What's in a Name?

Johanna Jantzen

September 30, 2020

For instructors: If your students do not have internet access, R objects are available which contain the information retrieved from the online databases. Have your students read the files in (see Guide for Instructors for how to read R objects (.RDS) into R) and use those objects instead of downloading the data from the online database.

Start of module

Before doing any work in R, you typically need to load the libraries with the functions that you will need. These libraries have previously been installed for this R project. Let's load them that now by selecting those rows and clicking "Run" or pressing *Ctrl Enter* (Windows) or *Cmd Return* (Mac). The warnings that you see in red are expected.

```
library(rotl)
```

```
## Warning: package 'rotl' was built under R version 3.6.3
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ape)
```

```
## Warning: package 'ape' was built under R version 3.6.3
```

```
library(taxize)
```

```
## Warning: package 'taxize' was built under R version 3.6.3
```

```
##
## Attaching package: 'taxize'
```

```
## The following objects are masked from 'package:rotl':
##
##     synonyms, tax_name, tax_rank
```

The Open Tree of Life project, which can be accessed at https://opentreeoflife.github.io/, has the goal of synthesizing a phylogeny for all of life, representing the evolutionary relationships between these organisms. A phylogeny that represents all known relationships between organisms, from plants to animals to microbes, has been made available for public use. If you are completing the entire course, we will start to investigate these relationships, focusing first on the names of taxa that we are studying in this "What's in a name?" module.

Taxonomy is the system used to classify organisms into groups while Nomenclature is the system used to name these groups. There are several taxonomic databases that contain lists of species names and higher order taxonomies. This module will explore the names and diveristy of organisms in four **clades**: plants, insects, birds, and amphibians. Think about which of these four clades you would want to investigate further if you are continuing on to the phylogenetic and trait data modules in this course.

Import species lists

First, let's get a list of species to study. You can get species lists from any number of online sources (see Admin/Data_Attribution/Data_citations.txt for the sources of these species lists) or you can generate your own species lists for organisms you are interested in studing. We have selected these groups of organisms to generate five lists of taxa: 1) Four conifer **genera** (plural of genus) that can be found around the world (Plants) 2) The maple genus (*Acer*), including only the species found in Canada (Plants) 3) Ant species occurring in Alberta, Canada (Insects) 4) The nuthatch genus (*Sitta*) across its entire distribution (Birds) 5) Frog species occurring in BC, Canada (Amphibians)

For each group of organisms (and therefore species lists), we are also including an **outgroup** species for downstream phylogenetic analysis (see "Module 2: Phylogenetics" for more information on outgroups).

We can generate R objects containing the lists of species names or genus names that we are interested in. For most of the groups, we are just typing the names into lists using the **function** *c()*. For the ant clade, we are reading in a **.txt** file which contains the list of species names. You can also open this "Ant_names.txt" file by clicking on the file name in the "Files" corner of RStudio in the folder "Data".

```
> #For birds, we'll look at the genus Sitta or nuthatches. We also include one
    species of wren (Troglodytes) as an outgroup for phylogenetic analysis.
> birds <- c("Sitta", "Troglodytes aedon")
>
> #For insects, we're using a list of ant species from Alberta, Canada. We
    have an extra step of converting our object into a vector (list) instead
    of a dataframe.
> ants_df <- read.delim("../Data/Ant_names.txt", header = FALSE,
    stringsAsFactors = FALSE)
> str(ants_df)

## 'data.frame':    90 obs. of  1 variable:
##  $ V1: chr  "Dolichoderus taschenbergi" "Tapinoma sessile " "Brachymyrmex
    depilis " "Brachymyrmex obscurior " ...

> ants <- as.vector(ants_df$V1)
> str(ants)

##  chr [1:90] "Dolichoderus taschenbergi" "Tapinoma sessile " ...
```

```
> #For plants, we have two lists: a list of maple species from Canada with
    horse chestnut as an outgroup, and a list of conifer genera with Ginkgo as
    an outgroup.
> maples <- c("Acer circinatum", "Acer glabrum", "Acer macrophyllum", "Acer
    negundo", "Acer nigrum", "Acer pensylvanicum", "Acer rubrum", "Acer
    saccharum", "Acer saccharinum", "Acer spicatum", "Aesculus glabra")
> conifers <- c("Abies", "Thuja", "Picea", "Tsuga", "Ginkgo")
>
> #For amphibians, we have a list of frog species from BC, Canada with a
    salamander as an outgroup.
> frogs <- c("Anaxyrus boreas", "Spea intermontanus", "Pseudacris maculata", "
    Hyla regilla", "Rana aurora", "Ascaphus truei", "Rana luteiventris", "Rana
    pretiosa", "Lithobates catesbeiana", "Lithobates clamitans", "Lithobates
    pipiens", "Lithobates sylvaticus", "Taricha granulosa")
```

