Interactive visualization of community dynamics in temporal networks

[DTU Compute, Advanced Project in Digital Media Engineering, 2015]

Ulf Aslak Jensen
Master student, DTU
Lyngby, Denmark
s103072@student.dtu.dk

Sune Lehmann
Associate professor, DTU
Lyngby, Denmark
sljo@dtu.dk

Enys Mones
Postdoc, DTU
Lyngby, Denmark
enmo@dtu.dk

1. INTRODUCTION

Visualization of community structure in temporal networks is a relatively unexplored area of research. Efforts to understand these structures and build visualizations, have been made, but are mainly focused either on understanding dynamics through time series summary statistics, such as how the number of nodes/links or clustering coefficient develop over time[11, 12], or making simple animations where the viewer can control the time variable[8].

In this paper, we propose and demonstrate a novel design for interactive visualization of community dynamics in temporal networks. Our demonstration uses a temporal network of real-world human interactions throughout a day, recorded as pairwise bluetooth connections between personal mobile phones held by students at the Technical University of Denmark (DTU). Communities are computed using the Infomap algorithm[1], and as a result, the demonstration shows how participants move between communities in their social/academic network throughout time. The visualization is available at ulfaslak.com/Visualisation.

2. DATA

The data used for this study is formatted as links with timestamps, each corresponding to a physical meeting of distance less than 1.5 meters, between two participants of the the ongoing research project Sensible DTU[6]. Connections are logged every five minutes, asynchronously, by phones carried by the participants. As such the best time resolution one can get with this data is five minutes. For the sake of building a simple demonstration, only a single day is considered in this study, reducing the size of the data to 15639 edges and 393 nodes. The example day, a Monday, spans 24 hours from midnight to midnight. Hereinafter, the words *edge*, *link* and *connection* will be used interchangeably.

3. THEORY

This study makes use of the Infomap algorithm for multiplex networks, developed by Rosvall, et al.[1, 10, 13], to detect communities in the temporal network. A multiplex network is a network which, for each state that it assumes, is represented by a separate layer. In a temporal network, where connections are associated with time, states of such a network could e.g. be inferred by separating connections into a multitude of time bins. In the multiplex network, every *physical node* is then represented in each bin (or layer) as a *state node*. Furthermore, an important property of the multiplex network is that any set of state nodes, for a physical node, is fully connected to itself. Fig. 1 illustrates this concept.

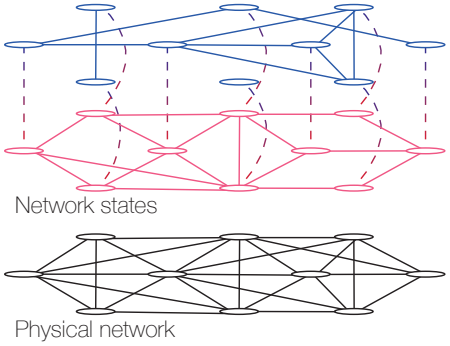


Figure 1: Sketch of a multiplex network. The full network, represented by the black graph, can be split into a number of states where state nodes are connected to each other. Edges in each network state are associated with separate entities, e.g. time span, type of interaction, etc..

Infomap utilizes the property that, for any network, when sectioning it into the communities that visually seems most appropriate, the *average per-step description length*, $L(\mathrm{M})$, of a random walker in the network is minimized[13]. In this setting, the description of each step in the walk is made using a system of codebooks which assign binary codenames to modules (i.e. communities) and their respective nodes. The lengths of these codenames are inversely proportional to the frequency at which they are used, such that rarely visited nodes and modules will get the longest, most expensive, codenames, and vice versa. The system uses is single *index codebook*, $\mathcal{Q}$, which stores module codenames, and is only used when the walker takes a step between two modules. Furthermore, it uses m *module codebooks*, $\mathcal{P}^i$, one for each module i, where m is the number of modules.

Infomap expresses the average per-step description length using The Map Equation:

$$L(\mathrm{M}) = q_{\curvearrowright} H(\mathcal{Q}) + \sum_{i=1}^{m} p^i_{\circlearrowright} H(\mathcal{P}^i) \qquad (1)$$

Recall that $L(\mathrm{M})$ is the average per-step description length of the random walker. In information theory, this is also called *entropy*. Eq. (1) is therefore a sum of two entropies,

each weighted with the rate at which they contribute. The left term is the average rate at which the walker steps from one community to another, $q_{\curvearrowright}$, multiplied by the frequency-weighted average length of codewords in the index codebook, $H(\mathcal{Q})$. The right term gives the average entropy of steps taken within communities, which becomes a sum over the probabilities $p^i_{\circlearrowright}$ of a step being taken within module i multiplied by the frequency-weighted average length of codewords in the module codebook, $H(\mathcal{P}^i)$.

