Package 'sen2r'

February 27, 2020

Type Package

Title Find, Download and Process Sentinel-2 Data

Version 1.3.2

Description Functions to download Sentinel-2 optical images
and perform preliminary processing operations.
'sen2r' provides the instruments required to easily perform
(and eventually automate) the steps necessary to build a complete
Sentinel-2 processing chain.
A Graphical User Interface to facilitate data processing is also provided.

License GPL-3

Encoding UTF-8

URL http://sen2r.ranghetti.info, https://github.com/ranghetti/sen2r

BugReports https://github.com/ranghetti/sen2r/issues

Depends R (>= 3.5.0)

Imports methods, sf, stars, data.table, raster, XML, jsonlite,
geojsonio, leaflet, leaflet.extras, mapedit, shiny, shinyFiles,
shinydashboard, shinyjs, shinyWidgets, foreach, parallel,
doParallel, httr

Suggests rgdal, spelling, geojsonlint, httptest, knitr, rmarkdown,
sys, tools, units, testthat (>= 2.1.0)

SystemRequirements GDAL (>= 2.1.2), PROJ (>= 4.9.1), GEOS (>= 3.4.2),
Cairo, Curl, NetCDF, jq, Protocol Buffers, V8, OpenSSL,
Libxml2.

VignetteBuilder knitr

RoxygenNote 7.0.2

Language en-GB

NeedsCompilation no

Author Luigi Ranghetti [aut, cre] (<https://orcid.org/0000-0001-6207-5188>),
Lorenzo Busetto [aut] (<https://orcid.org/0000-0001-9634-6038>)

Maintainer Luigi Ranghetti <luigi@ranghetti.info>

Repository CRAN

Date/Publication 2020-02-27 15:00:19 UTC

R topics documented:

`abs2rel` *Convert a path to a relative path*

Description

The function convert an absolute path to a relative path in respect to a reference. The longest common parent directory is taken as reference. Symbolic links are converted to original paths before performing the operation.

Usage

```
abs2rel(path, ref_path, mustWork = NA)
```

Arguments

`path`	The path to be converted (if it is not absolute, the current working directory is considered as its parent, and a warning is shown).
`ref_path`	The reference path to be compared to `path` to obtain the relative directory. *Important*: the path is considered as a directory also if it is the path of a file!
`mustWork`	(optional) logical: if TRUE an error is given if `path` or `ref_path` do not exists; if NA (default) then a warning; if FALSE nothing is shown.

Value

The relative path

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2017) <luigi.ranghetti@gmail.com>

Examples

```
# the reference path
(ref_path <- system.file(package="sen2r"))
# a path with a common parent with ref_path
(in_path_1 <- system.file(package="gdalUtils"))
# a path included in ref_path
(in_path_2 <- system.file("R/abs2rel.R", package="sen2r"))
# a path external to ref_path (in Linux)
(in_path_3 <- system.file(package="base"))
# an unexisting path
(in_path_4 <- gsub("sen2r","r2sen",ref_path))

abs2rel(in_path_1, ref_path)

abs2rel(in_path_2, ref_path)

suppressWarnings(abs2rel(in_path_3, ref_path))

suppressWarnings(abs2rel(in_path_4, ref_path, mustWork=FALSE))

suppressWarnings(abs2rel(ref_path, ref_path))
```

---

add_rgb_image *Add RGB product*

Description

Modal dialog to define an RGB image.

Usage

```
add_rgb_image(s2_bands)
```

Arguments

s2_bands	2-length list (one for TOA, one for BOA), each element being a list of S2 bands, as defined in s2_gui.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`build_example_param_file` *Build an example JSON parameter file*

Description

Function used to write JSON parameter file. A function is provided instead than a json file to ensure directories to match the user folder tree.

Usage

```
build_example_param_file(
  json_path = tempfile(fileext = "_sen2r_params.json"),
  overwrite = TRUE
)
```

Arguments

`json_path`	Path of the output file. Default is to save it on a temporary file, whose path is returned.
`overwrite`	Logical value: should existing output file be overwritten? (default: TRUE)

Value

The path of the created file.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
build_example_param_file()
```

check_gdal *Check GDAL installation*

Description

The function check that GDAL is installed and updated to the minimum required version (2.1.2).

Usage

```
check_gdal(abort = TRUE, gdal_path = NULL, force = FALSE, full_scan = FALSE)
```

Arguments

abort	Logical parameter: if TRUE (default), the function aborts in case no GDAL installation is found; if FALSE, a warning is shown and FALSE is returned.
gdal_path	(optional) Character: the path in which GDAL must be searched in. If NULL (default), search is performed in the whole file system.
force	(optional) Logical: if TRUE, install even if it is already installed (default is FALSE). Notice that, defining gdal_path, GDAL is searched again even if "force" = FALSE in case the existing installation is not in gdal_path.
full_scan	(optional) Logical: in Linux and MacOS, if gdal_path was not manually defined, GDAL is searched within the system path in case this argument is left to default value FALSE; instead, if TRUE, a full search is performed. In Windows, if the default OSGeo directory C:\\OSGeo4W64 exists, GDAL is searched there, instead in the main directory C:\\; setting full_scan to TRUE, is is always searched in the whole C:\\. This argument takes no effect if gdal_path was defined, since, in that case, a full search is always performed in gdal_path.

Value

Logical (invisible): TRUE in case the installation is ok, FALSE if GDAL is missing and abort=FALSE (otherwise, the function stops).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Use function
check_gdal()
```

```
# Check GDAL was imported
load_binpaths()$gdalinfo
```

`check_param_list` *Check a parameter list*

Description

Check that the parameter list (or JSON parameter file) is in the correct format, and then specified values are coherent with parameters.

Usage

```
check_param_list(pm, type = "string", check_paths = FALSE, correct = TRUE)
```

Arguments

`pm`	List of parameters or path of a JSON parameter file.
`type`	Type of the output (see print_message for details).
`check_paths`	Logical: if TRUE, the function checks required output paths to be provided; if FALSE (default) these checks are skipped.
`correct`	Logical: if TRUE (default), the function corrects some incoherences (e.g. timewindow of length 1 is transformed in length 2) and returns the corrected list as output; if false, only checking is performed, and the output is NULL if no errors occur.

Value

In case of errors, depending on `type` argument, output can be a vector of errors (if `type = "string"`), the first error occurred (if `type = "error"`) or a set of warnings (if `type = "warning"`). If no errors occur, output is the corrected parameter list if `correct = TRUE` or NULL otherwise.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`check_sen2r_deps` *Check package dependencies*

Description

The function allows to graphically check that all the dependencies are installed.

Usage

```
check_sen2r_deps()
```

Details

This package needs some external dependencies to run:

- Python
- GDAL
- Sen2Cor

This function opens a GUI which allows to check that these dependencies are installed. This check is highly suggested before using the library for the fist time, in order to avoid errors.

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
if (interactive()) {
  check_sen2r_deps()
}
```

compute_s2_paths *Compute names of S2 file to be generated*

Description

compute_s2_paths is an internal function (to be used within sen2r()) which computes the names of the required output image files (see details). The function was split from sen2r() because this code is called twice (and to shorten the main function).

list_sen2r_paths is a wrapper of sen2r, which runs sen2r until compute_s2_paths is called, returning the same list. It is a simple way to call compute_s2_paths with the same arguments if sen2r.

Usage

```
compute_s2_paths(
  pm,
  s2_list_l1c,
  s2_list_l2a,
  tmpdir,
  list_prods,
  force_tiles = FALSE,
  check_tmp = TRUE,
  ignorelist
)

list_sen2r_paths(
  param_list = NULL,
  gui = NA,
  preprocess = TRUE,
  s2_levels = c("l1c", "l2a"),
  sel_sensor = c("s2a", "s2b"),
  online = TRUE,
  apihub = NA,
  downloader = "builtin",
  overwrite_safe = FALSE,
  rm_safe = "no",
  step_atmcorr = "auto",
  max_cloud_safe = 100,
  timewindow = NA,
  timeperiod = "full",
  extent = NA,
  extent_name = "sen2r",
  s2tiles_selected = NA,
  s2orbits_selected = NA,
  list_prods = c("BOA"),
  list_rgb = NA,
```

```
  list_indices = NA,
  index_source = "BOA",
  rgb_ranges = NA,
  mask_type = NA,
  max_mask = 100,
  mask_smooth = 0,
  mask_buffer = 0,
  clip_on_extent = TRUE,
  extent_as_mask = FALSE,
  reference_path = NA,
  res = NA,
  res_s2 = "10m",
  unit = "Meter",
  proj = NA,
  resampling = "near",
  resampling_scl = "near",
  outformat = "GTiff",
  rgb_outformat = "GTiff",
  index_datatype = "Int16",
  compression = "DEFLATE",
  rgb_compression = "90",
  overwrite = FALSE,
  path_l1c = NA,
  path_l2a = NA,
  path_tiles = NA,
  path_merged = NA,
  path_out = NA,
  path_rgb = NA,
  path_indices = NA,
  path_subdirs = TRUE,
  thumbnails = TRUE,
  parallel = TRUE,
  processing_order = "by_step",
  use_python = TRUE,
  tmpdir = NA,
  rmtmp = TRUE,
  log = NA
)
```

Arguments

pm	List of input parameters.
s2_list_l1c	Names and paths of input SAFE level-1C products.
s2_list_l2a	Names and paths of input SAFE level-2A products.
tmpdir	Path of the temporary directory.
list_prods	Character vector with the values of the products to be processed (accepted values: "TOA", "BOA", "SCL", "TCI").

force_tiles	(optional) Logical: passed to safe_shortname (default: FALSE).
check_tmp	(optional) Logical: if TRUE (default), temporary files are also searched when exi names are computed; if FALSE, only non temporary files are searched.
ignorelist	Vector of output files to be ignored.
param_list	sen2r argument (refer to sen2r documentation).
gui	sen2r argument (refer to sen2r documentation).
preprocess	sen2r argument (refer to sen2r documentation).
s2_levels	sen2r argument (refer to sen2r documentation).
sel_sensor	sen2r argument (refer to sen2r documentation).
online	sen2r argument (refer to sen2r documentation).
apihub	sen2r argument (refer to sen2r documentation).
downloader	sen2r argument (refer to sen2r documentation).
overwrite_safe	sen2r argument (refer to sen2r documentation).
rm_safe	sen2r argument (refer to sen2r documentation).
step_atmcorr	sen2r argument (refer to sen2r documentation).
max_cloud_safe	sen2r argument (refer to sen2r documentation).
timewindow	sen2r argument (refer to sen2r documentation).
timeperiod	sen2r argument (refer to sen2r documentation).
extent	sen2r argument (refer to sen2r documentation).
extent_name	sen2r argument (refer to sen2r documentation).
s2tiles_selected	sen2r argument (refer to sen2r documentation).
s2orbits_selected	sen2r argument (refer to sen2r documentation).
list_rgb	sen2r argument (refer to sen2r documentation).
list_indices	sen2r argument (refer to sen2r documentation).
index_source	sen2r argument (refer to sen2r documentation).
rgb_ranges	sen2r argument (refer to sen2r documentation).
mask_type	sen2r argument (refer to sen2r documentation).
max_mask	sen2r argument (refer to sen2r documentation).
mask_smooth	sen2r argument (refer to sen2r documentation).
mask_buffer	sen2r argument (refer to sen2r documentation).
clip_on_extent	sen2r argument (refer to sen2r documentation).
extent_as_mask	sen2r argument (refer to sen2r documentation).
reference_path	sen2r argument (refer to sen2r documentation).
res	sen2r argument (refer to sen2r documentation).
res_s2	sen2r argument (refer to sen2r documentation).
unit	sen2r argument (refer to sen2r documentation).

proj	sen2r argument (refer to sen2r documentation).
resampling	sen2r argument (refer to sen2r documentation).
resampling_scl	sen2r argument (refer to sen2r documentation).
outformat	sen2r argument (refer to sen2r documentation).
rgb_outformat	sen2r argument (refer to sen2r documentation).
index_datatype	sen2r argument (refer to sen2r documentation).
compression	sen2r argument (refer to sen2r documentation).
rgb_compression	sen2r argument (refer to sen2r documentation).
overwrite	sen2r argument (refer to sen2r documentation).
path_l1c	sen2r argument (refer to sen2r documentation).
path_l2a	sen2r argument (refer to sen2r documentation).
path_tiles	sen2r argument (refer to sen2r documentation).
path_merged	sen2r argument (refer to sen2r documentation).
path_out	sen2r argument (refer to sen2r documentation).
path_rgb	sen2r argument (refer to sen2r documentation).
path_indices	sen2r argument (refer to sen2r documentation).
path_subdirs	sen2r argument (refer to sen2r documentation).
thumbnails	sen2r argument (refer to sen2r documentation).
parallel	sen2r argument (refer to sen2r documentation).
processing_order	sen2r argument (refer to sen2r documentation).
use_python	sen2r argument (refer to sen2r documentation).
rmtmp	sen2r argument (refer to sen2r documentation).
log	sen2r argument (refer to sen2r documentation).

Details

compute_s2_paths is structured in the following way:

1. Retrieve the file names expected to be present at the end of the processing chain (element exp) and already existing (exi);
2. Compute the file names expected to be created (elements req and new, see below) (this operation is done in reverse order).

   Meaning of the elements exi, exp, req and new (here and for all the script), which are defined foe each processing step:

   - exi: full names of the files already existing before launching the processing chain;
   - exp: full names of the files expected to be present at the end of the processing chain (already existing or not);
   - req: names of the files required by the step;
   - new: names of the required files not existing yet (expected to be created).

With overwrite=TRUE, all these vectors are equal because all is overwritten.

Value

A nested list:

- first elements are `exi`, `exp`, `req` and `new`;
- second elements deal with the processing step: `tiles`, `merged`, `warped`, `warped_scl`, `rgb`, `masked` and `indices`;
- third elements are related to output products.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) `<luigi@ranghetti.info>`

---

comsub *Find the longest common starting substring or directory*

Description

The function search for the longest common prefix between multiple strings.

Usage

```
comsub(data, sep = "")
```

Arguments

`data`	A vector of strings
`sep`	A character which is used to separate elements; default ("") is used to compare single characters; other useful alternatives are "/" (or "\\\\" in Windows) to find the longest common directory, or " " to compare words instead of characters.

Value

A character with the longest common initial substring

Note

Modified from a suggestion taken from stackoverflow.

Author(s)

Luigi Ranghetti, phD (2019) `<luigi@ranghetti.info>`

Examples

```
strings <- c("/home/user/git/sen2r",
             "/home/user/git_data/sen2r/ex/vrt/01_translate/")

comsub(strings)

comsub(strings, sep="/")
```

`create_indices_db` *Create the indices database*

Description

The internal function checks if indices.json (the database of spectral indices) already exists; if not, it downloads source files and creates it. Since this function depends on `xsltproc` executable (available only for Linux), this function can be used only from from Linux. It is not necessary, since a indices.json file is present in the package.

Usage

```
create_indices_db(xslt_path = NA, json_path = NA, force = FALSE)
```

Arguments

`xslt_path`	(optional) The path where to install `xsltml`, an external `xsltproc` script used to convert MathML index formulas to LaTeX (default: a subdirectory of the package).
`json_path`	(optional) The path of the output JSON file. *Warning*: to create a file which will be usable by the package, this option must be left to NA (default location is within the package installation). Edit this only to create the file in another place for external use.
`force`	(optional) Logical: if FALSE (default), the db is created only if missing or not updated; if TRUE, it is created in any case.

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`create_s2_dop` *Create the database of S2 orbits and doy of passage*

Description

The internal function build a database with the base DOY of passage across each Sentinel-2A orbit (which is used in function `s2_dop`).

Usage

```
create_s2_dop(json_path = NA, force = FALSE)
```

Arguments

`json_path`	(optional) The path of the output JSON file. *Warning*: to create a file which will be usable by the package, this option must be left to NA (default location is within the package installation). Edit this only to create the file in another place for external use.
`force`	(optional) Logical: if FALSE (default), the db is created only if missing or not updated; if TRUE, it is created in any case.

Value

The path of the json file

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`editModPoly` *Shiny Module Server for Geo Create, Edit, Delete*

Description

Shiny Module Server for Geo Create, Edit, Delete

Usage

```
editModPoly(
  input,
  output,
  session,
  leafmap,
  targetLayerId = NULL,
  sf = TRUE,
  record = FALSE,
  crs = 4326
)
```

Arguments

input	Shiny server function input
output	Shiny server function output
session	Shiny server function session
leafmap	leaflet map to use for Selection
targetLayerId	character identifier of layer to edit, delete
sf	logical to return simple features. sf=FALSE will return GeoJSON.
record	logical to record all edits for future playback.
crs	CRS (EPSG) to be used

Value

server function for Shiny module

Note

Slightly edited from mapedit::editMod in order to allow drawing only polygons.

---

expand_path *Expand a path with a parent directory*

Description

Accessory function which checks if a path is absolute or relative; if relative, use a specified parent directory instead than the working directory to expand it. Useful for functions which accept more than one path as arguments, in which one of them contains the absolute position, and the others do not.

Usage

