**What is a bad URL?**

We tackled a few URL issues:

* Some internal and external links were broken, but we did not know  
  which ones.
* A few links were short links ([bit.ly/blabla](http://bit.ly/blabla)) whereas it’s best to  
  store the actual link because the short link could break too.
* Some links were http links, although the same https link might work  
  and would be preferred over http for security.

**Please read if you try this at “home/”**

There are three main ingredients to our website spring/fall cleaning: R  
tools, elbow grease and *version control*! Most changes happened in a  
branch, and although one can’t possibly look in detail at a diff of more  
than one hundred files, we tried to be as careful as possible.

**From absolute to relative links**

To remove the absolute links, we resorted to using regular expressions.

library("magrittr")

# Identify the Markdown files to be examined

mds <- fs::dir\_ls("content", recurse = TRUE, glob = "\*.md")

mds <- mds[!grepl("\\/tutorials\\/", mds)]

# Function to fix each file if needed

fix\_ropensci <- function(filepath){

readLines(filepath) -> text

# We only edit files that had the issue

if (any(grepl("http(s)?\\:\\/\\/ropensci\\.org\\/", text))){

text %>%

stringr::str\_replace\_all("http(s)?\\:\\/\\/ropensci\\.org\\/", "/") %>%

writeLines(filepath)

}

}

purrr::walk(mds, fix\_ropensci)

Voilà!

**Broken URLs**

Now, what about the links that do not link to anything? We started by  
extracting all links together with the relevant file paths.

library("magrittr")

website\_source <- "/home/maelle/Documents/ropensci/roweb2"

mds <- fs::dir\_ls(website\_source, recurse = TRUE, glob = "\*.md")

mds <- mds[!grepl("\\/tutorials\\/", mds)]

get\_links <- function(filepath){

readLines(filepath) %>%

glue::glue\_collapse(sep = "\n") %>%

commonmark::markdown\_html(normalize = TRUE,

extensions = TRUE) %>%

xml2::read\_html() %>%

xml2::xml\_find\_all("//a") %>%

xml2::xml\_attr("href") -> urls

tibble::tibble(filepath = filepath,

url = urls)

}

all\_urls <- purrr::map\_df(mds, get\_links)

all\_urls <- all\_urls %>%

dplyr::mutate(url = stringr::str\_remove\_all(url, "#.\*"),

url = stringr::str\_remove(url, "\\/$"))

all\_urls

## # A tibble: 14,234 x 2

## filepath url

##

## 1 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://adamhsparks.netl>…

## 2 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://aldocompagnoni.w>…

## 3 /home/maelle/Documents/ropensci/roweb2/content/aut… <http://robitalec.ca>

## 4 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://alison.rbind.io>

## 5 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://dobb.ae>

## 6 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://thestudyofthehou>…

## 7 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://annakrystalli.me>

## 8 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://paleantology.com>…

## 9 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://aurielfournier.g>…

## 10 /home/maelle/Documents/ropensci/roweb2/content/aut… <https://faculty.washingt>…

## # … with 14,224 more rows

We chose a different method to find those within and outside of our  
website.

**Broken internal URLs**

We generated the sitemap from within the website folder to extract  
links.

cwd <- getwd()

setwd(website\_source)

p <- processx::process$new("hugo", args = "server", echo = TRUE)

## Running hugo server

Sys.sleep(120)

localhost <- "<http://localhost:1313>"

browseURL(localhost)

paste0(localhost, "/sitemap.xml") %>%

xml2::read\_xml() %>%

xml2::xml\_ns\_strip() %>%

xml2::xml\_find\_all("//loc") %>%

xml2::xml\_text() %>%

stringr::str\_remove\_all(localhost) %>%

stringr::str\_remove("\\/$") -> links

p$kill()

## [1] TRUE

setwd(cwd)

head(links)

## [1] "/authors/scott-chamberlain" "/tags/api"

## [3] "/authors" "/tags/http"

## [5] "/technotes/2019/12/11/http-testing" "/tags/mocking"

Let’s now extract the internal links we used in the content.

all\_urls %>%

dplyr::filter(!grepl("^http", url)) ->

internal\_urls

head(internal\_urls)

## # A tibble: 6 x 2

## filepath url

##

## 1 /home/maelle/Documents/ropensci/roweb2/content/blog… /community

## 2 /home/maelle/Documents/ropensci/roweb2/content/blog… /community

## 3 /home/maelle/Documents/ropensci/roweb2/content/blog… /blog/2013/05/10/introdu…

## 4 /home/maelle/Documents/ropensci/roweb2/content/blog… /about

## 5 /home/maelle/Documents/ropensci/roweb2/content/blog… /community

## 6 /home/maelle/Documents/ropensci/roweb2/content/blog… /contact

internal\_urls %>%

dplyr::filter(!url %in% links)

**Broken external URLs**

To identify broken external URLs, we ran  
crul::ok() on all  
of them and created a big spreadsheet of URLs to look at.

external\_urls <- dplyr::anti\_join(all\_urls, internal\_urls,

by = c("filepath", "url"))

unique\_urls <- unique(external\_urls[, "url"])

ok <- memoise::memoise(

ratelimitr::limit\_rate(crul::ok,

ratelimitr::rate(1, 1)))

get\_ok <- function(url){

message(url)

ok(url)

