Package 'rcrisp'

July 4, 2025

Title Automate the Delineation of Urban River Spaces

Version 0.1.4

Description Provides tools to automate the morphological delineation of riverside urban areas, based on a method introduced in Forgaci (2018) <doi:10.7480/abe.2018.31>. Delineation entails the identification of corridor boundaries, segmentation of the corridor, and delineation of the river space. The resulting delineation can be used to characterise spatial phenomena that can be related to the river as a central element.

License Apache License (>= 2)

URL https://cityriverspaces.github.io/rcrisp/, https://doi.org/10.5281/zenodo.15793526

BugReports https://github.com/CityRiverSpaces/rcrisp/issues

Depends R (>= 4.1.0)

Imports dbscan, dplyr, lwgeom, osmdata, rcoins, rlang, rstac, sf, sfheaders, sfnetworks, stringr, terra, tidygraph, units, visor

Suggests ggplot2, knitr, purrr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Claudiu Forgaci [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3218-5102>), Francesco Nattino [aut] (ORCID: <https://orcid.org/0000-0003-3286-0139>), Fakhereh Alidoost [ctb] (ORCID: <https://orcid.org/0000-0001-8407-6472>), Meiert Willem Grootes [ctb] (ORCID: <https://orcid.org/0000-0002-5733-4795>), Netherlands eScience Center [fnd]

Maintainer Claudiu Forgaci <c.forgaci@tudelft.nl>

Repository CRAN

Date/Publication 2025-07-04 21:40:09 UTC

Contents

as_bbox *Get the bounding box from the x object*

Description

If the x does not have a CRS, WGS84 is assumed.

Usage

```
as_bbox(x)
```

Arguments

x	Simple feature object (or compatible) or a bounding box, provided either as a matrix (with x, y as rows and min, max as columns) or as a vector (xmin, ymin, xmax, ymax)

Value

A bounding box as returned by `sf::st_bbox()`

Examples

```
library(sf)
bounding_coords <- c(25.9, 44.3, 26.2, 44.5)
bb <- as_bbox(bounding_coords)
class(bb)
st_crs(bb)
```

as_network *Create a network from a collection of line strings.*

Description

Create a network from a collection of line strings.

Usage

```
as_network(edges, flatten = TRUE, clean = TRUE)
```

Arguments

edges	A data frame with the network edges
flatten	Whether all intersections between edges should be converted to nodes
clean	Whether general cleaning tasks should be run on the generated network (see `clean_network()` for the description of tasks)

Value

A spatial network object

Examples

```
edges <- sf::st_sfc(
  sf::st_linestring(matrix(c(0, 0, 1, 1), ncol = 2, byrow = TRUE)),
  sf::st_linestring(matrix(c(0, 1, 1, 0), ncol = 2, byrow = TRUE))
)
sf::st_crs(edges) <- sf::st_crs("EPSG:4326")
as_network(edges)
```

---

bucharest_dambovita *rcrisp example delineation data for Bucharest*

Description

Delineation generated with rcrisp example data found at https://data.4tu.nl/datasets/f5d5e118-b5bd-4dfb-987f-f

Usage

```
bucharest_dambovita
```

Format

A list of sf objects representing:

valley The valley boundaries of river Dâmbovița.

corridor The corridor boundaries of river Dâmbovița.

segments Corridor segments of river Dâmbovița.

riverspace River space of Dâmbovița.

---

cache_directory *Return the cache directory used by the package*

Description

By default, the user-specific directory as returned by `tools::R_user_dir()` is used. A different directory can be used by setting the environment variable `CRISP_CACHE_DIRECTORY`. This can also be done by adding the following line to the `.Renviron` file: `CRISP_CACHE_DIRECTORY=/path/to/crisp/cache/dir`.

Usage

```
cache_directory()
```

Value

The cache directory used by rcrisp.

Examples

```
cache_directory()
```

---

check_cache *Check cache*

Description

A warning is raised if the cache size is > 100 MB or if it includes files older than 30 days.

