Package: freestiler (via r-universe)

June 29, 2026

Title Create Vector Tiles from Spatial Data

Version 0.3.0

Date 2026-06-24

Description A forked, windows-specific build of the original freestiler package which provides features to create vector tile archives in 'PMTiles' format from 'sf' spatial data frames. Supports 'Mapbox Vector Tile' ('MVT') and 'MapLibre Tile' ('MLT') output formats. Uses a 'Rust' backend via 'extendr' for fast, in-memory tiling with zero external system dependencies.

License MIT + file LICENSE

URL https://github.com/jimbrig/freestiler, http://docs.jimbrig.com/freestiler/

BugReports https://github.com/jimbrig/freestiler/issues

Depends R (>= 4.2)

Imports DBI, duckdb, httpuv, jsonlite, mapgl, sf, tools, viridisLite

Suggests arrow, knitr, quarto, stats, testthat (>= 3.0.0), withr

Config/rextendr/version 0.4.2.9000

Config/roxygen2/version 8.0.0

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

SystemRequirements Cargo (Rust's package manager), rustc >= 1.77.2

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev make libpng-dev libuv1-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev xz-utils zlib1g-dev libclang-dev

Repository https://jimbrig.r-universe.dev

Date/Publication 2026-06-29 00:45:22 UTC

RemoteUrl https://github.com/jimbrig/freestiler

RemoteRef HEAD

RemoteSha 3df643687d7c61e3ccb7781cf4c595545652f9ae

Contents

`freestile` *Create vector tiles from spatial data*

Description

Creates a PMTiles archive containing vector tiles from one or more sf data frames. Supports both Mapbox Vector Tile (MVT) and MapLibre Tile (MLT) formats, multi-layer output, feature dropping, point clustering, and feature coalescing.

Usage

```
freestile(
  input,
  output,
  layer_name = NULL,
  tile_format = "mvt",
  min_zoom = 0L,
  max_zoom = 14L,
  base_zoom = NULL,
  drop_rate = NULL,
  cluster_distance = NULL,
  cluster_maxzoom = NULL,
  coalesce = FALSE,
  simplification = TRUE,
  generate_ids = TRUE,
  overwrite = TRUE,
  quiet = FALSE
)
```

Arguments

`input`	An sf data frame, or a named list of sf/freestile_layer objects for multi-layer output.
`output`	Character. Path for the output .pmtiles file.

`layer_name` Character. Name for the tile layer. If NULL, derived from the output filename. Only used for single-layer input.

`tile_format` Character. Tile encoding format: `"mvt"` (default) for Mapbox Vector Tiles or `"mlt"` for MapLibre Tiles.

`min_zoom` Integer. Minimum zoom level (default 0).

`max_zoom` Integer. Maximum zoom level (default 14).

`base_zoom` Integer. Zoom level at and above which all features are present (no dropping). NULL (default) uses each layer's own max_zoom. The drop-rate curve is also computed relative to base_zoom, so lowering it produces gentler thinning at low zooms. Inspired by tippecanoe's `-B` / `--base-zoom`.

`drop_rate` Numeric. Exponential drop rate for feature thinning (e.g. 2.5). At each zoom level below base_zoom, features are retained at a rate of 1/drop_rate^(base_zoom - zoom). Points are thinned using spatial ordering; polygons/lines are thinned by area. NULL (default) disables drop-rate thinning.

`cluster_distance` Numeric. Pixel distance for point clustering. Points within this radius are merged into cluster features with a `point_count` attribute. NULL (default) disables clustering.

`cluster_maxzoom` Integer. Maximum zoom level for clustering. Above this zoom, individual points are shown. Default is max_zoom - 1.

`coalesce` Logical. Whether to merge features with identical attributes within each tile (default FALSE). Lines sharing endpoints are merged; polygons are grouped into MultiPolygons.

`simplification` Logical. Whether to snap geometries to the tile pixel grid at each zoom level (default TRUE). This provides zoom-adaptive simplification and prevents slivers between adjacent polygons.

