An Empirical Investigation of Forks as Variants in npm

John Businge
LORE Lab

Universiteit Antwerpen

Collaborators

John Businge
LORE Lab

Universiteit Antwerpen

Ahmed Zerouali
Soft Lab

VRIJE UNIVERSITEIT BRUSSEL

Alexandre Decan
Software Engineering lab

UMONS Université de Mons

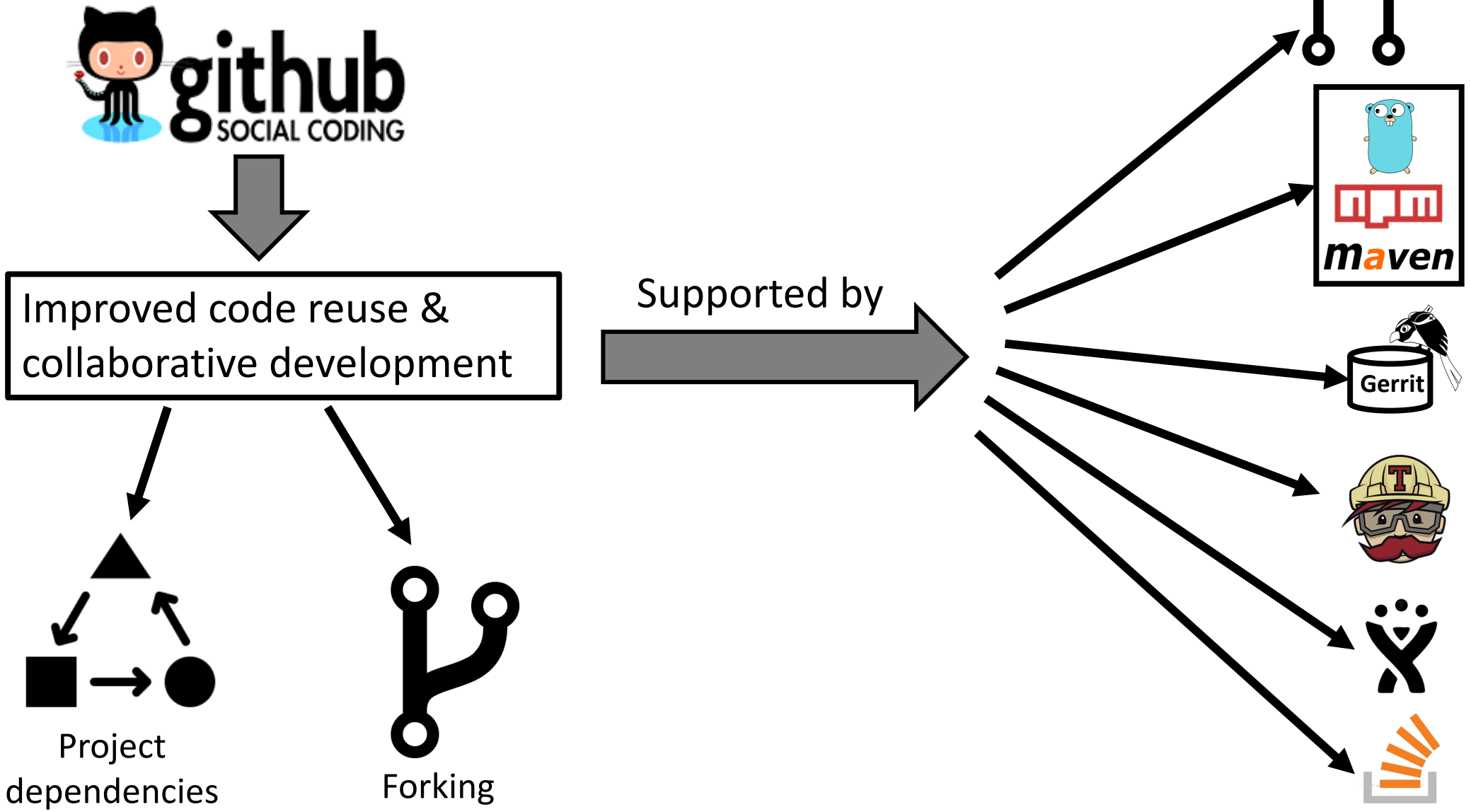
Tom Mens
Software Engineering lab

UMONS Université de Mons