Usage

```
check_cache()
```

Value

NULL

Examples

```
check_cache()
```

---

clean_network *Clean a spatial network.*

Description

Subdivide edges by adding missing nodes, (optionally) simplify the network (see `simplify_network()`), remove pseudo-nodes, and discard all but the main connected component.

Usage

```
clean_network(network, simplify = TRUE)
```

Arguments

network	A network object
simplify	Whether the network should be simplified with `simplify_network()`

Value

A cleaned network object

Examples

```
bucharest_osm <- get_osm_example_data()
edges <- dplyr::bind_rows(bucharest_osm$streets,
                          bucharest_osm$railways)
network <- sfnetworks::as_sfnetwork(edges, directed = FALSE)
clean_network(network)
```

---

clear_cache *Remove cache files*

Description

Remove files from cache directory either before a given date or entirely.

Usage

```
clear_cache(before_date = NULL)
```

Arguments

before_date	Date before which cache files should be removed provided as Date

Value

List of file paths of removed files

Examples

```
clear_cache()
```

default_stac_dem *Default STAC collection*

Description

Endpoint and collection ID of the default STAC collection where to access digital elevation model (DEM) data. This is the global Copernicus DEM 30 dataset hosted on AWS, as listed in the Earth-Search STAC API endpoint. Note that AWS credentials need to be set up in order to access the data (not the catalog). References:

- EarthSearch STAC API
- Copernicus DEM
- AWS Copernicus DEM datasets
- Data license

Usage

```
default_stac_dem
```

Format

An object of class `list` of length 2.

delineate *Delineate a corridor around a river.*

Description

Delineate a corridor around a river.

Usage

```
delineate(
  city_name,
  river_name,
  crs = NULL,
  network_buffer = NULL,
  buildings_buffer = NULL,
  corridor_init = "valley",
  dem = NULL,
  dem_buffer = 2500,
  max_iterations = 10,
  capping_method = "shortest-path",
  angle_threshold = 100,
  corridor = TRUE,
```

```
  segments = FALSE,
  riverspace = FALSE,
  force_download = FALSE,
  ...
)
```

Arguments

city_name	A place name as a string
river_name	A river name as a string
crs	The projected Coordinate Reference System (CRS) to use. If not provided, the suitable Universal Transverse Mercator (UTM) CRS is selected
network_buffer	Add a buffer (an integer in meters) around river to retrieve additional data (streets, railways, etc.). Default is 3000 m.
buildings_buffer	Add a buffer (an integer in meters) around the river to retrieve additional data (buildings). Default is 100 m.
corridor_init	How to estimate the initial guess of the river corridor. It can take the following values: • "valley": use the river valley boundary, as estimated from a Digital Elevation Model (DEM) (for more info see delineate_valley()) • numeric or integer: use a buffer region of the given size (in meters) around the river centerline • An sf::sf or sf::sfc object: use the given input geometry
dem	Digital elevation model (DEM) of the region (only used if corridor_init is "valley")
dem_buffer	Size of the buffer region (in meters) around the river to retrieve the DEM (only used if corridor_init is "valley" and dem is NULL).
max_iterations	Maximum number of iterations employed to refine the corridor edges (see corridor_edge()).
capping_method	The method employed to connect the corridor edge end points (i.e. to "cap" the corridor). See cap_corridor() for the available methods
angle_threshold	Only network edges forming angles above this threshold (in degrees) are considered when forming segment edges. See delineate_segments() and rcoins::stroke(). Only used if segments is TRUE.
corridor	Whether to carry out the corridor delineation
segments	Whether to carry out the corridor segmentation
riverspace	Whether to carry out the riverspace delineation
force_download	Download data even if cached data is available
...	Additional (optional) input arguments for retrieving the DEM dataset (see get_dem()). Only relevant if corridor_init is "valley" and dem is NULL

Value

A list with the corridor, segments, and riverspace geometries

Examples

```
delineate("Bucharest", "Dâmbovița")
```

`delineate_corridor` *Delineate a river corridor on a spatial network.*

Description

The corridor edges on the two river banks are drawn on the provided spatial network starting from an initial guess of the corridor (based e.g. on the river valley).

