

# Package ‘geoprofiler’

May 8, 2026

**Title** Perpendicular Line Transects for Geosciences

**Version** 0.0.3

**Description** Toolset to create perpendicular profile graphs and swath profiles. Method are based on coordinate rotation algorithm by Schaeben et al. (2024) <[doi:10.1002/mma.9823](https://doi.org/10.1002/mma.9823)>.

**License** GPL (>= 3)

**URL** <https://tobiste.github.io/geoprofiler/>

**BugReports** <https://github.com/tobiste/geoprofiler/issues>

**Depends** R (>= 4.1.0)

**Imports** dplyr, sf, tectonicr, terra, units

**Suggests** ggplot2, knitr, rmarkdown, spelling, testthat (>= 3.0.0), tidyterra

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**LazyData** true

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Tobias Stephan [aut, cre] (ORCID:  
<<https://orcid.org/0000-0002-9290-014X>>)

**Maintainer** Tobias Stephan <tobias.stephan1@yahoo.com>

**Repository** CRAN

**Date/Publication** 2025-12-11 19:20:13 UTC

## Contents

draw . . . . .	2
line_ends . . . . .	3
locations_example . . . . .	3
point_distance . . . . .	4
profile_azimuth . . . . .	4
profile_coords . . . . .	5
profile_length . . . . .	6
profile_line . . . . .	7
profile_points . . . . .	7
raster_example . . . . .	8
swath_profile . . . . .	9
swath_stats . . . . .	10
<b>Index</b>	<b>12</b>

---

draw	<i>Draw a profile line or a point to retrieve coordinates</i>
------	---

---

### Description

Opens a plot window showing the the map with the data, where the user can click profile coordinates.

### Usage

```
get_coordinates(x, n = 1, type = "o", col = "#B63679FF", ...)
```

```
draw_profile(x, n = 10, ...)
```

### Arguments

x	sf object
n	the maximum number of points to locate. Valid values start at 1.
type	One of "n", "p", "l" or "o". If "p" or "o" the points are plotted; if "l" or "o" they are joined by lines.
col	color of line or point
...	additional graphics parameters used if type != "n" for plotting the locations.

### Value

sf object of the profile.

---

line_ends	<i>Extract End Points of a Line</i>
-----------	-------------------------------------

---

**Description**

Extract End Points of a Line

**Usage**

```
line_ends(x)
```

**Arguments**

x                   sf line object

**Value**

sf point object

**Examples**

```
p1 <- data.frame(lon = -90.8, lat = 48.6) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
profile_points(p1,
  profile.azimuth = 135, profile.length = 10000,
  crs = sf::st_crs("EPSG:26915")
) |>
  profile_line() |>
  line_ends()
```

---

locations_example	<i>Example sf data set</i>
-------------------	----------------------------

---

**Description**

example dataset

**Usage**

```
data('locations_example')
```

**Format**

An object of class sf

**Examples**

```
data("locations_example")
head(locations_example)
```

---

point_distance	<i>Distance Between Points</i>
----------------	--------------------------------

---

**Description**

This uses the **haversine** formula (by default) to calculate the great-circle distance between two points, i.e., the shortest distance over the earth's surface.

**Usage**

```
point_distance(a, b, ...)
```

**Arguments**

a	lon, lat coordinate of point 1
b	lon, lat coordinate of point 2
...	parameters passed to <code>tectonicr::dist_greatcircle()</code>

**Value**

units object giving the distance

**Examples**

```
berlin <- c(13.4, 52.517) # lon, lat  
tokyo <- c(139.767, 35.7) # lon, lat  
point_distance(berlin, tokyo)
```

---

profile_azimuth	<i>Azimuth Between Profile Points</i>
-----------------	---------------------------------------

---

**Description**

Azimuth Between Profile Points

**Usage**

```
profile_azimuth(x)
```

**Arguments**

x	sf point object. First point marks the start point.
---	---

**Details**

If only two points are given, the azimuth is calculated using triangulation from the tectonic package. If more than two points are given, the azimuth is calculated using linear interpolation in the coordinate reference frame given by profile.

