

# Package: PEcAn.data.remote (via r-universe)

November 4, 2024

**Type** Package

**Title** PEcAn Functions Used for Extracting Remote Sensing Data

**Version** 1.8.0.9000

**Author** Mike Dietze, Bailey Morrison

**Maintainer** Bailey Morrison <b Morrison@bnl.gov>

**Description** PEcAn module for processing remote data. Python module requirements: requests, json, re, ast, pandas, sys. If any of these modules are missing, install using pip install <module name>.

**Imports** curl, DBI, furr, future, glue, ncdf4, PEcAn.DB, PEcAn.utils, purrr, XML, sp, MODISTools (>= 1.1.0), reticulate, PEcAn.logger, magrittr, PEcAn.remote, rlang, terra, doParallel, parallel, foreach

**Suggests** data.table, dplyr, getPass, GEDI4R, ggplot2, grDevices, httr, jsonlite, lubridate, raster, reshape, sf, testthat (>= 1.0.2), tibble, utils

**Remotes** github::VangiElia/GEDI4R

**License** BSD\_3\_clause + file LICENSE

**Copyright** Authors

**LazyLoad** yes

**LazyData** FALSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Repository** <https://pecanproject.r-universe.dev>

**RemoteUrl** <https://github.com/PecanProject/pecan>

**RemoteRef** HEAD

**RemoteSha** caad7b3f8386df43eaf44f9316459f66ffc69b0b

## Contents

call_MODIS . . . . .	2
construct_remotedata_filename . . . . .	4
download.LandTrendr.AGB . . . . .	5
download.NLCD . . . . .	6
download.thredds.AGB . . . . .	7
extract.LandTrendr.AGB . . . . .	8
extract_NLCD . . . . .	9
extract_phenology_MODIS . . . . .	10
GEDI_AGB_download . . . . .	11
GEDI_AGB_extract . . . . .	12
GEDI_AGB_plot . . . . .	13
GEDI_AGB_prep . . . . .	13
getnetrc . . . . .	15
grid2netcdf . . . . .	15
l4_download . . . . .	16
Landtrendr_AGB_prep . . . . .	18
MODIS_LAI_prep . . . . .	19
MODIS_LC_prep . . . . .	20
NASA_CMV_finder . . . . .	21
NASA_DAAC_download . . . . .	21
NASA_DAAC_URL . . . . .	23
Prep_AGB_IC_from_2010_global . . . . .	24
read_remote_registry . . . . .	25
regrid . . . . .	26
remotedata_db_check . . . . .	26
remotedata_db_insert . . . . .	28
remote_process . . . . .	29
set_stage . . . . .	30
SMAP_SMP_prep . . . . .	31
<b>Index</b>	<b>33</b>

---

call\_MODIS

*call\_MODIS*

---

### Description

Get MODIS data by date and location

### Usage

```
call_MODIS(
  var,
  product,
  band,
  site_info,
```

```

    product_dates,
    outdir = NULL,
    run_parallel = FALSE,
    ncores = NULL,
    package_method = "MODISTools",
    QC_filter = FALSE,
    progress = FALSE
)

```

### Arguments

var	the simple name of the modis dataset variable (e.g. lai)
product	string value for MODIS product number
band	string value for which measurement to extract
site_info	Bety list of site info for parsing MODIS data: list(site_id, site_name, lat, lon, time_zone)
product_dates	a character vector of the start and end date of the data in YYYYJJJ
outdir	where the output file will be stored. Default is NULL and in this case only values are returned. When path is provided values are returned and written to disk.
run_parallel	optional method to download data paralleize. Only works if more than 1 site is needed and there are >1 CPUs available.
ncores	number of cpus to use if run_parallel is set to TRUE. If you do not know the number of CPU's available, enter NULL.
package_method	string value to inform function of which package method to use to download modis data. Either "MODISTools" or "reticulate" (optional)
QC_filter	Converts QC values of band and keeps only data values that are excellent or good (as described by MODIS documentation), and removes all bad values. qc_band must be supplied for this parameter to work. Default is False. Only MODISTools option.
progress	TRUE reports the download progress bar of the dataset, FALSE omits the download progress bar. Default is TRUE. Only MODISTools option. Requires Python3 for reticulate method option. There are a number of required python libraries. sudo -H pip install numpy suds netCDF4 json depends on the MODISTools package version 1.1.0

### Author(s)

Bailey Morrison

### Examples

```

## Not run:
site_info <- list(
  site_id = 1,
  site_name = "test",
  lat = 44,

```

```

lon = 90,
time_zone = "UTC")
test_modistools <- call_MODIS(
  var = "lai",
  product = "MOD15A2H",
  band = "Lai_500m",
  site_info = site_info,
  product_dates = c("2001150", "2001365"),
  outdir = NULL,
  run_parallel = TRUE,
  ncores = NULL,
  package_method = "MODISTools",
  QC_filter = TRUE,
  progress = FALSE)

## End(Not run)