```
expand_path(path, parent = getwd(), silent = TRUE, normalize = TRUE)
```

Arguments

path	The path name (character) to check ad eventually expand.
parent	The parent directory (character) to use if path is relative (default value: the working directory).
silent	Logical value: if TRUE (default), no message are shown; if FALSE, a message inform if parent were applied or not; if NA, a warning is returned if path is expanded, nothing if it is already an absolute path.
normalize	Logical value: if TRUE (default), the path is normalised (normalizePath is applied); if FALSE it is simply appended.

Value

The path eventually expanded.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

---

fix_envi_format *Fix ENVI outputs*

Description

Internal function which changes some elements of output ENVI files:

- file extension is set to .dat if .envi (in case of files created by writeRaster) is found, and the header is edited properly,
- and band names are set in the header file (in particular, SR band names include wavelengths and names like NIR, SWIR; other products shows the product name as band name);
- SCL headers include information about class names and colours.

Usage

```
fix_envi_format(infiles)
```

Arguments

infiles	A vector of input filenames, in the sen2r naming convention (safe_shortname) and ENVI format.

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`gdalwarp_grid` *Warp a raster file aligning it on the grid of another file*

Description

The function applies `gdalwarp` to build rasters with the same projection, resolution and grid alignment of another raster. If not specified, the output format of each file is the same of the corresponding source file.

Usage

```
gdalwarp_grid(srcfiles, dstfiles, ref, of = NULL, r = NULL, tmpdir = tempdir())
```

Arguments

`srcfiles`	A vector of input file paths (managed by GDAL).
`dstfiles`	A vector of input file paths.
`ref`	Path of the raster taken as reference.
`of`	The output format (use the short format name). Default is the format of every input filename.
`r`	Resampling_method (`"near"|"bilinear"|"cubic"|"cubicspline"|"lanczos"|"average"|"mode"|"ma`
`tmpdir`	(optional) Path where intermediate files (.prj) will be created. Default is a temporary directory.

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Define file names
ex_sel <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_BOA_10.tif",
  package = "sen2r"
)
ex_ref <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_SCL_10.tif",
  package = "sen2r"
)
ex_out <- tempfile(fileext = "_BOA_out.tif")

# Run function
sen2r:::gdalwarp_grid(ex_sel, ex_out, ref = ex_ref)

# Show output
oldpar <- par(mfrow = c(1,3), mar = rep(0,4))
image(stars::read_stars(ex_sel), rgb = 4:2, maxColorValue = 3500)
par(mar = rep(2/3,4)); image(stars::read_stars(ex_ref))
par(mar = rep(0,4)); image(stars::read_stars(ex_out), rgb = 4:2, maxColorValue = 3500)
par(oldpar)
```

`gdal_abs2rel` *Convert absolute from/to relative paths in a virtual file*

Description

The two functions read the content of a GDAL virtual file (VRT) and check the presence of paths to linked files.

gdal_abs2rel scans the presence of absolute paths: when an absolute path has a common parent directory with the path in which the VRT is, this is replaced with a relative. This is useful when VRT are on a remote driver, which can be mounted to several points.

gdal_rel2abs checks the presence of relative paths, and replace them with the corresponding absolute path (symbolic links are followed). This is useful to grant that VRT can be moved (if the files they link to are not moved).

Usage

```
gdal_abs2rel(in_vrt, out_vrt = NA)

gdal_rel2abs(in_vrt, out_vrt = NA)
```

Arguments

`in_vrt`	The path of the VRT to be read.
`out_vrt`	(optional) The path of the output VRT file (default is to overwrite `in_vrt`).

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Load a VRT containing a relative path
ex_vrt <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_RGB432B_10.vrt",
  package = "sen2r"
)
abs_vrt <- tempfile(fileext = "_abs.vrt")
rel_vrt <- tempfile(fileext = "_rel.vrt")
gdal_rel2abs(ex_vrt, abs_vrt)
gdal_abs2rel(ex_vrt, rel_vrt)

# Show differences
ex_vrt_content <- readLines(ex_vrt)
abs_vrt_content <- readLines(abs_vrt)
rel_vrt_content <- readLines(rel_vrt)
ex_vrt_content[ex_vrt_content != abs_vrt_content] # Original line
abs_vrt_content[ex_vrt_content != abs_vrt_content] # Modified line
rel_vrt_content[ex_vrt_content != rel_vrt_content] # No edits
```

---

gdal_warp *Clip, reproject and warp raster files*

Description

The function applies gdalwarp to clip, reproject and/or warp raster files. If not specified, the output format of each file is the same of the corresponding source file.

Usage

```
gdal_warp(
  srcfiles,
  dstfiles,
  of = NULL,
  co = NULL,
  ref = NULL,
  mask = NULL,
```

```
  tr = NULL,
  t_srs = NULL,
  r = NULL,
  dstnodata = NULL,
  overwrite = FALSE,
  tmpdir = NA,
  rmtmp = TRUE
)
```

Arguments

srcfiles	A vector of input file paths (managed by GDAL).
dstfiles	A vector of corresponding output file paths.
of	The output format (use the short format name). Default is the format of every input filename.
co	Character. Passes a creation option to the output format driver. Multiple -co options may be listed. See format specific documentation for legal creation options for each format.
ref	Path of the raster taken as reference: if provided, parameters regarding the output grid (alignment, resolution and extent) are taken from this raster. To set differently some of these values, specify also other values of mask and/or tr. t_srs parameter value is always ignored when ref is provided.
mask	Spatial path or object from which to take the extent of output files. If it is a polygon, this is used as masking layer; otherwise, only the bounding box is considered. If both ref and mask are provided, this parameter will overlay the extent of the reference raster. In order to take only the grid from res and not to clip on its extent, set mask=NA. Notice that the output projection is never taken from mask.
tr	Numeric. (c(xres,yres)). set output file resolution (in target georeferenced units). If bot ref and tr are provided, tr is rounded in order to match the exact extent.
t_srs	Target spatial reference set (character). The coordinate systems that can be passed are anything supported by st_crs2.
r	Resampling_method ("near"|"bilinear"|"cubic"|"cubicspline"|"lanczos"|"average"|"mode"|"ma
dstnodata	Set nodata values for output bands (different values can be supplied for each band). If more than one value is supplied all values should be quoted to keep them together as a single operating system argument. New files will be initialized to this value and if possible the nodata value will be recorded in the output file. Use a value of NA to ensure that nodata is not defined. A vector with the same length of srcfiles can be supplied, in order to specify different nodata values for each input file. If this argument is not used then nodata values will be copied from the source datasets. At the moment it is not possible to set different values for different srcfiles (use multiple calls of the functions).
overwrite	Logical value: should existing output files be overwritten? (default: FALSE)

tmpdir	(optional) Path where intermediate files (maskfile) will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.
rmtmp	(optional) Logical: should temporary files be removed? (Default: TRUE)

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
#' # Define file names
ex_sel <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_RGB432B_10.tif",
  package = "sen2r"
)
ex_ref <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_SCL_10.tif",
  package = "sen2r"
)
crop_poly <- system.file("extdata/vector/dam.geojson", package = "sen2r")
crop_line <- sf::st_cast(sf::read_sf(crop_poly), "LINESTRING")

# Simple clip
test1 <- tempfile(fileext = "_test1.tif")
gdal_warp(ex_sel, test1, mask = crop_line)

# Clip and mask
test2 <- tempfile(fileext = "_test2.tif")
gdal_warp(ex_sel, test2, mask = crop_poly)

# Show output
crop_bbox <- sf::st_as_sfc(sf::st_bbox(crop_line))
oldpar <- par(mfrow = c(1,3), mar = rep(0,4))
image(stars::read_stars(ex_sel), rgb = 1:3)
plot(crop_line, add = TRUE, col = "blue", lwd = 2)
plot(crop_bbox, add = TRUE, border = "red", lwd = 2)
image(stars::read_stars(test1), rgb = 1:3)
plot(crop_bbox, add = TRUE, border = "red", lwd = 2)
image(stars::read_stars(test2), rgb = 1:3)
plot(crop_line, add = TRUE, col = "blue", lwd = 2)

# Warp on a reference raster
```

```
test3 <- tempfile(fileext = "_test3.tif")
gdal_warp(ex_sel, test3, ref = ex_ref)

# Show output
par(mfrow = c(1,3))
par(mar = rep(0,4)); image(stars::read_stars(ex_sel), rgb = 1:3)
par(mar = rep(2/3,4)); image(stars::read_stars(ex_ref))
par(mar = rep(0,4)); image(stars::read_stars(test3), rgb = 1:3)

# Reproject all the input file
test4 <- tempfile(fileext = "_test4.tif")
gdal_warp(ex_sel, test4, t_srs = 32631)

# Reproject and clip on a bounding box
test5 <- tempfile(fileext = "_test5.tif")
gdal_warp(ex_sel, test5, t_srs = "EPSG:32631", mask = stars::read_stars(test1))

# Reproject and clip on polygon (masking outside)
test6 <- tempfile(fileext = "_test6.tif")
gdal_warp(ex_sel, test6, t_srs = "31N", mask = crop_poly)

# Show output
crop_line_31N <- sf::st_transform(crop_line, 32631)
test1_bbox <- sf::st_as_sfc(sf::st_bbox(stars::read_stars(test1)))
test1_bbox_31N <- sf::st_transform(test1_bbox, 32631)
par(mfrow = c(1,4), mar = rep(0,4))
image(stars::read_stars(ex_sel), rgb = 1:3)
plot(crop_line, add = TRUE, col = "blue", lwd = 2)
plot(test1_bbox, add = TRUE, border = "red", lwd = 2)
image(stars::read_stars(test4), rgb = 1:3)
image(stars::read_stars(test5), rgb = 1:3)
plot(test1_bbox_31N, add = TRUE, border = "red", lwd = 2)
image(stars::read_stars(test6), rgb = 1:3)
plot(crop_line_31N, add = TRUE, col = "blue", lwd = 2)

# Use a reference raster with a different projection
test7 <- tempfile(fileext = "_test7.tif")
gdal_warp(ex_sel, test7, ref = test6)

# Use a reference raster with a different projection
# and specify a different bounding box
test8 <- tempfile(fileext = "_test8.tif")
gdal_warp(ex_sel, test8, mask = stars::read_stars(test1), ref = test6)

# Use a reference raster with a different projection and a mask
test9 <- tempfile(fileext = "_test9.tif")
gdal_warp(ex_sel, test9, mask = crop_poly, ref = test6)

# Show output
par(mfrow = c(1,4), mar = rep(0,4))
image(stars::read_stars(ex_sel), rgb = 1:3)
plot(crop_line, add = TRUE, col = "blue", lwd = 2)
image(stars::read_stars(test7), rgb = 1:3)
```

```
plot(crop_line_31N, add = TRUE, col = "blue", lwd = 2)
image(stars::read_stars(test8), rgb = 1:3)
plot(test1_bbox_31N, add = TRUE, border = "red", lwd = 2)
image(stars::read_stars(test9), rgb = 1:3)
plot(crop_line_31N, add = TRUE, col = "blue", lwd = 2)

par(oldpar)
```

`geograbber_process` *Preprocess (clip, reproject, reshape, mask) SAFE products*

Description

The function is a simplified wrapper of `sen2r()` to preprocess existing SAFE products. No online research is performed, neither sen2cor is applied to existing L1C products. Output products are GeoTIFF files clipped to a desired

Usage

```
geograbber_process(
  path_safe,
  path_out,
  bbox = NA,
  list_prods = c("BOA"),
  res = NA,
  ...
)
```

Arguments

`path_safe`	Path of the directory in which SAFE products are searched.
`path_out`	Path of the directory in which Sentinel-2 output products are generated.
`bbox`	(optional) Four-length numeric vector with the bounding box of the output products (`xmin`, `ymin`, `xmax`, `ymax`), in geographic coordinates.
`list_prods`	(optional) Character vector with the values of the products to be processed (accepted values: `"TOA"`, `"BOA"`, `"SCL"`, `"TCI"`). Default is `"BOA"`.
`res`	(optional) Numeric vector of length 2 with the x-y resolution for output products. NA (default) means that the resolution is kept as native.
`...`	Other additional sen2r arguments: `extent_name`, `list_indices`, `index_source`, `mask_type`, `max_mask`, `mask_smooth`, `mask_buffer`, `clip_on_extent`, `extent_as_mask`, `reference_path`, `res_s2`, `unit`, `proj`, `resampling`, `resampling_scl`, `outformat`, `index_datatype`, `compression`, `overwrite`, `path_tiles`, `path_merged`, `path_indices`, `path_subdirs`, `thumbnails`, `parallel`, `tmpdir`, `rmtmp`. See sen2r documentation for details.

gipp_init *Copy L2A_GIPP.xml in sen2r*

Description

Internal function to copy L2A_GIPP.xml from default Sen2Cor directory to sen2r. After that, user will allow editing Sen2Cor options in sen2r without affecting standalone Sen2Cor behaviour.

Usage

```
gipp_init(gipp_sen2r_path = NA, force = FALSE, dem_warning = FALSE)
```

Arguments

gipp_sen2r_path	Character path of the output GIPP XML file. By default it is equal to NA (meaning the default sen2r GIPP path).
force	Logical: if TRUE, the file is copied in any case; if FALSE (default), only if it does not yet exist.
dem_warning	TEMPORARY ARGUMENT Logical: if TRUE, a warning about the fact that DEM_Directory XML parameter was not overwritten is shown (default is FALSE). This argument will be removed when use_dem = TRUE will become the default.

Value

TRUE if the file was copied, FALSE elsewhere (invisible output)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

Examples

```
## Not run:
gipp_init()

## End(Not run)
```

give_write_permission *Give permission to write settings on disk*

Description

In interactive mode, ask users for permission to create a .sen2r settings directory, in which to store files required by the packages. The function can be used also in non-interactive mode by setting `agree = TRUE`. The function has no effect if the directory already exists.

Usage

```
give_write_permission(agree = NA)
```

Arguments

agree	Logical: if TRUE, allow creating the hidden directory; if FALSE, do not allow it; if NA (default), the permission is asked to the user in interactive mode (in non-interactive mode, the permission is denied).

Value

Logical: if TRUE, R was authorised saving in the directory; if FALSE, it was not and a temporary directory is being used.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2019) <lbusett@gmail.com>

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

init_python *Initialise python*

Description

Internal function to set the environmental variables required to run Python-based GDAL utilities on Windows.

Usage

```
init_python()
```

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

`install_aria2` *Download and install aria2.*

Description

This function download and install standalone version of aria2 for Windows.

Usage

```
install_aria2(aria2_dir, force = FALSE)
```

Arguments

`aria2_dir`	Path where aria2 executable will be installed.
`force`	(optional) Logical: if TRUE, install even if it is already installed (default is FALSE).

Value

The path of aria2

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
## Not run:
# Run only on Windows
install_aria2(aria2_dir = tempdir())
# ( use a non-temporary folder path instead of tempdir() )

## End(Not run)
```

install_sen2cor *Download and install (or link) Sen2Cor*

Description

install_sen2cor() downloads and installs a standalone version of Sen2Cor.

link_sen2cor() links an existing standalone version of Sen2Cor to sen2r.

Usage

```
install_sen2cor(sen2cor_dir = NA, version = "2.5.5", force = FALSE)

link_sen2cor(sen2cor_dir)
```

Arguments

sen2cor_dir	Path where sen2cor will be installed or searched (by default it is a subdirectory "sen2cor" of the default sen2r directory).
version	(optional) Character: Sen2Cor version (one among '2.5.5' - default - and '2.8.0').
force	(optional) Logical: if TRUE, installs sen2cor even if it is already found in sen2cor_dir (default is FALSE).

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
## Not run:
install_sen2cor(sen2cor_dir = tempdir())
# ( use a non-temporary folder path instead of tempdir() )

## End(Not run)
```

`list_indices` *List spectral indices*

Description

Return a table with attributes of the spectral indices computable with the package.

Usage

```
list_indices(values, pattern = "", all = FALSE)
```

Arguments

`values`	A vector of attributes which will be returned, being one or more within the followings: • `n_index`: internal index identifiers; • `name`: index name; • `longname`: index description; • `link`: URL to the index description page; • `s2_formula`: expression containing the formula to compute the index; • `s2_formula_mathml`: MathML version of the formula; • `checked`: logical (TRUE for verified indices); • `a`, `b`, `x`: parameter values (NA for non required parameters).
`pattern`	A regular expression on index names.
`all`	Logical: if TRUE, all the indices retrieved from IDB are returned; if FALSE (default), only indices checked by the authors are returned.

