



What R We Counting?

The Quest to Quantify R Usage in Biostatistics Organizations

Ben Arancibia



**As more and more Biostatistics Organizations
begin to adopt open-source tools, leaders might
start asking this question...**

How much R or Python are our clinical study teams using?



Director of Data Science in GSK Biostatistics

**Scientific and Adoption Lead for
Open Source for GSK R&D**

**Passionate about learning and
teaching**

How do you answer how much R our clinical study teams are using?

A realization

Count number of outputs in a study?

Do you count

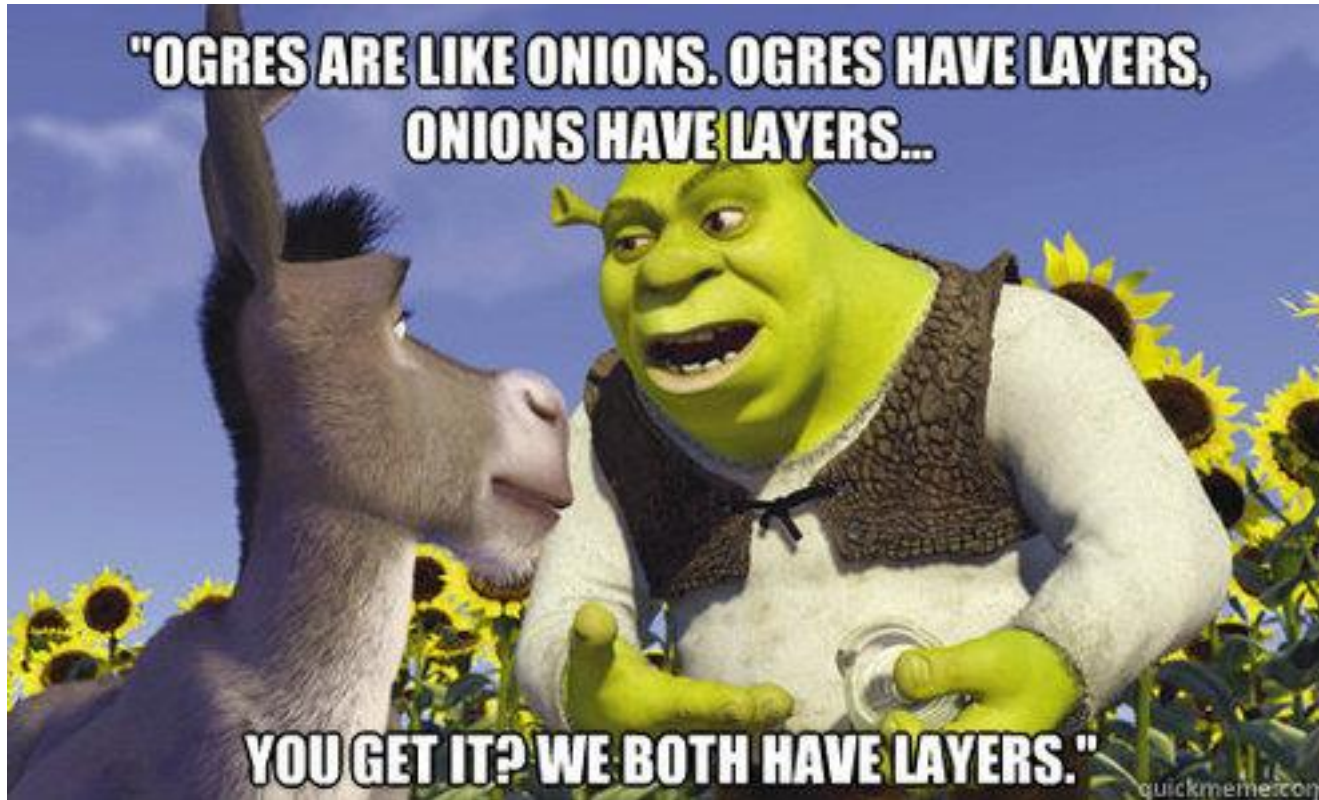
Do you care about
languages?

Do you timebox it?

**Answering such a simple
question “How much R or Python
are our clinical study teams
using?” is really hard...**

r QC

ter



R Adoption is like onions. Onions have layers, R Adoption has layers... You get it, they both have layers.

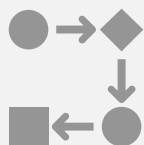
How we peeled back the layers

Why did our senior stakeholders ask us this question?

Since 2020 Putting together the Pieces



Platform



Process



Tools



Pipeline



Training



Two Commitments in August 2023

**All central tools to be built
using open-source languages**

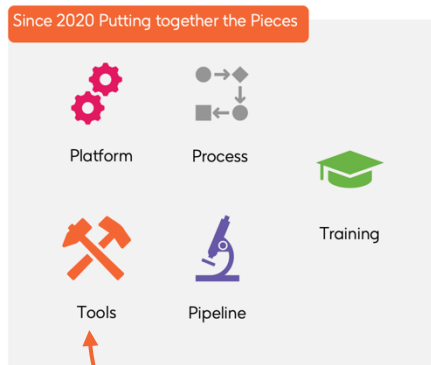
**50% of our code to be open-
source by end of 2025**

How we answered how much R are our clinical study teams using?



