Package: PEcAn.data.land (via r-universe)

September 18, 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.

Depends R (>= 3.5.0)

Imports coda, curl, dplyr, dplR, fs, future, furrr, httr, lubridate, magrittr, mvtnorm, ncdf4 (>= 1.15), neonUtilities, neonstore, swfscMisc, PEcAn.benchmark, PEcAn.DB, PEcAn.logger, PEcAn.remote, PEcAn.utils, PEcAn.visualization, purrr, rjags, rlang, sf, sirt, sp, stringr, terra, tidyr, tidyselect, traits, XML (>= 3.98-1.4)

Suggests dataone, datapack, getPass, glue, PEcAn.settings, redland, raster, reticulate, testthat (>= 1.0.2)

License BSD_3_clause + file LICENSE

Copyright Authors

LazyData true

Encoding UTF-8

RoxygenNote 7.3.2

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

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

RemoteRef HEAD

RemoteSha f22a7c4bbc532e4551f7bc9624cef649da317ac1

Contents

`BADM` *Biomass and soil data from FluxNet sites*

Description

Contains data from 246 Fluxnet sites. Variables include aboveground and belowground biomass in various pools, plus soil texture/chemistry/horizonation/C&N stocks.

Usage

```
BADM
```

Format

## 'BADM' A data frame with 12,300 rows and 13 columns:

SITE_ID Fluxnet code for the site

LOCATION_ELEV, LOCATION_LAT, LOCATION_LON site coordinates

Date Measurement date

GROUP_ID TODO

VARIABLE_GROUP category, eg abovground biomass or soil chemistry

VARIABLE, DATAVALUE key and value for each measured variable

NA_L1CODE, NA_L1NAME, NA_L2CODE, NA_L2NAME numeric IDs and names for the Level 1 and level 2 ecoregions where this site is located

Source

Originally from Fluxnet <https://fluxnet.org/badm-data-product/>, but the provenence and age of this specific file is not clear.

`BADM_IC_process` *BADM_IC_process*

Description

BADM_IC_process

Usage

```
BADM_IC_process(settings, dir, overwrite = TRUE)
```

Arguments

`settings`	pecan xml settings
`dir`	output dir which you want to store the IC netcdf file
`overwrite`	Flag for overwriting the IC file.

Value

a list of paths to generated and stored IC files.

`buildJAGSdata_InventoryRings` *Format ring & plot data for JAGA*

Description

builds the JAGS data object for the tree ring / inventory fusion code also sets all the priors

Usage

```
buildJAGSdata_InventoryRings(combined, inc.unit.conv = 0.1)
```

Arguments

`combined`	object returned from matchInventoryRings. Matrix with both increment and plot data
`inc.unit.conv`	conversion factor from loaded increments to cm (of radius)

Value

list

Author(s)

Michael Dietze

`Clean_Tucson` *Clean_Tucson*

Description

tree core QAQC

Usage

```
Clean_Tucson(file)
```

Arguments

`file`	WinDendro output

`cohort2pool` *cohort2pool*

Description

Converts .rds files into pool netcdf files.

Usage

```
cohort2pool(dat, allom_param = NULL, dbh_name = "DBH")
```

Arguments

`dat`	veg_info file
`allom_param`	parameters for allometric equation, a and b. Based on base-10 log-log linear model (power law)
`dbh_name`	Default is "DBH". This is the column name in the veg_file that represents DBH. May differ depending on data source.

Details

cohort2pool function Calculates total biomass using veg cohort file.

Author(s)

Saloni Shah

Examples

```
## Not run:
veg_file <- "~/downloads/FFT_site_1-25665/FFT.2008.veg.rds"
cohort2pool(veg_File = veg_file, allom_param = NULL)

## End(Not run)
```

---

dataone_download *DataONE download*

Description

Adapts the dataone::getDataPackage workflow to allow users to download data from the DataONE federation by simply entering the doi or associated package id

Usage

```
dataone_download(
  id,
  filepath = "/fs/data1/pecan.data/dbfiles",
  CNode = "PROD",
  lazyLoad = FALSE,
  quiet = FALSE
)
```

Arguments

id	"The identifier of a package, package metadata or other package member" – dataone r
filepath	path to where files will be stored
CNode	character, passed to ‘dataone::CNode‘
lazyLoad	"A logical value. If TRUE, then only package member system metadata is downloaded and not data. The default is FALSE." – dataone R
quiet	"A ’logical’. If TRUE (the default) then informational messages will not be printed." – dataone R

Author(s)

Liam P Burke, <lpburke@bu.edu>

Examples

```
## Not run:
dataone_download(id = "doi:10.6073/pasta/63ad7159306bc031520f09b2faefcf87",
filepath = "/fs/data1/pecan.data/dbfiles")

## End(Not run)
```

`download.SM_CDS` *Download CDS soil moisture data for the SDA workflow.*

Description

Download CDS soil moisture data for the SDA workflow.

Usage

```
download.SM_CDS(
  outfolder,
  time.points,
  overwrite = FALSE,
  auto.create.key = FALSE
)
```

Arguments

`outfolder`	physical paths to where the unziped soil moisture files are downloaded.
`time.points`	A vector contains each time point within the start and end date.
`overwrite`	flag determine if we want to overwrite existing files when downloading.
`auto.create.key`	flag determine if we want to automatically create the credential file.

Details

Introduction on how to play with the CDS python API to correctly build the python environment with the cdsapi installed, you need to follow those steps. 1. Install miniconda. create a directory to install minicaonda ‘mkdir -p ~/miniconda3‘ 2. Download latest miniconda version. ‘wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh -O ~/miniconda3/miniconda.sh‘ 3. run the install script. ‘bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3‘ 4. delete the intall script. ‘rm -rf ~/miniconda3/miniconda.sh‘ 5. add a conda initialize to your bash ‘~/miniconda3/bin/conda init bash‘ 6. Verify the installaton, you need to restart your session first. ‘conda list‘ 7. Create Python environment. ‘conda update conda‘ ‘conda create -n myenv python=3.9 –yes‘ 8. Activate your python env. ‘conda activate myenv‘ 9. Install the cdsapi package. ‘pip install cdsapi‘ in the meantime, you might encounter several issues saying XXXX dependency is not available. to solve this issue, you just need to install those dependencies before hand. 10. Create CDS account. go to ‘https://cds.climate.copernicus.eu/api-how-to#install-the-cds-api-key‘ website. create an account. 11. Create CDS personel token. run this function. go to ‘https://cds.climate.copernicus.eu/api-how-to#install-the-cds-api-key‘ website. copy and paste url and key to the prompt window.