Value

A data.frame with the required information. The table contains also the following attributes:

- `creation_date`: timestamp of the creation date of the indices archive;
- `pkg_version`: version of the `sen2r` package used to create the indices archive.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Show index names
list_indices(c("name","longname"))

# Return the MSAVI2 formula
list_indices("s2_formula", "^MSAVI2$")

# Return all index names (including unchecked)
list_indices("name", all = TRUE)
```

`load_binpaths` *Load the paths of external executables*

Description

Internal function to load the paths of executables from the JSON where they are saved when installed.

Usage

```
load_binpaths(bins = NULL)
```

Arguments

`bins`	Character vector with one of more of the following values: "gdal", sen2cor", "aria2", "python". If an executable corresponding to the passed `bins` value is not found in the JSON, it is checked (when possible).

Value

The list of the paths

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Load only existing paths
binpaths <- load_binpaths()
binpaths


## Not run:
```

```
# Load paths, forcing to check GDAL and sen2cor
binpaths <- load_binpaths(c("gdal", "sen2cor"))
binpaths

## End(Not run)
```

`load_extent_bbox` *Insert an extent*

Description

Internal functions and modal dialogs to specify an extent in the GUI.

Usage

```
load_extent_bbox()

load_extent_vectfile()

load_extent_draw(extent_ns_name)
```

Arguments

`extent_ns_name`	Name of the namespace to be used

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`mountpoint` *Return the mountpoint of the input directory (if it is mounted)*

Description

The function checks if the input directory is a subdirectory of a mountpoint of a certain protocol. At the moment, it works only on unix operating systems.

Usage

```
mountpoint(path, protocol = NA)
```

Arguments

path	The path to be checked
protocol	(optional) Vector of protocol types. If NA (default), all the protocols are considered.

Value

The path of the parent mountpoint for mounted directories; if the input directory is not mounted, NULL is returned. NULL is returned also if the operating system is not unix (together with a warning message). An attribute "protocol" contains the protocol of the mountpoint.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

---

nn *Replace NULL with character()*

Description

Internal function: return character(0) instead of NULL. This is sometimes needed not to return error when applying some functions.

Usage

```
nn(x)
```

Arguments

x	Input variable

Value

`character(0)` if `x==NULL`, `x` elsewhere

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
tryCatch(basename(NULL), error = print) # error
basename(character()) # ok
basename(sen2r:::nn(NULL)) # ok
```

`normalize_path` *Express file paths in canonical Form depending on the operating system*

Description

Accessory function wrapper for `normalizePath()` in Linux and `shortPathName(normalizePath())` in Windows.

Usage

```
normalize_path(path, ...)
```

Arguments

`path`	character vector of file paths
`...`	additional parameters passed to normalizePath (i.e. `mustWork`).

Value

The paths normalized.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

path_check *Check a path*

Description

Accessory functions to check that a directory exists and the user have write permissions on it (to be used in a Shiny context)

Usage

```
path_check(path, mustbe_empty = FALSE, mustbe_writable = TRUE)
```

Arguments

path	string full path to a folder
mustbe_empty	logical if TRUE, accept only empty directories
mustbe_writable	logical if TRUE, accept only directories with write permissions

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Lorenzo Busetto, PhD (2019) <lbusett@gmail.com>

print_message *Print a message*

Description

A common interface for printing messages of several types.

Usage

```
print_message(
  ...,
  type,
  sep = "",
  date = FALSE,
  date_format = "",
  width = 0.9 * getOption("width"),
  indent = TRUE,
  exdent = TRUE,
  prefix = "",
  initial = prefix
)
```

Arguments

`...`	R objects which are concatenated.
`type`	Type of the output .Accepted values: • 'message' for a diagnostic message; • 'string' for a character output; • 'cat' for the output of `cat` function; • 'error' and 'warning' for an error or warning message. Intentionally, no default value is defined.
`sep`	(optional) character used to separate input values (default is nothing).
`date`	Logical value: set `TRUE` to place the date before the message and after the prefix (this is useful for logs or time consuming operations); default is FALSE.
`date_format`	Format of the date (see `strftime`) for the definition of the format). The default format is `'\%Y-\%m-\%d \%H:\%M:\%S'`.
`width`	Positive integer: target column for wrapping lines in the output (set to `Inf` for no wrapping).
`indent`	Non-negative integer: indentation of the first line in a paragraph It can be also a logical: in this case, if TRUE (default) the value is optimised in order to align first line with the followings.
`exdent`	Non-negative integer: indentation of subsequent lines in paragraphs. It can be also a logical: in this case, if TRUE (default) the value is optimised in order to align lines with the first line.
`prefix`	Character: prefix for each line except the first.
`initial`	Character: prefix for the first line.

Details

Several functions print messages in different formats (message, error, warning, cat, R output) and with different syntaxes (concatenating parameters or accepting a single argument, appending a new line, etc.). This accessory function provides a common interface for different types: several arguments are accepted and concatenated with the `sep` argument; the format is defined with the `format` argument; a date is optionally placed before the message.

Value

Message (in the defined format).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

projpar *Return a parameter used in a WKT projection*

Description

Return the value of a parameter (the name or the unit) present in the WKT of the given CRS.

Usage

```
projpar(x, par, abort = FALSE)

projname(x, abort = FALSE)
```

Arguments

x	The CRS to be named (any st_crs2 input is accepted).
par	Character corresponding to the parameter name (it can be one among "name" and "unit" - case insensitive).
abort	logical: if TRUE, the function aborts in case an invalid CRS is passed; if FALSE (default), the function returns NA, and a warning is shown.

Value

A character with the content of the parameter, and an attribute `crs` with the input projection checked using `sf::st_crs()`.

Note

The old function, which was searching for a generic parameter parsing the WKT, was deprecated: now `projpar()` only accepts `par = "name"` and `par = "unit"`, and `projname()` is an alias for `projpar(...,par = "name")`.

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

Examples

```
projpar(4326, "name")
projpar(4326, "unit")


projname(4326)
```

raster2rgb *Produce an RGB image from a singleband raster file.*

Description

Internal function to create JPEG or PNG images from a singleband raster file. This function is used by s2_thumbnails, and it will be exported when it would be more generalised.

Usage

```
raster2rgb(
  in_rast,
  out_file = NULL,
  palette = "generic_ndsi_2",
  minval = -1,
  maxval = 1,
  bigtiff = FALSE,
  tmpdir = NA
)
```

Arguments

in_rast	Path of the input multiband raster.
out_file	(optional) Path of the output RGB JPEG image; if NULL (default), a RasterLayer will be returned.
palette	Path of the palette file to be used (cpt or txt), or character value of a builtin palette ("SCL", "NDVI", the default "generic_ndsi" or "generic_ndsi_2").
minval	(to be implemented) the value corresponding to the minimum value of the palette (for now, only -1 is used). A quantile will be also accepted.
maxval	(to be implemented) the value corresponding to the maximum value of the palette (for now, only 1 is used). A quantile will be also accepted.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.
tmpdir	(optional) Path where intermediate files (VRT) will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.

Value

The path of the output image; alternatively, the output image as RasterLayer (if out_rast = NULL).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`raster_metadata` *Get metadata from raster paths*

Description

This accessory function extract some useful metadata from a vector of raster paths.

Usage

```
raster_metadata(raster_paths, meta = "all", format = "data.table")
```

Arguments

`raster_paths`	A vector of raster paths.
`meta`	Vector with the desired metadata: one or more values among 'res', 'size', 'bbox', 'proj', 'unit', 'outformat', 'type'. Alternatively meta = 'all' (default) allows to return all metadata.
`format`	One between `data.table` (default), `data.frame` and `list`.

Value

A data.table, data.frame or list of the output metadata.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Define product names
examplenames <- c(
  system.file("tif/L7_ETMs.tif", package="stars"),
  system.file("nc/bcsd_obs_1999.nc", package = "stars"),
  system.file("extdata/out/S2A2A_20190723_022_Barbellino_BOA_10.tif",
    package = "sen2r")
)


# Return metadata as data.table
raster_metadata(examplenames)
```

```
# Return some metadata as data.table
raster_metadata(examplenames, c("res", "size", "bbox", "outformat"))

# Return some metadata as list
raster_metadata(examplenames, c("res", "size", "bbox", "proj"), format = "list")

# Output with an invalid raster
examplenames <- c(
  examplenames,
  system.file("extdata/settings/gdal_formats.json", package="sen2r")
)
raster_metadata(examplenames, c("bbox", "proj"))
```

`read_gipp` *Manage GIPP parameters for Sen2Cor*

Description

`read_gipp()` reads Ground Image Processing Parameters (GIPP) from the default sen2r GIPP path or from an XML file.

`set_gipp()` modifies values of a list of GIPP in an XML file (or creates a new XML file with the desired GIPP).

Usage

```
read_gipp(gipp_names, gipp_path = NA)

set_gipp(gipp = list(), gipp_path = NA, use_dem = NA)
```

Arguments

`gipp_names`	Character vector with the names of the parameters to be read.
`gipp_path`	Character path of the GIPP XML file to be read (`read_gipp()`) or written (`set_gipp()`). In `read_gipp()`, if NA (default), the default sen2r GIPP path is read; in `set_gipp()`, setting this argument is mandatory (see details).
`gipp`	(optional) Ground Image Processing Parameters (GIPP) (see `sen2cor()` for further details). Elements whose name is missing in the XML file are skipped.
`use_dem`	Logical, determining if a DEM should be set for being used for topographic correction in the XML specified with argument `gipp_path` (see `sen2cor()` for further details).

Details

In `set_gipp()`, editing /resetting the default sen2r GIPP XML file was disabled to grant code reproducibility among different machines (an error is returned if `gipp_path` is not set). Users who want to do that (being aware of the risk doing that) must explicitly define the argument `gipp_path` as the path of the default GIPP file, which is `file.path(dirname(attr(load_binpaths(),"path")),"sen2r_L2A_GIPP.xml")`.

Value

`read_gipp()` returns a named list of GIPP with the required parameters (values not found in the XML are skipped).

`set_gipp()` returns NULL (the function is called for its side effects).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

Examples

```
# Read default values
read_gipp(c("dem_directory", "dem_reference"))
# Set the use of a topographic correction
set_gipp(use_dem = TRUE, gipp_path = gipp_temp <- tempfile())
# Read the parameters in the created temporary files
read_gipp(c("DEM_Directory", "DEM_Reference"), gipp_path = gipp_temp)
# Set not to use a topographic correction
set_gipp(use_dem = FALSE, gipp_path = gipp_temp <- tempfile())
# This is equivalent to:
# set_gipp(
#   list(DEM_Directory = NA, DEM_Reference = NA),
#   gipp_path = gipp_temp <- tempfile()
# )
# Read again the parameters in the created temporary files
read_gipp(c("DEM_Directory", "DEM_Reference"), gipp_path = gipp_temp)
```

---

read_scihub_login *Import / export / check SciHub username and password*

Description

Read the SciHub login information (`read_scihub_login()`), save new username and password (`write_scihub_login()`) or check their validity (`check_scihub_login()`). Login information is stored in a file `apihub.txt` inside the ".sen2r" subfolder of the home directory. This functions allow to read or write this file, and to edit them from inside the GUI.

Usage

```
read_scihub_login(apihub_path = NA)

check_scihub_login(username, password)

check_scihub_connection()

write_scihub_login(
  username,
  password,
  apihub_path = NA,
  check = TRUE,
  append = FALSE
)
```

Arguments

apihub_path	Path of the file in which login information is saved. If NA (default) it is automatically read from the package default location.
username	SciHub username.
password	SciHub password.
check	Logical: if TRUE (default), new credentials are checked before writing them on apihub_path (if they are invalid, an error is provided); if FALSE, they are directly written.
append	Logical: if TRUE, new credentials are added to the ones existing within apihub_path; if FALSE (default), apihub_path is replaced with the new ones.

Details

Notice that new/recently updated SciHub credentials are recognised by API Hub with a delay of about one week (see https://scihub.copernicus.eu/twiki/do/view/SciHubWebPortal/APIHubDescription for details).

For this reason, newly created credentials can not immediately be used by sen2r, and password edits on old credentials are not immediately recognised.

Value

read_scihub_login returns a matrix of credentials, in which username is in the first column, password in the second.

check_scihub_login returns TRUE if credentials are valid, FALSE elsewhere.

check_scihub_connection returns TRUE if internet connection is available and SciHub is accessible, FALSE otherwise.

write_scihub_login returns NULL.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Lorenzo Busetto, phD (2019) <lbusett@gmail.com>

Examples

```
check_scihub_login("user", "user")
write_scihub_login("user", "user")
read_scihub_login()
check_scihub_connection()
```

`s2_calcindices` *Compute maps of spectral indices*

Description

Create maps of a set of spectral indices. Since `gdal_calc.py` is used to perform computations, output files are physical rasters (no output VRT is allowed).

Usage

```
s2_calcindices(
  infiles,
  indices,
  outdir = ".",
  parameters = NULL,
  source = c("TOA", "BOA"),
  format = NA,
  subdirs = NA,
  tmpdir = NA,
  compress = "DEFLATE",
  bigtiff = FALSE,
  dataType = "Int16",
  scaleFactor = NA,
  proc_mode = "raster",
  parallel = FALSE,
  overwrite = FALSE,
  .log_message = NA,
  .log_output = NA
)
```

Arguments

`infiles`	A vector of input filenames. Input files are paths of BOA (or TOA) products already converted from SAFE format to a format managed by GDAL (use s2_translate to do it); their names must be in the sen2r naming convention (safe_shortname).

`indices` Character vector with the names of the required indices. Values should be included in names corresponding to the Abbreviations of the following indices: IDB.

`outdir` (optional) Full name of the output directory where the files should be created (default: current directory). `outdir` can bot be an existing or non-existing directory (in the second case, its parent directory must exists). If it is a relative path, it is expanded from the common parent directory of `infiles`.

`parameters` (optional) Values of index parameters. This variable must be a named list, in which each element is a list of parameters, i.e.: `parameters = list("SAVI" = list("a" = 0.5))` Values can be both numeric values or band names (e.g. "band_1"). If not specified, parameters are set to default values.

`source` (optional) Vector with the products from which computing the indices. It can be `"BOA"`, `"TOA"` or both (default). If both values are provided, indices are computed from the available products (`"TOA"` if TOA is available, BOA if BOA is available); in the case both are available, two files are produced (they can be distinguished from the level component - S2x1C or S2x2A - in the filename).

`format` (optional) Format of the output file (in a format recognised by GDAL). Default is the same format of input images (or "GTiff" in case of VRT input images).

`subdirs` (optional) Logical: if TRUE, different indices are placed in separated `outfile` subdirectories; if FALSE, they are placed in `outfile` directory; if NA (default), subdirectories are created only if more than a single spectral index is required.

`tmpdir` (optional) Path where intermediate files (GTiff) will be created in case `format` is `"VRT"`.

`compress` (optional) In the case a GTiff format is present, the compression indicated with this parameter is used.

`bigtiff` (optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.

`dataType` (optional) Numeric datatype of the output rasters. if "Float32" or "Float64" is chosen, numeric values are not rescaled; if `"Int16"` (default) or `"UInt16"`, values are multiplicated by `scaleFactor` argument; if `"Byte"`, values are shifted by 100, multiplicated by 100 and truncated at 200 (so that range -1 to 1 is coerced to 0-200), and nodata value is assigned to 255.

`scaleFactor` (optional) Scale factor for output values when an integer datatype is chosen (default values are 10000 for `"Int16"` and `"UInt16"`, 1E9 for `"Int32"` and `"UInt32"`). Notice that, using `"UInt16"` and `"UInt32"` types, negative values will be truncated to 0.

`proc_mode` (optional) Character: if `"gdal_calc"`, `gdal_calc` routines are used to compute indices; if `"raster"` (default) or `"stars"`, R functions are instead used (using respectively `raster` or `stars` routines). **Notes**:

1. there is a difference in which the two modes manage values out of ranges (e.g. -32768 to 32767 in Int16 and 0 to 255 in Byte): `"raster"` and `"stars"` modes set these values to NA, `"gdal_calc"` mode clip them to the minimum/maximum values;
2. `proc_mode = "stars"` is experimental (the performance of this mode must be optimised).

parallel	(optional) Logical: if TRUE, the function is run using parallel processing, to speed-up the computation for large rasters. The number of cores is automatically determined; specifying it is also possible (e.g. parallel = 4). If FALSE (default), single core processing is used. Multiprocess masking computation is always performed in singlecore mode
overwrite	Logical value: should existing output files be overwritten? (default: FALSE)
.log_message	(optional) Internal parameter (it is used when the function is called by sen2r()).
.log_output	(optional) Internal parameter (it is used when the function is called by sen2r()).

