

Package 'gdalUtilities'

July 18, 2019

Type Package

Title Wrappers for 'GDAL' Utilities Executables

Version 1.0.0

Date 2019-07-12

Author Joshua O'Brien

Maintainer Joshua O'Brien <joshmobrien@gmail.com>

Description R's 'sf' package ships with self-contained 'GDAL' executables, including a bare bones interface to several of the 'GDAL'-related utility programs collectively known as the 'GDAL' utilities. For each of those utilities, this package provides an R wrapper whose formal arguments closely mirror those of the 'GDAL' command line interface. All of the utilities operate on data stored in files and typically write their output to other files. As a result, processing data stored in any of R's more common spatial formats (i.e. those supported by the 'sp', 'sf', and 'raster' packages) will require first writing to disk, then processing with the package's wrapper functions before reading back into R.

License GPL (>= 2)

URL <https://github.com/JoshOBrien/gdalUtilities/>

BugReports <https://github.com/JoshOBrien/gdalUtilities/issues/>

Depends raster

Imports sf

Suggests rasterVis, RColorBrewer, testthat, gdalUtils

RoxygenNote 6.1.1

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2019-07-18 06:36:55 UTC

R topics documented:

gdalUtilities-package	2
gdalbuildvrt	3
gdaldem	5
gdalinfo	6
gdalwarp	7
gdal_grid	9
gdal_rasterize	11
gdal_translate	12
gRasterize	14
nearblack	15
ogr2ogr	16

gdalUtilities-package

Wrappers for 'GDAL' Utilities Executables

Description

R's 'sf' package ships with self-contained 'GDAL' executables, including a bare bones interface to several of the 'GDAL'-related utility programs collectively known as the 'GDAL' utilities. For each of those utilities, this package provides an R wrapper whose formal arguments closely mirror those of the 'GDAL' command line interface. All of the utilities operate on data stored in files and typically write their output to other files. As a result, processing data stored in any of R's more common spatial formats (i.e. those supported by the 'sp', 'sf', and 'raster' packages) will require first writing to disk, then processing with the package's wrapper functions before reading back into R.

Details

The DESCRIPTION file:

```

Package:      gdalUtilities
Type:         Package
Title:        Wrappers for 'GDAL' Utilities Executables
Version:      1.0.0
Date:         2019-07-12
Author:       Joshua O'Brien
Maintainer:   Joshua O'Brien <joshmobrien@gmail.com>
Description:  R's 'sf' package ships with self-contained 'GDAL' executables, including a bare bones interface to several of
License:      GPL (>= 2)
URL:          https://github.com/JoshOBrien/gdalUtilities/
BugReports:   https://github.com/JoshOBrien/gdalUtilities/issues/
Depends:      raster
Imports:      sf
Suggests:    rasterVis, RColorBrewer, testthat, gdalUtils
RoxygenNote: 6.1.1

```

Encoding: UTF-8

Index of help topics:

gRasterize	Fast rasterize for Spatial objects
gdalUtilities-package	Wrappers for 'GDAL' Utilities Executables
gdal_grid	Interface to GDAL's gdal_grid utility
gdal_rasterize	Interface to GDAL's gdal_rasterize utility
gdal_translate	Interface to GDAL's gdal_translate utility
gdalbuildvrt	Interface to GDAL's gdalbuildvrt utility
gdaldem	Interface to GDAL's gdaldem utility
gdalinfo	Interface to GDAL's gdalinfo utility
gdalwarp	Interface to GDAL's gdalwarp utility
nearblack	Interface to GDAL's nearblack utility
ogr2ogr	Interface to GDAL's ogr2ogr utility

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Joshua O'Brien

Maintainer: Joshua O'Brien <joshmobrien@gmail.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

```
## Optional simple examples of the most important functions
## Use \dontrun{} around code to be shown but not executed
```

gdalbuildvrt *Interface to GDAL's gdalbuildvrt utility*

Description

This function provides an interface mirroring that of the GDAL command-line app `gdalbuildvrt`. For a description of the utility and the arguments that it takes, see the documentation at <https://gdal.org/programs/gdalbuildvrt.html>.

