Package 'overtureR'

October 15, 2026

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

Version 0.3.2

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, curl, duckdbfs, ggplot2, httr, jsonlite, knitr, rmarkdown, spelling, testthat (>= 3.0.0), withr

Config/Needs/website btw, ellmer, mapgl, purrr, tibble

Config/testthat/edition 3

Encoding UTF-8

Language en-US

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 (>= 1.1.0), glue, rlang, sf

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

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 2026-10-15 19:30:02 UTC

Contents

as_overture . . . 2
clear_overture_cache . . . 3
collect.overture_call . . . 4
config_extensions . . . 5
latest_overture_release . . . 5
open_curtain . . . 6
overture_releases . . . 7
overture_sf_methods . . . 8
overture_types . . . 9
record_overture . . . 9
sf_as_dbplyr . . . 11
stage_conn . . . 12

Index **14**

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

Description

Adds the `overture_call` class to a `tbl_sql` object. `open_curtain()` does this for you; call it directly on a lazy table of Overture data you built yourself, so that `collect()` returns `sf` and `record_overture()` knows the type and theme of the data.

Usage

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

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. See `overture_types()` for the valid values.
`theme`	Inferred from type by default. Must be set if type is "*" or `NULL`.
`release`	The Overture release the data came from, such as `"2026-08-19.0"`, or `NULL` if unknown.

Value

A tbl_sql object with the additional class `overture_call` and an `overture_playbill` attribute: a list with `type`, `theme` and `release`.

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, "division")

strike_stage(conn)
```

---

clear_overture_cache *Clear overtureR's catalog cache*

Description

overtureR caches what it reads from Overture's STAC catalog (the list of types in a release, and the bounding box of every Parquet file) in memory and on disk under `tools::R_user_dir("overtureR", "cache")`. Releases never change, so the cache never goes stale, but you can clear it here. Set `options(overturer_cache = FALSE)` to keep the cache in memory only, or `options(overturer_cache_dir = )` to move it.

Usage

```
clear_overture_cache()
```

Value

The cache directory, invisibly.

Examples

```
## Not run:
clear_overture_cache()

## End(Not run)
```

`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).

Details

The geometry column is read back as well-known binary and converted with `sf::st_as_sfc()`. If the column is already binary (for example, after a `mutate(geometry = ST_AsWKB(geometry))`), it is used as is. If it is neither DuckDB `GEOMETRY` nor binary, the result is returned as a plain data frame.

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.

latest_overture_release *Discover the latest available Overture Maps release*

Description

`open_curtain()` needs a release date/version to build its S3 path (e.g. `"2026-08-19.0"`). Overture cuts a new release roughly monthly, so hardcoding one quickly goes stale. This queries Overture's STAC catalog (https://stac.overturemaps.org/catalog.json) for its current `latest` release, so callers don't have to track releases themselves or wait on a package update. The result is cached for the session.

Usage

```
latest_overture_release(conn = NULL, refresh = FALSE)
```

Arguments

conn	A duckdb connection. Uses the cached session connection by default.
refresh	If `TRUE`, bypass the session cache and re-query the catalog.

Details

If the catalog can't be reached, the newest release in the package's local catalog cache is used with a warning (Overture removes releases after a few months, so it may itself be gone). With no cache, the call fails with an error; pass `base_url` to `open_curtain()` to work offline or from a local copy.

Value

A string identifying the latest release, e.g. `"2026-08-19.0"`.

Examples

