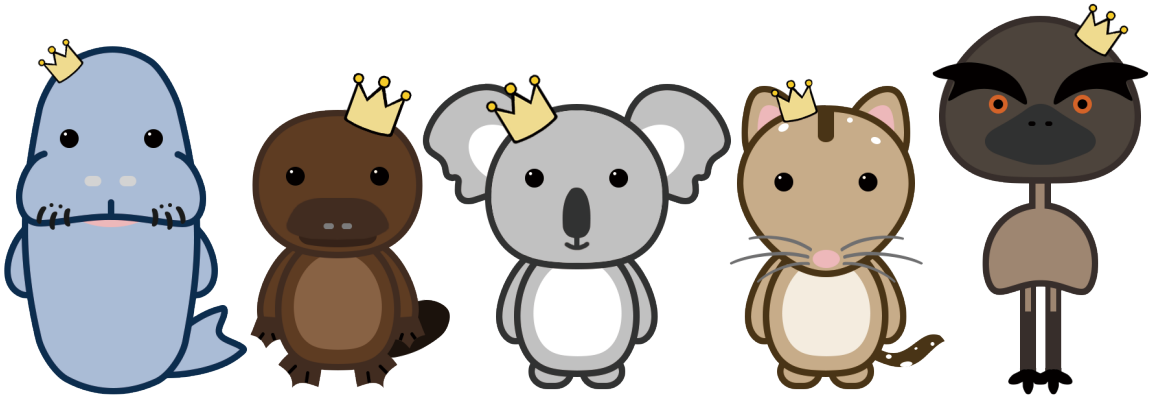
Introduction to specieshindex

Jessica Tam, Malgorzata (Losia) Lagisz, Shinichi Nakagawa, Will Cornwell

December 08, 2020



There are many ways to gauge research influence. We can simply sum up the number of publications or citations, which is a straight forward way without the need of any complicated mathematical formulas. On the other hand, there are more in-depth ways of quantifying research influence by using statistical indices. Some examples include the h-index, m-index, i10 index, h5 index, etc.

H-index (Hirsch 2005) is one of the main methods to measure publication influence objectively. It is calculated using the formula h = total publications (n) that have at least been cited **n** times, after sorting the publications by their citation counts in a descending order.

Currently, h-index is commonly used for assessing influence and academic productivity of authors, research institutions and journals (Braun, Glänzel, and Schubert 2005). Recently, this index has been also applied to quantify the research effort and influence associated with different species of living organisms (Fleming and Bateman 2016).

With this package, we can create a similar profile for different living organisms that would show their citation metrics, including total publications, total citations, h-index, etc. This will allow us to understand which species are receiving more research interest and develop the solutions to rectify the biases.

Installation

To get this package to work, make sure you have the following packages installed.

```
# Installation from GitHub
install.packages("rscopus")
install.packages("taxize")
install.packages("XML")
install.packages("httr")
```

```
install.packages("dplyr")
devtools::install_github("jessicatytam/specieshindex", force = TRUE, build_vignettes = FALSE)
```

```
# Load the library
library(specieshindex)
```

You can find the full list of functions here.

Workflow

Specieshindex package performs two main steps that simplify calculating h-indices for different species:

1. Connecting to Scopus and downloading citation data for a given species
2. Computing various indices (including h-index) for a given species

Below, we outline how to perform these steps using our package.

Connecting to Scopus

Make sure you are connected to the internet via institutional access or acquire a VPN from your institution if you are working from home. Alternatively, the functions will also run if you are already a subscriber of Scopus.

Before initiating the retrieval of any data, make sure you have registered an API key from Scopus. Otherwise, the functions will not run (and is in fact illegal to download the data). You can register your key here. Use your API key as the function parameter "myAPI".

The searches in Scopus are performed using species Latin name. The relevant functions take two parts of a Latin binominal species name as their main parameters, e.g. "*Bettongia*", "*penicillata*". Make sure you are using binomial names instead of common names as required by the code.

We have implemented functions that check the available number of relevant records (Count) and functions that download these records from Scopus (Fetch) for a given species. Count and Fetch functions have two versions, with suffixes T and TAK, respectively. In these functions, the suffix 'T' is for title and 'TAK' is for title+abstract+keywords. These terms are used to indicate where in the Scopus records you want your search terms to be found in. While 'T' only finds the publications with the binomial species name in the title, 'TAK' will find records with the name in title, abstract or keywords.