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,
  dryrun = FALSE)
```

Arguments

gdalfile	Character vector supplying file paths to one or more input datasets.
output.vrt	Character. Path to output VRT file. Typically, output file will have suffix ".vrt".
...	Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.
tileindex, resolution, te, tr, tap, separate, b, sd	See the GDAL project's gdalbuildvrt documentation ¹ for details.
allow_projection_difference, q, addalpha, hidenodata	See the GDAL project's gdalbuildvrt documentation ² for details.
srcnodata, vrtnodata, a_srs, r, input_file_list, overwrite	See the GDAL project's gdalbuildvrt documentation ³ for details.
dryrun	Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

Examples

```
## Prepare file paths
td <- tempdir()
out_vrt <- file.path(td, "out.vrt")
layer1 <-
  system.file("external/tahoe_lidar_bareearth.tif",
              package = "gdalUtils")
layer2 <-
  system.file("external/tahoe_lidar_highesthit.tif",
              package = "gdalUtils")

## Build VRT and check that it works
gdalbuildvrt(gdalfile = c(layer1, layer2), output.vrt = out_vrt)
gdalinfo(out_vrt)
```

¹<https://gdal.org/programs/gdalbuildvrt.html>

²<https://gdal.org/programs/gdalbuildvrt.html>

³<https://gdal.org/programs/gdalbuildvrt.html>

 gdaldem

Interface to GDAL's gdaldem utility

Description

This function provides an interface mirroring that of the GDAL command-line app `gdaldem`. For a description of the utility and the arguments that it takes, see the documentation at <https://gdal.org/programs/gdaldem.html>.

Usage

```
gdaldem(mode, input_dem, output_map, ..., of, compute_edges, alg, b, co, q,
        z, s, az, alt, combined, multidirectional, p, trigonometric,
        zero_for_flat, color_text_file = character(0), alpha,
        exact_color_entry, nearest_color_entry, dryrun = FALSE)
```

Arguments

<code>mode</code>	Character, one of "hillshade", "slope", "color-relief", "TRI", "TPI", "roughness", indicating which of the available processing modes is to be used.
<code>input_dem</code>	Path to a GDAL-supported readable DEM datasource.
<code>output_map</code>	Character. Path to a GDAL-supported output file.
<code>...</code>	Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.
<code>of, compute_edges, alg, b, co, q, z, s, az, alt, combined</code>	See the GDAL project's <code>gdaldem</code> documentation ⁴ for details.
<code>multidirectional, p, trigonometric, zero_for_flat</code>	See the GDAL project's <code>gdaldem</code> documentation ⁵ for details.
<code>color_text_file, alpha, exact_color_entry, nearest_color_entry</code>	See the GDAL project's <code>gdaldem</code> documentation ⁶ for details.
<code>dryrun</code>	Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

⁴<https://gdal.org/programs/gdaldem.html>

⁵<https://gdal.org/programs/gdaldem.html>

⁶<https://gdal.org/programs/gdaldem.html>

Examples

```
## Prepare file paths
td <- tempdir()
in_dem <- system.file("extdata/maunga.tif", package = "gdalUtilities")
out_slope <- file.path(td, "slope.tif")
out_shade <- file.path(td, "shade.tif")
out_aspect <- file.path(td, "aspect.tif")

## Apply DEM processing
gdaldem("slope", in_dem, out_slope)
gdaldem("shade", in_dem, out_shade)
gdaldem("aspect", in_dem, out_aspect)