Value

A vector containing file paths to the downloaded files.

Author(s)

Dongchen Zhang

`download_NEON_soilmoist` *Download NEON Soil Water Content and Soil Salinity data by date and site name*

Description

Download NEON Soil Water Content and Soil Salinity data by date and site name

Usage

```
download_NEON_soilmoist(
  site,
  avg = "all",
  var = "all",
  startdate = NA,
  enddate = NA,
  outdir
)
```

Arguments

`site`	four letter NEON site code name(s). If no site is specified, it will download all of them (chr) (e.g "BART" or c("SRER", "KONA", "BART"))
`avg`	averaging interval (minutes): 1, 30, or both ("all") . default returns both
`var`	variable of interest: "SWC" (soil water content) or "SIC" (soil ion content) or both ("all") default returns both. Both variables will be saved in outdir automatically (chr)
`startdate`	start date as YYYY-mm. If left empty, all data available will be downloaded (chr)
`enddate`	start date as YYYY-mm. If left empty, all data available will be downloaded (chr)
`outdir`	out directory to store the following data: .rds list files of SWC and SIC data for each site and sensor position, sensor positions .csv for each site, variable description .csv file, readme .csv file

Value

List of specified variable(s) AND prints the path to output folder

Author(s)

Juliette Bateman

Examples

```
## Not run:
test <- download_NEON_soilmoisture(
  site = c("SRER", "BART", "KONA"),
  avg = 30,
  var = "SWC",
  startdate = "2019-01",
  enddate = "2020-01",
  outdir = getwd())
## End(Not run)
```

---

download_package_rm *download_packages*

Description

Uses resource_map and dataone::getPackage to download the data into a BagItFile. Then utils::unzip unzips the data and stores in the user's directory.

Usage

```
download_package_rm(
  resource_map,
  directory,
  CNode = "PROD",
  download_format = "application/bagit-097",
  overwrite_directory = TRUE
)
```

Arguments

`resource_map`	the resource map that corresponds to the given data package
`directory`	location that download.packages places the data
`CNode`	defaults to "PROD"
`download_format`	typically "application/bagit-097". Other possible formats currently unknown.
`overwrite_directory`	boolean that indicates whether or not the function should overwrite the directory

Value

results of download

ens_veg_module *Sampling/ensemble module*

Description

Sampling/ensemble module

Usage

```
ens_veg_module(
  getveg.id,
  dbparms,
  input_veg,
  outfolder,
  machine,
  start_date,
  end_date,
  n.ensemble,
  new_site,
  host
)
```

Arguments

getveg.id	list, input.id and dbfile.id of the IC file in intermediate pecan standard
dbparms	list, settings$database info reqired for opening a connection to DB
input_veg	list, this is a sublist of settings$run$inputs that has info about source, id, metadata of the requested IC file
outfolder	path to where the processed files will be written
machine	data frame, DB info regarding localhost machine id/hostname etc.
start_date	date in "YYYY-MM-DD" format, in case of source==FIA it's the settings$run$start.date, otherwise start_date of the IC file in DB
end_date	date in "YYYY-MM-DD" format, in case of source==FIA it's the settings$run$end.date, otherwise end_date of the IC file in DB
n.ensemble	integer, ensemble member number
new_site	data frame, id/lat/lon/name info about the site
host	list, host info as in settings$host, host$name forced to be "localhost" upstream

Author(s)

Istem Fer

`EPA_ecoregion_finder` *EPA_ecoregion_finder*

Description

This function is designed to find the level1 and level2 code ecoregions for a given lat and long. You can learn more about ecoregions here: `https://www.epa.gov/eco-research/ecoregions`.

Usage

```
EPA_ecoregion_finder(Lat, Lon, folder.path = NULL)
```

Arguments

`Lat`	numeric latitude
`Lon`	numeric longitude
`folder.path`	path to the directory where you store the shape files of L1 and L2 ecoregion maps.

Value

a dataframe with codes corresponding to level1 and level2 codes as two columns

`extract.stringCode` *extract.stringCode*

Description

extract.stringCode

Usage

```
extract.stringCode(x, extractor = from.TreeCode)
```

Arguments

`x`	string to decode
`extractor`	function to apply

extract_FIA *extract_FIA*

Description

extract_FIA

Usage

```
extract_FIA(lon, lat, start_date, end_date, gridres = 0.075, dbparms, ...)
```

Arguments

`lon`	site longitude
`lat`	site latitude
`start_date`	"YYYY-MM-DD"
`end_date`	"YYYY-MM-DD"
`gridres`	taken from input_veg, DEFAULT = 0.075
`dbparms`	taken from settings object
`...`	Additional parameters

Author(s)

Istem Fer

extract_NEON_veg *extract_NEON_veg*

Description

extract_NEON_veg

Usage

```
extract_NEON_veg(
  lon,
  lat,
  start_date,
  end_date,
  store_dir,
  neonsites = NULL,
  ...
)
```

Arguments

`lon`	site longitude, passed from ic_process
`lat`	site latitude, passed from ic_process
`start_date`	"YYYY-MM-DD", used to download NEON datasets for desired time period
`end_date`	"YYYY_MM_DD", used to download NEON datasets for desired time period
`store_dir`	location where you want to store downloaded NEON files
`neonsites`	prepared datasets table from NEON using neonstore::neon_sites(api = "https://data.neonscience.org/api/v(.token = Sys.getenv("NEON_TOKEN"))
`...`	Additional parameters

Value

veg_info object to be passed to extract_veg within ic_process

Author(s)

Alexis Helgeson and Michael Dietze

Examples

```
start_date = as.Date("2020-01-01")
end_date = as.Date("2021-09-01")
```

`extract_SM_CDS` *Extract CDS soil moisture data for the SDA workflow.*

Description

Extract CDS soil moisture data for the SDA workflow.

Usage

```
extract_SM_CDS(
  site_info,
  time.points,
  in.path,
  out.path = NULL,
  allow.download = TRUE,
  search_window = 10
)
```

Arguments

`site_info`	Bety list of site info including site_id, lon, and lat.
`time.points`	A vector contains each time point within the start and end date.
`in.path`	physical paths to where the unziped soil moisture files are downloaded.
`out.path`	Where the final CSV file will be stored.
`allow.download`	Flag determine if we want to automatic download files if they are not available.
`search_window`	time length (days) for locate available soil moisture values.

Value

A data frame containing soil moisture and sd for each site and each time step.

