

Package 'gdalUtilities'

May 8, 2026

Type Package

Title Wrappers for 'GDAL' Utilities Executables

Version 1.2.5

Date 2023-08-09

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 '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. The utilities operate on data stored in files and typically write their output to other files. Therefore, to process data stored in any of R's more common spatial formats (i.e. those supported by the 'sf' and 'terra' packages), first write them to disk, then process them with the package's wrapper functions before reading the outputted results back into R. GDAL function arguments introduced in GDAL version 3.5.2 or earlier are supported.

License GPL (>= 2)

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

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

Imports sf (>= 1.0-11)

Suggests terra, stars, RColorBrewer, testthat

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Repository CRAN

Date/Publication 2023-08-10 16:40:02 UTC

Contents

gdalUtilities-package	2
gdalbuildvrt	3
gdaldem	5
gdalinfo	7
gdalmdiminfo	8
gdalmdimtranslate	10
gdalUtilities-defunct	11
gdalwarp	12
gdal_grid	14
gdal_rasterize	17
gdal_translate	19
nearblack	21
ogr2ogr	23
Index	27

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 '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. The utilities operate on data stored in files and typically write their output to other files. Therefore, to process data stored in any of R's more common spatial formats (i.e. those supported by the 'sf' and 'terra' packages), first write them to disk, then process them with the package's wrapper functions before reading the outputted results back into R. GDAL function arguments introduced in GDAL version 3.5.2 or earlier are supported.

Details

The DESCRIPTION file:

```

Package:      gdalUtilities
Type:         Package
Title:        Wrappers for 'GDAL' Utilities Executables
Version:      1.2.5
Date:         2023-08-09
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 'C
License:      GPL (>= 2)
URL:          https://github.com/JoshOBrien/gdalUtilities/
BugReports:   https://github.com/JoshOBrien/gdalUtilities/issues/
Imports:      sf (>= 1.0-11)

```

Suggests: terra, stars, RColorBrewer, testthat
 RoxygenNote: 7.2.3
 Encoding: UTF-8

Index of help topics:

gdalUtilities-defunct	Defunct function(s) in the gdalUtilities package
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
gdalmdiminfo	Interface to GDAL's gdalmdiminfo utility
gdalmdimtranslate	Interface to GDAL's gdalmdimtranslate utility
gdalwarp	Interface to GDAL's gdalwarp utility
nearblack	Interface to GDAL's nearblack utility
ogr2ogr	Interface to GDAL's ogr2ogr utility

Wrappers for 'GDAL' Utilities Executables.

Author(s)

Joshua O'Brien

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

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,
    optim,
    q,
    addalpha,
    hidenodata,
    srcnodata,
    vrtnodata,
    ignore_srcmaskband,
    a_srs,
    r,
    oo,
    input_file_list,
    strict,
    non_strict,
    overwrite,
    config_options = character(0),
    dryrun = FALSE
)

```

Arguments

<code>gdalfile</code>	Character vector supplying file paths to one or more input datasets.
<code>output.vrt</code>	Character. Path to output VRT file. Typically, output file will have suffix ".vrt".
<code>...</code>	Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.
<code>tileindex, resolution, te, tr, tap, separate, b, sd</code>	See the GDAL project's gdalbuildvrt documentation for details.
<code>allow_projection_difference, q, optim, addalpha, hidenodata</code>	See the GDAL project's gdalbuildvrt documentation for details.
<code>srcnodata, vrtnodata, ignore_srcmaskband, a_srs, r, oo</code>	See the GDAL project's gdalbuildvrt documentation for details.
<code>input_file_list, strict, non_strict, overwrite</code>	See the GDAL project's gdalbuildvrt documentation for details.
<code>config_options</code>	A named character vector with GDAL config options, of the form <code>c(option1=value1, option2=value2)</code> . (See here for a complete list of supported config options.)
<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

Silently returns path to `output.vrt`.

Author(s)

Joshua O'Brien

Examples

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

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

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,
    igor,
    p,
    trigonometric,
    zero_for_flat,
    color_text_file = character(0),
    alpha,
    exact_color_entry,
    nearest_color_entry,
    config_options = character(0),
    dryrun = FALSE
  )

