O'Reilly, has now based its brand on covers featuring  
beautiful gravures of animals.

Recently, while wondering what the name of R for Data Science bird was  
again (I thought it was a kea!), I was thrilled to find [the whole  
O'Reilly menagerie](https://www.oreilly.com/animals.csp), i.e.Â a list of  
books and corresponding animals! The website also features a link to ["A  
short history of the O'Reilly animalsâ€�](https://www.oreilly.com/ideas/a-short-history-of-the-oreilly-animals)  
that was an amazing read. In it was noted that "The animals are in  
trouble.â€�, with a few examples of endangered species. It inspired me to  
actually try and assess the conservation status of O'Reilly animals  
using responsible webscraping, taxonomic name resolving and IUCN Redlist  
API queryingâ€¦

**Scraping the menagerie: an utter delight!**

I had a great time webscraping the menagerie, not only thanks to my now  
reasonable experience doing such things, but also thanks to

* my using the wonderful polite package for  
  webscraping, that makes me feel  
  so good about myself. Read more about this package below

library("magrittr")

polite::scrape(session, params = "?A0=ALLSTAT") %>%

rvest::xml\_nodes("li") %>%

rvest::xml\_nodes("a") %>%

rvest::html\_attr("href") %>%

purrr::keep(function(x) stringr::str\_detect(x, "\\/cgi-bin\\/webadmin\\?A1\\=")) %>%

stringr::str\_remove("\\/cgi\\-bin\\/webadmin\\?A1\\=ind") %>%

stringr::str\_remove("\\&L\\=ALLSTAT") -> date\_strings

This is not very elegant but this got me only the “1807” and such I needed for the rest of the scraping. polite::scrape is a wrapper to both httr::GET and httr::content and does the rate limiting (by default 1 call every 5 seconds, delay parameter of polite::bow) and memoising. I actually ended up only scraping emails metadata from 2007 because it took ages to parse the 2006 page.

date\_strings <- date\_strings[stringr::str\_length(date\_strings) != 2]

# or

date\_strings <- purrr::discard(date\_strings, function(x) stringr::str\_length(x) == 2)

I created a function getting the metadata out of each archive page. The trickiest points here were:

* That the rows of the archive table could have two classes, which is the way alternate coloring was obtained. I therefore used | in XPath '//tr[@class="normalgroup"]|//tr[@class="emphasizedgroup"]'.
* That there was no different class/formatting for subject, date, sender, so I got all of them at once, and then used the modulo operator, %%, to assign them to the right vector.

get\_emails\_meta\_by\_date <- function(date\_string, session){

message(date\_string)

params <- glue::glue("?A1=ind{date\_string}&L=ALLSTAT&F=&S=&O=T&H=0&D=0&T=0")

everything <- try(polite::scrape(session, params = params),

silent = TRUE)

# at the time of writing one couldn't pass encoding to scrape

# but now one can https://github.com/dmi3kno/polite/issues/6#issuecomment-409268730

if(is(everything, "try-error")){

everything <- httr::GET(paste0(home\_url,

params)) %>%

httr::content(encoding = "latin1")

}

everything <- everything %>%

# there are two classes that correspond

# to the table having two colours of rows!

rvest::xml\_nodes(XPath = '//tr[@class="normalgroup"]|//tr[@class="emphasizedgroup"]') %>%

rvest::xml\_nodes("span")

everything %>%

rvest::xml\_nodes(XPath = "//td") %>%

rvest::xml\_nodes("span") %>%

rvest::xml\_nodes("a") %>%

rvest::html\_text() -> subjects

everything %>%

rvest::xml\_nodes(XPath = "//td[@nowrap]") %>%

rvest::xml\_nodes(XPath = "p[@class='archive']") %>%

rvest::html\_text() -> big\_mess

senders <- big\_mess[seq\_along(big\_mess) %% 3 == 1]

senders <- stringr::str\_remove(senders, " \\<\\[log in to unmask\\]\\>")

dates <- big\_mess[seq\_along(big\_mess) %% 3 == 2]

dates <- lubridate::dmy\_hms(dates, tz = "UTC")

sizes <- big\_mess[seq\_along(big\_mess) %% 3 == 0]

sizes <- stringr::str\_remove(sizes, " lines")

sizes <- as.numeric(sizes)

tibble::tibble(subject = subjects,

sender = senders,

date = dates,

size = sizes) %>%

readr::write\_csv(glue::glue("data/emails\_meta{date\_string}.csv"))

}

I chose to save the metadata of each archive page in its own csv in order to make my workflow less breakable. I could have used purrr::map\_df but then it’d be harder to re-start, and it was hard on memory apparently.

fs::dir\_create("data")

purrr::walk(date\_strings,

get\_emails\_meta\_by\_date,

session = session)

**Analyzing ALLSTAT jobs**

**Filtering jobs**

ALLSTAT encourages you to use keywords in emails’ subjects, so many job openings contain some variant of “job”, and that’s the sample on which I shall work.

library("magrittr")

library("magrittr")

fs::dir\_ls("../../static/data/allstat") %>%

purrr::map\_df(readr::read\_csv) -> emails

jobs <- dplyr::filter(emails,

stringr::str\_detect(subject,

"[Jj][Oo][Bb]"))

Out of 32761 emails I got 9245 job openings.

