From Aristotle to Ringelmann

Using Big Data and Network Science to Understand Productivity in Software Teams

Prof. Dr. Ingo Scholtes

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

&

SNF-Professor for Data Analytics
University of Zurich

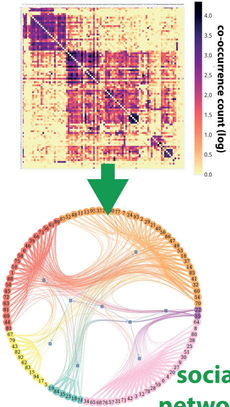
ingo.scholtes@uni-wuerzburg.de

Talk @ Inetum User Conference

2022/11/01

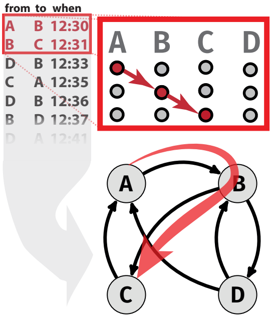
Machine Learning for Complex Networks

Machine Learning in Noisy Relational Data

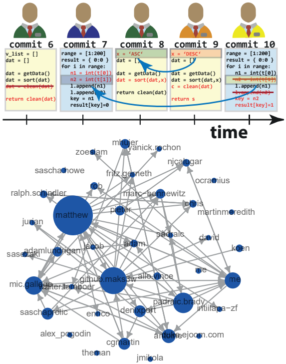


social network?

Graph Learning in Time Series Data



Data Science in Social Organizations



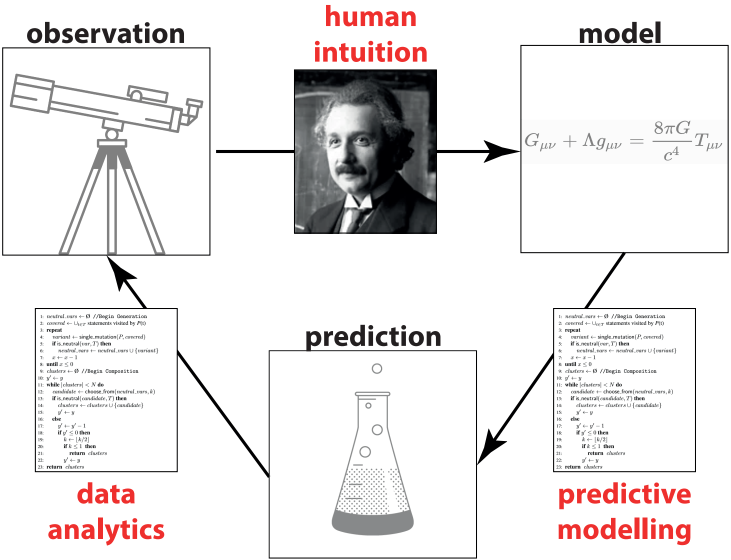
efficient team?

Horsehead Nebula
1375 light-years
December 2021

Great Orion Nebula
1344 light-years
January 2022

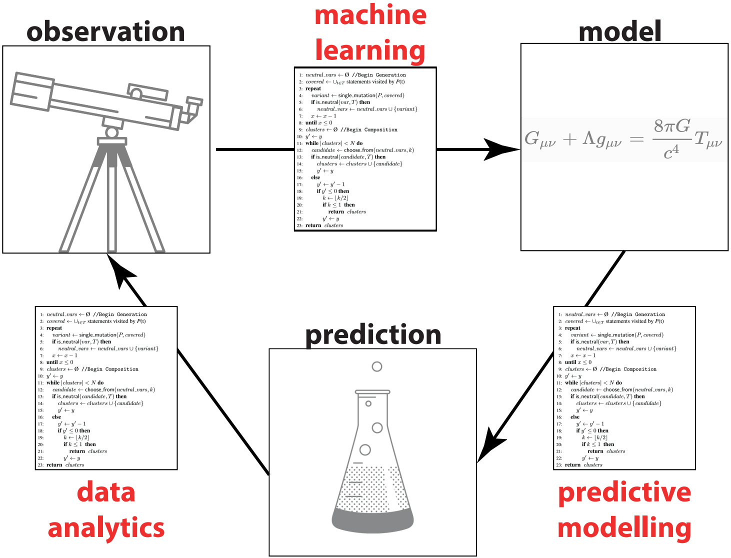
Andromeda Galaxy
2.5 mio light-years
December 2021

