**Searching repositories on GitHub**

**Study design **

GitHub’s V3 API offers a search  
endpoint,  
that however gives you less results than doing the same search via the  
web interface, even when  
using pagination right (or at least, when believing I use pagination  
right!). I’m however willing to use that sub-sample as basis for my  
study of language popularity. It’s actually a sub-sub-sample, since I’m  
only looking at Advent of code projects published on GitHub.

In order to circumvent the sub-sub-sampling a bit, I’ll do the search in  
two steps:

* Searching for Advent of code 2018 in general among repos, and  
  extracting the language of the repos.
* Searching for Advent of code 2018 *for each of these languages  
  separately* and extracting the total count of hits.

Note that I am not filtering the repos by activity, so some of them  
could very well have been created for a few days only. If they are empty  
though, they do not get assigned a language.

**Actual queries**

I first defined a function to search the API whilst respecting the rate  
limiting. I even erred on the side of caution and queried very slowly.

.search <- function(page){

gh::gh("GET /search/repositories",

q = "adventofcode 2018",

page = page,

fork = FALSE)

}

search <- ratelimitr::limit\_rate(.search,

ratelimitr::rate(10, 60))

I then wrote two other functions to help me rectangle the API output for  
each repository.

empty\_null <- function(x){

if(is.null(x)){

""

}else{

x

}

}

rectangle <- function(item){

tibble::tibble(full\_name = item$full\_name,

language = empty\_null(item$language))

}

I created a function putting these two pieces together.

get\_page <- function(page){

results <- try(search(page), silent = TRUE)

# an early return

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

return(NULL)

}

purrr::map\_df(results$items,

rectangle)

}

And I then ran the following pipeline.

total\_count <- search(1)$total\_count

pages <- 1:(ceiling(total\_count/100))

results <- purrr::map\_df(pages, get\_page)

results <- unique(results)

languages <- unique(results$language)

languages <- languages[languages != ""]

This got me 814 repos, with 46 non empty languages. Repo names are quite  
varied: rdmueller/aoc-2018, petertseng/adventofcode-rb-2018,  
NiXXeD/adventofcode, Arxcis/adventofcode2018,  
Stupremee/adventofcode-2018, phaazon/advent-of-code-2k18.

With that information obtained, I was able to run a query by language.

.get\_one\_language\_count <- function(language){

gh::gh("GET /search/repositories",

q = glue::glue("adventofcode 2018&language:{language}"),

fork = FALSE)$total\_count -> count

tibble::tibble(language = language,

count = count)

}

get\_one\_language\_count <- ratelimitr::limit\_rate(.get\_one\_language\_count,

ratelimitr::rate(10, 60))

counts <- purrr::map\_df(languages,

get\_one\_language\_count)

In total, the counts table contains information about 2080  
repositories, a bit less than half the number of Advent of code 2018  
repositories I’d find via the web interface.

**Advent of Code’s languages popularity**

I’ll concentrate on the 15 most popular languages in the sample, which  
automatically excludes R with… 8 repositories only.

library("ggplot2")

library("ggalt")

library("hrbrthemes")

library("magrittr")

counts %>%

dplyr::arrange(- count) %>%

head(n = 15) %>%

dplyr::mutate(language = reorder(language, count)) %>%

ggplot() +

geom\_lollipop(aes(language, count),

size = 2, col = "salmon") +

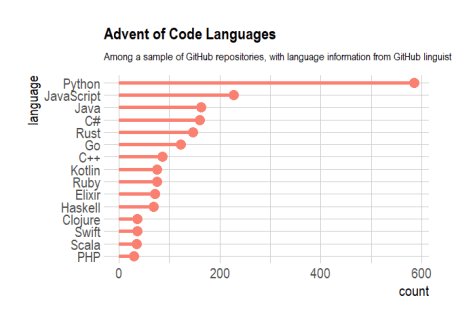
hrbrthemes::theme\_ipsum(base\_size = 16,

axis\_title\_size = 16) +

coord\_flip() +

ggtitle("Advent of Code Languages",

subtitle = "Among a sample of GitHub repositories, with language information from GitHub linguist")



**Conclusion**

In this post I used GitHub V3 API to get a glimpse at the popularity of languages used to solve the Advent of Code. Further work could include looking at the *completion* of the challenge by language, potentially using the GitHub activity of each repo as an (imperfect) proxy.

puzzles… Do *you* participate to Advent of Code this year? If so, with  
which language and why?