Value

A vector with the names of the created products.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

Examples

```
# Define file names
ex_in <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_BOA_10.tif",
  package = "sen2r"
)

# Run function
ex_out <- s2_calcindices(
  infiles = ex_in,
  indices = "EVI",
  outdir = tempdir(),
  dataType = "Float32"
)
ex_out

# Show output
oldpar <- par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(ex_in), rgb = 4:2, maxColorValue = 3500)
par(mar = rep(2/3,4)); image(stars::read_stars(ex_out))
par(oldpar)
```

s2_defNA *Set NA value of a specific product type*

Description

Internal function to determine the NA value to be used for each product type (except for spectral indices, whose NA value is managed by s2_calcindices).

Usage

```
s2_defNA(prod_types)
```

Arguments

prod_types	Character vector of the input product types

Value

Numeric NA values (NA not to set any NA value), corresponding to prod_types.

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
sen2r:::s2_defNA("BOA")
sen2r:::s2_defNA(c("BOA","BOA","SCL","TCI"))
```

s2_dop *Return the Dates Of Passage over some orbits*

Description

The function allows to know which Sentinel-2 passages should pass over certain orbits during a defined time interval. Dates are intended to be in UTC time. Notice that this is the expected calendar: some unexpected events (e.g. technical problems, or early working phases during first stages of acquisition) could cause the data unavailability even if an acquisition was expected. Notice also that some orbits (030, 073 and 116) acquire across UTC midnight: in this cases, the date is assumed to be the one of the acquisition after midnight (which corresponds to the date in local time).

Usage

```
s2_dop(s2_orbits = 1:143, timewindow = "10 days", mission = c("2A", "2B"))
```

Arguments

s2_orbits	A vector of Sentinel-2 orbits (as integer numbers or 3-length character). Default is all the 143 orbits.
timewindow	Temporal window for querying: Date object of length 1 (single day) or 2 (time window). Is it possible to pass also integer (or difftime) values, which are interpreted as the next n days (if positive) or the past n days (if negative). Also strings which can be interpreted as time ranges are accepted (see examples). Default is the next 10 days (one cycle).
mission	(optional) Vector with the desired Sentinel-2 missions ("2A", "2B" or both). Default is both.

Value

A data table with the dates (column "date"), the missions (column "mission") and the orbits (column "orbit"). An empty data table with the same structure is returned if no passages were found with the passed settings.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# All the passages in a cycle of 10 days over all the orbits
s2_dop()

# The passages in the current month over two orbits
s2_dop(c("022", "065"), "this month")

# The dates in which Sentinel-2A will pass in next six weeks over one orbit
s2_dop("022", "6 weeks", mission = "2A")$date

# The date in which Sentinel-2A would be passed in the last 10 days over one orbit
s2_dop("022", "-10 days", mission = "2A")$date

# All the orbits covered today
s2_dop(timewindow = Sys.Date(), mission = "2B")$orbit

# The passages in a fixed time window for one orbit
s2_dop(65, as.Date(c("2018-08-01", "2018-08-31")))

# A research with no passages found
s2_dop(22, "2018-08-16", mission = "2A")
```

s2_download *Download S2 products.*

Description

The function downloads S2 products. Input filenames must be elements obtained with s2_list function (each element must be a URL, and the name the product name).

Usage

```
s2_download(
  s2_prodlist = NULL,
  downloader = "builtin",
  apihub = NA,
  tile = NULL,
  outdir = ".",
  order_lta = TRUE,
  overwrite = FALSE
)
```

Arguments

s2_prodlist	Named character: list of the products to be downloaded, in the format safelist (see safelist). Alternatively, it can be the path of a JSON file exported by s2_order.
downloader	Executable to use to download products (default: "builtin"). Alternatives are "builtin" or "aria2" (this requires aria2c to be installed).
apihub	Path of the "apihub.txt" file containing credentials of SciHub account. If NA (default), the default location inside the package will be used.
tile	Deprecated argument
outdir	(optional) Full name of the existing output directory where the files should be created (default: current directory).
order_lta	Logical: if TRUE (default), products which are not available for direct download are ordered from the Long Term Archive; if FALSE, they are simply skipped.
overwrite	Logical value: should existing output archives be overwritten? (default: FALSE)

Value

Vector character with the list ot the output products (being downloaded or already existing).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

Lorenzo Busetto, phD (2019) <lbusett@gmail.com>

Examples

```
## Not run:
single_s2 <- paste0("https://scihub.copernicus.eu/apihub/odata/v1/",
  "Products(\'c7142722-42bf-4f93-b8c5-59fd1792c430\')/$value")
names(single_s2) <- "S2A_MSIL1C_20170613T101031_N0205_R022_T32TQQ_20170613T101608.SAFE"
# (this is equivalent to:
# single_s2 <- example_s2_list[1]
# where example_s2_list is the output of the example of the
# s2_list() function)

# Download the whole product
s2_download(single_s2, outdir=tempdir())

#' # Download the whole product - using aria2
s2_download(single_s2, outdir=tempdir(), downloader = "aria2")

# Download more products, ordering the ones stored in the Long Term Archive
pos <- sf::st_sfc(sf::st_point(c(-57.8815,-51.6954)), crs = 4326)
time_window <- as.Date(c("2018-02-21", "2018-03-20"))
list_safe <- s2_list(spatial_extent = pos, time_interval = time_window)
s2_download(list_safe, outdir=tempdir())

## End(Not run)
```

---

s2_gui *Launch the GUI for Sentinel-2 products*

Description

Launch the GUI to set parameters for the processing chain of Sentinel-2 products.

Usage

```
s2_gui(param_list = NULL, thunderforest_api = NA)
```

Arguments

param_list	List of parameters for initialising the GUI values (if empty, default values are used).
thunderforest_api	Character value with the API for thunderforest layers (now not used).

Value

A list of parameters.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`s2_list` *Retrieve list of available S2 products.*

Description

The function retrieves the list of available Sentinel-2 products satisfying given search criteria.

Usage

```
s2_list(
  spatial_extent = NULL,
  tile = NULL,
  orbit = NULL,
  time_interval = c(Sys.Date() - 10, Sys.Date()),
  time_period = "full",
  level = "auto",
  apihub = NA,
  max_cloud = 100,
  availability = "ignore",
  output_type = "deprecated"
)
```

Arguments

`spatial_extent`	A valid spatial object object of class `sf`, `sfc` or `sfg`
`tile`	string array Sentinel-2 Tiles to be considered string (5-length character)
`orbit`	string array Sentinel-2 orbit numbers to be considered
`time_interval`	Dates to be considered, as a temporal vector (class POSIXct or Date, or string in `YYYY-mm-dd` format) of length 1 (specific day) or 2 (time interval).
`time_period`	(optional) Character: • "full" (default) means that all the images included in the time window are considered;

- "seasonal" means that only the single seasonal periods in the window are used (i.e., with a time window from 2015-06-01 to 2017-08-31, the periods 2015-06-01 to 2015-08-31, 2016-06-01 to 2016-08-31 and 2017-06-01 to 2017-08-31 are considered).

`level` Character vector with one of the following: - "auto" (default): check if level-2A is available on SciHub: if so, list it; if not, list the corresponding level-1C product - "L1C": list available level-1C products - "L2A": list available level-2A products

`apihub` Path of the "apihub.txt" file containing credentials of SciHub account. If NA (default), the default location inside the package will be used.

`max_cloud` Integer number (0-100) containing the maximum cloud level of the tiles to be listed (default: no filter).

`availability` Character argument, determining which products have to be returned:

- `"online"` : only archive names already available for download are returned;
- `"lta"`: only archive names stored in the Long Term Archive are returned;
- `"check"`: all archive names are returned, checking if they are available or not for download (see "Value" to know how to distinguish each other);
- `"ignore"` (default): all archive names are returned, without doing the check (running the function is faster).

`output_type` Deprecated (use `as.data.table` to obtain a data.table).

Value

An object of class safelist. The attribute `online` contains logical values: in case `availability != "ignore"`, values are TRUE / FALSE for products available for download / stored in the Long Term Archive; otherwise, values are set to NA.

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2019) <lbusett@gmail.com> - Inspired by function `getSentinel_query` of package `getSpatialData` by J. Schwalb-Willmann

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
pos <- sf::st_sfc(sf::st_point(c(9.85,45.81)), crs = 4326)
time_window <- as.Date(c("2016-05-01", "2017-07-30"))

# Full-period list
example_s2_list <- s2_list(
  spatial_extent = pos,
```

```
  tile = "32TNR",
  time_interval = time_window,
  orbit = "065"
)
print(example_s2_list)
# Print the dates of the retrieved products
safe_getMetadata(example_s2_list, "sensing_datetime")

# Seasonal-period list
example_s2_list <- s2_list(
  spatial_extent = pos,
  tile = "32TNR",
  time_interval = time_window,
  time_period = "seasonal"
)
print(example_s2_list)
# Print the dates of the retrieved products
safe_getMetadata(example_s2_list, "sensing_datetime")
```

---

s2_mask *Apply cloud masks*

Description

s2_mask Applies a cloud mask to a Sentinel-2 product. Since raster functions are used to perform computations, output files are physical rasters (no output VRT is allowed).

s2_perc_masked computes the percentage of cloud-masked surface. The function is similar to s2_mask, but it returns percentages instead of masked rasters.

Usage

```
s2_mask(
  infiles,
  maskfiles,
  mask_type = "cloud_medium_proba",
  smooth = 0,
  buffer = 0,
  max_mask = 80,
  outdir = "./masked",
  tmpdir = NA,
  rmtmp = TRUE,
  save_binary_mask = FALSE,
  format = NA,
  subdirs = NA,
  compress = "DEFLATE",
  bigtiff = FALSE,
  parallel = FALSE,
```

```
  overwrite = FALSE,
  .log_message = NA,
  .log_output = NA
)

s2_perc_masked(
  infiles,
  maskfiles,
  mask_type = "cloud_medium_proba",
  tmpdir = NA,
  rmtmp = TRUE,
  parallel = FALSE
)
```

Arguments

infiles	A vector of input filenames. Input files are paths of products already converted from SAFE format to a format managed by GDAL (use s2_translate to do it); their names must be in the sen2r naming convention (safe_shortname).
maskfiles	A vector of filenames from which to take the information about cloud coverage (for now, only SCL products have been implemented). It is not necessary that `maskfiles` elements strictly match `infiles` ones. Input files are paths of products already converted from SAFE format to a format managed by GDAL (use s2_translate to do it); their names must be in the sen2r naming convention (safe_shortname).
mask_type	(optional) Character vector which determines the type of mask to be applied. Accepted values are: • `"nomask"`: do not mask any pixel; • `"nodata"`: mask pixels checked as "No data" or "Saturated or defective" in the SCL product (all pixels with values are maintained); • `"cloud_high_proba"`: mask pixels checked as "No data", "Saturated or defective" or "Cloud (high probability)" in the SCL product; • ‘"cloud_medium_proba": mask pixels checked as "No data", "Saturated or defective" or "Cloud (high or medium probability)" in the SCL product; • `"cloud_and_shadow"`: mask pixels checked as "No data", "Saturated or defective", "Cloud (high or medium probability)" or "Cloud shadow" in the SCL product; • `"clear_sky"`: mask pixels checked as "No data", "Saturated or defective", "Cloud (high or medium probability)", "Cloud shadow", "Unclassified" or "Thin cirrus" in the SCL product (only pixels classified as clear-sky surface - so "Dark area", "Vegetation", "Bare soil", "Water" or "Snow" - are maintained); • `"land"`: mask pixels checked as "No data", "Saturated or defective", "Cloud (high or medium probability)", "Cloud shadow", "Dark area", "Unclassified", "Thin cirrus", "Water" or "Snow" in the SCL product (only pixels classified as land surface - so "Vegetation" or "Bare soil" - are maintained);

- a string in the following form: "scl_n_m_n", where n, m and n are one or more SCL class numbers. E.g. string "scl_0_8_9_11" can be used to mask classes 0 ("No data"), 8-9 ("Cloud (high or medium probability)") and 11 ("Snow").

smooth (optional) Numerical (positive): the size (in the unit of inmask, typically metres) to be used as radius for the smoothing (the higher it is, the more smooth the output mask will result). Default is 0 (no smoothing is applied).

buffer (optional) Numerical (positive or negative): the size of the buffer (in the unit of inmask, typically metres) to be applied to the masked area after smoothing it (positive to enlarge, negative to reduce). Default is 0 (no buffer).

max_mask (optional) Numeric value (range 0 to 100), which represents the maximum percentage of allowed masked surface (by clouds or any other type of mask chosen with argument mask_type) for producing outputs. Images with a percentage of masked surface greater than max_mask% are not processed (the list of expected output files which have not been generated is returned as an attribute, named skipped). Default value is 80. Notice that the percentage is computed on non-NA values (if input images had previously been clipped and masked using a polygon, the percentage is computed on the surface included in the masking polygons).

outdir (optional) Full name of the output directory where the files should be created (default: "masked" subdir of current directory). outdir can bot be an existing or non-existing directory (in the second case, its parent directory must exists). If it is a relative path, it is expanded from the common parent directory of infiles.

tmpdir (optional) Path where intermediate files (VRT) will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.

rmtmp (optional) Logical: should temporary files be removed? (Default: TRUE). This parameter takes effect only if the output files are not VRT (in this case temporary files cannot be deleted, because rasters of source bands are included within them).

save_binary_mask (optional) Logical: should binary masks be exported? Binary mask are intermediate rasters used to apply the cloud mask: pixel values can be 1 (no cloud mask), 0 (cloud mask) or NA (original NA value, i.e. because input rasters had been clipped on the extent polygons). If FALSE (default) they are not exported; if TRUE, they are exported as MSK prod type (so saved within outdir, in a subdirectory called "MSK" if subdirs = TRUE). Notice that the presence of "MSK" products is not checked before running sen2r(), as done for the other products; this means that missing products which are not required to apply cloud masks will not be produced.

format (optional) Format of the output file (in a format recognised by GDAL). Default is the same format of input images (or "GTiff" in case of VRT input images).

subdirs (optional) Logical: if TRUE, different indices are placed in separated outfile subdirectories; if FALSE, they are placed in outfile directory; if NA (default), subdirectories are created only if more than a single product is required.

compress	(optional) In the case a GTiff format is present, the compression indicated with this parameter is used.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.
parallel	(optional) Logical: if TRUE, masking is conducted using parallel processing, to speed-up the computation for large rasters. The number of cores is automatically determined; specifying it is also possible (e.g. parallel = 4). If FALSE (default), single core processing is used. Multiprocess masking computation is always performed in singlecore mode if format != "VRT" (because in this case there is no gain in using multicore processing).
overwrite	(optional) Logical value: should existing output files be overwritten? (default: FALSE)
.log_message	(optional) Internal parameter (it is used when the function is called by sen2r()).
.log_output	(optional) Internal parameter (it is used when the function is called by sen2r()).

Value

s2_mask returns a vector with the names of the created products. An attribute "toomasked" contains the paths of the outputs which were not created cause to the high percentage of cloud coverage.

s2_perc_masked returns a names vector with the percentages of masked surfaces.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Define file names
ex_in <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_RGB432B_10.tif",
  package = "sen2r"
)
ex_mask <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_SCL_10.tif",
  package = "sen2r"
)

# Run function
ex_out <- s2_mask(
  infiles = ex_in,
  maskfiles = ex_mask,
  mask_type = "land",
  outdir = tempdir()
)
```

```
ex_out

# Show output
oldpar <- par(mfrow = c(1,3))
par(mar = rep(0,4)); image(stars::read_stars(ex_in), rgb = 1:3)
par(mar = rep(2/3,4)); image(stars::read_stars(ex_mask))
par(mar = rep(0,4)); image(stars::read_stars(ex_out), rgb = 1:3)
par(oldpar)
```

`s2_merge` *Merge S2 tiles with the same date and orbit*

Description

The function merge the input Sentinel-2 files with the same date, orbit number, product type and file format. Outputs are a set of products in the same format of corresponding input files.

Usage

```
s2_merge(
  infiles,
  outdir = ".",
  subdirs = NA,
  tmpdir = NA,
  rmtmp = TRUE,
  format = NA,
  compress = "DEFLATE",
  bigtiff = FALSE,
  vrt_rel_paths = NA,
  out_crs = NA,
  parallel = FALSE,
  overwrite = FALSE,
  .log_message = NA,
  .log_output = NA
)
```

Arguments

`infiles`	A vector of input filenames. Input files are paths of products already converted from SAFE format to a format managed by GDAL (use s2_translate to do it); their names must be in the sen2r naming convention (safe_shortname).
`outdir`	(optional) Full name of the output directory where the files should be created (default: current directory). `outdir` can bot be an existing or non-existing directory (in the second case, its parent directory must exists). If it is a relative path, it is expanded from the common parent directory of `infiles`.

subdirs	(optional) Logical: if TRUE, different output products are placed in separated outfile subdirectories; if FALSE, they are placed in outfile directory; if NA (default), subdirectories are created only if infiles relate to more than a single product.
tmpdir	(optional) Path where intermediate files (VRT) will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.
rmtmp	(optional) Logical: should temporary files be removed? (Default: TRUE). This parameter takes effect only if the output files are not VRT (in this case temporary files cannot be deleted, because rasters of source bands are included within them).
format	(optional) Format of the output file (in a format recognised by GDAL). Default is to maintain each input format.
compress	(optional) In the case a GTiff format is present, the compression indicated with this parameter is used.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.
vrt_rel_paths	(optional) Logical: if TRUE (default on Linux), the paths present in the VRT output file are relative to the VRT position; if FALSE (default on Windows), they are absolute. This takes effect only with format = "VRT".
out_crs	(optional) output CRS, in any format accepted by st_crs2 (default: the CRS of the first input file). The tiles with CRS different from out_crs will be reprojected (and a warning returned).
parallel	(optional) Logical: if TRUE, the function is run using parallel processing, to speed-up the computation for large rasters. The number of cores is automatically determined; specifying it is also possible (e.g. parallel = 4). If FALSE (default), single core processing is used.
overwrite	Logical value: should existing output files be overwritten? (default: FALSE)
.log_message	(optional) Internal parameter (it is used when the function is called by sen2r()).
.log_output	(optional) Internal parameter (it is used when the function is called by sen2r()).