Learning



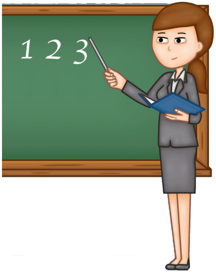
observation
human
intuition
model
$G_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4} T_{\mu\nu}$
prediction
1: neutral_vars ← ∅ //Begin Generation
2: covered ← ∪t∈T statements visited by P(t)
3: repeat
4: variant ← single_mutation(P, covered)
5: if is_neutral(var, T) then
6: neutral_vars ← neutral_vars ∪ {variant}
7: x ← x − 1
8: until x ≤ 0
9: clusters ← ∅ //Begin Composition
10: y′ ← y
11: while |clusters| < N do
12: candidate ← choose_from(neutral_vars, k)
13: if is_neutral(candidate, T) then
14: clusters ← clusters ∪ {candidate}
15: y′ ← y
16: else
17: y′ ← y′ − 1
18: if y′ ≤ 0 then
19: k ← ⌈k/2⌉
20: if k ≤ 1 then
21: return clusters
22: y′ ← y
23: return clusters
data
analytics
predictive
modelling

Machine learning



Machine learning - A primer

supervised learning
"learn" model
in labeled data



glossary: supervised techniques

- linear regression
- support vector machines
- artificial neural networks
- logistic regression

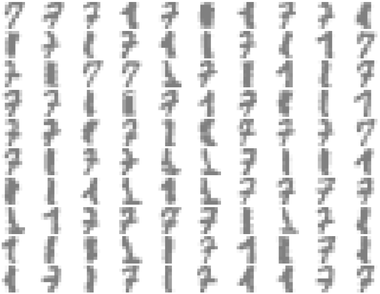
unsupervised learning
detect patterns
in unlabeled data

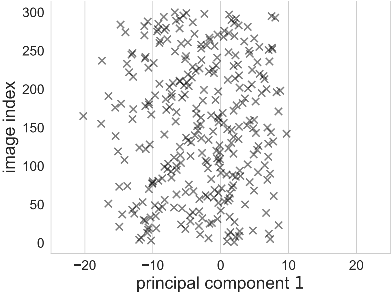
glossary: unsupervised techniques

- principal component analysis
- autoencoders
- representation learning
- k-means clustering

Classification problems

- problem: **high-dimensional data**
  - 8 x 8 Pixel $\Rightarrow \mathbb{N}^{64}$
  - reality: millions of pixels
- solution: **dimensionality reduction**
  - embed data in low-dimensional **feature space** $\mathbb{F}$
  - example: use d first **principal components** of data, i.e. $\mathbb{F} = \mathbb{R}^d$ ($d \ll 64$)





glossary: classification problem

- for feature space $\mathbb{F}$ and discrete classes $\mathbb{C}$ find **classifier** $C : \mathbb{F} \to \mathbb{C}$
- fundamental problem in **pattern recognition** and **machine learning**

Statistical binary classification

- use **training data** to compute **probability** $P(C = c|x)$ that object with feature x belongs to class c

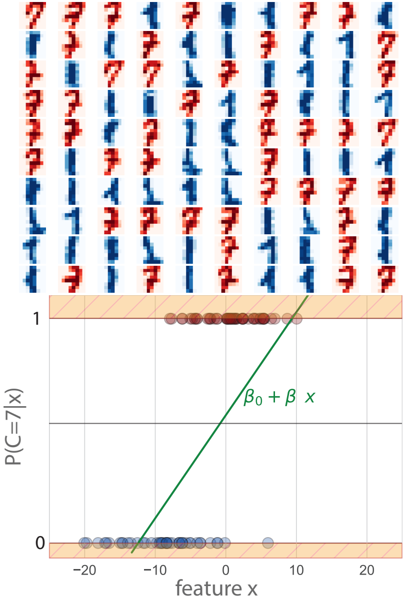
- for **dichotomous** $\mathbb{C} = \{1, 7\}$ consider

$$\mathbf{P(C = 7|x)} = \beta_0 + \beta_1 x$$

i.e. model for **linear relationship** between feature and class probability

- we need **transformation** σ such that

$$\mathbf{P(C = 7|x)} = \sigma\left(\beta_0 + \beta_1 x\right) \in [0, 1]$$



From linear to logistic model

- consider linear model for **logarithm of odds**

$$\log \frac{P(C=7|x)}{P(C=1|x)} = \beta_0 + \beta_1 x$$

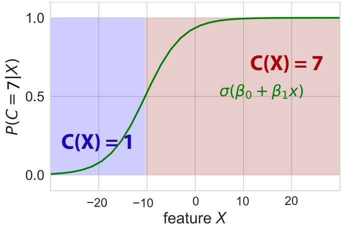
- we have

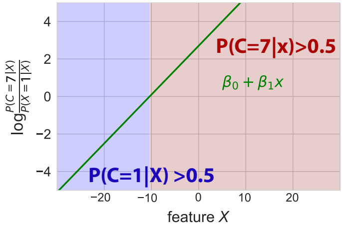
$$\frac{P(C=7|x)}{P(C=1|x)} = \frac{P(C=7|x)}{1-P(C=7|x)} = e^{\beta_0+\beta_1 x}$$

