

Package: luna (via r-universe)

October 14, 2026

Type Package

Title Tools for Satellite Remote Sensing (Earth Observation) Data Processing

Version 0.4-1

Date 2026-06-06

Encoding UTF-8

Language en-US

Depends R (>= 4.0.0), terra (>= 1.6.41)

LinkingTo Rcpp, RcppArmadillo

Imports methods, Rcpp (>= 1.0-10)

Suggests httr, rvest, xml2, jsonlite, meteor, signal

Maintainer Robert J. Hijmans <r.hijmans@gmail.com>

Description Tools for acquiring and (pre-) processing satellite remote sensing data. Including for downloading data from NASA such as LANDSAT and MODIS.

License GPL (>=3)

BugReports <https://github.com/rspatial/luna/issues/>

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev libproj-dev

Repository <https://rspatial.r-universe.dev>

Date/Publication 2026-06-29 22:32:26 UTC

RemoteUrl <https://github.com/rspatial/luna>

RemoteRef HEAD

RemoteSha 1762b80a1e5f867c43ac2c0a1eb0ddf801a0a0f0

Contents

luna-package	2
earthdataLogin	2
erosLogin	3

filterVI	5
getAVHRR	6
getCGLS	7
getLandsat	8
getModis	10
getNASA	11
getProducts	13
mesma	14
modis_mask	15
modisDate	16
modisExtent	17
panSharpen	17
productInfo	19
Index	21

luna-package	<i>Tools for Satellite Remote Sensing (Earth Observation) Data Processing</i>
--------------	---

Description

Tools for acquiring and (pre-) processing satellite remote sensing data. Including for downloading data from NASA such as LANDSAT and MODIS.

earthdataLogin	<i>Authenticate against NASA Earthdata Login (URS)</i>
----------------	--

Description

Validate a NASA Earthdata Login (URS) username/password against <https://urs.earthdata.nasa.gov/> and return a session object that can be passed to `getNASA` via its `auth` argument.

A successful call confirms up front that the credentials are correct, so subsequent `getNASA(download = TRUE, auth = ...)` calls will fail fast on auth errors rather than appearing to succeed when files happen to be cached on disk (issue #40).

Usage

```
earthdataLogin(username, password, verbose = TRUE)
```

Arguments

- username character. Earthdata Login user name. Sign up at <https://urs.earthdata.nasa.gov/users/new>.
- password character. Earthdata Login password.
- verbose logical. If TRUE (the default), print a status message before and after the credential probe.

Details

Validation is performed by issuing a HTTP GET request to <https://urs.earthdata.nasa.gov/api/users/tokens> with HTTP Basic authentication. The endpoint is read-only (it returns a list of the user's existing API tokens, if any) and reliably returns HTTP 200 for valid credentials and HTTP 401 for invalid ones.

The returned session object holds the paths to a temporary netrc file and cookie jar that the rest of luna uses to authenticate downloads. Both files are removed when the session is garbage-collected or when R exits.

In addition to a working Earthdata Login account, downloading a particular product requires that you have accepted that product's End User License Agreement (EULA) at <https://urs.earthdata.nasa.gov/>. Authentication can succeed (so `earthdataLogin()` returns a session) while subsequent downloads of an EULA-restricted product still fail with HTTP 401/403; in that case the failure is reported by `getNASA()`.

Value

An `earthdata_session` object (an environment) with elements `username`, `netrc_file`, `cookie_file`, `validated`, and `created`.

See Also

[getNASA](#), [erosLogin](#)

Examples

```
## Not run:
auth <- earthdataLogin("my_user", "my_password")
auth  # prints a one-line summary

# reuse the same session for many downloads, without re-authenticating
f <- getNASA(product = "MOD09A1",
             start_date = "2019-05-17", end_date = "2019-05-23",
             aoi = c(33, 34, -5, -4),
             download = TRUE, path = tempdir(),
             auth = auth)

## End(Not run)
```

erosLogin

Authenticate against the USGS EROS login system

Description

Validate a USGS Earthdata Login username/password against <https://ers.cr.usgs.gov/> and return a session object that can be passed to [getLandsat](#) via its `auth` argument.

