Package 'gpkg'

February 20, 2024

Type Package

Title Utilities for the Open Geospatial Consortium 'GeoPackage' Format

Version 0.0.8

Maintainer Andrew Brown <brown.andrewg@gmail.com>

Description Build Open Geospatial Consortium 'GeoPackage' files (<https://www.geopackage.org/>). 'GDAL' utilities for reading and writing spatial data are provided by the 'terra' package. Additional 'GeoPackage' and 'SQLite' features for attributes and tabular data are implemented with the 'RSQLite' package.

URL https://humus.rocks/gpkg/, https://github.com/brownag/gpkg

BugReports https://github.com/brownag/gpkg/issues

Imports utils, methods, DBI

Suggests RSQLite, terra (>= 1.6), tinytest, dplyr, dbplyr, knitr, rmarkdown

License CC0

Depends R (>= 3.1.0)

RoxygenNote 7.2.3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Andrew Brown [aut, cre]

Repository CRAN

Date/Publication 2024-02-20 02:00:02 UTC

R topics documented:

---

geopackage geopackage *Constructors*

Description

geopackage Constructors

Usage

```
geopackage(x, ...)

## S3 method for class 'list'
geopackage(x, dsn = NULL, connect = FALSE, ...)

## S3 method for class 'missing'
geopackage(x, connect = FALSE, pattern = "Rgpkg", tmpdir = tempdir(), ...)

## S3 method for class 'SQLiteConnection'
geopackage(x, connect = FALSE, ...)

## S3 method for class 'geopackage'
geopackage(x, ...)

## S3 method for class 'character'
geopackage(x, connect = FALSE, ...)
```

Arguments

x	list of SpatVectorProxy, SpatRaster, data.frame; or a character containing path to a GeoPackage file; or an SQLiteConnection to a GeoPackage. If missing, a temporary file with .gpkg extension is created in `tempdir`.
...	Additional arguments [not currently used]
dsn	Path to GeoPackage File (may not exist)
connect	Connect to database and store connection in result? Default: `FALSE`
pattern	used only when `x` is missing (creating temporary file GeoPackage), passed to `tempfile()`; default `"Rgpkg"`
tmpdir	used only when `x` is missing (creating temporary file GeoPackage), passed to `tempfile()`; default `tempdir()`

Value

A *geopackage* object

---

gpkg_2d_gridded_coverage_ancillary

Get `gpkg_2d_gridded_coverage_ancillary` *Table*

Description

Get `gpkg_2d_gridded_coverage_ancillary` Table

Usage

```
gpkg_2d_gridded_coverage_ancillary(x)
```

Arguments

x	A *geopackage* object, path to a GeoPackage or an *SQLiteConnection*

Value

a data.frame containing columns `id`, `tile_matrix_set_name`, `datatype`, `scale`, `offset`, `precision`, `data_null`, `grid_cell_encoding`, `uom`, `field_name`, `quantity_definition`

gpkg_add_contents *Add, Remove, Update and Create* gpkg_contents *table and records*

Description

gpkg_add_contents(): Add a record to gpkg_contents

gpkg_update_contents(): Add and remove gpkg_contents records to match existing tables

gpkg_delete_contents(): Delete a record from gpkg_contents based on table name

gpkg_create_contents(): Create an empty gpkg_contents table

Usage

```
gpkg_add_contents(
  x,
  table_name,
  description = "",
  template = NULL,
  query_string = FALSE
)

gpkg_update_contents(x)

gpkg_delete_contents(x, table_name, query_string = FALSE)

gpkg_create_contents(x, query_string = FALSE)
```

Arguments

x	A *geopackage*
table_name	Name of table to add or remove record for in *gpkg_contents*
description	Default ""
template	Default NULL uses global EPSG:4326 with bounds -180,-90:180,90
query_string	*logical*. Return SQLite statement rather than executing it? Default: FALSE

Value

logical. TRUE on successful execution of SQL statements.

gpkg_add_metadata_extension *Add 'Metadata' extension*

Description

Adds the "Metadata" extension tables.

Usage

```
gpkg_add_metadata_extension(x)
```

Arguments

x	a geopackage

Value

0 (invisible). Called for side effects.

gpkg_add_relatedtables_extension *Add 'Related Tables' extension*

Description

Adds the "Related Tables" extension tables.