Assessment

1) To check that you have loaded the text file containing the ant species names properly and were able to run the lines of code above, how many species of ants were in the file?

Resolving names with TNRS

When dealing with species names, it's imporant to consider that there may be errors or inconsistencies in the names that you are using. The names in our lists could be old names or **synonyms** which may have been the result of taxonomic changes (such as moving a species from one genus to another) or could contain spelling or **orthographic** errors. Before we proceed with any data collection or analyses using these names, let's first resolve any of these "issues" using the Taxonomic Name Resolution Service (TNRS).

We are using the **objects** generated by the code above with the **function** *tnrs_match_names* which does what it says: matches our species and genus names with names in the TNRS database. We assign the output from this **function** to a new **object**. It is good to use informative names for your **objects** so that you know what they contain. In this case, one new **object** contains the *birds* names that have been resolved so we use the name *birds_resolved*. You can name objects whatever you like (within reason) but short(ish) and concise names are best. Names also need to be one **string** so in place of spaces, you should use "_","-", or"." to separate words, or you can use camel case like this: *birdsResolved*.

```
> #Match names using the taxonomic name resolution service
> #These lines of code will not produce any output to the screen
> birds_resolved <- tnrs_match_names(birds)
> ants_resolved <- tnrs_match_names(ants)
> maples_resolved <- tnrs_match_names(maples)
> conifers_resolved <- tnrs_match_names(conifers)
> frogs_resolved <- tnrs_match_names(frogs)
```

Exploring the dataframes of resolved names

Now that we have checked our names against the TNRS, we can see if there have been any name changes or errors. Let's look at the dataframes that were output by the *tnrs_match_names* **function**. There are several ways to look at our data.

We can look at the entire object by just **running** the object. This works best when the objects are small (that is, they do not have too many rows or columns). We can also inspect just the first few rows using the

function *head()* or we can select specific rows or columns by either subsetting using a number within *[ ]* or the name using *$*.

Remember that we can use a variety of **functions** to get more information about the dimensions and structure of **objects**: * *nrow()* - get the number of rows of a dataframe/table * *ncol()* - get the number of columns of a dataframe/table * *rownames()* - get the names of the rows of a dataframe/table * *colnames()* - get the names of the columns of a dataframe/table * *dim()* - get the dimensions (height and width) of a dataframe/table * *str()* - get the structure of an object * *class()* - get the class of an object * *summary()* - get a summary of an object * *length()* - get the length of an object (eg a list)

```
> #Inspect entire bird object
> birds_resolved

##       search_string       unique_name approximate_match ott_id is_synonym
## 1             sitta             Sitta             FALSE 603922      FALSE
## 2 troglodytes aedon Troglodytes aedon             FALSE 293378      FALSE
##            flags number_matches
## 1 sibling_higher              1
## 2                             1

> nrow(maples_resolved)

## [1] 11

> #Get the first few rows of the ants object
> head(ants_resolved)

##               search_string               unique_name approximate_match
    ott_id
## 1 dolichoderus taschenbergi Dolichoderus taschenbergi             FALSE
    3258539
## 2         tapinoma sessile           Tapinoma sessile              TRUE
    1008036
## 3     brachymyrmex depilis       Brachymyrmex depilis              TRUE
    225004
## 4   brachymyrmex obscurior     Brachymyrmex obscurior              TRUE
    3259305
## 5   camponotus herculeanus     Camponotus herculeanus              TRUE
    436669
## 6   camponotus laevigatus      Camponotus laevigatus               TRUE
    854285
##   is_synonym          flags number_matches
## 1      FALSE                             1
## 2      FALSE                             2
## 3      FALSE                             1
## 4      FALSE                             1
## 5      FALSE sibling_higher              3
## 6      FALSE sibling_higher              1

> #Get the first and third rows of the maples object
> maples_resolved[c(1,3),]

##       search_string       unique_name approximate_match ott_id is_synonym
    flags
## 1   acer circinatum   Acer circinatum             FALSE 191948      FALSE
```

```
## 3 acer macrophyllum Acer macrophyllum              FALSE 538695       FALSE
##   number_matches
## 1              1
## 3              1

> #Get the first to third rows and the first to fifth columns of the conifers
    object
> conifers_resolved[1:3, 1:5]

##   search_string unique_name approximate_match ott_id is_synonym
## 1         abies       Abies             FALSE 994065      FALSE
## 2         thuja       Thuja             FALSE 994095      FALSE
## 3         picea       Picea             FALSE 517942      FALSE

> #Get the "unique_name" and "is_synonym" columns of the frogs object
> frogs_resolved$unique_name

##  [1] "Anaxyrus boreas"     "Spea intermontana"   "Pseudacris maculata"
##  [4] "Pseudacris regilla"  "Rana aurora"         "Ascaphus truei"
##  [7] "Rana luteiventris"   "Rana pretiosa"       "Rana catesbeiana"
## [10] "Rana clamitans"      "Rana pipiens"        "Rana sylvatica"
## [13] "Taricha granulosa"

> frogs_resolved$is_synonym

##  [1] FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
    TRUE
## [13] FALSE

> #For those datasets which now include the correct species names, we can make
     our new species name lists
> frog_species_list <- frogs_resolved$unique_name
> maple_species_list <- maples_resolved$unique_name
> ant_species_list <- ants_resolved$unique_name
```