Usage

```
delineate_corridor(
  network,
  river,
  corridor_init = 1000,
  max_width = 3000,
  max_iterations = 10,
  capping_method = "shortest-path"
)
```

Arguments

`network`	The spatial network to be used for the delineation
`river`	A (MULTI)LINESTRING simple feature geometry representing the river centerline
`corridor_init`	How to estimate the initial guess of the river corridor. It can take the following values: • numeric or integer: use a buffer region of the given size (in meters) around the river centerline • An `sf::sf` or `sf::sfc` object: use the given input geometry
`max_width`	(Approximate) maximum width of the corridor. The spatial network is trimmed by a buffer region of this size around the river
`max_iterations`	Maximum number of iterations employed to refine the corridor edges (see `corridor_edge()`).
`capping_method`	The method employed to connect the corridor edge end points (i.e. to "cap" the corridor). See `cap_corridor()` for the available methods

Value

A simple feature geometry representing the river corridor

Examples

```
bucharest_osm <- get_osm_example_data()
network <- rbind(bucharest_osm$streets, bucharest_osm$railways) |>
  as_network()
delineate_corridor(network, bucharest_osm$river_centerline)
```

`delineate_riverspace` *Delineate the space surrounding a river*

Description

Delineate the space surrounding a river

Usage

```
delineate_riverspace(
  river,
  occluders = NULL,
  density = 1/50,
  ray_num = 40,
  ray_length = 100
)
```

Arguments

`river`	List with river surface and centerline
`occluders`	Geometry of occluders
`density`	Density of viewpoints
`ray_num`	Number of rays
`ray_length`	Length of rays in meters

Value

Polygon geometry with the riverspace

Examples

```
## Not run:
  bucharest_osm <- get_osm_example_data()
  delineate_riverspace(bucharest_osm$river_surface, bucharest_osm$buildings)

## End(Not run)
```

delineate_segments *Split a river corridor into segments*

Description

Segments are defined as corridor subregions separated by river-crossing transversal lines that form continuous strokes in the network.

Usage

```
delineate_segments(corridor, network, river, angle_threshold = 100)
```

Arguments

corridor	The river corridor as a simple feature geometry
network	The spatial network to be used for the segmentation
river	The river centerline as a simple feature geometry
angle_threshold	Only consider angles above this threshold (in degrees) to form continuous strokes in the network. See `rcoins::stroke()` for more details.

Value

Segment polygons as a simple feature geometry

Examples

```
bucharest_osm <- get_osm_example_data()
corridor <- bucharest_dambovita$corridor
network <- rbind(bucharest_osm$streets, bucharest_osm$railways) |>
  as_network()
river <- bucharest_osm$river_centerline |> sf::st_geometry()
delineate_segments(corridor, network, river)
```

delineate_valley *Extract the river valley from the DEM*

Description

The slope of the digital elevation model (DEM) is used as friction (cost) surface to compute the cost distance from any grid cell of the raster to the river. A characteristic value (default: the mean) of the cost distance distribution in a region surrounding the river (default: a buffer region of 2 km) is then calculated, and used to threshold the cost-distance surface. The resulting area is then "polygonized" to obtain the valley boundary as a simple feature geometry.

Usage

```
delineate_valley(dem, river)
```

Arguments

dem	Digital elevation model of the region
river	A simple feature geometry representing the river

Value

River valley as a simple feature geometry

Examples

```
bucharest_osm <- get_osm_example_data()
bucharest_dem <- get_dem_example_data()
delineate_valley(bucharest_dem, bucharest_osm$river_centerline)
```

---

dem_to_cog *Write DEM to cloud optimized GeoTiff file as specified location*

Description

Write DEM to cloud optimized GeoTiff file as specified location

Usage

```
dem_to_cog(dem, fpath, output_directory = NULL)
```

Arguments

dem	to write to file
fpath	filepath for output. If no output directory is specified (see below) fpath is parsed to determine the output directory
output_directory	where file should be written. If specified fpath is treated as filename only.