Value

A vector with the names of the merged products (just created or already existing).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

s2_order *Order S2 products.*

Description

The function orders S2 products from Long Term Archive.

Usage

```
s2_order(
  s2_prodlist = NULL,
  export_prodlist = TRUE,
  delay = 0.5,
  apihub = NA,
  service = NA,
  reorder = TRUE
)
```

Arguments

s2_prodlist	Named character: list of the products to be ordered, in the format `safelist` (see safelist). Alternatively, it can be the path of a JSON file exported by a previous execution of s2_order, in case the user wants, for any reason, to resubmit the order.
export_prodlist	Logical or character: if TRUE (default), the list of ordered products is saved in a JSON text file, so to be easily retrievable at a later stage with safe_is_online or s2_download; if FALSE, no output files are generated. It is also possible to pass the path of an existing folder in which the JSON file will be saved (otherwise, a default path is used).
delay	Numeric: time frame (in seconds) to leave between two consecutive orders. Default is 0.5 seconds: use a higher value if you encountered errors (i.e. not all the products were correctly ordered).
apihub	Path of the "apihub.txt" file containing credentials of SciHub account. If NA (default), the default location inside the package will be used.
service	Character: it can be `"dhus"` or `"apihub"`, in which cases the required service is forced instead that the one present in the URLs passed through argument `s2_prodlist`. If NA (default), the service present in the URLs is maintained.
reorder	Logical: If TRUE, and a json file exported by s2_order is passed as argument to the function, try to order again also the `"available"` and `"ordered"` S2 datasets. Otherwise, only order the `"notordered"` ones.

Value

A named vector, containing the subset of `s2_prodlist` elements which were ordered. Moreover, the vector includes the following attributes:

- "available" with the elements of s2_prodlist which were already available for download,
- "notordered" with the elements of s2_prodlist which were not ordered for any reasons,
- "path" (only if argument export_prodlist is not FALSE) with the path of the json file in which the list of the products (ordered, available and not ordered) was saved (if export_prodlist = TRUE).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Lorenzo Busetto, phD (2020) <lbusett@gmail.com>

Examples

```
# Generate the lists of products
pos <- sf::st_sfc(sf::st_point(c(-57.8815,-51.6954)), crs = 4326)
time_window <- as.Date(c("2018-02-21", "2018-03-20"))
list_safe <- s2_list(spatial_extent = pos, time_interval = time_window)
print(list_safe)
# (at the time the documentation was written, this list was containing 5
# archives already available online and 2 stored in the Long Term Archive)

# Order the products
ordered_prods <- s2_order(list_safe)

# Check in a second time if the product was made available
(order_path <- attr(ordered_prods, "path"))
safe_is_online(order_path)
```

---

s2_rgb *Create RGB images from S2 reflectance products.*

Description

Function to create RGB images from Sentinel-2 reflectances.

Usage

```
s2_rgb(
  infiles,
  prod_type = NA,
  rgb_bands = 4:2,
  scaleRange = NA,
```

```
  outdir = NA,
  subdirs = NA,
  format = NA,
  compress = "DEFLATE",
  bigtiff = FALSE,
  tmpdir = NA,
  rmtmp = TRUE,
  parallel = TRUE,
  overwrite = FALSE,
  .log_message = NA,
  .log_output = NA
)
```

Arguments

infiles	A vector of input filenames. Input files are paths of products already converted from SAFE format to a format managed by GDAL (use s2_translate to do it); their names must be in the sen2r naming convention (safe_shortname).
prod_type	(optional) Output product (see safe_shortname for the list of accepted products). If not provided, it is retrieved from the file name.
rgb_bands	(optional) 3-length integer vector, which the number of the bands to be used respectively for red, green and blue. Default is 4:2 (true colours). It is also possible to pass a list of 3-length integer vectors in order to create multiple RGB types for each input file. Notice that this is the actual number name of the bands: so, to use i.e. BOA band 11 (1610nm) use the number 11, even if band 11 is the 10th band of a BOA product (because band 10 is missing).
scaleRange	(optional) Range of valid values. If can be a 2-length integer vector (min-max for all the 3 bands) or a 6-length vector or 3x2 matrix (min red, min green, min blue, max red, max green, max blue). Default is to use c(0,2500) for bands 1-5; c(0,7500) bands 6-12.
outdir	(optional) Full name of the existing output directory where the files should be created. Default is the same directory of input reflectance files.
subdirs	(optional) Logical: if TRUE, different indices are placed in separated outfile subdirectories; if FALSE, they are placed in outfile directory; if NA (default), subdirectories are created only if more than a single spectral index is required.
format	(optional) Format of the output file (in a format recognised by GDAL). Default is the same format of input images (or "GTiff" in case of VRT input images).
compress	(optional) In the case a GTiff format is present, the compression indicated with this parameter is used.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.
tmpdir	(optional) Path where intermediate files (VRT) will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.
rmtmp	(optional) Logical: should temporary files be removed? (Default: TRUE)

parallel	(optional) Logical: if TRUE, the function is run using parallel processing, to speed-up the computation for large rasters. The number of cores is automatically determined; specifying it is also possible (e.g. parallel = 4). If FALSE (default), single core processing is used.
overwrite	(optional) Logical value: should existing thumbnails be overwritten? (default: TRUE)
.log_message	(optional) Internal parameter (it is used when the function is called by sen2r()).
.log_output	(optional) Internal parameter (it is used when the function is called by sen2r()).

Value

A vector with the names of the created images.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Define file names
ex_in <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_BOA_10.tif",
  package = "sen2r"
)

# Run function
ex_out <- s2_rgb(
  infiles = ex_in,
  rgb_bands = list(c(11,8,4),c(9,5,4)),
  scaleRange = list(c(0,7500), matrix(c(rep(0,3),8500,6000,4000),ncol=2)),
  outdir = tempdir(),
  compress = 50
)
ex_out

# Show output
oldpar <- par(mfrow = c(1,3), mar = rep(0,4))
image(stars::read_stars(ex_in), rgb = 4:2, maxColorValue = 3500)
image(stars::read_stars(ex_out[1]), rgb = 1:3)
image(stars::read_stars(ex_out[2]), rgb = 1:3)
par(oldpar)
```

`s2_thumbnails` *Create thumbnails for S2 products.*

Description

Function to create thumbnail images for Sentinel-2 products. BOA and TOA multiband images are rendered as false colour JPEG images; SCL maps are rendered as 8-bit PNG; other singleband images (like spectral indices) are rendered as JPEG images with a standard colour palette. Output images are georeferenced.

Usage

```
s2_thumbnails(
  infiles,
  prod_type = NA,
  rgb_type = "SwirNirR",
  dim = 1024,
  scaleRange = NA,
  outdir = NA,
  tmpdir = NA,
  rmtmp = TRUE,
  overwrite = FALSE
)
```

Arguments

`infiles`	A vector of input filenames. Input files are paths of products already converted from SAFE format to a format managed by GDAL (use s2_translate to do it); their names must be in the sen2r naming convention (safe_shortname).
`prod_type`	(optional) Output product (see safe_shortname for the list of accepted products). If not provided, it is retrieved from the file name.
`rgb_type`	(optional) For BOA and TOA products, this value determine the type of false colours to be used for the thumbnails: • `"SwirNirR"` (default) for SWIR-NIR-Red; • `"NirRG"` for NIR-Red-Green; • `"RGB"` for true colours;
`dim`	Integer value, with the maximum greater dimension in pixels (width or height) of the output images (default: 1024 px). If this is lower than the corresponding dimension of the maps, maps are rescaled before producing the thumbnails; otherwise the original dimensions are maintained. To keep the original size in any case, set `dim = Inf`.
`scaleRange`	(optional) Range of valid values. If not specified (default), it is automatically retrieved from the product type. Default ranges for BOA and TOA products are 0 to 8000 (`rgb_type = "SwirNirR"`), 0 to 7500 (`"NirRG"`) and 0 to 2500 (`"RGB"`). For spectral indices, default range is -1 to 1 for Float products, -10000

	to 10000 for Int and 0 to 200 for Byte; for "Zscore" products, default range is -3 to 3 for Float and -3000 to 3000 for Int. It can be useful i.e. to stretch BOA "dark" products.
`outdir`	(optional) Full name of the existing output directory where the files should be created. Default is a subdirectory (named "thumbnails") of the parent directory of each input file.
`tmpdir`	(optional) Path where intermediate files (VRT) will be created. Default is a temporary directory. If `tmpdir` is a non-empty folder, a random subdirectory will be used.
`rmtmp`	(optional) Logical: should temporary files be removed? (Default: TRUE)
`overwrite`	(optional) Logical value: should existing thumbnails be overwritten? (default: TRUE)

Value

A vector with the names of the created images.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`s2_tiles` *Load Sentinel-2 tiles*

Description

Load the vector object of the Sentinel-2 tiles. When the function is run for the first time, it downloads the vector file from the sen2r GitHub repository and it saves it on disk.

Usage

```
s2_tiles()
```

Value

An sf spatial object containing the extent of the tiles.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Retrieve all the tiles
s2tiles <- s2_tiles()

# Extract a subset of all the tiles
httr::GET(
  "https://raw.githubusercontent.com/ranghetti/sen2r/devel/utils/vector/ch_bound.rds",
  httr::write_disk(ch_path <- tempfile())
)
ch <- readRDS(ch_path)
s2tiles_ch <- s2tiles[suppressMessages(sf::st_intersects(ch, s2tiles))[[1]],]
s2_coords <- sf::st_coordinates(suppressWarnings(sf::st_centroid(s2tiles_ch)))

# Show the tiles
plot(s2tiles_ch$geometry, border = "blue")
plot(ch, border = "red", add = TRUE)
text(s2_coords[,1], s2_coords[,2], s2tiles_ch$tile_id, col = "blue", cex = .75)

# Use function tiles_intersects() to exclude unuseful tiles.
```

---

s2_translate *Convert from SAFE format*

Description

The function build a virtual raster from a Sentinel2 SAFE product, eventually translating it in another spatial format. For now, only L1C and L2a with long name (< 2016/12/06) are recognised. Output vrt is at 10m resolution.

Usage

```
s2_translate(
  infile,
  outdir = ".",
  subdirs = NA,
  tmpdir = NA,
  rmtmp = TRUE,
  prod_type = NULL,
  tiles = NA,
  res = "10m",
  format = "VRT",
  compress = "DEFLATE",
  bigtiff = FALSE,
  vrt_rel_paths = NA,
  utmzone = "",
  overwrite = FALSE
)
```

Arguments

infile	Full path of the input SAFE folder (alternatively, full path of the xml file of the product with metadata).
outdir	(optional) Full name of the output directory where the files should be created (default: current directory). outdir can bot be an existing or non-existing directory (in the second case, its parent directory must exists). If it is a relative path, it is expanded from the directory of infile.
subdirs	(optional) Logical: if TRUE, different output products are placed in separated outdir subdirectories; if FALSE, they are placed in outdir directory; if NA (default), subdirectories are created only if prod_type has length > 1.
tmpdir	(optional) Path where intermediate files (VRT) will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.
rmtmp	(optional) Logical: should temporary files be removed? (Default: TRUE). This parameter takes effect only if the output files are not VRT (in this case temporary files cannot be deleted, because rasters of source bands are included within them).
prod_type	(optional) Vector of types to be produced as outputs (see safe_shortname for the list of accepted values). Default is reflectance ("TOA" for level 1C, "BOA" for level 2A).
tiles	(optional) Character vector with the desired output tile IDs (id specified IDs are not present in the input SAFE product, they are not produced). Default (NA) is to process all the found tiles.
res	(optional) Spatial resolution (one between '10m', '20m' or '60m'); default is '10m'. Notice that, choosing '10m' or '20m', bands with lower resolution will be rescaled to res. Band 08 is used with res = '10m', band 08A with res = '20m' and res = '60m'.
format	(optional) Format of the output file (in a format recognised by GDAL). Default value is "VRT" (Virtual Raster).
compress	(optional) In the case a GTiff format is chosen, the compression indicated with this parameter is used.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.
vrt_rel_paths	(optional) Logical: if TRUE (default on Linux), the paths present in the VRT output file are relative to the VRT position; if FALSE (default on Windows), they are absolute. This takes effect only with format = "VRT".
utmzone	(optional) UTM zone of output products (default: the first one retrieved from input granules), being a 3-length character (e.g. "32N"). Note that this function does not perform reprojections: if no granules refer to the specified UTM zone, no output is created.
overwrite	Logical value: should existing output files be overwritten? (default: FALSE)

Value

A vector with the names of the created output files (just created or already existing).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
## Not run:
s2_l1c_example <- file.path(
  "/existing/path",
  "S2A_MSIL1C_20170603T101031_N0205_R022_T32TQQ_20170603T101026.SAFE")
s2_l1c_example <- file.path(
  "/existing/path",
  "S2A_MSIL2A_20170603T101031_N0205_R022_T32TQQ_20170603T101026.SAFE")

# Create a single TOA GeoTIFF in the same directory
s2_translate(s2_l1c_example, format="GTiff")

# Create a single BOA VRT with a custom name
s2_translate(
  s2_l2a_example,
  "/new/path/example_sentinel2_sr.vrt",
  vrt_rel_paths = TRUE
)

# Create three products (ENVI) in the same directory at 60m resolution
s2_translate(
  s2_example,
  format = "ENVI",
  prod_type = c("BOA","TCI","SCL"),
  res = "60m",
  subdirs = TRUE
)

## End(Not run)
```

---

safelist-class *Format for SAFE archive lists*

Description

`safelist` is a format thought to manage lists of SAFE Sentinel.2 archives. It is a named character in which names are SAFE codes (e.g. `S2A_MSIL2A_20170507T102031_N0205_R065_T32TNR_20170507T102319.SAFE`), and values are URLs used to retrieve them from ESA API Hub (e.g. https://scihub.copernicus.eu/apihub/odata/v1/Products('a4 db7b-4ba8-9b09-53027ab0d7ab')/$value). Some attributes may be included, basically information retrieved by function s2_list containing product metadata. Moreover, the attribute `online` (retrieved

by function safe_is_online may contain logical values (TRUE for products available for download, FALSE for products stored in the Long Term Archive).

The class can be generated as an output of function s2_list, or converting named characters (with the same structures), data.frames or data.tables (including the columns `name` and `url`) using as (see examples). Objects of class `safelist` can be converted to named character, data.frames or data.tables (see examples). The conversion to data.frame / data.table is useful for reading hidden attributes.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
pos <- sf::st_sfc(sf::st_point(c(9.85,45.81)), crs = 4326)
time_window <- as.Date(c("2017-05-01", "2017-05-31"))

## Create an object of class safelist
list_safe <- s2_list(spatial_extent = pos, time_interval = time_window)
list_safe
class(list_safe)
attr(list_safe, "sensing_datetime") # extract an hidden attribute from a safelist

## Convert to other classes
(s2_char <- as.character(list_safe)) # convert to a simple named character
(s2_df <- as.data.frame(list_safe)) # convert to a data.frame
library(data.table)
(s2_dt <- as.data.table(list_safe)) # convert to a data.table
library(sf)
(s2_sf <- st_as_sf(list_safe)) # convert to sf

## Convert from other classes
as(s2_char, "safelist") # this causes the loss of hidden attributes
as(s2_df, "safelist") # this (and followings) maintain attributes as columns
as(s2_dt, "safelist")
as(s2_sf, "safelist")
```

`safe_getMetadata` *Get information from S2 file name or metadata*

Description

The function `safe_getMetadata()` scans a Sentinel2 product (main path or granule xml file) to retrieve information about the product.

The accessory function `rm_invalid_safe()` remove a SAFE archive in the case it is not recognised by `safe_getMetadata()`.

The accessory function `safe_isvalid()` scan the SAFE name to understand if it is a valid SAFE.

