Package 'gdalUtils'

October 10, 2015

Maintainer Jonathan Asher Greenberg <gdalUtils@estarcion.net>

License GPL (>= 2)

Title Wrappers for the Geospatial Data Abstraction Library (GDAL) Utilities

Type Package

LazyLoad yes

Author Jonathan Asher Greenberg and Matteo Mattiuzzi

Description Wrappers for the Geospatial Data Abstraction Library (GDAL) Utilities.

Version 2.0.1.7

Date 2015-10-08

Depends R (>= 2.14.0)

Imports sp, foreach, R.utils, raster, rgdal

SystemRequirements GDAL binaries

Repository CRAN

Repository/R-Forge/Project gdalutils

Repository/R-Forge/Revision 133

Repository/R-Forge/DateTimeStamp 2015-10-09 02:05:20

Date/Publication 2015-10-10 22:50:40

NeedsCompilation no

R topics documented:

`align_rasters` *Aligns raster files*

Description

Aligns a raster to a reference raster.

Usage

```
align_rasters(unaligned, reference, dstfile, output_Raster = FALSE,
  nThreads = 1, verbose = FALSE, ...)
```

Arguments

`unaligned`	Character. The filename of a raster to be aligned to the reference raster.
`reference`	Character. The filename of a raster to be used as the reference for the alignment. Syncing will use the reference's projection, resolution, and extent.
`dstfile`	Character. The filename of the synchronized output file.
`output_Raster`	Logical. Return output dst_dataset as a RasterBrick?

nThreads	Numeric or Character. If numeric, the number of threads to use. Setting to > 1 enables multithreaded execution. Can also be "ALL_CPUS" to use all available CPUS. Default is 1.
verbose	Logical. Enable verbose execution? Default is FALSE.
...	parameters to be passed to gdalwarp (e.g. resampling approach).

Details

Aligns a raster to the extent and projection of a reference raster and matches the resolution of the reference raster. This is helpful in preparing multiple files of different projections, resolutions, extents, and rotations for performing map algebra or change detection.

Value

Either NULL or a RasterBricks depending on whether output_Raster is set to TRUE.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>)

See Also

gdalwarp

---

batch_gdal_translate *batch_gdal_translate*

Description

Runs gdal_translate on a batch of files

Usage

```
batch_gdal_translate(infiles, outdir, outsuffix = "_conv.tif",
  pattern = NULL, recursive = FALSE, verbose = FALSE, ...)
```

Arguments

infiles	Character. A directory or a character vector of files (including their path). If a directory, all files matching the pattern will be converted.
outdir	Character. Output directory to save the output files.
outsuffix	Character. The suffix to append to the input filename (minus its extension) to generate the output filename(s).
pattern	Character. If infiles is a directory, this is used to limit the file it is searching for.
recursive	Logical. If infiles is a directory, should files be searched for recursively?
verbose	Logical. Enable verbose execution? Default is FALSE.
...	Parameters to pass to gdal_translate

Details

This function is designed to run gdal_translate in batch mode. Files are passed to the function either directly as a character vector of filenames, or by passing it a directory and (typically) a search pattern (e.g. pattern=".tif"). gdal_translate will execute based on parameters passed to it, and the output file will be named based on the input file (stripped of its extension), with the outsuffix appended to it.

If a parallel engine is started and registered with foreach, this program will run in parallel (one gdal_translate per worker).

Value

Either a list of NULLs or a list of RasterBricks depending on whether output_Raster is set to TRUE.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>)

References

http://www.gdal.org/gdal_translate.html

See Also

gdal_translate, list.files

Examples

```
## Not run:
input_folder <- system.file("external",package="gdalUtils")
list.files(input_folder,pattern=".tif")
output_folder <- tempdir()
# library(spatial.tools)
# sfQuickInit() # from package spatial.tools to launch a parallel PSOCK cluster
batch_gdal_translate(infiles=input_folder,outdir=output_folder,
outsuffix="_converted.envi",of="ENVI",pattern=".tif$")
list.files(output_folder,pattern="_converted.envi$")
# sfQuickStop() # from package spatial.tools to stop a parallel PSOCK cluster

## End(Not run)
```

---

gdaladdo *gdaladdo*

Description

R wrapper for gdaladdo: builds or rebuilds overview images

Usage

```
gdaladdo(filename, levels, r, b, ro, clean, oo, ignore.full_scan = TRUE,
  verbose = FALSE)
```

Arguments

filename	Character. The file to build overviews for (or whose overviews must be removed).
levels	Numeric. A list of integral overview levels to build. Ignored with clean=TRUE option.
r	Character. ("nearest"|"average"|"gauss"|"cubic"|"average_mp"|"average_magphase"|"mode") Select a resampling algorithm. Default is "nearest".
b	Numeric. (available from GDAL 1.10) Select an input band band for overview generation. Band numbering starts from 1. Multiple -b switches may be used to select a set of input bands to generate overviews.
ro	Logical. (available from GDAL 1.6.0) open the dataset in read-only mode, in order to generate external overview (for GeoTIFF especially).
clean	Logical. (available from GDAL 1.7.0) remove all overviews.
oo	Character. NAME=VALUE. (starting with GDAL 2.0) Dataset open option (format specific)
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdaladdo' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/gdaladdo.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdaladdo.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
filename  <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
temp_filename <- paste(tempfile(),".tif",sep="")
file.copy(from=filename,to=temp_filename,overwrite=TRUE)
gdalinfo(filename)
gdaladdo(r="average",temp_filename,levels=c(2,4,8,16),verbose=TRUE)
gdalinfo(temp_filename)
}
```

---

gdalbuildvrt *gdalbuildvrt*

Description

R wrapper for gdalbuildvrt: Builds a VRT from a list of datasets

Usage

```
gdalbuildvrt(gdalfile, output.vrt, tileindex, resolution, te, tr, tap, separate,
  b, sd, allow_projection_difference, q, addalpha, hidenodata, srcnodata,
  vrtnodata, a_srs, r, input_file_list, overwrite, ignore.full_scan = TRUE,
  verbose = FALSE, ...)
```

Arguments

gdalfile	Character. Input files (as a character vector) or a wildcard search term (e.g. "*.tif")
output.vrt	Character. Output VRT file.
tileindex	Logical. Use the specified value as the tile index field, instead of the default value with is 'location'.
resolution	Character. ("highest"|"lowest"|"average"|"user") In case the resolution of all input files is not the same, the -resolution flag enables the user to control the way the output resolution is computed. 'average' is the default. 'highest' will pick the smallest values of pixel dimensions within the set of source rasters. 'lowest' will pick the largest values of pixel dimensions within the set of source rasters. 'average' will compute an average of pixel dimensions within the set of source rasters. 'user' is new in GDAL 1.7.0 and must be used in combination with the -tr option to specify the target resolution.

`te` Numeric. c(xmin,ymin,xmax,ymax) (starting with GDAL 1.7.0) set georeferenced extents of VRT file. The values must be expressed in georeferenced units. If not specified, the extent of the VRT is the minimum bounding box of the set of source rasters.

`tr` Numeric. c(xres,yres) (starting with GDAL 1.7.0) set target resolution. The values must be expressed in georeferenced units. Both must be positive values. Specifying those values is of course incompatible with highest|lowest|average values for -resolution option.

`tap` Logical. (GDAL >= 1.8.0) (target aligned pixels) align the coordinates of the extent of the output file to the values of the -tr, such that the aligned extent includes the minimum extent.

`separate` Logical. (starting with GDAL 1.7.0) Place each input file into a separate stacked band. In that case, only the first band of each dataset will be placed into a new band. Contrary to the default mode, it is not required that all bands have the same datatype.

`b` Numeric. (GDAL >= 1.10.0) Select an input band to be processed. Bands are numbered from 1. If input bands not set all bands will be added to vrt

`sd` Numeric. (GDAL >= 1.10.0) If the input dataset contains several subdatasets use a subdataset with the specified number (starting from 1). This is an alternative of giving the full subdataset name as an input.

`allow_projection_difference`
Logical. (starting with GDAL 1.7.0) When this option is specified, the utility will accept to make a VRT even if the input datasets have not the same projection. Note: this does not mean that they will be reprojected. Their projection will just be ignored.

`q` Logical. (starting with GDAL 1.7.0) To disable the progress bar on the console.

`addalpha` Logical. (starting with GDAL 1.7.0) Adds an alpha mask band to the VRT when the source raster have none. Mainly useful for RGB sources (or grey-level sources). The alpha band is filled on-the-fly with the value 0 in areas without any source raster, and with value 255 in areas with source raster. The effect is that a RGBA viewer will render the areas without source rasters as transparent and areas with source rasters as opaque. This option is not compatible with -separate.

`hidenodata` Logical. (starting with GDAL 1.7.0) Even if any band contains nodata value, giving this option makes the VRT band not report the NoData. Useful when you want to control the background color of the dataset. By using along with the -addalpha option, you can prepare a dataset which doesn't report nodata value but is transparent in areas with no data.

`srcnodata` Character. (starting with GDAL 1.7.0) Set nodata values for input 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. If the option is not specified, the intrinsic nodata settings on the source datasets will be used (if they exist). The value set by this option is written in the NODATA element of each ComplexSource element. Use a value of None to ignore intrinsic nodata settings on the source datasets.

vrtnodata	Character. (starting with GDAL 1.7.0) Set nodata values at the VRT band level (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. If the option is not specified, intrinsic nodata settings on the first dataset will be used (if they exist). The value set by this option is written in the NoDataValue element of each VRTRasterBand element. Use a value of None to ignore intrinsic nodata settings on the source datasets.
a_srs	Character. (starting with GDAL 1.10) Override the projection for the output file. The srs_def may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.
r	Character. ("nearest" (default) | "bilinear" | "cubic" | "cubicspline" | "lanczos" | "average" | "mode"). (GDAL >= 2.0) Select a resampling algorithm.
input_file_list	Character. To specify a text file with an input filename on each line.
overwrite	Logical. Overwrite the VRT if it already exists.
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.
...	Additional arguments.

Details

This is an R wrapper for the 'gdalbuildvrt' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/gdalbuildvrt.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdalbuildvrt.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
```

```
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
layer1 <- system.file("external/tahoe_lidar_bareearth.tif", package="gdalUtils")
layer2 <- system.file("external/tahoe_lidar_highesthit.tif", package="gdalUtils")
output.vrt <- paste(tempfile(),".vrt",sep="")
gdalbuildvrt(gdalfile=c(layer1,layer2),output.vrt=output.vrt,separate=TRUE,verbose=TRUE)
gdalinfo(output.vrt)
}
```

---

gdaldem *gdaldem*

Description

R wrapper for gdaldem: Tools to analyze and visualize DEMs. (since GDAL 1.7.0)