`generate_ids` Logical. Whether to assign sequential feature IDs (default TRUE).

`overwrite` Logical. Whether to overwrite existing output file (default TRUE).

`quiet` Logical. Whether to suppress progress messages (default FALSE).

Details

Input data in any coordinate reference system (CRS) is automatically reprojected to WGS84 (EPSG:4326) before tiling.

Value

The output file path (invisibly).

Examples

```
## Not run:
library(sf)
nc <- st_read(system.file("shape/nc.shp", package = "sf"))
```

```
# Single layer
freestile(nc, "nc.pmtiles", layer_name = "counties")

# Multi-layer
pts <- st_centroid(nc)
freestile(
  list(counties = nc, centroids = pts),
  "nc_layers.pmtiles"
)

# With dropping and coalescing
freestile(nc, "nc_drop.pmtiles", drop_rate = 2.5, coalesce = TRUE)

# With point clustering
freestile(pts, "pts.pmtiles", cluster_distance = 50, cluster_maxzoom = 8)

## End(Not run)
```

`freestile_file` *Create vector tiles from a spatial file*

Description

Reads a GeoParquet, GeoPackage, Shapefile, or other spatial file directly into the tiling engine. Input data in any coordinate reference system is automatically reprojected to WGS84 (EPSG:4326) before tiling.

Usage

```
freestile_file(
  input,
  output,
  layer_name = NULL,
  tile_format = "mvt",
  min_zoom = 0L,
  max_zoom = 14L,
  base_zoom = NULL,
  drop_rate = NULL,
  cluster_distance = NULL,
  cluster_maxzoom = NULL,
  coalesce = FALSE,
  simplification = TRUE,
  overwrite = TRUE,
  quiet = FALSE,
  engine = "geoparquet"
)
```

Arguments

input	Character. Path to the input spatial file.
output	Character. Path for the output .pmtiles file.
layer_name	Character. Name for the tile layer. If NULL, derived from the output filename.
tile_format	Character. "mvt" (default) or "mlt".
min_zoom	Integer. Minimum zoom level (default 0).
max_zoom	Integer. Maximum zoom level (default 14).
base_zoom	Integer. Zoom level at and above which all features are present. NULL (default) uses max_zoom.
drop_rate	Numeric. Exponential drop rate. NULL (default) disables.
cluster_distance	Numeric. Pixel distance for clustering. NULL disables.
cluster_maxzoom	Integer. Max zoom for clustering. Default max_zoom - 1.
coalesce	Logical. Whether to merge features with identical attributes (default FALSE).
simplification	Logical. Whether to snap geometries to the tile pixel grid (default TRUE).
overwrite	Logical. Whether to overwrite existing output (default TRUE).
quiet	Logical. Whether to suppress progress (default FALSE).
engine	Character. Backend engine: "geoparquet" (default, for GeoParquet files) or "duckdb" (for any file format DuckDB supports).

Details

The GeoParquet engine requires compilation with FREESTILER_GEOPARQUET=true. The DuckDB engine uses the Rust DuckDB backend when included in the build (enabled by default for native builds), or falls back to the R duckdb package. Control backend selection with options(freestiler.duckdb_backend = "auto"|"rust"|"r").

Value

The output file path (invisibly).

Examples

```
## Not run:
freestile_file("data.parquet", "output.pmtiles")
freestile_file("data.gpkg", "output.pmtiles", engine = "duckdb")

## End(Not run)
```

freestile_h3 *Create vector tiles with dynamic H3 hexagonal binning*

Description

Aggregates points into H3 hexagons at zoom-appropriate resolutions and writes a PMTiles archive in which low zooms show coarse hexagons, intermediate zooms show progressively finer hexagons, and zooms at or above `base_zoom` show individual points. Aggregations (count, sum, mean, etc.) are computed in DuckDB via the H3 community extension; the function then assembles the per-resolution hex layers and the raw-point layer via `freestile()`.