It is recommend that you run a quick search using functions `CountSpT()` or `CountSpTAK()` before actually fetching the records since this can be a tedious process. Both functions give a search count of the number of records found on the Scopus database.

For example, you can run:

```
# Title only
CountSpT(genus = "Bettongia", species = "penicillata", APIkey = "API key")

#Title, abstract, and keywords
CountSpTAK(genus = "Bettongia", species = "penicillata", APIkey = "API key")
```

There is an optional parameter `additionalkeywords` that can be included to make your search more precise. This is achieved by expanding your query string and restrict the search to records that include specified

combination of the additional keywords. The additional keywords can be a single word, e.g. "conservation", but better yet, you can parse a string with Boolean operators, e.g. "conserv* OR protect* OR reintrod* OR restor*". The Boolean operator "AND" has already been implemented in the code to link the Latin name and the additional string. To learn more about search language, you can visit here and here.

Here are some examples of searches including additional keywords:

```
# Title only
CountSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(consrv* OR protect* OR reintrod* OR restor*)", APIkey = "API key")
CountSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(popul* W/5 ecolog*)", APIkey = "API key")
CountSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(breed* OR reprod* OR fertil* OR fecund*)", APIkey = "API key")
CountSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(popul* AND NOT popular)", APIkey = "API key")
CountSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "NOT (phylog* OR taxonom*)", APIkey = "API key")

# Title, abstract, and keywords
CountSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(consrv* OR protect* OR reintrod* OR restor*)", APIkey = "API key")
CountSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(popul* W/5 ecolog*)", APIkey = "API key")
CountSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(breed* OR reprod* OR fertil* OR fecund*)", APIkey = "API key")
CountSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(popul* AND NOT popular)", APIkey = "API key")
CountSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "NOT (phylog* OR taxonom*)", APIkey = "API key")
```

If no additional keywords are parsed, the extraction will include all publications related to the species. In this case, you MUST include `API key =` to specify your API key, otherwise your key will be read as an additional keyword and result in an error.

`CountSpTAK()` should yield more results since it also looked for the binomial name in the abstract and keywords, in addition to the title. Now that you know exactly how many publications there are for this species, and it is greater than 0, you can proceed to fetch the data from Scopus. If your record count is 0, there might be a problem with the Latin name of the species, e.g. a typo.

As you may expect, some species have been studied more than the others (Simko 2015; Krause and Robinson 2017), resulting in more publications. Therefore, if there are more than a few thousand citation records found on Scopus, it will take a considerably longer time to download them all, unless you are using a computer with high processing power and fast internet connection.

To fetch the citation records, you can run:

```
# Title only
FetchSpT(genus = "Bettongia", species = "penicillata", APIkey = "API key")

#Title, abstract, and keywords
FetchSpTAK(genus = "Bettongia", species = "penicillata", APIkey = "API key")
```

Similar to the Count functions above, Fetch functions also have the argument `additionalkeywords` that allows you to refine your search.

For example, you can run:

```
# Title only
FetchSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(consrv* OR protect* OR reintrod* OR restor*)", APIkey = "API key")
FetchSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(popul* W/5 ecolog*)", APIkey = "API key")
FetchSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(breed* OR reprod* OR fertil* OR fecund*)", APIkey = "API key")
FetchSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "(popul* AND NOT popular)", APIkey = "API key")
FetchSpT(genus = "Bettongia", species = "penicillata",
         additionalkeywords = "NOT (phylog* OR taxonom*)", APIkey = "API key")

# Title, abstract, and keywords
FetchSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(consrv* OR protect* OR reintrod* OR restor*)", APIkey = "API key")
FetchSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(popul* W/5 ecolog*)", APIkey = "API key")
FetchSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(breed* OR reprod* OR fertil* OR fecund*)", APIkey = "API key")
FetchSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "(popul* AND NOT popular)", APIkey = "API key")
FetchSpTAK(genus = "Bettongia", species = "penicillata",
           additionalkeywords = "NOT (phylog* OR taxonom*)", APIkey = "API key")
```

In larger datasets, duplicates may appear, although usually in low numbers. `FetchSpT()` and `FetchSpTAK()` will clean up the dataset by removing them if found. This is done by locating the publications with identical names and then removing the subsequent entry (or entries). The first occurrence of the record will be kept, which is normally the original record.

Possible errors

If the error `Bad Request (HTTP 400)` occurs in this step, try connecting to the internet via institutional access, instead of the one at home. This error might be caused by some authorisation issues.