```

---

```

construct_remotedata_filename
      construct_remotedata_filename

```

---

## Description

construct remotedata module file names

## Usage

```

construct_remotedata_filename(
  source,
  collection,
  siteid,
  scale = NULL,
  projection = NULL,
  qc = NULL,
  algorithm = NULL,
  out_process_data = NULL
)

```

## Arguments

source	source
collection	collection or product requested from the source
siteid	shortform of siteid
scale	scale, NULL by default
projection	projection, NULL by default
qc	qc_parameter, NULL by default
algorithm	algorithm name to process data, NULL by default
out_process_data	variable name requested for the processed file, NULL by default

**Value**

remotedata\_file\_names

**Author(s)**

Ayush Prasad

**Examples**

```
## Not run:
remotedata_file_names <- construct_remotedata_filename(
  source="gee",
  collection="s2",
  siteid="0-721",
  scale=10.0
  projection=NULL
  qc=1.0,
  algorithm="snap",
  out_process_data="lai")

## End(Not run)
```

---

download.LandTrendr.AGB

*download.LandTrendr.AGB*

---

**Description**

download.LandTrendr.AGB

**Usage**

```
download.LandTrendr.AGB(
  outdir,
  target_dataset = "biomass",
  product_dates = NULL,
  product_version = "v1",
  con = NULL,
  run_parallel = TRUE,
  ncores = NULL,
  overwrite = FALSE
)
```

**Arguments**

outdir            Where to place output  
target\_dataset   Which LandTrendr dataset to download? Default = "biomass"  
product\_dates    What data product dates to download

product_version	Optional. LandTrend AGB is provided with two versions, v0 and v1 (latest version)
con	Optional database connection. If specified then the code will check to see
run_parallel	Logical. Download and extract files in parallel?
ncores	Optional. If run_parallel=TRUE how many cores to use? If left as NULL will select max number -1
overwrite	Logical. Overwrite existing files and replace with new versions

**Value**

data.frame summarize the results of the function call

**Author(s)**

Shawn Serbin

**Examples**

```
## Not run:
outdir <- "~/scratch/abg_data/"
product_dates <- c(1990, 1991, 1995) # using discontinuous, or specific years
product_dates2 <- seq(1992, 1995, 1) # using a date sequence for selection of years
product_version = "v1"

results <- PEcAn.data.remote::download.LandTrendr.AGB(outdir=outdir,
  product_dates = product_dates,
  product_version = product_version)

results <- PEcAn.data.remote::download.LandTrendr.AGB(outdir=outdir,
  product_dates = product_dates2,
  product_version = product_version)

## End(Not run)
```

---

download.NLCD

*download.NLCD*

---

**Description**

Downloads and unzips the National Land Cover Database <http://www.mrlc.gov/nlcd2011.php>. Will automatically insert into PEcAn database if database connection provided.

**Usage**

```
download.NLCD(outdir, year = 2011, con = NULL)
```

**Arguments**

outdir	Directory to download NLCD to
year	which NLCD year to download. Only 2001 and 2011 are currently supported.
con	Optional database connection. If specified then the code will check to see if the file already exists in PEcAn before downloading, and will also create a database entry for new downloads

**Author(s)**

Mike Dietze

---

download.thredds.AGB    *download.thredds.AGB*

---

**Description**

download.thredds.AGB

**Usage**

```
download.thredds.AGB(
  outdir = NULL,
  site_ids,
  run_parallel = FALSE,
  ncores = NULL
)
```

**Arguments**

outdir	Where to place output
site_ids	What locations to download data at?
run_parallel	Logical. Download and extract files in parallel?
ncores	Optional. If run_parallel=TRUE how many cores to use? If left as NULL will select max number -1

**Value**

data.frame summarize the results of the function call

**Author(s)**

Bailey Morrison

**Examples**

```
## Not run:
outdir <- "~/scratch/abg_data/"
results <- PEEAn.data.remote::download.thredds.AGB(outdir=outdir,
  site_ids = c(676, 678, 679, 755, 767, 1000000030, 1000000145, 1000025731),
  run_parallel = TRUE, ncores = 8)

## End(Not run)
```

---

```
extract.LandTrendr.AGB
```

```
extract.LandTrendr.AGB
```

---

**Description**

```
extract.LandTrendr.AGB
```

**Usage**

```
extract.LandTrendr.AGB(
  site_info,
  dataset = "median",
  buffer = NULL,
  fun = "mean",
  data_dir = NULL,
  product_dates = NULL,
  output_file = NULL,
  ...
)
```

