```
## Not run:
clear.scratch(settings)

## End(Not run)
```

`convert.expr` *Convert expression to variable names*

Description

Convert expression to variable names

Usage

```
convert.expr(expression)
```

Arguments

`expression`	expression string

Value

list

Author(s)

Istem Fer

`datetime2cf` *Convert POSIXct date-time to CF-style date-time*

Description

Convert POSIXct date-time to CF-style date-time

Usage

```
datetime2cf(datetime, unit, ...)
```

Arguments

`datetime`	POSIXct datetime, or object that can be to POSIXct via `as.POSIXct`
`unit`	Target CF-style unit (e.g. "days since 2010-01-01")
`...`	Additional arguments to `as.POSIXct`. A common one is `tz` for time-zone (e.g. `tz = "UTC"`).

Value

Numeric value of date-time in target CF unit

Examples

```
datetime2cf("1990-10-05", "days since 1990-01-01", tz = "UTC")
```

`datetime2doy` *Extract Julian day from CF or POSIXct date-times*

Description

This gets around the fact that most functions for calculating Julian Day do not support non-integer days.

Usage

```
datetime2doy(datetime, tz = "UTC")

cf2doy(value, unit, tz = "UTC")
```

Arguments

datetime	POSIXct datetime, or object that can be to POSIXct via as.POSIXct
tz	Time zone of result (default = "UTC")
value	Numeric value of CF date-time
unit	CF style unit (e.g. "days since 2010-01-01")

Value

Numeric Julian date

Author(s)

Alexey Shiklomanov

Examples

```
datetime2doy("2010-01-01") # 1
datetime2doy("2010-01-01 12:00:00") # 1.5
cf2doy(0, "days since 2007-01-01")
cf2doy(5, "days since 2010-01-01") # 6
cf2doy(5, "days since 2010-01-01") # 6
```

---

days_in_year *Number of days in a year*

Description

Calculate number of days in a year based on whether it is a leap year or not.

Usage

```
days_in_year(year, leap_year = TRUE)
```

Arguments

year	Numeric year (can be a vector)
leap_year	Default = TRUE. If set to FALSE will always return 365

Value

integer vector, all either 365 or 366

Author(s)

Alexey Shiklomanov

Examples

```
days_in_year(2010)  # Not a leap year -- returns 365
days_in_year(2012)  # Leap year -- returns 366
days_in_year(2000:2008)  # Function is vectorized over years
```

`distn.stats` *Distribution Stats*

Description

Implementation of standard equations used to calculate mean and sd for a variety of named distributions different

Usage

```
distn.stats(distn, a, b)
```

Arguments

`distn`	named distribution, one of 'beta', 'exp', 'f', 'gamma', 'lnorm', 'norm', 't',
`a`	numeric; first parameter of `distn`
`b`	numeric; second parameter of `distn`

Value

vector with mean and standard deviation

Author(s)

David LeBauer

Examples

```
distn.stats('norm', 0, 1)
```

distn.table.stats *Helper function for computing summary statistics of a parametric distribution*

Description

return mean and standard deviation of a distribution for each distribution in a table with `colnames = c('distn', 'a', 'b')`, e.g. in a table of priors

Usage

```
distn.table.stats(distns)
```

Arguments

`distns`	table of distributions; see examples

Value

named vector of mean and SD

Author(s)

David LeBauer

download.url *Try and download a file.*

Description

This will download a file, if retry is set and 404 is returned it will wait until the file is available. If the file is still not available after timeout tries, it will return NA. If the file is downloaded it will return the name of the file

Usage

```
download.url(url, file, timeout = 600, .opts = list(), retry = TRUE)
```

Arguments

`url`	the url of the file to download
`file`	the filename
`timeout`	number of seconds to wait for file (default 600)
`.opts`	list of options for curl, for example to download from a protected site use list(userpwd=userpass, httpauth = 1L)
`retry`	retry if url not found yet, this is used by Brown Dog

Value

returns name of file if successful or NA if not.

