

# Package ‘gpkg’

April 3, 2026

**Type** Package

**Title** Utilities for the Open Geospatial Consortium 'GeoPackage' Format

**Version** 0.0.14

**Maintainer** Andrew Brown <brown.andrew@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), gdalraster, tinytest, sf, dplyr, dbplyr, knitr, rmarkdown

**License** CC0

**Depends** R (>= 3.1.0)

**RoxygenNote** 7.3.3

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Andrew Brown [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-4565-533X>>)

**Repository** CRAN

**Date/Publication** 2026-04-03 05:10:10 UTC

## Contents

geopackage . . . . .	2
gpkg_2d_gridded_coverage_ancillary . . . . .	4

gpkg_2d_gridded_tile_ancillary . . . . .	4
gpkg_add_contents . . . . .	5
gpkg_add_metadata_extension . . . . .	6
gpkg_add_relatedtables_extension . . . . .	6
gpkg_connect . . . . .	7
gpkg_contents . . . . .	8
gpkg_create_2d_gridded_coverage_ancillary . . . . .	9
gpkg_create_dummy_features . . . . .	9
gpkg_create_empty_features . . . . .	10
gpkg_create_empty_grid . . . . .	11
gpkg_create_geometry_columns . . . . .	12
gpkg_create_spatial_view . . . . .	13
gpkg_creation_options . . . . .	14
gpkg_default_nodata . . . . .	15
gpkg_execute . . . . .	16
gpkg_extensions . . . . .	17
gpkg_list_contents . . . . .	17
gpkg_list_tables . . . . .	18
gpkg_query . . . . .	18
gpkg_read . . . . .	19
gpkg_source . . . . .	20
gpkg_spatial_ref_sys . . . . .	20
gpkg_sqlite_tables . . . . .	21
gpkg_tables . . . . .	22
gpkg_table_pragma . . . . .	22
gpkg_tile_set_data_null . . . . .	25
gpkg_update_table . . . . .	25
gpkg_validate . . . . .	26
gpkg_write . . . . .	27
gpkg_write_attributes . . . . .	28

<b>Index</b>	<b>30</b>
--------------	-----------

---

geopackage	geopackage <i>Constructors</i>
------------	--------------------------------

---

## Description

geopackage() (alias gpkg()) creates an S3 object of class geopackage.

## 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 = tmpdir(), ...)

## 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, ...)

gpkg(x, ...)

```

### 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 tmpdir.
...	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 tmpdir()

### Details

Several geopackage() methods are provided:

- `geopackage(x=<list>)`: creates a new GeoPackage object from a heterogeneous list of inputs
- `geopackage(x=<missing>)`: creates a new empty GeoPackage file in tmpdir
- `geopackage(x=<SQLiteConnection>)`: creates a GeoPackage object from an existing *SQLite* connection
- `geopackage(x=<character>)`: creates a GeoPackage object from a path to an existing GeoPackage file

### 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\_2d\_gridded\_tile\_ancillary

*Get gpkg\_2d\_gridded\_tile\_ancillary Table*

---

**Description**

Get gpkg\_2d\_gridded\_tile\_ancillary Table

**Usage**

gpkg\_2d\_gridded\_tile\_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,
  data_type = NULL,
  description = "",
  srs_id = NULL,
  ext = NULL,
  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	<i>A geopackage</i>
table_name	Name of table to add or remove record for in <i>gpkg_contents</i>
data_type	<i>character</i> . One of: 2d-gridded-coverage, "features", "attributes". Default NULL will attempt to auto-detect table type based on gpkg_table_pragma() information; falls back to "attributes" if raster or vector data are not detected.
description	Default: ""
srs_id	<i>integer</i> . Spatial Reference System ID. Must be defined in gpkg_spatial_ref_sys table.
ext	<i>numeric</i> . A numeric vector of length four specifying the bounding box extent.
template	Deprecated. A list containing elements "srsid" and "ext".
query_string	<i>logical</i> . 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**

∅ (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**

∅ (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)

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

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

gpkg_connection(x, disconnect = FALSE)

## Default S3 method:
gpkg_connection(x, disconnect = FALSE)
```

**Arguments**

<code>x</code>	A <i>geopackage</i> or <i>SQLiteConnection</i> object
<code>disconnect</code>	Set attribute 'disconnect' on <i>SQLiteConnection</i> object to auto-disconnect? Default: FALSE

**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\_2d\_gridded\_coverage\_ancillary

*Create 2D Gridded Coverage and Tile Ancillary Tables*


---

### Description

Create 2D Gridded Coverage and Tile Ancillary Tables

### Usage

```
gpkg_create_2d_gridded_coverage_ancillary(x)
```

```
gpkg_create_2d_gridded_tile_ancillary(x)
```

### Arguments

x                    A *geopackage* object

### Value

gpkg\_create\_2d\_gridded\_coverage\_ancillary(): *integer*. Result of running sequential gpkg\_execute() statements.

gpkg\_create\_2d\_gridded\_tile\_ancillary(): *integer*. Result of running sequential gpkg\_execute() statements.