Let's see if you can get some more information about of these objects.

Assessment

2) How many rows does the *maples_resolved* object have? What function(s) can you use to calculate that?
3) How many columns does the *birds_resolved* object have? What function(s) can you use to calculate that?
4) How many taxa (from all five datasets) have synonyms? Which column contains this information?
5) Which dataset(s) (**object(s)**) contain(s) taxa that have no taxonomic problems, ambiguities or name changes/synonyms (at the taxonomic level studied)? Explain.
6) Give an example of an **orthographic** or spelling error from one of the datasets (**objects**) and one example of a synonym due to the movement from a species from one genus to another.
7) What might you conclude about the taxonomy of the ant species? Do you have confidence in the names that were matched?
8) Are you able to determine which of the five datasets contains the most species? Why or why not?

Retrieving names for lower taxonomic levels

You may have noticed that for some of the datasets, we have lists of **species binomials** composed of **genus** names and **specific epithets**. For others, however, we only have the **genus** name and no information on the **species** within that **genus**. If we want to get a list of all the species that are in a genus, we can use one of these **functions**: *downstream()* or *children()* from the **package** *taxize*.

downstream() can obtain names for different taxonomic levels below the one you **query**. For example, if you are looking for a list of **genera** within the plant **family** "Sapindaceae", the **function** would look like: *downstream("Sapindaceae", downto = "genus", db = "itis")*. Alternatively, you could get the list of **species** within this family by specifying: *downstream("Sapindaceae", downto = "species", db = "itis")*.

children() is slightly different in that this **function** only retrieves the names at the taxonomic level immediately below the one queried. So if you input a list of **genera**, you will receive the **species** names, while if you input a list of **families**, you would receive a list of **genera**.

These **functions** can access several different taxonomic databases. In this module, we will use the Integrated Taxonomic Information System (ITIS) which can also be accessed through https://www.itis.gov/.

```
> #Examples of *downstream()* and *children()* functions
> #The first element in the function can either be a taxonomic name in quotes
    (eg "Sapindaceae") or an R object that contains a list of taxon names (eg
    conifers)
>
> #Here, we have specified to get names down to the species level
> sapindaceae_species <- downstream("Sapindaceae", downto = "species", db = "
    itis")

## ==  1 queries  ===============

##
## Retrieving data for taxon 'Sapindaceae'

## v  Found:  Sapindaceae
## ==  Results  =================
##
## * Total: 1
## * Found: 1
## * Not Found: 0

> #The children function only retrieves the level immediately below, so here
    we are retrieving the genus names
> sapindaceae_genera <- children("Sapindaceae", db = "itis")

## ==  1 queries  ===============

##
## Retrieving data for taxon 'Sapindaceae'

## v  Found:  Sapindaceae
## ==  Results  =================
##
## * Total: 1
## * Found: 1
## * Not Found: 0
```

