request: an http requests DSL

Scott Chamberlain, rOpenSci
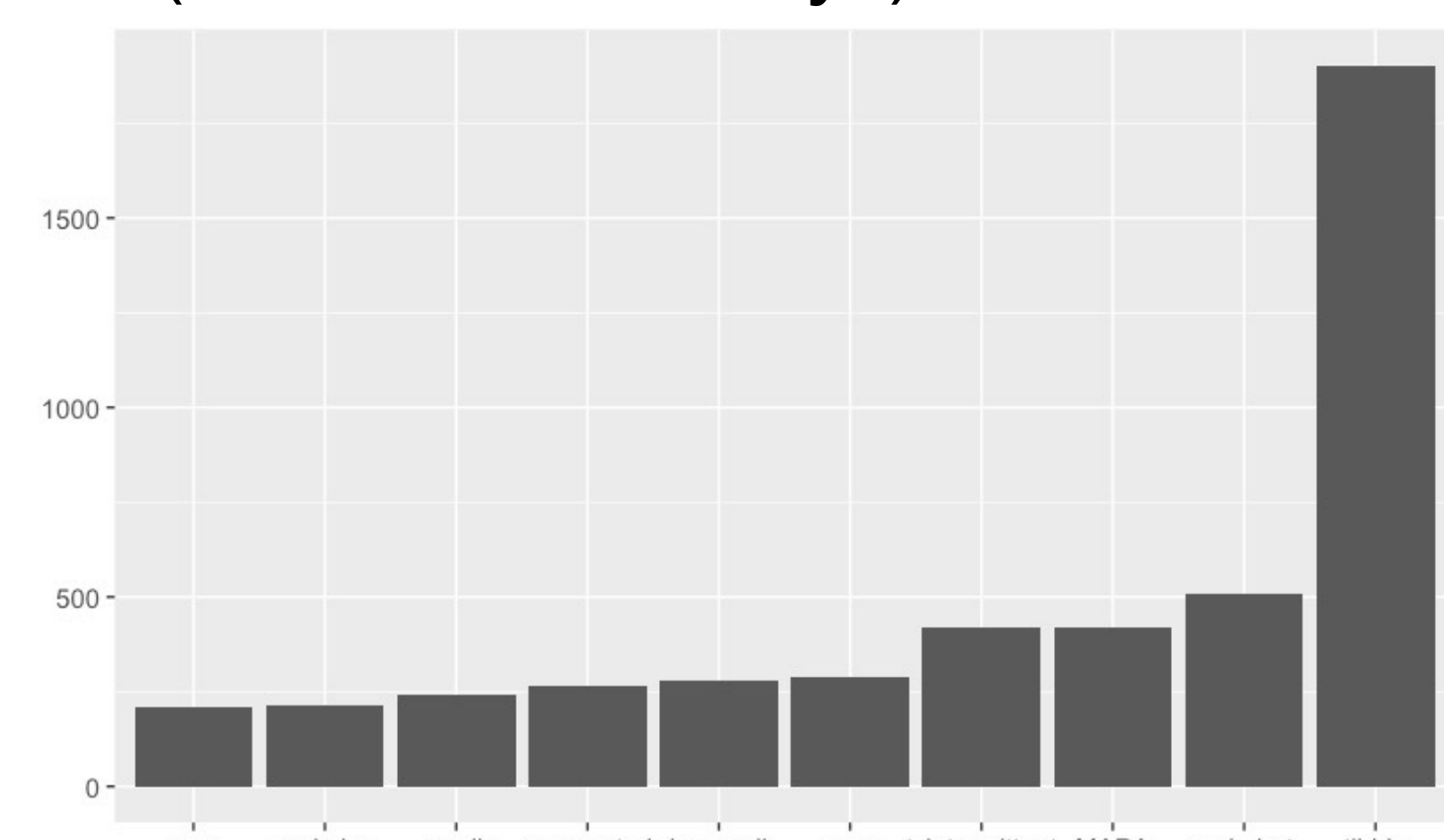
ROpenSci

🐦 @sckottie

request

github: *sckott/request*

CRAN: request

Philosophy

- Reduce cognitive load

- Make it easy as possible to do HTTP requests

- Get to a data.frame faster to take advantage of downstream tools: *dplyr, data.table, tidyr, ggplot2*, etc

- APIs are trending towards a common standard:

  - REST type

  - transport over HTTP

  - data format: JSON

- Make assumptions about what most users want to do (**GET** requests), and what data they will get back (**JSON**) -> makes things easier - fallbacks for other verbs and data types

- Give back data.frame (a *tibble*) if possible

- inspiration from Python's *httpie*

```
x <- api('cranlogs.r-pkg.org') %>% api_path(trending)
x %>%
mutate(increase = as.numeric(increase)) %>%
arrange(desc(increase)) %>%
top_n(10) %>%
ggplot(aes(reorder(package, increase), increase) +
geom_bar(stat = "identity")
```

request package features

Full and partial URLs
```
api('http://localhost:9200')
api('localhost/9200')
api(':9200')
api('9200')
```

Paths

- Build up URL piece by piece

```
api_path(the, red, fox)
# gives 'the/red/fox'
```

Path templating

```
x <- list(user = 'a', repo = 'b')
api_template('{{user}}/{{repo}}/', x)
```

Query

```
api_query(q = ecology, wt = json)
```

Pipes

- It's easy to use pipes or not

```
'httpbin.org/get' %>% api()
http(api('httpbin.org/get'))
```

Evaluate on last pipe
- We don't perform http request if not piped

```
# http request made
'httpbin.org/get' %>% api()

# http request NOT made
api('httpbin.org/get')

# call http explicitly
api('httpbin.org/get') %>% http()
```

NSE & SE
```
api('https://api.github.com') %>%
    api_path(repos, ropensci, rgbif)
```

```
api('https://api.github.com') %>%
    api_path_('repos', 'ropensci', 'rgbif')
```

Automatic POST w/ body
```
api('http://httpbin.org') %>%
    api_body(foo = 'bar')
```

Write to disk helper
```
api('api.github.com') %>%
    api(orgs, ropensci, events) %>%
    api_write(ff <- tempfile())
```

Examine request
```
api('api.github.com') %>%
    api(orgs, ropensci, events) %>%
    peep()
```

Features in the works
Paging

- Will be able to handle all paging scenarios, automating paging for the user
```
api('api.github.com') %>%
    api_paging(limit(size = 1000))
```

Retry

- Smart handling of retries after failures
```
api('api.github.com') %>%
    api_retry(n = 5, time = 3)
```

Rate limit

- Smart rate limit handling - choose how to handle reaching rate limits

Swap out backend?

- Use *curl* or *RCurl* instead of *httr* - right now uses httr

Flexible error handling

- Via pkg in dev *sckott/fuaxpaus*

More HTTP verbs

- Only GET and POST right now, add all others, with automatic verb selection given inputs