Characterisation of Complex Networks from their Topologies

A thesis submitted to The University of Manchester for the degree of
Doctor of Philosophy
in the Faculty of Science and Engineering

2019

Oliver A Blanthorn

School of Engineering
Department of Computer Science

Contents

Word count: 36,000

List of Figures

List of Tables

List of Algorithms

Abstract

Complex networks appear everywhere in the world. They describe everything from the Internet and transport networks to neuronal networks and social networks. Characterising these networks—that is, describing them quantitatively—is vitally important to understanding them, comparing them and tracking their temporal evolution. The connection patterns, or topologies, of networks are often very different across different networks and sometimes even between different parts of a single network, or over time in temporal networks. This work focuses particularly on these connection patterns and how they can characterise networks at multiple scales: we use the topology to characterise nodes, communities and entire networks.

On the subject of node characterisation, we demonstrate the merits of using colour in graphlet profiles when characterising ego-networks, using this technique to discover that, in scientific co-authorship networks, the proportion of graphlets of a specific kind centred on an author early in their career is correlated with that author's final academic career length. These graphlets which are correlated with career length are 4-stars; induced subgraphs containing four nodes where one node is the author and the other three nodes only have a link to the author.

For whole-network characterisation, we use matrix decomposition of data matrices formed from topological properties such as degree-mixing matrices and graphlet profiles to clearly distinguish between real networks and network models and track changes in networks over time. We introduce the term 'piecewise static networks' to describe the application of standard static network analysis to temporal networks.

Finally, on community characterisation, we introduce a new way of assessing the goodness-of-fit for tensor decompositions of temporal networks by applying adjusted mutual information and employ this technique to throw more light on software dependency networks—where we use it to highlight the uniqueness of the R language's software package ecosystem due to R's 'rolling-release cycle' policy—and semantic networks—where we start to study the evolution of the English language over the past thousand years.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

"Muad'Dib learned rapidly because his first training was in how to learn. And the first lesson of all was the basic trust that he could learn. It's shocking to find how many people do not believe they can learn, and how many more believe learning to be difficult. Muad'Dib knew that every experience carries its lesson."

— Frank Herbert, *Dune*

This work bears only my name but it is the product of everyone who helped me along the way to reach this point. I shall do a great disservice to most of these people as I only have room to thank a few.

First and foremost, I must acknowledge that I owe a great debt to my supervisor, Dr Eva Navarro-López. I'm certain this would be a far lesser work without her unwavering support and strength. I don't know how she does it.

I would also like to thank my family. Mum, thanks for always pushing me to work hard and ensuring that I always had the opportunity to have a great education. Thanks for looking after the koi pond and the watercress. Tara, keep chasing whatever makes you happy, whether that's ecology, field sites, or your daughter Eilah! I'm equal parts sorry and glad that the experimental work for thesis was a little less involved than your chasing primates across the Congo. Eilah, I can't wait to see what you do. Dad: this thesis is really for you. Now that you can't read this I think it is safe for me to admit that, yes, being forced to play 'Mavis Beacon Teaches Typing' on MS-DOS before I was allowed a quota of fun video-games was perhaps a little useful and that your abject refusal to give me a straight answer to any question probably instilled the curiosity (and, if we're honest, intransigency) that meant I was able to complete this thesis.

Finally, Harri: I am perpetually baffled why you put up with this curmudgeon, but I am glad that you have. Thanks for all of your support, especially in these last few months while I produced this thesis and gradually morphed into a troglodyte.

I'm grateful that this work was supported as part of an Engineering and Physical Sciences Research Council (EPSRC) grant, project reference EP/I028099/1.

Availability of data and material

The source for this thesis is available at github.com/bovine3dom/thesis. I believe this is a fairly comprehensive list of all the code used in this work. If an interested reader feels we have omitted any crucial routines, please file an issue on the repository for the thesis and we will endeavour to produce it.

Chapter 3

The programmes described in this chapter are available from:

- https://github.com/bovine3dom/ColoredGraphs.jl,
- https://github.com/bovine3dom/Nauty.jl,
- https://github.com/bovine3dom/Graphlets.jl.

Chapter 4

The co-authorship network used in this chapter is available from:

- https://aminer.org/dynamic_coauthor.

Chapter 5

The software used in this chapter is available from:

- https://github.com/bovine3dom/GraphPCA.jl.

Chapter 6

The data used in this research are available from:

- https://www.nceas.ucsb.edu/interactionweb/html/thomps_towns.html for the ecological networks and
- https://sites.google.com/site/ucinetsoftware/datasets/covert-networks/suffragettes is the Suffragette network.

Chapter 8

The data used in this research are available from:

- https://doi.org/10.5281/zenodo.2536573, and our code is
- https://doi.org/10.5281/zenodo.2610369, and
- https://github.com/bovine3dom/evolution-of-software-communities.

Chapter 9

Linguistic evolution: the proprietary data used in this research are available with agreement from the OED under an academic licence from developer.oxforddictionaries.com/our-data/. Please note that the research necessarily requires the data to be cached which is only permissible under special agreement. The code written to crawl and analyse the data is available at https://github.com/bovine3dom/evolution-of-english.

Nomenclature

A	adjacency matrix
$a_{i \leftarrow j}$	entry of the adjacency matrix corresponding to the link from j to i
ALS	alternating least squares
AMI	adjusted mutual information
APG-TF	alternating proximal gradient tensor factorisation
β_{reduced}	a coefficient of a variable in a linear regression multiplied by the standard deviation of that variable
C	global clustering coefficient
CANDECOMP	canonical decomposition
C_i	local clustering coefficient of the node i
CP	CANDECOMP/PARAFAC
CRAN	Comprehensive R Archive Network
δ_{ij}	Kronecker delta
D	degree correlation matrix
D	network diameter
DOM	document object model
$\mathcal{E}$	a set of edges
EPSRC	Engineering and Physical Sciences Research Council
FOSS	free and open-source software
$\mathcal{G}$	a graph

GCC	global clustering coefficient
HTTP	hypertext transfer protocol
L	Laplacian matrix
LIDA	Leeds Institute for Data Analytics
NNCP	non-negative CANDECOMP/PARAFAC
NRSS	normalised residual sum of squares
OED	Oxford English Dictionary
PARAFAC	parallel factor analysis
PCA	principal component analysis
PyPI	Python Package Index
R	the number of components in a tensor decomposition
R	the R programming language
SVD	singular value decomposition
$\boldsymbol{t}$	a transformed feature vector, e.g. projected onto its principal components
$\mathcal{V}$	a set of vertices
$\boldsymbol{w}_k$	the k^{th} principal component
$\boldsymbol{x}$	a feature vector corresponding to an example
X	a data matrix whose rows are feature vectors

Part I

Introduction

Chapter 1

Introduction

"I've suffered for my music,
and now it's your turn"

— Neil Innes, *A protest song*

Networks are all around us. They characterise the Internet, social networks, patterns of brain activity, transport systems and financial networks that define our lives. Humanity often meddles with these networks, for example in transportation networks where we add traffic lights to control traffic, financial networks where politicians suggest transaction taxes and social networks where parents try to send their children to a good school in the hope that they will be influenced in a good way by friends. But we only have a shallow understanding of how these networks work. Surprising (and controversial!) results are still published in this very young field (Broido & Clauset, 2019). It is our hope that the new tools this thesis provides will be used to further this understanding.

This thesis is particularly concerned with ways of characterising networks and parts of networks, i.e. developing ways of comparing and distinguishing networks or parts of networks, from looking at their topologies, or wiring patterns. This topological approach is attractive as the wiring patterns of real-world networks often change at a much slower rate than the dynamics on those networks; for example, the amount of road-traffic changes with the time of day but new roads take years to build (Gross & Sayama, 2009). The connections between parts of a system are often fairly easy to observe compared to the equations governing the dynamics—it is far easier to observe where a road leads than it is to understand exactly what caused a driver to pick that road, for example. Data on wiring patterns are therefore often more readily available than they are for node dynamics. We take a multi-scale approach where we consider characterisation of complex networks at three scales: at the scale of nodes, communities and whole networks.

This first part serves as an introduction to the thesis. The first chapter presents a brief in-

troduction to network science and the characterisation of networks; the next chapter is a more rigorous and mathematically-involved primer on the background of this thesis and the state-of-the-art of characterisation in network science and related techniques.

1.1 A short history of graph theory and network science

Network science has a long and winding history. We shall summarise some of it here in the hope of explaining more about what network science is and why we think they are subjects of such importance.

1.1.1 The seven bridges of Königsberg

Königsberg, now known as Kaliningrad, is a beautiful Baltic city. There are two islands in the centre of the city (see figure 1.1) which historically had seven bridges connecting them. The question whether it was possible to walk across each bridge exactly once required new techniques. Euler (1741) proved the walk was impossible, laying the foundations for graph theory along the way. He noticed that the topology, what we would now call a multigraph, was key to solving the problem: once you represent each bank or island as a node, with the bridges between them as links, it follows that, for a node which one is not currently on, in order to cross each link exactly once and not become stuck on the node, there must be an even number of links going to it, which we call an even degree (nodes at which one starts or ends the path must have odd degrees for the opposite reason). However, all of the nodes in the network have odd degree, as can be seen in figure 1.1, and so the walk was not possible.

Two bridges were lost in the bombing of Königsberg in World War II, along with most of the historic city centre, a tragedy compounded about a year later by the Soviet expulsion of the native population. However, the destruction of the two bridges does at least mean that one can enjoy walks crossing each bridge once, as the two banks of the river now have even degrees and the islands have odd degrees.

1.1.2 Random graphs

In the interests of brevity, we now skip over 200 years of fascinating results from graph theory to 1959, when Erdős & Rényi first proposed a model of random graphs. Their model, explored in more depth in section 2.9.2, was the first to be used to rigorously explore what it meant for a graph to have randomness. They showed that random graphs are very likely to be totally connected (i.e. a route is possible between every pair of nodes) with a relatively low number of edges and that the average distance between randomly chosen nodes grows very slowly with the total number of nodes. This decoupling of the distance between nodes and the total number of nodes is a key feature of real world networks and allows information to cross networks very quickly.

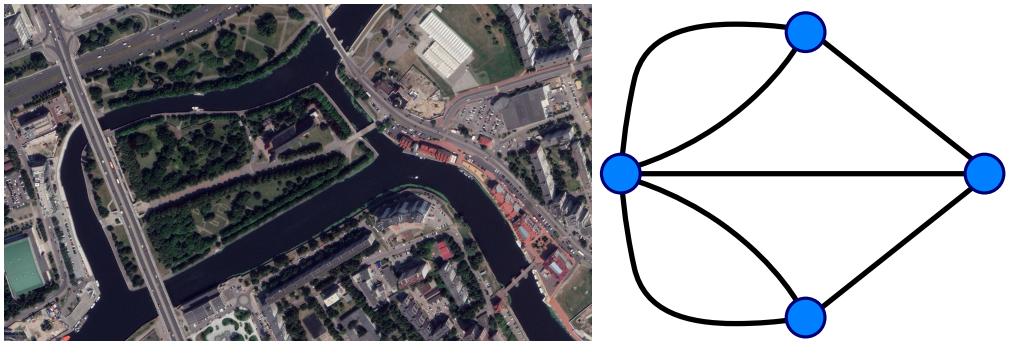

Figure 1.1 – Left: A satellite image of the centre of modern day Königsberg, now Kaliningrad. Five of the seven bridges from Euler's time remain. The last bastions of the two demolished bridges are just visible about one-third of the way across the image from the left; one has been turned into a berth for a boat. Map data from Google and Maxar Technologies. Right: Königsberg's seven bridges represented as a network (Chris Martin, 2006)

1.1.3 Small-world networks

These random graphs captured an important property of real-world networks: the average distance, or path length, between nodes is often short. This has been known intuitively for a very long time: 'it's a small world' has been a common refrain when one finds that a stranger has a mutual friend for at least a hundred years. This well-known phenomenon was first proved by Milgram (1967) in a famous experiment where he gave addressed letters out at random to people who were instructed to either send the letter to the addressee directly if they knew them personally, or forward the letter to the person they thought was most likely to know the addressee. Milgram's results are shown in figure 1.2: the modal number of intermediaries used was six, so the modal path length was seven.

There is a problem with the random graphs introduced by Erdős & Rényi, however. It is unlikely that linked nodes share neighbours, a property often found in real networks; in the jargon of the field, they have a very low 'clustering coefficient', whereas real networks often have a high clustering coefficient. Granovetter (1977) explored this issue and argued that social networks should be considered as cliques interacting across weak (e.g. long distance) links. Watts & Strogatz (1998) developed a model based on these ideas: a nearest neighbour ring network with weak-ties thrown in at random. They showed that a network needs very few of these weak ties before the path length loses direct proportionality with the number of nodes; this slow growth in path length is the small-world effect.

1.1.4 Scale-free networks

It was soon found that many real-world networks had degree distributions that looked nothing like the small-world or random graphs; Barabási & Albert (1999) found that real-world degree distributions often had 'fat tails' with a few nodes having the lion's share of the links. They

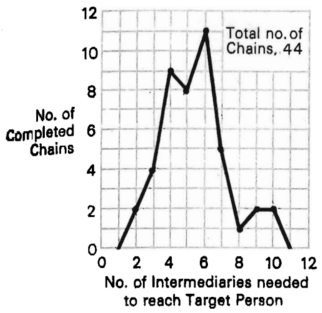

Figure 1.2 – The path-length distribution from an experiment on the social network of the United States (Milgram, 1967)

proposed a model of preferential attachment—that new nodes added to a network tend to link to nodes with high degree—which resulted in networks with degree distributions which followed a power-law, as they had observed in real networks. There has been some recent controversy over suggestions that these are perhaps less common than was previously thought if one uses a very strict definition of what it means to follow a power-law (Broido & Clauset, 2019). This is not a debate we seek to enter.

1.1.5 Motifs and graphlets

Milo *et al.* (2002) formalised the intuitive idea that the local structures of networks were essential to their function. They devised a method of calculating how unusual a local structure was—i.e. how unlikely it was to appear by chance—and used this to extract small graphs characteristic of graphs from several categories, including the Internet, food webs and gene regulatory networks.

1.2 Characterisation

Characterisation is key to many areas of life. When we purchase a car, we want to know what colour it is, how much it costs, how fast it goes, how much fuel it consumes and whether we can fit the shopping in the boot; when purchasing a house we want to know how often it floods, how much the asking price is and how close it is to a school. We use these characteristics to categorise things: if it goes fast and I can't fit my shopping in the boot, it's a sports car; if it was alive in the cretaceous period and looks like a giant lizard, I'm calling it a dinosaur. Characterisation's innate attraction is evident by the success of the 'Top Trumps' series of card games where one holds fact-sheets for examples of things within a certain field, be it sports cars, dinosaurs or Star Wars. This desire to categorise and compare is part of being human.

Perhaps the most famous and successful example of using characterisation to more deeply understand part of our universe are Hertzsprung-Russell diagrams (Russell, 1914), an example

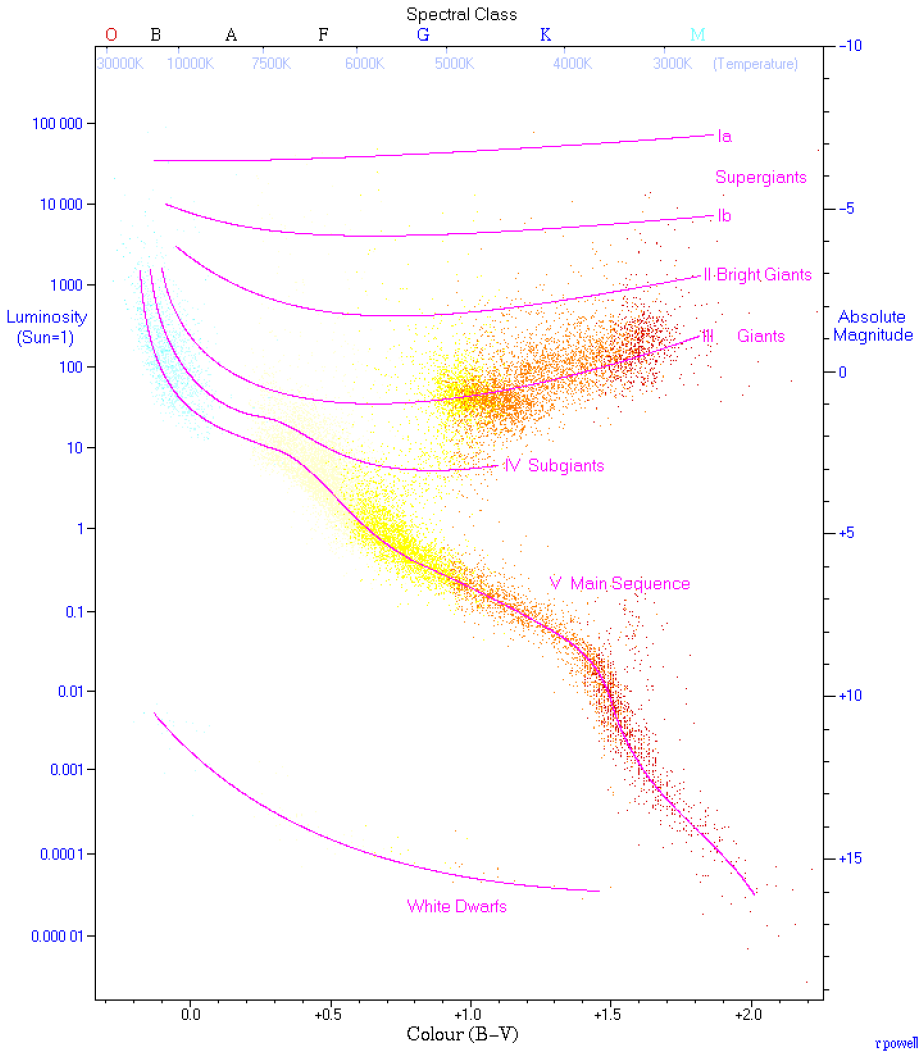

Figure 1.3 – A Hertzsprung-Russell diagram (adapted from Richard Powell, 2007) showing stars with their energy output plotted against their surface temperature. Approximate trajectories grouping stars with the same life cycles have been drawn on in purple

of which is shown in figure 1.3. By simply plotting the luminosity (total power radiated) against the surface temperature or colour (which are directly related) for observed stars, a striking pattern emerges and, with a little interpretation and background knowledge, the various paths a star may take throughout its life become clear. All stars start following the 'main sequence' in the top left, becoming redder as they age. Massive stars become 'giants' while smaller stars stay on the main sequence. Our ultimate goal is to make advances like this possible in the field of network science.

1.3 Contributions

Throughout the course of this thesis, we present numerous novel contributions to the field. We shall enumerate them here:

- Software adapting an existing graphlet enumeration routine for use on coloured graphs is described in chapter 3.

- On academic co-authorship networks, collaboration with authors who do not directly collaborate with each other was found to increase the number of years in which an author published papers. This finding had a robust degree of statistical significance. This is detailed in chapter 4.

- The use of coloured graphlet profiles compared to colour-blind graphlet profiles when characterising nodes from their ego-networks was found to provide more detailed results with a higher degree of statistical significance. This is shown in chapter 4.

- Networks can be distinguished using principal component analysis on data matrices formed from simple network representations and metrics. This is given in chapter 6.

- Adjusted mutual information can be used to aid in choosing the number of communities in which to cluster a network with tensor decomposition. We present this in chapters 7.1 and 8.

- Software ecosystem evolution: we can see how distinct different software dependency ecosystems look, shaped, we theorise, by the decisions made by the package-management designers. This is given in chapter 8.

- English language evolution: we present preliminary results on how we might apply tensor decompositions to a temporal linguistic network. In our preliminary results presented in chapter 9, we see changes in the language over time. We explore how such an analysis could be extended.

1.4 Publications and talks

- Coloured graphlet profiles as a predictor of career length in scientific co-authorship networks (talk and paper). In: 4th World Conference on Complex Systems: Emergence, self-organization, nonlinear dynamics and complexity. Institute of Electrical and Electronics Engineers (Blanthorn & Navarro-López, 2019a).

  – This paper has been invited for further publication after expansion in a special issue of *Journal of Systems Science and Complexity* in October 2019.

- Evolution of communities of software: using tensor decompositions to compare software ecosystems. Accepted 23 August 2019, awaiting publication. Applied Network Science (Blanthorn *et al.*, 2019).

- Tracking the evolution of the English language as a network with temporal clustering (unpublished): presented at Conference on Complex Systems, Nanyang Technological University, Singapore (Blanthorn & Navarro-López, 2019b).

1.5 Outline of the thesis

In the rest of this part, the necessary background and the state of the art is given. The remainder of the thesis is split into three parts following our three main areas of contribution. In part II, we characterise nodes by their local network topology through the use of graphlet profiles. In part III we characterise whole networks by creating high-dimensional representations of them and then collapsing these into low-dimensional representations useful for comparison. In part IV we characterise whole temporal networks by the evolution of their communities.

Bibliography

Barabási, Albert-László, & Albert, Réka. 1999. Emergence of scaling in random networks. *science*, **286**(5439), 509–512.

Blanthorn, Oliver, & Navarro-López, Eva. 2019a. Coloured graphlet profiles as a predictor of career length in scientific co-authorship networks. *In: 4th World Conference on Complex Systems: Emergence, self-organization, nonlinear dynamics and complexity*. Institute of Electrical and Electronics Engineers.

Blanthorn, Oliver, & Navarro-López, Eva. 2019b. *Tracking the evolution of the English language as a network with temporal clustering*. Unpublished: to be presented at Conference on Complex Systems, Nanyang Technological University, Singapore.

Blanthorn, Oliver, Caine, Colin M, & Navarro-López, Eva. 2019. *Evolution of communities of software: using tensor decompositions to compare software ecosystems*. Unpublished: accepted by Applied Network Science, Springer, 23 August 2019.

Broido, Anna D, & Clauset, Aaron. 2019. Scale-free networks are rare. *Nature communications*, **10**(1), 1017.

Erdős, Paul, & Rényi, Alfréd. 1959. On random graphs I. *Publicationes mathematicae (debrecen)*, **6**(1959), 290–297.

Euler, Leonhard. 1741. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum petropolitanae*, 128–140.

Granovetter, Mark S. 1977. The strength of weak ties. *Pages 347–367 of: Social networks*. Elsevier.

Gross, Thilo, & Sayama, Hiroki. 2009. Adaptive Networks. *Pages 1–8 of:* Gross, Thilo, & Sayama, Hiroki (eds), *Adaptive Networks*. Understanding Complex Systems. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-01284-6_1.

Milgram, Stanley. 1967. The small world problem. *Psychology today*, **2**(1), 60–67.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, **298**(5594), 824–827.

Russell, Henry Norris. 1914. Relations between the spectra and other characteristics of the stars. *Popular astronomy*, **22**(May), 275–294.

Watts, Duncan J., & Strogatz, Steven H. 1998. Collective dynamics of small-world networks. *Nature*, **393**(6684), 440–442.

Chapter 2

Background

"The thing that hath been, it is that which shall be;
and that which is done is that which shall be done:
and there is no new thing under the sun."
— Ecclesiastes 1:9, *King James Bible*

Graph theory had a long history before it spawned modern network science at the turn of the century. We do not have the luxury of enumerating all of its achievements or anything close to all of its or network science's achievements in this thesis. Instead, we aim only to furnish the reader with the mathematical background and advances required to understand this work and the reasons it takes the directions it does.

The reader is advised to skim through this section on first reading. It may be useful to revisit this chapter with the aid of the index at the rear of this book when unfamiliar terms appear in the later chapters. This chapter comprises quite a few sections: section 2.1 introduces some of the basic mathematical foundations needed to understand this thesis; section 2.2 introduces mathematical descriptions of networks and graphs; sections 2.3 and 2.4 explore matrix and tensor decomposition which we employ in parts III and IV; sections 2.5, 2.6 and 2.7 describe some current methods for characterising networks at multiple scales and are probably the most important sections in this chapter as they form the basis of the work we perform in the rest of the thesis; section 2.9 uses tools introduced in the preceding sections to explain what makes complex networks complex and gives some examples of complex networks; section 2.10 provides a brief explanation of the statistical methods we employ in chapter 4; and finally section 2.11 mentions some of the third-party software used to produce this thesis and the results therein.

2.1 Basic mathematical tools

Here we specify the most basic mathematical concepts and notations necessary for understanding this work.

2.1.1 A note on notation

$A = [a_{ij}]$ means that the matrix A comprises the elements $\{a_{ij}\}$, such that the $(i, j)^{\text{th}}$ element of A is a_{ij} (Cullis *et al.*, 1913). In this work, we generally use Roman capitals to refer to whole matrices or tensors and lowercase letters to refer to scalars or elements of a matrix; lowercase bold letters refer to vectors. Additionally, we adopt the convention that

$$A = [a_{ij}] \implies \left(\boldsymbol{a}_j = \sum_i a_{ij}\hat{\boldsymbol{i}}\right) \wedge \left(\boldsymbol{a}_i = \sum_j a_{ij}\hat{\boldsymbol{j}}\right),$$

where $\hat{\boldsymbol{i}}$ corresponds to the i^{th} unit column vector and $\hat{\boldsymbol{j}}$ to the j^{th} row vector, i.e. we form row and column vectors from matrix columns and rows by omitting the relevant indices.

2.1.2 Kronecker product

The Kronecker product is a binary operator, $\otimes$, and is defined thusly for two matrices:

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \dots & a_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & \dots & a_{mn}\boldsymbol{B} \end{bmatrix},$$

i.e. each term in the left-hand matrix becomes the coefficient to a block of the right-hand matrix. In component form, this can be represented as

$$(\boldsymbol{A} \otimes \boldsymbol{B})_{p(r-1)+v,q(s-1)+w} = a_{rs}b_{vw},$$

where A is an $m \times n$ matrix and B is an $p \times q$ matrix (Van Loan, 2000).

2.1.3 Tensor product

The tensor product is an extension of the outer product to two tensors, A and B is defined component-wise such that

$$(A \otimes B)_{a,\dots,n,\alpha,\dots,\nu} = a_{a,\dots,n}b_{\alpha,\dots,\nu}.$$

Predominantly, we will use it with vectors (rank 1 tensors) where it becomes

$$(\boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c})_{ijk} = a_i b_j c_k.$$

2.1.4 Reduction function

The reduction function, $\text{reduce}_{n=1}^{N}(\odot, b_n)$, returns the reduction of $\{b_n | n \in 1..N\}$ by applying an arbitrary associative binary operator $\odot$ repeatedly, i.e.

$$\text{reduce}_{n=1}^{N}(\odot, b_n) = b_1 \odot b_2 \odot b_3 \odot \ldots \odot b_N.$$

2.1.5 Frobenius norm

The Frobenius norm of a matrix A is

$$\|A\|_{\text{F}} = \sqrt{\sum_{i,j} |a_{ij}|^2} = \sqrt{\text{Tr}(A^\top A)}.$$

2.2 Networks

Networks are the key focus of this thesis. In this section we introduce them with more rigour than the introduction and provide some of their standard mathematical descriptions through the lens of graph theory.

Networks are models which allow us to explore the relationships between things and the large-scale impact of local relationships (Strogatz, 2001a). A complex network is a network with a large number of things and non-trivial wiring rules, the study of which started with Erdős & Rényi (1959) and was reinvigorated in the early 2000s with the discovery of scale-free and small world networks (Watts & Strogatz, 1998; Barabási & Albert, 1999). Complex networks are introduced in section 2.9 after we have the mathematical tools with which to explore them more thoroughly.

More rigorously, a network, or graph, is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of N nodes or vertices, with labels such that $\mathcal{V} = \{v_i | i \in \mathbb{N} \wedge i \in [1, N]\}$, and $\mathcal{E}$ is a set of 2-element subsets of $\mathcal{V}$. When referring to the node v_i, one often refers simply to the label i as these are unique. The elements of $\mathcal{E}$ may be ordered or unordered depending on whether the network is directed or not and represent the links or edges between the nodes. This wiring pattern is often represented as the adjacency matrix $A = [a_{i \leftarrow j}]$ where the value of the entry $a_{i \leftarrow j}$ indicates the weighting of the link from node j to node i. An entry of zero corresponds to there being no link. A small network is shown in figure 2.1.

2.2.1 Important concepts

In this subsection, we introduce the simplest tools we require for the discussion of networks.

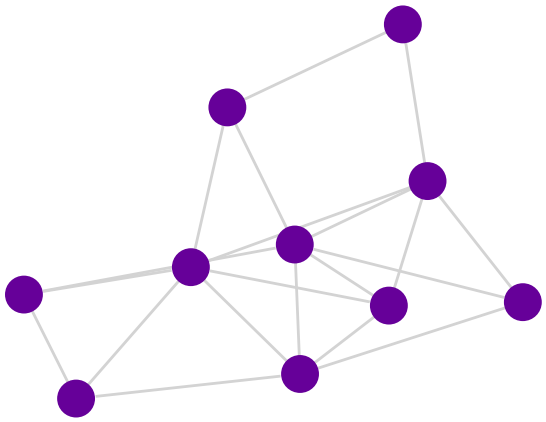

Figure 2.1 – A small network with 10 nodes (purple circles) and and fewer than 20 edges (grey lines)

Degree

The degree of a node in a network is the number of edges connected to the node. It is calculated as

$$k_i = \sum_{j=1}^{N} a_{i \leftarrow j}.$$

If the network is directed (i.e. $A \neq A^\intercal$), each node has an in-degree, k^{in}, and out-degree, k^{out}, showing the number of inbound and outbound links respectively. Intuitively, the first moment of these is identical, $\overline{k^{\text{in}}} = \overline{k^{\text{out}}} = \overline{k}$:

$$\overline{k^{\text{in}}} = \frac{\text{sum of all in-degrees}}{\text{number of nodes}} = \frac{\sum_{j=1}^{N} k_j^{\text{in}}}{N} = \frac{\sum_{j=1}^{N} (\sum_{i=1}^{N} a_{i \leftarrow j})}{N}$$

$$\overline{k^{\text{out}}} = \frac{\sum_{i=1}^{N} k_i^{\text{out}}}{N} = \frac{\sum_{i=1}^{N} (\sum_{j=1}^{N} a_{i \leftarrow j})}{N} = \frac{\sum_{j=1}^{N} \sum_{i=1}^{N} a_{i \leftarrow j}}{N} = \overline{k^{\text{in}}} = \overline{k},$$

so one often refers just to the 'mean degree' on a directed network rather than 'mean in-degree' or 'mean out-degree'.

The degree distribution of a network, $P(k)$, gives the probability of a randomly-chosen node having a certain degree (Newman, 2010). It is one of the main methods used to characterise networks.

Adjacency matrix

As introduced in section 2.2, the elements of the adjacency matrix, $\boldsymbol{A}$, are defined as:

$$a_{i \leftarrow j} = a_{ij} = \begin{cases} 1 & \text{if there is an edge from node } j \text{ to node } i \\ 0 & \text{otherwise.} \end{cases}$$

In a weighted network, the values correspond to the weights of the links which may be any real numbers (rather than just unity or zero); if *multi-edges*—multiple edges between two

nodes—are permitted, the entries in the adjacency matrix correspond to the number of edges present; *self-edges* or *self-loops* are represented by entries on the diagonal. Adjacency matrices and adjacency tensors, as will be introduced later in this chapter, are sometimes referred to as 'network representations' as they encode all the information one needs to reconstruct the topology of a network (Spinrad, 2003).

Subgraphs

A subgraph $G = (V, E)$ is a subset of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if and only if

$$(V \subseteq \mathcal{V}) \wedge (E \subseteq (\mathcal{E} \cap V^2)).$$

An induced subgraph is a subgraph wholly specified by the supergraph and a subset of nodes; all edges existing between nodes in that subset of the supergraph are included in the induced subgraph, so

$$\begin{aligned} \mathcal{G}_{\text{induced}} : \mathcal{P}(\mathcal{V}) &\mapsto \{G | G \subseteq \mathcal{G}\} \\ S &\to (S, \{(v_i, v_j) | v_i, v_j \in S \wedge (v_i, v_j) \in \mathcal{E}\}), \end{aligned}$$

where $\mathcal{P}(\mathcal{V})$ is the powerset of all vertices, i.e. the space containing all possible subsets of vertices.

Paths

A path p is a sequence of edges such that each consecutive pair of edges is such that the first edge ends and the second edge starts at the same node. A closed path is a path that starts and ends at a single node.

Graph colour

Coloured graphs are graphs whose nodes fall into discrete categories. A colouring, π, of $\mathcal{G}$ is a function mapping from $\mathcal{V}$ to some set Π, such that $\pi(i)$ returns the colour of the node i.

Ego-networks

An n-ego-network is a subgraph induced by the node of interest, the ego, and all of its neighbours a certain distance n or less from the ego; these neighbours are called the alters (Burt, 1980).

Colour can be applied trivially to ego-networks by setting the ego to be one colour and the alters to another.

Multilayer networks

A multilayer network is a network with multiple edge types. An intuitive example of a multilayer network is a transport network where each layer represents a mode of transport (air,

rail, road, etc.) and the nodes are physical locations and thus shared between all layers. We touch on the concept of multilayer networks in chapter 9.

2.2.2 Temporal or evolving networks

Almost no real-world systems are static: they change over time. The topologies of their network representations are no exception—they also change with time. We call these temporal networks; other authors sometimes call them dynamic, adaptive, or evolving networks. There are three main approaches to defining network wiring patterns which change with time.

Piecewise static networks

The simplest way to consider an evolving network is as a totally ordered set of adjacency matrices,

$$(A_t | t \in T),$$

i.e. as a sequence of static networks. This technique can be beneficial as it allows all of the standard network metrics to be employed, which will be used in part III. To the best of our knowledge, the term 'piecewise static' is our invention. We are unaware of any other terms for it although many authors apply it without giving it a name (e.g. Bagrow & Bollt, 2019).

Multilayer networks and adjacency tensors

An adjacency tensor is an extension of the adjacency matrix to multilayer temporal networks. They can take various forms, but a common one is made of elements

$$a^t_{i\alpha \leftarrow j\beta} = \begin{cases} 1 & \text{if there is an edge from node } j \text{ in layer } \beta \text{ to node } i \text{ in layer } \alpha \text{ at time } t \\ 0 & \text{otherwise,} \end{cases}$$

where the upper and lower indices are used solely for legibility and do not imply any special transformation rules.

Continuous time networks

The final way to consider temporal networks is in continuous time. This is an active area of research that we do not consider in this work. These networks are often described by a function per pair of nodes such that

$$\begin{aligned} A_{ij} : \mathbb{R} &\to \{1, 0\} \\ t &\mapsto A_{ij}(t). \end{aligned}$$

Although most real-world networks operate in continuous time, any data about them are usually quantised. Continuous time representations of real-world networks are therefore often interpolations of adjacency tensors.

2.3 Matrix decomposition

A matrix decomposition factorises a matrix into a product of matrices, e.g. a matrix X might be factorised such that $X = TW^{-1}$. These decompositions are useful for a variety of purposes; here we focus on ones that help us to reduce the number of dimensions required to approximate some data contained in a matrix X. We will exploit this technique in part III.

Principal component analysis (PCA) is a type of matrix decomposition (Wold *et al.*, 1987). The purpose of PCA is to find a new co-ordinate system such that the greatest variance is found on the first co-ordinate, the second greatest variance on the second co-ordinate and so on and that each co-ordinate is orthogonal, as shown graphically in figure 2.2. The transformation is defined by a set of unit vectors, $\{\boldsymbol{w}_k\}$, that map each row vector corresponding to an individual observation, $\boldsymbol{x}_i$, where $\overline{\boldsymbol{x}} = \mathbf{0}$, to a new vector, $\boldsymbol{t}_i$, containing the transformed co-ordinates, where

$$\boldsymbol{t}_i = \sum_k \boldsymbol{x}_i \cdot \boldsymbol{w}_k \hat{\boldsymbol{k}},$$

where $\hat{\boldsymbol{k}}$ is the k^{th} unit vector of the transformed space. It can therefore be seen that the first transformation vector, $\boldsymbol{w}_1$, must satisfy

$$\boldsymbol{w}_1 = \arg\max \left\{\sigma^2(\boldsymbol{t}'_1)\right\} = \arg\max \left\{\sum_i (\boldsymbol{t}'_1)_i^2\right\},$$

where $\sigma^2(\boldsymbol{t}'_1)$ is the variance of the vector $\boldsymbol{t}'_1 = [\boldsymbol{x}_i \cdot \boldsymbol{w}_1]$. If we stack the row vectors into a matrix $X = \{\boldsymbol{x}_i\}$, we can rewrite this equation such that

$$\boldsymbol{w}_1 = \arg\max \left\{\frac{\sum_i (\boldsymbol{x}_i \cdot \boldsymbol{w}_1)^2}{\|\boldsymbol{w}\|^2}\right\} = \arg\max \left\{\frac{\boldsymbol{w}_1^\intercal X^\intercal X \boldsymbol{w}_1}{\boldsymbol{w}_1^\intercal \boldsymbol{w}_1}\right\},$$

which is a Rayleigh quotient and therefore its maximum value is the largest eigenvalue of $X^\intercal X$ (Strang, 1993), i.e. the eigenvalue of the sample covariance matrix, which is obtained when $\boldsymbol{w}_1$ is the corresponding eigenvector. All subsequent eigenvectors and values correspond to the next best orthogonal directions of variance. These eigenvalues and eigenvectors can be obtained computationally cheaply through singular value decomposition (SVD, Halko *et al.*, 2011).

Therefore, the complete set of transformed co-ordinates is given by

$$T = XW,$$

where $T = \{\boldsymbol{t}_i\}$, so each row corresponds to a single observation and $W = \{\boldsymbol{w}_k\}$, where $\boldsymbol{w}_k$ is the eigenvector corresponding to the k^{th} largest eigenvalue. Since $\boldsymbol{w}_k$, the principal components, are arranged such that the first corresponds to the most variance, we can turn this into a dimension reduction by only considering the first few components, that is, the first few columns of W, we will have lost the least amount of variance possible.

For applications of this technique in network science, see section 2.6.8.