- and thus

$$P(C=7|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}} := \sigma(\beta_0+\beta_1 x)$$

- $\sigma(x) := \frac{1}{1+e^{-x}}$ is called **logistic function**





glossary: logistic regression

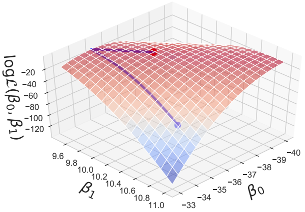
- statistical classifier with **linear decision boundary** between classes
- logistic "activation" function maps linear model to class probabilities $\longrightarrow$ cf. linear perceptron in neural networks

Fitting the model to training data

- for given β_0, β_1 logistic model gives probability $\mathbf{P(C = 7|x)}$ for any feature x

- for **training data** with features x_i and known classes c_i we can use $P(c_i|x_i)$ to compute **quality of model** $\mathcal{L}(\beta_0, \beta_1)$

- we use **gradient ascent algorithm** to find optimal parameters $\hat{\beta}_0, \hat{\beta}_1$ that **maximise quality of fitted model**



trajectory of gradient ascent algorithm in likelihood manifold for training data

observation

- we use **heuristic optimisation** to “learn” optimal model parameters in training data
- machine learning = **statistics + optimization**

gradient ascent optimization

1. start at random point $(\beta_0, \beta_1) \in \mathbb{R}^2$
2. move small step along **local gradients**
3. repeat 2 until convergence

Application to training images

let us apply **logistic regression** to our training images (using first principal component, i.e. features $x \in \mathbb{R}$)

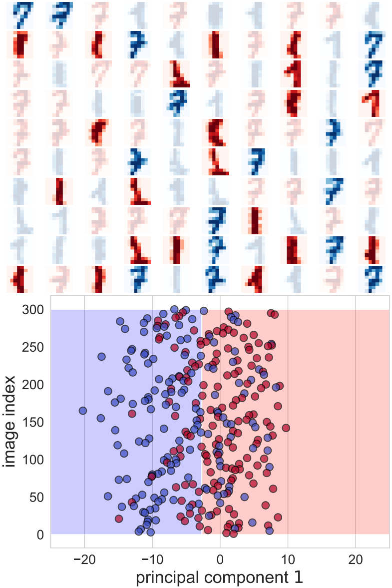
1. **learn model** in training data

gradient ascent algorithm

$$\hat{\beta}_0 = -37.5 \qquad \hat{\beta}_1 = 10.2$$

2. **classify images** based on feature x

3. we obtain **training error** of 33%



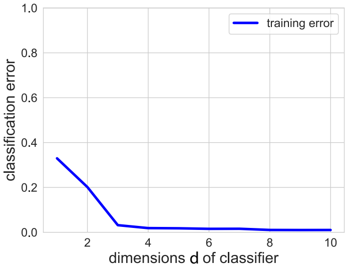
Increasing model complexity ...

- fit d-dimensional logistic regression classifier $C : \mathbb{R}^d \to \{1, 7\}$ based on **first d** principal components
- how does number of model dimensions affect **training error**?

model dimensions	training error
1	33 %
2	19 %
3	3 %
≥ 8	0 %

glossary: underfitting

- one-dimensional model **underfits** pattern in training data
- we can **reduce training error** by increasing model dimensionality



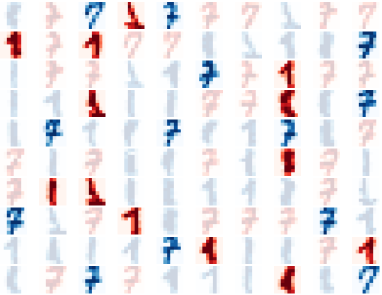
Generalisation to new data?