Examples

```
## Not run:
download.url('http://localhost/', index.html)

## End(Not run)
```

`download_file` *Simple function to use ncftpget for FTP downloads behind a firewall.*

Description

Requires ncftpget and a properly formatted config file in the users home directory

Usage

```
download_file(url, filename, method)
```

Arguments

`url`	complete URL for file download
`filename`	destination file name
`method`	Method of file retrieval. Can set this using the `options(download.ftp.method=[method])` in your Rprofile. example options(download.ftp.method="ncftpget")

Author(s)

Shawn Serbin, Rob Kooper

Examples

```
## Not run:
download_file("http://lib.stat.cmu.edu/datasets/csb/ch11b.txt","~/test.download.txt")

download_file("
  ftp://ftp.cdc.noaa.gov/Datasets/NARR/monolevel/pres.sfc.2000.nc",
  "~/pres.sfc.2000.nc")

## End(Not run)
```

full.path *Creates an absolute path to a folder.*

Description

This will take a folder and make it into an absolute folder name. It will normalize the path and prepend it with the current working folder if needed to get an absolute path name.

Usage

```
full.path(folder)
```

Arguments

folder	folder for file paths.

Value

absolute path

Author(s)

Rob Kooper

Examples

```
full.path('pecan')
```

get.ensemble.inputs *get.ensemble.inputs*

Description

Splits climate met for SIPNET

Usage

```
get.ensemble.inputs(settings, ens = 1)
```

Arguments

settings	PEcAn settings list
ens	ensemble number. default = 1

Value

find correct ensemble inputs

Author(s)

Mike Dietze and Ann Raiho

`get.parameter.stat` *Get Parameter Statistics*

Description

Gets statistics for LaTeX - formatted table

Usage

```
get.parameter.stat(mcmc.summary, parameter)
```

Arguments

`mcmc.summary`	probably produced by `summary.mcmc`
`parameter`	name of parameter to extract, as character

Value

table with parameter statistics

Author(s)

David LeBauer

Examples

```
## Not run: get.parameter.stat(mcmc.summaries[[1]], 'beta.o')
```

`get.quantiles` *Get Quantiles*

Description

Returns a vector of quantiles specified by a given <quantiles> xml tag

Usage

```
get.quantiles(quantiles.tag)
```

Arguments

`quantiles.tag`	specifies tag used to specify quantiles

Value

vector of quantiles

Author(s)

David LeBauer

`get.run.id` *returns an id representing a model run*

Description

Provides a consistent method of naming runs; for use in model input files and indices

Usage

```
get.run.id(run.type, index, trait = NULL, pft.name = NULL, site.id = NULL)
```

Arguments

`run.type`	character, can be any character; currently 'SA' is used for sensitivity analysis, 'ENS' for ensemble run.
`index`	unique index for different runs, e.g. integer counting members of an ensemble or a quantile used to which a trait has been perturbed for sensitivity analysis
`trait`	name of trait being sampled (for sensitivity analysis)
`pft.name`	name of PFT (value from pfts.names field in database)
`site.id`	optional site id .This is could be necessary for multisite write=false ensembles.

Value

id representing a model run

Author(s)

Carl Davidson, David LeBauer

Examples

```
get.run.id('ENS', left.pad.zeros(1, 5))
get.run.id('SA', round(qnorm(-3),3), trait = 'Vcmax')
```

get.sa.sample.list *get sensitivity samples as a list*

Description

get sensitivity samples as a list

Usage

```
get.sa.sample.list(pft, env, quantiles)
```

Arguments

pft	list of samples from Plant Functional Types
env	list of samples from environment parameters
quantiles	quantiles at which to obtain samples from parameter for sensitivity analysis

Value

sa.sample.list

get.sa.samples *Get sensitivity analysis samples*

Description

Samples parameters for a model run at specified quantiles.

Usage

```
get.sa.samples(samples, quantiles)
```

Arguments

samples	random samples from trait distribution
quantiles	list of quantiles to at which to sample, set in settings file

Details

Samples from long (>2000) vectors that represent random samples from a trait distribution. Samples are either the MCMC chains output from the Bayesian meta-analysis or are randomly sampled from the closed-form distribution of the parameter probability distribution function. The list is indexed first by trait, then by quantile.

Value

a list of lists representing quantile values of trait distributions

Author(s)

David LeBauer

`get.stats.mcmc` *Further summarizes output from summary.mcmc*

Description

Further summarizes output from summary.mcmc

Usage

```
get.stats.mcmc(mcmc.summary, sample.size)
```

Arguments

`mcmc.summary`	probably produced by `summary.mcmc`
`sample.size`	passed as 'n' in returned list

Value

list with summary statistics for parameters in an MCMC chain

Author(s)

David LeBauer

`left.pad.zeros` *Left Pad Zeros*

Description

left padded by zeros up to a given number of digits.

