In this new post, we're taking a break from modern birding data in [our  
birder's series](https://ropensci.org/tags/birder)â€¦ let's explore  
gorgeous drawings from a natural history collection! Armed with  
rOpenSci's packages binding powerful C++ libraries and open taxonomy  
data, how much information can we automatically extract from images?  
Maybe not much, but we'll at least have explored image manipulation,  
optical character recognition (OCR), language detection, taxonomic name  
resolution with rOpenSci's packages.

**Free natural history images and appropriate R tooling!**

A long time ago I had bookmarked the [Flickr account of the Biodiversity  
Heritage Library (BHL)](https://www.flickr.com/people/biodivlibrary/).  
So many beautiful images of biodiversity, moreover free to use! In  
particular, I [downloaded all pictures from one of the Birds of  
Australia albums](https://www.flickr.com/photos/biodivlibrary/sets/72157694191194992).

I wanted to try to extract the bird names from images using packages of  
Jeroen Ooms', rOpenSci's post-doc hacker & C(++)-bindings wizard. For  
that I worked with magick for  
image manipulation, tesseract  
for optical character recognition (OCR),  
cld2/cld3  
for language detectionâ€¦ Quite the armory!

**OCR bird naming workflow, piece by piece**

In this section, we explain the different elements of our R workflow:  
preparing images, extracting text, resolving taxonomic names.

**Image preparation**

I saved the pictures locally in a "birdsâ€� folder. Yes, I click-buttoned  
instead of using [the Flickr API](https://www.flickr.com/services/api/)  
for which I Write a minimal R  
wrapperâ€¦ I don't do *everything*  
with R scripts (yet).

Minimal wrapper

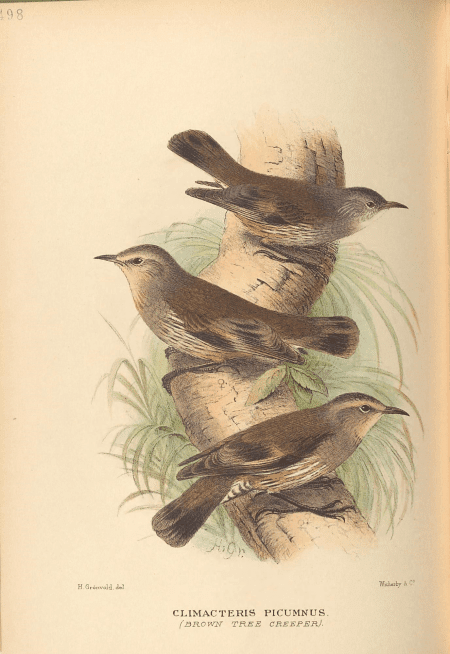
|  |
| --- |
| the <- new.env(parent = emptyenv()) |
|  |  |
|  | the$url <- "https://api.flickr.com/services/rest" |
|  | the$key <- "ed724990662ac2b8c236726a34226ea9" |
|  | the$secret <- "6f53c97ac97a0d78" |
|  |  |
|  | #' Authenticate a session |
|  | #' |
|  | #' This will spawn a browser window to allow the user to accept the |
|  | #' authentication. |
|  | #' @param permission The desired permission for the session. |
|  | #' @param key The API key, a default one is provided, but please register a new |
|  | #' one yourself if you will be doing many requests. |
|  | #' @param secret The API secret. |
|  | #' @export |
|  | authenticate <- function(permission = c("read", "write", "delete"), |
|  | key = the$key, secret = the$secret) { |
|  |  |
|  | permission <- match.arg(permission) |
|  |  |
|  | app <- httr::oauth\_app("R to flickr", key, secret) |
|  |  |
|  | endpoint <- httr::oauth\_endpoint( |
|  | request = "https://www.flickr.com/services/oauth/request\_token", |
|  | authorize = "https://www.flickr.com/services/oauth/authorize", |
|  | access = "https://www.flickr.com/services/oauth/access\_token") |
|  |  |
|  | the$token <- httr::oauth1.0\_token(endpoint, app) |
|  | invisible() |
|  | } |
|  |  |
|  | flickr\_VERB <- function(verb) { |
|  | function(method, ...) { |
|  | if (is.null(the$token)) { |
|  | stop("First authenticate with `authenticate()`", call. = FALSE) |
|  | } |
|  |  |
|  | query <- list(...) |
|  |  |
|  | signature <- httr::oauth\_signature(the$url, |
|  | method = verb, |
|  | app = the$token$app, |
|  | token = the$token$credentials$oauth\_token, |
|  | token\_secret = the$token$credentials$oauth\_token\_secret, |
|  | other\_params = c(list(method = method, format = "json", "nojsoncallback" = 1), query)) |
|  |  |
|  | response <- httr::VERB(verb, the$url, query = signature, query = query) |
|  |  |
|  | heads <- httr::headers(response) |
|  |  |
|  | if (httr::status\_code(response) >= 300) { |
|  | cond <- structure(list( |
|  | call = sys.call(-1), |
|  | content = httr::content(response), |
|  | headers = httr::headers(response), |
|  | message = paste0("flickr API error:", heads$status)), |
|  | class = c("condition", "error")) |
|  | stop(cond) |
|  | } |
|  | httr::content(response) |
|  | } |
|  | } |
|  |  |
|  | #' Perform a flickr request |
|  | #' |
|  | #' \code{flickr\_GET} performs a GET request, \code{flickr\_POST} performs a POST |
|  | #' request. Use \code{\link{authenticate}} to authenticate your session prior |
|  | #' to connecting. |
|  | #' @param method The flickr API method to use \url{https://www.flickr.com/services/api/}. |
|  | #' @param ... additional arguments passed to \code{\link[httr]{VERB}}. |
|  | #' @rdname flickr\_request |
|  | #' @examples |
|  | #' \dontrun{ |
|  | #' flickr\_GET("flickr.photo.search", text = "UseR2016") |
|  | #' } |
|  | #' @export |
|  | flickr\_GET <- flickr\_VERB("GET") |
|  |  |
|  | #' @rdname flickr\_request |
|  | #' @export |
|  | flickr\_POST <- flickr\_VERB("POST") |