Usage

```
freestile_h3(
  input,
  output,
  agg = "count",
  hex_layer_prefix = "h3",
  point_layer_name = "points",
  min_zoom = 0L,
  max_zoom = 14L,
  base_zoom = NULL,
  h3_resolutions = NULL,
  source_crs = NULL,
  db_path = NULL,
  tile_format = "mvt",
  fade = FALSE,
  fade_overlap = 1L,
  overwrite = TRUE,
  quiet = FALSE
)
```

Arguments

`input`	An `sf` data frame of POINT geometry, or a character SQL query that returns a geometry column when executed against DuckDB. (DuckDB's spatial functions such as `ST_Read()` and `read_parquet()` are available.)
`output`	Character. Path for the output `.pmtiles` file.
`agg`	Aggregation specification: • `"count"` (default): single `count = COUNT(*)` property. • Named character vector of SQL aggregation expressions, e.g. `c(n = "COUNT(*)", avg_pop = "AVG(pop)", total = "SUM(pop)")`. • Named list of `prop_name = c(fn, column)` for callers who don't want to write SQL, e.g. `list(n = c("count", "*"), avg_pop = c("mean", "pop"))`. Supported `fn` values: `"count"`, `"sum"`, `"mean"`/`"avg"`, `"min"`, `"max"`, `"median"`.

`hex_layer_prefix`
: Character. Prefix for the per-resolution hex MVT layer names. Default `"h3"` produces `"h3_r01"`, `"h3_r02"`, etc.

`point_layer_name`
: Character. MVT layer name for raw points (default `"points"`).

`min_zoom`, `max_zoom`
: Integer. Global zoom range (default 0–14).

`base_zoom`
: Integer or NULL. Zoom level at and above which raw points take over from hex aggregations. Default `NULL` resolves to `max_zoom - 2L`, clamped so `min_zoom <= base_zoom <= max_zoo` If `base_zoom == min_zoom`, no hex layers are created.

`h3_resolutions`
: Optional override of the zoom -> H3 resolution mapping. Accepts `NULL` (use built-in defaults), an unnamed integer vector with `length(min_zoom:(base_zoom - 1L))` entries mapped positionally, or a named integer vector with names that parse to integer zoom levels (sparse overrides; defaults fill the rest). All resolutions must be integers in `0:15`. The same resolution appearing in non-contiguous zoom runs (e.g. zooms 4–5 and 8) is rejected.

`source_crs`
: Character or NULL. CRS of geometry returned by SQL input, e.g. `"EPSG:4326"` (default for `sf` input, since `sf` is auto-transformed to WGS84) or `"EPSG:3857"`. Ignored for `sf` input. For SQL input, if `NULL`, a warning is emitted and the geometry is assumed to be `"EPSG:4326"`.

`db_path`
: Character or NULL. Path to an existing DuckDB database file, or `NULL`/`""` (default) for an in-memory database.

`tile_format`
: Character. `"mvt"` (default) or `"mlt"`.

`fade`
: Logical. If `TRUE`, adjacent hex layer zoom windows overlap by `fade_overlap` zooms so `view_h3_tiles()` can cross-fade between resolutions. Default `FALSE` (disjoint windows, clean zoom breaks).

`fade_overlap`
: Integer. Zooms of overlap on each side (only used when `fade = TRUE`, default `1L`).

`overwrite`
: Logical. Whether to overwrite an existing output file (default `TRUE`).

`quiet`
: Logical. Suppress progress messages (default `FALSE`).

Details

Each distinct H3 resolution becomes its own MVT source-layer (named `"<hex_layer_prefix>_r<resolution>"`, e.g. `"h3_r05"`); raw points are emitted as a separate source-layer (`point_layer_name`). With the default `fade = FALSE`, per-layer zoom windows are disjoint so the rendered map swaps between resolutions cleanly. With `fade = TRUE`, adjacent windows overlap by `fade_overlap` zooms so the companion `view_h3_tiles()` helper can cross-fade between resolutions.

