A Small Example

David Wu

2024-11-11

Here, we will clean and process a small database into a temporal network of patient transfers.

All the functionality is provided in the `hospinet` library.

```python
import hospinet

import polars as pl
import networkx as nx

from pprint import pprint
from matplotlib import pyplot as plt # for later
```

We load the database (in the form of a csv), and clean it using the `hospinet.cleaner` submodule. We specify the column names, since they are different from the (standardised) default.

```python
source_db = hospinet.cleaner.ingest_csv("./data/admissions.csv", convert_dates=True)
source_db.head(5)
```

patient	hospital	admission	discharge
i64	str	datetime[s]	datetime[s]
3	”D”	2020-04-02 02:56:19.917759	2021-06-04 14:58:15.715780
2	”E”	2020-04-10 13:56:21.043894	2021-06-07 14:35:45.036138
3	”B”	2020-04-15 03:22:06.938845	2020-05-27 10:14:20.476332
2	”A”	2020-04-20 20:33:17.858555	2020-05-23 10:31:03.186415
1	”E”	2020-04-26 03:36:11.245264	2021-01-08 13:36:08.182081

```python
cleaned_db = hospinet.cleaner.clean_database(
    source_db,
    delete_missing="record",
    delete_errors="record",
    subject_id = 'patient',
    facility_id = 'hospital',
    admission_date = 'admission',
    discharge_date = 'discharge',
    retain_auxiliary_data = True,
)
# encode dates to numerics
first_date = cleaned_db.select(pl.col('Adate').min()).item()
numeric_db = hospinet.cleaner.normalise_dates(
    cleaned_db,
    cols = ['Adate', 'Ddate'],
    ref_date = first_date
)
```

```
INFO::hospinet::Checking existence of columns...
INFO::hospinet::Column existence OK.
INFO::hospinet::Standardising column names...
INFO::hospinet::Coercing types...
INFO::hospinet::Type coercion done.
INFO::hospinet::Checking for missing values...
INFO::hospinet::Checking for erroneous records...
INFO::hospinet::Removing duplicate records...
INFO::hospinet::Finding and fixing overlapping records...
INFO::hospinet::Attempting up to 100 iterations
INFO::hospinet::Iteration 0: 144 entries; 92 overlaps; 0.0031626970157958567 s
INFO::hospinet::Iteration 1: 171 entries; 62 overlaps; 0.0034291570191271603 s
INFO::hospinet::Iteration 2: 194 entries; 50 overlaps; 0.003166757000144571 s
INFO::hospinet::Iteration 3: 215 entries; 44 overlaps; 0.0029579480178654194 s
INFO::hospinet::Iteration 4: 234 entries; 38 overlaps; 0.0034177099587395787 s
INFO::hospinet::Iteration 5: 250 entries; 34 overlaps; 0.002265217015519738 s
INFO::hospinet::Iteration 6: 265 entries; 32 overlaps; 0.0024220430059358478 s
INFO::hospinet::Iteration 7: 276 entries; 28 overlaps; 0.0031610269797965884 s
INFO::hospinet::Iteration 8: 286 entries; 21 overlaps; 0.0027002019924111664 s
INFO::hospinet::Iteration 9: 294 entries; 18 overlaps; 0.0024816050427034497 s
INFO::hospinet::Iteration 10: 302 entries; 16 overlaps; 0.0023534849751740694 s
INFO::hospinet::Iteration 11: 310 entries; 16 overlaps; 0.0027466000174172223 s
INFO::hospinet::Iteration 12: 317 entries; 15 overlaps; 0.0024385990109294653 s
INFO::hospinet::Iteration 13: 324 entries; 14 overlaps; 0.003173984994646162 s
INFO::hospinet::Iteration 14: 330 entries; 13 overlaps; 0.003205916960723698 s
INFO::hospinet::Iteration 15: 335 entries; 12 overlaps; 0.0031857180292718112 s
INFO::hospinet::Iteration 16: 339 entries; 9 overlaps; 0.0026347980019636452 s
INFO::hospinet::Iteration 17: 243 entries; 7 overlaps; 0.0026418190100230277 s
INFO::hospinet::Iteration 18: 188 entries; 5 overlaps; 0.0023560529807582498 s
INFO::hospinet::Iteration 19: 120 entries; 4 overlaps; 0.002990390988998115 s
INFO::hospinet::Iteration 20: 122 entries; 4 overlaps; 0.003335022018291056 s
```