I created two dummy variables to indicate the presence of data scientist or statistician in the description. With the definition below, the “statistician” category might contain “biostatitisticians” which is fine by me.

jobs <- dplyr::mutate(jobs,

data\_scientist = stringr::str\_detect(subject,

"[Dd]ata [Ss]cientist"),

statistician = stringr::str\_detect(subject,

"[Ss]tatistician"))

176 subjects contain the word “data scientist”, 2546 the word “statistician.”, 20 both.

dplyr::filter(jobs, data\_scientist, statistician) %>%

dplyr::select(subject, sender, date) %>%

knitr::kable()

Are the job titles synonymous for the organizations using slashes? I am especially puzzled by “Senior Medical statistician / Real-world data scientist”! I filtered them out and created a category variable.

jobs <- dplyr::filter(jobs,

!(data\_scientist&statistician))

jobs <- dplyr::mutate(jobs,

category = dplyr::case\_when(data\_scientist ~ "data scientist",

statistician ~ "statistician",

TRUE ~ "other"),

category = factor(category,

levels = c("statistician",

"data scientist",

"other"),

ordered = TRUE))

jobs <- dplyr::mutate(jobs,

year = lubridate::year(date),

year = as.factor(year))

Here are some examples of positions for each:

head(jobs$subject[jobs$category=="statistician"])

## [1] "FW: JOB: Senior/Lead Statistician - GlaxoSmithKline"

## [2] "JOB - Clinical Trial Statisticians and SAS Programmers"

## [3] "JOB - Medical statistician in Salvador, Brazil (2-3 years)"

## [4] "JOB - Medical Statistician, Oxford"

## [5] "JOB OPPORTUNITY - 39408 - Entry-Level Statisticians - GSK"

## [6] "JOB: Biostatistician - Contract/Consultancy Opportunity - Switzer land"

head(jobs$subject[jobs$category=="data scientist"])

## [1] "JOB: Data Scientist, Unilever"

## [2] "JOB: Data Scientist, Unilever"

## [3] "JOB: Data Scientist / Machine Learning (BELFAST)"

## [4] "JOB: Data Scientists in Belfast"

## [5] "JOBS: Data Scientists"

## [6] "[Job] Hacker or Botnet Developer, Data Scientist (Telecommute)"

head(jobs$subject[jobs$category=="other"])

## [1] "Job - Data manager Dept of Haematology, Imperial College, London"

## [2] "Re: JOB - Health Economic Modelling and Value Demonstrations"

## [3] "JOB - Lectureship / readership"

## [4] "JOB - Mathematical Modeller at the Health Protection Agency, Centre for Infections"

## [5] "JOB - PhD position"

## [6] "JOB - SAS Senior Statistical Programmer, Oxford"

**Are data scientists on the rise?**

library("ggplot2")

