Package: metrictrails (via r-universe)

October 23, 2026

Title Processing of GNSS Streams from Sports Watches

Version 1.0.0

Description Tools to read, clean and standardise GNSS (Global Navigation Satellite System) streams recorded by sports watches and similar devices, with a focus on trail and mountain running. The pipeline reads GPX files, applies a naming convention, sanitises coordinates (Kalman, LOESS, Gaussian kernel or Savitzky-Golay smoothing, and red flags for implausible readings), resamples the trajectory to a uniform spatial resolution derived from the Nyquist criterion, and replaces elevation with values from a Digital Elevation Model. Derived measures include speed, slope, elevation gain and kilometre-effort, Tobler hiking function parameters fitted by quantile regression, comparison between streams, detection of passes along a route and map matching to a known reference trail by dynamic programming. Methods follow Sanchez and Villena (2020) <doi:10.1177/1754337120918975>, Sanchez et al. (2024) <doi:10.1109/ICECET61485.2024.10698606>, Sanchez et al. (2025) <doi:10.1177/17543371251341660> and Martinez, Sanchez and Diaz (2026) <doi:10.1109/ICECET65726.2026.11632531>.

URL https://github.com/raimun2/metrictrails

BugReports https://github.com/raimun2/metrictrails/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

Depends R (>= 4.1)

Imports magrittr, sf, stats, terra, tibble, utils

Suggests curl, knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://raimun2.r-universe.dev

Date/Publication 2026-10-23 00:57:55 UTC

RemoteUrl https://github.com/raimun2/metrictrails

RemoteRef HEAD

RemoteSha 240bbf5d9dbe19e98892cc8df438de728d9ed22b

Contents

compare_stream . . . 2
detect_stream . . . 3
differential_stream . . . 5
ele_correction . . . 6
fractal_stream . . . 7
get_dem . . . 8
locate_stream . . . 9
match_stream . . . 10
project_stream . . . 12
read_stream . . . 13
resample_dem . . . 14
resample_stream . . . 15
resolution_stream . . . 16
sanitize_stream . . . 17
strava_metadata . . . 20
strava_streams . . . 21
summarise_stream . . . 22
tobler_speed . . . 23
tobler_stream . . . 24
uniform_stream . . . 25
write_stream . . . 26

Index **28**

`compare_stream` *Compare two streams*

Description

Spatial agreement between two streams, based on the distance from every reading of one stream to the nearest reading of the other (in the UTM zone of `a`). Directed measures are reported in both directions (`_ab`: from `a` to `b`; `_ba`: from `b` to `a`):

- `hausdorff`: maximum nearest-neighbour distance (m); the symmetric value is the larger of both directions.

- `separation`: median nearest-neighbour distance (m).
- `metric_score`: share of readings whose nearest-neighbour distance is at most `threshold` (%).

Separation follows the implementation used in Sanchez, Nieto-Jimenez & Riveros-Matthey (2025): the median of the directed nearest-neighbour distances. Streams should be resampled to the same resolution before comparing them (see `resample_stream()`).

Usage

```
compare_stream(a, b, threshold = 1, dims = 2)
```

Arguments

a, b	Streams with `lon` and `lat` (and `ele` when `dims = 3`).
threshold	Distance (m) used by the metric score.
dims	2 for horizontal distance, 3 to include elevation.

Value

A one-row tibble.

References

Sanchez, R., Nieto-Jimenez, C., & Riveros-Matthey, C. (2025). High-precision mapping of mountain terrain for human locomotion research: A multimodal geospatial framework. *2025 15th IEEE International Conference on Pattern Recognition Systems (ICPRS)*. doi:10.1109/ICPRS66293.2025.11302827

Examples

```
act = read_stream(system.file("extdata", "gpx_activity.gpx", package = "metrictrails"))
route = read_stream(system.file("extdata", "gpx_route.gpx", package = "metrictrails"))
compare_stream(resample_stream(act, res = 2), resample_stream(route, res = 2))
```

`detect_stream` *Detect the passes of a stream along a route*

Description

Predicate and locator: does a stream follow a route, and where? The stream is not transformed (compare with `match_stream()`, which aligns a stream known to follow a route).

The procedure follows, in metres:

1. the stream must come within `tolerance` of the route;
2. at least `min_completion` of the route readings must lie within `tolerance` of the stream;

3. every time the stream comes within `tolerance` of the start (or of the end) of the route a candidate pass begins; it ends at the first arrival within `tolerance` of the opposite end of the route after covering half of the route length (or, if there is none, after the route length plus `tolerance`);
4. the coverage of a pass is the share of route readings within `tolerance` of the pass; passes with coverage of at least `min_coverage` are `"match"`, the others `"partial"`;
5. the direction of a pass is `"forward"` when the order of the readings agrees with the order of their nearest route readings (positive Spearman correlation), otherwise `"reverse"`.

Distances are measured between readings, so both streams should have a resolution finer than `tolerance` (see `resample_stream()`).

Usage

```
detect_stream(
  data,
  route,
  tolerance = 50,
  min_completion = 1,
  min_coverage = 0.97,
  min_partial = 0.25
)
```

Arguments

`data`	A stream with `lon` and `lat`.
`route`	The route, with `lon` and `lat`, ordered from start to end.
`tolerance`	Distance (m) within which the stream is on the route.
`min_completion`	Share of the route that must be covered by the whole stream.
`min_coverage`	Share of the route that a pass must cover to be a `"match"`.
`min_partial`	Passes covering less than this share of the route are not reported. Overlapping passes are reduced to the one with the highest coverage.

