Osmenrich: An R package to enrich geocoded data

Erik-Jan van Kesteren[1, 2], **Leonardo Jaya Vida**[1, 2], **and Jonathan de Bruin**[1, 2]

1 Utrecht University **2** ODISSEI

DOI: DOIunavailable

Software

- Review
- Repository
- Archive

Editor: Pending Editor

Reviewers:

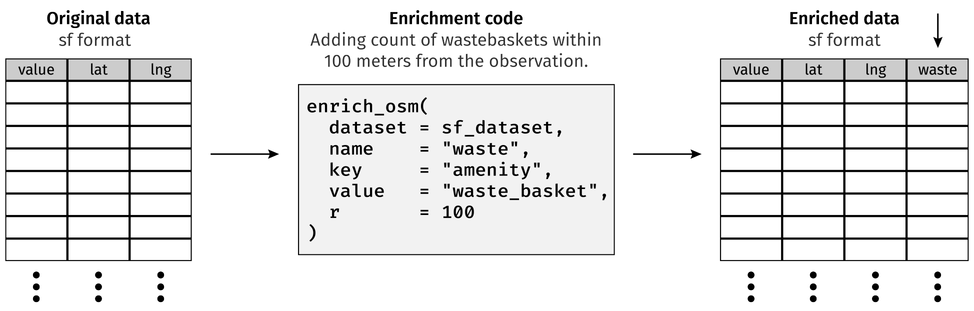
- @Pending Reviewers

Submitted: N/A
Published: N/A

Summary

The `osmenrich` package provides a user-friendly way to enrich geographic datasets in `R`, for example observations on a map, with geographic features around those observations and to weight these features by their distance or duration from the observations. This package builds on existing infrastructure to interface with OpenStreetMap (`osmdata`, Padgham et al. (2017)), and works well with the existing ecosystem of packages for (geospatial) data analysis, namely `sf` (E. Pebesma, 2018) and the `tidyverse` (Hadley Wickham et al., 2019). Thus, this package streamlines and standardizes the process going from a raw dataset with observations and locations to a `tidy` (Wickham, 2014), rich dataset with multiple relevant geographical features about those locations. Figure Figure 1 shows graphically the basic workflow of `osmenrich`.

The R package `osmenrich` is available on GitHub.



```
enrich_osm(
  dataset = sf_dataset,
  name    = "waste",
  key     = "amenity",
  value   = "waste_basket",
  r       = 100
)
```

Figure 1: Basic workflow of `osmenrich`.

Statement of need

Geographic data is valuable in research where the environment influences the process under investigation. For example, the `osmenrich` package is useful in the analysis of data retrieved from citizen science projects such as plastic spotter or the great backyard bird count. At the same time, within the R ecosystem multiple software solutions exist for extracting data from geographic information systems Walker (2020). However, to include these geographic data in further analysis (e.g. carrying out kriging in `gstat`, E. J. Pebesma (2004)), the data often need further processing and, crucially, *aggregation*. Within this problem space, the contributions of `osmenrich` are as follows:

- Creating a user-friendly interface to OpenStreetMap, abstracting away the necessary API calls.
- Defyining standardized ways to aggregate geographic information based on kernels (see section).

- Allowing distance measures based on routing, such as duration by foot or distance by car (see section).

Using our package, researchers can focus on research questions, rather than spending time figuring out how to aggregate geographic data. The `osmenrich` package can be especially useful answering questions relative to identifying relevant features to explain the process under investigation within a determined distance from observations.

Before describing the main function and features of this package, we introduce the grammar used in this paper. We call "*reference points*" the objects with geocoded data that a researcher wants to enrich, while "*feature points*" the objects the researcher is interested in retrieving. If a dataset contains geocoded data, with the `osmenrich` package one can extract information about real-world objects (*feature points*) around each of the *reference points* contained in the dataset, compute the distance/duration between them and enrich the initial dataset with this information. The result is a `tidy sf` dataset.

