Statistical Network Analysis

Prof. Dr. Ingo Scholtes

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

ingo.scholtes@uni-wuerzburg.de

Lecture 01
Motivation

October 20, 2021

Notes:

- **Lecture L01: Motivation** 20.10.2021
- **Educational objective:** This motivating lecture gives an overview of the course. We further introduce why the network perspective is important for the analysis of complex systems across disciplines.
  - Introducing the Chair of Informatics XV
  - From data science to network science
  - A brief history of network science
  - Overview of the course

Another unusal semester

- welcome to our course in an **almost normal** winter semester
- following a faculty recommendation, first session held in virtual format
- please respect the **CoViD-19 rules**
  - 3G rule applies on Campus
  - masks mandatory if distance of 1.5 m cannot be maintained
  - contact tracing via UniNow app
  - up-to-date information at

    → https://www.uni-wuerzburg.de/corona/studierende/
- preference for **presence, hybrid, or mixed format** in coming weeks?

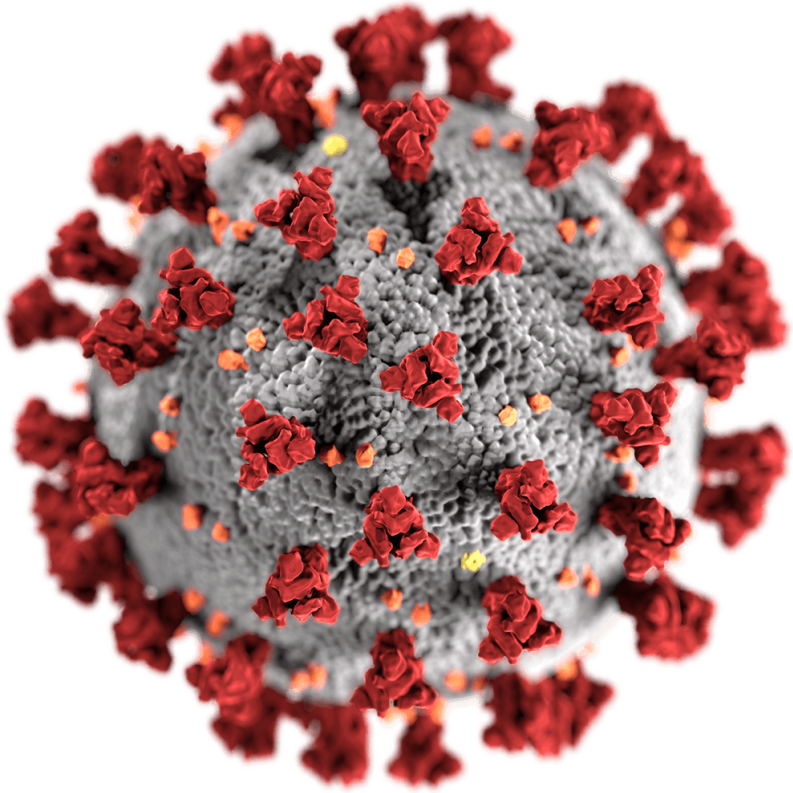


illustration of SARS-CoV-2 virus

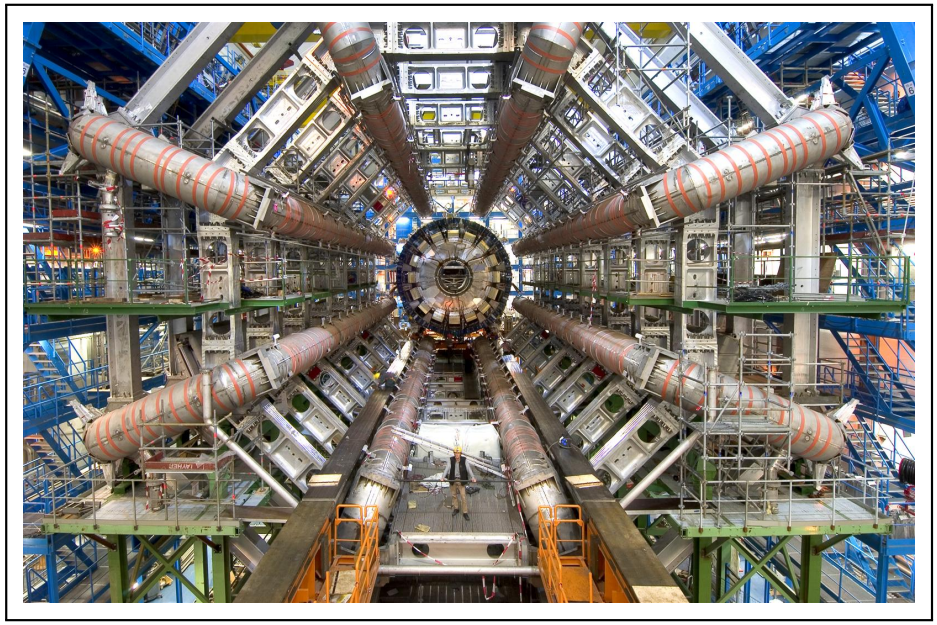
image credit: CDC/ Alissa Eckert, MS; Dan Higgins, MAMS, public domain

Notes:

- Before introducing our chair and our course, I welcome you to an almost normal winter semester. Be assured that your health is our primary concern and we will not do anything that puts your or our health at risk.
- Please respect the current CoViD-19 rules that apply on Campus!
- We understand that you may face challenges, e.g. regarding the infrastructure/office available at home. We do our best to make participation in this course convenient for you. Following a recommendation by the faculty, we have decided to hold the first lecture in a virtual format. We will be happy to consider your feedback and preference about presence sessions in the coming weeks.
- In any case, please respect the Webiquette, copyright and privacy regulations! You are not allowed to secretly record meetings or sessions.

Who am I?

- **diploma in Computer Science** from University of Trier (2005)
- designed **large-scale distributed Event Monitoring system** of ATLAS detector at CERN (2004–2008)
- **doctorate degree** in the area of massive-scale distributed systems from University of Trier (2011)
- **postdoc** at Chair of Systems Design at ETH Zürich (2011 – 2016)



the ATLAS detector, recording data on 40 million particle-particle collisions per second at 2 MB per collision

image credit: CERN

Notes:

- Before introducing the topic of this course, let me introduce myself and the Chair of Machine Learning for Complex Networks (since I only joined University of Würzburg eight weeks ago).
- I am 41 years old, and studied computer science at the University of Trier from 2000 to 2005. I entered the field of large-scale data processing and analysis in 2004, when I started a joint project with CERN. Between 2004 and 2008 I developed the Event Monitoring System of the ATLAS detector, which is now being used to distribute particle collision event data to thousands of physicists.
- In 2011 I completed my doctorate degree at the University of Trier. My thesis focussed on the modelling of complex structures and dynamical processes in networks.

Who am I?

- **Vertretungsprofessor** at Karlsruhe Institute of Technology (2016)
- **senior assistant and lecturer** at ETH Zürich (2017 – 2018)
- **SNSF Professor of Data Analytics** at University of Zurich (since 2018)
- **Professor of Data Analytics** at University of Wuppertal (2019 – 2021)
- **Professor of Machine Learning for Complex Networks** at Julius-Maximilians-Universität Würzburg (since Aug. 2021)

Alte Mainbrücke in Würzburg

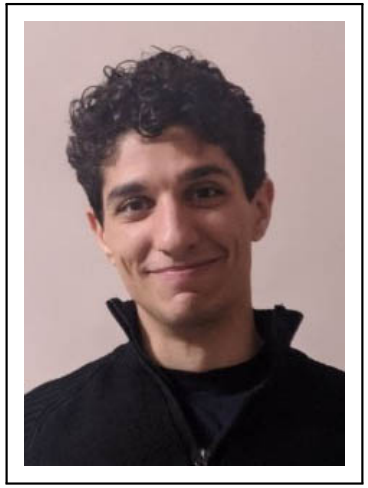
image credit: Ingo Scholtes

Notes:

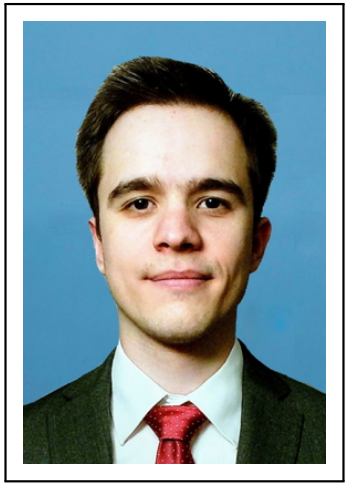
- I continued my focus on data analysis and complex networks during a postdoc at ETH Zürich. At ETH Zürich, I developed a first version of this course under the name “Complex Networks’ back in 2014. This course was mainly targeted at an interdisciplinary audience of physicists, mathematicians, and life scientists and became increasingly popular over the years.
- In 2018 I joined University of Zürich as an SNF Professor, where I revised the course at to focus on “Statistical Network Analysis”. In 2019 I joined University of Wuppertal, where I taught the same course in a 4+2 hr format, i.e. two lectures and one exercise session per week.
- Since August 2021 I hold the Chair of Machine Learning for Complex Networks at University of Wuppertal. Different from the previous edition at University of Wuppertal, this year’s course will be a 2+2 hr format, i.e. one lecture and one exercises session per week.
- For the first time, practical demonstrations in `python` will be consistently integrated into the lectures. Please be patient in case there are any issues!

Chair of Informatics XV

Dr. Christoph Gote[1]



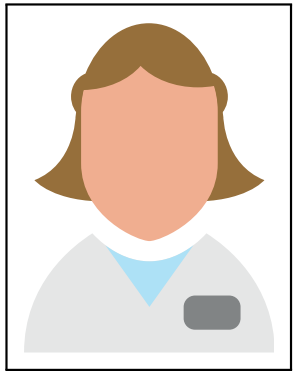
Vincenzo Perri[2]



Luka Petrovic[2]

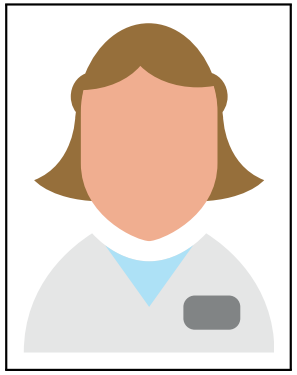
Dr. Jürgen Hackl[2,3]

Prof. Dr. Ingo Scholtes[2,4]

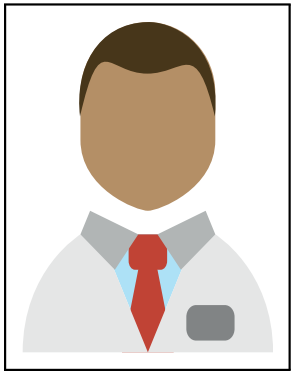


Leonore Röseler[2]

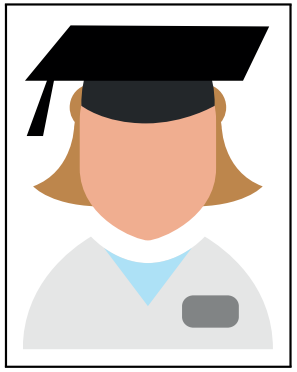
Dr. Unai Alvarez-Rodriguez[2]



N.N.[4]



N.N.[2]



Dr. N.N.[4]

[1] *Chair of Systems Design, ETH Zürich* / [2] *Department of Informatics, Universität Zürich*
[3] *Assistant Professor at University of Liverpool* / [4] *Chair of Informatics XV, University of Würzburg*

Notes:

- The Chair of Machine Learning for Complex Networks is an international and interdisciplinary team with two physical locations. Five members of our team are employees at the Department of Informatics at the University of Zurich. One external team member is employed at ETH Zürich but closely working with us. Our postdoc Jürgen Hackl is also an assistant professor at the University of Liverpool. The nationalities of our team include German, Austrian, Italian, Spanish, Serbian and soon Turkish, Kosovarian, and Singaporian! Our team members have backgrounds in computer science, civil engineering, control engineering, psychology, mathematics and theoretical physics. We appreciate the different view points and interests that results from this diversity. However, all members of our team speak a common "scientific language", which is the language of statistical data analysis, machine learning, and mathematical and computational modelling.
- We are still in the process of building the team and there are still open positions for PhD students and postdocs. If you know someone that may be interested in one of those positions, feel free to refer her or him to me!
- We also have open positions for student assistants (HiWis) to assist us in software development. Please refer to the website of our chair!

Teaching experience

Universität Trier, DE

- Rechnerarchitektur — 2007 – 2009
- Schaltnetze und Schaltwerke — 2008
- Rechnernetze — 2011

ETH Zürich, CH

- Complex Networks — 2014 – 2018

Karlsruher Institut für Technologie (KIT), DE

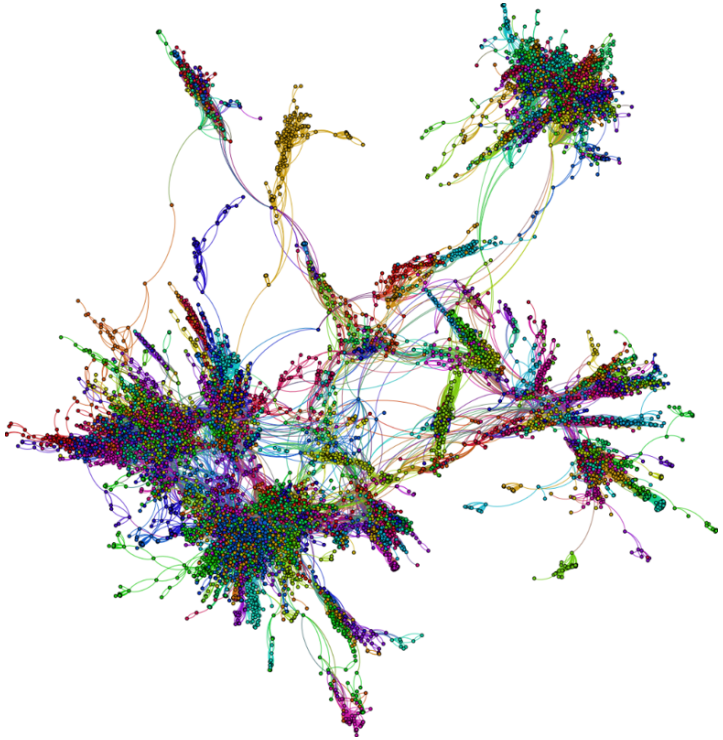
- Informationssysteme für eCommerce — SoSe 2016

