# infostop Documentation

***Release 0.1.8***

**Ulf Aslak**

**Sep 17, 2021**

# Contents

CHAPTER 1

# About this project

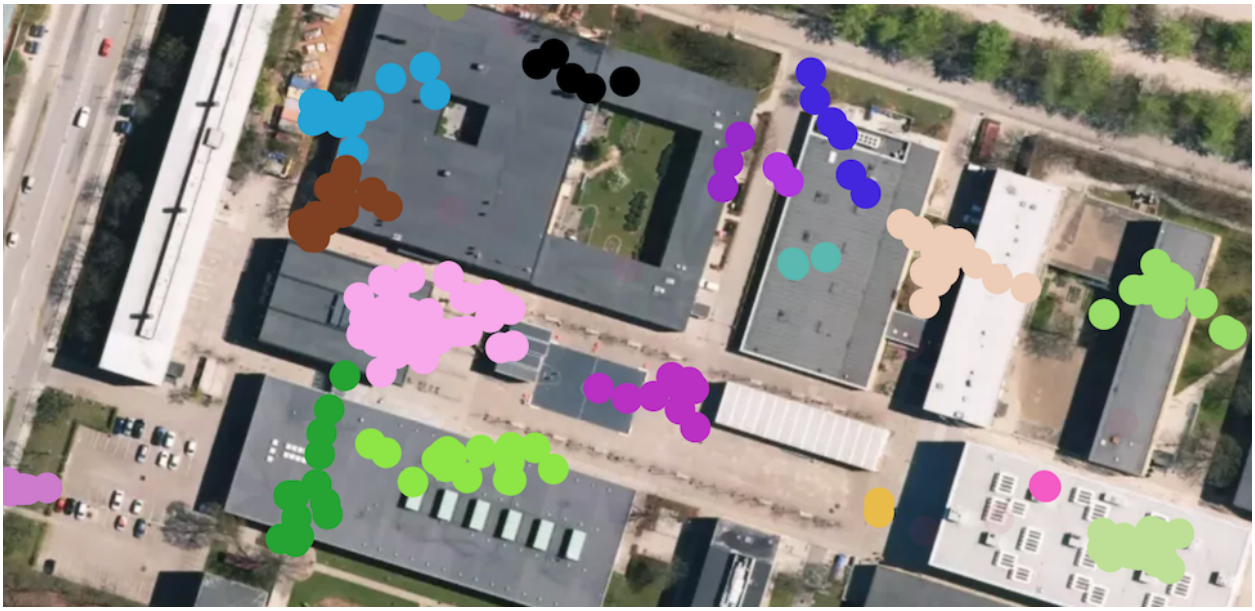
Infostop is a minimal and fast tool for performing network clustering on spacial data, and in particular it is well suited for detecting stop locations, or points of interest, in GPS data.

## 1.1 Quick example

For unlabeled GPS trace *data*:

```
>>> import infostop
>>> import numpy as np
    >>> data
    array([[ 55.75259295,  12.34353885 ],
           [ 55.7525908 ,  12.34353145 ],
           [ 55.7525876 ,  12.3435386  ],
           ...,
           [ 63.40379175,  10.40477095 ],
           [ 63.4037841 ,  10.40480265 ],
           [ 63.403787  ,  10.4047871  ]])
    >>> labels = infostop.label_trace(data)
    >>> np.hstack([data, labels.reshape(-1, 1)])
    array([[ 55.75259295,  12.34353885,   0.        ],
           [ 55.7525908 ,  12.34353145,   0.        ],
           [ 55.7525876 ,  12.3435386 ,  1-.        ],
           ...,
           [ 63.40379175,  10.40477095, 164.        ],
           [ 63.4037841 ,  10.40480265, 164.        ],
           [ 63.403787  ,  10.4047871 , 164.        ]])
```

The coordinates can now be plotted onto a map, colored by the assigned labels:



## 1.2 Why should I use Infostop

### 1.2.1 Pros

- Quick Python based API for finding important places in your GPS data
- Installable with pip, no compiling needed
- No external program needed
- Cross-platform
- More meaningful clustering solutions than DBSCAN (or similar) based methods

### 1.2.2 Cons

- Not as fast as DBSCAN based methods (although still relatively fast)
- No plotting options yet

## 1.3 Install

```
pip install infostop
```

Make sure to read the `README.md` in the public repository for notes on dependencies and installation.