Usage

```
gpkg_add_relatedtables_extension(x)
```

Arguments

x	a geopackage

Value

0 (invisible). Called for side effects.

gpkg_connect *Create SQLite Connection to GeoPackage*

Description

Method for creating and connecting SQLiteConnection object stored within geopackage object.

Usage

```
gpkg_connect(x)

## S3 method for class 'geopackage'
gpkg_connect(x)

## S3 method for class 'character'
gpkg_connect(x)

gpkg_is_connected(x)

## S3 method for class 'geopackage'
gpkg_is_connected(x)

gpkg_disconnect(x)

## S3 method for class 'geopackage'
gpkg_disconnect(x)

## S3 method for class 'SQLiteConnection'
gpkg_disconnect(x)
```

Arguments

x	A *geopackage* or *SQLiteConnection* object

Details

The S3 method for geopackage objects does not require the use of assignment to create an object containing an active SQLiteConnection. e.g. gpkg_connect(g) connects the existing geopackage object g

Value

A DBIConnection (SQLiteConnection) object. NULL on error.

If x is *geopackage*, the disconnected object is returned. If x is a *SQLiteConnection*, logical (TRUE if successfully disconnected).

gpkg_contents *Get* gpkg_contents *or* gpkg_ogr_contents *Table*

Description

These functions provide convenient access to the contents of the standard GeoPackage tables of the same name.

Usage

```
gpkg_contents(x, create = FALSE)

gpkg_ogr_contents(x)
```

Arguments

x	A *geopackage* object, path to a GeoPackage or an *SQLiteConnection*
create	Create table gpkg_contents if does not exist? Default: "

Value

gpkg_contents(): a *data.frame* containing columns table_name, data_type, identifier, description, last_change, min_x, min_y, max_x, max_y, srs_id

gpkg_ogr_contents(): a *data.frame* containing columns table_name and feature_count.

gpkg_create_dummy_features *Create a Dummy Feature Dataset in a GeoPackage*

Description

This function creates a minimal (empty) feature table and gpkg_geometry_columns table entry.

Usage

```
gpkg_create_dummy_features(x, table_name = "dummy_feature", values = NULL)
```

Arguments

x	A *geopackage* object
table_name	A table name; default "dummy_feature"
values	Values to use for new table. Defaults to default geometry name ("geom"), with generic GEOMETRY data type, with no spatial reference system.

Details

This is a workaround so that `gpkg_vect()` (via `terra::vect()`) will recognize a GeoPackage as containing geometries and allow for use of OGR query utilities. The "dummy table" is not added to `gpkg_contents` and you should not try to use it for anything. The main purpose is to be able to use `gpkg_vect()` and `gpkg_ogr_query()` on a GeoPackage that contains only gridded and/or attribute data.

Value

logical. `TRUE` on success.

See Also

`gpkg_vect()` `gpkg_ogr_query()`

`gpkg_create_spatial_view` *Create a Spatial View*

Description

Create a Spatial View

Usage

```
gpkg_create_spatial_view(
  g,
  viewname,
  viewquery,
  geom_column = "geom",
  geometry_type_name = "GEOMETRY",
  spatialite_computed = FALSE,
  data_type = "features",
  srs_id = 4326,
  z = 0,
  m = 0
)
```

Arguments

`g`	a geopackage
`viewname`	*character*. Name of view.
`viewquery`	*character*. Query for view contents.
`geom_column`	*character*. Column name of view geometry. Default: `"geom"`
`geometry_type_name`	*character*. View geometry type. Default: `"GEOMETRY"`

spatialite_computed
: *logical*. Register definition of geom_column as the result of a Spatialite spatial function via "gdal_spatialite_computed_geom_column" extension. Default: FALSE

data_type
: *character*. View data type. Default "features"

srs_id
: *integer*. Spatial Reference System ID. Default: 4326 (WGS84)

z
: *integer*. Default: 0

m
: *integer*. Default: 0

Value

integer. Returns 1 if a new record in gpkg_geometry_columns is successfully made.

---

gpkg_creation_options *GeoPackage Creation Options*

Description

GeoPackage Creation Options

Usage

```
gpkg_creation_options
```

Format

An object of class data.frame with 32 rows and 4 columns.