It is important to make sure you have spelt the genus and species name accurately when entering the function. If unsure, you can go to the Global Names Resolver for verification. This is where the above functions are linked to for checking via the package `taxize` (Chamberlain and Szöcs 2013). If the desired species is not in the index, it is possible that it has just been discovered and not officially listed yet, or the name has been changed and is no longer in use. Whether or not old synonyms are accepted depends on if they still exist in the database.

Simple counts

These functions do simple calculations with various parameters for the sets of downloaded citations for a given species. This information could provide insights into the behaviour of the other, more complicated, indices.

`TotalPub()` sums up the total publications from the search.

For example:

```
TotalPub(Woylie)
#> [1] 113
```

`TotalCite()` sums up the total citations of all the records from the search.

For example:

```
TotalCite(Woylie)
#> [1] 1903
```

`TotalJournals()` counts the number of unique journals that the species have appeared in.

For example:

```
TotalJournals(Woylie)
#> [1] 55
```

`TotalArt()` counts the number of publications that are articles.

For example:

```
TotalArt(Woylie)
#> [1] 110
```

`TotalRev()` counts the number of publications that are reviews.

For example:

```
TotalRev(Woylie)
#> [1] 3
```

`ARRatio()` calculates the percentage ratio of the number of articles:reviews, and rounds the percentage to 4 significant figures.

For example:

```
ARRatio(Woylie)
#> [1] "97.35 : 2.655"
```

`YearsPublishing()` counts the number of years since the first publication of the species.

For example:

```
YearsPublishing(Woylie)
#> [1] 43
```

Statistical indices

Statistical indices can provide deeper insights than the simple counts and ratios described above. They take a few parameters into consideration, such as time, total citations, total publications, and their distributions, to compute the index. Here we implemented the following statistical indices of research impact for species: h-index, h5 index, m-index, and the i10 index.

Use `SpHindex()` to compute the h-index (Hirsch 2005) for a given set of records. It sorts the citation counts in descending order and then finds the largest number of publications (n) that have EACH been cited at least **n** times.

For example:

```
SpHindex(Woylie)
#> [1] 26
```

The h5 index computes the h-index for the past 5 years.

For example:

```
SpH5(Woylie)
#> [1] 7
```

You can calculate h-index for publications published after a certain date. To subset the time with a specific lower limit, you can use `SpHAfterdate()`. Make sure the date parameter is in the exact format of yyyy-mm-dd. Using a different format or using slashes or hyphens will return an error.

For example:

```
SpHAfterdate(Woylie, "2000-01-01")
#> [1] 20
```

The m-index is an h-index adjusted for the age of publication. The m-index is calculated using the `SpMindex()` function, which first computes the h-index and then divides it by the number of years of publication activity , i.e. number of years since the oldest publication in the dataset was published:

$$m = h \,/\, years\ publishing$$

For example:

```
SpMindex(Woylie)
#> [1] 0.605
```

The i10 index is another variant of the h-index, which only takes into account publications with at least 10 citations. The i10 index can be calculated using `Spi10()`, which simply counts the publications that has 10 or more citations.

For example:

```
Spi10(Woylie)
#> [1] 54
```

All indices in one go

Rather than calling each function to calculate the indices in turn, you can get them all in one go. The `Allindices()` function conveniently returns a dataframe with most of the above indices. This allows for quick creation of dataframes with multiple species, analyses and plotting.

For example:

```
Allindices(Woylie, genus = "Bettongia", species = "penicillata")
#>           genus_species     species     genus publications citations journals
#> 1 Bettongia_penicillata penicillata Bettongia          113      1903       55
#>   articles reviews years_publishing  h     m i10 h5
#> 1      110       3               43 26 0.605  54  7
```

For multiple species, `rbind()` can be used to stitch the outputs together, which will be used below.

Worked examples

To demonstrate how the specieshindex package could be helpful in the real world, I will be comparing the citations of 4 species of marsupials. They are the woylie (*Bettongia penicillata*), quokka (*Setonix brachyurus*), platypus (*Ornithorhynchus anatinus*), and koala (*Phascolarctos cinereus*). I used `FetchSpTAK()` for all of these datasets. I have included these sets of data in the package but you can also extract the data from Scopus for the most updated records.

```
dim(Woylie)
#> [1] 113  20
dim(Quokka)
#> [1] 242  20
dim(Platypus)
#> [1] 321  20
dim(Koala)
#> [1] 773  20
```

As shown here, there were 113 records (publications) for the woylie, 242 records for the quokka, 321 records for the platypus, and 773 records for the koala. They all have a total of 20 variables, e.g. citations, journals, DOI, etc., representing bibliometric information downloaded from Scopus for each publication. I have skipped the steps of fetching the data from Scopus since it had already been demonstrated above.