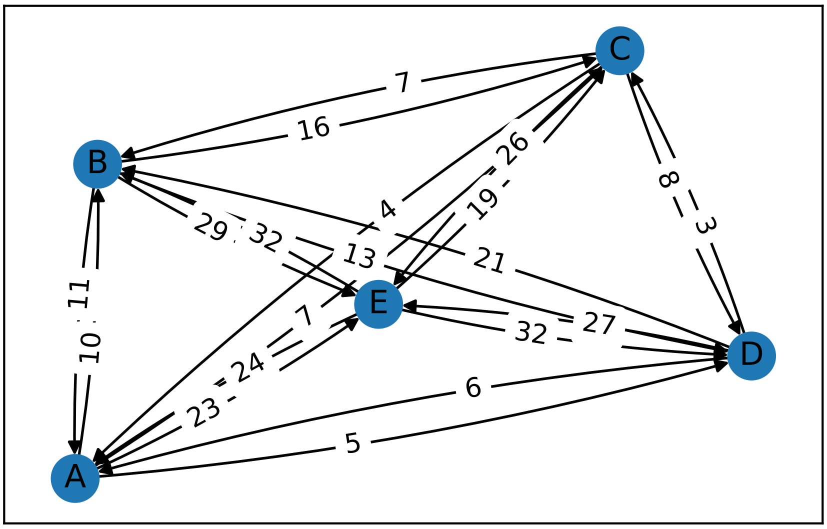
```
INFO::hospinet::Iteration 21: 123 entries; 3 overlaps; 0.0027989139780402184 s
INFO::hospinet::Iteration 22: 124 entries; 2 overlaps; 0.0029063019901514053 s
INFO::hospinet::Iteration 23: 125 entries; 2 overlaps; 0.0029028860153630376 s
INFO::hospinet::Iteration 24: 126 entries; 2 overlaps; 0.002851495984941721 s
INFO::hospinet::Iteration 25: 127 entries; 2 overlaps; 0.002019411011133343 s
INFO::hospinet::Iteration 26: 128 entries; 2 overlaps; 0.0025438889861106873 s
INFO::hospinet::Iteration 27: 129 entries; 2 overlaps; 0.002472137042786926 s
INFO::hospinet::Iteration 28: 130 entries; 2 overlaps; 0.002954298979602754 s
INFO::hospinet::Iteration 29: 131 entries; 2 overlaps; 0.002511174010578543 s
INFO::hospinet::Iteration 30: 132 entries; 2 overlaps; 0.0024062119773589075 s
INFO::hospinet::Iteration 31: 133 entries; 2 overlaps; 0.0019665530417114496 s
INFO::hospinet::Iteration 32: 134 entries; 2 overlaps; 0.0026819600025191903 s
INFO::hospinet::Iteration 33: 135 entries; 2 overlaps; 0.0019945429521612823 s
INFO::hospinet::Iteration 34: 135 entries; 1 overlaps; 0.0023768480168655515 s
INFO::hospinet::Iteration 35: 0 entries; 0 overlaps; 0.002146399987395853 s
INFO::hospinet::History of non-overlapping patient records:
INFO::hospinet::[0, 0...{16x}, 99, 57, 70, 0...{15x}, 135]
INFO::hospinet::0 overlaps remaining after iterations
```

We can then process this into a TemporalNetwork object directly using the `from_presence` class method

```
network = hospinet.TemporalNetwork.from_presence(
    numeric_db,
    discretisation=1,
)
```

This object is also a `networkx.DiGraph` object, so we can use the native plotting functionality. We project this down to static nodes first, so that we don't get the full temporal graph.

```
static_network = network.to_static()
pos = nx.spring_layout(static_network, seed=1451)
edge_labels = {
        tuple(edge): f"{attr}"
        for *edge, attr in static_network.edges(data='weight')
    }
nx.draw_networkx(static_network, pos=pos, connectionstyle='arc3,rad=0.05')
nx.draw_networkx_edge_labels(
    static_network,
    pos=pos,
    edge_labels=edge_labels,
    label_pos=0.4,
    connectionstyle='arc3,rad=0.05'
);
```



We provide some basic indexing support via methods:

```
print("Hospitals with presence at time 15: ", network.locs_at_time(15))
print("Corresponding nodes: ", network.nodes_at_time(15))
print("When hospital D is occupied: ")
pprint(network.times_for_place('D'), width=70, compact=True)
```

```
Hospitals with presence at time 15:  {'B', 'E'}
Corresponding nodes:  [('B', 15), ('E', 15)]
When hospital D is occupied:
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 38, 39, 40, 41, 42, 43,
 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 71,
 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 113,
 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127,
 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155,
 156, 157, 158, 159, 160, 175, 181, 182, 183, 184, 185, 186, 187, 188,
 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202,
 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216,
 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230,
 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244,
 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258,
 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
 278, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304,
 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318,
 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 345, 346,
 347, 348, 349, 350, 351, 352, 353, 355, 356, 357, 358, 359, 376, 377,
 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391,
 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405,
 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419,
 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433,
 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445}
```