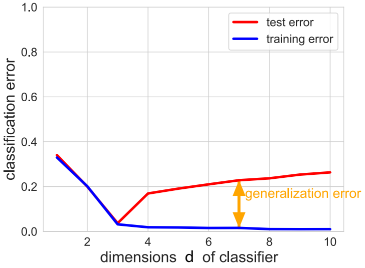
- how does our model **generalise to new data**?
- apply trained classifier to **test set** of images **not used during training**
- how does number of dimensions d of classifier affect **test error**?

$d > 3$: test error $\gg$ training error

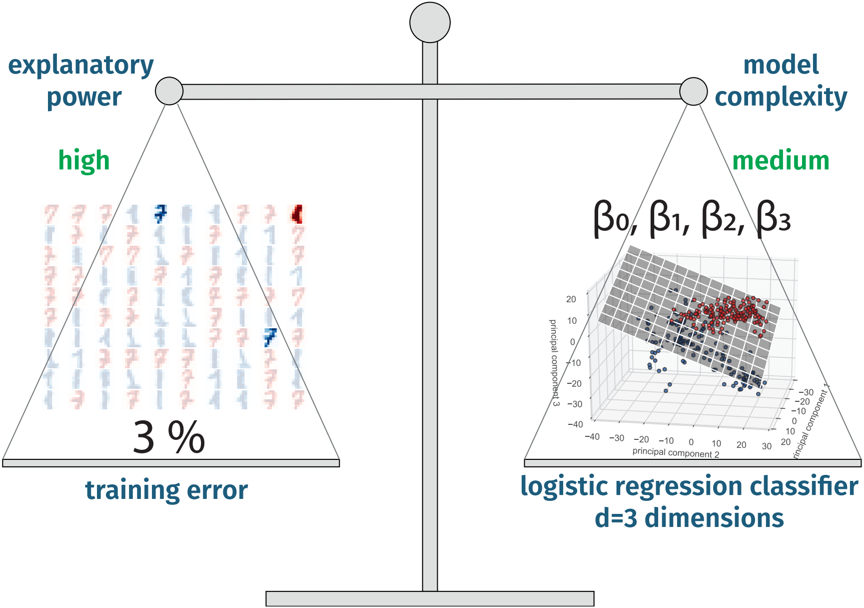
glossary: overfitting

- increasing number of dimensions **reduces generalisability** of our model
- use of complex models with many dimensions can lead to **overfitting**

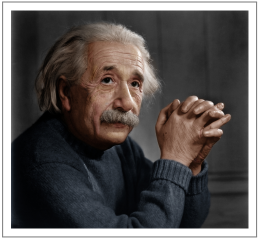




Model generalisability



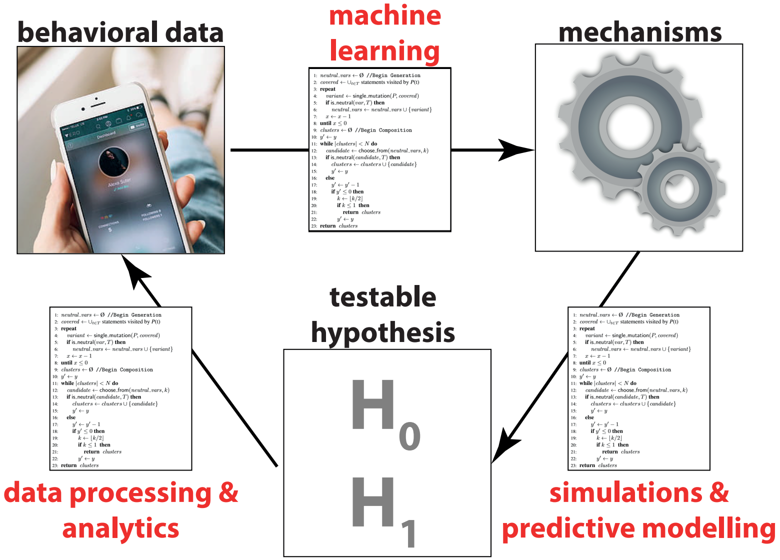
Epistemological challenges of ML



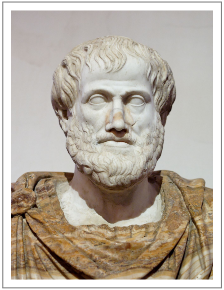
fundamental challenge in ML: how to learn generalisable models

- find **simplest model** that still **explains the data**
- requires **model selection techniques**, e.g. cross-validation, risk minimization, minimum description length, etc.

Natural Science Social Science Computational Social Science



From Aristotle ... to Ringelmann ...