Comparing indices

Next, we calculate research productivity and influence indices using the `Allindices()` function:

```
W <- Allindices(Woylie, genus = "Bettongia", species = "penicillata")
Q <- Allindices(Quokka, genus = "Setonix", species = "brachyurus")
P <- Allindices(Platypus, genus = "Ornithorhynchus", species = "anatinus")
K <- Allindices(Koala, genus = "Phascolarctos", species = "cinereus")
```

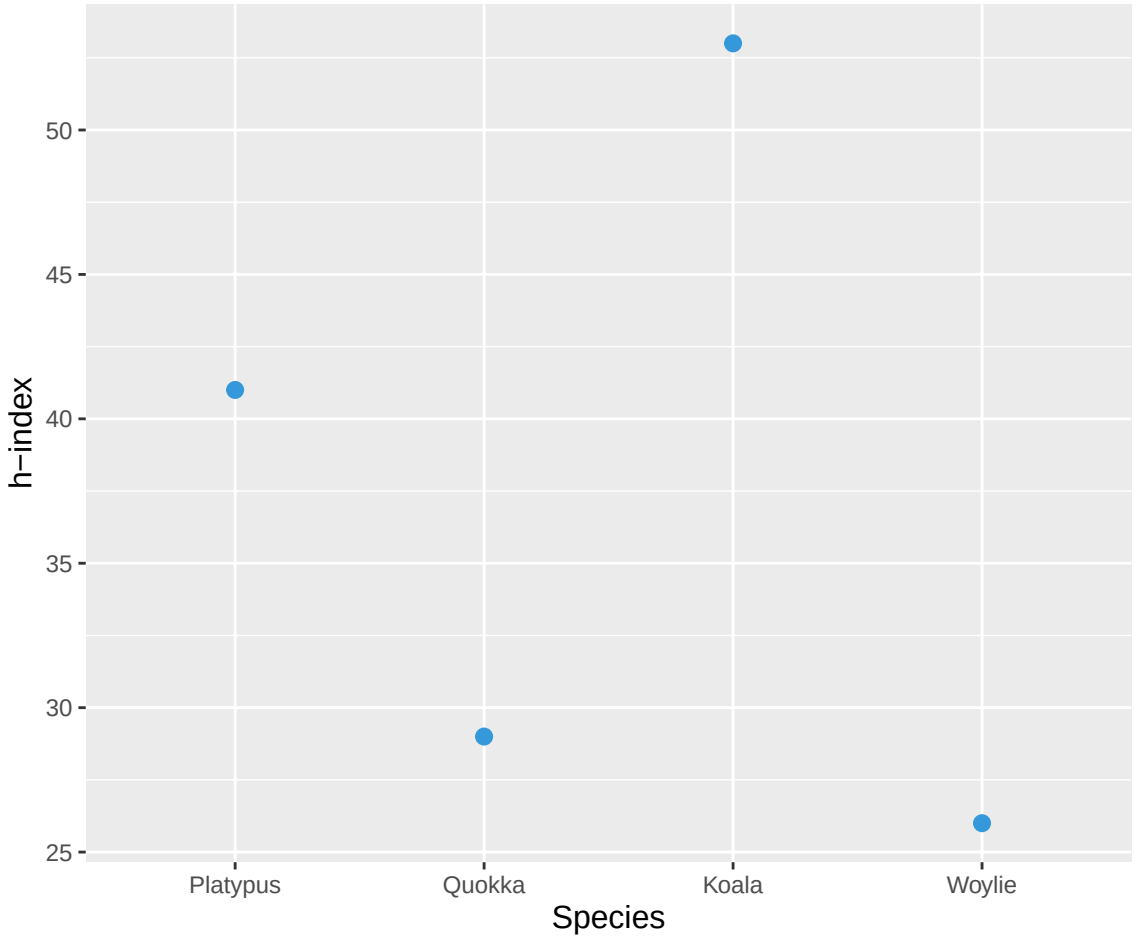
If you want to compare and analyse them against each other, you can use the function `rbind()` to combine indices for all four species into a single dataframe:

```
CombineSp <- rbind(W, Q, P, K) #combining the citation records
CombineSp
#>              genus_species     species           genus publications citations
#> 1    Bettongia_penicillata penicillata       Bettongia          113      1903
#> 2       Setonix_brachyurus  brachyurus         Setonix          242      3427
#> 3 Ornithorhynchus_anatinus    anatinus Ornithorhynchus          321      6365
#> 4   Phascolarctos_cinereus    cinereus   Phascolarctos          773     14291
#>   journals articles reviews years_publishing  h     m i10 h5
#> 1       55      110       3               43 26 0.605  54  7
#> 2      107      237       5               66 29 0.439 121  4
#> 3      153      308      13               67 41 0.612 177  7
#> 4      227      744      29              139 53 0.381 427 14
```