Universität Zürich, CH

- Statistical Network Analysis — HS 2018
- Big Data, AI, and Society — FS 2019

BUW, DE

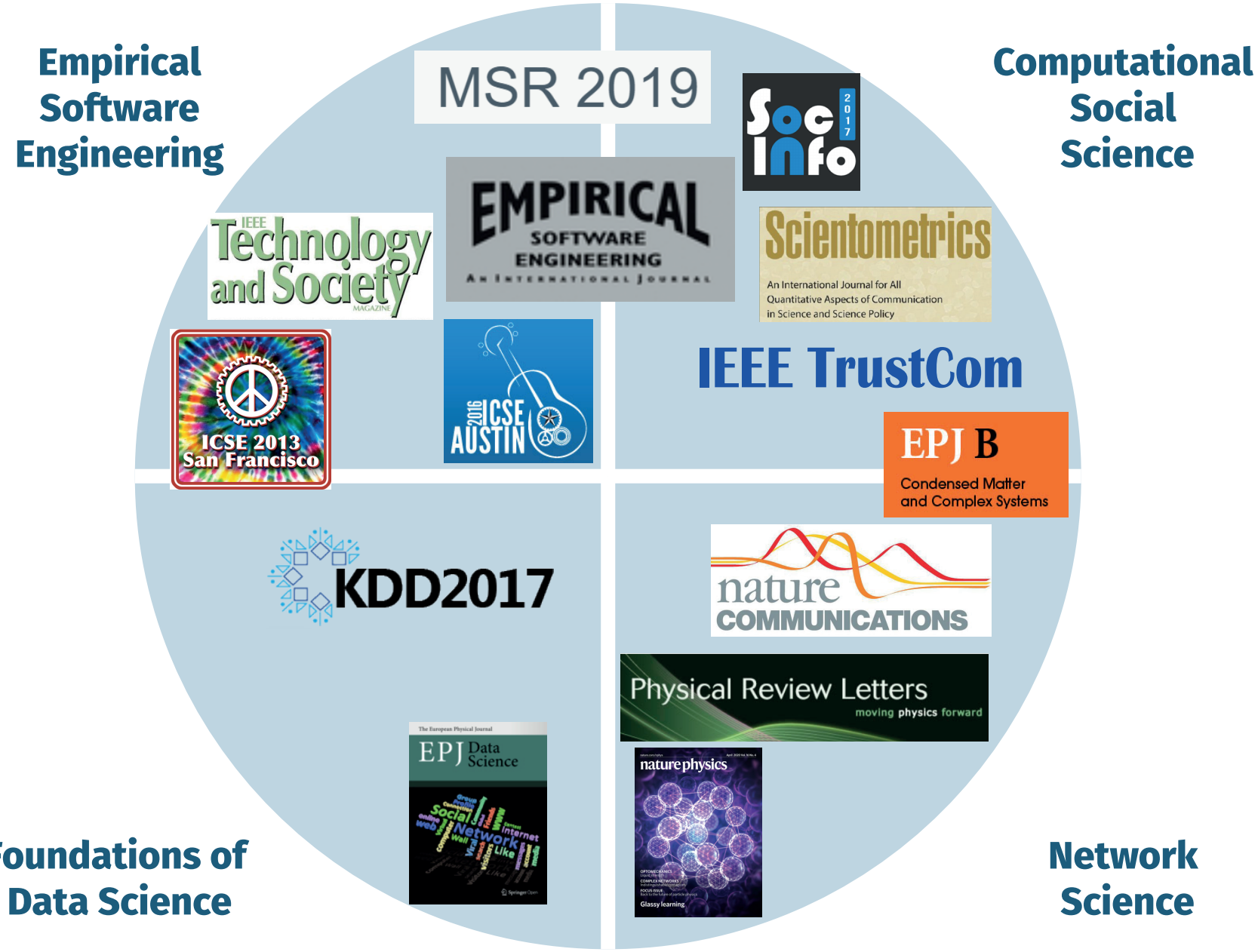
- Einführung in Datenbanken — WiSe 2020/2021
- Introduction to Data Science — WiSe 2020/2021
- Statistical Network Analysis — SoSe 2021
- Big Data, AI, and Society — SoSe 2021



Notes:

- The table above gives an overview of the courses that I have taught at universities in Germany and Switzerland over the past 14 years. I have taught courses in technical computer science (e.g. processor design), networks and distributed systems, information systems, and databases but my main teaching interests are in data science, network science, and computational social science.

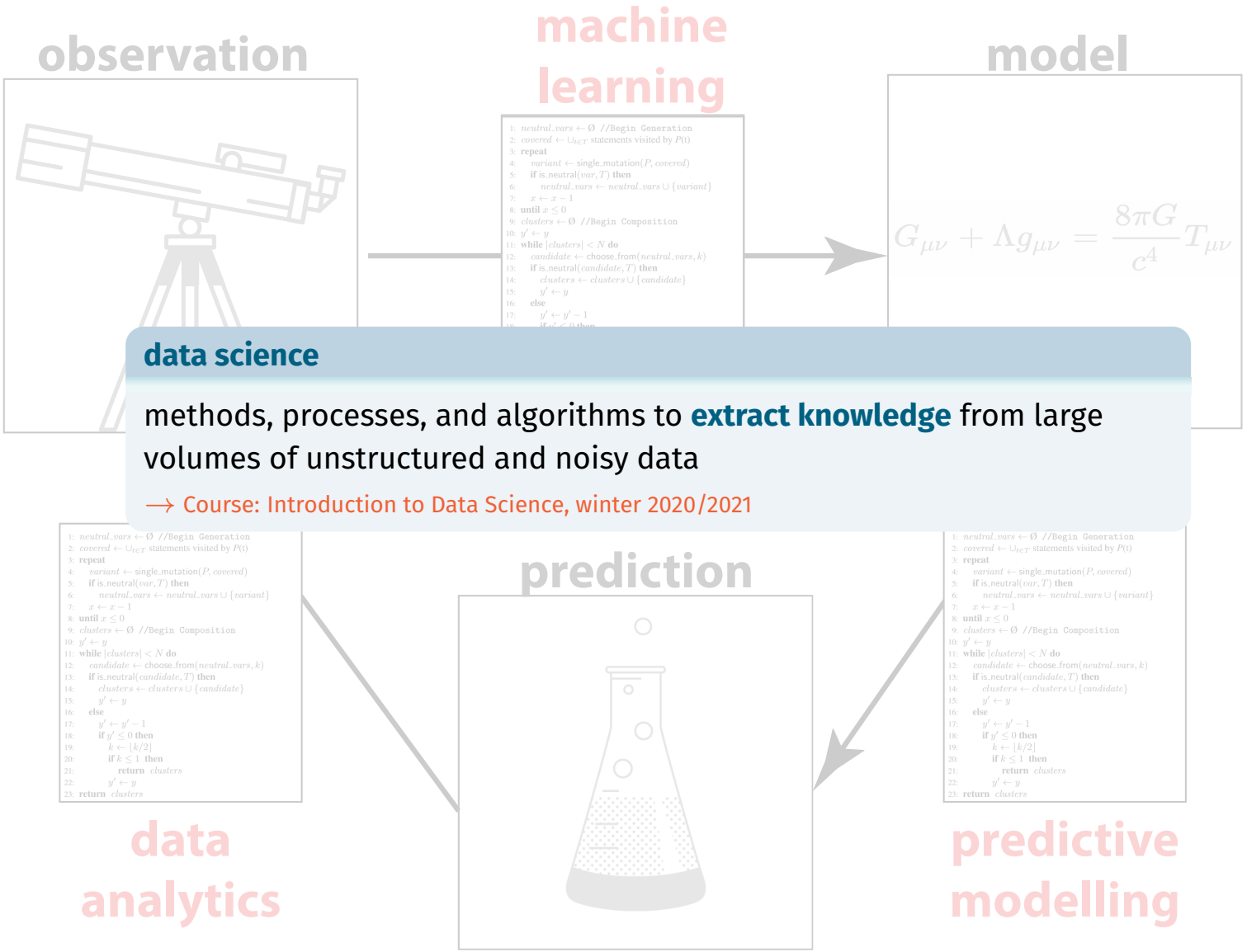
Research profile



Notes:

- The research profile of our Chair again includes several topics and disciplines. Above this is illustrated by some exemplary works and venues at which we have published over the past few years.
- Our work interest are the theoretical and methodological foundations of data science and network science. We are interested in open issues that arise in large, high-dimensional, noisy, and time-resolved data on complex systems and we develop methods that address those issues. We also apply data science, network analysis, and machine learning in practical settings, including but not limited to software engineering, recommender systems, and computational social science, i.e. the study of social systems by means of computational methods.
- In many of our recent applied works we use data science and statistical network analysis to better understand teams, companies, and other social organizations based on large-scale behavioural data. We recently started projects in which we combine network science and machine learning to support decision-making in areas like medical diagnostics, teaching, or traffic management.

What is Data Science?



data science

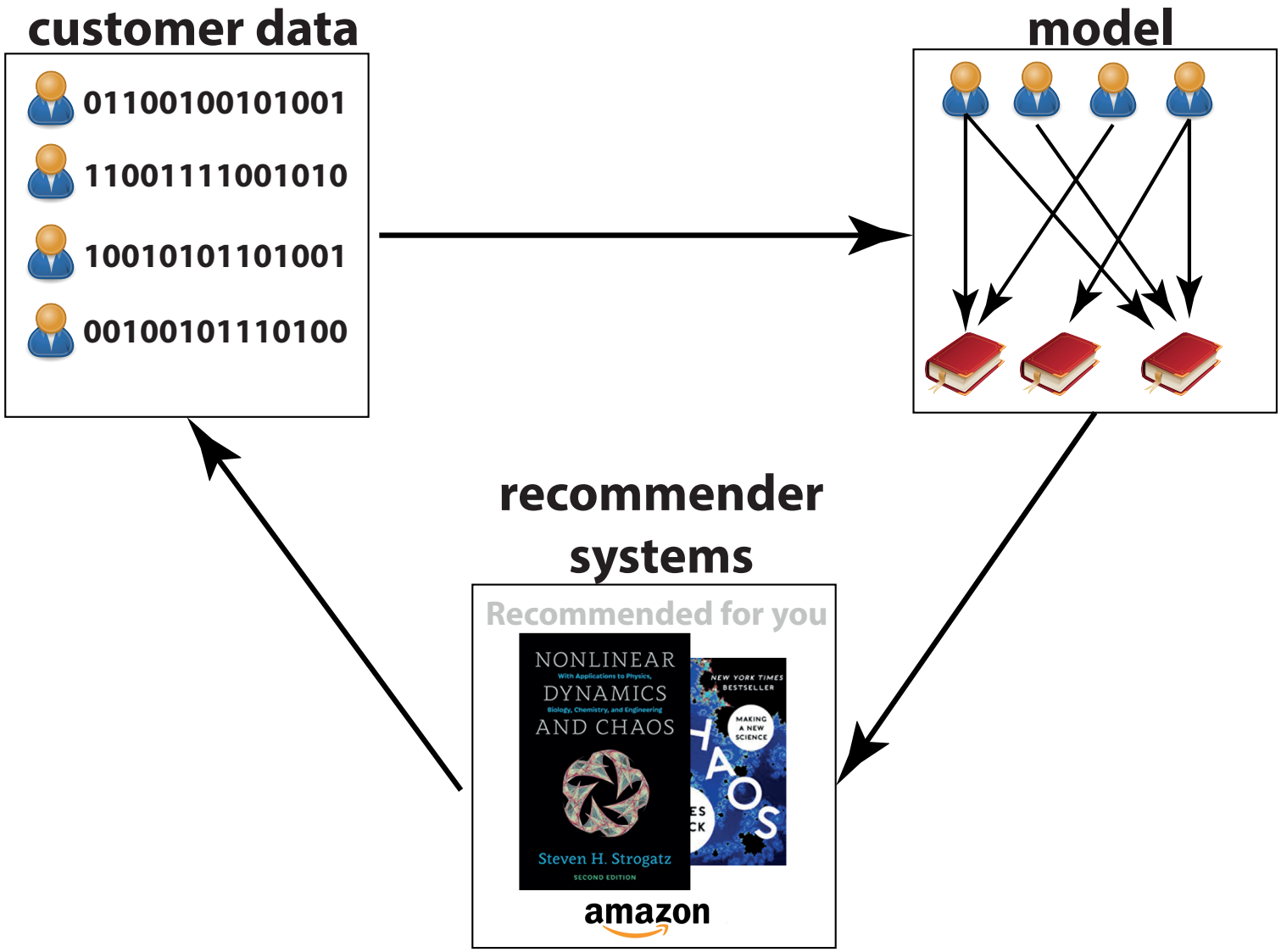
methods, processes, and algorithms to **extract knowledge** from large volumes of unstructured and noisy data

→ Course: Introduction to Data Science, winter 2020/2021

Notes:

- Before we can understand what “network science” is, we first need to understand what “data science” or “science” is. The word science (from Latin “scientia” for “knowledge”) refers to the **systematic process by which we extract knowledge from experience** (cf. Greek “empeiría” for experience). This experience is based on the observation of a phenomenon in our environment. Classical examples are astronomical observations (e.g. of stars, planets, or galaxies).
- We are interested in a model that describes (or explains, from Latin “explano” for “make plain” or “make clear”) our observations. A **good model provides a compact summary of our observations and allows to derive predictions** that can be used to falsify our model in future observations.
- We call this systematic process the “scientific method” and digital technologies have revolutionised it, e.g. by improving our ability to derive predictions through large-scale simulations or test predictions in vast amounts of data. However, the task of deriving a good model that can be used to make predictions is largely left to human intuition and creativity.
- Human cognitive abilities are limited, which is why we are interested in **machine learning** techniques that can be used to further automate knowledge generation. Fitting this view, the term “data science” collectively refers to methods, processes, and algorithms that can be used to extract knowledge from large volumes of structured and unstructured data.

Applications of data science



Notes:

- We illustrate data science in three exemplary applications. When people hear artificial intelligence, many will think of autonomous cars, which follow the same model. The observation is provided by sensors like cameras, radar, lidar, etc. Sensor data are merged into a model of the environment that is used to control the car.
- A second example, to which we are exposed on a daily basis, are social media and other web platforms. Most of them use behavioral data of users to create models that explain (and predict) which content or products customers are likely to appreciate. Those are then turned into recommender systems that either shape our newsfeed, suggest people that we should follow, or highlight products that we may buy.
- A third example, is the use of data science and machine learning in life sciences. We can use large data, e.g. from patients, genetic sequencing, or microarray experiments to learn models that explain clinical observations or highlight interesting patterns. Predictions of such models can then be used to develop new therapies for diseases.
- These three examples have in common that they follow the general process of "data science", a term which highlights the role of data but leaves open the question which type of model is being used. However, many applications of data science in different disciplines also have in common that they use **network models** that capture interactions between different elements or components of a system. The term "network science" emphasises the special role of networks in the scientific process. A remark about the current situation: Network models of interactions between human and viral proteins were recently used to identify drug targets, see

  https://news.northeastern.edu/2020/04/02/network-scientists-identify-40-new-drugs-to-test-against-covid-19/

Network models of complex systems

complex systems

A **complex system** is a system that …

- consists of a **large number of interacting elements**, and
- exhibits **collective or emergent properties** that cannot be easily explained by properties of its elements.

