Package: hfsubsetR (via r-universe)

December 5, 2024

Type Package

Title Hydrofabric Subsetter

Description Subset Hydrofabric Data in R.

Version 0.3.2

Maintainer Mike Johnson <mike.johnson@noaa.gov>

BugReports https://github.com/lynker-spatial/hfsubsetR

URL https://github.com/lynker-spatial/hfsubsetR

Depends R (>= 4.2)

Imports arrow, DBI, dplyr, dbplyr, glue, httr, jsonlite, nhdplusTools, sf, methods

Suggests testthat

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Config/testthat/edition 3

Config/testthat/parallel true

Collate 'OGRSQLConnection.R' 'OGRSQLDriver.R' 'OGRSQLResult.R' 'find_origin.R' 'get_subset.R' 'hfsubsetR-package.R' 'query.R' 'query_source.R' 'query_source_arrow.R' 'query_source_sf.R' 'query_subset.R' 'sf_arrow.R' 'sf_ogr.R' 'utils.R' 'zzz.R'

Config/pak/sysreqs cmake libgdal-dev gdal-bin libgeos-dev libicu-dev libpng-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev

Repository https://owp-spatial.r-universe.dev

RemoteUrl https://github.com/lynker-spatial/hfsubsetR

RemoteRef HEAD

RemoteSha 4146a8cde8725f2d79974a7d3ea28ce1164fd6dc

Contents

`as_ogr` *Delayed read for vector resources*

Description

A lazy data frame for GDAL vector data sources. as_ogr is DBI compatible and designed to work with dplyr.

Usage

```
as_ogr(x, layer, ..., query = NA, ignore_lyrs = "gpkg_|rtree_|sqlite_")

## S3 method for class 'character'
as_ogr(x, layer, ..., query = NA, ignore_lyrs = "gpkg_|rtree_|sqlite_")

## S3 method for class 'OGRSQLConnection'
as_ogr(x, layer, ..., query = NA, ignore_lyrs = "gpkg_|rtree_|sqlite_")
```

Arguments

`x`	the data source (file path, url, or database connection)
`layer`	layer name (varies by driver, may be a file name without extension); in case `layer` is missing, `st_read` will read the first layer of `dsn`, give a warning and (unless `quiet = TRUE`) print a message when there are multiple layers, or give an error if there are no layers in `dsn`. If `dsn` is a database connection, then `layer` can be a table name or a database identifier (see `Id`). It is also possible to omit `layer` and rather use the `query` argument.
`...`	parameter(s) passed on to st_as_sf
`query`	SQL query to pass in directly
`ignore_lyrs`	pattern for layers to be ignored description

Details

The output of ‘as_ogr()‘ is a ’tbl_OGRSQLConnection‘ that extends ‘tbl_dbi‘ and may be used with functions and workflows in the normal DBI way, see [OGRSQL()] for the as_ogr DBI support.

To obtain an in memory data frame use an explict ‘collect()‘ or ‘st_as_sf()‘. A call to ‘collect()‘ is triggered by ‘st_as_sf()‘ and will add the sf class to the output.

Value

a ’tbl_OGRSQLConnection’

`dbConnect,OGRSQLDriver-method`
dbConnect

Description

dbConnect for sources that can be read by package sf

Usage

```
## S4 method for signature 'OGRSQLDriver'
dbConnect(drv, DSN = "", readonly = TRUE, ...)
```

Arguments

`drv`	OGRSQLDriver created by `OGRSQL()`
`DSN`	data source name
`readonly`	open in readonly mode (‘TRUE‘ is the only option)
`...`	ignored

Details

The ’OGRSQL’ available is documented with GDAL: https://gdal.org/user/ogr_sql_dialect.html

find_origin *Find an origin from indexed IDs*

Description

Find an origin from indexed IDs

Usage

```
find_origin(
  network,
  id,
  type = c("id", "comid", "hl_uri", "poi_id", "nldi_feature", "xy")
)
```

Arguments

network	A ‘dplyr‘-compatible object.
id	A queryable identifier of type ‘type‘.
type	An index type describing ‘id‘.