Although USGS uses the same "Earthdata Login" credentials as NASA, the actual authentication flow for Landsat downloads is different: USGS requires an interactive form-login on `ers.cr.usgs.gov`

that establishes a session cookie. `erosLogin()` performs that flow once and stores the resulting curl handle in the session object, so subsequent `getLandsat(download = TRUE, auth = ...)` calls reuse the same authenticated session.

Usage

```
erosLogin(username, password, verbose = TRUE)
```

Arguments

username	character. USGS Earthdata Login user name. Sign up at https://ers.cr.usgs.gov/register .
password	character. USGS Earthdata Login password.
verbose	logical. If TRUE (the default), print a status message before and after the form-login probe.

Details

Authentication is performed by GETting the EROS login page, scraping the CSRF token from its HTML form, and POSTing a form-encoded request with the supplied credentials. Success is confirmed by checking that the resulting HTML contains a "Sign Out" link.

The session holds a libcurl handle whose cookies authenticate subsequent download requests against `landsatlook.usgs.gov`. The session attempts a logout when the object is garbage-collected.

If the EROS UI changes such that the login flow stops working, `erosLogin()` fails with an informative error. In that case consider using the **rstac** package or downloading manually from <https://earthexplorer.usgs.gov/>.

Value

An `eros_session` object (an environment) with elements `username`, `handle`, `validated`, and `created`.

See Also

[getLandsat](#), [earthdataLogin](#)

Examples

```
## Not run:
auth <- erosLogin("my_user", "my_password")
auth  # prints a one-line summary

# reuse the session across multiple Landsat scenes
f <- getLandsat(product = "landsat-c211",
  start_date = "2019-05-01", end_date = "2019-05-31",
  aoi = c(33, 34, -5, -4),
  bands = c("blue", "green", "red", "nir08"),
  download = TRUE, path = tempdir(),
  auth = auth)
```

```
## End(Not run)
```

filterVI

Gapfill and filter time series satellite data

Description

Estimate missing values using smooth gap-filled time series data with a two-stage Savitzky-Golay smoothing filter.

Usage

```
filterVI(x, ...)  
  
fillVI(x, method = "natural")
```

Arguments

x	numeric or SpatRaster representing time series signal to be filtered
method	type of spline to be used. Possible values are "fmm", "natural", "periodic", "monoH.FC" and "hyman"
...	additional arguments: method: as above p1: filter order for first pass n1: filter length for first pass (must be odd) p2: filter order for inner passes n2: filter length for inner passes (must be odd)

Details

Performs smoothing of time series data using a two pass Savitzky-Golay smoothing filter. Second pass is an iterative method to minimize the difference between actual and fitted values that are significantly lower than the actual values. Missing values designated as NA are automatically interpolated with spline. For more details on the arguments see [splinefun](#) and [sgolayfilt](#).

Value

Filtered signal of length(x)

Examples

```
# apply on single pixel  
#vi <- ....  
#s <- filterVI(vi)  
  
# for time series raster  
#fevi <- app(evi, filterVI)
```

getAVHRR

Find and download AVHRR NDVI Climate Data Record (CDR)

Description

Find and download AVHRR NDVI CDR global layers for a time period. The function returned is list of filenames that may have been downloaded.

Usage

```
getAVHRR(start_date, end_date, path, download=FALSE, overwrite=FALSE, update=FALSE, quiet=FALSE, ...)
```

Arguments

start_date	character. Start date for the data requested formatted yyyy-m-d
end_date	Character. end date for the data requested formatted yyyy-m-d
path	character. Path to download data to
download	logical. If TRUE, the files are downloaded. If FALSE (the default), only the file names that match the date range are returned
overwrite	logical. If TRUE, existing files are overwritten
update	logical. If TRUE, the cached index of available AVHRR NDVI CDR files is refreshed
quiet	logical. If TRUE, messages and the download progress bar are suppressed
...	Additional arguments passed to download.file (for example method)

Value

Character vector of file names pointing to the downloaded files

Examples

```
## Not run:
start <- "2019-05-17"
end <- "2019-05-23"
path <- tempdir()
ff <- getAVHRR(start, end, path, download=FALSE)

## End(Not run)
```

`getCGLS`*Download CGLS products*

Description

Download data provided by the Copernicus Global Land Service (<https://land.copernicus.eu/global/>)
You need to register at <https://land.copernicus.eu/global/> and create a username and password.

See <https://land.copernicus.eu/global/products/> for a product overview and product details and see <https://land.copernicus.vgt.vito.be/manifest/> for an overview for data availability.