Usage

```
left.pad.zeros(num, digits = 5)
```

Arguments

`num`	number to be padded (integer)
`digits`	number of digits to add

Details

returns a string representing a given number

Value

num with zeros to the left

Author(s)

Carl Davidson

`listToArgString` *format a list of arguments as one comma-separated string*

Description

format a list of arguments as one comma-separated string

Usage

```
listToArgString(l)
```

Arguments

`l`	a named list of function arguments

Value

A string containing named argument/value pairs separated by commas

Author(s)

Ryan Kelly

load.modelpkg *Load model package*

Description

Load model package

Usage

```
load.modelpkg(model)
```

Arguments

model	name of model

Value

FALSE if function returns error; else TRUE

Author(s)

David LeBauer

Examples

```
## Not run: require.modelpkg(BioCro)
```

load_local *Load an* RData *file into a list*

Description

Instead of polluting the current environment, this allows you to read an RData file into a list object of whatever name you choose.

Usage

```
load_local(file)
```

Arguments

file	a (readable binary-mode) connection or a character string giving the name of the file to load (when tilde expansion is done).

Value

List, with names corresponding to object names in file

Author(s)

Alexey Shiklomanov

Examples

```
x <- 1:10
y <- 11:15
tmp <- tempfile()
save(x, y, file = tmp)
my_list <- load_local(tmp)
rm(tmp)
```

`match_file` *Match a file*

Description

Return a list of files given a full prefix and optional suffix. Optionally, confirm that the right number of files are returned. If the wrong number of files is returned, throw an error.

Usage

```
match_file(path_prefix, suffix = NULL, expect = NULL)
```

Arguments

`path_prefix`	Full path and file prefix
`suffix`	File suffix, as character (default = `NULL`)
`expect`	Number of files expected to be returned (default = `NULL`)

Details

If `path_prefix` points to a directory, then all files inside that directory that match the suffix (if provided) are returned.

Value

Character vector of matched file names, as full paths.

mcmc.list2init *Convert mcmc.list to initial condition list*

Description

Used for restarting MCMC code based on last parameters sampled (e.g. in JAGS)

Usage

```
mcmc.list2init(dat)
```

Arguments

dat	mcmc.list object

Value

list

Author(s)

Mike Dietze

met2model.exists *checks that met2model function exists*

Description

Checks if met2model.<model> exists for a particular model

Usage

```
met2model.exists(model)
```

Arguments

model	model package name

Value

logical

misc.are.convertible *function to check whether units are convertible by misc.convert function*

Description

function to check whether units are convertible by misc.convert function

Usage

```
misc.are.convertible(u1, u2)
```

Arguments

u1	unit to be converted from, character
u2	unit to be converted to, character

Value

logical

Author(s)

Istem Fer, Shawn Serbin

misc.convert *conversion function for the unit conversions that udunits cannot handle but often needed in PEcAn calculations*

Description

conversion function for the unit conversions that udunits cannot handle but often needed in PEcAn calculations

Usage

```
misc.convert(x, u1, u2)
```

Arguments

x	convertible values
u1	unit to be converted from, character
u2	unit to be converted to, character

Value

val converted values

Author(s)

Istem Fer, Shawn Serbin

`mstmipvar` *return MstMIP variable as ncvar*

Description

returns a MstMIP variable as a ncvar based on name and other parameters passed in.

Usage

```
mstmipvar(
  name,
  lat = NULL,
  lon = NULL,
  time = NULL,
  nsoil = NULL,
  silent = FALSE
)
```

Arguments

`name`	of variable
`lat`	latitude if dimension requests it
`lon`	longitude if dimension requests it
`time`	time if dimension requests it
`nsoil`	nsoil if dimension requests it
`silent`	logical: suppress log messages about missing variables?

Value

ncvar based on MstMIP definition

Author(s)

Rob Kooper

need_packages *Check if required packages are installed, and throw an informative error if not.*

Description

Check if required packages are installed, and throw an informative error if not.

Usage

```
need_packages(...)
```

Arguments

...	Package names, as characters. Can be passed as individual arguments, character vectors, or any combination thereof.