**Arguments**

<code>site_info</code>	list of site info for parsing AGB data: list(site_id, site_name, lat, lon, time_zone)
<code>dataset</code>	Which LandTrendr dataset to parse, "median" or "stdv".Default: "median"
<code>buffer</code>	Optional. operate over desired buffer area (not yet implemented)
<code>fun</code>	Optional function to apply to buffer area. Default - mean
<code>data_dir</code>	directory where input data is located. Can be NUL if con is specified
<code>product_dates</code>	Process and extract data only from selected years. Default behavior (product_dates = NULL) is to extract data from all available years in BETYdb or data_dir
<code>output_file</code>	Path to save LandTrendr_AGB_output.RData file containing the output extraction list (see return)
<code>...</code>	Additional arguments, currently ignored



**Value**

list of two containing the median AGB values per pixel and the corresponding standard deviation values (uncertainties)

**Author(s)**

Shawn Serbin, Alexey Shiklomanov

**Examples**

```
## Not run:

# Example 1 - using BETYdb site IDs to extract data
# Database connection (optional)

con <- PEcAn.DB::db.open(
  list(user='bety', password='bety', host='localhost',
        dbname='bety', driver='PostgreSQL',write=TRUE))

site_ID <- c(2000000023,1000025731,676,1000005149) # BETYdb site IDs
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 = site_ID, .con = con))
suppressWarnings(qry_results <- DBI::dbSendQuery(con,site_qry))
suppressWarnings(qry_results <- DBI::dbFetch(qry_results))
site_info <- list(site_id=qry_results$id, site_name=qry_results$sitename, lat=qry_results$lat,
lon=qry_results$lon, time_zone=qry_results$time_zone)
data_dir <- "~/scratch/agb_data/"

results <- extract.LandTrendr.AGB(site_info, "median", buffer = NULL, fun = "mean",
data_dir, product_dates, output_file)

## End(Not run)
```

---

extract\_NLCD

*extract.NLCD*

---

**Description**

Based on codes from Christy Rollinson and from Max Joseph (<http://mbjoseph.github.io/2014/11/08/nlcd.html>)

**Usage**

```
extract_NLCD(buffer, coords, data_dir = NULL, con = NULL, year = 2011)
```

**Arguments**

buffer	search radius (meters)
coords	data frame containing elements 'long' and 'lat'. Currently just supports single point extraction.
data_dir	directory where input data is located. Can be NUL if con is specified
con	connection to PEcAn database. Can be NULL if data_dir is specified
year	which NLCD year to extract. If data_dir not provided, must be one of '2001' or '2011'

**Value**

dataframe of fractional cover of different cover classes

**Author(s)**

Mike Dietze

---

extract\_phenology\_MODIS

*Get MODIS phenology data by date and location*

---

**Description**

Get MODIS phenology data by date and location

**Usage**

```
extract_phenology_MODIS(
  site_info,
  start_date,
  end_date,
  outdir,
  run_parallel = TRUE,
  ncores = NULL
)
```

**Arguments**

site_info	A dataframe of site info containing the BETYdb site ID, site name, latitude, and longitude, e.g.
start_date	Start date to download data
end_date	End date to download data
outdir	Path to store the outputs
run_parallel	optional method to download data parallelly. Only works if more than 1 site is needed and there are >1 CPUs available.
ncores	number of cpus to use if run_parallel is set to TRUE. If you do not know the number of CPU's available, enter NULL.

**Value**

the path for output file The output file will be saved as a CSV file to the outdir. Output column names are "year", "site\_id", "lat", "lon", "leafofday", "leafoffday", "leafon\_qa", "leafoff\_qa"

**Author(s)**

Qianyu Li

---

GEDI_AGB_download	<i>Download GEDI AGB data for the GEDI AGB extract function.</i>
-------------------	--

---

**Description**

Download GEDI AGB data for the GEDI AGB extract function.

**Usage**

```
GEDI_AGB_download(
  start_date,
  end_date,
  outdir,
  extent,
  nfile.min = 0,
  nrow.min = 0,
  gradient = 0
)
```

**Arguments**

start_date	start date (date with YYYY-MM-DD format) for downloading GEDI AGB from remote database.
end_date	end date (date with YYYY-MM-DD format) for downloading GEDI AGB from remote database.
outdir	Directory where the final CSV file will be stored.
extent	the XY box (in degrees) for downloading GEDI AGB file.
nfile.min	the minimum required file number to be downloaded and extracted, default is 0.
nrow.min	the minimum required observation number to be extracted, default is 0.
gradient	the gradient for iteratively enlarge the extent if the nfile.min or nrow.min are not reached, default is 0. If nfile.min or nrow.min is 0 this will be skipped.

**Value**

A data frame containing AGB and sd for the target spatial and temporal extent.

**Author(s)**

Dongchen Zhang

---

GEDI_AGB_extract	<i>Extract L4A GEDI above ground biomass data for the GEDI AGB prep function.</i>
------------------	---

---

### Description

Extract L4A GEDI above ground biomass data for the GEDI AGB prep function.