The next section has some code that is a bit more complicated but uses the functions we tested with Sapindaceae above. As you run these lines, see if you can figure out how each line works.

```
> #Now we will get species names for those datasets without species names (ie
    birds, conifers)
>
> #Note that for our bird dataset, we don't want to get species name
    information for the outgroup because it is already a species binomial not
    a genus name
> birds

## [1] "Sitta"              "Troglodytes aedon"

> #Although we could just type the name "Sitta" in as our query, we will edit
    our list by
> #omitting the last entry (the outgroup) from the list. While it would be
    easy to do this manually for
> #our very short list of taxa, if we had a longer list it wouldn't be
    practical
> #We'll break this down into its component steps
> #Note: you do not need to change any of the code for this section
> #Don't worry if you do not understand the code; this may be a first
    introduction for you
> #This is a demonstration of how to manipulate objects in R so you can see
    what code can do
>
> #First, we want to know how long the list is
> length(birds)

## [1] 2

> #We want to get rid of the last entry (the Nth entry for a list of length N)
     for this query so we use the '-' symbol to indicate we want to remove
    this entry, using the [ ] we had used earlier to subset our data.
> #length() is a function to get the length of a list (see above)
> birds[-(length(birds))]

## [1] "Sitta"

> #We now have the version of the list we want for the query so we can insert
    it into our function
> species_birds <- downstream(birds[-(length(birds))], downto = "species", db
   = "itis")

## ==  1 queries  ===============

##
## Retrieving data for taxon 'Sitta'

## v  Found:  Sitta
## ==  Results  ================
##
## * Total: 1
## * Found: 1
## * Not Found: 0
```

```
> #We can compare the other function for getting species names for same bird
    dataset. Are the results the same?
> species_birds2 <- children(birds[-(length(birds))], db = "itis")

## ==  1 queries  ===============

##
## Retrieving data for taxon 'Sitta'

## v  Found:  Sitta
## ==  Results  =================
##
## * Total: 1
## * Found: 1
## * Not Found: 0

> #If we want to create our final list of bird species including our outgroup,
     we can now add our last entry from the original taxon list back (omitting
     the "-" now becasue we want to include only that entry instead of
    exlcuding it)
>
> bird_species_list <- c(species_birds$Sitta$taxonname, birds[(length(birds))
    ])
>
> #We can now get species names for conifers - all the names in the list are
    genus names so we can query the entire list
> species_conifers <- downstream(conifers, downto = "species", db = "itis")

## ==  5 queries  ===============

##
## Retrieving data for taxon 'Abies'

## v  Found:  Abies

##
## Retrieving data for taxon 'Thuja'

## v  Found:  Thuja

##
## Retrieving data for taxon 'Picea'

## v  Found:  Picea

##
## Retrieving data for taxon 'Tsuga'

## v  Found:  Tsuga

##
## Retrieving data for taxon 'Ginkgo'
```

```
## v  Found:  Ginkgo
## ==  Results  ================
##
## * Total: 5
## * Found: 5
## * Not Found: 0

> #Now you have a list of dataframes for the conifers with a different
    dataframe for each genus, but you want the names as a single list
> species_conifers

## $Abies
##       tsn parentname parenttsn rankname                      taxonname
    rankid
## 1   18032       Abies     18031  species                  Abies balsamea
    220
## 2  181824       Abies     18031  species                  Abies amabilis
    220
## 3  181825       Abies     18031  species                 Abies bracteata
    220
## 4  181826       Abies     18031  species                  Abies concolor
    220
## 5  181829       Abies     18031  species                   Abies fraseri
    220
## 6  181830       Abies     18031  species                Abies lasiocarpa
    220
## 7  181834       Abies     18031  species                 Abies magnifica
    220
## 8  181835       Abies     18031  species                   Abies procera
    220
## 9  183277       Abies     18031  species                   Abies lowiana
    220
## 10 183284       Abies     18031  species                   Abies grandis
    220
## 11 194774       Abies     18031  species              Abies guatemalensis
    220
## 12 500948       Abies     18031  species               Abies X shastensis
    220
## 13 506607       Abies     18031  species                      Abies alba
    220
## 14 564982       Abies     18031  species                 Abies homolepis
    220
## 15 822548       Abies     18031  species                  Abies veitchii
    220