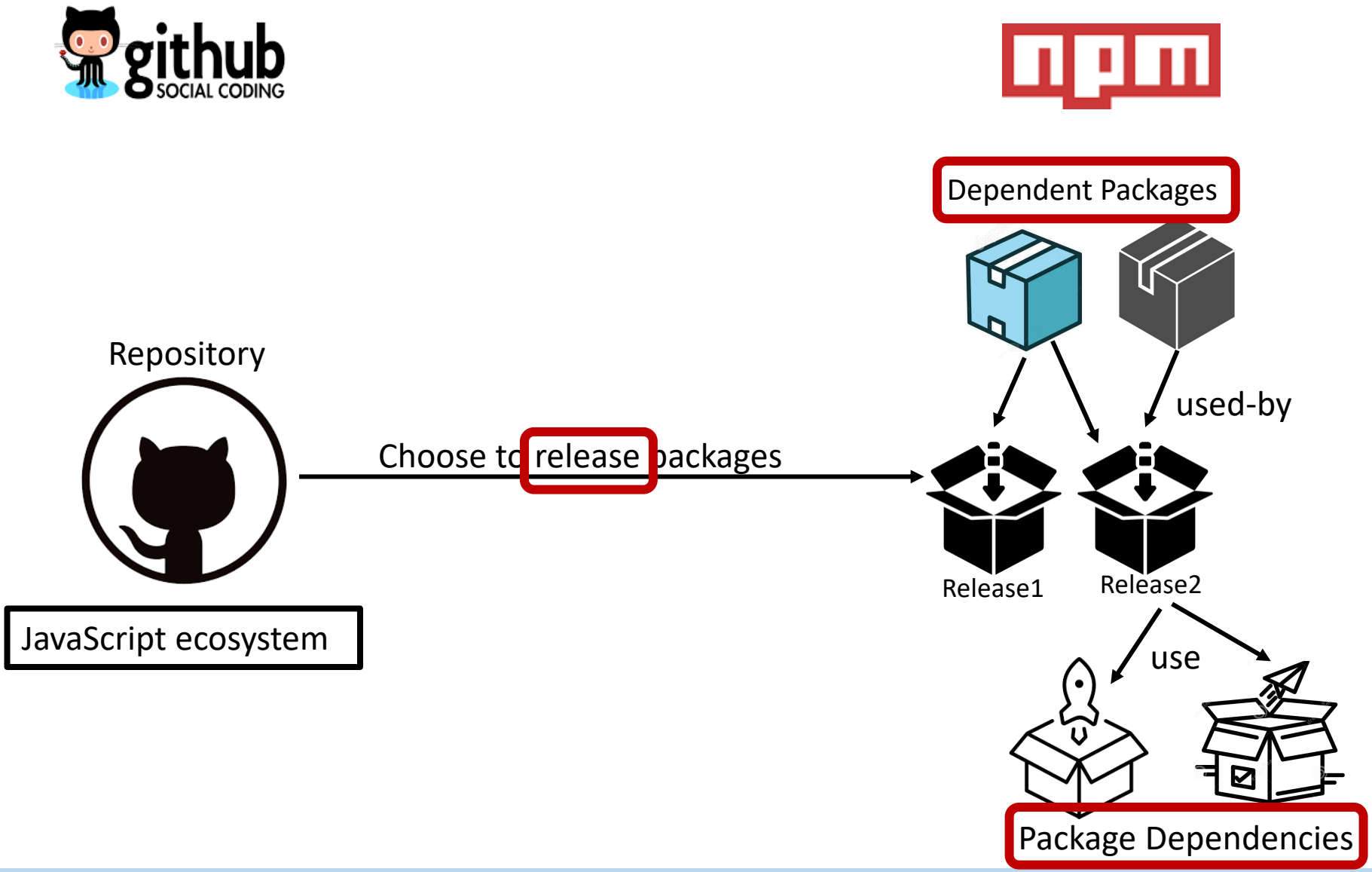
Serge Demeyer
LORE Lab

Universiteit Antwerpen

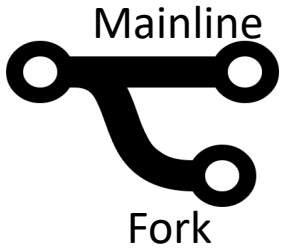
Introduction



Introduction – Terminology

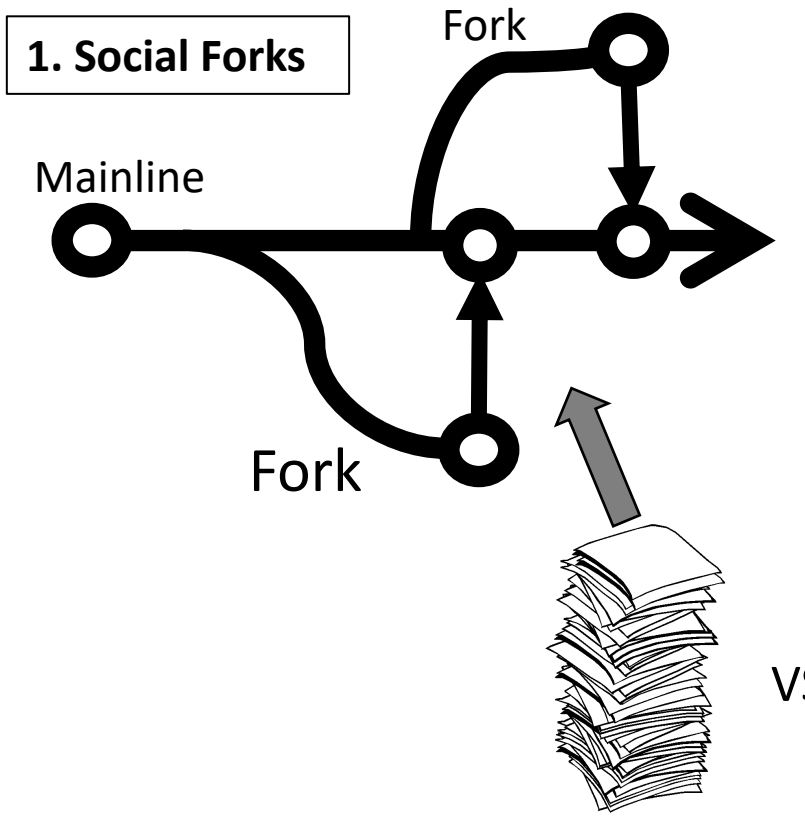


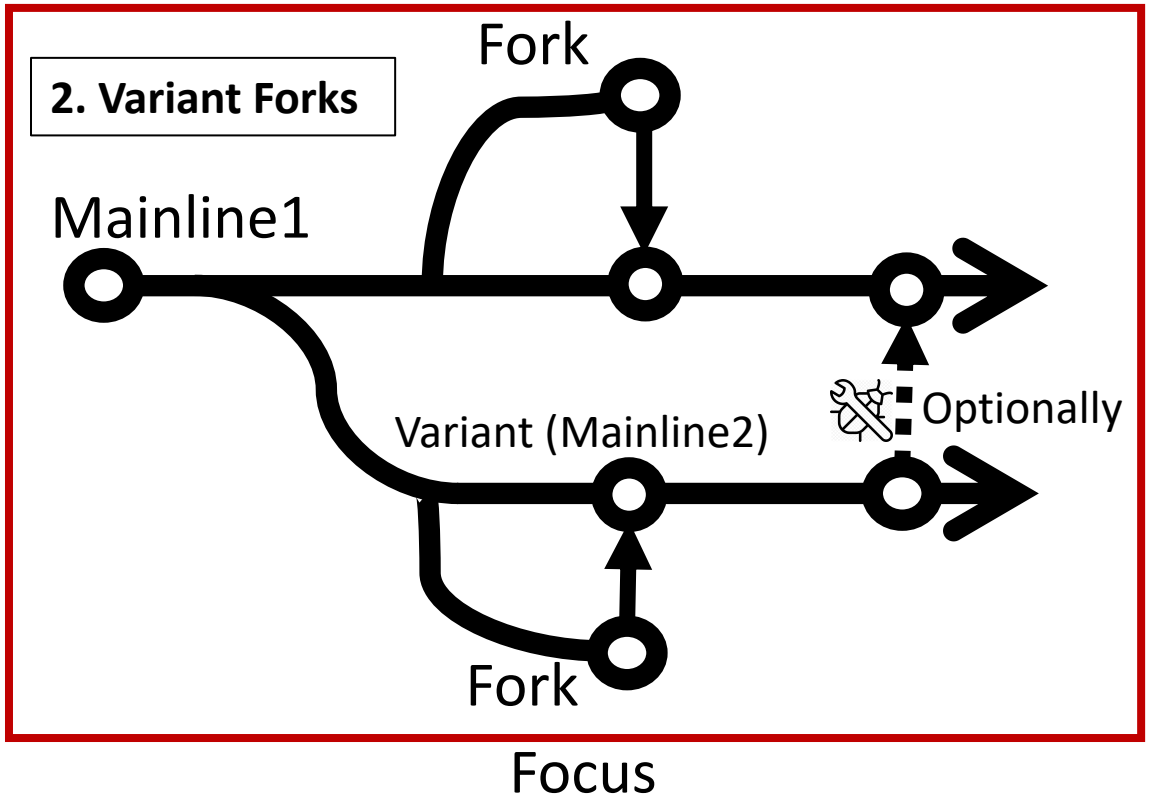
Introduction - Research focus



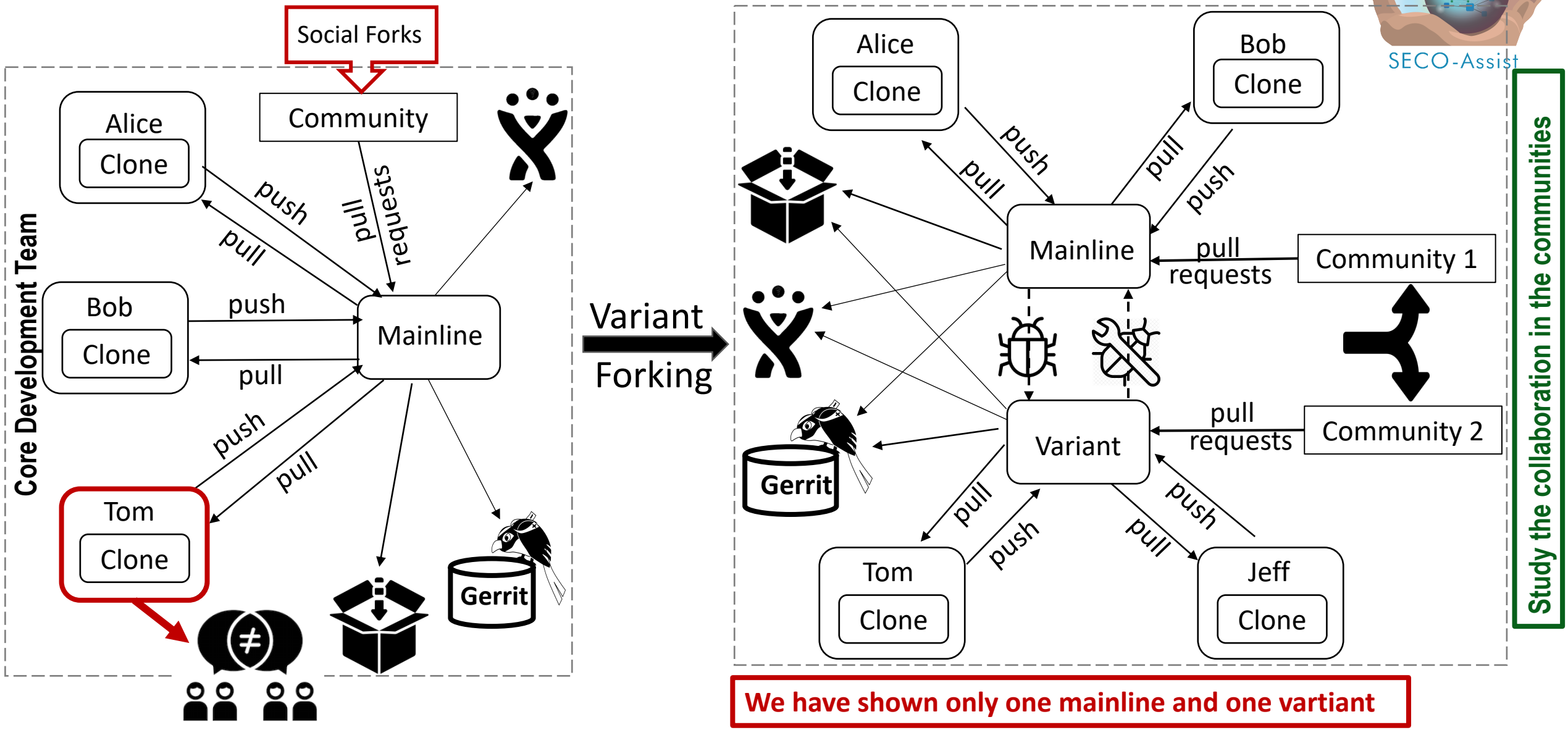
Two main reasons for forking