ggplot(jobs) +

geom\_bar(aes(year, fill = category)) +

viridis::scale\_fill\_viridis(discrete = TRUE) +

theme(legend.position = "bottom") +

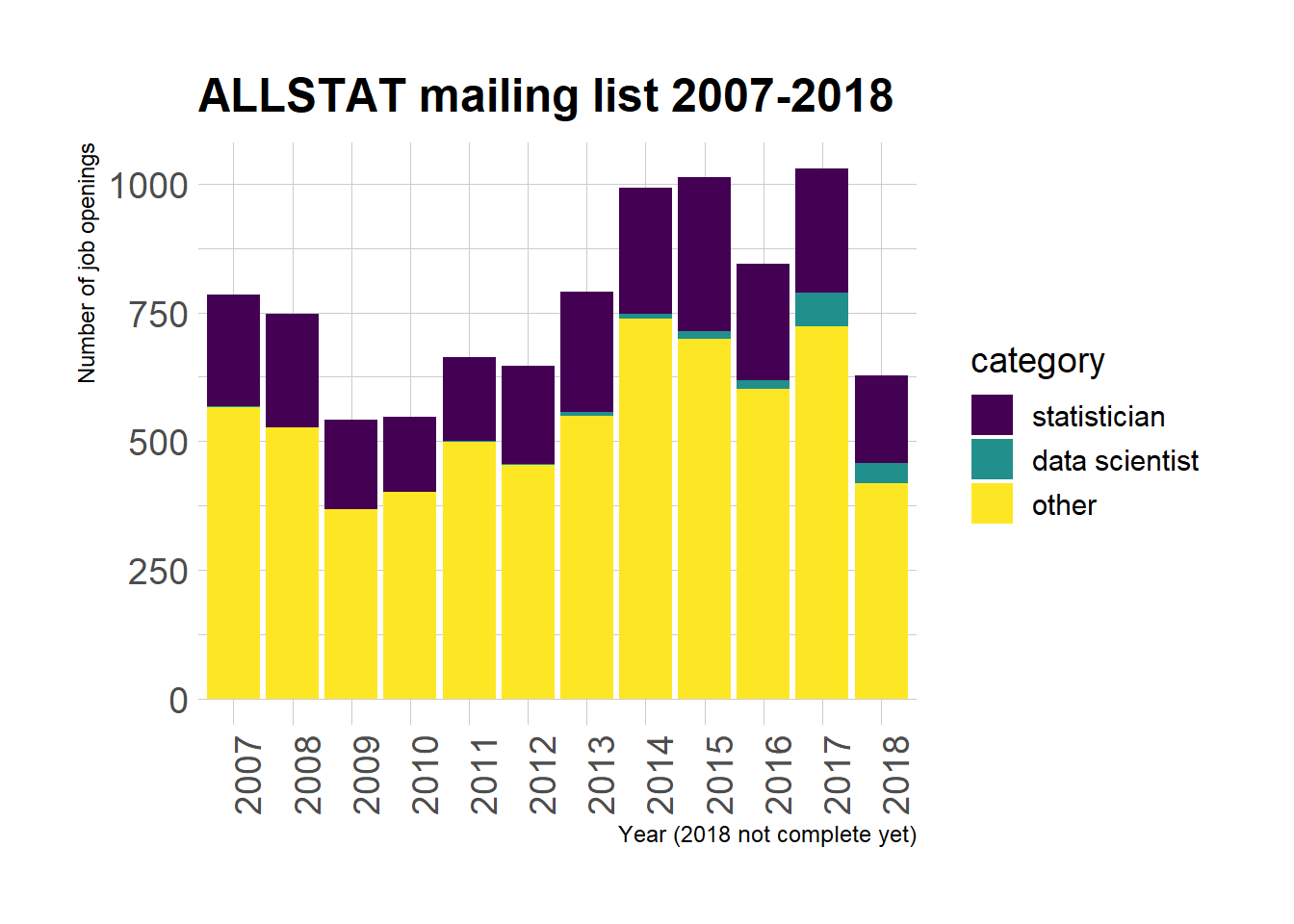
hrbrthemes::theme\_ipsum(base\_size = 14) +

xlab("Year (2018 not complete yet)") +

ylab("Number of job openings") +

theme(axis.text.x = element\_text(angle = 90, hjust = 1)) +

ggtitle("ALLSTAT mailing list 2007-2018")



I like this bar plot that shows how the total number of job openings fluctuates, but it’s hard to see differences in proportions.