## 16 822690       Abies     18031  species Abies concolor X Abies grandis
    220
##
## $Thuja
##       tsn parentname parenttsn rankname          taxonname rankid
## 1   18044       Thuja     18043  species      Thuja plicata    220
## 2  505490       Thuja     18043  species Thuja occidentalis    220
##
## $Picea
##       tsn parentname parenttsn rankname         taxonname rankid
```

```
## 1   18034       Picea     18033  species       Picea rubens    220
## 2  183289       Picea     18033  species        Picea abies    220
## 3  183290       Picea     18033  species  Picea breweriana    220
## 4  183291       Picea     18033  species Picea engelmannii    220
## 5  183295       Picea     18033  species       Picea glauca    220
## 6  183302       Picea     18033  species      Picea mariana    220
## 7  183307       Picea     18033  species      Picea pungens    220
## 8  183309       Picea     18033  species  Picea sitchensis    220
## 9  194777       Picea     18033  species    Picea X lutzii    220
## 10 822549       Picea     18033  species      Picea omorika    220
## 11 822580       Picea     18033  species    Picea smithiana    220
## 12 822581       Picea     18033  species      Picea glehnii    220
##
## $Tsuga
##       tsn parentname parenttsn rankname          taxonname rankid
## 1 183397       Tsuga    183396  species   Tsuga canadensis    220
## 2 183399       Tsuga    183396  species  Tsuga caroliniana    220
## 3 183400       Tsuga    183396  species Tsuga heterophylla    220
## 4 183402       Tsuga    183396  species  Tsuga mertensiana    220
## 5 505616       Tsuga    183396  species   Tsuga X jeffreyi    220
##
## $Ginkgo
##       tsn parentname parenttsn rankname     taxonname rankid
## 1 183269      Ginkgo    183264  species Ginkgo biloba    220
##
## attr(,"class")
## [1] "downstream"
## attr(,"db")
## [1] "itis"
```

```
> #We will now do some data manipulation that R makes simpler and more
    reproducible than cutting and pasting in Excel
>
> #We can retrieve the list of species within each genus using the $ symbol
    which retrieves a specific column
> #For example, within the 'species_conifers' list, we can specify the 'Abies'
    dataframe and the 'taxonname' column which includes the species names
> species_conifers$Abies
```

```
##       tsn parentname parenttsn rankname                    taxonname
    rankid
## 1   18032       Abies     18031  species                Abies balsamea
    220
## 2  181824       Abies     18031  species                Abies amabilis
    220
## 3  181825       Abies     18031  species               Abies bracteata
    220
## 4  181826       Abies     18031  species                Abies concolor
    220
## 5  181829       Abies     18031  species                 Abies fraseri
    220
## 6  181830       Abies     18031  species              Abies lasiocarpa
    220
## 7  181834       Abies     18031  species               Abies magnifica
    220
```

```
## 8  181835      Abies     18031  species                  Abies procera
    220
## 9  183277      Abies     18031  species                  Abies lowiana
    220
## 10 183284      Abies     18031  species                  Abies grandis
    220
## 11 194774      Abies     18031  species            Abies guatemalensis
    220
## 12 500948      Abies     18031  species             Abies X shastensis
    220
## 13 506607      Abies     18031  species                    Abies alba
    220
## 14 564982      Abies     18031  species                Abies homolepis
    220
## 15 822548      Abies     18031  species                 Abies veitchii
    220
## 16 822690      Abies     18031  species Abies concolor X Abies grandis
    220
```

```
> species_conifers$Abies$taxonname
```

```
##  [1] "Abies balsamea"              "Abies amabilis"
##  [3] "Abies bracteata"             "Abies concolor"
##  [5] "Abies fraseri"               "Abies lasiocarpa"
##  [7] "Abies magnifica"             "Abies procera"
##  [9] "Abies lowiana"               "Abies grandis"
## [11] "Abies guatemalensis"         "Abies X shastensis"
## [13] "Abies alba"                  "Abies homolepis"
## [15] "Abies veitchii"              "Abies concolor X Abies grandis"
```

```
> #We will take these five lists of names and join them into one list using
    the c() function
> conifer_species_list <- c(species_conifers$Abies$taxonname, species_conifers
    $Thuja$taxonname, species_conifers$Picea$taxonname, species_conifers$Tsuga
    $taxonname, species_conifers$Ginkgo$taxonname)
>
> #We now have five lists of species for our five groups of organisms
> # conifer_species_list
> # bird_species_list
> # frog_species_list
> # maple_species_list
> # ant_species_list
```