```

Arguments

mode	Character, one of "hillshade", "slope", "color-relief", "TRI", "TPI", "roughness", indicating which of the available processing modes is to be used.
input_dem	Path to a GDAL-supported readable DEM datasource.
output_map	Character. Path to a GDAL-supported output file.
...	Here, a placeholder argument that forces users to supply exact names of all subsequent formal arguments.
of, compute_edges, alg, b, co, q, z, s, az, alt, combined	See the GDAL project's gdaldem documentation for details.
multidirectional, igor, p, trigonometric, zero_for_flat	See the GDAL project's gdaldem documentation for details.
color_text_file, alpha, exact_color_entry, nearest_color_entry	See the GDAL project's gdaldem documentation for details.
config_options	A named character vector with GDAL config options, of the form c(option1=value1, option2=value2). (See here for a complete list of supported config options.)
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 output_map.

Author(s)

Joshua O'Brien

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(terra)) {
  op <- par(mfcol = c(1, 2))
  plot(rast(in_dem), main = "elevation")
  plot(rast(out_slope), main = "slope")
  plot(rast(out_shade), main = "hillshade")
  plot(rast(out_aspect), main = "aspect")
  par(op) ## Reset parameters to preexisting values
}
```

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,  
  nofl,  
  checksum,  
  proj4,  
  listmdd,  
  mdd,
```

```

    wkt_format,
    sd,
    oo,
    IF,
    dryrun = FALSE,
    config_options = character(0),
    quiet = 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 documentation for details.
nofl, checksum, proj4, listmdd, mdd, wkt_format, sd, oo, IF	See the GDAL project's gdalinfo documentation 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.
config_options	A named character vector with GDAL config options, of the form c(option1=value1, option2=value2). (See here for a complete list of supported config options.)
quiet	Logical (default FALSE). If TRUE, suppress printing of output to the console.

Value

Silently returns a character vector containing the information returned by the `gdalinfo` utility.

Author(s)

Joshua O'Brien

Examples

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

gdalmdiminfo

Interface to GDAL's gdalmdiminfo utility

Description

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

Usage

```
gdalmdiminfo(
  datasetname,
  ...,
  oo,
  arrayoption,
  detailed,
  nopretty,
  array,
  limit,
  stats,
  IF,
  dryrun = FALSE,
  config_options = character(0),
  quiet = 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.
oo, arrayoption, detailed, nopretty, array, limit, stats, IF	the GDAL project's gdalmdiminfo documentation 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.
config_options	A named character vector with GDAL config options, of the form c(option1=value1, option2=value2). (See here for a complete list of supported config options.)
quiet	Logical (default FALSE). If TRUE, suppress printing of output to the console.

Value

Silently returns a character vector containing the information in JSON format returned by the gdalmdiminfo utility.

Author(s)

Joshua O'Brien

Examples

```
ff <- system.file("nc/cropped.nc", package = "sf")
gdalmdiminfo(ff)
```

gdalmdimtranslate *Interface to GDAL's gdalmdimtranslate utility*

Description

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

Usage

```
gdalmdimtranslate(
    src_filename,
    dst_filename,
    ...,
    co,
    IF,
    of,
    array,
    group,
    subset,
    scaleaxes,
    oo,
    config_options = character(0),
    dryrun = FALSE
)
```

Arguments

<code>src_filename</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>co, IF, of, array, group, subset, scaleaxes, oo</code>	See the GDAL project's gdalmdimtranslate documentation for details.
<code>config_options</code>	A named character vector with GDAL config options, of the form <code>c(option1=value1, option2=value2)</code> . (See here for a complete list of supported config options.)
<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

Silently returns path to `dst_filename`.

Author(s)

Joshua O'Brien

Examples

```
## A simple dataset bundled with the sf package
FF <- system.file("nc/cropped.nc", package = "sf")
td <- tempdir()
out_tiff <- file.path(td, "out.tiff")
gdalinfo(FF)
gdalmdimtranslate(FF, out_tiff, array = "sst")
gdalinfo(out_tiff)

## A more interesting dataset bundled with the stars package
if(require(terra)) {
  FF <- system.file("nc/reduced.nc", package = "stars")
  gdalinfo(FF)
  td <- tempdir()
  out_1_tiff <- file.path(td, "out_1.tiff")
  gdalmdimtranslate(FF, out_1_tiff, array = "sst")
  plot(rast(out_1_tiff),
       main = "Sea Surface Temperature\n(2x2 degree cells)")
  ## Translate to a tiff, coarsen by a factor of 5
  out_2_tiff <- file.path(td, "out_2.tiff")
  gdalmdimtranslate(FF, out_2_tiff, array = "sst",
                   scaleaxes = "lon(5),lat(5)")
  plot(rast(out_2_tiff),
       main = "Sea Surface Temperature\n(10x10 degree cells)")
}
```

gdalUtilities-defunct *Defunct function(s) in the gdalUtilities package*

Description

These functions have been removed from this package.