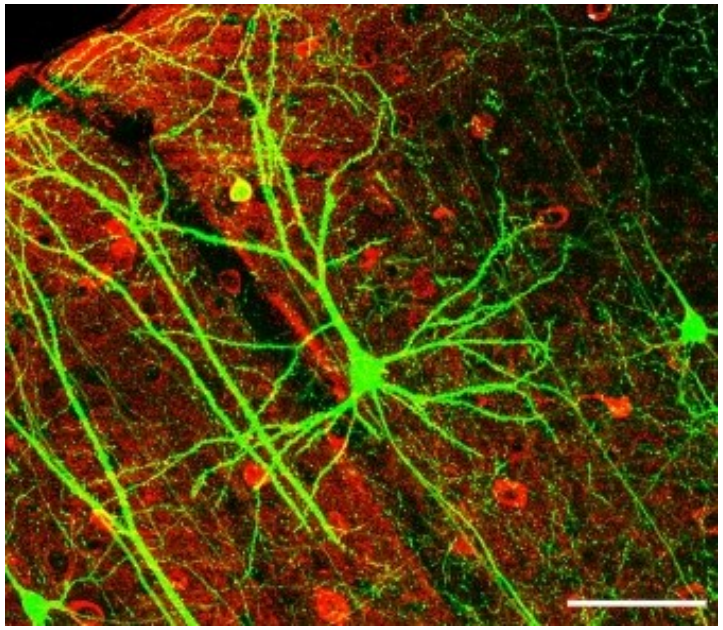
We distinguish between **complex** and **complicated** systems.

complex networks

A **graph** or **network** is a tuple $G = (V, E)$ consisting of a set V of **nodes** and a set $E \subseteq V \times V$ of **edges**.
A network where edges model pair-wise interactions between elements of a complex system is called **complex network**.

network science

methods, processes, and algorithms to **analyse, model, and predict the structure and dynamics of complex networks** based on **relational data**



the **brain** is an example for a **complex** system

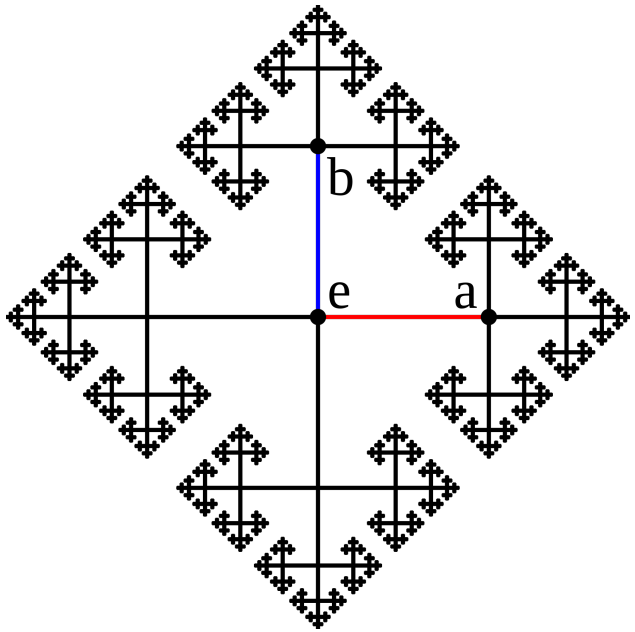
image credit: Pyramidal neurons in a mouse cerebral cortex, by Nrets, Wikimedia Commons, CC BY 2.5

Notes:

- Network models are important because we often study **complex systems** that consist many interacting elements. We **distinguish complex vs. complicated systems**:
  - A computer processor is an example for a **complicated** system. It fulfills the first condition as it consists of billions interacting transistors. But it has been designed such that the behavior of the processor can be understood (and predicted) based on a model of its elements and the way in they are composed. Thanks to clear interfaces between hierarchically arranged system components, we can thus reduce the system behaviour to the behaviour of its components.
  - This is impossible in **complex systems**, which exhibit collective, emergent, or systemic properties that cannot be easily explained based on the properties of components. The brain is an example for a complex system. Even if we understand how individual neurons work and how they are connected, we still do not understand the collective behaviour of billions of interconnected components (creating intelligence, conscience, creativity, etc.).
- Engineered systems can still exhibit emergent properties (e.g. unexpected phenomena in power grids or processors), i.e. they can be complex systems.
- **Complex networks** are graphical models of complex systems, where elements are abstracted as **vertices or nodes** and interactions are represented as **links or edges**. Different communities use different terms. In our course we will use “graph” or “network” synonymously. The use of network models requires **data capturing relations between objects**. Network science is a cornerstone of data science, but network science is more than data science (some methods are not based on data).

First Age: Theoretical Foundations

- since 1730s: **graph theory**
  - Königsberg bridges → L Euler, 1736
  - tree graphs → A Cayley, 1857
  - graph coloring → A Kempe, 1879
- since 1750s: **graphs** and **topology**
  - Euler's polyhedron theorem → L Euler, 1758
  - Cayley graphs → A Cayley, 1878
  - graph planarity → A K Kuratowski, 1930
- since 1950s: **random graphs** and **graph spectra**
  - random graph model → EN Gilbert, 1959
  - spectra of finite graphs → L Collatz, U Sinogowitz, 1957
  - graphs with given degrees → M Molloy, B Reed, 1995
- since 1990s: **complex networks**
  - synchronization dynamics → DJ Watts, S Strogatz, 1998
  - scale-free networks → AL Barabási, R Albert, 1999
  - epidemic spreading → R Pastor-Satorras, A Vespignani, 2001



Cayley graph of free group with generators a and b

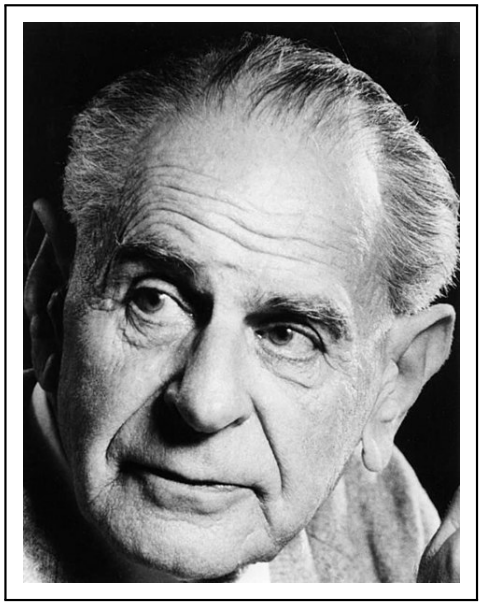
image credit: Wikipedia, public domain

Notes:

- Let us briefly introduce a personal view on the history of network science, which we can coarsely organize in three different historical “ages” (analogous to the three historical ages in Tolkien’s mythology.
- In the first age, the theoretical foundations of graph theory and network science were created. The first works in graph theory can be traced back to the works of the Swiss mathematician Leonhard Euler in the 1730s. Euler used graphs to address the famous Koenigsberg bridges problem, i.e. the question whether there is a walk through the city of Koenigsberg that crosses each of those bridges once and only once.
- In the 1750s, the study of graphs led to the mathematical field of topology and again one of the first works in this area can be traced back to Leonhard Euler. His polyhedron theorem relates the number of vertices, faces, and edges of polyhedra. This theorem can be generalized to planar graphs, which can be thought as models that capture the topology of a polyhedron irrespective of a bending of edges.
- In the 1950s, first works on random graph models combined graphs and probability theory. Subsequent works studied eigenvalue spectra of graphs and random graphs with given degree sequences.
- Finally, the 1990s saw an explosion of interest in random graphs with complex characteristics, such as heterogeneous degree distributions. Moreover, a number of works studied the evolution of dynamical processes like, e.g. synchronization or epidemic spreading, on such networks.

The rise of empiricism

- 18th century: from **rationalism** to **empiricism**
  - rationalism: science = **objective** search for infallible truths → F Bacon, 1620
  - idea that all human knowledge is based on **experience** ($\epsilon\mu\pi\epsilon\iota\rho\iota\alpha$)
- 19th century: **errors & dependencies in data**
  - normal distribution, least square regression → CF Gauss, 1809
  - uncertainty in orbital predictions → P-S Laplace, 1812
  - significance tests → K Pearson, 1900
- 20th century: **critical rationalism**
  - rationalism: science = objective search for **infallible truths** → F Bacon, 1620
  - focus on **falsifiable** theories with **testable predictions**



Sir Karl R. Popper
1902 – 1994

critical rationalism

"[Einstellung] die zugibt, dass ich mich irren kann, dass du recht haben kannst und dass wir zusammen vielleicht der Wahrheit auf die Spur kommen werden."

→ Karl Popper: "Die offene Gesellschaft und Ihre Feinde I", 1944

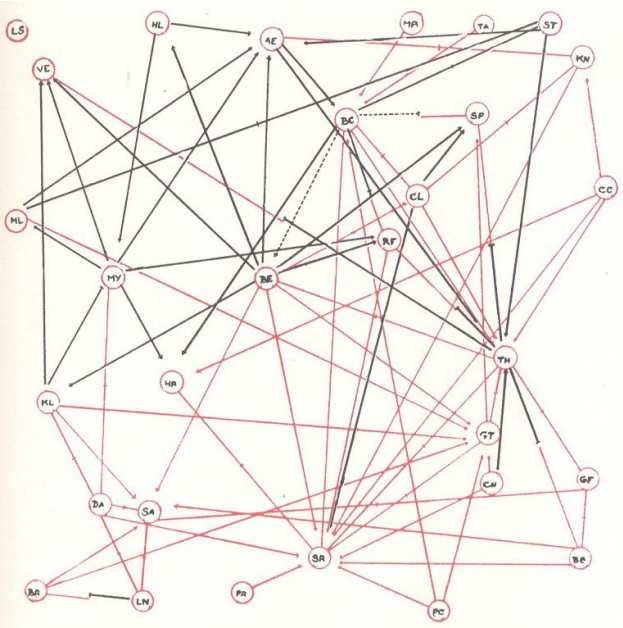
image credit: LSE library, public domain

Notes:

- …

Second Age: Interdisciplinary Network Science

- since 1880s: **chemical and biological systems**
  - **graphs** of chemical molecules → JJ Sylvester, 1878
  - food webs → Charles Elton, 1927
  - neural networks → F Rosenblatt, 1958
  - gene regulatory networks → S Kauffmann, 1969
- since 1930s: **social systems**
  - sociograms → J Moreno, 1934
  - closeness centrality → A Bavelas, 1950
  - clustering → A Rapoport, 1953
  - weak ties → M Granovetter, 1973
- since 1960s: **critical infrastructure**
  - transportation networks → JW Godfrey, 1969
  - Internet topology analysis → Faloutsos[3], 1999
  - economic networks → F Schweitzer et al., 2009



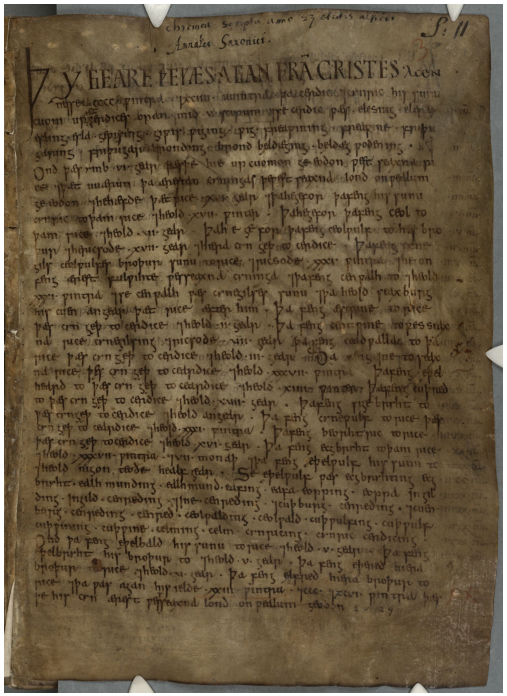
social network of a cottage family

figure from → J Moreno 1934

Notes:

Big Data & Machine Intelligence

- up to 2003: humanity has produced approx. 1 exabyte of data
- 1994: **World Wide Web**
  - Amazon (1994)
  - Google (1998)
  - Facebook (2004)
- 2021: **data everywhere**
  - > 3 billion smartphones
  - 1 exabyte of data produced every 48 hours
- since 1960s: **AI hype** (reloaded)
  - 1960s: first AI hype
  - 1980s: AI winter → J Lighthill, 1973
  - 2000s: deep learning
  - GPUs = “cheap” learning
  - inflated (?) expectations in industry/media



Winchester manuscript from Anglo-Saxon Chronicle ca. 890 AD

exabyte

1 exabyte = 1 billion GB
≈ 1 trillion average images

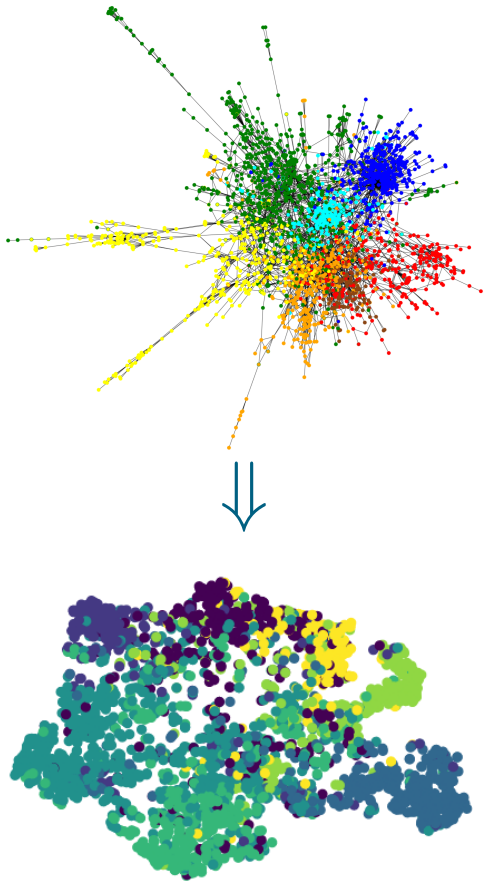
image credit: public domain

Notes:

