Package: PEcAnRTM (via r-universe)

March 14, 2025

Type Package

Title PEcAn Functions Used for Radiative Transfer Modeling

Version 1.7.3.9000

Description Functions for performing forward runs and inversions of radiative transfer models (RTMs). Inversions can be performed using maximum likelihood, or more complex hierarchical Bayesian methods. Underlying numerical analyses are optimized for speed using Fortran code.

Depends R (>= 2.10)

Imports PEcAn.logger, MASS, coda, lubridate (>= 1.6.0), PEcAn.assim.batch, BayesianTools

Suggests minpack.lm, mclust, PEcAn.utils, PEcAn.ED2, XML, rmarkdown, testthat (>= 1.0.2), knitr (>= 1.42), pwr

OS_type unix

License BSD_3_clause + file LICENSE

Copyright Authors

LazyLoad yes

LazyData FALSE

Encoding UTF-8

VignetteBuilder knitr, rmarkdown

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Config/pak/sysreqs cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libmagick++-dev gsfonts jags libicu-dev libpng-dev libxml2-dev libnetcdf-dev libssl-dev libproj-dev python3 libsqlite3-dev libudunits2-dev libx11-dev xz-utils zlib1g-dev

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

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

RemoteRef HEAD

RemoteSha 97e61070b67901b2fa9aa727c73fdaf98a69a70c

RemoteSubdir modules/rtm

Contents

`burnin.thin` *Burn-in and thinning of MCMC samples*

Description

Burn-in and thinning of MCMC samples

Usage

```
burnin.thin(
  samples,
  target = 5000,
  burnin.ratio = 2,
  auto = TRUE,
  burnin = NULL,
  thin = NULL
)
```

Arguments

`samples`	Matrix of MCMC samples
`target`	Target number of samples (default = 5000). Only applicable if auto=TRUE.
`burnin.ratio`	Fraction of samples to burn-in; i.e. 2 means to remove first 1/2 of samples, 3 means 1/3, etc. (default = 2). Only applicable if auto=TRUE.
`auto`	Whether or not to perform automatic burnin and thin based on target number of samples.
`burnin`	Number of samples to discard as burnin (auto must be FALSE)
`thin`	Thinning interval (auto must be FALSE)

cbind.spectra *Combine spectra by wavelength*

Description

Combine spectra by wavelength

Usage

```
## S3 method for class 'spectra'
cbind(...)
```

Arguments

...	Spectra to combine

check.convergence *Check convergence of multiple MCMC chains*

Description

Uses Gelman multivariate Gelman-Rubin diagnostic to check if multiple MCMC chains have converged

Usage

```
check.convergence(jags_out, threshold = 1.1, verbose = TRUE, ...)
```

Arguments

jags_out	mcmc.list object (from coda package) containing samples from MCMC chains.
threshold	Gelman-Rubin diagnostic parameter threshold. Default = 1.1
verbose	If TRUE, print convergence result. Default = TRUE
...	Additional arguments to gelman.diag (from coda package)

Value

List length 3 containing the following: * convergence: Logical. Whether or not convergence was achieved. * diagnostics: Numerical value of Gelman-Rubin diagnostics for each parameter and multivariate diagnostic * error: Logical. Whether or not an error occured in the Gelman-Rubin calculation.

default.settings.prospect *Defult inversion settings for PROSPECT 5 models*

Description

Default settings for PROSPECT inversion

Usage

```
default.settings.prospect
```

Format

An object of class `list` of length 18.

defparam *Get default parameters*

Description

Extract default parameter values from `model.list`

Usage

```
defparam(modname)
```

Arguments

`modname`	Model name. Must match `modname` in `model.list`

Value

Named vector of default parameter values

dtnorm *Truncated normal distribution density*

Description

Calculates the log density of a univariate truncated normal variable

Usage

```
dtnorm(x, mu, sd, MIN)
```

Arguments

x	A random variable
mu	The mean parameter of the distribution; NOTE this is not equal to the mean
sd	The standard deviation parameter of the distribution
MIN	Value at which the truncation takes place

Value

Numeric; log density of the distribution, or -1e15 if the x < MIN

Author(s)

Alexey Shiklomanov

EDR *ED radiative transfer module (EDR) wrapper function*

Description

This function provides a convenient way to call the ED radiative transfer module (EDR, which simulates full spectral return of an ED patch for a given point in time) directly from R.

