

# Analyzing the effect of introducing time as a component in Python dependency graphs

Author: Andrei Purcaru

Supervisors: Georgios Gousios, Diomidis Spinellis



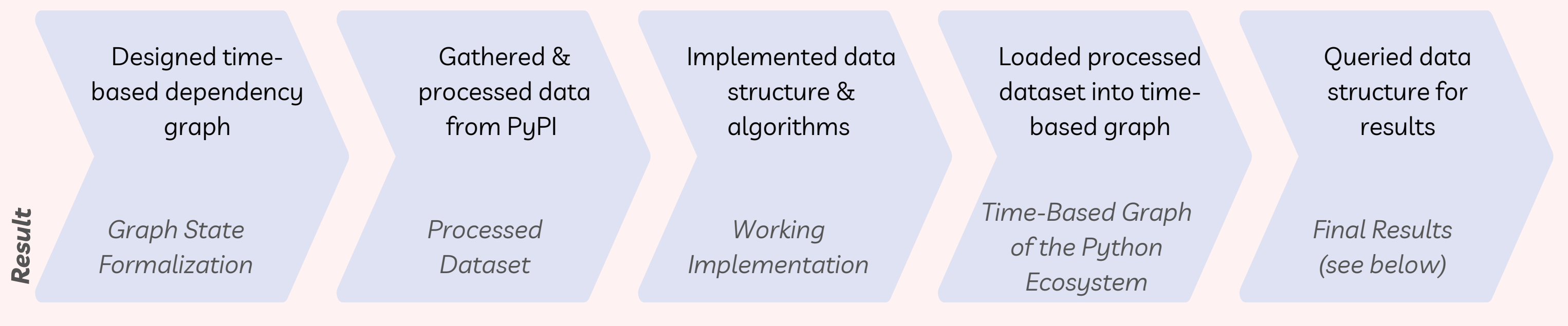
## Background

- Usage of libraries improves development efficiency [1]
- **But** adding dependencies can introduce vulnerabilities
- Visualizing a project's dependencies can be achieved with dependency graphs
- **But** existing tools only target current releases
- No way to check the dependencies of a library at a given time

## Objectives

- Create a **time-based dependency graph data structure**
- Analyze the **most used Python packages** at various points in time

## Methodology



## Results

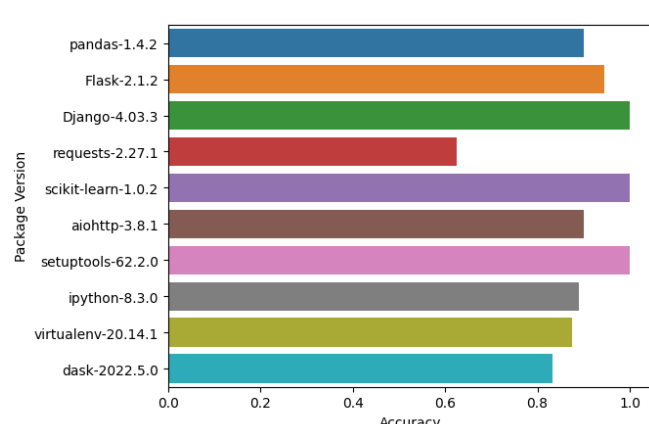
### 1 Formalization of the graph state

$$G_{[t_1, t_2]} = (V, E)$$

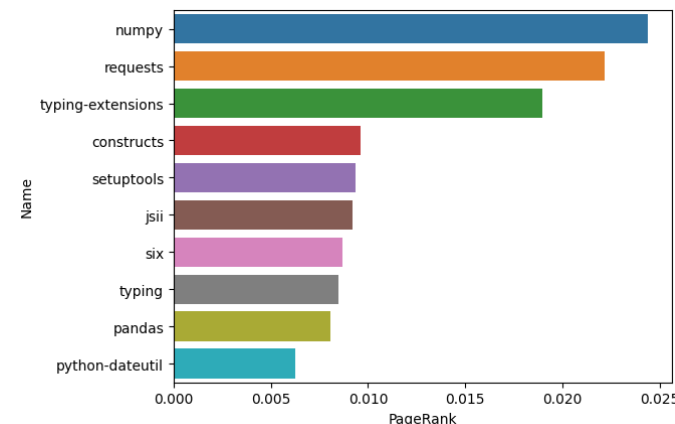
Where:

- $t_1, t_2$  are the timestamps that define the time frame of the state
- $V$  is the set of all package versions, where  $v \in V$  if  $v_{timestamp} \in [t_1, t_2]$
- $E$  is the set of edges, where an edge  $e = (a, b)$  exists only if the package version  $a$  depends on package version  $b$
- $G$  is a directed graph

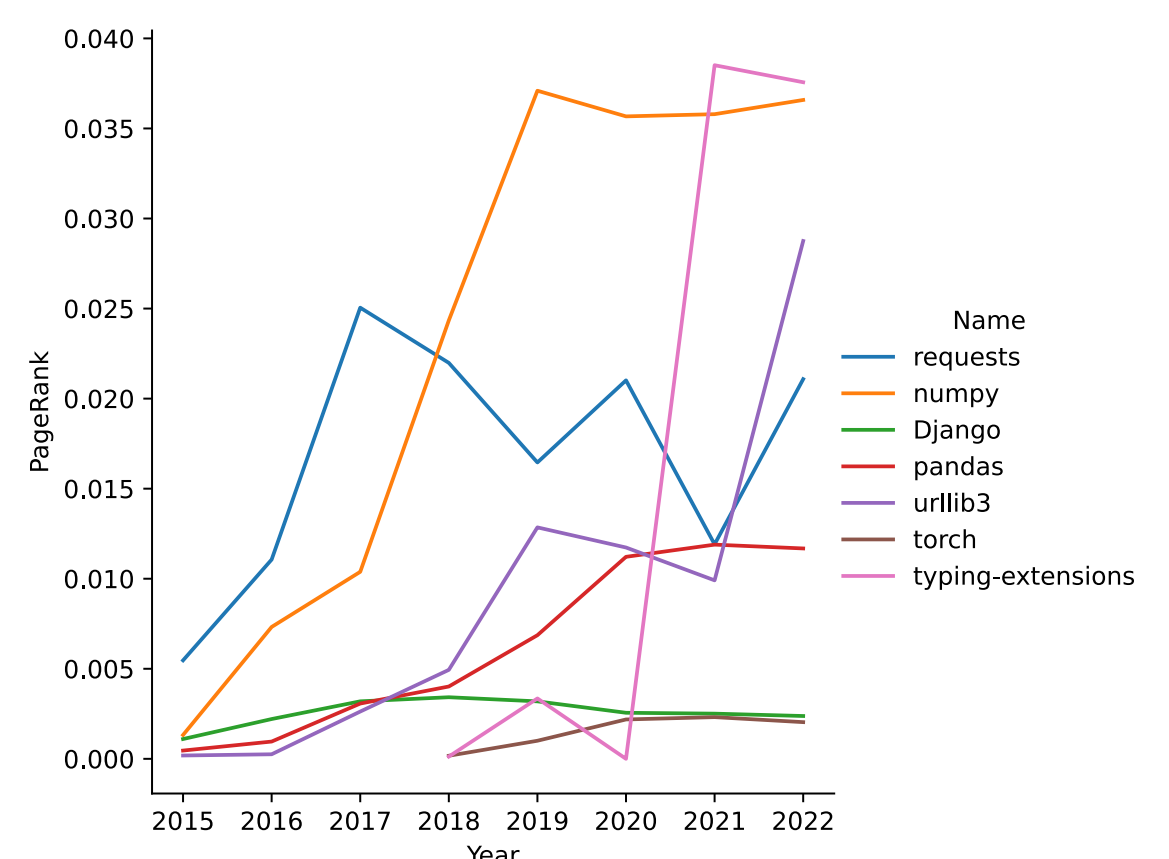
### 2 89.6% resolution accuracy



### 3 Ranking of the Top 10 most important Python packages



### 4 Popularity of selected packages throughout time (2015-2022)



## Conclusion

- The Python ecosystem has evolved over the years, with developers using packages now more than in 2015
- There aren't any packages that once removed would collapse the entire ecosystem
- The time-based dependency graph data structure represents a step into what could be the future of dependency graph analysis

## Terminology

- **Dependency** - a library that provides some functionality to other libraries
- **Dependency Graph** - a graph where each node represents a specific version of a library and the edges represent the dependency/dependent relationship
- **Transitive Dependency** - an indirect dependency resulting when the direct dependencies also have their own dependencies

### Related literature

[1] Parastoo Mohagheghi and Reidar Conradi. Quality, productivity and economic benefits of software reuse: a review of industrial studies. Empirical Software Engineering, 12(5):471–516, 2007.