The Infomap algorithm finds communities, by way of iteratively recording $L(\mathrm{M})$ for a random walker in the network with varying community configurations. To avoid situations where the walker gets "stuck" in parts of the network, for each step it has a small probability to teleport to a random node, and as such it becomes a *random surfer*[13]. This theoretical framework also works for multiplex networks, because sets of state nodes are fully connected[10], allowing the surfer access to all parts of the network. Connections between state nodes are then represented using edges of weight corresponding to the inter-layer *relax rates*, i.e. the layer-transition probabilities $p^{i,j}_{\updownarrow}$, of the surfer. In this study $p^{i,j}_{\updownarrow}$ is set to obey a decay function:

$$p^{i,j}_{\updownarrow} = e^{-\Delta t \frac{\log 2}{\tau}}, \tag{2}$$

where Δt is time difference between layers i and j, and τ is half life. To get an intuition of what τ means, one can imagine that when it goes to infinity there will be unit relax rate between all layers, and as such the natural time ordering of the layers will vanish.

4. DESIGN AND PROTOTYPING

Inventing a design for the visualization was a collaborative and highly iterative process between the authors and other members of the Sensible DTU group. The goal was to make a visualization that could explain the transfer of members between communities at different times. Early inspiration was found in the 2012 New York Times visualization 'Over the Decades, How States Have Shifted' created by Mike Bostock[9] (see Appendix A).

4.1 Current design iterations

Fig. 2 shows all iterations of the design. The first iteration is illustrated by the sketch in Fig. 2a. By visual comparison it highly resembles Bostock's visualization in that states are now communities (rectangles) that, furthermore, contain members (little circles). The idea is that one can hover over a community to highlight it and all its members in all time layers, illustrated with a blue color, hence showing how members come and go. The problem, however, is that the data in this study contains significantly more time steps than that of Bostock's. Therefore, in the next iteration, shown in Fig. 2b, each person is visualized as a trace of squares colored by community in a large matrix of width and height corresponding to the total number of participants and number of time layers respectively. To prove the feasibility of such a design, a first implementation was made using D3.js as illustrated in Fig. 2c. While this prototype is hard to interpret by the uninitiated, it actually provided a solid stepping stone in the design process. As can be seen from the figure, time steps of the communities, distinguished by color, align poorly because some emerge and vanish in each time step, and as such their horizontal placement varies a lot. This is, however, easy to correct for, by simply designating horizontal coordinates to each rectangle respecting the location of their communal predecessors. The implementation of this led to what is shown in Fig. 2d. Here, a more fitting color scheme is used, and the height of the rectangles is reduced to correspond to five minute layers. Considering the differences between Fig. 2c and 2d an obvious one, besides the alignment of rectangles, is that dyads (communities of two members) have been filtered out. While they are interesting in their own rights, for the purpose of this visualization they are found to create too much visual noise. Finally a graph, showing the state of the network in a given time layer, is added, allowing the user to better inspect specific network states and changes between them. A small text box is also included, allowing the user to read a short 'About' introduction to the visualization, and press the 'Guide' tab, to read about features and hot-keys.

Importantly, it should be noted that the design includes interaction features, which are not illustrated in Fig. 2d, such as when hovering over community strips, other communities turn opaque proportionally to how many members they share with the one in focus. Other interactions such as clicking communities to see basic information, changing time steps to update the graph using key arrows and shift-hold for slow layer-transition is also included in the design.

4.1.1 Evaluation

As mentioned, the current design was developed in a collaborative process. As such, each iteration was subject to long discussions between the authors, and was evaluated in a *quick and dirty* fashion, using fellow researchers and students as test persons. Commonly, critique was raised around the design of the strip view, stating that it was unclear whether the horizontal arrangement of communities had any significance. Reversely, much positive feedback was given in relation to the format of the visualization, where many stated that the interactive features made it "fun", and "feel alive".