Usage

```
safe_getMetadata(
  s2,
  info = "all",
  format = "default",
  simplify = TRUE,
  abort = TRUE,
  allow_oldnames = FALSE
)

rm_invalid_safe(s2, req_res = c("10m", "20m", "60m"), allow_oldnames = FALSE)

safe_isvalid(
  s2,
  allow_oldnames = FALSE,
  check_file = TRUE,
  req_res = c("10m", "20m", "60m")
)
```

Arguments

`s2` Sentinel-2 products, which can be:

- a list of products in the format `safelist` (see safelist);
- a vector of SAFE paths;
- a vector of paths of xml product files with metadata. If the product does not exist locally, the function can run only with option `info = "nameinfo"` (see below).

`info` (optional) A character vector with the list of the metadata which should be provided. Accepted values are:

- `"all"` (default): all the retrievable metadata are provided;
- `"fileinfo"`: only the metadata obtained by scanning the file name and product structure (without opening it with GDAL) are provided.
- `"nameinfo"`: only the metadata obtained by scanning the file name are provided (it is faster and there is no need to have downloaded yet the file).
- a vector of single specific information (one or more from the followings):
  - `"name"` (SAFE name - this is always returned);
  - `"validname"` (TRUE or FALSE);
  - `"exists"` (TRUE or FALSE);

- "prod_type" ('singlegranule' or 'product');
- "version" ('old' or 'compact');
- "tiles" (vector with the tiles ID available in the product);
- "utm" (vector with the UTM zones used in the product);
- "xml_main" (name of the main XML file with metadata);
- "xml_granules" (names of the XML with granule metadata);
- "level" ('1C' or '2A');
- "creation_datetime", "id_tile", "mission", "centre", "file_class", "id_orbit", "orbit_number", "sensing_datetime", "id_baseline": metadata specific of the product type and version (they are returned only if obtainable for the specified input);
- "clouds", "direction", "orbit_n", "preview_url", "proc_baseline", "level", "sensing_datetime", "nodata_value", "saturated_value": information retrieved from the metadata stored in the XML file;
- "res": resolutions with all the output products available;
- "jp2list" (data.frame with the list of the JP2 band files - asking for this info will cause format to be coerced to "list").

Notice that the required info are returned only if available; i.e., if some info requiring existing files are asked by the user, but input SAFE do not exist, only info retrievable by the SAFE name are returned.

format — Output format, being one of the followings:

- "data.table" and "data.frame": a table with one row per s2 input and one column per required info;
- "list": a list (one element per s2 input) in which each element is a list of the required info;
- "vector": a list (one element per info) in which each element is a named vector (with s2 length and names) with the required info;
- "default" (default): "vector" if info is of length 1; "data.table" otherwise.

simplify — Logical parameter, which applies in case s2 is of length 1: in this case, if TRUE (default) and format is "list" or "vector", a single info list or vector is returned; if FALSE, a list of length 1 (containing the list or vector of the required s2 product) is returned.

abort — Logical parameter: if TRUE (default), the function aborts in case some inputs are not recognised, or if some files do not exists (in case some info elements require the files to be present); if FALSE, a warning is shown.

allow_oldnames — Logical parameter: if TRUE, old (long) name products are managed (metadata are returned, and they are considered valid; if FALSE (default), they are considered as non-supported files. Note that, from sen2r version 1.1.0, oldname products are no more supported within processing chains, so this function is deprecated and no more supported; moreover, it will be removed in next releases.

req_res — Character: vector of variable length (0 to 3) containing the names of the spatial resolution to be checked (one or more among "10m", "20m" and "60m"). In

case of level 2A-products, the existence of the JP2 files with the required resolutions necessary for sen2r processing chains (spectral bands and SCL) is checked, determining the result of the check. Default is c("10m","20m","60m"), since Sen2Cor by default produces all of these resolutions. NULL can be used not to scan for JP2 content. In case of level-1C products, in which each layer band is available in a specific resolution, any of the previous values causes all JP2 layers to be checked, while NULL causes no scan to be performed (as in the case of L2A). In safe_isvalid(), this argument is ignored if check_file = FALSE.

check_file Logical: if TRUE (default), the content of the provided paths is checked; if FALSE, only the validity of SAFE names is tested.

Value

safe_getMetadata() returns a data.table, a data.frame or a list (depending on argument format) with the output metadata;

rm_invalid_safe() returns a named vector (with the length of s2) with TRUE if the s2 product was removed, FALSE elsewhere.

safe_isvalid() returns a named vector (with the length of s2) with TRUE if the product is a valid SAFE, FALSE if not.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Define product name
s2_examplenames <- c(
  "S2A_MSIL1C_20190723T101031_N0208_R022_T32TNS_20190723T121220.SAFE",
  "S2A_MSIL1C_20190723T101031_N0208_R022_T32TNR_20190723T121220.SAFE"
)

# Return the information retrievable from the file names (files are not scanned)
safe_getMetadata(s2_examplenames, info="nameinfo")

# Return some specific information without scanning files
safe_getMetadata(s2_examplenames, info=c("level", "id_tile"))

# Return a single information without scanning files
# (in this case, the default output is a vector instead than a data.table)
safe_getMetadata(s2_examplenames, info="level")

# Check if the products are valid existing SAFE archives
safe_isvalid(s2_examplenames)

# Check if the product names are valid SAFE names
```

```
safe_isvalid(s2_examplenames, check_file = FALSE)
safe_isvalid("invalid_safe_name.SAFE", check_file = FALSE)

## Not run:
# Download a sample SAFE archive (this can take a while)
s2_exampleurl <- c(
  "S2A_MSIL1C_20190723T101031_N0208_R022_T32TNS_20190723T121220.SAFE" =
    paste0("https://scihub.copernicus.eu/apihub/odata/v1/",
           "Products('19bbde60-992b-423d-8dea-a5e0ac7715fc')/$value")
)
s2_download(s2_exampleurl, outdir=tempdir())
s2_examplepath <- file.path(tempdir(), names(s2_exampleurl))

# Return all the available information
safe_getMetadata(s2_examplepath)

# Return some specific information
safe_getMetadata(s2_examplepath, info=c("clouds", "direction"))

# Return a single information
safe_getMetadata(s2_examplepath, info="nodata_value")

# Check if the downloaded SAFE is valid
safe_isvalid(s2_examplepath)

# Delete it if it is not recognised
rm_invalid_safe(s2_examplepath)


## End(Not run)
```

`safe_is_online` *Check if SAFE is available for download*

Description

The function checks if the required SAFE archives are available for download, or if they have to be ordered from the Long Term Archive.

Usage

```
safe_is_online(s2_prodlist = NULL, apihub = NA, verbose = TRUE)
```

Arguments

`s2_prodlist`	Named character: list of the products to be checked, in the format `safelist` (see safelist). Alternatively, it can be the path of a JSON file exported by s2_order.
`apihub`	Path of the "apihub.txt" file containing credentials of SciHub account. If NA (default), the default location inside the package will be used.
`verbose`	Logical: if TRUE, provide processing messages summarising how many of the SAFE archives in `s2_prodlist` are available online.

Value

A logical vector of the same length and names of the SAFE products passed with `s2_prodlist`, in which each element is TRUE if the corresponding SAFE archive is available for download, FALSE if it is not or NA in case of errors with the SAFE url.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Lorenzo Busetto, phD (2020) <lbusett@gmail.com>

Examples

```
# Generate the lists of products
pos <- sf::st_sfc(sf::st_point(c(-57.8815,-51.6954)), crs = 4326)
time_window <- as.Date(c("2018-02-21", "2018-03-20"))
list_safe <- s2_list(spatial_extent = pos, time_interval = time_window)
# (at the time the documentation was written, this list was containing 5
# archives already available online and 2 stored in the Long Term Archive)

# Check for availability
safe_is_online(list_safe)
```

---

safe_shortname *Rename products using a shorten convention*

Description

This function renames a Sentinel-2 product in order to obtain shorten names. See the details for the structure of the adopted schema (named "sen2r naming convention"). The function applies only to compact product names (not to single granule names), since it is thought to be applied to entire products. Old long names are no more supported.

Usage

```
safe_shortname(
  prod_name,
  prod_type = NULL,
  ext = NULL,
  res = "10m",
  tiles = NULL,
  force_tiles = NULL,
  full.name = TRUE,
```

```
  set.seed = NULL,
  multiple_names = NULL,
  abort = FALSE
)
```

Arguments

prod_name	Input Sentinel-2 product name (it is not required that the file exists).
prod_type	(optional) Output product (default: TOA for L1C, BOA for L2A); see the details for the list of accepted products.
ext	(optional) Extension of the output filename (default: none).
res	(optional) Spatial resolution (one between '10m', '20m' or '60m'); default is '10m'. Notice that, choosing '10m' or '20m', bands with lower resolution will be rescaled to res. Band 08 is used with res = '10m', band 08A with res = '20m' and res = '60m'.
tiles	Deprecated (no more used).
force_tiles	Deprecated (no more used).
full.name	Logical value: if TRUE (default), all the input path is maintained (if existing); if FALSE, only basename is returned.
set.seed	Deprecated (no more used).
multiple_names	Deprecated (no more used).
abort	Logical parameter: if TRUE, the function aborts in case prod_type is not recognised; if FALSE (default), a warning is shown.

Details

ESA Sentinel-2 naming convention is particularly long-winded. So, the convention here adopted, named "sen2r naming convention", follows this schema:

S2mll_yyyymmdd_rrr_ttttt_ppp_rr.fff

where:

- S2mll (length: 5) shows the mission ID (S2A or S2B) and the product level (1C or 2A);
- yyyymmdd (length: 8) is the sensing date (e.g. 20170603 for 2017-06-03) ; the hour is skipped, since a single sensor can not pass two times in a day on the same tile);
- rrr (length: 3) is the relative orbit number (e.g. 022);
- ttttt (length: 5) is the tile number (e.g. 32TQQ);
- ppp (length: 3) is the output product, being one of these: *for level 1C:*
  - TOA: 13-bands Top-Of-Atmosphere Reflectance; *for level 2A:*
  - BOA: 13-bands Bottom-Of-Atmosphere Reflectance;
  - TCI: True Colour Image (3-band RGB 8-bit image);
  - AOT: Aerosol Optical Thickness;
  - WVP: Water Vapour;
  - SCL: Scene Classification Map;

  - CLD: Quality Indicators for cloud probabilities;
  - SNW: Quality Indicators for snow probabilities;
  - VIS: TODO Visibility (used for AOT);
- rr (length: 2) is the original minimum spatial resolution in metres (10, 20 or 60);
- fff (length: variable, generally 3) is the file extension.

Value

Output product name

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
safe_shortname("S2A_MSIL1C_20170603T101031_N0205_R022_T32TQQ_20170603T101026.SAFE", ext="tif")
```

---

sen2cor *Correct L1C products using Sen2Cor*

Description

The function uses Sen2Cor to manually correct L1C products. Standalone version of sen2cor (version 2.8.0 or 2.5.5) is used.

Usage

```
sen2cor(
  l1c_prodlist = NULL,
  l1c_dir = NULL,
  outdir = NULL,
  proc_dir = NA,
  tmpdir = NA,
  rmtmp = TRUE,
  gipp = NULL,
  use_dem = NA,
  tiles = NULL,
  parallel = FALSE,
  timeout = 0,
  overwrite = FALSE,
  .log_message = NA,
  .log_output = NA
)
```

Arguments

`l1c_prodlist`	List of L1C product names to be corrected. They can be both product names with full/relative path or only names of SAFE products (in this case, also l1c_dir argument must be provided). SAFE products must be unzipped. Note that, at this stage, all products must be in the same directory (this will be fixed).
`l1c_dir`	Full or relative path of input L1C products. If NULL (default), `l1c_prodlist` must already be a vector of full paths.
`outdir`	Directory where output L2A products will be placed. If NULL (default), each product is left in the parent directory of `l1c_prodlist`.
`proc_dir`	(optional) Directory where processing is applied. If NA (default), processing is done in `l1c_dir` and output L2A product is then moved to `outdir`, unless `l1c_dir` is a subdirectory of a SAMBA mountpoint under Linux: in this case, L1C input products are copied in a temporary directory (specified with argument `tmpdir`), processing is done there and then L2A is moved to `outdir`. This is required under Linux systems when `l1c_dir` is a subdirectory of a unit mounted with SAMBA, otherwise Sen2Cor would produce empty L2A products.
`tmpdir`	(optional) Path where processing is performed if a temporary working directory is required (see argument `proc_dir`). Be sure `tmpdir` not to be a SAMBA mountpoint under Linux. Default is a temporary directory. If `tmpdir` is a non-empty folder, a random subdirectory will be used.
`rmtmp`	(optional) Logical: should temporary files be removed? (Default: TRUE)
`gipp`	(optional) Ground Image Processing Parameters (GIPP) to be passed to Sen2Cor. It is possible to specify both the path of an existing XML file or a list of parameters in the form `parameter_name = "value"`, where `parameter_name` is the name of the parameter as specified in the XML file (case insensitive), and `"value"` is the character value which the user wants to set (notice that, in the case the user wants to specify the value `NONE`, both `"NONE"` and `NA` can be used, but not `NULL`, which has the effect to maintain the value specified in the XML file). For details about the GIPP parameters, refer to the Sen2Cor documentation (v. 2.5.5 or 2.8.0: see the "Schemas of the GIPP file" at the end of each page). *Note*: this argument takes effect only in the current execution of `sen2cor()` function.
`use_dem`	(optional) Logical: if TRUE, Sen2Cor is set to use a Digital Elevation Model for topographic correction (reflecting what is done for Level-2A SAFE images provided by ESA Hub); if FALSE, it is set not to perform topographic correction (reflecting the current default Sen2Cor behaviour); if NA (default), the option set in the XML GIPP configuration file used by sen2r (stored in the default sen2r settings directory) is respected; in case the user never edited it, the current default setting is not to perform topographic correction. *Notes*: 1. if TRUE, the path used to read or store DEM files and the online source used to download missing DEM tiles are respectively the `DEM_Directory` and `DEM_Reference` parameters set in the default sen2r GIPP XML file (the user can read them with the function `read_gipp(c("DEM_Directory","DEM_Reference"))`). In case one or both these parameters were set to `"NONE"`, a subdirectory

"srtm90" of the default sen2r directory is used as DEM directory, and/or the CGIAR SRTM 90m is set as online source. To set another directory or reference, use argument gipp in the form gipp = list(DEM_Directory = tempdir(),DEM_Reference ="another_reference",...) (replacing tempdir() with the desired path and specifying the online resource).

2. Currently the default value is NA in order to grant backward compatibility. In a future release of sen2r, the default value will be set to TRUE, so to grant homogeneity between Level-2A products downloaded from ESA Hub and generated using Sen2Cor.

tiles — Vector of Sentinel-2 Tile strings (5-length character) to be processed (default: process all the tiles found in the input L1C products).

parallel — (optional) Logical: if TRUE, Sen2Cor instances are launched in parallel using multiple cores; if FALSE (default), they are launched in series on a single core. The number of cores is automatically determined; specifying it is also possible (e.g. parallel = 4).

timeout — Integer value: number of minutes after which killing Sen2Cor if it is still running (default, 0, means that this is never done). This can be useful in case Sen2Cor produced an error without exiting from Python (leaving a standing process running).

overwrite — Logical value: should existing output L2A products be overwritten? (default: FALSE)

.log_message — (optional) Internal parameter (it is used when the function is called by sen2r()).

.log_output — (optional) Internal parameter (it is used when the function is called by sen2r()).