Third Age: Machine Learning for Networks

- since 1960s: **graph clustering**
  - graph partitioning → BW Kernighan and S Lin, 1970
  - stochastic block model → PW Holland et al., 1983
  - modularity optimization → M Newman, 2004
  - flow compression → M Rosvall and C Bergstrom, 2007
- since 1990s: **recommender systems**
  - PageRank → L Page, S Brin, 1998
  - Matrix Factorization → D Billsus and MJ Pazzani, 1998
  - SimRank → G Jeh and J Widom, 2002
  - Link Prediction → L Getoor et al., 2002
- since 2010s: **graph neural networks**
  - DeepWalk → B Perozzi et al., 2014
  - LINE → J Tang et al, 2015
  - GCN → T Kipf and M Welling, 2017
  - GraphSage → WL Hamilton et al., 2017

citation network



vector space representation
generated by graph convolutional network

Notes:

Example: Triaging bug reports

- **bug tracking** in Open Source Software (OSS) communities is a challenging task

example: Mozilla Firefox

out of 113,000 bug reports submitted between April 2002 and June 2012, 79 % were categorised as **not reproducible**, **duplicate** or **incomplete** (>89,000 reports)

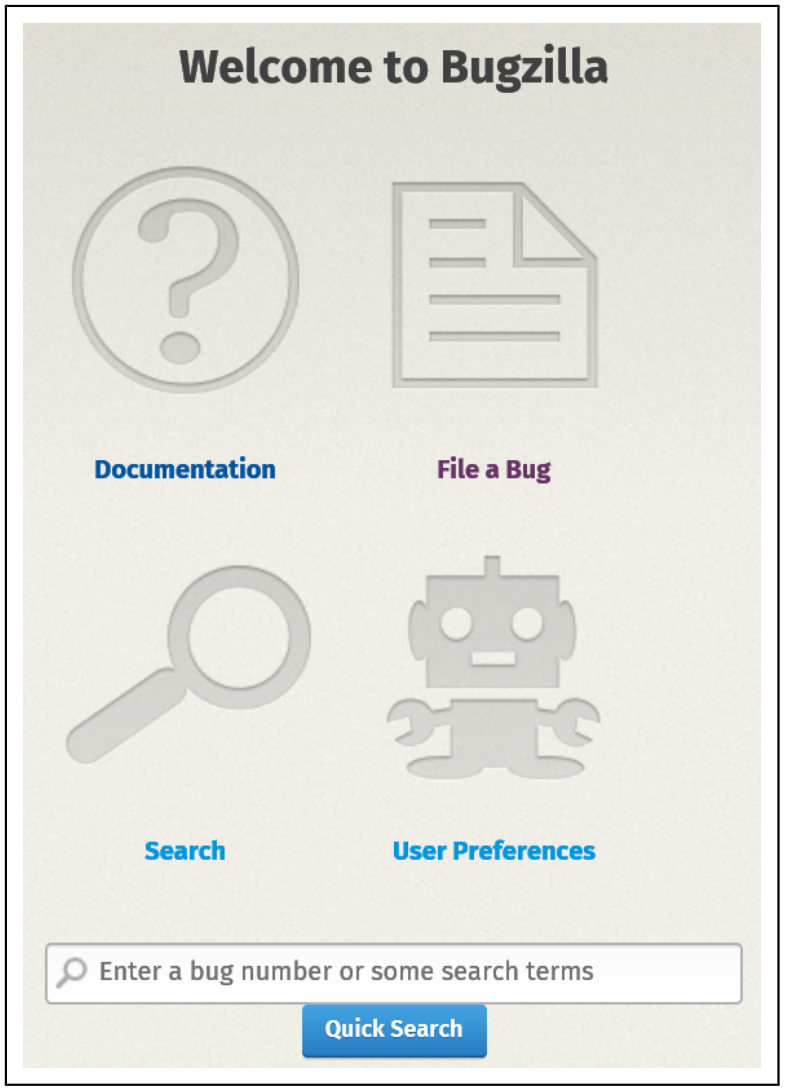
→ M Zanetti, I Scholtes, C Tessone, F Schweitzer, ICSE 2013

research questions

- what distinguishes **invalid** from **valid** bug reports?
- can we **predict valid bug reports** at the time of submission?

data set

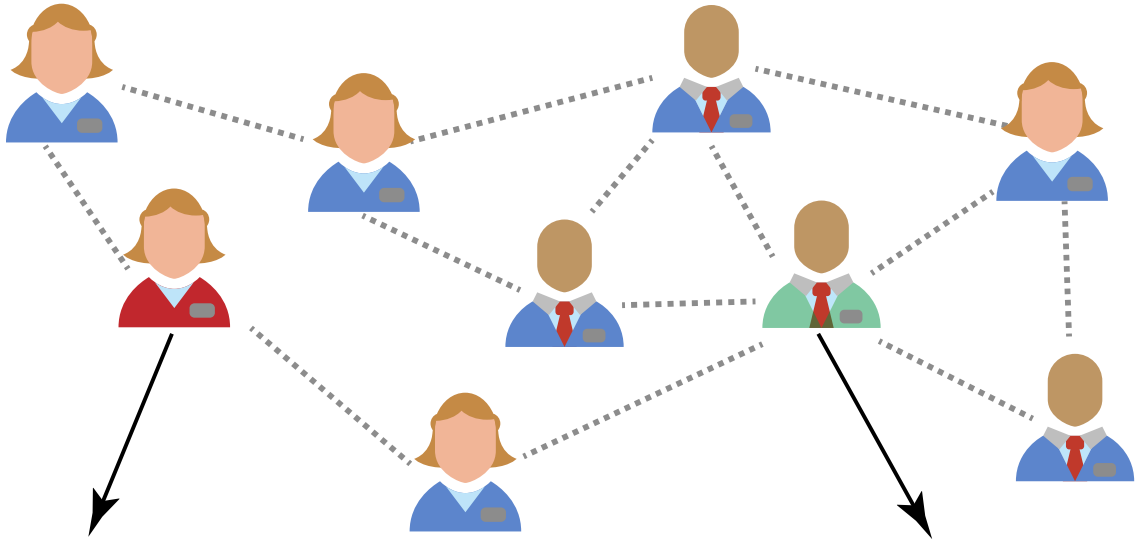
> 700,000 bug reports and 5.8 million user records from OSS projects Firefox, Thunderbird, Eclipse, and NetBeans

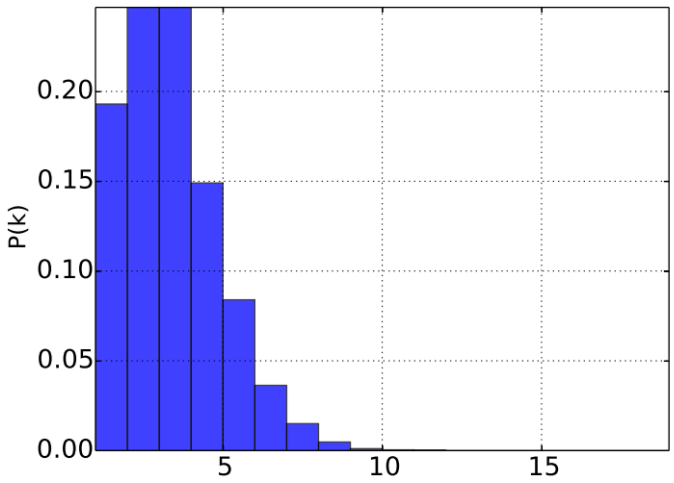


Notes:

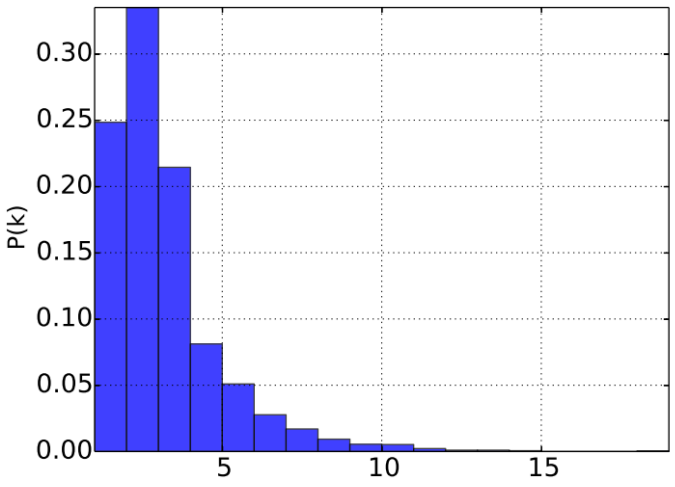
- Let us consider one concrete example, which demonstrates how we can combine network analysis with machine learning. The example is based on a work that we published at a major software engineering conference back in 2013. It addresses the challenge of triaging bug reports in the bug trackers of Open Source Software (OSS) communities. The term “triage” (from French “trier” for sorting or selecting) refers to the process of determining the order in which patients are treated in an emergency situation (you have probably heard the term in recent news).
- Bug or issue tracking platforms, like *Bugzilla*, are widely used to (i) allow users to report software defects and/or missing functionality, and (ii) to coordinate and track the work of developers as they work towards a solution of these defects. A common problem in OSS Communities, is that development teams are overloaded by the vast number of reports. Even worse, our analysis of bug trackers in major OSS communities revealed that the majority of those bug reports are *invalid*, i.e. they either (i) do not refer to actual bugs, (ii) are duplicates of already known issues, or (iii) do not contain enough information to be reproducible.
- Considering the limited resources of OSS projects, we need tools/recommender systems to help filter bug reports. In our study we have combined network science and machine learning to address this problem.

Network analysis and machine learning





eigenvector centralities of users contributing INVALID bug reports

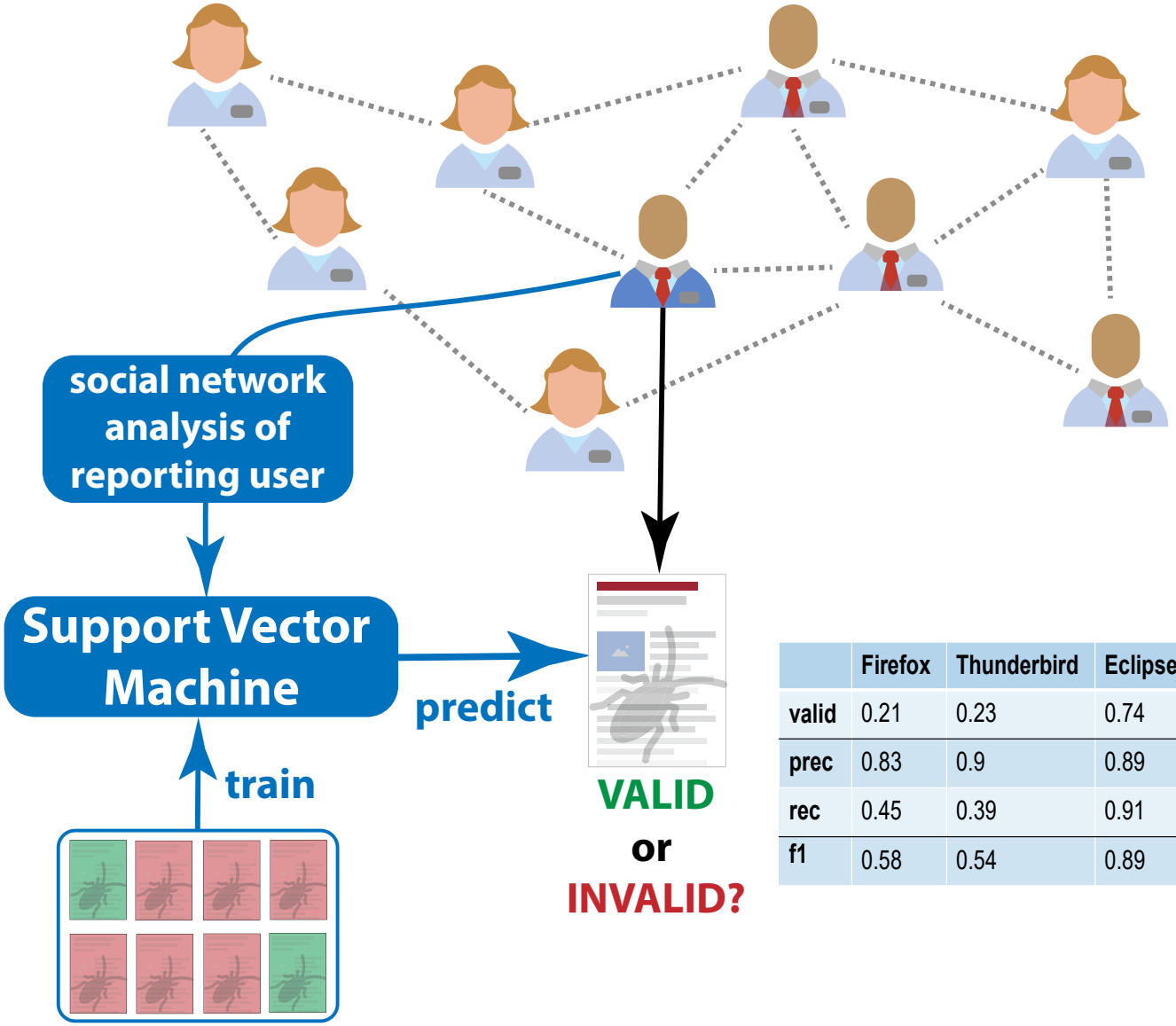


eigenvector centralities of users contributing VALID bug reports

Notes:

- Existing works on the automated triaging of bug reports have considered the textual content of reports. In our study, we have taken a different approach, i.e., we only consider the social context of those users that submit bug reports, i.e. we analyse their position in the **collaboration and communication network of a community**, which we can can construct based on relational data.
- We construct a network model of the community based on dyadic interactions between users, in particular, users forwarding information on bug reports to each other or users assigning tasks to each other. We can now identify users that contribute valid or invalid bug reports, and use social network analysis to assess the position of corresponding nodes in the network model. Here, we specifically used a centrality measure, which yields a value that captures how central the position of a node is in the network. → more on this in L03
- We then perform a statistical analysis of all 700,000 bug reports. A statistical hypothesis test reveals that users reporting valid bug reports are significantly more central in the network than those reporting invalid bug reports.

Network analysis and machine learning



	Firefox	Thunderbird	Eclipse	NetBeans
valid	0.21	0.23	0.74	0.63
prec	0.83	0.9	0.89	0.79
rec	0.45	0.39	0.91	0.87
f1	0.58	0.54	0.89	0.83

Notes:

- The finding that there is a statistically significant relationship between the position of users in the network and the question whether they submit valid or invalid bug reports is the basis for a combination with a machine learning classifier. The idea is to compute multiple network measures and use those measures as positions in a feature space. The outcome of bug reports along with the features of reporting users is used to train a support vector machine (see course Introduction to Data Science). Importantly, we exclusively compute network measures based on interactions that have happened at the time of the submission of the bug report (i.e. we exclusively use information available at the time of submission).
- We can validate the trained classifier in a cross-validation experiment and use precision/recall and f-score to evaluate the predictive power of our classifier.