Usage

```
EDR(
  img_path,
  ed2in_path,
  spectra_list,
  trait.values,
  soil_reflect_path = system.file("extdata", "soil_reflect_par.dat", package =
    "PEcAnRTM"),
  wood_reflect_path = system.file("extdata", "wood_reflect_par.dat", package =
    "PEcAnRTM"),
```

```
  par.wl = 400:2499,
  nir.wl = 2500,
  edr_exe_path = NULL,
  output.path = dirname(normalizePath(ed2in_path, mustWork = TRUE)),
  settings = list(model = list(revision = "git", config.header = NULL)),
  singularity_args = list(),
  clean = FALSE,
  stderr = TRUE,
  verbose_error = TRUE,
  ...
)
```

Arguments

`img_path`	Path to Singularity container (usually a `.simg` file)
`ed2in_path`	Path to ED2IN file.
`spectra_list`	List of spectral data matrices. Names must exactly match the PFTs given in `trait.values`. Each item must be a matrix of reflectance (labeled 'R' or 'reflectance') and transmittance (labeled 'T' or 'transmittance') values. If the matrix is not of class `spectra` (see `spectra()`), it must also have a column of wavelength values (labeled 'wl'). Such a matrix is returned by default by all versions of PROSPECT in this package.
`trait.values`	Named, hierarchical list of trait values for generating config.xml file. Names must be PFTs, and must exactly match names of `spectra_list`.
`soil_reflect_path`	Path to soil reflectance file (defaults to spectra in package `extdata`)
`wood_reflect_path`	Path to wood reflectance file (defaults to spectra in package `extdata`)
`par.wl`	Vector of wavelengths defining PAR region
`nir.wl`	Vector of wavelengths defining NIR region
`edr_exe_path`	If `img_path` is `NULL`, a full path to the EDR executable. Ignored otherwise.
`output.path`	Directory in which to execute the run. Defaults to `dirname(ed2in_path)`.
`settings`	PEcAn settings list. Default is `list(model = list(revision = "git", config.header = NULL))`.
`singularity_args`	Additional arguments to be passed to `singularity run` (before)
`clean`	Logical. If `TRUE`, remove all files generated by this function (e.g. cloned history file, ED2IN, output HDF files).
`stderr`	Logical. If `TRUE` (default), internalize `system2` results as R character vector. `TRUE` is recommended because it allows EDR to check its execution and to run more quietly.
`verbose_error`	Logical. If `TRUE` (default), spit out the full ED if EDR execution fails.
`...`	Additional arguments to `system2`

Author(s)

Alexey Shiklomanov

`EDR.preprocess.history` *Preprocess history file for EDR*

Description

Locate history file based on path and prefix, copy to specified output directory, and rename to correct time.

Usage

```
EDR.preprocess.history(
  history.path,
  output.path,
  datetime,
  history.prefix = "history"
)
```

Arguments

`history.path`	Path to directory containing history file.
`output.path`	Directory in which to execute the run.
`datetime`	POSIX date and time for run
`history.prefix`	String describing the history file prefix in `history.path`. Default = 'history'

`fortran_data_module` *List to FORTRAN data module*

Description

Convert R list to a Fortran `/data/` module block

Usage

```
fortran_data_module(dat, types, modname, fname = paste0(modname, ".f90"))
```

Arguments

`dat`	Named list object for creating module. List names will be used to initialize FORTRAN variabiles.
`types`	Character vector of FORTAN types (e.g. real*8, integer)
`modname`	Name of the module. We suggest the format 'MOD_yourmodname'.
`fname`	Output file name. Defaults to 'yourmodname.f90'

Details

For models with large constants (e.g. absorption features in the PROSPECT model), it may be preferable to store these in FORTRAN90 modules. However, manually creating and formatting these files is tedious. This script allows you to automatically generate module files from R lists. It automatically interprets the object lengths as array dimensions (only vectors are supported right now – higher dimension arrays may be in the future) and splits long data into rows of 10. Currently, only numeric data are supported (i.e. no characters).