Value

pkgs, invisibly

Author(s)

Alexey Shiklomanov

Examples

```
# Only need ::: because package isn't exported.
# Inside a package, just call `need_packages`
PEcAn.utils:::need_packages("stats", "methods") # Always works
try(PEcAn.utils:::need_packages("notapackage"))
```

newxtable *New xtable*

Description

utility to properly escape the '%' sign for latex

Usage

```
newxtable(
  x,
  environment = "table",
  table.placement = "ht",
  label = NULL,
  caption = NULL,
  caption.placement = NULL,
  align = NULL
)
```

Arguments

x	data.frame to be converted to latex table
environment	can be 'table'; 'sidewaystable' if using latex rotating package
table.placement, label, caption, caption.placement, align	passed to xtable

Value

Latex version of table, with percentages properly formatted

Author(s)

David LeBauer

---

n_leap_day *n_leap_day*

Description

number of leap days between two dates

Usage

```
n_leap_day(start_date, end_date)
```

Arguments

start_date, end_date	dates in any format recognized by as.Date

Author(s)

Mike Dietze

paste.stats *Paste Stats*

Description

A helper function for building a LaTex table.

Usage

```
paste.stats(median, lcl, ucl, n = 2)
```

Arguments

median	50-percent quantile
lcl	lower confidence limit
ucl	upper confidence limit
n	significant digits for printing. Passed to tabnum

Details

Used by get.parameter.stat.

Author(s)

David LeBauer

Examples

```
paste.stats(3.333333, 5.00001, 6.22222, n = 3)
# [1] "$3.33(5,6.22)$"
```

pdf.stats *Probability Distribution Function Statistics*

Description

Calculate mean, variance statistics, and CI from a known distribution

Usage

```
pdf.stats(distn, A, B)
```

Arguments

distn	name of distribution used by R (beta, f, gamma, lnorm, norm, weibull)
A	first parameter
B	second parameter

Value

list with mean, variance, and 95 CI

Author(s)

David LeBauer

PEcAn *R package to support PEcAn, the Predictive Ecosystem Analyzer*

Description

Instructions for the use of this package are provided in the project documentation https://pecanproject.github.io/documentation.html.

Details

Project homepage: pecanproject.org

Description of PEcAn

The Predictive Ecosystem Analyzer (PEcAn) is a scientific workflow management tool that is designed to simplify the management of model parameterization, execution, and analysis. The goal of PEcAn is to streamline the interaction between data and models, and to improve the efficacy of scientific investigation. PEcAn is an open source utility that encapsulates:

1. acquisition of meteorological inputs
2. synthesis of physiological trait data as the posterior distribution of a Bayesian meta-analysis
3. sampling trait meta-analysis posterior distributions to parameterize ensembles of ED2 and other ecophysiological models
4. probabilistic forecasts
5. postprocessing to constrain forecasts and model parameters with field, meterological, eddy flux, and spectral data, and
6. provenance tracking

PECAn integrates available data into ecological forecasts by running ensembles of a terrestrial ecosystem model that is parameterized by the posterior distribution from a meta-analysis of available plant trait data. These trait data are assembled from field research and primary literature, and are stored in a PostgreSQL database. Current development focused on biofuel crops uses BETYdb. In addition to generating forecasts that reflect available data, PEcAn quantifies the contribution of

each parameter to model uncertainty. This information informs targeted data collection and synthesis efforts that most efficiently reduce forecast uncertainty.

Current development is focused on developing PEcAn into a real-time data assimilation and forecasting system. This system will provide a detailed analysis of the past and present ecosystem functioning that seamlessly transitions into forecasts.

Author(s)

Maintainer: Rob Kooper <kooper@illinois.edu>

Authors:

- Mike Dietze <dietze@bu.edu>
- David LeBauer <dlebauer@email.arizona.edu>
- Xiaohui Feng <feng22@illinois.edu>
- Dan Wang
- Carl Davidson <davids14@illinois.edu>
- Shawn Serbin <sserbin@bnl.gov>
- Shashank Singh <shashanksingh819@gmail.com>
- Chris Black <chris@ckblack.org>
- Tanishq Jain <tanishqjain010@gmail.com>

Other contributors:

- University of Illinois, NCSA [copyright holder]

---

r2bugs.distributions *convert R parameterizations to BUGS paramaterizations*

Description

R and BUGS have different parameterizations for some distributions. This function transforms the distributions from R defaults to BUGS defaults. BUGS is an implementation of the BUGS language, and these transformations are expected to work for bugs.