Author(s)

Dongchen Zhang

`extract_soil_gssurgo` *Extract soil data from gssurgo*

Description

Extract soil data from gssurgo

Usage

```
extract_soil_gssurgo(
  outdir,
  lat,
  lon,
  size = 1,
  radius = 500,
  depths = c(0.15, 0.3, 0.6)
)
```

Arguments

`outdir`	Output directory for writing down the netcdf file
`lat`	Latitude
`lon`	Longitude
`size`	Ensemble size
`radius`	radius in meters is used to take soil type samples around the site
`depths`	Standard set of soil depths in m to create the ensemble of soil profiles with.

Value

It returns the address for the generated soil netcdf file

Author(s)

Hamze Dokoohaki

Examples

```
## Not run:
   outdir  <- "~/paleon/envTest"
   lat     <- 40
   lon     <- -80
   PEcAn.data.land::extract_soil_gssurgo(outdir, lat, lon)

## End(Not run)
```

`extract_soil_nc` *Extract soil data from the gridpoint closest to a location*

Description

Extract soil data from the gridpoint closest to a location

Usage

```
extract_soil_nc(in.file, outdir, lat, lon)
```

Arguments

`in.file`	path to netcdf file containing soil data
`outdir`	directory in which to write netcdf file of extracted data. Output filename will be the same as input filename.
`lat, lon`	location in decimal degrees. Data will be extracted from the point in ‘in.file‘ that is nearest this

Value

path to netCDF file containing extracted data

Examples

```
## Not run:
in.file <- "~/paleon/env_paleon/soil/paleon_soil.nc"
outdir  <- "~/paleon/envTest"
lat     <- 40
lon     <- -80
PEcAn.data.land::extract_soil_nc(in.file,outdir,lat,lon)

## End(Not run)
```

---

extract_veg *extract_veg*

Description

Function queries a DB to extract veg info downstream

Usage

```
extract_veg(
  new_site,
  start_date,
  end_date,
  source,
  gridres,
  format_name = NULL,
  machine_host,
  dbparms,
  outfolder,
  overwrite = FALSE,
  input_veg = input_veg,
  ...
)
```

Arguments

`new_site`	new_site object passed from ic_process includes lat, lon, id, and name
`start_date`	"YYYY-MM-DD"
`end_date`	"YYYY-MM-DD"
`source`	taken from input$source, passed from ic_process
`gridres`	only used for source = "FIA"
`format_name`	DEFAULT=NULL
`machine_host`	passed from ic_process
`dbparms`	taken from settings object, passed from ic_process
`outfolder`	passed from ic_process, location where to store files

`overwrite`	DEFAULT = FALSE
`input_veg`	passed from input object in ic_process
`...`	Additional parameters

Value

results object to be passed back to get.veg.module

Author(s)

Istem Fer and Alexis Helgeson

`fia.to.psscss` *Create pss/css files based on data in the fia database*

Description

Create pss/css files based on data in the fia database

Usage

```
fia.to.psscss(
  settings,
  lat = as.numeric(settings$run$site$lat),
  lon = as.numeric(settings$run$site$lon),
  year = lubridate::year(settings$run$start.date),
  gridres = 0.075,
  min.year = year - 5,
  max.year = year + 5,
  overwrite = FALSE
)
```

Arguments

`settings`	PEcAn settings object
`lat`, `lon`	site location in decimal degrees. Defults to values passed in ‘settings‘.
`year`	defaults to year of start date passed in settings
`gridres`	grid resolution in degrees
`min.year`, `max.year`	limits on years of FIA data to look for
`overwrite`	logical: regenerate files already in the database?

Value

modified settings, invisibly

Author(s)

Mike Dietze, Rob Kooper, Ryan Kelly

`format_identifier` *format_identifier*

Description

This function is for formatting purposes. It simply inserts the doi or id that the user wishes to query into Solr format so that it is compatible with the dataoneR query functionality in the PEcAn function

Usage

```
format_identifier(id)
```

Arguments

`id`	the doi or other identifier linked to the package in DataONE

Value

returns the id in the proper format for querying the DataONE Federation (using solrQuery syntax)

Author(s)

Liam P Burke, `<lpburke@bu.edu>`

`from.Tag` *from.Tag*

Description

from.Tag

Usage

```
from.Tag(x)
```

Arguments

`x`	string to decode

`from.TreeCode` *from.TreeCode*

Description

from.TreeCode

Usage

```
from.TreeCode(x)
```

Arguments

`x`	string to decode

`get.attributes` *Retrieve attribute information from a vector or raster layer*

Description

Function to extract attribute information from vector or raster data layer.

Usage

```
get.attributes(file, coords)
```

Arguments

`file`	vector or raster layer
`coords`	vector containin xmin,ymin,xmax,ymax defing the bounding box for subset

Author(s)

Shawn P. Serbin

Examples

```
## Not run:
file <- Sys.glob(file.path(R.home(), 'library', 'PEcAn.data.land','data','*.kml'))
out <- get.attributes(file=file,coords=c(-95,42,-84,47))
print(out)

## End(Not run)
```

`get.soil` *get.soil*

Description

Get Soil

Usage

```
get.soil(lat, lon, soil.nc = soil.nc)
```

Arguments

`lat`	latitude
`lon`	longitude
`soil.nc`	netCDFe file with soil data

Value

usda soil class

Author(s)

David LeBauer

`get_resource_map` *get_resource_map*

Description

Locates data in DataONE and returns the resource_map or a message indicating that there is no corresponding resource_map for the given id

Usage

```
get_resource_map(id, CNode = "PROD")
```

Arguments

`id`	the doi or other identifier linked to the package in DataONE
`CNode`	default is "PROD"

Value

return the resource_map or a message indicating that there is no corresponding resource_map for the given id

get_veg_module *Load/extract + match species module*

Description

Load/extract + match species module

Usage

```
get_veg_module(
  input_veg,
  outfolder,
  start_date,
  end_date,
  dbparms,
  new_site,
  host,
  machine_host,
  overwrite
)
```

Arguments

`input_veg`	list, this is a sublist of settings$run$inputs that has info about source, id, metadata of the requested IC file
`outfolder`	path to where the processed files will be written
`start_date`	date in "YYYY-MM-DD" format, in case of source==FIA it's the settings$run$start.date, otherwise start_date of the IC file in DB
`end_date`	date in "YYYY-MM-DD" format, in case of source==FIA it's the settings$run$end.date, otherwise end_date of the IC file in DB
`dbparms`	list, settings$database info reqired for opening a connection to DB
`new_site`	data frame, id/lat/lon/name info about the site
`host`	list, host info as in settings$host, host$name forced to be "localhost" upstream
`machine_host`	local machine hostname, e.g. "pecan2.bu.edu"
`overwrite`	logical flag for convert_input

Author(s)

Istem Fer

`gSSURGO.Query` *This function queries the gSSURGO database for a series of map unit keys*

Description

This function queries the gSSURGO database for a series of map unit keys

Usage

```
gSSURGO.Query(
  mukeys,
  fields = c("chorizon.sandtotal_r", "chorizon.silttotal_r", "chorizon.claytotal_r")
)
```

Arguments

`mukeys`	map unit key from gssurgo
`fields`	a character vector of the fields to be extracted. See details and the default argument to find out how to define fields.

