Package 'overtureR'

May 9, 2026

Title Load 'Overture' Datasets as 'dbplyr' and 'sf'-Ready Data Frames

Version 0.2.5

Description An integrated R interface to the 'Overture' API (<https://docs.overturemaps.org/>). Allows R users to return 'Overture' data as 'dbplyr' data frames or materialized 'sf' spatial data frames.

License MIT + file LICENSE

Suggests bench, duckdbfs, ggplot2, httr, jsonlite, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/arthurgailes/overtureR, https://arthurgailes.github.io/overtureR/

BugReports https://github.com/arthurgailes/overtureR/issues

Imports DBI, dbplyr, dplyr (>= 1.0.0), duckdb (>= 1.0.0), glue, rlang, sf

VignetteBuilder knitr

NeedsCompilation no

Author Arthur Gailes [aut, cre, cph] (ORCID: <https://orcid.org/0009-0006-8176-8653>)

Maintainer Arthur Gailes <agailes1@gmail.com>

Repository CRAN

Date/Publication 2025-04-21 14:00:02 UTC

Contents

`as_overture` *Convert a tbl_sql object to a overture_call object*

Description

This function adds the overture_call class to a tbl_sql object. It is primarily used internally#' by the open_curtain() function but can also be used directly on tbl_sql #' objects representing Overture Maps data.

Usage

```
as_overture(x, type, theme = get_theme_from_type(type))
```

Arguments

`x`	A tbl_sql object representing an Overture Maps dataset.
`type`	A string specifying the type of overture dataset to read. Setting to "*" or `NULL` will read all types for a given theme.
`theme`	Inferred from type by default. Must be set if type is "*" or NULL

Details

The function adds the overture_call class as the first class of the object

Value

A tbl_sql object with the additional class overture_call and attributes overture_type and overture_theme.

Examples

```
# The open_curtain() function already uses as_overture() internally,
# but you can also use it directly:
conn <- stage_conn()
division <- open_curtain("division", tablename = "test")

class(division)

# views
division2 <- tbl(conn, "test")
division2 <- as_overture(division2)
```

```
exit_stage(conn)
```

`collect.overture_call` *Convert dbplyr table to sf Object*

Description

Collects a lazy dbplyr view and materializes it as an in-memory `sf` table. `collect_sf` is a deprecated alias.

Usage

```
## S3 method for class 'overture_call'
collect(x, ..., geom_col = "geometry", crs = 4326)

collect_sf(...)
```

Arguments

`x`	A lazy data frame backed by a database query.
`...`	Further arguments passed to `dplyr::collect()`.
`geom_col`	The name of the geometry column. Will auto-detect names matching 'geom'.
`crs`	The coordinate reference system to use for the geometries, specified by its EPSG code. The default is 4326 (WGS 84).

Value

An 'sf' object with the dataset converted to spatial features.

Examples

```
bbox <- c(xmin = -120.5, ymin = 35.5, xmax = -120.0, ymax = 36.0)
lazy_tbl <- open_curtain("building", bbox)
collect(lazy_tbl)
```

config_extensions *Check duckdb extension and config settings*

Description

Check duckdb extension and config settings

Usage

```
config_extensions(conn)
```

Arguments

conn	A connection to a duckdb database.

open_curtain *Retrieve (Spatially Filtered) Overture Datasets*

Description

Fetches overture data from AWS. If a bounding box is provided, it applies spatial filtering to only include records within that area. The core code is copied from duckdbfs, which deserves all credit for the implementation

Usage

```
open_curtain(
  type,
  spatial_filter = NULL,
  theme = get_theme_from_type(type),
  conn = NULL,
  as_sf = FALSE,
  mode = "view",
  tablename = NULL,
  read_opts = list(),
  base_url = "s3://overturemaps-us-west-2/release/2025-03-19.0",
  bbox = NULL
)
```

Arguments

type	A string specifying the type of overture dataset to read. Setting to "*" or NULL will read all types for a given theme.
spatial_filter	An object to spatially filter the result.
theme	Inferred from type by default. Must be set if type is "*" or NULL

conn	A connection to a duckdb database.
as_sf	If TRUE, return an sf dataframe
mode	Either "view" (default) or "table". If "table", will download the dataset into memory.
tablename	The name of the table to create in the database.
read_opts	A named list of key-value pairs passed to DuckDB's read_parquet
base_url	Allows user to download data from a different mirror, such as a local directory, or a alternative release.
bbox	alias for spatial_filter. may be deprecated in the future.