1. To fix bugs/feature in the mainline and merge the fork thereafter **(social fork)**.
2. To use the mainline code as a starting point of a new & related project **(variant fork)**.





Introduction – Illustration of variant forking



Introduction – Concrete example

- creationix/wheat is a mainline hosted on GitHub
  - It has 136 forks
  - 2 of the 136 forks (**sun11/wheat** and **frodare/barley**) have their package releases distributed on npm.
  - The **2 are variant forks** and the majority of the **134 are social forks**

	Releases	Dependencies	Dependent packages	Dependent projects
creationix/wheat (M)	13	77	2	23
frodare/barley (V)	1	8	0	0
sun11/wheat (V)	1	6	1	1

software family

We comapre mainlines vs variants for the four technical aspects.

Goal and Research questions

Ultimate Goal

To empirically investigating the socio-technical evolution of **software families** within the **npm software ecosystem**.

1. **Social** – collaboration aspects
2. **Technical** – package releases, dependencies and dependents.

First Step!

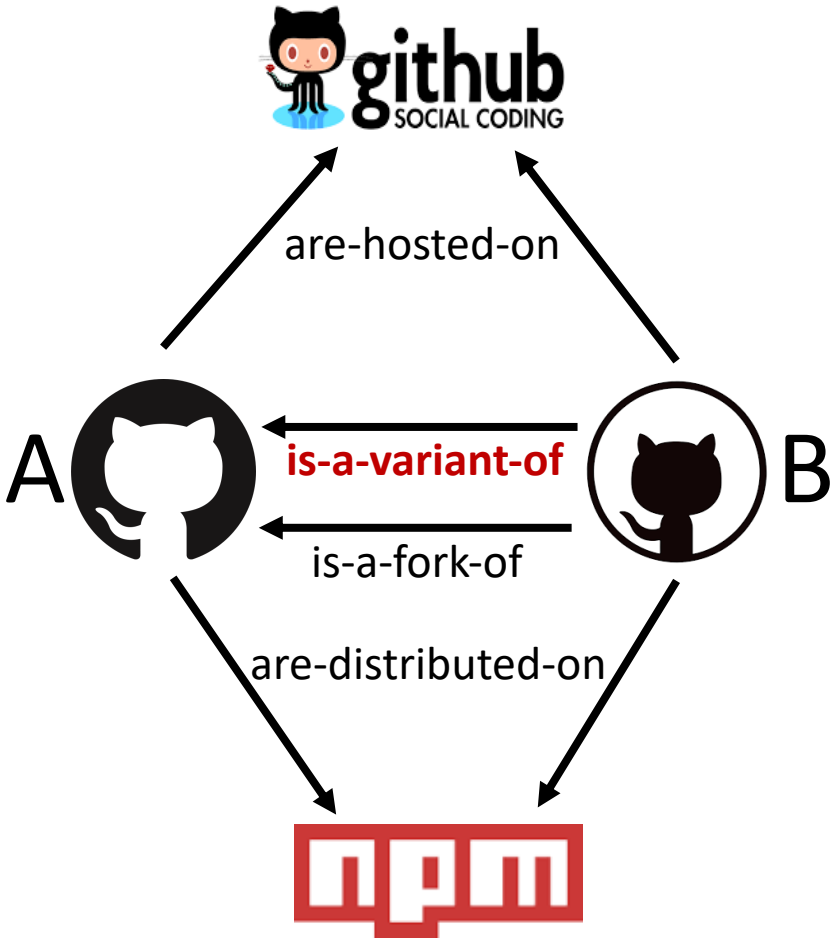
To perform an exploratory investigation on the evolution of variants focusing on their technical aspects.

