

Package: sfarrow (via r-universe)

October 29, 2024

Title Read/Write Simple Feature Objects ('sf') with 'Apache' 'Arrow'

Version 0.4.1

Date 2021-10-25

Description Support for reading/writing simple feature ('sf') spatial objects from/to 'Parquet' files. 'Parquet' files are an open-source, column-oriented data storage format from Apache (<https://parquet.apache.org/>), now popular across programming languages. This implementation converts simple feature list geometries into well-known binary format for use by 'arrow', and coordinate reference system information is maintained in a standard metadata format.

License MIT + file LICENSE

URL <https://github.com/wcjochem/sfarrow>,
<https://wcjochem.github.io/sfarrow/>

BugReports <https://github.com/wcjochem/sfarrow/issues>

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports sf, arrow, jsonlite, dplyr,

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository <https://wcjochem.r-universe.dev>

RemoteUrl <https://github.com/wcjochem/sfarrow>

RemoteRef HEAD

RemoteSha aff5deed42bec078799bc46a43975badb59a05eb

Contents

read_sf_dataset	2
st_read_feather	3
st_read_parquet	4
st_write_feather	5
st_write_parquet	6
write_sf_dataset	7
Index	9

read_sf_dataset	<i>Read an Arrow multi-file dataset and create sf object</i>
-----------------	--

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 <code>arrow::open_dataset</code> or an <code>arrow_dplyr_query</code>
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.

Value

object of class `sf`

See Also

[open_dataset](#), [st_read](#), [st_read_parquet](#)

Examples

```
# read spatial object
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)

# create random grouping
nc$group <- sample(1:3, nrow(nc), replace = TRUE)

# use dplyr to group the dataset. %>% also allowed
nc_g <- dplyr::group_by(nc, group)

# write out to parquet datasets
tf <- tempfile() # create temporary location
on.exit(unlink(tf))
# partitioning determined by dplyr 'group_vars'
write_sf_dataset(nc_g, path = tf)

list.files(tf, recursive = TRUE)

# open parquet files from dataset
ds <- arrow::open_dataset(tf)

# create a query. %>% also allowed
q <- dplyr::filter(ds, group == 1)

# read the dataset (piping syntax also works)
nc_d <- read_sf_dataset(dataset = q)

nc_d
plot(sf::st_geometry(nc_d))
```

st_read_feather	<i>Read a Feather file to sf object</i>
-----------------	---

Description

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

Usage

```
st_read_feather(dsn, col_select = 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
...	additional parameters to pass to FeatherReader

Details

Reference for the metadata used: <https://github.com/geopandas/geo-arrow-spec>. These are standard with the Python GeoPandas library.

Value

object of class `sf`

See Also

[read_feather](#), [st_read](#)

Examples

```
# load Natural Earth low-res dataset.
# Created in Python with GeoPandas.to_feather()
path <- system.file("extdata", package = "sfarrow")

world <- st_read_feather(file.path(path, "world.feather"))

world
plot(sf::st_geometry(world))
```

st_read_parquet	<i>Read a Parquet file to sf object</i>
-----------------	---

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.

Value

object of class [sf](#)

See Also

[read_parquet](#), [st_read](#)

Examples

```
# load Natural Earth low-res dataset.
# Created in Python with GeoPandas.to_parquet()
path <- system.file("extdata", package = "sfarrow")

world <- st_read_parquet(file.path(path, "world.parquet"))

world
plot(sf::st_geometry(world))
```

st_write_feather	<i>Write sf object to Feather file</i>
------------------	--

Description

Convert a simple features spatial object from [sf](#) and write to a Feather file using [write_feather](#). Geometry columns (type [sfc](#)) are converted to well-known binary (WKB) format.

Usage

```
st_write_feather(obj, dsn, ...)
```

Arguments

<code>obj</code>	object of class sf
<code>dsn</code>	data source name. A path and file name with <code>.parquet</code> extension
<code>...</code>	additional options to pass to write_feather

Value

`obj` invisibly

See Also

[write_feather](#)

Examples

```
# read spatial object
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)

# create temp file
tf <- tempfile(fileext = '.feather')
on.exit(unlink(tf))

# write out object
st_write_feather(obj = nc, dsn = tf)

# In Python, read the new file with geopandas.read_feather(...)
# read back into R
nc_f <- st_read_feather(tf)
```

st_write_parquet	<i>Write sf object to Parquet file</i>
------------------	--

Description

Convert a simple features spatial object from `sf` and write 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, ...)
```

Arguments

<code>obj</code>	object of class <code>sf</code>
<code>dsn</code>	data source name. A path and file name with <code>.parquet</code> extension
<code>...</code>	additional options to pass to <code>write_parquet</code>

Value

`obj` invisibly

See Also

[write_parquet](#)

Examples

```
# read spatial object
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)

# create temp file
tf <- tempfile(fileext = '.parquet')
on.exit(unlink(tf))

# write out object
st_write_parquet(obj = nc, dsn = tf)

# In Python, read the new file with geopandas.read_parquet(...)
# read back into R
nc_p <- st_read_parquet(tf)
```

write_sf_dataset	<i>Write sf object to an Arrow multi-file dataset</i>
------------------	---

Description

Write sf object to an Arrow multi-file dataset

Usage

```
write_sf_dataset(
  obj,
  path,
  format = "parquet",
  partitioning = dplyr::group_vars(obj),
  ...
)
```

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 <code>dplyr::group_vars</code>
...	additional arguments and options passed to <code>arrow::write_dataset</code>

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.

Value

obj invisibly

See Also

[write_dataset](#), [st_read_parquet](#)

Examples

```
# read spatial object
nc <- sf::st_read(system.file("shape/nc.shp", package="sf"), quiet = TRUE)

# create random grouping
nc$group <- sample(1:3, nrow(nc), replace = TRUE)

# use dplyr to group the dataset. %>% also allowed
nc_g <- dplyr::group_by(nc, group)

# write out to parquet datasets
tf <- tempfile() # create temporary location
on.exit(unlink(tf))
# partitioning determined by dplyr 'group_vars'
write_sf_dataset(nc_g, path = tf)

list.files(tf, recursive = TRUE)

# open parquet files from dataset
ds <- arrow::open_dataset(tf)

# create a query. %>% also allowed
q <- dplyr::filter(ds, group == 1)

# read the dataset (piping syntax also works)
nc_d <- read_sf_dataset(dataset = q)

nc_d
plot(sf::st_geometry(nc_d))
```


Index

FeatherReader, [3](#)

filter, [2](#)

group_by, [7](#)

open_dataset, [2](#)

ParquetFileReader, [4](#)

read_feather, [4](#)

read_parquet, [4](#), [5](#)

read_sf_dataset, [2](#)

select, [2](#)

sf, [2](#), [4-7](#)

st_read, [2](#), [4](#), [5](#)

st_read_feather, [3](#)

st_read_parquet, [2](#), [4](#), [8](#)

st_write_feather, [5](#)

st_write_parquet, [6](#)

write_dataset, [8](#)

write_feather, [5](#)

write_parquet, [6](#)

write_sf_dataset, [7](#)