Value

A tibble with one row per pass: `pass`, `start_row`, `end_row` (rows of `data`), `direction`, `coverage` and `status`. It has no rows when the stream does not follow the route; the reason is given in `attr(x, "diagnosis")`.

See Also

`match_stream()`, `compare_stream()`

Examples

```
act = read_stream(system.file("extdata", "gpx_activity.gpx", package = "metrictrails"))
route = read_stream(system.file("extdata", "gpx_route.gpx", package = "metrictrails"))
detect_stream(act, route)
```

differential_stream *Spatial and temporal derivatives of a stream*

Description

Adds, for every reading, the differences with the previous reading of the same segment and the quantities derived from them.

Always computed:

- `delta_d`: horizontal (2D) distance (m), and `distance`, its cumulative sum along the stream (distance between segments is not counted).
- `turn_angle`: absolute change of heading between the previous and the current step (degrees, 0 to 180).

With `ele`: `delta_ele`, `dplus` and `dminus` (elevation gained and lost, both positive), `delta_d3d` (3D distance) and `slope` (`delta_ele / delta_d`, rise over run; `NA` when the reading is idle).

With `time`: `delta_time`, `speed` (horizontal, m/s), `accel` (change of `speed` over time, m/s^2), `pace` (min/km); with `ele` also `vert_speed`, `vert_accel` and `speed_3d`.

The first reading of every segment gets 0 for differences and `NA` for rates.

Usage

```
differential_stream(data, distance = c("utm", "haversine"), idle = 0.1)
```

Arguments

`data`	A stream with `lon` and `lat`.
`distance`	How horizontal distance is computed: `"utm"` (Euclidean distance in the UTM zone of the stream) or `"haversine"` (great circle on a sphere of radius 6371008.8 m).
`idle`	Horizontal distance (m) below which slope and heading are not defined.

Value

The stream with the new columns.

References

Sanchez, R., Nieto-Jimenez, C., & Riveros-Matthey, C. (2025). High-precision mapping of mountain terrain for human locomotion research: A multimodal geospatial framework. *2025 15th IEEE International Conference on Pattern Recognition Systems (ICPRS)*. doi:10.1109/ICPRS66293.2025.11302827

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
stream = read_stream(f) %>% uniform_stream() %>% differential_stream()
summary(stream$speed)
```

`ele_correction` *Replace the elevation of a stream with values from a DEM*

Description

Extracts the elevation of every reading from a Digital Elevation Model and replaces the raw elevation (or adds it as `ele_dem`). Elevation from an open DEM gives consistent elevation gain across devices, independently of the altitude sensor (Sanchez & Villena, 2020). This is the last step of the pipeline, after `resample_stream()`.

Usage

```
ele_correction(
  data,
  dem = NULL,
  z = 14,
  extract = c("bilinear", "simple"),
  replace = TRUE
)
```

Arguments

`data`	A stream with `lon` and `lat`.
`dem`	A `terra::SpatRaster` or a path to a raster file. When `NULL`, a DEM is downloaded with `get_dem()` at zoom `z`.
`z`	Zoom level used when `dem` is `NULL` (see `get_dem()`).
`extract`	`"bilinear"` (interpolate between the four nearest cells) or `"simple"` (value of the cell).
`replace`	If `TRUE`, `ele` is replaced; otherwise the column `ele_dem` is added.

Value

The stream with the new elevation.

References

Sanchez, R., & Villena, M. (2020). Comparative evaluation of wearable devices for measuring elevation gain in mountain physical activities. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 234(4), 312-319. doi:10.1177/1754337120918975

Examples

```
f = system.file("extdata", "gpx_route.gpx", package = "metrictrails")
dem = system.file("extdata", "test_dem.tif", package = "metrictrails")
stream = ele_correction(read_stream(f), dem = dem, replace = FALSE)
summary(stream$ele - stream$ele_dem)
```

`fractal_stream` *Fractal dimension of a stream*

Description

Estimates the fractal dimension of the horizontal path of a stream with the divider (Richardson) method. A divider of size `r` is walked along the path: from the current point, the next point is the first point of the path at a straight-line distance `r`. The measured length is $L(r) = N(r) \cdot r +$ remainder, and for a fractal curve $L(r) \propto r^{1-D}$, so the dimension is $D = 1 - b$, where `b` is the slope of the least-squares regression of $\log L(r)$ on $\log r$.

A straight line has `D = 1`; values above 1 measure how much length is gained when the path is measured with finer steps, i.e. the geometric complexity that makes distance and elevation gain depend on the spatial resolution (Sanchez et al., 2025). Only horizontal 2D distance (UTM) is used. Each segment is walked separately and their lengths are added.

By default the divider sizes are `n_scales` values evenly spaced in log scale between twice the median distance between moving readings (finer dividers would only measure the straight segments between readings) and one tenth of the path length.

Usage

```
fractal_stream(data, scales = NULL, n_scales = 10)
```

Arguments

`data`	A stream with `lon` and `lat`.
`scales`	Optional vector of divider sizes in metres.
`n_scales`	Number of divider sizes when `scales` is `NULL`.

Value

A list with `dimension` (the estimate), `r_squared` of the log-log regression, and `scaling`, a tibble with the divider size `r`, the number of steps `n` and the length `length` (m).

References

Mandelbrot, B. (1967). How long is the coast of Britain? Statistical self-similarity and fractional dimension. *Science*, 156(3775), 636-638. doi:10.1126/science.156.3775.636

Sanchez, R., Egli, P. E., Jornet, K., Duggan, M., & Besomi, M. (2025). How fractal complexity distorts distance and elevation gain in trail and mountain running: The case for course measurement standardization. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*. doi:10.1177/17543371251341660

See Also

`summarise_stream()`, `resample_stream()`

Examples