ggplot(jobs) +

geom\_bar(aes(year, fill = category),

position = "fill") +

viridis::scale\_fill\_viridis(discrete = TRUE) +

theme(legend.position = "bottom") +

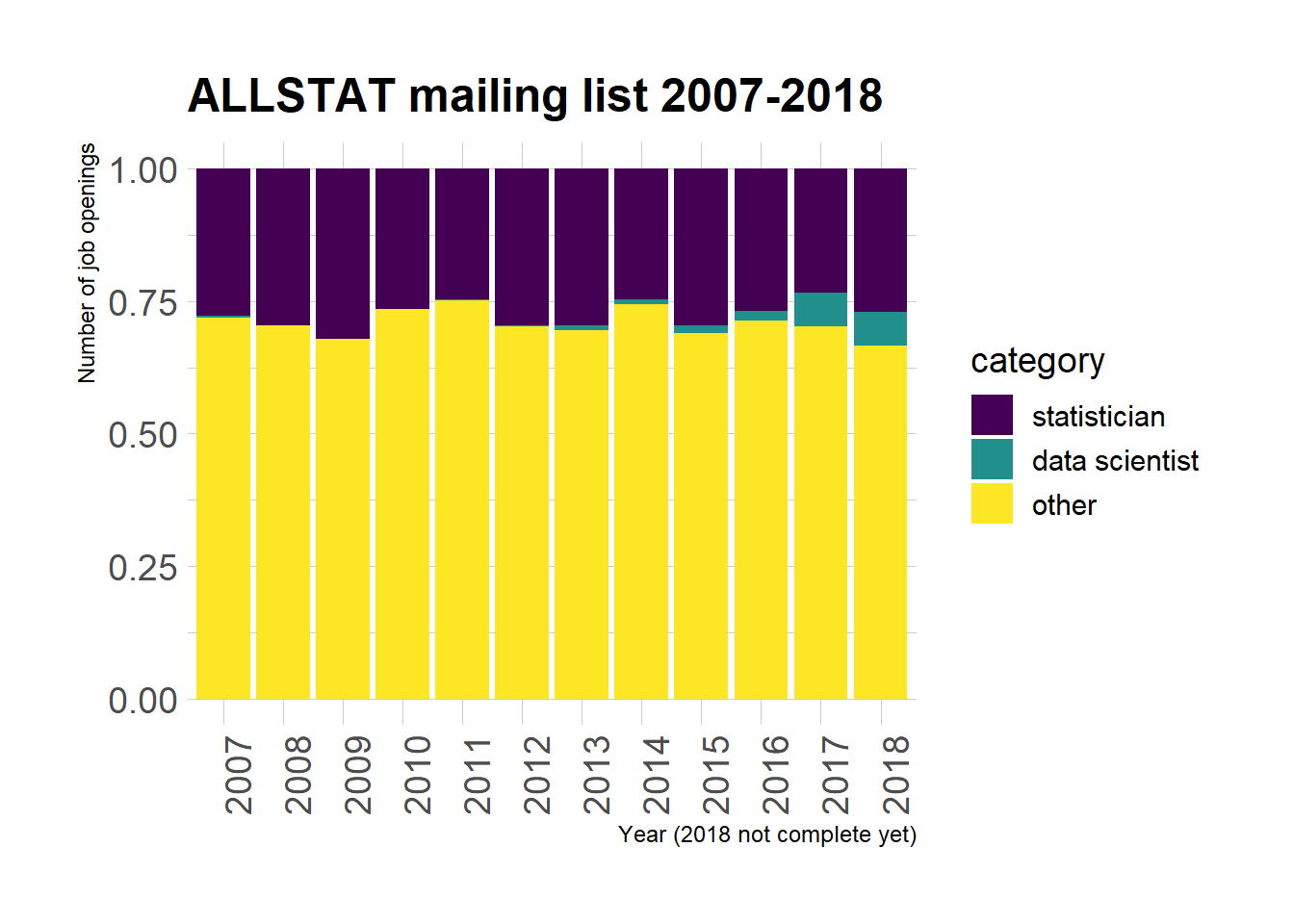
hrbrthemes::theme\_ipsum(base\_size = 14) +

xlab("Year (2018 not complete yet)") +

ylab("Number of job openings") +

theme(axis.text.x = element\_text(angle = 90, hjust = 1))+

ggtitle("ALLSTAT mailing list 2007-2018")



According to this plot, although there seems to be more and more data scientists’ jobs advertised on ALLSTAT… Statisticians don’t need to get worried just yet.

**Who offers data scientists’ jobs?**

dplyr::count(jobs, category,

sender) %>%

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

dplyr::arrange(category, - n) %>%

dplyr::filter(sender %in% sender[1:5])

## # A tibble: 15 x 3

## # Groups: category [3]

## category sender n

## <ord> <chr> <int>

## 1 statistician James Phillips 106

## 2 statistician Sabrina Andresen 82

## 3 statistician James Miller 45

## 4 statistician Angela Smythe 40

## 5 statistician Helena Newman-Mitchell 37

## 6 data scientist James Phillips 86

## 7 data scientist Sportradar HR 5

## 8 data scientist Jason Howlin 4

## 9 data scientist Deborah Gee 3

## 10 data scientist Christos Mitas 2

## 11 other James Phillips 671

## 12 other Angela Smythe 223

## 13 other Helena Newman-Mitchell 103

## 14 other Jason Howlin 91

## 15 other James Miller 85

Seeing James Phillips’ name so often made me have a look at their emails: this person sends emails on the behalf of a website called StatsJobs.com! We can also assume that other super-senders actually work for job aggregators of some sort.

**What are the openings about?**

To make a more thorough description of the different categories, one would need to get the email bodies, which I decided against for this post. I simply used the subjects, and compared word usage between the “data scientist” and “statistician” categories as in this chapter of the Tidy text mining book by Julia Silge and David Robinson.

library("tidytext")

data("stop\_words")

words <- dplyr::filter(jobs, category != "other") %>%

unnest\_tokens(word, subject, token = "words") %>%

dplyr::filter(!word %in% stop\_words$word,

!word %in% c("job", "statistician",

"jobs", "statisticians",

"data", "scientist",

"scientists",

"datascientistjobs"))

word\_ratios <- words %>%

dplyr::count(word, category) %>%

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

dplyr::filter(sum(n) >= 10) %>%

dplyr::ungroup() %>%

tidyr::spread(category, n, fill = 0) %>%

dplyr::mutate\_if(is.numeric, dplyr::funs((. + 1) / sum(. + 1))) %>%

dplyr::mutate(logratio = log(`data scientist` / statistician)) %>%

dplyr::arrange(desc(logratio))