Value

The input DEM. This function is used for the side-effect of writing values to a file.

Examples

```
bucharest_dem <- get_dem_example_data()
dem_to_cog(bucharest_dem, "bucharest_dem.tif")
```

flatten_network *Flatten a network by adding points at apparent intersections.*

Description

All crossing edges are identified, and the points of intersections are injected within the edge geometries. Note that the injected points are not converted to network nodes (this can be achieved via sfnetworks' `sfnetworks::to_spatial_subdivision()`, which is part of the tasks that are included in `clean_network()`.

Usage

```
flatten_network(network)
```

Arguments

network	A network object

Details

The functionality is similar to sfnetworks' `sfnetworks::st_network_blend()`, but in that case an external point is only injected to the closest edge.

Value

A network object with additional points at intersections

Examples

```
bucharest_osm <- get_osm_example_data()
edges <- dplyr::bind_rows(bucharest_osm$streets,
                          bucharest_osm$railways)
network <- sfnetworks::as_sfnetwork(edges, directed = FALSE)
flatten_network(network)
```

get_dem *Access digital elevation model (DEM) for a given region*

Description

Access digital elevation model (DEM) for a given region

Usage

```
get_dem(
  bb,
  dem_source = "STAC",
  stac_endpoint = NULL,
  stac_collection = NULL,
  crs = NULL,
  force_download = FALSE
)
```

Arguments

bb	A bounding box, provided either as a matrix (rows for "x", "y", columns for "min", "max") or as a vector ("xmin", "ymin", "xmax", "ymax"), in lat/lon coordinates (WGS84 coordinate referece system)
dem_source	Source of the DEM: • If "STAC" (default), DEM tiles are searched on a SpatioTemporal Asset Catalog (STAC) end point, then accessed and mosaicked to the area of interest
stac_endpoint	URL of the STAC API endpoint (only used if dem_source is "STAC"). For more info, see get_stac_asset_urls()
stac_collection	Identifier of the STAC collection to be queried (only used if dem_source is "STAC"). For more info, see get_stac_asset_urls()
crs	Coordinate reference system (CRS) which to transform the DEM to
force_download	Download data even if cached data is available

Value

DEM as a terra SpatRaster object

Examples

```
bb <- get_osm_bb("Bucharest")
get_dem(bb)
```

---

get_dem_example_data *Get example DEM data*

Description

This function retrieves example Digital Elevation Model (DEM) data from a persistent URL on the 4TU.ResearchData data repository, and it can be used in examples and tests.

Usage

```
get_dem_example_data()
```

Value

A SpatRaster object containing the DEM data.

Examples

```
get_dem_example_data()
```

---

get_osmdata *Retrieve OpenStreetMap data for a given location*

Description

Retrieve OpenStreetMap data for a given location, including the city boundary, the river centreline and surface, the streets, the railways, and the buildings

Usage

```
get_osmdata(
  city_name,
  river_name,
  network_buffer = NULL,
  buildings_buffer = NULL,
  city_boundary = TRUE,
  crs = NULL,
  force_download = FALSE
)
```

Arguments

city_name	A character string with the name of the city.
river_name	A character string with the name of the river.
network_buffer	Buffer distance in meters around the river to get the streets and railways, default is 0 means no network data will be downloaded
buildings_buffer	Buffer distance in meters around the river to get the buildings, default is 0 means no buildings data will be downloaded
city_boundary	A logical indicating if the city boundary should be retrieved. Default is TRUE.
crs	An integer with the EPSG code for the projection. If no CRS is specified, the default is the UTM zone for the city.
force_download	Download data even if cached data is available

Value

An list with the retrieved OpenStreetMap data sets for the given location

Examples

```
get_osmdata("Bucharest", "Dâmbovița")
```

`get_osm_bb` *Get the bounding box of a city*

Description

Get the bounding box of a city

Usage

```
get_osm_bb(city_name)
```

Arguments

`city_name`	The name of the city

Value

A bbox object with the bounding box of the city

Examples

```
get_osm_bb("Bucharest")
```

`get_osm_buildings` *Get OpenStreetMap buildings*

Description

Get buildings from OpenStreetMap within a given buffer around a river.