4.2 Future design iterations

The design shown in Fig. 2d is implemented and live at ulfaslak.com/Visualisation, but is, at this point, still considered an iteration in an ongoing development process. Future versions of the visualization will include a number of features that help better explain the transfer of members between communities. Below is given a list of desired features:

- To make the strip view more visually appealing, some further "cleaning up" and development must be done. This includes, first of all, allowing the user to filter communities by member size and to change the time scale to inspect multiple days. Furthermore, the horizontal ordering of strips should be done "in a natural fashion" using D3's force layout adding attractive force between strips in proportion to how similar they are. This would hopefully remove any confusion about the significance of horizontal ordering.
- The user should be able to explore individual participants in the same way that she is able to explore

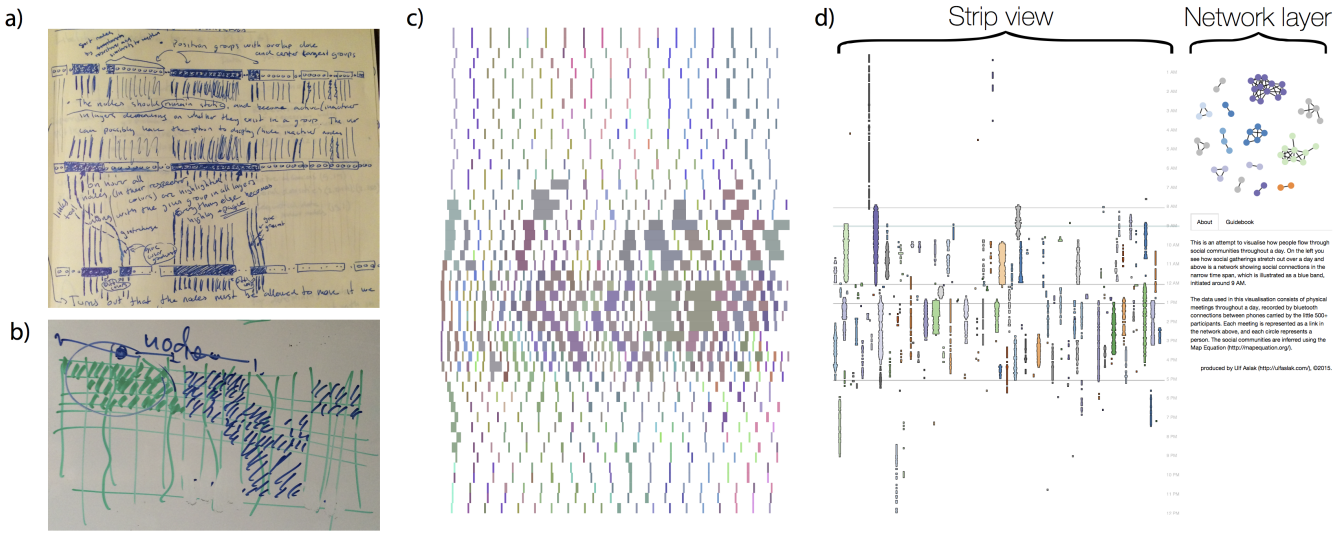


Figure 2: The most important steps in the design process. ***a)*** **Starting point of the design process. This design carries much resemblance to Bostock's (Appendix A), where instead of states in each time layer, it uses groups that enclose circles representing people, who move between groups in layers represented with curves.** ***b)*** **It was discovered that the design needed to represent far more time layers than Bostock's, and as such, a matrix-like design was imagined, where rows and columns represent time layers and people, respectively.** ***c)*** **First implemented prototype. In design, it is largely equivalent to** ***b***, **where rectangles, representing communities in time layers, have width and height corresponding to number of members and time span, respectively.** ***d)*** **Latest implemented prototype. The strip view is similar to** ***c***, **except groups of two are removed, time layer spans are reduced to five minutes, and community layers are horizontally ordered, such that for a given community, the rectangles that represent it in every layer will have the same center-point on the horizontal axis, unless the community is discontinuous. See Appendix B for larger figures.**

communities in the implemented demonstration. This should include options to explore statistics such as which communities and people the individual is most associated with, how many interactions she makes with groups and individuals, when she is most socially active, etc.

- When the user focuses on a community which is present in the time layer that the graph considers, Bezier curves should extend from this layer of the community and connect to other layers of other communities in the next or previous time steps, to show how members flow into an out of the specific community layer. This idea is a legacy from the imagined prototype (see Fig. 2a).