word\_ratios %>%

dplyr::group\_by(logratio < 0) %>%

dplyr::top\_n(15, abs(logratio)) %>%

dplyr::ungroup() %>%

dplyr::mutate(word = reorder(word, logratio)) %>%

ggplot(aes(word, logratio, fill = logratio < 0)) +

geom\_col(show.legend = FALSE) +

coord\_flip() +

ylab("log odds ratio (data scientist / statistician)") +

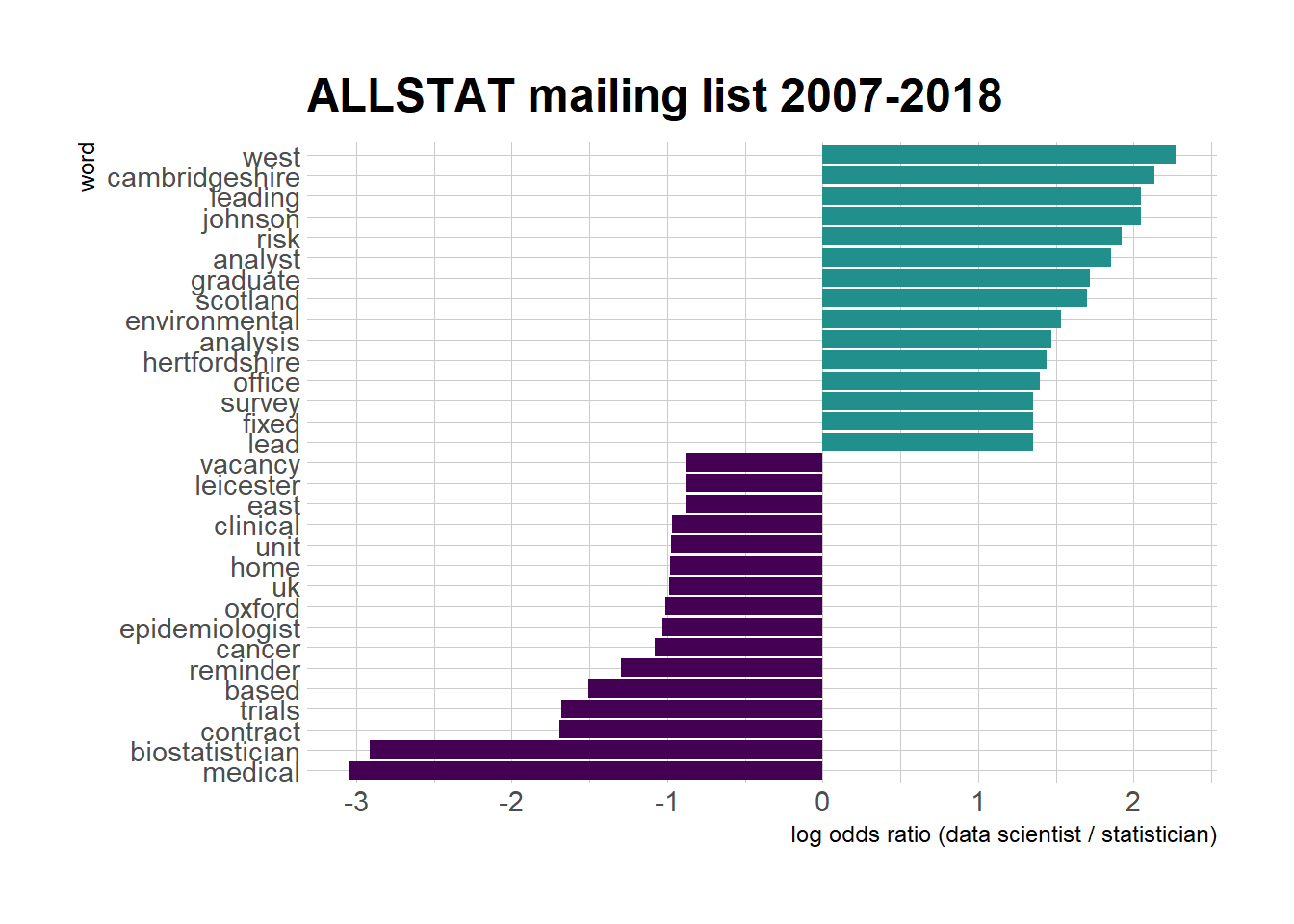
scale\_fill\_manual(name = "",

values = c("#21908CFF", "#440154FF"),

labels = c("data scientist", "statistician")) +

hrbrthemes::theme\_ipsum(base\_size = 11)+

ggtitle("ALLSTAT mailing list 2007-2018")



What does this figure show? On the left are words more frequent in statistician job openings, on the right words more frequent in data scientist job openings. Well, there might be geographical clusters for each of the category, which I don’t believe though: is the data scientist vs. statistician debate a Cambridgeshire vs. Oxford battle? I was surprised that no word related to academia made an appearance because I thought “university” or an equivalent word would be more representative of statisticians. I am less surprised, though, by words such as “trials”, “medical”, “clinical” being more prevalent in the “statistician” category. The word “fixed” as in “fixed-term” contract is more prevalent for data scientist job openings, which doesn’t sound too cool?