After preparing the data, we are ready to make some plots. For example, using the package `ggplot2`, we can plot the h-index and m-index. When comparing the indices, it is important to remember that their scales may differ drastically. Therefore, putting the variables in the same plot may generate plots that do not make a lot of sense.

We first plot the h-index:

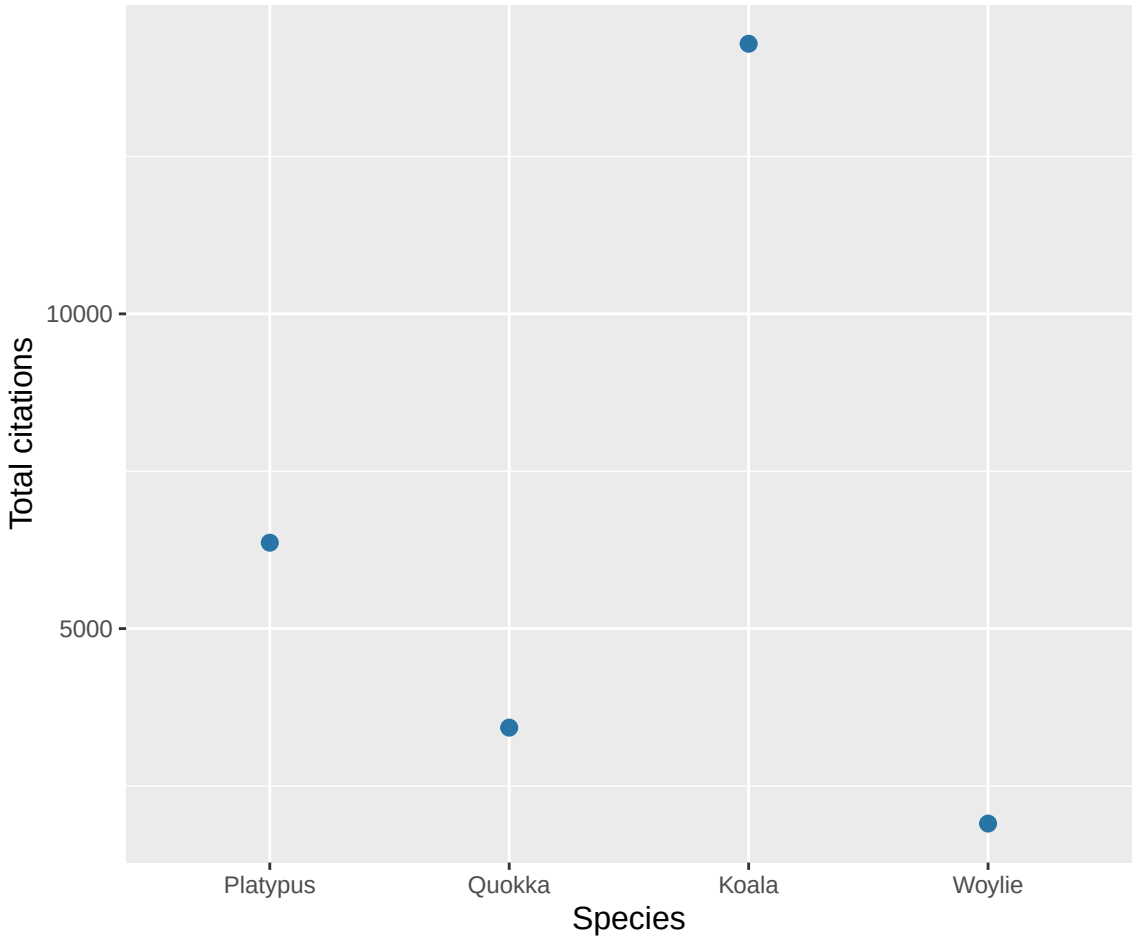
```r
# h-index
library(ggplot2)
ggplot(CombineSp, aes(x = species)) +
  geom_point(aes(y = h,
                 colour = "H-index"),
             size = 2.5) +
  labs(x = "Species",
       y = "h-index",
       colour = "Index") +
  scale_x_discrete(labels = c("Platypus", "Quokka", "Koala", "Woylie")) +
  scale_colour_manual(values = c("H-index" = "#3498DB")) +
  theme(axis.title = element_text(size = 12),
        axis.text = element_text(size = 9),
        legend.position = "none")
```