```
f = system.file("extdata", "gpx_route.gpx", package = "metrictrails")
fd = fractal_stream(read_stream(f))
fd$dimension
fd$scaling
```

`get_dem` *Download a Digital Elevation Model for a stream*

Description

Downloads the Terrain Tiles (GeoTIFF) hosted on Amazon Web Services that intersect a corridor of `buffer` metres around the stream, mosaics them and projects the result to the UTM zone of the stream. Only the tiles along the path are downloaded, not every tile of its bounding box; cells of the bounding box outside those tiles are `NA`. The tiles combine SRTM and other sources; their resolution depends on the zoom level `z`. The default, `z = 14`, is the finest level (about 4 m at the latitude of Santiago, Chile, and 4.8 m at the equator) and is used for every stream regardless of its extent, so that elevations are always comparable; large areas simply need more tiles.

When `res` is given, the DEM is refined (or coarsened) with `resample_dem()`. Refining a public DEM with bilinear interpolation to sub-metre resolution reduces elevation gain errors to levels comparable to a LiDAR model (Sanchez et al., 2024).

Usage

```
get_dem(
  data,
  z = 14,
  res = NULL,
  method = "bilinear",
  buffer = 500,
  max_tiles = Inf
)
```

Arguments

`data`	A stream with `lon` and `lat`.
`z`	Zoom level of the tiles, from 1 to 14 (default 14).
`res`	Optional target resolution in metres, passed to `resample_dem()`.
`method`	Resampling method passed to `resample_dem()`.
`buffer`	Width (m) of the corridor around the stream.
`max_tiles`	Optional maximum number of tiles to download (`Inf` by default). A message reports the number of tiles when it exceeds 50.

Value

A `terra::SpatRaster` in the UTM zone of the stream.

References

Sanchez, R., Egli, P., Besomi, M., & Truffello, R. (2024). Assessing the impact of Digital Elevation Model resolution on elevation gain estimations in trail running. *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET)*. doi:10.1109/ICECET61485.2024.10698606

Terrain Tiles on AWS: https://registry.opendata.aws/terrain-tiles/

Examples

```
f = system.file("extdata", "gpx_route.gpx", package = "metrictrails")
dem = get_dem(read_stream(f))
dem
```

`locate_stream` *Locate a stream along a reference*

Description

Linear referencing: every reading of `data` is projected orthogonally onto the polyline of `reference`, giving its position along the reference and its signed distance to it. Unlike `match_stream()`, each reading is located independently, without monotonicity or speed constraints, so noisy readings can jump back and forth along the reference.

Each reading is projected on its nearest segment of the reference. Distances are measured in the UTM zone of the reference.

Usage

```
locate_stream(data, reference)
```

Arguments

`data`	A stream with `lon` and `lat`.
`reference`	The reference, with `lon` and `lat`, ordered along its path.

Value

The stream with:

- `s_ref`: distance along the reference of the projected point (m);
- `offset_cross`: signed distance to the reference (m, positive to the left of the reference direction);
- `lon_ref`, `lat_ref`: coordinates of the projected point.

See Also

`match_stream()`, `detect_stream()`

Examples

```
act = read_stream(system.file("extdata", "gpx_activity.gpx", package = "metrictrails"))
route = read_stream(system.file("extdata", "gpx_route.gpx", package = "metrictrails"))
located = locate_stream(act, route)
summary(abs(located$offset_cross))
```

`match_stream` *Match a stream to a known reference trail*

Description

Temporal alignment of a stream along a reference trail that the athlete is known to have followed (for example, rounds on a trail measured with a high-precision protocol). For every reading the function estimates where the athlete was along the reference, under the constraints that the athlete never moves backwards and never exceeds a physiologically plausible speed for the local slope.

The unknown is the sequence of reference vertices $S = (j_1, \ldots, j_n)$, non-decreasing along the direction of travel, that minimises

$$J(S) = \sum_i d^2(obs_i, ref_{j_i}) + \lambda^{-2} \sum_i pen(j_{i-1}, j_i, \Delta t_i)$$

where d is the horizontal (or 3D) distance and pen the squared excess of the horizontal, vertical and total speeds along the reference over the Tobler ceiling (`tobler_speed() * tol_factor`). Transitions whose total speed exceeds the ceiling (with slope clipped to `slope_clip`) are not allowed; staying on the same vertex is always allowed, so a solution always exists. Because J is a sum of terms that depend on one reading or on two consecutive readings, the global minimum is found exactly with dynamic programming (Viterbi algorithm).

The reference should come out of the full pipeline (sanitized, resampled at a fine resolution, with DEM elevation), because the matched stream takes its coordinates and elevation from it.

Usage