Usage

```
getCGLS(product, start_date, end_date, resolution="1km",  
        version=1, path, username, password, ...)
```

Arguments

<code>product</code>	character. For example: "fapar", "fcover", "lai", "ndvi", "ssm", "swi", "lst"
<code>resolution</code>	character. one of "1km", "300m" or "100m"
<code>version</code>	positive integer. product version
<code>start_date</code>	character or Date. Start date for the data requested formatted yyyy-m-d
<code>end_date</code>	Character or Date. End date for the data requested formatted yyyy-m-d
<code>path</code>	character. Path name indicating where to store the data
<code>username</code>	character. EOSDIS user name (see Details)
<code>password</code>	character. EOSDIS password (see Details)
<code>...</code>	Additional arguments passed to <code>download.file</code>

Value

SpatRaster

See Also

[getNASA](#)

getLandsat

*Find and download Landsat products***Description**

Find and (optionally) download Landsat scenes for a specific product, area, and time period.

The function searches the USGS Landsat STAC catalog at <https://landsatlook.usgs.gov/stac-server/>. It is the successor to the old, NASA CMR-based implementation of `getLandsat()`: NASA CMR no longer indexes USGS Landsat granules, so the previous default product "Landsat_8_OLI_TIRS_C1" (Landsat 8 Collection 1) and the AWS public S3 bucket `landsat-pds` are both retired.

When `download = FALSE` (the default), `getLandsat()` returns a character vector of asset URLs that match the search and the optional bands filter. These can be inspected, filtered, or downloaded with any tool of your choice.

When `download = TRUE`, the function logs in to EROS using the supplied authentication session (or, deprecated, raw username/password), downloads each asset, and returns a vector of file paths. Downloads are validated against authentication-failure HTML pages so that no broken file is ever saved silently.

Usage

```
getLandsat(product = "landsat-c211", start_date, end_date, aoi,
  download = FALSE, path,
  username, password, auth = NULL,
  bands = NULL,
  limit = 1000, overwrite = FALSE,
  verbose = TRUE, ...)
```

Arguments

<code>product</code>	character. STAC collection id. One of "landsat-c211" (Level-1, all sensors), "landsat-c212-sr" (Level-2 Surface Reflectance), or "landsat-c212-st" (Level-2 Surface Temperature). Legacy CMR short names such as "Landsat_8_OLI_TIRS_C1" are translated to "landsat-c211" with a message.
<code>start_date</code>	character or Date. Start date for the search, formatted yyyy-m-d.
<code>end_date</code>	character or Date. End date for the search, formatted yyyy-m-d.
<code>aoi</code>	numeric vector of four elements (xmin, xmax, ymin, ymax) encompassing the area of interest. Or a <code>SpatExtent</code> or any object from which an extent can be extracted.
<code>download</code>	logical. If TRUE the matching assets are downloaded to path.
<code>path</code>	character. Directory in which to store downloaded files.
<code>username</code>	character. USGS Earthdata Login user name. <i>Deprecated</i> - prefer creating an auth session up front via erosLogin (see Details).
<code>password</code>	character. USGS Earthdata Login password. <i>Deprecated</i> - see username.

auth	an eros_session returned by erosLogin . Required for downloads unless the deprecated username/password arguments are supplied.
bands	optional character vector of asset keys (e.g. c("blue", "green", "red", "nir08")) to restrict the result. The match is case-insensitive. If NULL (the default) all assets of every matching scene are returned, including band TIFs and metadata files such as MTL.txt and ANG.txt.
limit	positive integer. Maximum number of scenes to retrieve from the STAC search.
overwrite	logical. If TRUE, existing files are re-downloaded.
verbose	logical. If FALSE, progress bars and per-file cache/progress messages are suppressed.
...	Additional arguments. The legacy server and version arguments are accepted for backwards compatibility but ignored, with a warning.

Details

Landsat data are distributed by USGS, not NASA. To download data you need a USGS Earthdata Login account; you can sign up for free at <https://ers.cr.usgs.gov/register>. The same Earthdata Login can be used with NASA EOSDIS (see [getNASA](#)), but the actual authentication flow is different - USGS requires an interactive form-login on ers.cr.usgs.gov.