### Usage

```
GEDI_AGB_extract(
  site_info,
  start_date,
  end_date,
  outdir,
  nfile.min = 0,
  nrow.min = 0,
  buffer = 0.01,
  gradient = 0
)
```

### Arguments

site_info	A list including site_id, longitude, and latitude.
start_date	target start date (date with YYYY-MM-DD format) for preparing GEDI AGB from remote or local database.
end_date	end date (date with YYYY-MM-DD format) for preparing GEDI AGB from remote or local database.
outdir	Directory where the final CSV file will be stored.
nfile.min	the minimum required file number to be downloaded and extracted, default is 0.
nrow.min	the minimum required observation number to be extracted, default is 0.
buffer	buffer distance (in degrees) for locate GEDI AGB searching box (default is 0.01 [~ 1 km]).
gradient	the gradient for iteratively enlarge the extent if the nfile.min or nrow.min are not reached, default is 0. If nfile.min or nrow.min is 0 this will be skipped.

### Value

A list of AGB data frames for each site.

### Author(s)

Dongchen Zhang

---

GEDI_AGB_plot	<i>Plot GEDI AGB observations around the target site.</i>
---------------	---

---

**Description**

Plot GEDI AGB observations around the target site.

**Usage**

```
GEDI_AGB_plot(outdir, site.id, start_date, end_date)
```

**Arguments**

outdir	Where the plot PNG file will be stored.
site.id	Unique ID for the target site.
start_date	start date (date with YYYY-MM-DD format) for filtering out the existing CSV file.
end_date	end date (date with YYYY-MM-DD format) for filtering out the existing CSV file.

**Author(s)**

Dongchen Zhang

---

GEDI_AGB_prep	<i>Prepare L4A GEDI above ground biomass (AGB) data for the state data assimilation (SDA) workflow. This function is built upon the modified 'l4_download' function within the 'GEDI4R' package in need for a better parallel computation.</i>
---------------	--

---

**Description**

Prepare L4A GEDI above ground biomass (AGB) data for the state data assimilation (SDA) workflow. This function is built upon the modified 'l4\_download' function within the 'GEDI4R' package in need for a better parallel computation.

**Usage**

```
GEDI_AGB_prep(
  site_info,
  time_points,
  outdir = file.path(getwd(), "GEDI_AGB"),
  buffer = 0.01,
  search_window = "3 month"
)
```

**Arguments**

site_info	A list including site_id, longitude, and latitude.
time_points	A vector of date contains target dates (in YYYY-MM-DD).
outdir	Directory where the final CSV file will be stored.
buffer	buffer distance (in degrees) for locate GEDI AGB searching box (default is 0.01 [~ 1 km]).
search_window	search window (any length of time. e.g., 3 month) for locate available GEDI AGB values.

**Details**

During the first use, users will be ask to enter their Earth Explore login Information for downloading the data. If you don't have already an account, register at <https://urs.earthdata.nasa.gov/users/new>. These information will be saved in outdir as a netrc file. This function uses the foreach package for downloading files in parallel, with the doParallel configuration. If a file with the same name is already presented in outdir it will be overwrite.

**Value**

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

**Author(s)**

Dongchen Zhang

**Examples**

```
## Not run:
settings <- PEEAn.settings::read.settings("pecan.xml")
site_info <- settings %>%
  purrr::map(~.x[['run']] ) %>%
  purrr::map('site')%>%
  purrr::map(function(site.list){
    #conversion from string to number
    site.list$lat <- as.numeric(site.list$lat)
    site.list$lon <- as.numeric(site.list$lon)
    list(site_id=site.list$id, lat=site.list$lat, lon=site.list$lon, site_name=site.list$name)
  })%>%
  dplyr::bind_rows() %>%
  as.list()
time_points <- seq(start.date, end.date, by = time.step)
buffer <- 0.01
outdir <- getwd()
GEDI_AGB <- GEDI_AGB_prep(site_info, time_points, outdir, buffer)

## End(Not run)
```

---

getnetrc	<i>Function for building netrc file with access credentials</i>
----------	---

---

**Description**

Function for building netrc file with access credentials

**Usage**

```
getnetrc(dl_dir)
```

**Arguments**

dl\_dir            Directory where the netrc file will be stored.

**Value**

file path of the netrc file.

---

grid2netcdf	<i>grid2netcdf</i>
-------------	--------------------

---

**Description**

Write gridded data to netcdf file

**Usage**

```
grid2netcdf(gdata, date = "9999-09-09", outfile = "out.nc")
```

**Arguments**

gdata            gridded data to write out  
date             currently ignored; date(s) from 'gdata' are used instead  
outfile          name for generated netCDF file.