Statistical network analysis

microscopic perspective

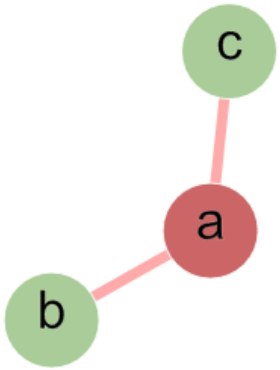
focus: **properties of individual nodes and links**

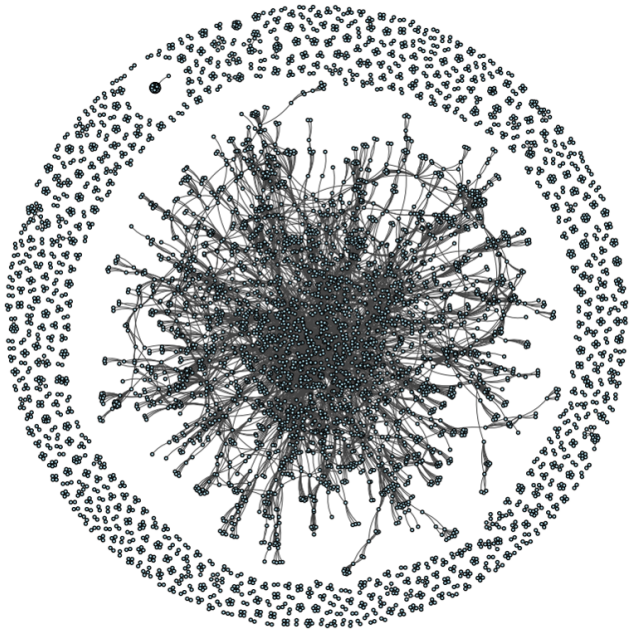
- what is the position of node a?
- why are nodes b and c not connected?
- basis for social network analysis, recommender systems, and feature extraction

macroscopic perspective

focus: **statistical regularities & collective properties**

- statistical models of network topologies
- (stochastic) models for dynamical processes
- basis for pattern recognition, statistical learning, and predictive analytics





challenge: micro-macro link

- explain **systemic properties** based on model of nodes/links
- basis to **design and control complex networks**

Notes:

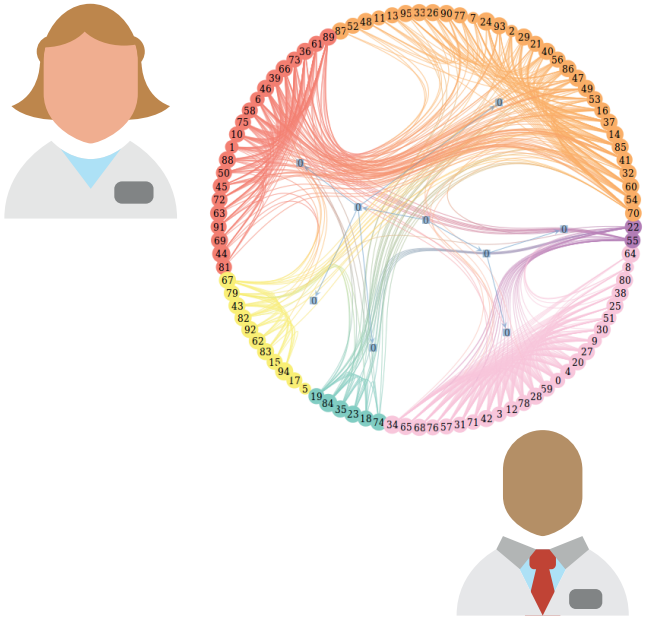
- This example of combining network science and machine learning is based on one perspective taken by network scientists. This microscopic perspective emphasises the properties of individual nodes and links. We can, for instance, use social network analysis techniques like centrality measures to assess the position of a particular node a in a network. In the small example on the left we find that nodes b and c are similar in the sense that they are both not located in the center and both are connected to central node a. Using social science theories about the role of links, we could ask **why** b and c are not connected. Even more, we could use the similarity of b and c to predict that they **should** be connected, which yields a simple link prediction algorithm often used in recommender systems. Such a microscopic analysis is useful in small or moderately large networks, or if we are interested in a particular node or link.
- We are often interested in the system as a whole rather than in a particular element. Consider the collaboration network shown on the right. Each node is one author in the software engineering community, and a link exists between authors that have jointly authored a paper. Here, we could study statistical regularities and systemic properties of the topology that are not due to a particular node or link. For instance, we could study the average distance between authors, the distribution of coauthors, how many parts of the community are disconnected, or search for patterns in the network that are due to different topics. The underlying macroscopic perspective is the basis for statistical network analysis, inference and machine learning applications in networks.

Challenge I: unreliable data

- real data contain **noise, errors, and biases**
  - missing/spurious interactions
  - ambiguous node identifiers
  - small and/or non-uniform samples

network data $\neq$ network model

- need **inference and learning techniques** to obtain reliable results/predictions
  - null models for complex networks
  - efficient model selection/hypothesis testing
  - Bayesian model inference



data: promimity sensing data from RealityMining project at MIT

Notes:

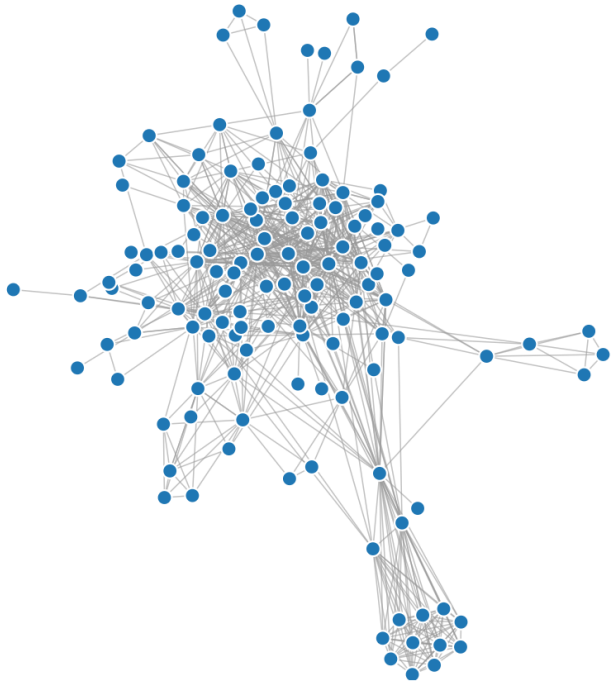
- Advances in network science have led to many important applications. However, a number of challenges still remain. Closing this introduction of network science, I want to present three selected three challenges that shape cutting-edge research in network science. A first challenge is due to the quality of network data. In the early days of network science, data, e.g., on protein-protein interactions, communication networks, or social ties between humans was seen as virtually equivalent to the grpah or network used to model those systems.
- However, in reality data is never a perfect representation of reality. Data contains noise, errors, or biases that must be accounted for in the modelling process. As an example consider the recent application of contact tracing apps, e.g. CoronaWarn app. The example on the right shows a data set that has been obtained via such a smartphone app. In the social sciences, we can use such data to construct a matrix, where each cell in the matrix counts the number of observed proximities between a pair of students (here: MIT students).
- But is this enough to make statements about the underlying social network that drives these proximities? Some of the observed proximities will be incidental, e.g. students randomly passing by each other on their way to the mensa. Other unobserved proximities may be due to sensor errors or missing data. And finally, data are subject to sampling biases, where the most active users are represent more strongly than less active users. To find meaningful pattern in networks, we must account for the unreliable nature of data, using appropriately calibrated null models, Bayesian inference, model selection and hypothesis testing techniques.

Challenge II: time series data

- we increasingly have access to **time-resolved data on complex networks**
  - time-stamped interactions
  - time series data on processes in networks
  - event sequences

> time dimension invalidates results of network analysis

- need **time-aware network analysis**
  - detect **temporal patterns** in node/link activities
  - model **causal paths** in temporal networks
  - understand **interplay of time and topology** in temporal networks



static network projection of (dynamic) character co-occurrences in “The Lord of the Rings”

Notes:

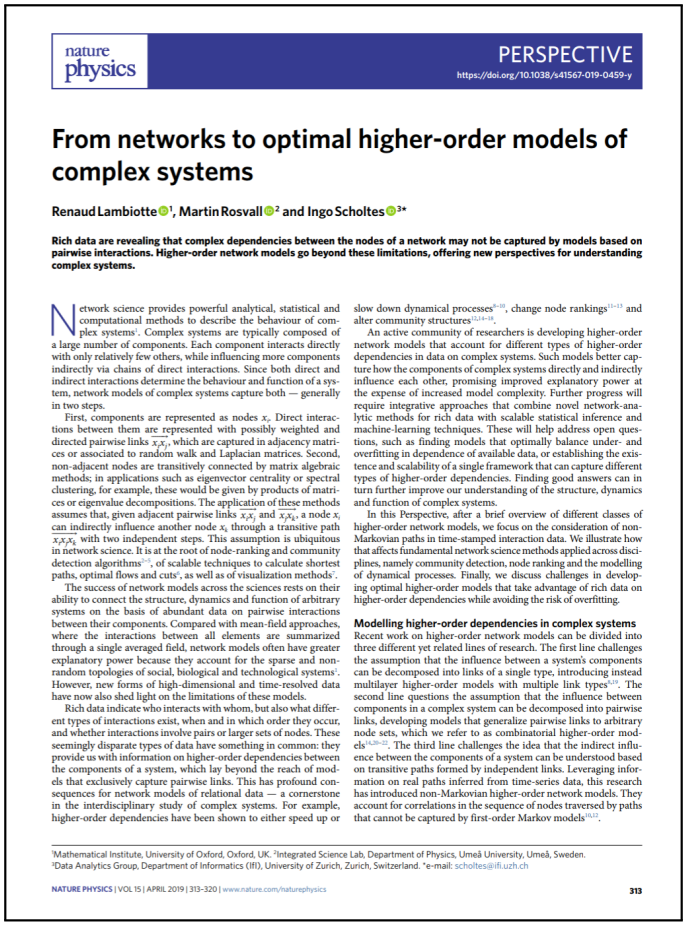
- A second challenge is due to the growing amount of time series data on complex systems, i.e. data that not only capture the topology of interactions between elements, but also when and in which order or sequence those interactions occur.
- The proximity data from the previous slide is an example for a time series data set that naturally gives rise to a temporal network, i.e. a network where links additionally carry time stamps. Such temporal networks cannot directly be analysed with graph-theoretic concepts, since graphs are inherently static objects.
- To address this issue, the network science community is actively developing time-aware methods to analyse time series data on networked systems. This includes methods to detect correlations and patterns in the activation sequence of links or nodes, modelling techniques that study so-called causal paths in temporal networks, i.e. paths by which nodes can influence each other over time (more on this in theory lecture L12). In general we need methods that improve our understanding for the complex interplay between time and topology in complex systems, which has largely been ignored in previous studies.

Challenge III: complex interactions

- in real systems **elements interact in complex ways**
  - non-dyadic interactions
  - multiple types of (dyadic) interactions
  - interactions via specific paths

network models oversimplify interactions in complex systems

- need **higher-order network models**
  - **generalise network models** beyond (single type of) dyadic links
  - topological data analysis
  - hypergraphs, higher-dimensional graphs, multiplex networks, simplicial complexes



nature physics PERSPECTIVE
https://doi.org/10.1038/s41567-019-0459-y

From networks to optimal higher-order models of complex systems

Renaud Lambiotte[1], Martin Rosvall[2] and Ingo Scholtes[3]*

Rich data are revealing that complex dependencies between the nodes of a network may not be captured by models based on pairwise interactions. Higher-order network models go beyond these limitations, offering new perspectives for understanding complex systems.

Network science provides powerful analytical, statistical and computational methods to describe the behaviour of complex systems[1]. Complex systems are typically composed of a large number of components. Each component interacts directly with only relatively few others, while influencing more components indirectly via chains of direct interactions. Since both direct and indirect interactions determine the behaviour and function of a system, network models of complex systems capture both — generally in two steps.

First, components are represented as nodes x_i. Direct interactions between them are represented with possibly weighted and directed pairwise links $\overrightarrow{x_i x_j}$, which are captured in adjacency matrices or associated to random walk and Laplacian matrices. Second, non-adjacent nodes are transitively connected by matrix algebraic methods; in applications such as eigenvector centrality or spectral clustering, for example, these would be given by products of matrices or eigenvalue decompositions. The application of these methods assumes that, given adjacent pairwise links $\overrightarrow{x_i x_j}$ and $\overrightarrow{x_j x_k}$, a node x_i can indirectly influence another node x_k through a transitive path $\overrightarrow{x_i x_j x_k}$ with two independent steps. This assumption is ubiquitous in network science. It is at the root of node-ranking and community detection algorithms[2–5], of scalable techniques to calculate shortest paths, optimal flows and cuts[6], as well as of visualization methods[7].

The success of network models across the sciences rests on their ability to connect the structure, dynamics and function of arbitrary systems on the basis of abundant data on pairwise interactions between their components. Compared with mean-field approaches, where the interactions between all elements are summarized through a single averaged field, network models often have greater explanatory power because they account for the sparse and non-random topologies of social, biological and technological systems[8]. However, new forms of high-dimensional and time-resolved data have now also shed light on the limitations of these models.

Rich data indicate who interacts with whom, but also what different types of interactions exist, when and in which order they occur, and whether interactions involve pairs or larger sets of nodes. These seemingly disparate types of data have something in common: they provide us with information on higher-order dependencies between the components of a system, which lay beyond the reach of models that exclusively capture pairwise links. This has profound consequences for network models of relational data — a cornerstone in the interdisciplinary study of complex systems. For example, higher-order dependencies have been shown to either speed up or slow down dynamical processes[9–10], change node rankings[11–13] and alter community structures[14,15–18].

An active community of researchers is developing higher-order network models that account for different types of higher-order dependencies in data on complex systems. Such models better capture how the components of complex systems directly and indirectly influence each other, promising improved explanatory power at the expense of increased model complexity. Further progress will require integrative approaches that combine novel network-analytic methods for rich data with scalable statistical inference and machine-learning techniques. These will help address open questions, such as finding models that optimally balance under- and overfitting in dependence of available data, or establishing the existence and scalability of a single framework that can capture different types of higher-order dependencies. Finding good answers can in turn further improve our understanding of the structure, dynamics and function of complex systems.

In this Perspective, after a brief overview of different classes of higher-order network models, we focus on the consideration of non-Markovian paths in time-stamped interaction data. We illustrate how that affects fundamental network science methods applied across disciplines, namely community detection, node ranking and the modelling of dynamical processes. Finally, we discuss challenges in developing optimal higher-order models that take advantage of rich data on higher-order dependencies while avoiding the risk of overfitting.