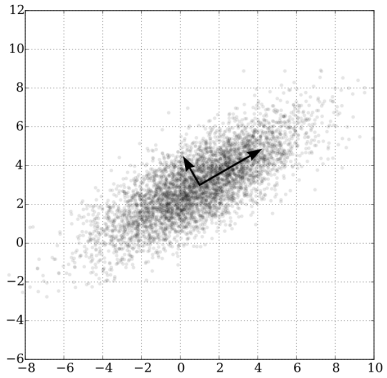

Figure 2.2 – Samples from a two-dimensional Gaussian distribution centred at (1,3) with the principal components plotted: the first component points towards the top right following the direction of maximum variance and the second component is orthogonal to the first (Nicoguaro, 2016)

2.4 Tensor decomposition

Tensor decompositions can be seen as an extension of matrix decompositions: matrix decompositions factorise a matrix into a product of matrices; tensor decompositions factorise a tensor into a tensor product of tensors. In this work, we choose to decompose tensors into the tensor product of rank-1 tensors (i.e. vectors). A order-N tensor X can be represented as

$$X = \sum_{r=1}^{R} \text{reduce}_{n=1}^{N}(\otimes, \boldsymbol{a}_{nr}), \tag{2.1}$$

where $\otimes$ is the tensor product and each $\boldsymbol{a}_{nr} \in \mathbb{R}^{d_n}$ where d_n is the dimension of the n^{th}-mode of the tensor X. If a smaller R than is necessary is chosen, the decomposition becomes an approximation. When an order N tensor is decomposed into RN vectors, the total number of dimensions changes from $\prod_{n=1}^{N} d_n$ to $\sum_{r=1}^{R} \sum_{n=1}^{N} d_n$, which in all sane applications will be smaller. Tensor decompositions are also known as canonical polyadic decompositions, tensor rank decompositions, tensor factorisations and CANDECOMP/PARAFAC (Harshman *et al.*, 1970; Carroll & Chang, 1970), reflecting the number of times the technique has been independently rediscovered.

To find the tensor decomposition of a tensor X, we need to minimise

$$\|X - \sum_{r=1}^{R} \text{reduce}_{n=1}^{N}(\otimes, \boldsymbol{a}_{nr})\|_F,$$

where $\boldsymbol{a}_{nr}$ are parameters to be found (R is generally chosen from domain knowledge). The most common way to approach this is to use an alternating least squares (ALS) method where one optimises for a single component at a time using least squares, holding all other

components fixed, and then iterating through each component repeatedly until convergence (Kolda & Bader, 2009). In this work, we were not especially concerned with using the most computationally efficient method so picked alternating proximal gradient tensor factorisation (APG-TF, Xu & Yin, 2013) as it was fast enough and had a public implementation. For our applications, we impose a further constraint such that

$$\forall i, n, r : a_{inr} \geq 0,$$

in order to aid interpretation as all of our adjacency tensors are non-negative. Such a decomposition is often referred to as non-negative CANDECOMP/PARAFAC (NNCP) decomposition. Tensor decompositions are generally used in network science for community detection and characterisation in temporal networks through decomposing their adjacency tensors, as we explore in section 2.7 and part IV.

2.4.1 Finding locally optimal numbers of components

In the literature, by far the most popular method for finding the optimal number of components, R, or communities in network science, to decompose a tensor into is to already know the answer from domain knowledge. Gauvin *et al.* (2014) did this when they chose to split their pupil-interaction network into the same number of communities as there were classes, into which the children were split.

In this work, we generally don't know the how many communities there should be; in many cases it is not even clear if 'how many communities are there?' is a meaningful question. We therefore must use some metrics to help us determine the best number of communities.

The normalised residual sum of squares or relative error, NRSS_R, is a measure of reconstruction error and is one approach used with tensor decompositions (Papalexakis *et al.*, 2012). It can be used to inform one's choice of R by looking for the number of components where the rate at which the residual sum of squares decreases as the number of components increases becomes linear, commonly known as an elbow. We define the normalised residual sum of squares as:

$$\text{NRSS}_R(X) = \frac{\|X - \sum_{r=1}^{R} \text{reduce}_{n=1}^{N}(\otimes, \boldsymbol{a}_{nr})\|_F}{\|X\|_{\text{F}}}, \tag{2.2}$$

where $\|X\|_{\text{F}}$ is the Frobenius norm of X.

Another approach in the literature is 'CORCONDIA', introduced by Bro & Kiers (2003). It attempts to measure how closely some assumptions made by CANDECOMP/PARAFAC decompositions are followed by the resultant decomposition. As with NRSS, one uses it to choose a value for R based on where an 'elbow' is visible.

2.5 Node characterisation

The smallest scale of a network is that which considers nodes. They are therefore often the easiest thing to improve in a real-world network, for example, one might upgrade traffic lights at a road junction, and so being able to describe them is vitally important. We apply node characterisation extensively in chapter 4 where we consider how academic researchers might change the way they choose co-authors for their papers with a view to increasing the length of their academic careers.

Functions which characterise the importance of nodes are commonly called 'centralities'.

2.5.1 Characterisation from local topology

The most obvious and most commonly used node centrality is degree centrality. It is simply the degree of a node and obviously gives some measure of a nodes importance. Other, more involved centralities are discussed in this section.

Local clustering coefficient

The immediate neighbourhood of a node v_i is $N_i = \{v_j | e_{ij} \in E \vee e_{ji} \in E\}$. Note that $v_i \notin N_i$ unless self-loops are permitted. The local clustering coefficient for undirected graphs (Newman, 2010) is then

$$\begin{aligned} C_i &= \frac{\text{number of links in subgraph induced by neighbourhood}}{\text{number of possible links in neighbourhood}} \\ &= \frac{2|\{e_{jk}|(v_j, v_k \in N_i) \wedge (e_{jk} \in E)\}|}{k_i(k_i - 1)}. \end{aligned}$$

Note that this is equivalent to the global clustering coefficient, introduced in section 2.6.6, on the 1-ego-network of the node v_i.

Ego-networks

As was just touched on, nodes can be characterised by the characterisation of their ego-networks. A number of authors have explored this possibility. Akoglu *et al.* (2010) use simple criteria—number of edges, etc.—from a node's ego-network to characterise it. A similar approach is employed by others who use graphlet profiles (explored in section 2.6.7) on ego-networks to characterise nodes (O'Callaghan *et al.*, 2012; Cunningham *et al.*, 2013). For more detail on the characterisation of networks see section 2.6. Other approaches applying machine learning techniques exist (Rossi *et al.*, 2017).

2.5.2 Characterisation from global topology

A node's position within the whole network is also often worth considering. Imagine a very close-knit group of friends who do not interact much with people outside of that group of

friends. From local topological measures, people within that group of friends may appear very important when they are globally very isolated.

Betweenness centrality

The betweenness centrality of a node v_i is the normalised sum of the ratios of the number of shortest paths that pass through the node,

$$g(v_i) = \frac{2\sum_{i\neq j\neq k}\sigma_{ij}(v_k)}{(N-1)(N-2)\sum_{i\neq j\neq k}(\sum_k \sigma_{ij}(v_k))},$$

where $\sigma_{i\leftarrow j}(v_k)$ is the number of geodesic paths from node v_j to node v_i which pass through node v_k and N is the number of nodes in the graph (Freeman, 1978). Intuitively, it is a measure of how many short paths a node lies on, so 'bridges' that connect two clusters have a high betweenness centrality.

Closeness centrality

Closeness centrality (Rochat, 2009), H, is a measure of how close a node is to all other nodes. It is defined as

$$H(v_i) = N\sum_j (d(v_i, v_j)^{-1}),$$

where

$$d(v_i, v_j)^{-1} = \begin{cases} (\text{the distance between } v_i \text{ and } v_j)^{-1} & \text{if } v_i \text{ is connected to } v_j, \\ 0 & \text{otherwise.} \end{cases}$$

Trophic level

The trophic level of a node in a directed graph (Levine, 1980), $s(v_i)$, is defined such that

$$s(v_i) - 1 = \begin{cases} 0 & \text{if } k_i^{\text{in}} = 0, \\ \frac{\sum_j a_{ij}s(v_j)}{k_i^{\text{in}}} & \text{otherwise.} \end{cases}$$

It is predominantly used in ecological contexts where it denotes how 'far up' a food chain a node is.

2.5.3 Graph embeddings and representation learning

A graph embedding maps nodes to a low dimensional space. This is a very active area of research because such embeddings are ideal as inputs to machine learning techniques where one is trying to predict things about nodes from their position within the topology. Hamilton *et al.* (2017) and Goyal & Ferrara (2018) provide thorough reviews of the subject. It is not an area we have time to explore deeply in this thesis, but it would be remiss of us if we failed to mention the similarities between graph embedding and the work we perform in part III where we map entire graphs to a low dimensional space.

2.6 Network characterisation

Network characterisation is the task of describing a network mathematically such that it can be compared with other networks. For a good review of the subject, see Costa *et al.* (2007). Many network characteristics are termed metrics in the sense that 'they record something about the network'; this does not mean that they satisfy the identity of indiscernibles (i.e. $d(x, y) = 0 \iff x = y$ for any metric d) or any of the other conditions for a formal mathematical metric.

Here, we will only present the work most relevant to our own. First, we start with the simplest: networks are often characterised by the number of nodes, $N = \|\mathcal{V}\|$, and the number of edges, $\|\mathcal{E}\|$. The density of an undirected graph is $\rho = \frac{2\|\mathcal{E}\|}{N(N-1)}$.

2.6.1 Laplacian

The Laplacian of a network is defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A},$$

where $\boldsymbol{D}$ is a diagonal matrix of the degrees of each node, $\boldsymbol{D} = \{\delta_{ij} k_i\} = \{\delta_{ij} \sum_k a_{ik}\}$, and $\boldsymbol{A}$ is the adjacency matrix (Newman, 2010, p. 152).

2.6.2 Average path length

The path length between two nodes is the length of the route taken between the two nodes. For an unweighted network, this is the number of edges such a path would touch; for a weighted network it would be the sum of the weights of those edges. The shortest, or geodesic, path length is the shortest possible route between two nodes; often this is just abbreviated to 'path length' in the literature. A key network characteristic is the average length of all shortest paths.

2.6.3 Diameter

The diameter of a network, D, is the length of the longest geodesic path in the network. In very large networks, one might instead take the 90$^{\text{th}}$ percentile of the path-length distribution to ensure a more representative picture is gained.

2.6.4 Path efficiency

The path efficiency, E, of a graph is

$$E(\mathcal{G}) = \frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{d(v_i, v_j)},$$

where $d(v_i, v_j)$ is the distance between node v_i and node v_j, where $d(v_i, v_j)^{-1} = 0 \iff$ there is no path between v_i and v_j. It is closely related to the average path length and has the added benefit of naturally handling disconnected components.

2.6.5 Degree correlation matrices

The degree mixing matrix or degree correlation matrix, D, (Newman, 2003) is defined such that each entry

$$d_{ij} = P(\text{edge chosen at random connects a node of degree } i \text{ to a node of degree } j).$$

Examples can be found in figures 2.3, 2.4 and 2.5. Entries in the off-diagonals correspond to disassortative mixing—nodes linking to nodes unlike themselves (at least in terms of degree)—and diagonal entries (or close to diagonal) correspond to assortative mixing.

2.6.6 Global clustering coefficient

The global clustering coefficient (GCC), C, is a measure of how many triangles there are in a network and is closely related to transitivity, i.e. for an undirected graph, $(v_i, v_j) \in \mathcal{E} \wedge (v_j, v_k) \in \mathcal{E} \implies (v_k, v_i) \in \mathcal{E}$. Precisely,

$$C = \frac{\text{number of closed paths of length 2}}{\text{number of paths of length 2}};$$

e.g. in a network of social connections, it measures the probability that if Alice knows Bob and Bob knows Charlie, Charlie knows Alice (Newman, 2010).

2.6.7 Graphlets and motifs

Motifs are characteristic subgraphs, tallied by automorphism group—that is, graphs whose nodes can be relabelled to produce an identical adjacency matrix—of a graph. The 'characteristic' refers to subgraphs that appear more often than one would expect from a null model such as a configuration model (Milo *et al.*, 2002). They described them as:

> patterns of inter-connections occurring in complex networks at numbers that are significantly higher than those in randomized networks.

Graphlets are a similar concept, originally with the distinction that the subgraphs must be induced and that no significance test or normalisation is required (Przulj, 2007). Many authors now use motifs to refer to induced subgraphs, i.e. subgraphs defined by a subset of nodes and the edges between those nodes. Graphlets have also been used to characterise nodes through their ego-networks (Cunningham *et al.*, 2013; O'Callaghan *et al.*, 2012) through choosing a few graphlets to track over time and counting the frequencies of their appearances.

Coloured graphlets

The idea of graphlets can be expanded to match two coloured subgraphs if they can have their nodes relabelled so that they have the same adjacency matrix and each node must have the same colour as its counterpart in the other subgraph (Wernicke & Rasche, 2006).

We believe this particularly pertinent for exploring networks such as ego-networks, where there are two distinct classes of node, the ego and the alters.

Graph isomorphism

Two graphs are isomorphic if their adjacency matrices can be made identical after relabelling their nodes. A stricter form of isomorphism demands that after this relabelling the graph colourings match up, i.e. for two relabelled graphs $\mathcal{G}$ and $\mathcal{G}'$, $\pi_{\mathcal{G}}(i) = \pi_{\mathcal{G}'}(i)\ \forall i$. There are a handful of routines for determining whether graphs are isomorphic—see McKay & Piperno (2014) for a detailed review—but for small graphs they have broadly the same characteristics. Throughout this work, we use nauty (McKay & Piperno, 2014) as it supports colour. Nauty takes a graph and returns a 'canonical representation' string, canon, such that

$$\text{canon}(\mathcal{G}) = \text{canon}(\mathcal{G}') \iff \mathcal{G} \text{ and } \mathcal{G}' \text{ are isomorphic.}$$

Graphlet enumeration

There are many methods for counting graphlets and motifs (Masoudi-Nejad *et al.*, 2012). Since that review, some new faster techniques have emerged: (Meira *et al.*, 2014; Ahmed *et al.*, 2016). Our main criterion was that the the routine should consider node colour when determining isomorphism. FANMOD (Wernicke & Rasche, 2006) is the only existing algorithm we are aware of which meets this demand. Kavosh (Kashani *et al.*, 2009) does not support colour out of the box but uses nauty which does. Faster methods exist but do not support graph colour which we explore in chapter 3.1.

Graphlet profile

The result of graphlet enumeration is a graphlet profile: a mapping from graphlets to frequencies. A graphlet profile of graphlets of a certain size is a vector $\boldsymbol{g}_n$. There are many schools of thought on how one ought to normalise graphlet profiles. In this work, we solely use graphlet ratio profiles (Cunningham *et al.*, 2013) such that

$$\hat{\boldsymbol{g}}_n = \frac{\boldsymbol{g}_n}{|\boldsymbol{g}_n|}.$$

Other methods include z-scores (Milo *et al.*, 2002) which attempt to create 'significance' profiles where graphlets which occur more often than random chance would suggest receive higher scores. We do not apply this in our work as it is computationally expensive because many configuration models for each graph must be generated and because we compare our graphs with many other graphs, as we discuss in chapters 3.1 and 5.

2.6.8 Matrix decompositions for network characterisation

As can be seen merely from the number of subsections in the network characterisation section, network characterisation is a potentially very high-dimensional problem. As was introduced in section 2.3, one approach to making high-dimensional problems tractable is to apply matrix decompositions. Costa *et al.* (2007, pp. 221-222) and Bounova & de Weck (2012) explore the use of PCA on data matrices constructed from standard network metrics such as the clustering coefficient. Costa *et al.* find that they cannot distinguish between networks with the few features they consider and instead resort to canonical variable analysis, a close variant of PCA which uses labelled networks (i.e. network A is type 1, network B is type 2, network C is type 1, etc.) to maximise the distance between different labels. Bounova & de Weck use rather more features and find that they can distinguish between different kinds of networks without needing to label them. Ronqui & Travieso (2015) take an interesting approach where they construct features from the pairwise correlation of node centralities within a network (e.g. the correlation between node degree and local clustering coefficient) and they also find that they can distinguish between types of network after applying PCA to a data or feature matrix formed of these features.

However, our view is that these authors have somewhat missed the point of PCA: it can be used for feature selection and extraction, i.e. an unbiased selection of the best features to explain the variance in some data. In our view, one should therefore aim to give PCA the highest-dimensional representation of the data that can be reasonably found so that it may choose the best features itself. Instead, the authors cited have fed PCA a selection of 'human-crafted' features. In the field of machine learning, this practice is termed 'feature engineering' and has been found to lead to worse results, most notably in the field of computer vision to the point where feature engineering is deemed 'classical computer vision' (Lee, 2015). In part III, we explore simple high-dimensional descriptions of networks that can be used in data matrices and see whether they still have the power to distinguish between types of network.

2.6.9 Information-theoretic measures

There has been some progress in using metrics from information theory—a field which largely deals with the differences between distributions—to characterise networks (Carpi *et al.*, 2011; Schieber *et al.*, 2016, 2017). They are broadly based on the Shannon entropy, H, of a discrete probability distribution $P(k)$,

$$H = -\sum_k P(k) \log_2 P(k),$$

where $P(k)$ in network science is usually the degree distribution (Costa *et al.*, 2007, p. 167). A common metric on this space is the square root of the Jensen-Shannon divergence,

$$\mathcal{J}^{1/2}[P_1, P_2] = \sqrt{H\left[\frac{P_1 + P_2}{2}\right] - \frac{H[P_1] + H[P_2]}{2}}.$$

Schieber *et al.* (2017) take an approach which has a nice symmetry to the approach we take in parts II and III: they characterise whole networks by first characterising nodes and then comparing the distributions of characterised nodes using $\mathcal{J}$. Bagrow & Bollt (2019) apply $\mathcal{J}$ to

$$b_{lk} = \text{the number of nodes who have } k \text{ nodes at distance } l,$$

the network portrait (Bagrow *et al.*, 2008). They show that small instances of Erdős-Rényi and Barabási-Albert models are distinguishable with intra-model distances smaller than inter-model distances. They also provide an example of it measuring changes in temporal networks, highlighting a key issue with $\mathcal{J}$: as it is always positive, there is no indication of the 'direction' of change. In the authors' temporal network example, the network clearly takes a 'step back' at one time step to a simpler, more regular network, before it becomes more complex again, which $\mathcal{J}$ cannot represent.

We are unaware of any attempt to compare these various techniques in a quantitative manner.

2.7 Community characterisation

Many of the techniques described in section 2.6 can be trivially applied to communities, that is, induced subgraphs whose nodes are related in some manner, by considering the subgraphs induced by those subsets of nodes. In this work, we predominantly focus on community characterisation in temporal networks, although many of the techniques could be applied to multilayer or static networks.

The first step in community characterisation is to determine the composition of communities. There are countless routines to determine these, for example: the Louvain algorithm (Aynaud & Guillaume, 2010), statistical null models (Bassett *et al.*, 2013), algorithms which exploit the historic community structure of the network (He & Chen, 2015; He *et al.*, 2017), Markov models (Rosvall *et al.*, 2014), semi-definite programming (Tantipathananandh & Berger-Wolf, 2011), gravitational relationship between nodes (Yin *et al.*, 2017) and temporal matrix factorisation (Yu *et al.*, 2017), amongst others. Machine learning techniques (Savić *et al.*, 2019c), genetic algorithms (Folino & Pizzuti, 2014), consensus clustering (Aynaud & Guillaume, 2010) and tensor factorisation (Gauvin *et al.*, 2014; Araujo *et al.*, 2016; Verma & Bharadwaj, 2017) have only recently been used for the detection of communities in temporal networks.

Of these, tensor factorisation has a number of advantages. It is a kind of soft clustering, where nodes have membership strengths to multiple communities, and so it lends itself naturally to situations where nodes are in multiple communities or community membership is not well-defined. It can deal with clustering across multilayer networks and is especially useful at characterising the communities of temporal networks, where it can produce measures of the activity level of each community over time (Gauvin *et al.*, 2014).

In part IV, we apply tensor decomposition or factorisation, as was introduced in section 2.4. We chose this approach above all the others because it simultaneously performs soft clustering of the network—i.e. nodes are given 'strengths' of membership to all communities rather than only being in a single community—and characterises these communities by giving their activity over time, which no other technique can do.

Previous works applying tensor decompositions to temporal networks have had very clearly defined 'real' communities—such as pupils separated into classes at school (Gauvin *et al.*, 2014)—or have not been concerned with picking an optimal number of components (Kang *et al.*, 2012). Tensor decompositions have also been used for community detection in static multilayer networks. Generally in these works, the 'ground truth' (i.e. 'real') communities have been known. We consider how we can extend this technique to areas where there are no ground truth communities in chapter 7.1, which we apply in chapter 8.

2.8 Edge characterisation

We do not consider the characterisation of edges in this thesis, but we mention it here for completeness. Edges share many similarities with nodes: edge betweenness centrality follows readily from node betweenness centrality and the ego-network of an edge can be defined with ease. Other more exotic measures have been explored, such as those based on κ-path centrality, an extension of betweenness centrality to 'fairly short' rather than shortest paths (De Meo *et al.*, 2012).

2.9 Complex networks

Complex networks are large networks with non-trivial topologies: e.g. with degree mixing matrices that exhibit assortativity or disassortativity, community structure where there are groups of nodes within which nodes more densely linked to each other than outside the groups, and average path lengths that are not functions of the number of nodes in the network, or at most, grow very slowly with the number of nodes in the network. A complex networked system consists of many interacting agents that communicate with each other over a web of complex interconnections. In the formalism of network science, the agents, or 'things', are called nodes, which are connected together by links or edges; other formalisms such as multi-agent systems could be applied but are not explored in this work.

2.9.1 Dynamics (of topology)

Complex networks are often formed by dynamical processes; they grow and change over time. Complex networks are generally created by many agents making small decisions, such as websites choosing which other websites to link to, or people choosing which other people to send emails to, or neurons making links with nearby neurons. Strogatz (2001a) and Gross & Sayama (2009) give a very good introduction to the field for the uninitiated reader.

Complex networks often have complex dynamics on each node, influenced by the topology—for example, in a social network, each person could be considered 'ill' or 'healthy' and the likelihood of each state depends on the number of 'ill' and 'healthy' friends they are in contact with; this is the SI (susceptible-infected) model. Interested readers should see Danon *et al.* (2011) and Pastor-Satorras *et al.* (2015) for detailed reviews. This is not something we have room to consider explicitly in this work, although it is of course considered implicitly in many chapters where we consider the real-world characteristics of nodes and theorise about how the topology might affect these characteristics.

2.9.2 Network models

Dealing with real-world networks poses a number of issues, such as being inflexible in the number of nodes there are or poor labelling. Network models are convenient for the generation of synthetic networks with specific parameters and we use them extensively in part III as paragons by which we can test the quality of our characterisation.

Random networks

A random network is created by choosing a random subset of all of the possible links between nodes. The links are each present with some probability, p. This model therefore has two parameters: $G(n, p)$, where n is the number of nodes[1]. This type of network was first introduced by Erdős & Rényi (1959) and can be considered the 'mother' of all research into complex networks, as they were the first people to consider large, random networks, as opposed to the traditional, regular graphs that were studied before. These networks have many interesting properties, such as the sudden appearance of a giant component which contains a large fraction of the nodes as p is increased. An example implementation of the model is given in algorithm 2.1. The degree distribution and degree mixing matrix of an instance of an Erdős-Rényi model are shown in figure 2.3 where the Poisonnian nature of both can be seen.

[1]It should be noted that Erdős and Rényi originally formulated their model with a number of nodes and a number of edges as its parameters. This is equivalent to having an edge probability which we feel captures a more important aspect of the model.

Algorithm 2.1 Erdős-Rényi random graph model implementation in Python 3 (Schult, 2008)

```
# G(n, p)
def Erdős_Rényi_graph(n, p, seed=None, directed=False):
    if directed:
        edges = itertools.permutations(range(n), 2)
        G = nx.DiGraph()
    else:
        edges = itertools.combinations(range(n), 2)
        G = nx.Graph()
    G.add_nodes_from(range(n))
    if p <= 0:
        return G
    if p >= 1:
        return complete_graph(n, create_using=G)

    for e in edges:
        if seed.random() < p:
            G.add_edge(*e)
    return G
```

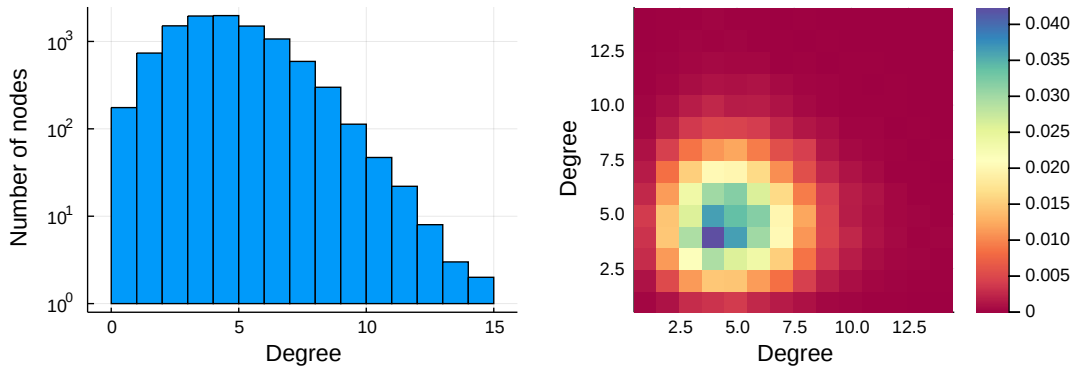

Figure 2.3 – The degree distribution (left) and the degree mixing matrix (right) of an instance of an Erdős-Rényi model with 10,000 nodes and 20,000 edges

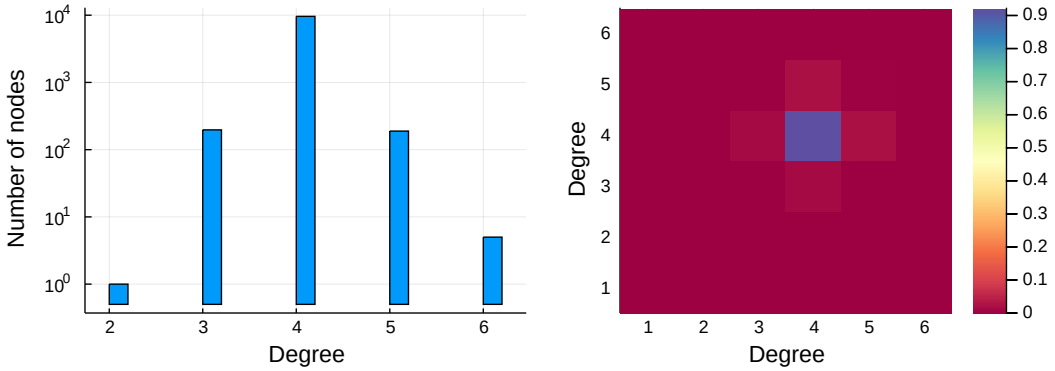

Figure 2.4 – The degree distribution (left) and the degree mixing matrix (right) of an instance of a Watts-Strogatz model with 10,000 nodes and 20,000 edges

Small-world

A small-world network is one whose average path length, L, grows with the network such that $L \propto \log N$, whilst maintaining a high clustering coefficient. The first people to propose a model that could generate such a network were Watts & Strogatz (1998), whose network starts out as a nearest-neighbour ring network. Then, a random shortcut is created from each node with a set probability. This then also has three parameters: $G(n, p, m)$ where n is the number of nodes, p is the probability of each edge being rewired a random node and m is the initial degree of the nodes. These networks are called small-world because their path efficiencies are much greater than the corresponding nearest-neighbour network. An example implementation of the model is given in algorithm 2.2. The degree distribution and degree mixing matrix of an instance of a Watts-Strogatz model are shown in figure 2.4. The difference with figure 2.3 is striking; the distributions are very tight.

Scale-free

Scale-free networks are networks with power-law degree distributions, $P(k) \propto k^{-\alpha}$, where α is a constant and generally lies such that $2 < \alpha < 3$, with so-called 'fat tails' where a small number of nodes have a very large number of links (Barabási & Albert, 1999). They are called scale-free because their degree distributions look the same at different scales—e.g. if one takes the k-core of a scale-free network, that is, a subgraph where every node has at least degree k, the resultant network is also a scale-free network. The most famous model of these networks, and the only scale-free model we consider in this work, is the Barabási-Albert model. This model grows a network from m unconnected nodes. Nodes are added and connected to m pre-existing nodes chosen with probability proportional to the degree of those nodes. These models therefore have two main parameters: $G(n, m)$ where n is the number of nodes and m is the number of links to add with each node. An implementation is shown in algorithm

Algorithm 2.2 Watts-Strogatz small-world graph model implementation in Python 3 (Schult, 2008)

```
# G(n,p,m)=watts_strogatz_graph(n,m,p)
def watts_strogatz_graph(n, k, p, seed=None):
    if k >= n:
        raise NetworkXError("k>=n, choose smaller k or larger n")

    G = Graph()
    nodes = list(range(n))  # nodes are labeled 0 to n-1
    # connect each node to k/2 neighbors
    for j in range(1, k // 2 + 1):
        targets = nodes[j:] + nodes[0:j]
        G.add_edges_from(zip(nodes, targets))
    # rewire edges from each node
    # loop over all nodes in order (label) and neighbors in order (distance)
    # no self loops or multiple edges allowed
    for j in range(1, k // 2 + 1):  # outer loop is neighbors
        targets = nodes[j:] + nodes[0:j]
        # inner loop in node order
        for u, v in zip(nodes, targets):
            if seed.random() < p:
                w = seed.choice(nodes)
                # Enforce no self-loops or multiple edges
                while w == u or G.has_edge(u, w):
                    w = seed.choice(nodes)
                    if G.degree(u) >= n - 1:
                        break  # skip this rewiring
                else:
                    G.remove_edge(u, v)
                    G.add_edge(u, w)
    return G
```

Algorithm 2.3 Barabási-Albert scale-free graph model implementation in Python 3 (Schult, 2008)

```
# G(n, m)
def Barabási_albert_graph(n, m, seed=None):
    # Add m initial nodes (m0 in Barabási-speak)
    G = empty_graph(m)
    # Target nodes for new edges
    targets = list(range(m))
    # List of existing nodes, with nodes repeated
    # once for each adjacent edge
    repeated_nodes = []
    # Start adding the other n-m nodes. The first node is m.
    source = m
    while source < n:
        # Add edges to m nodes from the source.
        G.add_edges_from(zip([source] * m, targets))
        # Add one node to the list for each new edge just created.
        repeated_nodes.extend(targets)
        # And the new node "source" has m edges to add to the list.
        repeated_nodes.extend([source] * m)
        # Now choose m unique nodes from the existing nodes
        # Pick uniformly from repeated_nodes
        # (preferential attachment)
        targets = _random_subset(repeated_nodes, m, seed)
        source += 1
    return G
```

2.3. The degree distribution and degree mixing matrix of an instance of a Barabási-Albert (Barabási & Albert, 1999) model are shown in figure 2.5. The contrast with figures 2.3 and 2.4 is immediate—there are very clear 'fat tails' where a handful of nodes have a very large number of links, and are fairly likely to be linked to by nodes of all degrees despite their small number. Recent research has suggested that true scale-free networks are rarer than once thought (House *et al.*, 2015; Broido & Clauset, 2019).

Configuration model

The configuration model (Molloy & Reed, 1995a) takes a degree sequence, $\{k_i | i \in \mathcal{E}\}$, and returns a randomly rewired graph with the same degree sequence. These models are predominantly used to generate random graphs similar to other graphs with a given degree distribution.

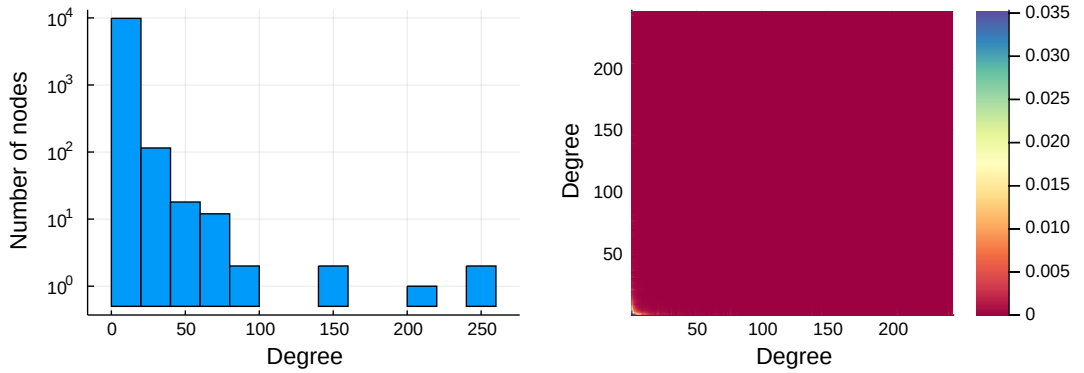

Figure 2.5 – The degree distribution (left) and the degree mixing matrix (right) of an instance of a Barabási-Albert model with 10,000 nodes and 20,000 edges. As this is a scale-free network, the majority of the nodes are in the bottom left of each subfigure, especially with the degree mixing matrix

2.9.3 Real-world networks

Real-world networks are a key component of this thesis: complex networks are predominantly of interest to us because they represent real-world systems. We refer back to them in every part: part II explores an academic co-authorship network, part III characterises a covert collaboration network and a selection of food webs and part IV analyses a number of temporal software dependency networks and a temporal linguistic network.

Collaboration networks

There are many types of social collaboration network. In this thesis, we only consider three.

Academic co-authorship networks describe academics who have written papers or other publications together. In a co-authorship network, the nodes are individual authors and the links between the authors represent collaborations.

One of the main challenges with real-world co-authorship networks is disambiguation: the ability to identify when two authors with the same name are different people. Datasets exist where this procedure has been completed (Zhuang *et al.*, 2013).

As with other temporal networks made from data where events are recorded as instantaneous but in reality have some unknown duration, a window size must be chosen for which the events are extended. For example, a 3-year window size means that a co-authorship registered in a certain year is extended to also include the two years before it; many collaboration network datasets effectively set the window size to infinity and consider the network as if it were static. This author's own co-authorship network is shown in figure 2.6.

For a detailed overview, see Kumar (2015). Some work has been conducted with graphlets and motifs on co-authorship networks (Krumov *et al.*, 2011; Choobdar *et al.*, 2012a; Chakraborty

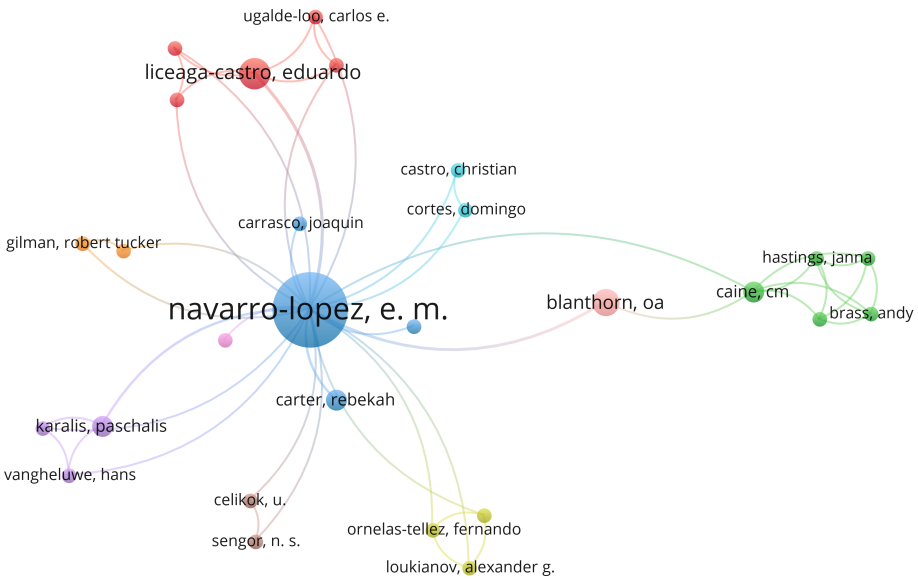

Figure 2.6 – This author's co-authorship 2-ego-network with a window size of ∞ visualised using VOS-Viewer (Van Eck & Waltman, 2009). Node size is proportional to number of papers published with people in the network shown

et al., 2015), but coloured graphlets have been neglected; some authors have considered the link between node centralities, career length and citation counts (Li *et al.*, 2013b).

Clandestine collaboration networks are networks of human collaboration operating in a covert fashion. In this work, we briefly consider the network of Suffragettes, women who campaigned for universal suffrage in the UK (Edwards & Crossley, 2009) in chapter 6. See Broccatelli *et al.* (2016) for a thorough treatment.

Software dependency networks are directed networks of collaboration between computer programmes (Decan *et al.*, 2018a). Modern software development is often a collaborative effort involving many authors through the re-use and sharing of code through software libraries. Many of these libraries and software packages are open-source and developed in the open on sites such as `GitHub`, so there is a large amount of data available about these networks.

We explore these networks in detail in chapter 9.

Semantic networks

A semantic network is one where the nodes are individual meanings of a word—lemmas, in the jargon. For example, here are two of the lemmas relating to 'mean' from the Oxford English Dictionary (OED, Coleridge *et al.*, 2019):

> mean: The average of a set of numerical values, as calculated by adding them together and dividing by the number of terms in the set.
>
> mean: Inferior in rank or quality; unpleasant.

which leads to one of this author's favourite jokes:

> The University of California Statistics Department: where mean is normal and deviation standard.

A hierarchical linguistic network links lemmas in a categorical fashion; for example, 'the world' would have an outgoing link to 'Asia' which in turn would have an outgoing link to 'China' and 'brick' may be pointed to by 'house' which might be pointed to by 'shelter'. See Solé *et al.* (2010) for a good introduction to language networks. We consider a temporal linguistic network in chapter 9.

The majority of work studying semantic networks is concerned with improving how machines can 'understand' it by changing the way semantic networks are presented to machine learning techniques (Anisimov *et al.*, 2014; Nakatsuji, 2016; Piao *et al.*, 2017). Kay *et al.* (2009) provide similar data to that which we use in chapter 9 (Coleridge *et al.*, 2019). Some analysis has been performed on this with the intent of uncovering interesting patterns in the language for humans rather than machines (Alexander, 2012; Alexander & Kay, 2014), but no-one has really applied state-of-the-art network science techniques to this data in an attempt to understand its evolution.

For a detailed survey on detecting semantic change, such that temporal semantic networks can be created, see Tahmasebi *et al.* (2018).

Ecological networks

Ecological networks represent the interactions between species. In this work, we predominantly consider food webs where $a_{i \leftarrow j} = 1 \iff$ species i predates species j; the direction of a link shows the flow of energy in the food web (Montoya *et al.*, 2006). This leads to several interesting properties: cycles are very rare and the diameters of the networks, closely related to the maximum trophic level, tend to be small due to the unavoidable loss of energy. Recent work has suggested that the lack of cycles could be linked to trophic levels (Johnson & Jones, 2017).