Source

GDAL - GPKG – GeoPackage raster, GDAL - GPKG – GeoPackage vector

---

gpkg_execute *Execute an SQL statement in a GeoPackage*

Description

Execute an SQL statement in a GeoPackage

Usage

```
gpkg_execute(x, statement, ..., silent = FALSE)
```

Arguments

x	A *geopackage* object
statement	An SQLite statement
...	Additional arguments to RSQLite::dbExecute()
silent	Used to suppress error messages, passed to try(). Default: FALSE.

Value

Invisible result of RSQLite::dbExecute(); or try-error on error.

---

gpkg_list_contents *List Tables Registered in a GeoPackage* gpkg_contents

---

Description

Get a vector of grid, feature and attribute table names registered in GeoPackage.

Usage

```
gpkg_list_contents(x, ogr = FALSE)
```

Arguments

x	A *geopackage* object, path to a GeoPackage or an *SQLiteConnection*
ogr	Intersect gpkg_contents table name result with OGR tables that are listed in gpkg_ogr_contents? Default: FALSE

Value

character. Vector of grid, feature and attribute table names registered in GeoPackage.

See Also

gpkg_contents() gpkg_list_tables()

gpkg_list_tables *List Tables in a GeoPackage*

Description

List Tables in a GeoPackage

Usage

```
gpkg_list_tables(x)
```

Arguments

x	A *geopackage* object, path to a GeoPackage or an *SQLiteConnection*

Value

a character vector with names of all tables and views in the database

gpkg_query *Query a GeoPackage for Tabular Result*

Description

`gpkg_ogr_query()`: an alias for `gpkg_query(..., ogr=TRUE)`

Usage

```
gpkg_query(x, query, ogr = FALSE, ...)

gpkg_ogr_query(x, query, ...)
```

Arguments

x	A *geopackage* object
query	*character*. An SQLite/Spatialite/GeoPackage query. The `query` argument is forwarded to `sql` argument when `ogr=TRUE`.
ogr	*logical*. Use the OGR query interface (via `terra::query()`). See details. Default: `FALSE` uses 'RSQLite' driver instead of 'terra'.
...	Additional arguments to `terra::query()` (such as `start`, `n`, `vars`, `where`, `extent`, `filter`) are passed when `ogr=TRUE` (or using alias `gpkg_ogr_query()`). Otherwise not used.

Details

The GeoPackage driver supports OGR attribute filters. Provide filters in the SQLite dialect, as they will be executed directly against the database. If Spatialite is used, a recent version (4.2.0) is needed and cast operators are required to transform GeoPackage geometries to Spatialite geometries. A variety of SQL functions are available, see: https://gdal.org/drivers/vector/gpkg.html#sql-functions

Value

a *data.frame* result of `RSQLite::dbGetQuery()` or *SpatVector* result from `terra::query()`.

---

gpkg_read *Read data from a GeoPackage*

Description

Experimental: This function is being evaluated for its scope compared to other more general functions that perform similar operations (i.e. `gpkg_tables()`).

Usage

```
gpkg_read(x, connect = FALSE, quiet = TRUE)
```

Arguments

`x`	Path to GeoPackage
`connect`	Connect to database and store connection in result? Default: `FALSE`
`quiet`	Hide printing of gdalinfo description to stdout. Default: `TRUE`

Value

A *geopackage* object (list containing tables, grids and vector data)

---

gpkg_source *Get Source File of a* geopackage *object*

Description

Get Source File of a *geopackage* object

Usage

```
gpkg_source(x)

## S3 method for class 'geopackage'
gpkg_source(x)
```

Arguments

x	A *geopackage* object

Value

character file path

---

gpkg_sqlite_tables *GeoPackage Dataset*

Description

GeoPackage Dataset

GeoPackage SQLite Tables

Usage

```
gpkg_sqlite_tables
```

Format

a `data.frame` with 1 column (`"table_name"`) and 10 rows

Source

GDAL - GPKG – GeoPackage raster, GDAL - GPKG – GeoPackage vector

---

gpkg_tables *Get Tables from a* geopackage *object*

Description

Get Tables from a *geopackage* object

Usage

```
gpkg_tables(x, collect = FALSE, pragma = FALSE)

## S3 method for class 'geopackage'
gpkg_tables(x, collect = FALSE, pragma = FALSE)
```

Arguments

x	A *geopackage* object
collect	Default: FALSE. Should tables be materialized as 'data.frame' objects in memory? (i.e. not "lazy") Default: FALSE; if TRUE 'dbplyr' is not required. Always TRUE for pragma=TRUE (pragma information are always "collected").
pragma	Default: FALSE. Use gpkg_table_pragma() instead of gpkg_table()? The former does not require 'dbplyr'.