Authentication. The recommended way to supply credentials is to call [erosLogin](#) once and pass the returned session via auth:

```
auth <- erosLogin("my_user", "my_password")
getLandsat(..., download = TRUE, auth = auth)
```

[erosLogin\(\)](#) validates the credentials up front, so authentication errors are caught immediately rather than only when a download is attempted, and the session can be reused across many [getLandsat\(\)](#) calls without re-authenticating. The username/password arguments are still accepted for backwards compatibility but are deprecated and may emit a warning. They are ignored when auth is supplied.

Caching. When download = TRUE and an asset already exists in path (and overwrite = FALSE), [getLandsat\(\)](#) re-uses the local copy and does *not* contact USGS. With verbose = TRUE this is announced as a "reading from cache: ..." message. Use [erosLogin](#) to validate credentials independently of the cache.

For programmatic access at scale, also consider the official USGS M2M API (<https://m2m.cr.usgs.gov/>) or the **rstac** package, which both target the same STAC catalog.

Value

A character vector. When download = FALSE, asset URLs that match the search and any bands filter. When download = TRUE, the local file paths of successfully downloaded assets.

See Also

[erosLogin](#), [getProducts](#), [getNASA](#)

Examples

```
## Not run:
sdate <- "2019-05-01"
edate <- "2019-05-30"
area <- c(33, 34, -5, -4)
dir <- tempdir()

# list all assets for matching scenes
urls <- getLandsat("landsat-c211", start_date = sdate, end_date = edate,
                  aoi = area, download = FALSE)
head(urls)

# only the four optical bands of every matching scene
urls <- getLandsat("landsat-c211", start_date = sdate, end_date = edate,
                  aoi = area, download = FALSE,
                  bands = c("blue", "green", "red", "nir08"))

# download a Surface Reflectance scene (requires Earthdata Login credentials)
# auth <- erosLogin("your_user", "your_pwd")
# files <- getLandsat("landsat-c212-sr",
#                   start_date = sdate, end_date = edate,
#                   aoi = area, download = TRUE, auth = auth,
#                   bands = c("blue", "green", "red", "nir08"),
#                   path = dir)

## End(Not run)
```

getModis

Obsolete

Description

Obsolete. See [getNASA](#)

Usage

```
getModis(...)
```

Arguments

... Arguments passed to [getNASA](#)

getNASA

*Find and download NASA products***Description**

Find and download NASA data for a specific product, area, and time period. The function returned is list of filenames that may have been downloaded.

It is important to first consult [getProducts](#) to discover what products and version numbers are available, and which server to use.

Usage

```
getNASA(product, start_date, end_date, aoi, version=NULL, download=FALSE, path,
         username, password, auth=NULL,
         server="LPDAAC_ECS", limit=100000, overwrite=FALSE,
         pattern=NULL, verbose=TRUE, ...)
```

Arguments

product	character. Supported products can be found using getProducts
start_date	character. Start date for the data requested formatted yyyy-m-d
end_date	Character. end date for the data requested formatted yyyy-m-d
aoi	numeric vector of four elements (minimum longitude, maximum longitude, minimum latitude, maximum latitude) encompassing the area of interest. It can also be a SpatExtent, or any object from which an extent can be extracted
version	character. product version number
download	logical. If TRUE data will be downloaded unless it is present in the download directory
path	character. Path name indicating where to store the data
username	character. NASA Earthdata Login user name. <i>Deprecated</i> - prefer creating an auth session up front via earthdataLogin (see Details).
password	character. NASA Earthdata Login password. <i>Deprecated</i> - see username.
auth	an earthdata_session returned by earthdataLogin . Required for downloads unless the deprecated username/password arguments are supplied.
server	character. The server to download the data from
limit	positive integer
overwrite	logical. If TRUE, existing files are overwritten
pattern	optional character vector of regular expressions. Granules whose filename does not match any of the patterns are dropped. The match is case-insensitive and uses grepl . When NULL all granules are kept.
verbose	logical. If FALSE, progress bars and per-file cache/progress messages printed during downloading are suppressed
...	Additional arguments. These can be product specific. See Details)

Details

This function is an R interface to the EOSDIS system. Access is free, but it requires a "Earthdata" account. If you do not have an account, you can sign up here: <https://urs.earthdata.nasa.gov/users/new>

In addition to having an account, before you can download a particular product you must accept its End User License Agreement (EULA). The EULA is presented when you first try to download a product through the EOSDIS web interface, and can also be reviewed under your profile at <https://urs.earthdata.nasa.gov/>. If your credentials are wrong or you have not accepted the EULA, downloads will fail with an authentication error rather than silently saving an HTML error page as if it were data.