Usage

```
gRasterize(...)
```

Arguments

... Function arguments

Details

gRasterize was removed due to its dependency on the **raster** package, on which **gdalUtilities** no longer Depends. The source for gRasterize may still be found (and sourced, using `devtools::source_gist()`) at <https://gist.github.com/JoshOBrien/7cf19b8b686e6d6230a78a1a9799883b>.

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,  
  ct,  
  to,  
  vshift,  
  novshift,  
  s_coord_epoch,  
  t_coord_epoch,  
  order,  
  tps,  
  rpc,  
  geoloc,  
  et,  
  refine_gcps,  
  te,  
  te_srs,  
  tr,  
  tap,  
  ts,  
  ovr,  
  wo,  
  ot,  
  wt,  
  r,  
  srcnodata,  
  dstnodata,
```

```

    srcalpha,
    nosrcalpha,
    dstalpha,
    wm,
    multi,
    q,
    IF,
    of,
    co,
    cutline,
    cl,
    cwhere,
    csql,
    cblend,
    crop_to_cutline,
    overwrite,
    nomd,
    cvmd,
    setci,
    oo,
    doo,
    config_options = character(0),
    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.
s_srs, t_srs, ct, to, vshift, novshift	See the GDAL project's gdalwarp documentation for details.
s_coord_epoch, t_coord_epoch, order, tps, rpc, geoloc, et	See the GDAL project's gdalwarp documentation for details.
refine_gcps, te, te_srs, tr, tap, ts, ovr, wo, ot, wt, r, srcnodata	See the GDAL project's gdalwarp documentation for details.
dstnodata, srcalpha, nosrcalpha, dstalpha, wm, multi, q, IF, of, co	See the GDAL project's gdalwarp documentation for details.
cutline, cl, cwhere, csql, cblend, crop_to_cutline, overwrite	See the GDAL project's gdalwarp documentation for details.
nomd, cvmd, setci, oo, doo	See the GDAL project's gdalwarp documentation for details.
config_options	A named character vector with GDAL config options, of the form c(option1=value1, option2=value2) (See here for a complete list of supported config options.)
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 dstfile.

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")

## Check that it worked
if(require(terra)) {
  op <- par(mfcol = c(1, 2))
  r1 <- plot(rast(in_tif), main = "Original raster")
  r2 <- plot(rast(out_tif), main = "Warped raster")
  par(op) ## Reset preexisting parameters
}
```

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,
  tr,
  outsize,
  a_srs,
  zfield,
  z_increase,
  z_multiply,
  a,
  spat,
  clipsrc,
  clipsrcsql,
  clipsrclayer,
  clipsrcwhere,
  l,
  where,
  sql,
  co,
  q,
  config_options = character(0),
  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.

`ot, of, txe, tye, tr, outsize, a_srs, zfield, z_increase, z_multiply`
See the GDAL project's [gdal_grid documentation](#) for details.

`a, spat, clipsrc, clipsrcsql, clipsrclayer, clipsrcwhere`
See the GDAL project's [gdal_grid documentation](#) for details.

`l, where, sql, co, q`
See the GDAL project's [gdal_grid documentation](#) for details.

`config_options` A named character vector with GDAL config options, of the form `c(option1=value1, option2=value2)`. (See [here](#) for a complete list of supported config options.)

`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 `dst_filename`.

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(requireNamespace("terra", quietly = TRUE)) {
  library(terra)
  plot(rast(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,  
    add,  
    l,  
    where,  
    sql,  
    dialect,  
    of,  
    a_srs,  
    to,  
    co,  
    a_nodata,  
    init,  
    te,  
    tr,  
    tap,  
    ts,  
    ot,  
    optim,  
    q,  
    config_options = character(0),  
    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, add, l, where, sql, dialect, of	See the GDAL project's gdal_rasterize documentation for details.
a_srs, to, co, a_nodata, init, te, tr, tap, ts, ot, optim, q	See the GDAL project's gdal_rasterize documentation for details.
config_options	A named character vector with GDAL config options, of the form c(option1=value1, option2=value2). (See here for a complete list of supported config options.)
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 `dst_filename`.

Author(s)

Joshua O'Brien

Examples