```
latest_overture_release()
```

`open_curtain` *Retrieve (Spatially Filtered) Overture Datasets*

Description

Fetches overture data from AWS. If a spatial filter 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(),
  predicate = "intersects",
  release = NULL,
  base_url = NULL,
  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. See `overture_types()` for the valid values.
`spatial_filter`	An object to spatially filter the result: a named numeric vector or `sf::st_bbox()` bounding box, an `sf` or `sfc` object, the name of a table in `conn`, or another `dbplyr` lazy table with a `geometry` column. `sf` filters in another coordinate reference system are transformed to EPSG:4326 (Overture's) before filtering.
`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. union_by_name defaults to TRUE when type is "*", because the types in a theme have different columns.
predicate	How a feature must relate to spatial_filter to be kept: "intersects" (default), "within" (the feature lies entirely inside the filter) or "contains" (the feature contains the whole filter).
release	An Overture release, such as "2026-08-19.0". Defaults to getOption("overturer_release"), then to the latest release found by latest_overture_release(). See overture_releases() for the releases Overture still hosts. When base_url is set, release only labels the result.
base_url	Read from a different mirror, such as a local directory from record_overture(). Defaults to the S3 path of release.
bbox	alias for spatial_filter. may be deprecated in the future.

Details

When spatial_filter is set and base_url points at an Overture release or at a directory written by record_overture(), open_curtain() reads only the Parquet files whose bounding box touches the filter, using the file list in Overture's STAC catalog or the local copy's manifest (see overture_types() and clear_overture_cache()). This turns a cold query over hundreds of files into one over a handful. Set options(overturer_prune = FALSE) to always read the whole partition.

To pin every query in a session to one release, set options(overturer_release = "2026-08-19.0").

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)

# pin a release so the script returns the same rows next month
open_curtain("building", bbox, release = "2026-08-19.0")

# only buildings entirely inside the box
open_curtain("building", bbox, predicate = "within")
```

---

overture_releases *List the Overture releases still online*

Description

Overture publishes a release about once a month and removes old ones after a few months. This reads the releases its STAC catalog currently lists, so you can pick one for open_curtain(release =) or check that a pinned release is still available.

Usage

```
overture_releases(conn = NULL)
```

Arguments

conn	A duckdb connection. Uses the cached session connection by default.

Value

A character vector of releases, newest first, such as `c("2026-08-19.0", "2026-07-22.0")`.

Examples

```
overture_releases()
```

---

overture_sf_methods *Extent and coordinate reference system without collecting*

Description

`sf::st_bbox()` and `sf::st_crs()` methods for lazy `overture_call` tables, so you can read a query's extent and coordinate reference system straight from DuckDB, without pulling the rows into R with `collect()`.

Usage

```
## S3 method for class 'overture_call'
st_crs(x, ...)

## S3 method for class 'overture_call'
st_bbox(obj, ...)
```

Arguments

...	Unused, for compatibility with the generics.
obj, x	An `overture_call` object, as returned by `open_curtain()`.

Details

`st_bbox()` runs one `ST_Extent_Agg()` query over the geometry column. `st_crs()` reads the coordinate reference system from DuckDB's typed `GEOMETRY` column (for example `GEOMETRY('OGC:CRS84')` on duckdb >= 1.5), falling back to EPSG:4326, which is what Overture stores.

Value

`st_bbox()` returns an `sf::st_bbox()` object; `st_crs()` returns an `sf::st_crs()` object.

Examples

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

`overture_types` *List the dataset types in an Overture release*

Description

Reads the type-to-theme table for a release from Overture's STAC catalog, so new types (such as `bathymetry`) appear without a package update. The answer is cached per release. If the catalog can't be reached, the package's built-in table is returned with a warning.

Usage

```
overture_types(release = latest_overture_release(conn), conn = NULL)
```

Arguments

`release`	An Overture release, such as `"2026-08-19.0"`. Defaults to the latest release.
`conn`	A duckdb connection. Uses the cached session connection by default.

Value

A data frame with columns `type` and `theme`.

Examples

```
overture_types()
```

`record_overture` *Download Overture Maps data to a local directory*

Description

Writes the rows of an `overture_call` to Parquet files under `output_dir`, in Overture's own `theme=<theme>/type=<type>/` layout, and returns a new `overture_call` that reads from the copy. Each `type=` directory also gets an `_overture.json` manifest recording the source release and the bounding box of every file, so `open_curtain()` on the copy skips files by location just as it does on S3. `snapshot_overture()` defaults `output_dir` to `tempdir()` and `overwrite` to `TRUE`.