Value

Vector character with the list ot the output products (being corrected or already existing).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
## Not run:
# Download an L1C SAFE product
example_s2_list <- s2_list(
  spatial_extent = st_sfc(st_point(c(12.0, 44.8)), crs=st_crs(4326)),
  tile = "32TQQ",
  time_interval = as.Date(c("2017-05-01","2017-07-30"))
)
s2_download(example_s2_list, outdir = tempdir())

# Correct it applying a topographic correction
```

```
sen2cor(
  names(example_s2_list)[1],
  l1c_dir = tempdir(),
  outdir = tempdir(),
  use_dem = TRUE
)

## End(Not run)
```

---

sen2r *Find, download and preprocess Sentinel-2 images*

Description

The function is a wrapper to perform the entire processing chain to find, download and pre-process Sentinel-2 data. Input is a set of parameters that can be passed with a list or file (parameter `param_list`) or singularly (see the descriptions of all the other parameters).

Usage

```
sen2r(
  param_list = NULL,
  gui = NA,
  preprocess = TRUE,
  s2_levels = "l2a",
  sel_sensor = c("s2a", "s2b"),
  online = TRUE,
  order_lta = TRUE,
  apihub = NA,
  downloader = "builtin",
  overwrite_safe = FALSE,
  rm_safe = "no",
  step_atmcorr = "auto",
  sen2cor_use_dem = NA,
  sen2cor_gipp = NA,
  max_cloud_safe = 100,
  timewindow = NA,
  timeperiod = "full",
  extent = NA,
  extent_name = "sen2r",
  s2tiles_selected = NA,
  s2orbits_selected = NA,
  list_prods = NA,
  list_rgb = NA,
  list_indices = NA,
  index_source = "BOA",
  rgb_ranges = NA,
  mask_type = NA,
```

```
  max_mask = 100,
  mask_smooth = 0,
  mask_buffer = 0,
  clip_on_extent = TRUE,
  extent_as_mask = FALSE,
  reference_path = NA,
  res = NA,
  res_s2 = "10m",
  unit = "Meter",
  proj = NA,
  resampling = "near",
  resampling_scl = "near",
  outformat = "GTiff",
  rgb_outformat = "GTiff",
  index_datatype = "Int16",
  compression = "DEFLATE",
  rgb_compression = "90",
  overwrite = FALSE,
  path_l1c = NA,
  path_l2a = NA,
  path_tiles = NA,
  path_merged = NA,
  path_out = NA,
  path_rgb = NA,
  path_indices = NA,
  path_subdirs = TRUE,
  thumbnails = TRUE,
  parallel = FALSE,
  processing_order = "by_groups",
  use_python = NA,
  tmpdir = NA,
  rmtmp = TRUE,
  log = NA
)
```

Arguments

param_list	(optional) List of input parameters: it can be both an R list or the path of a JSON file. If some parameters are passed both as elements of param_list and as function arguments, the values passed as function arguments are considered. If some parameters are missing in param_list and are not provided as arguments, default values will be used. Use the function s2_gui() to create a complete list of parameters. If param_list is NULL (default), values given with the parameters below (or default values for parameters not provided) are used.
gui	(optional) Logical: if TRUE, function s2_gui() is launched before starting to process in order to set or load parameters; if FALSE, the function uses parameters passed with param_list or with other function arguments. Default is FALSE if param_list is not NULL, TRUE elsewhere.

`preprocess` (optional) Logical: TRUE (default) to perform also preprocessing steps, FALSE not to (do only find, download and atmospheric correction).

`s2_levels` (optional) Character vector of length 1 or 2, with Sentinel-2 levels required for processing steps or as output. This parameter is used only if `preprocess = FALSE` (otherwise, the required levels are derived from `list_prods`). Accepted values: "l1c" and "l2a"; default: "l2a".

`sel_sensor` (optional) Character vector of length 1 or 2, with Sentinel-2 sensors to be used. Accepted values: "s2a" and "s2b"; default: c("s2a","s2b").

`online` (optional) Logical: TRUE (default) to search for available products on SciHub (and download if needed); FALSE to work only with already downloaded SAFE products.

`order_lta` (optional) Logical: TRUE (default) to order products from the Long Term Archive if unavailable for direct download; FALSE to simply skip them (this option has effect only in online mode).

`apihub` Path of the text file containing credentials of SciHub account. If NA (default), the default location inside the package will be used.

`downloader` (optional) Character value corresponding to the executable which should be used to download SAFE products. It could be one among "builtin" (default) and "aria2". If aria2 is not installed, built-in method will be used instead.

`overwrite_safe` (optional) Logical: TRUE to overwrite existing products with products found online or manually corrected, FALSE (default) to skip download and atmospheric correction for products already existing.

`rm_safe` (optional) Character: should SAFE products be deleted after preprocessing? "yes" (or "all") means to delete all SAFE; "no" (default) not to delete; "l1c" to delete only Level-1C products.

`step_atmcorr` (optional) Character vector to determine how to obtain Level-2A SAFE products:

- `"auto"` (default) means that L2A is first searched on SciHub: if found, it is downloaded, if not, the corresponding Level-1C is downloaded and sen2cor is used to produce L2A;
- `"scihub"` means that Sen2Cor is always used from L1C products downloaded from SciHub;
- `"l2a"` means that they are downloaded if available on SciHub, otherwise they are skipped (sen2cor is never used).

`sen2cor_use_dem` (optional) Logical, determining if a DEM should be used for topographic correction by Sen2Cor (see the documentation of `sen2cor()` - argument `use_dem` for further details). Currently the default value is NA in order to grant backward compatibility: in this case, the option set in the XML GIPP configuration file used by sen2r (stored in the default sen2r settings directory) is respected.

Note: in a future release of sen2r, the default value will be set to TRUE, so to grant homogeneity between Level-2A products downloaded from ESA Hub and generated using Sen2Cor.

`sen2cor_gipp` (optional) Ground Image Processing Parameters (GIPP) to be passed to Sen2Cor (see the documentation of `sen2cor()` - argument `gipp` - for details about the

usage of this argument). Default value (NA) corresponds to an empty list of parameters.

`max_cloud_safe` (optional) Integer number (0-100) containing the maximum cloud level of each SAFE to be considered (default: no filter). It it used to limit the research of SAFE products to "good" images, so it is applied only to non-existing archives (existing SAFE are always used). In this sense, this parameter is different from `max_mask`, which can be used to set a maximum cloud coverage over output extents. Notice also that this value is used to filter on the basis of the metadata "Cloud cover percentage" associated to each SAFE, so it is not based on the cloud mask defined with the processing options.

`timewindow` (optional) Temporal window for querying: Date object of length 1 (single day) or 2 (time window). Default is NA, meaning that no filters are used if online = FALSE, and all found images are processed; if online = TRUE, last 90 days are processed. Is it possible to pass also integer (or difftime) values, which are interpreted as the last n days.

`timeperiod` (optional) Character:

- "full" (default) means that all the images included in the time window are considered;
- "seasonal" means that only the single seasonal periods in the window are used (i.e., with a time window from 2015-06-01 to 2017-08-31, the periods 2015-06-01 to 2015-08-31, 2016-06-01 to 2016-08-31 and 2017-06-01 to 2017-08-31 are considered).

`extent` (optional) Spatial extent on which to clip products (it can be both the path of a vector file or a geoJSON). Default is NA for offline mode (meaning no extent: all found tiles are entirely used); in online mode, a sample extent is used as default.

`extent_name` (optional) Name of the area set as extent, to be used in the output file names. Default is "sen2r" The name is an alphanumeric string which cannot contain points nor underscores, and that cannot be a five-length string with the same structure of a tile ID (two numeric and three uppercase character values).

`s2tiles_selected` (optional) Character vector with the Sentinel-2 tiles to be considered (default is NA, meaning all the tiles).

`s2orbits_selected` (optional) Character vector with the Sentinel-2 orbits to be considered (still to be implemented; for now, all the accepted values are listed).

`list_prods` (optional) Character vector with the values of the products to be processed (accepted values: "TOA", "BOA", "SCL", "TCI"). Default is no one (NA).

`list_rgb` (optional) Character vector with the values of the RGB images to be produced. Images are in the form RGBrgbx, where:

- x is B (if source is BOA) or T (is source is TOA);
- r g and b are the the number of the bands to be used respectively for red, green and blue, in hexadecimal format. Notice that this is the actual number name of the bands: so, to use i.e. BOA band 11 (1610nm) use the value "b", even if band 11 is the 10th band of a BOA product (because band 10 is missing). (e.g., RGB432B, RGB843B) Default is no one (NA).

`list_indices` (optional) Character vector with the values of the spectral indices to be computed. Default is no one (NA).

`index_source` (optional) Character value: if "BOA" (default), indices are computed from BOA values; if "TOA", non corrected reflectances are instead used (be careful to use this setting!).

`rgb_ranges` (optional) Range of valid values to be used for RGB products. Values must be provided in the same scale used within SAFE and BOA/TOA products (0-10000, corresponding to reflectances * 10000). If can be a 2-length integer vector (min-max for all the 3 bands) or a 6-length vector or 3x2 matrix (min red, min green, min blue, max red, max green, max blue). Default is to use c(0,2500) for bands 2, 3 and 4; c(0,7500) for other bands. In case `list_rgb` is a vector of length > 1, `rgb_ranges` must be a list of the same length (otherwise, the same range values will be used for all the RGB products).

`mask_type` (optional) Character value which determines the categories in the Surface Classification Map to be masked (see `s2_mask()` for the accepted values). Default (NA) is not to mask.

`max_mask` (optional) Numeric value (range 0 to 100), which represents the maximum percentage of allowed masked surface (by clouds or any other type of mask chosen with argument `mask_type`) for producing outputs. Images with a percentage of masked surface greater than `max_mask`% are not processed (the list of expected output files which have not been generated is returned as an attribute, named "skipped"). Default value is 80. This parameter is different from `max_cloud_safe`, because:

1. it is computed over the selected extent;
2. it is computed starting from the cloud mask defined as above. Notice that the percentage is computed on non-NA values (if input images had previously been clipped and masked using a polygon, the percentage is computed on the surface included in the masking polygons).

`mask_smooth` (optional) Numeric positive value: the smoothing radius (expressed in unit of measure of the output projection, typically metres) to be applied to the cloud mask by function s2_mask.

`mask_buffer` (optional) Numeric value: the buffering radius (expressed in unit of measure of the output projection, typically metres) to be applied to the cloud mask by function s2_mask. Default value (0) means that no buffer is applied; a positive value causes an enlargement of the masked area; a negative value cause a reduction.

`clip_on_extent` (optional) Logical: if TRUE (default), output products and indices are clipped to the selected extent (and resampled/reprojected); if FALSE, the geometry and extension of the tiles is maintained.

`extent_as_mask` (optional) Logical: if TRUE, pixel values outside the `extent` polygon are set to NA; if FALSE (default), all the values within the bounding box are maintained.

`reference_path` (optional) Path of the raster file to be used as a reference grid. If NA (default), no reference is used.

`res` (optional) Numeric vector of length 2 with the x-y resolution for output products. Default (NA) means that the resolution is kept as native.

`res_s2` (optional) Character value corresponding to the native Sentinel-2 resolution to be used. Accepted values are "10m" (default), "20m" and "60m".

`unit` (optional) Character value corresponding to the unit of measure with which to interpret the resolution (for now, only "Meter" - the default value - is supported).

`proj` (optional) Character string with the pro4string of the output resolution. default value (NA) means not to reproject.

`resampling` (optional) Resampling method (one of the values supported by `gdal_translate`: `"near"` (default), `"bilinear"`, `"cubic"`, `"cubicspline"`, `"lanczos"`, `"average"` or `"mode"`).

`resampling_scl` (optional) Resampling method for categorical products (for now, only SCL): one among `"near"` (default) and `"mode"`.

`outformat` (optional) Format of the output file (in a format recognised by GDAL). Default is `"GTiff"`. Value `"BigTIFF"` can be used to generate a GeoTIFF with the option BigTIFF

`rgb_outformat` (optional) Format of the output RGB products (in a format recognised by GDAL). Default is `"GTiff"`.

`index_datatype` (optional) Numeric datatype of the output spectral indices (see s2_calcindices.

`compression` (optional) In the case GTiff is chosen as output format, the compression indicated with this parameter is used (default is "DEFLATE").

`rgb_compression`

(optional) In the case `GTiff` is chosen as output format for RGB products, the compression indicated with this parameter is used (default is `"DEFLATE"`). In the cases GTiff or JPEG are chosen as output format for RGB products, this parameter can also be a 1-100 integer value, which is interpreted as the compression level for a JPEG compression.

`overwrite` (optional) Logical value: should existing output files be overwritten? (default: FALSE).

`path_l1c` (optional) Path of the directory in which Level-1C SAFE products are searched and/or downloaded. If not provided (default), a temporary directory is used.

`path_l2a` (optional) Path of the directory in which Level-2A SAFE products are searched, downloaded and/or generated. If not provided (default), a temporary directory is used.

`path_tiles` (optional) Path of the directory in which Sentinel-2 tiles (as generated by s2_translate) are searched and/or generated. If not provided (default), a temporary directory is used, and files are generated as virtual rasters; otherwise, they are generated in the format specified with `outformat` parameter.

`path_merged` (optional) Path of the directory in which Sentinel-2 tiles merged by orbit (as generated by s2_merge) are searched and/or generated. If not provided (default), a temporary directory is used, and files are generated as virtual rasters; otherwise, they are generated in the format specified with `outformat` parameter.

`path_out` (optional) Path of the directory in which Sentinel-2 output products are searched and/or generated. If not provided (default), a temporary directory is used.

`path_rgb` (optional) Path of the directory in RGB products are searched and/or generated. If not provided (default), `path_out` is used.

`path_indices` (optional) Path of the directory in which files of spectral indices are searched and/or generated. If not provided (default), `path_out` is used.

`path_subdirs` (optional) Logical: if TRUE (default), a directory for each output product or spectral index is generated within `path_tiles`, `path_merged`, `path_out` and `path_indices`; if FALSE, products are put directly within them.

`thumbnails` (optional) Logical: if TRUE (default), a thumbnail is added for each product created. Thumbnails are JPEG or PNG georeferenced small images (width or height of 1024 pixels) with default colour palettes (for more details, see the help window in the GUI). They are placed in a subdirectory of the products names `"thumbnails"`. If FALSE, they are not created.

`parallel` (optional) Logical or integer: setting to TRUE, the processing is executed using multiple cores in order to speed up the execution. Parallelisation is performed on groups of dates. The number of cores is automatically determined; specifying it is also possible (e.g. `parallel = 4`). If FALSE (default), the processing chain is forced to run with a single core (this can be useful if multiple sen2r instances are run in parallel).

`processing_order`

(optional) Character string: order used to execute the processing chain (this affects the speed of computation and the usage of system resources). Values can be one of the followings:

- `"4"` or `"by_groups"` (default): it provides a good compromise between processing speed and disk usage. Processing is done as follows:
  1. the list of required SAFE and output product names is computed;
  2. the required dates are grouped in g groups, where g is the number of dates divided by the number of CPU;
  3. groups are then processed sequentially; for each group:
     - the required SAFE archives are downloaded;
     - Sen2Cor is applied in parallel using one core per L1C SAFE archive;
     - the remaining processing operations are executed using parallel R sessions (one core for each date).
- `"2"` or `"by_date"`: this allows minimising the requirements of disk usage (in particular if SAFE archives are deleted after processing). It is similar to the default execution, but each group is composed by a single date: so the disk space occupied by SAFE archives and temporary files is lower, but it is generally slower than the default one because parallel computation over dates for products' generation is not possible.
- `"3"` or `"mixed"`: this allows maximising CPU usage and processing speed. The cycle on groups is ignored, and all the required SAFE are first of all downloaded and/or produced, and then dates are processed in parallel. This mode is faster than the default mode, but it requires all SAFE archives to be downloaded and processed before performing subsequent steps, thus increasing disk space requirements.
- `"1"` or `"by_step"`: this is the legacy mode, in which the cycle on groups is ignored as well as the parallel computation over dates. All SAFE archives are first downloaded/processed, then the processing steps are performed sequentially. This mode is similar to the previous one in terms of disk usage but it is slightly slower; its advantage are the lower RAM requirements.

use_python	Deprecated argument
tmpdir	(optional) Path where intermediate files will be created. Default is a temporary directory (unless outformat = "VRT": in this case, default is a subdirectory named ".vrt" within path_out).
rmtmp	(optional) Logical: should temporary files be removed? (Default: TRUE). rmtmp is forced to FALSE if outformat = "VRT".
log	(optional) Character string with the path where the package messages will be redirected. Default (NA) is not to redirect (use standard output). A two-length character with two paths (which can also coincide) can be used to redirect also the output: in this case, the first path is the path for messages, the second one for the output.

Value

A vector with the paths of the files which were created (excluded the temporary files); NULL otherwise. The vector includes some attributes:

- cloudcovered with the list of images not created due to the higher percentage of cloud covered pixels;
- missing with the list of images not created due to other reasons;
- procpath with the path of a json parameter file, created after each sen2r() run, containing the parameters used in the execution of the function;
- ltapath with the path of a json file containing the list of the SAFE Sentinel-2 archives eventually ordered in Long Term Archive.
- status with a data.frame summarising the status of the processing (see sen2r_process_report()).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2020) <luigi@ranghetti.info>

Lorenzo Busetto, phD (2020) <lbusett@gmail.com>

Examples

```
# Open an interactive section
if (interactive()) {
  sen2r()
}

# Launch a processing from a saved JSON file (here we use an internal function
# to create a testing json file - this is not intended to be used by final users)
json_path <- build_example_param_file()

out_paths_2 <- sen2r(json_path)
```

```
# Notice that passing the path of a JSON file results in launching
# a session without opening the gui, unless gui = TRUE is passed.