* the webpage having really good structured html with specific  
  classes.

The menagerie is divided into pages of 20 books, so I mapped over all  
possible offsets up to the number of animals indicated on the website,  
1227.

library("magrittr")

home\_url <- "https://www.oreilly.com/animals.csp"

session <- polite::bow(home\_url,

user\_agent = "MaÃ«lle Salmon https://masalmon.eu/")

get\_twenty <- function(offset, session){

# offset parameter to get all books 20 by 20

params <- glue::glue("?x-o={offset}")

# scraping with content parameter

# cf https://github.com/dmi3kno/polite/issues/6

# https://www.oreilly.com/animals.csp?x-o=720 was problematic

# (German characters)

page <- polite::scrape(session, params = params,

content = "text/html;charset=iso-8859-1")

# get all animal rows

rows <- rvest::xml\_nodes(page,

xpath = "//div[@class='animal-row']")

# extract book titles

rows %>%

rvest::xml\_nodes(xpath = "a[@class='book']") %>%

rvest::xml\_nodes(xpath = "h1[@class='book-title']") %>%

rvest::html\_text() -> book\_titles

rows %>%

rvest::xml\_nodes(xpath = "h2[@class='animal-name']") %>%

rvest::html\_text() -> animal\_names

tibble::tibble(book = book\_titles,

animal = animal\_names)

}

no\_animals <- 1227 # by hand!

offsets <- (0:floor(no\_animals/20))\*20

purrr::map\_df(offsets, get\_twenty, session = session) %>%

readr::write\_csv("oreilly\_animals.csv")

I got 1134 rows, each corresponding to a book, with animals potentially  
repeated.

animals

## # A tibble: 1,134 x 2

## book animal

##

## 1 Mobile Design and Development 12-Wired Bird of Paradise

## 2 Windows PowerShell for Develop~ 3-Banded Armadillo

## 3 Jakarta Commons Cookbook Aardvark

## 4 Clojure Cookbook Aardwolf

## 5 Ubuntu: Up and Running Addax, aka Screwhorn Antelope

## 6 Social eCommerce Adjutant (Storks)

## 7 BioBuilder Aegina Citrea, narcomedusae, jellyfish

## 8 JRuby Cookbook African Civet

## 9 C# 5.0 Pocket Reference African Crowned Crane aka Grey Crowned~

## 10 Programming C# 5.0 African Crowned Crane aka Grey Crowned~

## # ... with 1,124 more rows

In the short history of animals, Edie Freedman mentions having  
discovered "that there were intriguing correspondences between specific  
technologies and specific animalsâ€�. This made me curious about my last  
name, Salmon!

animals %>%

dplyr::filter(stringr::str\_detect(animal, "[Ss]almon")) %>%

knitr::kable()

| **book** | **animal** |
| --- | --- |
| Values, Units, and Colors | Salmon |
| CSS Text | Salmon |
| CSS Fonts | Salmon |
| Selectors, Specificity, and the Cascade | Salmon |
| Transitions and Animations in CSS | Salmon2 |

I have no idea what trait of salmons make them good at design, other  
than my not sharing that trait with them.

**From animals common names to scientific names?**

Now, you'll have noticed the names of animals are written in English. My  
ultimate goal being the querying of IUCN Red List API, and this API only  
accepting scientific names (contrary to the website of the same  
organization), I needed to resolve the common names to scientific names.  
This is a hard problem! My strategy here was:

* Cleaning the names a bit to remove the parts after "akaâ€� for  
  instance.

clean <- function(animal){

semiclean <- animal %>%

stringr::str\_remove\_all("aka.\*") %>%

stringr::str\_remove\_all("\\,.\*") %>%

stringr::str\_remove\_all("\\(.\*")

if(semiclean == "12-Wired Bird of Paradise"){

semiclean <- "Twelve-Wired Bird of Paradise"

}

if(semiclean == "3-Banded Armadillo"){

semiclean <- "Three-Banded Armadillo"

}

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

}

animals <- dplyr::mutate(animals, animal\_clean = purrr::map\_chr(animal, clean))

* Using the rOpenSci taxize  
  package that has a handy  
  comm2sci function. This function works for anyone, but it's better  
  to request a key for the database used, EOL by default (see e.g.  
  taxize::use\_eol() for more info).
* Not being too optimistic since the databases taxize queries cannot  
  do wonders, no matter how good they are.