Modelling higher-order dependencies in complex systems

Recent work on higher-order network models can be divided into three different yet related lines of research. The first line challenges the assumption that the influence between a system's components can be decomposed into links of a single type, introducing instead multilayer higher-order models with multiple link types[8,19]. The second line questions the assumption that the influence between components in a complex system can be decomposed into pairwise links, developing models that generalize pairwise links to arbitrary node sets, which we refer to as combinatorial higher-order models[14,20–22]. The third line challenges the idea that the indirect influence between the components of a system can be understood based on transitive paths formed by independent links. Leveraging information on real paths inferred from time-series data, this research has introduced non-Markovian higher-order network models. They account for correlations in the sequence of nodes traversed by paths that cannot be captured by first-order Markov models[23,12].

[1]Mathematical Institute, University of Oxford, Oxford, UK. [2]Integrated Science Lab, Department of Physics, Umeå University, Umeå, Sweden. [3]Data Analytics Group, Department of Informatics (IFI), University of Zurich, Zurich, Switzerland. *e-mail: scholtes@ifi.uzh.ch

→ Lambiotte, Rosvall, Scholtes, Nature Physics, 2019

Notes:

- A third and final challenge results from the observation that the elements in real complex systems interact in more complex ways than what we can capture with (simple) network models. We can, for instance have interactions between more than two elements (e.g. due to a interaction with all elements at a certain distance). In this case we say that the interaction is non-dyadic, i.e. it can involve more than two nodes. Moreover, in many systems we can have more than a single type of (dyadic) links. Indeed, the very first illustration of a social network by Jacob Moreno has two different types of links (friendship and foes), that are depicted by coloured links. As a final example, elements can interact via specific paths, i.e. a set of links that is always activated sequentially in a specific order.
- We can detect such complex interaction types in increasingly rich data on complex systems, but we currently lack techniques to consider them in the modelling process.
- In a recent perspective article we have highlighted this problem in the modelling of complex systems. We further gave a brief overview over different types of **higher-order network modelling techniques**, i.e. approaches to generalise networks such that they can capture more than a single type of dyadic links. Examples include hypergraphs, higher-order networks, multiplex and multi-layer networks or simplicial complexes in topological data analysis.
- These techniques have become a major research focus in the network science, and a lot of interesting methods and applications remain to be discovered.

Who are you?

Questions

- what did you study in your **BSc degree**?
- what is your **MSc program**?
- do you have **experience** in data science or machine learning?
- what are **your interests** in network analysis?
- what are **your expectations** for this course?

Notes:

- This brings us to the last part of today's lecture, in which I want to give you an overview of the course and explain some organisational issues.
- Having introduced me, our chair, and the topic discussed in this course, I would love to hear more about your background, your experience in data science and machine learning, and your interest in complex networks.
- What specific systems that can be studied from a network perspective are you interested in? What are your expectations for this course?

Lectures

Theoretical Concepts

We introduce key concepts and techniques in interdisciplinary statistical network analysis.

→ handout of slides with notes on WueCampus

Practical Demonstrations

We demonstrate how these concepts and techniques can be practically applied in `python`.

→ jupyter notebooks in gitLab

script, lecture handouts and link to git repository available on WueCampus

→ https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2021_wise_sna_notebooks

image credit: Randall Munroe, CC-BY-NC 2.5

Notes:

- This is the second edition of this course at the University of Wuppertal. We will adopt a similar concept like for the course “Introduction to Data Science”, which some of you have attended. We will combine a series of theory lectures (i.e. traditional lectures with slides just like the lecture today) with an accompanying series of practice lectures, in which we interactively show how to do network science in python.
- The handouts of the lecture slides contain basic comments on each slide, which you can complement with your individual notes. The material of the practice lectures are jupyter notebooks, which you can run yourself and which you can use as a seed for your own analyses. The practice lectures will deepen and extend the contents of the theory lectures, and they will allow us to practically explore many of the theoretical concepts and models.
- Due to the fact that this is the second Corona summer semester, this year we have a special situation in the sense that all of lecture recordings of the 2020 edition are already available on Moodle! I would thus suggest that we do not repeat the recording of the theory and practice lectures. Instead, I want to adopt a teaching format in which we use the lecture slots for an interactive session. Which format would you prefer? How should we best use the weekly lecture slots?

I. Motivation and Introduction

L01 What is Network Science? **20.10.2021**

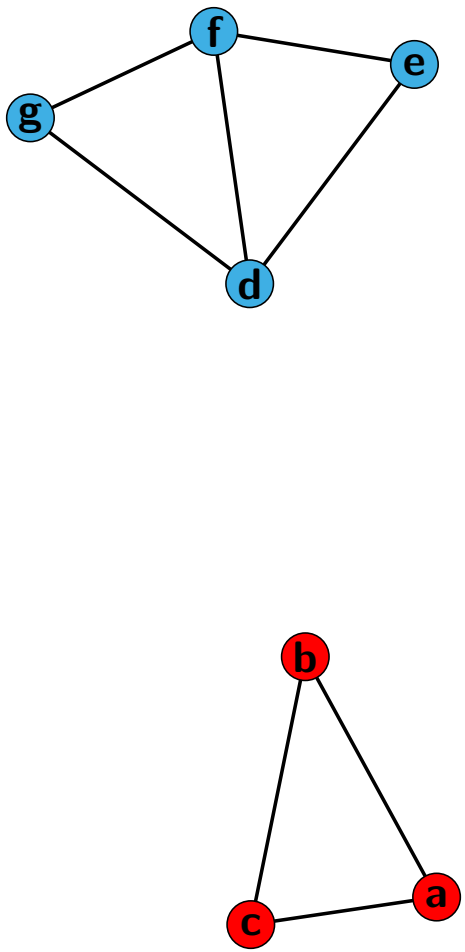
We give an overview of the course and introduce history and interdisciplinary applications of network science.

L02 Foundations of Graph Theory **27.10.2021**

We learn how to mathematically represent networks, how to compute shortest paths and show how to identify maximally connected components.

L03 Modularity and Centralities **03.11.2021**

We show how we can use centrality measures to assess the importance of nodes and how we can detect cluster structures based on the modularity measure.



undirected network with two connected components

Notes:

- Let us study the syllabus of the course. In the first chapter, we motivate and introduce basic concepts in graph theory and statistical network analysis, and show how we can create, analyse, and visualise networks in python.
- Lecture 2 introduces foundations of graph theory, shows how we can mathematically represent networks, and how we can compute shortest paths and connected components.
- In Lecture 3 we show how we can use centrality measures to find important nodes in a network and how we can use the measure of modularity to detect community structures, i.e. groups of nodes that are more connected to each other than to nodes in other groups.

II. Statistical Ensembles of Networks

L04 Random Graph Ensembles **10.11.2021**

We introduce basic statistical ensembles of random graphs and derive expressions for the probability and degree distribution of microstates.

Buss- und Bettag **17.11.2021**

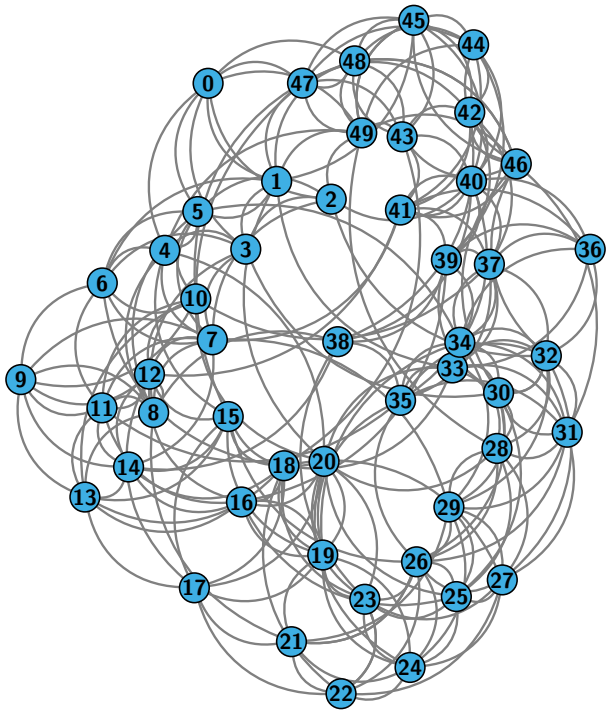
public holiday, no lecture

L05 Small-World Networks **24.11.2021**

We compare the diameter and clustering coefficient of random graphs with real social networks. We introduce the small-world property and explain how it affects the navigability of networks.

L06 Degree-based Ensembles **01.12.2021**

We introduce the configuration model and show how we to make analytical statements about macroscopic properties of a network based on their degree distribution.



network with small-world property

Notes:

- The second chapter introduces statistical ensembles, the key methodological foundation of this course. Various variants of ensembles will keep us busy for several weeks. They will provide the basis to study the expected properties of random networks with given degree distribution, including their connectivity and their robustness.

II. Statistical Ensembles of Networks 2/2

L07 Generating Functions **08.12.2021**

We introduce generating functions and show how we can apply them to analyze the expected properties of random networks with given degree distribution.

L08 Percolation Transition **15.12.2021**

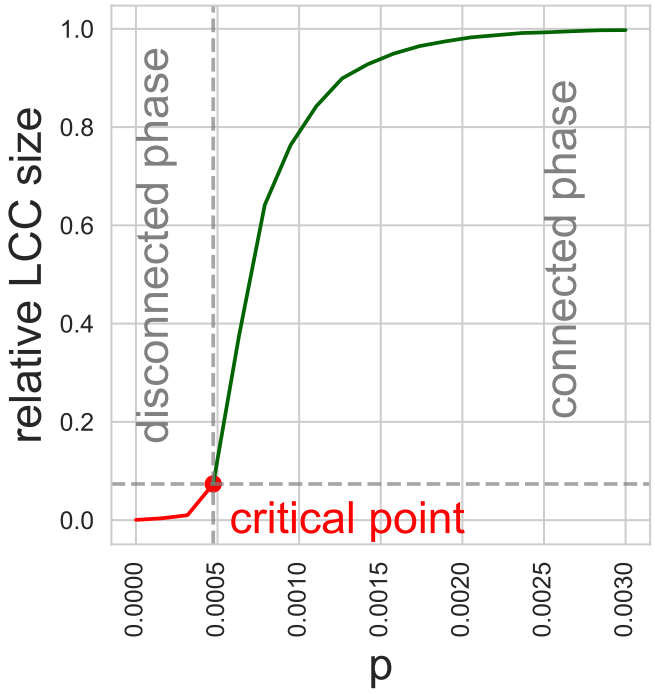
We use generating functions to derive the critical point at which a giant connected component emerges or disappears in random networks with given degree distribution.

L09 Scale-free networks **22.12.2021**

We use generating functions to study properties of scale-free networks. We highlight fallacies in the application of statistical ensembles to real networks.

Christmas break **24.12.2021 – 09.01.2022**

public holiday, no lecture



percolation transition in random graphs

Notes:

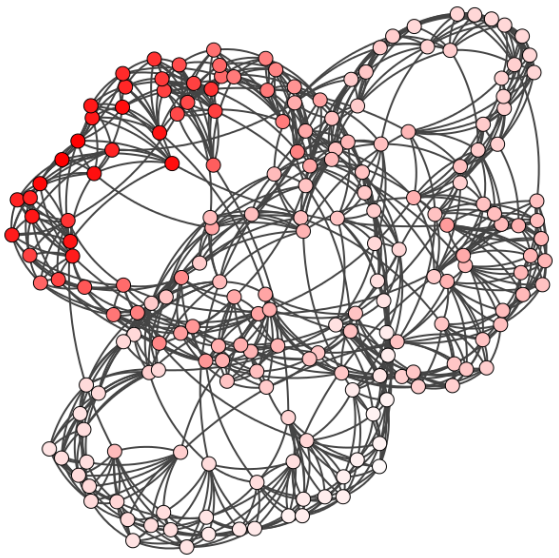
IV. Dynamical Processes in Networks

L10 Random Walks in Networks **12.01.2022**

We explain how we can use Markov chains to model diffusion processes in networks and show how random walks can be used to identify important nodes.

L11 Spectral Analysis **19.01.2022**

We show how we can predict evolution of dynamical processes based on eigenvalues and eigenvectors of matrix representations that capture the topology of networks.



evolution of diffusion process in small-world network

Notes:

- In a third chapter, we study the modelling of dynamical processes in networks. You will learn how we can quantitatively assess how the macroscopic topology of a network influences dynamical processes like random walks and diffusion, and how we can use this knowledge to find interesting patterns in networks.

IV. Dynamics of Networks

L12 Network Evolution **25.01.2022**

We show how we can model equilibrium dynamics of network topologies and introduce a method to predict the stationary macrostate of growing networks.

L13 Temporal Networks and Higher-Order Models **02.02.2022**

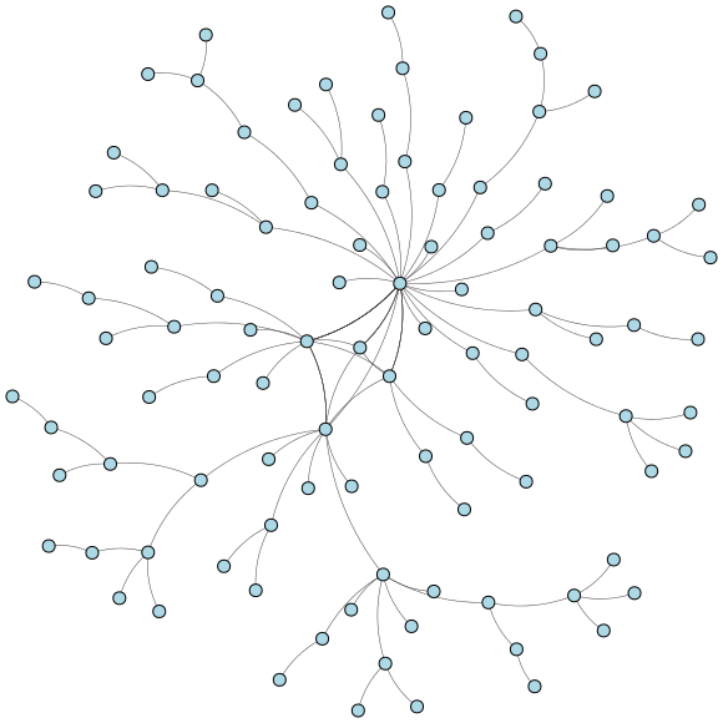
We show that the arrow of time in dynamic networks can invalidate network analysis results and explain how this can be addressed via so-called higher-order network models.

L14 Repetitorium **09.02.2022**

As a preparation for the final exam, you can bring your open questions and ask me to repeat specific parts of the course.