In this work, the real-world food webs we consider are provided by Thompson & Townsend (2003). We explore similarities between these and other networks in chapter 6.

2.10 Statistical methods

When dealing with noisy real-world data and many parameters, it is often beneficial to apply statistical techniques to ensure that what appear to be firm results are unlikely to be due to random chance. The methods from this section are applied particularly in part II where incautious analysis of our high-dimensional graphlet profiles would put us at a high-risk of finding spurious correlations.

2.10.1 Multiple linear regression

Multiple linear or multilinear regression is a simple method for investigating the relationship between many continuous so-called explanatory variables x_i and a single dependent variable, y, with

$$y = \Sigma_i \beta_i x_i,$$

where β_i are coefficients to be determined, commonly by the least-squares method (Montgomery *et al.*, 2012). β_{reduced} coefficients, the coefficients multiplied by the standard deviation of each explanatory variable, are often reported to ease interpretation of the coefficients. Each reduced coefficient corresponds to the change in the dependent variable one could reasonably expect as a result of increasing the corresponding explanatory variable from a figure representative of the samples with a low amount of that variable, to a figure representative of samples with a high amount of that variable.

2.10.2 Statistical significance

When sampling data from a population—i.e. choosing a subset of that population—and performing an experiment on that sample, it is possible that any conclusions drawn from that sample are simply due to the composition of the sample and are not representative of the population as a whole. For example, assume we have a population represented by a matrix

$$P = [p_{ij}] = [\delta_{ij}],$$

where δ_{ij} is the Kronecker delta, and we happen to sample the diagonal of the matrix. We would be led to believe that $p_{ij} = 1$. Statistical significance tests aim to quantify the likelihood of us being misled in such a manner (Fisher, 1925). More rigorously, a statistical significance test calculates the probability that an experiment would give a result at least as extreme at the one received given that the null hypothesis is true; this probability is called the 'p-value'. The null hypothesis will vary according to the experiment performed but generally corresponds to a 'negative' result. For example, if we propose that two variables are linked such that $y = \beta x + c$, the null hypothesis is that $\beta = 0$, that there is no link. One generally chooses a threshold for

Activity	Software used
General programming	Julia (Bezanson *et al.*, 2012), Python (Van Rossum & Drake Jr, 1995)
Exploratory data analysis	Jupyter notebook (Kluyver *et al.*, 2016), Julia
Communication	LaTeX, LyX[2], natbib, ebgaramond, fancyhdr, beamer, microtype, pandoc[3]
Visualisation	Plots.jl[4], GR[5], Gadfly.jl[6], Gephi (Bastian *et al.*, 2009), VOSViewer (Van Eck & Waltman, 2009)
Final data analysis	Julia, MATLAB, Python, LightGraphs.jl (Bromberger, 2017), ScikitLearn.jl[7] (Pedregosa *et al.*, 2011), JuliaDB[8], Query.jl[9], MATLAB.jl[10], TensorToolbox (Bader & Kolda, 2007)
Literature management	Mendeley, Zotero

Table 2.1 – List of software used

the p-value below which results are accepted or rejected; such a value is called the significance level. A significance level of 5% is generally used for preliminary studies, and corresponds to a confidence level of 95%, meaning that 19 times out of 20 a result is real and not due to sampling error.

We apply a specific implementation of a significance test, the F-test (Bates *et al.*, 2019), to test whether the coefficients in our linear regression are statistically significant. Its p-values are sometimes referred to as $Pr(>|t|)$.

2.10.3 Bonferroni correction

The significance levels mentioned above are only valid in the context of a single, isolated, hypothesis test (where the null hypothesis is that the coefficient is zero). Testing many hypotheses simultaneously, for example with a multilinear regression, greatly increases the probability of false positives. We therefore apply a Bonferroni correction to guard against this—we check the p-values against a corrected significance level of $\frac{\alpha}{n}$, where α is the desired p-value and n is the number of hypotheses tested (Bonferroni, 1950).

2.11 Software used

This thesis was made possible through the efforts of many software developers. Some of the software used in the course of creating this work is listed in table 2.1.

[2] https://www.lyx.org/
[3] https://pandoc.org/
[4] https://github.com/JuliaPlots/Plots.jl
[5] https://github.com/sciapp/gr
[6] https://github.com/GiovineItalia/Gadfly.jl
[7] https://github.com/cstjean/ScikitLearn.jl
[8] https://github.com/JuliaComputing/JuliaDB.jl
[9] https://github.com/queryverse/Query.jl
[10] https://github.com/JuliaInterop/MATLAB.jl

Bibliography

Ahmed, Nesreen K, Willke, Theodore L, & Rossi, Ryan A. 2016. Estimation of local subgraph counts. *Pages 586–595 of: 2016 IEEE international conference on big data (big data)*. IEEE.

Akoglu, Leman, McGlohon, Mary, & Faloutsos, Christos. 2010. Oddball: Spotting anomalies in weighted graphs. *Pages 410–421 of: Pacific-asia conference on knowledge discovery and data mining*. Springer.

Alexander, Marc. 2012. Patchworks and Field-Boundaries: Visualizing the History of English. *Pages 82–83 of: Dh*.

Alexander, Marc, & Kay, Christian. 2014. The spread of RED in the Historical Thesaurus of English. *Colour studies: A broad spectrum*, 126–139.

Anisimov, Anatoly, Marchenko, Oleksandr, Taranukha, Volodymyr, & Vozniuk, Taras. 2014. Semantic and syntactic model of natural language based on tensor factorization. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, **8455 LNCS**, 51–54.

Araujo, Miguel, Hung, San-Chuan, Araujo, Miguel, & Faloutsos, Christos. 2016. Distributed community detection on edge-labeled graphs using spark. *In: 12th international workshop on mining and learning with graphs (mlg)*, vol. 113.

Aynaud, Thomas, & Guillaume, Jean-Loup. 2010. Static community detection algorithms for evolving networks. *Pages 513–519 of: 8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks*. IEEE.

Bader, Brett W., & Kolda, Tamara G. 2007. Efficient MATLAB computations with sparse and factored tensors. *Siam journal on scientific computing*, **30**(1), 205–231.

Bagrow, James P, & Bollt, Erik M. 2019. An information-theoretic, all-scales approach to comparing networks. *Applied network science*, **4**(1), 45.

Bagrow, James P, Bollt, Erik M, Skufca, Joseph D, & Ben-Avraham, Daniel. 2008. Portraits of complex networks. *Epl (europhysics letters)*, **81**(6), 68004.

Barabási, Albert-László, & Albert, Réka. 1999. Emergence of scaling in random networks. *science*, **286**(5439), 509–512.

Bassett, Danielle S., Porter, Mason A., Wymbs, Nicholas F., Grafton, Scott T., Carlson, Jean M., & Mucha, Peter J. 2013. Robust detection of dynamic community structure in networks. *Chaos*, **23**(1).

Bastian, Mathieu, Heymann, Sebastien, & Jacomy, Mathieu. 2009. Gephi: an open source software for exploring and manipulating networks. *In: Third international aaai conference on weblogs and social media*.

Bates, Douglas, Kornblith, Simon, Noack, Andreas, Borregaard, Michael Krabbe, Bouchet-Valat, Milan, White, John Myles, Arslan, Alex, Kleinschmidt, Dave, Lendle, Sam, Mogensen, Patrick Kofod, & et al. 2019. Juliastats/glm.jl: v1.3.1. Aug.

Bezanson, Jeff, Karpinski, Stefan, Shah, Viral B, & Edelman, Alan. 2012. Julia: A fast dynamic language for technical computing. *arxiv preprint arxiv:1209.5145*.

Bonferroni, Carlo. 1950. Sulle medie multiple di potenze. *Bollettino dell'unione matematica italiana*, **5**(3-4), 267–270.

Bounova, Gergana, & de Weck, Olivier. 2012. Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles. *Physical review E*, **85**(1), 016117.

Bro, Rasmus, & Kiers, Henk A.L. 2003. A new efficient method for determining the number of components in PARAFAC models. *Journal of chemometrics*, **17**(5), 274–286.

Broccatelli, Chiara, Everett, Martin, & Koskinen, Johan. 2016. Temporal dynamics in covert networks. *Methodological innovations*, **9**(Apr.), 2059799115622766.

Broido, Anna D, & Clauset, Aaron. 2019. Scale-free networks are rare. *Nature communications*, **10**(1), 1017.

Bromberger, Seth. 2017. JuliaGraphs/LightGraphs.jl. Sept.

Burt, Ronald S. 1980. Models of Network Structure. *Annual review of sociology*.

Carpi, Laura C., Rosso, Osvaldo A., Saco, Patricia M., & Ravetti, Martín Gómez. 2011. Analyzing complex networks evolution through Information Theory quantifiers. *Physics letters a*, **375**(4), 801–804.

Carroll, J Douglas, & Chang, Jih-Jie. 1970. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, **35**(3), 283–319.

Chakraborty, Tanmoy, Ganguly, Niloy, & Mukherjee, Animesh. 2015. An author is known by the context she keeps: significance of network motifs in scientific collaborations. *Social network analysis and mining*, **5**(1), 1–21.

Choobdar, S., Ribeiro, P., Bugla, S., & Silva, F. 2012a. Comparison of Co-authorship Networks across Scientific Fields Using Motifs. *2012 IEEE/ACM international conference on advances in social networks analysis and mining*, 147–152.

Coleridge, Herbert, Furnivall, Frederick, Murray, James, March, Francis, Bradley, Henry, Craigie, William, & Onions, Charles Talbut. 2019. *Oxford English Dictionary (Online)*. Oxford University Press.

Costa, Luciano da F., Rodrigues, Francisco A., Travieso, Gonzalo, & Villas Boas, P. R. 2007. Characterization of complex networks: A survey of measurements. *Advances in physics*, **56**(1), 167–242.

Cullis, Cuthbert Edmund, *et al.* 1913. Chapter i: Introduction of rectangular matrices and determinoids. *Pages 1–21 of: Matrices and determinoids, volume i*. Cornell University Library.

Cunningham, Padraig, Harrigan, Martin, Wu, Guangyu, & O'Callaghan, Derek. 2013. Characterizing ego-networks using motifs. *Network science*, **1**(2), 170–190.

Danon, Leon, Ford, Ashley P, House, Thomas, Jewell, Chris P, Keeling, Matt J, Roberts, Gareth O, Ross, Joshua V, & Vernon, Matthew C. 2011. Networks and the epidemiology of infectious disease. *Interdisciplinary perspectives on infectious diseases*, **2011**.

De Meo, Pasquale, Ferrara, Emilio, Fiumara, Giacomo, & Ricciardello, Angela. 2012. A novel measure of edge centrality in social networks. *Knowledge-based systems*, **30**, 136–150.

Decan, Alexandre, Mens, Tom, & Grosjean, Philippe. 2018a. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical software engineering*, 1–36.

Edwards, Gemma, & Crossley, Nick. 2009. Measures and meanings: Exploring the egonet of Helen Kirkpatrick Watts, militant suffragette. *Methodological innovations online*, **4**(1), 37–61.

Erdős, Paul, & Rényi, Alfréd. 1959. On random graphs I. *Publicationes mathematicae (debrecen)*, **6**(1959), 290–297.

Fisher, Ronald A. 1925. Statistical methods for research workers oliver and boyd. *Edinburgh, scotland*, **6**.

Folino, Francesco, & Pizzuti, Clara. 2014. An evolutionary multiobjective approach for community discovery in dynamic networks. *IEEE transactions on knowledge and data engineering*, **26**(8), 1838–1852.

Freeman, Linton C. 1978. Centrality in social networks conceptual clarification. *Social networks*, **1**(3), 215–239.

Gauvin, Laetitia, Panisson, André, & Cattuto, Ciro. 2014. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *Plos one*, **9**(1).

Goyal, Palash, & Ferrara, Emilio. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-based systems*, **151**, 78–94.

Gross, Thilo, & Sayama, Hiroki. 2009. Adaptive Networks. *Pages 1–8 of:* Gross, Thilo, & Sayama, Hiroki (eds), *Adaptive Networks*. Understanding Complex Systems. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-01284-6_1.

Halko, Nathan, Martinsson, Per-Gunnar, & Tropp, Joel A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *Siam review*, **53**(2), 217–288.

Hamilton, William L, Ying, Rex, & Leskovec, Jure. 2017. Representation learning on graphs: Methods and applications. *arxiv preprint arxiv:1709.05584*.

Harshman, Richard A, *et al.* 1970. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis.

He, Jialin, & Chen, Duanbing. 2015. A fast algorithm for community detection in temporal network. *Physica A: Statistical mechanics and its applications*, **429**, 87–94.

He, Jialin, Chen, Duanbing, Sun, Chongjing, Fu, Yan, & Li, Wenjun. 2017. Efficient stepwise detection of communities in temporal networks. *Physica A: Statistical mechanics and its applications*, **469**, 438–446.

House, Thomas, Read, Jonathan M, Danon, Leon, & Keeling, Matthew J. 2015. Testing the hypothesis of preferential attachment in social network formation. *Epj data science*, **4**(1), 13.

Johnson, Samuel, & Jones, Nick S. 2017. Looplessness in networks is linked to trophic coherence. *Proceedings of the national academy of sciences*, **114**(22), 5618–5623.

Kang, U, Papalexakis, Evangelos, Harpale, Abhay, & Faloutsos, Christos. 2012. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. *Pages 316–324 of: Proceedings of the 18th ACM sigkdd international conference on knowledge discovery and data mining*. ACM.

Kashani, Zahra Razaghi Moghadam, Ahrabian, Hayedeh, Elahi, Elahe, Nowzari-Dalini, Abbas, Ansari, Elnaz Saberi, Asadi, Sahar, Mohammadi, Shahin, Schreiber, Falk, & Masoudi-Nejad, Ali. 2009. Kavosh: a new algorithm for finding network motifs. *Bmc bioinformatics*, **10**(1), 318.

Kay, Christian, Roberts, Jane, Samuels, Michael, & Wotherspoon, Irené. 2009. *Historical thesaurus of the Oxford English dictionary*. Oxford University Press.

Kluyver, Thomas, Ragan-Kelley, Benjamin, Pérez, Fernando, Granger, Brian E, Bussonnier, Matthias, Frederic, Jonathan, Kelley, Kyle, Hamrick, Jessica B, Grout, Jason, Corlay, Sylvain, *et al.* 2016. Jupyter notebooks-a publishing format for reproducible computational workflows. *Pages 87–90 of: Elpub*.

Kolda, Tamara G., & Bader, Brett W. 2009. Tensor Decompositions and Applications. *Siam review*, **51**(3), 455–500.

Krumov, L., Fretter, C., Müller-Hannemann, M., Weihe, K., & Hütt, M. T. 2011. Motifs in co-authorship networks and their relation to the impact of scientific publications. *European physical journal b*, **84**(4), 535–540.

Kumar, Sameer. 2015. Co-authorship networks: A review of the literature. *Aslib journal of information management*, **67**(1), 55–73.

Lee, Andy. 2015. Comparing deep neural networks and traditional vision algorithms in mobile robotics.

Levine, Stephen. 1980. Several measures of trophic structure applicable to complex food webs. *Journal of theoretical biology*, **83**(2), 195–207.

Li, Eldon Y., Liao, Chien Hsiang, & Yen, Hsiuju Rebecca. 2013b. Co-authorship networks and research impact: A social capital perspective. *Research policy*, **42**(9), 1515–1530.

Masoudi-Nejad, A., Schreiber, F., & Kashani, Z.R.M. 2012. Building blocks of biological networks: a review on major network motif discovery algorithms. *IET systems biology*, **6**(5), 164–174.

McKay, Brendan D., & Piperno, Adolfo. 2014. Practical graph isomorphism, II. *Journal of symbolic computation*, **60**(Jan.), 94–112.

Meira, Luis AA, Máximo, Vinícius R., Fazenda, Álvaro L., & Da Conceição, Arlindo F. 2014. Acc-motif: accelerated network motif detection. *IEEE/ACM transactions on computational biology and bioinformatics (TCBB)*, **11**(5), 853–862.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, **298**(5594), 824–827.

Molloy, Michael, & Reed, Bruce. 1995a. A critical point for random graphs with a given degree sequence. *Random structures & algorithms*, **6**(2-3), 161–180.

Montgomery, Douglas C, Peck, Elizabeth A, & Vining, G Geoffrey. 2012. *Introduction to linear regression analysis*. Vol. 821. John Wiley & Sons.

Montoya, José M, Pimm, Stuart L, & Solé, Ricard V. 2006. Ecological networks and their fragility. *Nature*, **442**(7100), 259.

Nakatsuji, Makoto. 2016. Semantic sensitive simultaneous tensor factorization. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, **9981 LNCS**, 411–427.

Newman, M. E. J. 2003. Mixing patterns in networks. *Phys. rev. e*, **67**(Feb), 026126.

Newman, M. E. J. 2010. *Networks: an introduction*. Oxford ; New York: Oxford University Press. OCLC: ocn456837194.

O'Callaghan, Derek, Harrigan, Martin, Carthy, Joe, & Cunningham, Pádraig. 2012. Network Analysis of Recurring YouTube Spam Campaigns. 531–534.

Papalexakis, Evangelos E., Faloutsos, Christos, & Sidiropoulos, Nicholas D. 2012 (July). Parcube: Sparse parallelizable tensor decompositions. *Pages 521–536 of: Joint european conference on machine learning and knowledge discovery in databases*, vol. 10. Springer.

Pastor-Satorras, Romualdo, Castellano, Claudio, Van Mieghem, Piet, & Vespignani, Alessandro. 2015. Epidemic processes in complex networks. *Rev. mod. phys.*, **87**(3), 925–979.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, **12**(Oct), 2825–2830.

Piao, Scott, Dallachy, Fraser, Baron, Alistair, Demmen, Jane, Wattam, Steve, Durkin, Philip, McCracken, James, Rayson, Paul, & Alexander, Marc. 2017. A time-sensitive historical thesaurus-based semantic tagger for deep semantic annotation. *Computer speech and language*, **46**, 113–135.

Przulj, N. 2007. Biological network comparison using graphlet degree distribution. *Bioinformatics*, **23**(2), e177–e183.

Rochat, Yannick. 2009. *Closeness centrality extended to unconnected graphs: The harmonic centrality index*. Tech. rept.

Ronqui, JRF, & Travieso, Gonzalo. 2015. Analyzing complex networks through correlations in centrality measurements. *Journal of statistical mechanics: Theory and experiment*, 9.

Rossi, Ryan A., Zhou, Rong, & Ahmed, Nesreen K. 2017. Deep Feature Learning for Graphs. 1–11.

Rosvall, Martin, Esquivel, Alcides V., Lancichinetti, Andrea, West, Jevin D., & Lambiotte, Renaud. 2014. Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications*, **5**, 1–13.

Savić, Miloš, Kurbalija, Vladimir, Bosnić, Zoran, & Ivanović, Mirjana. 2019c. Feature selection based on community detection in feature correlation networks. *Computing*.

Schieber, Tiago A., Carpi, Laura, Frery, Alejandro C., Rosso, Osvaldo A., Pardalos, Panos M., & Ravetti, Martín G. 2016. Information theory perspective on network robustness. *Physics letters a*, **380**(3), 359–364.

Schieber, Tiago A, Carpi, Laura, Díaz-Guilera, Albert, Pardalos, Panos M, Masoller, Cristina, & Ravetti, Martín G. 2017. Quantification of network structural dissimilarities. *Nature communications*, **8**, 13928.

Schult, Daniel A. 2008. Exploring network structure, dynamics, and function using NetworkX. *Pages 11–15 of: In Proceedings of the 7th Python in Science Conference (SciPy)*.

Solé, Ricard V, Corominas-Murtra, Bernat, Valverde, Sergi, & Steels, Luc. 2010. Language networks: Their structure, function, and evolution. *Complexity*, NA–NA.

Spinrad, Jeremy P. 2003. *Efficient graph representations*. American Mathematical Society.

Strang, Gilbert. 1993. *Introduction to linear algebra*. Vol. 3. Wellesley-Cambridge Press Wellesley, MA.

Strogatz, S. H. 2001a. Exploring complex networks. *Nature*, **410**(6825), 268–276.

Tahmasebi, Nina, Borin, Lars, & Jatowt, Adam. 2018. Survey of Computational Approaches to Lexical Semantic Change.

Tantipathananandh, Chayant, & Berger-Wolf, Tanya Y. 2011. Finding communities in dynamic social networks. *Proceedings - IEEE international conference on data mining, icdm*, 1236–1241.

Thompson, RM, & Townsend, CR. 2003. *Interaction Web DataBase*.

Van Eck, Nees, & Waltman, Ludo. 2009. Software survey: Vosviewer, a computer program for bibliometric mapping. *Scientometrics*, **84**(2), 523–538.

Van Loan, Charles F. 2000. The ubiquitous kronecker product. *Journal of computational and applied mathematics*, **123**(1-2), 85–100.

Van Rossum, Guido, & Drake Jr, Fred L. 1995. *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Verma, Ankita, & Bharadwaj, Kamal K. 2017. Identifying community structure in a multi-relational network employing non-negative tensor factorization and GA k-means clustering. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, **7**(1), 1–22.

Watts, Duncan J., & Strogatz, Steven H. 1998. Collective dynamics of small-world networks. *Nature*, **393**(6684), 440–442.

Wernicke, Sebastian, & Rasche, Florian. 2006. FANMOD: A tool for fast network motif detection. *Bioinformatics*, **22**(9), 1152–1153.

Wold, Svante, Esbensen, Kim, & Geladi, Paul. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, **2**(1), 37–52.

Xu, Yangyang, & Yin, Wotao. 2013. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *Siam journal on imaging sciences*, **6**(3), 1758–1789.

Yin, Guisheng, Chi, Kuo, Dong, Yuxin, & Dong, Hongbin. 2017. An approach of community evolution based on gravitational relationship refactoring in dynamic networks. *Physics letters, section a: General, atomic and solid state physics*, **381**(16), 1349–1355.

Yu, Wenchao, Aggarwal, Charu C., & Wang, Wei. 2017. Temporally Factorized Network Modeling for Evolutionary Network Analysis. 455–464.

Zhuang, Honglei, Sun, Yihan, Tang, Jie, Zhang, Jialin, & Sun, Xiaoming. 2013. Influence Maximization in Dynamic Social Networks. *Pages 1313–1318 of: Data mining (icdm), 2013 IEEE 13th international conference on*.

Part II

Node characterisation from local topology

Chapter 3

Graphlets: colour, profiles and node characterisation

"If I could remember the names of all these particles, I'd be a botanist"

— Enrico Fermi

Node characterisation, as introduced in section 2.5, describes single nodes such that they can be compared with other nodes in a network or, perhaps, with other nodes in other networks. Here, we focus on characterising nodes from their local topology as opposed to being concerned with any dynamical equations governing state vectors associated with those nodes.

Of the works summarised in section 2.5, we believed that graphlet profiles (Cunningham *et al.*, 2013) showed the most promise as their technique could be extended to use graph colour to denote the ego and thus retain the information about the position of a node under consideration within its local topology. Additionally, this technique could serve as an input for the decompositions discussed in part III.

This part is a discussion of our contributions to this technique and includes an example application using coloured graphlet profiles to investigate the relationship between local topology on academic collaboration networks and academic career length. A paper based on this work was presented at a conference for which proceedings are available (Blanthorn & Navarro-López, 2019a).

The rest of this chapter and part is organised as follows:

- section 3.1 provides details on extensions to the algorithms used to create the coloured graphlet profiles of a network,

- section 3.2 describes how we apply graphlet profiles to the ego-networks of nodes to characterise them,
- and chapter 4 shows how this technique can be used to explore a real-word collaboration network. Importantly, we find that our graphlet profiles are not degenerate to the standard node centralities or monochromatic graphlet profiles—i.e. the new technique is worthwhile.

3.1 Coloured graphlet profiles

As described in section 2.6.7, the graphlet profile of a graph is a mapping from automorphism groups corresponding to a certain number of nodes to the number of times subgraphs belonging to those automorphism groups appear in a graph.

Coloured graphlet profiles are much less well supported in software than monochromatic graphlet profiles. Of the algorithms introduced in section 2.6.7 on page 41, only FANMOD (Wernicke & Rasche, 2006) supports coloured graphlets. However, FANMOD is far from the state-of-the-art in graphlet enumeration; it is many times slower than more modern routines (Masoudi-Nejad *et al.*, 2012).

Kavosh is substantially faster than FANMOD (Kashani *et al.*, 2009) and uses nauty (McKay & Piperno, 2014) to tally graphlets by their automorphism group. Nauty has built-in support for graph colour. There are other graphlet enumeration routines which are quicker for monochromatic graphs (Masoudi-Nejad *et al.*, 2012), but adapting these for coloured graphs seemed like a far larger undertaking. It was therefore decided that adapting Kavosh to work with coloured graphs struck a good balance between being a useful contribution to the field and the amount of time required to design and build. In our testing, the original Kavosh implementation suffered from memory leaks. As the algorithm is quite short, we re-implemented it from scratch in Julia, which required a number of changes which we will enumerate in this chapter.

3.1.1 Calling nauty from Julia

We wrote a small package to call nauty from Julia and a small type for coloured graphs as the first step before Kavosh could be reimplemented. The most important elements of these packages are enumerated here:

- Creation of a partition array of node labels such that all nodes of the same colour are in a single contiguous block, as shown in algorithm 3.1.
- Creation of labels for nauty describing where the changes in colour are—an array where zeroes denote that a node is the same colour as the preceding node in the partition, and non-zeroes denote a change in colour, as shown in algorithm 3.1.

Algorithm 3.1 Implementation in Julia of labelling and partition for use of nauty on coloured graphs

```
function nautylabelspartition(g;implicit_colour=false,ego=0)
    colorsarray::Array{Array{Int64,1},1} = !implicit_colour?
                    # if the colours are explicitly specified
                    # create the array of colours from that
                    [collect(filter_vertices(
                         g,(graph, vertex) -> begin
                             get(
                                 props(graph, vertex), :color, ""
                             ) == c
                         end
                    )) for c in colors(g)]:
        # otherwise denote the ego as a different colour to
        # the alters
        ego==1 ? [[1],collect(2:number_of_vertices(g))]:
        ego==0 ? [collect(1:number_of_vertices(g))]:
        ego==number_of_vertices(g) ? [collect(1:ego-1),[ego]]:
        [[ego],[1:ego-1..., ego+1:number_of_vertices(g)...]]

    # flatten the array and subtract one from all indices
    # as nauty makes the unfortunate choice of numbering
    # nodes from 0.
    labels = vcat(colorsarray.-1...)

    # set the last element of each colour block to be zero
    # to denote a change in colour
    partition = vcat(
        [begin
            z[end]=0; z
        end for z in ones.(Cint,size.(colorsarray))]...
    )
    return (labels,partition)
end
```

- The ability to call nauty from Julia, shown in algorithm 3.2.

3.1.2 Kavosh adaptation

The key changes made to the algorithm to support colour and work in Julia are as follows:

- Hashing and equality operators for Julia on the nauty return values to enable us to use them as keys for a dictionary, as shown in algorithm 3.3.

Algorithm 3.2 Interface to nauty from Julia

```
struct nautyreturn
    canong::NautyGraph
    partition::Array{Cint}
end

# Get the canonical form of a graph from nauty given the colouring
function baked_canonical_form_color(g, labelling, partition)
    g = lg_to_nauty(g)
    (num_vertices, num_setwords) = size(g, 1, 2)
    stats = statsblk()
    outgraph = zero(g)
    orbits = zero(labelling)
    ccall((:baked_options_color, LIB_FILE), Cvoid,
            (NautyGraphC, Ptr{Cint},
             Ptr{Cint}, Ptr{Cint},
             Ref{statsblk}, Cint,
             Cint, NautyGraphC),
        g, labelling, partition,
        orbits, stats, num_setwords,
        num_vertices, outgraph
    )
    return nautyreturn(outgraph, partition)
end

function nauty(g; egonet=false, ego=0 )
    labels, partition = nautylabelspartition(
        g;model_t=egonet,ego=ego
    )
    nautyrtn = baked_canonical_form_color(g,labels,partition)
    return nautyrtn
end
```

Algorithm 3.3 Implementation of equality and hashing operators on nauty return values in Julia

```
function Base.hash(m::nautyreturn)
    return hash((m.canong,m.partition))
end

function Base.:(==)(a::nautyreturn, b::nautyreturn)
    # two canonical graphs are the same if they have the same layout
    # and the same colourings
    return a.canong == b.canong && a.partition == b.partition
end
```

- As Kavosh iterates through each unique (per node labels) induced subgraph, we increment the counter corresponding to the key corresponding to the nauty return value for that graph. This routine is not shown here for brevity; it may be found online from links in chapter 'Availability of data and material' on page 14.

3.1.3 Limitations

Our implementation of Kavosh only supports graphlets of at least size two and graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a maximum of two colours, i.e.[1] $|\{\pi(i) : i \in \mathcal{V}\}| = 2$, where $\pi(i)$ is the colour of the node i. One of these colours must have a maximum of one node, that is, if we let $\Pi = \{\pi(i) : i \in \mathcal{V}\}$, then

$$\min(\{|\{i : \pi(i) = a\}| : a \in \Pi\}) \leq 1.$$

This limitation is because nauty has no memory of the entire graph and so it cannot tell the difference, for example, between a red node connected to a green node and a yellow node connected to a blue node on separate calls.

Support for all coloured graphs could be added by adding a new disconnected node of each colour to every subgraph as it is sent to nauty acting as a key. This was not necessary to distinguish between the single ego and alters in ego-networks.

3.2 Node characterisation from graphlet profiles of ego-networks

Graphlets, or induced motifs, have long been used to find important medium-scale structures in directed networks (Milo *et al.*, 2002). The works introduced in section 2.6.7 such as Cunningham *et al.* (2013) have a clear area for improvement: they do not retain information about the location of the node of interest within its local topology. Our coloured graphlet profiles offer a clear way to retain this information from the employment of coloured ego-networks, where the ego is given a distinct colour from the alters, as the location of the node of interest (the ego) is denoted by its colour. Here we present a method using the composition of coloured graphlets in ego-networks to characterise nodes:

1. For each node, extract its n-ego-network

2. Compose coloured graphlet profiles for the ego-network of each node, where the ego is one colour and the alters are another, for each size of graphlet considered

[1] here we use the notation $\{i : i \in S\} \equiv \{i \mid i \in S\}$ to prevent ambiguity with the cardinality of a set

3. For each node, combine the graphlet profiles into a single vector to characterise that node; these vectors can then be used in further analysis such as multi-linear regression as described in section 4.1.5.

3.2.1 Ego-networks and coloured graphlets

The ego in each ego-network is given a colour distinct from all of the other nodes, used in the coloured subgraph matching. This allows us to understand the importance of an ego within a graphlet, if it is there at all.

Each graphlet is given a label consisting of a decimal followed by an alphabetic character. These labels follow the convention that the number before the decimal place is the number of nodes in the graphlet, the number after the decimal place labels the monochromatic automorphism group in ascending order of the Shannon entropy of their degree distributions (so less-uniform distributions have a higher label (Costa *et al.*, 2007, pp. 191-192)) and the character signifies colour differences between graphlets within the same monochromatic automorphism group, as shown in figure 3.1.

3.2.2 From graphlet profiles to node characterisation

Similar to previous work (Cunningham *et al.*, 2013) as described in section 2.6.7, we normalise each graphlet profile of graphlets with n nodes, $\mathbf{g_n}$, by the total number of induced subgraphs of size n in the ego-network,

$$\hat{g}_{ni} = \frac{g_{ni}}{\sum_i g_{ni}} \ , \tag{3.1}$$

where each ni corresponds to a specific automorphism group i of size n. We normalise predominantly to allow for meaningful comparison between networks of different sizes.

This is sufficient compared to the much more computationally expensive significance profiles approach of comparing the counts to the the counts in ensemble of configuration models, calculating a so-called z-score (Milo *et al.*, 2002; Kashani *et al.*, 2009; Meira *et al.*, 2014) as we are later comparing these profiles with many other profiles: we are not concerned with how they compare to configuration models. We want to know how they differ from other networks in same domain.

After normalisation, we combine the graphlet profiles for each graphlet size considered in a single vector, $\mathbf{g}$, where each dimension corresponds to a specific graphlet such that profiles between nodes are directly comparable.

3.3 Conclusion

We have described a technique for characterising nodes and networks from coloured graphlet profiles, extending previous monochromatic treatments. We will employ it in part III as part

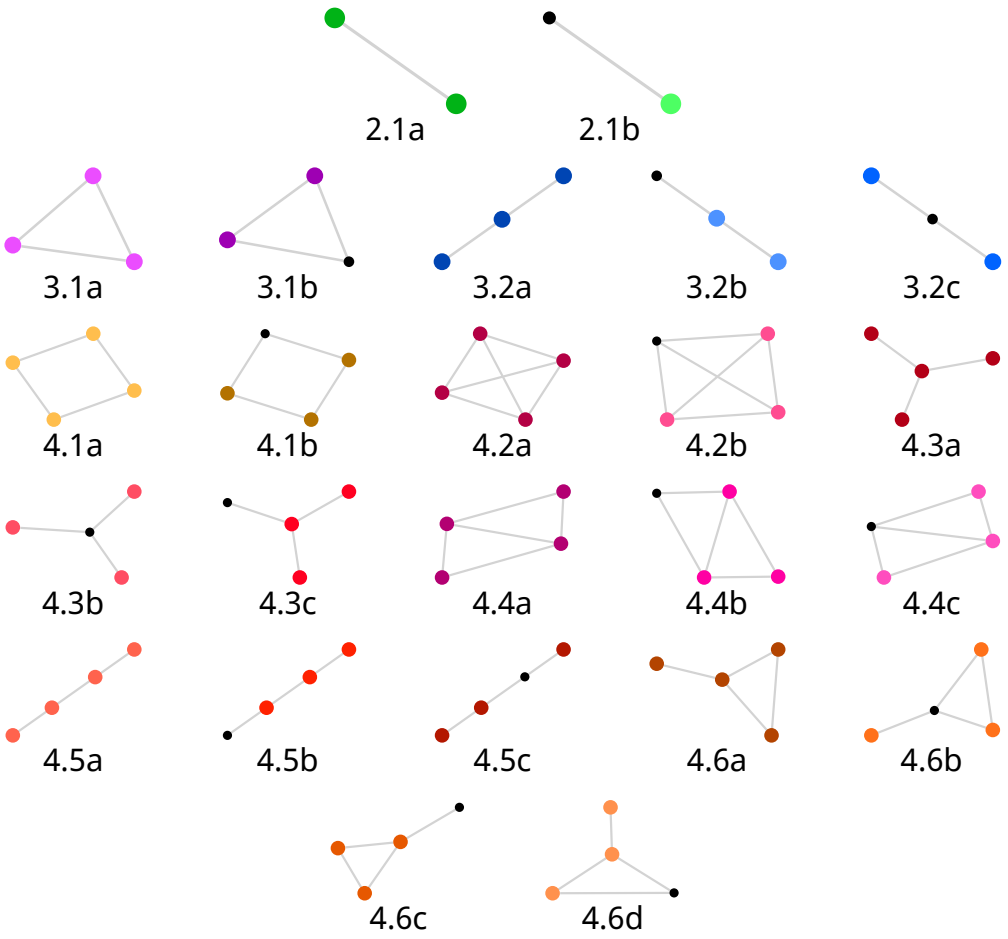

Figure 3.1 – All possible undirected graphlets with at most a single differently-coloured node (i.e, the possible graphlets within ego-networks where the ego is one colour and the alters are another colour) up to size 4. The smaller black node corresponds to the ego, if it exists, and the coloured nodes are the alters. The graphlets are labelled as described in Sect. 3.2.1; the colours are chosen such that similar graphlets have similar colours

of a data matrix to characterise whole networks. In the next chapter, we demonstrate its usefulness compared to previous monochromatic treatments and the standard node centralities in one example application on a real-world piecewise static network.

Bibliography

Blanthorn, Oliver, & Navarro-López, Eva. 2019a. Coloured graphlet profiles as a predictor of career length in scientific co-authorship networks. *In: 4th World Conference on Complex Systems: Emergence, self-organization, nonlinear dynamics and complexity*. Institute of Electrical and Electronics Engineers.

Costa, Luciano da F., Rodrigues, Francisco A., Travieso, Gonzalo, & Villas Boas, P. R. 2007. Characterization of complex networks: A survey of measurements. *Advances in physics*, **56**(1), 167–242.

Cunningham, Padraig, Harrigan, Martin, Wu, Guangyu, & O'Callaghan, Derek. 2013. Characterizing ego-networks using motifs. *Network science*, **1**(2), 170–190.

Kashani, Zahra Razaghi Moghadam, Ahrabian, Hayedeh, Elahi, Elahe, Nowzari-Dalini, Abbas, Ansari, Elnaz Saberi, Asadi, Sahar, Mohammadi, Shahin, Schreiber, Falk, & Masoudi-Nejad, Ali. 2009. Kavosh: a new algorithm for finding network motifs. *Bmc bioinformatics*, **10**(1), 318.

Masoudi-Nejad, A., Schreiber, F., & Kashani, Z.R.M. 2012. Building blocks of biological networks: a review on major network motif discovery algorithms. *IET systems biology*, **6**(5), 164–174.

McKay, Brendan D., & Piperno, Adolfo. 2014. Practical graph isomorphism, II. *Journal of symbolic computation*, **60**(Jan.), 94–112.

Meira, Luis AA, Máximo, Vinícius R., Fazenda, Álvaro L., & Da Conceição, Arlindo F. 2014. Acc-motif: accelerated network motif detection. *IEEE/ACM transactions on computational biology and bioinformatics (TCBB)*, **11**(5), 853–862.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, **298**(5594), 824–827.

Wernicke, Sebastian, & Rasche, Florian. 2006. FANMOD: A tool for fast network motif detection. *Bioinformatics*, **22**(9), 1152–1153.

Chapter 4

Career length in academic collaboration networks

"Time is a great teacher, it is said; the misfortune is that he is a brutal teacher who kills all his students"
— Hector Berlioz

Our technique introduced in chapter 3.2 is used here to predict the numbers of years researchers are active from their temporal collaboration networks. We compare our success with simpler metrics; we find that the use of coloured graphlets improves predictive performance compared to colour-blind graphlets; that 4-star graphlets centred on an author early in their career are predictors of an eventual long career; and that this effect is not degenerate to centralities, i.e. graphlets have a contribution to career length which is not explained by other centralities.

