Package: gps2gtfs (via r-universe)

June 7, 2026

Type Package

Title High-Performance GPS to GTFS Converter

Version 0.1.0

Description Preprocesses raw GPS trajectory data of public transit and transforms it to GTFS format. Provides a high-performance R port of the 'gps2gtfs' Python package by Aaivu (Ratneswaran et al., 2023) <doi:10.1109/ICCT56969.2023.10075789>. Heavy computational tasks are offloaded to an extremely fast compiled Rust backend or a C++ (Rcpp) backend. Automatic backend selection prefers Rust, then Rcpp, and falls back to a slower pure R implementation utilizing 'data.table' and 'sf'.

License MIT + file LICENSE

Depends R (>= 4.2)

Imports data.table, Rcpp (>= 1.0.0)

Suggests sf, testthat (>= 3.0.0)

Config/testthat/edition 3

LinkingTo Rcpp

Config/rextendr/version 0.5.0

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

SystemRequirements Cargo (Rust's package manager), rustc >= 1.65.0, xz

Config/pak/sysreqs xz-utils libclang-dev

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

Date/Publication 2026-06-06 14:24:52 UTC

RemoteUrl https://github.com/e-kotov/gps2gtfs

RemoteRef HEAD

RemoteSha 41c0412ddc3b7c7b206a3c176b627c1bf857b617

Contents

`g2g_clean_gps` *Clean Raw GPS Data*

Description

Cleans raw GPS data by removing records with latitude or longitude equal to zero, parsing the device time, and sorting by device ID, date, and time.

Usage

```
g2g_clean_gps(raw_gps_df, projected = NULL)
```

Arguments

`raw_gps_df`	A data.frame containing raw GPS data. Must include columns: `id`, `deviceid`, `latitude`, `longitude`, `devicetime`, and `speed`.
`projected`	Logical. Is the coordinates data already projected? Default is NULL (auto-detect).

Value

A sorted `data.table` with additional `date` and `time_str` columns.

Examples

```
data(g2g_data_gps)
cleaned_gps <- g2g_clean_gps(g2g_data_gps)
head(cleaned_gps)
```

g2g_data_gps *GPS Trajectory Subset*

Description

A lightweight subset of raw bus GPS data from Kandy, Sri Lanka. This dataset is intended for examples and testing of the `gps2gtfs` pipeline.

Usage

```
g2g_data_gps
```

Format

A data frame with 1045 rows and 6 variables:

id Unique identifier for the GPS record.

deviceid Unique identifier for the tracking device (bus).

devicetime Timestamp of the GPS ping (UTC).

latitude Latitude in WGS-84 degrees.

longitude Longitude in WGS-84 degrees.

speed Recorded speed of the vehicle.

Source

Original data from the Python `gps2gtfs` package.

References

Ratneswaran, S., & Thayasivam, U. (2023). Extracting potential Travel time information from raw GPS data and Evaluating the Performance of Public transit - a case study in Kandy, Sri Lanka. *2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, 1-7. doi:10.1109/ICCT56969.2023.10075789

g2g_data_stops *Bus Stops Data*

Description

Sample data containing the coordinates and metadata for bus stops. This dataset works together with `g2g_data_gps` to extract stop times.

Usage

```
g2g_data_stops
```

Format

A data frame with 23 rows and 6 variables:

stop_id Unique identifier for the bus stop.

route_id Route identifier the stop belongs to.

direction Direction of the route the stop serves.

latitude Latitude in WGS-84 degrees.

longitude Longitude in WGS-84 degrees.

address Address or name of the stop location.

Source

Original data from the Python `gps2gtfs` package.

References

Ratneswaran, S., & Thayasivam, U. (2023). Extracting potential Travel time information from raw GPS data and Evaluating the Performance of Public transit - a case study in Kandy, Sri Lanka. *2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, 1-7. doi:10.1109/ICCT56969.2023.10075789

`g2g_data_terminals` *Bus Terminals Data*

Description

Sample data containing the coordinates for bus route terminals. Used to define the start and end of trips.

Usage

```
g2g_data_terminals
```

Format

A data frame with 2 rows and 4 variables:

terminal_id Unique identifier for the bus terminal.

terminal_name Name of the terminal.

latitude Latitude in WGS-84 degrees.

longitude Longitude in WGS-84 degrees.

Source

Original data from the Python `gps2gtfs` package.