```
match_stream(
  data,
  reference,
  params,
  direction = c("auto", "forward", "reverse"),
  by = NULL,
  dims = 2,
  lambda = 1,
  tol_factor = 1.1,
  slope_clip = 0.6,
  max_cells = 5e+07
)
```

Arguments

`data`	A stream with `lon`, `lat` and `time` (and `ele` when `dims = 3`).
`reference`	The reference trail, with `lon`, `lat` and preferably `ele`, ordered along the trail.
`params`	Tobler parameters, output of `tobler_stream()` or a vector accepted by `tobler_speed()`.
`direction`	Direction of travel along the reference: `"auto"` (from the order of the projected readings), `"forward"` or `"reverse"`.
`by`	Optional name of a column splitting `data` into independent rounds; each round is matched separately (and its direction detected separately).
`dims`	`2` (horizontal distance) or `3` (include elevation) in the distance term.
`lambda`	Weight of the speed penalty (larger values weaken it).
`tol_factor`	Tolerance factor applied to the Tobler ceiling.
`slope_clip`	Slope clipping used by the hard speed constraint.
`max_cells`	Maximum size of the dynamic programming table (readings x vertices) for one round.

Value

The stream with:

- `lon`, `lat`, `ele`: position of the matched reference vertex (the original values are kept in `lon_obs`, `lat_obs`, `ele_obs`);
- `ref_index`, `s_ref`: matched vertex and its distance along the reference;
- `offset_d`: horizontal distance between the original reading and the matched vertex (m);
- `offset_along`, `offset_cross`: the same offset decomposed along the local direction of travel of the reference (positive when the reading is ahead) and across it (positive to the left), so that `offset_d^2 = offset_along^2 + offset_cross^2`;
- `speed_h`, `speed_v`, `speed_total`, `slope_ref`: kinematics along the reference after matching.

A quality-control report is stored in `attr(x, "match_report")`: readings per reference vertex (maximum and number of vertices above `max(2, ceiling(n / m))`), share of transitions at the speed ceiling, objective value and offset summaries, per round.

References

Sanchez, R. (2025). Human trajectory mapping in ecological contexts: a framework for reliable space-time localization. *ANZSCPB Conference 2025*, Sippy Downs, Australia.

Forney, G. D. (1973). The Viterbi algorithm. *Proceedings of the IEEE*, 61(3), 268-278. doi:10.1109/PROC.1973.9030

See Also

`tobler_stream()`, `detect_stream()`

Examples

```
act = read_stream(system.file("extdata", "gpx_activity.gpx", package = "metrictrails"))
route = read_stream(system.file("extdata", "gpx_route.gpx", package = "metrictrails"))
dem = system.file("extdata", "test_dem.tif", package = "metrictrails")
ref = route %>% resample_stream(res = 5) %>% ele_correction(dem = dem)
act = act[1:300, ] %>% uniform_stream() %>% sanitize_stream() %>%
  ele_correction(dem = dem)
params = c(v1 = 3.5, m1 = 0.1, c_plus = 3.2, c_minus = -0.5)
matched = match_stream(act, ref, params)
summary(matched$offset_d)
```

---

project_stream *Project a stream to metric coordinates*

Description

Adds the coordinates of every reading in a metric CRS, by default the WGS 84 / UTM zone that contains the centre of the stream. This is the projection used internally by every function of the package that measures distances.

Usage

```
project_stream(data, epsg = NULL)
```

Arguments

data	A stream with lon and lat.
epsg	Optional EPSG code of the target CRS. When NULL, the UTM zone of the stream is used (326xx north, 327xx south).

Value

The stream with columns x and y (metres); the EPSG code is stored in attr(x, "epsg").

See Also

locate_stream()

Examples

```
f = system.file("extdata", "gpx_route.gpx", package = "metrictrails")
route = project_stream(read_stream(f))
attr(route, "epsg")
head(route[, c("lon", "lat", "x", "y")])
```

read_stream *Read a GNSS stream from a GPX file*

Description

Reads the track points (or route points) of a GPX file with `sf::st_read()` and returns them as a tibble with one row per reading.

Coordinates are returned as `lon` and `lat` (WGS 84 degrees), elevation as `ele` and the time signature as `timestamp` (POSIXct, UTC). Each GPX track segment (`trkseg`) becomes a value of the column `segment`, so that later processing never interpolates across a recording gap. Point extensions such as the Garmin `TrackPointExtension` (heart rate, cadence, temperature, power) are returned as numeric columns with their original tag names (e.g. `hr`, `cad`, `atemp`).

Only GPX is supported. Other formats can be converted to GPX beforehand.

Usage

```
read_stream(file, type = c("auto", "track", "route"))
```

Arguments

`file`	Path to a `.gpx` file.
`type`	Which GPX geometry to read: `"auto"` (track points if present, otherwise route points), `"track"` or `"route"`.

Value

A tibble with columns `lon`, `lat`, and, when available, `ele`, `timestamp`, `segment` and extension variables.

See Also

`uniform_stream()`, `write_stream()`

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
stream = read_stream(f)
head(stream)
```

`resample_dem` *Change the resolution of a DEM*

Description

Refines or coarsens a Digital Elevation Model.

- `"bilinear"` and `"near"` resample the DEM to a grid of `res` metres. Bilinear interpolation is recommended to refine a coarse public DEM; nearest neighbour performs poorly at fine resolutions (Sanchez et al., 2024).
- `"mean"`, `"median"`, `"min"` and `"max"` aggregate blocks of cells to a coarser resolution (factor `round(res / current resolution)`).

Usage

```
resample_dem(
  dem,
  res,
  method = c("bilinear", "near", "mean", "median", "min", "max")
)
```

Arguments

`dem`	A `terra::SpatRaster` (or a path to a raster file) in a metric CRS, such as the output of `get_dem()`.
`res`	Target resolution in metres.
`method`	One of `"bilinear"`, `"near"`, `"mean"`, `"median"`, `"min"`, `"max"`.

Value

A `terra::SpatRaster`.

References

Sanchez, R., Egli, P., Besomi, M., & Truffello, R. (2024). Assessing the impact of Digital Elevation Model resolution on elevation gain estimations in trail running. *2024 International Conference on Electrical, Computer and Energy Technologies (ICECET)*. doi:10.1109/ICECET61485.2024.10698606

Examples

```
dem = terra::rast(system.file("extdata", "test_dem.tif", package = "metrictrails"))
dem_utm = terra::project(dem, "EPSG:32719")
resample_dem(dem_utm, res = 5)
```

`resample_stream` *Resample a stream to a uniform spatial resolution*

Description

Resamples each segment of a stream along its horizontal 2D path, so that the result does not depend on how fast the athlete moved or on the recording rate of the device (the "defractalizer").