DuckDB and the H3 community extension are required. With `sf` input the data is written to DuckDB via a temporary Parquet (or `dbWriteTable`) roundtrip; with character SQL input, the user query is wrapped in a temporary view inside DuckDB.

Value

The output file path (invisibly).

Point volume at `base_zoom`

The raw-point layer is encoded without thinning: every input point is written to every tile that contains it from `base_zoom` up (and from `base_zoom - fade_overlap` when `fade = TRUE`). For very large inputs the tiles at the first point zoom can get heavy. Raising `base_zoom` defers points to higher zooms, where each tile covers less area and so holds fewer points. Per-layer feature dropping for the points layer is on the roadmap.

Antimeridian and polar cells

Hexagons that cross the antimeridian are split at +/-180 degrees so they render correctly on both sides of the dateline instead of as world-spanning slivers. The rare cells that contain a pole receive the same split and render approximately (the polygon edge follows the cell boundary vertices, so the polar cap itself is not filled).

See Also

`freestile()`, `view_h3_tiles()`

Examples

```
## Not run:
library(sf)
pts <- st_as_sf(data.frame(
  x = runif(50000, -100, -80),
  y = runif(50000, 30, 45),
  w = rnorm(50000, 100, 10)
), coords = c("x", "y"), crs = 4326)

freestile_h3(pts, "wind.pmtiles",
  agg = c(n = "COUNT(*)", avg_w = "AVG(w)"),
  min_zoom = 2, max_zoom = 12, base_zoom = 10)

view_h3_tiles("wind.pmtiles", agg_column = "n")

# Cross-fade between resolutions
freestile_h3(pts, "wind_fade.pmtiles",
  agg = "count",
  min_zoom = 2, max_zoom = 12, base_zoom = 10,
  fade = TRUE)

## End(Not run)
```

`freestile_layer` *Create a layer specification with per-layer zoom range*

Description

Wraps an sf object with optional per-layer zoom range overrides for use in multi-layer tile generation.

Usage

```
freestile_layer(input, min_zoom = NULL, max_zoom = NULL)
```

Arguments

input	An sf data frame.
min_zoom	Integer. Minimum zoom level for this layer. If NULL, uses the global min_zoom from freestile().
max_zoom	Integer. Maximum zoom level for this layer. If NULL, uses the global max_zoom from freestile().

Value

A freestile_layer object (list with class attribute).

Examples

```
## Not run:
library(sf)
nc <- st_read(system.file("shape/nc.shp", package = "sf"))
roads <- st_read("roads.shp")

freestile(
  list(
    counties = freestile_layer(nc, min_zoom = 0, max_zoom = 10),
    roads = freestile_layer(roads, min_zoom = 8, max_zoom = 14)
  ),
  "layers.pmtiles"
)

## End(Not run)
```

---

freestile_query *Create vector tiles from a DuckDB SQL query*

Description

Executes a SQL query via DuckDB's spatial extension and pipes the results into the tiling engine. Uses the Rust DuckDB backend when included in the build (enabled by default for native builds), or falls back to the R duckdb package. Control backend selection with options(freestiler.duckdb_backend = "auto"|"rust"|"r").

Usage

```
freestile_query(
  query,
  output,
  db_path = NULL,
  layer_name = NULL,
  tile_format = "mvt",
  min_zoom = 0L,
  max_zoom = 14L,
  base_zoom = NULL,
  drop_rate = NULL,
  cluster_distance = NULL,
  cluster_maxzoom = NULL,
  coalesce = FALSE,
  simplification = TRUE,
  overwrite = TRUE,
  quiet = FALSE,
  source_crs = NULL,
  streaming = "auto"
)
```