Usage

```
get_osm_buildings(aoi, crs = NULL, force_download = FALSE)
```

Arguments

aoi	Area of interest as sf object or bbox
crs	Coordinate reference system as EPSG code
force_download	Download data even if cached data is available

Value

An sf object with the buildings

Examples

```
bb <- get_osm_bb("Bucharest")
crs <- get_utm_zone(bb)
get_osm_buildings(bb, crs)
```

---

get_osm_city_boundary *Get the city boundary from OpenStreetMap*

Description

This function retrieves the city boundary from OpenStreetMap based on a bounding box with the OSM tags "place:city" and "boundary:administrative". The result is filtered by the city name.

Usage

```
get_osm_city_boundary(
  bb,
  city_name,
  crs = NULL,
  multiple = FALSE,
  force_download = FALSE
)
```

Arguments

bb	Bounding box of class bbox
city_name	A character string with the name of the city
crs	Coordinate reference system as EPSG code
multiple	A logical indicating if multiple city boundaries should be returned. By default, only the first one is returned.
force_download	Download data even if cached data is available

Value

An sf object with the city boundary

Examples

```
bb <- get_osm_bb("Bucharest")
crs <- get_utm_zone(bb)
get_osm_city_boundary(bb, "Bucharest", crs)
```

---

get_osm_example_data *Get example OSM data*

Description

This function retrieves example OpenStreetMap (OSM) data from a persistent URL on the 4TU.ResearchData data repository, and it can be used in examples and tests.

Usage

```
get_osm_example_data()
```

Value

A list of sf objects containing the OSM data.

Examples

```
get_osm_example_data()
```

---

get_osm_railways *Get OpenStreetMap railways*

Description

Get OpenStreetMap railways

Usage

```
get_osm_railways(
  aoi,
  crs = NULL,
  railway_values = "rail",
  force_download = FALSE
)
```

Arguments

aoi	Area of interest as sf object or bbox
crs	Coordinate reference system as EPSG code
railway_values	A character or character vector with the railway values to retrieve.
force_download	Download data even if cached data is available

Value

An sf object with the railways

Examples

```
bb <- get_osm_bb("Bucharest")
crs <- get_utm_zone(bb)
get_osm_railways(bb, crs)
```

---

get_osm_river *Get the river centreline and surface from OpenStreetMap*

Description

Get the river centreline and surface from OpenStreetMap

Usage

```
get_osm_river(bb, river_name, crs = NULL, force_download = FALSE)
```

Arguments

bb	Bounding box of class bbox
river_name	The name of the river
crs	Coordinate reference system as EPSG code
force_download	Download data even if cached data is available

Value

A list with the river centreline and surface

Examples

```
bb <- get_osm_bb("Bucharest")
crs <- get_utm_zone(bb)
get_osm_river(bb, "Dâmbovița", crs)
```

get_osm_streets *Get OpenStreetMap streets*

Description

Get OpenStreetMap streets

Usage

```
get_osm_streets(aoi, crs = NULL, highway_values = NULL, force_download = FALSE)
```

Arguments

aoi	Area of interest as sf object or bbox
crs	Coordinate reference system as EPSG code
highway_values	A character vector with the highway values to retrieve. If left NULL, the function retrieves the following values: "motorway", "trunk", "primary", "secondary", "tertiary"
force_download	Download data even if cached data is available

Value

An sf object with the streets

Examples

```
bb <- get_osm_bb("Bucharest")
crs <- get_utm_zone(bb)
get_osm_streets(bb, crs)
```

get_river_aoi *Get an area of interest (AoI) around a river, cropping to the bounding box of a city*

Description

Get an area of interest (AoI) around a river, cropping to the bounding box of a city

