$_{t}acomaDocumentation$

Release 0.0.27

Benjamin F. Maier

Sep 14, 2018

Contents

CHAPTER 1

About this project

Tacoma is an acronym for *(T)empor(A)l (CO)ntact (M)odeling and (A)nalysis*. It is a joint C++/Python-package for the modeling and analysis of undirected and unweighted temporal networks, with a focus on (but not limited to) human face-to-face contact networks.

1.1 Quick example

In order to download the SocioPatterns 'Hypertext 2009'-dataset and visualize it interactively, do the following.

```
>>> import tacoma as tc
>>> from tacoma.interactive import visualize
>>> temporal_network = tc.download_and_convert_sociopatterns_hypertext_2009()
100% [..................................................] 67463 / 67463
>>> visualize(temporal_network, frame_dt = 20)
```

This is the result:

1.2 Why should I use tacoma?

1.2.1 Pros

- networks are natively described in continuous time
- two main native formats to describe temporal networks (`_tacoma.edge_lists` and `_tacoma.edge_changes`), a third way, a sorted list of `on`-intervals for each edge called `tc.edge_trajectories` is available, but algorithms work on the two native formats only
- the simple portable file-format `.taco` as a standardized way to share temporal network data (which is just the data dumped to a `.json`-file, a simple file format readable from a variety of languages)

- easy functions to produce surrogate temporal networks from four different models
- easy way to simulate Gillespie (here, epidemic spreading) processes on temporal networks
- easy framework to develop new Gillespie-simulations algorithms on temporal networks
- multiple and simple ways to interactively visualize temporal networks
- simple functions to manipulate temporal networks (slice, concatenate, rescale time, sample, bin, convert)
- simple functions to analyze structural and statistical properties of temporal networks (mean degree, degree distribution, group size distribution, group life time distributions, etc.)
- fast algorithms due to C++-core (*fast* as in *faster than pure Python*)
- relatively fast and easy to compile since it only depends on the C++11-stdlib and pybind11 without the large overhead of `Boost`

1.2.2 Cons

- no support for directed temporal networks yet
- no support for weighted temporal networks yet

1.3 Install

If you get compiling errors, make sure that pybind11 is installed.

```
$ git clone https://github.com/benmaier/tacoma
$ pip install ./tacoma
```

Note that a C++11-compiler has to be installed on the system before installing `tacoma`.

CHAPTER 2

Temporal network classes

Undirected and unweighted temporal networks are composed of N nodes and up to $m_{\max} = N(N+1)/2$ edges, where each edge (i,j) can be described as a series of events where the edge is either switched on or switched off. One way of expressing that is to define the temporal adjacency matrix

$$A_{ij}(t) = \begin{cases} 1, & (i,j) \text{ connected at time } t \\ 0, & \text{else.} \end{cases}$$

In *tacoma*, we will interpret temporal networks as if they were recorded in an experiment. We expect that over the course of time $t_0 \leq t < t_{\max}$ in which we record activity, we will encounter N nodes from the node set $V = 0, 1, \ldots, N-1$ (nodes posses an integer label).

The experiment begins at time t_0, where the network consists of an edge set $E_0 \subseteq \{i, j : V \times V, i < j\}$. Then, each time the network changes, we denote that time by an entry in a time vector t. Each entry in the time vector corresponds to a network change event and thus to a change in the edge set. We call the total number of change events N_e, such that the vector t has N_e entries. In between consecutive times, the network is constant. After the last recorded event, we kept the experiment running until the maximum time $t_{\max}$ without observing any change and stopped recording at $t_{\max}$.

There's three data structures implemented in this package, all of which capture the situation described above in different ways and are useful in different situations.

2.1 Edge lists

The class `_tacoma.edge_lists` consists of a collection of complete edge lists, each time the network changes, a complete edge list of the network after the change is saved. It has the following attributes.

- N : The total number of nodes
- t : A vector of length $N_e + 1$. The 0-th entry contains the time of the beginning of the experiment t_0
- *edges* : A vector of length $N_e + 1$ where each entry contains an edge list, describing the network after the change which occured at the corresponding time in t. The 0-th entry contains the edge list of the beginning of the experiment t_0

- $t_{\max}$: The time at which the experiment ended.

Additionally,

2.2 Edge changes

The class `_tacoma.edge_changes` consists of a collection of both edges being created and edges being deleted. It has the following attributes.

- N : The total number of nodes.
- t_0 : The time of the beginning of the experimen.
- *edges_initial* : The edge list of the beginning of the experiment at t_0.
- t : A vector of length N_e, each time corresponding to a change in the network.
- $t_{\max}$: The time at which the experiment ended.

Additionally,

2.3 Edge trajectories

CHAPTER 3

Relevant C++-core classes

CHAPTER 4

API module

CHAPTER 5

Analysis module

A reference to `tacoma.analysis`.

CHAPTER 6

Tool module

CHAPTER 7

Drawing module

CHAPTER 8

Models conversion module

CHAPTER 9

Flockwork module

CHAPTER 10

Loading model parameters module