Value

a list of SpatVectorProxy, SpatRaster, data.frame (lazy tbl?)

---

gpkg_table_pragma *Lazy Access to Tables by Name*

Description

gpkg_table_pragma(): Get information on a table in a GeoPackage (without returning the whole table).

gpkg_table(): access a specific table (by name) and get a "lazy" tibble object referencing that table

gpkg_collect(): alias for gpkg_table(..., collect=TRUE)

gpkg_tbl(): shorthand for gpkg_table(..., collect=FALSE)(default) that always returns a 'dplyr' object.

Usage

```
gpkg_table_pragma(x, table_name = NULL, ...)

## S3 method for class 'character'
gpkg_table_pragma(x, table_name = NULL, ...)

## S3 method for class 'SQLiteConnection'
gpkg_table_pragma(x, table_name, ...)

## S3 method for class 'geopackage'
gpkg_table_pragma(x, table_name = NULL, ...)

gpkg_table(x, table_name, collect = FALSE, query_string = FALSE, ...)

## Default S3 method:
gpkg_table(x, table_name, collect = FALSE, query_string = FALSE, ...)

gpkg_collect(x, table_name, query_string = FALSE, ...)
```

```
gpkg_tbl(x, table_name, ...)

gpkg_rast(x, table_name = NULL, ...)

gpkg_vect(x, table_name, ...)
```

Arguments

x	A *geopackage* object or character path to GeoPackage file
table_name	*character*. One or more table names; for gpkg_table_pragma() if table_name=NULL returns a record for each table. gpkg_table() requires table_name be specified
...	Additional arguments. In gpkg_table() arguments in ... are passed to dplyr::tbl(). For gpkg_table_pragma(), ... arguments are (currently) not used. For gpkg_rast() additional arguments are passed to terra::rast(). For gpkg_vect() additional arguments (such as proxy=TRUE) are passed to terra::vect().
collect	*logical*. Materialize a data.frame object in memory? Default: FALSE requires 'dbplyr' package. TRUE uses 'RSQLite'.
query_string	*logical*. Return SQLite query rather than executing it? Default: FALSE

Value

gpkg_table(): A 'dbplyr' object of class *tbl_SQLiteConnection*

gpkg_collect(): An object of class *data.frame*

gpkg_tbl(): An object of class *tbl_SQLiteConnection*

gpkg_rast(): A 'terra' object of class *SpatRaster*

gpkg_vect(): A 'terra' object of class *SpatVector* (may not contain geometry columns)

Examples

```
tf <- tempfile(fileext = ".gpkg")

r <- terra::rast(system.file("extdata", "dem.tif", package = "gpkg"))

gpkg_write(r,
           destfile = tf,
           RASTER_TABLE = "DEM1",
           FIELD_NAME = "Elevation")

gpkg_write(r,
           destfile = tf,
           append = TRUE,
           RASTER_TABLE = "DEM2",
           FIELD_NAME = "Elevation")

g <- geopackage(tf)
```

```
# inspect gpkg_contents table
gpkg_table(g, "gpkg_contents")

gpkg_vect(g, "gpkg_contents")

# materialize a data.frame from gpkg_2d_gridded_tile_ancillary
library(dplyr, warn.conflicts = FALSE)

gpkg_table(g, "gpkg_2d_gridded_tile_ancillary") %>%
  dplyr::filter(tpudt_name == "DEM2") %>%
  dplyr::select(mean, std_dev) %>%
  dplyr::collect()

gpkg_disconnect(g)
```

`gpkg_tile_set_data_null` *Set* `data_null` *Metadata for a GeoPackage Tile Dataset*

Description

Set `data_null` Metadata for a GeoPackage Tile Dataset

Usage

```
gpkg_tile_set_data_null(x, name, value, query_string = FALSE)
```

Arguments

`x`	A *geopackage* object, path to a GeoPackage or an *SQLiteConnection*
`name`	character. Tile matrix set name(s) (`tile_matrix_set_name`)
`value`	numeric. Value to use as "NoData" (`data_null` value)
`query_string`	logical. Return SQLite query rather than executing it? Default: `FALSE`

Value

logical. `TRUE` if number of `data_null` records updated is greater than `0`.

gpkg_update_table *Update a Table by Name*

Description

For a given table, set column `updatecol` to scalar `updatevalue` where column `wherecol` is in vector `wherevector`.