Value

A network origin. If a single origin is not found, then an exception is raised.

get_hydrofabric *Download a Hydrofabric Geopackage*

Description

Downloads a hydrofabric Geopackage from a specified URL and saves it to a local file.

Usage

```
get_hydrofabric(
  url = "https://lynker-spatial.s3-us-west-2.amazonaws.com/hydrofabric",
  version = "2.2",
  domain = "conus",
  type = "nextgen",
  outfile = NULL,
  overwrite = FALSE
)
```

Arguments

`url`	A character string specifying the base URL of the hydrofabric repository. Defaults to ‘’https://lynker-spatial.s3-us-west-2.amazonaws.com/hydrofabric’‘.
`version`	A character string indicating the version of the hydrofabric to download. Defaults to ‘’2.2’‘.
`domain`	A character string specifying the geographic domain of the hydrofabric. Defaults to ‘’conus’‘.
`type`	A character string indicating the type of hydrofabric. Defaults to ‘’nextgen’‘.
`outfile`	A character string specifying the path to save the downloaded file. If ‘NULL‘, the file will not be saved. Defaults to ‘NULL‘.
`overwrite`	A logical value indicating whether to overwrite an existing file. Defaults to ‘FALSE‘.

Value

The function returns the path to the downloaded file (‘outfile‘).

Examples

```
## Not run:
# Download the default hydrofabric file
get_hydrofabric(outfile = "conus_nextgen.gpkg")

# Specify a different domain and version
get_hydrofabric(
  version = "3.0",
  domain = "hawaii",
  outfile = "hawaii_nextgen.gpkg",
  overwrite = TRUE
)

## End(Not run)
```

`get_subset` *Build a hydrofabric subset*

Description

Build a hydrofabric subset

Usage

```
get_subset(
  id = NULL,
  comid = NULL,
  hl_uri = NULL,
  poi_id = NULL,
  nldi_feature = NULL,
  xy = NULL,
  lyrs = c("divides", "flowpaths", "network", "nexus"),
  gpkg = NULL,
  source = "s3://lynker-spatial/hydrofabric",
  hf_version = "2.2",
  type = "nextgen",
  domain = "conus",
  outfile = NULL,
  overwrite = FALSE
)
```

Arguments

`id`	hydrofabric id. datatype: string / vector of strings e.g., 'wb-10026' or c('wb-10026', 'wb-10355')
`comid`	NHDPlusV2 COMID. datatype: int / vector of int e.g., 61297116 or c(61297116 , 6129261)
`hl_uri`	hydrolocation URI. datatype: string / vector of string / a url e.g., HUC12-010100100101 or c(HUC12-010100100101 , HUC12-010100110104)
`poi_id`	POI identifier. datatype: int / vector of int e.g., 266387 or c(266387, 266745)
`nldi_feature`	list with names 'featureSource' and 'featureID' where 'featureSource' is derived from the "source" column of the response of dataRetrieval::get_nldi_sources() and the 'featureID' is a known identifier from the specified 'featureSource'. datatype: a url e.g., 'https://labs.waterdata.usgs.gov/api/nldi/linked-data/census2020-nhdpv2'
`xy`	Location given as vector of XY in EPSG:4326 (longitude, latitude, crs)
`lyrs`	layers to extract
`gpkg`	a local gpkg file
`source`	hydrofabric source (local root directory or s3 link)
`hf_version`	hydrofabric version
`type`	hydrofabric type
`domain`	hydrofabric domain
`outfile`	If gpkg file path is provided, data will be written to a file.
`overwrite`	overwrite existing outfile file path. Default is FALSE

`get_vpu_fabric` *Get VPU Fabric*

Description

Retrieve and Process Vector Processing Unit (VPU) Hydrofabric Layers

This function retrieves and optionally filters spatial data layers from a GeoPackage (GPKG) based on a specified Vector Processing Unit ID (VPU ID). The function can either return the filtered layers as a list or write them to an output file.