Usage

```
get_river_aoi(river, city_bbox, buffer_distance)
```

Arguments

river	A list with the river centreline and surface geometries
city_bbox	Bounding box around the city
buffer_distance	Buffer size around the river

Value

An sf object in lat/lon coordinates

Examples

```
bb <- get_osm_bb("Bucharest")
river <- get_osm_river(bb, "Dâmbovița")
get_river_aoi(river, bb, buffer_distance = 100)
```

---

get_stac_asset_urls *Retrieve the URLs of all the assets intersecting a bbox from a STAC API*

Description

Retrieve the URLs of all the assets intersecting a bbox from a STAC API

Usage

```
get_stac_asset_urls(bb, endpoint = NULL, collection = NULL)
```

Arguments

bb	A bounding box, provided either as a matrix (rows for "x", "y", columns for "min", "max") or as a vector ("xmin", "ymin", "xmax", "ymax"), in lat/lon coordinates (WGS84 coordinate referece system)
endpoint	URL of the STAC API endpoint. To be provided together with stac_collection, or leave blank to use defaults (see default_stac_dem)
collection	Identifier of the STAC collection to be queried. To be provided together with stac_endpoint, or leave blank to use defaults (see default_stac_dem)

Value

A list of URLs for the assets in the collection overlapping with the specified bounding box

Examples

```
bb <- get_osm_bb("Bucharest")
get_stac_asset_urls(bb)
```

get_utm_zone *Get the UTM zone of a spatial object*

Description

Get the UTM zone of a spatial object

Usage

```
get_utm_zone(x)
```

Arguments

x	Bounding box or geometry object

Value

The EPSG code of the UTM zone

Examples

```
# Get EPSG code for UTM zone of Bucharest
bb <- get_osm_bb("Bucharest")
get_utm_zone(bb)
```

load_dem *Retrieve DEM data from a list of STAC assets*

Description

Load DEM data from a list of tiles, crop and merge using a given bounding box to create a raster DEM for the specified region. Results are cached, so that new queries with the same input parameters will be loaded from disk.

Usage

```
load_dem(bb, tile_urls, force_download = FALSE)
```

Arguments

bb	A bounding box, provided either as a matrix (rows for "x", "y", columns for "min", "max") or as a vector ("xmin", "ymin", "xmax", "ymax")
tile_urls	A list of tiles where to read the DEM data from
force_download	Download data even if cached data is available

Value

Raster DEM, retrieved and retiled to the given bounding box

Examples

```
bb <- get_osm_bb("Bucharest")
tile_urls <- get_stac_asset_urls(bb)
load_dem(bb, tile_urls)
```

`osmdata_as_sf` *Retrieve OpenStreetMap data as sf object*

Description

Query the Overpass API for a key:value pair within a given bounding box (provided as lat/lon coordiates). Results are cached, so that new queries with the same input parameters will be loaded from disk.

Usage

```
osmdata_as_sf(key, value, aoi, force_download = FALSE)
```

Arguments

`key`	A character string with the key to filter the data
`value`	A character string with the value to filter the data
`aoi`	An area of interest, provided either as as sf object or "bbox" or as a vector ("xmin", "ymin", "xmax", "ymax")
`force_download`	Download data even if cached data is available

Value

An sf object with the retrieved OpenStreetMap data

Examples

```
bb <- get_osm_bb("Bucharest")
osmdata_as_sf("highway", "motorway", bb)
```

`reproject` *Reproject a raster or vector dataset to the specified coordinate reference system (CRS)*

Description

Reproject a raster or vector dataset to the specified coordinate reference system (CRS)

Usage

```
reproject(x, crs, ...)
```

Arguments

`x`	Raster or vector object
`crs`	CRS to be projected to
`...`	Optional arguments for raster or vector reproject functions

Value

Object reprojected to specified CRS

Examples

```
# Reproject a raster to EPSG:4326
r <- terra::rast(matrix(1:12, nrow = 3, ncol = 4), crs = "EPSG:32633")
reproject(r, 4326)
```

Index