Note that for each species, the first scientific name returned is  
selected, because there's no other criterion to go by. That's how I'll  
end up with a Salmon catfish for Salmon, too bad.

animal\_names <- unique(animals$animal\_clean)

# scientific names

good\_comm2sci <- memoise::memoise(taxize::comm2sci)

get\_name <- function(common\_name){

sci\_names <- good\_comm2sci(common\_name)

# don't get the name of who defined the species

sci\_name <- stringr::word(sci\_names[[1]][1], start=1, end = 2)

tibble::tibble(common\_name = common\_name,

sci\_name = sci\_name)

}

scientific\_names <- purrr::map\_df(animal\_names, get\_name)

animals <- dplyr::left\_join(animals,

scientific\_names,

by = c("animal\_clean" = "common\_name"))

I got names for 694 books, out of 1134, getting 555 animals. It's not  
bad, but this number also needs to be treated with caution. See for  
instance:

animals %>%

dplyr::filter(stringr::str\_detect(animal, "Galapagos")) %>%

knitr::kable()

| **book** | **animal** | **animal\_clean** | **sci\_name** |
| --- | --- | --- | --- |
| PHP Cookbook | Galapagos Land Iguana | Galapagos Land Iguana | Conolophus marthae |
| Upgrading to PHP 5 | Galapagos Tortoise | Galapagos Tortoise | Chelonoidis nigra |

I noticed the iguana while perusing my results, and a quick internet  
search taught me that there are *three* species of terrestrial iguanas  
in the Galapagos, the most common one, and the one probably present on  
the book cover, being Conolophus subcristatus, not Conolophus marthae!  
I've noticed a few other mistakes, so I'll need to handle the results  
with care. I now wish the menagerie had a bit more Latin in it!

**Querying the IUCN Red List**

Indeed, scientific names of species are the key to a wealth of data!  
Traits data, taxonomic  
informationâ€¦ and conservation  
status thanks to the [IUCN Red List](http://www.iucnredlist.org/), an  
impressive assessment of species at the global scale. One can  
programmatically query it using the rOpenSci rredlist  
package! That's what I did, adding  
a waiting time of 2 seconds between API calls. Note  
that I have an API key because I asked for it, see more info by typing  
rredlist::rl\_use\_iucn() after installing rredlist, and be patient  
since it can last a few days before one gets one.

slow\_rl\_search <- ratelimitr::limit\_rate(rredlist::rl\_search,

rate = ratelimitr::rate(1, 2))

get\_status <- function(sci\_name){

message(sci\_name)

results <- slow\_rl\_search(sci\_name)$result

if(!is.null(results)){

results$sci\_name <- sci\_name

}

results

}

animals <- dplyr::filter(animals, !is.na(sci\_name))

purrr::map\_df(unique(animals$sci\_name), get\_status) %>%

readr::write\_csv("oreilly\_animals\_status.csv")

status <- readr::read\_csv("oreilly\_animals\_status.csv")

animals <- readr::read\_csv("oreilly\_animals\_scientific.csv")

status <- dplyr::filter(status, !is.na(category))

animals <- dplyr::left\_join(animals, status, by = "sci\_name")

str(animals)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 1134 obs. of 32 variables:

## $ book : chr "Mobile Design and Development" "Windows PowerShell for Developers" "Jakarta Commons Cookbook" "Clojure Cookbook" ...

