A while ago we  
[onboarded](https://github.com/ropensci/onboarding/issues/130) an  
exciting package, [codemetar](https://github.com/ropensci/codemetar/)  
by [Carl Boettiger](https://ropensci.org/about/#team). codemetar is an R specific  
information collector and parser for the [CodeMeta  
project](https://github.com/codemeta). In particular, codemetar can  
digest metadata about an R package in order to fill [the terms  
recognized by CodeMeta](https://codemeta.github.io/terms/). This means  
extracting information from DESCRIPTION but also from e.g. continuous  
integration[1] badges in the README! In this note, we’ll take advantage  
of codemetar::extract\_badges function to explore the diversity of  
badges worn by the READMEs of CRAN packages.

**Why codemetar::extract\_badges, and how**

CodeMeta recognized terms include information we’ve been getting from  
badges:

* “contIntegration”, URLs to continuous integration services such as  
  Travis, Appveyor, Codecov;
* “review”, information about review of the software if there was one.  
  codemetar recognizes information from the peer-review badge we add  
  to the README of onboarded packages thanks to work by [Karthik  
  Ram](https://ropensci.org/about/#team).

The list might get longer, so instead of using regular expressions on  
the README text, we extract and memoize[2] all badges at once to a  
data.frame that we then query. The badges extraction is based on (in the  
dev branch of codemetar):

* conversion to XML using commonmark (see [my recent tech  
  note](https://ropensci.org/technotes/2018/09/05/commonmark/)) which works well for the  
  Markdown badges e.g.  
  [![Travis-CI Build Status](https://travis-ci.org/ropensci/codemetar.svg?branch=master)](https://travis-ci.org/ropensci/codemetar).
* extraction and parsing on the first HTML table there is if there’s  
  any, which is necessary for [badges in table, which we’ve been  
  discussing for our  
  guidelines](https://github.com/ropensci/onboarding-meta/issues/56).

Note that the CRAN version of codemetar already features  
extract\_badges, but with a badges table creation based on regular  
expressions only. Here is codemetar::extract\_badges in action [on the  
README of the drake package](https://github.com/ropensci/drake):

library("magrittr")

codemetar::extract\_badges("https://raw.githubusercontent.com/ropensci/drake/master/README.md") %>%

knitr::kable()

| **text** | **link** | **image\_link** |
| --- | --- | --- |
| ropensci\_footer | [https://ropensci.org](https://ropensci.org/) | <http://ropensci.org/public_images/github_footer.png> |
| JOSS | <https://doi.org/10.21105/joss.00550> | <http://joss.theoj.org/papers/10.21105/joss.00550/status.svg> |
| Licence | <https://www.gnu.org/licenses/gpl-3.0.en.html> | [https://img.shields.io/badge/licence-GPL–3-blue.svg](https://img.shields.io/badge/licence-GPL--3-blue.svg) |
| AppVeyor | <https://ci.appveyor.com/project/ropensci/drake> | <https://ci.appveyor.com/api/projects/status/4ypc9xnmqt70j94e?svg=true&branch=master> |
| rOpenSci | <https://github.com/ropensci/onboarding/issues/156> | <https://badges.ropensci.org/156_status.svg> |
| minimal R version | <https://cran.r-project.org/> | <https://img.shields.io/badge/R%3E%3D-3.3.0-blue.svg> |
| Travis | <https://travis-ci.org/ropensci/drake> | <https://travis-ci.org/ropensci/drake.svg?branch=master> |
| CRAN | <http://cran.r-project.org/package=drake> | <http://www.r-pkg.org/badges/version/drake> |
| downloads | <http://cran.rstudio.com/package=drake> | <http://cranlogs.r-pkg.org/badges/drake> |
| Codecov | <https://codecov.io/github/ropensci/drake?branch=master> | <https://codecov.io/github/ropensci/drake/coverage.svg?branch=master> |
| Zenodo | <https://zenodo.org/badge/latestdoi/82609103> | <https://zenodo.org/badge/82609103.svg> |
| Project Status: Active â€“ The project has reached a stable, usable state and is being actively developed. | <http://www.repostatus.org/#active> | <http://www.repostatus.org/badges/latest/active.svg> |

Quite handy for our metadata collection!