Arguments

query	Character. A SQL query that returns a geometry column. DuckDB spatial functions like `ST_Read()` and `read_parquet()` are available. Multi-statement SQL is supported: setup statements (e.g., `LOAD h3;`) are executed first, then the final SELECT is used for tiling.
output	Character. Path for the output .pmtiles file.
db_path	Character. Path to a DuckDB database file, or NULL (default) for an in-memory database.
layer_name	Character. Name for the tile layer. If NULL, derived from the output filename.
tile_format	Character. `"mvt"` (default) or `"mlt"`.
min_zoom	Integer. Minimum zoom level (default 0).
max_zoom	Integer. Maximum zoom level (default 14).
base_zoom	Integer. Zoom level at and above which all features are present. NULL (default) uses max_zoom.
drop_rate	Numeric. Exponential drop rate. NULL (default) disables.
cluster_distance	Numeric. Pixel distance for clustering. NULL disables.
cluster_maxzoom	Integer. Max zoom for clustering. Default max_zoom - 1.
coalesce	Logical. Whether to merge features with identical attributes (default FALSE).
simplification	Logical. Whether to snap geometries to the tile pixel grid (default TRUE).
overwrite	Logical. Whether to overwrite existing output (default TRUE).

quiet	Logical. Whether to suppress progress (default FALSE).
source_crs	Character or NULL. CRS of the geometry returned by query, for example "EPSG:4326" or "EPSG:4267". Used only by the R duckdb fallback; ignored by the Rust DuckDB backend.
streaming	Character. DuckDB query execution mode: "auto" (default) enables the streaming point pipeline for large queries, "always" forces it, and "never" uses the existing in-memory path.

Details

When using the R fallback, source_crs must be supplied explicitly so the query result can be interpreted or reprojected correctly. Pass "EPSG:4326" if the SQL already returns WGS84 geometry, or the source CRS string (for example "EPSG:4267") to have DuckDB reproject to WGS84 before tiling. For file-based input where the CRS is embedded in the file, use freestile_file() with engine = "duckdb" instead, which auto-detects the source CRS.

Value

The output file path (invisibly).

Examples

```
## Not run:
# Query a GeoParquet file
freestile_query(
  "SELECT * FROM read_parquet('data.parquet') WHERE pop > 50000",
  "output.pmtiles"
)

# Query a Shapefile
freestile_query(
  "SELECT * FROM ST_Read('counties.shp')",
  "counties.pmtiles"
)

# Query with an existing DuckDB database
freestile_query(
  "SELECT * FROM my_table WHERE region = 'West'",
  "west.pmtiles",
  db_path = "my_database.duckdb"
)

## End(Not run)
```

`serve_tiles` *Serve PMTiles files via local HTTP server with CORS*

Description

Start a local HTTP server to serve PMTiles files with CORS headers and HTTP range request support. This allows PMTiles to be consumed by mapgl and MapLibre GL JS. The server runs in the background and can be stopped with `stop_server()`.

Usage

```
serve_tiles(path, port = 8080)
```

Arguments

`path`	Path to a directory containing PMTiles files, or a single PMTiles file. If a single file, its directory will be served.
`port`	Port number for the HTTP server. Default is 8080.

Details

If a server is already running on the requested port, it is stopped first.

The server uses httpuv (a dependency of Shiny) to serve static files with the CORS and range-request headers that PMTiles requires. Works well for files up to ~1 GB. For larger files, consider an external server like `npx http-server /path --cors -c-1`.

Value

Invisibly returns a list with `url`, `port`, and `dir`. The server handle is stored internally so it can be stopped with `stop_server()`.

See Also

`stop_server()`, `view_tiles()`

Examples

```
## Not run:
# Serve a directory
serve_tiles("/tmp/tiles")

# Serve a single file (its directory is served)
serve_tiles("us_bgs.pmtiles")

# Stop when done
stop_server()

## End(Not run)
```

stop_server *Stop a local tile server*

Description

Stop a local tile server

Usage

```
stop_server(port = NULL)
```

Arguments

port	Port number to stop, or NULL to stop all running servers.

Value

Invisibly returns TRUE if a server was stopped.

See Also

serve_tiles()

Examples

```
## Not run:
serve_tiles("tiles/")
stop_server()        # stop all
stop_server(8080)    # stop specific port

## End(Not run)
```

view_h3_tiles *View an H3 hexagonal-binning PMTiles archive*

Description

Reads the metadata from a PMTiles archive produced by freestile_h3() and builds a mapgl map with one fill layer per H3 resolution plus a circle layer for raw points. Automatically detects whether the archive was built with fade = FALSE (disjoint zoom windows, clean breaks) or fade = TRUE (overlapping windows, cross-fade) and styles accordingly.