## View results
if(require(rasterVis)) {
  lp <- function(f) {
    levelplot(raster(f), main = substitute(f),
              margin = FALSE, colorkey = FALSE)
  }
  plot(lp(in_dem),      split = c(1,1,2,2))
  plot(lp(out_slope),  split = c(2,1,2,2), newpage = FALSE)
  plot(lp(out_shade),  split = c(1,2,2,2), newpage = FALSE)
  plot(lp(out_aspect), split = c(2,2,2,2), newpage = FALSE)
}
```

 gdalinfo

Interface to GDAL's gdalinfo utility

Description

This function provides an interface mirroring that of the GDAL command-line app `gdalinfo`. For a description of the utility and the arguments that it takes, see the documentation at <https://gdal.org/programs/gdalinfo.html>.

Usage

```
gdalinfo(datasetname, ..., json, mm, stats, approx_stats, hist, nogcp,
          nomd, norat, noct, checksum, listmdd, mdd, nofl, sd, proj4, oo, config,
          dryrun = FALSE)
```

Arguments

`datasetname` Path to a GDAL-supported readable datasource.

`...` Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.

json, mm, stats, approx_stats, hist, nogcp, nomd, norat, noct
 See the GDAL project's gdalinfo documentation⁷ for details.

checksum, listmdd, mdd, nofl, sd, proj4, oo, config
 See the GDAL project's gdalinfo documentation⁸ for details.

dryrun Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

Silently returns path to datasetname.

Author(s)

Joshua O'Brien

Examples

```
ff <- system.file("extdata/maunga.tif", package = "gdalUtilities")
gdalinfo(ff)
```

gdalwarp

Interface to GDAL's gdalwarp utility

Description

This function provides an interface mirroring that of the GDAL command-line app gdalwarp. For a description of the utility and the arguments that it takes, see the documentation at <https://gdal.org/programs/gdalwarp.html>.

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, co, cutline, cl, cwhere, csq1,
  cblend, crop_to_cutline, overwrite, nomd, cvmd, setci, oo, doo, config,
  dryrun = FALSE)
```

Arguments

srcfile Character. Path to a GDAL-supported readable datasource.

dstfile Character. Path to a GDAL-supported output file.

... Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.

⁷<https://gdal.org/programs/gdalinfo.html>

⁸<https://gdal.org/programs/gdalinfo.html>

`s_srs, t_srs, to, order, tps, rpc, geoloc, et, refine_gcps`
 See the GDAL project's [gdalwarp documentation](#)⁹ for details.

`te, te_srs, tr, tap, ts, ovr, wo, ot, wt, r, srcnodata, dstnodata`
 See the GDAL project's [gdalwarp documentation](#)¹⁰ for details.

`dstalpha, wm, multi, q, of, co, cutline, cl, cwhere, csq, cblend`
 See the GDAL project's [gdalwarp documentation](#)¹¹ for details.

`crop_to_cutline, overwrite, nomd, cvmd, setci, oo, doo, config`
 See the GDAL project's [gdalwarp documentation](#)¹² for details.

`dryrun` Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

Examples

```
## Prepare file paths
td <- tempdir()
in_tif <- file.path(td, "tahoe.tif")
gcp_tif <- file.path(td, "tahoe_gcp.tif")
out_tif <- file.path(td, "tahoe_warped.tif")

## Set up some ground control points, then warp
file.copy(system.file("extdata/tahoe.tif", package = "gdalUtilities"),
          in_tif)
## Four numbers: column, row, x-coord, y-coord
gcp <- matrix(c(100, 300, -119.93226, 39.28977, ## A
               0, 300, -119.93281, 39.28977, ## B
               100, 400, -119.93226, 39.28922, ## C
               0, 400, -119.93281, 39.28922, ## lower-left
               400, 0, -119.93067, 39.29136, ## upper-right
               400, 400, -119.93062, 39.28922, ## lower-right
               0, 0, -119.93281, 39.29141), ## upper-left
             ncol = 4, byrow = TRUE)

## Add ground control points. (For some reason, this drops CRS, so
## it needs to be explicitly given via `a_srs` argument.)
gdal_translate(in_tif, gcp_tif, gcp = gcp, a_srs = "EPSG:4326")
gdalwarp(gcp_tif, out_tif, r = "bilinear")
```