We compare our coloured graphlet profiles to other methods of characterisation of nodes, namely the standard centralities such as node degree, betweenness centrality and the local clustering coefficient. We apply this node characterisation technique in a specific application: predicting the 'lifetime' of researchers from the shape of their collaborations.

The research lifetime of researchers, that is, the number of years they are active publishing research, follows a power-law, so the overwhelming majority of researchers have short research lifetimes, as shown in figure 4.1 and found in other research (Milojević *et al.*, 2018). It has been suggested that this distribution is due to the voluntary career changes of PhD students (Biryukov & Dong, 2010), but this would imply a bi-modal distribution which we do not see in figure 4.1. However, other research has found that there is evidence that lifetime distributions are often 'U'-shaped, which are clearly bi-modal (Fedotov & Stage, 2017). This could be consistent with our data as our distributions are truncated as detailed in section 4.1.6. 53% of all UK workers do not wish to change career (Priddy, 2015), suggesting that a large proportion

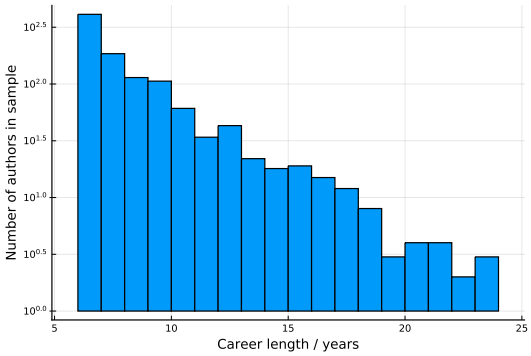

Figure 4.1 – A log-lin histogram of showing the distribution of researcher lifetimes in the subsample of the Arnetminer co-authorship dataset Zhuang *et al.* (2013) used throughout this chapter. The lowest career length is 5 years and the highest career length is 23 years. Both of these artefacts are due to choices we have made to ensure we do not have an incomplete picture of any careers

of career changes away from academia are involuntary. Our motivation is to find information that could help early career researchers extend their careers in academia, as many of them clearly wish to do so. The shape of a collaboration network is relatively easy to change as one chooses who to collaborate with. Our work aims to uncover whether particular shapes of research collaboration predict career longevity and to explore the importance of colour, or nodes which fall into discrete categories, in ego-networks.

In previous work, uncoloured graphlets in co-authorship networks have been used to predict success in terms of citation count (Krumov *et al.*, 2011). It was found that 'box motifs', i.e. 4-cycles, were beneficial. Whether this effect could be better explained by other centralities was not explored. More information on academic collaboration networks was given in section 2.9.3 on page 50.

The rest of this chapter is organised as follows: section 4.1 explains our methodological contributions; section 4.2 demonstrates the use of our technique with a real-world application; and section 4.3 summarises our results and possible directions for future work.

In our application, the ego is the author whose lifetime we predict and the alters are their co-authors, their co-authors' co-authors and so on.

4.1 Methods

We predict the lifetimes of nodes in an evolving complex network using a simple linear regression model that combines all of our features. This allows us to disregard degenerate features. Our features consist of graphlet profiles and the standard node centralities. These methods are explored in full in this subsection.

4.1.1 Ego-networks and coloured graphlets

We follow the technique introduced in section 3.2.1. We chose to consider the 3-ego-networks of each node, where 3 corresponds to the maximum distance from the node of interest. Three was chosen as it was found that any more than three did not improve predictions and led to somewhat increased computational complexity, considerably so in the worst cases; fewer than three would have meant that some 4-graphlets were underrepresented.

4.1.2 Graphlet profiles

The graphlet profiles are calculated as in section 3.2.2 with the following caveat: we produce coloured and uncoloured graphlet profiles and compare them; we refer to them as the coloured and uncoloured or monochromatic treatments. Monochromatic graphlets can appear in both types of profile but we refer to them with a trailing 'a' in the coloured treatment to signify that they represent the complete absence of the ego. Since we use the graphlet profile as part of multilinear regression, we in effect compare the graphlets from each ego-network with every other ego-network to determine what is out of the ordinary and what is normal, further suggesting that our choice of normalisation in section 3.2.2 is sound.

Graphlet profiles for graphlets of size $n = 2, 3, 4$ were calculated. Larger graphlets are not used due to their increased computational complexity and our experimental findings that they did not improve predictions, in agreement with previous studies such as Cunningham *et al.* (2013).

4.1.3 Other centralities considered

Table 4.1 summarises all of the features we consider in this work, many of which were introduced in sections 2.5 and 2.6. We refer to all of the features that are not graphlet profiles as the 'standard centralities', but not all of the features are centralities in the traditional sense as they characterise graphs rather than nodes. However, because we are using them to describe the ego-network of specific nodes, they can in some sense be considered to characterise those nodes, so we take the liberty of referring to them as centralities.

4.1.4 Estimation of confidence intervals

We estimate errors via a 'bootstrap method': the standard error is calculated for the correlation with lifetime for ten subsamples of the graphlet profiles. This error is then scaled by the square root of the number of subsamples. The error bars plotted correspond to twice the standard error and are therefore approximations of the 95% confidence interval, assuming a normal distribution.

Table 4.1 – Features used to characterise egos in ego-networks. See section 2.5 for more detail

Feature	Description	Range
Betweenness[a]	Prop. of shortest paths passing through ego	$[0, 1]$
Degree[a]	Number of edges attached to ego	$[0, \infty)$
Closeness[a]	Reciprocal of distance of ego to alters	$(0, 1]$
Clustering	Ratio of triangles present to possible triangles	$[0, 1]$
Ego-clustering	Clustering of 1-ego-network of ego	$[0, 1]$
Edges	Number of edges in ego-network	$[0, \infty)$
Graphlet profile	Normalised graphlet counts as described	$[0, 1]^d$

[a] Commonly known with the suffix 'centrality', omitted here for brevity. NB: ^{d} is the number of unique graphlets and is not a footnote; it is equal to 24 in the coloured treatment and 9 in the monochromatic treatment

4.1.5 Multiple linear regression

As introduced in section 2.10.1, multiple linear regression is a simple method for investigating the relationship between many continuous so-called explanatory variables x_i and a single dependent variable, y, (Montgomery *et al.*, 2012) with

$$y = \Sigma_i \beta_i x_i.$$

The explanatory variables are the features enumerated in table 4.1 and the dependent variable in our application is the research lifetime of the authors. The β_{reduced} coefficients,

$$\beta_{i_{\text{reduced}}} = \frac{\beta_i}{\text{std}(x_i)},$$

where std is the standard deviation of the variable x_i, are particularly important for our application as the variables have very different ranges, as shown in table 4.1.

We apply a Bonferroni correction (Bonferroni, 1950) as explained in section 2.10.3 to ensure that the large number of statistical tests we perform are valid. Since we have 30 features—each of the items in table 4.1, our corrected significance level corresponding to a 95% confidence level is approximately 0.0017. The software package GLM.jl (Bates *et al.*, 2019) was used for prediction.

4.1.6 Experimental design and data

The dataset chosen was Arnetminer's disambiguated temporal collaboration network (Zhuang *et al.*, 2013). The window size on this dataset is 3 years: each collaboration is instantaneous in the original datasets, but Arnetminer expands these collaborations to exist for three years past this. The resolution is a single year, with data for every year from 1986

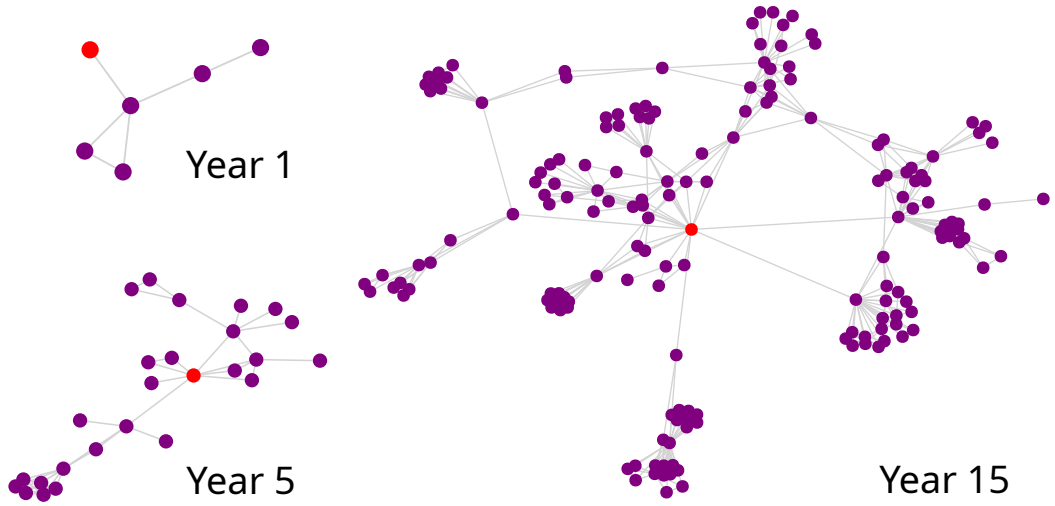

Figure 4.2 – The 3-ego-networks, anti-clockwise from top-left, of one researcher's first, fifth and fifteenth year in their career as a researcher. The ego is shown in red; the alters are shown in purple

up to and including 2012. The growth of a particularly successful researcher's collaboration network is shown in figure 4.2.

The data were cleaned in order to consider researchers for whom we have a full picture of their careers. Any researchers who were active in the first year of the dataset were removed to exclude researchers who started their careers before the data began and any researchers still active at the end of the dataset, with the assumption that these researchers may not have finished their careers yet, were also removed. This eliminated approximately 50% of the dataset. 10,000 authors were then sampled to reduce the computational complexity. Finally, any authors that did not have careers of at least 5 years were discarded—effectively discarding all PhD students—as this was found to make prediction easier with stronger correlations, leaving us with about 2,000 authors. The lifetime distribution is shown in figure 4.1.

We postulate the following:

Hypothesis 1 *Some components of graphlet profiles of researchers' co-authorship ego-networks at an early stage of their careers correlate with their career lengths, after controlling for potentially confounding variables.*

Hypothesis 2 *Graphlet profiles coloured to distinguish egos from the alters correlate more strongly with career length.*

Finally, we conclude this section with a summary of the experimental design explained in this section:

1. clean data—ensure that we have a complete life cycle for all nodes considered,

2. choose a subsample of these nodes to limit computation time,
3. calculate lifetimes of the chosen nodes,
4. extract the 3-ego-networks of each node at a specific early stage in their life cycle,
5. extract features: calculate centralities and graphlet profiles on these ego-networks,
6. calculate the linear correlation of features with lifetime,
7. check the pairwise correlation between features to check for possible degeneracy,
8. and finally, perform multiple linear regression to determine whether correlations are better explained by other features.

4.2 Results

4.2.1 Correlation of features with lifetime

The correlation of each graphlet with career longevity is shown in figure 4.4. There are no strong correlations, but we can see that 4.5c, 3.2c, 4.3b and 4.6b have weak positive correlations. If one cross-references with figure 4.3, it can be seen that these are the graphlets where the ego is central and of higher degree than its neighbours. Contrary to what one might expect, 4.2b and 3.1b, the all-to-all graphlets which in some sense correspond to 'maximum collaboration', have a very weak negative correlation with lifetime.

On the whole, the uncoloured treatment has weaker correlations with lifetime and misses out on some subtlety: where monochrome graphlets have equivalent coloured graphlets that are both negatively and positively correlated with lifetime, the net effect is to have an overall correlation which is closer to zero, which could be misleading. This is especially evident with graphlet 4.3: 4.3b is one of the strongest positive predictors of lifetime, but 4.3a is negatively correlated in the coloured treatment, albeit not significantly so. This leads to 4.3 having no overall correlation with lifetime in the uncoloured treatment, which could explain why previous work found this graphlet to be unremarkable (Krumov *et al.*, 2011, where our 4.3 graphlet corresponds to their 'Motif 3').

The higher correlation with lifetime for 3.2c and 4.3b could be because they are proxies for the number of papers published; 4.5c could be a proxy for the number of papers published by your collaborators. The fact that the correlation is stronger than that of degree centrality could be because degree centrality is a poor measure of the number of papers published in a co-authorship network; the negative correlation of the clustering coefficients is further evidence in favour of this as a higher number of authors per paper will result in a higher clustering coefficient.

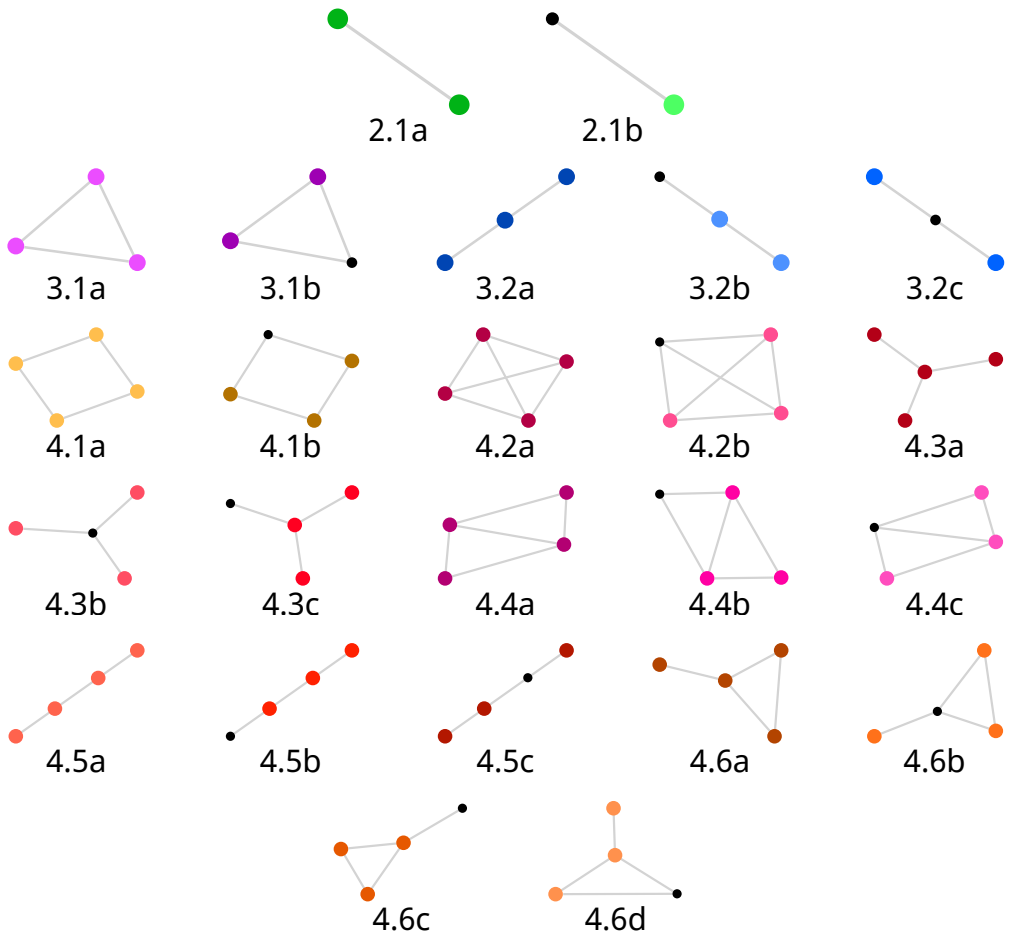

Figure 4.3 – All possible undirected graphlets with at most a single differently-coloured node (i.e, the possible graphlets within ego-networks where the ego is one colour and the alters are another colour) up to size 4. The smaller black node corresponds to the ego, if it exists, and the coloured nodes are the alters. The graphlets are labelled as described in Sect. 3.2.1; the colours are chosen such that similar graphlets have similar colours. This figure is identical to figure 3.1 and is included again here for the benefit of the reader

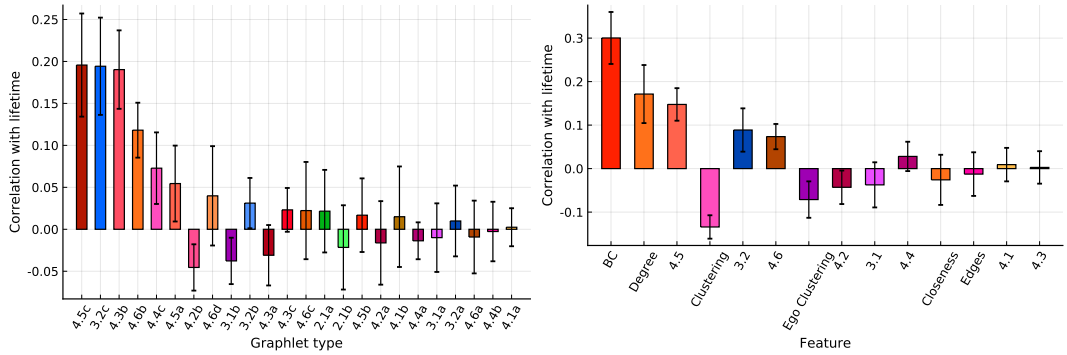

Figure 4.4 – The correlation of each graphlet frequency with lifetime for graphlets in the coloured treatment (left) and uncoloured treatment (right). The right-hand figure also includes the standard node centralities

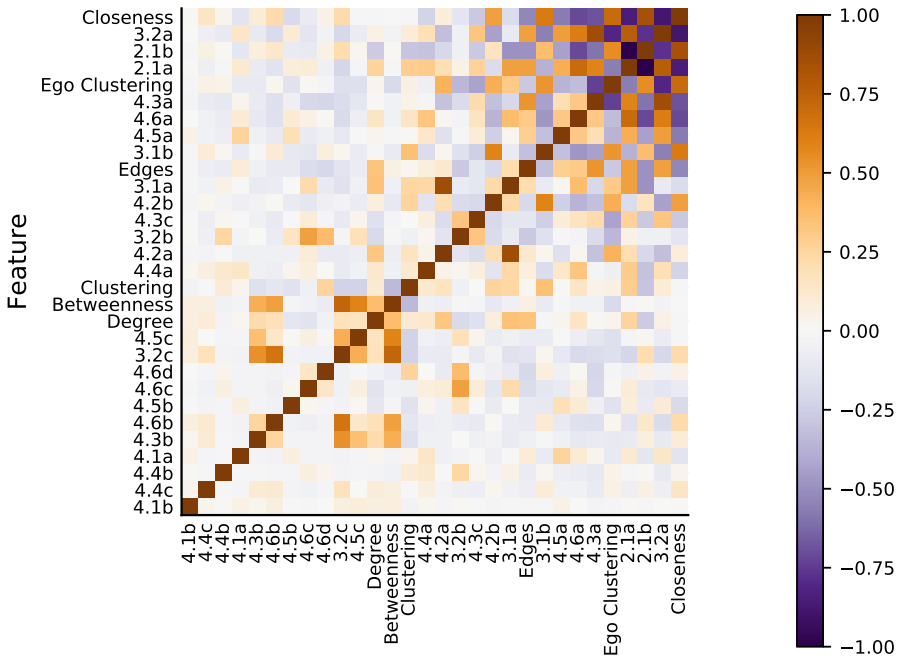

Figure 4.5 – A pairwise correlation matrix of the various features on the subsample of co-authorship 3-ego-networks

To investigate whether graphlet profiles add any new insight compared to the standard node centralities, pairwise correlation analysis on the graphlet profile components and the centralities was performed, as shown in figure 4.5. It can be seen that the monochromatic graphlets 4.5a, 4.3a, 4.6a, 3.2a and 2.1a are closely related, which is as expected as they are all subgraphs of 4.6a. Of particular interest is the that graphlets which correlate with lifetime also correlate with the various centralities: 4.3b, 4.5c, 4.6b and 3.2c all correlate strongly with betweenness centrality. Generally, this correlation plot was unsurprising and confirmed the need for further analysis to rule out degeneracies.

4.2.2 Prediction of researcher lifetimes

Due to the high pairwise correlations between some of the centralities and the graphlets that were most highly correlated with lifetime, we tested for degeneracy amongst the features by performing linear regression analysis. The results can be seen in table 4.2 and table 4.3. Graphlet profile components with particularly insignificant results have been omitted for brevity. Many of the most strongly correlated graphlets with lifetime, 4.5c, 3.2c and 4.6b, were found to be degenerate to betweenness centrality. It can be seen that many of the centralities are much more significant than the graphlet profile components.

Comparing table 4.2 with table 4.3, we can accept hypothesis 2: on the whole, the coloured graphlet profiles have more significant and stronger relationships with lifetime than in the monochrome treatment. We can also accept hypothesis 1: graphlet 4.3b is strongly and signific-

Table 4.2 – Regression analysis for graphlets in the monochromatic treatment versus research lifetime

Feature	β	Std.Error	t value	Pr($>\|t\|$)
3.1a	0.4	0.45	0.9	0.3688
4.3a	1.8	0.78	2.3	0.0227
4.5a	2.9	0.92	3.1	0.0018
Betweenness	3.9	0.5	7.7	$< 1e-13$
Clustering	-0.69	0.22	-3.2	0.0016
Degree	0.1	0.024	4.4	$< 1e-4$
Edges	-0.0026	0.00086	-3.0	0.0027

The features are described in detail in table 4.1; for the meanings of the other columns see section 4.1.5.

Table 4.3 – Regression analysis for coloured graphlets versus research lifetime

Feature	β	β_{reduced}	Std.Error	t value	Pr($>\|t\|$)
3.1a	1.8	10.0	6.9	1.5	0.1385
3.1b	0.95	4.0	1.6	2.6	0.0095
4.3a	-4.1	-0.95	3.4	-1.2	0.2280
4.3b	10.0	0.34	2.7	3.8	0.0001
4.3c	1.3	0.077	1.8	0.69	0.4882
Betweenness	4.2	0.74	0.75	5.7	$< 1e-7$
Clustering	-0.82	-0.33	0.23	-3.5	0.0005
Degree	0.12	0.42	0.028	4.3	$< 1e-4$

The features are described in detail in table 4.1; for the meanings of the other columns see section 4.1.5

antly linked with career lifetime, even after accounting for degree and betweenness centrality, with which it is correlated. It is displayed in figure 4.6. The large coefficient suggests that the graphlet's presence could explain a large amount of research lifetime, which could warrant further investigation. There are no other graphlets which are significantly linked to lifetime.

Some of the centralities are highly significant; betweenness centrality in particular has a very significant link, although it should be noted that it can only explain 4 years of lifetime. The significance of betweenness centrality contradicts earlier research, although the significant but small influence of degree on lifetime is in agreement (Li *et al.*, 2013b).

Curiously, graphlets 4.1a and 4.1b do not appear to have any link with career length, despite their previously being found to be important for success as measured by citation count (Krumov *et al.*, 2011). This distinction could be because that previous work looked at motifs (i.e, all subgraphs, not just induced ones), but one would expect to see the 4.2 and 4.4 graphlets,

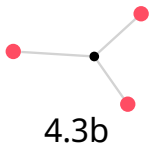

Figure 4.6 – Graphlet 4.3b from figure 4.4, the only graphlet found to have a significant link with career lifetime, after controlling for many standard node centralities

of which 4.1 is a subgraph, correlating positively with lifetime with some certainty, which is not the case. 4.2 seems to be negatively correlated with lifetime, if at all.

Finally, it is pertinent to note that none of the features tested have a very large impact on career-length, especially when considering the reduced coefficients, suggesting that there is a large element of random chance, or that some important factor has been left out of our analysis, which is consistent with other work (Milojević *et al.*, 2018).

4.3 Conclusions

We have demonstrated that coloured graphlet profiles can be predictors for a real-world characteristic of nodes and this predictive power is not explained by the other standard centralities; particularly, degree centrality and betweenness centrality.

In our application, the 4.3b graphlet, the 4-star centred on the author of interest, was found to be the strongest predictor of career length in researchers in the fifth year of their career. Therefore, the best advice we can give to young researchers is that they should try to collaborate broadly with people who do not collaborate with each other. However, the correlations we have found do come with the usual caveat: there could be some other confounding variables that are at play—for example, it could be that the kind of people who collaborate with lots of people are interdisciplinarians who are particularly intelligent and therefore were more likely to be successful anyway, or that the success of the 4.3b graphlet compared to degree centrality could be because 4.3b requires more papers to be published than degree centrality; indeed, other work has found that the number of papers published is one of the main factors in determining career-length (Milojević *et al.*, 2018). This could be solved by a more thorough study with more information about each author. It could also be beneficial to use a dataset spanning more time as our current approach limits the maximum career length to 23 years—we are forced disregard any researcher with a career longer than that in order to ensure that we have a complete picture of all the careers of the researchers we consider.

Finally, it would be interesting to see if our technique has similar or greater success in other applications. Analysis of any network where there are interesting properties on the nodes which are linked to the medium scale structure of the network could plausibly benefit; for example, it could be used to detect groups of users on a social network spreading information

in an inauthentic fashion.

Our main conclusion is that when characterising a node through its ego-network, it can be useful to retain the information as to which node is the ego and which are the alters. This information can be used by existing graph packages by colouring the ego one colour and the alters another.

Bibliography

Bates, Douglas, Kornblith, Simon, Noack, Andreas, Borregaard, Michael Krabbe, Bouchet-Valat, Milan, White, John Myles, Arslan, Alex, Kleinschmidt, Dave, Lendle, Sam, Mogensen, Patrick Kofod, & et al. 2019. Juliastats/glm.jl: v1.3.1. Aug.

Biryukov, Maria, & Dong, Cailing. 2010. Analysis of computer science communities based on DBLP. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, **6273 LNCS**, 228–235.

Bonferroni, Carlo. 1950. Sulle medie multiple di potenze. *Bollettino dell'unione matematica italiana*, **5**(3-4), 267–270.

Cunningham, Padraig, Harrigan, Martin, Wu, Guangyu, & O'Callaghan, Derek. 2013. Characterizing ego-networks using motifs. *Network science*, **1**(2), 170–190.

Fedotov, Sergei, & Stage, Helena. 2017. Anomalous metapopulation dynamics on scale-free networks. *Physical review letters*, **118**(9), 098301.

Krumov, L., Fretter, C., Müller-Hannemann, M., Weihe, K., & Hütt, M. T. 2011. Motifs in co-authorship networks and their relation to the impact of scientific publications. *European physical journal b*, **84**(4), 535–540.

Li, Eldon Y., Liao, Chien Hsiang, & Yen, Hsiuju Rebecca. 2013b. Co-authorship networks and research impact: A social capital perspective. *Research policy*, **42**(9), 1515–1530.

Milojević, Staša, Radicchi, Filippo, & Walsh, John P. 2018. Changing demographics of scientific careers: The rise of the temporary workforce. *Proceedings of the national academy of sciences*, **115**(50).

Montgomery, Douglas C, Peck, Elizabeth A, & Vining, G Geoffrey. 2012. *Introduction to linear regression analysis*. Vol. 821. John Wiley & Sons.

Priddy, Steve. 2015. *LSBF Careers Report: Are UK professionals looking to change careers?* Tech. rept. London School of Business & Finance, London.

Zhuang, Honglei, Sun, Yihan, Tang, Jie, Zhang, Jialin, & Sun, Xiaoming. 2013. Influence Maximization in Dynamic Social Networks. *Pages 1313–1318 of: Data mining (icdm), 2013 IEEE 13th international conference on*.

Part III

Network characterisation through matrix decompositions

Chapter 5

Construction of data matrices and matrix decomposition

"Things will explain themselves to you if you just know how to listen"

— Dad

The dimensionality, or the number of degrees of freedom, in the topology of complex networks naively appears to be very high—of the order of N^2, where N is the number of nodes in a network. In reality, many of those dimensions are redundant. Evidence for this was found in the previous part in figure 4.5 where it was shown that 'features' of networks are highly correlated, in line with previous work (Bounova & de Weck, 2012; Ronqui & Travieso, 2015). It stands to reason that, if one could combine the dimensions in such a way that only the 'true' degrees of freedom for complex networks were dimensions in some reduced space, that space could make characterising complex networks substantially easier. A more general form of this problem is well-recognised in many other fields; techniques for addressing it are termed 'dimension reductions', as was explored in section 2.6.8.

In this part, we build on work introduced in sections 2.6 and 2.6.8. There is some prior work applying dimension reductions to the task of characterising whole networks; similar techniques were employed by Costa *et al.* (2007, pp. 221-234), Bounova & de Weck (2012) and Ronqui & Travieso (2015). The graphlet profiles explored in the previous part are also explored here.

The first task in finding a low-dimensional representation of a network is to find a high-dimensional representation of a network to which dimension reductions may be applied. That is the focus of this chapter, which is split into two sections: section 5.1 introduces the process by which we construct data matrices and section 5.2 briefly discusses how we apply dimension reductions to these.

5.1 Construction of a data matrix for network characterisation

The standard form for high-dimensional representations of observations is the data matrix. 'Data matrix' is a term from the field of machine learning which we adopt; each column is termed a 'feature' and each row is called an 'example'. In our application, each example corresponds to a network. Each feature is an observable property of a network which is directly comparable across all networks considered; for example, one feature might be 'first moment of the degree distribution' and another might be 'network diameter'. The creation of data matrices allows us to easily apply standard techniques from other fields to characterise networks, as was explored by previous work (Costa *et al.*, 2007; Bounova & de Weck, 2012; Ronqui & Travieso, 2015). However, as was discussed in section 2.6.8, these previous works used existing metrics as features which we believe is somewhat against the spirit of machine learning; one should apply it to the fullest representation of the data possible and allow the technique to discover which features are the most important. In this chapter, we explore how we can provide high-dimensional descriptions of networks to these data matrices.

5.1.1 Requirements for data matrix

Each feature in the data matrix must be directly comparable between examples; if two examples have the same entry for a feature, it must mean that this observable property is in some way the same. An additional constraint is that all of the features must have physical dimensions consistent with the transformation applied by the machine learning techniques used; for example, a linear transformation demands that the features must have the same physical dimensions, because the addition of features must be a valid operation. We normalise our features to make them dimensionless. It is important to note that there is often no penalty to including features that do not provide discriminatory power, as the machine learning procedures will just discard them. Clearly, some features must have some discriminatory power for the procedure to provide anything useful. In this and the next chapter, we consider the features separately such that we can understand their usefulness separately. In a real application one would concatenate all of the relevant feature vectors into a single feature vector, which could increase the richness of the results.

5.1.2 Features considered

In this section, we present a series of metrics and network representations and discuss whether they meet our criteria for features. In addition to the absolute requirements mentioned above, a good feature should also have the following properties:

1. similar networks should have small Euclidean distances between one another,

2. dissimilar networks, such as configuration models and the original networks, should have a fairly large distance between one another,

3. invariance with respect to the number of nodes in the graph.

In our view, as was discussed in section 2.6.8, features should also be simple—each feature should have as little 'human design' in it as possible to allow the machine learning technique to make its own decisions.

All matrices are converted into feature vectors by simply flattening them such that

$$x_{i+(j-1)n} = a_{ij},$$

where $\boldsymbol{x}$ is the feature vector corresponding to the $n \times m$ matrix A.

Adjacency matrices

Adjacency matrices do not initially appear to satisfy the requirements on features: there is no guarantee that node labels (i.e. $\{i|v_i \in \mathcal{V}\}$ where $i \in \mathbb{N}\backslash\{0\}$) between different graphs have the same meaning. We therefore relabel the nodes such that the node with the lowest degree has the lowest label and the node with the highest degree has the highest label; any nodes with the same degree are labelled randomly. This does mean that only networks with the same number of nodes can be meaningfully considered, even with padding—the decomposition would otherwise spot the hard cut-off as small networks had empty entries for any node labels higher than their number of nodes. It is unclear whether similar networks will have small distances between each other, nor whether dissimilar networks will have large distances between each other. This is something we will determine in testing in the next chapter. The feature is, however, very simple—it is the most common representation of networks.

We note with interest that node labels in synthetic graphs often hold a great deal of meaning. In many implementations of Watts-Strogatz small-world networks, nodes with adjacent or near-adjacent labels are neighbours and so have a far greater probability of being linked than nodes with labels which are very far apart. Similarly, Barabási-Albert models are usually constructed such that nodes with lower labels have higher degrees than nodes with higher labels as the first nodes added to the network are most likely to have links added to them. It is therefore very easy to be misled by the quality of these matrices; one receives very good separation between separate models due to the wealth of information contained in their labels. This information is clearly not 'real' in the sense that one is generally unable to construct a real network with labels which are as useful as that, so we take great care to discard it.

Laplacian matrices

Laplacian matrices, similar to adjacency matrices, do not appear to satisfy the requirements on features as node labels are not directly comparable. Relabelling the nodes based on their

degree mitigates this. Since we only consider networks with no self-loops, the Laplacian as a feature vector is clearly equivalent to including the degree sequence at the end of the adjacency matrix. As such, the Laplacian matrix adheres to our criteria in the same way as the adjacency matrices.

Degree correlation matrices

The degree mixing matrix or degree correlation matrix, D, is defined such that each entry

$$d_{i \leftarrow j} = P(\text{edge chosen at random connects a node of degree } j \text{ to a node of degree } i).$$

As it is a joint probability distribution, it is already normalised and suitable for usage with networks of any number of nodes. The degree mixing matrices are quick to calculate and take relatively little memory as, for most real-world networks, the maximum degree in a network grows very slowly with the size of the network. Networks with the same degree correlations will appear closely together; whether this is a good measure of similarity is an open question. For example, two Watts-Strogatz models with parameters of two and four neighbours could appear totally unrelated. The correlation matrix is simple, however; it can be seen as an adjacency matrix tallied up by degree and normalised.

Graphlet profiles

We generate graphlet profiles using the routines introduced in chapter 3.1 but without any consideration of colour. We apply the same normalisation presented in sections 2.6.7, 3.2.2 and 4.1.2, again reasoning that we don't need to compare our networks to configuration models when we're comparing them to many other networks. With regards to our criteria, similar graphs will clearly have little difference between them as they will have similar graphlet profiles. However, graphlet profiles do not consider the global topology of networks—for example, a network with a single connected component could potentially appear the same as a network with many disconnected components—so dissimilar networks might not appear far apart. Whether this is a problem in practice remains to be seen. Graphlet profiles are somewhat arbitrarily constructed as they have a couple of parameters: the size of the graphlets and the way that they are normalised. They are clearly simpler than some other network metrics such as betweenness centrality, however.

5.2 Dimension reductions

As was mentioned as the start of this chapter, these data matrices are very high-dimensional characterisations of the networks. To make these characterisations interpretable, we must compress the number of dimensions to a more manageable number, such as two. We do this via matrix decomposition.

To decompose our data matrices, we use principal component analysis (PCA, Wold *et al.*, 1987) as introduced in section 2.3. Many other techniques could be applied—related work applies multidimensional scaling or canonical variable analysis (Costa *et al.*, 2007; Schieber *et al.*, 2017). PCA was chosen because of its widespread usage and computational simplicity. We discuss this further and show our results in the following chapter.

Bibliography

Bounova, Gergana, & de Weck, Olivier. 2012. Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles. *Physical review E*, **85**(1), 016117.

Costa, Luciano da F., Rodrigues, Francisco A., Travieso, Gonzalo, & Villas Boas, P. R. 2007. Characterization of complex networks: A survey of measurements. *Advances in physics*, **56**(1), 167–242.

Ronqui, JRF, & Travieso, Gonzalo. 2015. Analyzing complex networks through correlations in centrality measurements. *Journal of statistical mechanics: Theory and experiment*, 9.

Schieber, Tiago A, Carpi, Laura, Díaz-Guilera, Albert, Pardalos, Panos M, Masoller, Cristina, & Ravetti, Martín G. 2017. Quantification of network structural dissimilarities. *Nature communications*, **8**, 13928.

Wold, Svante, Esbensen, Kim, & Geladi, Paul. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems*, **2**(1), 37–52.

Chapter 6

Application of matrix decomposition to characterisation of networks

> *"[Experimentalists must] seek and examine, as if they were indifferent divine beings, what is and not what pleases."*
>
> — Johann Wolfgang von Goethe, *Experiment as mediator between subject and object, 1792*

The data matrices introduced in the previous chapter are used here to characterise networks through the application of principal component analysis (PCA), a kind of matrix decomposition. We explore each feature described in the previous chapter in turn. We demonstrate the usage of our technique in distinguishing synthetic networks, distinguishing between networks and their configuration models, characterising piecewise static networks and assessing network damage.

In this section we consider only unweighted, undirected and uncoloured networks, with no multi-edges or self-loops. All networks we consider are converted to this form before any further processing is carried out; directed edges are replaced with undirected ones.

The features considered can clearly all be applied to directed and undirected networks simultaneously, but we believe this is unsound. In an directed network, $a_{i \leftarrow j} = 0$ does not have the same implications as in an undirected network where $a_{i \leftarrow j} = 0 \iff a_{j \leftarrow i} = 0$; the absence of an entry in an adjacency or Laplacian matrix means a very different thing. In practice, we found that PCA immediately separated the directed and undirected networks and lost all nuance within directed and undirected networks, which we do not show for brevity. To compare the two kinds of networks meaningfully while preserving directedness would require substantially more research and thought than we have had time for. It is a matter of regret that we thereby contribute to the general neglect directed graphs face in network

science (Johnson, 2019).

This chapter comprises five sections: first, in section 6.1 we discuss the synthetic and real-world networks we attempt to characterise in this chapter; section 6.2 characterises static networks; section 6.3 shows how we can distinguish between networks and configuration models thereof for some of our data matrices; section 6.4 explores how we can track the evolution of real and synthetic piecewise static networks; and finally section 6.5 summarises our findings from this chapter and provides an outline of some possible directions for further research.

6.1 Types of network considered

Here, we will briefly mention the different networks we attempt to characterise throughout this chapter. See sections 2.9.2 and 2.9.3 for more detail.

6.1.1 Synthetic networks

We use four types of synthetic networks, introduced in section 2.9.2, with various parameters, holding the number of nodes and density constant where features are sensitive to them. These networks are Erdős-Rényi (Erdős & Rényi, 1959), Barabási-Albert (Barabási & Albert, 1999), Watts-Strogatz (Watts & Strogatz, 1998) and configuration models (Newman & Girvan, 2004). The specific implementations of these models are from the LightGraphs.jl package for Julia (Bromberger, 2017). The Watts-Strogatz models are shown with various rewiring probabilities. The exact parameters are detailed in each figure and table 6.1.

6.1.2 Real-world networks

We only consider two types of real-world network as the figures became too hard to interpret when there were too many types of network displayed at once. This problem could be solved with creating interactive plots where metadata could be displayed about the networks as one hovered over them. Additionally, due to the fact that the adjacency and Laplacian networks can only compare networks with the same number of nodes, the real-world networks are left out for these as they are not all the same size.