We can also plot the total citations counts for comparison, since citations is an important factor in contributing to h-index.

```r
# Total citations
ggplot(CombineSp, aes(x = species)) +
geom_point(aes(y = citations,
               colour = "Citations"),
           size = 2.5) +
```

```
labs(x = "Species",
     y = "Total citations",
     colour = "Index") +
scale_x_discrete(labels = c("Platypus", "Quokka", "Koala", "Woylie")) +
scale_colour_manual(values = c("Citations"  = "#2874A6")) +
theme(axis.title = element_text(size = 12),
      axis.text = element_text(size = 9),
      legend.position = "none")
```

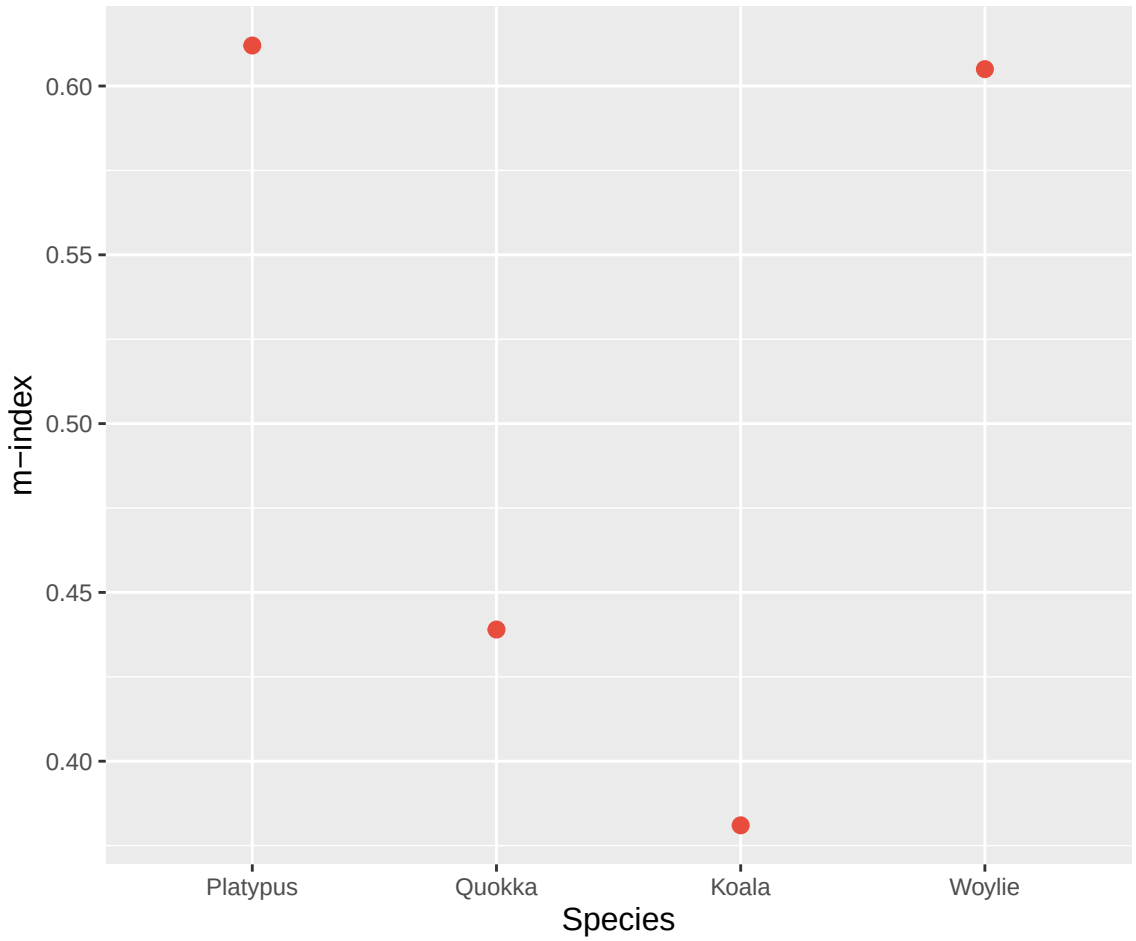


It seems like the h-index values are quite consistent with the total numbers of citations, i.e. the more the citations in total, the higher the h-index score. Therefore, the woylie, with the lowest number of citations has the lowest h-index. On the other hand, the koala, with the highest number of citations has the highest h-index.

