Conscious R packages maintenance

Maciej Nasiński

University of Warsaw

Introduction

Supplementary utils for CRAN maintainers and R package developers in the pacs package. The wide range of tools to achieve healthy R environment and make the R developer life easier. Each of the function was inspired be everyday challenges which have to be faced by the experienced R developer **in the production quality agile project**. These tools should be universal in different work environments.

Objectives

1. Control Complexity of developed packages.
2. Support the process of adding/updating packages.
3. Validate the library.

Reproducible R Environment

- `Docker`/`Podman`, a full reproducibility.
- `renv`, a python pip in the R world. Helps manage the R packages used in a project; it (at least not yet) does not provide any tools for associating a particular version of R with a project.
- RStudio Package Manager (RSPM), useful when handling private company packages.

Tinyverse

The R community `tinyverse` movement is a tryout to create a new package development standards. The clue is the TINY part here. The last years of R package development were full of high number of dependencies temptations. `tinyverse` means as least dependencies as possible as R package dependencies matter. Every dependency you add to your project is an invitation to break your project.

More information is available in the `tinyverse` vignette.

Think Tiny as a Developer

Minimize dependencies and avoid installing “nice to have” packages. The objective should be to have zero dependencies in an R package. That is why `tinytest` was created or `renv` has zero dependencies.

Check a package dependencies exposed to the end users with `pacs::pac_deps_user`, these are packages installed with the `install.packages`. When adding a package to Depends, Imports or LinkingTo, then we impact end users when they invoke `install.packages`. On the other hand if a package is added to Suggests, this impacts mainly developers e.g. `R CMD CHECK`, check all developer exposed packages with `pacs::pac_deps_dev`.

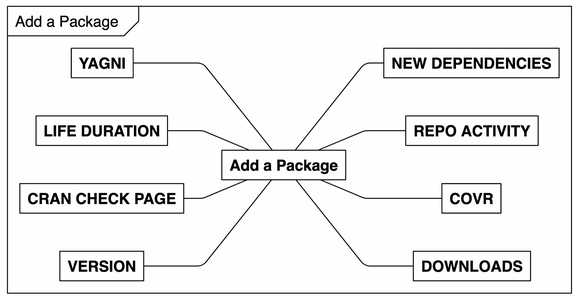
The main strategy should be to prefer usage of `pacs::pacs_base()` packages (already installed in R) and packages with only a few dependencies (the best zero like `jsonlite`).

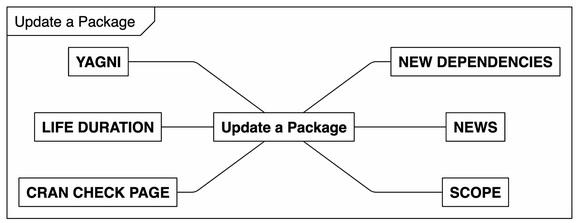
One method of reducing number of dependencies (exposed to end users) is to transfer the package from Imports to Suggests and load it in the delayed manner, or not include it at all. Check the `tinyverse` vignette for more details.

```r
func <- function() {
  if (requireNamespace("PACKAGE", quietly = TRUE)) {
    # regular code
  } else {
    stop("Please install the PACKAGE to use the func function")
  }
}
```

Add/Update a package

Important topics to consider to have **a secure, light and healthy library**. We should be careful when adding/updating packages. The updating part could be surprising nevertheless e.g. in bigger projects we want to have a stable library for each product release.
Tools: `pacs`, `remotes`, CRAN packages websites, cranlogs, and repository website like github.
Example: We think of adding/updating the `parsnip` package.

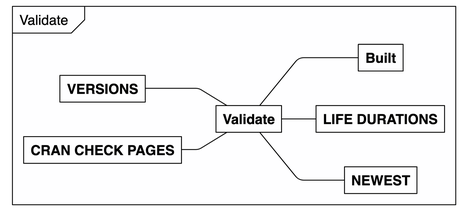




- **COVR**, the coverage of the code in the tests (expected more than 90%); e.g. click the covr badge.
- **CRAN CHECK PAGE**, CRAN checks on variety of servers; `pacs::pac_checkpage("parsnip")`.
- **DOWNLOADS**, popularity of a package; cranlogs app, worth to compare with similar packages.
- **LIFE DURATION**, life duration of the version where we should expect at least 14 days; `pacs::pac_timemachine("parsnip")` or `pacs::pac_lifeduration("parsnip")`.
- **NEW DEPENDENCIES**, what packages have to be added/updated where we want to use light packages with only a few dependencies; `remotes::package_deps("parsnip")`.
- **NEWS**, checking the NEWS file to get latest updates and their significance; `utils::news(package = "parsnip")`; CRAN NEWS PAGE
- **REPO ACTIVITY** how quick issues are answered and how often new commits are added; GITHUB REPO
- **SCOPE**, changes in DECRIPTION/NAMESPACE files between current and the new version; `pacs::pac_compare_versions("parsnip")` and `pacs::pac_compare_namespace("parsnip")`.
- **VERSION**, prefer mature version like 1.2.8 or 0.8.7 not a low 0.1.0; `pacs::pac_description("parsnip")` or `pacs::pac_last("parsnip")`.
- **YAGNI** You aren’t gonna need it, an extreme programming (XP) rule. Do we really need this package/update/feature?
- **OTHER**

Validate

We could validate numerous things like: R library, R package or `renv` lock file. Even if we will not find any obvious problems we will need a proof that the environment is **validated** which is often the authorities’ requirement. R library could be easily broken with functions like `remotes::install_version` or `remotes::install_github(..., dependencies = FALSE)`. On the other hand the `renv` lock file could be built manually or by an external automation pipeline. We could control many risks with `pacs::lib_validate`, `pacs::pac_validate` and `pacs::lock_validate` functions.



- **LIFE DURATIONS**, life duration of the version where we should expected at least 14 days.
- **VERSIONS**, do we have the same versions as required by DESCRIPTION files.
- **NEWEST**, is the newest version.
- **BUILT**, packages installed with a previous version of R could not work correctly with the new version of R. Mostly for local usage (without `renv`) and a minor (last number) R version update.
- **CRAN CHECK PAGE**, CRAN checks on variety of servers. We could extract checks for all CRAN packages with `pacs::checked_packages()`.
- **OTHER**

```r
# assesing life durations could be time consuming for a bigger library.
pacs::lib_validate(
  checkred = list(
    scope = c("ERROR", "FAIL"),
    flavors = pacs::cran_flavors()$Flavor
  ),
  lifeduration = FALSE
)
```

Conclusion

Think `tiny` and be minimalist for better UX and easier development work.

Contact

Maciej Nasinski

References

- https://polkas.github.io/pacs/
- https://remotes.r-lib.org/
- https://rstudio.github.io/renv/articles/renv.html
- https://cranlogs.r-pkg.org/
- https://r-pkgs.org/
- http://blog.obeautifulcode.com/R/How-R-Searches-And-Finds-Stuff/
- https://cran.r-project.org/doc/manuals/r-release/R-exts.html