Details

Full documention of available tables and their relationships can be found here `www.sdmdataaccess.nrcs.usda.gov/QueryHelp.aspx` There have been occasions where NRCS made some minor changes to the structure of the API which this code is where those changes need to be implemneted here. Fields need to be defined with their associate tables. For example, sandtotal is a field in chorizon table which needs to be defined as chorizon.sandotal_(r/l/h), where r stands for the representative value, l stands for low and h stands for high. At the moment fields from mapunit, component, muaggatt, and chorizon tables can be extracted.

Value

a dataframe with soil properties. Units can be looked up from database documentation

Examples

```
## Not run:
 PEcAn.data.land::gSSURGO.Query(
   mukeys = 2747727,
   fields = c(
     "chorizon.cec7_r", "chorizon.sandtotal_r",
     "chorizon.silttotal_r","chorizon.claytotal_r",
     "chorizon.om_r","chorizon.hzdept_r","chorizon.frag3to10_r",
     "chorizon.dbovendry_r","chorizon.ph1to1h2o_r",
     "chorizon.cokey","chorizon.chkey"))

## End(Not run)
```

`IC_ISCN_SOC` *Extract ISCN SOC initial conditions from existing ISCN database.*

Description

Extract ISCN SOC initial conditions from existing ISCN database.

Usage

```
IC_ISCN_SOC(site_info, ens = 100, ecoregion.path = NULL)
```

Arguments

`site_info`	Bety list of site info including site_id, lon, and lat.
`ens`	ensemble number.
`ecoregion.path`	path to the directory where you store the shape files of L1 and L2 ecoregion maps.

Value

A data frame containing sampled SOC, each row represent each site.

Author(s)

Dongchen Zhang

`ic_process` *ic_process*

Description

ic_process

Usage

```
ic_process(settings, input, dir, overwrite = FALSE)
```

Arguments

`settings`	pecan settings list
`input`	Taken from settings$run$inputs. This should include id, path, and source
`dir`	settings$database$dbfiles
`overwrite`	Default = FALSE. whether to force ic_process to proceed

Author(s)

Istem Fer, Hamze Dokoohaki

id_resolveable *id_resolveable*

Description

Uses dataone::query from dataoneR to query DataONE. Prints result if data exists

Usage

```
id_resolveable(id, return_result = TRUE, CNode = "PROD")
```

Arguments

`id`	the doi or other identifier linked to the package in DataONE
`return_result`	boolean that returns or suppresses result of query. defaults to TRUE.
`CNode`	CNode="PROD"

Value

returns message indicating wether or not the id resolves to data in the DataONE federation and information about said data.

InventoryGrowthFusion *InventoryGrowthFusion*

Description

this code fuses forest inventory data with tree growth data (tree ring or dendrometer band) for the same plots. Code is a rewrite of Clark et al 2007 Ecol Appl into JAGS

Usage

```
InventoryGrowthFusion(
  data,
  cov.data = NULL,
  time_data = NULL,
  n.iter = 5000,
  n.chunk = n.iter,
  n.burn = min(n.chunk, 2000),
  random = NULL,
  fixed = NULL,
  time_varying = NULL,
  burnin_plot = FALSE,
  save.jags = "IGF.txt",
  z0 = NULL,
  save.state = TRUE,
  restart = NULL
)
```

Arguments

`data`	list of data inputs
`cov.data`	covariate data
`time_data`	required if time_varying is provided
`n.iter`	total number of iterations across all chunks
`n.chunk`	number of MCMC steps to evaluate at a time. Will only return LAST. If restarting, second number in vector is chunk to start from
`n.burn`	number of steps to automatically discard as burn-in
`random`	whether or not to include random effects
`fixed`	formula for fixed effects
`time_varying`	formula for time-varying effects
`burnin_plot`	logical: display a plot of the burnin steps?
`save.jags`	logical: Save the generated JAGS script?
`z0`	initial conditions for state variable
`save.state`	whether or not to include inferred DBH in output (can be large). Enter numeric value to save.state periodically (in terms of n.chunk)
`restart`	final mcmc.list from previous execution. NULL for new run. TRUE to save final state for new run.

Value

an mcmc.list object

Note

Requires JAGS

`InventoryGrowthFusionDiagnostics`
InventoryGrowthFusionDiagnostics

Description

InventoryGrowthFusionDiagnostics

Usage

```
InventoryGrowthFusionDiagnostics(jags.out, combined = NULL)
```

Arguments

`jags.out`	output mcmc.list from InventoryGrowthFusion
`combined`	data output from matchInventoryRings

Author(s)

Michael Dietze

`iscn_soc` *Soil organic carbon (SOC) density based on eco-region level 2 code from the ISCN database.*

Description

Contains 200 ensemble SOC data from 43 level 2 eco-regions across North America. Variable include SOC densities in g/cm2.

Usage

```
iscn_soc
```

Format

## ‘iscn_soc‘ A data frame with 200 rows and 43 columns:

rows 1 to 200 ensemble members

columns 43 level 2 ecoregion codes across North America

Source

https://iscn.fluxdata.org/wp-content/uploads/sites/23/2019/05/ISCN_ALL_DATA_DATASET_1-1.xlsx

---

`load_veg` *load_veg*

Description

uses ‘PEcAn.benchmark::load_data()‘ to get veg data

Usage

```
load_veg(
  new_site,
  start_date,
  end_date,
  source_id,
  source,
  icmeta = NULL,
  format_name = NULL,
  machine_host,
  dbparms,
  outfolder,
  overwrite = FALSE,
  ...
)
```