We next plot the m-index to see how it differs from the h-index. Recalling that m-index is calculated by dividing the h-index by the number of years of publication activity ($m = h/years\ publishing$), let us have a closer inspection at both the m-index:

```
# m-index
ggplot(CombineSp, aes(x = species)) +
  geom_point(aes(y = m,
                 colour = "M-index"),
             size = 2.5) +
```

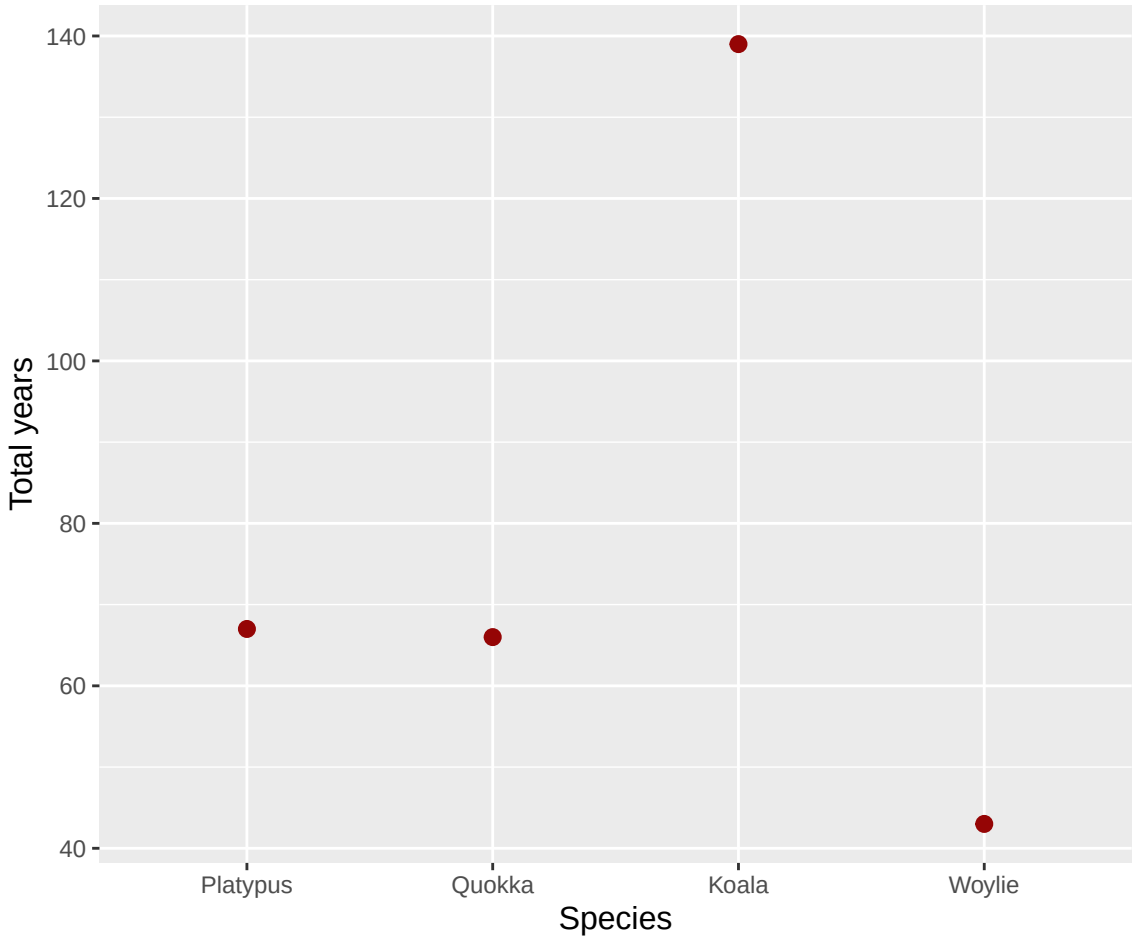
```
  labs(x = "Species",
       y = "m-index",
       colour = "Index") +
  scale_x_discrete(labels = c("Platypus", "Quokka", "Koala", "Woylie")) +
  scale_colour_manual(values = c("M-index"  = "#E74C3C")) +
  theme(axis.title = element_text(size = 12),
        axis.text = element_text(size = 9),
        legend.position = "none")
```