Research Questions

- **RQ0**: How prevalent are software families in the JavaScript ecosystem on GitHub?
- **RQ1 – R3**: How do the distributions of package releases, package dependencies, dependent packages/projects compare for mainlines vs variants?

Methods and Dataset

Variant of a repository

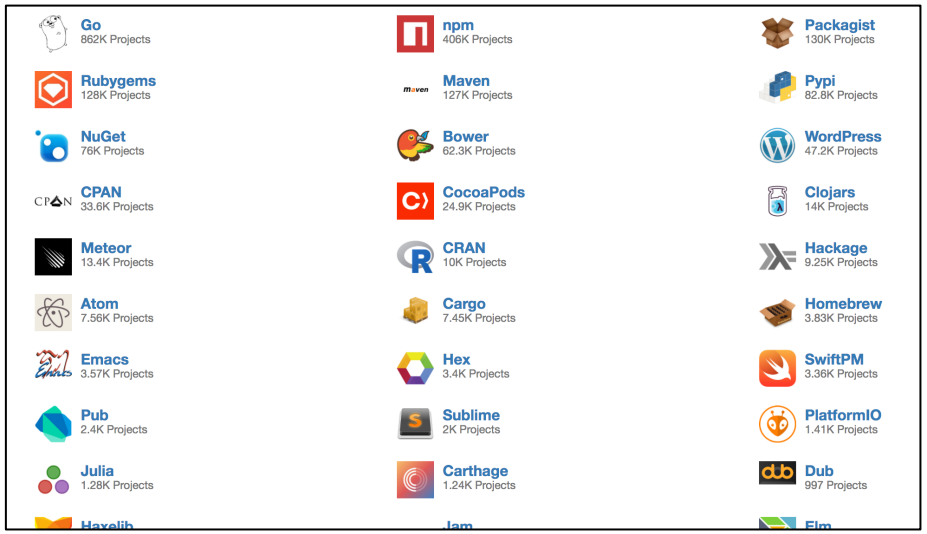