Usage

```
r2bugs.distributions(priors, direction = "r2bugs")
```

Arguments

priors	data.frame with columns distn = distribution name, parama, paramb using R default parameterizations.
direction	One of "r2bugs" or "bugs2r"

Value

priors dataframe using JAGS default parameterizations

Author(s)

David LeBauer, Ben Bolker

Examples

```
priors <- data.frame(distn = c('weibull', 'lnorm', 'norm', 'gamma'),
                     parama = c(1, 1, 1, 1),
                     paramb = c(2, 2, 2, 2))
r2bugs.distributions(priors)
```

`read.output` *Read model output*

Description

Reads the output of a single model run

Usage

```
read.output(
  runid,
  outdir,
  start.year = NA,
  end.year = NA,
  variables = "GPP",
  dataframe = FALSE,
  pft.name = NULL,
  ncfiles = NULL,
  verbose = FALSE,
  print_summary = TRUE
)
```

Arguments

`runid`	the ID distinguishing the model run. Can be omitted if `ncfiles` is set.
`outdir`	the directory that the model's output was sent to. Can be omitted if `ncfiles` is set.
`start.year, end.year`	first and last year of output to read. Specify as a date-time (only the year portion is used) or as a four-digit number or string. If `NA`, reads all years found in `outdir`.
`variables`	Character vector of variables to be read from model output. Default = `"GPP"`. If `NULL`, try to read all variables in output file..

dataframe	Logical: if TRUE, will return output in a data.frame format with a posix column. Useful for PEcAn.benchmark::align.data and plotting.
pft.name	character string, name of the plant functional type (PFT) to read PFT-specific output. If NULL no PFT-specific output will be read even the variable has PFT as a dimension.
ncfiles	Custom character vector of full paths to NetCDF files. If NULL (default), this list is constructed automatically by looking for YYYY.nc files in file.path(outdir, runid).
verbose	Logical. If TRUE, print status as every year and variable is read, as well as all NetCDF diagnostics (from verbose argument to, e.g., ncdf4::nc_open()) (default = FALSE).
print_summary	Logical. If TRUE (default), calculate and print a summary of the means of each variable for each year.

Details

Generic function to convert model output from model-specific format to a common PEcAn format. This function uses MsTMIP variables except that units of (kg m-2 d-1) are converted to kg ha-1 y-1. Currently this function converts Carbon fluxes: GPP, NPP, NEE, TotalResp, AutoResp, HeteroResp, DOC_flux, Fire_flux, and Stem (Stem is specific to the BioCro model) and Water fluxes: Evaporation (Evap), Transpiration (TVeg), surface runoff (Qs), subsurface runoff (Qsb), and rainfall (Rainf).

For more details, see the MsTMIP variables documentation.

Value

If dataframe = FALSE, a vector of output variables. If dataframe = TRUE, a data.frame of output variables with POSIXct timestamps added (posix column). The posix column is in seconds after January 1 of start.year, or 1970 if start.year is not provided.

Author(s)

Michael Dietze, David LeBauer, Alexey Shiklomanov

---

read_web_config *Read* config.php *file into an R list*

Description

Read config.php file into an R list

Usage

```
read_web_config(
  php.config = "../../web/config.php",
  parse = TRUE,
  expand = TRUE
)
```

Arguments

php.config	Path to config.php file
parse	Logical. If TRUE (default), try to parse numbers and unquote strings.
expand	Logical. If TRUE (default), try to perform some variable substitutions.