Usage

```
gdaldem(mode, input_dem, output, of, compute_edges, alg, b, co, q, z, s, az,
  alt, combined, p, trigonometric, zero_for_flat, color_text_file, alpha,
  exact_color_entry, nearest_color_entry, output_Raster = FALSE,
  ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

mode	Character. ("hillshade"|"slope"|"aspect"|"color-relief"|"TRI"|"TPI"|"roughness")
input_dem	Character. The input DEM raster to be processed.
output	Character. The output raster produced.
of	Character. Select the output format. The default is GeoTIFF (GTiff). Use the short format name.
compute_edges	Logical. (GDAL >= 1.8.0) Do the computation at raster edges and near nodata values.
alg	Character. "ZevenbergenThorne" (GDAL >= 1.8.0) Use Zevenbergen & Thorne formula, instead of Horn's formula, to compute slope & aspect. The litterature suggests Zevenbergen & Thorne to be more suited to smooth landscapes, whereas Horn's formula to perform better on rougher terrain.
b	Numeric. Select an input band to be processed. Bands are numbered from 1.
co	Character. (GDAL >= 1.8.0) Passes a creation option ("NAME=VALUE") to the output format driver. Multiple -co options may be listed. See format specific documentation for legal creation options for each format.
q	Logical. Suppress progress monitor and other non-error output.
z	Numeric. (mode=="hillshade") vertical exaggeration used to pre-multiply the elevations.

s	Numeric. (mode=="hillshade" | mode=="slope) ratio of vertical units to horizontal. If the horizontal unit of the source DEM is degrees (e.g Lat/Long WGS84 projection), you can use scale=111120 if the vertical units are meters (or scale=370400 if they are in feet).
az	Numeric. (mode=="hillshade") azimuth of the light, in degrees. 0 if it comes from the top of the raster, 90 from the east, ... The default value, 315, should rarely be changed as it is the value generally used to generate shaded maps.
alt	Numeric. (mode=="hillshade") altitude of the light, in degrees. 90 if the light comes from above the DEM, 0 if it is raking light.
combined	Character. (mode=="hillshade") "combined shading" (starting with GDAL 1.10) a combination of slope and oblique shading.
p	Logical. (mode=="slope") if specified, the slope will be expressed as percent slope. Otherwise, it is expressed as degrees.
trigonometric	Logical. (mode=="aspect") return trigonometric angle instead of azimuth. Thus 0deg means East, 90deg North, 180deg West, 270deg South.
zero_for_flat	Logical. (mode=="aspect") By using those 2 options, the aspect returned by gdaldem aspect should be identical to the one of GRASS r.slope.aspect. Otherwise, it's identical to the one of Matthew Perry's aspect.cpp utility.
color_text_file	Character. (mode=="color-relief") text-based color configuration file (see Description).
alpha	Logical. (mode=="color-relief") add an alpha channel to the output raster.
exact_color_entry	Logical. (mode=="color-relief") use strict matching when searching in the color configuration file. If none matching color entry is found, the "0,0,0,0" RGBA quadruplet will be used.
nearest_color_entry	Logical. (mode=="color-relief") use the RGBA quadruplet corresponding to the closest entry in the color configuration file.
output_Raster	Logical. Return output dst_dataset as a RasterBrick?
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdaldem' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://www.gdal.org/gdaldem.html), or, in some cases, use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

The user can choose to (optionally) return a RasterBrick of the output file (assuming raster/rgdal supports the particular output format).

Value

NULL or if(output_Raster), a RasterBrick.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Matthew Perry, Even Rouault, Howard Butler, and Chris Yesson (GDAL developers).

References

http://www.gdal.org/gdaldem.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && require(rgdal) && valid_install)
{
# We'll pre-check for a proper GDAL installation before running these examples:
gdal_setInstallation()
if(!is.null(getOption("gdalUtils_gdalPath")))
{
input_dem  <- system.file("external/tahoe_lidar_highesthit.tif", package="gdalUtils")
plot(raster(input_dem),col=gray.colors(256))

# Hillshading:
# Command-line gdaldem call:
# gdaldem hillshade tahoe_lidar_highesthit.tif output_hillshade.tif
output_hillshade <- gdaldem(mode="hillshade",input_dem=input_dem,
output="output_hillshade.tif",output_Raster=TRUE,verbose=TRUE)
plot(output_hillshade,col=gray.colors(256))

# Slope:
# Command-line gdaldem call:
# gdaldem slope tahoe_lidar_highesthit.tif output_slope.tif -p
output_slope <- gdaldem(mode="slope",input_dem=input_dem,
output="output_slope.tif",p=TRUE,output_Raster=TRUE,verbose=TRUE)
plot(output_slope,col=gray.colors(256))

# Aspect:
# Command-line gdaldem call:
# gdaldem aspect tahoe_lidar_highesthit.tif output_aspect.tif
output_aspect <- gdaldem(mode="aspect",input_dem=input_dem,
output="output_aspect.tif",output_Raster=TRUE,verbose=TRUE)
```

```
plot(output_aspect,col=gray.colors(256))
}
}
```

---

gdalinfo *gdalinfo*

Description

R wrapper for gdalinfo

Usage

```
gdalinfo(datasetname, mm, stats, approx_stats, hist, nogcp, nomd, nrat, noct,
  nofl, checksum, proj4, oo, mdd, sd, version, formats, format, optfile, config,
  debug, raw_output = TRUE, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

datasetname	Character. A raster dataset name. It can be either file name.
mm	Logical. Force computation of the actual min/max values for each band in the dataset?
stats	Logical. Read and display image statistics. Force computation if no statistics are stored in an image.
approx_stats	Logical. Read and display image statistics. Force computation if no statistics are stored in an image. However, they may be computed based on overviews or a subset of all tiles. Useful if you are in a hurry and don't want precise stats.
hist	Logical. Report histogram information for all bands.
nogcp	Logical. Suppress ground control points list printing. It may be useful for datasets with huge amount of GCPs, such as L1B AVHRR or HDF4 MODIS which contain thousands of them.
nomd	Logical. Suppress metadata printing. Some datasets may contain a lot of metadata strings.
nrat	Logical. Suppress printing of raster attribute table.
noct	Logical. Suppress printing of color table.
nofl	Logical. (GDAL >= 1.9.0) Only display the first file of the file list.
checksum	Logical. Force computation of the checksum for each band in the dataset.
proj4	Logical. (GDAL >= 1.9.0) Report a PROJ.4 string corresponding to the file's coordinate system.
oo	Character. (starting with GDAL 2.0) NAME=VALUE. Dataset open option (format specific).
mdd	Character. Report metadata for the specified domain.

`sd` Numeric. (GDAL >= 1.9.0) If the input dataset contains several subdatasets read and display a subdataset with specified number (starting from 1). This is an alternative of giving the full subdataset name.

`version` Logical. Report the version of GDAL and exit.

`formats` Logical. List all raster formats supported by this GDAL build (read-only and read-write) and exit. The format support is indicated as follows: 'ro' is read-only driver; 'rw' is read or write (ie. supports CreateCopy); 'rw+' is read, write and update (ie. supports Create). A 'v' is appended for formats supporting virtual IO (/vsimem, /vsigzip, /vsizip, etc). A 's' is appended for formats supporting subdatasets. Note: The valid formats for the output of gdalwarp are formats that support the Create() method (marked as rw+), not just the CreateCopy() method.

`format` Character. List detailed information about a single format driver. The format should be the short name reported in the –formats list, such as GTiff.

`optfile` Character. Read the named file and substitute the contents into the commandline options list. Lines beginning with # will be ignored. Multi-word arguments may be kept together with double quotes.

`config` Character. Sets the named configuration keyword to the given value, as opposed to setting them as environment variables. Some common configuration keywords are GDAL_CACHEMAX (memory used internally for caching in megabytes) and GDAL_DATA (path of the GDAL "data" directory). Individual drivers may be influenced by other configuration options.

`debug` Character. Control what debugging messages are emitted. A value of ON will enable all debug messages. A value of OFF will disable all debug messages. Another value will select only debug messages containing that string in the debug prefix code.

`raw_output` Logical. Dump the raw output of the gdalinfo (default=TRUE). If not, attempt to return a clean list (not all parameters will be retained, at present).

`ignore.full_scan` Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.

`verbose` Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdalinfo' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://www.gdal.org/gdalinfo.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

By default, this will return the gdalinfo as a character vector, one line of the output per element. The user can choose raw_output=FALSE for a cleaner format (similar to GDALinfo in the rgdal package), although not all parameters are preserved.

Value

character (if raw_output=TRUE) or list (if raw_output=FALSE).

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) and Matteo Mattiuzzi (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdalinfo.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
src_dataset <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
# Command-line gdalinfo call:
# gdalinfo tahoe_highrez.tif
gdalinfo(src_dataset,verbose=TRUE)
}
```

---

gdallocationinfo *gdallocationinfo*

Description

R wrapper for gdallocationinfo: raster query tool

Usage

```
gdallocationinfo(srcfile, x, y, coords, xml, lifonly, valonly, b, overview,
  l_srs, geoloc, wgs84, oo, raw_output = TRUE, ignore.full_scan = TRUE,
  verbose = FALSE)
```

Arguments

srcfile	Character. The source GDAL raster datasource name.
x	Numeric. X location of target pixel. By default the coordinate system is pixel/line unless -l_srs, -wgs84 or -geoloc supplied.
y	Numeric. Y location of target pixel. By default the coordinate system is pixel/line unless -l_srs, -wgs84 or -geoloc supplied.

coords	Character or Matrix. Filename of coordinates (space separated, no header) or a matrix of coordinates.
xml	Logical. The output report will be XML formatted for convenient post processing.
lifonly	Logical. The only output is filenames production from the LocationInfo request against the database (ie. for identifying impacted file from VRT).
valonly	Logical. The only output is the pixel values of the selected pixel on each of the selected bands.
b	Numeric. band. Selects a band to query. Multiple bands can be listed. By default all bands are queried.
overview	Numeric. overview_level. Query the (overview_level)th overview (overview_level=1 is the 1st overview), instead of the base band. Note that the x,y location (if the coordinate system is pixel/line) must still be given with respect to the base band.
l_srs	Character. srs def. The coordinate system of the input x, y location.
geoloc	Logical. Indicates input x,y points are in the georeferencing system of the image.
wgs84	Logical. Indicates input x,y points are WGS84 long, lat.
oo	Character. "NAME=VALUE". (starting with GDAL 2.0) Dataset open option (format specific)
raw_output	Logical. Dump the raw output of the gdallocationinfo (default=TRUE). If not, attempt to return a matrix of data.
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdallocationinfo' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://www.gdal.org/gdallocationinfo.html), or, in some cases, use R vectors to achieve the same end.

This utility is intended to provide a variety of information about a pixel. Currently it reports three things:

The location of the pixel in pixel/line space. The result of a LocationInfo metadata query against the datasource - currently this is only implemented for VRT files which will report the file(s) used to satisfy requests for that pixel. The raster pixel value of that pixel for all or a subset of the bands. The unscaled pixel value if a Scale and/or Offset apply to the band. The pixel selected is requested by x/y coordinate on the commandline, or read from stdin. More than one coordinate pair can be supplied when reading coordinatesis from stdin. By default pixel/line coordinates are expected. However with use of the -geoloc, -wgs84, or -l_srs switches it is possible to specify the location in other coordinate systems.

The default report is in a human readable text format. It is possible to instead request xml output with the -xml switch.

For scripting purposes, the -valonly and -lifonly switches are provided to restrict output to the actual pixel values, or the LocationInfo files identified for the pixel.

It is anticipated that additional reporting capabilities will be added to gdallocationinfo in the future.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

Character or matrix (if valonly=T & raw_output=F)

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdallocationinfo.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
src_dataset <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
# Raw output of a single coordinate:
gdallocationinfo(srcfile=src_dataset,x=10,y=10)

# A matrix of coordinates and a clean, matrix output:
coords <- rbind(c(10,10),c(20,20),c(30,30))
gdallocationinfo(srcfile=src_dataset,coords=coords,valonly=TRUE,raw_output=FALSE)
}
```

---

gdalmanage *gdalmanage*

Description

R wrapper for gdalmanage: Identify, delete, rename and copy raster data files

Usage

```
gdalmanage(mode, datasetname, newdatasetname, r, u, f,
  ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

mode	Character. Mode of operation. "identify" | "copy" | "rename" | "delete". See details.
datasetname	Character. Raster file to operate on.
newdatasetname	Character. For copy and rename modes, you provide a source filename and a target filename, just like copy and move commands in an operating system.
r	Logical. Recursively scan files/folders for raster files.
u	Logical. Report failures if file type is unidentified.
f	Character. format. Specify format of raster file if unknown by the application. Uses short data format name (e.g. GTiff).
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdalmanage' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/gdalmanage.html), or, in some cases, can use R vectors to achieve the same end.

Mode of operation

- mode="identify",datasetname: List data format of file.
- mode="copy",datasetname,newdatasetname: Create a copy of the raster file with a new name.
- mode="rename",datasetname,newdatasetname: Change the name of the raster file.
- mode="delete",datasetname: Delete raster file.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

Character.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdalmanage.html

Examples

```
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
# Using identify mode
# Report the data format of the raster file by using the identify mode
 # and specifying a data file name:
src_dataset <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
gdalmanage(mode="identify",datasetname=src_dataset)

# Recursive mode will scan subfolders and report the data format:
src_dir <- system.file("external/", package="gdalUtils")
gdalmanage(mode="identify",datasetname=src_dir,r=TRUE)

## Not run:
# Using copy mode
# Copy the raster data:
file_copy <- tempfile(fileext=".tif")
gdalmanage(mode="copy",src_dataset,file_copy)
file.exists(file_copy)

# Using rename mode
# Rename the raster data:
new_name <- tempfile(fileext=".tif")
gdalmanage(mode="rename",file_copy,new_name)
file.exists(new_name)

# Using delete mode
# Delete the raster data:
gdalmanage(mode="delete",new_name)
file.exists(new_name)

## End(Not run)
}
```

---

gdalsrsinfo *gdalsrsinfo*

Description

R wrapper for gdalsrsinfo: lists info about a given SRS in number of formats (WKT, PROJ.4, etc.)