Arguments

`new_site`	list passed to ‘load_data‘
`start_date`, `end_date`	date range to look up
`source_id`	input id to look up in DB
`source`	name of data source (used in file naming)
`icmeta`	metadata for initial conditions
`format_name`	file format to look for
`machine_host`	hostname of machine where the data lives
`dbparms`	parameters to use when opening connection to database
`outfolder`	path to write results
`overwrite`	Logical: replace existing files? NOTE: Currently ignored!
`...`	Additional arguments, currently ignored

Author(s)

Istem Fer

`matchInventoryRings` *matchInventoryRings*

Description

matchInventoryRings

Usage

```
matchInventoryRings(
  trees,
  rings,
  extractor = "TreeCode",
  nyears = 30,
  coredOnly = TRUE
)
```

Arguments

`trees, rings`	codes from which to extract IDs
`extractor`	function to call, specified without its initial ‘to.‘ e.g. "TreeCode" calls ‘to.TreeCode‘
`nyears`	number of years to extract
`coredOnly`	logical: Only include trees with data from 2000?

match_pft *match_pft*

Description

Matches BETYdb species IDs to model-specific PFTs

Usage

```
match_pft(
  bety_species_id,
  pfts,
  query = NULL,
  con = NULL,
  allow_missing = FALSE,
  model = NULL
)
```

Arguments

`bety_species_id`	vector of BETYdb species IDs
`pfts`	settings$pfts. List of pfts with database matching based on name
`query`	Default is NULL. query to BETY db.
`con`	database connection, if NULL use traits package
`allow_missing`	flag to indicate that settings file does not need to match exactly
`model`	Default is NULL. This is the BETY model ID for matching pfts to the correct model.

Value

table of BETYdb PFT IDs matched to species IDs

Author(s)

Mike Dietze, Istem Fer

`match_species_id` *Match BETY species ID.*

Description

Parses species codes in input data and matches them with the BETY species ID.

Usage

```
match_species_id(
  input_codes,
  format_name = "custom",
  bety = NULL,
  translation_table = NULL,
  ...
)
```

Arguments

`input_codes`	Character vector of species codes
`format_name`	Species code format name (see details)
`bety`	BETY connection object
`translation_table`	Data frame with custom translation table (see details).
`...`	additional arguments, currently ignored

Details

`format_name` can be one of the following:

`usda` USDA Plants database symbol (e.g. QURU, TSCA)

`fia` FIA species code

`latin_name` Scientific name, as "Genus species"; must match exactly and unambiguously to `scientificname` field in BETY

`custom` A data frame matching BETY IDs (column name `bety_species_id`) to input codes (column name `input_code`). This data frame must be passed via the `translation_table` argument.

Value

`data.frame` containing the following columns:

`input_code`	Character provided as input
`bety_species_id`	Big integer species ID, unique and specific to BETY
`genus`	Genus part of Latin name, from BETY
`species`	Species part of Latin name, from BETY

Author(s)

Alexey Shiklomanov <ashiklom@bu.edu>, Istem Fer

Examples

```
## Not run:
con <- PEcAn.DB::db.open(list(
  driver = "Postgres",
  dbname = 'bety',
  user = 'bety',
  password = 'bety',
  host = 'localhost')
)
input_codes <- c('ACRU', 'PIMA', 'TSCA')
format_name <- 'usda'
match_species_id(input_codes = input_codes,
                 format_name = format_name,
                 bety = con)

## End(Not run)
```

`mpot2smoist` *Convert a matric potential to a soil moisture*

Description

Convert a matric potential to a soil moisture

Usage

```
mpot2smoist(
  mpot,
  soil_water_potential_at_saturation,
  soil_hydraulic_b,
  volume_fraction_of_water_in_soil_at_saturation
)
```

Arguments

`mpot`	water potential (cm H2O)
`soil_water_potential_at_saturation`	water potential when soil is saturated (cm H2O)
`soil_hydraulic_b`	pore-size distribution parameter for Campbell (1974) water content model
`volume_fraction_of_water_in_soil_at_saturation`	VSWC when soil is saturated (numeric in range 0-1)

Value

volumetric soil water content

`netcdf.writer.BADM` *netcdf.writer.BADAM*

Description

netcdf.writer.BADAM

Usage

```
netcdf.writer.BADM(lat, long, siteid, outdir, ens)
```

Arguments

`lat`	numeric latitude
`long`	numeric longitude
`siteid`	site id as a string
`outdir`	output dir which you want to store the IC netcdf file
`ens`	ensemble members, passed on to ‘pool_ic_list2netcdf‘

Value

a dataframe with file, host, mimetype, formatname, startdate, enddate and dbfile.name columns

`parse.MatrixNames` *parse.MatrixNames*

Description

parse.MatrixNames

Usage

```
parse.MatrixNames(w, pre = "x", numeric = FALSE)
```

Arguments

`w`	mcmc object containing matrix outputs
`pre`	prefix (variable name) for the matrix variable to be extracted
`numeric`	boolean, whether to coerce class to numeric

Value

matrix

Author(s)

Michael Dietze

`partition_roots` *partition_roots*

Description

Given a vector of root size thresholds (lower bound of each) and a vector of corresponding root carbon values, partition_roots checks if the input can be partitioned along the .002 m threshold between fine and coarse roots and returns a list containing the summed values for fine and coarse. If there are fewer than two thresholds or none within .0005 m of .002 m, returns NULL. Meant to be used in conjunction with standard variable root_carbon_content with rtsize dimension, extracted from netcdf.

Usage

```
partition_roots(roots, rtsize)
```

Arguments

`roots`	vector of root carbon values in kg C m-2
`rtsize`	vector of lower bounds of root size class thresholds in m, length greater than one and equal to roots. Must contain threshold within .0005 m of .002 m

Value

list containing summed fine root and coarse root carbon (2 values)

Author(s)

Anne Thomas

`plot2AGB` *plot2AGB*

Description

convert composite ring & census data into AGB

Usage

```
plot2AGB(combined, out, outfolder, allom.stats, unit.conv = 0.02)
```

Arguments

`combined`	data frame merging plot inventory and tree ring data
`out`	MCMC samples for diameter (sample x tree)
`outfolder`	output folder for graphs & data
`allom.stats`	Allometry statistics computed by ‘AllomAve‘
`unit.conv`	area conversion from sum(kg/tree) to kg/area

Author(s)

Mike Dietze `<dietze@bu.edu>`

`pool_ic_list2netcdf` *pool_ic_list2netcdf*

Description

Converts input list containing standard dimensions and variables (named values) for initial conditions to a netcdf file, input to pool-based models.

Usage

```
pool_ic_list2netcdf(input, outdir, siteid, ens = NA)
```

Arguments

`input`	list with two elements: list of netcdf dimensions (dims, with named values) and list of variables (vals, with named values)
`outdir`	directory to write netcdf file
`siteid`	site id
`ens`	Default is NA. Ensemble members.