Value

Named list of variable-value pairs set in config.php

Author(s)

Alexey Shiklomanov, Michael Dietze, Rob Kooper

Examples

```
## Not run:
# Read Docker configuration and extract the `dbfiles` and output folders.
docker_config <- read_web_config(file.path("..", "..", "docker", "web", "config.docker.php"))
docker_config[["dbfiles_folder"]]
docker_config[["output_folder"]]

## End(Not run)
```

---

retry.func *Retry function X times before stopping in error*

Description

Retry function X times before stopping in error

Usage

```
retry.func(
  expr,
  isError = function(x) inherits(x, "try-error"),
  maxErrors = 5,
  sleep = 0
)
```

Arguments

expr	The function to try running
isError	function to use for checking whether to try again. Must take one argument that contains the result of evaluating expr and return TRUE if another retry is needed
maxErrors	The number of times to retry the function
sleep	How long to wait before retrying the function call

Value

retval returns the results of the function call

Author(s)

Shawn Serbin <adapted from https://stackoverflow.com/questions/20770497/how-to-retry-a-statement-on-error>

Examples

```
## Not run:
  file_url <- paste0("https://thredds.daac.ornl.gov/",
      "thredds/dodsC/ornldaac/1220",
      "/mstmip_driver_global_hd_climate_lwdown_1999_v1.nc4")
dap <- retry.func(
  ncdf4::nc_open(file_url),
  maxErrors=10,
  sleep=2)

## End(Not run)
```

---

robustly *Adverb to try calling a function* n *times before giving up*

Description

Adverb to try calling a function n times before giving up

Usage

```
robustly(.f, n = 10, timeout = 0.2, silent = TRUE)
```

Arguments

.f	Function to call.
n	Number of attempts to try
timeout	Timeout between attempts, in seconds
silent	Silence error messages?

Value

Modified version of input function

Examples

```
rlog <- robustly(log, timeout = 0.3)
try(rlog("fail"))
## Not run:
 nc_openr <- robustly(ncdf4::nc_open, n = 10, timeout = 0.5)
 nc <- nc_openr(url)
 # ...or just call the function directly
 nc <- robustly(ncdf4::nc_open, n = 20)(url)
 # Useful in `purrr` maps
 many_vars <- purrr::map(varnames, robustly(ncdf4::ncvar_get), nc = nc)

## End(Not run)
```

---

rsync *R implementation of rsync*

Description

rsync is a file copying tool in bash

Usage

```
rsync(args, from, to, pattern = "")
```

Arguments

args	rsync arguments (see man rsync)
from	source
to	destination
pattern	file pattern to be matched

Value

nothing, transfers files as a side effect

Author(s)

David LeBauer

Shawn Serbin

seconds_in_year *Number of seconds in a given year*

Description

Number of seconds in a given year

Usage

```
seconds_in_year(year, leap_year = TRUE, ...)
```

Arguments

year	Numeric year (can be a vector)
leap_year	Default = TRUE. If set to FALSE will always return 31536000.
...	additional arguments, all currently ignored

Author(s)

Alexey Shiklomanov

Examples

```
seconds_in_year(2000)  # Leap year -- 366 x 24 x 60 x 60 = 31622400
seconds_in_year(2001)  # Regular year -- 365 x 24 x 60 x 60 = 31536000
seconds_in_year(2000:2005)  # Vectorized over year
```

sendmail *Sends email. This assumes the program sendmail is installed.*

Description

Sends email. This assumes the program sendmail is installed.

Usage

```
sendmail(from, to, subject, body)
```

Arguments

from	the sender of the mail message
to	the receipient of the mail message
subject	the subject of the mail message
body	the body of the mail message

Value

nothing

Author(s)

Rob Kooper

Examples

```
## Not run:
sendmail('bob@example.com', 'joe@example.com', 'Hi', 'This is R.')

## End(Not run)
```

---

ssh *R implementation of SSH*

Description

R implementation of SSH

Usage

```
ssh(host, ..., args = "")
```

Arguments

host	(character) machine to connect to
...	Commands to execute. Will be passed as a single quoted string
args	futher arguments

---

standard_vars *Standardized variable names and units for PEcAn*

Description

A lookup table giving standard names, units and descriptions for variables in PEcAn input/output files. Originally based on the MsTMIP standards, with additions to accomodate a wider range of model inputs and outputs. The standard_vars table replaces both `mstmip_vars` and `mstmip_local`, both of which are now deprecated.