# Launch a processing using function arguments
safe_dir <- file.path(dirname(attr(load_binpaths(), "path")), "safe")
out_dir_3 <- tempfile(pattern = "Barbellino_")
out_paths_3 <- sen2r(
  gui = FALSE,
  step_atmcorr = "l2a",
  extent = system.file("extdata/vector/barbellino.geojson", package = "sen2r"),
  extent_name = "Barbellino",
  timewindow = as.Date("2017-07-03"),
  list_prods = c("TOA","BOA","SCL"),
  list_indices = c("NDVI","MSAVI2"),
  list_rgb = c("RGB432T", "RGB432B", "RGB843B"),
  mask_type = "cloud_medium_proba",
  max_mask = 80,
  path_l1c = safe_dir,
  path_l2a = safe_dir,
  path_out = out_dir_3
)

# Launch a processing based on a JSON file, but changing some parameters
# (e.g., the same processing on a different extent)
out_dir_4 <- tempfile(pattern = "Scalve_")
out_paths_4 <- sen2r(
  param_list = json_path,
  extent = system.file("extdata/vector/scalve.kml", package = "sen2r"),
  extent_name = "Scalve",
  path_out = out_dir_4
)


# Show outputs (loading thumbnails)

# Generate thumbnails names
thumb_3 <- file.path(dirname(out_paths_3), "thumbnails", gsub("tif$", "jpg", basename(out_paths_3)))
thumb_3[grep("SCL", thumb_3)] <-
  gsub("jpg$", "png", thumb_3[grep("SCL", thumb_3)])
thumb_4 <- file.path(dirname(out_paths_4), "thumbnails", gsub("tif$", "jpg", basename(out_paths_4)))
thumb_4[grep("SCL", thumb_4)] <-
  gsub("jpg$", "png", thumb_4[grep("SCL", thumb_4)])

oldpar <- par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(thumb_3[grep("BOA", thumb_3)]), rgb = 1:3)
image(stars::read_stars(thumb_3[grep("SCL", thumb_3)]), rgb = 1:3)

par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(thumb_3[grep("MSAVI2", thumb_3)]), rgb = 1:3)
image(stars::read_stars(thumb_3[grep("NDVI", thumb_3)]), rgb = 1:3)

par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(thumb_3[grep("RGB432B", thumb_3)]), rgb = 1:3)
```

```
image(stars::read_stars(thumb_3[grep("RGB843B", thumb_3)]), rgb = 1:3)

par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(thumb_4[grep("BOA", thumb_4)]), rgb = 1:3)
image(stars::read_stars(thumb_4[grep("SCL", thumb_4)]), rgb = 1:3)

par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(thumb_4[grep("MSAVI2", thumb_4)]), rgb = 1:3)
image(stars::read_stars(thumb_4[grep("NDVI", thumb_4)]), rgb = 1:3)

par(mfrow = c(1,2), mar = rep(0,4))
image(stars::read_stars(thumb_4[grep("RGB432B", thumb_4)]), rgb = 1:3)
image(stars::read_stars(thumb_4[grep("RGB843B", thumb_4)]), rgb = 1:3)

par(oldpar)
```

`sen2r_getElements` *Get information from S2 short name*

Description

This accessory function extracts metadata included in the name of a Sentinel-2 product which follows the sen2r naming convention (see safe_shortname).

Usage

```
sen2r_getElements(s2_names, format = "data.table", abort = TRUE)
```

Arguments

`s2_names`	A vector of Sentinel-2 product names in the sen2r naming convention.
`format`	One between `data.table` (default), `data.frame` and `list`.
`abort`	Logical parameter: if TRUE (default), the function aborts in case any of `s2_names` is not recognised; if FALSE, a warning is shown, and a list with only the element "type"='unrecognised' is returned.

Value

A data.table, data.frame or list of the output metadata.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
# Define product name
fs2nc_examplename <-
  "/path/of/the/product/S2A1C_20170603_022_32TQQ_TOA_20.tif"

# Return metadata
sen2r_getElements(fs2nc_examplename)
```

`sen2r_process_report` *Summarise processing report*

Description

Internal function used to summarise results of sen2r execution at the end of a processing..

Usage

```
sen2r_process_report(
  s2_list_ordered,
  s2names = NULL,
  pm = NULL,
  cloudlist_path = character(0),
  ignorelist_path = character(0),
  s2_list_cloudcovered = NA,
  s2_list_failed = NA,
  s2_list_cloud_ignored = NA,
  s2_list_failed_ignored = NA,
  download_only = FALSE,
  s2_downloaded = NA,
  s2_skipped = NA,
  s2_corrected = NA
)
```

Arguments

`s2_list_ordered`	List containing the lists of ordered/notordered LTA S2 images.
`s2names`	Output of `compute_s2_paths()`.
`pm`	List containing sen2r processing parameters.
`cloudlist_path`	Internal parameter.
`ignorelist_path`	Internal parameter.
`s2_list_cloudcovered`	Internal parameter.
`s2_list_failed`	Internal parameter.

`s2_list_cloud_ignored`
: Internal parameter.

`s2_list_failed_ignored`
: Internal parameter.

`download_only`
: Logical: if TRUE, it indicates that the processing to be summarised only involved download (`pm$preprocess = FALSE`).

`s2_downloaded`
: Internal parameter.

`s2_skipped`
: Internal parameter.

`s2_corrected`
: Internal parameter.

Value

A data.frame summarising the report, and containing the following columns:

- `time`: date/time of report creation;
- `n_req_tot_dates`: number of dates to be processed based on the query;
- `n_ondisk_dates`: number of dates for which all expected products are already on disk;
- `n_proc_dates`: number of date for which some products were computed in the current run;
- `n_complete_out`: number of dates for which processing is "complete", and all products were created;
- `n_failed_dates`: number of dates for which processing is "complete", but some products were not created for unexpected reasons;
- `n_cloudy_dates`: number of dates for which processing is "complete", but some products were not created because cloudiness in the spatial extent was above `max_mask`;
- `n_notonline_dates`: number of date for which processing is "incomplete", because not all require images are online;
- `n_ordered_imgs`: number of images correctly ordered from LTA;
- `n_notordered_imgs`: number of images for which LTA order failed;
- `n_downloaded`: number of images downloaded during current run;
- `n_skipped`: number of required images not downloaded because already on disk (note: this does not include images that would be needed to process a date for which all products are already on disk);
- `n_corrected`: number of images atmospherically corrected using sen2cor;
- `completed`: logical, indicating if processing can be considered "complete" (it is set to TRUE in case `n_notonline_dates = 0`).

Note

License: GPL 3.0

Author(s)

Lorenzo Busetto, phD (2020) <lbusett@gmail.com>

smooth_mask *Buffer cloud masks*

Description

Internal function (used by s2_mask) which smooths and buffers a 0-1 mask image in order to reduce the roughness of the mask obtained from SCL classification (which is done pixel by pixel). See details.

Usage

```
smooth_mask(
  inmask,
  binpaths,
  tmpdir = tempdir(),
  radius = 250,
  buffer = 250,
  namask = NULL,
  bigtiff = FALSE
)
```

Arguments

inmask	The path of the input 0-1 mask (where 0 represents the area to be masked, 1 the clean surface).
binpaths	list of paths of binaries.
tmpdir	(optional) Path where intermediate files (VRT) will be created. Default is a temporary directory.
radius	(optional) Numerical (positive): the size (in the unit of inmask, typically metres) to be used as radius for the smoothing (the higher it is, the more smooth the output mask will result).
buffer	(optional) Numerical (positive or negative): the size of the buffer (in the unit of inmask, typically metres) to be applied to the masked area after smoothing it (positive to enlarge, negative to reduce).
namask	(optional) The path of an input 0-1 mask where 0 represents the area of the original file with NA values (which should not be smoothed / buffered). Default (NULL) means that no NA values are present.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE).

Value

The path of the smoothed mask.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

`stack2rgb` *Produce an RGB image from a multiband raster file.*

Description

Internal function to create JPEG images from a multiband raster file. This function is used by s2_thumbnails, and it will be exported when it would be more generalised.

Usage

```
stack2rgb(
  in_rast,
  out_file = NULL,
  bands = 1:3,
  minval = 0,
  maxval = 10000,
  format = "JPEG",
  compress = "90",
  bigtiff = FALSE,
  tmpdir = NA
)
```

Arguments

`in_rast`	Path of the input multiband raster.
`out_file`	(optional) Path of the output RGB JPEG image; if NULL (default), a RasterBrick will be returned.
`bands`	(optional) 3-length integer argument, with the position of the three bands to be used respectively for red, green and blue.
`minval`	(optional) the value corresponding to black (default: 0). Also a 3-length vector is accepted (min values for red, green and blue respectively).
`maxval`	(optional) the value corresponding to white (default: 10000). Also a 3-length vector is accepted (max values for red, green and blue respectively).
`format`	(optional) Format of the output file (in a format recognised by GDAL). Default is JPEG.

compress	(optional) In the case a GTiff format is present, the compression indicated with this parameter is used. In the case a JPEG format is present, the compression indicates the quality (integer, 0-100). In the case a GTiff format is present and an integer 0-100 number is provided, this is interpreted as the quality level of a JPEG compression.
bigtiff	(optional) Logical: if TRUE, the creation of a BigTIFF is forced (default is FALSE). This option is used only in the case a GTiff format was chosen.
tmpdir	(optional) Path where intermediate files will be created. Default is a temporary directory. If tmpdir is a non-empty folder, a random subdirectory will be used.

Value

The path of the output image; alternatively, the output image as RasterBrick (if out_rast = NULL).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

---

start_trace *Trace functions which create files*

Description

Some functions create output files during their execution; some of them are slow, so the probability that something goes wrong during their execution is not null, with the consequence that output files are incomplete, or that undesired temporary files are left on the filesystem.

This functions is though to manage these situations.

trace_function runs a function and checks if errors occur during its execution; in this case, the files created by it are deleted (only if the timestamp of the files is subsequent to the time of execution of the function). If the code interrupts before the function ends, the paths of the files intended to be created by the function are saved within the package, so that they can be easily deleted in a second time; to do it, simply run clean_traces.

Other intermediate functions are used internally: start_trace saves the paths of the files intended to be created within a text file; end_trace deletes this text file (it is used by trace_function when a function stops without errors); clean_trace deletes this text file and the intended output files (it is used by trace_function when a function stops with errors).

Usage

```
start_trace(trace_files, trace_funname)

end_trace(tracename)

clean_trace(tracename)

trace_function(trace_fun, trace_files, trace_funname = NA, ...)

clean_traces(trace_funname = NA)
```

Arguments

trace_files	Vector of the files intended to be created by fun (for now, providing it is mandatory). Also temporary files can be indicated here.
trace_funname	The name of the function to be run (in start_trace) or to be cleaned (in clean_traces; if NA, all the traces are cleaned).
tracename	The path of the text file containing the log information generated by start_trace.
trace_fun	The function to be run.
...	Arguments of the function fun

Value

NULL (the function is called for its side effects)

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

---

str_pad2 *Pad a string.*

Description

Vectorised over string, width and pad. This is an internal function doing the same thing of stringr::str_pad (except for parameters 'width' and 'length' which must be of length 1), but without depending on package stringi.

Usage

```
str_pad2(string, width, side = c("left", "right", "both"), pad = " ")
```

Arguments

string	A character vector.
width	Minimum width of padded strings.
side	Side on which padding character is added (left, right or both).
pad	Single padding character (default is a space).

Value

A character vector.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
rbind(
  str_pad2("hadley", 30, "left"),
  str_pad2("hadley", 30, "right"),
  str_pad2("hadley", 30, "both")
)

# All arguments are vectorised except side
str_pad2(c("a", "abc", "abcdef"), 10)

# Longer strings are returned unchanged
str_pad2("hadley", 3)
```

---

st_as_text_2 *Return WKT or WKT2 basing on the installed rgdal version*

Description

This is a convenience temporary function which returns the WKT representation of a CRS, using sf::st_as_text in case PROJ < 3, rgdal::CRS otherwise. This has the advantage to perform precise transformations with PROJ >=3, and to avoid conversion errors (see here). This function will be deleted whenever sf will manage WKT2.

Usage

```
st_as_text_2(x, pretty = FALSE)
```

Arguments

x	object of class sfg, sfc or crs
pretty	logical; if TRUE, print human-readable well-known-text representation of a coordinate reference system

Value

Well-known Text representation of simple feature geometry or coordinate reference system

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
sen2r:::st_as_text_2(sf::st_crs(32632))
```

---

st_crs2 *Retrieve coordinate reference system from sf or sfc object*

Description

This function is a wrapper for sf::st_crs, unless threating numeric character strings as integers, and accepting also UTM timezones, paths of spatial files and paths of text files containing WKT like .prj (see details) .

Usage

```
st_crs2(x, ...)
```

Arguments

x	numeric, character, or object of class sf or sfc, being: • EPSG code: numeric (e.g. 32632) or character (in the form "32632" or "EPSG:32632"); • UTM zone: numeric (e.g. 32, interpreted as 32 North) or character (e.g. "32" or "32N" for zone 32 North, "32S" for 32 South); • WKT test: passed as character string or as path of a text file containing it (e.g. the path of a .prj file); • PROJ.4 string, passed as character (e.g. "+proj=utm +zone=32 +datum=WGS84 +units=m +no_defs" (**NOTE**: this representation is deprecated with PROJ >= 6 – see http://rgdal.r-forge.r-project.org/articles/PROJ6_GDAL3.html – so a warning is returned using it, unless the string contains only the epsg code – e.g. "+init=epsg:32632", in which case the EPSG code is taken); • path of a spatial file (managed by sf::st_read or stars::read_stars), passed as character string of length 1; • spatial file of class sf or sfc.
...	other parameters passed to sf::st_crs.

Details

See sf::st_crs for details.

Value

An object of class crs of length 2.

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
## CRS from EPSG
st_crs2(32609)
st_crs2("EPSG:32609")

## CRS from UTM zone
st_crs2(9)
st_crs2("09")
st_crs2("9N")
st_crs2("09S")

## CRS from WKT (string or path)
(wkt_32n <- sf::st_as_text(sf::st_crs(32609)))
st_crs2(wkt_32n)
writeLines(wkt_32n, wkt_32n_path <- tempfile())
st_crs2(wkt_32n_path)

## CRS from spatial file path
raster_path <- system.file(
  "extdata/out/S2A2A_20190723_022_Barbellino_BOA_10.tif",
  package="sen2r"
)
vector_path <- system.file(
  "extdata/vector/barbellino.geojson",
  package="sen2r"
)
st_crs2(raster_path)
st_crs2(vector_path)

## CRS from spatial files
st_crs2(stars::read_stars(raster_path))
st_crs2(raster::raster(raster_path))
st_crs2(sf::read_sf(vector_path))


## CRS from PROJ.4 string
```

```
# (avoid using this with PROJ >= 6!)
st_crs2("+init=epsg:32609") # this makes use of the EPSG code
st_crs2("+proj=utm +zone=9 +datum=WGS84 +units=m +no_defs")
```

`suppress_warnings` *Selective suppress warnings*

Description

Suppress warnings matching particular regular expressions.

Usage

```
suppress_warnings(.expr, .f)
```

Arguments

`.expr`	Code to evaluate
`.f`	A regular expression (which will be passed to `grepl()`).

Details

See https://stackoverflow.com/questions/16517795/selective-suppresswarnings-that-filters-by-regular-expression

Value

The warning message as character string, invisibly.

`tiles_intersects` *Select the tiles intersecting the extent*

Description

Function which returns the tile IDs of the Sentinel-2 tiles which overlap a provided extent.

Usage

```
tiles_intersects(extent, all = FALSE, out_format = "id", .s2tiles = NULL)
```

Arguments

extent	sf object with the spatial extent.
all	logical: if TRUE, all the tiles overlapping the extent are provided; if FALSE (default), unnecessary tiles are skipped. Unnecessary tiles are tiles which overlaps the extent for an area already covered by another tile. In case the extent is all included in an overlapping area, only one of the two candidate tiles is returned (the first in alphabetical order).
out_format	character: if "sf", the spatial object of the overlapping tiles is returned; if "id" (default), a character vector with the tile IDs.
.s2tiles	output of s2_tiles() function (it is possible to pass it in order to speed up the execution; otherwise leave to NULL and it will be generated within the function).

Value

the tiles intersecting the extent (see argument out_format).

Note

License: GPL 3.0

Author(s)

Luigi Ranghetti, phD (2019) <luigi@ranghetti.info>

Examples

```
ex_extent <- sf::st_read(
  system.file("extdata/vector/scalve.kml", package = "sen2r"),
  quiet = TRUE
)
ex_extent <- ex_extent[1,]

# Tile ID of the required S2 tile
tiles_intersects(ex_extent)

# Tile ID of all the overlapping S2 tiles
tiles_intersects(ex_extent, all = TRUE)

# Spatial object with the required tile
sel_tiles <- tiles_intersects(ex_extent, out_format = "sf")
plot(sf::st_geometry(sel_tiles)); plot(sf::st_geometry(ex_extent), add=TRUE, col="yellow")

# Spatial object with the overlapping S2 tiles
sel_tiles <- tiles_intersects(ex_extent, all = TRUE, out_format = "sf")
plot(sf::st_geometry(sel_tiles)); plot(sf::st_geometry(ex_extent), add=TRUE, col="yellow")
```

Index