Author(s)

Anne Thomas

`pool_ic_netcdf2list` *pool_ic_netcdf2list*

Description

Converts netcdf containing standard dimensions and variables for pool-based initial conditions, created by pool_ic_list2netcdf, back into list format

Usage

```
pool_ic_netcdf2list(nc.path)
```

Arguments

`nc.path`	path to netcdf file containing standard dimensions and variables

Value

list with two elements: list of netcdf dimensions (dims, with named values) and list of variables (vals, with named values)

Author(s)

Anne Thomas

prepare_pools *prepare_pools*

Description

Calculates pools from given initial condition values, deriving complements where necessary/possible if given TotLivBiomass

Usage

```
prepare_pools(nc.path, constants = NULL)
```

Arguments

`nc.path`	path to netcdf file containing standard dimensions and variables; currently supports these variables: TotLivBiom, leaf_carbon_content, LAI, AbvGrndWood, root_carbon_content, fine_root_carbon_content, coarse_root_carbon_content, litter_carbon_content, soil_organic_carbon_content, soil_carbon_content, wood_debris_carbon_content
`constants`	list of constants; must include SLA in m2 / kg C if providing LAI for leaf carbon

Value

list of pool values in kg C / m2 with generic names

Author(s)

Anne Thomas

put_veg_module *Match species to PFTs + veg2model module*

Description

Match species to PFTs + veg2model module

Usage

```
put_veg_module(
  getveg.id,
  dbparms,
  input_veg,
  pfts,
  outfolder,
  n.ensemble,
  dir,
  machine,
  model,
  start_date,
  end_date,
  new_site,
  host,
  overwrite
)
```

Arguments

`getveg.id`	list, input.id and dbfile.id of the IC file in intermediate pecan standard
`dbparms`	list, settings$database info reqired for opening a connection to DB
`input_veg`	list, this is a sublist of settings$run$inputs that has info about source, id, metadata of the requested IC file
`pfts`	list, same as settings$pfts
`outfolder`	path to where the processed files will be written
`n.ensemble`	integer, ensemble member number
`dir`	dir path to dbfiles on local machine
`machine`	data frame, DB info regarding localhost machine id/hostname etc.
`model`	model name, e.g. "ED2"
`start_date`	date in "YYYY-MM-DD" format, in case of source==FIA it's the settings$run$start.date, otherwise start_date of the IC file in DB
`end_date`	date in "YYYY-MM-DD" format, in case of source==FIA it's the settings$run$end.date, otherwise end_date of the IC file in DB
`new_site`	data frame, id/lat/lon/name info about the site
`host`	list, host info as in settings$host, host$name forced to be "localhost" upstream
`overwrite`	logical flag for convert_input

Author(s)

Istem Fer

`Read.IC.info.BADM` *Read.IC.info.BADM*

Description

This function returns a dataframe of plant biomass, root and soil carbon for a set of lat and long coordinates. This function first finds the level1 and level2 ecoregions for the given coordinates, and then tries to filter BADM database for those eco-regions. If no data found in the BADM database for the given lat/longs eco-regions, then all the data in the database will be used to return the initial condition. All the variables are also converted to kg/m^2.

Usage

```
Read.IC.info.BADM(lat, long)
```

Arguments

`lat`	numeric latitude
`long`	numeric longitude

Value

a dataframe with 7 columns of Site, Variable, Date, Organ, AGB, soil_organic_carbon_content, litter_carbon_content. Variable in the return object refers to what this value was called inside BADM database.

Examples

```
## Not run:
  badm_test <- Read.IC.info.BADM(45.805925,-90.07961)

## End(Not run)
```

`Read_Tucson` *Read_Tucson*

Description

wrapper around read.tucson that loads a whole directory of tree ring files and calls a 'clean' function that removes redundant records (WinDendro can sometimes create duplicate records when editing)

Usage

```
Read_Tucson(folder)
```

Arguments

`folder`	path to read files from. Will read all files at this path matching "TXT", "rwl", or "rw"

sample_ic *sample_ic*

Description

sample_ic

Usage

```
sample_ic(
  in.path,
  in.name,
  start_date,
  end_date,
  outfolder,
  n.ensemble,
  machine_host,
  source,
  bin_var = "DBH",
  bin_size = 10,
  bin_herb_soil = TRUE,
  ...
)
```

Arguments

`in.path`	path to folder of the file to be sampled
`in.name`	file name of the file to be sampled
`start_date`	date in "YYYY-MM-DD" format
`end_date`	date in "YYYY-MM-DD" format
`outfolder`	dir path, whete to write the file
`n.ensemble`	integer, ensemble member number
`machine_host`	localhost name, e.g. "pecan2.bu.edu"
`source`	string to appear in file names, e.g. "PalEON"
`bin_var`	variable you would like to sample by, DEFAULT is DBH
`bin_size`	bin size for sampling, DEFAULT is 10
`bin_herb_soil`	if we want to use bin size for both herb and soil sampling
`...`	Other inputs

Author(s)

Istem Fer

`sclass` *This function determines the soil class number based on the fraction of sand, clay, and silt*

Description

This function determines the soil class number based on the fraction of sand, clay, and silt

Usage

```
sclass(sandfrac, clayfrac)
```

Arguments

`sandfrac, clayfrac`	numeric vectors with values in range 0 to 1. Silt fraction is assumed to be the difference between (sand+clay) and 1

Value

vector of integers identifying textural class of each input layer. Possible values are 1 through 17; NB these are NOT the same class boundaries as the 12 USDA soil texture classes.

Examples

```
sclass(0.3,0.3)
```

`shp2kml` *Convert shapefile to KML*

Description

Convert ESRI shapefile (*.shp) to keyhole markup language (KML) file format

Usage

```
shp2kml(
  dir,
  ext,
  kmz = FALSE,
  proj4 = NULL,
  color = NULL,
  NameField = NULL,
  out.dir = NULL
)
```

Arguments

`dir`	Directory of GIS shapefiles to convert to kml/kmz
`ext`	File extension for files to convert to kml/kmz. Defaults to ESRI shapefile, '.shp'. [Place holder for other potential vector files to conver to kml]
`kmz`	TRUE/FALSE. Option to write out file as a compressed kml. Requires zip utility
`proj4`	OPTIONAL. Define output proj4 projection string. If set, input vector will be reprojected to desired projection. Not yet implemented.
`color`	OPTIONAL. Fill color for output kml/kmz file
`NameField`	OPTIONAL. Define names for individual features in KML/KMZ file
`out.dir`	OPTIONAL. Output directory for converted files

Author(s)

Shawn P. Serbin

Examples

```
## Not run:
dir <- Sys.glob(file.path(R.home(), 'library', 'PEcAn.data.land','data'))
out.dir <- path.expand('~/temp')
shp2kml(dir,'.shp',kmz=FALSE,NameField='STATE',out.dir=out.dir)
system(paste('rm -r ',out.dir))

## End(Not run)
```

`soil.units` *Get standard units for a soil variable*

Description

Given SSURGO names for soil properties, looks up their standard units. Note that names must match exactly.