Author(s)

Alexey Shiklomanov

Examples

```
w <- 3.2
x <- 1:5
y <- 6:15
z <- seq(exp(1), pi, length.out=42)
l <- list(x=x, y=y, z=z) ## NOTE that names must be explicitly declared
l.types <- c('real','integer', 'real*4', 'real*8')
fortran_data_module(l, l.types, 'testmod',
  file.path(tempdir(), "testmod.f90"))

x <- runif(10)
y <- rnorm(10)
z <- rgamma(10, 3)
d <- data.frame(x,y,z) ## NOTE that data.frames are just named lists
d.types <- rep('real*8', ncol(d))
fortran_data_module(d, d.types, 'random',
  file.path(tempdir(), "random.f90"))
```

`foursail` *SAIL model*

Description

R wrapper for 4SAIL model

Usage

```
foursail(refl, tran, rsoil, param)
```

Arguments

`refl`	input leaf reflectance from 400-2500nm (can be measured or modeled)
`tran`	input leaf transmittance from 400-2500nm (can be measured or modeled)
`rsoil`	input soil reflectance spectra from 400-2500nm (can be measured or modeled)

param	Vector of SAIL parameter values: * LIDFa: Leaf angle distribution function - parameter a * LIDFb: Leaf angle distribution function - parameter b * TypeLIDF: Leaf angle distribution function type (1 or 2) * LAI: Leaf area index * q: Hot spot effect parameter * tts: Solar zenith angle * tto: Observer zenith angle * psi: Sun-sensor azimuth angle

Value

Spectra matrix (see spectra()) (2101 x 4) of reflectance factors for wavelengths 400 to 2500nm: * bi-hemispherical reflectance (rddt) * hemispherical directional (rsdt) * directional hemispherical (rdot) * bi-directional (rsot)

Author(s)

Shawn Serbin

Alexey Shiklomanov

---

generalized_plate_model

Generalized plate model

Description

This is the fundamental physical model underlying the PROSPECT family of leaf RTMs.

Usage

```
generalized_plate_model(k, refractive, N)
```

Arguments

k	Specific absorption coefficient (400 - 2500nm)
refractive	Refractive index (400 - 2500nm)
N	Effective number of mesophyll layers (see prospect())

generate.noise *Generate autocorrelated spectral noise*

Description

Generate autocorrelated spectral noise

Usage

```
generate.noise(n = 2101, sigma = 1e-04, fw = 201, fsd = 6)
```

Arguments

n	Length of output vector (default = 2101)
sigma	Gaussian noise standard deviation (default=1e-4)
fw	Filter width. Will be coerced to an odd number if even (default = 201).
fsd	Scaling factor for filter standard deviation (default = 6)

get.EDR.output *Read EDR output*

Description

Read EDR output

Usage

```
get.EDR.output(path = getwd())
```

Arguments

path	Path to directory containing albedo_par/nir.dat files

`invert.auto` *Inversion with automatic convergence checking*

Description

Inversion with automatic convergence checking

Usage

```
invert.auto(
  observed,
  invert.options,
  return.samples = TRUE,
  save.samples = NULL,
  quiet = FALSE,
  parallel = TRUE,
  parallel.cores = NULL,
  parallel.output = "/dev/null"
)
```

Arguments

`observed`	Vector, matrix, or data frame (coerced to matrix) of observed values. For spectral data, wavelengths are rows and spectra are columns. Dimensions must align with the output of `model`.
`invert.options`	Parameters related to inversion.
`return.samples`	Include full samples list in output. Default = `TRUE`.
`save.samples`	Save samples to file as the inversion proceeds (useful for debugging). If `NULL`, do not save samples. Default = `NULL`.
`quiet`	Suppress progress bar and status messages. Default=FALSE
`parallel`	Logical. Whether or not to run multiple chains in parallel on multiple cores (default = `TRUE`).
`parallel.cores`	Number of cores to use for parallelization. If `NULL` (default), allocate one fewer than detected number of cores.
`parallel.output`	Filename (or " for stdout) for printing parallel outputs. Use with caution. Default = `'/dev/null'`.

Details

Performs an inversion via the `invert.custom` function with multiple chains and automatic convergence checking. Convergence checks are performed using the multivariate Gelman-Rubin diagnostic.

Parameters specific to `invert.auto` are described here. For the remaining parameters, see `invert.custom()`.

- `model` – The model to be inverted. This should be an R function that takes `params` as input and returns one column of `observed` (nrows should be the same). Constants should be implicitly included here.
- `nchains` – Number of independent chains.
- `inits.function` – Function for generating initial conditions.
- `ngibbs.max` – Maximum number of total iterations (per chain). DEFAULT = 5e6
- `ngibbs.min` – Minimum number of total iterations (per chain). DEFAULT = 5000.
- `ngibbs.step` – Number of iterations between convergence checks. Default = 1000.
- `run_first` – Function to run before running sampling. Takes parallel inputs list containing runID, initial values, and resume (NULL) as an argument.
- `calculate.burnin` – If `TRUE`, use `PEcAn.assim.batch::autoburin` function to calculate burnin. Otherwise, assume burnin is `min(niter/2, iter_conv_check)`.
- `threshold` – Maximum value of the Gelman-Rubin diagnostic for determining convergence. Default = 1.1

Value

List including `results` (summary statistics), `samples` (`mcmc.list` object, or `NA` if `return.samples=FALSE`), and other information.