Final exam **16.02.2022 TBC**

preliminary date, please consider announcement of final date on Moodle!



growing network with preferential attachment

Notes:

- In a final chapter, we study the dynamics of networks, i.e. networks in which the nodes or edges are dynamic. We will study how we can model evolving networks with equilibrium and non-equilibrium dynamics.
- We then study how the arrow of time leads to interesting *causal topologies* in temporal networks, and how we can study such causal topologies based on higher-order network models.
- In the final week of the semester, we offer a repetitorium. You can bring any questions that you encountered in the exam preparation and you can ask me to repeat selected chapters of the course.

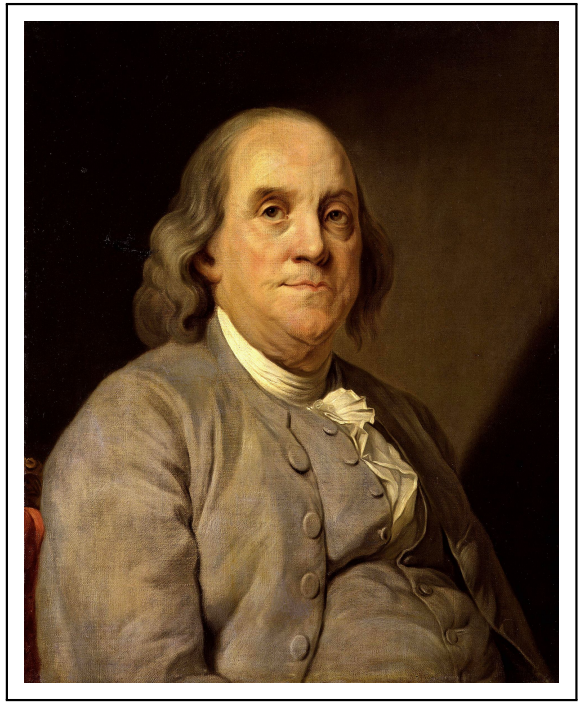
Exercises

weekly exercise sheet

- deepen your knowledge of theoretical concepts
- get a feeling for possible exam questions
- practically apply network analysis in `python`
- gain bonus points for the exam

preliminary exercise schedule

exercise sheet E01	27/10/21 – 03/11/21
exercise sheet E02	03/11/21 – 10/11/21
exercise sheet E03	10/11/21 – 24/11/21
exercise sheet E04	24/11/21 – 01/12/21
exercise sheet E05	01/12/21 – 08/12/21
exercise sheet E06	08/12/21 – 15/12/21
exercise sheet E07	15/12/21 – 22/12/21
exercise sheet E08	22/12/21 – 12/01/22
exercise sheet E09	12/01/22 – 19/01/22
exercise sheet E10	19/01/22 – 26/01/22



Benjamin Franklin

1706 – 1790

quote

"Tell me and I forget, teach me and I may remember, involve me and I learn" → Benjamin Franklin

exercises on Wednesday 12:30 - 14:00

Notes:

- The exercises will consist of weekly assignments that cover both theoretical concepts, proofs, calculations as well as small examples/experiments that can can be solved in `python`.
- The solutions to the assignments must be uploaded to WueCampus before the deadline. Correct solutions will yield bonus points that help you to pass the exam. You also have the chance to present your solutions in the exercise session, in which case you get additional bonus points that can improve your grade (but will not help you to pass the exam).
- The exercise schedule is given above and there will be a total of ten exercise sheets. The first sheet will be released next week and the first exercise session will take place today in two weeks.
- Don't worry if you have not worked with `python` before. In today's exercise session I will introduce some basics of what you will need in the course. I will also make available a short tutorial on WueCampus.

Self-study and exam

final exam

written, closed-book exam of 120 minutes

- no `python` skills required
- exercises = bonus points
- preliminary exam date: **16/02/2021 (TBC)**

exam registration

mandatory registration via WueStudy

- from 01.10.2021 - 31.01.2022 (Zeitraumgruppe C10)
- “Ausgewählte Kapitel der Informatik” (318820)
- “Ausgewählte Kapitel der Intelligenten Systeme” (318914)

assisting self-study at home

each lecture includes 10 - 12 **self-study questions**

- repeat most important issues
- test your comprehension
- useful for exam preparation

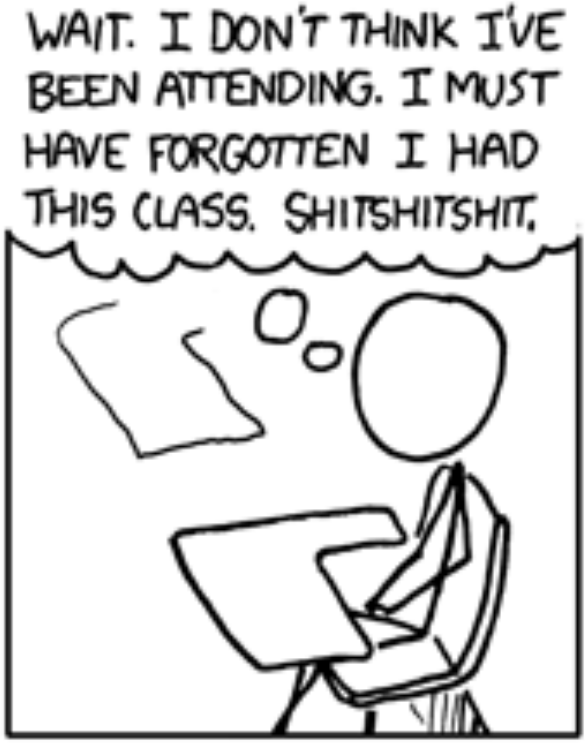
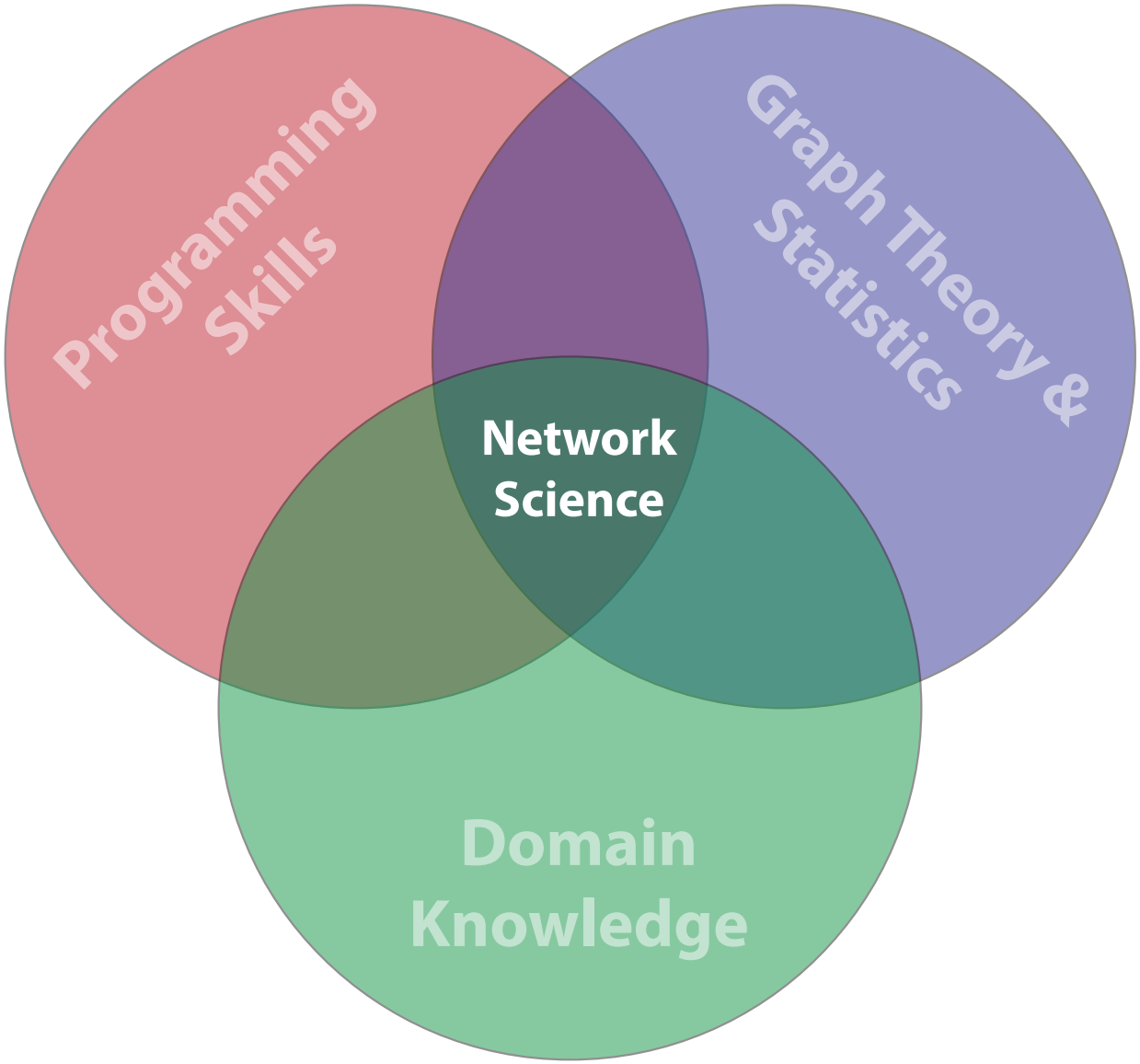


image credit: Randall Munroe, CC-BY-SA

Notes:

- The final exam will be a two hour closed book exam, i.e. there are no books or lecture slides allowed. Wou may be allowed to use a single, hand-written sheet during the exam but I will finally decide about this later. For the exam, you will not have to program, so the emphasis is on a conceptual understanding of the methods rather than on programming skills.
- We further assist you in the self-studies at home, which are an important component of any course and is actually accounted for in the ECTS credits as well. As a reminder, this course gives 5 credit points, which corresponds to a workload of 125 - 150 hours. The 28 lectures and exercise sessions amount to 56 hours, which leaves an additional six hours per week to (i) complete the exercises, (ii) brush up your python and programming skills, and (iii) work through the slides, jupyter notebooks, and literature that we distribute.
- To assist you in the self-study and exam preparation, we include a set of self-study questions at the end of each lecture. You should be able to answer them based on the slides and you can see them as a proxy for the questions asked in the exam (along with the tasks in the exercise sheets).

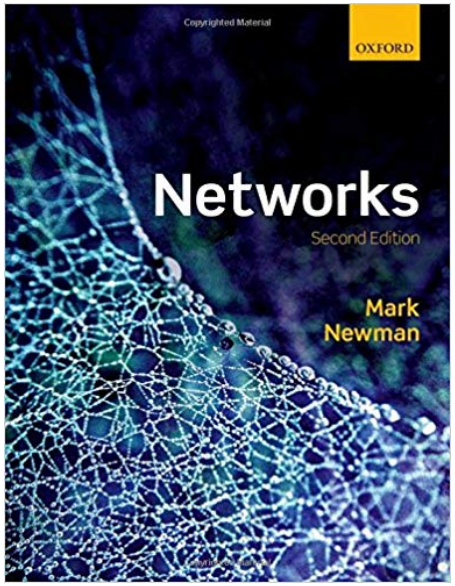
Course concept



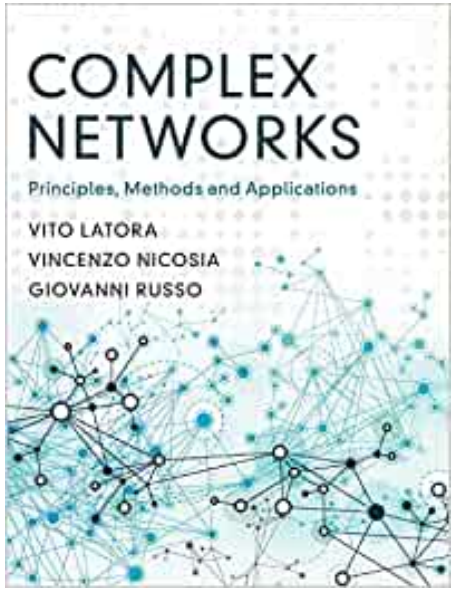
Notes:

- Similar to data science, network science requires theoretical knowledge in mathematics, graph theory, and statistics, practical skills in programming and data analysis, as well as domain expertise.
- In this course, we will not be able to teach you the domain expertise needed to apply network science in a specific application setting, but we will be happy to assist you in finding projects outside this course (e.g. through labs, master thesis topics, or student assistant positions).