Usage

```
soil.units(varname = NA)
```

Arguments

`varname`	character vector. See details

Details

Supported variables are:

- `soil_depth`
- `soil_cec`
- `fraction_of_clay_in_soil`
- `fraction_of_sand_in_soil`
- `fraction_of_silt_in_soil`
- `fraction_of_gravel_in_soil`
- `volume_fraction_of_water_in_soil_at_saturation`
- `volume_fraction_of_water_in_soil_at_field_capacity`
- `volume_fraction_of_condensed_water_in_dry_soil`
- `volume_fraction_of_condensed_water_in_soil_at_wilting_point`
- `soilC`
- `soil_ph`
- `soil_bulk_density`
- `soil_type`
- `soil_hydraulic_b`
- `soil_water_potential_at_saturation`
- `soil_hydraulic_conductivity_at_saturation`
- `thcond0`
- `thcond1`
- `thcond2`
- `thcond3`
- `soil_thermal_conductivity`
- `soil_thermal_conductivity_at_saturation`
- `soil_thermal_capacity`
- `soil_albedo`

Value

character matrix with columns `var` and `unit`

Examples

```
soil.units("soil_albedo")
```

soil2netcdf *Save soil texture & parameters in PEcAn standard netCDF CF*

Description

A table of standard names and units can be displayed by running soil.units() without any arguements

Usage

```
soil2netcdf(soil.data, new.file)
```

Arguments

soil.data	List of soil variables in standard names & units. Minimum is soil_depth and two of [sand, silt, clay]. Bulk density encouraged.
new.file	filename (including path) for output

Details

soil_params is called internally to estimate additional soil physical parameters from sand/silt/clay & bulk density. Will not overwrite any provided values

Need to expand to alternatively take soil_type (texture class) as an input

On output, soil_type named class is converted to a number because netCDF is a pain for storing strings. Conversion back can be done by load(system.file ("data/soil_class.RData",package = "PEcAn.data.land")) and then soil.name[soil_n]

Value

none

Examples

```
## Not run:  soil.data <- list(fraction_of_sand_in_soil = c
 (0.3,0.4,0.5), fraction_of_clay_in_soil = c(0.3,0.3,0.3), soil_depth = c
 (0.2,0.5,1.0))

soil2netcdf(soil.data,"soil.nc")
## End(Not run)
```

soilgrids_soilC_extract

soilgrids_soilC_extract

Description

soilgrids_soilC_extract function A function to extract total soil organic carbon for a single or group of lat/long locationsbased on user-defined site location from SoilGrids250m version 2.0 : https://soilgrids.org

Usage

```
soilgrids_soilC_extract(site_info, outdir = NULL, verbose = TRUE)
```

Arguments

`site_info`	A dataframe of site info containing the BETYdb site ID, site name, latitude, and longitude, e.g. (site_id, site_name, lat, lon)
`outdir`	Optional. Provide the results as a CSV file (soilgrids_soilC_data.csv)
`verbose`	Provide progress feedback to the terminal? TRUE/FALSE

Value

a dataframe containing the total soil carbon values and the corresponding standard deviation values (uncertainties) for each location Output column names are c("Site_ID","Site_Name","Latitude","Longitude", "Total_soilC","Std_soilC")

Author(s)

Qianyu Li, Shawn P. Serbin

Examples

```
## Not run:

# Example 1 - using the modex.bnl.gov BETYdb and site IDs to extract data
db <- 'betydb'
host_db <- 'modex.bnl.gov'
db_port <- '5432'
db_user <- 'bety'
db_password <- 'bety'

bety <- list(user='bety', password='bety', host=host_db,
dbname='betydb', driver=RPostgres::Postgres(),write=FALSE)

con <- DBI::dbConnect(drv=bety$driver, dbname=bety$dbname, host=bety$host,
password=bety$password, user=bety$user)
```

```
suppressWarnings(site_qry <- glue::glue_sql("SELECT *, ST_X(ST_CENTROID(geometry)) AS lon,
ST_Y(ST_CENTROID(geometry)) AS lat FROM sites WHERE id IN ({ids*})",
ids = c("676","622","678","766","764"), .con = con))

suppressWarnings(qry_results.1 <- DBI::dbSendQuery(con,site_qry))
suppressWarnings(qry_results.2 <- DBI::dbFetch(qry_results.1))
DBI::dbClearResult(qry_results.1)
DBI::dbDisconnect(con)

site_info <- qry_results.2
verbose <- TRUE
system.time(result_soc <- PEcAn.data.land::soilgrids_soilC_extract(site_info=site_info,
verbose=verbose))
result_soc


## End(Not run)
```

Soilgrids_SoilC_prep *Prepare Soilgrids SoilC data for the SDA workflow.*

Description

Prepare Soilgrids SoilC data for the SDA workflow.

Usage

```
Soilgrids_SoilC_prep(
  site_info,
  start_date,
  end_date,
  time_points,
  outdir = NULL,
  export_csv = FALSE
)
```

Arguments

site_info	Bety list of site info including site_id, lon, and lat.
start_date	Start date of SDA workflow.
end_date	End date of SDA workflow.
time_points	A vector contains each time point within the start and end date.
outdir	Where the final CSV file will be stored.
export_csv	Decide if we want to export the CSV file.

Value

A data frame containing AGB median and sd for each site and each time step.