**Value**

writes netCDF file

**Author(s)**

David LeBauer

---

l4\_download

---

*DOWNLOAD GEDI level 4A data from DAACL.ORNL*


---

### Description

Download all GEDI footprints from [the official repository](#) that intersect a study area, defined as an extent in lon/lat coordinates. The footprints are located within the global latitude band observed by the International Space Station (ISS), nominally 51.6 degrees N and S and reported for the period 2019-04-18 to 2020-09-02

### Usage

```
l4_download(
  ul_lat,
  lr_lat,
  ul_lon,
  lr_lon,
  ncore = parallel::detectCores() - 1,
  from = NULL,
  to = NULL,
  outdir = getwd(),
  just_path = F,
  subset = NULL
)
```

### Arguments

ul_lat	Numeric: upper left latitude.
lr_lat	Numeric: lower right latitude.
ul_lon	Numeric: upper left longitude.
lr_lon	Numeric: lower right longitude.
ncore	Numeric: numbers of core to be used if the maximum core available is less than the number of files to be download. Default to the number of cores available minus one.
from	Character: date from which the data search starts. In the form "yyyy-mm-dd".
to	Character: date on which the data search end. In the form "yyyy-mm-dd".
outdir	Character: path of the directory in which to save the downloaded files. Default to the working directory. If it doesn't exist it will be created. Ignored if just_path=TRUE.
just_path	Logical: if TRUE return a character vector of available files without downloading them. Default to FALSE.
subset	Numeric vector of indices for downloading a subset of files instead of all. If is not numeric it will be ignored silently.



## Details

During the first use, users will be asked to enter their Earth Explore login information for downloading the data. If you don't have an account already, register at <https://urs.earthdata.nasa.gov/users/new>. This information will be saved in `outdir` as a `netrc` file. This function uses the `foreach` package for downloading files in parallel, with the `doParallel` configuration. If a file with the same name is already present in `outdir` it will be overwritten.

## Value

List of file paths in `outdir`.

## Author(s)

Elia Vangi

## Examples

```
## Not run:
#retrieve Italy bound
bound <- sf::st_as_sf(raster::getData('GADM', country='ITA', level=1))
ex <- raster::extent(bound)
ul_lat <- ex[4]
lr_lat <- ex[3]
ul_lon <- ex[2]
lr_lon <- ex[1]
from <- "2020-07-01"
to <- "2020-07-02"
#get just files path available for the searched parameters
l4_download(ul_lat=ul_lat,
            lr_lat=lr_lat,
            ul_lon=ul_lon,
            lr_lon=lr_lon,
            from=from,
            to=to,
            just_path=T
)

#download the first 4 files

l4_download(ul_lat=ul_lat,
            lr_lat=lr_lat,
            ul_lon=ul_lon,
            lr_lon=lr_lon,
            from=from,
            to=to,
            just_path=F,
            outdir = tempdir(),
            subset=1:4)

## End(Not run)
```

---

Landtrendr\_AGB\_prep     *Prepare Landtrendr AGB data for the SDA workflow.*

---

### Description

Prepare Landtrendr AGB data for the SDA workflow.

### Usage

```
Landtrendr_AGB_prep(
  site_info,
  start_date,
  end_date,
  time_points,
  AGB_indir,
  outdir = NULL,
  export_csv = TRUE,
  allow_download = FALSE,
  buffer = NULL,
  skip_buffer = TRUE
)
```

### Arguments

site_info	Bety list of site info including site_id, site_name, 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.
AGB_indir	Where the Landtrendr AGB data can be accessed.
outdir	Where the final CSV file will be stored.
export_csv	Decide if we want to export the CSV file.
allow_download	If data is missing, should we download the missing data?
buffer	buffer area to calculate the min var of AGB data.
skip_buffer	flag to skip calculating min var based on buffer area.

### Value

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

### Author(s)

Dongchen Zhang

---

MODIS_LAI_prep	<i>Prepare MODIS LAI data for the SDA workflow.</i>
----------------	---

---

**Description**

Prepare MODIS LAI data for the SDA workflow.

**Usage**

```
MODIS_LAI_prep(  
  site_info,  
  time_points,  
  outdir = NULL,  
  search_window = 30,  
  export_csv = FALSE,  
  sd_threshold = 20  
)
```

**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.
outdir	Where the final CSV file will be stored.
search_window	search window for locate available LAI values.
export_csv	Decide if we want to export the CSV file.
sd_threshold	Threshold for filtering out any estimations with unrealistic high standard error, default is 20. The QC check will be skipped if it's set as NULL.

**Value**

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

**Author(s)**

Dongchen Zhang

---

`MODIS_LC_prep`*Prepare MODIS land cover data for the SDA workflow.*

---

### Description

Prepare MODIS land cover data for the SDA workflow.