We should also have a look at the related variable years_publishing (number of years since first publication for a given species).

```
# Years publishing
ggplot(CombineSp, aes(x = species)) +
  geom_point(aes(y = years_publishing,
                 colour = "Years publishing"),
             size = 2.5) +
  labs(x = "Species",
       y = "Total years",
       colour = "Index") +
  scale_x_discrete(labels = c("Platypus", "Quokka", "Koala", "Woylie")) +
  scale_colour_manual(values = c("Years publishing" = "#950505")) +
```

```
theme(axis.title = element_text(size = 12),
      axis.text = element_text(size = 9),
      legend.position = "none")
```



In order to understand what yields a high m-index score, we should recall that the m-index is a ratio. A high m-index can be achieved by combination of a higher h-index and a lower `years_publishing` value. This explains why koala publications have a relatively low m-index, albeit highest h-index. Research on koalas has been produced for over 3 times longer than that on the woylie. Although the h-index of woylie publications was half of that of the koala, it had only been in the scientific scene for a few decades. Comparing the quokka to the platypus, we can see that their publications have both been around for a similar duration of time. 66 years for the quokka and 67 years for the platypus respectively. Since the platypus has a higher h-index than the quokka, this is thus consistent with the higher m-index score here.

Limitations

Currently, the input option for search requests is limited to species Latin names. Although common names are synonymous with the binomial names, they are comparatively much less explicit. There could be multiple species using the same common name, which will require extra steps to specify which is the particular species that is being studied. Moreover, there are other fields of research that use species' common names in the titles of their work without meaning the actual species, e.g. as a product/project name.

Another limitation is that only species-level searches using binomial names are enabled at the moment. This is because of homonymous (Remsen 2016) genus names between some plant and animal taxa. For example, the genus *Prunella* represents both a taxon of herbaceous plants and passerines. Additional parameters will have to be added to identify the exact taxon in question.

Finally, requests must be sent via a subscriber's internet provider. This include universities and individuals who have paid for annual Scopus subscription. This is easily addressed by connecting to your institution's network. Hence, it might not work at all if you are working from home. The error `Bad Request (HTTP 400)` will be returned in this case.

Implications and future uses

Although the development of this package is still in its infancy, it is nonetheless a valuable tool for reserchers looking to examine research biases between species. Some functions that could be added in the future include search requests for higher classification levels, e.g. genus, class, etc., and to include other emerging indices that are being used to calculate research influence. The results of the species h-index will show us the research interest each taxon receives. More investigations will then follow to explore the reasons why and how the knowledge gap affected species conservation and the depth of understanding we have for them.

Acknowledgements

I acknowledge the contributions of the authors of the dependency packages: `rscopus`, `taxize`, `XML`, `httr`, `dplyr`.

References

Braun, T., W. Glänzel, and A. Schubert. 2005. "A Hirsch-Type Index for Journals [1]." *Scientist* 19 (22): 8. https://www.scopus.com/inward/record.uri?eid=2-s2.0-28944445322&partnerID=40&md5=9f03ec93906f77c8de67db2d6fe506e8.

Chamberlain, S. A., and E. Szöcs. 2013. "Taxize: Taxonomic Search and Retrieval in R." *F1000Research* 2. https://doi.org/10.12688/f1000research.2-191.v2.

Fleming, P. A., and P. W. Bateman. 2016. "The Good, the Bad, and the Ugly: Which Australian Terrestrial Mammal Species Attract Most Research?" *Mammal Review* 46 (4): 241–54. https://doi.org/10.1111/mam.12066.

Hirsch, J. E. 2005. "An Index to Quantify an Individual's Scientific Research Output." *Proceedings of the National Academy of Sciences of the United States of America* 102 (46): 16569–72. https://doi.org/10.1073/pnas.0507655102.

Krause, M., and K. Robinson. 2017. ""Charismatic Species and Beyond: How Cultural Schemas and Organisational Routines Shape Conservation"." *Conservation and Society* 15 (3): 313–21. https://doi.org/10.4103/cs.cs_16_63.

Remsen, David. 2016. "The Use and Limits of Scientific Names in Biological Informatics." *ZooKeys* 550: 207–23. https://doi.org/10.3897/zookeys.550.9546.

Simko, I. 2015. "Analysis of Bibliometric Indicators to Determine Citation Bias." *Palgrave Communications* 1. https://doi.org/10.1057/palcomms.2015.11.