"the whole is more than the sum of its parts" → Aristotle, 384 – 322 BC "many things have a plurality of parts and are not merely a complete aggregate but instead some kind of a whole beyond its parts"
→ Aristotle, 384 – 322 BC

"as soon as one couples two or several [men] to the same load, the work performed by each of them, at the same level of fatigue, decreases"
→ Ringelmann, 1913

What makes teams efficient?

- **Ringelmann effect**: members in larger teams are less productive

  → M Ringelmann: Recherches sur les moteurs animés: Travail de l'homme, 1913

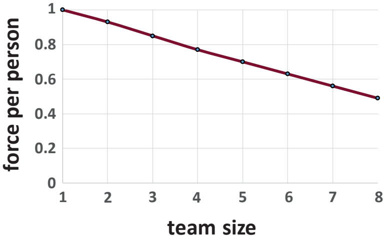
- classical experiment in **social psychology**

Brooks' law of software project management

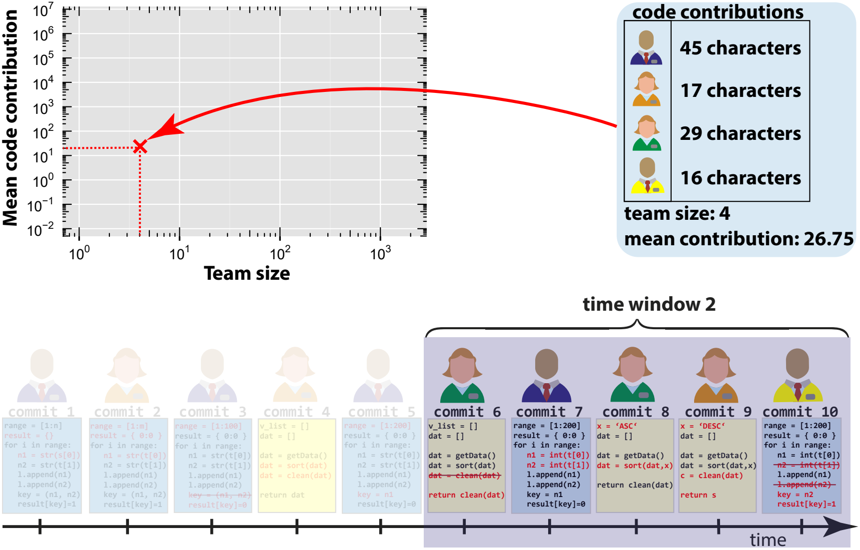
"Adding [wo]manpower to a late software project makes it later." → Fred Brooks, The Mythical Man Month, 1975

- possible **mechanisms**
  1. motivation: "social loafing"
  2. coordination overhead

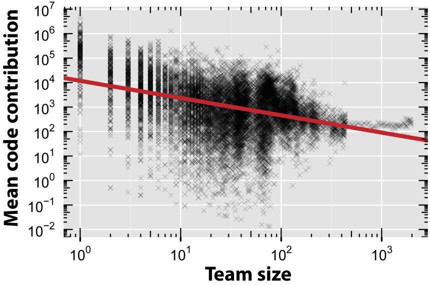
Maximilien Ringelmann
1861 – 1931



Repository mining



Testing the Ringelmann hypothesis



- we recover a **Ringelmann effect** across four orders of magnitude
- large fraction of unexplained variance = **limited use as predictive model**

key findings

- repository mining allows to test and quantify the **Ringelmann effect** at global scale

  → I Scholtes, P Mavrodiev, F Schweitzer, Empirical Software Engineering, 2016
- **new ways to operationalize** social science theories

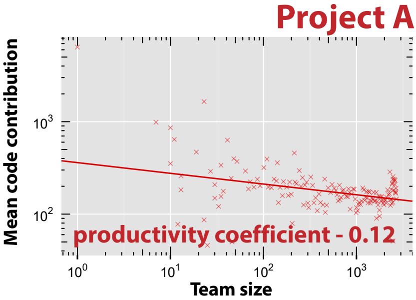
Why are some teams more efficient?

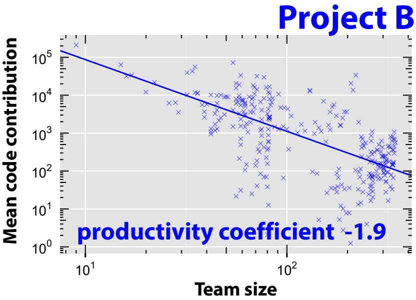
- we can apply our analysis to **individual projects** in our corpus
- cannot reject Ringelmann hypothesis for **any project**
- strong **project-dependent differences** of effect strength

example projects

- **project A**: doubling of team size → increase of output by approx. 90 %
- **project B**: doubling of team size → increase of output by approx. 10 %

- can we explain these differences?