Read the source for extract\_badges  
[here](https://github.com/ropensci/codemetar/blob/dev/R/extract_badges.R)  
and see how it’s used in [this  
script](https://github.com/ropensci/codemetar/blob/dev/R/guess_metadata.R).  
You can also compare [codemetar’s  
README](https://github.com/ropensci/codemetar/blob/master/README.md)  
with its  
[codemeta.json](https://github.com/ropensci/codemetar/blob/master/codemeta.json)  
e.g. [the lines for  
“contIntegration”](https://github.com/ropensci/codemetar/blob/master/codemeta.json#L445).

Now since, codemetar::extract\_badges exports a nice data.frame for any  
README with badges, and is exported, it’d be too bad not to use it to  
gain insights from many, many READMEs!

**Extract badges from CRAN packages**

In this exploration we shall concentrate on CRAN packages that indicate  
a GitHub repo link under the URL field of DESCRIPTION. By the way, if  
you don’t indicate such links in DESCRIPTION of your package yet, you  
can (and should) run  
[usethis::use\_github\_links](http://usethis.r-lib.org/reference/use_github_links.html).

**Get links to GitHub repos from CRAN information**

I reckon that I could have also used BugReports like Steven M Mortimer  
did [in his great analysis of CRAN downloads and GitHub  
stars](https://stevenmortimer.com/most-starred-r-packages-on-github/). I  
am unsure of how one can get the link to and the content of the README  
of packages that don’t Rbuildignore their README, such as  
[codemetar](https://cran.r-project.org/web/packages/codemetar/index.html)  
(see under Material). The imperfect sample I collected will do for this  
note.

Here’s how I got all the repo owners and names:

cran\_db <- tools::CRAN\_package\_db()

# only packages that have a GitHub repo

github\_cran <- dplyr::filter(cran\_db[, c("Package", "URL")],

stringr::str\_detect(URL, "github\\.com"))

# will need to keep only the URL to the repo

select\_github\_repo <- function(URL){

URLs <- stringr::str\_split(URL, pattern = ",", simplify = TRUE)

github\_repo <- URLs[stringr::str\_detect(URLs, "github\\.com")][1]

github\_repo <- stringr::str\_remove(github\_repo, "\\#.\*$")

github\_repo <- stringr::str\_remove(github\_repo, "\\#.\*[ \\(.\*\\)]")

github\_repo <- stringr::str\_remove(github\_repo, "/$")

stringr::str\_replace(github\_repo,".\*\\.com\\/", "")

}

github\_cran <- dplyr::group\_by(github\_cran, Package)

github\_cran <- dplyr::mutate(github\_cran, github = select\_github\_repo(URL))

github\_cran <- tidyr::separate(github\_cran, github, "\\/",

into = c("owner", "repo"))

github\_cran <- dplyr::ungroup(github\_cran)

# Not very general

github\_cran$repo[which(github\_cran$Package == "webp")] <- "webp"

I needed a bit of string cleaning mostly to deal with the URLs of  
[Jeroen Ooms](https://ropensci.org/about/#team)’ packages, see e.g. [this  
one](https://cran.r-project.org/web/packages/antiword/index.html). I  
guess I could have cleaned even more, but it was good enough for this  
exploration.

**Get all badges**

Then for each repo I queried the download URL of the [preferred  
README](https://developer.github.com/v3/repos/contents/#get-the-readme)  
via GitHub’s V3 API, using [the gh  
package](https://github.com/r-lib/gh). The preferred README is the one  
GitHub displays on the repo landing page. I used  
codemetar::extract\_badges, of course. I rate-limited the basic  
function using [ratelimitr](https://github.com/tarakc02/ratelimitr).

library("magrittr")

github\_cran <- readr::read\_csv("data/github\_cran\_links.csv")

.get\_badges <- function(owner, repo){

message(paste(owner, repo, sep = "/"))

readme <- try(gh::gh("GET /repos/:owner/:repo/readme",

owner = owner, repo = repo),

silent = TRUE)

if(inherits(readme, "try-error")){

return(NULL)

}else{

badges <- codemetar::extract\_badges(readme$download\_url)

if(nrow(badges)>0){

badges$owner <- owner

badges$repo <- repo

}

return(badges)