Usage

```
standard_vars
```

Format

data frame, all columns character

Variable.Name Short name suitable for programming with

standard_name Name used in the NetCDF CF metadata conventions

Units Standard units for this variable. Do not call variables by these names if they are in different units. See `ud_convert` for conversions to and from non-standard units

Long.Name Human-readable variable name, suitable for e.g. axis labels

Category What kind of variable is it? (Carbon pool, N flux, dimension, input driver, etc)

var_type Storage type (character, integer, etc)

dim1,dim2,dim3,dim4 Dimensions across which is this variable allowed to vary. Dimension names are themselves standard vars and must be present in the table with category "Dimension"

Description Further details. For composite measures, list the variables it is calculated from

---

status *PEcAn workflow status tracking*

Description

Records the progress of a PEcAn workflow by writing statuses and timestamps to a STATUS file. Use these each time a module starts, finishes, or is skipped.

Usage

```
status.start(name, file = NULL)

status.end(status = "DONE", file = NULL)

status.skip(name, file = NULL)

status.check(name, file = NULL)
```

Arguments

`name`	one-word description of the module being checked or recorded, e.g. "TRAIT", "MODEL", "ENSEMBLE"
`file`	path to status file. If NULL, taken from `settings` (see details)
`status`	one-word summary of the module result, e.g. "DONE", "ERROR"

Details

All of these functions write to or read from a STATUS file in your run's output directory. If the file is not specified in the call, they will look for a `settings` object in the global environment and use `<settings$outdir>/STATUS` if possible.

Since the status functions may be called inside error-handling routines, it's important that they not produce new errors of their own. Therefore if the output file doesn't exist or is not writable, rather than complain the writer functions (`status.start`, `status.end`, `status.skip`) will print to the console and `status.check` will simply return 0.

Value

For `status.start`, `status.end`, and `status.skip`: NULL, invisibly

For `status.check`, an integer: 0 if module not run, 1 if done, -1 if error

Functions

- `status.start()`: Record module start time
- `status.end()`: Record module completion time and status
- `status.skip()`: Record that module was skipped
- `status.check()`: Look up module status from file

Author(s)

Rob Kooper

---

summarize.result *Summarize results of replicate observations in trait data query*

Description

Summarize results of replicate observations in trait data query

Usage

```
summarize.result(result)
```

Arguments

`result`	dataframe with results of trait data query

Value

result with replicate observations summarized

Author(s)

David LeBauer, Alexey Shiklomanov

tabnum *Table numbers*

Description

Convert number to n significant digits

Usage

```
tabnum(x, n = 3)
```

Arguments

x	numeric value or vector
n	number of significant figures

Value

x rounded to n significant figures

Author(s)

David LeBauer

Examples

```
tabnum(1.2345)
tabnum(1.2345, n = 4)
```

temp.settings *Create a temporary settings file*

Description

Uses tempfile function to provide a valid temporary file (OS independent) Useful for testing functions that depend on settings file Reference: http://stackoverflow.com/a/12940705/199217

Usage

```
temp.settings(settings.txt)
```

Arguments

settings.txt	character vector to be written

Value

character vector written to and read from a temporary file

Author(s)

David LeBauer

`timezone_hour` *Timezone Hour*

Description

Returns the number of hours offset to UTC for a timezone.

Usage

```
timezone_hour(timezone)
```

Arguments

`timezone`	to be converted

Value

hours offset of the timezone

Author(s)

Rob Kooper

Examples

```
## Not run:
timezone_hour('America/New_York')

## End(Not run)
```

to_ncdim *Make some values into an NCDF dimension variable*

Description

Units and longnames are looked up from the standard_vars table

Usage

```
to_ncdim(dimname, vals)
```

Arguments

dimname	character vector, standard dimension name (must be in PEcAn.utils::standard_vars)
vals	values of dimension; can be single value or vector

Value

ncdim defined according to standard_vars

Author(s)

Anne Thomas

to_ncvar *Define an NCDF variable*

Description

Define an NCDF variable

Usage

```
to_ncvar(varname, dims)
```

Arguments

varname	character vector, standard variable name (must be in PEcAn.utils::standard_vars)
dims	list of previously defined ncdims (function will match subset of dims for this variable in standard_vars; can include other dims–enables lapply.)

Value

ncvar defined according to standard_vars

Author(s)

Anne Thomas

trait.lookup *Dictionary of terms used to identify traits in ed, filenames, and figures*

Description

Dictionary of terms used to identify traits in ed, filenames, and figures

Usage

```
trait.lookup(traits = NULL)
```

Arguments

traits	a vector of trait names, if traits = NULL, all of the traits will be returned.