More specifically the GitHub APIs

Exploring what information is created by tools



All studies must use version control

Releases 30

v1.1.1 Latest
on Jun 18, 2024

[+ 29 releases](#)

Packages

No packages published

Contributors 89



[+ 75 contributors](#)

Deployments 423

✓ github-pages 4 hours ago

[+ 422 deployments](#)

Languages

● R 100.0%

Ask the R Ecosystem and It Provides

REST API / Metrics /

REST API endpoints for repository statistics

Use the REST API to fetch the data that GitHub uses for visualizing different types of repository activity.

About repository statistics [↗](#)

You can use the REST API to fetch the data that GitHub uses for visualizing different types of repository activity.

Best practices for caching [↗](#)

Computing repository statistics is an expensive operation, so we try to return cached data whenever possible. If the data hasn't been cached when you query a repository's statistics, you'll receive a `202` response; a background job is also fired to start compiling these statistics. You should allow the job a short time to complete, and then submit the request again. If the job has completed, that request will receive a `200` response with the statistics in the response body.

Repository statistics are cached by the SHA of the repository's default branch; pushing to the default branch resets the statistics cache.

Statistics exclude some types of commits [↗](#)

The statistics exposed by the API match the statistics shown by [different repository graphs](#).

To summarize this:

- All statistics exclude merge commits.
- Contributor statistics also exclude empty commits.

Get the weekly commit activity [↗](#)

gh 1.4.1 Reference Articles ▾ Changelog

Search for



gh

Minimalistic client to access GitHub's [REST](#) and [GraphQL](#) APIs.

Installation

Install the package from CRAN as usual:

```
install.packages("gh")
```

Install the development version from GitHub:

```
pak::pak("r-lib/gh")
```

Usage

```
library(gh)
```

Use the `gh()` function to access all API endpoints. The endpoints are listed in the [documentation](#).

The first argument of `gh()` is the endpoint. You can just copy and paste the API endpoints from the documentation. Note that the leading slash must be included as well.

From <https://docs.github.com/rest/reference/repos#list-repositories-for-a-user> you can copy and paste `GET /users/{username}/repos` into your `gh()` call. E.g.

LINKS

[View on CRAN](#)
[Browse source code](#)
[Report a bug](#)

LICENSE

[Full license](#)
MIT + file LICENSE

COMMUNITY

[Code of conduct](#)

CITATION

[Citing gh](#)

DEVELOPERS

[Gábor Csárdi](#)
Maintainer, contributor
[Jennifer Bryan](#)
Author
[Hadley Wickham](#)
Author
 positt
Copyright holder, funder

An unconventional way to use {gh}

{gh} is primarily used for managing GitHub PATs

```
install.packages("gh")
library(gh)

#Set the environment variable
Sys.setenv(GITHUB_PAT = "your_personal_access_token")

my_repos <- gh("GET /users/{username}/repos", username = "Fake Username")
vapply(my_repos, "[", "", "name")
```

BUT

It can also be used for complex queries

```
#function to get repo level information for all repos in gsk tech using graphql
get_all_repos<-function(auth = gh::gh_token()){

  # 100 repos at a time - need to paginate thru
  #(note that gh package offers limit=.Inf for REST API which paginates under the hood)
  hasNextPage <- TRUE
  endCursor <- NA
  repo_list <- vector(mode = "list")

  system.time(
    while(hasNextPage){

      query <- paste0('query {
        organization(login: "ORG NAME") {
          repositories(first: 100,
            after: ', jsonlite::toJSON(endCursor, auto_unbox = TRUE), ') {
            pageInfo {
              hasNextPage
              endCursor
            }
            edges{
              node{
                name
                diskUsage
                createdAt
                pushedAt
                updatedAt
                templateRepository{
                  name
                }
              }
            }
          }
        }
      }')

      # Fetch only top 10 languages
      languages (first:10){
        totalSize
        edges {
          size
          node{
            name
          }
        }
      }
    }
  )
}
```

**Started exploring and pulling data through the
API when...the layers reappear**

Endless Possibilities

What do you track?

Files Counts?
File Size?
Commits?
File counts and size by department?
Non-study repos vs study repos?

What metrics do you decide to create?

Summary vs predictive?

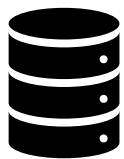
Deal with weird use cases?

Shiny - creates HTML, JS, but R is minimal.