}

unique\_urls <- unique\_urls %>%

dplyr::group\_by(url) %>%

dplyr::summarise(ok = get\_ok(url))

external\_urls <- dplyr::left\_join(external\_urls, unique\_urls,

by = "url")

external\_urls <- dplyr::arrange(external\_urls, url)

parse\_one\_post <- function(path){

if (grepl("\\\_index", path)){

return(NULL)

}

lines <- suppressWarnings(readLines(path, encoding = "UTF-8"))

yaml <- blogdown:::split\_yaml\_body(lines)$yaml

yaml <- glue::glue\_collapse(yaml, sep = "\n")

yaml <- yaml::yaml.load(yaml)

meta <- tibble::tibble(date = anytime::anydate(yaml$date),

author = yaml$authors,

title = yaml$title,

software\_peer\_review = "Software Peer Review" %in% yaml$tags,

type = dplyr::if\_else(grepl("\\/blog\\/", path),

"blog post", "tech note"),

filepath = path)

meta

}

info <- purrr::map\_df(mds[grepl("blog", mds)|grepl("technotes",mds)], parse\_one\_post)

info <- dplyr::group\_by(info, filepath) %>%

dplyr::summarize(date = date[1],

author = toString(author),

title = title[1],

type = type[1])

bad\_urls <- dplyr::filter(external\_urls, !ok)

bad\_urls <- dplyr::left\_join(bad\_urls, info, by = "filepath")

readr::write\_csv(bad\_urls, "urls.csv")

From that spreadsheet hundreds of links were examined **manually**! When  
there was a replacement link, we used it thanks to a code looping over  
all links. For the about 50 links without replacement, we amended the  
posts by hand to make sure to take context into account (e.g. removing  
the link vs. removing the whole sentence presenting it).

Sometimes you’ll get an error for the HEAD request but not the GET  
request.

# use get verb instead of head

crul::ok("<http://animalnexus.ca>")

## [1] FALSE

crul::ok("<http://animalnexus.ca>", verb = "get")

## [1] TRUE

Sometimes you’ll need an user-agent whose name does not contain “curl”,  
which the default user-agent of crul contains (crul:::make\_ua() is  
libcurl/7.58.0 r-curl/4.3 crul/0.9.1.9991).

# some urls will require a different useragent string

# they probably regex the useragent string

crul::ok("<https://doi.org/10.1093/chemse/bjq042>")

## GnuTLS recv error (-54): Error in the pull function.

## [1] FALSE

crul::ok("<https://doi.org/10.1093/chemse/bjq042>", verb = "get", useragent = "foobar")

## [1] TRUE

**From short to long links**

We only identified short links using the [bit.ly](http://bit.ly) service. We found the  
corresponding link by running the function below. There were actually  
only 4 short links so that was quick.

get\_long <- function(url){

crul::HttpClient$new(url)$get()$url

}

get\_long("<http://bit.ly/2JfrzmE>")

## [1] "<https://www.timeanddate.com/worldclock/fixedtime.html?msg=rOpenSci+Community+Call+on+Reproducible+Research+with+R&iso=20190730T09&p1=791&ah=1>"

**http vs https**

We proceeded as previously when checking external links, except we used  
better settings for crul::ok().

http <- dplyr::filter(all\_urls, grepl("http\\:", url))

http <- dplyr::mutate(http, https = sub("http\\:", "https:", url))

unique\_urls <- unique(http[, "https"])

ok <- memoise::memoise(

ratelimitr::limit\_rate(crul::ok,

ratelimitr::rate(1, 1)))

get\_ok <- function(url){

message(url)

ok(url, verb = "get", useragent = "Maëlle Salmon checking links")

}

unique\_urls <- unique\_urls %>%

dplyr::group\_by(https) %>%

dplyr::summarise(ok = get\_ok(https))

http <- dplyr::left\_join(http, unique\_urls, by = "https")

httpsok <- dplyr::filter(http, ok)

modify\_url <- function(index, df = httpsok) {

row <- df[index,]

readLines(row$filepath) %>%

stringr::str\_replace\_all(row$url, row$https) %>%

writeLines(row$filepath)

}

purrr::walk(seq\_len(nrow(httpsok)), modify\_url)

dotgithub <- dplyr::filter(all\_urls, urltools::domain(url) == "[ropensci.github.io](http://ropensci.github.io)")

make\_docs\_url <- function(url, ropensci\_pkgs = ropensci\_pkgs) {

message(url)

newurl <- url

urltools::domain(newurl) <- "[docs.ropensci.org](http://docs.ropensci.org)"

if (crul::ok(newurl, verb = "get", useragent = "Maëlle Salmon checking links")) {

return(newurl)

} else {

return(url)

}

}

dotgithub <- dotgithub %>%

dplyr::group\_by(url) %>%

dplyr::mutate(newurl = make\_docs\_url(url))

modify\_url <- function(index, df = dotgithub) {

row <- df[index,]

readLines(row$filepath) %>%

stringr::str\_replace\_all(row$url, row$newurl) %>%

writeLines(row$filepath)

}

purrr::walk(seq\_len(nrow(dotgithub)), modify\_url)

**Conclusion**

In this tech note we saw how to use a combination of regular  
expressions, commonmark, xml2 and crul to identify links to be fixed in  
Markdown content.