Value

An dbplyr lazy dataframe, or an sf dataframe if as_sf is TRUE

Examples

```
bbox <- c(xmin = -120.5, ymin = 35.5, xmax = -120.0, ymax = 36.0)
open_curtain("building", bbox)
```

---

record_overture *Download Overture Maps Data to Local Directory*

Description

This function downloads Overture Maps data to a local directory, maintaining the same partition structure as in S3. snapshot_overture defaults 'output_dir' to tempdir() and overwrite to TRUE.

Usage

```
record_overture(curtain_call, output_dir, overwrite = FALSE, write_opts = NULL)

snapshot_overture(
  curtain_call,
  output_dir = tempdir(),
  overwrite = TRUE,
  write_opts = NULL
)
```

Arguments

curtain_call	A overture_call object.
output_dir	The directory where the data will be saved.
overwrite	Logical, if FALSE (default), existing directories will not be overwritten.
write_opts	a character vector passed to DuckDB's COPY command.

Value

Another tbl_lazy. Use `dplyr::show_query()` to see the generated query, and use `dplyr::collect()` to execute the query and return data to R.

An 'overture_call' for the downloaded data

See Also

DuckDB documentation on partitioned writes

Examples

```
broadway <- c(xmin = -73.99, ymin = 40.76, xmax = -73.98, ymax = 40.76)
buildings <- open_curtain("building", spatial_filter = bbox)
local_buildings <- record_overture(buildings, tempdir(), overwrite = TRUE)
```

`sf_as_dbplyr` *Registeran sf object as a DuckDB virtual table*

Description

A thin wrapper around `duckdb::duckdb_register()` that creates a virtual table, then selects the geometry column to DuckDB.'s GEOMETRY type in the returned `dbplyr` representation. Mostly useful for join and spatial operations within DuckDB. No data is copied.

Usage

```
sf_as_dbplyr(
  conn,
  name,
  sf_obj,
  geom_only = isFALSE(inherits(sf_obj, "sf")),
  overwrite = FALSE,
  ...
)
```

Arguments

`conn`	A DuckDB connection, created by `dbConnect()`.
`name`	The name for the virtual table that is registered or unregistered
`sf_obj`	sf object to be registered to duckdb
`geom_only`	if TRUE, only the geometry column is registered. Always FALSE for sfc or sfg objects
`overwrite`	Should an existing registration be overwritten?
`...`	additional arguments passed to duckdb_register

Details

Behind the scenes, this function creates an initial view (name_init) with the geometry stored as text via sf::st_as_text. It then creates the view name which replaces the geometry column with DuckDB's internal geometry type.

Value

a dbplyr lazy table

Examples

```
library(sf)

con <- stage_conn()
sf_obj <- st_sf(a = 3, geometry = st_sfc(st_point(1:2)))
sf_as_dbplyr(con, "test", sf_obj)

DBI::dbDisconnect(con)
```

---

stage_conn *create a cachable duckdb connection. In dev*

Description

stage_conn is primarily intended for internal use by other overtureR functions. However, it can be called directly by the user whenever it is desirable to have direct access to the connection object. The core code is copied from duckdbfs, which deserves all credit for the implementation

Usage

```
stage_conn(
  dbdir = ":memory:",
  read_only = FALSE,
  bigint = "numeric",
  config = list(),
  ...
)

strike_stage(conn = stage_conn())
```

Arguments

dbdir	Location for database files. Should be a path to an existing directory in the file system. With the default (or ""), all data is kept in RAM.

read_only	Set to TRUE for read-only operation. For file-based databases, this is only applied when the database file is opened for the first time. Subsequent connections (via the same drv object or a drv object pointing to the same path) will silently ignore this flag.
bigint	How 64-bit integers should be returned. There are two options: "numeric" and "integer64". If "numeric" is selected, bigint integers will be treated as double/numeric. If "integer64" is selected, bigint integers will be set to bit64 encoding.
config	Named list with DuckDB configuration flags, see https://duckdb.org/docs/configuration/overview#configuration-reference for the possible options. These flags are only applied when the database object is instantiated. Subsequent connections will silently ignore these flags.
...	Further arguments passed to DBI::dbConnect
conn	A duckdb_connection object

Details

When first called (by a user or internal function), this function both creates a duckdb connection and places that connection into a cache (overturer_conn option). On subsequent calls, this function returns the cached connection, rather than recreating a fresh connection.

This frees the user from the responsibility of managing a connection object, because functions needing access to the connection can use this to create or access the existing connection. At the close of the global environment, this function's finalizer should gracefully shutdown the connection before removing the cache.

strike_stage closes the connection.

Value

a duckdb::duckdb()connection object

Examples

```
con <- stage_conn()
strike_stage(con)
```

Index