```
if(require(terra)) {
  ## Prepare file paths of example shapefile and template raster file
  vect_file <- system.file("ex/lux.shp", package = "terra")
  td <- tempdir()
  rast_file <- file.path(td, "lux_rast.tif")

  ## Construct and save an appropriately sized 'empty' raster
  LUX <- vect(vect_file)
  lonlatratio <- 1 / cospi(mean(geom(LUX)[, "y"]) / 180)
  rr <- rast(ext(LUX),
             resolution = c(lonlatratio * 0.01, 0.01),
             crs = crs(LUX), vals = NA)

  ## Note: this next line warns that raster is empty
  writeRaster(rr, filename = rast_file, overwrite = TRUE)

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

gdal_translate	<i>Interface to GDAL's gdal_translate utility</i>
----------------	---

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,  
    IF,  
    of,  
    b,  
    mask,  
    expand,  
    outsize,  
    tr,  
    r,  
    scale,  
    exponent,  
    unscale,  
    srcwin,  
    projwin,  
    projwin_srs,  
    srs,  
    epo,  
    eco,  
    a_srs,  
    a_coord_epoch,  
    a_ullr,  
    a_nodata,  
    a_scale,  
    a_offset,  
    colorinterp,  
    mo,  
    co,  
    nogcp,  
    gcp,  
    q,  
    sds,
```

```

stats,
noxmp,
norat,
oo,
sd_index,
config_options = character(0),
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, IF, of, b, mask, expand, outsize, tr, r, scale, exponent`
See the GDAL project's [gdal_translate documentation](#) for details.

`unscale, srcwin, projwin, projwin_srs, srs, epo, eco`
See the GDAL project's [gdal_translate documentation](#) for details.

`a_srs, a_coord_epoch, a_ullr, a_nodata, a_scale, a_offset`
See the GDAL project's [gdal_translate documentation](#) for details.

`colorinterp` Along with `colorinterp`, arguments named `colorinterp_bn`, where `bn` refers to the number of a band are also allowed. See the GDAL project's [gdal_translate documentation](#) for details.

`mo, co, nogcp, gcp, q, sds, stats, norat, noxmp, oo, sd_index`
See the GDAL project's [gdal_translate documentation](#) for details.

`config_options` A named character vector with GDAL config options, of the form `c(option1=value1, option2=value2)`. (See [here](#) for a complete list of supported config options.)

`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 `dst_dataset`.

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(terra)) {

  r1 <- rast(in_raster)
  r1[is.na(r1)] <- 0
  r1 <- as.factor(r1)
  rat <- levels(r1)[[1]]
  rat[["landcover"]] <- c("water", "land")
  levels(r1) <- rat

  r2 <- rast(out_raster)
  r2[is.na(r2)] <- 0
  r2 <- as.factor(r2)
  rat <- levels(r2)[[1]]
  rat[["landcover"]] <- c("water", "land")
  levels(r2) <- rat

  op <- par(mfcol = c(1, 2))
  plot(r1, col = c("lightblue", "brown"), legend = FALSE)
  plot(r2, col = c("lightblue", "brown"), legend = FALSE)
  par(op) ## Reset pre-existing parameters
}
```

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,
  white,
  color,
  near,
  nb,
  setalpha,
```

```

    setmask,
    q,
    co,
    config_options = character(0),
    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, white, color, near, nb, setalpha, setmask, q, co</code>	See the GDAL project's nearblack documentation for details.
<code>config_options</code>	A named character vector with GDAL config options, of the form <code>c(option1=value1, option2=value2)</code> . (See here for a complete list of supported config options.)
<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

Silently returns path to `o`.

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", near = 150)

## Check that it worked
if(require(terra)) {
  op <- par(mfcol = c(1, 2))
  r1 <- plot(rast(a_rast))
  r2 <- plot(rast(b_rast))
  par(op) ## Reset preexisting parameters
}

```

`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,  
    ct,  
    preserve_fid,  
    fid,  
    limit,  
    oo,  
    doo,  
    gt,  
    ds_transaction,  
    clipsrc,  
    clipsrcsql,
```