The two types of network we do consider are a series of ecological food web networks (Thompson & Townsend, 2003) and a 112-person clandestine network corresponding to social ties within the social network of the Suffragette, Emmeline Pankhurst (Edwards & Crossley, 2009). More detail on these real-world networks is given in section 2.9.3.

6.2 Characterisation of static networks

We first demonstrate the general use of all of the features considered on static networks. Later in the chapter, we drop the graphlet sizes which were found to be too small to be useful or

Graph/model name	Abbreviation	Parameters/description
Erdős-Rényi	ER	0.8% edge selection probability
Barabási-Albert	BA	2 edges per additional node
Watts-Strogatz	WS	4 neighbours per node, various rewiring probabilities
Configuration model	Cfg model	The degree sequence of a graph $\mathcal{G}$
Food webs	Eco	Real-world ecological food webs with around a hundred nodes and hundreds of links each (Thompson & Townsend, 2003)
Suffragette social tie network	Suf	A network representing social interactions between members of the Suffragette suffragist movement with 112 nodes and 2690 edges (Edwards & Crossley, 2009)

Table 6.1 – Models considered and their parameters; all models had 500 nodes and 1000 edges

too large to be practical from a computational perspective.

6.2.1 Adjacency and Laplacian matrices

The dimension reductions of the Laplacian and adjacency matrices, figures 6.2 and 6.1 respectively, show similar results. The Laplacian matrices show a greater separation than the adjacency matrices. Both figures show that there is a clear separation between the scale-free graphs on one side and the small-world and random graphs on the other. Strikingly in figure 6.2 and less clearly in figure 6.1, one can observe that as Watts-Strogatz models have their wiring probability increased, they become less like the Erdős-Rényi random graphs, which is quite counter-intuitive. One would expect as more randomness was added to a nearest-neighbour graph that it would become more like a random graph. This warrants further investigation, so we will now examine the principal components in detail.

The first two principal components are shown reconstructed as matrices in figures 6.3 and 6.4. We have enhanced the contrast in these plots (and figure 6.7) by mapping the value of each cell, x, such that

$$x \mapsto \operatorname{sgn}(x)\sqrt{|x|},$$

where sgn is the signum or sign function. This transformation improves legibility around

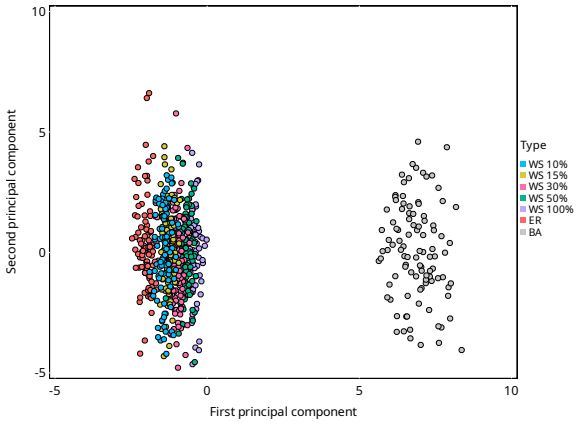

Figure 6.1 – Adjacency matrices of various networks whose nodes are labelled according to degree, as detailed in chapter 5, projected onto their first two principal components. Each point corresponds to a single network and the abbreviations used are explained in table 6.1

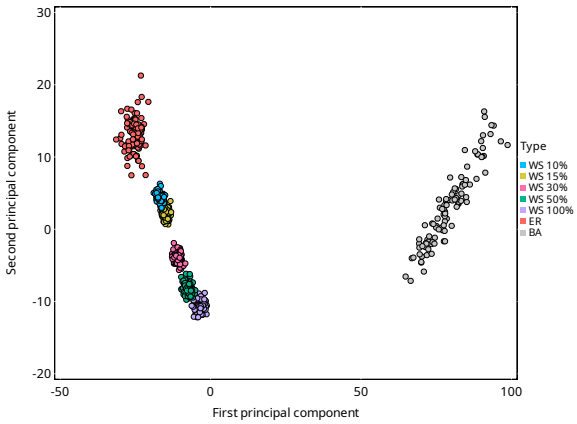

Figure 6.2 – Laplacian matrices of various networks whose nodes are labelled according to degree, as detailed in chapter 5, projected onto their first two principal components. Each point corresponds to a single network and the abbreviations used are explained in table 6.1

zero, aiding interpretation. The contrast was then further enhanced after plotting by editing the 'curves' in an image editor.

The first component of the adjacency matrix, shown in figure 6.3, has positive elements for nodes of higher degree linking to nodes of all degrees, and negative elements for nodes of lower degree linking to nodes of lower degree; i.e. it looks like the adjacency matrix of a scale-free network. This explains why the scale-free network is on the right hand side of figure 6.1. In figure 6.1, there is no discernible pattern along the second principal component. It is therefore unsurprising that when one inspects the second component in figure 6.3, it appears to be uniformly random.

The components relating to the Laplacian matrices, shown in figure 6.4, seem to be composed

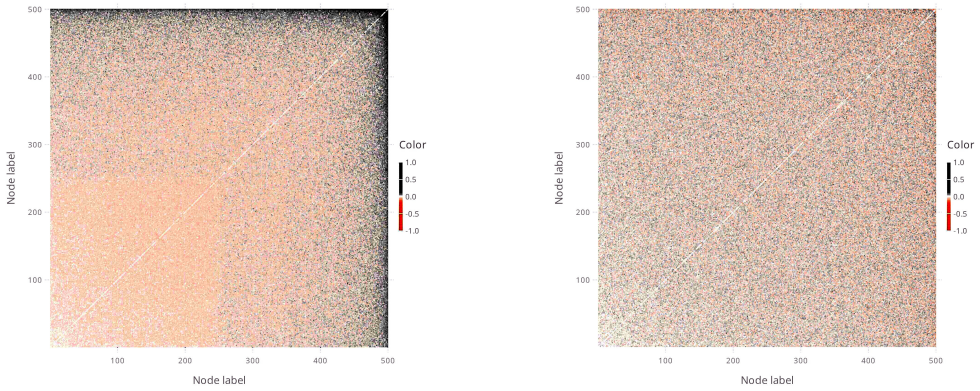

Figure 6.3 – The first (left) and second (right) principal components of the adjacency matrices corresponding to those in figure 6.1, reconstructed as matrices. Contrast has been enhanced as described in 6.2.1

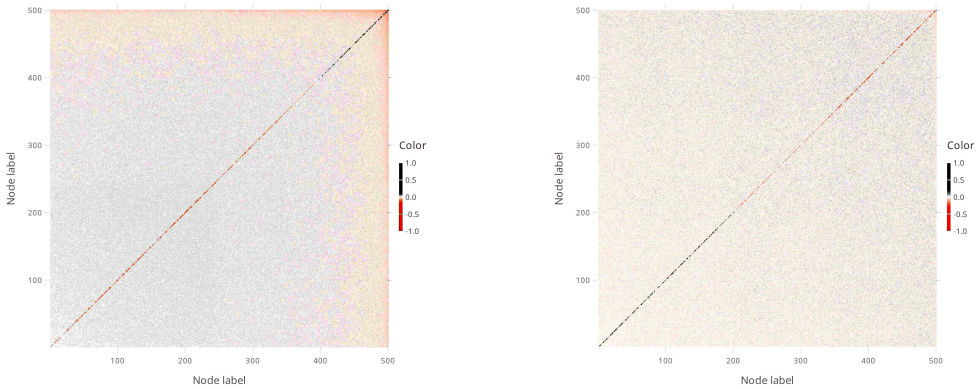

Figure 6.4 – The first (left) and second (right) principal components of the Laplacian matrices corresponding to those in figure 6.2, reconstructed as matrices. Contrast has been enhanced as described in 6.2.1. The diagonals are shown separately in figure 6.2

from the components given for the adjacency matrices with the addition of a diagonal, as one might expect given the definition of the Laplacian matrix. What is unexpected is that the first component seems to have had its adjacency matrix negated but without any effect on where the scale-free networks lie in the projected space. The diagonals are interesting and are understood more readily from figure 6.5: the first component's diagonal suggests that the first moment of the degree of the network is tightly centred on the highest degree nodes; i.e. they are scale-free networks, with a smooth negative region for the rest of the nodes. The diagonal of the second component is quite pronounced: it has discrete steps where degree strength falls at around 50, 200 and 350 nodes, before it peaks sharply in the positive direction around node 500. If we take the negative direction of this, it represents a degree distribution where most nodes have a degree of one of four levels.

From this, the positions of the networks can be explained: Erdős-Rényi graphs have a negative

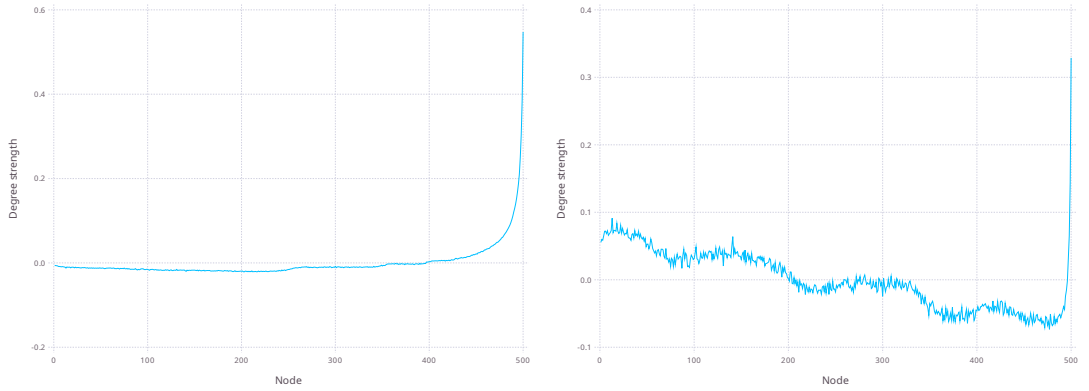

Figure 6.5 – The diagonals of the first (left) and second (right) principal components of the Laplacian matrices corresponding to those in figure 6.2, reconstructed as matrices. These diagonals correspond to the degree

first component, corresponding to a smooth, fairly even degree distribution, and a positive second component which cancels out the spike at the end of the first component. Watts-Strogatz graphs have varying levels of a negative first component and the second component can become positive, which vary in tandem according to the rewiring probability. More rewiring probability leads to more distance from the Erdős-Rényi graphs. To understand this, one must go back to the algorithm in section 2.9.2. Only a single link per node can be rewired, so a hundred-percent rewiring probability only corresponds to (number of nodes)·(number of neighbours)$^{-1}$ edges being rewired, and these rewired edges will have half of their ends evenly distributed throughout the nodes. We therefore must have a degree distribution which is quite tightly centred on the original distribution (such as the one shown in figure 2.4 on page 47), hence the large discrete steps in the second component, which the degree distributions of the Erdős-Rényi models (such as the one shown in figure 2.3 on page 46) do not possess. The Watts-Strogatz graphs with a low rewiring probability will have tight degree distributions and so the diagonal of the Laplacian matrix will be a constant, which is closer to the first principal component than it is to the second. These facts all explain why the increased rewiring probability moves Watts-Strogatz further from Erdős-Rényi for the Laplacian matrices, which is the opposite to intuition.

It is still unclear why the plots of the adjacency matrix suggest that Erdős-Rényi graphs are closer to Watts-Strogatz with low rewiring probability.

6.2.2 Degree correlation

The decomposed degree correlation matrices (figure 6.6) show good separation between different models. It illustrates clearly how, with rewiring, a Watts-Strogatz model becomes more like an Erdős-Rényi random graph as the rewiring probability increases from the bottom-right to the middle-top. It would be pertinent to note that this is exactly the opposite behaviour

to that seen in figures 6.1 and 6.2, which we explore in this section.

The figure also shows the similarities between the ecological networks, the Suffragette network and the Barabási-Albert network which are all clustered together in the top left.

Figure 6.7 shows how the projection manages such good separation between the scale-free and small-world networks. The first principal component has negative elements for low-degree nodes linking to high-degree nodes—i.e. it is negative for the types of degree mixing matrices one would expect from a scale-free graph, which is why the scale-free models appear on the left in figure 6.6. It has positive elements for nodes of degree 4 linking to nodes of degree 4 or 3, and some strength for nodes of degree 5 linking to nodes of degree 5. This matches the parameters in table 6.1 so the positive direction along this component therefore represents nearest-neighbour and potentially small-world networks.

The second principal component, also shown in figure 6.7, has a strong positive element for nodes of degree 4 linking to nodes of degree 4, surrounded by a 'smear' of negative elements for, e.g. nodes of degree 5 linking to nodes of degree 5. It has positive elements for nodes of low degree linking to nodes of high degree, with a characteristic striping going out to very high degree, signifying the small numbers of nodes with high degree in scale-free networks. This principal component therefore does double duty: its positive end represents nearest-neighbour networks with no shortcuts and scale-free networks; its negative end represents Poisonnian degree distributions such as those found in Erdős-Rényi graphs or small-world graphs with a very high number of shortcuts.

There are some clear limitations of this technique, however. It is very sensitive to the density of the small-world and Erdős-Rényi networks: any increase in density would move the 'blobs' seen in the bottom left of the reconstructed components diagonally upwards, such that the projection would portray the networks as totally unrelated. It is less sensitive to the density of scale-free networks, however, as the smooth fall-off in the likelihood of higher-degree nodes linking to higher-degree nodes means that networks with higher average degree will just appear as more 'extreme' versions of the less dense scale-free networks. Additionally, unlike the adjacency and Laplacian matrices, there is no requirement that the networks are the same size.

6.2.3 Graphlets

It is important to use the smallest graphlet possible: the complexity of the graphlet enumeration algorithm we use scales with the number of unique subgraphs of the size of the graphlet in the graph. For random graphs of constant density, the number of unique subgraphs of size n is approximately $\binom{E}{n} \propto E^n$, where E is the number of edges in the graph. We therefore tested a few sizes of graphlet and chose the one which struck the best balance between computation time and discriminatory power.

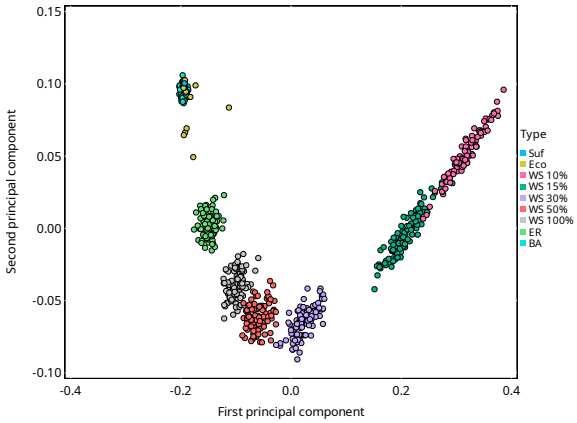

Figure 6.6 – Degree correlation matrices of various networks, as detailed in chapter 5, projected onto their first two principal components. Each point corresponds to a single network and the abbreviations used are explained in table 6.1

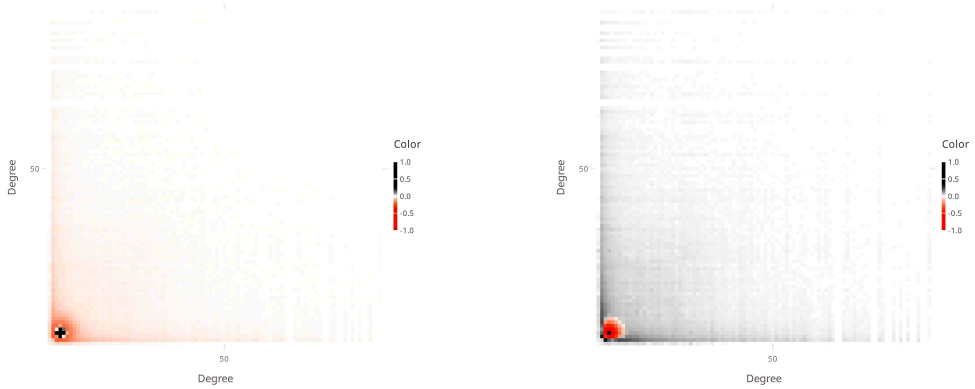

Figure 6.7 – The first (left) and second (right) principal components of the degree correlation matrices corresponding to those in figure 6.6, reconstructed as matrices. Contrast has been enhanced as described in 6.2.1

3-graphlets

The figure for the three-graphlets, figure 6.8, is not particularly informative; it does not show much separation between the different graphs and seems to be totally linear with no second principal component. Watt-Strogatz graphs with a rewiring probability of 10% can be distinguished from 30%. All of the others are hard to distinguish with many networks totally obscured by others. This is because an uncoloured 3-graphlet profile is a proxy for the global clustering co-efficient: it simply measures the percentages of paths of length 3 which are closed and open. It is therefore unsurprising that it is one-dimensional. It is interesting to note that the Suffragette network appears to have a very different clustering coefficient to all of the other graphs.

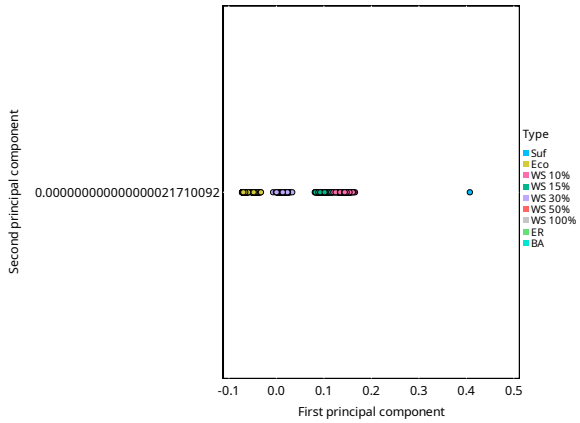

Figure 6.8 – A data matrix formed from the graphlet profiles of 3-graphlets of various networks of various types. The networks are shown projected onto their first two principal components; each point is a single network and the abbreviations used are explained in table 6.1

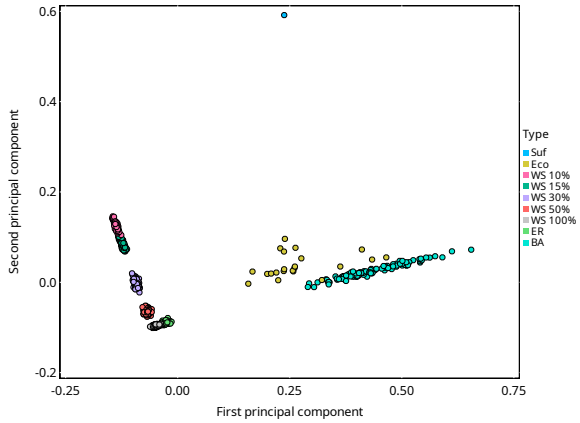

Figure 6.9 – A data matrix formed from the graphlet profiles of 4-graphlets of various networks of various types. The networks are shown projected onto their first two principal components; each point is a single network and the abbreviations used are explained in table 6.1

4-graphlets

The figure for the four-graphlets, figure 6.9, is much more informative. There is clear separation between many of the models and the similarities between the Barabási-Albert model and the ecological networks are clear. It is interesting that the Suffragette network is so much of an outlier in this plot compared to the degree correlation figure (figure 6.6); it is presumably due to the high clustering coefficient as was seen in figure 6.8. The transformation of the Watts-Strogatz model into something more like the Erdős-Rényi model is once again clear from the middle-left to the bottom-left.

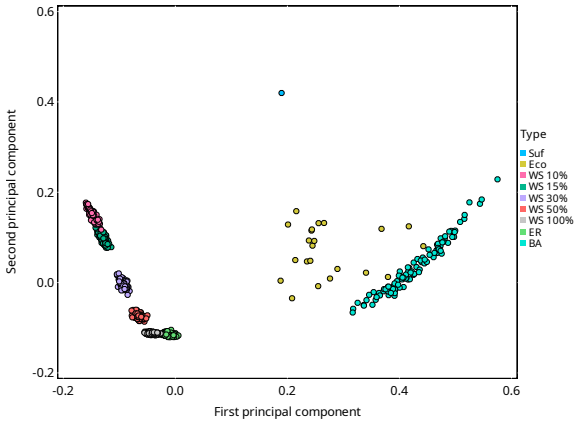

Figure 6.10 – A data matrix formed from the graphlet profiles of 5-graphlets of various networks of various types. The networks are shown projected onto their first two principal components; each point is a single network

5-graphlets

The figure for the five-graphlets, figure 6.10, also shows separation between the different graphs. Curiously, compared to figure 6.9, there is slightly more separation between the Barabási-Albert model and the ecological networks. This could be because the number of trophic levels in ecological networks is often no higher than five (Lindeman, 1942), whereas long chains are quite possible in Barabási-Albert models. There is not a great deal of difference between this and the case with the four-graphlets, but this takes far longer to run (approximately ten minutes rather than ten seconds), so from now on we only consider four-graphlets.

6.3 Characterisation of configuration models

A configuration model, as described in section 2.9.2, generates graphs similar to an existing graph. Here, the configuration model ensures that the generated graphs have the same degree sequence, $\{k_i | v_i \in \mathcal{V}\}$, as the original graph. Whether our techniques can distinguish between 'real graphs' and their configuration models is a good test of how well they characterise the networks; configuration models are only superficially similar to the original networks (Newman & Girvan, 2004).

Figures 6.11, 6.12 and 6.13, corresponding to adjacency, Laplacian and degree correlation matrices respectively, show no difference between the configuration models and the original networks. Figure 6.14 from the four-graphlets shows a clear difference, which is unsurprising as configuration models are well-known for reducing the clustering coefficient, which graphlet profiles can capture. It is a surprise that the degree correlation matrix performed so badly, as our configuration models only impose the constraint that the degree sequence is retained; all edges are otherwise assigned at random, so one would expect the correlations to be lost. Presumably

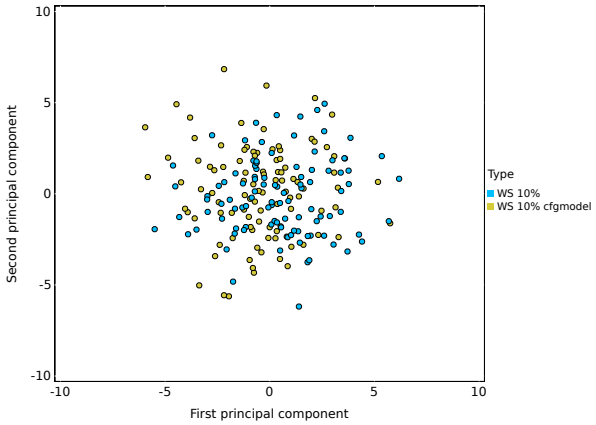

Figure 6.11 – Adjacency matrices, as in figure 6.1, projected onto their first two principal components of Watts-Strogatz synthetic networks and configuration models of those networks. The abbreviations used are explained in table 6.1

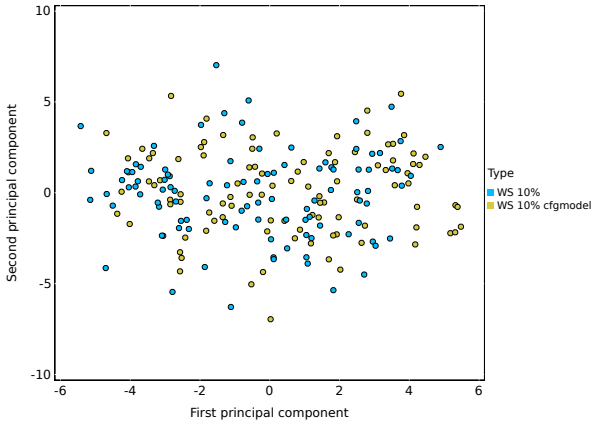

Figure 6.12 – Laplacian matrices, as in figure 6.2, projected onto their first two principal components of Watts-Strogatz synthetic networks and configuration models of those networks. The abbreviations used are explained in table 6.1

the homogeneity of the Watts-Strogatz model has ensured that the degree correlations are preserved. It could be fruitful to run this on more models, particularly more heterogeneous ones, and their configuration models and extract the principal components.

6.4 Characterising temporal networks

Our ultimate vision with this section is to eventually end up with something analogous to the famous Hertzsprung-Russell diagrams (Russell, 1914) from stellar astronomy where we can plot the life cycle of networks and categorise new networks based on where they appear within our projected diagrams. This clearly doesn't make much sense when comparing networks across domains—it strikes us as unlikely that a food web is likely to turn into a road network,

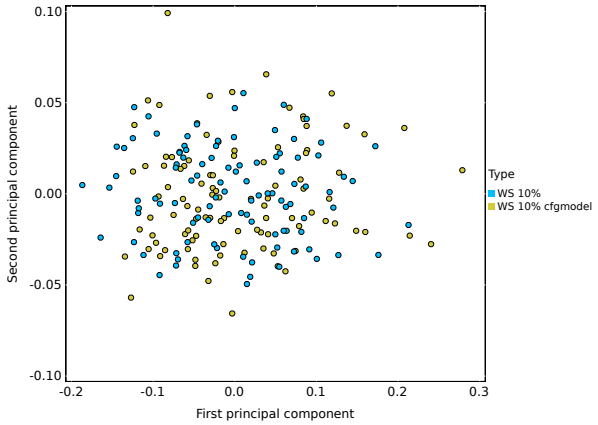

Figure 6.13 – Degree mixing matrices, as in figure 6.6, projected onto their first two principal components of Watts-Strogatz synthetic networks and configuration models of those networks. The abbreviations used are explained in table 6.1

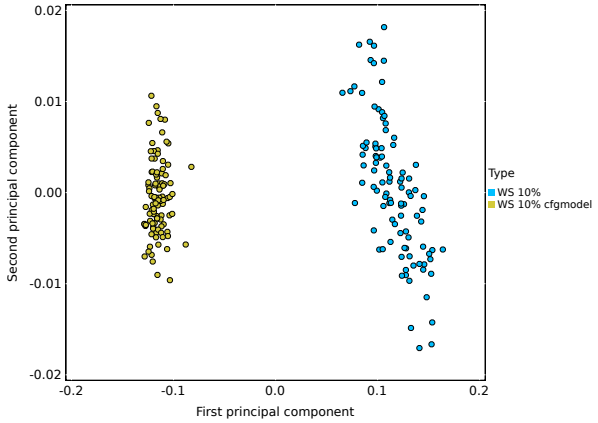

Figure 6.14 – 4-graphlet profiles, as in figure 6.9, projected onto their first two principal components of Watts-Strogatz synthetic networks and configuration models of those networks. The abbreviations used are explained in table 6.1

for example—so in the second part of this section we return to the collaboration network investigated in chapter 4. Our first section considers damage to synthetic networks.

6.4.1 Synthetic network damage

Figures 6.15 and 6.16 show the same mix of synthetic networks characterised by their projections onto their first two principal components according to their degree correlation matrices shown in previous sections. We have added a new network: a Watts-Strogatz network which has had a single link, chosen uniformly at random, severed at each time-step. The journey of the network from something indistinguishable to a small-world network to something more like a scale-free network can clearly be seen as it travels from right to left. It is interesting to note that the space it travels through is a kind of 'no-man's land' where no undamaged

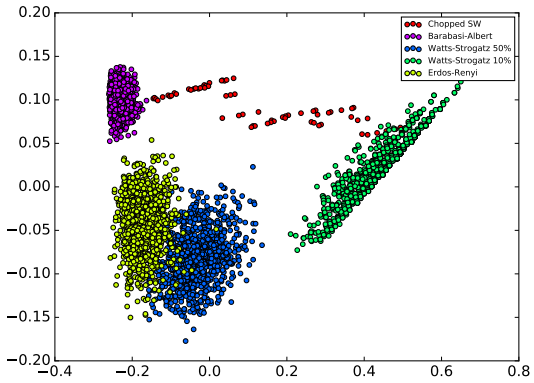

Figure 6.15 – Degree mixing matrices projected onto their first two principal components, as in figure 6.6, for 100-node networks via an 'online' method. Particularly of note are the 'Chopped SW' networks which show a network created according to a Watts-Strogatz model with a 10% rewiring probability where a random link has been removed at each time step which can be see as the network moves from right to left. The abbreviations used are explained in table 6.1

network is projected.

The only difference between the two figures is that figure 6.15 is only 'trained' on the undamaged networks, a so-called 'online' method, whereas 6.16 is 'trained' on all of the networks, an 'offline' approach; i.e. in the online approach, the transformation matrix W is calculated from a data matrix X which excludes the undamaged network. This could be important for real-world applications where data are streaming in in real-time, as a very large ensemble of reference networks could be used to find the projection matrices once and new data could be projected as it was added. It can be seen that there are limited differences between the two figures; predominantly that the end-point of the damaged small-world network is not exactly within the scale-free cluster in the 'trained on all networks' version, i.e. figure 6.16.

6.4.2 Characterisation of ego-networks

The techniques described in this part can be applied to the problem of describing and comparing individual nodes within a graph by considering the neighbourhood of each node as a separate graph. This work expands on that of chapter 4: we apply matrix decompositions to the coloured and monochromatic graphlet profiles of authors in an academic collaboration network. Cunningham *et al.* (2013) propose a similar technique for characterising nodes by the decomposition of data matrices formed from their graphlet profiles but do not use colour to denote the position of the ego or examine the effect of colour. Here, in contrast to chapter 4, we try to understand how the patterns of collaboration change over a researcher's career rather than trying to determine which graphlets make a long career more likely.

As in chapter 4, we use disambiguated data from Arnetminer (Tang *et al.*, 2008), covering

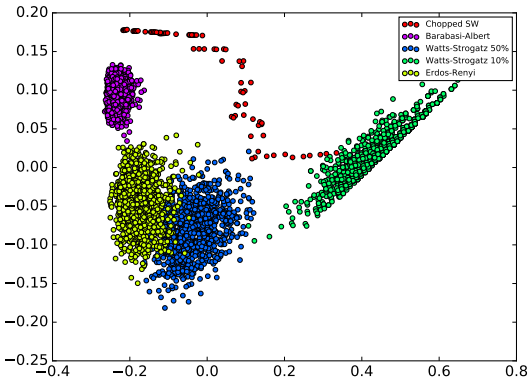

Figure 6.16 – Degree mixing matrices projected onto their first two principal components, as in figure 6.6, for 100-node networks via an 'offline' method. Particularly of note are the 'Chopped SW' networks which show a network created according to a Watts-Strogatz model with a 10% rewiring probability where a random link has been removed at each time step which can be see as the network moves from right to left. The abbreviations used are explained in table 6.1

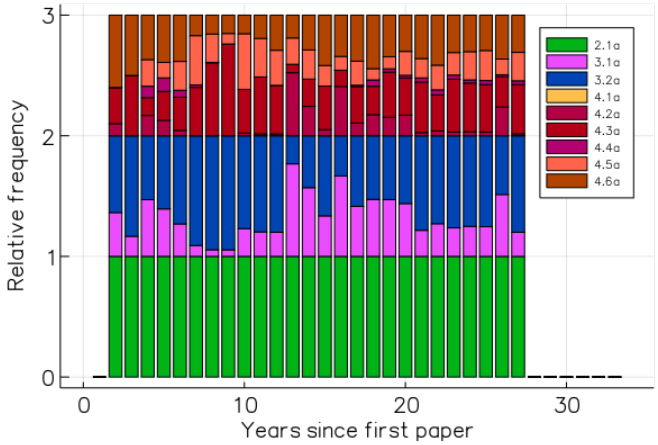

Figure 6.17 – The proportion of monochrome graphlets of each type found in the 3-ego-networks of the author known as '7' in the Arnetminer dataset. The graphlet labels correspond to those in figure 6.18.

the years 1986–2012. The window size was set to three years.

We picked one author as an example to investigate, known only as 'number 7', as they seemed to have a relatively long career. Most authors do not: they fizzle out after a few years, as was shown in figure 4.1. Figures 6.17 and 6.18 show the changes in the graphlet profiles of the 3-ego-networks of our author of interest. These diagrams are quite hard to interpret; one must cross-reference with figure 3.1. With some effort, it can be seen from figure 6.17 that the diversity of graphlets and the number of more complex graphlets (graphlets which are less likely to appear by random chance) increases as time goes on. These more complex graphlets include long chains and loops.

This similarly can be seen in figure 6.18, with the added complication that as time goes on,

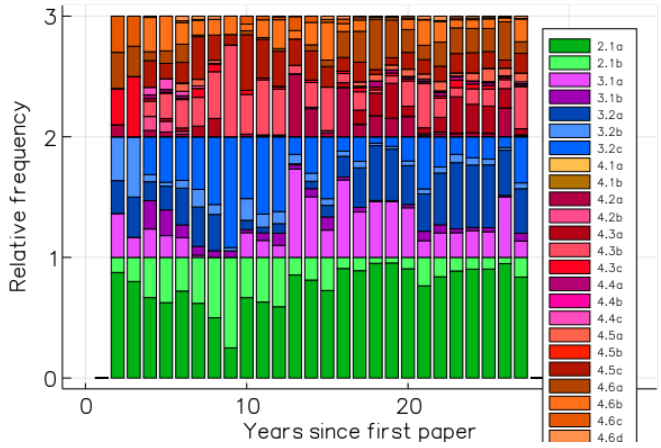

Figure 6.18 – The proportion of coloured graphlets of each type found in the 3-ego-networks of the author known as '7' in the Arnetminer dataset. The graphlet labels correspond to those in figure 3.1.

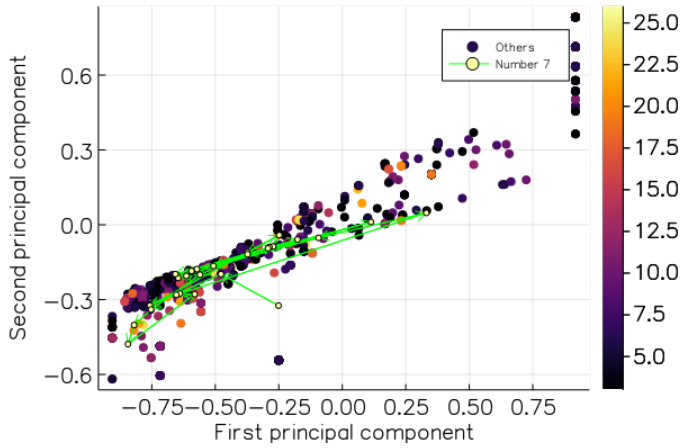

Figure 6.19 – A data matrix formed from the monochromatic graphlet profiles of 4-graphlets of the ego-networks of author '7' from each year of their career the Arnetminer dataset and a random sample of other authors. The networks are shown projected onto their first two principal components; each point is a single network. The colours of the other authors corresponds to their age in years. Author '7"s life cycle is shown with yellow circles for each year and green arrows showing the direction from one year to the next

the ego can be initially be found in more of the graphlets, but this subsides after the first decade—i.e. the 'a' variants of graphlets become more common—and could potentially reflect the author becoming more of a bridge for knowledge rather than an active researcher themselves.

In some sense, the same information can be seen in figure 6.20. The author starts life in an area sparsely populated by other authors, briefly goes towards the mass of authors residing around $(0.8, -0.25)$, and then spends most of their life in regions that are sparsely populated. We can therefore tell that this author is unusual, at least with respect to how they co-author papers.

There are clear limitations of this matrix decomposition approach on display here. Figures

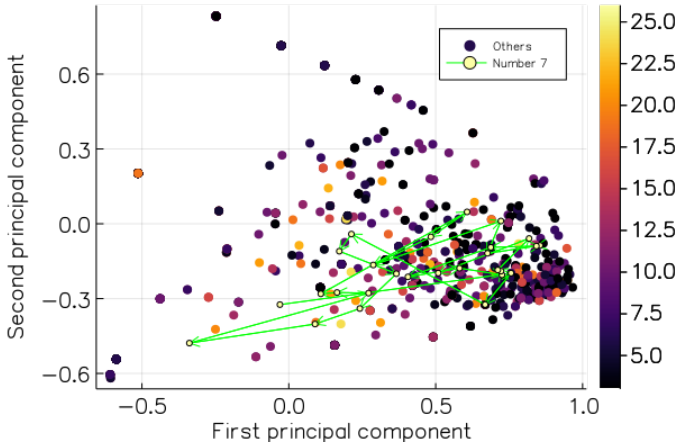

Figure 6.20 – A data matrix formed from the coloured graphlet profiles of 4-graphlets of the ego-networks of author '7' from each year of their career the Arnetminer dataset and a random sample of other authors. The networks are shown projected onto their first two principal components; each point is a single network. The colours of the other authors corresponds to their age in years. Author '7"s life cycle is shown with yellow circles for each year and green arrows showing the direction from one year to the next

6.19 and 6.20, where the nodes have been coloured according to their ultimate career-length, show no clear relationship between position and lifetime. The more traditional analysis in chapter 4 found a statistically significant positive link between a particular graphlet (4.3b) and career-length. Unlisted in that chapter but mentioned here for completeness, we also statistically tested for any link between the principal components found here and career-length and found none. As was shown in chapter 4 and particularly figure 4.4, when it comes to academic collaboration networks, a large portion of the variation in topology has nothing to do with research career-length. Given that PCA is predominantly concerned with preserving variation, it is unsurprising that it is of little use for investigating a particular hidden variable that is not associated with a great deal of variance.

6.5 Conclusions & further work

We have shown that matrix decompositions of data matrices formed from network representations and graphlet profiles can be used to discriminate between different types of synthetic networks and real-world networks. Furthermore, its use in displaying the damage done to a network was demonstrated. This work is all novel; the closest previous work had only considered data matrices formed from centrality distributions or metrics (Costa *et al.*, 2007; Bounova & de Weck, 2012; Ronqui & Travieso, 2015).

There is an obvious criticism of much of this chapter: there is no quantitative comparison of our technique with existing metrics. In the absence of a more thorough study, we can compare our plots with the plots made by Bounova & de Weck (2012). Our technique seems to show a similar level of separation between networks as the best metrics and decompositions

in their work.

However, comparison with other attempts at characterising networks in the literature is only a small part of the story. Complex networks are only of interest because of the systems they represent. Therefore the true test of any metric that describes how different networks are should consider how different the dynamics those networks support is and a measure of the amount of damage done to a network (as discussed in section 6.4.1) should be a measure of how much damage is done to the underlying node dynamics; this would be a fascinating avenue to explore for further work.