5. IMPLEMENTATION

This section describes all the steps that turn raw network data into an interactive visualization. The steps can be sectioned into two main stages: Community detection, and visualization. In the following, it is understood that, unless otherwise stated, the implementation uses Python for data handling and formatting, and Javascript (D3.js) for visualization. Note that it is considered out of context to describe all but the most general parts of the implementation pipeline. Interested readers are, however, welcome to explore in-depth aspects of the architecture by reading the code which is largely self-contained and open-sourced on Github[5].

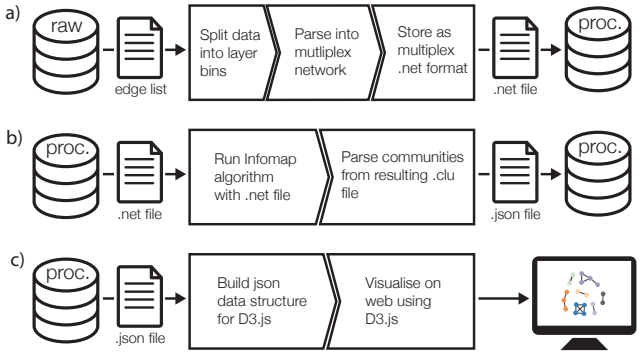


Figure 3: High-level illustration of the pipeline for visualizing a temporal social network with communities inferred by Infomap. ***a*** **and** ***b*** **summarize preprocessing and community detection steps, respectively.** ***c*** **illustrates visualization steps.**

The implementation pipeline is illustrated, in very high level terms, by Fig. 3. *a* shows how raw data is converted into a processed file that represents the chosen subset as a multiplex network. *b* shows how this file is used as input to the Infomap algorithm which yields the communities that are then stored in another file. *c* illustrates that this file is used to build the data structure which acts as motor for the D3 visualization.

5.1 Community detection

This stage can be partitioned into data pre-processing (Fig. 3a) and computation (Fig. 3b). These parts of described in the following two subsections.

5.1.1 Data pre-processing

The minimum time resolution of five minutes is chosen for binning raw data by layer. This results in 288 layer-bins (minutes per day divided by 5 minutes), each storing connections within their respective five minute bins.

The binned data is parsed into a multiplex network (Sec. 3 describes multiplex networks). In representation of the binned data, *intra*-layer edge weights are set to integer values corresponding to the number of times each edge exists in each time layer. *Inter*-layer edges connects each set of states for each physical node, with edge weights corresponding to relax rate, given by Eq. (2). Here, τ is set to be equal to the width of the layer-bins that the network is sectioned into (300 seconds), such that $p_{\updownarrow}^{1,2} = 0.5$, $p_{\updownarrow}^{1,3} = 0.25$ and so on. The multiplex network is reformatted and stored in the file format *multiplex Pajek format* with `.net` file extension[1].

5.1.2 Computation

With a valid `.net` formatted representation of the multilayer network, the Infomap algorithm is run from the command-line using a C++ implementation developed by Rosvall et al.[1]. It computes the communities of the network, and outputs the result in `.clu` format, from which communities are parsed, using Python, into a `json` datastructure, that stores community-member pairs for each layer.

5.2 Visualization

The Javascript library D3.js is used for implementation of the design illustrated in Fig. 2d. D3 allows for fast data-driven manipulation of the Document Object Model (DOM) of an html web page, which proves extremely powerful when making interactive visualizations. Below is given a description of the implementation at the time of writing. The reader should be aware that since this is still at project in development, features may be added in the future.

5.2.1 Data structure

D3.js is a fast data-driven visualization framework. One of the reasons that it is so fast is that it is meant to do very little data manipulation on the fly, but instead just look up values in a data structure, and as such a great deal of data structure engineering is required to create something that runs fast and feels responsive. In this implementation, the results from the Infomap algorithm are processed, using Python (see Fig. 3c), into a big data structure that stores geometric data for drawing the community strips, similarity values between all community pairs, clustered networks in all layers for making graphs as well as meta data specifying various global parameters. To inspect the data structure, users may go to the visualization, open an inspector in the browser, go to the Javascript console and type 'dataset', to print the data structure variable.

5.2.2 Current implementation

In the following, the most important aspects of the current implementation of the visualization is presented. If readers are interested in learning more about the project, they may view the code on Github[5]. The current implementation of our proposed method for visualization of temporal networks is served at ulfaslak.com/Visualisation.