`invert.custom` *Bayesian inversion of a model*

Description

Performs an inversion of an arbitrary model using a modified Metropolis Hastings algorithm with block sampling. This may be slightly slower than the implementation in Fortran, but is much more customizable, as the model can be any R function.

Usage

```
invert.custom(
  observed,
  invert.options,
  quiet = FALSE,
  return.resume = FALSE,
  runID = NULL
)
```

Arguments

`observed`	Vector, matrix, or data frame (coerced to matrix) of observed values. For spectral data, wavelengths are rows and spectra are columns. Dimensions must align with the output of `model`.
`invert.options`	R list object containing inversion settings. See details.

quiet	Suppress progress bar and status messages. Default=FALSE
return.resume	If TRUE, return results as list that includes current Jump distribution (useful for continuing an ongoing run) and acceptance rate. Default = FALSE.
runID	Run-unique ID. Useful for parallel runs. Default=NULL

Details

inversion.options contains the following:

- inits – Vector of initial values of model parameters to be inverted.
- ngibbs – Number of MCMC iterations
- prior.function – Function for use as prior. Should take a vector of parameters as input and return a single value – the sum of their log-densities – as output.
- param.mins – Vector of minimum values for inversion parameters
- param.maxs – Vector of minimum values for inversion parameters
- model – The model to be inverted. This should be an R function that takes params and runID as input and returns one column of observed (nrows should be the same). Constants should be implicitly included here.
- adapt – Number of steps for adapting covariance matrix (i.e. adapt every 'n' steps). Default=100
- adj_min – Minimum threshold for rescaling Jump standard deviation. Default = 0.1.
- target – Target acceptance rate. Default=0.234, based on recommendation for multivariate block sampling in Haario et al. 2001
- do.lsq – Perform least squares optimization first (see invert.lsq), and use outputs to initialize Metropolis Hastings. This may improve mixing time, but risks getting caught in a local minimum. Default=FALSE
- catch_error – If TRUE (default), wrap model in tryCatch to prevent sampling termination on model execution error.

References

- Haario, Heikki; Saksman, Eero; Tamminen, Johanna. An adaptive Metropolis algorithm. Bernoulli 7 (2001), no. 2, 223–242. http://projecteuclid.org/euclid.bj/1080222083.

---

invert.lsq *Least squares model inversion*

Description

Performs a least-squares inversion of an arbitrary radiative transfer model (passed as an R function). The inversion attempts to minimize the sum of residual least squares between modeled and observed spectra via the Levenberg-Marquardt algorithm (nls.lm function from the minpack.lm package).

Usage

```
invert.lsq(observed, inits, model, lower = NULL, upper = NULL)
```

Arguments

observed	Vector of observations (e.g. a reflectance spectrum).
inits	Vector of initial conditions for the parameters.
model	An R function that calls the RTM and returns the error to be minimized. Be sure to include constants here.
lower	Lower bounds on parameters (default=NULL, which means -Inf).
upper	Upper bounds on parameters (default=NULL, which means +Inf).

Author(s)

Alexey Shiklomanov

---

invert_bt *Perform Bayesian inversion using BayesianTools package*

Description

Use samplers from the BayesianTools package to fit models to data. Like `invert.auto`, this will continue to run until convergence is achieved (based on Gelman diagnostic) *and* the result has enough samples (as specified by the user; see Details).

Usage

```
invert_bt(observed, model, prior, custom_settings = list(), loglike = NULL)
```

Arguments

observed	Vector of observations. Ignored if `loglike` is not `NULL`.
model	Function called by log-likelihood. Must be `function(params)` and return a vector equal to `length(observed)` or `nrow(observed)`. Ignored if `loglike` is not `NULL`.
prior	BayesianTools prior object.
custom_settings	Nested settings list. See Details.
loglike	Custom log likelihood function. If `NULL`, use `rtm_loglike()` with provided `observed` and `model`.

