Package: nvdb2osmr (via r-universe)

May 26, 2026

Title Convert NVDB Data to OSM

Version 0.1.0

Description High-performance conversion of Swedish NVDB data to OpenStreetMap format. Uses DuckDB for spatial IO and Rust for topological simplification.

License MIT + file LICENSE

Imports DBI, duckdb, jsonlite, mirai, cli, glue

Suggests rosmium, testthat (>= 3.0.0), yaml

Remotes e-kotov/rosmium@f1fe5d46ae78745dec4d8c667a509424993c6c81

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

SystemRequirements Cargo (Rust package manager), rustc; the crate 'libR-sys' must compile without error. Osmium Tool (for PBF merging and sorting).

Config/rextendr/version 0.4.2

Config/pak/sysreqs cmake xz-utils libclang-dev

Repository https://e-kotov.r-universe.dev

Date/Publication 2026-02-25 11:09:13 UTC

RemoteUrl https://github.com/e-kotov/nvdb2osmr

RemoteRef HEAD

RemoteSha 86625fb879e1bd2e722c4cfda4eb4bd12624a6a2

Contents

`get_column_mappings` *Get NVDB Column Name Mappings*

Description

Returns a named list mapping short ASCII column names (as they appear in GDB/GeoParquet files) to long descriptive Swedish names. This is useful for documentation and understanding what each column represents.

Usage

```
get_column_mappings()
```

Details

The column mappings are stored in `inst/extdata/column_mappings.yaml` and intentionally contain non-ASCII characters (Swedish åääö) for human reference. Package code should always use the short ASCII names.

Value

A named list where names are short column names and values are long descriptive Swedish names.

Examples

```
## Not run:
mappings <- get_column_mappings()
mappings$Vagnr_10370  # "Driftbidrag statligt/Vägnr"

# Get all columns related to roads (väg)
mappings[grepl("vag", names(mappings), ignore.case = TRUE)]

## End(Not run)
```

`get_long_name` *Get Long Column Name*

Description

Look up the descriptive (long) name for a given short ASCII column name.

Usage

```
get_long_name(short_name)
```

Arguments

short_name	Character string with the short column name (e.g., "Vagnr_10370")

Value

Character string with the long descriptive name, or the input if not found

Examples

```
## Not run:
get_long_name("Vagnr_10370")  # "Driftbidrag statligt/Vägnr"
get_long_name("Klass_181")    # "Funktionell vägklass/Klass"

## End(Not run)
```

---

list_columns *List Available NVDB Columns*

Description

Returns a data frame with all available column mappings, useful for exploring the dataset structure.

Usage

```
list_columns(pattern = NULL)
```

Arguments

pattern	Optional regex pattern to filter column names

Value

A data frame with columns short_name and long_name

Examples

```
## Not run:
# All columns
list_columns()

# Columns related to speed (hastighet)
list_columns("hastighet")

# Columns with "vag" in the name
list_columns("vag")

## End(Not run)
```

nvdb_to_pbf *Convert NVDB data to OSM PBF using parallel processing (WKB optimized)*

Description

Convert NVDB data to OSM PBF using parallel processing (WKB optimized)

Usage

```
nvdb_to_pbf(
  input_path,
  output_pbf,
  municipality_codes = NULL,
  county_codes = NULL,
  split_by = c("municipality", "county", "none"),
  use_geoparquet = "auto",
  global_node_prepass = c("auto", "on", "off"),
  simplify_method = "refname",
  presplit = FALSE,
  max_retries = 2,
  duckdb_memory_limit_gb = 4,
  duckdb_threads = 1
)
```