Usage

```
gdalsrsinfo(srs_def, p, V, o, as.CRS = FALSE, ignore.full_scan = TRUE,
  verbose = FALSE)
```

Arguments

srs_def	Character. A raster dataset name. It can be either file name.
p	Logical. Pretty-print where applicable (e.g. WKT).
V	Logical. Validate SRS.
o	Character. Output type ("default"|"all"|"wkt_all"|"proj4"|"wkt"|"wkt_simple"|"wkt_noct"|"wkt_esri"|"map
as.CRS	Logical. Return a CRS object? Default=FALSE.
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdalsrsinfo' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://www.gdal.org/gdalsrsinfo.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

If as.CRS is set to TRUE, 'o' will automatically be set to "proj4" and the output will be coerced to a CRS object for use with sp.

Value

character

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) and Matteo Mattiuzzi (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdalsrsinfo.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
src_dataset <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
```

```
# Command-line gdalsrsinfo call:
# gdalsrsinfo -o proj4 tahoe_highrez.tif
gdalsrsinfo(src_dataset,o="proj4",verbose=TRUE)
# Export as CRS:
gdalsrsinfo(src_dataset,as.CRS=TRUE,verbose=TRUE)
}
```

---

gdaltindex *gdaltindex*

Description

R wrapper for gdaltindex: Builds a shapefile as a raster tileindex

Usage

```
gdaltindex(index_file, gdal_file, f, tileindex, write_absolute_path,
  skip_different_projection, t_srs, src_srs_name, src_srs_format, lyr_name,
  output_Vector = FALSE, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

index_file	Character. The name of the output file to create/append to. The default shapefile will be created if it doesn't already exist, otherwise it will append to the existing file.
gdal_file	Character. The input GDAL raster files, can be multiple files separated by spaces. Wildcards my also be used. Stores the file locations in the same style as specified here, unless -write_absolute_path option is also used.
f	Character. format. The OGR format of the output tile index file. Default is Esri Shapefile.
tileindex	Character. field_name. The output field name to hold the file path/location to the indexed rasters. The default tile index field name is location.
write_absolute_path	Logical. The absolute path to the raster files is stored in the tile index file. By default the raster filenames will be put in the file exactly as they are specified on the command line.
skip_different_projection	Logical. Only files with same projection as files already inserted in the tileindex will be inserted (unless -t_srs is specified). Default does not check projection and accepts all inputs.
t_srs	Character. target_srs. Geometries of input files will be transformed to the desired target coordinate reference system. Using this option generates files that are not compatible with MapServer < 6.4. Default creates simple rectangular polygons in the same coordinate reference system as the input rasters.
src_srs_name	Character. field_name. The name of the field to store the SRS of each tile. This field name can be used as the value of the TILESRS keyword in MapServer >= 6.4.

src_srs_format	Character. type. The format in which the SRS of each tile must be written. Types can be AUTO, WKT, EPSG, PROJ.
lyr_name	Character. name. Layer name to create/append to in the output tile index file.
output_Vector	Logical. Return output dst_filename as a Spatial* object. Currently only works with f="ESRI Shapefile".
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdaltindex' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdaltindex format (http://www.gdal.org/gdaltindex.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL that has the right drivers as specified with the "of" (output format) parameter.

The user can choose to (optionally) return a SpatialPolygonsDataFrame of the output file.

Value

NULL or if(output_Vector), a SpatialPolygonsDataFrame.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdaltindex.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(rgdal) && valid_install)
{
# Modified example from the original gdaltindex documentation:
src_folder <- system.file("external/", package="gdalUtils")
output_shapefile <- paste(tempfile(),".shp",sep="")
```

```
# Command-line gdalwarp call:
# gdaltindex doq_index.shp external/*.tif
gdaltindex(output_shapefile,list.files(path=src_folder,pattern=glob2rx("*.tif"),full.names=TRUE),
output_Vector=TRUE,verbose=TRUE)
}
```

---

gdaltransform *gdaltransform*

Description

R wrapper for gdaltransform: transforms coordinates

Usage

```
gdaltransform(srcfile, dstfile, coords, s_srs, t_srs, to, order, tps, rpc,
  geoloc, i, gcp, output_xy, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

srcfile	Character. File with source projection definition or GCP's. If not given, source projection is read from the command-line -s_srs or -gcp parameters.
dstfile	Character. File with destination projection definition.
coords	Matrix. A two-column matrix with coordinates.
s_srs	Character. source spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf file containing well known text.
t_srs	Character. target spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf file containing well known text.
to	Character. "NAME=VALUE". set a transformer option suitable to pass to GDALCreateGenImgProjTransformer2().
order	Numeric. order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.
tps	Logical. Force use of thin plate spline transformer based on available GCPs.
rpc	Logical. Force use of RPCs.
geoloc	Logical. Force use of Geolocation Arrays.
i	Logical. Inverse transformation: from destination to source.
gcp	Character. pixel line easting northing [elevation]: Provide a GCP to be used for transformation (generally three or more are required)
output_xy	Logical. (GDAL >= 2.0) Restrict output to "x y" instead of "x y z"

ignore.full_scan
: Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.

verbose
: Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdaltransform' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/gdaltransform.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

Numeric.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdaltransform.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
pts <- matrix(c(177502,311865,177503,311866),ncol=2,byrow=TRUE)
gdaltransform(s_srs="EPSG:28992",t_srs="EPSG:31370",coords=pts,verbose=TRUE)
}
```

gdalwarp *gdalwarp*

Description

R wrapper for gdalwarp: image reprojection and warping utility

Usage

```
gdalwarp(srcfile, dstfile, s_srs, t_srs, to, order, tps, rpc, geoloc, et,
  refine_gcps, te, te_srs, tr, tap, ts, ovr, wo, ot, wt, r, srcnodata,
  dstnodata, dstalpha, wm, multi, q, of = "GTiff", co, cutline, cl, cwhere,
  csql, cblend, crop_to_cutline, overwrite, nomd, cvmd, setci, oo, doo,
  output_Raster = FALSE, ignore.full_scan = TRUE, verbose = FALSE, ...)
```

Arguments

srcfile	Character. The source file name(s).
dstfile	Character. The destination file name.
s_srs	Character. source spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf file containing well known text.
t_srs	Character. target spatial reference set. The coordinate systems that can be passed are anything supported by the OGRSpatialReference.SetFromUserInput() call, which includes EPSG PCS and GCSes (ie. EPSG:4296), PROJ.4 declarations (as above), or the name of a .prf file containing well known text.
to	Character. set a transformer option suitable to pass to GDALCreateGenImgProjTransformer2().
order	Numeric. order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.
tps	Logical. Force use of thin plate spline transformer based on available GCPs.
rpc	Logical. Force use of RPCs.
geoloc	Logical. Force use of Geolocation Arrays.
et	Numeric. error threshold for transformation approximation (in pixel units - defaults to 0.125).
refine_gcps	Numeric. (GDAL >= 1.9.0) refines the GCPs by automatically eliminating outliers. Outliers will be eliminated until minimum_gcps are left or when no outliers can be detected. The tolerance is passed to adjust when a GCP will be eliminated. Note that GCP refinement only works with polynomial interpolation. The tolerance is in pixel units if no projection is available, otherwise it is in SRS units. If minimum_gcps is not provided, the minimum GCPs according to the polynomial model is used.

te	Numeric. (c(xmin,ymin,xmax,ymax)). set georeferenced extents of output file to be created (in target SRS).
te_srs	Character. srs_def. (GDAL >= 2.0) Specifies the SRS in which to interpret the coordinates given with -te. The srs_def may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT. This must not be confused with -t_srs which is the target SRS of the output dataset. -te_srs is a conveniency e.g. when knowing the output coordinates in a geodetic long/lat SRS, but still wanting a result in a projected coordinate system.
tr	Numeric. (c(xres,yres)). set output file resolution (in target georeferenced units)
tap	Logical. (GDAL >= 1.8.0) (target aligned pixels) align the coordinates of the extent of the output file to the values of the -tr, such that the aligned extent includes the minimum extent.
ts	Numeric. (c(width,height)). set output file size in pixels and lines. If width or height is set to 0, the other dimension will be guessed from the computed resolution. Note that -ts cannot be used with -tr
ovr	Character. (level | "AUTO" | "AUTO-n" | "NONE"). (GDAL >= 2.0) To specify which overview level of source files must be used. The default choice, AUTO, will select the overview level whose resolution is the closest to the target resolution. Specify an integer value (0-based, i.e. 0=1st overview level) to select a particular level. Specify AUTO-n where n is an integer greater or equal to 1, to select an overview level below the AUTO one. Or specify NONE to force the base resolution to be used.
wo	Character. Set a warp options. The GDALWarpOptions::papszWarpOptions docs show all options. Multiple -wo options may be listed.
ot	Character. For the output bands to be of the indicated data type.
wt	Character. Working pixel data type. The data type of pixels in the source image and destination image buffers.
r	Character. resampling_method. ("near"|"bilinear"|"cubic"|"cubicspline"|"lanczos"|"average"|"mode"|"ma See Description.
srcnodata	Character. Set nodata masking values for input 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. Masked values will not be used in interpolation. Use a value of None to ignore intrinsic nodata settings on the source dataset.
dstnodata	Character. 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 None to ensure that nodata is not defined (GDAL>=2.0). If this argument is not used then nodata values will be copied from the source dataset (GDAL>=2.0).
dstalpha	Logical. Create an output alpha band to identify nodata (unset/transparent) pixels.
wm	Numeric. Set the amount of memory (in megabytes) that the warp API is allowed to use for caching.

multi	Logical. Use multithreaded warping implementation. Multiple threads will be used to process chunks of image and perform input/output operation simultaneously.
q	Logical. Be quiet.
of	Character. Select the output format. The default is GeoTIFF (GTiff). Use the short format name.
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.
cutline	Character. Enable use of a blend cutline from the name OGR support datasource.
cl	Character. Select the named layer from the cutline datasource.
cwhere	Character. Restrict desired cutline features based on attribute query.
csql	Character. Select cutline features using an SQL query instead of from a layer with -cl.
cblend	Numeric. Set a blend distance to use to blend over cutlines (in pixels).
crop_to_cutline	Logical. (GDAL >= 1.8.0) Crop the extent of the target dataset to the extent of the cutline.
overwrite	Logical. (GDAL >= 1.8.0) Overwrite the target dataset if it already exists.
nomd	Logical. (GDAL >= 1.10.0) Do not copy metadata. Without this option, dataset and band metadata (as well as some band information) will be copied from the first source dataset. Items that differ between source datasets will be set to * (see -cvmd option).
cvmd	Character. (GDAL >= 1.10.0) Value to set metadata items that conflict between source datasets (default is "*"). Use "" to remove conflicting items.
setci	Logical. (GDAL >= 1.10.0) Set the color interpretation of the bands of the target dataset from the source dataset.
oo	Character. NAME=VALUE. (starting with GDAL 2.0) Dataset open option (format specific).
doo	Character. NAME=VALUE. (starting with GDAL 2.1) Output dataset open option (format specific).
output_Raster	Logical. Return output dst_dataset as a RasterBrick?
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.
...	Additional arguments.

Details

This is an R wrapper for the 'gdalwarp' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some

modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalwarp format (http://www.gdal.org/gdalwarp.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL that has the right drivers as specified with the "of" (output format) parameter.

The resampling_methods available are as follows:

- near: nearest neighbour resampling (default, fastest algorithm, worst interpolation quality).
- bilinear: bilinear resampling.
- cubic: cubic resampling.
- cubicspline: cubic spline resampling.
- lanczos: Lanczos windowed sinc resampling.
- average: average resampling, computes the average of all non-NODATA contributing pixels. (GDAL >= 1.10.0)
- mode: mode resampling, selects the value which appears most often of all the sampled points. (GDAL >= 1.10.0)
- max: maximum resampling, selects the maximum value from all non-NODATA contributing pixels. (GDAL >= 2.0.0)
- min: minimum resampling, selects the minimum value from all non-NODATA contributing pixels. (GDAL >= 2.0.0)
- med: median resampling, selects the median value of all non-NODATA contributing pixels. (GDAL >= 2.0.0)
- q1: first quartile resampling, selects the first quartile value of all non-NODATA contributing pixels. (GDAL >= 2.0.0)
- q3: third quartile resampling, selects the third quartile value of all non-NODATA contributing pixels. (GDAL >= 2.0.0)

The user can choose to (optionally) return a RasterBrick of the output file (assuming raster/rgdal supports the particular output format).

Value

NULL or if(output_Raster), a RasterBrick.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdalwarp.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && require(rgdal) && valid_install)
{
# Example from the original gdal_translate documentation:
src_dataset <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
# Command-line gdalwarp call:
# gdalwarp -t_srs '+proj=utm +zone=11 +datum=WGS84' raw_spot.tif utm11.tif
gdalwarp(src_dataset,dstfile="tahoe_highrez_utm11.tif",
t_srs='+proj=utm +zone=11 +datum=WGS84',output_Raster=TRUE,
overwrite=TRUE,verbose=TRUE)
}
```

`gdal_chooseInstallation` *gdal_chooseInstallation*

Description

Choose a GDAL installation based on certain requirements.