⁹<https://gdal.org/programs/gdalwarp.html>

¹⁰<https://gdal.org/programs/gdalwarp.html>

¹¹<https://gdal.org/programs/gdalwarp.html>

¹²<https://gdal.org/programs/gdalwarp.html>


```
## Check that it worked
if(require(rasterVis)) {
  r1 <- raster(in_tif)
  p1 <- levelplot(r1, margin = FALSE, colorkey = FALSE)
  r2 <- raster(out_tif)
  p2 <- levelplot(r2, margin = FALSE, colorkey = FALSE)
  plot(p1, split = c(1, 1, 2, 1))
  plot(p2, split = c(2, 1, 2, 1), newpage = FALSE)
}
```

gdal_grid

Interface to GDAL's gdal_grid utility

Description

This function provides an interface mirroring that of the GDAL command-line app `gdal_grid`. For a description of the utility and the arguments that it takes, see the documentation at https://gdal.org/programs/gdal_grid.html.

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, config,
  dryrun = FALSE)
```

Arguments

<code>src_datasource</code>	Character. Path to a GDAL-supported readable datasource.
<code>dst_filename</code>	Character. Path to a GDAL-supported output file.
<code>...</code>	Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.
<code>ot, of, txe, tye, outsize, a_srs, zfield, z_increase, z_multiply</code>	See the GDAL project's <code>gdal_grid</code> documentation ¹³ for details.
<code>a, spat, clipsrc, clipsrcsql, clipsrclayer, clipsrcwhere</code>	See the GDAL project's <code>gdal_grid</code> documentation ¹⁴ for details.
<code>l, where, sql, co, q, config</code>	See the GDAL project's <code>gdal_grid</code> documentation ¹⁵ for details.
<code>dryrun</code>	Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

¹³https://gdal.org/programs/gdal_grid.html

¹⁴https://gdal.org/programs/gdal_grid.html

¹⁵https://gdal.org/programs/gdal_grid.html

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

Examples

```
## Set up file paths
td <- tempdir()
dem_file <- file.path(td, "dem.csv")
vrt_header_file <- file.path(td, "tmp.vrt")
out_raster <- file.path(td, "tmp.tiff")

## Create file of points with x-, y-, and z-coordinates
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 = dem_file, row.names = FALSE)

## Prepare a matching VRT file
vrt_header <- c(
  '<OGRVRTDataSource>',
  '  <OGRVRTLayer name="dem">',
  paste0('    <SrcDataSource>', dem_file, '</SrcDataSource>'),
  '    <GeometryType>wkbPoint</GeometryType>',
  '    <GeometryField encoding="PointFromColumns" x="Easting" y="Northing" z="Elevation"/>',
  '  </OGRVRTLayer>',
  '</OGRVRTDataSource>'
)
cat(vrt_header, file = vrt_header_file, sep = "\n")

## Test it out
gdal_grid(src_datasource = vrt_header_file,
          dst_filename = out_raster,
          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")

## Check that it works
if(require(raster)) {
  plot(raster(out_raster))
  text(Northing ~ Easting, data = pts,
       labels = seq_len(nrow(pts)), cex = 0.7)
}
```

gdal_rasterize *Interface to GDAL's gdal_rasterize utility*

Description

This function provides an interface mirroring that of the GDAL command-line app `gdal_rasterize`. For a description of the utility and the arguments that it takes, see the documentation at https://gdal.org/programs/gdal_rasterize.html.

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, dryrun = FALSE)
```

Arguments

`src_datasource` Character. Path to a GDAL-supported readable datasource.

`dst_filename` Character. Path to a GDAL-supported output file.

`...` Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.

`b, i, at, burn, a, threeD, l, where, sql, dialect, of` See the GDAL project's `gdal_rasterize` documentation¹⁶ for details.