Main function

To enrich data, the `osmenrich` package uses the main function `enrich_osm()`. This function takes a dataset containing geocoded *reference points* in `sf` format, retrieves specified points from a local or remote OpenStreetMap server, computes the enrichment with the specified parameters and outputs an enriched `sf` dataset.

```
enrich_osm(
  dataset = sf_dataset,
  name = "waste",
  key = "amenity",
  value = "waste_basket",
  r = 100,
  # kernel = "uniform", # default kernel function
  # reduce_fun = "sum", # default aggregation function
  # distance = "spherical", # default type of distance measure
  # type = "points", # default type of shape retrieved from OpenStreetMap
)
```

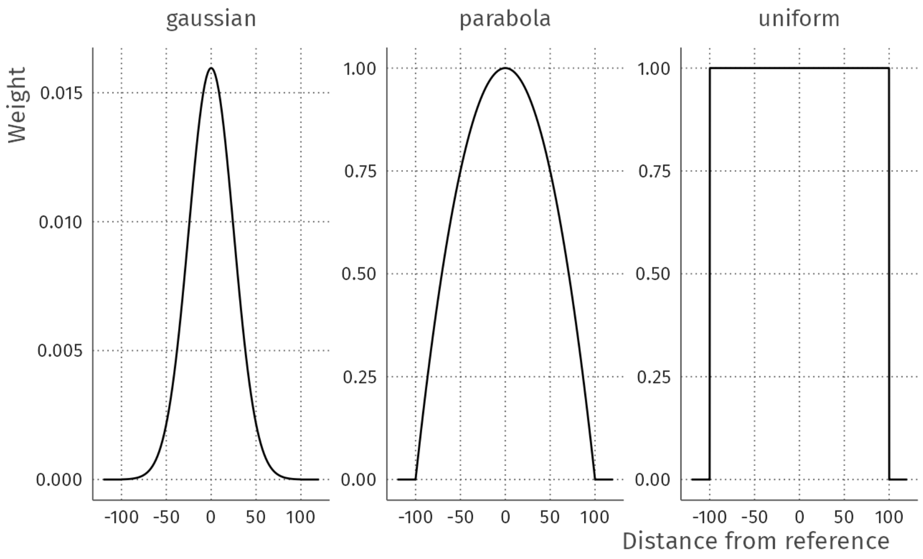
The example above shows an example of query and displays values that can be left to default. Specifically, the function uses the bounding box created by the *reference points* from the input dataset and searches for the specified *feature points* of type `type`, in OpenStreetMap using the parameters `key` and `value` within the radius `r` around each of the *reference points*. For each *reference points* it uses the `kernel`, `reduce_fun` and `distance` parameters, that can be left to their deafult values, to compute the enrichment and finally adds a newly created column named using the variable `name` containing the enriched data. See section section for an example usage of this function.

Kernels

Kernels are a tool to weight and aggregate the features retrieved from OpenStreetMap and play a center role in the data enrichment process used in `osmenrich`. Kernels first weight the points retrieved by their distances (or durations) from the reference points and then convert these vectors into single numbers. This conversion is done by specifying a kernel or by specifying a kernel and the aggregation function used to reduce these weighted vectors of points into single numbers.

There are three main variables involved in the specification of a kernel:

1. The radius parameter `r`, indicates the distance from each of the observation points on which the kernel will be applied.
2. The kernel parameter `kernel`, indicates the kernel function to use in weighting the retrieved points within the specified radius. The `osmenrich` package provides three different kernels to be used out-of-the-box (`uniform`, `gaussian` and `parabola`) and defaults to `kernel = uniform`. However, the user can also specify custom-made kernels as long as they follow the .



3. The aggregation function parameter `reduce_fun`, is used to choose the aggregation function that will be applied to the weighted points. This parameters defaults to `reduce_fun = sum`, however it accepts any standard `R` function, such as `mean` or `median`.