A further limitation of this technique is that because it relies so heavily on the variance in the data, it can be rendered useless by outliers in the data. This was shown by the fact that when we compared undirected and directed networks, that difference was found at the expense of all others. It was also shown in other testing where we neglected to first remove self-loops from our real world networks and, once again, that became the only difference that was noticeable. A related risk is that if one is interested in a characteristic that isn't associated with a great deal of variance in the topology, the technique is of limited use, as was shown in the stark contrast between the usefulness and interpretability of the results in section 6.4.2, where all we could say is that 'there's some variation' and 'this particular author's graphlet profile changes a lot over time', compared to section 4.2, where we showed that the 4.3b graphlet is correlated with career length after controlling for centralities, in the previous chapter. It would also be desirable to thoroughly test our techniques on directed networks, especially the data matrices formed from graphlet profiles, as direction greatly increases the total number of possible graphlets. Recent work has found that food webs fall neatly into categories when clustered based on directed graphlet profiles (Klaise & Johnson, 2017).

Our selection of graph models and real-world networks could also be improved. Recent work has suggested that many real-world networks are not as close to scale-free networks as once thought (House *et al.*, 2015; Broido & Clauset, 2019). Our technique could potentially inform this debate given enough real-world networks and competing models with approximately scale-free nature; the real-world networks should simply appear closest to their closest models in the projected space.

Schieber *et al.* (2017) published some related work after this work was conducted. Their approach is similar to ours; they form high dimensional descriptions of networks from characterising nodes and then apply a dimension reduction (specifically, multi-dimensional scaling) to see how networks are similar or dissimilar. The authors test their metric on automorphism groups, ensuring that their metric returns zero for distances within the group, which is a sound way of testing network characterisation. Our reduced Laplacian and adjacency network data matrices should return near zero for this test, since they cannot tell the difference between models and their configuration models, but we cannot be certain without testing. The degree

mixing matrices and graphlet profiles, however, are clearly invariant within automorphism groups so we can be certain that these data matrices would pass this test.

It would also be of interest to apply other dimension reductions and machine learning techniques to our data matrices. McInnes *et al.* (2018) in particular created a novel dimension reduction, UMAP, which is particularly computationally efficient, making it possible to analyse larger adjacency and Laplacian matrices than was possible at the time this work was conducted. Additionally, these 'manifold-preserving' dimension reductions are far less susceptible to outliers than variance-based dimension reductions such as PCA, so we could see better distinction between networks which are similar. Torres *et al.* (2019) have applied UMAP to networks through a data matrix constructed from a novel graph representation. It should be noted that many of these competing dimension reductions are not invertible transformations; i.e. one cannot map back to the original space from the reduced space. This greatly harms their interpretability; figures such as figure 6.7 would not be possible.

Rossi *et al.* (2018) have conducted similar work on the creation of data matrices, which they call feature matrices. The authors use graphlets as features by which they characterise networks for the purpose of inter-network comparison; and graphlets of ego-networks to characterise nodes.

In the next part, we apply a related technique—tensor decomposition—to discover and characterise communities in networks. There, each tensor represents a single network whereas in this chapter each matrix represented an ensemble of networks; this change in approach (previously pursued by other authors) is what allows us to characterise communities rather than whole networks.

Bibliography

Barabási, Albert-László, & Albert, Réka. 1999. Emergence of scaling in random networks. *science*, **286**(5439), 509–512.

Bounova, Gergana, & de Weck, Olivier. 2012. Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles. *Physical review E*, **85**(1), 016117.

Broido, Anna D, & Clauset, Aaron. 2019. Scale-free networks are rare. *Nature communications*, **10**(1), 1017.

Bromberger, Seth. 2017. JuliaGraphs/LightGraphs.jl. Sept.

Costa, Luciano da F., Rodrigues, Francisco A., Travieso, Gonzalo, & Villas Boas, P. R. 2007. Characterization of complex networks: A survey of measurements. *Advances in physics*, **56**(1), 167–242.

Cunningham, Padraig, Harrigan, Martin, Wu, Guangyu, & O'Callaghan, Derek. 2013. Characterizing ego-networks using motifs. *Network science*, **1**(2), 170–190.

Edwards, Gemma, & Crossley, Nick. 2009. Measures and meanings: Exploring the ego-net of Helen Kirkpatrick Watts, militant suffragette. *Methodological innovations online*, **4**(1), 37–61.

Erdős, Paul, & Rényi, Alfréd. 1959. On random graphs I. *Publicationes mathematicae (debrecen)*, **6**(1959), 290–297.

House, Thomas, Read, Jonathan M, Danon, Leon, & Keeling, Matthew J. 2015. Testing the hypothesis of preferential attachment in social network formation. *Epj data science*, **4**(1), 13.

Johnson, Samuel. 2019. Digraphs are different: Why directionality matters in complex systems. *arxiv preprint arxiv:1908.07025*.

Klaise, Janis, & Johnson, Samuel. 2017. The origin of motif families in food webs. *Scientific reports*, **7**(1), 16197.

Lindeman, Raymond L. 1942. The trophic-dynamic aspect of ecology. *Ecology*, **23**(4), 399–417.

McInnes, Leland, Healy, John, & Melville, James. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arxiv preprint arxiv:1802.03426*.

Newman, M. E. J., & Girvan, M. 2004. Finding and evaluating community structure in networks. *Physical review E*, **69**(2), 026113.

Ronqui, JRF, & Travieso, Gonzalo. 2015. Analyzing complex networks through correlations in centrality measurements. *Journal of statistical mechanics: Theory and experiment*, 9.

Rossi, Ryan A, Zhou, Rong, & Ahmed, Nesreen K. 2018. Deep inductive network representation learning.

Russell, Henry Norris. 1914. Relations between the spectra and other characteristics of the stars. *Popular astronomy*, **22**(May), 275–294.

Schieber, Tiago A, Carpi, Laura, Díaz-Guilera, Albert, Pardalos, Panos M, Masoller, Cristina, & Ravetti, Martín G. 2017. Quantification of network structural dissimilarities. *Nature communications*, **8**, 13928.

Tang, Jie, Zhang, Jing, Yao, Limin, Li, Juanzi, Zhang, Li, & Su, Zhong. 2008. ArnetMiner: Extraction and Mining of Academic Social Networks. *Proceeding of the 14th ACM sigkdd international conference on knowledge discovery and data mining - kdd 08*, 990.

Thompson, RM, & Townsend, CR. 2003. *Interaction Web DataBase*.

Torres, Leo, Suárez-Serrato, Pablo, & Eliassi-Rad, Tina. 2019. Non-backtracking cycles: length spectrum theory and graph mining applications. *Applied network science*, **4**(1), 41.

Watts, Duncan J., & Strogatz, Steven H. 1998. Collective dynamics of small-world networks. *Nature*, **393**(6684), 440–442.

Part IV

Temporal community characterisation

Chapter 7

Finding an optimum number of communities

"Life is mostly grief and labour
Two things get you through
Chortling when it hits a neighbour
Whingeing when it hits you"

— Kingsley Amis on local optima

This part explores temporal community characterisation through the use of tensor decompositions. As was explored in section 2.7, tensor decompositions are a good method of characterising temporal communities as they provide a detailed view into their evolution. They are the sole method of community characterisation we directly consider in this part.

Tensor decompositions are mathematically closely related to concepts introduced in part III. Here, we focus on using the decomposition techniques to detect communities in temporal networks and then calculate membership and relative activity of those communities over time. We decompose adjacency tensors, as introduced in section 2.2.2 on page 33, relating to single networks rather than data matrices relating to multiple networks. In this way we find similarities within rather than across networks. This approach gives us more detail about the behaviour within each network—compare this with the previous part where we distilled networks down to a single point—and is often easier to interpret. A clear example of this is in our novel application of the technique to software dependency networks in chapter 8, where the difference between CRAN, the R language's repository, and other software ecosystems can be seen as predominantly due to CRAN's rolling-release cycle (Blanthorn *et al.*, 2019); if we had performed the same analysis as in part III, we may well have seen the point relating to CRAN moving around more than the other ecosystems, but we would not have known why.

Tensor decompositions also make dealing with multilayer networks possible. The decompositions can be viewed as a kind of soft clustering, making them natural for use in networks where community membership is loosely defined.

In this part, a new technique for determining a locally optimal number of components into which a tensor representing a temporal complex network should be decomposed is introduced in section 7.1. Its use is demonstrated in chapters 8 and 9, where it is used to analyse two types of directed temporal networks: software collaboration networks, where we have shown the impact that the rolling-release cycle in R has on its software ecosystem. This chapter is simply split into two sections: the first presents our technique for finding locally optimal numbers of communities and the second provide a short summary of the chapter and technique.

7.1 Locally optimal numbers of communities for decomposition

One of the major downsides to tensor decompositions is that one must choose the number of communities or components, R, the current approaches to which are given in 2.4.1. In the rest of this chapter, we present our approach to finding locally optimal numbers of communities. Due to the local nature of the technique, where we compare decompositions of certain numbers of communities with higher and lower numbers of communities, we cannot make any statement about whether the number of communities is globally optimal.

As will be seen in section 8.3.2, unlike previous works, in practice we found that the NRSS, the normalised residual sum of squares, was not useful for determining locally optimal numbers of components because no 'elbows' were visible (Blanthorn *et al.*, 2019). We also found that CORCONDIA (Bro & Kiers, 2003) gave no pronounced 'elbows' either, but, as we were using our own implementation, we felt that we could not be certain that it was not due to an implementation error and therefore excluded it from our results in section 8.3.2. Previous works have had very clearly defined 'real' communities—such as pupils separated into classes at school (Gauvin *et al.*, 2014)—or have not been concerned with picking an optimal number of components (Kang *et al.*, 2012), but our applications (chapters 8 and 9) have no ground truth. This could account for the differences in utility of NRSS. As is explored in section 7.1.2, we reason that if multiple tensor decompositions arrive at similar clusterings—i.e. partitions of $\mathcal{V}$ which are approximately the same—despite the stochastic nature of the decomposition algorithms, those clusterings are likely to be the right ones. We therefore turned to another metric for evaluating clustering methods: adjusted mutual information, a measure of the level of agreement between clusterings.

7.1.1 Measuring the level of agreement between clusterings

Adjusted mutual information is a measure of the level of agreement between two sets of clusterings, independent of cluster labels. If the two clusterings split the elements into the same subsets, it is maximal; if the two clusterings have the same level of overlap between their subsets as if they had been partitioned randomly, it is 0.

More rigorously, the adjusted mutual information (AMI) between two partitions U and V, where $U = \{U_1, U_2, \ldots\}$ is a finite-length partition of some set of length N and V is another finite partition of that set, is

$$\text{AMI}(U, V) = \frac{I(U;V) - E[I(U;V)]}{\max(S(U), S(V)) - E[I(U;V)]},$$

where the entropy S is

$$S(U) = -\sum_i \frac{|U_i|}{N} \log \frac{|U_i|}{N},$$

with $0 \cdot \log(0) = 0$, the mutual information between U and V is

$$I(U;V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|},$$

and $E[W]$ is the expectation of a random variable W. Here, it can be calculated using a hypergeometric model of all possible random clusterings (Vinh *et al.*, 2009) which we omit for brevity. AMI measures the amount of agreement between two clusterings. It has an upper bound of 1. An AMI score of 0 corresponds to the amount of agreement one would expect if one of the clusterings was totally random and 1 corresponds to perfect agreement between the two sets of clusters. The AMI score can be negative if there is more disagreement between the clusters than one would expect from random chance. It is hard to interpret the meaning of scores between 0 and 1, but it is important to bear in mind that anything greater than 0 is a good score in the sense that it is better than random guessing. AMI is quite sensitive to the number of clusters and awards higher scores to pairs of clusterings that agree on the same number of assignments but have a larger number of clusters, to the limit where if each element is given its own cluster the AMI must be 1.

7.1.2 Application of AMI to choosing the number of components in tensor decompositions

The first task is to consider which values of R to test. Generally in this work, we don't have any knowledge as to the likely number of communities. We therefore start with two, since $R = 1$

will always give a maximal AMI. We run the decomposition multiple times for each likely R to determine whether the decomposition is stable, that is, if multiple runs produce the same results. As with many experiments, the exact number of runs required will depend on the exact application: as the number of runs increases the standard error of the mean pairwise AMI will reduce. This makes it easier to determine with more certainty where there is a peak in AMI. In our experiments, we found that thirty repeats were sufficient. We reason that a good choice of the number of communities to decompose a tensor into should result in the tensor decomposition algorithms choosing roughly the same nodes to assign to each community as this implies that true communities are not being artificially split up. That would result in nodes being assigned fairly randomly to each of the communities a true community was split into and so would not result in what we term a stable decomposition.

We look to see if nodes appear in the same communities as each other by considering the mean pairwise adjusted mutual information between repeated runs for a single R. We use hard clustering for this process by assigning nodes to communities for each non-temporal factor from the maximum strength, i.e. for a single-layer temporal network, $\text{Community}(i, \boldsymbol{v}) = \text{argmax}(v_{ir})$, for each factor matrix in equation 2.1 on page 35. For many applications it is clear which factor to prioritise; for a single-layered temporal network with a scale-free out-degree distribution but a normal in-degree distribution, such as the software dependency networks considered in chapter 8, it is probably best to consider communities based on the factor corresponding to the in-degree distribution, as the out-degree factor will be very sparse.

Since we are using AMI to compare clusterings with each other and have far fewer clusters (or communities) than nodes, we believe the interpretability of the absolute value of the AMI is not particularly important. We are concerned primarily with the relative changes in AMI as we move from one number of communities to another; a peak in AMI with respect to the number of communities means that that number of communities is more stable and is therefore more likely to be an optimal number of communities. We therefore plot AMI against R and look for the point with a negative second derivative—the second local maxima—to guard against the initial advantage that very low numbers of communities have with AMI. Rather than an an 'elbow' in the plot of AMI versus the number of communities, we therefore look for the second maxima in a graph that looks like a 'U-bend'.

7.1.3 Subjective checks

One can check visually to see if the temporal activity of communities looks similar across multiple decompositions into a single number of communities. If the decomposition is unstable, i.e. if multiple runs produce wildly different results, it suggests that ground-truth communities are being merged or split up. Another check that can be performed is to look at the metadata associated with nodes from each community, making a qualitative judgement

as to whether nodes in a single community are similar. For example, if a numerical computing software package and a front-end web development software package are in the same community in a software dependency network, the number of communities is probably too small. A final check is whether adding an extra component adds interesting new behaviour to the temporal community activity, or if it instead seems to break an existing community into noisy subcommunities. All of these approaches are quite subjective, so there is quite a large uncertainty in exactly how many communities is ideal for any application. This is not a major issue as we are not particularly interested in the exact number of communities our applications have; indeed, we do not believe this is a meaningful question as there are no 'ground truth' communities, i.e. communities which have some real basis, for example, the classes children are assigned to at school. We are interested in the activity over time of communities and the membership of these communities—it does not matter a great deal if two communities are seen as erroneously one in our analysis.

7.2 Summary

In this chapter we have explored a method for determining a locally optimal number of communities, R, to decompose a tensor into. In the next chapter, we apply this technique to analyse temporal collaboration networks: software dependency networks corresponding to several languages.

Bibliography

Blanthorn, Oliver, Caine, Colin M, & Navarro-López, Eva. 2019. *Evolution of communities of software: using tensor decompositions to compare software ecosystems*. Unpublished: accepted by Applied Network Science, Springer, 23 August 2019.

Bro, Rasmus, & Kiers, Henk A.L. 2003. A new efficient method for determining the number of components in PARAFAC models. *Journal of chemometrics*, **17**(5), 274–286.

Gauvin, Laetitia, Panisson, André, & Cattuto, Ciro. 2014. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *Plos one*, **9**(1).

Kang, U, Papalexakis, Evangelos, Harpale, Abhay, & Faloutsos, Christos. 2012. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. *Pages 316–324 of: Proceedings of the 18th ACM sigkdd international conference on knowledge discovery and data mining*. ACM.

Vinh, Nguyen Xuan, Epps, Julien, & Bailey, James. 2009. Information theoretic measures for clusterings comparison: Is a correction for chance necessary? *Pages 1073–1080 of: Proceedings of the 26th annual international conference on machine learning*. ICML '09. New York, NY, USA: ACM.

Chapter 8

Community detection in temporal software dependency networks

"Computers are useless. They can only give you answers."

— Pablo Picasso

Factorisation of tensors, and our own work on determining the optimal number of communities from chapter 7.1, is used in this chapter to explore the dynamics of communities of software and then compare these dynamics between languages on a dataset of approximately 1 million software projects. We establish that adjusted mutual information is a valid metric by which to assess the number of communities in a tensor decomposition and find that there are striking differences between the communities found across different software ecosystems and that communities do experience large and interpretable changes in activity over time. The differences between the elm and R software ecosystems, which see some communities decline over time and the more conventional software ecosystems of Python, Java and JavaScript, which do not see many declining communities, are particularly marked. A paper describing some of this work has been accepted and will shortly be published (Blanthorn *et al.*, 2019).

8.1 Importance of communities in software dependency networks

Contemporary software authors routinely depend on and re-use the software packages of authors with whom they have no contact. This unco-ordinated process creates what have recently been called 'software ecosystems' (Decan *et al.*, 2018a): extensive networks of interdependent software components that are used and maintained by large communities of contributors all over the world. These ecosystems are complex networks whose nodes and

edges both evolve over time. Throughout this chapter, terminology such as 'package', 'package manager', 'dependency' follow the usual software engineering convention, as documented by (Decan *et al.*, 2018a). With this work we would like to contribute to the debate on software dependencies, which recently re-emerged due to the hijacking of the `event-stream` package after previous incidents (Baldwin, 2018; Schlueter, 2016a; Durumeric *et al.*, 2014), by giving a clearer picture of the structure of software dependency networks. For this, we propose a novel framework to model and analyse the formation, long-term behaviour and change with time of communities of software packages and compare these behaviours across several programming languages.

The evolution of software and package dependency networks has been extensively studied by using network science techniques over the past 15 years. An early such work is Myers (2003), which has been followed by others (Pan *et al.*, 2011; Xu *et al.*, 2005; Zheng *et al.*, 2008). A recent survey on this topic is given by Savić *et al.* (2019b). Methods and tools used for social networks have also been applied to analyse software evolution (Chatzigeorgiou & Melas, 2012), reinforcing the importance of the social component in evolving software networks. There has been recent success in applying standard network metrics to analyse how software ecosystems have evolved with time (Decan *et al.*, 2018a,b).

Collaborative software networks—mainly in free and open-source software (FOSS) environments—can be considered as adaptive, evolving or temporal networks. Most of the published works highlight the fact that software networks exhibit scale-free network properties with a power-law-type node degree distribution (Cai & Yin, 2009; Lian Wen *et al.*, 2009; Louridas *et al.*, 2008) and a clear preferential attachment in the network growth process (Ai *et al.*, 2013; Chaikalis & Chatzigeorgiou, 2015), confirming the heterogeneity and hierarchical characteristics of networks of software. Other key properties in software systems like maintainability and reliability (Chong & Lee, 2015), robustness (Gao *et al.*, 2014) and modularity (Zanetti & Schweitzer, 2012) have been also analysed using the complex network paradigm.

The identification of communities in software dependency networks that evolve over time is one of the main motivations of this chapter. General background on characterisation of communities in temporal networks was given in section 2.7. There has been much less work on finding clusters or communities in software dependency networks: some representative works are Dietrich *et al.* (2008); Paymal *et al.* (2011); Concas *et al.* (2013) and Savić *et al.* (2012). However, it is still a challenge to give satisfactory solutions for the dynamic treatment of these clusters and inter-language comparison. (Savić *et al.*, 2012), for example, give some dynamic treatment of communities within the 'class collaboration' network of Apache Ant. This is a single piece of software in which the classes are more witting participants in the software than in general software dependency networks where the authors of a package may have no idea where their work is being used. Additionally, the community detection methods used are

static, so it is the metrics on the types of communities found at each version of Apache Ant which are compared rather than the qualities of any individual communities being tracked across time.

The novelty of our work is the detection of communities in temporal software dependency networks, the use of tensor decompositions on software ecosystems and the use of adjusted mutual information (AMI, introduced in chapter 7.1) to assist in choosing the number of communities. Additionally, some of the networks we study are amongst the largest temporal networks to which tensor factorisation has been applied, although there have been studies considering much larger static networks (Kang *et al.*, 2012).

This chapter addresses three research questions corresponding to unsolved problems in collaborative large-scale software development and evolution over time:

- RQ_1 What are the differences between different software ecosystems?
- RQ_2 What do communities of commonly-used-together software packages look like?
- RQ_3 How do these communities change with time?

By answering these questions, we identify communities of packages in the ecosystems of several languages. Namely: Elm; JavaScript; Rust; Python; R; and Java. These languages were chosen because data on their package ecosystems are readily available and they represent a variety of uses and ages: low-level systems languages; scientific computing; and web development; with histories between 3 and 20 years long.

The rest of this chapter is structured as follows. Firstly, in section 8.2, we detail the data, mathematical tools, software and hardware used to conduct this study. Next, section 8.3 presents our results and provides some discussion of why they might have arisen. The final section, section 8.4, compares our results with prior work and provides a summary of this works' limitations while highlighting further avenues of research.

8.2 Methods

8.2.1 What is a community?

Software packages associated with a single programming language and package manager form an *ecosystem* and a *community* is a collection of packages which are more likely to depend on one another than depend on packages from outside that community. Communities may be related by some theme and by co-occurrence of activity over time. They might be collections of numerical computing packages, or a community of packages that use a particular library, or a community of older packages that became obsolete. Packages may be in more than one community, which is modelled by vectors of continuous strengths of membership to each

community as detailed in the 'Tensor decompositions' section below. Communities may wax and wane over time.

Further detail on how we calculate communities and membership strengths thereof is given in section 8.2.4, chapter 7.1 and section 8.2.5.

8.2.2 Shape of the data

We considered the following package managers (corresponding to specific languages): elm-get (Elm), npm (JavaScript), crates.io (Rust), PyPI (Python), CRAN (R) and Maven (Java). All publicly registered packages for these package managers were included in our analysis. To simplify our analysis, we do not consider specific versions of dependent packages—each package depends on some other packages at each time, each of which is assumed to be the most up-to-date package available at that time. We only consider packages registered with these package managers and do not consider cross-platform (inter-language) dependencies. The data cover 1 million projects listed in package managers. This is sourced from Libraries.io (Katz, 2018). The network of Elm dependencies at the final time-step is shown in figure 8.1.

Brief introduction to each ecosystem

Here, we will briefly introduce each language and ecosystem considered.

Elm (elm-lang.org) is a relatively small new language created in 2012 as a functional language for developing web applications. It is unusually focused on a single domain and unusually restrictive in what can be expressed in listed packages. The language authors explicitly discourage proliferation of packages in favour of a unified and carefully designed standard package for each task. It aims to eliminate run-time errors and many classes of logical errors through static analysis and careful language design.

Python is a very popular general-purpose interpreted programming language. It was first released in 1991. Its package manager, PyPI (https://pypi.org/), was released in 2002. Python is notable for having a large standard library; many features such as HTTP and GUI support that would be external packages in other languages are included by default. We therefore suspect that Python's package ecosystem will look quite different to many of the other languages we consider. It is perhaps pertinent to note that some of the packages listed in PyPI are designed to be used by end-users, not developers.

JavaScript is another popular general-purpose interpreted programming language. It was first released in 1995 as part of the Netscape browser and was initially used for client-side scripts on web pages, but now has several server-side implementations, most notably Node.js. JavaScript has a very small standard library, with the exception of the DOM interface (Document Object Model—a representation of a web page), which is included. We therefore expect most

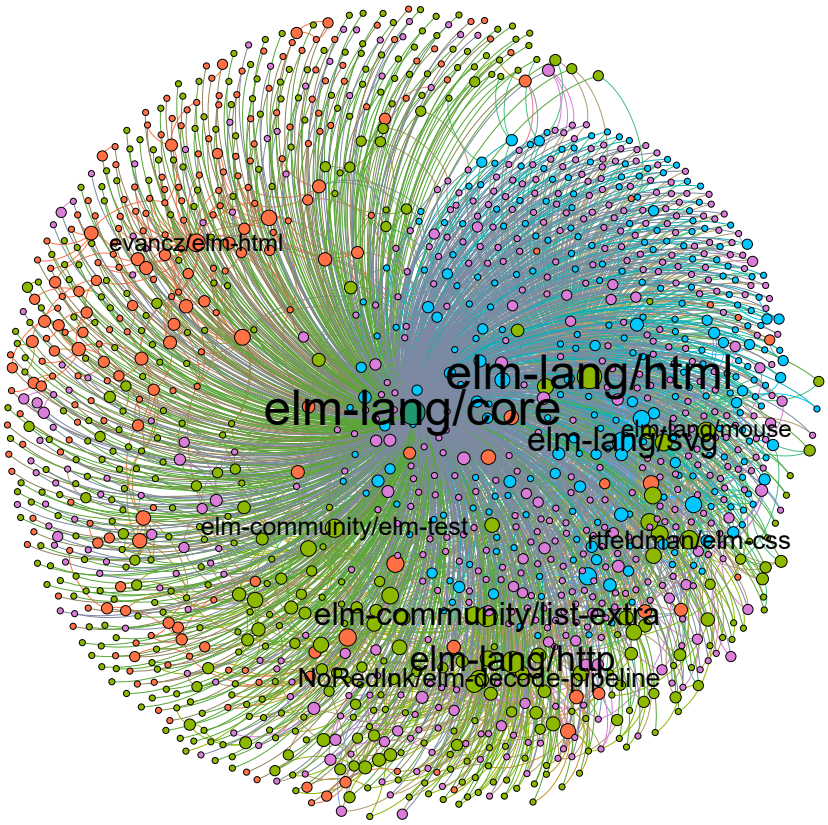

Figure 8.1 – Elm network visualised using `Gephi`. Node and label size corresponds to out-degree. Each node i is coloured according to the strongest strength of community membership given by $\boldsymbol{\kappa}_i$ as described in the section 8.2. Legend: light-green corresponds to the community we labelled as `elm-lang/core`, purple to `elm-lang/http`, blue to `elm-lang/html` and orange to `evancz/elm-html`, dark-green to unassigned nodes, i.e. those without any dependencies

JavaScript packages to have to depend on many other packages. JavaScript's most common package manager is npm (npmjs.com), the Node.js package manager, which was released in 2010. The JavaScript ecosystem we describe in this paper is the public npm registry, which is the *de facto* canonical list of JavaScript packages. Despite its name, the registry is used for all types of JavaScript, not just server-side applications. Like PyPI, some of the packages listed in npm are designed to be used by end-users, not developers.

Rust is another relatively new systems programming language. Unlike many of the other ecosystems we have considered, it is statically compiled. One of its main aims is safe concurrent programming. We suspect that because of this conservative focus, packages written in Rust will have relatively few dependencies. Rust's package manager is Cargo, which was released in 2014 and its registry is crates.io. Like PyPI, some of the packages listed on crates.io are designed to be used by end-users, not developers.

R is an interpreted programming language for statistical computing. It was first released in 1993. Its standard library for statistical computing is large. Its package manager is the Comprehensive R Archive Network (CRAN cran.r-project.org).

Java is a general-purpose typed compiled programming language, first released in 1995. Its standard library is large. Java has no official package manager, but the most widely used is Apache Maven (maven.apache.org), which was released in 2004.

8.2.3 Representing the data

We represent each language's network as a tensor, $\boldsymbol{A}_{\text{lang}}$, where each

$$a_{ijt} \in \{0, 1\}$$

denotes whether the package i depends on the package j at time t. We consider links as going from j to i, so the direction of each link reflects the flow of code. Software packages have unweighted directed links to packages that they depend on at time t according to the most recently available version of the software package at that time. The time is quantised into time-steps. We chose a resolution of one month on the basis that packages are long-lived and change dependencies rarely; we looked at other resolutions but they didn't qualitatively change our results.

8.2.4 Tensor decompositions

As was detailed in section 2.4, a rank-N tensor X can be approximated as

$$X \approx \sum_{r=1}^{R} \text{reduce}_{n=1}^{N}(\otimes, \boldsymbol{a}_{nr}),$$

where $\text{reduce}_{n=1}^{N}(\odot, b_n)$ returns the reduction of $\{b_n | n \in 1..N\}$ by applying an arbitrary associative binary operator $\odot$ repeatedly, i.e.

$$\text{reduce}_{n=1}^{N}(\odot, b_n) = b_1 \odot b_2 \odot b_3 \odot \ldots \odot b_N;$$

$\otimes$ is the tensor product; and each $\boldsymbol{a}_{nr} \in \mathbb{R}^{d_n +}$ where d_n is the dimension of the n^{th}-mode of the tensor X.

Each $\boldsymbol{a}_{nr}$ for a specific n is called a factor and maps its indices (in our case, software packages or time) to strength of membership to the community r. The total number of communities or components, R, is a fixed chosen parameter. Each $\boldsymbol{a}_{nr}$ for a specific r is called a community or component. The concept of a temporal index having strengths of community membership may seem odd; it is perhaps more easily understood as the amount of community activity at a certain time (Gauvin *et al.*, 2014).

There are many algorithms for performing tensor decompositions of this kind, as we discussed in section 2.4. We selected a non-negative method with a public implementation that would allow us to later extend our analysis to large rank-N tensors: Alternating Proximal Gradient (APG-TF[1], Xu & Yin, 2013), an efficient non-negative CANDECOMP/PARAFAC (CP) method. We think the use of tensor decomposition (Kolda & Bader, 2009) is better suited to community detection in temporal networks than other static methods, such as the Louvain algorithm (Aynaud & Guillaume, 2010), because tensor decompositions explicitly integrate evolution over time and can scale very efficiently for large graphs.

For our specific application, we approximate our temporal adjacency matrix as

$$a_{i\leftarrow j,t} \approx \sum_{r=1}^{R} \kappa_{ir} h_{jr} \tau_{tr}, \tag{8.1}$$

where κ_{ri} are the elements of a matrix $K \in (\mathbb{R}^+)^{N_p \times R}$ that describe the strengths of community membership of each node i based on outgoing links, h_{rj} are the elements of a matrix $H \in (\mathbb{R}^+)^{N_p \times R}$ which represent the strengths of community membership of each node j based on incoming links, τ_{rt} are the elements of a matrix $T \in (\mathbb{R}^+)^{N_p \times R}$ that describe the activity levels of each community r at each time step t and N_p and N_t are the number of packages considered and the number of time steps chosen, respectively.

We will now describe H and K in more detail. We will use two facts about software dependency networks (Decan *et al.*, 2018a):

- the number of outgoing links from each package (number of packages that depend on a package) follows a power-law;
- the number of incoming links to each package (number of packages that a package depends on) is fairly evenly distributed.

Since H represents groups of nodes which are linked to by similar nodes, it is very sparse with only a few hub nodes having any community membership due to high strengths in H. K is much more evenly distributed as it represents groups of nodes which link to similar nodes; most packages are strongly associated with at least one community.

[1]Implementation: https://www.caam.rice.edu/~optimization/bcu/ncp/

Term	Description
dependency	a package whose code is used by another package (the dependent)
dependent package	a package which uses another package's code (the dependency)
$A = [a_{i \leftarrow j,t}]$	the adjacency tensor of the software dependency matrix where the direction of links denotes the flow of code from dependencies to dependent packages
i, $\max(i) = N_p$	the index corresponding to packages with incoming links, i.e. dependencies
j, $\max(j) = N_p$	the index corresponding to packages with outgoing links, i.e. dependent packages
t, $\max(t) = N_t$	an index corresponding to time measured in months elapsed since records began
r, $\max(r) = R$	an index corresponding to communities
$K = [\kappa_{ir}]$	a matrix which comprises strengths of membership of package i to community r based on their dependencies for all i, r
$H = [h_{jr}]$	a matrix which comprises strengths of membership of package j to community r based on dependent packages for all j, r
$T = [\tau_{tr}]$	a matrix which comprises temporal activities of community r at time t for all r, t

Table 8.1 – Nomenclature for decomposed adjacency tensors of software dependency networks

If a community has significantly declined from its peak according to a plot of $\boldsymbol{\tau}_r$, we will define its lifetime as the full width at half maximum. A summary of the relevant symbols and terms is provided in table 8.1 for the reader's convenience.

8.2.5 Choosing the number of communities, R

One of the major downsides to tensor decompositions is that one must choose the number of communities, R, as explored in chapter 7.1. For this application, we employ all techniques mentioned: NRSS, AMI and the more subjective measures mentioned in section 7.1.3, adapted to the application of software communities by looking at the metadata associated with a sample of software projects (e.g. the names of the packages, their authors, the descriptions of the packages) from each community and making a qualitative judgement as to whether they are similar. For example, if a numerical computing package and a front-end web development package are in the same community, the number of communities is probably too small.

8.2.6 Labelling communities

H is a sparse matrix that denotes the main packages that are depended on in each community. The package name associated with a community r is the name of the package p_r that is maximal in $\boldsymbol{h}_r$, such that p_r does not have a higher $\boldsymbol{h}_{r'}$ value in any other community r'. That is,

$$p_r = \text{indmax}([\text{mask}(H)]_r),$$

where $\text{indmax}(\boldsymbol{x})$ of any vector $\boldsymbol{x}$ returns the index of its maximum value and

$$[\text{mask}(M)]_{ij} = \begin{cases} m_{ij} & \text{if } m_{ij} = \max(\boldsymbol{m}_i) \\ 0 & \text{otherwise,} \end{cases} \tag{8.2}$$

where m_{ij} and $\boldsymbol{m}_i$ are the elements and column vectors of an arbitrary matrix M.

8.2.7 Hardware and software used

Julia (Bezanson *et al.*, 2012), LightGraphs.jl (Bromberger, 2017), Plots.jl and Jupyter (Kluyver *et al.*, 2016) with IJulia were used for exploratory data analysis, data pre-processing, figures and coordinating external software. The actual tensor decompositions were performed using MATLAB.jl, MATLAB and TensorToolbox (Bader & Kolda, 2007). Adjusted mutual information was calculated using ScikitLearn.jl and scikit-learn (Pedregosa *et al.*, 2011). The decompositions were performed on 20-core Xeon servers with 250GB of RAM provided by MARC1, part of the High Performance Computing and Leeds Institute for Data Analytics (LIDA) facilities at the University of Leeds, UK.

8.3 Results and discussion

We examine the pairwise average adjusted mutual information (AMI) score and $\text{NRSS}_R(X)$ as defined in equation 2.2 on page 36 to identify a suitable number of communities for each ecosystem. A high AMI would indicate that the communities created by the decomposition are stable across repeated runs of the algorithm. Troughs between peaks in our AMI plots would indicate that communities are being split inappropriately. We will interpret the first peak in AMI as the minimum valid number of communities for an ecosystem. Each community will then be labelled according to equation 8.2.

Table 8.2 shows that there are clear differences between the language ecosystems in all the statistics we calculate.Python packages have a mean number of 2.4 ± 2.9 dependencies or dependents (i.e. their mean degree is 2.4 ± 2.9) where JavaScript packages have 9.9 ± 22. Figures 8.2, 8.5, 8.8, 8.11, 8.14 and 8.17 show that the distributions of in- and out-degree look fairly similar in shape across all software ecosystems considered; broadly, all out-degree distributions follow a power-law and the in-degree distributions follow more Poisonnian

distributions. Respectively, this means that a few packages are depended on by most packages, and that packages have similar numbers of dependencies within ecosystems.

Npm, the JavaScript ecosystem, in an outlier here. Its in-degree distribution is bimodal: there are more packages with the highest numbers of dependencies than there are with middling numbers of dependencies. Upon further investigation, it became apparent that the distribution is skewed by a few hundred joke/malicious packages[2] and some irregular packages that list all their recursive dependencies (i.e. the dependencies of their dependencies and so on) as their direct dependencies[3], so they have performed a transitive closure on their dependency networks. We chose not to omit these packages from the decomposition as we hoped that tensor decomposition might discover them as a community; it did not.

We believe that the presence of a rich and well-used standard library—the functions included in the language without any other dependencies—reduces the mean node degree[4] of an ecosystem. Python (mean degree 2.4) has such a library, while JavaScript (mean degree 9.9) emphatically does not.

Elm (mean degree 2.6) features a small standard library compared to Python, but its library is very rich within its domain of web application development.

Rust (mean degree 4.4) features a small standard library by design (instead promoting and curating community packages). R (mean degree 5.3) features a fairly small standard library and many users now prefer the community `tidyverse` collection as a consistent community alternative.

Table 8.2 – Summary statistics for each ecosystem. Number of communities is the number we choose to plot and analyse. For elm-get, CRAN and PyPI, other numbers of communities also seemed reasonable. NRSS is not reported because it did not contribute to analysis. Communities are considered to have declined if their activity at the end of the study period is substantially lower than their peak activity. All numbers are provided to two significant figures. k refers to the in- and out-degrees, σ is the standard deviation and $\overline{k}$ is the arithmetic mean of both the in- and out-degree distributions. R is the number of communities determined to be the optimum.

Package manager	$\approx N$	$\overline{k}$	$\sigma(k_{\text{in}})$	$\sigma(k_{\text{out}})$	R	AMI	% declined
elm-get	1e3	2.6	1.8	42	4	0.67 ± 0.16	50
PyPI	1e6	2.4	2.9	28	6	0.62 ± 0.11	0
npm	1e6	9.9	22	450	3	0.35 ± 0.17	33
Cargo	1e5	4.4	4.2	60	8	0.73 ± 0.087	25
CRAN	1e3	5.3	5.4	110	7	0.61 ± 0.10	43
Maven	1e5	4.0	5.8	120	5	0.62 ± 0.11	20

[2] Such as neat-230

[3] @ckeditor5-build-inline and possibly react-misc-toolbox

[4] NB: the mean in- and out-degrees are always equal as was shown in section 2.2.1.

8.3.1 Ecosystem-specific results

We first provide detailed examination of each ecosystem. A broader overview and comparison between ecosystems is given in section 8.3.2.

Elm

Figure 8.3 shows community activity over time. `evancz/html` is the precursor of `elm-lang/html` and we can see an early community of packages using `evancz/html` that decays over time as the newer community using `elm-lang/html` emerges. We can also see the swift rise of a community of HTTP-using packages. This is perhaps particularly pronounced in Elm because the language is unusually restrictive (there was no official way to write HTTP packages before the package which became `elm-lang/http`) and young (there was a strong known demand for an obvious feature like HTTP support). In a more mature language, one might expect that the rise and fall of communities would be less extreme.

The fall in activity of the `evancz/html` community, shows that many earlier packages were abandoned and did not make the jump to Elm 0.17 and `elm-lang/html`[5]. Those that do make the jump appear in the `elm-lang/html` community as they switch to depending on that package. New packages generally do not depend on them, nor depend on packages that depend on them, explaining the fall in activity.

Elm is a young and deliberately unconventional and experimental language, so it is to be expected that many packages will be abandoned as the community experiments with different approaches.

The rapid changes in activity indicate that the Elm ecosystem is not stable, that is, communities have experienced large relative changes in activity recently.