Usage

```
gpkg_update_table(
  x,
  table_name,
  updatecol,
  updatevalue,
  wherecol = NULL,
  wherevector = NULL,
  query_string = FALSE
)
```

Arguments

`x`	A *geopackage* object, path to a GeoPackage or an *SQLiteConnection*.
`table_name`	*character*. Table name.
`updatecol`	*character*. Column to update.
`updatevalue`	*character*, *numeric*, etc.; A scalar value to set.
`wherecol`	*character*. Column used to constrain update.
`wherevector`	*character*, *numeric*, etc.; Vector of values where update should be made.
`query_string`	*logical*. Return SQLite query rather than executing it? Default: `FALSE`

Value

integer. Number of rows updated by executing `UPDATE` query. Or `character` SQL query string if `query_string=TRUE`.

gpkg_validate *Validate a GeoPackage*

Description

Checks for presence of required tables, valid values and other constraints.

Usage

```
gpkg_validate(x, diagnostics = FALSE)
```

Arguments

x	Path to GeoPackages
diagnostics	Return a list containing diagnostics (missing table names, invalid values, other errors)

Value

TRUE if valid. FALSE if one or more problems are found. For full diagnostics run with diagnostics = TRUE to return a list containing results for each input GeoPackage.

---

gpkg_write *Write data to a GeoPackage*

Description

Write data to a GeoPackage

Usage

```
gpkg_write(
  x,
  destfile,
  table_name = NULL,
  datatype = "FLT4S",
  append = FALSE,
  overwrite = FALSE,
  NoData = NULL,
  gdal_options = NULL,
  ...
)
```

Arguments

x	Vector of source file path(s), or a list containing one or more SpatRaster, SpatRasterCollection, or SpatVectorProxy objects.
destfile	Character. Path to output GeoPackage
table_name	Character. Default NULL name is derived from source file. Required if x is a *data.frame*.
datatype	Data type. Defaults to "FLT4S" for GeoTIFF files, "INT2U" otherwise. See documentation for terra::writeRaster().
append	Append to existing data source? Default: FALSE. Setting append=TRUE overrides overwrite=TRUE

overwrite	Overwrite existing data source? Default FALSE.
NoData	Value to use as GDAL NoData Value
gdal_options	Additional gdal_options, passed to terra::writeRaster()
...	Additional arguments are passed as GeoPackage "creation options." See Details.

Details

Additional, non-default GeoPackage creation options can be specified as arguments to this function. The full list of creation options can be viewed here or in the gpkg_creation_options dataset. The name of the argument is creation_option and the value is selected from one of the elements of values for that option.

Value

Logical. TRUE on successful write of at least one grid.

See Also

gpkg_creation_options

---

gpkg_write_attributes *Write or Remove Attribute Table in a GeoPackage*

Description

gpkg_write_attributes(): Specify a target geopackage and name for new table. For adding attributes, specify the new data as data.frame. The table name will be registered in the gpkg_contents table. Optionally include a custom description and/or use a template object to define the spatial extent associated with attribute data.

gpkg_remove_attributes(): Remove an attribute table and corresponding gpkg_contents record

Usage

```
gpkg_write_attributes(
  x,
  table,
  table_name,
  description = "",
  template = NULL,
  overwrite = FALSE,
  append = FALSE
)

gpkg_remove_attributes(x, table_name)
```

Arguments

x	A geopackage object
table	A data.frame
table_name	character. The name for table in x
description	Optional description. Default ""
template	A list (containing elements "ext" and "crs", or a terra object. These objects defining xmin/ymin/xmax/ymax and spatial reference system for the attribute table.
overwrite	Overwrite? Default FALSE
append	Append? Default FALSE

Value

logical. TRUE on successful table write or remove.

Index