library("magrittr")

filenames <- fs::dir\_ls("birds")

Each image shows a bird and its name. Images are either landscape- or  
portrait-oriented, but for the sake of simplicity, I'll act as if they  
were all portrait-oriented. *A possible easy and lazy fix here would be  
to duplicate images rotated (magick::image\_rotate) in all possible  
directions and then to apply the workflow to all 4 versions of each  
image, hoping to get good data from one of the rotated versions.*

magick::image\_read(filenames[1])



From that image, I wanted to extract the name indicated below the bird.  
To maximize the efficiency of OCR, I shall first prepare the image,  
since the accuracy of OCR depends on the quality of the input  
image which can be influenced a bit. This part could be tweaked even more, and  
in real life examples it'll be worth spending time trying different  
magick functions and parameter values. Since I have in mind the case  
of a bunch of images to be batch-processed, nothing is done by hand.

crop\_bird <- function(filename){

image <- magick::image\_read(filename)

height <- magick::image\_info(image)$height

# crop the top of the image

image <- magick::image\_crop(image,

paste0("+0+",round(0.75\*height))) %>%

# convert the image to black and white

magick::image\_convert(type = "grayscale") %>%

# increase brightness

magick::image\_modulate(brightness = 120) %>%

magick::image\_enhance() %>%

magick::image\_median() %>%

magick::image\_contrast()

# we'll need the filename later

attr(image, "filename") <- filename

return(image)

}

crop\_bird(filenames[1])



It does look cleaner now!

**Text extraction**

Now is the actual OCR step! The tesseract package provides bindings to  
the [Tesseract OCR  
engine](https://en.wikipedia.org/wiki/Tesseract_(software)), free  
software currently sponsored by Google. It is a powerful engine, with a  
ton of parameters. Here again, tweaking a lot is warranted. Particularly  
useful reads are tesseract vignette..  
The hocr package might be of  
interest for post-processing of OCR results.

One can use either tesseract::ocr for a file path, url, or raw vector  
to image, or magick::image\_ocr for a magick object which is quite  
handy in our pipeline.