Usage

```
record_overture(
  curtain_call,
  output_dir,
  overwrite = FALSE,
  write_opts = NULL,
  partition_by = NULL,
  grid = NULL,
  spatial_filter = NULL,
  ...
)

snapshot_overture(
  curtain_call,
  output_dir = tempdir(),
  overwrite = TRUE,
  write_opts = NULL,
  partition_by = NULL,
  grid = NULL,
  spatial_filter = NULL,
  ...
)
```

Arguments

curtain_call	An overture_call object, or the name of an Overture type (such as "building") to open with open_curtain() first.
output_dir	The directory where the data will be saved.
overwrite	If FALSE (default), output_dir must be empty. If TRUE, the theme=/type= directories being written are replaced; other files in output_dir are left alone.
write_opts	A character vector of extra options for DuckDB's COPY command, such as "ROW_GROUP_SIZE 100000". Use partition_by, not PARTITION_BY, to change the layout.
partition_by	Names of columns to partition by, below theme and type. Add columns with dplyr::mutate() first if needed.
grid	Cell size in degrees. If set, the copy is further partitioned into a grid of that size (columns x_cell and y_cell, the cell's south-west corner), so each file covers a compact area and a later spatial_filter skips most of them.
spatial_filter, ...	Passed to open_curtain() when curtain_call is a type name.

Value

An overture_call reading from the downloaded data. Use dplyr::show_query() to see its query and dplyr::collect() to bring the rows into R.

See Also

DuckDB documentation on partitioned writes

Examples

```
broadway <- c(xmin = -73.99, ymin = 40.755, xmax = -73.98, ymax = 40.762)
buildings <- open_curtain("building", spatial_filter = broadway)
local_buildings <- record_overture(buildings, tempdir(), overwrite = TRUE)

# or in one call
local_buildings <- record_overture(
  "building", tempdir(), overwrite = TRUE, spatial_filter = broadway
)
```

`sf_as_dbplyr` *Register an sf object as a DuckDB virtual table*

Description

A thin wrapper around `duckdb::duckdb_register()` that creates a virtual table, then casts 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 TRUE 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 well-known binary via `sf::st_as_binary`. It then creates the view `name` which replaces the geometry column with DuckDB's internal geometry type. DuckDB geometries carry no coordinate reference system; the geometry is registered in whatever system `sf_obj` uses.

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 or reuse a cached DuckDB connection*

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 = getOption("overturer_conn", NULL))
```

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. Defaults to the cached session connection, if there is one.

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. The `dbdir`, `read_only`, `bigint`, and `config` arguments only take effect when a connection is created.

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

as_overture, 2

clear_overture_cache, 3
clear_overture_cache(), *7*
collect(), *2*, *8*
collect.overture_call, 4
collect_sf *(*collect.overture_call*)*, 4
config_extensions, 5

DBI::dbConnect, *13*
dplyr::collect(), *4*, *10*
dplyr::mutate(), *10*
dplyr::show_query(), *10*
duckdb::duckdb(), *13*

latest_overture_release, 5
latest_overture_release(), *7*

open_curtain, 6
open_curtain(), *2*, *8–10*
overture_releases, 7
overture_releases(), *7*
overture_sf_methods, 8
overture_types, 9
overture_types(), *2*, *6*, *7*

record_overture, 9
record_overture(), *2*, *7*

sf_as_dbplyr, 11
snapshot_overture *(*record_overture*)*, 9
st_bbox.overture_call
        *(*overture_sf_methods*)*, 8
st_crs.overture_call
        *(*overture_sf_methods*)*, 8
stage_conn, 12
strike_stage *(*stage_conn*)*, 12