Usage

```
gdal_chooseInstallation(hasDrivers)
```

Arguments

`hasDrivers`	Character. Which drivers must be available?

Details

By default, the GDAL commands will use the installation found at getOption("gdalUtils_gdalPath")[[1]], which is the most recent version found on the system. If the user has more than one GDAL installed (more common on Windows and Mac systems than *nix systems), gdal_chooseInstallation can be used to choose an installation (perhaps not the most recent one) that has certain functionality, e.g. supports HDF4 formatted files.

Value

Numeric id of the most recent installation that matches the requirements.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>)

References

http://www.gdal.org/gdal_translate.html

Examples

```
## Not run:
# Choose the best installation that has both HDF4 and HDF5 drivers:
gdal_chooseInstallation(hasDrivers=c("HDF4","HDF5"))
# Get the version of this installation:
getOption("gdalUtils_gdalPath")[[
gdal_chooseInstallation(hasDrivers=c("HDF4","HDF5"))]]$version

## End(Not run)
```

`gdal_cmd_builder` *gdal_cmd_builder*

Description

Helper function for building GDAL commands.

Usage

```
gdal_cmd_builder(executable, parameter_variables = c(),
  parameter_values = c(), parameter_order = c(), parameter_noflags = c(),
  parameter_doubledash = c(), parameter_noquotes = c(),
  gdal_installation_id = 1)
```

Arguments

`executable`	Character. The GDAL command to use (e.g. "gdal_translate")
`parameter_variables`	List. A list of parameter names, organized by type.
`parameter_values`	List. A list of the parameters names/values.
`parameter_order`	Character. The order of the parameters for the GDAL command.
`parameter_noflags`	Character. Parameters which do not have a flag.
`parameter_doubledash`	Character. Parameters which should have a double dash "–".
`parameter_noquotes`	Character. Parameters which should not be wrapped in quotes (vector parameters only, at present).
`gdal_installation_id`	Numeric. The ID of the GDAL installation to use. Defaults to 1.

Details

This function takes the executable name (e.g. "gdal_translate"), a list of parameter names organized by logical, vector, scalar, character, repeatable, a list of values of these parameters, the order they should be used in the GDAL command, and a list of parameters that should not have a flag, and returns a properly formatted GDAL command (with the full path-to-executable) that should work with a system() call.

Sometimes, a user may not want to use the most recent GDAL install (gdal_installation_id=1), so the gdal_installation_id can be used to set a different install. This is often used with gdal_chooseInstallation if, for instance, the particular GDAL installation required needs a specific driver that may not be available in all installations.

In general, an end user shouldn't need to use this function – it is used by many of the GDAL wrappers within gdalUtils.

Value

Formatted GDAL command for use with system() calls.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>)

References

http://www.gdal.org/gdal_translate.html

Examples

```
## Not run:
# This builds a gdal_translate command.
executable <- "gdal_translate"

parameter_variables <- list(
logical = list(
varnames <- c("strict","unscale","epo",
"eco","q","sds","stats")),
vector = list(
varnames <- c("outsize","scale","srcwin",
"projwin","a_ullr","gcp")),
scalar = list(
varnames <- c("a_nodata")),
character = list(
varnames <- c("ot","of","mask","expand","a_srs",
"src_dataset","dst_dataset")),
repeatable = list(
varnames <- c("b","mo","co")))

parameter_order <- c(
"strict","unscale","epo","eco","q","sds","stats",
"outsize","scale","srcwin","projwin","a_ullr","gcp",
"a_nodata",
```

```
"ot","of","mask","expand","a_srs",
"b","mo","co",
"src_dataset","dst_dataset")

parameter_noflags <- c("src_dataset","dst_dataset")

# Now assign some parameters:
parameter_values = list(
src_dataset = "input.tif",
dst_dataset = "output.envi",
of = "ENVI",
strict = TRUE
)

cmd <- gdal_cmd_builder(
executable=executable,
parameter_variables=parameter_variables,
parameter_values=parameter_values,
parameter_order=parameter_order,
parameter_noflags=parameter_noflags)

cmd
system(cmd,intern=TRUE)

## End(Not run)
```

---

gdal_contour *gdal_contour*

Description

R wrapper for gdal_contour: builds vector contour lines from a raster elevation model

Usage

```
gdal_contour(src_filename, dst_filename, b, a, threeD, inodata, snodata, i,
  f = "ESRI Shapefile", dsco, lco, off, fl, nln, output_Vector = FALSE,
  ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

src_filename	Character. Any OGR supported readable datasource.
dst_filename	Character. The OGR supported output file.
b	Numeric. Picks a particular band to get the DEM from. Defaults to band 1.
a	Character. Provides a name for the attribute in which to put the elevation. If not provided no elevation attribute is attached.
threeD	Logical. (GDAL parameter '3d') Force production of 3D vectors instead of 2D. Includes elevation at every vertex.

inodata	Logical. Ignore any nodata value implied in the dataset - treat all values as valid.
snodata	Numeric. Input pixel value to treat as "nodata".
i	Numeric. Elevation interval between contours.
f	Character. Create output in a particular format, default is "ESRI Shapefiles".
dsco	Character. Dataset creation option (format specific). Follows "NAME=VALUE" format.
lco	Character. Layer creation option (format specific). Follows "NAME=VALUE" format.
off	Numeric. Offset from zero relative to which to interpret intervals.
fl	Character. Name one or more "fixed levels" to extract.
nln	Character. Provide a name for the output vector layer. Defaults to "contour".
output_Vector	Logical. Return output dst_filename as a Spatial* object. Currently only works with f="ESRI Shapefile".
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdal_contour' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdal_contour format (http://www.gdal.org/gdal_contour.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL that has the right drivers as specified with the "of" (output format) parameter.

The user can choose to (optionally) return a RasterBrick of the output file (assuming raster/rgdal supports the particular output format).

Value

output vector filename or SpatialLinesDataFrame object.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdal_contour.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && require(rgdal) && valid_install)
{
# Example from the original gdal_contour documentation:
#  gdal_contour -a elev dem.tif contour.shp -i 10.0
# Choose a DEM:
input_dem  <- system.file("external/tahoe_lidar_bareearth.tif", package="gdalUtils")
# Setup an output filename (shapefile):
output_shapefile <- paste(tempfile(),".shp",sep="")
contour_output <- gdal_contour(src_filename=input_dem,dst_filename=output_shapefile,
a="Elevation",i=5.,output_Vector=TRUE)
# Plot the contours using spplot:
spplot(contour_output["Elevation"],contour=TRUE)
}
```