The function below also filters results of the OCR using language  
detection. By only keeping text recognized as either Latin or English by  
one of the two language detection packages cld2 and cld3 that are  
interfaces to Google compact language detectors 2 and 3, one gets a  
first quality filter. If not doing that, the output would contain more  
unusable text.

get\_names <- function(image){

filename <- attr(image, "filename")

ocr\_options <- list(tessedit\_pageseg\_mode = 1)

text <- magick::image\_ocr(image, options = ocr\_options)

text <- stringr::str\_split(text, "\n", simplify = TRUE)

text <- stringr::str\_remove\_all(text, "[0-9]")

text <- stringr::str\_remove\_all(text, "[:punct:]")

text <- trimws(text)

text <- stringr::str\_remove\_all(text, "~")

text <- text[text != ""]

text <- tolower(text)

# remove one letter words

# https://stackoverflow.com/questions/31203843/r-find-and-remove-all-one-to-two-letter-words

text <- stringr::str\_remove\_all(text, " \*\\b[[:alpha:]]{1,2}\\b \*")

text <- text[text != ""]

# keep only the words that are recognized as either Latin

# or English by cld2 or cld3

if(length(text) > 0){

results <- tibble::tibble(text = text,

cld2 = cld2::detect\_language(text),

cld3 = cld3::detect\_language(text),

filename = filename)

results[results $cld2 %in% c("la", "en") |

results$cld3 %in% c("la", "en"),]

}else{

return(NULL)

}

}

(results1 <- filenames[1] %>%

magick::image\_read() %>%

get\_names())

## NULL

Nothing! Now if we replace magick::image\_read with the previously  
defined crop\_bird function that crops and cleans the imageâ€¦

(results2 <- filenames[1] %>%

crop\_bird() %>%

get\_names())

## # A tibble: 2 x 4

## text cld2 cld3 filename

##

## 1 climacteris picumnus la birds/n115\_w1150\_42399797481\_o.jpg

## 2 brown tree creeper en birds/n115\_w1150\_42399797481\_o.jpg

We get a result! So we see that the image transformation was quite  
useful.

Now, these names look fine, but how to be sure they're actually  
taxonomic names?

**Taxonomic name resolution**

The taxize package is a taxonomic toolbelt for R, providing access to many  
fantastic data sources and tools for taxonomy. One of them, the Global  
Name Resolver, provides, well, resolution of taxonomic names, sadly not  
common names. taxize::gnr\_resolve has many options, of which only one  
is used below: best\_match\_only = TRUE means it'll only return the best  
match from the different data sources.

latin <- results2$text[results2$cld2 == "la"|

results2$cld3 == "la"]

taxize::gnr\_resolve(latin,

best\_match\_only = TRUE)

## # A tibble: 1 x 5

## user\_supplied\_name submitted\_name matched\_name data\_source\_tit~ score

## \*

## 1 climacteris picum~ Climacteris pi~ Climacteris p~ NCBI 0.988

English names could be cleaned up a bit using the spelling  
package, also an rOpenSci package, since spelling::spell\_check\_text would output possible typos.

**OCR bird naming workflow in action!**

First the two steps image processing and OCR are applied to all images.

bird\_names <- purrr::map(filenames, crop\_bird) %>%

purrr::map\_df(get\_names)

Out of 51 images only 17 are present in the final table with possible  
names which is a bit disheartening, but one could surely do better in  
the image processing and OCR tweaking steps! Please  
also keep in mind that such natural history images are quite hard to  
parse.

The name resolution information can be added to this table.

safe\_resolve <- function(text){

results <- taxize::gnr\_resolve(text,

best\_match\_only = TRUE)

if(nrow(results) == 0){

list(NULL)

}else{

list(results)

}

}

bird\_names <- dplyr::group\_by(bird\_names, text) %>%

dplyr::mutate(gnr = ifelse(cld2 == "la" | cld3 == "la",

safe\_resolve(text),

list(NULL)))

We do not get much resolution, but we knew the names weren't very good  
to start with. A better (untested here!) idea here might be to get a  
full list of names of Australian birds, potentially leveraging the  
taxizedb package , and to then  
fuzzy-match them with the names we have.

unique(bird\_names$gnr)