Arguments

input_path	Path to input file (.gdb, .gpkg, or .geoparquet)
output_pbf	Path to final output .osm.pbf
municipality_codes	Optional vector of 4-digit municipality codes to process (default: all)
county_codes	Optional vector of 2-digit county codes to process (e.g., "01" for Stockholm)
split_by	Strategy for splitting the work: "municipality" (default), "county", or "none" (process whole file in one go).
use_geoparquet	Use GeoParquet for faster processing: "auto", TRUE, or FALSE (default: "auto")
global_node_prepass	Whether to build a global endpoint-node dictionary before split processing. One of "auto" (default), "on", or "off". For split processing (`split_by = "municipality"` or `"county"`), `"off"` is not allowed and raises an error because split mode requires global node prepass for boundary-safe connectivity.
presplit	Logical: whether to pre-split to temp files (default: FALSE)
max_retries	Maximum retries for failed municipalities (default: 2)
duckdb_memory_limit_gb	Memory limit for DuckDB in GB (numeric). Default 4.
duckdb_threads	Number of threads for DuckDB. Default 1 (ideal for parallel runs).

Details

This function supports parallel processing via the `mirai` package. To run in parallel, you must set up mirai daemons before calling this function, for example using `mirai::daemons(4)`. To shut down daemons after processing, call `mirai::daemons(0)`. If no daemons are configured, processing will happen sequentially.

Splitting by "municipality" is recommended for high-core counts as it provides more granular tasks (~290 tasks). "county" provides ~21 tasks. "none" handles everything in a single process (memory intensive for large areas).

Value

Path to output PBF file (invisibly)

`process_nvdb_fast` *Fast NVDB to PBF conversion using ported Rust algorithm (WKB optimized)*

Description

Fast NVDB to PBF conversion using ported Rust algorithm (WKB optimized)

Usage

```
process_nvdb_fast(
  gdb_path,
  output_pbf,
  municipality_code = NULL,
  county_code = NULL,
  simplify_method = "refname",
  node_id_start = 1L,
  way_id_start = 1L,
  duckdb_memory_limit_gb = 4,
  duckdb_threads = 1,
  verbose = TRUE,
  global_node_dict_path = NULL,
  area_code = NULL,
  prepass_rounding = "duckdb_1e7"
)
```

Arguments

`gdb_path`	Path to input file (GDB, GPKG, or GeoParquet)
`output_pbf`	Output PBF file path
`municipality_code`	4-digit municipality code to process (e.g., '2480')
`county_code`	2-digit county code to process (e.g., '24'). Used if municipality_code is NULL.

`simplify_method`
: Simplification method (default: "refname")

`node_id_start`
: Starting node ID for this chunk (default: 1)

`way_id_start`
: Starting way ID for this chunk (default: 1)

`duckdb_memory_limit_gb`
: Memory limit for DuckDB in GB (numeric). Default 4.

`duckdb_threads`
: Number of threads for DuckDB. Default 1.

`verbose`
: Print progress messages (default: TRUE)

`global_node_dict_path`
: Optional path to global endpoint-node dictionary parquet. If provided, per-segment start/end global node IDs are joined into the chunk. Required when `municipality_code` or `county_code` is supplied.

`area_code`
: Area code for this chunk (municipality or county code). Required when `global_node_dict_path` is provided.

`prepass_rounding`
: Rounding scheme for global node dictionary matching. Currently only `"duckdb_1e7"` is supported.

Value

Path to output PBF file (invisibly)

`process_nvdb_wkb` *Process NVDB data to OSM PBF (WKB optimized)*

Description

Optimized function using WKB geometries and direct R property columns. This avoids JSON serialization overhead for significant speedup.

Usage

```
process_nvdb_wkb(
  wkb_geoms,
  col_names,
  col_data,
  output_path,
  simplify_method = "refname",
  node_id_start = 1L,
  way_id_start = 1L
)
```

Arguments

wkb_geoms	List of raw WKB byte vectors (one per geometry)
col_names	Character vector of property column names
col_data	List of vectors (one per column), each same length as wkb_geoms
output_path	Path to write the output .osm.pbf file
simplify_method	Simplification method: "refname" (default), "recursive", "linear", "route", or "segment"
node_id_start	Starting ID for nodes (default: 1)
way_id_start	Starting ID for ways (default: 1)

Value

TRUE on success

Index