---

gdal_grid *gdal_grid*

Description

R wrapper for gdal_grid: creates regular grid from the scattered data

Usage

```
gdal_grid(src_datasource, dst_filename, ot, of, txe, tye, outsize, a_srs,
  zfield, z_increase, z_multiply, a, spat, clipsrc, clipsrcsql, clipsrclayer,
  clipsrcwhere, l, where, sql, co, q, output_Raster = FALSE,
  ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

src_datasource	Character. Any OGR supported readable datasource.
dst_filename	Character. The GDAL supported output file.
ot	Character. "type". For the output bands to be of the indicated data type.
of	Character. "format". Select the output format. The default is GeoTIFF (GTiff). Use the short format name.
txe	Numeric. c(xmin,xmax). Set georeferenced X extents of output file to be created.
tye	Numeric. c(ymin,ymax). Set georeferenced Y extents of output file to be created.

outsize	Numeric. c(xsize,ysize). Set the size of the output file in pixels and lines.
a_srs	Character. "srs_def". Override the projection for the output file. The srs_def may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.
zfield	Character. "field_name". Identifies an attribute field on the features to be used to get a Z value from. This value overrides Z value read from feature geometry record (naturally, if you have a Z value in geometry, otherwise you have no choice and should specify a field name containing Z value).
z_increase	Numeric. increase_value. Addition to the attribute field on the features to be used to get a Z value from. The addition should be the same unit as Z value. The result value will be Z value + Z increase value. The default value is 0.
z_multiply	Numeric. multiply_value. This is multiplication ratio for Z field. This can be used for shift from e.g. foot to meters or from elevation to deep. The result value will be (Z value + Z increase value) * Z multiply value. The default value is 1.
a	Character. [algorithm[:parameter1=value1][:parameter2=value2]...] Set the interpolation algorithm or data metric name and (optionally) its parameters. See INTERPOLATION ALGORITHMS and DATA METRICS sections for further discussion of available options.
spat	Numeric. c(xmin,ymin,xmax,ymax). Adds a spatial filter to select only features contained within the bounding box described by (xmin, ymin) - (xmax, ymax).
clipsrc	Numeric or Character. c(xmin,ymin,xmax,ymax)|WKT|datasource|spat_extent. Adds a spatial filter to select only features contained within the specified bounding box (expressed in source SRS), WKT geometry (POLYGON or MULTIPOLYGON), from a datasource or to the spatial extent of the -spat option if you use the spat_extent keyword. When specifying a datasource, you will generally want to use it in combination of the -clipsrclayer, -clipsrcwhere or -clipsrcsql options.
clipsrcsql	Character. Select desired geometries using an SQL query instead.
clipsrclayer	Character. "layername". Select the named layer from the source clip datasource.
clipsrcwhere	Character. "expression". Restrict desired geometries based on attribute query.
l	Character. "layername". Indicates the layer(s) from the datasource that will be used for input features. May be specified multiple times, but at least one layer name or a -sql option must be specified.
where	Character. "expression". An optional SQL WHERE style query expression to be applied to select features to process from the input layer(s).
sql	Character. "select_statement". An SQL statement to be evaluated against the datasource to produce a virtual layer of features to be processed.
co	Character. "NAME=VALUE". 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.
q	Logical. Suppress progress monitor and other non-error output.
output_Raster	Logical. Return output dst_filename as a RasterBrick?

`ignore.full_scan`
: Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.

`verbose`
: Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdal_grid' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdal_contour format (http://www.gdal.org/gdal_grid.html), or, in some cases, can use R vectors to achieve the same end.

INTERPOLATION ALGORITHMS

There are number of interpolation algorithms to choose from.

- invdist

  Inverse distance to a power. This is default algorithm. It has following parameters:

  - power: Weighting power (default 2.0).
  - smoothing: Smoothing parameter (default 0.0).
  - radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
  - radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
  - angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).
  - max_points: Maximum number of data points to use. Do not search for more points than this number. This is only used if search ellipse is set (both radii are non-zero). Zero means that all found points should be used. Default is 0.
  - min_points: Minimum number of data points to use. If less amount of points found the grid node considered empty and will be filled with NODATA marker. This is only used if search ellipse is set (both radii are non-zero). Default is 0.
  - nodata: NODATA marker to fill empty points (default 0.0).

- invdistnn

  (Since GDAL 2.1) Inverse distance to a power with nearest neighbor searching, ideal when max_points is used. It has following parameters:

  - power: Weighting power (default 2.0).
  - radius: The radius of the search circle, which should be non-zero. Default is 1.0.
  - max_points: Maximum number of data points to use. Do not search for more points than this number. Found points will be ranked from nearest to furthest distance when weighting. Default is 12.
  - min_points: Minimum number of data points to use. If less amount of points found the grid node is considered empty and will be filled with NODATA marker. Default is 0.
  - nodata: NODATA marker to fill empty points (default 0.0).

- average

  Moving average algorithm. It has following parameters:

  - radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
  - radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
  - angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).
  - min_points: Minimum number of data points to use. If less amount of points found the grid node considered empty and will be filled with NODATA marker. Default is 0.
  - nodata: NODATA marker to fill empty points (default 0.0). Note, that it is essential to set search ellipse for moving average method. It is a window that will be averaged when computing grid nodes values.

- nearest

  Nearest neighbor algorithm. It has following parameters:

  - radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
  - radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
  - angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).
  - nodata: NODATA marker to fill empty points (default 0.0).

- linear

  (Since GDAL 2.1) Linear interpolation algorithm.

  The Linear method performs linear interpolation by compution a Delaunay triangulation of the point cloud, finding in which triangle of the triangulation the point is, and by doing linear interpolation from its barycentric coordinates within the triangle. If the point is not in any triangle, depending on the radius, the algorithm will use the value of the nearest point or the nodata value.

  It has following parameters:

  - radius: In case the point to be interpolated does not fit into a triangle of the Delaunay triangulation, use that maximum distance to search a nearest neighbour, or use nodata otherwise. If set to -1, the search distance is infinite. If set to 0, nodata value will be always used. Default is -1.
  - nodata: NODATA marker to fill empty points (default 0.0).

DATA METRICS

Besides the interpolation functionality gdal_grid can be used to compute some data metrics using the specified window and output grid geometry. These metrics are:

- minimum: Minimum value found in grid node search ellipse.
- maximum: Maximum value found in grid node search ellipse.
- range: A difference between the minimum and maximum values found in grid node search ellipse.
- count: A number of data points found in grid node search ellipse.
- average_distance: An average distance between the grid node (center of the search ellipse) and all of the data points found in grid node search ellipse.

- average_distance_pts: An average distance between the data points found in grid node search ellipse. The distance between each pair of points within ellipse is calculated and average of all distances is set as a grid node value.

All the metrics have the same set of options:

- radius1: The first radius (X axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
- radius2: The second radius (Y axis if rotation angle is 0) of search ellipse. Set this parameter to zero to use whole point array. Default is 0.0.
- angle: Angle of search ellipse rotation in degrees (counter clockwise, default 0.0).
- min_points: Minimum number of data points to use. If less amount of points found the grid node considered empty and will be filled with NODATA marker. This is only used if search ellipse is set (both radii are non-zero). Default is 0.
- nodata: NODATA marker to fill empty points (default 0.0).

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL that has the right drivers as specified with the "of" (output format) parameter.

The user can choose to (optionally) return a RasterBrick of the output file (assuming raster/rgdal supports the particular output format).

Value

NULL or if(output_Raster), a RasterBrick.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdal_grid.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && valid_install)
{
# Create a properly formatted CSV:
temporary_dir <- tempdir()
tempfname_base <- file.path(temporary_dir,"dem")
tempfname_csv <- paste(tempfname_base,".csv",sep="")
```

```
pts <- data.frame(
Easting=c(86943.4,87124.3,86962.4,87077.6),
Northing=c(891957,892075,892321,891995),
Elevation=c(139.13,135.01,182.04,135.01)
)

write.csv(pts,file=tempfname_csv,row.names=FALSE)

# Now make a matching VRT file
tempfname_vrt <- paste(tempfname_base,".vrt",sep="")
vrt_header <- c(
'<OGRVRTDataSource>',
'\t<OGRVRTLayer name="dem">',
'\t<SrcDataSource>dem.csv</SrcDataSource>',
'\t<GeometryType>wkbPoint</GeometryType>',
 '\t<GeometryField encoding="PointFromColumns" x="Easting" y="Northing" z="Elevation"/>',
'\t</OGRVRTLayer>',
'\t</OGRVRTDataSource>'
)
vrt_filecon <- file(tempfname_vrt,"w")
writeLines(vrt_header,con=vrt_filecon)
close(vrt_filecon)

tempfname_tif <- paste(tempfname_base,".tiff",sep="")

# Now run gdal_grid:
setMinMax(gdal_grid(src_datasource=tempfname_vrt,
dst_filename=tempfname_tif,a="invdist:power=2.0:smoothing=1.0",
txe=c(85000,89000),tye=c(894000,890000),outsize=c(400,400),
of="GTiff",ot="Float64",l="dem",output_Raster=TRUE))
}
```

---

gdal_rasterize *gdal_rasterize*

Description

R wrapper for gdal_rasterize: burns vector geometries into a raster

Usage

```
gdal_rasterize(src_datasource, dst_filename, b, i, at, burn, a, threeD, l,
  where, sql, dialect, of, a_srs, co, a_nodata, init, te, tr, tap, ts, ot, q,
  output_Raster = FALSE, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

`src_datasource` Character. Any OGR supported readable datasource.

dst_filename	Character. The GDAL supported output file. Must support update mode access. Before GDAL 1.8.0, gdal_rasterize could not create new output files.
b	Numeric. The band(s) to burn values into. Multiple -b arguments may be used to burn into a list of bands. The default is to burn into band 1.
i	Logical. Invert rasterization. Burn the fixed burn value, or the burn value associated with the first feature into all parts of the image not inside the provided a polygon.
at	Logical. Enables the ALL_TOUCHED rasterization option so that all pixels touched by lines or polygons will be updated not just those one the line render path, or whose center point is within the polygon. Defaults to disabled for normal rendering rules.
burn	Numeric. A fixed value to burn into a band for all objects. A vector of burn options can be supplied, one per band being written to.
a	Character. Identifies an attribute field on the features to be used for a burn in value. The value will be burned into all output bands.
threeD	Logical. (GDAL parameter '3d') Indicates that a burn value should be extracted from the "Z" values of the feature. These values are adjusted by the burn value given by "-burn value" or "-a attribute_name" if provided. As of now, only points and lines are drawn in 3D.
l	Character. Indicates the layer(s) from the datasource that will be used for input features. May be specified multiple times, but at least one layer name or a -sql option must be specified.
where	Character. An optional SQL WHERE style query expression to be applied to select features to burn in from the input layer(s).
sql	Character. An SQL statement to be evaluated against the datasource to produce a virtual layer of features to be burned in.
dialect	Character. (starting with GDAL 2.1) The SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by passing OGRSQL. Starting with GDAL 1.10, the "SQLITE" dialect can also be used with any datasource.
of	Character. (GDAL >= 1.8.0) Select the output format. The default is GeoTIFF (GTiff). Use the short format name.
a_srs	Character. (GDAL >= 1.8.0) Override the projection for the output file. If not specified, the projection of the input vector file will be used if available. If incompatible projections between input and output files, no attempt will be made to reproject features. The srs_def may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.
co	Character. (GDAL >= 1.8.0) Passes a creation option ("NAME=VALUE") to the output format driver. Multiple -co options may be listed. See format specific documentation for legal creation options for each format.
a_nodata	Numeric. (GDAL >= 1.8.0) Assign a specified nodata value to output bands.
init	Numeric. (GDAL >= 1.8.0) Pre-initialize the output image bands with these values. However, it is not marked as the nodata value in the output file. If only one value is given, the same value is used in all the bands.

te	Numeric. c(xmin,ymin,xmax,ymax) (GDAL >= 1.8.0) set georeferenced extents. The values must be expressed in georeferenced units. If not specified, the extent of the output file will be the extent of the vector layers.
tr	Numeric. c(xres,yres) (GDAL >= 1.8.0) set target resolution. The values must be expressed in georeferenced units. Both must be positive values.
tap	Logical. (GDAL >= 1.8.0) (target aligned pixels) align the coordinates of the extent of the output file to the values of the -tr, such that the aligned extent includes the minimum extent.
ts	Numeric. c(width,height) (GDAL >= 1.8.0) set output file size in pixels and lines. Note that -ts cannot be used with -tr
ot	Character. (GDAL >= 1.8.0) For the output bands to be of the indicated data type. Defaults to Float64
q	Logical. (GDAL >= 1.8.0) Suppress progress monitor and other non-error output.
output_Raster	Logical. Return output dst_filename as a RasterBrick?
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'gdal_rasterize' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalwarp format (http://www.gdal.org/gdal_rasterize.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL that has the right drivers as specified with the "of" (output format) parameter.

The user can choose to (optionally) return a RasterBrick of the output file (assuming raster/rgdal supports the particular output format).

Value

NULL or if(output_Raster), a RasterBrick.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdal_rasterize.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && require(rgdal) && valid_install)
{
# Example from the original gdal_rasterize documentation:
# gdal_rasterize -b 1 -b 2 -b 3 -burn 255 -burn 0
#  -burn 0 -l tahoe_highrez_training tahoe_highrez_training.shp tempfile.tif
dst_filename_original  <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
# Back up the file, since we are going to burn stuff into it.
dst_filename <- paste(tempfile(),".tif",sep="")
file.copy(dst_filename_original,dst_filename,overwrite=TRUE)
#Before plot:
plotRGB(brick(dst_filename))
src_dataset <- system.file("external/tahoe_highrez_training.shp", package="gdalUtils")
tahoe_burned <- gdal_rasterize(src_dataset,dst_filename,
b=c(1,2,3),burn=c(0,255,0),l="tahoe_highrez_training",verbose=TRUE,output_Raster=TRUE)
#After plot:
plotRGB(brick(dst_filename))
}
```

---

gdal_setInstallation *gdal_setInstallation*

Description

Sets local GDAL installation options

Usage

```
gdal_setInstallation(search_path = NULL, rescan = FALSE,
  ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

search_path	Character. Force a search in a specified directory. This directory should contain the gdalinfo(.exe) executable. If a valid GDAL install is found in this path, this will force gdalUtils to use this installation. Remember to set rescan=TRUE if you have already set an install.
rescan	Logical. Force a rescan if neccessary (e.g. if you updated your GDAL install).
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This function searches the local system for valid installations of GDAL, and returns a list, one item per valid GDAL install, containing the path to the installation, the version, the release date, available drivers, and available python utilities. The list will be sorted by release date, so in general the first entry is the one that is used by the various GDAL utilities. Note that this will automatically run every time a GDAL wrapper function is called, so the user does not have to explicitly run it.

gdal_setInstallation is designed to invoke consecutively more rigorous searches in able to find a valid GDAL install. Understanding the search routine may help debug problems on your system. The order of the searches is as follows, noting that as soon as a valid install is found (determined by running gdalinfo –version and getting the correct output), gdal_setInstallation stops further searching:

1. Checks a pre-determined location given by the search_path parameter.
2. Checks using Sys.which(). This is typically defined in the system's PATH, so will override any other install.
3. Checks in common install locations (OS specific).
4. (optional, if ignore.full_scan=FALSE) Finally, if it can't find a valid GDAL install anywhere else, it will brute-force search the entire local system (which may take a long time).

Value

Sets an option "gdalUtils_gdalPath" with GDAL installation information.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) and Matteo Mattiuzzi

References

http://www.gdal.org/gdal_translate.html

Examples

```
## Not run:
# Assumes you have GDAL installed on your local machine.
getOption("gdalUtils_gdalPath")
gdal_setInstallation()
getOption("gdalUtils_gdalPath")
# If there is more than one installation of GDAL, this is the
# most recent installation:
getOption("gdalUtils_gdalPath")[[1]]
# The version number:
getOption("gdalUtils_gdalPath")[[1]]$version