## $ animal : chr "12-Wired Bird of Paradise" "3-Banded Armadillo" "Aardvark" "Aardwolf" ...

## $ animal\_clean : chr "Twelve-Wired Bird of Paradise" "Three-Banded Armadillo" "Aardvark" "Aardwolf" ...

## $ sci\_name : chr "Seleucidis melanoleuca" "Tolypeutes tricinctus" "Cucumis humifructus" "Proteles cristata" ...

## $ taxonid : int NA 21975 NA 18372 NA 22697721 NA 41589 22692046 22692046 ...

## $ scientific\_name : chr NA "Tolypeutes tricinctus" NA "Proteles cristata" ...

## $ kingdom : chr NA "ANIMALIA" NA "ANIMALIA" ...

## $ phylum : chr NA "CHORDATA" NA "CHORDATA" ...

## $ class : chr NA "MAMMALIA" NA "MAMMALIA" ...

## $ order : chr NA "CINGULATA" NA "CARNIVORA" ...

## $ family : chr NA "CHLAMYPHORIDAE" NA "HYAENIDAE" ...

## $ genus : chr NA "Tolypeutes" NA "Proteles" ...

## $ main\_common\_name : chr NA "Brazilian Three-banded Armadillo" NA "Aardwolf" ...

## $ authority : chr NA "(Linnaeus, 1758)" NA "(Sparrman, 1783)" ...

## $ published\_year : int NA 2014 NA 2015 NA 2016 NA 2015 2016 2016 ...

## $ category : chr NA "VU" NA "LC" ...

## $ criteria : chr NA "A2cd" NA NA ...

## $ marine\_system : logi NA FALSE NA FALSE NA FALSE ...

## $ freshwater\_system : logi NA FALSE NA FALSE NA TRUE ...

## $ terrestrial\_system: logi NA TRUE NA TRUE NA TRUE ...

## $ assessor : chr NA "Miranda, F., Moraes-Barros, N., Superina, M. & Abba, A.M." NA "Green, D.S." ...

## $ reviewer : chr NA "Loughry, J." NA "Dloniak, S.M.D. & Holekamp, E." ...

## $ aoo\_km2 : chr NA NA NA NA ...

## $ eoo\_km2 : chr NA "937000" NA NA ...

## $ elevation\_upper : int NA NA NA 2000 NA 550 NA 2500 NA NA ...

## $ elevation\_lower : int NA NA NA 0 NA 0 NA 0 0 0 ...

## $ depth\_upper : num NA NA NA NA NA NA NA NA NA NA ...

## $ depth\_lower : int NA NA NA NA NA NA NA NA NA NA ...

## $ errata\_flag : logi NA NA NA NA NA NA ...

## $ errata\_reason : chr NA NA NA NA ...

## $ amended\_flag : logi NA NA NA NA NA NA ...

## $ amended\_reason : chr NA NA NA NA ...

There are 1134 books, 499 with a conservation status from the IUCN Red  
List, although this includes "DDâ€� meaning "Data Deficientâ€�. I am  
hesitant to actually show the proportion of species in each category for  
those for which I got data for, because the resolution of common names  
to scientific names isn't certainâ€¦ Take the following table with a pinch  
of salt!

dplyr::count(animals, category) %>%

knitr::kable()

| **category** | **n** |
| --- | --- |
| CR | 14 |
| DD | 7 |
| EN | 42 |
| EW | 1 |
| EX | 6 |
| LC | 348 |
| LR/cd | 1 |
| LR/lc | 5 |
| LR/nt | 4 |
| NT | 23 |
| VU | 48 |
| NA | 635 |

See [the following page for more precise information about  
categories](https://en.wikipedia.org/wiki/IUCN_Red_List#IUCN_Red_List_Categories).  
LC is least concern. Let's have a look at the extinct species.

animals %>%

dplyr::filter(category == "EX") %>%

dplyr::select(book, animal, sci\_name) %>%

knitr::kable()

| **book** | **animal** | **sci\_name** |
| --- | --- | --- |
| Java Data Objects | Bilby, Rabbit-eared Bandicoot (Macrotis lagotis) | Macrotis leucura |
| Building and Testing with Gradle | Bush Wren | Xenicus longipes |
| Designing Mobile Payment Experiences | Crested Pigeon | Microgoura meeki |
| SSH, The Secure Shell: The Definitive Guide | Land Snail | Amastra crassilabrum |
| Java NIO | Pigfooted Bandicoot | Chaeropus ecaudatus |
| Java I/O | White Rabbit | Macrotis leucura |

I searched for the covers and names and could assess that in that table,  
there are 4 false positives due to the ambiguity of common names! Only  
the Bush wren and the Pigfooted Bandicoot got scientific names  
corresponding to what they look like, and are extinct, which is quite  
sad.

Now, to reverse-engineer what Edie Freedman wrote in the short history of  
O'Reilly animals, "Many of the animals that appear on our covers are  
critically endangered - the tarsier from Learning the vi & Vim Editors,  
the lorises from sed & awk, the Hawksbill turtle from Getting Started  
with CouchDB, the tiger from Running Mac OS X Tiger, and the African  
elephant on Hadoop: The Definitive Guide, just to name a few.â€�, let's  
look at what we got for them.

animals %>%

dplyr::filter(book %in%

c("Hadoop: The Definitive Guide",

"Learning the vi and Vim Editors",

"sed & awk",

"Getting Started with CouchDB",

"Running Mac OS X Tiger")) %>%

dplyr::select(book, animal, sci\_name, category) %>%

knitr::kable()

| **book** | **animal** | **sci\_name** | **category** |
| --- | --- | --- | --- |
| Hadoop: The Definitive Guide | African Elephant, young | Elephantulus rozeti | LC |
| Getting Started with CouchDB | Hawksbill Turtle | Eretmochelys imbricata | CR |
| sed & awk | Slender Loris "Awkâ€� | NA | NA |
| Running Mac OS X Tiger | Sumatran Tiger | Parantica tityoides | LR/nt |
| Learning the vi and Vim Editors | Tarsier, full-body, standing on hind feet, b/w engraving | Tarsius pelengensis | EN |

So all in all, we got some truth but also some wrong names and hence  
wrong conservation statuses!