Details

`custom_settings` is a list of lists, containing the following:

- `common` – BayesianTools settings common to both the initial and subsequent samples.
- `init` – BayesianTools settings for just the first round of sampling. This is most common for the initial number of iterations, which is the minimum expected for convergence.
- `loop` – BayesianTools settings for iterations inside the convergence checking `while` loop. This is most commonly for setting a smaller iteration count than in `init`.
- `other` – Miscellaneous (non-BayesianTools) settings, including:
  - `sampler` – String describing which sampler to use. Default is `DEzs`
  - `use_mpsrf` – Use the multivariate PSRF to check convergence. Default is `FALSE` because it may be an excessively conservative diagnostic.
  - `min_samp` – Minimum number of samples after burnin before stopping. Default is 5000.
  - `max_iter` – Maximum total number of iterations. Default is 1e6.
  - `lag.max` – Maximum lag to use for autocorrelation normalization. Default is `10 * log10(n)` (same as `stats::acf` function).
  - `save_progress` – File name for saving samples between loop iterations. If `NULL` (default), do not save progress samples.
  - `threshold` – Threshold for Gelman PSRF convergence diagnostic. Default is 1.1.
  - `verbose_loglike` – Diagnostic messages in log likelihood output. Default is TRUE.

See the BayesianTools sampler documentation for what can go in the `BayesianTools` settings lists.

`load.from.name` *Load object from an RData file*

Description

Load object from an RData file

Usage

```
load.from.name(filename, filepath = ".")
```

Arguments

`filename`	Full name (without path!) of RData file
`filepath`	Path of RData file (default='.')

lognorm.mu *Functions for default priors Lognormal mean parameters*

Description

Functions for default priors Lognormal mean parameters

Usage

```
lognorm.mu(mean, sd)
```

Arguments

mean	Sample mean
sd	Sample standard deviation

lognorm.sigma *Lognormal sigma parameter*

Description

Lognormal sigma parameter

Usage

```
lognorm.sigma(mean, sd)
```

Arguments

mean	Sample mean
sd	Sample standard deviation

matplot *Matplot generic method*

Description

Matplot generic method

Usage

```
matplot(...)
```

Arguments

...	arguments passed to methods

matplot.default *Matplot default method*

Description

Matplot default method

Usage

```
## Default S3 method:
matplot(...)
```

Arguments

...	arguments passed to matplot

matplot.spectra *Plot multiple spectra on same graph*

Description

Plot multiple spectra on same graph

Usage

```
## S3 method for class 'spectra'
matplot(spectra, ...)
```

Arguments

spectra	Vector (length = length(wavelengths)) or matrix (ncol = length(wavelengths))
...	Additional arguments to matplot

neff *Effective sample size*

Description

Calculate effective sample size of vector based on its autocorrelation.

Usage

```
neff(x, ...)
```

Arguments

x	A vector or time series
...	additional arguments passed to methods

params.prospect4 *PROSPECT 4 parameters*

Description

Shortcut lists for PROSPECT parameter names

Usage

```
params.prospect4
```

Format

An object of class character of length 4.

params.prospect5 *PROSPECT 5 parameters*

Description

PROSPECT 5 parameters

Usage

```
params.prospect5
```

Format

An object of class character of length 5.

params.prospect5b *PROSPECT 5B parameters*

Description

PROSPECT 5B parameters

Usage

```
params.prospect5b
```

Format

An object of class `character` of length 6.

params.prospectd *PROSPECT D parameters*

Description

PROSPECT D parameters

Usage

```
params.prospectd
```

Format

An object of class `character` of length 7.

params2edr *Convert named parameter vector to EDR-compatible inputs*

Description

Creates a nested list whose components are suitable for passing to EDR.

Usage

```
params2edr(params, sep = ".", prospect = TRUE, version = 5)
```

Arguments

params	Named parameter vector
sep	Separator between PFT name and trait name. Must be a single character (default = ".").
prospect	Logical. If TRUE (default), scan for PROSPECT traits and pass them to PROSPECT.
version	PROSPECT version

Details

If prospect = TRUE, parameters prefixed with prospect_ are passed to prospect with the specified version, and other parameters are passed to trait.values.

The regular expression defining the separation is greedy, i.e. temperate.Early_Hardwood.SLA will separate into temperate.Early_Hardwood and SLA (assuming the default sep = "."). Therefore, it is crucial that trait names not contain any sep characters (neither ED nor PROSPECT parameters should anyway). If this is a problem, use an alternate separator (e.g. |).

Note that using sep = "." allows this function to directly invert the default behavior of unlist. That is, calling unlist(params2edr(params, prospect = FALSE)$trait.values) will return the input vector of trait values. This makes unlist a convenient way to go from a trait.values list to a properly formatted params vector.

Because unused ED parameters in the config.xml are ignored, the PROSPECT parameters are saved in the trait.values object as well, which may be useful for debugging.

