The [seymour[](https://git.sr.hr/~hrbrmsts/seymour)](https://git.sr.hr/~hrbrmsts/seymour) Feedly API package has been updated to support subscribing to RSS/Atom feeds. [Previously](https://rud.is/b/2018/12/31/exploring-2018-r-bloggers-r-weekly-posts-with-feedly-the-seymour-package/) the package was intended to just treat your Feedly as a data source, but there was a compelling use case for enabling subscription support: subscribing to code repository issues. Sure, there’s already email notice integration for repository issues on most social coding platforms but if you always have Feedly up (like I usually do) having issues aggregated into a Feedly category may be a better way to keep tabs on what’s going on.

If you use GitLab, that platform already has RSS feeds for public repositories. GitHub users have to use either [RSSHub](https://docs.rsshub.app/en/) or [gh-feed](http://gh-feed.imsun.net/) to do the same (note that you can host your own instance of either of those tools).

If you have more than a few repos and want to have their issues shunted to Feedly you *could* go manually enter them into the Feedly UI but that could be a pain, especially if “more than a few” is in the dozens or hundreds. But, we have R and can automate this. I’m providing an example for GitHub (since most readers art still stuck on that legacy platform) via the gh package but the same methods can be done for GitLab using the [gitlabr[](https://blog.points-of-interest.cc/post/gitlabr/)](https://blog.points-of-interest.cc/post/gitlabr/) package.

First, we need to get the list of public GitHub issues you own:

library(gh)

library(purrr)

library(seymour) # git.sr.ht/~hrbrmstr/seymour, git[la|hu]b/hrbrmstr/seymour

gh::gh(

"/user/repos",

visibility = "public",

affiliation = "owner",

sort = "created", direction = "desc",

.token = Sys.getenv("GITHUB\_PAT") # see ?devtools::github\_pat

) -> gh\_repos

If you have more than 30 the gh package has a gh\_next() function which will enable you to paginate through all your repos.

Now you need to choose between gh-feed or RSSHub and prepend a special URL prefix to a user/repo string to create a usable RSS URL. These are the prefixes:

* <http://gh-feed.imsun.net/>
* <https://rsshub.app/github/>

We’ll use RSSHub for the remainder of the example.

Let’s do this first in base R. feedly\_subscribe() takes an RSS/Atom URL as a parameter and optionally supports supplying a title and a vector of Feedly category names to help organize your new addition. The title will be automagically intuited by Feedly if not supplied. We’ll iterate over the return value of our call to the GitHub API and add subscribe to each repo.

do.call(

rbind.data.frame,

lapply(sapply(gh\_repos, "[[", "name"), function(.x) {

feedly\_subscribe(

feed\_url = sprintf("https://rsshub.app/github/%s/%s", "hrbrmstr", .x),

category = "github issues"

)

})

) -> res

The res value is a data frame that just has the resultant metadata about feed ids and where they’re located.

Here’s the same thing tidyverse-style:

map\_chr(repos, "name") %>%

sprintf("https://rsshub.app/github/%s/%s", "hrbrmstr", .) %>%

map\_df(feedly\_subscribe, category = "github issues") -> res

**FIN**

There are two other addition to the seymour package: feedly\_subscriptions(). This convenience function just pulls a data frame of feeds you subscribe to. The same data could be retrieved via the existing “stream” functions but this new function is faster and more targeted. The other one is feedly\_categories() which you can use to identify the categories you have. The same data could be retrieved via the “collections” functions, but this function — again — is faster and more targeted.

As usual, kick the tyres and file issues/PRs as needed.