### References

<http://www.opengis.net/doc/IS/geopackage-gr/1.0>

---

gpkg\_create\_dummy\_features

*Create a Dummy Feature Dataset in a GeoPackage*


---

### Description

This function has been deprecated. Please use gpkg\_create\_empty\_features().

### 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**

Create a minimal (empty) feature table and gpkg\_geometry\_columns table entry.

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\\_create\\_empty\\_features\(\)](#) [gpkg\\_vect\(\)](#) [gpkg\\_ogr\\_query\(\)](#)

---

gpkg\_create\_empty\_features

*Create an empty feature table*

---

**Description**

Create an empty feature table and associated entries for gpkg\_spatial\_ref\_sys, and gpkg\_geometry\_columns.

**Usage**

```
gpkg_create_empty_features(
  x,
  table_name,
  column_name = "geom",
  geometry_type_name = "GEOMETRY",
  srs_id = 4326,
  z = 0L,
  m = 0L,
  contents = TRUE,
  description = "",
  ext = c(-180, -90, 180, 90)
)
```

**Arguments**

x	A geopackage Object
table_name	<i>character</i> . New table name.
column_name	<i>character</i> . Geometry column name. Default "geom"
geometry_type_name	<i>character</i> . Geometry type name. Default: "GEOMETRY"

srs_id	<i>integer</i> . Spatial Reference System ID. Must be defined in gpkg_spatial_ref_sys table.
z	<i>integer</i> . Default: 0
m	<i>integer</i> . Default: 0
contents	<i>logical</i> . If TRUE (default) add the new table to gpkg_contents table.
description	<i>character</i> . Description for gpkg_contents table. Default: ""
ext	<i>numeric</i> . A numeric vector of length four specifying the bounding box extent.

**Value**

*integer* result of gpkg\_execute(). Returns 1 if a new geometry record is appended to gpkg\_geometry\_columns table.

**See Also**

[gpkg\\_create\\_empty\\_grid\(\)](#)

---

gpkg\_create\_empty\_grid

*Create an empty grid table*

---

**Description**

Create an empty grid table and associated entries for gpkg\_spatial\_ref\_sys, gpkg\_2d\_gridded\_coverage\_ancillary, and gpkg\_2d\_gridded\_tile\_ancillary.

**Usage**

```
gpkg_create_empty_grid(
  x,
  tile_matrix_set_name,
  datatype = "integer",
  scale = 1,
  offset = 0,
  precision = 1,
  data_null = NULL,
  grid_cell_encoding = "grid-value-is-center",
  uom = NULL,
  field_name = "Height",
  quantity_definition = "Height",
  srs_id = 4326,
  contents = TRUE,
  description = "",
  ext = c(-180, -90, 180, 90)
)
```

**Arguments**

x	A <i>geopackage</i> object
tile_matrix_set_name	<i>character</i> . New table name.
datatype	<i>character</i> . Either "integer" (default) or "float".
scale	<i>numeric</i> . Default: 1.0
offset	<i>numeric</i> . Default: 0.0
precision	<i>numeric</i> . Default: 1.0
data_null	<i>numeric</i> . Default: NULL
grid_cell_encoding	<i>character</i> Default: "grid-value-is-center"
uom	<i>character</i> . Unit of measure. Default: NULL
field_name	<i>character</i> . Default: "Height".
quantity_definition	<i>character</i> . Default: "Height"
srs_id	<i>integer</i> . Spatial Reference System ID. Must be defined in <code>gpkg_spatial_ref_sys</code> table. Default: 4326
contents	<i>logical</i> . Include entry in <code>gpkg_contents</code> ? Default: TRUE
description	<i>character</i> . Description for <code>gpkg_contents</code> . Default: ""
ext	<i>numeric</i> . Length 4. Extent (c(xmin, ymin, xmax, ymax)) for <code>gpkg_contents</code> . Default: c(-180, -90, 180, 90)

**Value**

*integer*

---

gpkg\_create\_geometry\_columns

*GeoPackage Geometry Columns*

---

**Description**

Create `gpkg_geometry_columns` table to account for geometry columns within the database with `gpkg_create_geometry_columns()`. Register new geometry columns with `gpkg_add_geometry_columns()`.