`a_srs, co, a_nodata, init, te, tr, tap, ts, ot, q` See the GDAL project's `gdal_rasterize` documentation¹⁷ for details.

`dryrun` Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

Examples

```
if(require(raster)) {
  ## Prepare file paths of example shapefile and template raster file
  vect_file <- system.file("external/lux.shp", package = "raster")
```

¹⁶https://gdal.org/programs/gdal_rasterize.html

¹⁷https://gdal.org/programs/gdal_rasterize.html

```

td <- tempdir()
rast_file <- file.path(td, "lux_rast.tif")

## Construct and save an appropriately sized 'empty' raster
SPDF <- shapefile(vect_file)
lonlatratio <- 1 / cospi(mean(coordinates(SPDF)[,2]) / 180)
rr <- raster(extent(SPDF),
             resolution = c(lonlatratio * 0.01, 0.01),
             crs = proj4string(SPDF))
writeRaster(rr, filename = rast_file) ## Warns that raster is empty

## Rasterize polygon using empty raster and check that it worked
gdal_rasterize(vect_file, rast_file, a = "ID_2")
plot(raster(rast_file))
}

```

gdal_translate

Interface to GDAL's gdal_translate utility

Description

This function provides an interface mirroring that of the GDAL command-line app `gdal_translate`. For a description of the utility and the arguments that it takes, see the documentation at https://gdal.org/programs/gdal_translate.html.

Usage

```

gdal_translate(src_dataset, dst_dataset, ..., ot, strict, of, b, mask,
              expand, outsize, tr, r, scale, exponent, unscale, srcwin, projwin,
              projwin_srs, srs, epo, eco, a_srs, a_ullr, a_nodata, mo, co, gcp, q, sds,
              stats, norat, oo, sd_index, config, dryrun = FALSE)

```

Arguments

`src_dataset` Character. Path to a GDAL-supported readable datasource.

`dst_dataset` Character. Path to a GDAL-supported output file.

`...` Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.

`ot, strict, of, b, mask, expand, outsize, tr, r, scale, exponent`
See the GDAL project's `gdal_translate` documentation¹⁸ for details.

`unscale, srcwin, projwin, projwin_srs, srs, epo, eco`
See the GDAL project's `gdal_translate` documentation¹⁹ for details.

`a_srs, a_ullr, a_nodata, mo, co, gcp, q, sds, stats, norat`
See the GDAL project's `gdal_translate` documentation²⁰ for details.

¹⁸https://gdal.org/programs/gdal_translate.html

¹⁹https://gdal.org/programs/gdal_translate.html

²⁰https://gdal.org/programs/gdal_translate.html

`oo, sd_index, config` See the GDAL project's `gdal_translate` documentation²¹ for details.

`dryrun` Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

Examples

```
## Prepare file paths
td <- tempdir()
in_raster <- file.path(td, "europe.tif")
out_raster <- file.path(td, "europe_small.tif")
file.copy(system.file("extdata/europe.tif", package = "gdalUtilities"),
          to = td)

## Shrink a tiff by 50% in both x and y dimensions
gdal_translate(in_raster, out_raster, outsize = c("50%", "50%"))

## Check that it worked
if(require(rasterVis)) {
  r1 <- raster(in_raster)
  r1[is.na(r1)] <- 0
  r1 <- as.factor(r1)
  rat <- levels(r1)[[1]]
  rat[["landcover"]] <- c("water", "land")
  levels(r1) <- rat
  p1 <- levelplot(r1, margin = FALSE, colorkey = FALSE,
                 col.regions = c("lightblue", "brown"))

  r2 <- raster(out_raster)
  r2[is.na(r2)] <- 0
  r2 <- as.factor(r2)
  rat <- levels(r2)[[1]]
  rat[["landcover"]] <- c("water", "land")
  levels(r2) <- rat
  p2 <- levelplot(r2, margin = FALSE, colorkey = FALSE,
                 col.regions = c("lightblue", "brown"))

  plot(p1, split = c(1, 1, 2, 1))
  plot(p2, split = c(2, 1, 2, 1), newpage = FALSE)
```

²¹https://gdal.org/programs/gdal_translate.html

}

`gRasterize`*Fast rasterize for Spatial objects*

Description

Rasterize `Spatial*` objects using `gdal_rasterize`.