Usage

```
get_vpu_fabric(gpkg, vpuid = NULL, outfile = NULL)
```

Arguments

`gpkg`	A string specifying the path to the GeoPackage file.
`vpuid`	A vector of VPU IDs to filter the layers. If ‘NULL‘, no filtering is applied. Default is ‘NULL‘.
`outfile`	A string specifying the path to write the filtered layers to a new GeoPackage. If ‘NULL‘, the layers are returned as a list. Default is ‘NULL‘.

Details

The function reads all layers from the provided GeoPackage, excluding the "error" layer. For each layer, the data is optionally filtered by the provided ‘vpuid‘ and then processed into ‘sf‘ objects. If an output file path is provided, the filtered layers are written to a new GeoPackage. Otherwise, the layers are stored in a list and returned.

Value

If ‘outfile‘ is ‘NULL‘, returns a list where each element is a filtered spatial layer (‘sf‘ object). If ‘outfile‘ is provided, returns the path to the output GeoPackage.

Examples

```
## Not run:
# Example 1: Retrieve filtered layers as a list
fabric <- get_vpu_fabric("path/to/geopackage.gpkg", vpuid = c("01", "02"))

# Example 2: Write filtered layers to a new GeoPackage
get_vpu_fabric("path/to/geopackage.gpkg", vpuid = c("01", "02"), outfile = "output.gpkg")

## End(Not run)
```

OGRSQL — *OGRSQL OGRSQL driver, use to [dbConnect()] to a data source readable by sf*

Description

OGRSQL OGRSQL driver, use to [dbConnect()] to a data source readable by sf

Usage

```
OGRSQL()
```

query — *Initialize a new hfsubset query*

Description

Initialize a new hfsubset query

Usage

```
query()
```

Value

A ‘hfsubset_query‘ object

query_set_id — *Set the identifier of a query*

Description

Set the identifier of a query

Usage

```
query_set_id(
  query,
  identifier,
  type = c("id", "comid", "hl_uri", "poi_id", "nldi_feature", "xy")
)
```

Arguments

`query`	A ‘hfsubset_query‘ object
`identifier`	Identifier value
`type`	Identifier type

Value

‘query‘ with the identifier included

`query_set_layers` *Set the layers of a query*

Description

Set the layers of a query

Usage

```
query_set_layers(query, layers)
```

Arguments

`query`	A ‘hfsubset_query‘ object
`layers`	A ‘character‘ vector of layer names

Value

‘query‘ with the layers included

`query_set_sink` *Set the sink of a query*

Description

Set the sink of a query

Usage

```
query_set_sink(query, sink, overwrite = FALSE)
```

Arguments

`query`	A ‘hfsubset_query‘ object
`sink`	A character path to sink
`overwrite`	If TRUE, then if the sink exists, it should be overwritten

Value

‘query‘ with the sink included

`query_set_source` *Set the source of a query*

Description

Set the source of a query

Usage

```
query_set_source(query, src)
```

Arguments

`query`	A ‘hfsubset_query‘ object
`src`	A ‘hfsubset_query_source‘ object

Value

‘query‘ with the source included

See Also

query_source_arrow query_source_sf

`query_source_arrow` *Create a new ‘arrow‘ query source.*

Description

Create a new ‘arrow‘ query source.

Usage

```
query_source_arrow(srcname, ...)
```

Arguments

`srcname`	URI to ‘arrow‘-compatible dataset
`...`	Unused

Value

An ‘hfsubset_query_source_arrow‘ object

`query_source_sf` *Create a new ‘sf‘ query source.*

Description

Create a new ‘sf‘ query source.