Value

List containing spectra_list and trait.values, both objects needed by EDR.

Author(s)

Alexey Shiklomanov

---

plot.spectra *Plot spectra vs. wavelength*

Description

Plot spectra vs. wavelength

Usage

```
## S3 method for class 'spectra'
plot(spectra, type = "l", ...)
```

Arguments

spectra	Vector (length = length(wavelengths)) or matrix (ncol = length(wavelengths))
type	plot style, e.g. "l" for lines, "b" for lines and points
...	Additional arguments to plot

print.spectra *Print method for spectra S3 class*

Description

Print method for spectra S3 class

Usage

```
## S3 method for class 'spectra'
print(spectra, n = 10, ...)
```

Arguments

spectra	Object of class spectra
n	Max number of rows to print (show first n/2 and last n/2 rows)
...	Additional arguments to print

print_results_summary *Neatly print inversion results summary*

Description

Neatly print inversion results summary

Usage

```
print_results_summary(output)
```

Arguments

output	Output from invert.auto

Author(s)

Alexey Shiklomanov

prior.defaultvals.prospect *Default prior parameters for PROSPECT models*

Description

Default prior parameters for PROSPECT models

Usage

```
prior.defaultvals.prospect(sd.inflate = 3)
```

Arguments

sd.inflate	Standard deviation multiplier (default = 3)

priorfunc.prospect *Default PROSPECT 5 prior function*

Description

Default PROSPECT 5 prior function

Usage

```
priorfunc.prospect(pmu, psigma)
```

Arguments

pmu	Lognormal mu parameter
psigma	Lognormal sigma parameter

Details

Assumes lognormal distribution for all parameters. NOTE that prior on N is shifted by 1.

pro2s *Coupled PROSPECT-Two-stream model*

Description

Coupled PROSPECT-Two-stream model

Usage

```
pro2s(param, prospect.version = 5)
```

Arguments

`param`	Model parameters, in the following order: N, Cab, (Car, Cbrown), Cw, Cm, solar zenith angle, LAI, soil_moisture
`prospect.version`	Version of PROSPECT to use (4, 5, or '5B'; default=5)

Value

Spectra matrix (see `spectra()`) for wavelengths 400 to 2500nm containing the following columns:

- `alpha.c` – Direct ("collimated") reflectance
- `alpha.i` – Diffuse ("isotropic") reflectance
- `Tc` – Direct transmittance to background
- `Ti` – Diffuse reflectance to background
- `Ac` – Direct absorbance by canopy
- `Ai` – Diffuse absorbance by canopy

pro4sail *PRO4SAIL model*

Description

R wrapper for PRO4SAIL model

Usage

```
pro4sail(param)
```

Arguments

param	Vector of PRO4SAIL parameter values: * N: Effective number of leaf layers (>1) * Cab: Leaf chlorophyll content (ug/cm2) (>0) * Car: Leaf carotenoid content (ug/cm2) (>0) * Cbrown: Leaf brown matter content (ug/cm2) (>0) * Cw: Leaf water content (cm) (>0) * Cm: Leaf dry matter content (ug/cm2) (>0) * LIDFa: Leaf angle distribution function - parameter a * LIDFb: Leaf angle distribution function - parameter b * TypeLIDF: Leaf angle distribution function type (1 or 2) * LAI: Leaf area index * q: Hot spot effect parameter * tts: Solar zenith angle * tto: Observer zenith angle * psi: Sun-sensor azimuth angle * psoil: Fraction of soil moisture

Value

Spectra matrix (see spectra()) (2101 x 4) of reflectance factors for wavelengths 400 to 2500nm: * bi-hemispherical reflectance (rddt) * hemispherical directional (rsdt) * directional hemispherical (rdot) * bi-directional (rsot)

Author(s)

Alexey Shiklomanov

---

pro4saild *PRO4SAILD model*

Description

R wrapper for PRO4SAILD model

Usage

```
pro4saild(param)
```

Arguments

param	Vector of PRO4SAIL parameter values: * N: Effective number of leaf layers (>1) * Cab: Leaf chlorophyll content (ug/cm2) (>0) * Car: Leaf carotenoid content (ug/cm2) (>0) * Canth: Leaf anthocyanin content (ug/cm2) (>0) * Cbrown: Leaf brown matter content (ug/cm2) (>0) * Cw: Leaf water content (cm) (>0) * Cm: Leaf dry matter content (ug/cm2) (>0) * LIDFa: Leaf angle distribution function - parameter a * LIDFb: Leaf angle distribution function - parameter b * TypeLIDF: Leaf angle distribution function type (1 or 2) * LAI: Leaf area index * q: Hot spot effect parameter * tts: Solar zenith angle * tto: Observer zenith angle * psi: Sun-sensor azimuth angle * psoil: Fraction of soil moisture