The visualization is implemented as an HTML web page, which uses *Bootstrap* for styling[7], as well as custom styles defined in a CSS file, *D3* for DOM manipulation[3], and some *jQuery* for key-event listening[4]. All components rendered to the canvas by D3 are Scalable Vector Graphics (svg) objects.

The design has two main components: the *strip view*, and the *network layer* (see Fig. 2d). Each of these components are uncoupled in the implementation, but have events that when triggered, read from the same global parameters, which control whether either of the `shift` or `alt` keys are pressed, which community the cursor is currently hovering over, which community was most recently clicked and which time step the visualization is currently in. The design contains a number of features which are triggered by either of the keys `up-arrow`, `down-arrow`, `esc`, `alt` and `shift`.

The strip view consists of individual polygons. Each of these belong to a specific community, and is described as corner points listed in clockwise order. A community may be represented by multiple polygons, which is the case for communities that vanish at some points and reemerge at others. The network layer, is rendered using the D3 force layout[2]. The behavior of the network is controlled by a number of force parameters such as *gravity*, *link strength*, and *charge* which control attraction, repulsion, friction, temperature, etc. The slow motion effect is achieved by initiating the graph with a low temperature (i.e. energy), and a low friction parameter (corresponding to high friction in the physical world), and then increasing these values slowly, i.e. over a large number of rendering iterations. Furthermore, new nodes are initiated in the geometrical center of their neighbors, which makes the state transitions steady, since new nodes would have otherwise been initiated randomly, and possibly far away from their neighbors, to which they are attracted with great force, thus enabling undesired sling-shot like force-interactions.

6. DISCUSSION AND CONCLUSION

To be able to inspect a temporal network, state-by-state, using the arrow keys, is not a new approach, but combining this with a complete overview of all communities in a day is. Our proposed visualization design, therefore, has clear advantages over similar efforts, in that it gives the user everything she needs in order to perform close inspection of specific time intervals in a network, as well as the dynamics of how it changes over time; without losing track of the big picture. A further advantage of this approach to temporal network visualization is that the interactive format of the visualization, enabled by the D3 library, is "fun" and "feels alive", as noted by several test persons.

In conclusion, the current implementation of our proposed design for interactive visualization of community dynamics in temporal networks, has proven feasible, and well worth further development.

7. REFERENCES

[1] D. edler and m. rosvall, the mapequation software package, available online at. http://www.mapequation.org. Accessed: 2015-12-05.

[2] D3 force layout docs on github. https://github.com/mbostock/d3/wiki/Force-Layout. Accessed: 2015-12-06.

[3] Official website of data-driven-documents (d3). http://d3js.org/.

[4] Official website of jquery javascript library. https://jquery.com/.

[5] Project code on github. https://github.com/ulfaslak/InfomapSensibleDTU.

[6] Sensible dtu research project. https://www.sensible.dtu.dk/. Accessed: 2015-12-05.

[7] Twitter bootstrap official website. http://getbootstrap.com/.

[8] J.-w. Ahn, M. Taieb-Maimon, A. Sopan, C. Plaisant, and B. Shneiderman. Temporal visualization of social network dynamics: Prototypes for nation of neighbors. *Lecture Notes in Computer Science*, 6589:309–316, 2011.

[9] M. Bostock. Over the decades, how states have shifted. http://www.nytimes.com/, 2012. The New York Times.

[10] M. De Domenico, A. Lancichinetti, A. Arenas, and M. Rosvall. Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems. *Physical Review X*, 5(1):011027, 2015.