Usage

```
gRasterize(SPDF, r, field, filename = "")
```

Arguments

<code>SPDF</code>	A <code>Spatial*</code> object to be rasterized.
<code>r</code>	A <code>Raster*</code> object to be used as the rasterization template.
<code>field</code>	Character. The name of the numeric column in <code>data.frame()</code> that will be written to the output <code>Raster*</code> object.
<code>filename</code>	Character. Output filename (optional). If none is supplied, the resulting raster will be stored <code>inMemory</code> (unless it is too large, as determined by a call to <code>canProcessInMemory(y, 3)</code>

Details

For a 1000-by-1000 raster, `gRasterize` is more than 6 times faster than `raster::rasterize`. For a 2000-by-2000 raster, it is almost 12 times faster (6 seconds vs. 70 seconds on my Windows laptop).

I've modeled `gRasterize` arguments and behavior on that of `rasterize`. Like `rasterize`, it takes a `filename=` argument which defaults to "" in which case (unless it's determined internally that `!canProcessInMemory(rstr, 3)`) the returned raster is `'inMemory'`. Otherwise, if the raster is too large or if a filename is supplied, it's returned `fromDisk`.

Internally, `gdal_rasterize` by default writes to a file, and only optionally returns an `R Raster` object (when its `output_Raster = TRUE`); to get the raster `'inMemory'`, I use `readAll` (after a check that it's really OK, memory-wise to do so).

Value

A `RasterLayer` object containing a rasterized version of `SPDF`.

Author(s)

Joshua O'Brien

Examples

```
SPDF <- shapefile(system.file("external/lux.shp", package="raster"))
## rr <- raster(extent(SPDF), ncol=100, nrow=100, crs=proj4string(SPDF))
llratio <- 1/cos(pi*mean(coordinates(SPDF)[,2])/180)
rr <- raster(extent(SPDF),
             resolution=c(llratio*0.01, 0.01),
             crs=proj4string(SPDF))

## An example using an integer-valued field
rInt <- gRasterize(SPDF, rr, field = "ID_2")
plot(rInt, col = RColorBrewer::brewer.pal(name = "Paired", 12))
plot(SPDF, lwd = 3, border = "grey30", add = TRUE)

## An example using a character-valued field
rFac <- gRasterize(SPDF, rr, field = "NAME_2")
rasterVis::levelplot(rFac)
```

nearblack

Interface to GDAL's nearblack utility

Description

This function provides an interface mirroring that of the GDAL command-line app `nearblack`. For a description of the utility and the arguments that it takes, see the documentation at <https://gdal.org/programs/nearblack.html>.

Usage

```
nearblack(infile, o = infile, ..., of, co, white, color, near, nb,
          setalpha, setmask, q, dryrun = FALSE)
```

Arguments

<code>infile</code>	Character. Path to a GDAL-supported readable datasource.
<code>o</code>	Optionally, a character string giving the path to a GDAL-supported output file. If not supplied, defaults to <code>codeinfile=</code> , indicating that the input file should be modified in place.
<code>...</code>	Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.
<code>of, co, white, color, near, nb, setalpha, setmask, q</code>	See the GDAL project's <code>nearblack</code> documentation ²² for details.
<code>dryrun</code>	Logical (default <code>FALSE</code>). If <code>TRUE</code> , instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

²²<https://gdal.org/programs/nearblack.html>

Value

Silently returns path to datasetname.