Authentication. The recommended way to supply credentials is to call `earthdataLogin` once and pass the returned session object via `auth`:

```
auth <- earthdataLogin("my_user", "my_password")
getNASA(..., download = TRUE, auth = auth)
```

`earthdataLogin()` validates the credentials up front against URS, so authentication errors are caught immediately rather than only when a download is attempted. The session can be reused across many `getNASA()` calls without re-authenticating.

The username/password arguments are still accepted for backwards compatibility but are deprecated and may emit a warning. They are ignored when `auth` is supplied.

Caching. When `download = TRUE` and a granule already exists in path (and `overwrite = FALSE`), `getNASA()` re-uses the local copy and does *not* contact Earthdata. With `verbose = TRUE` this is announced as a "reading from cache: ..." message, so it is clear that no download (and no credential check) was performed for that file. Use `earthdataLogin` to validate credentials independently of the cache.

If no data is available between `start_date` and `end_date`, files for the closest dates are returned.

NASA periodically retires older product collections in favor of new ones (for example, MOD17A3H was superseded by MOD17A3HGF). When the requested product/version cannot be found, `getNASA` reports the available alternatives in the cached collection list and suggests similar product short names; consult `getProducts` for the full inventory.

Value

Character vector of file names pointing to the downloaded files (if `download=TRUE`)

See Also

[earthdataLogin](#), [getProducts](#), [getLandsat](#)

Examples

```
product <- "MOD09A1"
# information about the product
getProducts(product)
sdate <- "2019-05-17"
edate <- "2019-05-23"
```

```

area <- c(33, 34, -5, -4)
path <- tempdir()
# list available tiles
f <- getNASA(product = product,
              start_date = sdate, end_date = edate,
              aoi = area, download = FALSE, path=path)

f

# get the date from the filenames
modisDate(f)

# to download the tiles, set download = TRUE
# and provide your credentials via an explicit Earthdata Login session
#auth <- earthdataLogin("your_user", "your_password")
#f <- getNASA(product = product, auth = auth,
#             start_date = sdate, end_date = edate,
#             aoi = area, download = TRUE, path=path)

# Daymet bundles several variables (tmin, tmax, prcp, srad, vp, swe, dayl)
# under a single collection; use `pattern` to download only the ones you
# want instead of all of them:
#f <- getNASA(product = "Daymet_Daily_V4R1_2129", server = "ORNL_CLOUD",
#             start_date = "2020-06-29", end_date = "2020-07-01",
#             aoi = area, pattern = c("tmin", "tmax"),
#             download = TRUE, auth = auth, path = path)

```

getProducts

*Find NASA product names***Description**

Find available products in NASA's Common Metadata Repository (CMR) that can be downloaded. See [getNASA](#) for further use.

Usage

```
getProducts(product, ...)
```

Arguments

product	character
...	additional arguments. None implemented

Value

character

See Also

[getNASA](#)

Examples

```
# list of all available MODIS products
getProducts("^MOD|^MYD|^MCD")
```

mesma

Multiple Endmember Spectral Mixture Analysis (Spectral Unmixing)

Description

mesma performs a multiple endmember spectral mixture analysis on a multiband raster image.

For unmixing, a non-negative least squares (NNLS) regression which is using a sequential coordinate-wise algorithm (SCA) based on Franc et al. (2005).

Usage

```
## S4 method for signature 'SpatRaster'
mesma(x, em, iterate=400, tolerance=0.00000001, ...)
```

Arguments

x	SpatRaster. Usually representing a hyperspectral remotely sensed image
em	matrix or data.frame with spectral endmembers. Rows represent a single end-member of a class, columns represent the spectral bands (i.e. columns correspond to number of bands in img). The number of rows needs to be > 1
iterate	integer. Set maximum iteration per pixel. Processing time could increase the more iterations are made possible
tolerance	numeric. Tolerance limit representing a nearly zero minimal number
...	further arguments passed to writeRaster

Value

SpatRaster with one layer per end-member, with each value representing the estimated presence probability of the end-member per pixel (0 to 1), and an RMSE layer.