By default the resolution is derived from the stream itself with `resolution_stream()` (`res = "nyquist"`): the q-quantile of the distance between consecutive moving readings. Streams resampled at different resolutions are not directly comparable; to compare several streams, pass the same numeric `res` to all of them (at least the largest of their Nyquist resolutions).

Methods:

- `"arclength"` (default): points are placed every `L / floor(L / res)` metres along the 2D path of length `L` (a spacing equal to or slightly larger than `res`, keeping both ends), as in `defractalize()` of Sanchez, Nieto-Jimenez & Riveros-Matthey (2025).
- `"simplify"`: the path is densified to 0.2 m and then simplified with the Douglas-Peucker algorithm using `res` as tolerance, as in Sanchez et al. (2025). The spacing of the result is not uniform.

Resampling decimates the stream wherever the original readings are closer than `res`; the stream should be low-pass filtered before (see `sanitize_stream()`) to avoid aliasing. `time`, `timestamp`, `ele` and every other numeric column are interpolated linearly along the 2D distance; other columns take the value of the preceding original reading. Elevation is expected to be replaced from a DEM afterwards (`ele_correction()`).

Usage

```
resample_stream(
  data,
  res = "nyquist",
  q = 0.9,
  method = c("arclength", "simplify"),
  idle = 0.1
)
```

Arguments

`data`	A stream with `lon` and `lat`.
`res`	Target resolution in metres, or `"nyquist"`.
`q`	Quantile used when `res = "nyquist"`.
`method`	`"arclength"` or `"simplify"`.
`idle`	Distance (m) below which a reading is idle (see `resolution_stream()`).

Value

The resampled stream; the resolution used is stored in `attr(x, "res")`.

References

Sanchez, R., Egli, P. E., Jornet, K., Duggan, M., & Besomi, M. (2025). How fractal complexity distorts distance and elevation gain in trail and mountain running: The case for course measurement standardization. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*. doi:10.1177/17543371251341660

Sanchez, R., Nieto-Jimenez, C., & Riveros-Matthey, C. (2025). High-precision mapping of mountain terrain for human locomotion research: A multimodal geospatial framework. *2025 15th IEEE International Conference on Pattern Recognition Systems (ICPRS)*. doi:10.1109/ICPRS66293.2025.11302827

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
stream = read_stream(f) %>% uniform_stream() %>% sanitize_stream()
resampled = resample_stream(stream)
attr(resampled, "res")
```

---

resolution_stream *Spatial resolution of a stream*

Description

Describes the distribution of the horizontal distance between consecutive readings (`delta_d`, 2D, in metres) and derives the Nyquist resolution used by `resample_stream()`.

A GNSS device that records at a fixed rate (e.g. 1 Hz) produces a spatial resolution that is a random variable: `delta_d` is the speed times the sampling interval. Readings with `delta_d < idle` (stops) are excluded.

The spacing `res` is a high quantile (`q`) of `delta_d`. Spatial features with wavelength of at least `2 * res` are resolved almost everywhere along the track, and a uniform grid with spacing `res` represents exactly that band (its Nyquist wavelength is `2 * res`). A finer grid would add interpolated readings between sparse fixes; a coarser grid would discard resolved information.

Only horizontal geodesic distance is used: raw elevations come from sensor fusion or are rounded to the metre, and are replaced from a DEM at the end of the pipeline.

Usage

```
resolution_stream(data, q = 0.9, idle = 0.1)
```

Arguments

`data`	A stream with `lon` and `lat`, or a list of streams.
`q`	Quantile of `delta_d` that defines the resolution.
`idle`	Distance (m) below which a reading is considered idle.

Value

A tibble with one row per stream: number of readings, share of idle readings, mean, standard deviation and quantiles of `delta_d`, and `res`, the q-quantile. When `data` is a list, use `max(result$res)` as the common resolution to make the streams comparable.

References

Sanchez, R., Egli, P. E., Jornet, K., Duggan, M., & Besomi, M. (2025). How fractal complexity distorts distance and elevation gain in trail and mountain running: The case for course measurement standardization. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*. doi:10.1177/17543371251341660

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
resolution_stream(read_stream(f))
```

---

sanitize_stream — *Sanitize the coordinates of a stream and flag implausible readings*

Description

Cleans the horizontal position (`lon`, `lat`) of a GNSS stream and adds red flags for readings that are physically implausible. This is the third step of the pipeline, after `read_stream()` and `uniform_stream()` and before `resample_stream()`.

Red flags are computed on the raw values, before smoothing, and returned as logical columns:

flag	rule
`flag_coord`	missing coordinate, |lat| > 90, |lon| > 180 or (0, 0)
`flag_time`	time does not increase (`delta_t <= 0`)
`flag_speed`	horizontal speed from the previous reading > `v_max`, or above the Tobler ceiling when `tobler` is given
`flag_accel`	|acceleration| > `accel_max`
`flag_jump`	displacement > `edge_jump` among the first or last 3 readings
`flag_slope`	|slope| > `slope_max`
`flag_vert_speed`	|vertical speed| > `vert_speed_max`
`flag_idle_ele`	elevation changes while the horizontal position does not
`flag_gap`	`delta_t` > `gap`
`flag_low_res`	distance from the previous reading > `low_res`

Rules that need time are skipped when the stream has no `time` column, or when more than half of the readings exceed `v_max` (time is then not physical, e.g. routes exported with fake timestamps; a warning is given). Elevation flags are diagnostics of the device: elevation is replaced from a DEM at the end of the pipeline (see `ele_correction()`).