}

}

.get\_badges %>%

ratelimitr::limit\_rate(ratelimitr::rate(1, 1)) -> get\_badges

purrr::map2\_df(github\_cran$owner,

github\_cran$repo,

get\_badges) -> badges

**Remove non badges from the sample**

The way badges are recognized by codemetar::extract\_badges is not  
specific enough, it can include images formatted like badges that aren’t  
badges but instead either local images or images whose credit is shown  
as URL. To remove them from the sample, I used a strategy in two steps:

* I first had a look at the most common domains. For the 17 most  
  common of them, I accepted the images except for one, ropensci.org,  
  included because the *footer* our packages get is formatted as a  
  Markdown badge.
* For the remaining images, a bit more than 200, I used magick to  
  obtain their width and height, and filtered actual badges based on  
  their width/height ratio. Sometimes the link to the image wasn’t  
  even valid, which was also a reason for exclusion, since it revealed  
  the image was a local one.

# extract and parse URLs

badges %>%

dplyr::pull(image\_link) %>%

purrr::map\_df(urltools::url\_parse) -> parsed\_image\_links

# count hits by domain

parsed\_image\_links %>%

dplyr::count(domain, sort = TRUE) -> domain\_count

# these were manually inspected

# as legit badge providers

ok\_domain <- domain\_count$domain[1:17]

# keep the badges needing a check

tbd <- dplyr::filter(parsed\_image\_links,

! domain %in% ok\_domain)

# get their size ratio

get\_size <- function(url){

img <- try(magick::image\_read(url),

silent = TRUE)

if(inherits(img, "try-error")){

tibble::tibble(error = TRUE,

image\_link = url)

}else{

info <- magick::image\_info(img)

info$error <- FALSE

info$image\_link <- url

info

}

}

img\_info <- purrr::map\_df(urltools::url\_compose(tbd),

get\_size)

img\_info <- dplyr::mutate(img\_info, ratio = width/height)

# filter badges from images

img\_info <- dplyr::filter(img\_info,

ratio < 3|error)

# it'd have been wiser to use a row-wise workflow!

badges <- dplyr::filter(badges,

!image\_link %in% img\_info$image\_link,

!image\_link %in% stringr::str\_remove(img\_info$image\_link,

"/$"),

!tolower(image\_link) %in% img\_info$image\_link,

!tolower(image\_link) %in% stringr::str\_remove(img\_info$image\_link,

"/$"),

!stringr::str\_detect(image\_link,

"ropensci\\.org\\/public\\\_images\\/"))

readr::write\_csv(badges, "data/aaall\_badges.csv")

Don’t judge me by my filenaming skills. I was *maybe* a bit too  
enthusiastic!

**Analyze badges from CRAN packages**

I wanted to answer several questions about the badges of CRAN packages,  
beyond being just happy to have been able to collect so many of them.

**How many repos have at least one badge?**

github\_cran <- readr::read\_csv("data/github\_cran\_links.csv")

# the same repo can have been used by several packages!

badges <- readr::read\_csv("data/aaall\_badges.csv")

badges <- dplyr::distinct(badges)

nobadges <- dplyr::anti\_join(github\_cran, badges,

by = c("owner", "repo"))

There are 1277 packages without any badge (or rather said, without any  
badge that we identified) out of a sample of 3541 packages. That means  
64% have at least one badge. As a reminder, there are more than 13,000  
packages on CRAN so we’re only looking at a subset.

**Among the repos with badges, how many badges?**

library("magrittr")

badges %>%

dplyr::count(repo, owner,

sort = TRUE) -> badges\_count

badges\_count %>%

dplyr::summarise(median = median(n))

## # A tibble: 1 x 1

## median

##

## 1 4

library("ggplot2")

badges\_count %>%

ggplot() +

geom\_histogram(aes(n))+

hrbrthemes::theme\_ipsum(base\_size = 12,

axis\_title\_size = 12,

axis\_text\_size = 12) +

ggtitle("Number of badges per repo",

subtitle = "Among repos with at least one badge")

number of badges for READMEs with at least one
(histogram)

The median number of badges is 4, which corresponds to my gut feeling  
that the answer would be “a few”. I have a new question, what are the  
repos with the most badges?

most\_badges <- dplyr::filter(badges\_count,

n == max(n))

most\_badges

## # A tibble: 2 x 3

## repo owner n

##

## 1 gpuR cdeterman 13

## 2 psycho.R neuropsychology 13

You can browse them at <https://github.com/cdeterman/gpuR>,  
<https://github.com/neuropsychology/psycho.R>.