**Usage**

```
gpkg_create_geometry_columns(x)
```

```
gpkg_add_geometry_columns(
  x,
  table_name,
  column_name,
```

```

    geometry_type_name = "GEOMETRY",
    srs_id,
    z,
    m
)

```

### Arguments

x	A <i>geopackage</i> object, or path to GeoPackage file.
table_name	<i>character</i> . New table name.
column_name	<i>character</i> . Geometry column name. Default "geom"
geometry_type_name	<i>character</i> . Geometry type name. Default: "GEOMETRY"
srs_id	<i>integer</i> . Spatial Reference System ID. Must be defined in <code>gpkg_spatial_ref_sys</code> table.
z	<i>integer</i> . Default: 0
m	<i>integer</i> . Default: 0

### Value

*integer*. 1 if table created or row inserted successfully, 0 otherwise.

---

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	<i>character</i> . Name of view.
viewquery	<i>character</i> . Query for view contents.
geom_column	<i>character</i> . Column name of view geometry. Default: "geom"
geometry_type_name	<i>character</i> . View geometry type. Default: "GEOMETRY"
spatialite_computed	<i>logical</i> . Register definition of geom_column as the result of a Spatialite spatial function via "gdal_spatialite_computed_geom_column" extension. Default: FALSE
data_type	<i>character</i> . View data type. Default "features"
srs_id	<i>integer</i> . Spatial Reference System ID. Default: 4326 (WGS84)
z	<i>integer</i> . Default: 0
m	<i>integer</i> . 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\_default\_nodata    *Get Default NoData Value for GeoPackage Raster Datatypes*

---

## Description

Returns a default NoData value for a given raster datatype, following GDAL and terra conventions, and avoiding NaN values which produce a warning when passed as `data_null` to GDAL. NOTE: at this time (GDAL 3.12) only Byte (INT1U), Int16 (INT2S), UInt16 (INT2U) or Float32 (INT4S) datatypes are supported in the **GPKG raster driver**.

## Usage

```
gpkg_default_nodata(datatype)
```

## Arguments

`datatype`            character. A raster datatype code in terra format. Supported types are:

- Signed integers: "INT1S", "INT2S", "INT4S", "INT8S"
- Unsigned integers: "INT1U", "INT2U", "INT4U", "INT8U"
- Floats: "FLT4S" (single precision), "FLT8S" (double precision)

## Details

While NaN is technically a valid value for grids stored in a geopackage, it is generally preferred to use a numeric value, as NaN is not supported by SQLite 3 and is stored in the metadata column as SQL NULL. These defaults are used by `gpkg_write()` when `auto_nodata = TRUE`.

For signed integer types, the minimum representable value (e.g., `INT32_MIN`) is returned, as it is unlikely to occur in valid data. For unsigned integer types, the maximum representable value is returned (e.g., `UINT16_MAX`), with the exception of INT1U (byte) which returns 255 (following terra convention). For float types, the negative extreme value is returned, as GeoPackage metadata cannot represent NaN.

Datatype	Default NoData
INT1S	-128
INT2S	-32768
INT4S	-2147483648
INT8S	-9223372036854775808
INT1U	255
INT2U	65534
INT4U	4294967295
INT8U	18446744073709549568
FLT4S	-3.40282346638528860e+38
FLT8S	-1.79769313486231571e+308

Note: The INT8U default (18446744073709549568) is `UINT64_MAX - 1101` to account for floating-point precision loss when converting to double, following terra's convention. R does not have a native unsigned 64-bit integer type.

**Value**

numeric. The default NoData value for the given datatype, or NA\_real\_ if no default is available or the datatype is not recognized.

**See Also**

[gpkg\\_write\(\)](#) for using these defaults when writing rasters to GeoPackage

**Examples**

```
gpkg_default_nodata("FLT4S")
gpkg_default_nodata("INT2S")
gpkg_default_nodata("INT2U")
```

---

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 <i>geopackage</i> object
statement	An SQLite statement
...	Additional arguments to <code>RSQLite::dbExecute()</code>
silent	Used to suppress error messages, passed to <code>try()</code> . Default: FALSE.