References

Ratneswaran, S., & Thayasivam, U. (2023). Extracting potential Travel time information from raw GPS data and Evaluating the Performance of Public transit - a case study in Kandy, Sri Lanka. *2023 3rd International Conference on Intelligent Communication and Computational Techniques (ICCT)*, 1-7. doi:10.1109/ICCT56969.2023.10075789

`g2g_extract_trips` *Extract Trips from GPS Trajectories*

Description

Reads raw GPS and terminal CSV files, extracts trips, and optionally writes results to a CSV.

Usage

```
g2g_extract_trips(
  gps_data,
  terminals_data,
  terminals_buffer_radius,
  output_path = NULL,
  projected_crs = NULL,
  backend = "auto",
  projected = NULL
)
```

Arguments

`gps_data`	A data.frame or path to the raw GPS CSV.
`terminals_data`	A data.frame or path to the terminal coordinates CSV.
`terminals_buffer_radius`	Numeric. Buffer radius for terminals (in meters).
`output_path`	Character. Optional path to write output trip features as CSV. Default is `NULL` (no file written).
`projected_crs`	Numeric. The EPSG code of a projected coordinate system used for metric distance calculations. Required when `projected = TRUE`; otherwise a suitable UTM projection is selected when omitted.
`backend`	Character. The backend to use: `"auto"`, `"rust"`, `"rcpp"`, or `"pure_r"`. Automatic selection prefers Rust, then Rcpp, then pure R. Explicit unavailable backends produce an error.
`projected`	Logical. Whether plain-table coordinates are already projected. Out-of-bounds coordinates require explicit `TRUE`.

Value

A data.table containing extracted trip features.

Examples

```
data(g2g_data_gps)
data(g2g_data_terminals)
trips <- g2g_extract_trips(
  gps_data = g2g_data_gps,
  terminals_data = g2g_data_terminals,
  terminals_buffer_radius = 50
)
```

---

g2g_extract_trips_and_stop_times

Extract Trips and Stop Times from GPS Trajectories

Description

Reads raw GPS, terminal, and stop CSV files, extracts trips and stop times, and optionally writes results.

Usage

```
g2g_extract_trips_and_stop_times(
  gps_data,
  terminals_data,
  stops_data,
  terminals_buffer_radius,
  stops_buffer_radius,
  stops_extended_buffer_radius,
  output_trips_path = NULL,
  output_stops_path = NULL,
  projected_crs = NULL,
  backend = "auto",
  projected = NULL,
  stop_direction_map = NULL
)
```

Arguments

`gps_data`	A data.frame or path to the raw GPS CSV.
`terminals_data`	A data.frame or path to the terminal coordinates CSV.
`stops_data`	A data.frame or path to the bus stops coordinates CSV.
`terminals_buffer_radius`	Numeric. Buffer radius for terminals (in meters).
`stops_buffer_radius`	Numeric. Buffer radius for bus stops (in meters).

`stops_extended_buffer_radius`
: Numeric. Extended buffer radius for bus stops (in meters).

`output_trips_path`
: Character. Optional path to write output trip features as CSV. Default is `NULL` (no file written).

`output_stops_path`
: Character. Optional path to write output stop times as CSV. Default is `NULL` (no file written).

`projected_crs`
: Numeric. The EPSG code of a projected coordinate system used for metric distance calculations. Required when `projected = TRUE`; otherwise a suitable UTM projection is selected when omitted.

`backend`
: Character. The backend to use: `"auto"`, `"rust"`, `"rcpp"`, or `"pure_r"`. Automatic selection prefers Rust, then Rcpp, then pure R. Explicit unavailable backends produce an error.

`projected`
: Logical. Whether plain-table coordinates are already projected. Out-of-bounds coordinates require explicit `TRUE`.

`stop_direction_map`
: Optional named character vector mapping each raw stop-direction label to its starting terminal ID.

Value

A list containing two data.tables: `trips` and `stop_times` (with columns matching GTFS standard naming).

Examples

```
data(g2g_data_gps)
data(g2g_data_terminals)
data(g2g_data_stops)
result <- g2g_extract_trips_and_stop_times(
  gps_data = g2g_data_gps,
  terminals_data = g2g_data_terminals,
  stops_data = g2g_data_stops,
  terminals_buffer_radius = 50,
  stops_buffer_radius = 30,
  stops_extended_buffer_radius = 50
)
head(result$trips)
head(result$stop_times)
```

Index