Data

Extracted from

Dataset

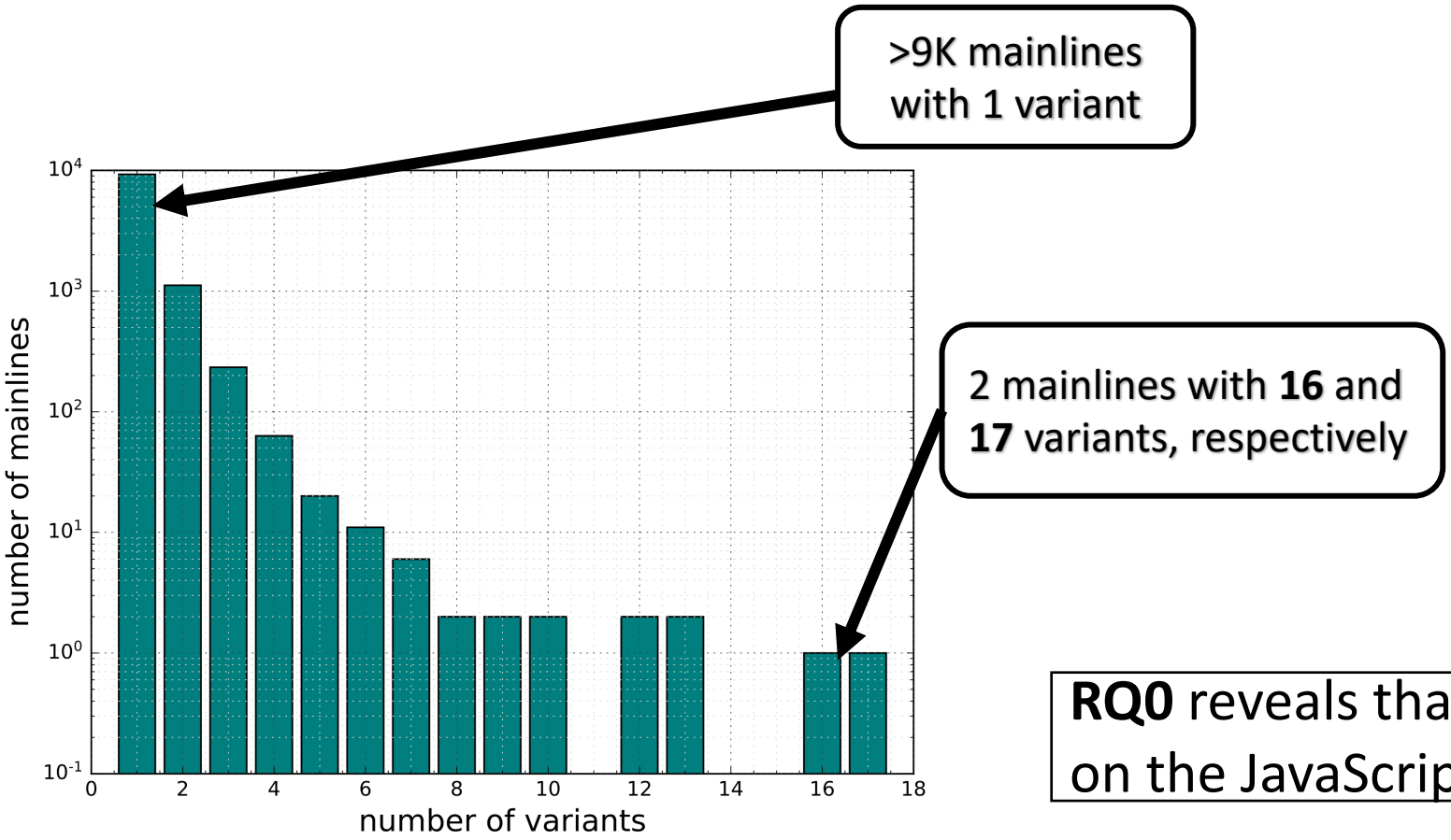
Libraries.io (Release 1.6.0)



Results – Number of Variants

RQ0: How prevalent are software families in the JavaScript ecosystem on GitHub?

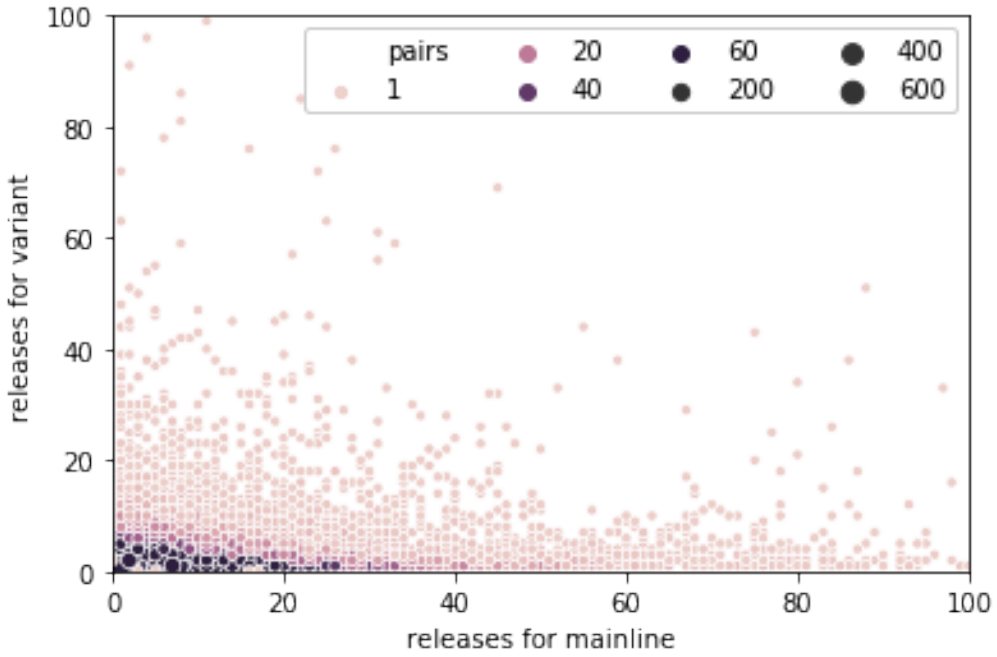
- **10,743 mainlines with 12,813 variants in total.**



RQ0 reveals that software families indeed exist on the JavaScript ecosystem on GitHub

Results – Package releases

RQ1: How do the distributions of package releases in mainlines and their variants compare to each other?



From the graph we observe a good number of variants being maintained in parallel with their mainline counterparts.

Examples: variant-releases > mainline-releases

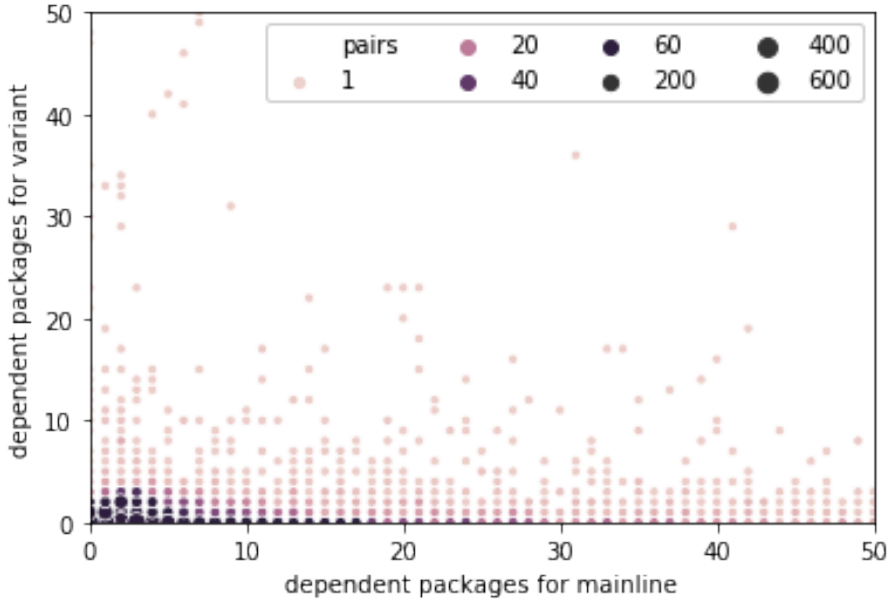
#	Mainline	Variant	Mainline Releases	Variant Releases	Diff
1	weex-pack	weexpack	1	129	128
2	restyped-giphy-api	restyped-staffjoy-api	1	116	115
3	cogs-javascript-sdk	cogs-sdk	5	104	99
4	gulp-galen	gulp-galenframework	11	99	88

Last updated:

- Mainline – **5 years ago**
- Variant – **1 year ago**

Results – Dependent packages

RQ3: Do variant packages exhibit dependent packages more than their mainline counterparts?



Examples: Variant-dependent-packages > Mainine-dependent-packages

#	Mainline	Variant	Mainline Dependent Packages	Variant Dependent Packages	Diff
1	selenium	selenium-server	97	2046	1949
2	replace2	replace	0	1043	1043
3	grunt-mocha-screenshot	grunt-mocha	2	651	649
4	mocha-istanbul	grunt-mocha-istanbul	606	987	381

We do observe some variants having more dependent packages compared to their mainline counterparts

Last maintained:

- selenium **9 years ago**
- selenium-server **2 years ago**

Conclusion & Future work

- We performed an exploratory study on the evolution of variants focusing on their technical aspects
  1. We have identified large number of variants
  2. We have observed that some of these variants are actively being maintained and other developers are interested in using them

Future work

- Empirically a detailed investigating on the socio-technical evolution of **software families** within the **npm software ecosystem**.
  1. **Social** – collaboration aspects
  2. **Technical** – Commits, package releases, dependencies and dependents.

- Collaboration is most welcome
- Happy to share our dataset ☺

Thank you for Listening !