Author(s)

Dongchen Zhang

`soil_class` *Default parameters for calculating soil properties from sand & clay content*

Description

Default parameters for calculating soil properties from sand & clay content

Usage

```
soil_class
```

Format

## 'soil_class' A list with 26 entries:

air.cond, h2o.cond, sand.cond, silt.cond, clay.cond thermal conductivity, W m^-1 K^-1

air.hcap, sand.hcap, silt.hcap, clay.hcap heat capacity, J m^-3 K^-1

kair, ksand, ksilt, kclay relative conductivity factor

fieldcp.K hydraulic conductance at field capacity, mm day^-1

grav gravity acceleration, m s^-2

soil.key Abbreviations for each of 18 soil texture classes, e.g. "SiL", "LSa"

soil.name Names for 18 soil texture classes, e.g. "Sand", "Silty clay"

soilcp.MPa soil water potential when air-dry, MPa

soilld.MPa soil water potential at critical water content, MPa

soilwp.MPa soil water potential at wilting point, MPa

stext.lines list of 18 lists, each giving minimum and maximum sand/silt/clay contents for a soil texture class

stext.polygon list of 18 lists, each giving corner points in the soil texture triangle for a soil texture class

texture data frame with 13 rows and 21 columns, giving default parameter values for 13 named soil textures

theta.crit critical water content (fractional soil moisture at which plants start dropping leaves), m^3 m^-3

xclay.def default volume fraction of sand in each of 18 soil texture classes

xsand.def default volume fraction of clay in each of 18 soil texture classes

Source

The hydraulic parameters are derived from Cosby et al 1984, "A Statistical Exploration of the Relationships of Soil Moisture Characteristics to the Physical Properties of Soils", Water Resources Research 20(6): 682-690. This implementation comes from one provided by the ED2 model, plus ‘texture.csv‘ from a source not recorded. Package ‘PEcAn.linkages‘ contains an identical texture.csv, also with no obvious source label. See also comments in soil_utils.R

`soil_params` *Estimate soil parameters from texture class or sand/silt/clay*

Description

Estimate soil parameters from texture class or sand/silt/clay

Usage

```
soil_params(
  soil_type = NULL,
  sand = NULL,
  silt = NULL,
  clay = NULL,
  bulk = NULL
)
```

Arguments

`soil_type`	USDA Soil Class. See Details
`sand`	percent sand
`silt`	percent silt
`clay`	percent clay
`bulk`	soil bulk density (optional, kg m-3)

Details

* Specify _either_ soil_type or sand/silt/clay. soil_type will be ignored if sand/silt/clay is provided * If only 2 out of sand/silt/clay are provided, it will be assumed they sum to 100 * Valid soil class options: "Sand","Loamy sand","Sandy loam","Silt loam","Loam", "Sandy clay loam","Silty clay loam","Clayey loam", "Sandy clay","Silty clay","Clay","Peat","Bedrock", "Silt","Heavy clay","Clayey sand","Clayey silt" * Based on ED2/R-utils/soilutils.r * Hydraulics based on Cosby et al 1984, using table 4 and equation 1 (which is incorrect it should be saturated moisture potential over moisture potential)

Value

list of soil hydraulic and thermal parameters

Examples

```
sand <- c(0.3, 0.4, 0.5)
clay <- c(0.3, 0.3, 0.3)
soil_params(sand=sand,clay=clay)
```

`soil_process` *Module for managing soil texture extraction*

Description

Module for managing soil texture extraction

Usage

```
soil_process(settings, input, dbfiles, overwrite = FALSE, run.local = TRUE)
```

Arguments

`settings`	PEcAn settings list
`input`	PEcAn input list
`dbfiles`	directory to write database files
`overwrite`	overwrite previous results (boolean)
`run.local`	logical: Run only on the current machine? If FALSE, runs on ‘settings$host‘ (which might turn out to be the current machine)

Value

path to soil file

`subset_layer` *Function to subset and clip a GIS vector or raster layer by a bounding box or clip/subset layer (e.g. shapefile/KML)*

Description

Function to subset and clip a GIS vector or raster layer by a bounding box or clip/subset layer (e.g. shapefile/KML)

Usage

```
subset_layer(
  file,
  coords = NULL,
  sub.layer = NULL,
  clip = FALSE,
  out.dir = NULL,
  out.name = NULL
)
```

Arguments

`file`	input file to be subset
`coords`	vector with xmin,ymin,xmax,ymax defing the bounding box for subset
`sub.layer`	Vector layer defining the subset region
`clip`	clip geometries to bounding box/subset layer? TRUE/FALSE
`out.dir`	output directory for subset layer. Defaults to location of input file. Can also set to 'pwd'
`out.name`	filename for subset layer. Defaults to original filename with the suffix *.sub

Author(s)

Shawn P. Serbin

Examples

```
## Not run:
# Test dataset
file <- Sys.glob(file.path(R.home(), 'library', 'PEcAn.data.land','data','*.shp'))
out.dir <- path.expand('~/temp')
# with clipping enabled
subset_layer(file=file,coords=c(-95,42,-84,47),clip=TRUE,out.dir=out.dir)
# without clipping enables
subset_layer(file=file,coords=c(-95,42,-84,47),out.dir=out.dir)
system(paste('rm -r',out.dir,sep=''))

## End(Not run)
```

to.Tag *to.Tag*

Description

to.Tag

Usage

```
to.Tag(SITE, PLOT, SUBPLOT, TAG = NULL)
```

Arguments

`SITE, PLOT, SUBPLOT`	ignored
`TAG`	string (or coercible to)

`to.TreeCode` *to.TreeCode*

Description

to.TreeCode

Usage

```
to.TreeCode(SITE, PLOT, SUBPLOT, TAG = NULL)
```

Arguments

`SITE, PLOT, SUBPLOT, TAG`	strings (or coercible to)

`write_ic` *write_ic*

Description

write_ic

Usage

```
write_ic(
  in.path,
  in.name,
  start_date,
  end_date,
  outfolder,
  model,
  new_site,
  pfts,
  source = input_veg$source,
  overwrite = FALSE,
  n.ensemble,
  host.inputargs,
  ...
)
```

Arguments

`in.path`	file path to rds file with IC data
`in.name`	file name of IC data
`start_date`	YYYY-MM-DD
`end_date`	YYYY-MM-DD
`outfolder`	Location to store function outputs
`model`	BETY model ID
`new_site`	Site info including lat, lon, and BETT site ID
`pfts`	list settings$pfts.
`source`	Data source as saved in the BETY db
`overwrite`	DEfault is FALSE. Option to overwrite existing files.
`n.ensemble`	number of ensemble members
`host.inputargs`	host info taken from settings object
`...`	Additional parameters

Author(s)

Istem Fer

`write_veg` *write_veg*

Description

Function to save intermediate rds file

Usage

```
write_veg(outfolder, start_date, veg_info, source)
```

Arguments

`outfolder`	output folder
`start_date`	start date
`veg_info`	vegetation data to be saved
`source`	name of data source (used in file naming)

Index