Author(s)

Joshua O'Brien

Examples

```
td <- tempdir()
a_rast <- file.path(td, "a.tif")
b_rast <- file.path(td, "b.tif")
file.copy(system.file("extdata/tahoe.tif", package = "gdalUtilities"),
          a_rast)
file.copy(system.file("extdata/tahoe.tif", package = "gdalUtilities"),
          b_rast)
nearblack(a_rast, b_rast, of = "GTiff")
nearblack(a_rast, of = "GTiff", near = 100)
```

 ogr2ogr

Interface to GDAL's ogr2ogr utility

Description

This function provides an interface mirroring that of the GDAL command-line app `ogr2ogr`. For a description of the utility and the arguments that it takes, see the documentation at <https://gdal.org/programs/ogr2ogr.html>.

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, dryrun = FALSE)
```

Arguments

`src_datasource_name`

Character. Path to a GDAL-supported readable datasource.

dst_datasource_name
Character. Path to a GDAL-supported output file.

... Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.

layer, f, append, overwrite, update, select, progress, sql, dialect
See the GDAL project's ogr2ogr documentation²³ for details.

where, skipfailures, spat, spat_srs, geomfield, dsco, lco, nln, nlt
See ogr2ogr documentation²⁴.

dim, a_srs, t_srs, s_srs, preserve_fid, fid, oo, doo, gt
See the See ogr2ogr documentation²⁵.

ds_transaction, clipsrc, clipsrcsql, clipsrclayer, clipsrcwhere
See ogr2ogr documentation²⁶.

clipdst, clipdstsql, clipdstlayer, clipdstwhere, wrapdateline
See ogr2ogr documentation²⁷.

datelineoffset, simplify, segmentize, fieldTypeToString
See See ogr2ogr documentation²⁸.

mapFieldType, unsetFieldWidth, splitlistfields, maxsubfields
See ogr2ogr documentation²⁹.

explodecollections, zfield, gcp, order, tps, fieldmap, addfields
See ogr2ogr documentation³⁰.

relaxedFieldNameMatch, forceNullable, unsetDefault, unsetFid
See ogr2ogr documentation³¹.

nomd, mo See ogr2ogr documentation³².

dryrun Logical (default FALSE). If TRUE, instead of executing the requested call to GDAL, the function will print the command-line call that would produce the equivalent output.

Value

None. Called instead for its side effect.

Author(s)

Joshua O'Brien

²³<https://gdal.org/programs/ogr2ogr.html>

²⁴<https://gdal.org/programs/ogr2ogr.html>

²⁵<https://gdal.org/programs/ogr2ogr.html>

²⁶<https://gdal.org/programs/ogr2ogr.html>

²⁷<https://gdal.org/programs/ogr2ogr.html>

²⁸<https://gdal.org/programs/ogr2ogr.html>

²⁹<https://gdal.org/programs/ogr2ogr.html>

³⁰<https://gdal.org/programs/ogr2ogr.html>

³¹<https://gdal.org/programs/ogr2ogr.html>

³²<https://gdal.org/programs/ogr2ogr.html>

Examples

```
## Prepare file paths
td <- tempdir()
in_shp <- system.file("external/lux.shp",
                      package = "raster")
out_merc <- file.path(td, "mercator.shp")
out_utm <- file.path(td, "utm.shp")

## Reproject to 'WGS 84/World Mercator'
## https://en.wikipedia.org/wiki/Mercator\_projection
ogr2ogr(in_shp, out_merc, t_srs = "EPSG:3395")
## Reproject to 'Lambert conformal conic projection'
## https://en.wikipedia.org/wiki/Lambert\_conformal\_conic\_projection
ogr2ogr(in_shp, out_utm, t_srs = "EPSG:42304")

if(require(raster)) {
  op <- par(mfcol = c(1,2))
  plot(shapefile(out_merc), main = "WGS 84",
        border = "darkgrey", col = gray.colors(12))
  plot(shapefile(out_utm), main = "LCC",
        border = "darkgrey", col = gray.colors(12))
  par(op)
}
```