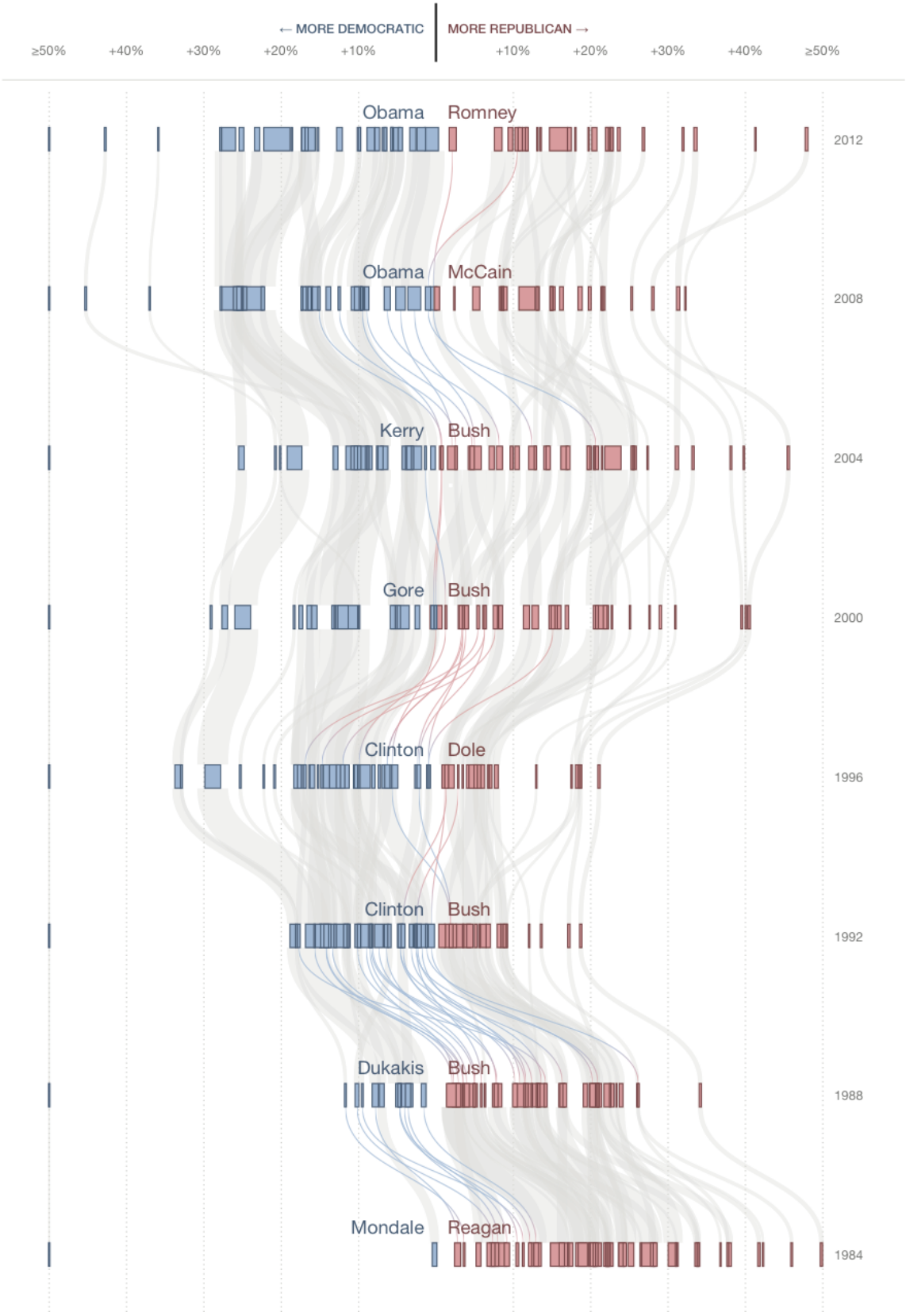
[11] L. Gauvin, A. Panisson, and C. Cattuto. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *PLoS ONE*, 9(1):1, 2014.

[12] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science, Science, Science, Science Magazine, Science (wash D C), Science Express, Sciencexpress*, 328(5980):876–878, 2010.

[13] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. 2012.