Note

Depending on iterate and tolerance settings, the sum of estimated presence probabilities per pixel varies around 1.

Author(s)

Jakob Schwalb-Willmann

References

Franc, V., Hlaváč, V., & Navara, M. (2005). Sequential coordinate-wise algorithm for the non-negative least squares problem. In: International Conference on Computer Analysis of Images and Patterns (pp. 407-414). Berlin, Heidelberg.

Examples

```
lsat <- rast(system.file("ex/lsat.tif", package="luna"))

# endmember spectra: water and land
em_names <- c("water", "land")
pts <- data.frame(class=em_names, cell = c(47916,5294))
em <- lsat[pts$cell]
rownames(em) <- em_names

# unmix the image for water and land
probs <- mesma(lsat, em)

# take a look
hist(probs$water)
plot(probs$water, col = c("white","blue"))
hist(probs$land)
plot(probs$land, col = c("white","brown"))
```

modis_mask	<i>MODIS mask</i>
------------	-------------------

Description

Create a "mask" for a MODIS image based your selection of bad quality pixels, using the Quality Assessment (QA) band.

Usage

```
modis_mask(x, bits, qmat, ...)
```

Arguments

x	SpatRaster
bits	single integer value: 16 or 32
qmat	three-column matrix of data.frame. The first and second columns are positive integers (or character representation thereof) indicating the start and end bits (starting counting at one, not at zero!). The third column has the patterns to be rejected, stored as comma-separated characters. See example
...	additional arguments, none implemented

Value

SpatRaster

Examples

```
## Not run:
f <- "MOD09A1.A2011009.h21v08.006.2015216112834.hdf"
mod <- rast(f)
shortnames <- substr(names(mod), 86, 120)
names(mod) <- shortnames

qc <- mod[[12]]

from <- c(1,3,11,12) #,16)
to <- c(2,6,11,14) #,17)
reject <- c("10,11", "1100,1101,1110,1111", "1", "000,110,111") # "11")

qmat <- cbind(from, to, reject)

m <- modis_mask(qc, 16, qmat)
plot(m)

## End(Not run)
```

modisDate	<i>Extract the date from a MODIS filename</i>
-----------	---

Description

Extract the date that is encoded in a MODIS filename

Usage

```
modisDate(filename)
```

Arguments

filename character. MODIS product filenames

Value

data.frame

See Also

[getNASA](#)

Examples

```
modisDate("MOD09A1.A2019137.h21v09.006.2019150085736.hdf")
```

modisExtent	<i>Get the extent of a MODIS tile</i>
-------------	---------------------------------------

Description

Get the extent of a MODIS tile from the h (horizontal; column number) and v (vertical; row number) identifier in the filename

Usage

```
modisExtent(f=NULL, h, v)
```

Arguments

f	character. filename that includes a h12v09 pattern
h	integer between 0 and 35
v	integer between 0 and 17

Value

SpatExtent

See Also

[modisDate](#)

Examples

```
f <- "MOD09A1.A2019137.h21v09.006.2019150085736.hdf"
modisExtent(f)

modisExtent(h=21, v=9)
```

panSharpen	<i>Panchromatic sharpening</i>
------------	--------------------------------

Description

Sharpen multispectral imagery using a higher-resolution panchromatic band. Three transforms are supported: "Brovey", "HSI", and "PCA".

Usage

```
## S4 method for signature 'SpatRaster,SpatRaster'
panSharpen(x, p, method="Brovey", weights=NULL, filename="", overwrite=FALSE, ...)
```

Arguments

x	SpatRaster with the multispectral layers (bands) to be sharpened.
p	Single-layer SpatRaster with the panchromatic band, at higher spatial resolution than x.
method	character. One of "Brovey", "HSI", or "PCA".
weights	optional numeric vector of weights, one per layer of x. Used to compute the intensity component as $I = \sum_i w_i B_i$ (after normalising the weights to sum to 1). When NULL (the default), I is the unweighted mean of the multispectral bands. Ignored (with a warning) when method = "PCA", which derives its own component weighting from the data.
filename	character. Optional output file. When supplied, the result is written via writeRaster .
overwrite	logical. If TRUE, an existing filename is overwritten.
...	additional arguments for writeRaster (only used when filename is supplied).

Details

x is first resampled to the geometry of p.