## End(Not run)
```

gdal_translate *gdal_translate*

Description

R wrapper for gdal_translate

Usage

```
gdal_translate(src_dataset, dst_dataset, ot, strict, of = "GTiff", b, mask,
  expand, outsize, tr, r, scale, exponent, unscale, srcwin, projwin,
  projwin_srs, epo, eco, a_srs, a_ullr, a_nodata, mo, co, gcp, q, sds, stats,
  norat, oo, sd_index, output_Raster = FALSE, ignore.full_scan = TRUE,
  verbose = FALSE, ...)
```

Arguments

src_dataset	Character. The source dataset name. It can be either file name, URL of data source or subdataset name for multi-dataset files.
dst_dataset	Character. The destination file name.
ot	Character. ("Byte"/"Int16"/"UInt16"/"UInt32"/"Int32"/"Float32"/"Float64"/"CInt16"/"CInt32"/"CFloat32 For the output bands to be of the indicated data type.
strict	Logical. Don't be forgiving of mismatches and lost data when translating to the output format.
of	Character. Select the output format. The default is GeoTIFF (GTiff). Use the short format name.
b	Numeric or Character. Select an input band band for output. Bands are numbered from 1. Multiple bands may be used to select a set of input bands to write to the output file, or to reorder bands. Starting with GDAL 1.8.0, band can also be set to "mask,1" (or just "mask") to mean the mask band of the first band of the input dataset.
mask	Numeric. (GDAL >= 1.8.0) Select an input band band to create output dataset mask band. Bands are numbered from 1. band can be set to "none" to avoid copying the global mask of the input dataset if it exists. Otherwise it is copied by default ("auto"), unless the mask is an alpha channel, or if it is explicitly used to be a regular band of the output dataset ("-b mask"). band can also be set to "mask,1" (or just "mask") to mean the mask band of the 1st band of the input dataset.
expand	Character. ("gray"|"rgb"|"rgba"). (From GDAL 1.6.0) To expose a dataset with 1 band with a color table as a dataset with 3 (RGB) or 4 (RGBA) bands. Useful for output drivers such as JPEG, JPEG2000, MrSID, ECW that don't support color indexed datasets. The 'gray' value (from GDAL 1.7.0) enables to expand a dataset with a color table that only contains gray levels to a gray indexed dataset.

`outsize` Numeric. (c(xsize[percentage],ysize[percentage])). Set the size of the output file. Outsize is in pixels and lines unless '%' is attached in which case it is as a fraction of the input image size.

`tr` Numeric. c(xres,yres). (starting with GDAL 2.0) set target resolution. The values must be expressed in georeferenced units. Both must be positive values. This is exclusive with -outsize and -a_ullr.

`r` Character. resampling_method. ("nearest"|"bilinear"|"cubic"|"cubicspline"|"lanczos"|"average"|"mode") (GDAL >= 2.0) Select a resampling algorithm.

`scale` Numeric. (c(src_min,src_max,dst_min,dst_max)). Rescale the input pixels values from the range src_min to src_max to the range dst_min to dst_max. If omitted the output range is 0 to 255. If omitted the input range is automatically computed from the source data.

`exponent` Numeric. (From GDAL 1.11) To apply non-linear scaling with a power function. exp_val is the exponent of the power function (must be postive). This option must be used with the -scale option. If specified only once, -exponent applies to all bands of the output image. It can be repeated several times so as to specify per band parameters. It is also possible to use the "-exponent_bn" syntax where bn is a band number (e.g. "-exponent_2" for the 2nd band of the output dataset) to specify the parameters of one or several specific bands.

`unscale` Logical. Apply the scale/offset metadata for the bands to convert scaled values to unscaled values. It is also often necessary to reset the output datatype with the -ot switch.

`srcwin` Numeric. (c(xoff,yoff,xsize,ysize)). Selects a subwindow from the source image for copying based on pixel/line location.

`projwin` Numeric. (c(ulx,uly,lrx,lry)). Selects a subwindow from the source image for copying (like -srcwin) but with the corners given in georeferenced coordinates.

`projwin_srs` Character. srs_def. (GDAL >= 2.0) Specifies the SRS in which to interpret the coordinates given with -projwin. The srs_def may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT. Note that this does not cause reprojection of the dataset to the specified SRS.

`epo` Logical. (Error when Partially Outside) (GDAL >= 1.10) If this option is set, -srcwin or -projwin values that falls partially outside the source raster extent will be considered as an error. The default behaviour starting with GDAL 1.10 is to accept such requests, when they were considered as an error before.

`eco` Logical. (Error when Completely Outside) (GDAL >= 1.10) Same as -epo, except that the criterion for erroring out is when the request falls completely outside the source raster extent.

`a_srs` Character. Override the projection for the output file. The srs_def may be any of the usual GDAL/OGR forms, complete WKT, PROJ.4, EPSG:n or a file containing the WKT.

`a_ullr` Numeric. (c(ulx,uly,lrx,lry)). Assign/override the georeferenced bounds of the output file. This assigns georeferenced bounds to the output file, ignoring what would have been derived from the source file.

a_nodata	Numeric. Assign a specified nodata value to output bands. Starting with GDAL 1.8.0, can be set to none to avoid setting a nodata value to the output file if one exists for the source file
mo	Character. ("META-TAG=VALUE"). Passes a metadata key and value to set on the output dataset if possible.
co	Character. ("NAME=VALUE"). 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.
gcp	Numeric. (c(pixel,line,easting,northing(,elevation))). Add the indicated ground control point to the output dataset. This option may be provided multiple times to provide a set of GCPs.
q	Logical. Suppress progress monitor and other non-error output.
sds	Logical. Copy all subdatasets of this file to individual output files. Use with formats like HDF or OGDI that have subdatasets.
stats	Logical. (GDAL >= 1.8.0) Force (re)computation of statistics.
norat	Logical. (GDAL >= 1.11) Do not copy source RAT into destination dataset.
oo	Character. NAME=VALUE. (starting with GDAL 2.0) Dataset open option (format specific)
sd_index	Numeric. If the file is an HDF4 or HDF5 file, which subdataset should be returned (1 to the number of subdatasets)? If this flag is used, src_dataset should be the filename of the multipart file. This parameter only works if the subdataset names follow the SUBDATASET_n_NAME convention.
output_Raster	Logical. Return output dst_dataset as a RasterBrick?
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.
...	Additional arguments.

Details

This is an R wrapper for the 'gdal_translate' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdal_translate format (http://www.gdal.org/gdal_translate.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL that has the right drivers as specified with the "of" (output format) parameter.

The user can choose to (optionally) return a RasterBrick of the output file (assuming raster/rgdal supports the particular output format).

Value

NULL or if(output_Raster), a RasterBrick.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdal_translate.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && require(rgdal) && valid_install)
{
# Example from the original gdal_translate documentation:
src_dataset <- system.file("external/tahoe_highrez.tif", package="gdalUtils")
# Original gdal_translate call:
# gdal_translate -of GTiff -co "TILED=YES" tahoe_highrez.tif tahoe_highrez_tiled.tif
gdal_translate(src_dataset,"tahoe_highrez_tiled.tif",of="GTiff",co="TILED=YES",verbose=TRUE)
# Pull out a chunk and return as a raster:
gdal_translate(src_dataset,"tahoe_highrez_tiled.tif",of="GTiff",
srcwin=c(1,1,100,100),output_Raster=TRUE,verbose=TRUE)
# Notice this is the equivalent, but follows gdal_translate's parameter format:
gdal_translate(src_dataset,"tahoe_highrez_tiled.tif",of="GTiff",
srcwin="1 1 100 100",output_Raster=TRUE,verbose=TRUE)
}
## Not run:
# Extract the first subdataset from an HDF4 file:
hdf4_dataset <- system.file("external/test_modis.hdf", package="gdalUtils")
gdal_translate(hdf4_dataset,"test_modis_sd1.tif",sd_index=1)

## End(Not run)
```

---

get_subdatasets *get_subdatasets*

Description

Returns HDF4, HDF5, and NetCDF subdataset names for standardized files.

Usage

```
get_subdatasets(datasetname, names_only = TRUE, verbose = FALSE)
```

Arguments

datasetname	Character. Input HDF4/5 or NetCDF file.
names_only	Logical. Return subdataset names only? Default=TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

Currently, this only returns the subdataset names of HDF4, HDF5, and NetCDF files, assuming they follow the SUBDATASET_n_NAME convention. This can be used with gdal_translate to extract a single subdataset (or with gdal_translate(...,sd_index=n)

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

character vector of subdataset names that can be used in gdal_translate.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) and Matteo Mattiuzzi (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/gdalinfo.html

Examples

```
## Not run:
hdf4_dataset <- system.file("external/test_modis.hdf", package="gdalUtils")
get_subdatasets(hdf4_dataset)

## End(Not run)
```

mosaic_rasters *Mosaic raster files using GDAL Utilities*

Description

Mosaic raster files using GDAL Utilities

Usage

```
mosaic_rasters(gdalfile, dst_dataset, output.vrt = NULL,
  output_Raster = FALSE, trim_margins = NULL, verbose = FALSE, ...)
```

Arguments

gdalfile	Character. Input files (as a character vector) or a wildcard search term (e.g. "*.tif")
dst_dataset	Character. The destination file name.
output.vrt	Character. Output VRT file. If NULL a temporary .vrt file will be created.
output_Raster	Logical. Return output dst_dataset as a RasterBrick?
trim_margins	Numeric. Pre-crop the input tiles by a fixed number of pixels before mosaicking. Can be a single value or four values representing the left, top, right, and bottom margins, respectively.
verbose	Logical. Enable verbose execution? Default is FALSE.
...	Parameters to pass to gdalbuildvrt or gdal_translate.

Details

This function mosaics a set of input rasters (gdalfile) using parameters found in gdalbuildvrt and subsequently exports the mosaic to an output file (dst_dataset) using parameters found in gdal_translate. The user can choose to preserve the intermediate output.vrt file, but in general this is not needed.

Value

Either a list of NULLs or a list of RasterBricks depending on whether output_Raster is set to TRUE.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>)

See Also

gdalbuildvrt, gdal_translate

Examples

```
# We'll pre-check to make sure there is a valid GDAL install
# and that raster and rgdal are also installed.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(raster) && require(rgdal) && valid_install)
{
layer1 <- system.file("external/tahoe_lidar_bareearth.tif", package="gdalUtils")
layer2 <- system.file("external/tahoe_lidar_highesthit.tif", package="gdalUtils")
mosaic_rasters(gdalfile=c(layer1,layer2),dst_dataset="test_mosaic.envi",separate=TRUE,of="ENVI",
verbose=TRUE)
gdalinfo("test_mosaic.envi")
}
```

`nearblack` *nearblack*

Description

R wrapper for nearblack: convert nearly black/white borders to black

Usage

```
nearblack(infile, o, of, co, white, color, near, nb, setalpha, setmask, q,
  output_Raster = FALSE, overwrite = FALSE, ignore.full_scan = TRUE,
  verbose = FALSE)
```

Arguments

`infile`	Character. The input file. Any GDAL supported format, any number of bands, normally 8bit Byte bands.
`o`	Character. outfile. The name of the output file to be created. Newly created files are created with the HFA driver by default (Erdas Imagine - .img)
`of`	Character. format. (GDAL 1.8.0 or later) Select the output format. Use the short format name (GTiff for GeoTIFF for example).
`co`	Character. "NAME=VALUE". (GDAL 1.8.0 or later) 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. Only valid when creating a new file.
`white`	Logical. Search for nearly white (255) pixels instead of nearly black pixels.
`color`	Numeric. c1,c2,c3...cn. (GDAL >= 1.9.0) Search for pixels near the specified color. May be specified multiple times. When -color is specified, the pixels that are considered as the collar are set to 0.

near	Numeric. dist. Select how far from black, white or custom colors the pixel values can be and still considered near black, white or custom color. Defaults to 15.
nb	Numeric. non_black_pixels. number of non-black pixels that can be encountered before the giving up search inwards. Defaults to 2.
setalpha	Logical. (GDAL 1.8.0 or later) Adds an alpha band if the output file is specified and the input file has 3 bands, or sets the alpha band of the output file if it is specified and the input file has 4 bands, or sets the alpha band of the input file if it has 4 bands and no output file is specified. The alpha band is set to 0 in the image collar and to 255 elsewhere.
setmask	Logical. (GDAL 1.8.0 or later) Adds a mask band to the output file, or adds a mask band to the input file if it does not already have one and no output file is specified. The mask band is set to 0 in the image collar and to 255 elsewhere.
q	Logical. (GDAL 1.8.0 or later) Suppress progress monitor and other non-error output.
output_Raster	Logical. Return outfile as a RasterBrick?
overwrite	Logical. If output file exists, OR if output file is not set (which would defualt to overwriting the input file), allow overwriting?
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

#' This is an R wrapper for the 'nearblack' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/nearblack.html), or, in some cases, can use R vectors to achieve the same end.