Usage

```
view_h3_tiles(
  input,
  agg_column = NULL,
  stops = NULL,
  palette = "viridis",
  hex_opacity = 0.9,
  point_color = "#0868ac",
  point_radius = NULL,
  background_style = NULL,
  hex_layer_prefix = "h3",
  point_layer_name = "points",
  port = 8080
)
```

Arguments

input	Path to a local .pmtiles file produced by freestile_h3().
agg_column	Character or NULL. Aggregation column to drive the hex color scale. If NULL, the first non-h3_id numeric field in the metadata is used.
stops	List with values and colors (equal length, sorted by values) defining the shared color scale across all hex layers. If NULL, the documented quick-look default is used.
palette	Character. Named palette used to generate default colors when stops is NULL. Currently supports "viridis" (default), "magma", "plasma", "cividis", "inferno", "rocket", "mako", "turbo" if viridisLite is installed; otherwise falls back to a five-color blue ramp.
hex_opacity	Numeric. Peak fill opacity for hex layers (default 0.9). In fade mode this is the opacity at each layer's center zoom.
point_color	Character. Color for the raw-point circles (default "#0868ac").
point_radius	mapgl expression or NULL. If NULL, a zoom-interpolated default is used.
background_style	mapgl style passed to mapgl::maplibre(); if NULL uses mapgl::maplibre() defaults.
hex_layer_prefix	Character. Prefix used when the archive was written. Must match the value passed to freestile_h3(); default "h3".
point_layer_name	Character. Name of the raw-points MVT layer in the archive. Must match the value passed to freestile_h3(); default "points".
port	Integer. Port for the local PMTiles server (default 8080).

Details

The default color scale is a documented quick-look ramp (5 evenly spaced breaks across 1, 10, 100, 1000, 10000). For production maps, pass an explicit stops = list(values = ..., colors = ...) derived from your data. Embedding real aggregation statistics in PMTiles metadata is on the roadmap.

Value

A mapgl map object.

See Also

freestile_h3(), view_tiles()

Examples

```
## Not run:
view_h3_tiles("wind.pmtiles", agg_column = "n",
  stops = list(values = c(1, 10, 100, 1000, 10000),
               colors = viridisLite::viridis(5)))

## End(Not run)
```

---

view_tiles *Quickly view a PMTiles file on an interactive map*

Description

Starts a local tile server (if needed) and creates an interactive mapgl map showing the tileset. Layer type and styling are auto-detected from the PMTiles metadata when possible.

Usage

```
view_tiles(
  input,
  layer = NULL,
  layer_type = NULL,
  color = NULL,
  opacity = 0.5,
  port = 8080,
  promote_id = NULL
)
```

Arguments

input	Path to a local .pmtiles file.
layer	Character. Source layer name to display. If NULL (default), the first layer in the tileset is used.
layer_type	Character. Map layer type: "fill", "line", or "circle". If NULL, auto-detected from the PMTiles metadata geometry type.
color	Fill, line, or circle color. Default is "navy" for fill and line layers, "steelblue" for circle layers.

opacity	Numeric opacity (0–1). Default is 0.5.
port	Port for the local tile server. Default is 8080.
promote_id	Character. Property name to use as the feature ID for hover interactivity. If NULL, no feature promotion is used.

Value

A mapgl map object (can be piped into further mapgl operations).

See Also

serve_tiles(), freestile()

Examples

```
## Not run:
freestile(nc, "nc.pmtiles", layer_name = "counties")
view_tiles("nc.pmtiles")

# Override auto-detection
view_tiles("roads.pmtiles", layer_type = "line", color = "red")

# Point data
view_tiles("airports.pmtiles", layer_type = "circle", color = "orange")

## End(Not run)
```

Index