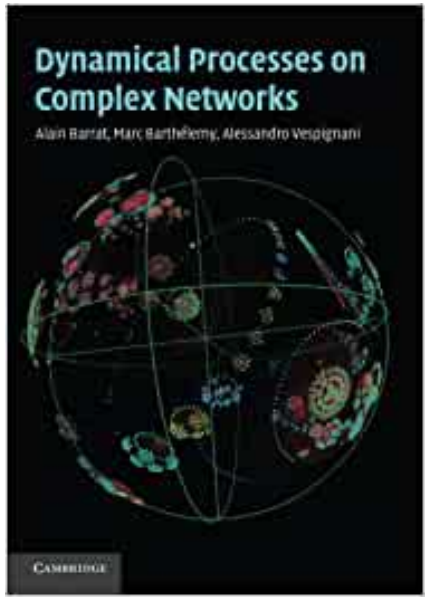
Literature



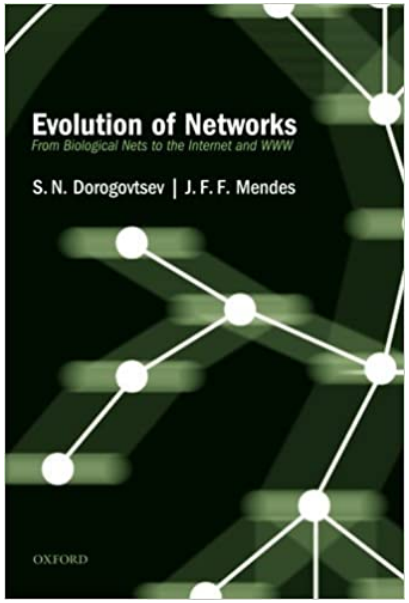
Mark Newman: **Networks**, 2nd edition, Oxford University Press, 2018



Vito Latora, Vincenzo Nicosia, Giovanni Russo: **Complex Networks: Principles, Methods, and Applications**, Cambridge University Press, 2017



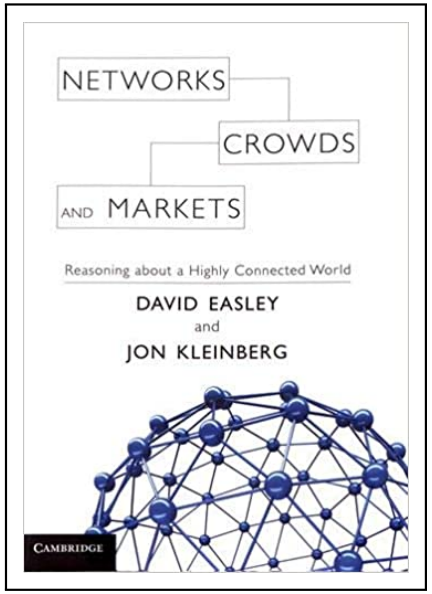
Alain Barrat, Marc Barthélemy, Alessandro Vespignani: **Dynamical Processes on Complex Networks**, Cambridge University Press, 2008



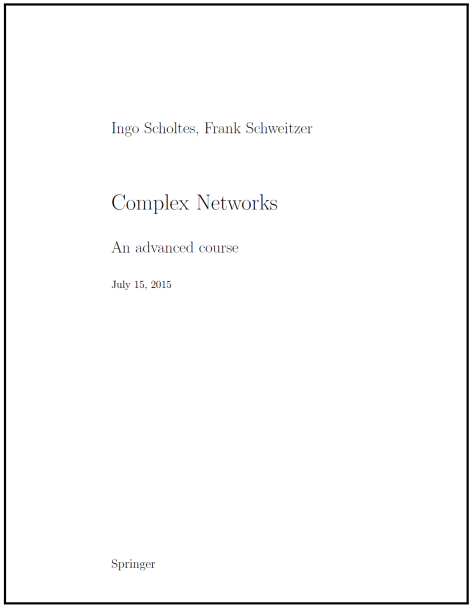
Sergey N Dorogovtsev, Jose FF Mendes: **Evolution of Networks: From Biological Nets to the Internet and WWW**, Oxford University Press, 2003

Notes:

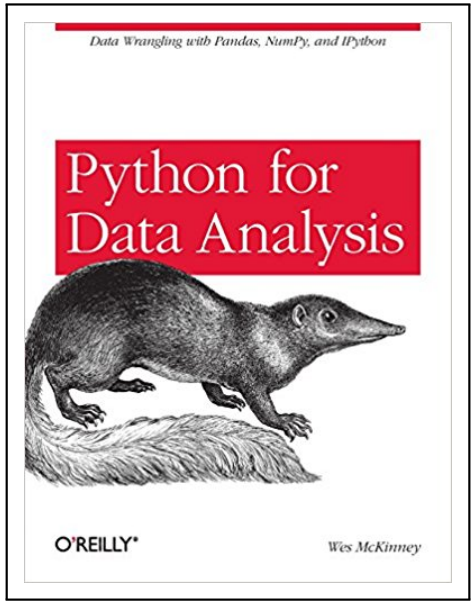
Literature



David Easley, Jon Kleinberg: **Networks, Crowds, and Markets: Reasoning about a highly interconnected world**, Cambridge University Press, 2010



Ingo Scholtes, Frank Schweitzer: **Complex Networks: An Advanced Course**, in preparation, selected chapters on Moodle, 2020



Wes McKinney: **Python for Data Analysis**, 2nd Edition, O'Reilly, 2017

PATHPY

https://www.pathpy.net/

see tutorial and practice lecture notebooks

Notes:

Let's learn from each other!

Humboldtsches Bildungsideal

University education should not be job-focused, but educational training that is independent of economic interests. [...] University should be a place of [...] **exchange between all involved in the scientific process**. → Wikipedia

Wilhelm von Humboldt
1767 – 1835

- we need a **pleasant learning environment**
  - golden rule: there are **no** stupid questions
  - feel free to **ask anything at any time**
- **consultation hours** and **advice**
  - contact me in case of problems/concerns
  - 24/7 via E-Mail :)
  - personal meetings upon request

`ingo.scholtes@uni-wuerzburg.de`

Notes:

- An important rule for this course: you should not be afraid to ask questions, no matter how simple or “stupid” you think they are. There will be no laughing about you and there will be no public shaming. You should be able to learn in an environment free of fear.
- If you have problems that you don’t want to address in the plenum, contact me via E-Mail or ask for an appointment during my speaking hours.
- Please consider posting your questions to the Q&A forum on Moodle rather than sending them via E-Mail. This will allow other students to benefit from the answer to your question, and other students may have an answer even before I managed to reply.

Today's exercise …

- basics of **data & network science** with `python`
  - work with `Visual Studio Code`
  - literate programming with `jupyter`
  - set up your network science environment
  - collaborate via `git`
  - import and visualise networks in `pathpy`
- `pathpy` is a **network and path analysis package** developed at our chair
- we provide **jupyter notebooks**
  - available via gitLab
  - see link in Moodle
- see you **today at 12:30** via Zoom!

P01-02: Introduction to pathpy

April 23 2020

Now that we know the basics of python, let us introduce some basics of pathpy, a network analysis and visualisation package developed in my group.

Compared to many other packages, pathpy has a couple of advantages. First, it is particularly easy to install as it has no dependencies that are not included in a default anaconda installation. Second, pathpy is fully documented, comes with a tutorial and has a user-friendly API. It also provides interactive HTML visualisations that can be directly displayed inside jupyter notebooks, which makes it particularly suitable for educational settings. Third, it supports the analysis and visualisation of time series data on networked systems, such as time-stamped edges or data on paths in networks.

We first need to install pathpy on the terminal. The latest release version pathpy3 is available for download from the python package index pip. You can install it via the command:

```
pip install pathpy3
```

pathpy requires python 3.x. If you have both python 2.x and python 3.x installed, you can explicitly install a package for python 3 by using the command pip3 instead of pip. Note that python 2 is deprecated since April 2020, so you should always use python 3.

Note that you may need to manually uninstall any previous major version of pathpy, e.g. uninstall the package pathpy2 in case you have it installed. You can check this via the command pip list. Uninstalling prior versions is necessary because all pathpy versions use the same namespace pathpy and for reasons of backwards compatibility we still provide older major versions in case someone needs them. If for some reason you want to quickly switch between different major versions, we recommend to use virtual environments in Anaconda.

Rather than using the release version, for this course I would recommend to *install the development version* from github. This is useful, because it is likely that we will add additional functions or fix issues as we progress with the course. You can install the latest development version of pathpy via:

```
pip install git+git://github.com/pathpy/pathpy.git
```

Alternatively, you can clone the repository to your local machine by running the command:

```
git clone https://github.com/pathpy/pathpy
```

This will create a new directory pathpy on your machine. You can change to this directory and run pip install -e ., which will *install pathpy as an editable python module*. Technically speaking, rather than copying the code to a separate directory in your python module path, this will create a symbolic link to the current directory. This has the advantage that you can update your local installation of pathpy by entering the directory and running git pull. After this the new version will be immediately available in all python processes launched after the update. This approach allows us to update your local pathpy installations by means of a simple git push. Note that after updating pathpy you must always restart the python process (or jupyter kernel) before changes are effective.

Now that we have installed pathpy we can import it in python and assign the local alias pp:

```
import pathpy as pp
import numpy as np
```

```
[04-09 17:08:55: DEBUG] pathpy version 3.0.0a1
[04-09 17:08:55: DEBUG] platform is win32
[04-09 17:08:55: DEBUG] pathpy runs in a vs code environment
```

If the import statement completes without error message, the installation was successful and we can now use pathpy to generate, analyse, and visualise networks.

Creating networks

For this purpose pathpy provides the Network class. Calling the constructor will return an instance that represents an empty network with no nodes and no links. By default, networks in pathpy are directed. If you want to create an undirected network you can pass the parameter directed=False in the constructor.

Printing the Network object will give a short string summary which tells whether the network is directed or undirected, as well as the number of unique nodes and links.

PATHPY

`https://github.com/pathpy/pathpy`

Notes:

- In today's exercise session, I will introduce the basic data science and network analysis ecosystem in python, which will be the foundation for the practical demonstrations and some of the exercises.

Self-study questions

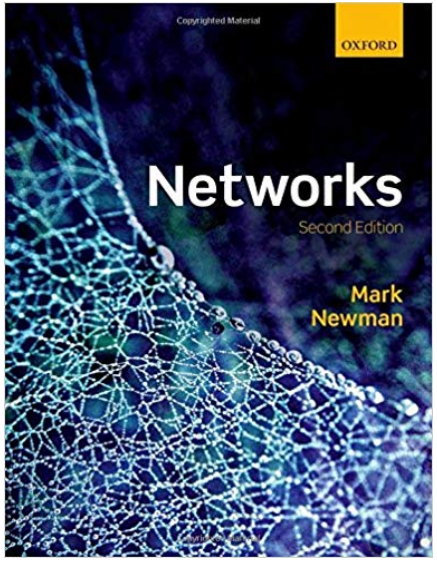
1. Explain the role of graphs and network models in data science.
2. Give two examples for complex systems that can be studied from a network perspective?
3. Explain the micro and macro perspective on networked analysis. What distinguishes them and what are exemplary applications.
4. Search the network science literature and identify an exemplary study that applies concepts from statistical physics in the study of complex networks.
5. Investigate how the modelling of epidemics in networks is used to fight pandemics.
6. List and explain three key challenges in the study of complex systems from a network perspective.
7. What is a hypergraph and how is it related to a graph?
8. Give an examples for a system that can naturally be modelled by an undirected or directed network respectively.
9. Give an example how network analysis can be combined with statistical classification techniques.

Notes:

References

reading list

- M Newman: **Networks**, Oxford University Press, 2010
  $\longrightarrow$ Chapter 1
- D Easley, J Kleinberg: **Networks, Crowds, and Markets: Reasoning about a highly interconnected world**, Cambridge University Press, 2010 $\longrightarrow$ Chapter 1
- V Latora, V Nicosia, G Russo: **Complex Networks: Principles, Methods, and Applications**, Cambridge University Press, 2017 $\longrightarrow$ Chapter 1
- S Boccaletti, V Latora, Y Moreno, M Chavez, DU Hwang: **Complex networks: Structure and dynamics**, Physics Reports Vol. 424, No. 4–5, pp. 175-308, 2006
- MS Zanetti, I Scholtes, C Tessone, F Schweitzer: **Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities**, In Proceedings of ICSE'13, pp. 1032-1041, 2013
- R Lambiotte, M Rosvall, I Scholtes: **From networks to optimal higher-order models of complex systems**, Nature Physics, 2019



Categorizing Bugs with Social Networks: A Case Study on Four Open Source Software Communities

Marcelo Serrano Zanetti, Ingo Scholtes, Claudio Juan Tessone, and Frank Schweitzer
Chair of Systems Design – http://www.sg.ethz.ch/ – ETH Zurich, Switzerland

Abstract—Efficient bug triaging procedures are an important precondition for successful collaborative software engineering projects. Triaging bugs can become a laborious task particularly in open source software (OSS) projects with a large base of comparably inexperienced part-time contributors. In this paper, we propose an efficient and practical method to identify valid bug reports which a) refer to an actual software bug, b) are not duplicates and c) contain enough information to be processed right away. Our classification is based on sim measures for the social embeddedness of bug reporters in the project's collaboration network. We demonstrate its applicability in a case study, using a comprehensive data set of more than 700,000 bug reports obtained from the BUGZILLA installation of four major OSS communities, for a period of more than ten years. For those projects that exhibit the lowest fraction of valid bug reports, we find that the bug reporters' position in the collaboration network is a strong indicator for the quality of bug reports. Based on this finding, we develop an automated classification scheme that can easily be integrated into bug tracking platforms and analyze its performance in the considered OSS communities. A support vector machine (SVM) to identify valid bug reports based on the sim measures yields a precision of up to 90.3% with an recall of up to 90.5%. With this, we show that quantitative measures of social embeddedness are a promising approach for the identification of bugs that are eventually fixed. Furthermore, our study highlights the potential of using quantitative measures of social organization in collaborative software engineering. It also opens a broad perspective for the integration of social awareness in the design of support infrastructures.

I. Introduction

Triaging and processing bug reports is an important task in collaborative software engineering which can crucially affect product quality, project reputation, user motivation and thus the long-term success of a project. Practical experience from large open source software (OSS) projects confirms that – particularly in projects with large numbers of comparably inexperienced part-time contributors– the process of triaging, categorizing and prioritizing bug reports can become a laborious and difficult task that consumes considerable resources. Both the importance and complexity of this problem can be illustrated by a simple example: Out of the more than 64,000 bug reports that have been resolved by the community of the MOZILLA FIREFOX project, more than 50,000 (or ≈ 78%) of these reports have eventually been identified either as *duplicate* of known bugs, *invalid* reports that refer to a user error rather than a software issue or *incomplete* reports which lack basic information required to reproduce the alleged bug. The magnitude of this problem in large-scale projects calls for (semi-)automated techniques that assist bug handling communities in the triaging and prioritization of bug reports. The provision of methods which are able to automatically identify *valid* bug reports with high precision can have huge implications for practitioners of distributed software engineering: Being able to filter, assign and prioritize those bug reports that likely result in a bug fix can significantly improve the responsiveness of support communities. Furthermore, a temporary deferral of those bug reports that are likely to be duplicates, invalid or incomplete to a moderation queue can considerably alleviate the effort required for bug triaging. It can also be used to automatically enforce the adherence to community guidelines, e.g. by automatically asking original reporters to reconfirm that reported bugs are neither duplicates nor incomplete.

Due to the importance for practical software engineering, a number of different approaches for the automated classification of bug reports have been studied, among them approaches based on the automated assessment of information provided by bug reports [1–4], natural language processing [5–7], the temporal dynamics of bug handling processes [8], coordination patterns [9], or the reputation of bug reporters [10–12]. Based on a unique data set containing the full history of more than 704,000 bug reports in four major OSS communities, in this paper we consider to what extent automated bug classification techniques can be based on *quantitative measures for the social embeddedness of bug reporters in the project's community*. We particularly address this question from the perspective of complex, evolving collaboration networks and the computation of node-centric metrics that capture structural properties like centrality and clustering.

Our contributions to the current state of research are:

- We study relations between the centrality of bug reporters and the eventual outcome of the bug triaging process. For the four OSS communities studied in this paper, we find strong evidence for the hypothesis that the centrality of users in the collaboration network is indicative for the quality of bug reports.
- We show that quantitative measures for the bug reporter's position in the collaboration network can be used for an automated classification of valid bug reports. For the four studied OSS communities, we find that this classification achieves a precision of up to 90.3% with an associated recall of 38.0%.

With this, we extend previous works that have studied automated classification of bugs that are eventually fixed. In particular, we use a more comprehensive data set, more

978-1-4673-3076-3/13/$31.00 @ 2013 IEEE 1032 ICSE 2013, San Francisco, CA, USA Software Engineering in Practice

Notes: