UMONS
Université de Mons

Faculté
des Sciences

A Measurement Framework for Analyzing Technical Lag in Open-Source Software Ecosystems

Presented by:
Ahmed Zerouali

Advisor:
Dr. Tom Mens

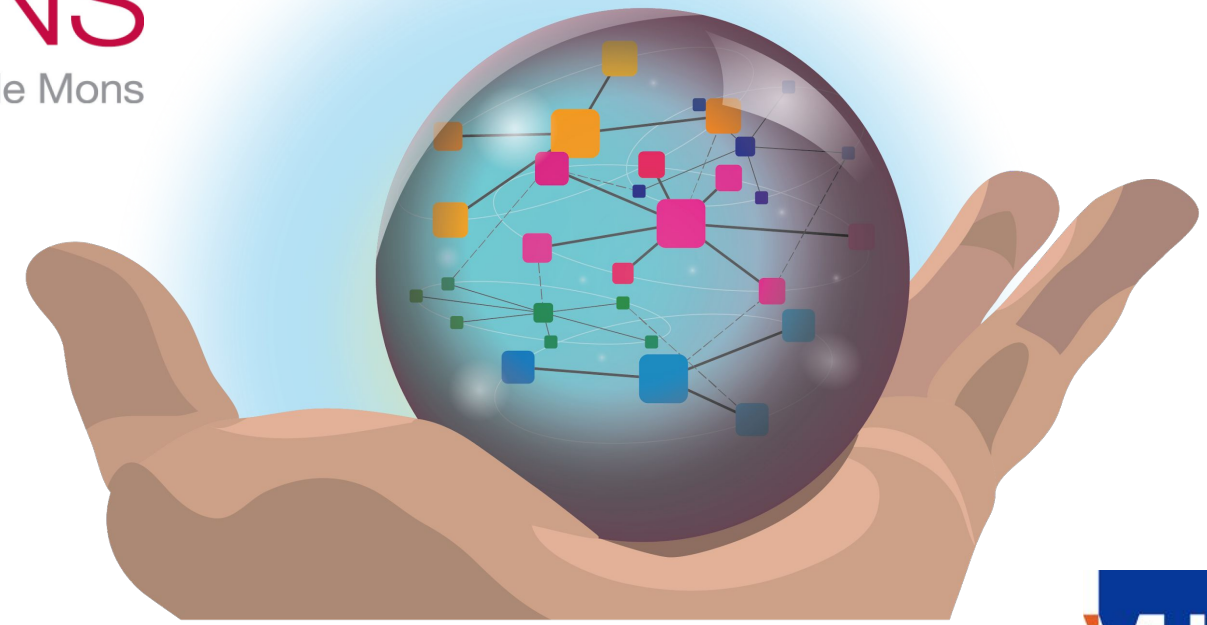
Jury:

Dr. Olivier Delgrange
Dr. Alexandre Decan
Dr. Jesus Gonzalez-Barahona
Dr. Alexander Serebrenik

Public PhD Defense
Software Engineering Lab, Université de Mons - Belgium, 4 September 2019

Belgian Excellence of Science Research Project 2018- 2021

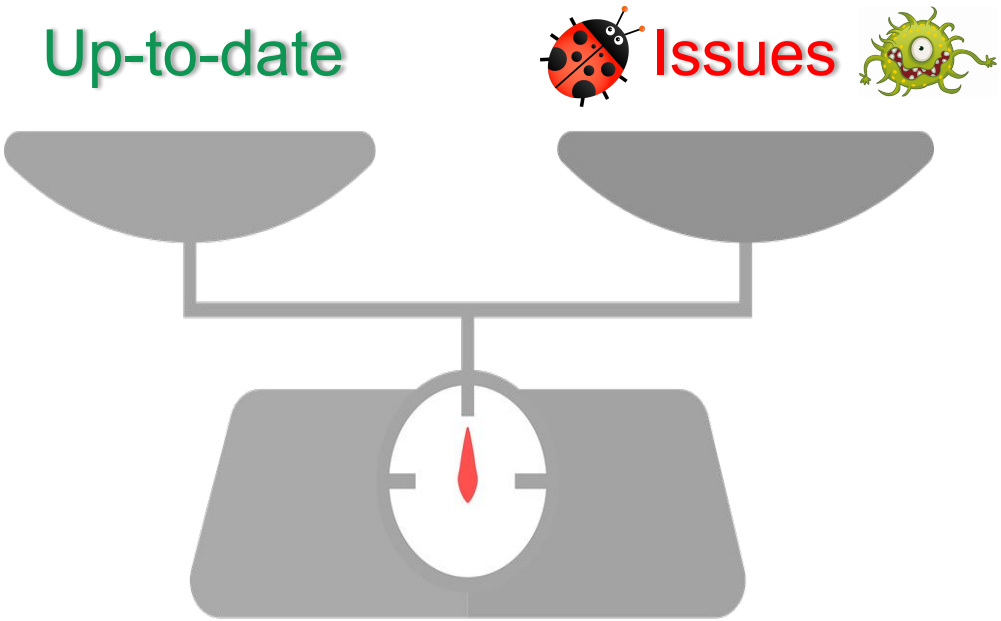
https://secoassist.github.io/



SECO-Assist

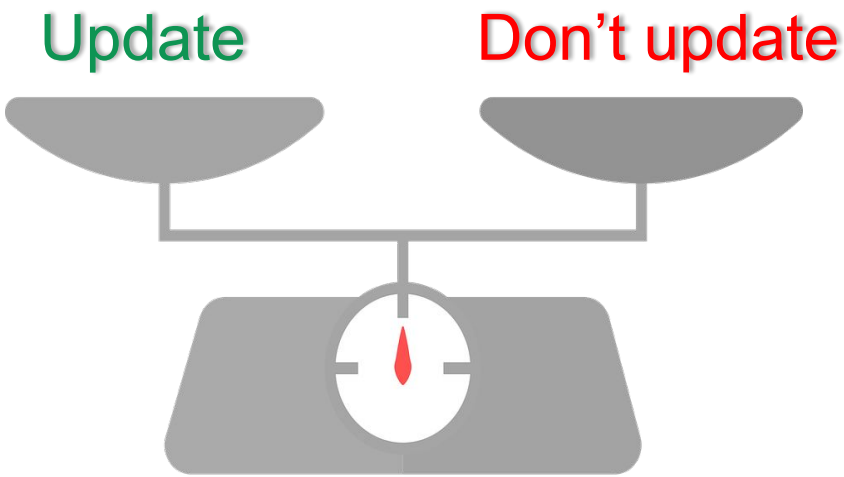
Background

Focus



Focus

*How can we help software developers to decide **when** and **why** they should update* **?**



Published papers

TITLE	YEAR
ConPan: a tool to analyze packages in software containers A Zerouali, V Cosentino, G Robles, JM Gonzalez-Barahona, T Mens Proceedings of the 16th International Conference on Mining Software …	2019
On the impact of outdated and vulnerable javascript packages in docker images A Zerouali, V Cosentino, T Mens, G Robles, JM Gonzalez-Barahona 2019 IEEE 26th International Conference on Software Analysis, Evolution and …	2019
On the diversity of software package popularity metrics: An empirical study of npm A Zerouali, T Mens, G Robles, JM Gonzalez-Barahona 2019 IEEE 26th International Conference on Software Analysis, Evolution and …	2019
On the Relation between Outdated Docker Containers, Severity Vulnerabilities, and Bugs A Zerouali, T Mens, G Robles, JM Gonzalez-Barahona 2019 IEEE 26th International Conference on Software Analysis, Evolution and …	2019
A formal framework for measuring technical lag in component repositories—and its application to npm A Zerouali, T Mens, J Gonzalez-Barahona, A Decan, E Constantinou, ... Journal of Software: Evolution and Process, e2157	2019
[Engineering Paper] Graal: The Quest for Source Code Knowledge V Cosentino, S Dueñas, A Zerouali, G Robles, JM Gonzalez-Barahona 2018 IEEE 18th International Working Conference on Source Code Analysis and …	2018
An empirical analysis of technical lag in npm package dependencies A Zerouali, E Constantinou, T Mens, G Robles, J González-Barahona International Conference on Software Reuse, 95-110	2018
An empirical comparison of the development history of cloudstack and eucalyptus A Zerouali, T Mens Proceedings of the 2017 International Conference on Smart Digital …	2017
Analyzing the evolution of testing library usage in open source Java projects A Zerouali, T Mens 2017 IEEE 24th International Conference on Software Analysis, Evolution and …	2017

Technical Lag

Technical lag
> Background

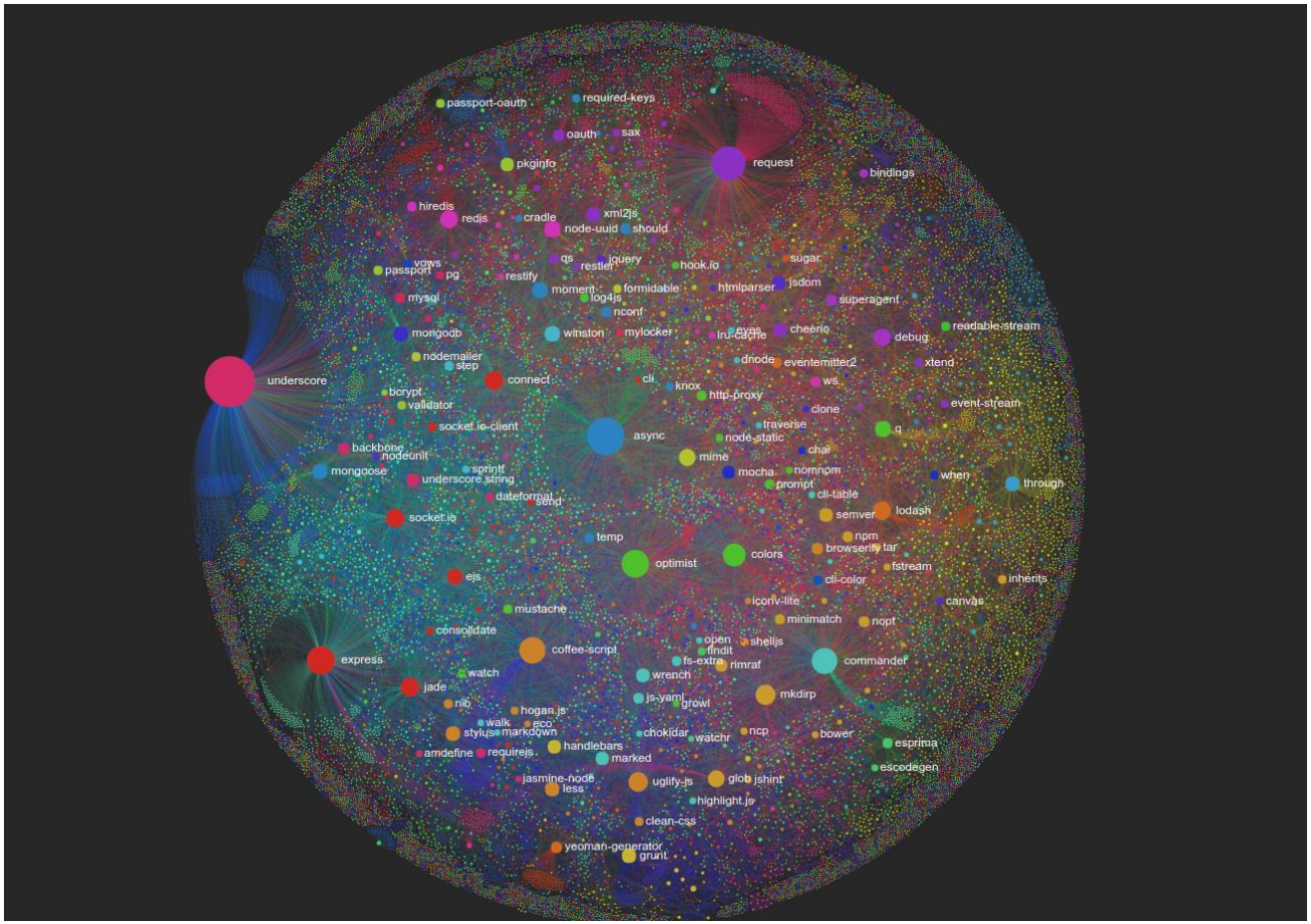
Technical lag[1]: the increasing **difference** between deployed software packages and the **ideal** available upstream packages.

- **Ideal:** stability, security, functionality, recency, etc.
- **Difference:** version updates, bugs, vulnerabilities, lines of code, commits, etc.

[1] Gonzalez-Barahona, et al. "Technical Lag in Software Compilations: Measuring How Outdated a Software Deployment Is." *IFIP International Conference on Open Source Systems*. Springer, Cham, 2017.

Technical lag
> Background

+20M dependencies



Credits: https://exploring-data.com/vis/npm-packages-dependencies/

Technical lag
> Background

youtube-player

5.5.2 • Public • Published 4 months ago

Readme | 3 Dependencies

Dependencies (3)

debug load-script sister

Dev Dependencies (15)

ava babel-cli babel-plugin-add-module-exports

babel-plugin-transform-flow-strip-types babel-plugin-transform-ob

babel-preset-env babel-register chai eslint eslint-config-canonic

flow-copy-source husky npm-watch semantic-release

package.json

```
13  "dependencies": {
14    "debug": "^2.6.6",
15    "load-script": "^1.0.0",
16    "sister": "^3.0.0"
17  },
18  "description": "YouTube IFrame Player API abstraction.",
19  "devDependencies": {
20    "ava": "^0.19.1",
21    "babel-cli": "^6.24.1",
22    "babel-plugin-add-module-exports": "^0.2.1",
23    "babel-plugin-transform-flow-strip-types": "^6.22.0",
24    "babel-plugin-transform-object-rest-spread": "^6.23.0",
25    "babel-preset-env": "1.4.0",
26    "babel-register": "^6.24.1",
27    "chai": "^3.5.0",
28    "eslint": "^3.19.0",
29    "eslint-config-canonical": "^8.2.0",
30    "flow-bin": "^0.45.0",
31    "flow-copy-source": "^1.1.0",
32    "husky": "^0.13.3",
33    "npm-watch": "^0.1.9",
34    "semantic-release": "^6.3.2"
35  },
36  "keywords": [
```

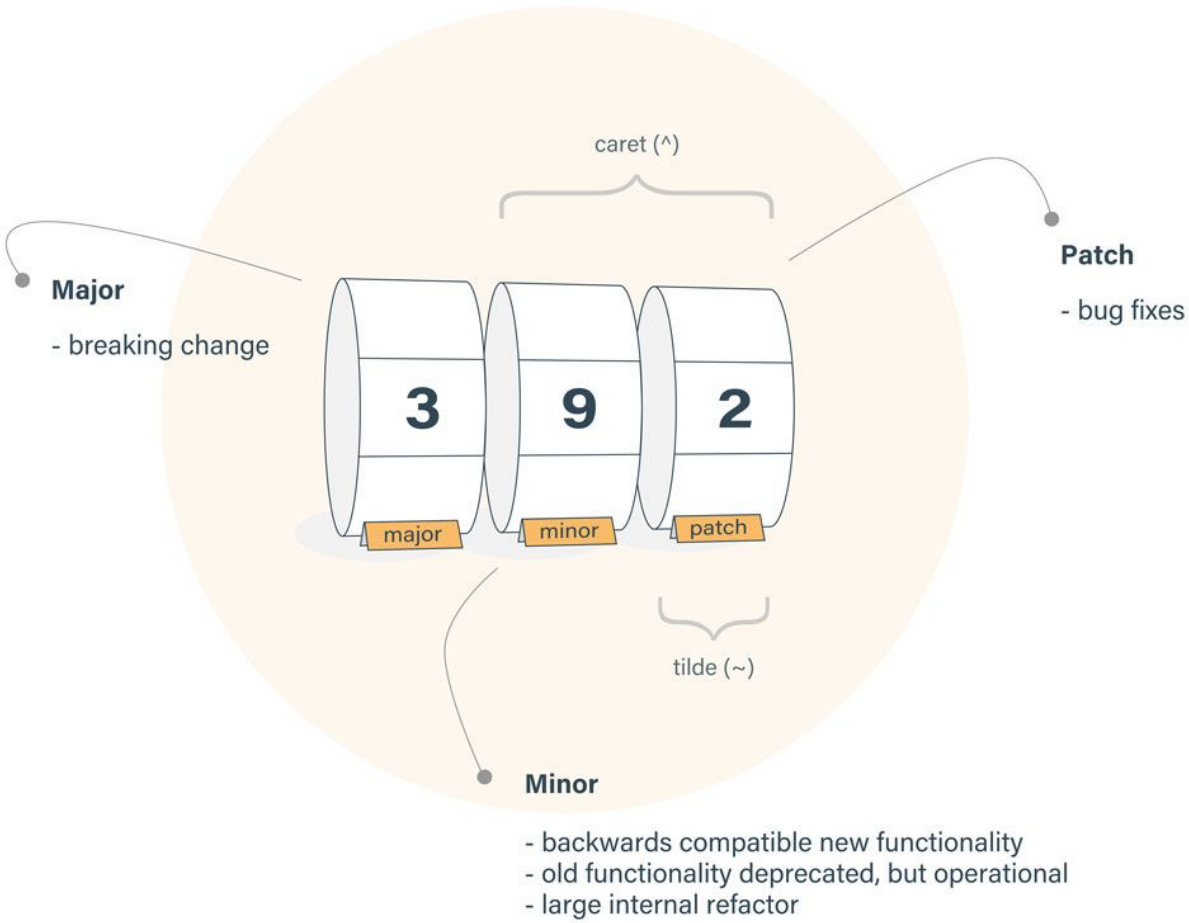
Technical lag
> Background

Semantic Versioning

Major	Minor	Bugfix
Incompatible changes	Added features	Squashing bugs

Examples: 0.0.1, 1.0.0, 1.2.3, 1.2.3-beta

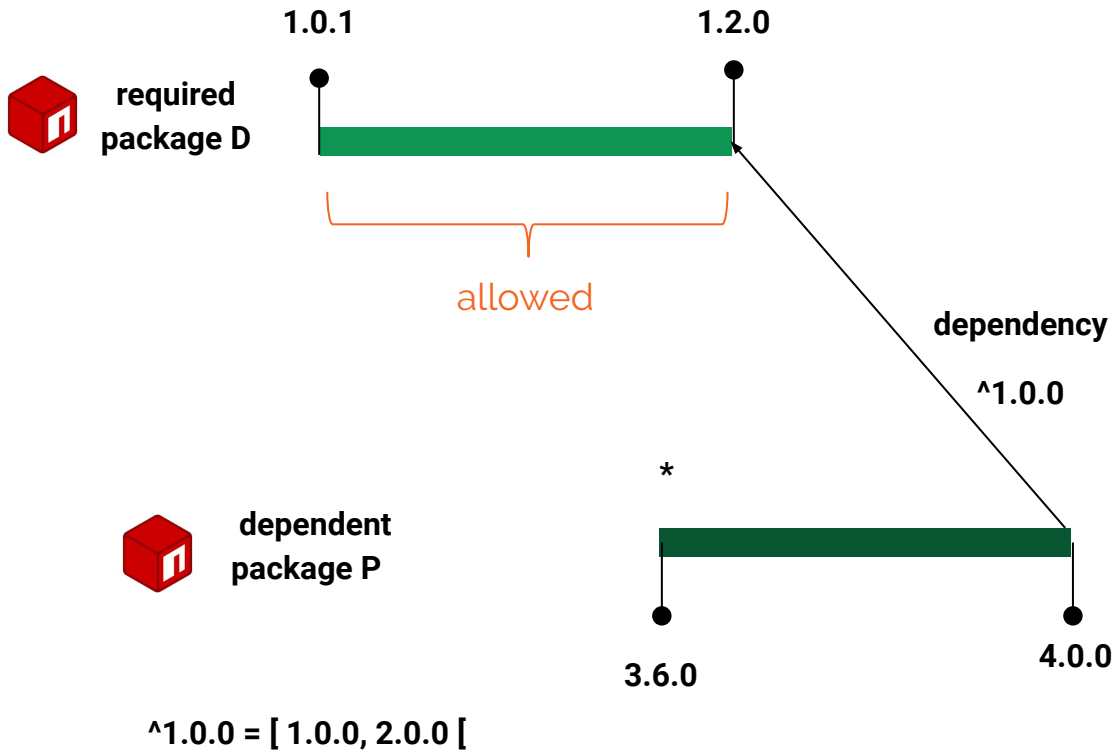
Technical lag
> Background



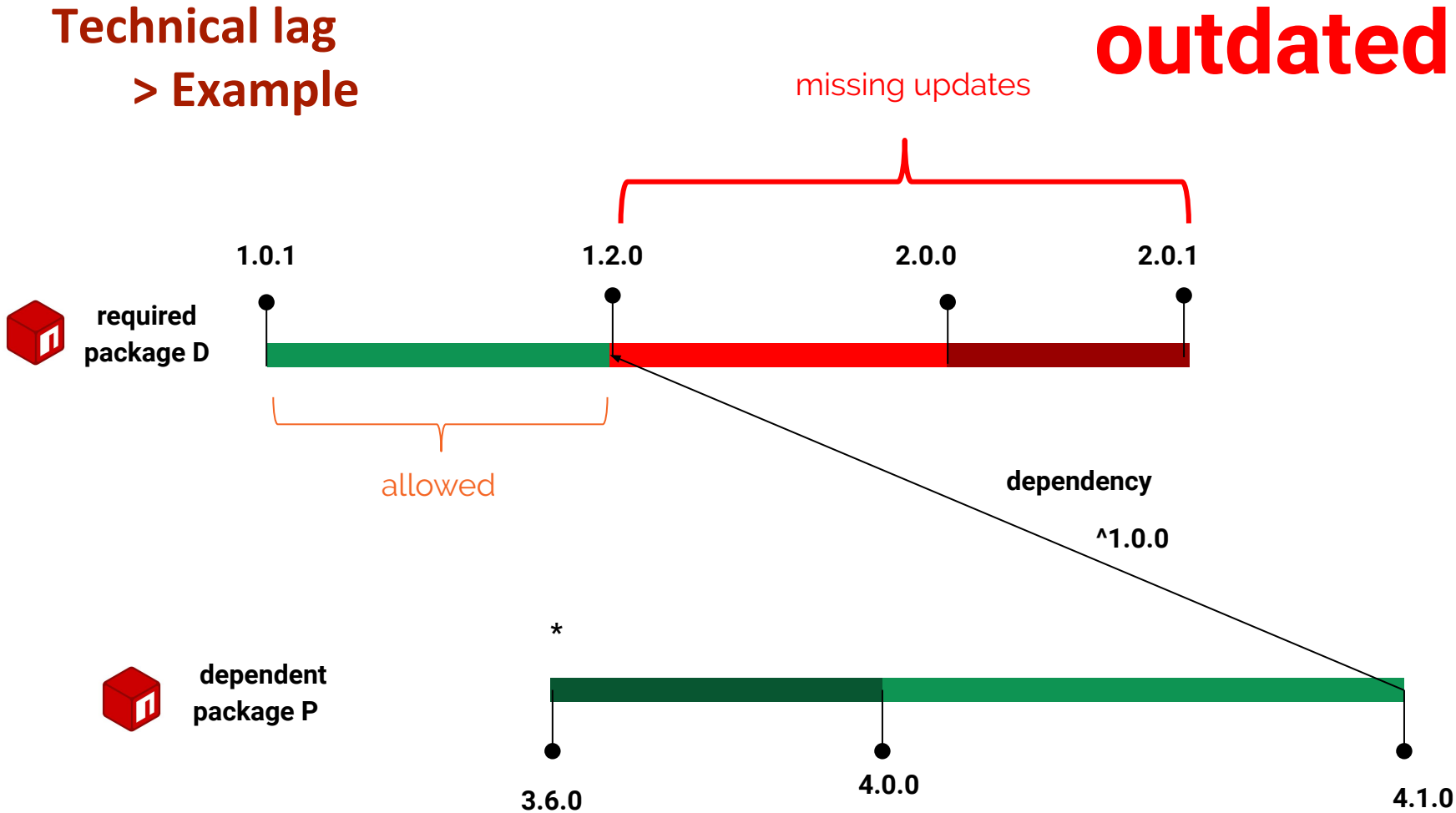
Other: *, ==1.2.3, >1.2.3, <1.2.3, 1.2.x, 1.x.x

Technical lag
> Example

up-to-date

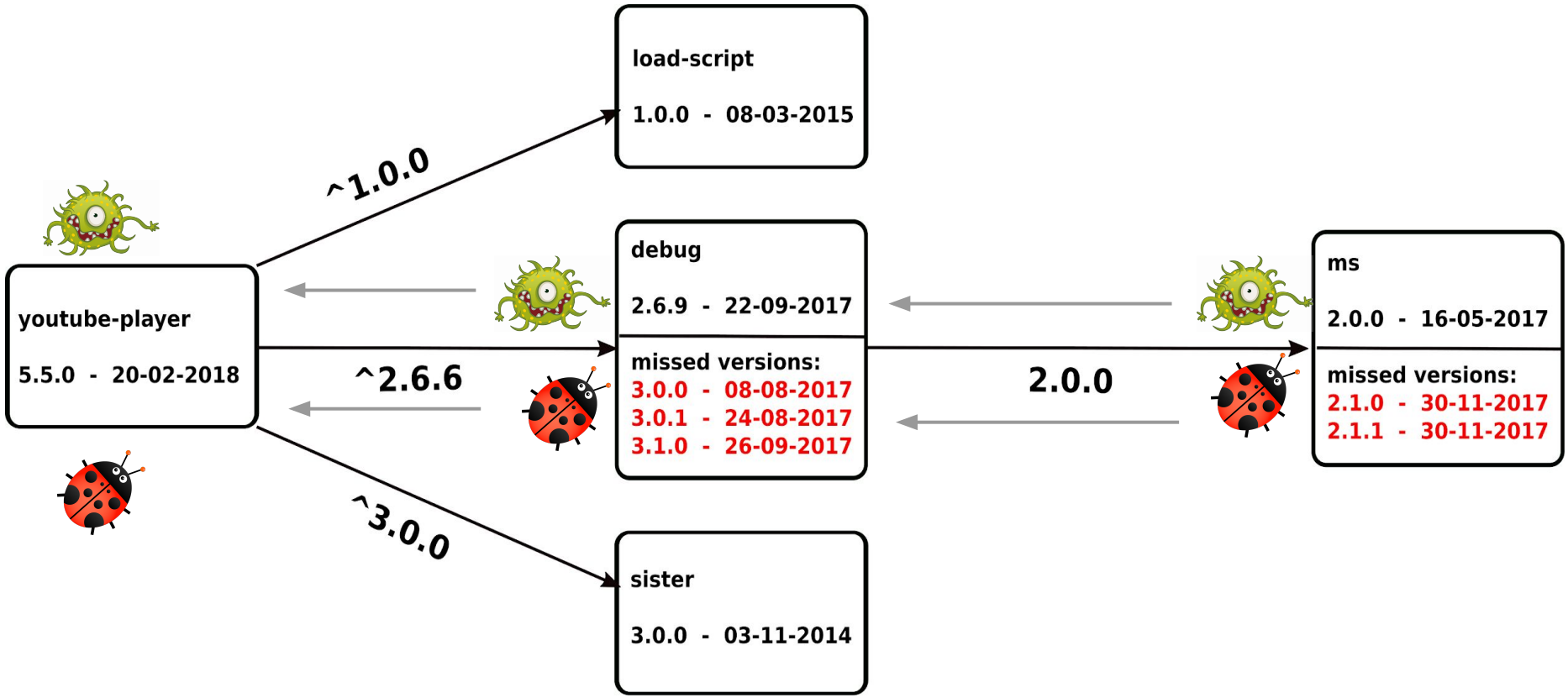


Technical lag
> Example

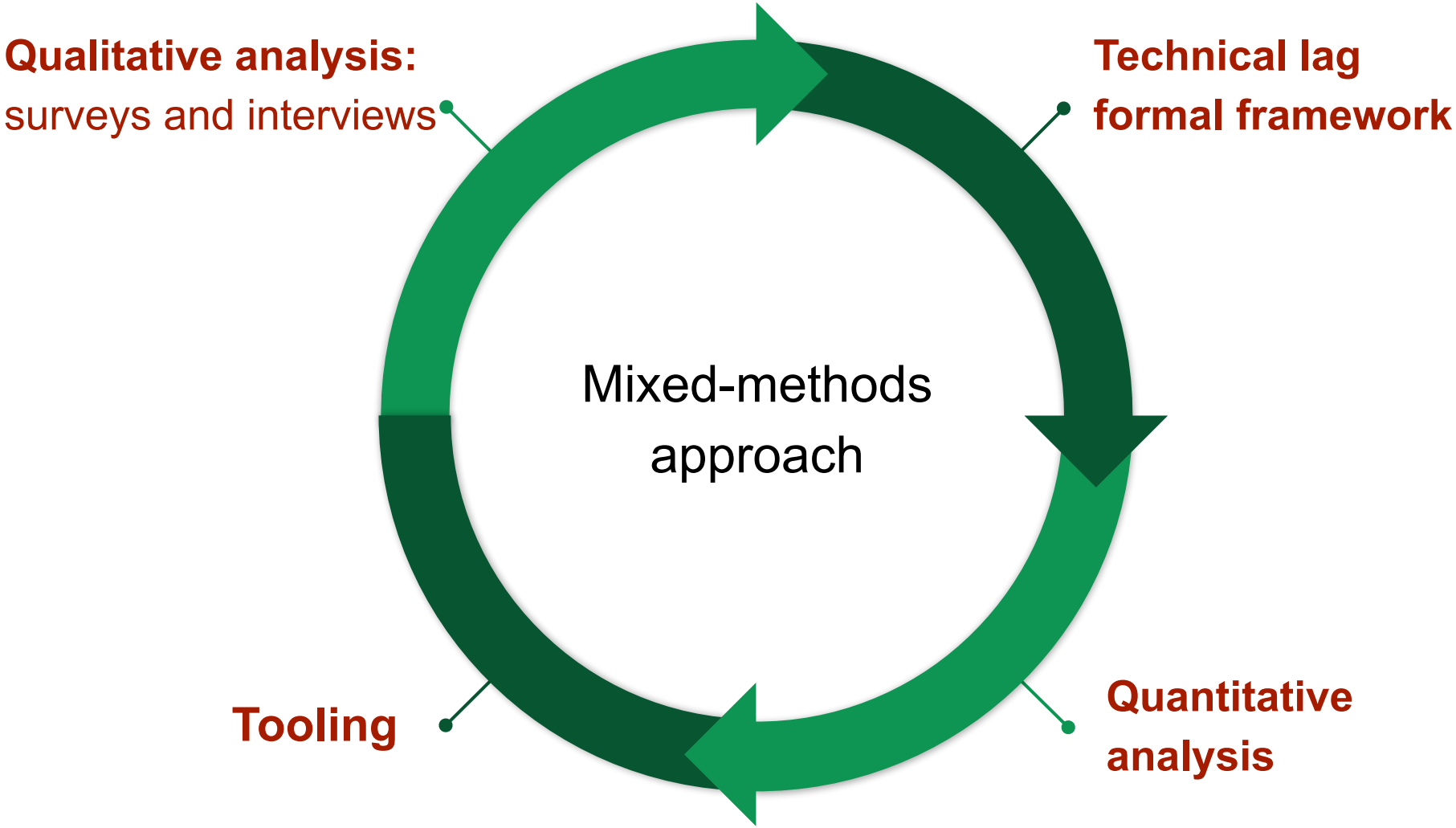


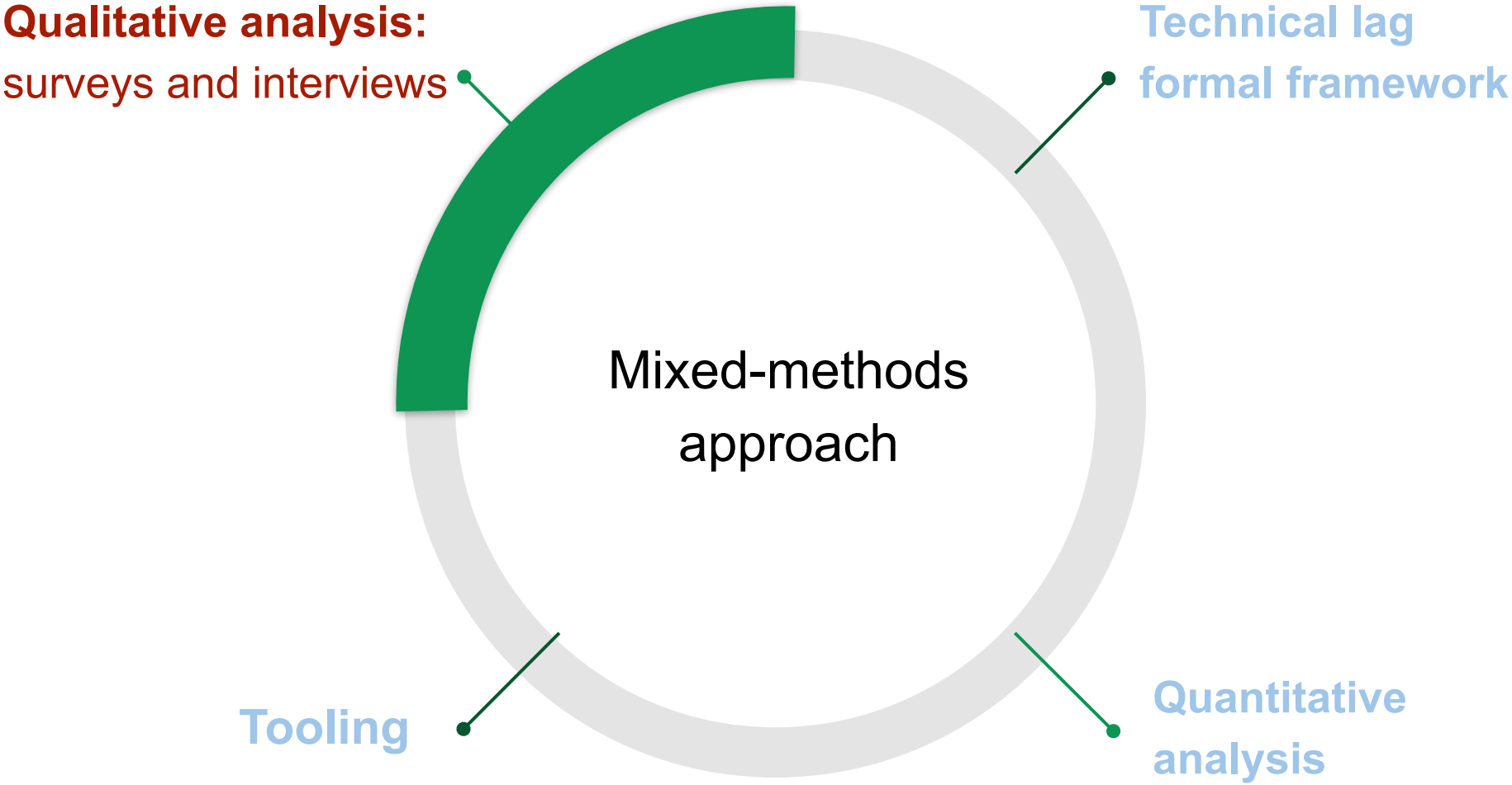
^1.0.0 = [1.0.0, 2.0.0 [

Technical lag
> Example



Followed research approach




Qualitative analysis:
surveys and interviews
Technical lag
formal framework
Mixed-methods
approach
Tooling
Quantitative
analysis

A Framework for Technical Lag
> Qualitative analysis

Semi-structured Interviews:

- 5 software practitioners
- Place: FOSDEM
- Highly educated interviewees with an average of 10 years of experience

Technical Lag is important, especially if we mix between the benefits of updating and the effort needed to do that.

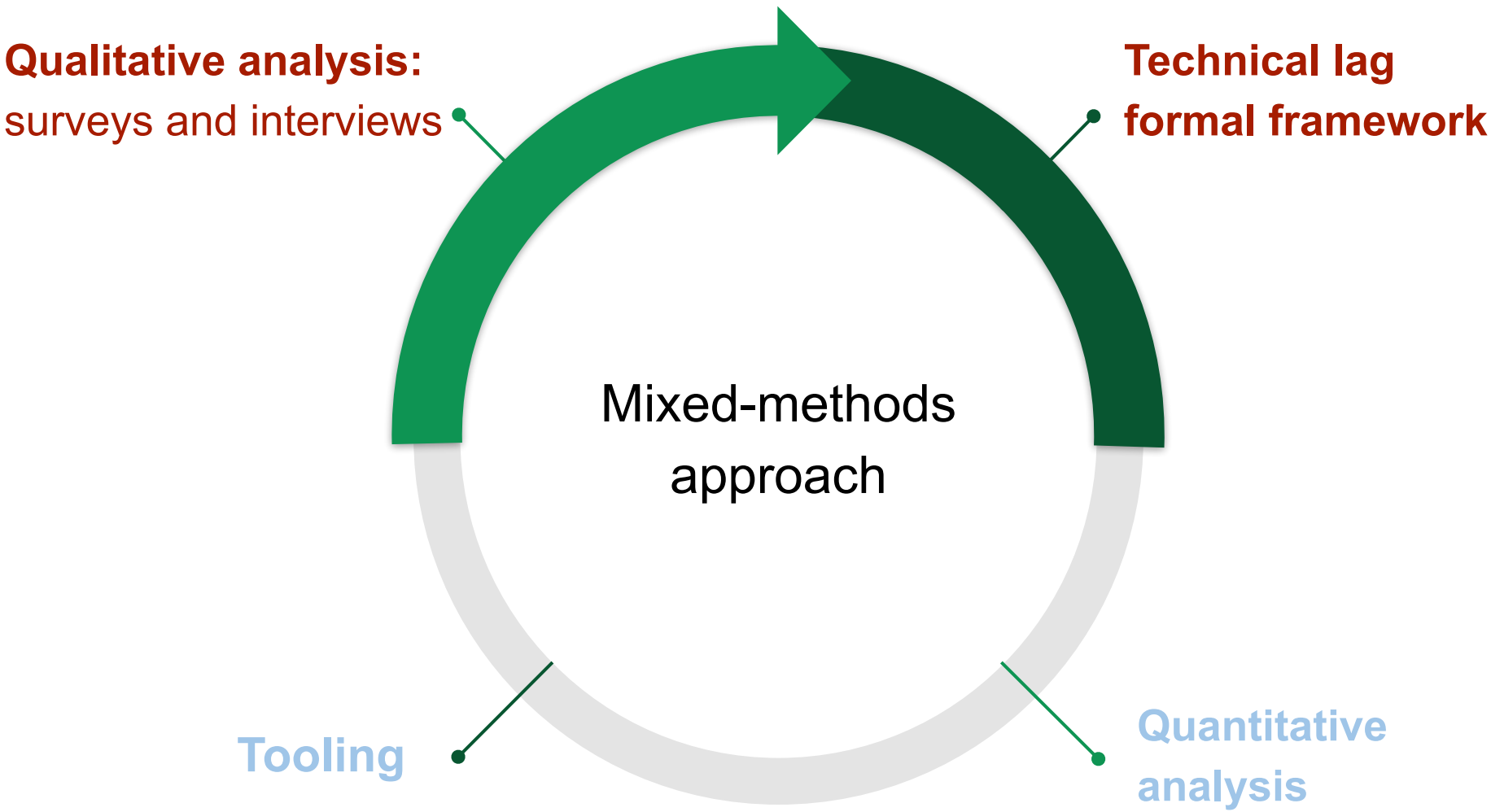
A Framework for Technical Lag
> Qualitative analysis

Online surveys:

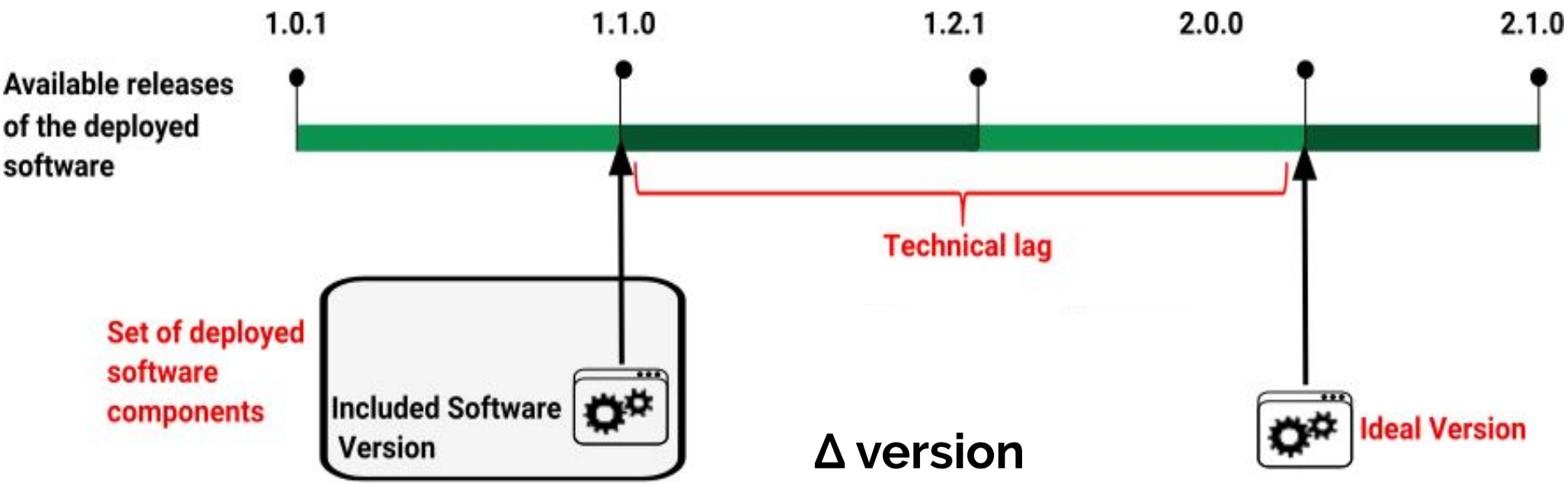
- ➢ 17 candidates (from facebook groups)
- ➢ Highly educated interviewees with an average of 3 years of experience

MCQ: What would be the most appropriate (ideal) version of a software library to use?

- ★ Most stable (14)
- ★ Latest available (9)
- ★ Most documented (7)
- ★ Most secure (5)


Qualitative analysis:
surveys and interviews
Technical lag
formal framework
Mixed-methods
approach
Tooling
Quantitative
analysis

Technical lag



Δ version
Δ time
Δ bugs
Δ vulnerabilities

A Framework for Technical Lag

$$\mathcal{F} = (\mathcal{C}, \mathcal{L}, ideal, delta, agg)$$

- $\mathcal{C}$ is a set of component releases
- $\mathcal{L}$ is a set of possible lag values
- ideal : $\mathcal{C} \rightarrow \mathcal{C}$ is a function returning the “ideal” component release
- delta : $\mathcal{C}$ x $\mathcal{C} \rightarrow \mathcal{L}$ is a function computing the difference between two component releases
- agg : $\mathbb{P}(\mathcal{L}) \rightarrow \mathcal{L}$ is a function aggregating the results of a set of lags

A Framework for Technical Lag

Given a technical lag framework $\mathcal{F}$, we define:

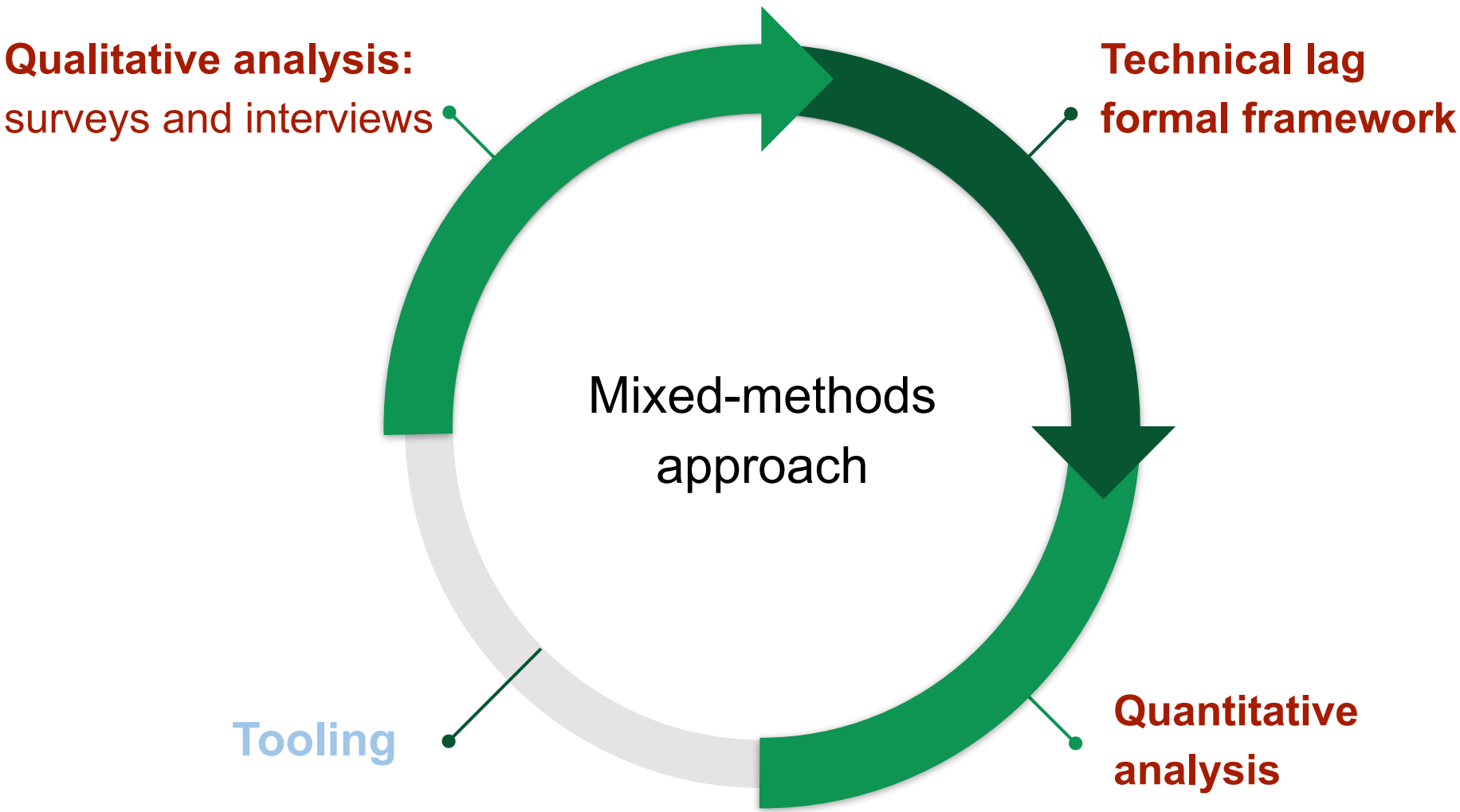
Technical lag

$$\mathbf{techlag}_{\mathcal{F}} : \mathcal{C} \rightarrow \mathcal{L} \text{ such that } \mathbf{techlag}_{\mathcal{F}}(c) = delta(c, ideal(c))$$

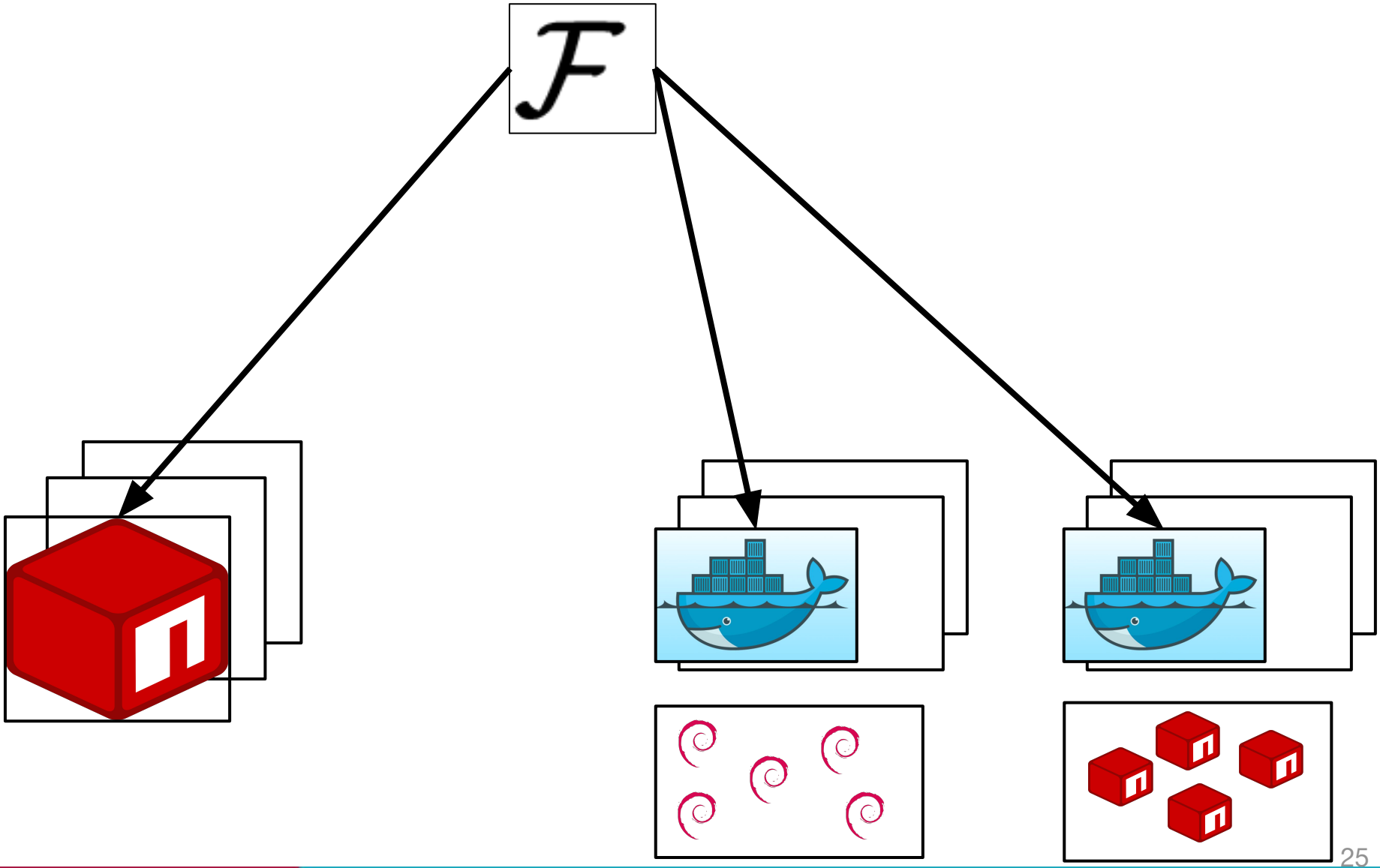
Aggregated Technical lag

Let $D \subseteq \mathcal{C}$, be a set of components, then:

$$\mathbf{agglag}_{\mathcal{F}} : \mathbb{P}(\mathcal{C}) \rightarrow \mathcal{L} \text{ such that } \mathbf{agglag}(\mathbf{D})_{\mathcal{F}} = agg(\{\mathbf{techlag}_{\mathcal{F}}(c) \mid \forall c \in D\})$$


Qualitative analysis:
surveys and interviews
Technical lag
formal framework
Mixed-methods
approach
Tooling
Quantitative
analysis

A Framework for Technical Lag
> Framework Instantiation



Technical Lag in npm Packages

Technical Lag in npm Packages

Time-based instantiation of $\mathcal{F}$:

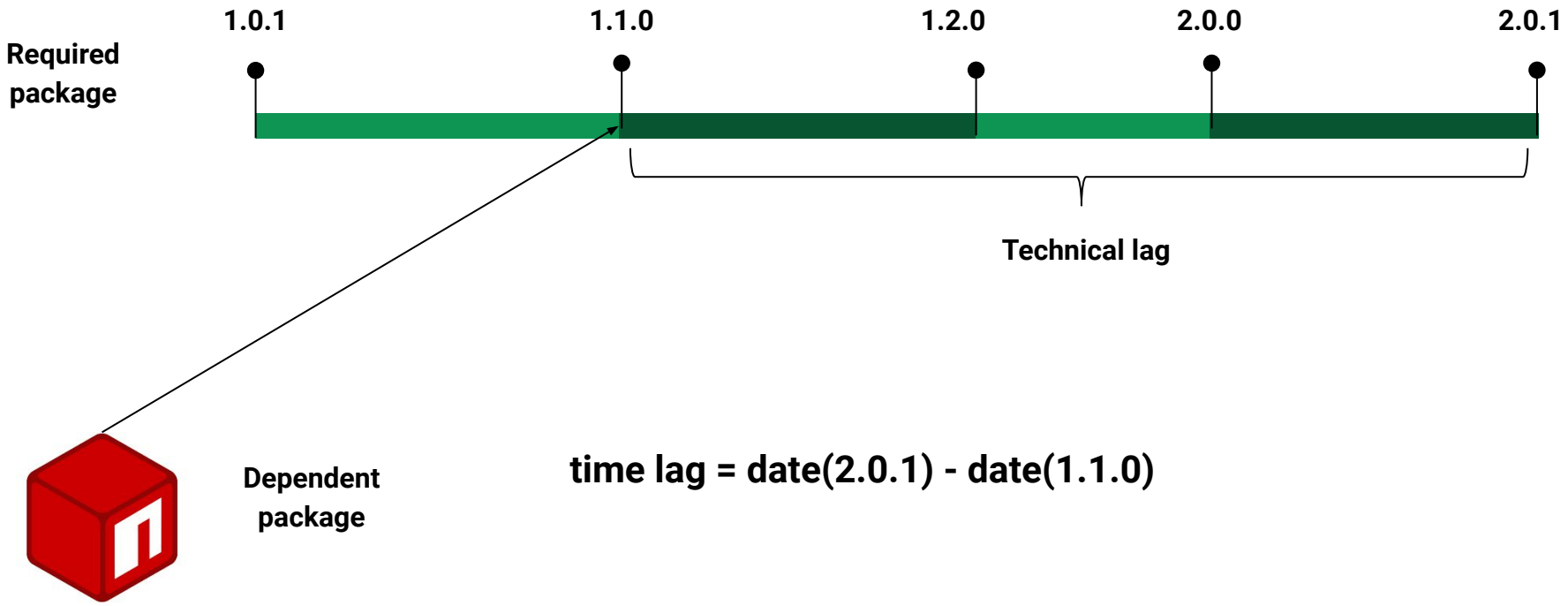
- **Ideal**: Highest available version
- **Delta**: **Time Lag = date(ideal) - date(used)**
- **Aggregation:** Max

Technical Lag in npm Packages

Version-based instantiation of $\mathcal{F}$:

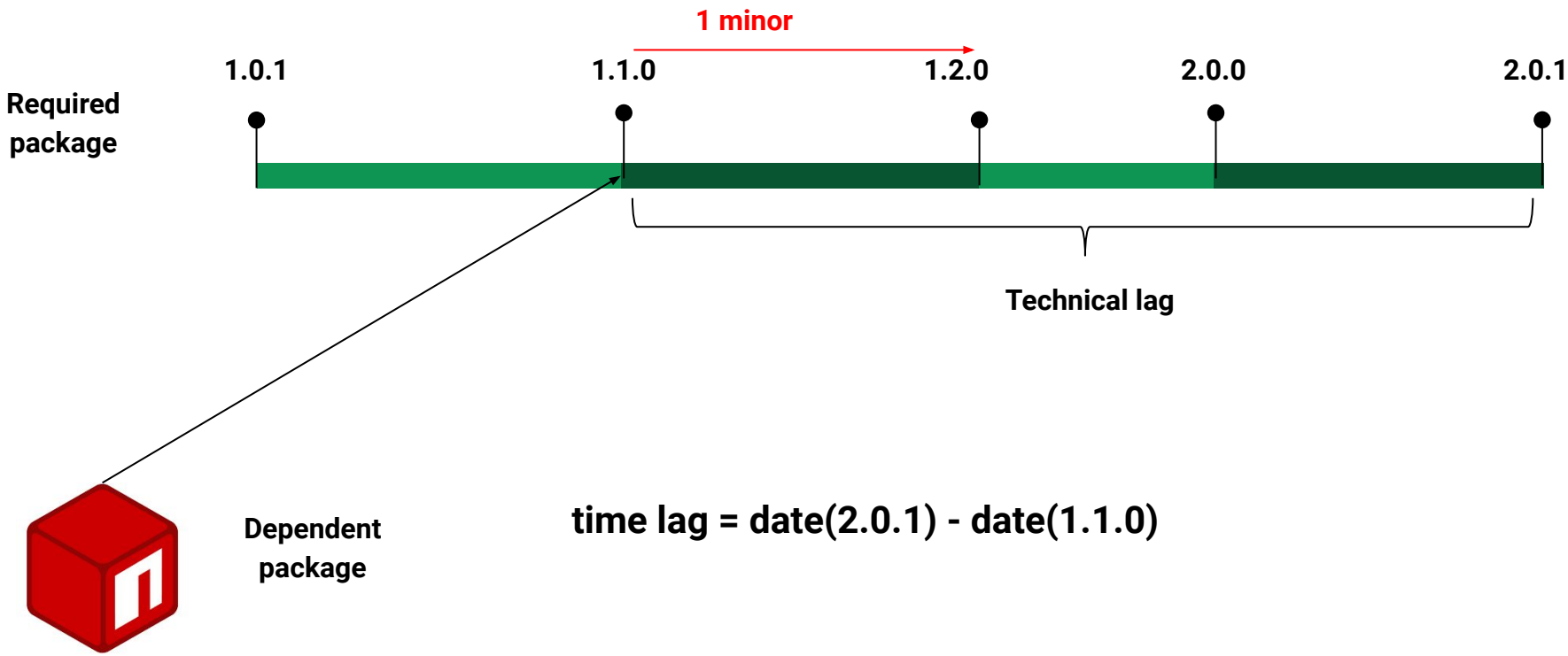
- **Ideal**: Highest available version
- **Delta**: **Version lag = (ΔMajor, ΔMinor, ΔPatch)**
- **Aggregation:** Sum

Technical Lag in npm Packages
> Time-based instantiation



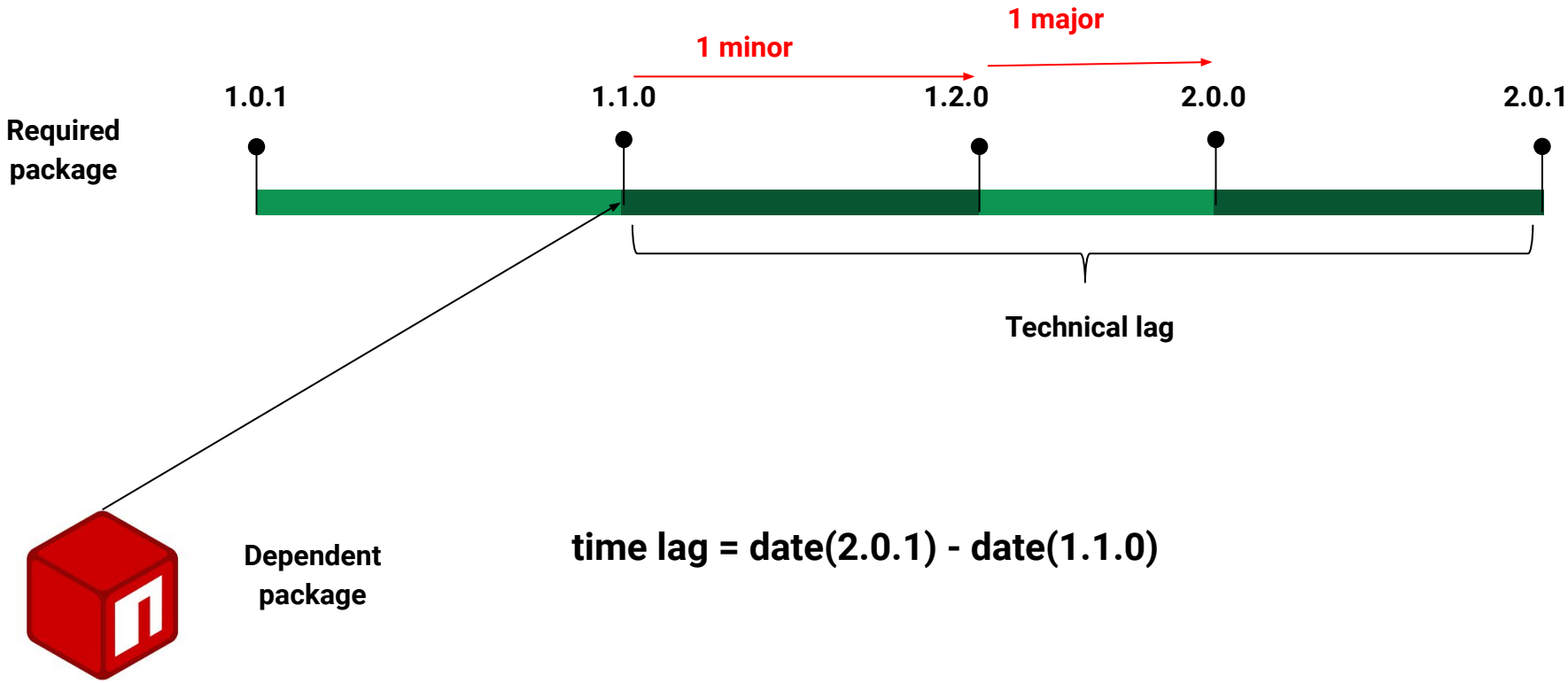
time lag = date(2.0.1) - date(1.1.0)

Technical Lag in npm Packages
> Version-based instantiation



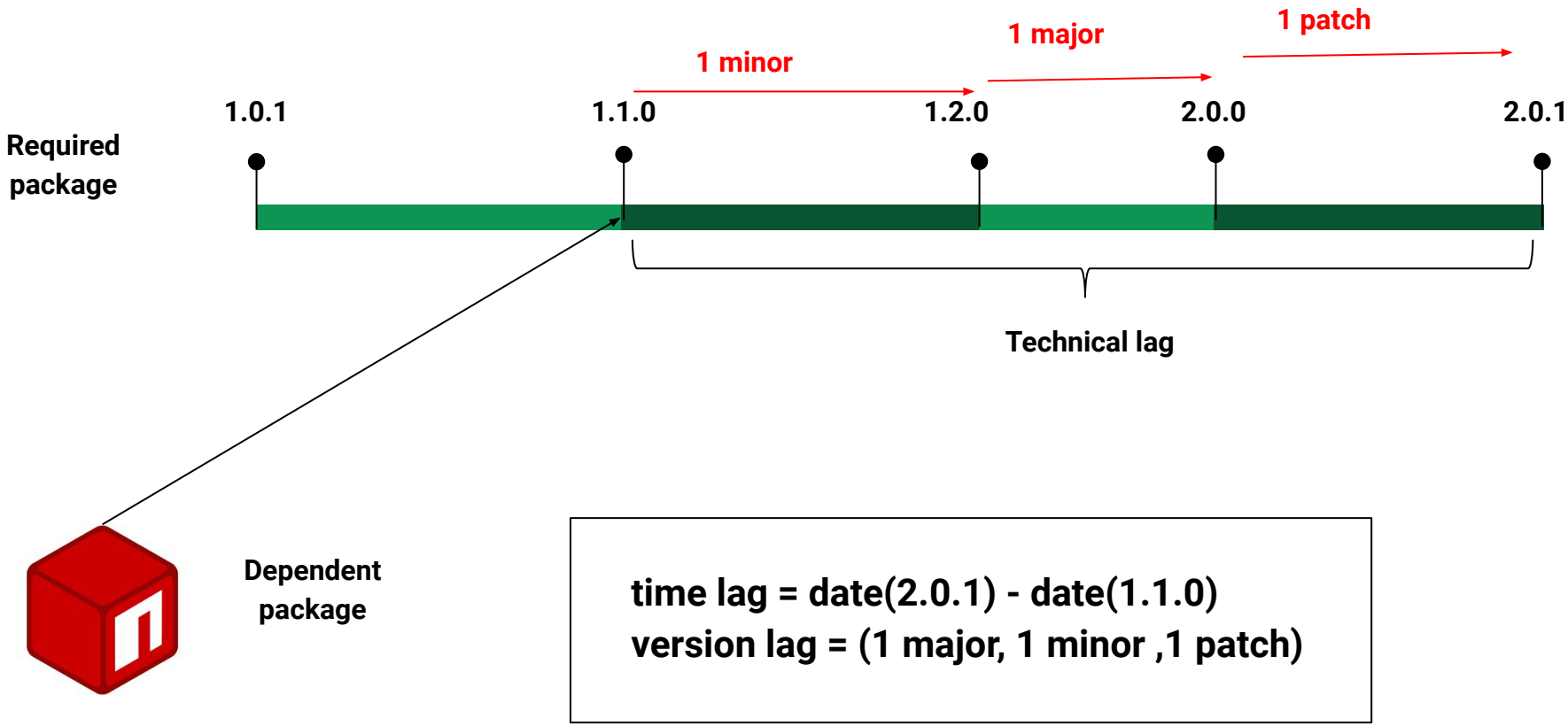
time lag = date(2.0.1) - date(1.1.0)

Technical Lag in npm Packages
> Version-based instantiation



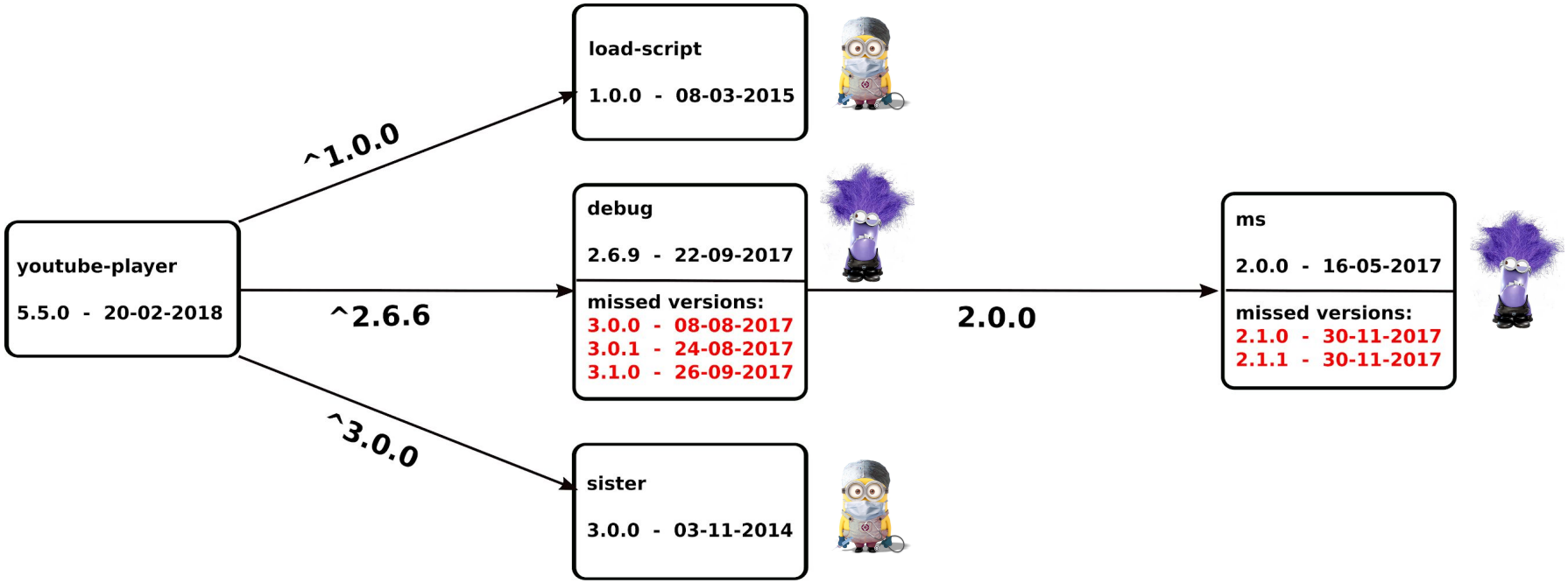
time lag = date(2.0.1) - date(1.1.0)

Technical Lag in npm Packages
> Version-based instantiation



time lag = date(2.0.1) - date(1.1.0)
version lag = (1 major, 1 minor ,1 patch)

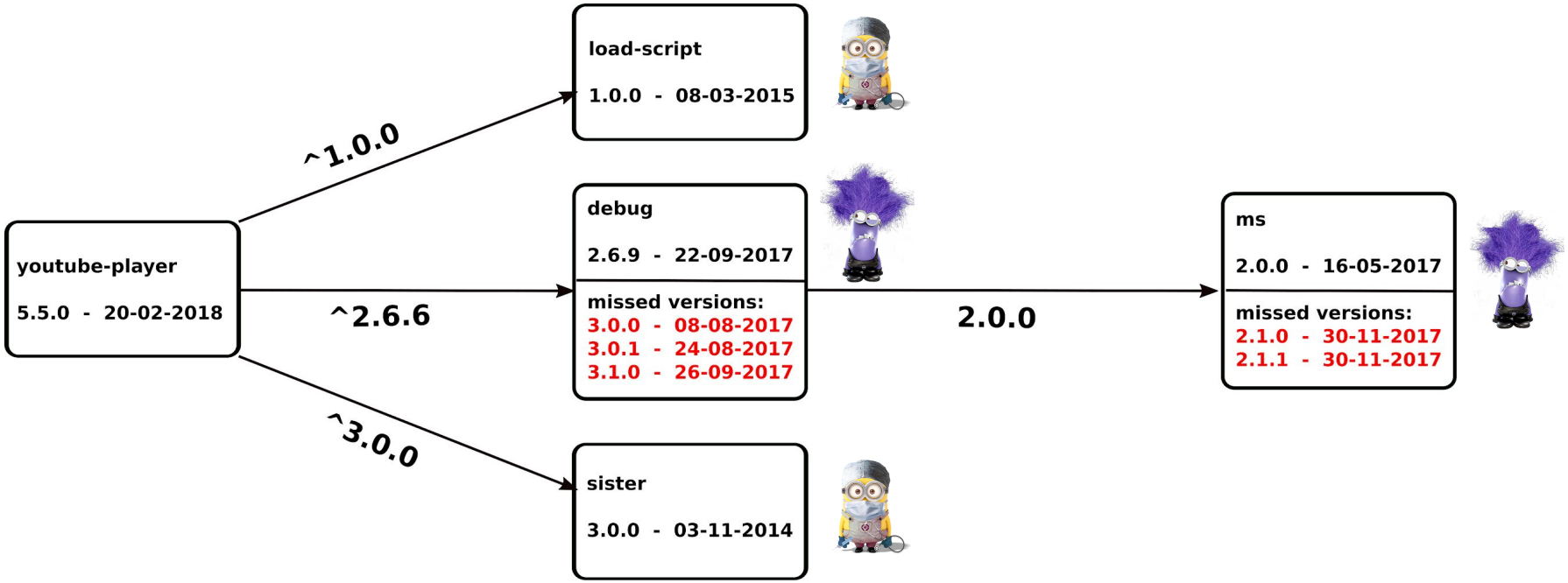
Technical Lag in npm Packages
> Example



Time-based technical lag for the package *debug*:

- *ideal* (2.6.9) = 3.1.0
- time_lag(2.6.9) = 26-09-2017 - 22-09-2017 = 4 days
- version_lag(2.6.9) = (1,1,1)

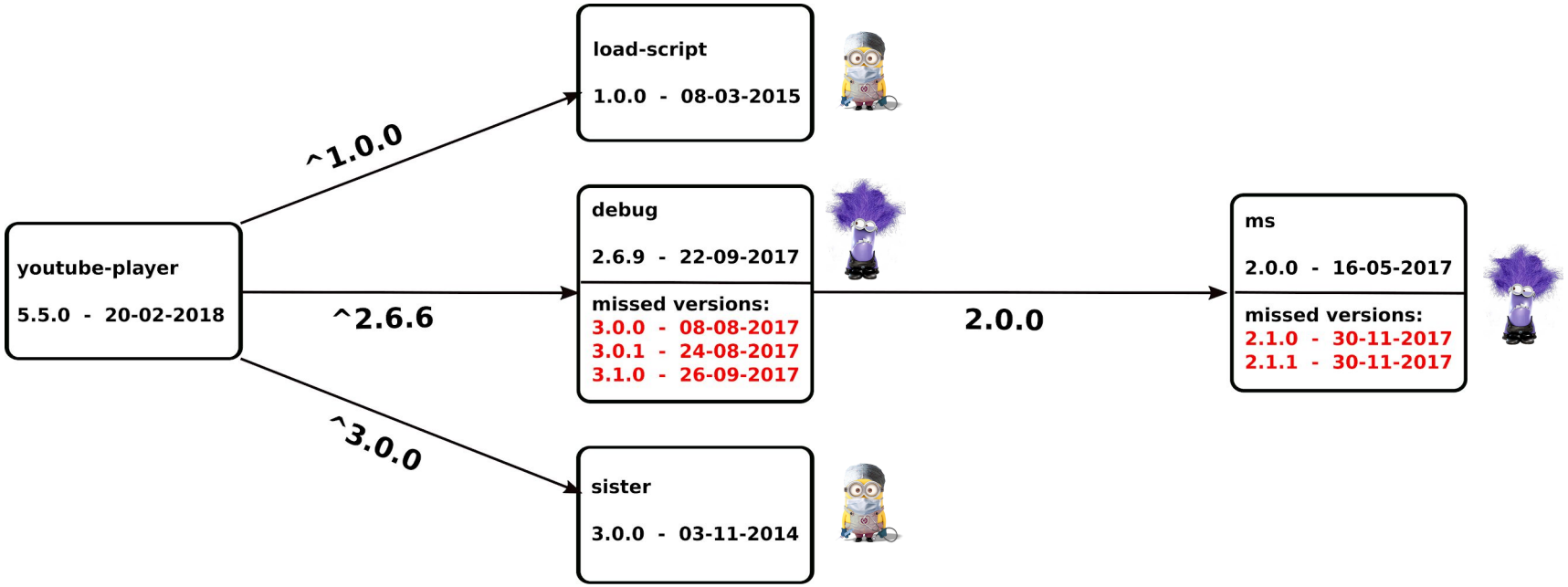
Technical Lag in npm Packages
> Example



Time-based technical lag for the package *ms*:

- *time_lag*(2.0.0) = 198 days

Technical Lag in npm Packages
> Example



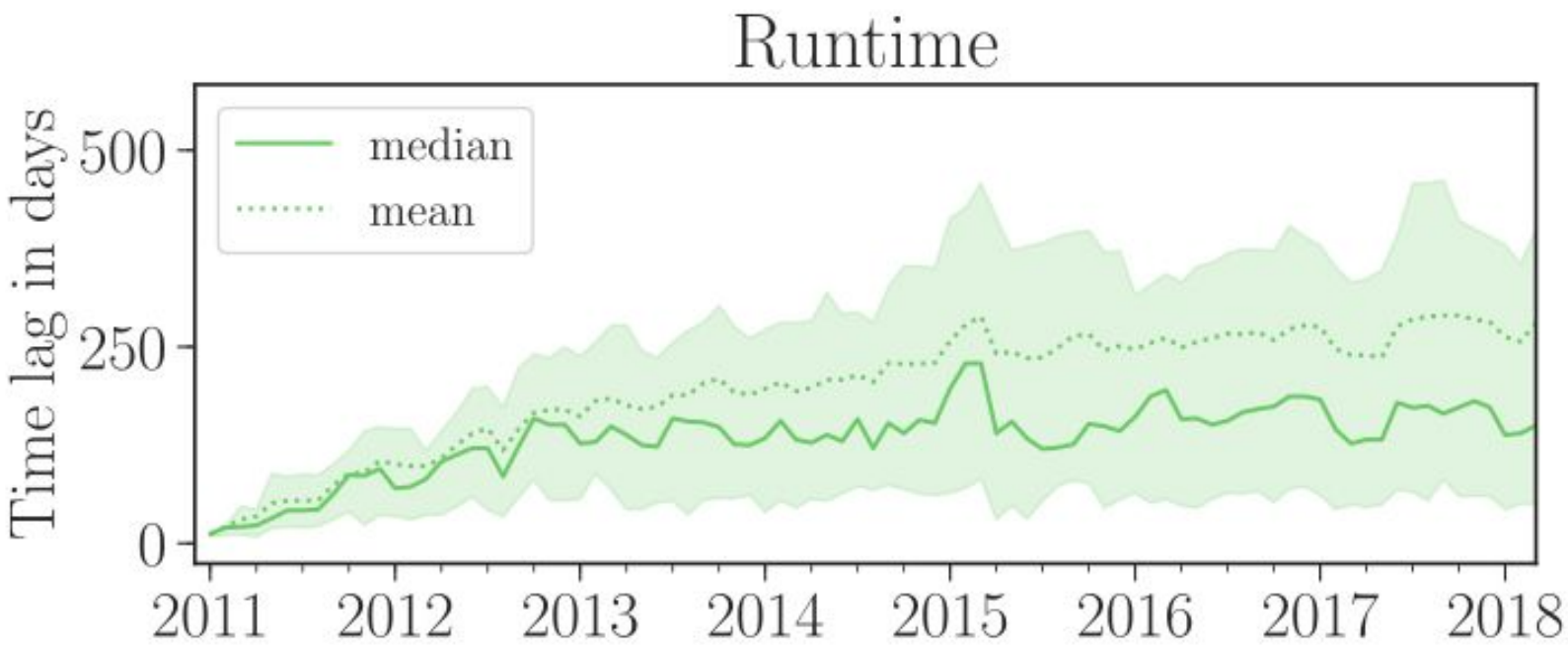
Aggregated Time-based technical lag for the package release youtube-player@5.5.0:

- *time_lag*(debug@2.6.9) = **4 days**
- *time_lag*(ms@2.0.0) = **198 days**

➔ *agglag*({debug@2.6.9, ms@2.0.0}) = *max* (4 days, 198 days) **= 198 days**

Technical Lag in npm Packages

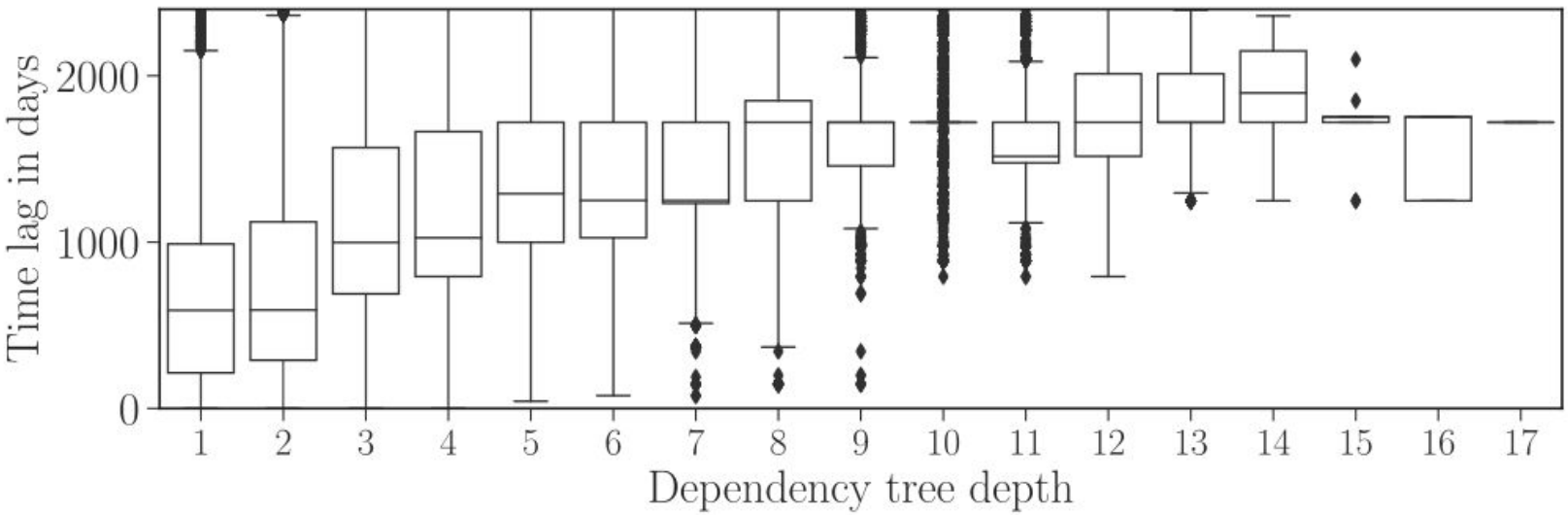
Time lag induced by direct dependencies in npm package releases:



New package releases have an increased technical lag.
Technical lag is induced by version constraints

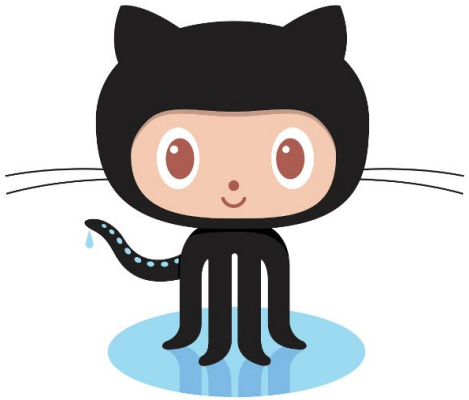
Technical Lag in npm Packages

Time lag induced by transitive runtime dependencies in npm package releases:



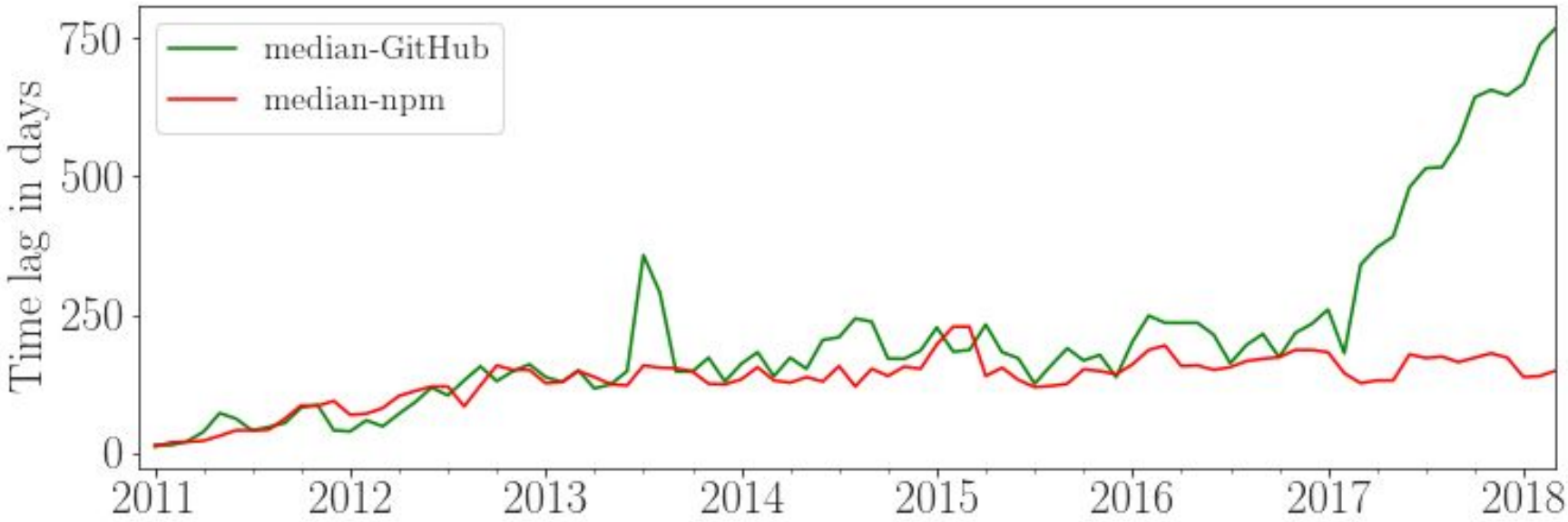
Technical lag is accumulated from a level to another in the dependency tree.

Technical Lag in npm applications



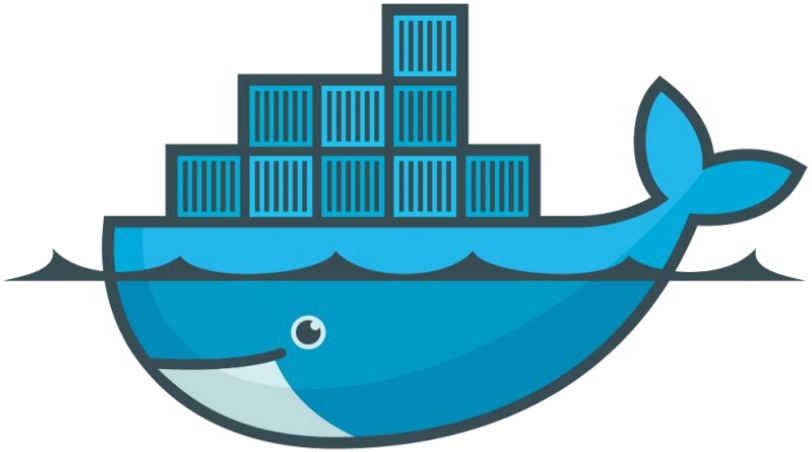
Technical Lag in npm applications

Time lag induced by direct dependencies in GitHub applications:



Technical lag in GitHub applications is higher than in npm package releases

Docker Containers

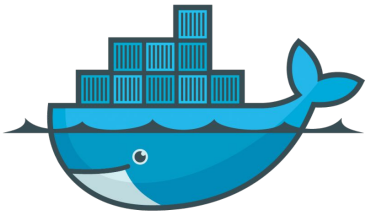


docker

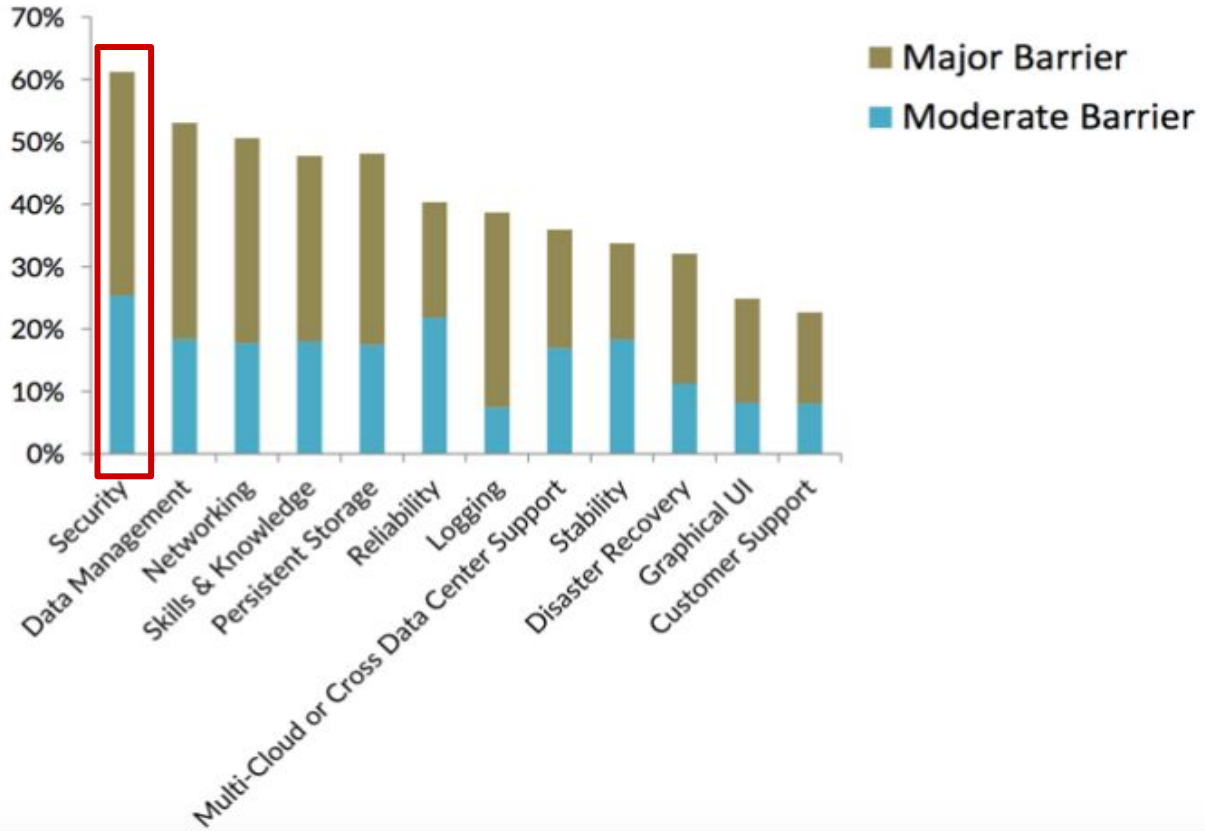
Technical Lag in Docker Containers > Background

- *Containers* are isolated bundles of software packages
- *Docker* is one of the main tools for containerisation
- DockerHub is the largest repository for container images

Technical Lag in Docker Containers
> Background



What are the biggest barriers to putting containers in a production environment? - ClusterHQ



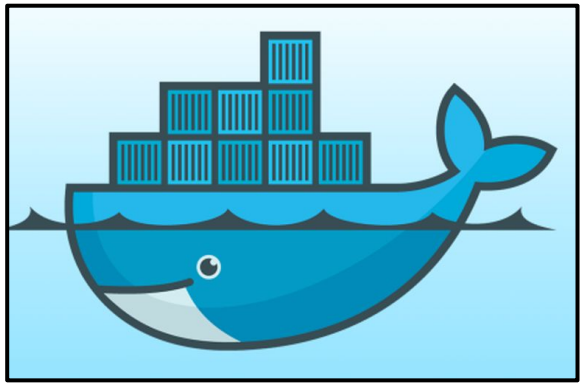
Technical Lag in Docker Containers > Background

#11

Six months ago our surveys showed that users who considered image security focused on simple CVE scanning on the operating system. In this latest survey we are encouraged to see more focus on the other artifacts within the image, many of which are not covered by traditional scanning solutions.

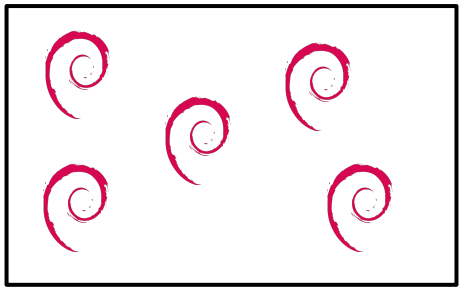
anchore

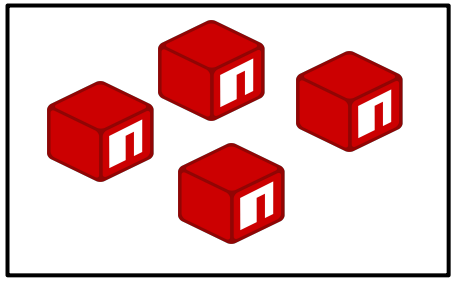
Technical Lag in Docker Containers
> Focus



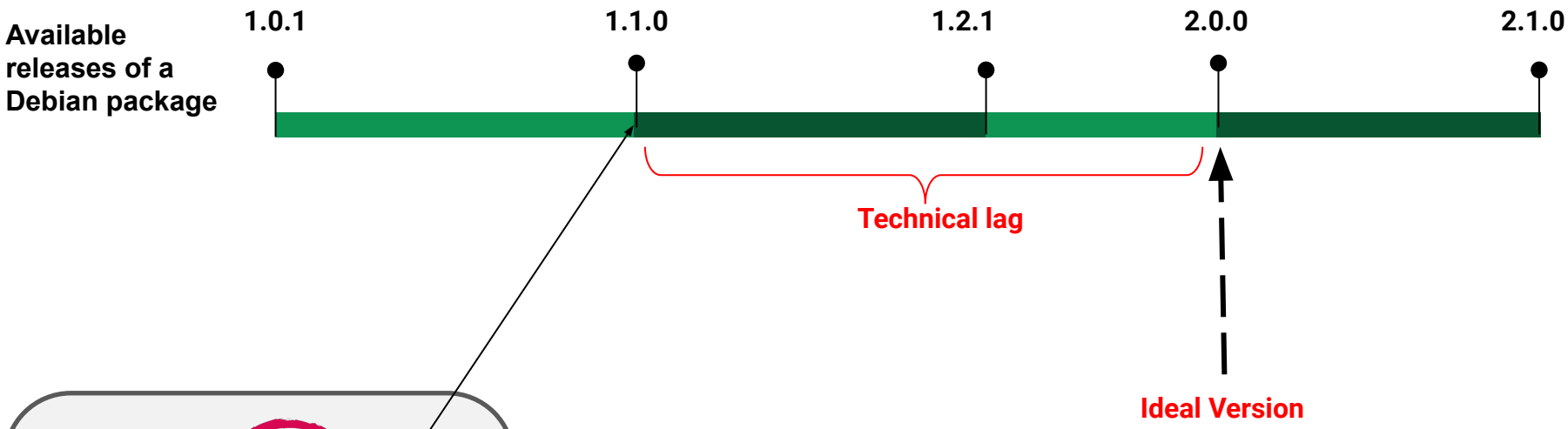
debian

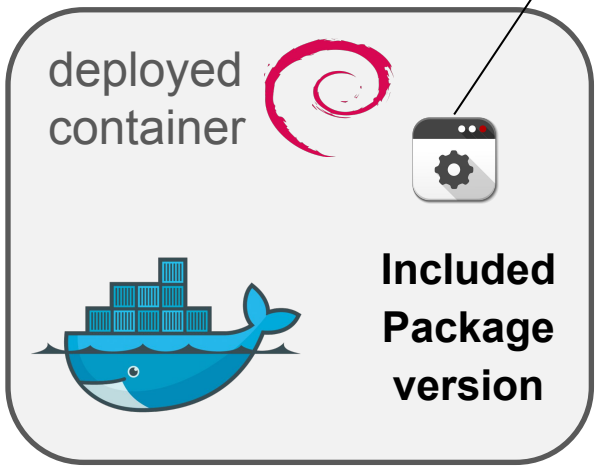
npm





Technical Lag in Docker Containers
> Background

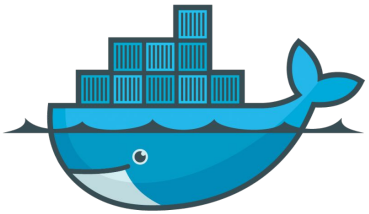




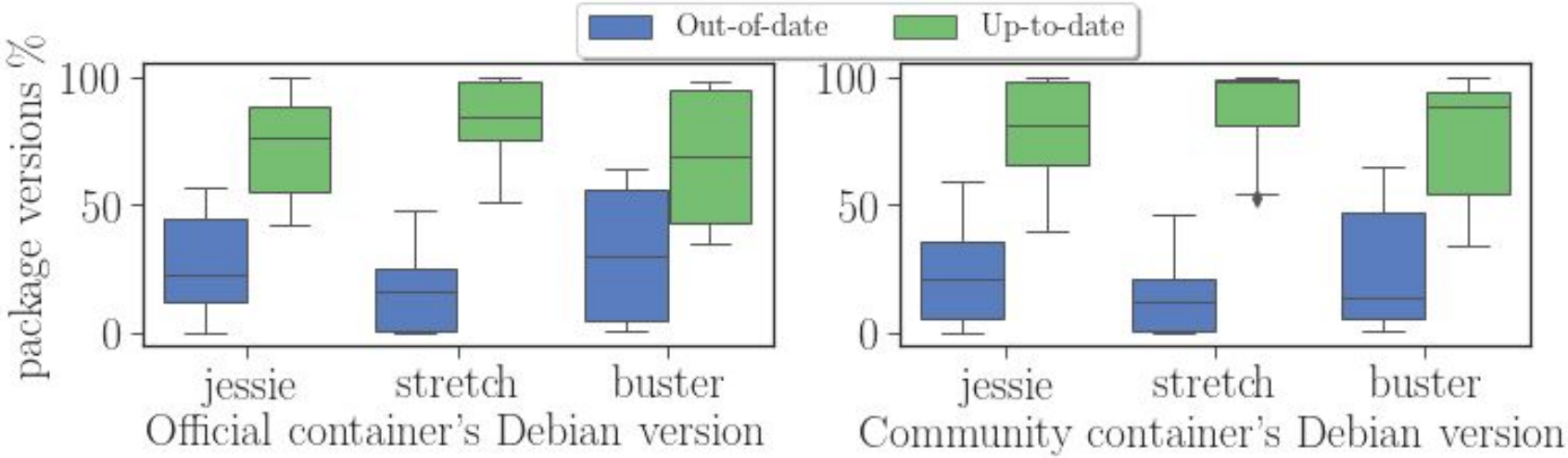
technical lag =

- Δ Versions (freshness)
- Δ Vulnerabilities (security)
- Δ Bugs (stability)

Technical Lag in Docker Containers
> Software Freshness



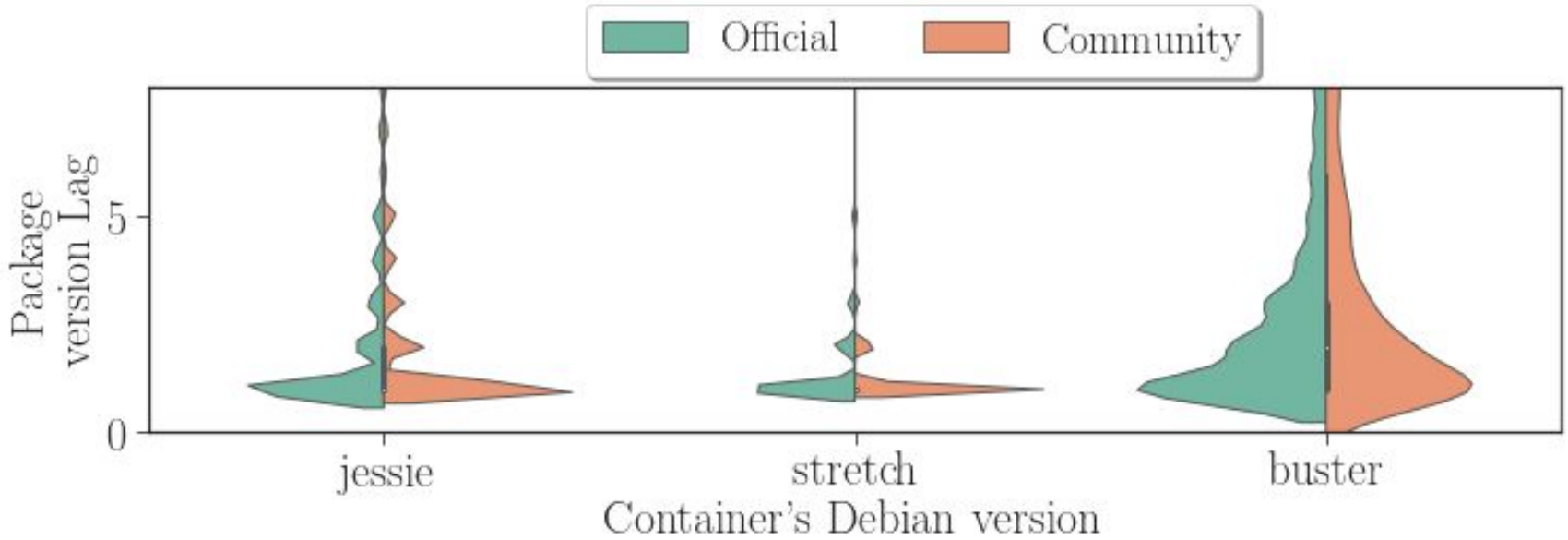
How outdated are images?
IDEAL = LATEST



The majority of packages in Debian containers is up-to-date...
... but most of the images contain outdated packages.

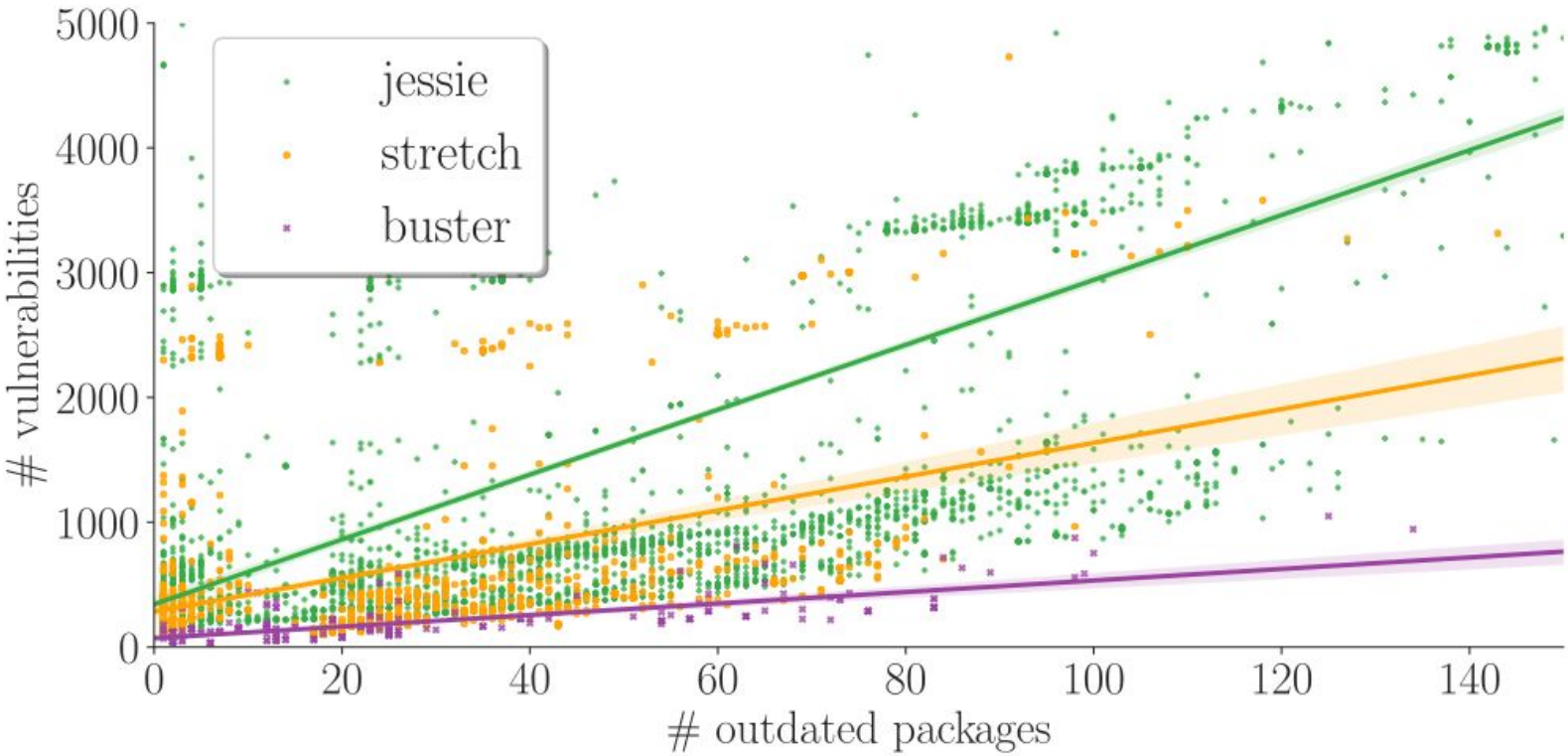
Technical Lag in Docker Containers
> Software Freshness

What is the version lag induced by the used Debian package releases?
IDEAL = LATEST



Outdated Debian packages in Docker containers induce a median version lag of 1 version.

Technical Lag in Docker Containers
> Software Security

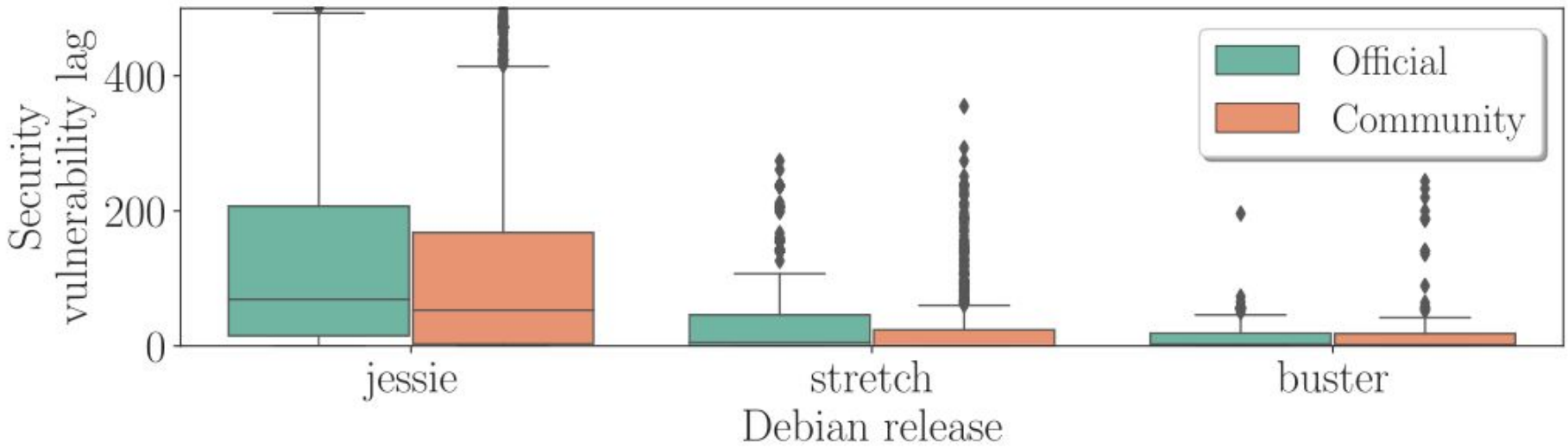


The number of vulnerabilities depends on the Debian release, and is moderately correlated with the number of outdated packages in a container.

Technical Lag in Docker Containers
> Software Security

Can we reduce security lag in DockerHub container images?

IDEAL = Most Secure

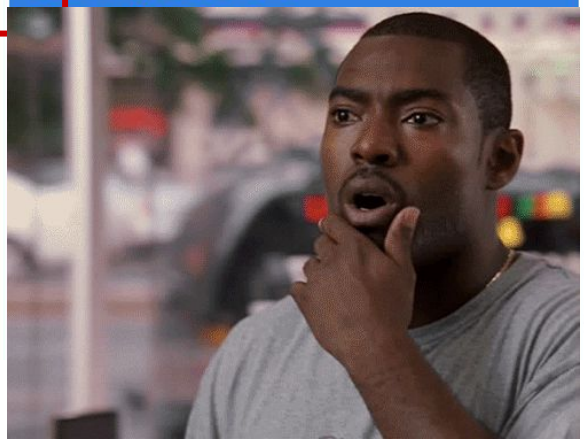


Few packages (2.5%) need to be downgraded in order to have the most secure version.

Technical Lag in Docker Containers
> Survey

#11

Six months ago our surveys showed that users who considered image security focused on simple CVE scanning on the operating system. In this latest survey we are encouraged to see more focus on the other artifacts within the image, many of which are not covered by traditional scanning solutions.



Technical Lag in Docker Containers

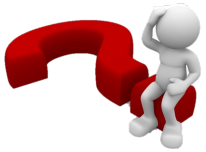
Security vulnerabilities in npm JavaScript

"37% of websites include a JavaScript library with a known open source vulnerability"

T. Lauinger et al. "*Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web*", NDSS 2017.

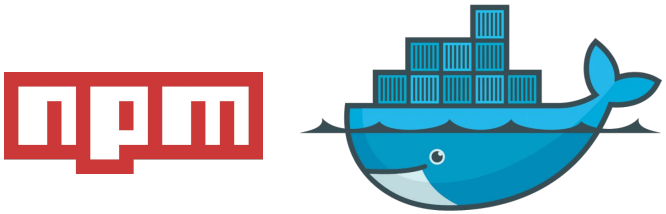
1 out of 3 depending on a vulnerable npm package never update their dependency and remain vulnerable

A. Decan et al. "*On the impact of security vulnerabilities in the npm package dependency network*", MSR 2018.

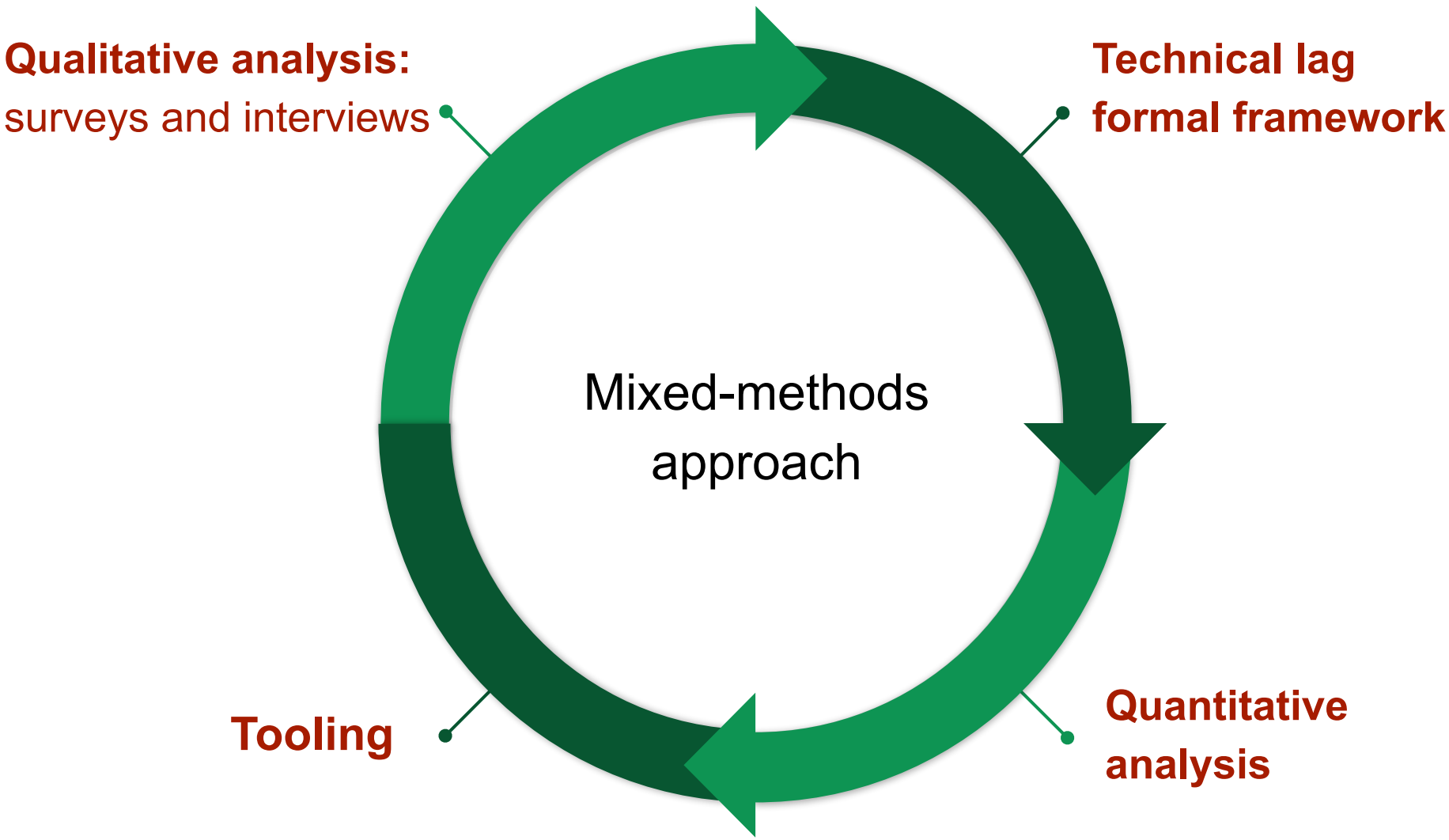


So what about Docker containers having npm packages?

Technical Lag in Docker Containers



- ★ Old Node images might be missing many updates, including one major update.
- ★ All official Node-based images have vulnerable npm packages, with an average of 16 security vulnerabilities per image.
- ★ Older images are more likely to have more vulnerabilities.


Qualitative analysis:
surveys and interviews
Technical lag
formal framework
Mixed-methods
approach
Tooling
Quantitative
analysis

ConPan: A tool to analyze packages in software containers
> Existing tools

Image Analysis | Policy Bundles | Events | Configuration

Image Analysis ▸ docker.io ▸ dnurmi/myapplication ▸ latest ▸ sha256:ccd1b4c1e042616e91e0ba44b6d69744287ed

Overview | Compliance | Vulnerabilities | Contents | Changelog | Build Summary

Compliance — Policy Bundle: Tuned Security — Final Action: STOP

Total: 388 — Stop, Warn — 99.7%

Vulnerabilities

Total: 387 — High, Medium, Low — 45.0%, 54.0%

Print Report

Image Data

Repository / Tag: dnurmi/myapplication:latest — This is the most recent tagged image

Image ID:

Image Digest: sha256:ccd1b4c1e042616e91e0ba44b6d69744287ed...

Image Detected:

Image Scanned: 2 days ago on Thu, 02 Aug 2018 at 15:01:37 PDT

Operating System:

QUAY | Repositories | Tutorial | Docs | Blog | Quay User

exmaple/nginx — 99009dfc5e95

Quay Security Scanner has detected **55** vulnerabilities.

Patches are available for **15** vulnerabilities.

- **9** High-level vulnerabilities.
- **9** Medium-level vulnerabilities.
- **7** Low-level vulnerabilities.
- **22** Negligible-level vulnerabilities.
- **8** Unknown-level vulnerabilities.

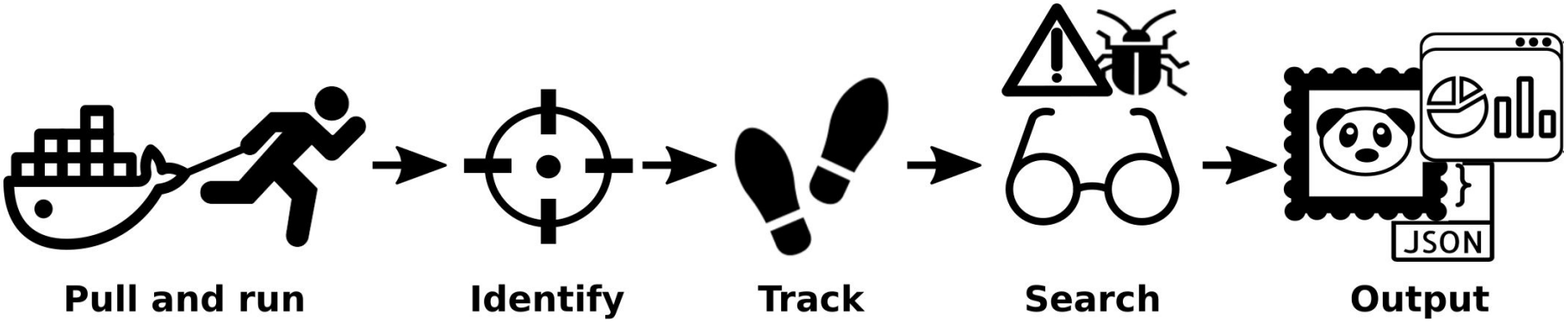
15% 16% 16% 13% 40%

Image Vulnerabilities — Filter Vulnerabilities... — Only show fixable

CVE	SEVERITY	PACKAGE	CURRENT VERSION	FIXED IN VERSION	INTRODUCED IN IMAGE
CVE-2016-2108	10 / 10	openssl	1.0.1k-3+deb8u4	1.0.1k-3+deb8u5	RUN apt-key adv --keyserver hkp...
CVE-2016-3191	High	pcre3	2:8.35-3.3+deb8u2	2:8.35-3.3+deb8u3	ADD file:b5391cb13172fb513dbfca...

ConPan: A tool to analyze packages in software containers
> Overview

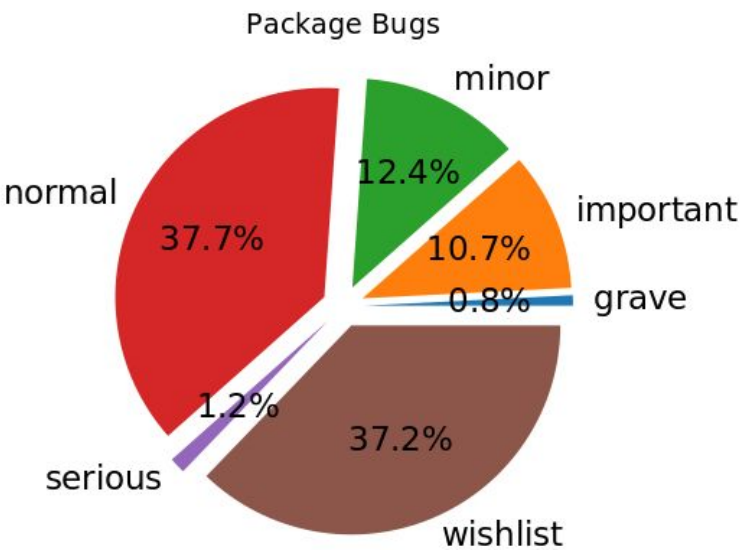
ConPan(*): "**CON**tainer **P**ackages **AN**alyzer"

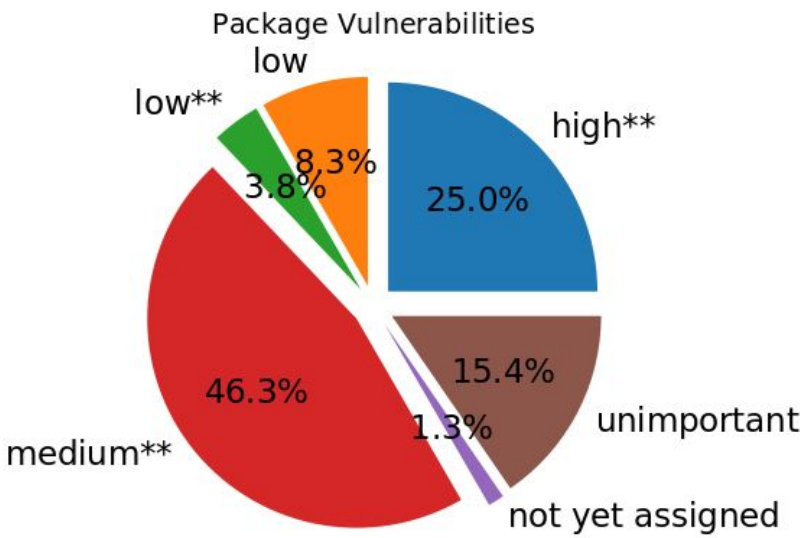


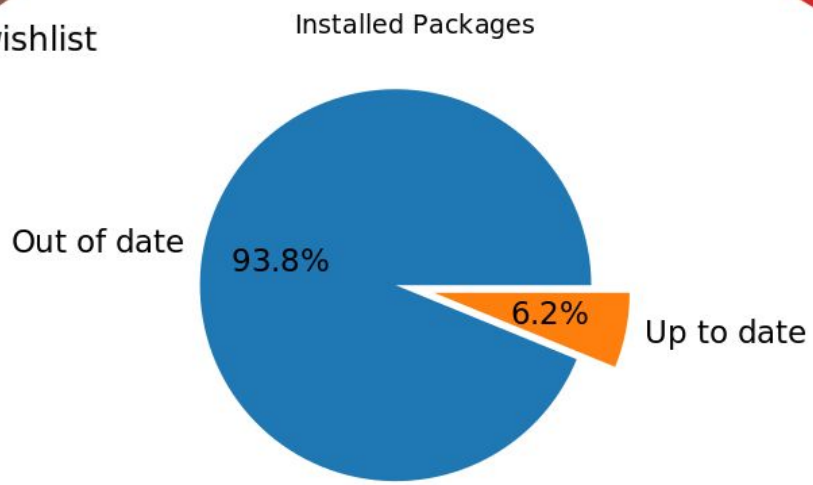
(*): https://github.com/neglectos/ConPan

ConPan: A tool to analyze packages in software containers
> From the CLI

Example: $ conpan -p debian -c google/mysql -d ~/ConPan/data/debian/







ConPan: A tool to analyze packages in software containers
> From the API

```
[1]: #! /usr/bin/env python3
     from conpan.conpan import ConPan

[2]: # Parameters
     kind = 'debian'
     image = 'google/mysql'
     dir_data = '/home/neglectos/ConPan/data/debian/'

[3]: cp = ConPan(packages=kind, image=image, dir_data=dir_data)

[4]: (general_info, installed_packages, tracked_packages, vulnerabilities, bugs) = cp.analyze()

     Connecting to DockerHub... Done
     Pulling the Docker image... Done
     Extracting installed packages... Done
     Tracking installed packages... Done
     Identifying vulnerabilities... Done
     Identifying other kind of bugs... Done
```

ConPan: A tool to analyze packages in software containers
> From the API

```
general_info
```

```
{'description': 'MySQL server for Google Compute Engine',
 'star_count': '18',
 'pull_count': '46959',
 'full_size': '96687899',
 'last_updated': '2015-11-13T01:19:18.235940Z'}
```

```
tracked_packages.head(2)
```

package	version	release_number	debian	release_snapshot	date	source	source_version	outdate
libpcre3	1:8.30-5	7.3	wheezy	jessie	20130506	pcre3	1:8.30-5	8.0
libreadline6	6.2+dfsg-0.1	7.3	wheezy	jessie	20130506	readline6	6.2+dfsg-0.1	7.0

ConPan: A tool to analyze packages in software containers
> From the API

```
vulnerabilities.head(2)
```

source	source_version	package	version	date	outdate	urgency	status	fixed_version	debianbug	cve
libdbd-mysql-perl	4.021-1	libdbd-mysql-perl	4.021-1+b1	20130506	10.0	high**	open	undefined	866818	CVE-2017-10788
pcre3	1:8.30-5	libpcre3	1:8.30-5	20130506	8.0	high**	resolved	2:8.35-3.3+deb8u2	undefined	CVE-2015-2328

```
bugs.head(2)
```

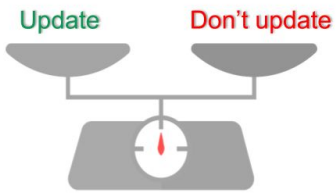
debianbug	source	found_in	fixed_in	type	status	severity	arrival	last_modified	source_version	date
780323	coreutils	8.13-3.3	8.20-1	archived	done	critical	2015-03-12 03:15:02	2019-03-25 07:27:13	8.13-3.5	20130506
705268	ifupdown	0.7.7	0.7.41	archived	done	critical	2013-04-12 09:00:02	2013-06-04 07:30:56	0.7.8	20130506

Summary and Outlook

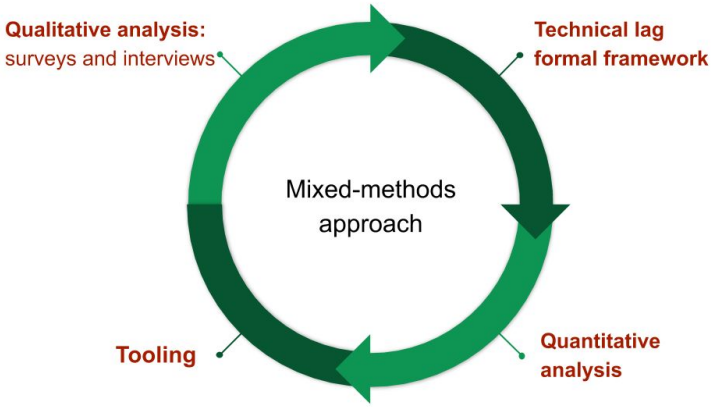
Summary

Focus

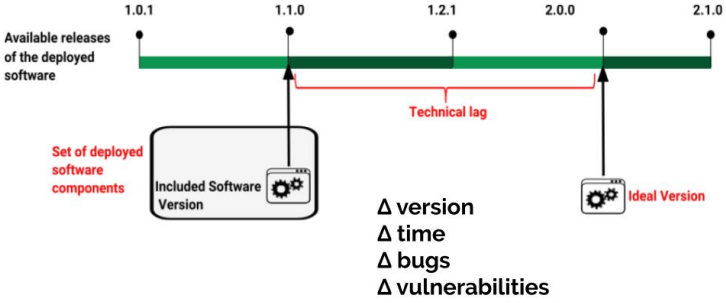
*How can we help software developers to decide **when** and **why** they should update* **?**



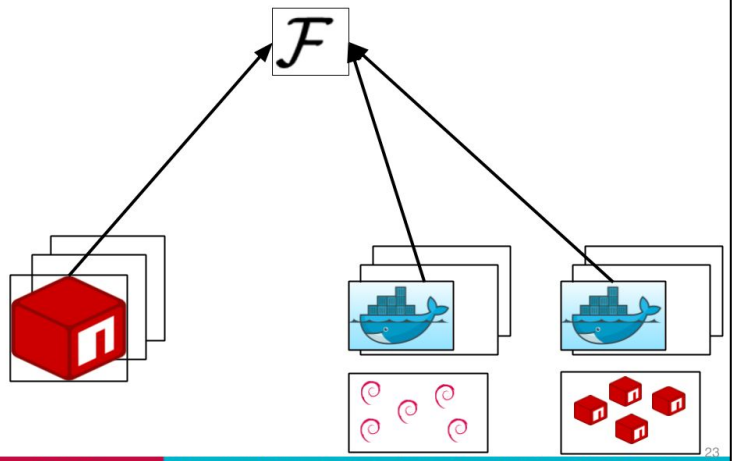
Technical lag
> Followed research approach



Technical lag



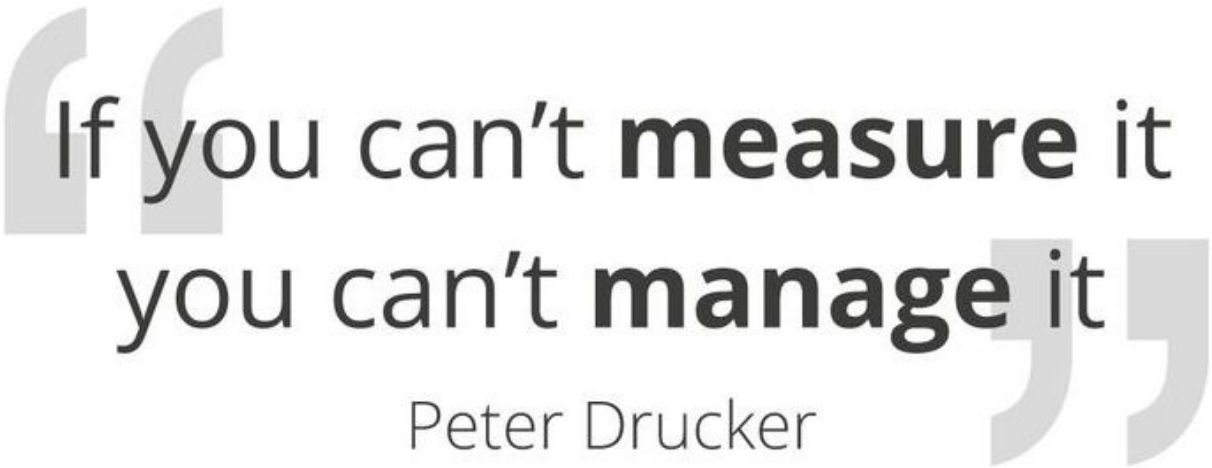
A Framework for Technical Lag
> Framework Instantiation



Future Work

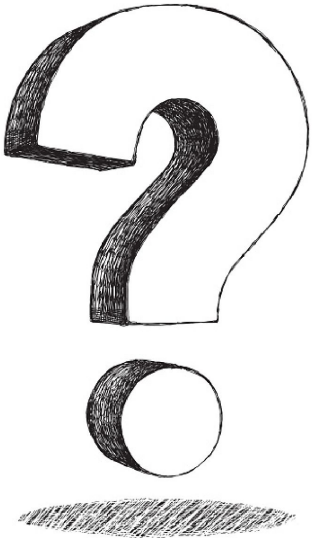
- Other instantiations of the technical lag
  - Effort needed to reduce the technical lag
- Extend and enhance ConPan
- Cross ecosystems comparison
- Promote technical lag to be used by software developers

Conclusion

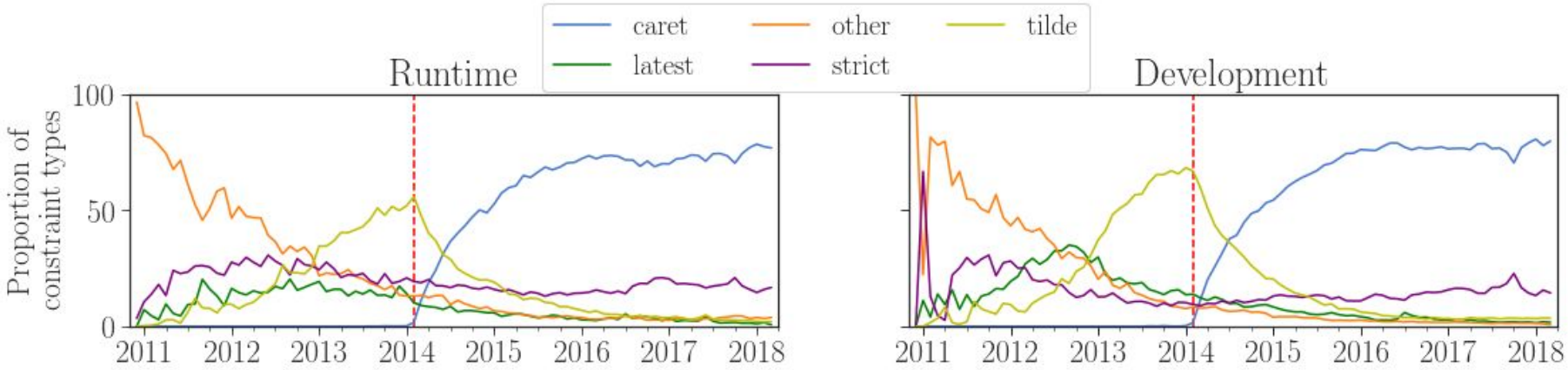


The technical lag could help open source software developers and deployers to keep their software in a healthy shape.


THANK YOU

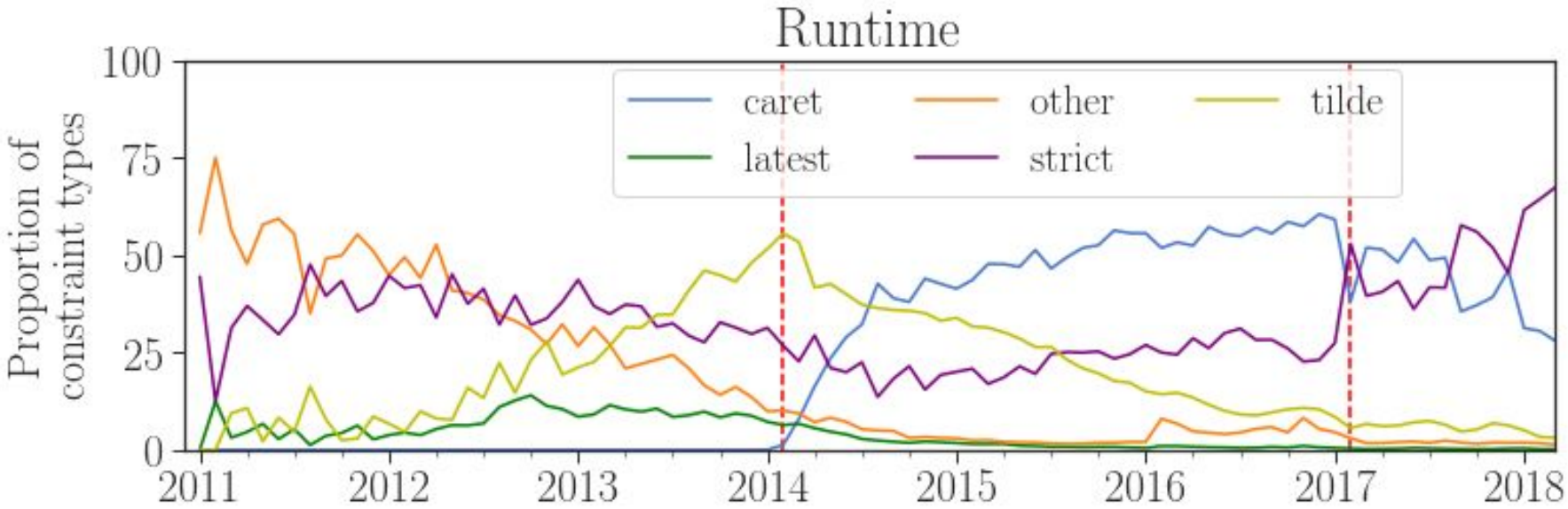


Evolution of Dependency constraints in npm packages



- Caret (^) usage is increasing over time.
- Caret introduction coincides with Major version lag increase.

Evolution of Dependency constraints in GitHub applications



The usage of strict constraint is much higher in external applications