Brovey. For each multispectral band B_i , the Brovey transform returns $B'_i = B_i \cdot P/I$. Without weights, I is the per-pixel mean of the multispectral bands. With weights, I is the weighted sum, equivalent to the algorithm used by GDAL's `gdal_pansharp.py` (the default GDAL pansharp-ening method).

HSI The implementation follows the additive (also called *fast*) IHS formulation: substituting the intensity component with a pan band that has been histogram-matched to I is algebraically equivalent to $B'_i = B_i + (P' - I)$, where P' is the pan band linearly rescaled so its mean and standard deviation match those of I .

PCA. The multispectral stack is forward-projected onto its principal components using [prcomp](#); the first principal component (the dominant variance direction, analogous to a data-driven intensity) is then replaced with the pan band rescaled to its mean and sd, and the inverse rotation is applied to recover sharpened bands. Columns of the rotation matrix are flipped so that each component's loadings sum positively. PCA pan-sharpening is data-driven and well-suited to multispectral inputs with more than 3 bands, where Brovey and IHS rely on heuristic intensity definitions.

If you want behaviour equivalent to `gdal_pansharp.py`, use method = "Brovey" with weights reflecting the spectral overlap of each multispectral band with the panchromatic band; for sensors where the pan band covers the visible plus NIR (e.g. Landsat 8 OLI band 8) typical weights are roughly c(R=0.21, G=0.27, B=0.42, NIR=0.10).

Value

SpatRaster with the same number of layers as x, at the resolution of p.

See Also

[colorize](#)

Examples

```
library(terra)

f <- system.file("ex/lsat.tif", package = "luna")
ms <- rast(f)[[1:3]] # B (B1), G (B2), R (B3)

# Synthesise a higher-resolution "panchromatic" band for the example.
# In practice, `p` would be the actual sensor pan band.
pan <- disagg(mean(rast(f)[[1:4]]), 2)

# Brovey (GDAL default)
b1 <- panSharpen(ms, pan, method = "Brovey")

# Weighted Brovey, matching gdal_pansharpen.py semantics
b2 <- panSharpen(ms, pan, method = "Brovey",
  weights = c(0.42, 0.27, 0.21))

# Additive IHS (works for any number of bands)
h <- panSharpen(ms, pan, method = "HSI")

# PCA (data-driven; recommended for >3 bands)
pc <- panSharpen(ms, pan, method = "PCA")

# Compare resolutions
res(ms)
res(pan)
res(b1)
```

productInfo

NASA product information

Description

Get complete product information in the corresponding product website. The information is displayed in the default browser. See [getProducts](#) for available options of products.

Usage

```
productInfo(product, ...)
```

Arguments

product	character
...	additional arguments (non implemented)

See Also

[getProducts](#)

Examples

```
## Not run:  
# MODIS in LP DAAC with version information  
productInfo(product = "MCD18A1")  
# Without any version or server information  
productInfo(product = "LANDSAT_SURFACE_REFLECTANCE_L8_OLI_TIRS")  
  
## End(Not run)
```

Index

- * **spatial**
 - modis_mask, [15](#)
- colorize, [18](#)
- download.file, [6](#)
- earthdataLogin, [2](#), [4](#), [11](#), [12](#)
- erosLogin, [3](#), [3](#), [8](#), [9](#)
- fillVI (filterVI), [5](#)
- filterVI, [5](#)
- getAVHRR, [6](#)
- getCGLS, [7](#)
- getLandsat, [3](#), [4](#), [8](#), [12](#)
- getModis, [10](#)
- getNASA, [2](#), [3](#), [7](#), [9](#), [10](#), [11](#), [13](#), [16](#)
- getProducts, [9](#), [11](#), [12](#), [13](#), [19](#)
- grepl, [11](#)
- luna (luna-package), [2](#)
- luna-package, [2](#)
- mesma, [14](#)
- mesma, SpatRaster-method (mesma), [14](#)
- modis_mask, [15](#)
- modisDate, [16](#), [17](#)
- modisExtent, [17](#)
- panSharpen, [17](#)
- panSharpen, SpatRaster, SpatRaster-method
(panSharpen), [17](#)
- prcomp, [18](#)
- print.earthdata_session
(earthdataLogin), [2](#)
- print.eros_session (erosLogin), [3](#)
- productInfo, [19](#)
- sgolayfilt, [5](#)
- splinefun, [5](#)
- writeRaster, [14](#), [18](#)