

OS14: Unveiling R Package Risk Assessment

Comparative Analysis of Metadata Extraction
Techniques

Malte.Stein@Cytel.com



The Clinical Data
Science Conference

EU 2024

10-13 November
Strasbourg Convention Centre



Strasbourg Cathedral

Agenda

01

Requirements on risk assessment for Cytel

R package basics and use case example
Cytel risk-based approach

02

Extraction methods

API and web scrapping

03

Case study

Experience on extraction different metadata for risk assessment

04

Conclusion

Overall experience

Requirement for risk assessment

Why do we need risk assessment

Available packages

- Only **base** and **recommended** R packages are **controlled** by the R Foundation
- **Other packages** developed by anyone (**open source**)

CRAN

- CRAN (Comprehensive R Archive Network) provides thousands of R packages (>20k)
 - **open source**
 - different levels of quality and maintenance

Note: Other repositories not part with this paper/presentation

Requirement for risk assessment

Cytel's risk-based approach (simplified example)

Step 1: Single Package risk category

- Assign risk categories C1 (best) to C5 (Worst) per criteria

Criteria	Criteria Category
Test coverage: 81% / Download rates year: 132k	C2
Archived: No	C1
License: Ok	C1

- Worst category (C2) will be the single package risk category

Step 2: Overall Package risk category

- Analyzing all single package risk categories of depend packages

Depend packages	Risk Category
Package A	C3
Package B	C1

- The **worst single** category (C3) of the package and recursive depend packages is the **overall** package risk category

Methods: API and web scrapping

Explanation

API (Application Programming Interface)

- A tool for requesting data directly from a server, bypassing the website's interface.
 - Offers a **standardized** and **documented** way to access data
 - **Limited** to predefined data and **rate limits**

Web Scraping

- Extracts data from a webpage's **HTML structure**.
 - More **flexible** than APIs
 - **Less stable**, as changes in layout or dynamic content can cause failures.

Methods: API and web scrapping

Example test coverage for dplyr

Link to test coverage in README

- Link as badge:



- HTML code: ``

API:

```
> jsonlite::read_json("https://api.codecov.io/api/v2/gh/tidyverse/repos/dplyr/totals")$totals$coverage
[1] 90.94
```

Web scrapping:

```
> xml2::xml_text(xml2::read_html("https://codecov.io/gh/tidyverse/dplyr/branch/main/graph/badge.svg"))
[1] "codecovcodecov91%91%"
```

Methods: API and web scrapping

Conclusions

Issues during metadata extraction

- **Using APIs** on source pages (e.g., GitHub) is working well for a **few number of packages**.
 - For **all CRAN package** not possible due to **request limits**.
- Implementation of **web scrapping** on source pages was **complex and not stable over time**.

Recommendation

- Avoid using API or web scrapping as much as possible or **use only stable or internal sources**
 - E.g., APIs from internal Posit package manager or official CRAN webpage

Case study – Description file

General information

Content of description file

- The **key information** of a package is stored **in** the **package description** file. The following elements are commonly included:
 - Name and Version
 - Title and description
 - Authors, Maintainer
 - License
 - Imports, Depends, Suggests
- **CRAN** is **providing** also additional **database** for the latest version of each package on <https://cran.r-project.org/web/packages/packages.rds> (or with function **tools::CRAN_package_db()**)

Case study – CRAN checks

General information

What are the CRAN checks?

- **CRAN checks** are automated **tests** that ensure R packages meet **quality**, compatibility, and **CRAN policy** standards before being accepted on CRAN. The R CMD checks are part of it.

Where to find?

- <https://cran.r-project.org/web/checks> (webpage)
- https://cran.r-project.org/web/checks/check_details.rds

How to run R CMD checks?

- `devtools::check(pkg_source_path)`
- `riskmetric::assess_r_cmd_check(riskmetric::pkg_ref(pkg_source_path))`

Fun fact:

Total running time of checks for all CRAN packages are between 12-42 days depending on system.

Case study – Test coverage

Extraction methods

Test Coverage

- The test coverage is the percentage of code, which is covered by test.
- **Covr** R package to **calculate** and share the test coverage

Published Test coverage

- Link in **README** file or others supporting webpage
- Test coverage information shared on **Coverall.io** or **Codecov.io**



Recalculation of test coverage

- With `covr::package_coverage()` the test coverage can be **recalculated with the source code** of the package
 - Package must be **installed** from source **with tests** and **full source code** is needed

Case study – Test coverage

Result from README

Test Coverage from README Files (Snapshot of September 19, 2024)

Test coverage	All (incl. N/A)	$\geq 0\%$	$\geq 50\%$	$\geq 75\%$	$\geq 90\%$	$\geq 95\%$
Number of packages	21438	1762	1537	1257	839	545

- **Similar results** can be generated **using APIs**, but extraction from README files via **web scrapping** is much **easier**.

Case study – Download rates

Metadata types

Available metadata

- Raw **download logs** from the RStudio CRAN mirror (cran.rstudio.com) are **available** for data since October 2012:
 - Detailed information, such as the package version, operating system, and R version

Available functions

- `cranlogs::cran_downloads(package = "dplyr", from = "2023-01-01", to = "2024-01-01")`
 - only total **download rates for all version** of a package

Download Rates using `cranlogs::cran_downloads` (Snapshot of September 10, 2024):

Downloads within 1 year	≥ 0	≥ 2500	$\geq 5k$	$> 10k$	$> 20k$	$\geq 500k$	$\geq 1M$
Number of packages	21274	14742	6732	3769	2598	485	306

Conclusion

Overall experience

Data collection

- **latest versions:** relative **easily** with all methods
 - use stable sources/databases for all packages, before establishing methods on single packages
- **previous versions:** can be **challenging** or must be recreated manually
 - Solution: ongoing automatic data collection

API vs. web scrapping vs. riskmetric

- **all CRAN packages: time-consuming or limited** by request caps
- **few packages:** manageable but scaling up is challenging.
 - The best extraction methods is depending on the type of data or the source.

Remaining challenges

- No industry standard for risk assignment (e.g., min download rates needed)
- Test coverage not part of description file of a package.



Thanks
Any Questions?

Malte Stein
Malte.Stein@Cytel.com