**Value**

Azimuth as units object

**See Also**

[profile\\_length\(\)](#)

**Examples**

```
p1 <- data.frame(lon = -90.8, lat = 48.6) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")

profile_points(p1,
  profile.azimuth = 135, profile.length = 10000,
  crs = sf::st_crs("EPSG:26915")
) |>
  profile_azimuth()
```

---

profile\_coords

*Profile Coordinates*

---

**Description**

Project points on a cross section given by a starting point and the direction

**Usage**

```
profile_coords(x, profile, azimuth = NULL, drop.units = TRUE)
```

**Arguments**

x	'sf' object
profile	'sf' object of the profile or the profile's starting point.
azimuth	numeric. Direction (in degrees) emanating from starting point. Is ignored when profile contains two points or is a LINESTRING.
drop.units	logical. Whether the return should show the units or not.

**Value**

tibble where X is the distance along the profile line. Y is the distance across the profile line. (units of X and Y depend on coordinate reference system).

**Author(s)**

Tobias Stephan

**Examples**

```

data(locations_example)
p1 <- data.frame(lon = -90.8, lat = 48.6) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
profile_crds <- profile_coords(locations_example, profile = p1, azimuth = 135)
head(profile_crds)

# Plot the transformed coordinates
plot(profile_crds)

```

---

profile_length	<i>Length of Profile</i>
----------------	--------------------------

---

**Description**

Length of Profile

**Usage**

```
profile_length(x, ...)
```

**Arguments**

x	sf line object
...	(optional) passed on to <a href="#">s2::s2_distance()</a>

**Value**

units object when coordinate system is set.

**See Also**[profile\\_azimuth\(\)](#)**Examples**

```

p1 <- data.frame(lon = -90.8, lat = 48.6) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
profile_points(p1,
  profile.azimuth = 135, profile.length = 10000,
  crs = sf::st_crs("EPSG:26915")
) |>
  profile_line() |>
  profile_length()

```

---

profile_line	<i>Combine Points to a Line</i>
--------------	---------------------------------

---

**Description**

Combine Points to a Line

**Usage**

```
profile_line(x)
```

**Arguments**

x                    sf point object

**Value**

sf line object

**See Also**

[profile\\_points\(\)](#)

**Examples**

```
p1 <- data.frame(lon = -90.8, lat = 48.6) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
profile_points(p1,
  profile.azimuth = 135, profile.length = 10000,
  crs = sf::st_crs("EPSG:26915")
) |>
  profile_line()
```

---

profile_points	<i>Profile End Point</i>
----------------	--------------------------

---

**Description**

Create a end point along a profile line starting at a point with a defined direction and length.

**Usage**

```
profile_points(
  start,
  profile.azimuth,
  profile.length,
  crs = st_crs(start),
  return.sf = TRUE
)
```

**Arguments**

`start`                `sf` point object.  
`profile.azimuth`        numeric or units object. Direction of profile in degrees if numeric.  
`profile.length`        numeric or units object.  
`crs`                    Coordinate reference system. Should be parsed by `sf::st_crs()`.  
`return.sf`             logical. Should the profile points be returned as a `sf` object (TRUE, the default) object or as a `data.frame`.

**Value**

class depends on `return.sf`.

**Note**

Use metric values (meters, kilometers, etc) in case of a projected coordinate reference frame, and degree when geographical coordinate reference frame.