Top Line Metrics Important to GSK Biostatistics

Relative R Size



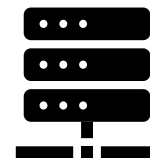
R code size out of (R and SAS)

Any R



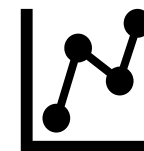
Total Repos with any usage of R

Substantial R



Repos where R code size (out of R and SAS) is $\geq$ Target Percent

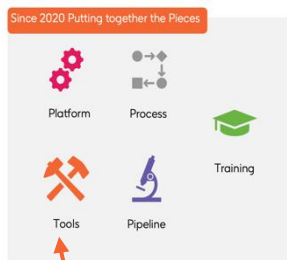
Growth of R



Rate of change of R code in a repo and organization

Layers Reappear

Noticed other things....



Useful Data

- Commits
- Pull Requests
- Merge Conflicts
- Number of Branches
- Repo Life (days)
- Repo Pull Request Reviewers
- Repo Names
- Studies with multiple repos
- Irregular commits
- Unmerged pull requests

Two New Projects Utilizing the GitHub API

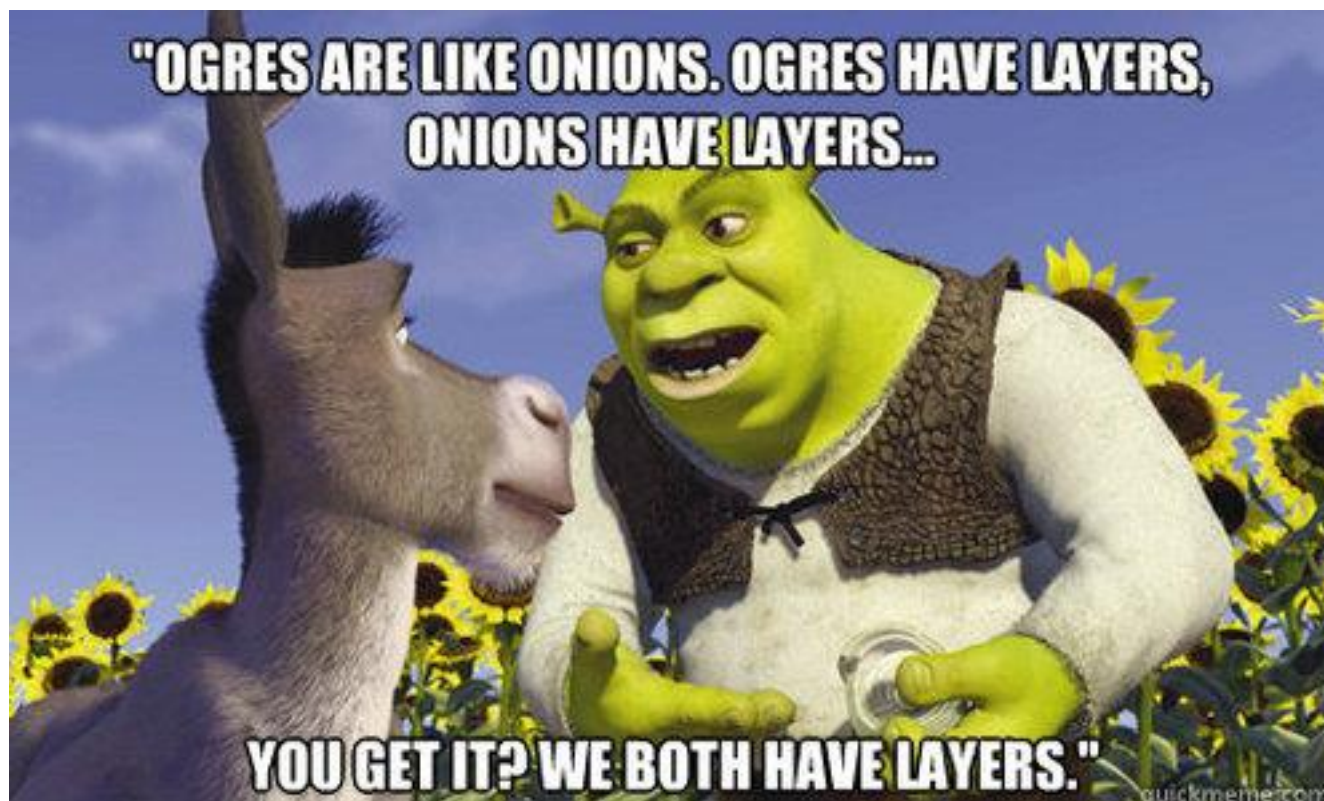
GitHub Health

GitHub is hard to learn...how do we support users in this transition. Can we proactively find studies that are struggling and support them.

Asset Catalogue

What tools exist out there for individual study teams that can be scaled across the entire Biostatistics Organization? Are our internal tools actually being used?

How do you answer how much R our clinical study teams are using?



It's all about the layers!

GSK