High AMI for communities two and four suggest that the Elm ecosystem is not diverse as it is well described by small numbers of communities.

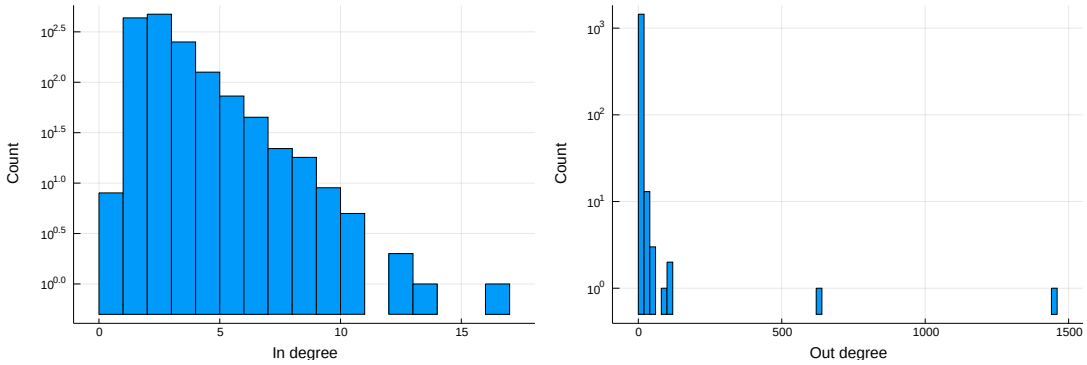

Figure 8.2 – In- and out-degree distributions for the Elm package ecosystem

[5] `evancz/html` is deprecated in Elm 0.17

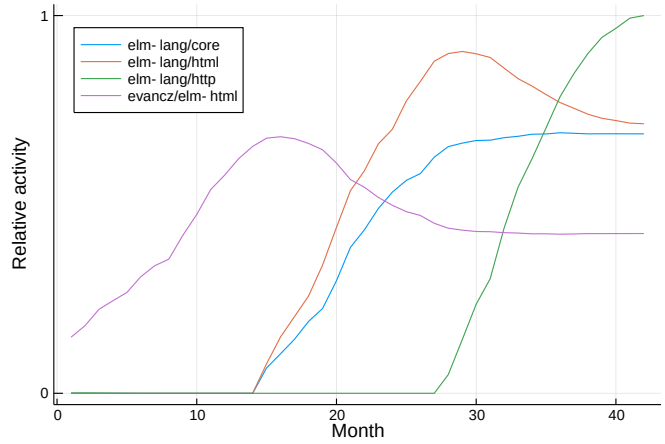

Figure 8.3 – Elm community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

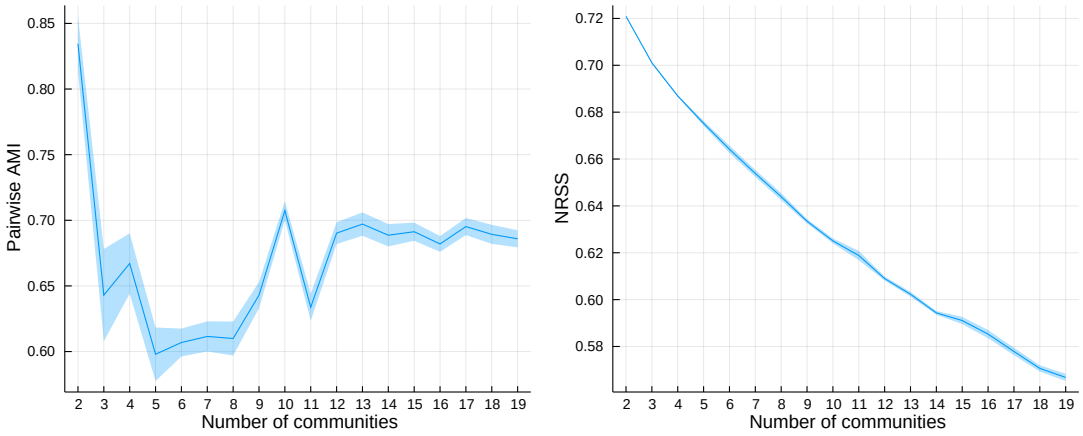

Figure 8.4 – Choosing the number of communities: Elm. Left: the mean pairwise adjusted mutual information of the community-node membership across repeated decompositions. 1 is a perfectly stable assignment to communities; 0 is perfectly random. The shaded area corresponds to twice the standard error of the mean. Right: the mean normalised residual sum of squares error for repeated decompositions on the Elm network against the number of components. The shaded area corresponds to twice the standard error of the mean above and below the line

Python

Figure 8.6 demonstrates how these communities change over time: it is interesting that there are no communities which have declined particularly. One community, `django`, a web application framework, seems to have stagnated. Google Trends data suggest that interest in it has reached a plateau compared to frameworks in other languages such as `React` for JavaScript.

The growth of communities, as shown in figure 8.6, seems to be monotonic. There do not appear to be any major deprecated packages, which is surprising, given the impending deprecation of Python version 2, which is incompatible with Python version 3. This could be

due to the popular package `six` which provides compatibility helpers for Python 2 and 3.

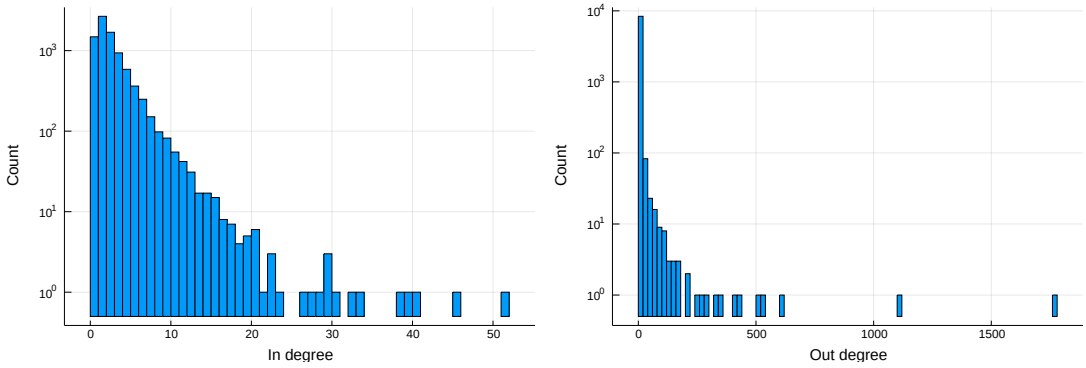

Figure 8.5 – In- and out-degree distributions for the PyPI package ecosystem

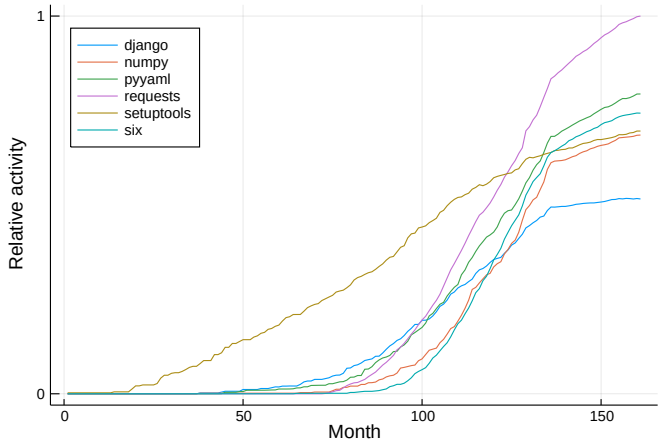

Figure 8.6 – PyPI community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

JavaScript

The AMI plot for npm is given in figure 8.10. We chose not to calculate the NRSS values due to computational constraints due to the far greater size of the dataset corresponding to npm compared to the other ecosystems we considered. The AMI for every number of communities we trialled is very low and although we use three communities in the time activity plot, figure 8.9, we are not particularly confident that the communities picked are meaningful. It is plausible that some larger number of communities would fit the data better, however we trialled up to 50 communities with no success.

Rust

The AMI and NRSS plots for Rust are shown in figure 8.13. The AMI plot strongly indicates a first peak at 8 communities.

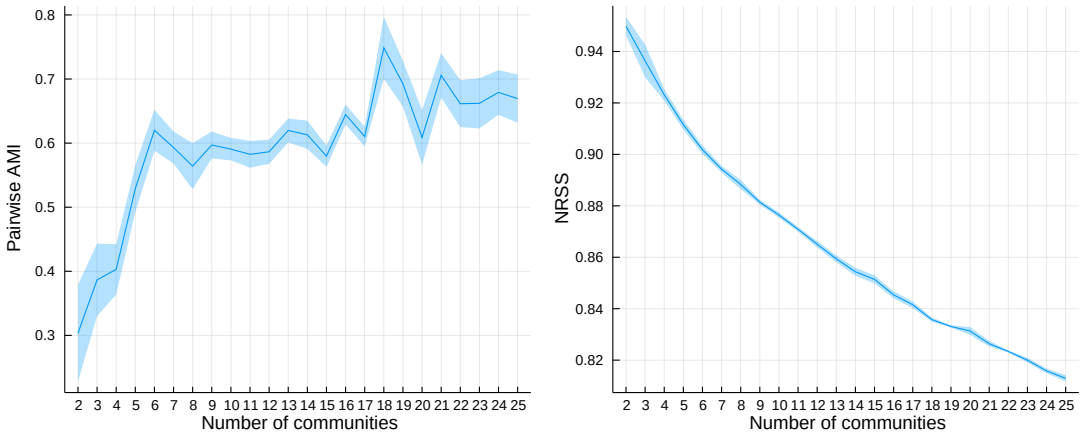

Figure 8.7 – Choosing the number of communities: PyPI. Left: the mean pairwise adjusted mutual information of the community-node membership across repeated decompositions. 1 is a perfectly stable assignment to communities; 0 is perfectly random. The shaded area corresponds to twice the standard error of the mean. Right: the mean normalised residual sum of squares error for repeated decompositions on the PyPI network against the number of components. The shaded area corresponds to twice the standard error of the mean above and below the line

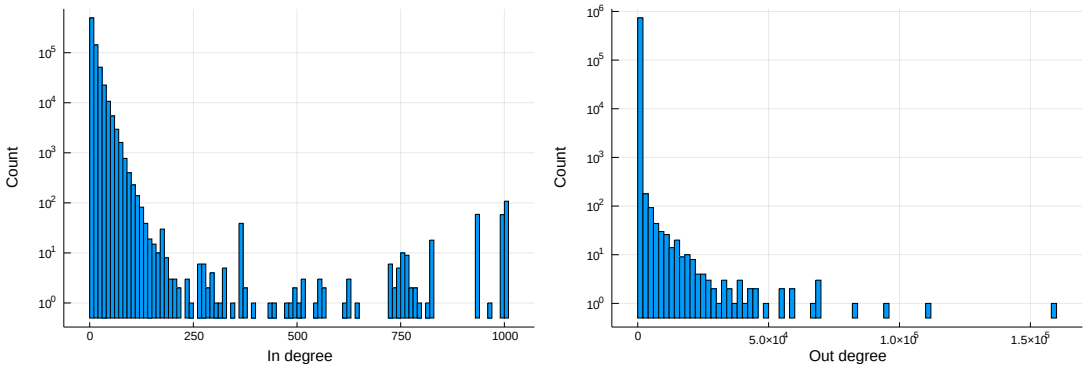

Figure 8.8 – In- and out-degree distributions for the npm package ecosystem

Figure 8.12 demonstrates how these communities change over time. Of particular note are the communities centred around `rustc-serialize` and `serde`. `rustc-serialize` has been deprecated and officially replaced with the `serde` package. The decline in activity of the `rustc-serialize` community and rise of the `serde` community is quite distinct and clearly occurs at the same time.

The community labelled `winapi` is also easily interpretable. The K factor for this community is bimodal and the size of the higher strength cluster indicates that there are around 450 packages in this community. A brief review of these packages suggests that they are low-level cross-platform packages for interacting with the operating system. These packages were amongst the first written and predate the package manager, which explains their steep initial rise.

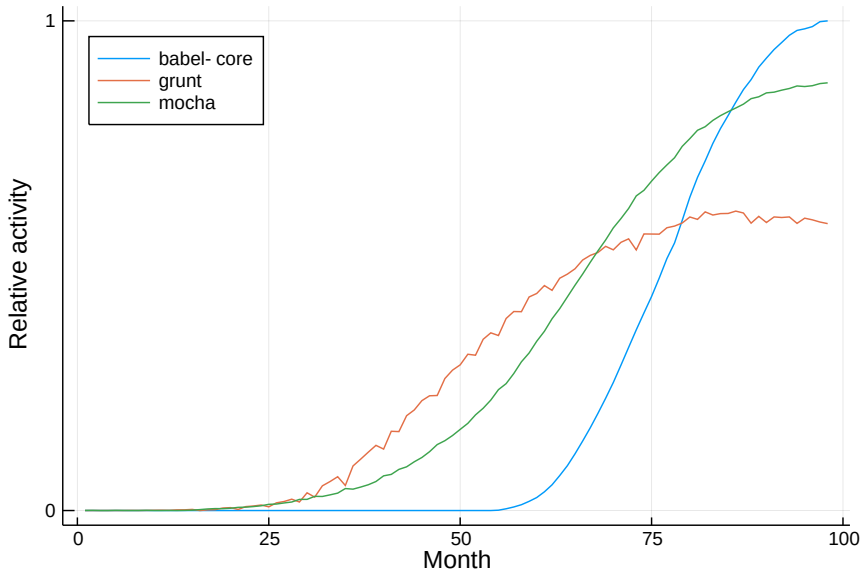

Figure 8.9 – npm community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

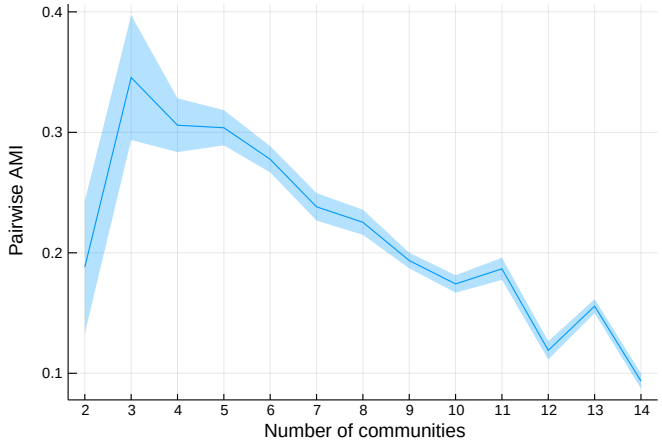

Figure 8.10 – npm adjusted mutual information against components: The pairwise adjusted mutual information of the community-node membership across repeated decompositions. 1 is a perfectly stable assignment to communities; 0 is perfectly random. The shaded area corresponds to twice the standard error of the mean above and below the line

R

As shown in figure 8.16, AMI suggests that decomposition into seven or eight communities is acceptable as this is where the peak is located. That CRAN can be stably decomposed into two communities suggests that the ecosystem is not very diverse, which is as we would expect from a language focused on scientific and statistical computing.

It can be seen in figure 8.15 that when decomposed into seven communities, distinctive waves of activity over time exist. We believe these waves are driven by CRAN's rolling release model

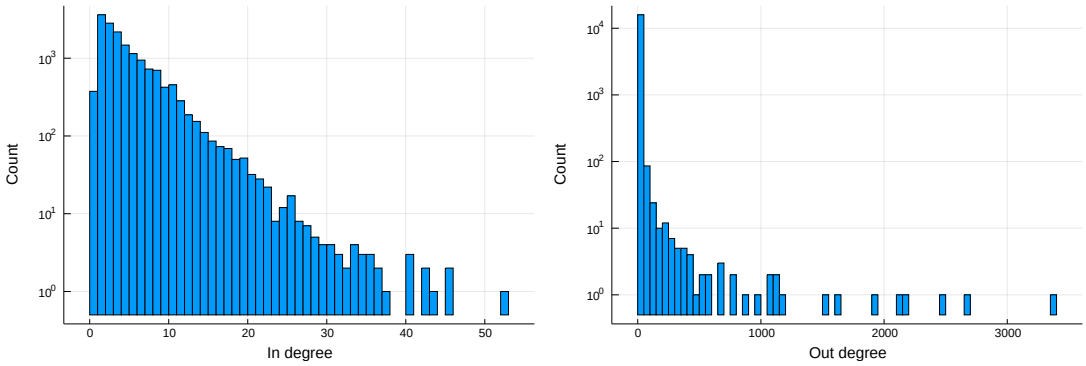

Figure 8.11 – In- and out-degree distributions for the Cargo package ecosystem

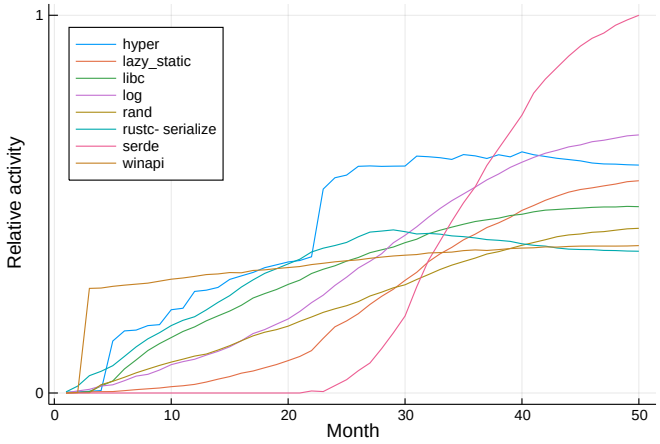

Figure 8.12 – Cargo community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

which requires packages to be actively maintained or de-listed. This is a strong incentive for package authors to switch dependencies if e.g. one of two testing libraries is actively maintained. The de-listing also means that older, unused packages are removed from the dependency graph over time, unlike any of the other studied ecosystems.

Visible in figure 8.15 and an exemplar of this behaviour, `RUnit` is an older unmaintained testing library and `testthat` is a more modern replacement. Unlike any other long-established language in this analysis, there is a clear and steep decline of the earlier library in favour of the competitor[6].

Despite R being an old language, some of its communities have grown at a rapid rate, specifically those relating to `knitr` and `testthat`. This suggests that, despite being an older language, R's ecosystem is still subject to significant change.

Defining community lifetime as the full width at half maximum, we can see from figure 8.15 that communities tend to stay active for approximately 4 to 8 years.

[6]Package popularity indicated on METACRAN: https://cranlogs.r-pkg.org/

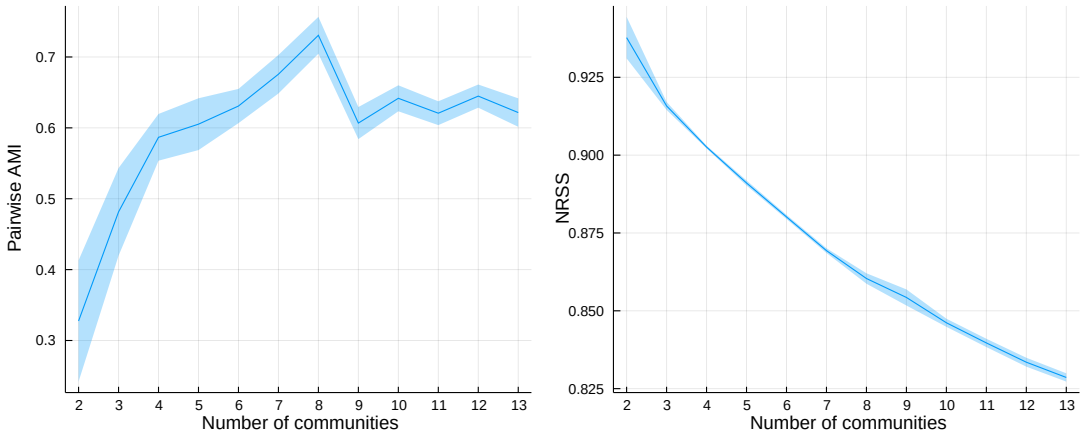

Figure 8.13 – Choosing the number of communities: Cargo. Left: the mean pairwise adjusted mutual information of the community-node membership across repeated decompositions. 1 is a perfectly stable assignment to communities; 0 is perfectly random. The shaded area corresponds to twice the standard error of the mean. Right: the mean normalised residual sum of squares error for repeated decompositions on the Cargo network against the number of components. The shaded area corresponds to twice the standard error of the mean above and below the line

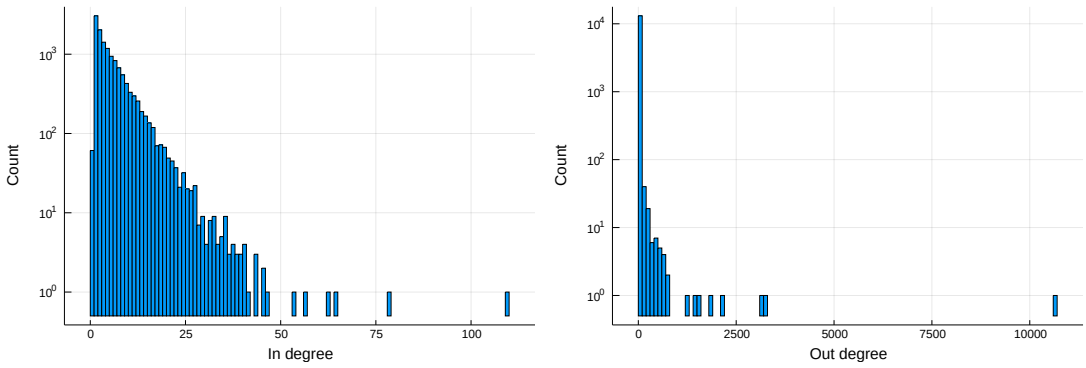

Figure 8.14 – In- and out-degree distributions for the CRAN package ecosystem

Java

The AMI plot, figure 8.19, has an unambiguous peak at five communities. AMI is very low for two communities and comparatively low for three and four, suggesting that Maven has a wider diversity of communities than CRAN or elm-get. These communities include a distinct Scala community, a kind of dialect of Java, centred around the Scala standard library.

Figure 8.18 demonstrates how these communities change over time. There is a general upwards trend and some noise that may be spurious. As noted by Decan *et al.* (2018a), there may be issues with the Maven dataset that are affecting these results.

8.3.2 Summary and inter-ecosystem comparison

In this section we provide a brief summary of the results and compare the ecosystems broadly.

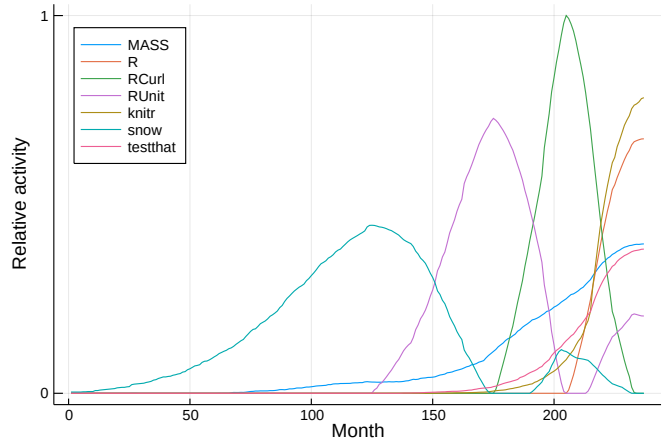

Figure 8.15 – CRAN community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

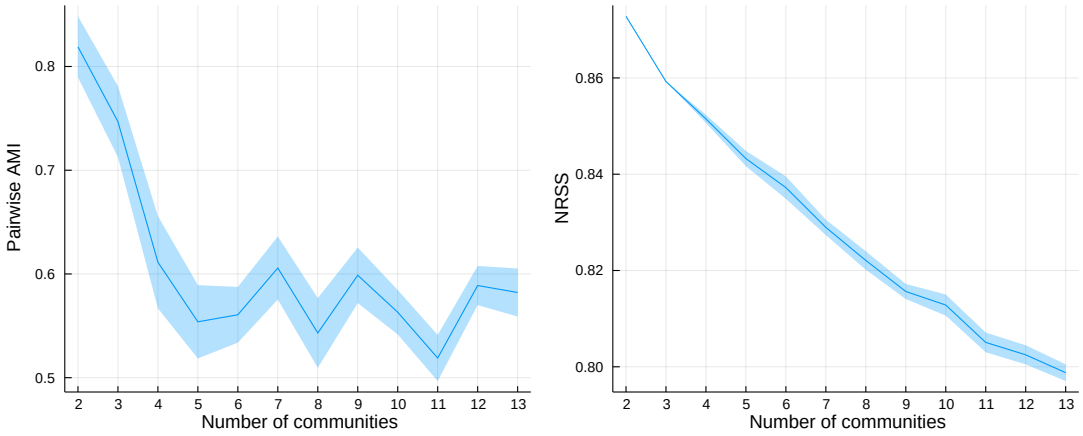

Figure 8.16 – Choosing the number of communities: CRAN. Left: the mean pairwise adjusted mutual information of the community-node membership across repeated decompositions. 1 is a perfectly stable assignment to communities; 0 is perfectly random. The shaded area corresponds to twice the standard error of the mean. Right: the mean normalised residual sum of squares error for repeated decompositions on the CRAN network against the number of components. The shaded area corresponds to twice the standard error of the mean above and below the line

Number of communities

Figures 8.4, 8.7, 8.10, 8.13, 8.16 and 8.19 show the mean AMI and NRSS for each number of communities we evaluated for each ecosystem. We computed 10 decompositions for each number of communities we tested in each ecosystem.

Identifying a suitable number of communities was easiest by looking for early peaks in the AMI plots, which correlated well with visual inspection of decomposition stability mentioned in section 7.1.3 and our manual checks on package names. The NRSS plots have no obvious elbows and contributed nothing to our analysis; we include them as a relevant negative result showing that NRSS is not useful on some real-world networks.

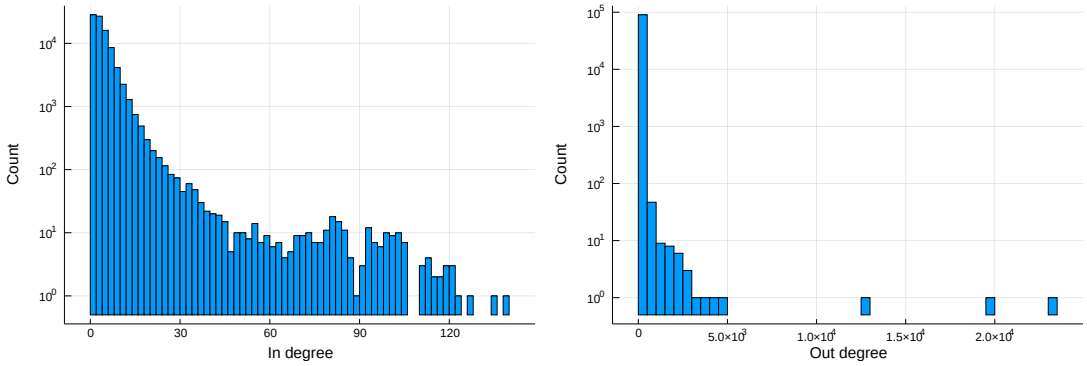

Figure 8.17 – In- and out-degree distributions for the Maven package ecosystem

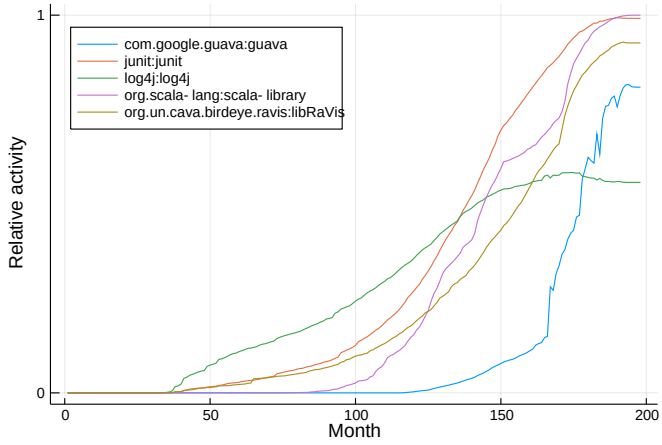

Figure 8.18 – Maven community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

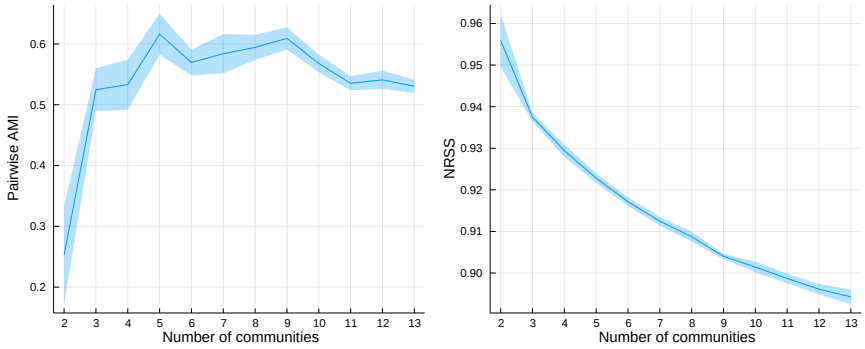

Figure 8.19 – Choosing the number of communities: Maven. Left: the mean pairwise adjusted mutual information of the community-node membership across repeated decompositions. 1 is a perfectly stable assignment to communities; 0 is perfectly random. The shaded area corresponds to twice the standard error of the mean. Right: the mean normalised residual sum of squares error for repeated decompositions on the Maven network against the number of components. The shaded area corresponds to twice the standard error of the mean above and below the line.

For Elm, CRAN and PyPI, AMI indicates more than one 'good' number of communities to

split the ecosystem into. We have chosen the smallest good number except when that number was two.

We observed interesting differences between ecosystems in this metric: Elm and R have relatively high AMIs at relatively low numbers of components while other languages such as JavaScript had low AMIs for all of the numbers of components we tried, suggesting that a more representative number of components might be much higher. This tells us something about the diversity of an ecosystem: the more communities it has, the more diverse it must be, suggesting that R and Elm are less diverse than the other language considered. This matches up neatly with our domain knowledge that Elm is almost exclusively used for creating websites and R for performing statistical analysis, whereas the other languages are much more general purpose.

Activity over time

Figures 8.3, 8.6, 8.9, 8.12, 8.15 and 8.18 show the relative activity of communities in each ecosystem over time for a representative decomposition; i.e. they are plots of τ_{rt} from equation 8.1 against time, t.

The labels are reasonably stable across decompositions when AMI is high and they appear to be informative in most ecosystems. The relative size and activity over time of the communities can often be matched with what we know about the their constituent packages and the ecosystem from other sources.

The community activity plots show clear differences between the various ecosystems. Elm, R and Rust all contain communities that decline in activity over time. Some languages such as Elm, R and Rust have 'peaky' community activity where packages have been replaced by others. Java, Python and JavaScript do not exhibit such behaviour; suggesting that their communities are more stable—their composition does not change much over time in terms of nodes or links between those nodes. The '% declined' column in table 8.2 summarises this behaviour.

Community composition

Comparing ecosystems, we observe that Python has communities centred around making websites (`django`), scientific computing (`numpy`), or interpreting configuration files (`pyyaml`). By contrast, Elm's communities are all to do with building web applications and R's communities are mostly focused towards scientific computing. This suggests, uncontroversially, that Elm and R are less general purpose languages, as their communities are full of packages doing similar things, than Python, where the communities do very different things.

8.4 Conclusions

We have found a large amount of variation between different software ecosystems; some, such as Python, are stable and long-lived and others, such as Elm, have packages that have been important but short-lived due to deprecation. It is our suspicion that a large amount of the variation between languages is due to the size and usefulness of their standard libraries. However, some of the differences, especially those found by the tensor decompositions, could be because early communities which fall out of use are ignored by the decomposition in favour of later, larger communities. It would be interesting to investigate whether this effect is real.

Based on our analysis, we have the following answers to our research questions:

- RQ_1 What are the differences between different software ecosystems?

There are differences in the number of communities of software packages, the rate at which communities of software packages gain and lose popularity as measured by how often they are dependant on each other and in the overall trajectory of the growth of these communities.

- RQ_2 What do communities of commonly-used-together software packages look like?

Similar to the languages as a whole, the communities have fairly homogeneous in-degree distributions but heterogeneous power-law out-degree distributions.

- RQ_3 How do these communities change with time?

The general rule is 'up and to the right': all of the ecosystems we considered increased in activity as time went on, in agreement with Decan *et al.* (2018a). This is unsurprising as it is relatively rare for a package to lose dependencies—it can fall out of contemporary use in that new packages will not choose to depend on it, but the packages that previously depended on it will tend to continue to depend on it as rewriting them to use alternative packages often requires significant effort.

Our results agree with that found in other works, especially that of Decan *et al.* (2018a). Figure 8.15 in particular shows the effect of CRAN's strict rolling-release model where out-of-date packages are archived (meaning that they are no longer as easy to install); figure 8.12 shows how the Rust ecosystem seems to start very suddenly as the package manager came relatively late into the language's life.

For Java specifically, our work does not show much overlap with the communities found by Šubelj & Bajec (2011). This could be due to differences between their static and our dynamic community detection, but it could also be that the validity of our results is questionable for Maven as the `Libraries.io` data for Maven is incomplete (Decan *et al.*, 2018a).

The absolute level of the adjusted mutual information at low numbers of communities seems to reflect the diversity of an ecosystem: R and elm, both domain specific languages, have high levels of AMI for 2 and 3, whereas all of the general purpose languages we consider do not.

Methodologically, we found that using adjusted mutual information as a metric by which to choose the number of communities led to (predominantly) explainable communities that appeared to be stable across multiple runs of the decompositions and across multiple values of the number of communities chosen. Our labelling algorithm, equation 8.2 worked well for our application, giving plausible and comprehensible labels and could plausibly work well for any communities in networks with power-law degree distributions. The stability of an ecosystem can be seen from our activity over time plots, i.e. the $\boldsymbol{\tau}_r$ vectors.

Further work could involve investigating larger components for languages for which we currently have fairly low AMI scores. It would also be fairly straight-forward to extend our technique to consider the co-authorship network that creates the software as another layer in the network. Knowledge of how the ecosystem evolves organically could be used to detect fraudulent packages, especially by unknown authors. The creation of models of software ecosystem evolution from simple sets of rules (for example, 10% of packages are deprecated every 6 months) to try to replicate our results synthetically could also prove insightful.

Our work could have applications in helping designers of software ecosystems to make informed choices; it is clear that, for example, CRAN's rolling release policy has a big impact on its software communities. One could imagine using clusterings like those we have created to determine whether part of a software ecosystem was on the wane and was likely to be replaced soon. Such considerations would make sense if one was choosing dependencies for a project that was intended to last many years. Additionally, if one knows how communities form naturally, it becomes easier to pick out outliers, as we have seen with the npm 'joke' packages.

In conclusion, we have described the long term evolution of several software ecosystems by breaking them up into their constituent communities. We have been able to spot ground-truth events such as the deprecation of major software packages in the temporal activities of these communities. We have demonstrated striking differences between different software ecosystems, particularly with the rolling release cycle of R. Previous work has not had as much success highlighting these differences; we have demonstrated the use of applying state-of-the-art network science techniques for analysing software dependency networks.

In the next chapter we explore whether we can repeat our success in applying tensor decompositions to real-world networks in a completely different field: we explore applying them to networks representing human languages, specifically the English language as recorded by Coleridge *et al.* (2019).

Bibliography

Ai, Jun, Cai, Wei, Zhao, Hai, Li, Hui, & Xu, Jiu-Qiang. 2013. A modular attachment mechanism for software network evolution. *Physica A: Statistical mechanics and its applications*, **392**(9), 2025–2037.

Aynaud, Thomas, & Guillaume, Jean-Loup. 2010. Static community detection algorithms for evolving networks. *Pages 513–519 of: 8th international symposium on modeling and optimization in mobile, ad hoc, and wireless networks*. IEEE.

Bader, Brett W., & Kolda, Tamara G. 2007. Efficient MATLAB computations with sparse and factored tensors. *Siam journal on scientific computing*, **30**(1), 205–231.

Baldwin, Adam. 2018. *Details about the event-stream incident*.

Bezanson, Jeff, Karpinski, Stefan, Shah, Viral B, & Edelman, Alan. 2012. Julia: A fast dynamic language for technical computing. *arxiv preprint arxiv:1209.5145*.

Blanthorn, Oliver, Caine, Colin M, & Navarro-López, Eva. 2019. *Evolution of communities of software: using tensor decompositions to compare software ecosystems*. Unpublished: accepted by Applied Network Science, Springer, 23 August 2019.

Bromberger, Seth. 2017. JuliaGraphs/LightGraphs.jl. Sept.

Cai, Kai Yuan, & Yin, Bei Bei. 2009. Software execution processes as an evolving complex network. *Information sciences*, **179**(12), 1903–1928.

Chaikalis, Theodore, & Chatzigeorgiou, Alexander. 2015. Forecasting java software evolution trends employing network models. *IEEE transactions on software engineering*, **41**(6), 582–602.

Chatzigeorgiou, Alexander, & Melas, George. 2012. Trends in object-oriented software evolution: Investigating network properties. *Proceedings - international conference on software engineering*, 1309–1312.

Chong, Chun Yong, & Lee, Sai Peck. 2015. Analyzing maintainability and reliability of object-oriented software using weighted complex network. *Journal of systems and software*, **110**, 28–53.

Coleridge, Herbert, Furnivall, Frederick, Murray, James, March, Francis, Bradley, Henry, Craigie, William, & Onions, Charles Talbut. 2019. *Oxford English Dictionary (Online)*. Oxford University Press.

Concas, Giulio, Monni, Cristina, Orru, Matteo, & Tonelli, Roberto. 2013. A study of the community structure of a complex software network. *International workshop on emerging trends in software metrics, wetsom*, 14–20.

Decan, Alexandre, Mens, Tom, & Grosjean, Philippe. 2018a. An empirical comparison of dependency network evolution in seven software packaging ecosystems. *Empirical software engineering*, 1–36.

Decan, Alexandre, Mens, Tom, & Constantinou, Eleni. 2018b. On the evolution of technical lag in the npm package dependency network.

Dietrich, Jens, Yakovlev, Vyacheslav, McCartin, Catherine, Jenson, Graham, & Duchrow, Manfred. 2008. Cluster analysis of Java dependency graphs. *Page 91 of: Proceedings of the 4th ACM symposium on software visuallization - softvis '08*. New York, New York, USA: ACM Press.