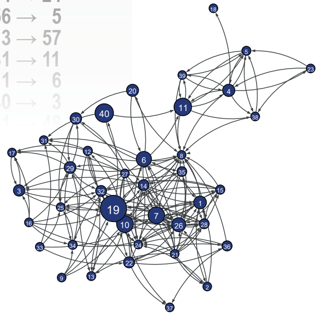
Machine Learning for Complex Networks

graphs and networks

- graph or network = universal **mathematical abstraction for complex system** consisting of many interacting parts
- important foundation to understand **collective phenomena** technical, biological, and social systems

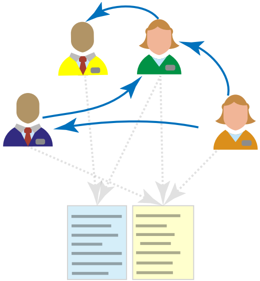
- can **network perspective** help us to explain differences between projects?
- problem: we lack data on on-/offline interactions between developers

from	→	to
10	→	15
43	→	38
22	→	57
19	→	35
2	→	25
48	→	31
30	→	21
8	→	7
55	→	17
20	→	27
3	→	40
51	→	5
4	→	24
56	→	5
3	→	57
31	→	11
1	→	6
40	→	3



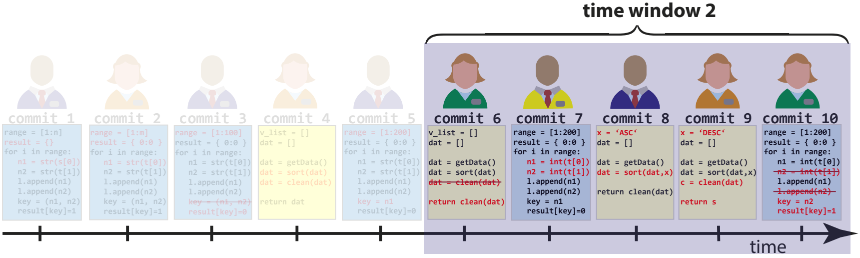
Inferring coordination networks

network of coordination requirements



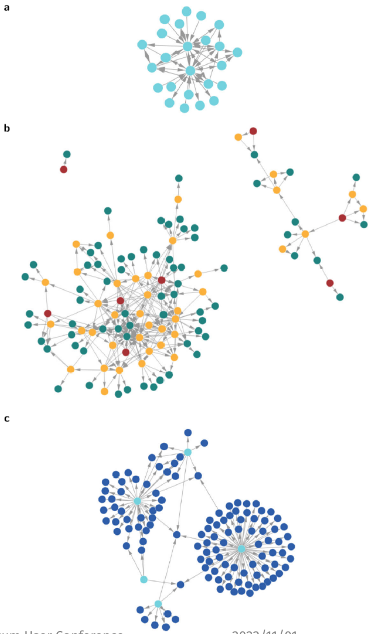
Open Source software `git2net`

- automates inference of **developer networks** from large git repositories
  → C Gote, I Scholtes, F Schweitzer, MSR 2019
- **2019 FOSS Impact Award**



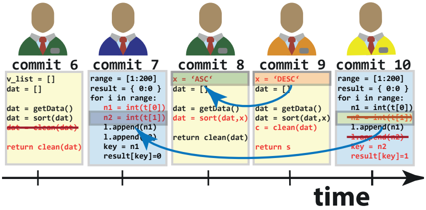
Network perspective on Ringelmann effect

- can **coordination networks** explain strength of Ringelmann effect?
- let us compare networks of **project A and B**
- how does **average number of coordination links per developer** change as team grows in size?
- **faster growth** of links related to **stronger decrease** of productivity
- can we study possible mechanisms at the level of **individual developers**?



Mechanism behind Ringelmann effect?

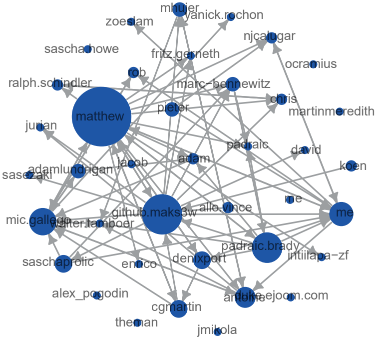
- we can study relationship between **code ownership** and **individual code production**
- **own code** = lines of code last edited by a given developer



working hypothesis

cognitive overhead involved in editing **own code** smaller compared to **foreign code**

- how does code ownership influence **time needed for edits**
- how does **team size** influence editing behavior?