**Value**

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

---

gpkg_extensions	<i>Get Geopackage Extensions</i>
-----------------	----------------------------------

---

**Description**

Get Geopackage Extensions

**Usage**

```
gpkg_extensions(x)
```

**Arguments**

x                    A geopackage

**Value**

a data.frame

---

gpkg_list_contents	<i>List Tables Registered in a GeoPackage</i>	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 'SQLite' 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/en/stable/drivers/vector/gpkg.html#sql-functions>>

**Value**

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

---

gpkg_read	<i>Read data from a GeoPackage</i>
-----------	------------------------------------

---

**Description**

This function creates a *geopackage* object with references to all tables from the GeoPackage source specified in `x`. For a simple list of tables see `gpkg_tables()`.

**Usage**

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

**Arguments**

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

**Value**

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

**See Also**

[gpkg\\_tables\(\)](#)

---

gpkg_source	<i>Get Source File of a geopackage object</i>
-------------	---

---

**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_spatial_ref_sys	<i>GeoPackage Spatial Reference System</i>
----------------------	--

---

**Description**

GeoPackage Spatial Reference System

**Usage**

```
gpkg_spatial_ref_sys(x)

gpkg_list_srs(x, column_name = "srs_id")

gpkg_create_spatial_ref_sys(x, default = TRUE, query_string = FALSE)

gpkg_add_spatial_ref_sys(
  x,
  srs_name = "",
  srs_id = NULL,
  organization = "",
  organization_coordsys_id = 0L,
  definition = "",
  description = "",
  query_string = FALSE)
```

```
)
gpkg_delete_spatial_ref_sys(x, srs_id = NULL)
```

### Arguments

x	A geopackage object
column_name	Default: "srs_id"
default	<i>logical</i> or <i>character</i> . If TRUE, or one or more of "cartesian", "geographic", or "crs84", then these default Spatial Reference Systems are added.
query_string	<i>logical</i> . Return SQL queries without executing? Default: FALSE
srs_name	<i>character</i> . Spatial Reference System Name, for example "WGS 84 geodetic"
srs_id	<i>integer</i> . Spatial Reference System ID, for example 4326L
organization	<i>character</i> . Organization, for example "EPSG"
organization_coordsys_id	<i>integer</i> . Organization Coordinate System ID, for example 4326L
definition	<i>character</i> . WKT2019 Coordinate Reference System description string
description	<i>character</i> . Description

### Value

gpkg\_spatial\_ref\_sys(): *data.frame*

gpkg\_list\_srs(): *vector* of values from specified column\_name

gpkg\_create\_spatial\_ref\_sys(): *integer*. Result of running sequential gpkg\_execute() statements. This method is run for the side-effect of creating the table if needed, and adding any "default" records.

gpkg\_add\_spatial\_ref\_sys(): *integer* result of executing SQL statement

gpkg\_delete\_spatial\_ref\_sys(): *integer* result of executing SQL statement

---

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	<i>Get Tables from a geopackage object</i>
-------------	--

---

**Description**

Get Tables from a *geopackage* object

**Usage**

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

## Default S3 method:
gpkg_tables(x, collect = TRUE, pragma = TRUE)
```

**Arguments**

x	A <i>geopackage</i> 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	<i>Lazy Access to Tables by Name</i>
-------------------	--------------------------------------

---

**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 *tbl\_SQLiteConnection* object referencing that table

gpkg\_collect(): Alias for gpkg\_table(..., collect=TRUE)

gpkg\_tbl(): Alias for gpkg\_table(..., collect=FALSE)(default) that *always* returns a *tbl\_SQLiteConnection* object.

gpkg\_rast(): Get a *SpatRaster* object corresponding to the specified *table\_name*

gpkg\_vect(): Get a *SpatVector* object corresponding to the specified *table\_name*

gpkg\_sf(): Get a *sf-tibble* object corresponding to the specified *table\_name*

**Usage**

```

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

## Default S3 method:
gpkg_table_pragma(x, table_name = NULL, ...)

gpkg_table(
  x,
  table_name,
  collect = FALSE,
  column_names = "*",
  query_string = FALSE,
  ...
)

## Default S3 method:
gpkg_table(
  x,
  table_name,
  collect = FALSE,
  column_names = "*",
  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, ...)