Now it's time for you to explore these databases further for your assessment.

Tip: Retrieving the data for some of the functions can take a little while so while you are waiting for one function to run, you can think about and write the code for the next question.

Assessment

9) How many **species** (including **hybrid species** which are indicated by "X") of *Abies* were retrieved? How many of *Picea*?
10) Pick a **genus** that you're fond of and retrieve the names of **species** within that **genus** using one of the two **functions** illustrated in this module. *Requires internet* - if no internet, write the code you would use to do this

11) Retrieve the taxonomic level below species for the frog dataset. What do you find? Do the same for the conifer dataset. What about for the conifer dataset? (Hint: start with the **objects** containing the lists of **species** for frogs and for conifers. Copy an example of the code given earlier and edit it in a new line.)
12) Now, which of our five datasets has the most species? How did you calculate that?

Saving lists of species for future analysis

Now that we have our lists of species, we may want to save those names, or we may want to save the dataframes of resolved names that contain the OTT IDs so that we can query the Open Tree of Life to retrieve a phylogeny for our species. We will do both here to prepare ourselves for the next module in which we retrieve and visualize phylogenies for our organisms of interest.

The **objects** that are created in R are not saved unless we tell the program to save them. In R Projects, sometimes the **environment** can be saved, so objects that were read into R can be used the next time you open the Project. However, sometimes the results you produce need to be able to be shared with others which requires you to save the **object** to a file.

Here, we will save our lists of species as **.txt** files and our dataframes containing the OTT IDs as **.csv** files. We need to specify a **path** to where the files will be saved. Because we are working in an R Project, the **path** starts where our script is, so first we move up one level to the "Module_1" folder (*../*), and then specify the **Output** folder (*../Output/*) and the filename of our choice (*../Output/xxxx_species_names.txt*). Each time you enter a new folder or move back a level in the folder structure (*../*) a new section of the path needs to be added (separated by */*). We specify that we want our output files of species names to be **tab delimited** using the **function** *write.table()* and the term *sep = "\t"*. This means that each column would be separated by a tab in the file. We specify that we want our dataframes including the OTT IDs to be saved as **comma delimited files** using the **function** *write.csv()*. The endings of the file names typically indicate what type of file it is (eg **.csv** versus **.txt**).

```
> #write the objects to files - saving these files for future reference
> write.table(conifer_species_list, "../Output/conifer_species_names.txt", sep
    = "\t", col.names = FALSE, row.names = FALSE)
> write.table(bird_species_list, "../Output/bird_species_names.txt", sep = "\t
    ", col.names = FALSE, row.names = FALSE)
> write.table(frog_species_list, "../Output/frog_species_names.txt", sep = "\t
    ", col.names = FALSE, row.names = FALSE)
> write.table(maple_species_list, "../Output/maple_species_names.txt", sep = "
    \t", col.names = FALSE, row.names = FALSE)
> write.table(ant_species_list, "../Output/ant_species_names.txt", sep = "\t",
     col.names = FALSE, row.names = FALSE)
>
> write.csv(birds_resolved, "../Output/birds_OTT_IDs.csv", row.names = FALSE)
> write.csv(conifers_resolved, "../Output/conifers_OTT_IDs.csv", row.names =
    FALSE)
> write.csv(frogs_resolved, "../Output/frogs_OTT_IDs.csv", row.names = FALSE)
> write.csv(maples_resolved, "../Output/maples_OTT_IDs.csv", row.names = FALSE
    )
> write.csv(ants_resolved, "../Output/ants_OTT_IDs.csv", row.names = FALSE)
```

Congratulations! You have resolved taxonomic issues for several groups of organisms, retrieved OTT IDs for these taxa, obtained a list of species names for higher order taxa, and saved your species names and dataframes as files for future analysis.