`infostop` depends on the following packages which will be installed by `pip` during the installation process

- `numpy>=0.14`
- `infomap>=3.0.12`

## 1.4 Bug reports & contributing

You can contribute to the public repository and raise issues there.

CHAPTER 2

# Label location trace

For a time-ordered series of coordinates, such as a GPS trace, annotate each point with a location label. Such a coordinate series may look like:

```
array([[ 55.75259295,  12.34353885 ],
       [ 55.7525908 ,  12.34353145 ],
       [ 55.7525876 ,  12.3435386  ],
       ...,
       [ 63.40379175,  10.40477095 ],
       [ 63.4037841 ,  10.40480265 ],
       [ 63.403787  ,  10.4047871  ]])
```

The trace can then be labeled using `infostop.detect.label_trace`:

```
labels = infostop.label_trace(data)
labels
# array([0, 0, -1, ..., 164, 164, 164])
```

And stacked with the input data:

```
    np.hstack([data, labels.reshape(-1, 1)])
array([[ 55.75259295,  12.34353885,   0.        ],
       [ 55.7525908 ,  12.34353145,   0.        ],
       [ 55.7525876 ,  12.3435386 ,  -1.        ],
       ...,
       [ 63.40379175,  10.40477095, 164.        ],
       [ 63.4037841 ,  10.40480265, 164.        ],
       [ 63.403787  ,  10.4047871 , 164.        ]])
```

Labels 0 and 164 in this example correspond to locations where the user was static, and possibly returned. The label -1 is given to location measurements which are recorded while the user is not stationary.

It is relevant for the user to consider first setting the `r1` parameter in `infostop.detect.label_trace`, which sets the threshold for the maximum distance between two consecutive location measurements allowed within the same stop location. As a rule of thumb, `r1` should be slightly greater than the standard deviation of the uncertainty distribution on the location measurement.

To get bigger stop locations consider increasing `r2`, which sets the distance threshold for link creation.

If the input data is not GPS points, the `distance_function` parameter should be set to a valid metric, and `r1` and `r2` should most likely be specified to something meaningful. Per default, Infostop assumes that points are (lat, lon) and uses the haversine distance function, where the thresholds are given in meters.

## 2.1 Label timestamped trace

In a third (rightmost) column, timestamps can be listed so the input series looks like:

```
array([[ 55.75259295,  12.34353885,  1356998400 ],
       [ 55.7525908 ,  12.34353145,  1356998700 ],
       [ 55.7525876 ,  12.3435386 ,  1356999000 ],
       ...,
       [ 63.40379175,  10.40477095,  1357085400 ],
       [ 63.4037841 ,  10.40480265,  1357085700 ],
       [ 63403787  ,  10.4047871  ,  1357086000 ]])
```

This is useful, as it allows the user to specify in the `infostop.detect.label_trace` function two new parameters: `min_staying_time` which sets a lower bound on how short stops can be, and `max_time_between` which sets an upper bound on the longest allowed time between two samples before a new stop event is created.

CHAPTER 3

# Label static coordinates

In cases where the user needs to cluster a collection of points (i.e. static coordinates) `infostop.detect.label_static_points` is useful. It creates a network of locations where locations that are closer than *r2* to each other are linked, and then clusters this network using Infomap. Since Infomap is a flowbased clustering algorithm, it can detect overlapping clusters (i.e. stop locations or points of interest) as seperate locations.

CHAPTER 4

# Label distance matrix

Given an `(N, N)` distance matrix for a set of coordinates `infostop.detect.label_distance_matrix` returns the same result as `infostop.detect.label_static_points` does for a `(N, 2)` array of coordinates.

CHAPTER 5

# Detect module

CHAPTER 6

# Utils module