This utility will scan an image and try to set all pixels that are nearly or exactly black, white or one or more custom colors around the collar to black or white. This is often used to "fix up" lossy compressed airphotos so that color pixels can be treated as transparent when mosaicking.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/nearblack.html

Examples

```
# None available at present.
```

---

ogr2ogr *ogr2ogr*

Description

R wrapper for ogr2ogr: converts simple features data between file formats

Usage

```
ogr2ogr(src_datasource_name, dst_datasource_name, layer, f, append, overwrite,
  update, select, progress, sql, dialect, where, skipfailures, spat, spat_srs,
  geomfield, dsco, lco, nln, nlt, dim, a_srs, t_srs, s_srs, preserve_fid, fid,
  oo, doo, gt, ds_transaction, clipsrc, clipsrcsql, clipsrclayer, clipsrcwhere,
  clipdst, clipdstsql, clipdstlayer, clipdstwhere, wrapdateline, datelineoffset,
  simplify, segmentize, fieldTypeToString, mapFieldType, unsetFieldWidth,
  splitlistfields, maxsubfields, explodecollections, zfield, gcp, order, tps,
  fieldmap, addfields, relaxedFieldNameMatch, forceNullable, unsetDefault,
  unsetFid, nomd, mo, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

src_datasource_name	Character. Input vector file.
dst_datasource_name	Character. Output vector file.
layer	Character. Layer to use.
f	Character. output file format name (default is ESRI Shapefile), some possible values are: "ESRI Shapefile", "TIGER", "MapInfo File", "GML", "PostgreSQL"
append	Logical. Append to existing layer instead of creating new
overwrite	Logical. Delete the output layer and recreate it empty.
update	Logical. Open existing output datasource in update mode rather than trying to create a new one
select	Character. Comma-delimited list of fields from input layer to copy to the new layer. A field is skipped if mentioned previously in the list even if the input layer has duplicate field names. (Defaults to all; any field is skipped if a subsequent field with same name is found.) Starting with OGR 2.0, geometry fields can also be specified in the list.
progress	Logical. (starting with GDAL 1.7.0) Display progress on terminal. Only works if input layers have the "fast feature count" capability.
sql	Character. SQL statement to execute. The resulting table/layer will be saved to the output.

dialect	Character. SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by passing OGRSQL. Starting with GDAL 1.10, the "SQLITE" dialect can also be used with any datasource.
where	Character. Attribute query (like SQL WHERE).
skipfailures	Logical. Continue after a failure, skipping the failed feature.
spat	Numeric. c(xmin,ymin,xmax,ymax) spatial query extents. Only features whose geometry intersects the extents will be selected. The geometries will not be clipped unless -clipsrc is specified
spat_srs	Character. srs_def. (OGR >= 2.0) Override spatial filter SRS.
geomfield	Character. (OGR >= 1.11) Name of the geometry field on which the spatial filter operates on.
dsco	Character. Dataset creation option (format specific).
lco	Character. Layer creation option (format specific).
nln	Character. Assign an alternate name to the new layer.
nlt	Character. Define the geometry type for the created layer. One of NONE, GEOMETRY, POINT, LINESTRING, POLYGON, GEOMETRYCOLLECTION, MULTIPOINT, MULTIPOLYGON or MULTILINESTRING. Add "25D" to the name to get 2.5D versions. Starting with GDAL 1.10, PROMOTE_TO_MULTI can be used to automatically promote layers that mix polygon or multipolygons to multipolygons, and layers that mix linestrings or multilinestrings to multilinestrings. Can be usefull when converting shapefiles to PostGIS (and other target drivers) that implements strict checks for geometry type.
dim	Numeric. (starting with GDAL 1.10) Force the coordinate dimension to val (valid values are 2 or 3). This affects both the layer geometry type, and feature geometries. Starting with GDAL 2.0, the value can be set to "layer_dim" to instruct feature geometries to be promoted to the coordinate dimension declared by the layer.
a_srs	Character. Assign an output SRS.
t_srs	Character. Reproject/transform to this SRS on output.
s_srs	Character. Override source SRS.
preserve_fid	Logical. Use the FID of the source features instead of letting the output driver to automatically assign a new one.
fid	Character. If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries. Note: if you want to select several features based on their feature id, you can also use the fact the 'fid' is a special field recognized by OGR SQL. So, '-where "fid in (1,3,5)"' would select features 1, 3 and 5.
oo	Character. "NAME=VALUE". (starting with GDAL 2.0) Input dataset open option (format specific).
doo	Character. "NAME=VALUE". (starting with GDAL 2.0) Destination dataset open option (format specific), only valid in -update mode.
gt	Numeric. group n features per transaction (default 200). Increase the value for better performance when writing into DBMS drivers that have transaction support.

`ds_transaction` Logical. (starting with GDAL 2.0) Force the use of a dataset level transaction (for drivers that support such mechanism), especially for drivers such as FileGDB that only support dataset level transaction in emulation mode.

`clipsrc` Character. [xmin ymin xmax ymax]|WKT|datasource|spat_extent: (starting with GDAL 1.7.0) clip geometries to the specified bounding box (expressed in source SRS), WKT geometry (POLYGON or MULTIPOLYGON), from a datasource or to the spatial extent of the -spat option if you use the spat_extent keyword. When specifying a datasource, you will generally want to use it in combination of the -clipsrclayer, -clipsrcwhere or -clipsrcsql options

`clipsrcsql` Character. Select desired geometries using an SQL query instead.

`clipsrclayer` Character. Select the named layer from the source clip datasource.

`clipsrcwhere` Character. Restrict desired geometries based on attribute query.

`clipdst` Character. (starting with GDAL 1.7.0) clip geometries after reprojection to the specified bounding box (expressed in dest SRS), WKT geometry (POLYGON or MULTIPOLYGON) or from a datasource. When specifying a datasource, you will generally want to use it in combination of the -clipdstlayer, -clipdstwhere or -clipdstsql options

`clipdstsql` Character. Select desired geometries using an SQL query instead.

`clipdstlayer` Character. Select the named layer from the destination clip datasource.

`clipdstwhere` Character. Restrict desired geometries based on attribute query.

`wrapdateline` Logical. (starting with GDAL 1.7.0) split geometries crossing the dateline meridian (long. = +/- 180deg).

`datelineoffset` Logical. (starting with GDAL 1.10) offset from dateline in degrees (default long. = +/- 10deg, geometries within 170deg to -170deg will be splited)

`simplify` Numeric. (starting with GDAL 1.9.0) distance tolerance for simplification. Note: the algorithm used preserves topology per feature, in particular for polygon geometries, but not for a whole layer.

`segmentize` Numeric. (starting with GDAL 1.6.0) maximum distance between 2 nodes. Used to create intermediate points

`fieldTypeToString` Character. (starting with GDAL 1.7.0) converts any field of the specified type to a field of type string in the destination layer. Valid types are : Integer, Real, String, Date, Time, DateTime, Binary, IntegerList, RealList, StringList. Special value All can be used to convert all fields to strings. This is an alternate way to using the CAST operator of OGR SQL, that may avoid typing a long SQL query.

`mapFieldType` Character. srctype|All=dsttype,... (starting with GDAL 2.0) converts any field of the specified type to another type. Valid types are : Integer, Integer64, Real, String, Date, Time, DateTime, Binary, IntegerList, Integer64List, RealList, StringList. Types can also include subtype between parenthesis, such as Integer(Boolean), Real(Float32), ... Special value All can be used to convert all fields to another type. This is an alternate way to using the CAST operator of OGR SQL, that may avoid typing a long SQL query. This is a generalization of -fieldTypeToString. Note that this does not influence the field types used by the source driver, and is only an afterwards conversion.

`unsetFieldWidth`
: Logical. (starting with GDAL 2.0) set field width and precision to 0.

`splitlistfields`
: Logical. (starting with GDAL 1.8.0) split fields of type StringList, RealList or IntegerList into as many fields of type String, Real or Integer as necessary.

`maxsubfields`
: Numeric. To be combined with -splitlistfields to limit the number of subfields created for each split field.

`explodecollections`
: Logical. (starting with GDAL 1.8.0) produce one feature for each geometry in any kind of geometry collection in the source file.

`zfield`
: Character. (starting with GDAL 1.8.0) Uses the specified field to fill the Z coordinate of geometries.

`gcp`
: Numeric. c(ungeoref_x,ungeoref_y,georef_x georef_y,elevation) (starting with GDAL 1.10.0) Add the indicated ground control point. This option may be provided multiple times to provide a set of GCPs.

`order`
: Numeric. (starting with GDAL 1.10.0) order of polynomial used for warping (1 to 3). The default is to select a polynomial order based on the number of GCPs.

`tps`
: Logical. (starting with GDAL 1.10.0) Force use of thin plate spline transformer based on available GCPs.

`fieldmap`
: Character. (starting with GDAL 1.10.0) Specifies the list of field indexes to be copied from the source to the destination. The (n)th value specified in the list is the index of the field in the target layer definition in which the n(th) field of the source layer must be copied. Index count starts at zero. There must be exactly as many values in the list as the count of the fields in the source layer. We can use the 'identity' setting to specify that the fields should be transferred by using the same order. This setting should be used along with the -append setting.

`addfields`
: Logical. (starting with GDAL 2.0) This is a specialized version of -append. Contrary to -append, -addfields has the effect of adding, to existing target layers, the new fields found in source layers. This option is usefull when merging files that have non-strictly identical structures. This might not work for output formats that don't support adding fields to existing non-empty layers.

`relaxedFieldNameMatch`
: Logical. (starting with GDAL 1.11) Do field name matching between source and existing target layer in a more relaxed way if the target driver has an implementation for it. [-relaxedFieldNameMatch] [-forceNullable]

`forceNullable`
: Logical. (starting with GDAL 2.0) Do not propagate not-nullable constraints to target layer if they exist in source layer.

`unsetDefault`
: Logical. (starting with GDAL 2.0) Do not propagate default field values to target layer if they exist in source layer.

`unsetFid`
: Logical. (starting with GDAL 2.0) Can be specify to prevent the new default behaviour that consists in, if the output driver has a FID layer creation option and we are not in append mode, to preserve the name of the source FID column and source feature IDs.

`nomd`
: Logical. (starting with GDAL 2.0) To disable copying of metadata from source dataset and layers into target dataset and layers, when supported by output driver.

mo — Character. "META-TAG=VALUE". (starting with GDAL 2.0) Passes a metadata key and value to set on the output dataset, when supported by output driver.

ignore.full_scan — Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.

verbose — Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'ogr2ogr' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/ogrinfo.html), or, in some cases, can use R vectors to achieve the same end.

PERFORMANCE HINTS

When writing into transactional DBMS (SQLite/PostgreSQL,MySQL, etc...), it might be beneficial to increase the number of INSERT statements executed between BEGIN TRANSACTION and COMMIT TRANSACTION statements. This number is specified with the -gt option. For example, for SQLite, explicitly defining -gt 65536 ensures optimal performance while populating some table containing many hundredth thousand or million rows. However, note that if there are failed insertions, the scope of -skipfailures is a whole transaction.

For PostgreSQL, the PG_USE_COPY config option can be set to YES for significantly insertion performance boot. See the PG driver documentation page.

More generally, consult the documentation page of the input and output drivers for performance hints.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

character

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/ogr2ogr.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
```

```
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
src_datasource_name <- system.file("external/tahoe_highrez_training.shp", package="gdalUtils")
dst_datasource_name <- paste(tempfile(),".shp",sep="")
ogrinfo(src_datasource_name,"tahoe_highrez_training")
# reproject the input to mercator
ogr2ogr(src_datasource_name,dst_datasource_name,t_srs="EPSG:3395",verbose=TRUE)
ogrinfo(dirname(dst_datasource_name),layer=remove_file_extension(basename(dst_datasource_name)))
}
```