Durumeric, Zakir, Li, Frank, Kasten, James, Amann, Johanna, Beekman, Jethro, Payer, Mathias, Weaver, Nicolas, Adrian, David, Paxson, Vern, Bailey, Michael, & Halderman, J Alex. 2014. The Matter of Heartbleed. *Pages 475–488 of: Proceedings of the 2014 conference on internet measurement conference*. IMC '14. New York, NY, USA: ACM.

Gao, Yichao, Zheng, Zheng, & Qin, Fangyun. 2014. Analysis of Linux kernel as a complex network. *Chaos, solitons and fractals*, **69**, 246–252.

Gauvin, Laetitia, Panisson, André, & Cattuto, Ciro. 2014. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *Plos one*, **9**(1).

Kang, U, Papalexakis, Evangelos, Harpale, Abhay, & Faloutsos, Christos. 2012. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. *Pages 316–324 of: Proceedings of the 18th ACM sigkdd international conference on knowledge discovery and data mining*. ACM.

Katz, Jeremy. 2018 (Dec.). *Libraries.io Open Source Repository and Dependency Metadata*.

Kluyver, Thomas, Ragan-Kelley, Benjamin, Pérez, Fernando, Granger, Brian E, Bussonnier, Matthias, Frederic, Jonathan, Kelley, Kyle, Hamrick, Jessica B, Grout, Jason, Corlay, Sylvain, *et al.* 2016. Jupyter notebooks-a publishing format for reproducible computational workflows. *Pages 87–90 of: Elpub*.

Kolda, Tamara G., & Bader, Brett W. 2009. Tensor Decompositions and Applications. *Siam review*, **51**(3), 455–500.

Lian Wen, Dromey, R.G., & Kirk, D. 2009. Software Engineering and Scale-Free Networks. *IEEE transactions on systems, man, and cybernetics, part b (cybernetics)*, **39**(4), 845–854.

Louridas, Panagiotis, Spinellis, Diomidis, & Vlachos, Vasileios. 2008. Power laws in software. *ACM transactions on software engineering and methodology*, **18**(1), 1–26.

Myers, Christopher R. 2003. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical review E - statistical physics, plasmas, fluids, and related interdisciplinary topics*, **68**(4), 1–15.

Pan, Weifeng, Li, Bing, Ma, Yutao, & Liu, Jing. 2011. Multi-granularity evolution analysis of software using complex network theory. *Journal of systems science and complexity*, **24**(6), 1068–1082.

Paymal, Prashant, Patil, Rajvardhan, Bhowmick, Sanjukta, & Siy, Harvey. 2011. Empirical Study of Software Evolution Using Community Detection. *Cs.unomaha.edu*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. 2011. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, **12**(Oct), 2825–2830.

Savić, Miloš, Radovanović, Miloš, & Ivanović, Mirjana. 2012. Community detection and analysis of community evolution in Apache Ant class collaboration networks. 229.

Savić, Miloš, Ivanović, Mirjana, & Jain, Lakhmi C. 2019b. Complex Networks in Software, Knowledge, and Social Systems. **148**.

Schlueter, Isaac. 2016a. *kik, left-pad, and npm*.

Šubelj, Lovro, & Bajec, Marko. 2011. Community structure of complex software systems: Analysis and applications. *Physica A: Statistical mechanics and its applications*, **390**(16), 2968–2975.

Xu, Jin, Gao, Yongqin, Christley, Scott, & Madey, Gregory. 2005. A topological analysis of the open source software development community. *Proceedings of the 38th Hawaii International Conference on System Sciences*, **00**(C), 1–10.

Xu, Yangyang, & Yin, Wotao. 2013. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *Siam journal on imaging sciences*, **6**(3), 1758–1789.

Zanetti, Marcelo Serrano, & Schweitzer, Frank. 2012. A Network Perspective on Software Modularity. *Arcs 2012*, 1–8.

Zheng, Xiaolong, Zeng, Daniel, Li, Huiqian, & Wang, Feiyue. 2008. Analyzing open-source software systems as complex networks. *Physica A: Statistical mechanics and its applications*, **387**(24), 6190–6200.

Chapter 9

Linguistic evolution

"Gaéð á wyrd swá hío scel.
Fate goes always as She must"
— Unknown author, *Beowulf c. 990*
and modern translation by B Slade

The English language has existed for approximately 1500 years and has therefore been exposed to massive societal and technological change. Previous work has shown that this exogenous change is visible in the language (Perc, 2012; Cuskley *et al.*, 2014). Our work conducted in chapter 8 proved that tensor decompositions can be applied to study communities in temporal real-world networks. In this chapter, we hope to understand the change in the English language more clearly through representing the language as a temporal network and performing temporal clustering on it.

As shown in figures 9.1 and 9.2, the English language has evolved in a fitful manner; its growth rate has been far from constant with many periods in which expansion was above historical norms. This chapter hopes to explore the causes of these expansions by identifying groups of words which appeared in the language at around the same time. We note that naive approaches, such as splitting the language up into regions, do not give a satisfactory answer except, perhaps, for the last 200 years which may have been driven by colonisation, as may be seen in figure 9.3 where the growth of words increases in regions colonised by the British around the 18th century.

We hope to be able to see the rise and fall of terminology related to specific fields, such as those used in sailing, the automotive industry and computing; we also hope to see changes from exposure to foreign influence, such as the Columbian exchange. This work has the potential to improve our understanding of how groups of words enter and leave the English language and how the rate at which that evolution occurs has changed over time.

This chapter is split into five sections: section 9.1 provides a brief introduction to the surrounding literature; section 9.2 presents the manner in which we present the network and decompose it; section 9.3 displays and discusses our results so far; section 9.4 explores how we might improve our results; and finally section 9.5 compares our work to previous work, providing a short summary.

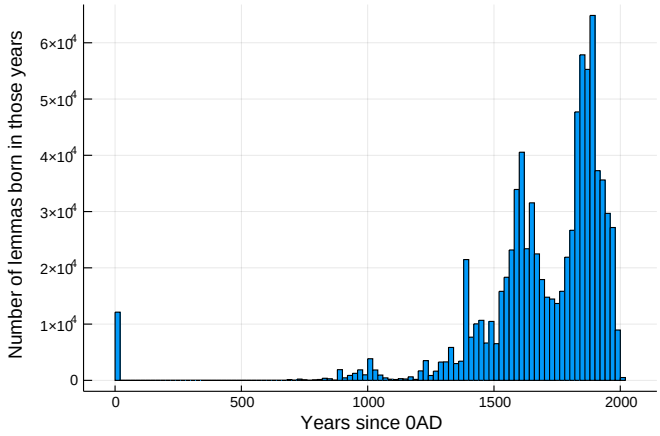

Figure 9.1 – The number of lemmas entering the English language over time as recorded by the Oxford English Dictionary (OED, Coleridge *et al.*, 2019). The large spike at the origin is due to how the OED records words for which there is no known date of first use—it assumes 0AD

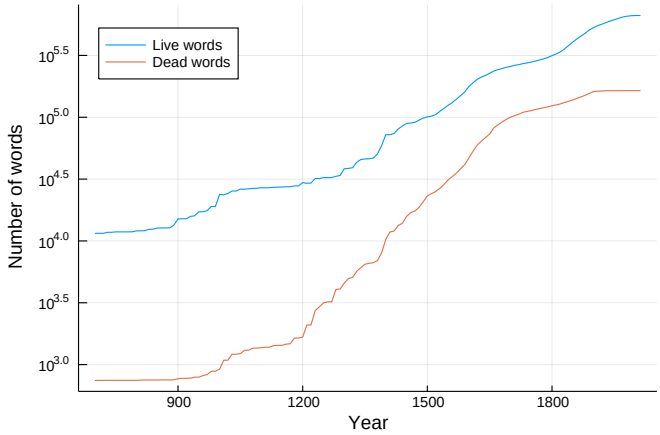

Figure 9.2 – The size of the English language over time: the number of lemmas marked as 'active' or 'inactive' in the Oxford English Dictionary (Coleridge *et al.*, 2019) over time

9.1 Background

As discussed in section 2.9.3 on page 51, there is some related work in this area where authors have applied tensor decompositions to semantic networks (Nakatsuji *et al.*, 2016). The applications are very different, however; the authors are trying to predict how users feel about

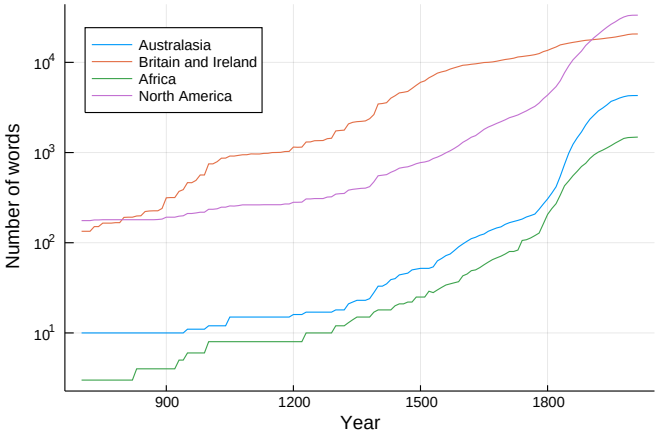

Figure 9.3 – The size of the English language over time per region: the number of lemmas marked as 'active' or 'inactive' in the Oxford English Dictionary (Coleridge *et al.*, 2019) over time, marked as predominantly used in the given region. NB: the given region corresponds to modern usage, so old words which are predominantly used in a different region to where they were first introduced will show up in the 'wrong' region; this explains why North America has many words listed for centuries before a word of English was spoken on that continent

products and activities and there is no temporal factor. There are also works utilising non-negative tensor decomposition to probe the syntax of language albeit again without any temporal factor (Anisimov *et al.*, 2014). There is some literature based on data similar to that which we use but again for completely different purposes; Piao *et al.* (2017), for example, built a programme to tag words to specific lemmas, single meanings of words as was explained in section 2.9.3. Similar data is available (Kay *et al.*, 2009) and some work has been done on visualising it (Alexander, 2012) and looking at the evolution of specific words over time (Alexander & Kay, 2014). There has also been interesting work on the evolution of languages as a syntactic network both from a theoretical (Dorogovtsev & Mendes, 2001) and empirical viewpoint (Solé *et al.*, 2010). For a thorough survey of detecting (rather than analysing) semantic change see Tahmasebi *et al.* (2018). We are unaware of any work applying tensor decompositions to temporal semantic networks in an attempt to better understand their evolution.

9.2 Methods

The bulk of this chapter follows the methods introduced in chapters 7.1 and 8. We made several application specific adjustments.

9.2.1 Building the network

We present a subset of the language from the Oxford English Dictionary (OED, Coleridge *et al.*, 2019) as a temporal network with an undirected adjacency tensor A which comprises

$$a_{ijt} = \alpha_i(t) \cdot \alpha_j(t) \cdot \begin{cases} 1 & \text{if lemma } i \text{ is a synonym or near-synonym of lemma } j, \\ 0 & \text{otherwise,} \end{cases}$$

where

$$\alpha_i(t) = \begin{cases} 1 & \text{if lemma } i \text{ is in usage at time } t, \\ 0 & \text{otherwise.} \end{cases}$$

We define near-synonymy as any words no more than one level above or below each other in the OED's hierarchical thesaurus; in practice this means that things like adverbs and adjectives with the same root (e.g quickly, quick) are linked as well as true synonyms (e.g. fast, quick). This is clearly a network with many disconnected components. There are approximately 800,000 lemmas in the dataset, although only approximately 600,000 of these have any links. There are 12000 lemmas which are so old that there is no estimate of their age; these are assumed to be in usage from 0AD, as shown in figure 9.1. The OED aims to be a comprehensive dictionary for all words in use since 1150 and so we only consider years after that in our analysis. We chose a temporal resolution of 10 years to speed up the computation time as we noted that that was approximately the timescale at which large changes occurred in the language by referencing figure 9.2.

We perform non-negative tensor factorisation (Gauvin *et al.*, 2014) using the APG-TF algorithm (Xu & Yin, 2013) on this network to extract R temporal communities of words. We decompose the the network such that

$$a_{ijt} \approx \sum_{r=1}^{r=R} \kappa_{ir}\kappa_{jr}\tau_{tr},$$

where $K = [\kappa_{ir}]$ signifies strength of lemma membership to communities and $T = [\tau_{tr}]$ gives the temporal activity of the communities. Note that here, unlike chapter 8, we only have two distinct factors rather than three as the network is undirected: $\forall i, j, t : a_{ijt} = a_{jit}$.

9.2.2 Naming the communities

We take a similar approach to that introduced in section 8.2.6. However, because lemmas with the strongest memberships are those with the most synonyms, they may not necessarily be very well known. To aid the ease of interpretation we therefore take the top 10 lemmas with the strongest memberships to each community and sort them in order of decreasing number of quotations listed in the OED, which we use as a crude proxy for how common words are.

We found that naming each community by the top three names in these lists was sufficient for us to distinguish between homonyms, i.e. words with different meanings which are spelt the same, bypass uncommon words and get some idea what the community represented.

9.3 Preliminary results

We found that all choices for the numbers of communities between two and fifty produced very noisy activity plots. This was unexpected, but it could be because the network is split into so many disconnected components. We therefore did not perform the analysis from chapter 7.1 of the optimum number of communities as there was clearly something wrong with our approach. We discuss possibilities for other approaches in the next section, but we did not have time to perform any of them in time for inclusion in this work.

However, we noticed that with a moving arithmetic average corresponding to 15 time intervals, i.e. 150 years, some pattern emerged. We present this smoothed temporal activity for the decomposition with in figure 9.4. There is some promise here: words in the same community appear to be related, with perhaps the exception of the 'farther, absinth, flea-dock' community. It is not clear whether the temporal activities have any bearing in reality; cross-referencing with Google Ngrams it is clear that 'mickle', for example, has been in declining usage, but 'full' and 'more' have been flat and it is not clear that words relating to 'full' and 'more' have dropped out of the language in the last hundred years or so—although it is quite possible that they have and that this author is just unaware of it. A more thorough study would be required.

Additionally, many of the communities are centred around flowers ('argrimony, water lily, woodruff') or insults ('dully, sap-headed, thickheadedness') which may point to a fatal flaw in our analysis: we heavily rely on the number of synonyms a word has—words are given community strengths based on whether they are synonymous with the same words and whether their links appear at the same time. This appears to be a poor approach as the number of synonyms appears to be skewed in favour of plants, for which there are many scientific and common names for similar plants which the OED classes as synonyms, and insults. Our desire to spot the rise of new technologies and cultural exchange has gone unmet.

9.4 Further work

The OED has a hierarchical thesaurus where words are linked to categories, and categories are linked to parent categories in a hierarchical fashion; e.g. 'cat' and 'pussycat' both belong to the 'felis domesticus (cat)' category which itself belongs to the 'family Felidae (feline)' category.

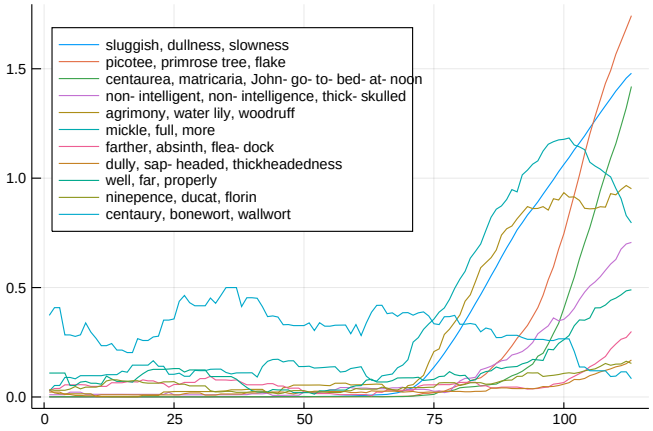

Figure 9.4 – Relative activity of each community at each time interval. The communities are named as detailed in section 9.2.2

We can present this as multi-layer temporal directed network with adjacency tensor

$$a_{i\leftarrow j}^{\alpha t} = \begin{cases} 1 & \text{if node } i \text{ is linked to node } j \text{ at time } t \text{ in layer } \alpha \\ 0 & \text{otherwise,} \end{cases}$$

where all other symbols retain their previous meaning and $\alpha \in \{\text{synonymy}, \text{hierarchical}\}$ denotes specify the type of relationship between two nodes. The hierarchical layer could be organised such that

$$a_{i\leftarrow j}^{\text{hierarchical},\, t} = \chi_i(t) \cdot \begin{cases} 1 & \text{if node } i \text{ is a child of node } j \text{ in the OED hierarchical thesaurus} \\ 0 & \text{otherwise,} \end{cases}$$

where

$$\chi_i(t) = \begin{cases} 1 & \text{if node } i \text{ is a lemma node or it has a path to one at time } t \\ 0 & \text{otherwise,} \end{cases}$$

a recursive definition which can be resolved by iterating up from the lemmas. We hope that this representation may prove more fruitful than the previous one as it relies less on synonymy and puts more weight on hierarchy.

Another possibility is to attempt to vary link strengths based on the rarity of words at each time interval; links in the hierarchy layer could have weights according to the mean rarity of all of their children and links in the synonymy layer could have links with weights based on the minimum rarity of the two nodes to which they are attached. The OED has some static data on rarity but it is not currently in the dataset we possess. There are detailed data on the frequency of words written in printed books over time for approximately the last two hundred years available from Google Ngrams, but these do not account for homonymy.

9.5 Conclusions

This is a much more wide-ranging analysis than previous work, e.g. Alexander & Kay (2014) where only a single category of words was considered (equivalent to our work if we had plotted the in-degree of semantic classes over time). However, we have not been able to extract much meaning out of our analysis, despite some promising preliminary results. We have presented suggestions for how this research could be taken forwards.

Bibliography

Alexander, Marc. 2012. Patchworks and Field-Boundaries: Visualizing the History of English. *Pages 82–83 of: Dh*.

Alexander, Marc, & Kay, Christian. 2014. The spread of RED in the Historical Thesaurus of English. *Colour studies: A broad spectrum*, 126–139.

Anisimov, Anatoly, Marchenko, Oleksandr, Taranukha, Volodymyr, & Vozniuk, Taras. 2014. Semantic and syntactic model of natural language based on tensor factorization. *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, **8455 LNCS**, 51–54.

Coleridge, Herbert, Furnivall, Frederick, Murray, James, March, Francis, Bradley, Henry, Craigie, William, & Onions, Charles Talbut. 2019. *Oxford English Dictionary (Online)*. Oxford University Press.

Cuskley, Christine F., Pugliese, Martina, Castellano, Claudio, Colaiori, Francesca, Loreto, Vittorio, & Tria, Francesca. 2014. Internal and external dynamics in language: Evidence from verb regularity in a historical corpus of english. *Plos one*, **9**(8).

Dorogovtsev, S. N., & Mendes, J. F F. 2001. Language as an evolving word web. *Proceedings of the royal society b: Biological sciences*, **268**(1485), 2603–2606.

Gauvin, Laetitia, Panisson, André, & Cattuto, Ciro. 2014. Detecting the community structure and activity patterns of temporal networks: A non-negative tensor factorization approach. *Plos one*, **9**(1).

Kay, Christian, Roberts, Jane, Samuels, Michael, & Wotherspoon, Irené. 2009. *Historical thesaurus of the Oxford English dictionary*. Oxford University Press.

Nakatsuji, Makoto, Toda, Hiroyuki, Sawada, Hiroshi, Zheng, Jin Guang, & Hendler, James A. 2016. Semantic sensitive tensor factorization. *Artificial intelligence*, **230**, 224–245.

Perc, Matjaž. 2012. Evolution of the most common english words and phrases over the centuries. *Journal of the royal society interface*, **9**(77), 3323–3328.

Piao, Scott, Dallachy, Fraser, Baron, Alistair, Demmen, Jane, Wattam, Steve, Durkin, Philip, McCracken, James, Rayson, Paul, & Alexander, Marc. 2017. A time-sensitive historical thesaurus-based semantic tagger for deep semantic annotation. *Computer speech and language*, **46**, 113–135.

Solé, Ricard V, Corominas-Murtra, Bernat, Valverde, Sergi, & Steels, Luc. 2010. Language networks: Their structure, function, and evolution. *Complexity*, NA–NA.

Tahmasebi, Nina, Borin, Lars, & Jatowt, Adam. 2018. Survey of Computational Approaches to Lexical Semantic Change.

Xu, Yangyang, & Yin, Wotao. 2013. A Block Coordinate Descent Method for Regularized Multiconvex Optimization with Applications to Nonnegative Tensor Factorization and Completion. *Siam journal on imaging sciences*, **6**(3), 1758–1789.

Part V

Conclusions

Chapter 10

Summary and discussion

"Nothing except a battle lost can be half so melancholy as a battle won."

— The Duke of Wellington,
letter from the field of Waterloo

Throughout this thesis, we have shown the value of characterisation at many different scales in networks. This final part will try to wrap up our work and suggest future avenues we or others might take. Here, we provide a recapitulation, highlighting some of our key findings, and explore areas where we think improvements could be made. This chapter is split into four sections: one for each of the three main parts of the thesis and a final section presenting our overall conclusions.

10.1 Node characterisation

Our first area of contribution in the thesis was in node characterisation. Building on previous work that had shown that the use of ego-networks to characterise nodes was a promising technique, we demonstrated the benefits of considering where the ego lay in these networks through the use of colour. In an academic co-authorship network, we found that the proportion of coloured graphlets of a certain kind—a star centred on the ego with 3 links to alters—were the second strongest predictors of career length of all the topological measures we tried, and that there were no statistically significant links between monochromatic graphlets and career length.

There is an obvious criticism here in that we did not include other real-world properties of the authors which could have confounded our analysis. Unfortunately, the disambiguated network we use did not contain that information and the authors were irreversibly anonymous. It would be interesting to see if the result held if 'total papers published' was also included in our analysis; to our eyes the 4.3b graphlet, shown again in figure 10.1, looks suspiciously like it

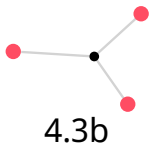

Figure 10.1 – Graphlet 4.3b from figure 4.4, the only graphlet found to have a significant link with academic career length, after controlling for many standard node centralities

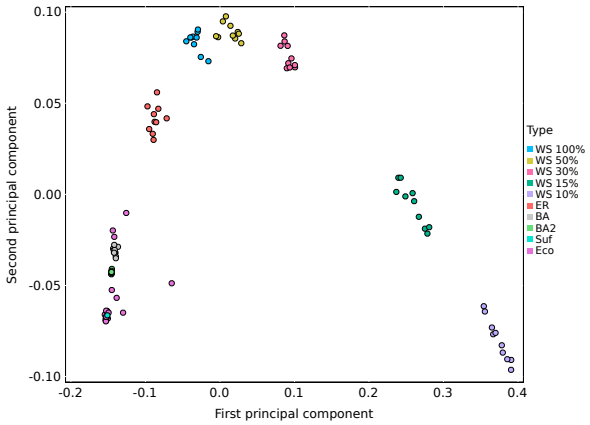

Figure 10.2 – PCA applied to the degree mixing matrices of various networks, as detailed in chapter 5, projected onto their first two principal components. Each point corresponds to a single network

could simply be a proxy for the number of papers published as it is the graphlet which requires the most papers published (if there was a link between any of the alters, that would reduce the number of papers published). The topology only explains a small fraction of research career length, so academic career length isn't solely determined by the topology of the co-authorship network, which is to be expected.

10.2 Network characterisation

We have shown that so-called 'unbiased' descriptions of networks can be used as an input to matrix decompositions of networks to highlight similarities and differences to them. We provided some indications that these 'unbiased' inputs were about as good as 'biased' ones but did not test it very rigorously. We showed that our technique can successfully distinguish between configuration models and original networks. An example of using the technique to track network damage was given. The similarities and dissimilarities between some real-world networks and synthetic networks were shown, repeated here in figure 10.2.

We built on our work in the first part and again showed that coloured graphlets have a far greater richness to them than monochromatic graphlets. The figures relating to the academic co-authorship network are hard to interpret. We suspect this is because, as was found in the first part, the topology only accounts for a small amount of variation in career length

and career length is hardly the only variable describing academic researchers. An interactive version of the plot, where one could hover over outliers and see who they were, could be interesting.

10.3 Community characterisation

In the previous part, we explored the use of tensor decomposition to characterise communities in large, real-world temporal networks. We found that existing techniques for determining sensible numbers of communities to decompose these networks into were not suited to the networks we considered and devised a novel one. This technique uses adjusted mutual information to determine which numbers of communities lead to the formation of similar communities on repeated runs, with the logic that if an optimising method with random elements finds similar communities each time, then these are plausible.

We applied this technique to several software dependency networks and found we could see interesting real-world behaviour reflected in the activity of these communities. Particularly, we could see the deprecation of packages and their effect on the network and we could observe how some of the rules governing some software dependency ecosystems shaped the activity of their communities over time, with the rolling-release cycle of R especially well-reflected, shown again in figure 10.3.

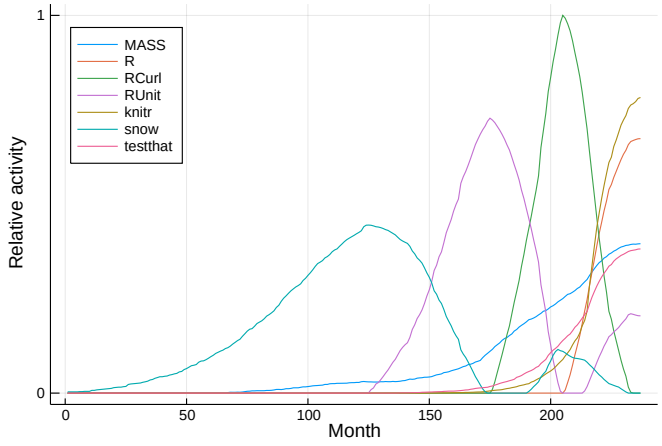

Figure 10.3 – CRAN community activity: Relative activity of each community in each month. The communities are named by important packages as detailed in the Methods section

We also demonstrated how this technique could be applied to semantic networks: we investigated how we would go about applying it to study the evolution of the English language. We hoped we would be able to see the impact of new technologies and societal change in the language but our results so far have been hard to interpret. We have provided suggestions for how we might proceed.

10.4 Summary

We have made contributions at three scales of characterisation: node (local), community (mid-range) and whole network (global). Our exploration of graphlet profiles to characterise nodes through their ego-networks led us to consider how we could characterise whole networks through their graphlet profiles; our investigation of the use of matrix decomposition on graphlet profiles led neatly to the concept of tensor decompositions, albeit applied to networks in a totally different way.

We hope that our work encourages more people to use tensor decompositions to analyse large real-world networks; our technique should help others to have the confidence to use decompositions in the knowledge that there are metrics by which they can evaluate their choice of the number of communities. Regarding our applications, we believe that there are many interesting problems that may have more light shed on them through applying network science techniques to software dependency networks.

We feel that we have only scratched the surface of what is possible. In our next and final chapter we shall enumerate some of these possibilities.

Chapter 11

Further work

"Adapt or perish, now as ever, is Nature's inexorable imperative."

— HG Wells, *The Mind at the End of its Tether*

There is room to expand on almost every part of this thesis. Each discovery has led us to more questions. In this final chapter, we explore some of these areas we think would be interesting for us and perhaps other researchers to explore.

11.1 Network characterisation

There are three main directions where our network characterisation could be improved.

11.1.1 Other uses of data matrices

As was mentioned in chapter 5, many different machine learning techniques can be applied to data matrices. DeepGL (Rossi *et al.*, 2017) is one example of work that has already attempted to do something similar by applying deep-learning techniques to networks. We are somewhat reluctant to go so far as our technique was able to distinguish well between many types of network without requiring labels. Indeed, we think the best way forward for our technique is to apply it to as many real-world networks as possible to see what sort of clusters emerge, if any.

11.1.2 Construction of synthetic networks

Any point in the transformed co-ordinates, $\hat{\boldsymbol{t}}$, from PCA can be reprojected back into the original space with

$$\hat{\boldsymbol{x}} = \sum_k t_k \boldsymbol{w}_k.$$

If we apply this to the principal components of adjacency matrices or Laplacian matrices, we can create synthetic weighted & directed networks from any point in the reduced space. We can therefore create a network 'midway' between two other kinds of network. It would be interesting to see what those look like; in the facial recognition subfield of computer vision these reconstructed examples are called 'eigenfaces' (Turk & Pentland, 1991). With absolutely no imagination required, we would dub them eigennetworks.

11.1.3 Quantitative comparison of our work with related work

There have been several related attempts to characterise networks through dimension reductions (Costa *et al.*, 2007; Bounova & de Weck, 2012; Ronqui & Travieso, 2015; Schieber *et al.*, 2017). There has not yet been an attempt to quantify the success or otherwise of these techniques or to compare them in a thorough manner. An experiment could be devised with a straightforward collection of synthetic and real networks from several categories. A clustering algorithm such as DBSCAN (Ester *et al.*, 1996) would then be run in the reduced spaces and the adjusted mutual information (AMI) between the clustering and the ground-truth labels could be determined. The dimension reduction with the highest AMI could be considered the best.

11.2 Graphlets

There are two main areas where we feel our work on graphlets could be extended.

11.2.1 Graphlet profile normalisation performance

Graphlets may appear in networks for uninteresting reasons: for example, in a denser graph, denser graphlets are likely to appear. We need some kind of benchmark by which we can tell how unusual a graphlet is. Most of the current approaches in the literature use the so-called z-score (Milo *et al.*, 2002): the number of standard deviations away from the mean the number of appearances of a graphlet are compared to an ensemble of 'similar' random networks. Kavosh and acc-Motif Kashani *et al.* (2009); Meira *et al.* (2014) use an ensemble of configuration models, as described in section 2.9.2. These routines are predominantly designed for use on a few networks; the authors are not particularly concerned about the performance of calculating graphlet profiles for many networks. They recommend generating a thousand similar graphs per graph.

That is too slow for many purposes where one wishes to compare many hundreds or thousands of real networks. We propose a faster, approximate technique: instead generate ten random graphs for each decile of density and interpolate the number of graphlets between them and

then calculate the z-scores for each graphlet in a real network,

$$z_i = \frac{\mu_i(\rho) - x_i}{\sigma_i(\rho)},$$

where

$$\begin{aligned}
\mu_i(\rho) &= w_-(\rho)\overline{x_{i\rho_-}} - w_+(\rho)\overline{x_{i\rho_+}}, \\
\sigma_i(\rho) &= w_-(\rho)\sigma(x_{i\rho_-}) - w_+(\rho)\sigma(x_{i\rho_+}), \\
w_+(\rho) &= \frac{\rho - \rho_+}{\rho_+ - \rho_-}, \\
w_-(\rho) &= \frac{\rho - \rho_-}{\rho_+ - \rho_-}, \\
x_i &= \frac{n_i + 1}{(\Sigma_j n_j) + \max(j)},
\end{aligned} \tag{11.1}$$

where z_i is the z-score of the i^{th} graphlet which occurs n_i times in a network of density ρ, ρ_+ and ρ_- are the densities of the closest ensembles of random graphs, $\overline{x}$ is the arithmetic mean of the set x and $\sigma(x)$ is its standard deviation and $x_{i\rho_+}$ and $x_{i\rho_-}$ are sets corresponding to x_i in each of the ten random graphs in the ensembles for the closest densities matching the graph of interest. A small but important detail that has not been covered by the existing literature is how to calculate the z-score of a graphlet that has never appeared in the ensemble. To do this, we suggest using the rule of succession (Zabell, 1989) which leads to equation 11.1, assuming *a priori* that each graphlet is equally likely to appear and updating that assumption in the face of additional evidence.

Any research investigating this novel graphlet normalisation procedure should carefully consider the implications of controlling only for density where existing methods control for degree sequences. It should be noted that in the limit of a large number of networks, this procedure is N times faster than the existing procedure, where N is the number of configuration models generated per graph, commonly a thousand.

11.2.2 Academic career length from collaboration network improvements

As we discussed in chapter 4, the graphlet most related to career length was the star graphlet, 4.3b, with no links between the co-authors of the author of interest. This graphlet is the one which requires the highest number of papers to be authored. A clear improvement, then, would be to include the number of papers written alongside other characteristics such as citation count to see whether the graphlet is still related to career length, taking those other factors into account. However, this could all be of limited interest, as the variance in that graphlet can only explain approximately a year of difference in career length.

Additionally, the network lends itself quite naturally for analysis through the lens of tensor decomposition. An additional layer representing papers, with links between the author and paper layer denoting authorship and links between papers denoting citation could also prove interesting to analyse; one could potentially spot fields forming and fading away.

11.3 Community characterisation

Our analysis of software ecosystems could be deepened by adding a layer relating authors to the packages to which they have contributed. Most of these packages are developed in the open, so these contributions could be weighted by the number of patches applied. It would be particularly interesting to see what happens to authors when they cease contributing to one package. Authors often contribute to more than one language ecosystem, so another enhancement could be to consider all software ecosystems together to try to analyse how authors move between them. The patterns we observed, particularly the rolling-release cycle of R, could lend themselves to simple theoretical models of ecosystems. Such models could be of great interest to those creating software ecosystems.

Similarly, our linguistic network could be expanded to consider multiple languages, with a layer for each language, where links between layers denote synonymy. One could then potentially observe the cross-pollination between languages which would be a fascinating endeavour.

In general, we feel that there is a large room for tensor decompositions to be applied to more data. For example, Reale *et al.* (2019) provide a fascinating exploration of the evolution of the study of ecological networks, but they study the network as a series of piecewise static networks. We believe that tensor decompositions would have greater insight in cases such as these.

A further symptom of the immaturity of tensor decompositions in network science is the lack of polished software. Multilayer and temporal networks are already somewhat neglected and tensor decompositions are no exception. Reducing the barrier to entry so that non-specialists could apply these dynamic community detection algorithms to their own networks could yield dividends in terms of the amount of output produced.

Lastly, it would be beneficial if there was a theoretical basis as to the soundness of using adjusted mutual information to choose the number of components into which a tensor should be decomposed. Our current arguments have only really suggested that it is plausible, which is supported by our results. Further work could also consider why CORCONDIA (Bro & Kiers, 2003) was not useful for picking the number of components in our applications, which appears to be at odds with other work (Sapienza *et al.*, 2018).

11.4 Characterisation of complex networks

There is a wonderful survey of the characterisation of complex networks (Costa *et al.*, 2007). At the time of writing, it is 12 years old and so misses out almost all of the works cited in this thesis. It is particularly silent on the application of machine learning techniques to networks, many of which had not been popularised at the time of its publication, although there are more recent reviews specifically on characterising nodes for machine learning (Hamilton *et al.*, 2017; Goyal & Ferrara, 2018) they do not consider network or community characterisation. The work of Costa *et al.* is well due an update.

11.5 Interplay between topology and node dynamics

As we have intimated in previous chapters, we believe that complex networks are interesting predominately because they reflect real-world networks. In these real-world networks, we rarely if ever care directly about the network topology: it is the topology's impact on the observable properties of the system, such as the amount of time it takes to travel from one place to another, or the diversity of an ecosystem, that we want to optimise. It is therefore crucial to link the topology to these observable properties, which is what we have done in chapters 4, 8 and 9. What we have not done is linked the topology to models of the dynamics that generate these observable properties. This is a very active field—see Menck *et al.* (2013) and Gao *et al.* (2016) for two examples—but little work has been done in terms of characterising networks by considering what sort of dynamics they support or encourage. We would like to see how well methods of characterising networks correlate with dynamical properties such as the resilience of the system.

■

Bibliography

Bounova, Gergana, & de Weck, Olivier. 2012. Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles. *Physical review E*, **85**(1), 016117.

Bro, Rasmus, & Kiers, Henk A.L. 2003. A new efficient method for determining the number of components in PARAFAC models. *Journal of chemometrics*, **17**(5), 274–286.

Costa, Luciano da F., Rodrigues, Francisco A., Travieso, Gonzalo, & Villas Boas, P. R. 2007. Characterization of complex networks: A survey of measurements. *Advances in physics*, **56**(1), 167–242.

Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei, *et al.* 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *Pages 226–231 of: Kdd*, vol. 96.

Gao, Jianxi, Barzel, Baruch, & Barabási, Albert-László. 2016. Universal resilience patterns in complex networks. *Nature*, **530**(7590), 307–312.

Goyal, Palash, & Ferrara, Emilio. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-based systems*, **151**, 78–94.

Hamilton, William L, Ying, Rex, & Leskovec, Jure. 2017. Representation learning on graphs: Methods and applications. *arxiv preprint arxiv:1709.05584*.

Kashani, Zahra Razaghi Moghadam, Ahrabian, Hayedeh, Elahi, Elahe, Nowzari-Dalini, Abbas, Ansari, Elnaz Saberi, Asadi, Sahar, Mohammadi, Shahin, Schreiber, Falk, & Masoudi-Nejad, Ali. 2009. Kavosh: a new algorithm for finding network motifs. *Bmc bioinformatics*, **10**(1), 318.

Meira, Luis AA, Máximo, Vinícius R., Fazenda, Álvaro L., & Da Conceição, Arlindo F. 2014. Acc-motif: accelerated network motif detection. *IEEE/ACM transactions on computational biology and bioinformatics (TCBB)*, **11**(5), 853–862.

Menck, Peter J., Heitzig, Jobst, Marwan, Norbert, & Kurths, Jürgen. 2013. How basin stability complements the linear-stability paradigm. *Nature physics*, **9**(2), 89–92.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., & Alon, U. 2002. Network Motifs: Simple Building Blocks of Complex Networks. *Science*, **298**(5594), 824–827.

Reale, Denis, Khelfaoui, Montiglio, Pierre-Olivier, & Gingras, Yves. 2019 (Aug.). *Mapping the dynamics of research networks in ecology and evolution using co-citation analysis (1975–2015)*.

Ronqui, JRF, & Travieso, Gonzalo. 2015. Analyzing complex networks through correlations in centrality measurements. *Journal of statistical mechanics: Theory and experiment*, 9.

Rossi, Ryan A., Zhou, Rong, & Ahmed, Nesreen K. 2017. Deep Feature Learning for Graphs. 1–11.

Sapienza, Anna, Bessi, Alessandro, & Ferrara, Emilio. 2018. Non-negative tensor factorization for human behavioral pattern mining in online games. *Information (switzerland)*, **9**(3).

Schieber, Tiago A, Carpi, Laura, Díaz-Guilera, Albert, Pardalos, Panos M, Masoller, Cristina, & Ravetti, Martín G. 2017. Quantification of network structural dissimilarities. *Nature communications*, **8**, 13928.

Turk, Matthew, & Pentland, Alex. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience*, **3**(1), 71–86.

Zabell, Sandy L. 1989. The rule of succession. *Erkenntnis*, **31**(2-3), 283–321.

Index