### Usage

```
MODIS_LC_prep(  
  site_info,  
  time_points,  
  outdir = NULL,  
  qc.filter = c("000", "001")  
)
```

### Arguments

<code>site_info</code>	Bety list of site info including <code>site_id</code> , lon, and lat.
<code>time_points</code>	A vector contains each time point within the start and end date.
<code>outdir</code>	Where the final CSV file will be stored.
<code>qc.filter</code>	values that will pass the QC check. the check will be skipped if it's NULL.

### Details

This function enables the feature of grabbing pre-extracted MODIS LC CSV files such that any site that has records will be skipped (See Line 33). In more detail, we will be loading the previous 'LC.csv' file, which contains previous extracted land cover records and trying to match that with current requests (location, time). Any requests that fail the match will be regarded as new extractions and combine with the previous 'LC.csv' file.

### Value

A data frame containing MODIS land cover types for each site and each time step.

### Author(s)

Dongchen Zhang

---

NASA_CMV_finder	<i>Create URL that can be used to request data from NASA DAAC server.</i>
-----------------	---

---

**Description**

Create URL that can be used to request data from NASA DAAC server.

**Usage**

```
NASA_CMV_finder(doi)
```

**Arguments**

doi	Character: data DOI on the NASA DAAC server, it can be obtained directly from the NASA ORNL DAAC data portal (e.g., GEDI L4A through <a href="https://daac.ornl.gov/cgi-bin/dsvviewer.pl?ds_id=2056">https://daac.ornl.gov/cgi-bin/dsvviewer.pl?ds_id=2056</a> ).
-----	---

**Value**

A list with each containing corresponding provider and concept ids given the data doi.

**Author(s)**

Dongchen Zhang

**Examples**

```
## Not run:
provider_conceptID <- NASA_CMV_finder("10.3334/ORNLDAAC/2183")

## End(Not run)
```

---

NASA_DAAC_download	<i>Parallel download data from the NASA ORNL DAAC server given period, spatial bounding box, and data DOI.</i>
--------------------	--

---

**Description**

Parallel download data from the NASA ORNL DAAC server given period, spatial bounding box, and data DOI.

**Usage**

```

NASA_DAAC_download(
  ul_lat,
  ul_lon,
  lr_lat,
  lr_lon,
  ncore = 1,
  from,
  to,
  outdir = getwd(),
  doi,
  netrc_file = NULL,
  just_path = FALSE
)

```

**Arguments**

ul_lat	Numeric: upper left latitude.
ul_lon	Numeric: upper left longitude.
lr_lat	Numeric: lower right latitude.
lr_lon	Numeric: lower right longitude.
ncore	Numeric: numbers of core to be used if the maximum core
from	Character: date from which the data search starts. In the form "yyyy-mm-dd".
to	Character: date on which the data search end. In the form "yyyy-mm-dd".
outdir	Character: path of the directory in which to save the downloaded files. Default is the current work directory(getwd()).
doi	Character: data DOI on the NASA DAAC server, it can be obtained directly from the NASA ORNL DAAC data portal (e.g., GEDI L4A through <a href="https://daac.ornl.gov/cgi-bin/dsviewer.pl?ds_id=2056">https://daac.ornl.gov/cgi-bin/dsviewer.pl?ds_id=2056</a> ).
netrc_file	Character: path to the credential file, default is NULL.
just_path	Boolean: if we just want the metadata and URL or proceed the actual download.

**Value**

A list containing meta data and physical path for each data downloaded.

**Author(s)**

Dongchen Zhang

**Examples**

```

## Not run:
ul_lat <- 35
ul_lon <- -121
lr_lat <- 33

```

```

lr_lon <- -117
from <- "2022-02-23"
to <- "2022-05-30"
doi <- "10.3334/ORNLDAAC/2183"
outdir <- "/projectnb/dietzelab/dongchen/SHIFT/test_download"
metadata <- NASA_DAAC_download(ul_lat = ul_lat,
                               ul_lon = ul_lon,
                               lr_lat = lr_lat,
                               lr_lon = lr_lon,
                               from = from,
                               to = to,
                               doi = doi,
                               just_path = T)

## End(Not run)

```

---

NASA\_DAAC\_URL

*Create URL that can be used to request data from NASA DAAC server.*


---

### Description

Create URL that can be used to request data from NASA DAAC server.

### Usage

```

NASA_DAAC_URL(
  base_url = "https://cmr.earthdata.nasa.gov/search/granules.json?pretty=true",
  provider,
  page_size = 2000,
  page = 1,
  concept_id,
  bbox,
  daterange = NULL
)

```

### Arguments

base_url	Character: base URL for the CMR search. default is "https://cmr.earthdata.nasa.gov/search/granules.json?"
provider	Character: ID of data provider from NASA DAAC. See 'NASA_CMRFinder' for more details.
page_size	Numeric: maximum requested length, default is 2000.
page	Numeric: which page of the URL, default is 1.
concept_id	Character: CMR Concept ID. See 'NASA_CMRFinder' for more details.
bbox	Numeric: vector of bounding box coordinates.
daterange	Character: vectors of the requested start and end dates. In the form "yyyy-mm-dd".