gpkg_sf(x, table_name, ...)

```

**Arguments**

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

`column_names` *character*. Used only when `collect=TRUE`. A *character* vector of column names to select from `table_name`.

`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)

`gpkg_sf()`: An *sf-tibble* object of class "sf", "tbl\_df". If the table contains no geometry column the result is a "tbl\_df".

### Examples

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

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

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

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

g <- geopackage(tf, connect = TRUE)

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

gpkg_contents(g)

# 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 <i>geopackage</i> object, path to a GeoPackage or an <i>SQLiteConnection</i>
name	character. Tile matrix set name(s) ( <i>tile_matrix_set_name</i> )
value	numeric. Value to use as "NoData" ( <i>data_null</i> 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 <i>geopackage</i> object, path to a <i>GeoPackage</i> or an <i>SQLiteConnection</i> .
table_name	<i>character</i> . Table name.
updatecol	<i>character</i> . Column to update.
updatevalue	<i>character, numeric, etc.</i> ; A scalar value to set.
wherecol	<i>character</i> . Column used to constrain update.
wherevector	<i>character, numeric, etc.</i> ; Vector of values where update should be made.
query_string	<i>logical</i> . Return <i>SQLite</i> 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	<i>Validate a GeoPackage</i>
---------------	------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

x	A <i>geopackage</i> object, or <i>character</i> path to <i>GeoPackage</i>
diagnostics	Return a list containing individual diagnostic test results (see <i>Details</i> )

**Details**

Several tests are run on the input *GeoPackage*, including:

- `required_tables`: *logical*. TRUE if `gpkg_contents` and `gpkg_spatial_ref_sys` tables exist
- `has_contents`: *logical*. TRUE if the number of rows in `gpkg_contents` table is greater than 0 and all tables listed in `gpkg_contents` are in the database
- `has_spatial_tables`: *logical*. TRUE if the number of tables in `gpkg_contents` with `data_type` "features" or "2d-gridded-coverage" is greater than 0

**Value**

*logical*. 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 test run on the input *GeoPackage*.

---

gpkg\_write                      *Write data to a GeoPackage*

---

### Description

Write data to a GeoPackage

### Usage

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

### Arguments

x	Vector of source file path(s), or a list containing one or more SpatRaster, SpatRasterCollection, SpatVectorProxy, or data.frame objects.
y	<i>character, geopackage, or DBIConnection.</i>
table_name	<i>character.</i> Default NULL name is derived from source file. Required if x is a <i>data.frame</i> .
datatype	<i>character.</i> Data type. Defaults to "FLT4S" for GeoTIFF files, "INT2U" otherwise. See documentation for terra::writeRaster().
append	<i>logical.</i> Append to existing data source? Default: FALSE. When TRUE, new data are added to the existing table(s) if they exist. For raster data, this adds a new subdataset (layer) to the GeoPackage (via APPEND_SUBDATASET=YES creation option). For attribute tables, this appends rows to the existing table. For vector data, this maps to the insert argument of terra::writeVector(). Setting append=TRUE overrides overwrite=TRUE.
overwrite	<i>logical.</i> Overwrite existing data source? Default FALSE. When TRUE, existing table(s) with the same name are dropped and recreated. Note that this only affects the specific tables being written; it does not delete other existing tables in the GeoPackage file. To completely overwrite an entire GeoPackage file, delete it first using unlink().
NoData	<i>numeric.</i> Value to use as GDAL NoData Value
gdal_options	<i>character.</i> Additional gdal_options, passed to terra::writeRaster()

auto_nodata	logical. If TRUE (default), automatically select a datatype-appropriate default NoData value when NoData = NULL, determined by <code>gpkg_default_nodata()</code> . Set to FALSE to use terra's default behavior (NaN for floats), which may trigger GDAL warnings. Ignored if NoData is explicitly specified.