Usage

```
query_source_sf(srcname, ...)
```

Arguments

`srcname`	Path or VSI URI to source
`...`	Unused

Value

An ‘hfsubset_query_source_sf‘ object

`query_subset` *Execute a query subset*

Description

Execute a query subset

Usage

```
query_subset(query)
```

Arguments

`query`	A ‘hfsubset_query‘ object

Value

A list of hydrofabric layers, or the path to the sink of the query `read_sf_dataset` *Read Parquet Dataset*

Description

Read an Arrow multi-file dataset and create `sf` object

Usage

```
read_sf_dataset(dataset, find_geom = FALSE)
```

Arguments

`dataset`	a `Dataset` object created by `arrow::open_dataset` or an `arrow_dplyr_query`
`find_geom`	logical. Only needed when returning a subset of columns. Should all available geometry columns be selected and added to to the dataset query without being named? Default is `FALSE` to require geometry column(s) to be selected specifically.

Details

This function is primarily for use after opening a dataset with `arrow::open_dataset`. Users can then query the `arrow` `Dataset` using `dplyr` methods such as `filter` or `select`. Passing the resulting query to this function will parse the datasets and create an `sf` object. The function expects consistent geographic metadata to be stored with the dataset in order to create `sf` objects. Adopted from wcjochem/sfarrow

Value

object of class `sf`

See Also

`open_dataset`, `st_read`, `st_read_parquet`

`st_as_sf` *Force collection of a OGR query Convert as_ogr to a data frame or sf object*

Description

Force collection of a OGR query Convert as_ogr to a data frame or sf object

Usage

```
## S3 method for class 'tbl_OGRSQLConnection'
st_as_sf(x, ...)
```

Arguments

x	output of [as_ogr()]
...	passed to [collect()]

Value

a data frame from ‘collect()‘, sf data frame from ‘st_as_sf()‘ (only if it contains an ‘sfc‘ geometry column)

`st_read_parquet` *Read a Parquet file to* `sf` *object*

Description

Read a Parquet file. Uses standard metadata information to identify geometry columns and coordinate reference system information.

Usage

```
st_read_parquet(dsn, col_select = NULL, props = NULL, ...)
```

Arguments

dsn	character file path to a data source
col_select	A character vector of column names to keep. Default is `NULL` which returns all columns
props	Now deprecated in `read_parquet`.
...	additional parameters to pass to `ParquetFileReader`

Details

Reference for the metadata used: `https://github.com/geopandas/geo-arrow-spec`. These are standard with the Python `GeoPandas` library. Adopted from wcjochem/sfarrow

Value

object of class `sf`

See Also

`read_parquet`, `st_read`

st_write_parquet — *Write sf object to Parquet file*

Description

Convert a simple features spatial object from `sf` to a Parquet file using `write_parquet`. Geometry columns (type `sfc`) are converted to well-known binary (WKB) format.

Usage

```
st_write_parquet(
  obj,
  dsn,
  hf_version = "2.2",
  license = "ODbL",
  source = "lynker-spatial",
  ...
)
```

Arguments

`obj`	object of class `sf`
`dsn`	data source name. A path and file name with .parquet extension
`hf_version`	dataset version
`license`	dataset license
`source`	dataset source
`...`	additional options to pass to `write_parquet`

Details

Adopted from wcjochem/sfarrow

Value

`obj` invisibly

See Also

`write_parquet`

`write_sf_dataset` *Write Parquet Dataset*

Description

Write `sf` object to an Arrow multi-file dataset

Usage

```
write_sf_dataset(
  obj,
  path,
  format = "parquet",
  partitioning = dplyr::group_vars(obj),
  hf_version = "2.2",
  license = "ODbL",
  source = "lynker-spatial",
  ...
)
```

Arguments

`obj`	object of class `sf`
`path`	string path referencing a directory for the output
`format`	output file format ("parquet" or "feather")
`partitioning`	character vector of columns in `obj` for grouping or the `dplyr::group_vars`
`hf_version`	dataset version
`license`	dataset license
`source`	dataset source
`...`	additional arguments and options passed to `arrow::write_dataset`

Details

Translate an `sf` spatial object to `data.frame` with WKB geometry columns and then write to an `arrow` dataset with partitioning. Allows for `dplyr` grouped datasets (using `group_by`) and uses those variables to define partitions. Adopted from wcjochem/sfarrow

Value

`obj` invisibly

See Also

`write_dataset`, `st_read_parquet`

Index