**Value**

A character of URL that can be used to request data.

**Author(s)**

Dongchen Zhang

**Examples**

```
## Not run:
provider <- "ORNL_CLOUD"
concept_id <- "C2770099044-ORNL_CLOUD"
bbox <- "-121,33,-117,35"
daterange <- c("2022-02-23", "2022-05-30")
URL <- NASA_DAAC_URL(provider = provider,
concept_id = concept_id,
bbox = bbox,
daterange = daterange)

## End(Not run)
```

---

Prep\_AGB\_IC\_from\_2010\_global

*Extract ensemble above ground biomass density from pre-existing GeoTIFF files for the SDA workflow. Note that, this function only works for those products who have both mean and uncertainty GeoTIFF images prepared. And it works under the 2010 Global AGB products: DOI: <https://doi.org/10.3334/ORNLDAAAC/1763>.*

---

**Description**

Extract ensemble above ground biomass density from pre-existing GeoTIFF files for the SDA workflow. Note that, this function only works for those products who have both mean and uncertainty GeoTIFF images prepared. And it works under the 2010 Global AGB products: DOI: <https://doi.org/10.3334/ORNLDAAAC/1763>.

**Usage**

```
Prep_AGB_IC_from_2010_global(site_info, paths.list, ens)
```

**Arguments**

site_info	Bety list of site info including site_id, lon, and lat.
paths.list	list containing file paths for ‘mean’ and ‘uncertainty’ datasets.
ens	ensemble number.



**Value**

A data frame containing sampled above ground biomass densities, each column represent each site.

**Author(s)**

Dongchen Zhang

---

`read_remote_registry` *read\_remote\_registry*

---

**Description**

read remote module registration files

**Usage**

```
read_remote_registry(source, collection)
```

**Arguments**

source	remote source, e.g gee or appears
collection	collection or product name

**Value**

list containing original\_name, pecan\_name, scale, qc, projection raw\_mimetype, raw\_formatname pro\_mimetype, pro\_formatname, coordtype

**Author(s)**

Istem Fer

**Examples**

```
## Not run:  
read_remote_registry(  
  "gee",  
  "COPERNICUS/S2_SR")  
  
## End(Not run)
```

---

regrid	<i>regrid</i>
--------	---------------

---

**Description**

Regrid dataset to even grid

**Usage**

```
regrid(latlon.data)
```

**Arguments**

latlon.data      dataframe with lat, lon, and some value to be regridded

**Value**

dataframe with regridded data

**Author(s)**

David LeBauer

---

remotedata_db_check	<i>remotedata_db_check</i>
---------------------	----------------------------

---

**Description**

check the status of the requested data in the DB

**Usage**

```
remotedata_db_check(  
  raw_file_name,  
  pro_file_name,  
  start,  
  end,  
  siteid,  
  siteid_short,  
  out_get_data,  
  algorithm,  
  out_process_data,  
  overwrite,  
  dbcon  
)
```

**Arguments**

raw_file_name	raw_file_name
pro_file_name	pro_file_name
start	start date requested by user
end	end date requested by the user
siteid	siteid of the site
siteid_short	short form of the siteid
out_get_data	out_get_data
algorithm	algorithm
out_process_data	out_process_data
overwrite	overwrite
dbcon	BETYdb con

**Value**

list containing remotefile\_check\_flag, start, end, stage\_get\_data, write\_raw\_start, write\_raw\_end, raw\_merge, existing\_raw\_file\_path, stage\_process\_data, write\_pro\_start, write\_pro\_end, pro\_merge, input\_file, existing\_pro\_file\_path, raw\_check, pro\_check

**Author(s)**

Ayush Prasad

**Examples**

```
## Not run:
dbstatus <- remotedata_db_check(
  raw_file_name,
  pro_file_name,
  start,
  end,
  siteid,
  siteid_short,
  out_get_data,
  algorithm,
  out_process_data,
  overwrite
  dbcon)

## End(Not run)
```

---

remotedata\_db\_insert *Insert the output data returned from rp\_control into BETYdb*

---

### Description

Insert the output data returned from rp\_control into BETYdb

### Usage

```
remotedata_db_insert(
  output,
  remotefile_check_flag,
  siteid,
  out_get_data,
  out_process_data,
  write_raw_start,
  write_raw_end,
  write_pro_start,
  write_pro_end,
  raw_check,
  pro_check,
  raw_mimetype,
  raw_formatname,
  pro_mimetype,
  pro_formatname,
  dbcon
)
```