`action` decides what happens to flagged readings before smoothing: `"flag"` only adds the columns; `"drop"` removes readings with a position flag and sets `ele` to `NA` where an elevation flag is raised;

"interpolate" replaces flagged positions (and flagged elevations) by linear interpolation between the nearest valid readings of the same segment. Readings with a flag_coord cannot be smoothed and are always removed.

Smoothing methods (per segment, in metric UTM coordinates):

- "kalman" (default): constant-velocity Kalman filter followed by a Rauch-Tung-Striebel smoother (rts = TRUE). Process noise is a white acceleration with standard deviation sigma_acc (m/s^2); measurement noise has standard deviation sigma_obs (m). The filter restarts after time gaps longer than gap and where time is missing or decreases. Without time, every method except "gaussian" runs along the cumulative 2D distance instead of time (arc-length parametrisation; sigma_acc then bounds the curvature, in 1/m).
- "loess": local polynomial regression of each coordinate against time (or cumulative distance when there is no time), with span alpha.
- "gaussian": Gaussian kernel regression against time with standard deviation bandwidth seconds (Sanchez & Villena, 2020).
- "savgol": Savitzky-Golay filter with window sg_window and polynomial order sg_order.
- "pipeline": the post-processing sequence of Martinez, Sanchez & Diaz (2026): Kalman filter, outlier detection on derived speed, acceleration and slope, linear interpolation of outliers, Savitzky-Golay filter and edge-jump correction. See Details.
- "none": flags only.

Usage

```
sanitize_stream(
  data,
  method = c("kalman", "loess", "gaussian", "savgol", "pipeline", "none"),
  action = c("flag", "drop", "interpolate"),
  smooth_ele = FALSE,
  v_max = 10,
  accel_max = 6,
  vert_speed_max = 1,
  slope_max = 1.19,
  gap = 30,
  low_res = 20,
  edge_jump = 30,
  idle = 0.1,
  tobler = NULL,
  tol_factor = 1.1,
  sigma_obs = 5,
  sigma_acc = 1,
  rts = TRUE,
  alpha = 0.05,
  bandwidth = 10,
  sg_window = 9,
  sg_order = 2,
  window = 25,
  jump_k = 1.5,
```

```
  local_k = 2
)
```

Arguments

data	A stream with lon and lat; time and ele are optional.
method	Smoothing method, see Description.
action	What to do with flagged readings: "flag", "drop" or "interpolate".
smooth_ele	If TRUE, ele is smoothed with the same method.
v_max	Maximum plausible horizontal speed (m/s).
accel_max	Maximum plausible absolute acceleration (m/s^2).
vert_speed_max	Maximum plausible absolute vertical speed (m/s).
slope_max	Maximum plausible absolute slope (rise over run); 1.19 is about 50 degrees.
gap	Time gap (s) reported by flag_gap.
low_res	Distance (m) between readings reported by flag_low_res.
edge_jump	Displacement (m) that defines a jump at the edges.
idle	Horizontal displacement (m) below which a reading is idle.
tobler	Optional Tobler parameters (output of tobler_stream()); when given, flag_speed uses the slope-dependent ceiling times tol_factor.
tol_factor	Tolerance factor applied to the Tobler ceiling.
sigma_obs, sigma_acc, rts	Kalman filter parameters.
alpha	Span of "loess".
bandwidth	Standard deviation (s) of the "gaussian" kernel.
sg_window, sg_order	Savitzky-Golay window (odd) and polynomial order.
window, jump_k, local_k	Outlier detection parameters of "pipeline".

Details

In "pipeline", a reading is an outlier when at least two of three criteria hold for any of the derived variables: (1) extreme value, outside [Q1 - 1.5 IQR, Q3 + 1.5 IQR]; (2) abrupt jump, absolute change from the previous reading larger than jump_k standard deviations of the changes; (3) local deviation, distance to the moving mean (window window) larger than local_k moving standard deviations. The paper reports the thresholds (window 25, jump 1.5, local deviation 2.0, Savitzky-Golay 9/2, edge jump 30 m) but not their exact formulation; the formulation above is the one implemented here. The uniform temporal interpolation of the paper is not applied, because spatial resampling (resample_stream()) follows in the metrictrails pipeline.

Value

The stream with smoothed lon, lat (and ele if smooth_ele), the flag_* columns, and a summary in attr(x, "sanitize_report").

References

Martinez, C., Sanchez, R., & Diaz, C. (2026). A post-processing methodology for improving kinematic consistency in GNSS-based running trajectories. *2026 6th International Conference on Electrical, Computer and Energy Technologies (ICECET)*. doi:10.1109/ICECET65726.2026.11632531

Sanchez, R., & Villena, M. (2020). Comparative evaluation of wearable devices for measuring elevation gain in mountain physical activities. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, 234(4), 312-319. doi:10.1177/1754337120918975

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
stream = read_stream(f) %>% uniform_stream() %>% sanitize_stream()
attr(stream, "sanitize_report")
```

---

strava_metadata *Metadata of the activities in strava_streams*

Description

Summary of the two activities in strava_streams as returned by the Strava API activities endpoint, one row per activity.

Usage