Specifying these variables in the `enrich_osm()` function, allows the user to choose the specify the type of weighting and aggregation to be applied on the features retrieved from OpenStreetMap.

```
enrich_osm(
  [...],
  r = 100, # radius for features retrieval
  kernel = 'gaussian', # weighting function
  reduce_fun = 'mean', # aggregation function
)
```

Routing

To retrieve *feature points* around the *reference points* and the distances (or durations) between these points, the `osmenrich` package makes use of an instance of OpenStreetMap and one or more instances of Open Source Routing Machine (OSRM) respectively.

The basic data enrichment will work without having to setup any one of these server locally, thanks to publicly available servers. However, for large data enrichment tasks and for tasks involving the computation of durations between *reference points* and *feature points* and/or the computation of custom distances or durations between these points (such as the distances between two points computed on a walking distance or cycling), the setup of one or more of these servers is required.

We created a GitHub repository hosting the instruction and the `docker_compose.yml` files needed to setup these servers. To facilitate the routing of users to the right setup for their need, we provide three use cases and their respective recommended setup:

1. **Base use-case**. The user only wants to enrich adding nearby features. The user is not interested in distances nor durations as measured by OSRM servers. The setup of local or remote instance is not recommended.
2. **Normal use-case**. The user only wants to enrich adding features for a large area. The user might be interested in distances but does not require a specific metric (i.e. car vs. foot vs. bike). Only the setup of the OpenStreetMap server is recommended. The OSRM connection will rely on public servers.
3. **Advanced use-case**. The user wants to enrich adding features for a large area and/or is interested in specific metrics for distances/durations (i.e. foot or bike). The setup of both the OpenStreetMap and OSRM servers is recommended.

Finally, to specify which distance metric to use, the user can set the parameter `distance` to the desired choice as in the example below.

```r
# Specify the address of local OSRM instance
# options(osrm.server = "http://localhost:<port>/")
# You can specify also the address of the Overpass (OSM) instance
# osmdata::set_overpass_url("http://localhost:<port>/api/interpreter")
enrich_osm(
  [...],
  distance = "distance_by_foot", # Specifying a routing using the OSRM server
)
```

Full usage

`osmenrich` is available on GitHub and can be installed and loaded into the R session with the remotes package from GitHub. Then the package can be loaded in the usual way:

```r
library(osmenrich)
```

As a brief example, suppose we have a dataset of common swift's nests provided openly by the city of Utrecht, the Netherlands. We want to enrich this dataset retrieving a proxy of natural material availability. For the sake of this example, we retrieve only "trees," as they represent a close enough proxy to natural materials. However, we could easily retrieve other data types, using the available list on OpenStreetMap at Map Features

```r
head(common_swift())
# Show the output of the command

bird_sf <-
  bird_sf %>%
  enrich_osm(
    name = "tree_1km",
    key = "natural",
    value = "tree",
    kernel = "gaussian",
    r = 1000
  )

bird_sf
# Show the output of the command
```

Acknowledgements

References

Cooley, D. (2020). *Googleway: Accesses google maps APIs to retrieve data and plot maps*. https://CRAN.R-project.org/package=googleway

Hadley Wickham et al. (2019). Welcome to the tidyverse. *Journal of Open Source Software*, *4*(43), 1686. https://doi.org/10.21105/joss.01686

Padgham, M., Rudis, B., Lovelace, R., & Salmon, M. (2017). Osmdata. *The Journal of Open Source Software*, *2*(14). https://doi.org/10.21105/joss.00305

Pebesma, E. (2018). Simple Features for R: Standardized Support for Spatial Vector Data. *The R Journal*, *10*(1), 439–446. https://doi.org/10.32614/RJ-2018-009

Pebesma, E. J. (2004). Multivariable geostatistics in S: The gstat package. *Computers & Geosciences*, *30*, 683–691.

Walker, K. (2020). *Mapboxapi: R interface to 'mapbox' web services*. https://CRAN.R-project.org/package=mapboxapi

Wickham, H. (2014). Tidy data. *Journal of Statistical Software, Articles*, *59*(10), 1–23. https://doi.org/10.18637/jss.v059.i10