### Arguments

output	output list from rp_control
remotefile_check_flag	remotefile_check_flag
siteid	siteid
out_get_data	out_get_data
out_process_data	out_process_data
write_raw_start	write_raw_start, start date of the raw file
write_raw_end	write_raw_end, end date of the raw file
write_pro_start	write_pro_start
write_pro_end	write_pro_end
raw_check	id, site_id, name, start_date, end_date, of the existing raw file from inputs table and file_path from dbfiles tables

pro_check	pro_check id, site_id, name, start_date, end_date, of the existing processed file from inputs table and file_path from dbfiles tables
raw_mimetype	raw_mimetype
raw_formatname	raw_formatname
pro_mimetype	pro_mimetype
pro_formatname	pro_formatname
dbcon	BETYdb con

**Value**

list containing raw\_id, raw\_path, pro\_id, pro\_path

**Author(s)**

Ayush Prasad

**Examples**

```
## Not run:
db_out <- remotedata_db_insert(
  output,
  remotefile_check_flag,
  siteid,
  out_get_data,
  out_process_data,
  write_raw_start,
  write_raw_end,
  write_pro_start,
  write_pro_end,
  raw_check,
  pro_check,
  raw_mimetype,
  raw_formatname,
  pro_mimetype,
  pro_formatname,
  dbcon)

## End(Not run)
```

---

remote_process	<i>remote_process</i>
----------------	-----------------------

---

**Description**

call rp\_control (from RpTools Python package) and store the output in BETY

**Usage**

```
remote_process(settings)
```

**Arguments**

settings      PEcAn settings list containing remotedata tags: source, collection, scale, projection, qc, algorithm, credfile, out\_get\_data, out\_process\_data, overwrite

**Author(s)**

Ayush Prasad, Istem Fer

**Examples**

```
## Not run:
remote_process(settings)

## End(Not run)
```

---

set_stage	<i>set_stage</i>
-----------	------------------

---

**Description**

set dates, stage and merge status for remote data download

**Usage**

```
set_stage(result, req_start, req_end, stage)
```

**Arguments**

result      dataframe containing id, site\_id, name, start\_date, end\_date from inputs table and file\_path from dbfiles table

req\_start    start date requested by the user

req\_end      end date requested by the user

stage        the stage which needs to be set, get\_remote\_data or process\_remote\_data

**Value**

list containing req\_start, req\_end, stage, merge, write\_start, write\_end

**Author(s)**

Ayush Prasad

**Examples**

```
## Not run:
raw_check <- set_stage(
  result,
  req_start,
  req_end,
  get_remote_data)

## End(Not run)
```

---

SMAP\_SMP\_prep

---

*Prepare SMAP Soil Moisture (SMP) data for the SDA workflow.*


---

**Description**

Prepare SMAP Soil Moisture (SMP) data for the SDA workflow.

**Usage**

```
SMAP_SMP_prep(
  site_info,
  start_date,
  end_date,
  time_points,
  outdir,
  search_window = 30,
  export_csv = TRUE,
  update_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, and the CSV file from GEE are stored.
search_window	search window for locate available SMP values.
export_csv	Decide if we want to export the CSV file.
update_csv	Decide if we want to update current CSV file given an updated SMAP_gee.csv file

**Value**

A data frame containing SMAP smp and sd for each site and each time step.

**Author(s)**

Dongchen Zhang



# Index

[call\\_MODIS](#), [2](#)  
[construct\\_remotedata\\_filename](#), [4](#)

[download.LandTrendr.AGB](#), [5](#)  
[download.NLCD](#), [6](#)  
[download.thredds.AGB](#), [7](#)

[extract.LandTrendr.AGB](#), [8](#)  
[extract\\_NLCD](#), [9](#)  
[extract\\_phenology\\_MODIS](#), [10](#)

[GEDI\\_AGB\\_download](#), [11](#)  
[GEDI\\_AGB\\_extract](#), [12](#)  
[GEDI\\_AGB\\_plot](#), [13](#)  
[GEDI\\_AGB\\_prep](#), [13](#)  
[getnetrc](#), [15](#)  
[grid2netcdf](#), [15](#)

[l4\\_download](#), [16](#)  
[Landtrendr\\_AGB\\_prep](#), [18](#)

[MODIS\\_LAI\\_prep](#), [19](#)  
[MODIS\\_LC\\_prep](#), [20](#)

[NASA\\_CMR\\_finder](#), [21](#)  
[NASA\\_DAAC\\_download](#), [21](#)  
[NASA\\_DAAC\\_URL](#), [23](#)

[Prep\\_AGB\\_IC\\_from\\_2010\\_global](#), [24](#)

[read\\_remote\\_registry](#), [25](#)  
[regrid](#), [26](#)  
[remote\\_process](#), [29](#)  
[remotedata\\_db\\_check](#), [26](#)  
[remotedata\\_db\\_insert](#), [28](#)

[set\\_stage](#), [30](#)  
[SMAP\\_SMP\\_prep](#), [31](#)