Big Data = Big Insights?

- **big repository data** = insights into how collaboration patterns influence productivity …
- …but recent studies came to **different conclusions** despite using similar data and methods
- example: **super-linear productivity** of developers in software teams

> […] **previous studies only analyzed a few hundred software development groups** working on highly popular software projects. In this paper, **we address these issues by expanding the analysis to millions of groups** […]
>
> ⟶ G Muric et al., ACM HCI, 2019

2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)

Big Data = Big Insights? Operationalising Brooks' Law in a Massive GitHub Data Set

Christoph Gote
cgote@ethz.ch
Chair of Systems Design, ETH Zurich
Zurich, Switzerland

Pavlin Mavrodiev
pmavrodiev@ethz.ch
Chair of Systems Design, ETH Zurich
Zurich, Switzerland

Frank Schweitzer
fschweitzer@ethz.ch
Chair of Systems Design, ETH Zurich
Zurich, Switzerland

Ingo Scholtes*
ingo.scholtes@uni-wuerzburg.de
Chair of Computer Science XV - Machine Learning for Complex Networks, Julius-Maximilians-Universität Würzburg
Würzburg, Germany

ABSTRACT

Massive data from software repositories and collaboration tools are widely used to study social aspects in software development. One question that several recent works have addressed is how a software project's size and structure influence team productivity, a question famously considered in *Brooks' law*. Recent studies using massive repository data suggest that developers in larger teams tend to be less productive than smaller teams. Despite using similar methods and data, other studies argue for a positive linear or even super-linear relationship between team size and productivity, thus contesting the view of software economics that software projects are *diseconomies of scale*.

In our work, we study challenges that can explain the disagreement between recent studies of developer productivity in massive repository data. We further provide, to the best of our knowledge, the largest, curated corpus of *GitHub* projects tailored to investigate the influence of team size and collaboration patterns on individual and collective productivity. Our work contributes to the ongoing discussion on the choice of productivity metrics in the operationalisation of hypotheses about determinants of successful software projects. It further highlights general pitfalls in big data analysis and shows that the use of bigger data sets does not automatically lead to more reliable insights.

ACM Reference Format:
Christoph Gote, Pavlin Mavrodiev, Frank Schweitzer, and Ingo Scholtes. 2022. Big Data = Big Insights? Operationalising Brooks' Law in a Massive GitHub Data Set. In *44th International Conference on Software Engineering (ICSE '22), May 21–29, 2022, Pittsburgh, PA, USA*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3510003.3510619

*Also with Data Analytics Group, Department of Informatics, University of Zurich.

ACM ISBN 978-1-4503-9221-1/22/05.
https://doi.org/10.1145/3510003.3510619

1 INTRODUCTION

Empirical research across disciplines is nowadays driven by the availability of big data and methods to process and analyse them efficiently. In empirical software engineering, massive data from software repositories and online collaboration tools are widely used to investigate social and human aspects in software development. This intersects with *computational social science*, which uses big data to test hypotheses about individual and collective human behaviour originally developed in sociology, social psychology, or organisational theory. Data-driven studies of developer productivity in large software projects are an exemplary case of how research in empirical software engineering can advance computational social science. The question of how factors like, e.g., team size, influence the productivity of team members was already addressed by Maximilien Ringelmann [1] in 1913. In social psychology, his finding that individual productivity tends to linearly decrease with team size is known as the *Ringelmann effect*. In software project management, a similar observation is famously paraphrased as *Brooks' law* [2]. Here, the anecdote that "adding manpower to a late project makes it later" captures that the overhead associated with growing team sizes can reduce team efficiency. Studies of collaborative software projects found evidence for a strong Ringelmann effect for different team sizes, programming languages, and development phases [3–6]. Other studies, however, found a positive linear or even super-linear relationship between the size of a team and the productivity of its members [7–9].