destfile	<i>character</i> . Path to output GeoPackage
...	Additional arguments are passed as GeoPackage creation options. See Details.

### Details

`gpkg_write()` can write multiple layers of different types (raster, vector, and attributes) in a single call when `x` is a list. If calling `gpkg_write()` multiple times to build a GeoPackage, use `append=TRUE` to add additional layers. To replace a specific table while preserving others, use `overwrite=TRUE`. Note that `overwrite=TRUE` only drops the specified `table_name` and does not affect other tables in the file.

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.

If `x` contains source file paths, any comma-separated value (CSV) files are treated as attribute data—even if they contain a geometry column. GeoPackage source file paths are always treated as vector data sources, and only one layer will be read from the source and written to the target. If you need to read raster data from a GeoPackage first create a `SpatRaster` from the layer of interest (see `gpkg_rast()`) before passing to `gpkg_write()`. If you need to read multiple layers from any multi-layer source read them individually into suitable objects. For a source GeoPackage containing multiple layers you can use `gpkg_read()` (returns a *geopackage* object) or `gpkg_tables()` (returns a *list* object).

### Value

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

### See Also

[gpkg\\_creation\\_options](#) [gpkg\\_default\\_nodata\(\)](#)

---

`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 existing table? Default FALSE
append	Append rows to existing table? Default FALSE. Setting append=TRUE overrides overwrite=TRUE.

**Value**

logical. TRUE on successful table write or remove.

# Index

- \* **datasets**
  - gpkg\_creation\_options, 14
  - gpkg\_sqlite\_tables, 21
- \* **io**
  - gpkg\_read, 19
  - gpkg\_write, 27
- geopackage, 2
- gpkg (geopackage), 2
- gpkg\_2d\_gridded\_coverage\_ancillary, 4
- gpkg\_2d\_gridded\_tile\_ancillary, 4
- gpkg\_add\_contents, 5
- gpkg\_add\_geometry\_columns
  - (gpkg\_create\_geometry\_columns), 12
- gpkg\_add\_metadata\_extension, 6
- gpkg\_add\_relatedtables\_extension, 6
- gpkg\_add\_spatial\_ref\_sys
  - (gpkg\_spatial\_ref\_sys), 20
- gpkg\_collect (gpkg\_table\_pragma), 22
- gpkg\_connect, 7
- gpkg\_connection (gpkg\_connect), 7
- gpkg\_contents, 8
- gpkg\_contents(), 17
- gpkg\_create\_2d\_gridded\_coverage\_ancillary, 9
- gpkg\_create\_2d\_gridded\_tile\_ancillary
  - (gpkg\_create\_2d\_gridded\_coverage\_ancillary), 9
- gpkg\_create\_contents
  - (gpkg\_add\_contents), 5
- gpkg\_create\_dummy\_features, 9
- gpkg\_create\_empty\_features, 10
- gpkg\_create\_empty\_features(), 10
- gpkg\_create\_empty\_grid, 11
- gpkg\_create\_empty\_grid(), 11
- gpkg\_create\_geometry\_columns, 12
- gpkg\_create\_spatial\_ref\_sys
  - (gpkg\_spatial\_ref\_sys), 20
- gpkg\_create\_spatial\_view, 13
- gpkg\_creation\_options, 14, 28
- gpkg\_default\_nodata, 15
- gpkg\_default\_nodata(), 28
- gpkg\_delete\_contents
  - (gpkg\_add\_contents), 5
- gpkg\_delete\_spatial\_ref\_sys
  - (gpkg\_spatial\_ref\_sys), 20
- gpkg\_disconnect (gpkg\_connect), 7
- gpkg\_execute, 16
- gpkg\_extensions, 17
- gpkg\_is\_connected (gpkg\_connect), 7
- gpkg\_list\_contents, 17
- gpkg\_list\_srs (gpkg\_spatial\_ref\_sys), 20
- gpkg\_list\_tables, 18
- gpkg\_list\_tables(), 17
- gpkg\_ogr\_contents (gpkg\_contents), 8
- gpkg\_ogr\_query (gpkg\_query), 18
- gpkg\_ogr\_query(), 10
- gpkg\_query, 18
- gpkg\_rast (gpkg\_table\_pragma), 22
- gpkg\_read, 19
- gpkg\_remove\_attributes
  - (gpkg\_write\_attributes), 28
- gpkg\_sf (gpkg\_table\_pragma), 22
- gpkg\_source, 20
- gpkg\_spatial\_ref\_sys, 20
- gpkg\_sqlite\_tables, 21
- gpkg\_table (gpkg\_table\_pragma), 22
- gpkg\_table\_pragma, 22
- gpkg\_tables, 22
- gpkg\_tables(), 19
- gpkg\_tbl (gpkg\_table\_pragma), 22
- gpkg\_tile\_set\_data\_null, 25
- gpkg\_update\_contents
  - (gpkg\_add\_contents), 5
- gpkg\_update\_table, 25
- gpkg\_validate, 26
- gpkg\_vect (gpkg\_table\_pragma), 22
- gpkg\_vect(), 10

`gpkg_write`, [27](#)

`gpkg_write()`, [15](#), [16](#)

`gpkg_write_attributes`, [28](#)