Value

Spectra matrix (see spectra()) (2101 x 4) of reflectance factors for wavelengths 400 to 2500nm: * bi-hemispherical reflectance (rddt) * hemispherical directional (rsdt) * directional hemispherical (rdot) * bi-directional (rsot)

Author(s)

Alexey Shiklomanov, Shawn Serbin

`prospect` *PROSPECT (4, 5, or 5B) model*

Description

R wrapper for PROSPECT models

Usage

```
prospect(param, version)
```

Arguments

`param`	Vector of PROSPECT parameter values: * N: Effective number of leaf layers (>1) * Cab: Leaf chlorophyll content (ug/cm2) (>0) * (5) Car: Leaf carotenoid content (ug/cm2) (>0) * (D) Canth: Leaf anthocyanin content (ug/cm2) (>0) * (5B, D) Cbrown: Leaf brown matter content (ug/cm2) (>0) * Cw: Leaf water content (cm) (>0) * Cm: Leaf dry matter content (ug/cm2) (>0)
`version`	PROSPECT version: 4, 5, or '5B'

Value

Object of class `spectra` (see `spectra()`) with simulated reflectance (column 1) and transmittance (column 2) from 400 to 2500 nm

Author(s)

Alexey Shiklomanov

`prospect_bt_prior` *Quick BayesianTools prior creator for PROSPECT model*

Description

Quick BayesianTools prior creator for PROSPECT model

Usage

```
prospect_bt_prior(version, custom_prior = list())
```

Arguments

`version`	PROSPECT version: 4, 5, or '5B'
`custom_prior`	List containing `param_name`, `distn`, `parama`, `paramb`, and `lower`

read.rsr.folder *Read and process RSR data from directory*

Description

Read and process RSR data from directory

Usage

```
read.rsr.folder(dir.path, type)
```

Arguments

dir.path	Directory containing RSR data
type	Type of sensor. Options are: landsat, avhrr

resample *Resample vector, matrix, or spectra*

Description

Convenient wrapper around base R's splinefun and approxfun. See stats::splinefun() and stats::approxfun() documentation for information on different spline methods and additional arguments.

Usage

```
resample(values, ...)

## Default S3 method:
resample(values, from, to, method = "fmm", ...)

## S3 method for class 'matrix'
resample(values, from, to, ...)

## S3 method for class 'spectra'
resample(values, to, ...)
```

Arguments

values	Vector or matrix of values, or object of class spectra(). Length of vector, or nrow of matrix must match length of from. For spectra, from argument is omitted because it is taken from wavelengths.
...	Additional arguments to stats::splinefun() or stats::approxfun()

from	X values for interpolation (for spectra objects, this is assumed to be the wavelengths attribute.)
to	Y values onto which to interpolate. For spectra objects, this should be new wavelengths.
method	One of the methods for stats::splinefun() (for polynomial and periodic splines) or stats::approxfun() (for constant or linear). Default is "fmm" (same as splinefun).

Value

Object of the same class as values, resampled to the to values.

`rsr.from.fwhm` *Generate relative spectral response (RSR) matrix based on FWHM data*

Description

Generate relative spectral response (RSR) matrix based on FWHM data

Usage

```
rsr.from.fwhm(wavelength, fwhm)
```

Arguments

wavelength	Vector of average band widths, as reported in FWHM data.
fwhm	Vector of full-width half maximum (FWHM) bandwidths, as reported in FWHM data.

`rtm_loglike` *Generic log-likelihood generator for RTMs*

Description

Generic log-likelihood generator for RTMs

Usage

```
rtm_loglike(nparams, model, observed, lag.max = NULL, verbose = TRUE, ...)
```

Arguments

nparams	number of parameters in model
model	function to minimize
observed	vector of observations
lag.max	passed to neff()
verbose	logical: print extra diagnostics during run?
...	additional arguments passed to model function

---

rtnorm *Random sampling from one-sided truncated normal distribution*

Description

Random sampling from one-sided truncated normal distribution

Usage

```
rtnorm(mu, sd, MIN)
```

Arguments

mu	The mean parameter of the truncated normal. NOTE that this is NOT equal to the mean of the distribution
sd	The standard deviation parameter of the truncated normal. Again, NOTE tht this is not the SD of the distribution.
MIN	The minimum value, which defines the truncation.