Value

a dataframe with id, the name used by ED and PEcAn database for a parameter; fileid, an abbreviated name used for files; figid, the parameter name written out as best known in english for figures and tables.

Examples

```
# convert parameter name to a string appropriate for end-use plotting
## Not run:
trait.lookup('growth_resp_factor')
trait.lookup('growth_resp_factor')$figid

# get a list of all traits and units in dictionary
trait.lookup()[,c('figid', 'units')]

## End(Not run)
```

transformstats *Transform misc. statistics to SE*

Description

Automates transformations of SD, MSE, LSD, 95%CI, HSD, and MSD to conservative estimates of SE. Method details and assumptions described in LeBauer 2020 Transforming ANOVA and Regression statistics for Meta-analysis. Authorea. DOI: https://doi.org/10.22541/au.158359749.96662550

Usage

```
transformstats(data)
```

Arguments

data	data frame with columns for mean, statistic, n, and statistic name

Value

data frame with statistics transformed to SE

Author(s)

David LeBauer

Examples

```
statdf <- data.frame(Y=rep(1,5),
                     stat=rep(1,5),
                     n=rep(4,5),
                     statname=c('SD', 'MSE', 'LSD', 'HSD', 'MSD'))
transformstats(statdf)
```

---

tryl *Test if function gives an error*

Description

adaptation of try that returns a logical value (FALSE if error)

Usage

```
tryl(FUN)
```

Arguments

FUN	function to be evaluated for error

Value

FALSE if function returns error; else TRUE

Author(s)

David LeBauer

Examples

```
tryl(1+1)
# TRUE
tryl(sum('a'))
# FALSE
```

ud_convert *Convert units*

Description

Unit conversion to replace the now-unmaintained `udunits2::ud.convert`

Usage

```
ud_convert(x, u1, u2)
```

Arguments

x	vector of class "numeric" or "difftime"
u1	string parseable as the units in which x is provided. If x is class "difftime", then u1 is not actually used. However, it still needs to be supplied and needs to be convertible to u2 for consistency.
u2	string parseable as the units to convert to

Value

numeric vector with values converted to units in u2

Author(s)

Chris Black

units_are_equivalent *Check if two unit strings are equivalent*

Description

This is to allow multiple forms of the same unit to work, such as `m/s` vs. `m s-1` or `K` and `Kelvin`.

Usage

```
units_are_equivalent(x, y)
```

Arguments

x	A unit string, as character
y	Another unit string for comparison, as character

Value

`TRUE` if equivalent, `FALSE` otherwise

Author(s)

Alexey Shiklomanov

`unit_is_parseable` *Check whether a string can be interpreted as a unit*

Description

Function will replace the now-unmaintained `udunits2::ud.is.parseable`

Usage

```
unit_is_parseable(unit)
```

Arguments

`unit`	A character string representing a type of units

Value

TRUE if the units is parseable, FALSE otherwise.

Author(s)

Tanishq Jain

Examples

```
unit_is_parseable("g/sec^2")
unit_is_parseable("kiglometters")
```

`vecpaste` *Convert vector to comma delimited string*

Description

vecpaste, turns vector into comma delimited string fit for SQL statements.

Usage

```
vecpaste(x)
```

Arguments

`x`	vector

Value

comma delimited string

`zero.bounded.density` *Zero bounded density using log density transform*

Description

Provides a zero bounded density estimate of a parameter. Kernel Density Estimation used by the `density` function will cause problems at the left hand end because it will put some weight on negative values. One useful approach is to transform to logs, estimate the density using KDE, and then transform back.

Usage

```
zero.bounded.density(x, bw = "SJ", n = 1001)
```

Arguments

`x`	data, as a numeric vector
`bw`	The smoothing bandwidth to be used. See 'bw.nrd'
`n`	number of points to use in kernel density estimate. See `density`

Value

data frame with back-transformed log density estimate

Author(s)

Rob Hyndman

References

M. P. Wand, J. S. Marron and D. Ruppert, 1991. Transformations in Density Estimation. Journal of the American Statistical Association. 86(414):343-353 http://www.jstor.org/stable/2290569

`zero.truncate` *Zero Truncate*

Description

Truncates vector at 0

Usage

```
zero.truncate(y)
```

Arguments

y	numeric vector

Value

numeric vector with all values less than 0 set to 0

Author(s)

unknown

Index