**How many unique badges are there?**

For counting types of badges, I’ll use the domain of image\_link. This  
is an approximation, since e.g. www.r-pkg.org offers several badges.

badges %>%

dplyr::pull(image\_link) %>%

purrr::map\_df(urltools::url\_parse) -> parsed\_image\_links

parsed\_image\_links %>%

dplyr::pull(domain) %>%

unique() %>%

sort() -> unique\_domains

length(unique\_domains)

## [1] 50

Not *that* many after all, so I’ll print all of them! A special mention  
to <https://github.com/ropensci/cchecksapi#badges> maintained under our  
GitHub organization by [Scott Chamberlain](https://ropensci.org/about/#team), to show the  
CRAN check status of your package!

Unique badge domains collapsed by glue::glue\_collapse(unique\_domains, sep = ", ", last = " and "): anaconda.org, api.codacy.com, api.travis-ci.org, app.wercker.com,  
assets.bcdevexchange.org, awesome.re, badge.fury.io,  
badges.frapsoft.com, badges.gitter.im, badges.herokuapp.com,  
badges.ropensci.org, bestpractices.coreinfrastructure.org,  
ci.appveyor.com, circleci.com, codeclimate.com, codecov.io,  
coveralls.io, cranchecks.info, cranlogs.r-pkg.org, depsy.org,  
dmlc.github.io, eddelbuettel.github.io, githubbadges.com,  
githubbadges.herokuapp.com, gitlab.com, hits.dwyl.io, i.imgur.com,  
img.shields.io, jhudatascience.org, joss.theoj.org, mybinder.org,  
popmodels.cancercontrol.cancer.gov, pro-pulsar-193905.appspot.com,  
raw.githubusercontent.com, readthedocs.org, saucelabs.com,  
semaphoreci.com, travis-ci.com, travis-ci.org,  
user-images.githubusercontent.com, usgs-r.github.io, www.nceas.ucsb.edu,  
www.ohloh.net, www.openhub.net, www.paypal.com, www.r-pkg.org,  
www.rdocumentation.org, www.repostatus.org, www.rpackages.io and  
zenodo.org.

**What are the most common badges?**

Note that this doesn’t take into account the fact that one domain can  
appear several times in a single README (Travis status for different  
branches for instance).

parsed\_image\_links %>%

dplyr::count(domain, sort = TRUE) %>%

head(n = 10) %>%

knitr::kable()

| **domain** | **n** |
| --- | --- |
| travis-ci.org | 1880 |
| www.r-pkg.org | 1804 |
| cranlogs.r-pkg.org | 1286 |
| img.shields.io | 882 |
| ci.appveyor.com | 698 |
| codecov.io | 656 |
| www.repostatus.org | 240 |
| zenodo.org | 197 |
| coveralls.io | 157 |
| www.rdocumentation.org | 86 |

The most common badges are Travis-CI badges, and METACRAN badges from  
[www.r-pkg.org](https://www.r-pkg.org/) and  
[cranlogs.r-pkg.org](https://cranlogs.r-pkg.org/). Now, “img.shields.io”  
is a service for badges of other things… which?

badges %>%

dplyr::filter(stringr::str\_detect(image\_link, "img\\.shields\\.io")) %>%

dplyr::count(text, sort = TRUE)

## # A tibble: 135 x 2

## text n

##

## 1 Coverage Status 196

## 2 License 91

## 3 lifecycle 62

## 4 CoverageStatus 54

## 5 38

## 6 packageversion 37

## 7 Last-changedate 32

## 8 Licence 28

## 9 minimal R version 28

## 10 Github Stars 19

## # ... with 125 more rows

Diverse things, in particular the [Tidyverse lifecycle  
badges](https://www.tidyverse.org/lifecycle/). After some discussion, we  
at rOpenSci have adopted the repostatus.org status badges in our  
guidelines… but are actually open to repos using both types of badges  
since their nomenclature can complement each other!

**Conclusion**

In this tech note I presented and used one of codemetar’s tools for R  
package metadata munging, extract\_badges. I extracted and analyzed  
badges information from the READMEs of all CRAN packages that indicate a  
GitHub repo in the URL field of DESCRIPTION. README badges are a way to  
show development status, test results, code coverage, peer-review merit,  
etc.; but can also be used as a machine-readable source of information  
about these same things.