# Test BacDive server speed

*Katrin Leinweber*

*2018-11-26*

## Generic function to record the dataset download

We will be running a search for some dozens datasets and downloading them through the webservice as `.json` files. From the benchmarked duration of the download in seconds, and the sum of the resulting `.json` files sizes, we can calculate the download speed in kilobytes per second.

This does not take potential compression between the server and the client into account. However: `better compression = shorter download time = higher download speed` in both reality and this test.

*Please note: In order to re-run this analysis, you'll need to have BacDiveR installed and set-up, or replace all `BacDiveR:::get_credentials()` occurances with your own credentials insertion code.*

```
test_speed <- function(IDs, dir) {
  dir.create(dir)

  mb <- microbenchmark::microbenchmark(times = 1,
    for (i in IDs) {
      json <- RCurl::getURL(URLencode(paste0(
        "https://bacdive.dsmz.de/api/bacdive/bacdive_id/", i, "/?format=json"
      )), userpwd = paste(BacDiveR:::get_credentials(), collapse = ":"), httpauth = 1L)
      write(json, file = paste0(dir, "/", i, ".json"))
    }
  )

  sec <- mb$time / 10^9
  kbytes_per_sec <- sum(file.size(list.files(dir, full.names = TRUE))) / 2^10 / sec
  return(kbytes_per_sec)
}
```

## Example 1: Downloading 100 datasets through the default taxon search

The following code from is from `BacDive_webservice_R_script.pdf`, but slightly adapted to write the individual `.json` files to disk, rather than convert them into R-native dataframes.

```
x <-
  RCurl::getURL(URLencode(
  "https://bacdive.dsmz.de/api/bacdive/taxon/Pseudomonas/?format=json"
  ), userpwd = paste(BacDiveR:::get_credentials(), collapse = ":"), httpauth = 1L)

xx <- jsonlite::fromJSON(x)
STR <- strsplit(unlist(xx$results), "/")
IDs <- as.numeric(sapply(STR,function(x) x[7]))

test_speed(IDs, dir = "BD_JSONs_taxon")
#> [1] 20.05327
```

## Example 2: Downloading through the advanced search interface

An advanced search for "Archaea" yields 605 datasets, see URL below. BacDiveR's code to download those is in `retrieve_search_results.R` and slightly adapted here:

```
queryURL <- "https://bacdive.dsmz.de/advsearch?advsearch=search&site=advsearch&searchparams[70][content
DL_param <- "&csv_bacdive_ids_advsearch=download"

# ensure that file with result IDs gets downloaded
if (!grepl(pattern = paste0(DL_param, "$"), x = queryURL)) {
  queryURL <- paste0(queryURL, DL_param)
}

cred <- BacDiveR:::get_credentials()
payload <- httr::GET(queryURL, httr::authenticate(cred[1], cred[2]))
payload <- httr::content(payload, as = "text", encoding = "UTF-8")
IDs <- strsplit(x = payload, split = "\\n")[[1]]

test_speed(IDs, dir = "BD_JSONs_adv")
#> [1] 39.01389
```

A `Warning in dir.create ... already exists` does not matter, because existing files are overwritten. Thus, the `file.size` extraction operators on exactly what was downloaded.

## Small client-side influence

Using `profvis` on the code the `test_speed()` calls will show that only a minority of the processing time is related to the different implementation details between the original code and BacDiveR's. The majority is `RCurl::getURL`, i.e. the download.