Details

Draws a random number and, if it doesn't fall within the specified range, resample using an adjusted Normal CDF. This isn't performed immediately because CDF sampling calls three functions – qnorm, runif, and pnorm–and therefore is much less efficient than a simple random sample.

Value

Numeric length one.

Author(s)

Alexey Shiklomanov

sensor.list *Sensor spectral response functions*

Description

Sensor spectral response functions

Usage

```
sensor.list
```

Format

An object of class `character` of length 12.

sensor.proper *Sensor list with proper names*

Description

Sensor list with proper names

Usage

```
sensor.proper
```

Format

An object of class `character` of length 12.

setup_edr *Setup EDR run*

Description

Using an existing ED2IN file as a template, create a new ED2IN and history file configured for running EDR.

Usage

```
setup_edr(
  ed2in,
  output_dir,
 datetime = ISOdatetime(ed2in[["IYEARA"]], ed2in[["IMONTHA"]], ed2in[["IDATEA"]], 12, 0,
    0, tz = "UTC"),
  ...
)
```

Arguments

ed2in	ED2IN list object (see PEcAn.ED2::read_ed2in).
output_dir	Directory in which run files will be stored
datetime	Date time object (or compliant string) at which to run EDR. Defaults to 12 noon on start date in ED2IN.
...	Additional arguments passed on to PEcAn.ED2::modify_ed2in

Value

Path to EDR-configured ED2IN file.

Author(s)

Alexey Shiklomanov

---

spectra *Spectra S3 class*

Description

Spectra S3 class

Usage

```
spectra(spectra, wavelengths = 400:2500)

is_spectra(spectra)
```

Arguments

spectra	Vector (length = length(wavelengths)) or matrix (ncol = length(wavelengths))
wavelengths	Wavelengths of spectra.

Functions

- is_spectra(): Test if object is class spectra

spectral.response *Convolution of spectra to sensor RSR*

Description

Convolution of spectra to sensor RSR

Usage

```
spectral.response(spec, sensor)
```

Arguments

spec	Full (1 nm) spectrum (vector)
sensor	Sensor name (string). See sensor.list

str.spectra *Structure of* spectra *object*

Description

Structure of spectra object

Usage

```
## S3 method for class 'spectra'
str(spectra, ...)
```

Arguments

spectra	Object of class spectra
...	additional arguments, currently ignored

summary_mvnorm *Multivariate normal fit*

Description

Fit multivariate normal to samples. Return means and covariance matrix as a long list (for easy construction of data.tables)

Usage

```
summary_mvnorm(samples)
```

Arguments

samples	Matrix of MCMC samples.

summary_simple *Simple summary statistics on MCMC samples*

Description

Calculate simple univariate summary statistics and return as named list

Usage

```
summary_simple(samples)
```

Arguments

samples	Matrix of MCMC samples

trim.rsr *Trim RSR matrix to wavelength limits*

Description

Trim RSR matrix to wavelength limits

Usage

```
trim.rsr(rsr, wl.min = 400, wl.max = 2500)
```

Arguments

rsr	RSR matrix
wl.min	Minimum wavelength (inclusive, default = 400)
wl.max	Maximum wavelength (inclusive, default = 2500)

wavelengths *Retrieve wavelengths from spectra object*

Description

Retrieve wavelengths from spectra object

Usage

```
wavelengths(spectra)
```

Arguments

spectra	Object of class spectra

[.spectra *Select spectra*

Description

Select spectra

Usage

```
## S3 method for class 'spectra'
spectra[i, j, drop = FALSE]
```

Arguments

spectra	Object of class spectra
i, j	indices specifying elements to extract or replace. See base::Extract for details.
drop	Coerce result to the lowest dimension possible? See base::drop() for details.

[[.spectra *Select spectra by wavelength*

Description

Select spectra by wavelength

Usage

```
## S3 method for class 'spectra'
spectra[[wavelength, j]]
```

Arguments

spectra	Object of class spectra
wavelength	Wavelength vector to select
j	index specifying elements to extract or replace. See base::Extract for details.

[[<-.spectra *Assign values to spectra by wavelength*

Description

Assign values to spectra by wavelength

Usage

```
## S3 replacement method for class 'spectra'
spectra[[wavelength, j]] <- value
```

Arguments

spectra	Object of class spectra
wavelength	Wavelength vector to select
j	index specifying elements to extract or replace. See base::Extract for details.
value	Vector or matrix of values to assign

Index