The fact that different works studying the same research question yield qualitatively different results, despite applying similar methods to data from similar or even identical sources, should concern us. Referring to the massive number of projects, commits, or developers covered in their studies, authors often corroborate their findings by the size of the data used to obtain them, thus implying that the analysis of bigger data automatically yields more reliable insights. This points to an important general issue relevant for empirical research beyond software engineering: Apart from advantages in terms of coverage, resolution, or statistical confidence, the use of big data also introduces new *threats* for the validity of results. To address this issue, in this work, we explore *four challenges in the analysis of big data*. We study those challenges in a massive GitHub data set and argue that they are likely to explain

262

⟶ C Gote, P Mavrodiev, I Scholtes, F Schweitzer, ICSE 2022

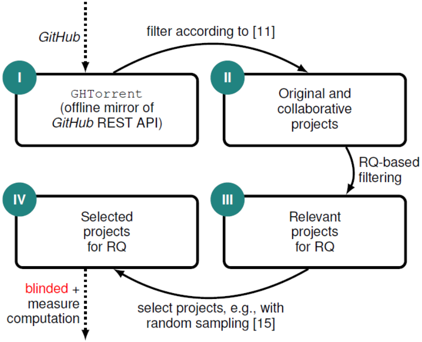
Challenge 1: Data quality

- `git` repository $\neq$ software project

example in `gitHub`

user `efarberger` hosts three repositories with > 18 million (!) commits

> "We recommend that researchers interested in performing studies using GitHub data [...] target the **data that can really provide information** towards answering their research questions." → E Kalliamvakou et al., MSR, 2014



- **repository selection and sampling pipeline** to systematically create balanced corpus of projects

only 1.4 % of 125 million projects on `gitHub` fulfill criteria of collaborative, active, and non-duplicate software projects proposed in empirical software engineering literature

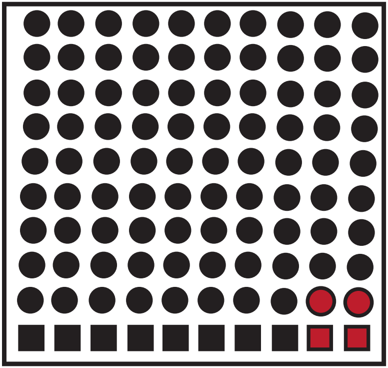
→ C Gote, P Mavrodiev, I Scholtes, F Schweitzer, ICSE 2022

Challenge 2: Population validity

- careless sampling can distort results in **unbalanced populations**

example in `gitHub`

- **heavily skewed** distribution of team sizes
- naive random sample likely to consist of projects with **smallest team sizes**
- not suitable to reason about effect of **team size** on developer productivity



- use **stratified random sampling** to obtain balanced sample of projects with **different team sizes**
- non-linear relationship suggests **optimal team size** of $\approx 10 - 20$ developers

Challenge 3: Omitted variable bias

- **omitted variables** can lead to **biased models** and question **causal interpretation** of found relationships

exemplary issue

- larger teams associated with larger mean in-degree, i.e. **more connections per developer**
- larger mean in-degree associated with **higher overall productivity** but **stronger negative effect** of team size

- variant of **Simpson's paradox** where effect in aggregate population is reversed compared to groups

- to isolate effect of team size we control for **interactions between team size and network characteristics**

	Commits	LevD	CycC	NLOC	HalEff
(IC)	2.72^{***}	10.85^{***}	5.01^{***}	6.68^{***}	15.91^{***}
	(0.11)	(0.14)	(0.15)	(0.13)	(0.23)
TS (log)	-0.22^{***}	-0.44^{***}	-0.42^{***}	-0.40^{***}	-0.48^{***}
	(0.03)	(0.04)	(0.04)	(0.04)	(0.06)
InD (log)	1.83^{***}	1.80^{***}	1.76^{***}	1.80^{***}	1.72^{***}
	(0.12)	(0.16)	(0.17)	(0.15)	(0.26)
TS$\times$InD	-0.18^{***}	-0.10	-0.08	-0.10^{*}	0.05
	(0.03)	(0.04)	(0.04)	(0.04)	(0.07)
FModR	-0.92^{***}	-1.79^{***}	-2.62^{***}	-2.29^{***}	-3.28^{***}
	(0.22)	(0.28)	(0.30)	(0.27)	(0.46)
R^2	0.47	0.47	0.46	0.50	0.34
Adj. R^2	0.46	0.47	0.46	0.49	0.33

$^{***}p < 0.001$; $^{**}p < 0.01$; $^{*}p < 0.05$

regression analysis controlling for network characteristics

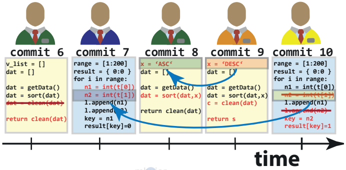
In summary

- data science and machine learning have become **vital parts of the scientific method** across disciplines

- massive repository data can yield **insights into software development processes**

- **graph learning** helps to understand how **topology of interactions** influences **collective behavior** …

- …but, we must carefully address **epistemological challenges in big data**



GIT2NET

https://github.com/gotec/git2net

PATHPY

http://www.pathpy.net