---

ogrinfo *ogrinfo*

Description

R wrapper for ogrinfo: lists information about an OGR supported data source

Usage

```
ogrinfo(datasource_name, layer, ro, q, where, spat, geomfield, fid, sql,
  dialect, al, so, fields, geom, oo, nomd, listmdd, mdd, nocount, noextent,
  formats, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

datasource_name	Character. The data source to open. May be a filename, directory or other virtual name. See the OGR Vector Formats list for supported datasources.
layer	Character. One or more layer names may be reported.
ro	Logical. Open the data source in read-only mode.
q	Logical. Quiet verbose reporting of various information, including coordinate system, layer schema, extents, and feature count.
where	Character. An attribute query in a restricted form of the queries used in the SQL WHERE statement. Only features matching the attribute query will be reported.
spat	Numeric. c(xmin,ymin,xmax,ymax) The area of interest. Only features within the rectangle will be reported.
geomfield	Character. (OGR >= 2.0) Name of the geometry field on which the spatial filter operates on.
fid	Numeric. If provided, only the feature with this feature id will be reported. Operates exclusive of the spatial or attribute queries. Note: if you want to select several features based on their feature id, you can also use the fact the 'fid' is a special field recognized by OGR SQL. So, '-where "fid in (1,3,5)"' would select features 1, 3 and 5.
sql	Character. Execute the indicated SQL statement and return the result.

`dialect` Character. SQL dialect. In some cases can be used to use (unoptimized) OGR SQL instead of the native SQL of an RDBMS by passing OGRSQL. Starting with GDAL 1.10, the "SQLITE" dialect can also be used with any datasource.

`al` Logical. List all features of all layers (used instead of having to give layer names as arguments).

`so` Logical. Summary Only: supress listing of features, show only the summary information like projection, schema, feature count and extents.

`fields` Character. ("YES"|"NO") (starting with GDAL 1.6.0) If set to NO, the feature dump will not display field values. Default value is YES.

`geom` Character. ("YES"|"NO"|"SUMMARY") (starting with GDAL 1.6.0) If set to NO, the feature dump will not display the geometry. If set to SUMMARY, only a summary of the geometry will be displayed. If set to YES, the geometry will be reported in full OGC WKT format. Default value is YES.

`oo` Character. "NAME=VALUE". (starting with GDAL 2.0) Dataset open option (format specific).

`nomd` Logical. (starting with GDAL 2.0) Suppress metadata printing. Some datasets may contain a lot of metadata strings.

`listmdd` Logical. (starting with GDAL 2.0) List all metadata domains available for the dataset.

`mdd` Character. (starting with GDAL 2.0) Report metadata for the specified domain. "all" can be used to report metadata in all domains.

`nocount` Logical. (starting with GDAL 2.0) Suppress feature count printing.

`noextent` Logical. (starting with GDAL 2.0) Suppress spatial extent printing.

`formats` Logical. List the format drivers that are enabled.

`ignore.full_scan` Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.

`verbose` Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'ogrinfo' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/ogrinfo.html), or, in some cases, can use R vectors to achieve the same end.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

character

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/ogrinfo.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(valid_install)
{
datasource_name <- system.file("external/tahoe_highrez_training.shp", package="gdalUtils")
# Display all available formats:
# Command-line ogrinfo call:
# ogrinfo --formats
ogrinfo(formats=TRUE)

# Get info on an entire shapefile:
# ogrinfo tahoe_highrez_training.shp
ogrinfo(datasource_name)

# Get info on the layer of the shapefile:
# ogrinfo tahoe_highrez_training.shp tahoe_highrez_training
ogrinfo(datasource_name,"tahoe_highrez_training")
}
```

---

ogrlineref *ogrlineref*

Description

R wrapper for ogrlineref: create a linear reference

Usage

```
ogrlineref(help_general, progress, quiet, f, dsco, lco, create, l, ln, lf, p,
  pn, pm, pf, r, rn, o, on, of, s, get_pos, x, y, get_coord, m, get_subline, mb,
  me, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

help_general	Logical. Show the usage.
progress	Logical. Show progress.
quiet	Logical. Suppress all messages except errors and results.
f	Character. format_name. Select an output format name. The default is to create a shapefile.
dsco	Character. "NAME=VALUE". Dataset creation option (format specific).
lco	Character. "NAME=VALUE". Layer creation option (format specific).
create	Logical. Create the linear reference file (linestring of parts).
l	Character. src_line_datasource_name. The path to input linestring datasource (e.g. the road)
ln	Character. layer_name. The layer name in datasource
lf	Character. field_name. The field name of uniq values to separate the input lines (e.g. the set of roads)
p	Character. src_repers_datasource_name. The path to linear references points (e.g. the road mile-stones)
pn	Character. layer_name. The layer name in datasource.
pm	Character. pos_field_name.The field name of distances along path (e.g. mile-stones values)
pf	Character. field_name. The field name of uniq values to map input reference points to lines
r	Character. src_parts_datasource_name. The path to linear reference file
rn	Character. layer_name. The layer name in datasource.
o	Character. dst_datasource_name. The path to output linear reference file (linestring datasource)
on	Character. layer_name. The layer name in datasource.
of	Character. field_name. The field name for storing the uniq values of input lines
s	Numeric. step. The part size in linear units.
get_pos	Logical. Return linear referenced postion for input X, Y
x	Numeric. long. Input X coordinate
y	Numeric. lat. Input Y coordinate
get_coord	Logical. Return point on path for input linear distance.
m	Numeric. position. The input linear distance
get_subline	Logical. Return the portion of the input path from and to input linear positions
mb	Numeric. position. The input begin linear distance.
me	Numeric. position. The input end linear distance
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'ogrlineref' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/ogrlineref.html), or, in some cases, can use R vectors to achieve the same end.

The utility can be used for:

- create linear reference file from input data
- return the "linear referenced" distance for the projection of the input coordinates (point) on the path
- return the coordinates (point) on the path according to the "linear referenced" distance
- return the portion of the path according to the "linear referenced" begin and end distances

The ogrlineref program can be used to create a linear reference - a file containing a segments of special length (e.g. 1 km in reference units) and get coordinates, linear referenced distances or sublines (subpaths) from this file. The utility not required the M or Z values in geometry. The results can be stored in any OGR supported format. Also some information writed to the stdout.

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/ogrlineref.html

Examples

```
# No examples ATM for this function.
```

`ogrtindex` *ogrtindex*

Description

R wrapper for ogrtindex: creates a tileindex

Usage

```
ogrtindex(output_dataset, src_dataset, lnum, lname, f, tileindex,
  write_absolute_path, skip_different_projection, accept_different_schemas,
  output_Vector = FALSE, ignore.full_scan = TRUE, verbose = FALSE)
```

Arguments

output_dataset	Character. Output tile index.
src_dataset	Character. Input geospatial files.
lnum	Numeric. n. Add layer number 'n' from each source file in the tile index.
lname	Character. name. Add the layer named 'name' from each source file in the tile index.
f	Character. output_format. Select an output format name. The default is to create a shapefile.
tileindex	Character. file_name. The name to use for the dataset name. Defaults to LOCATION.
write_absolute_path	Logical. Filenames are written with absolute paths.
skip_different_projection	Logical. Only layers with same projection ref as layers already inserted in the tileindex will be inserted.
accept_different_schemas	Logical. By default ogrtindex checks that all layers inserted into the index have the same attribute schemas. If you specify this option, this test will be disabled. Be aware that resulting index may be incompatible with MapServer!
output_Vector	Logical. Return output output_dataset as a Spatial* object. Currently only works with f="ESRI Shapefile".
ignore.full_scan	Logical. If FALSE, perform a brute-force scan if other installs are not found. Default is TRUE.
verbose	Logical. Enable verbose execution? Default is FALSE.

Details

This is an R wrapper for the 'ogrtindex' function that is part of the Geospatial Data Abstraction Library (GDAL). It follows the parameter naming conventions of the original function, with some modifications to allow for more R-like parameters. For all parameters, the user can use a single character string following, precisely, the gdalinfo format (http://gdal.org/ogrtindex.html), or, in some cases, can use R vectors to achieve the same end.

The ogrtindex program can be used to create a tileindex - a file containing a list of the identities of a bunch of other files along with there spatial extents. This is primarily intended to be used with MapServer for tiled access to layers using the OGR connection type.

If no -lnum or -lname arguments are given it is assumed that all layers in source datasets should be added to the tile index as independent records.

If the tile index already exists it will be appended to, otherwise it will be created.

It is a flaw of the current ogrtindex program that no attempt is made to copy the coordinate system definition from the source datasets to the tile index (as is expected by MapServer when PROJECTION AUTO is in use).

This function assumes the user has a working GDAL on their system. If the "gdalUtils_gdalPath" option has been set (usually by gdal_setInstallation), the GDAL found in that path will be used. If nothing is found, gdal_setInstallation will be executed to attempt to find a working GDAL.

Value

NULL or SpatialPolygonsDataFrame

Author(s)

Jonathan A. Greenberg (<gdalUtils@estarcion.net>) (wrapper) and Frank Warmerdam (GDAL lead developer).

References

http://www.gdal.org/ogrtindex.html

Examples

```
# We'll pre-check to make sure there is a valid GDAL install.
# Note this isn't strictly neccessary, as executing the function will
# force a search for a valid GDAL install.
gdal_setInstallation()
valid_install <- !is.null(getOption("gdalUtils_gdalPath"))
if(require(rgdal) && valid_install)
{
tempindex <- tempfile(fileext=".shp")
src_dir <- system.file("external/", package="gdalUtils")
src_files <- list.files(src_dir,pattern=".shp",full.names=TRUE)
ogrtindex(output_dataset=tempindex,src_dataset=src_files,
accept_different_schemas=TRUE,output_Vector=TRUE)
}
```

---

qm *qm*

Description

Wraps an input in quotation marks.

Usage

```
qm(x)
```

Arguments

x	Character or Numeric.

Value

A character string that begins and ends with quotation marks.

Author(s)

Jonathan A. Greenberg

Examples

```
{
qm("Hi!")
qm(42)
}
```

`remove_file_extension` *remove_file_extension*

Description

Strips a file extension from a filename.

Usage

```
remove_file_extension(filename, extension_delimiter = ".")
```

Arguments

`filename`	Character. The input filename.
`extension_delimiter`	Character. The extension or extension delimiter (default ".") to remove.

Author(s)

Jonathan A. Greenberg <spatial.tools@estarcion.net>

Examples

```
myfilename="my.file.gri"
remove_file_extension(myfilename,".")
remove_file_extension(myfilename,".file.gri")
```

tahoe_highrez_training

Point and polygon files for use with gdalUtils

Description

Point and polygon files for use with gdalUtils

Author(s)

Jonathan A. Greenberg <gdalUtils@estarcion.net>

Examples

```
## Not run:
tahoe_highrez_training_polygons <- readOGR(
dsn=system.file("external", package="gdalUtils"),layer="tahoe_highrez_training")
spplot(tahoe_highrez_training_polygons,zcol="Class")
tahoe_highrez_training_points <- readOGR(
dsn=system.file("external", package="gdalUtils"),layer="tahoe_highrez_training_points")
spplot(tahoe_highrez_training_points,zcol="SPECIES")

## End(Not run)
```

tahoe_lidar_bareearth.tif

Lidar-derived bare earth digital elevation model from the Lake Tahoe Basin.

Description

Lidar-derived bare earth digital elevation model from the Lake Tahoe Basin.

Author(s)

Jonathan A. Greenberg <gdalUtils@estarcion.net>

Examples

```
## Not run:
tahoe_lidar_bareearth <-
raster(system.file("external/tahoe_lidar_bareearth.tif", package="gdalUtils"))
plot(tahoe_lidar_bareearth)

## End(Not run)
```

tahoe_lidar_highesthit.tif *Lidar-derived highest hit (aka canopy) digital elevation model from the Lake Tahoe Basin.*

Description

Lidar-derived highest hit (aka canopy) digital elevation model from the Lake Tahoe Basin.

Author(s)

Jonathan A. Greenberg <gdalUtils@estarcion.net>

Examples

```
## Not run:
tahoe_lidar_highesthit <-
raster(system.file("external/tahoe_lidar_highesthit.tif", package="gdalUtils"))
plot(tahoe_lidar_highesthit)

## End(Not run)
```

test_modis.hdf *MODIS HDF4 file*

Description

MODIS HDF4 file

Author(s)

Jonathan A. Greenberg <gdalUtils@estarcion.net>

Examples

```
## Not run:
gdalinfo(system.file("external/test_modis.hdf", package="gdalUtils"))

## End(Not run)
```

Index