```
    clipsrclayer,  
    clipsrcwhere,  
    clipdst,  
    clipdstsql,  
    clipdstlayer,  
    clipdstwhere,  
    wrapdateline,  
    datelineoffset,  
    simplify,  
    segmentize,  
    makevalid,  
    fieldTypeToString,  
    unsetFieldWidth,  
    mapFieldType,  
    fieldmap,  
    splitlistfields,  
    maxsubfields,  
    resolveDomains,  
    explodecollections,  
    zfield,  
    gcp,  
    order,  
    tps,  
    s_coord_epoch,  
    t_coord_epoch,  
    a_coord_epoch,  
    addfields,  
    unsetFid,  
    emptyStrAsNull,  
    relaxedFieldNameMatch,  
    forceNullable,  
    unsetDefault,  
    nomd,  
    mo,  
    noNativeData,  
    config_options = character(0),  
    dryrun = FALSE  
)
```

Arguments

<code>src_datasource_name</code>	Character. Path to a GDAL-supported readable datasource.
<code>dst_datasource_name</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.

layer, f, append, overwrite, update, select, progress, sql, dialect
 See the GDAL project's [ogr2ogr documentation](#) for details.

where, skipfailures, spat, spat_srs, geomfield, dsco, lco, nln, nlt
 See [ogr2ogr documentation](#).

dim, a_srs, t_srs, s_srs, ct, preserve_fid, fid, limit, oo, doo, gt
 See the See [ogr2ogr documentation](#).

ds_transaction, clipsrc, clipsrcsql, clipsrclayer, clipsrcwhere
 See [ogr2ogr documentation](#).

clipdst, clipdstsql, clipdstlayer, clipdstwhere, wrapdateline
 See [ogr2ogr documentation](#).

datelineoffset, simplify, segmentize, makevalid, addfields
 See See [ogr2ogr documentation](#).

fieldmap, splitlistfields, maxsubfields
 See [ogr2ogr documentation](#).

resolveDomains, explodecollections, zfield, gcp, order, tps
 See [ogr2ogr documentation](#).

s_coord_epoch, t_coord_epoch, a_coord_epoch
 See [ogr2ogr documentation](#).

unsetFid, emptyStrAsNull, relaxedFieldNameMatch, forceNullable
 See See [ogr2ogr documentation](#).

unsetDefault, fieldTypeToString, unsetFieldWidth, mapFieldType
 See [ogr2ogr documentation](#).

nomd, mo, noNativeData
 See [ogr2ogr documentation](#).

config_options A named character vector with GDAL config options, of the form c(option1=value1, option2=value2). (See [here](#) for a complete list of supported config options.)

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 dst_datasource_name.

Author(s)

Joshua O'Brien

Examples

```
## Prepare file paths
td <- tempdir()
lux <- system.file("ex/lux.shp", package = "terra")
lux_merc <- file.path(td, "mercator.shp")
lux_lcc <- file.path(td, "lcc.shp")

## Reproject to 'WGS 84/World Mercator'
```

```
## https://en.wikipedia.org/wiki/Mercator\_projection
ogr2ogr(lux, lux_merc, t_srs = "EPSG:3395", overwrite = TRUE)
## Reproject to a Canadian 'Lambert conformal conic projection'
## https://en.wikipedia.org/wiki/Lambert\_conformal\_conic\_projection
ogr2ogr(lux, lux_lcc, t_srs = "EPSG:3347", overwrite = TRUE)

if(require(terra)) {
  op <- par(mfcol = c(1,2))
  plot(vect(lux_merc), main = "WGS 84",
        border = "darkgrey", col = gray.colors(12))
  plot(vect(lux_lcc), main = "LCC",
        border = "darkgrey", col = gray.colors(12))
  par(op)
}
```

Index

* package

- gdalUtilities-package, [2](#)

- gdal_grid, [14](#)
- gdal_rasterize, [17](#)
- gdal_translate, [19](#)
- gdalbuildvrt, [3](#)
- gdaldem, [5](#)
- gdalinfo, [7](#)
- gdalmdiminfo, [8](#)
- gdalmdimtranslate, [10](#)
- gdalUtilities (gdalUtilities-package), [2](#)
- gdalUtilities-defunct, [11](#)
- gdalUtilities-package, [2](#)
- gdalwarp, [12](#)
- gRasterize (gdalUtilities-defunct), [11](#)

- nearblack, [21](#)

- ogr2ogr, [23](#)