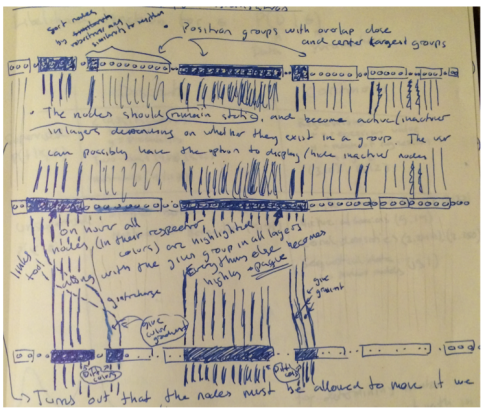
APPENDIX

A. OVER THE DECADES, HOW STATES HAVE SHIFTED



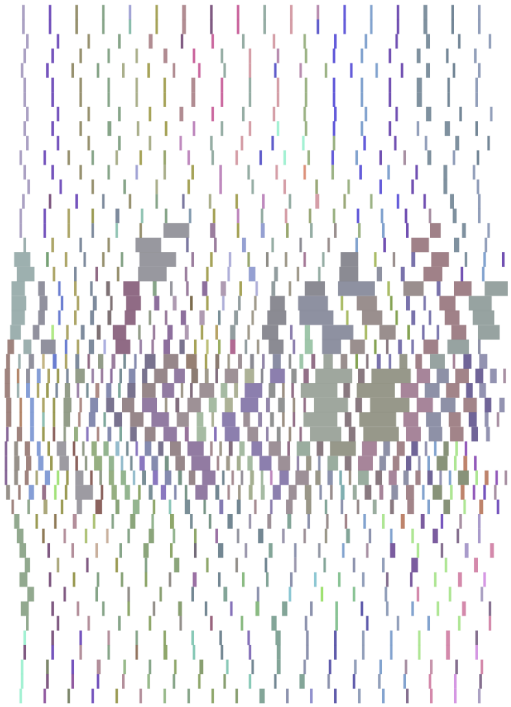
B. DESIGN PROTOTYPES, LARGE FIGURES

a)

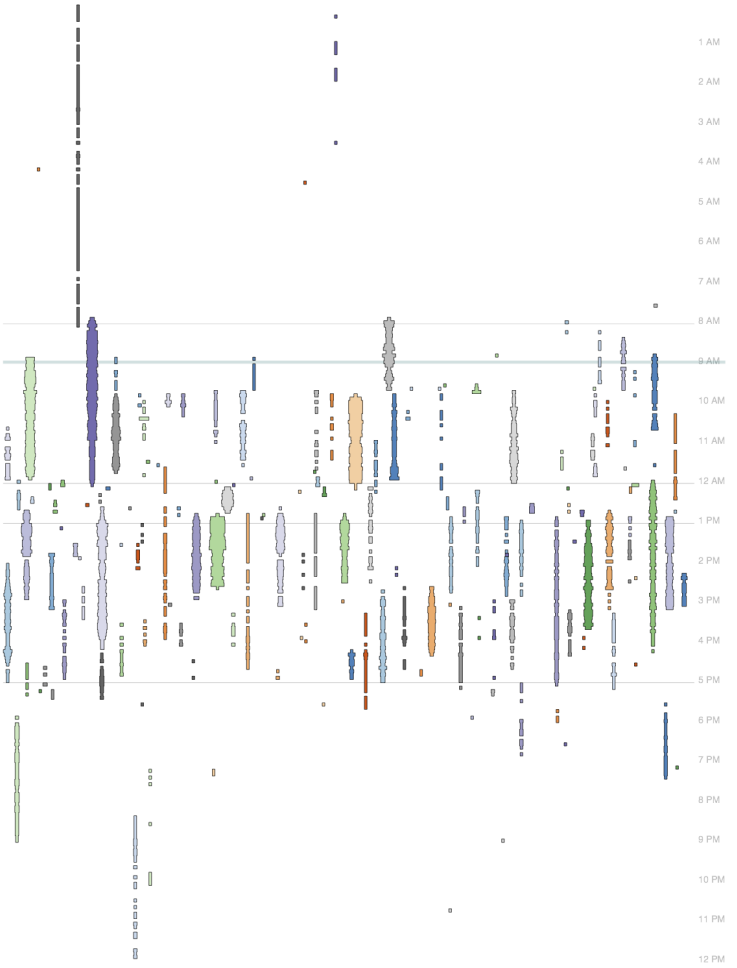


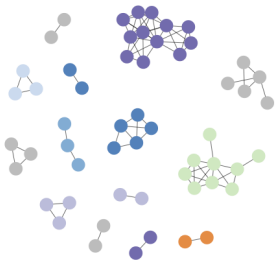
b)

c)



d)





About Guidebook

This is an attempt to visualise how people flow through social communities throughout a day. On the left you see how social gatherings stretch out over a day and above is a network showing social connections in the narrow time span, which is illustrated as a blue band, initiated around 9 AM.

The data used in this visualisation consists of physical meetings throughout a day, recorded by bluetooth connections between phones carried by the little 500+ participants. Each meeting is represented as a link in the network above, and each circle represents a person. The social communities are inferred using the Map Equation (http://mapequation.org/).

produced by Ulf Aslak (http://ulfaslak.com/), ©2015.