## [[1]]

## # A tibble: 1 x 5

## user\_supplied\_name submitted\_name matched\_name data\_source\_tit~ score

## \*

## 1 climacteris picum~ Climacteris pi~ Climacteris p~ NCBI 0.988

##

## [[2]]

## [1] NA

##

## [[3]]

## NULL

##

## [[4]]

## # A tibble: 1 x 5

## user\_supplied\_na~ submitted\_name matched\_name data\_source\_title score

## \*

## 1 austrodicaeum ii~ Austrodicaeum ~ Austrodicaeu~ The Interim Regis~ 0.75

##

## [[5]]

## # A tibble: 1 x 5

## user\_supplied\_name submitted\_name matched\_name data\_source\_tit~ score

## \*

## 1 melithreptus laet~ Melithreptus l~ Melithreptus ~ CU\*STAR 0.988

##

## [[6]]

## # A tibble: 1 x 5

## user\_supplied\_na~ submitted\_name matched\_name data\_source\_title score

## \*

## 1 rad isdlvorniode Rad isdlvorni~ Rad Baker & ~ The Interim Regist~ 0.75

**More data from the Biodiversity Heritage Library**

If you're interested in other types of data from the BHL, in addition to  
the images, have a look at the rbhl  
paclage, part of rOpenSci's suite,  
that interacts with the BHL API. One can e.g.Â search all books by the  
same author as the one we used images from.

author <- rbhl::bhl\_authorsearch("Gregory M Mathews")

books <- rbhl::bhl\_getauthortitles(creatorid = author$CreatorID)

head(books$FullTitle)

## [1] "A manual of the birds of Australia,"

## [2] "A list of the birds of the Phillipian sub-region : which do not occur in Australia. "

## [3] "A manual of the birds of Australia /"

## [4] "A list of the birds of Australia : containing the names and synonyms connected with each genus, species, and subspecies of birds found in Australia, at present known to the author /"

## [5] "Austral avian record; a scientific journal devoted primarily to the study of the Australian avifauna."

## [6] "Arcana, or, The museum of natural history : containing the most recent discovered objects : embellished with coloured plates, and corresponding descriptions : with extracts relating to animals, and remarks of celebrated travellers; combining a general survey of nature."

Or we could get all books whose title contains the words "birdsâ€� and  
"australiaâ€�.

australia\_birds <- rbhl::bhl\_booksearch(title = "birds Australia")

head(australia\_birds$FullTitle)

## [1] "Handbook to the birds of Australia. : [Supplementary material in Charles Darwin's copy]."

## [2] "An introduction to The birds of Australia /"

## [3] "The useful birds of southern Australia : with notes on other birds /"

## [4] "The Birds of Australia"

## [5] "The birds of Australia,"

## [6] "The birds of Australia,"

And to get the OCR results of the pages of the book we used, we could  
write:

library("magrittr")

# ocr=TRUE to extract OCR for all pages

rbhl::bhl\_getitempages("250938", ocr = TRUE) %>%

# for each page transform the type into a string

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

dplyr::mutate(page\_type = toString(PageTypes[[1]])) %>%

# keep only the illustration pages

# that are like the ones we used

dplyr::filter(page\_type == "Illustration") %>%

# from the data.frame extract the OCR

dplyr::pull(OcrText) %>%

head()

## [1] "491 \nFAL CUNCULUS LEUCOGASTER. \n( WHITE -BELLIED Â£ If BIKE - TIT) \nFALCUNCULUS FRONTATUS. \nSHRIKE - TIT). \n"

## [2] "492 \nA\*\* \nOREOICA GUTTURALIS. \n(CRESTED BELL-BIRD). \n"

## [3] "APHELOCEPHALA LEUCOPSIS \n( WHITE FACE ). \n"

## [4] "\* \nAPHELOCEPHALA PE CTORALIS. \n(CHE <3 TNUT -BREASTED WHITEFA CEj. \nAPHELOCEPHALA NIGRICINCTA. \n(BE A CK-BAH.DED WHITE FA CEj. \n"

## [5] "H . Gronvold. del. \nWitherLy & CÂ° \nSPHENOSTOMA CRIS TATUM \n(WEDGE BIEL). \n"

## [6] "49 6 \nH \n(jronvolcl. del. \nN E O SIT TA LE CJ C O CE PHAI.A. \n( WHITE Â¦ HE AID EE THE EH UN HE FL). \nNEOSHTA ALBATA \n(F IE E> T Ft E EE UNNEFlj. \nNEOSITTA CHRYSOPTERA \nf OFi. A. NGE - wing-e d tree runner). \nWitWLjA \n"

So there's quite a lot to explore!