```
strava_metadata
```

Format

A data frame with 2 rows and 61 variables.

Details

Main variables: `id` (activity identifier, links to strava_streams), `name`, `type`, `start_date` and `start_date_local` (ISO 8601), `distance` (km), `elapsed_time` and `moving_time` (s), `total_elevation_gain`, `elev_high` and `elev_low` (m), `average_speed` and `max_speed`, and `map.summary_polyline` (encoded polyline). See the Strava API reference for the remaining variables.

Source

https://developers.strava.com/docs/reference/

`strava_streams` *GNSS streams of two activities from the Strava API*

Description

Readings of two activities recorded in Santiago, Chile (a trail run on Manquehue hill and a cross-country ride), as returned by the Strava API streams endpoint, already following the metrictrails naming convention.

Usage

```
strava_streams
```

Format

A data frame with 4148 rows and 13 variables:

lon longitude, WGS 84 degrees

lat latitude, WGS 84 degrees

ele elevation reported by Strava, metres

time seconds since the start of the activity

cadence cadence, steps or revolutions per minute

distance cumulative distance computed by Strava, kilometres

grade_smooth terrain grade smoothed by Strava, percent

heartrate heart rate, beats per minute

moving whether Strava considers the athlete moving

temp temperature, degrees Celsius

velocity_smooth speed smoothed by Strava, km/h

id activity identifier, links to strava_metadata

type activity type (`"Run"` or `"Ride"`)

Source

https://developers.strava.com/docs/reference/

summarise_stream *Summary measures of a stream*

Description

Computes course-level measures of a stream, or of each group of a stream.

- `distance_2d`, `distance_3d`: horizontal and 3D length (m).
- `elevation_gain`, `elevation_loss`: cumulative positive and negative elevation change (m, both positive).
- `ele_min`, `ele_max`: elevation range (m).
- `mean_slope`, `iqr_slope`: mean and inter-quartile range of the slope of moving readings (%).
- `sinuosity`: path length over straight-line distance between the first and last readings. `NA` for closed paths (first and last readings closer than `closed_tol`), where the ratio is not defined.
- `sinuosity_local`: mean ratio between the path length of a window of `window` metres and the straight-line distance (chord) between its ends. Windows start every `window / 10` metres along the path and both ends are interpolated, so the measure does not depend on the sampling rate of the device. It is defined for open and closed paths alike, equals 1 for a straight line and grows with the curvature at the scale of `window`. Windows do not cross segments; `NA` when no segment is longer than `window`.
- `mean_turn`: mean heading change (degrees).
- `resolution`: mean horizontal distance between consecutive readings (m).
- `km_effort`: kilometre-effort, horizontal km plus elevation gain / 100.
- `pct_idle`: share of readings without horizontal displacement (%), and `idle_gain`, `idle_loss`: elevation gained and lost while idle.
- `duration`: elapsed time (s), when `time` is available.
- `fractal_dim`: fractal dimension of the horizontal path (divider method, see `fractal_stream()`), when `fractal = TRUE`.

Usage

```
summarise_stream(
  data,
  by = NULL,
  idle = 0.1,
  fractal = FALSE,
  window = 100,
  closed_tol = 50
)
```

Arguments

`data`	A stream with `lon` and `lat` (and optionally `ele`, `time`).
`by`	Optional name(s) of grouping columns (e.g. `"round"`).
`idle`	Horizontal distance (m) below which a reading is idle.
`fractal`	If `TRUE`, adds the fractal dimension (`NA` when the path is too short for its resolution).
`window`	Length (m) of the windows of `sinuosity_local`. It should be several times the resolution of the stream.
`closed_tol`	Distance (m) between the first and last readings below which the path is closed and `sinuosity` is `NA`.

Value

A tibble with one row per group.

References

Sanchez, R., Egli, P. E., Jornet, K., Duggan, M., & Besomi, M. (2025). How fractal complexity distorts distance and elevation gain in trail and mountain running: The case for course measurement standardization. *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*. doi:10.1177/17543371251341660

Sanchez, R., Nieto-Jimenez, C., & Riveros-Matthey, C. (2025). High-precision mapping of mountain terrain for human locomotion research: A multimodal geospatial framework. *2025 15th IEEE International Conference on Pattern Recognition Systems (ICPRS)*. doi:10.1109/ICPRS66293.2025.11302827

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
summarise_stream(read_stream(f) %>% uniform_stream())
```

`tobler_speed` *Speed predicted by the Tobler function*

Description

Evaluates the asymmetric Tobler function fitted by `tobler_stream()`.

Usage

```
tobler_speed(slope, params)
```

Arguments

`slope`	Slope (rise over run).
`params`	Output of `tobler_stream()`, or a numeric vector with `v1`, `m1`, `c_plus`, `c_minus` (in this order or named).

Value

Horizontal speed (m/s).

Examples

```
tobler_speed(c(-0.2, 0, 0.2), c(v1 = 3, m1 = -0.05, c_plus = 3.5, c_minus = -3.5))
```

---

tobler_stream *Tobler parameters of a stream*

Description

Fits an asymmetric Tobler-type hiking function to the relation between horizontal speed and slope of a stream:

$$v(m) = v_1 \exp(-c_+(m - m_1)) \quad \text{if } m \geq m_1$$

$$v(m) = v_1 \exp(-c_-(m - m_1)) \quad \text{if } m < m_1$$

where `m` is the slope (rise over run), `m1` the slope of maximum speed, `v1` the speed at `m1` (m/s), `c_plus` > `0` the decay uphill and `c_minus` < `0` the decay downhill.