**Examples**

```

p1 <- data.frame(lon = -90.8, lat = 48.6) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
profile_points(p1,
  profile.azimuth = 135, profile.length = units::set_units(10, "km"),
  crs = sf::st_crs("EPSG:26915")
)

```

---

raster\_example

*Example raster data set*


---

**Description**

example dataset

**Usage**

```
data('raster_example')
```

**Format**

An object of class `matrix`

**Examples**

```

data("raster_example")
head(raster_example)

```

---

swath_profile	<i>Swath Elevation Profile Statistics</i>
---------------	---

---

### Description

Calculate swath-profile values perpendicular to a straight baseline. The distance between samples and the number of samples can be specified, see arguments `k` and `dist`. Values of the swath-profile are extracted from a given raster file, see argument `raster`. CRS of raster and points have to be the same.

### Usage

```
swath_profile(  
  profile,  
  raster,  
  k = 1,  
  dist,  
  crs = terra::crs(raster),  
  method = c("bilinear", "simple")  
)
```

### Arguments

profile	either a <code>sf</code> object or a <code>matrix(ncol=2, nrow=2)</code> with x and y coordinates of beginning and end point of the baseline; each point in one row <b>column 1</b> x coordinates (or longitudes) <b>column 2</b> y coordinates (latitudes)
raster	Raster file ("SpatRaster" object as loaded by <code>terra::rast()</code> )
k	integer. number of lines on each side of the baseline
dist	numeric. distance between lines
crs	character. coordinate reference system. Both the raster and the profile are transformed into this CRS. Uses the CRS of raster by default.
method	character. method for extraction of raw data, see <code>terra::extract()</code> : default value: "bilinear"

### Details

The final width of the swath is:  $2k \times \text{dist}$ .

### Value

list.

`swath` matrix. Statistics of the raster measured along the lines

`data` list of numeric vector containing the data extracted from the raster along each line

`lines` swath lines as "sf" objects

**Source**

The algorithm is a modified version of "swathR" by Vincent Haburaj (<https://github.com/jjvhab/swathR>).

**See Also**

[swath\\_stats\(\)](#)

**Examples**

```
# Create a random raster
r <- terra::rast(ncol = 10, nrow = 10, xmin = -150, xmax = -80, ymin = 20, ymax = 60, crs = "WGS84")
terra::values(r) <- runif(terra::ncell(r))

# Create a random profile
profile <- data.frame(lon = c(-140, -90), lat = c(55, 25)) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
swath_profile(profile, r, k = 2, dist = 1)
```

---

swath\_stats

*Summary Statistics on Swath Elevation Profile*

---

**Description**

Statistics of the elevation data across a swath profile.

**Usage**

```
swath_stats(x, profile.length = 1)
```

**Arguments**

**x** list. The return object of [swath\\_profile\(\)](#)

**profile.length** numeric or units object. If NULL the fractional distance is returned, i.e. 0 at start and 1 at the end of the profile.

**Value**

data.frame

**See Also**

[swath\\_profile\(\)](#)

**Examples**

```
# Create a random raster
r <- terra::rast(ncol = 10, nrow = 10, xmin = -150, xmax = -80, ymin = 20, ymax = 60)
terra::values(r) <- runif(terra::ncell(r))

# Create a random profile
profile <- data.frame(lon = c(-140, -90), lat = c(55, 25)) |>
  sf::st_as_sf(coords = c("lon", "lat"), crs = "WGS84")
swath <- swath_profile(profile, r, k = 5, dist = 10)

swath_stats(swath, profile.length = profile_length(profile_line(profile)))
```

# Index

## \* datasets

locations\_example, 3  
raster\_example, 8

draw, 2

draw\_profile(draw), 2

get\_coordinates(draw), 2

line\_ends, 3

locations\_example, 3

point\_distance, 4

profile\_azimuth, 4

profile\_azimuth(), 6

profile\_coords, 5

profile\_length, 6

profile\_length(), 5

profile\_line, 7

profile\_points, 7

profile\_points(), 7

raster\_example, 8

s2::s2\_distance(), 6

sf::st\_crs(), 8

swath\_profile, 9

swath\_profile(), 10

swath\_stats, 10

swath\_stats(), 10

tectonicr::dist\_greatcircle(), 4

terra::extract(), 9

terra::rast(), 9