The model describes the frontier of human performance, so it is fitted to the `tau` quantile of speed, not to its mean. Two estimators are available, both using only base R:

- `"pinball"` (default): minimises the quantile (pinball) loss $\sum \rho_\tau(v_i - v(m_i))$ with `stats::optim()` (Nelder-Mead), starting from the `"binned"` solution and a few perturbations of it. This is the same estimator as nonlinear quantile regression (`quantreg::nlrq`).
- `"binned"`: groups the readings in slope bins of width `bin_width`, takes the `tau` quantile of speed in each bin with at least `min_bin` readings, and fits `log(speed)` with a weighted piecewise-linear `stats::lm()`; `m1` is chosen with `stats::optimize()`.

The stream should have gone through the pipeline (sanitized, resampled and with DEM elevation) so that slope and speed are reliable.

Usage

```
tobler_stream(
  data,
  tau = 0.9,
  method = c("pinball", "binned"),
  slope_range = c(-0.6, 0.6),
  min_speed = 0.1,
  bin_width = 0.02,
  min_bin = 10
)
```

Arguments

data	A stream with lon, lat, ele and time.
tau	Quantile of speed to model.
method	"pinball" or "binned".
slope_range	Slopes outside this range are ignored.
min_speed	Readings slower than this (m/s) are ignored (stops).
bin_width	Width of the slope bins of "binned".
min_bin	Minimum readings per bin.

Value

A one-row tibble with v1, m1, c_plus, c_minus, tau, n (readings used), r1 (pseudo R-squared for quantile regression, 1 - loss / loss of the intercept-only model) and method.

References

Tobler, W. (1993). Three presentations on geographical analysis and modeling. *National Center for Geographic Information and Analysis, Technical Report 93-1*.

Koenker, R., & Machado, J. A. F. (1999). Goodness of fit and related inference processes for quantile regression. *Journal of the American Statistical Association*, 94(448), 1296-1310.

See Also

tobler_speed()

Examples

```
f = system.file("extdata", "gpx_activity.gpx", package = "metrictrails")
dem = system.file("extdata", "test_dem.tif", package = "metrictrails")
stream = read_stream(f) %>% uniform_stream() %>% sanitize_stream() %>%
  resample_stream(res = 5) %>% ele_correction(dem = dem)
tobler_stream(stream)
```

---

uniform_stream *Uniform naming convention for a stream*

Description

Renames the columns of a stream to the convention used by every function of the package:

- lon: longitude (WGS 84 degrees)
- lat: latitude (WGS 84 degrees)
- ele: elevation or altitude (metres)
- timestamp: time signature of the reading (POSIXct, UTC)

- time: seconds since the first reading

Columns are matched against a dictionary of exact names (case-insensitive), e.g. lng, longitude or position_long for longitude and altitude or enhanced_altitude for elevation. Partial matches are not used, so columns such as moving_time or platform are never renamed. Other variables (heart rate, cadence, power, ...) are kept unchanged.

A column called time holding date-times (or ISO 8601 text) is treated as a timestamp. A numeric time column (e.g. Strava API streams) is kept as seconds. When time increases in less than half of the readings (e.g. routes exported by planners with a constant time), time and timestamp are removed with a warning.

Usage

```
uniform_stream(data)
```

Arguments

data	A data frame or tibble with a GNSS stream.

Value

A tibble with lon, lat (and ele, timestamp, time when available) as first columns.

Examples

```
data(strava_streams)
head(uniform_stream(strava_streams))

df = data.frame(Latitude = c(-33.40, -33.41), Longitude = c(-70.50, -70.51),
                Altitude = c(800, 805),
                Time = c("2024-01-01T10:00:00Z", "2024-01-01T10:00:05Z"))
uniform_stream(df)
```

---

write_stream *Write a GNSS stream to a GPX file*

Description

Writes a stream as a GPX 1.1 track. Each value of segment becomes a <trkseg>. Elevation (ele) and time (timestamp) are written when present.

Usage

```
write_stream(data, file, name = NULL)
```

Arguments

data	A stream with at least lon and lat (see uniform_stream()).
file	Path of the GPX file to create.
name	Optional track name.

Value

`file`, invisibly.

See Also

`read_stream()`

Examples

```
f = system.file("extdata", "gpx_route.gpx", package = "metrictrails")
route = read_stream(f)
out = tempfile(fileext = ".gpx")
write_stream(route, out)
nrow(read_stream(out)) == nrow(route)
```

Index

∗ **datasets**
    strava_metadata, 20
    strava_streams, 21

compare_stream, 2
compare_stream(), *4*

detect_stream, 3
detect_stream(), *9*, *11*
differential_stream, 5

ele_correction, 6
ele_correction(), *15*, *17*

fractal_stream, 7
fractal_stream(), *22*

get_dem, 8
get_dem(), *6*, *14*

locate_stream, 9
locate_stream(), *12*

match_stream, 10
match_stream(), *3*, *4*, *9*

project_stream, 12

read_stream, 13
read_stream(), *17*, *27*
resample_dem, 14
resample_dem(), *8*
resample_stream, 15
resample_stream(), *3*, *4*, *6*, *7*, *16*, *17*, *19*
resolution_stream, 16
resolution_stream(), *15*

sanitize_stream, 17
sanitize_stream(), *15*
sf::st_read(), *13*
stats::lm(), *24*
stats::optim(), *24*
stats::optimize(), *24*
strava_metadata, 20, *21*
strava_streams, *20*, 21
summarise_stream, 22
summarise_stream(), *7*

tobler_speed, 23
tobler_speed(), *11*, *25*
tobler_stream, 24
tobler_stream(), *11*, *19*, *23*

uniform_stream, 25
uniform_stream(), *13*, *17*, *26*

write_stream, 26
write_stream(), *13*