

Material suplementario para “eidosapi, un paquete para estandarizar la taxonomía de especies en España”

Héctor Miranda-Cebrián*¹

(1) Instituto Pirenaico de Ecología - CSIC

Autor de correspondencia*: Héctor Miranda-Cebrián [hectorm94@gmail.com]

Palabras clave

Taxonomía; Estandarización; Nomenclatura; Lista Patrón

Keywords

Taxonomy; Standardization; Nomenclature; Checklist

Instalación

La instalación del paquete puede realizarse fácilmente desde R clonando el repositorio disponible en GitHub empleando la función `install_github` del paquete **remotes** (Cs'ardi et al., 2024).

```
# Instalación con remotes
remotes::install_github("https://github.com/hmirceb/eidos_api",
                        force = TRUE,
                        quiet = TRUE)

# Cargamos el paquete
library(eidosapi)
```

Ejemplos de uso

Busqueda de especies por nombre

Vamos a replicar un ejemplo de uso básico, buscar dos especies en la base de datos empleando la API. Para comprobar que la API también devuelve los sinónimos del taxón elegido vamos a emplear dos especies diferentes: el sapo partero ibérico (*Alytes cisternasii*), que no tiene sinónimos; y *Polygonum viviparum* que tiene varios. El procedimiento básico consiste en crear una tabla (*data frame*) con el género y la especie de cada taxón. De forma adicional podemos incluir una columna con la subespecie y la autoridad taxonómica que haya descrito el taxón, como se muestra en la Tabla 1. En el caso de que el taxón que nos interesa no tuviese subespecies o no conociésemos la autoría podemos omitir las columnas correspondientes o rellenarlas con *NA*.

genus	species	subspecies	scientificnameauthorship
Alytes	cisternasii		
Polygonum	viviparum		
Pinus	nigra	salzmannii	

```
# Tabla ejemplo:
taxa_list = data.frame(genus = c("Alytes", "Polygonum"),
                       species = c("cisternasii", "viviparum"))
eidos_results = eidosapi::eidos_taxon_by_name(
  taxon_list = taxa_list
)

# La tabla resultante tienes muchas columnas
# A modo de ejemplo se muestran solamente el género,
# especie, nombre completo, su identificador y el id
# del taxón aceptado:
eidos_results[c("supplied_genus", "supplied_species", "name",
                "idtaxon", "nametype", "acceptednameid")]
```

	supplied_genus	supplied_species		name	idtaxon
1	Alytes	cisternasii	Alytes cisternasii	Boscá, 1879	10909
2	Polygonum	viviparum	Polygonum viviparum	L.	7277
	nametype	acceptednameid			
1	Aceptado/válido	10909			
2	Sinónimo	32824			

```
# Obtendríamos el mismo resultado si en vez de una tabla
# usásemos un vector:
taxa_list = c("Alytes cisternasii", "Polygonum viviparum")
eidos_results = eidosapi::eidos_taxon_by_name(
  taxa_list = taxa_list
)
eidos_results[c("supplied_genus", "supplied_species", "name",
  "idtaxon", "nametype", "acceptednameid")]
```

	supplied_genus	supplied_species		name	idtaxon
1	Alytes	cisternasii	Alytes cisternasii	Boscá, 1879	10909
2	Polygonum	viviparum	Polygonum viviparum	L.	7277
	nametype	acceptednameid			
1	Aceptado/válido	10909			
2	Sinónimo	32824			

En caso de querer consultar una subespecie, esta puede escribirse como *Género especie subespecie* o *Género especie subsp. subespecie*.

```
# Usar el formato *Género especie subespecie* da resultados
# equivalentes a *Género especie subsp. subespecie*:
# Con subsp-
eidos_subsp1 = eidosapi::eidos_taxon_by_name(
  taxa_list = "Pinus nigra subsp. salzmannii"
)
head(
  eidos_subsp1[c("supplied_taxon", "supplied_species", "name",
    "idtaxon", "nametype", "acceptednameid")],
  n = 3)
```

	supplied_taxon	supplied_species		name	idtaxon	nametype
1	Pinus nigra salzmannii	nigra				
2	Pinus nigra salzmannii	nigra				
3	Pinus nigra salzmannii	nigra				
1		Pinus nigra	J.F. Arnold	7120	Aceptado/válido	
2		Pinus nigra	J.F. Arnold subsp. nigra	7121	Aceptado/válido	
3		Pinus nigra	subsp. salzmannii (Dunal) Franco	7122	Aceptado/válido	
	acceptednameid					
1	7120					
2	7121					
3	7122					

```
# Sin subsp.
eidos_subsp2 = eidosapi::eidos_taxon_by_name(
  taxon_list = "Pinus nigra salzmannii"
)
head(
  eidos_subsp2[c("supplied_taxon", "supplied_species", "name",
    "idtaxon", "nametype", "acceptednameid")],
  n = 3)
```

	supplied_taxon	supplied_species			
1	Pinus nigra salzmannii	nigra			
2	Pinus nigra salzmannii	nigra			
3	Pinus nigra salzmannii	nigra			
			name	idtaxon	nametype
1		Pinus nigra	J.F. Arnold	7120	Aceptado/válido
2		Pinus nigra	J.F. Arnold subsp. nigra	7121	Aceptado/válido
3	Pinus nigra subsp. salzmannii (Dunal)	Franco		7122	Aceptado/válido
	acceptednameid				
1	7120				
2	7121				
3	7122				

La tabla obtenida contiene las columnas correspondientes a la información que hayamos aportado, con sus nombres precedidos por el prefijo *supplied_*, y todas las columnas que devuelve la API de EIDOS por defecto. Entre estas columnas encontramos *idtaxon*, que nos permite hacer uso de otras funciones del paquete **eidosapi**. A este respecto, cabe destacar que a fecha de escritura de este documento la API de EIDOS cuenta con varias inconsistencias en la nomenclatura de las columnas de sus tablas, siendo la más importante que la columna *idtaxon* a veces aparece como *taxonid*. Todas las tablas producidas por cualquiera de las funciones del paquete **eidosapi** renombran la columna *taxonid* a *idtaxon* cuando sea necesario para mantener la consistencia.

Busqueda de especies por identificador

Estado de conservación

La columna *idtaxon* obtenida en el paso anterior contiene el identificador único para cada taxón de la base de datos. Si nos interesase saber si una especie presente en EIDOS, por ejemplo la gaviota de Audouin (*Larus audouinii*), tiene asociada alguna categoría de amenaza según los criterios de la UICN, solo tendríamos que obtener su identificador

con la función `eidos_taxon_by_name` y después emplearlo introduciendo en la función `eidos_conservation_by_id`.

```
# Buscamos el identificador por nombre:
eidos_results = eidosapi::eidos_taxon_by_name(
  taxon_list = "Larus audouinii"
)

# El identificador debería ser 14053:
print(eidos_results$idtaxon)
```

```
[1] 14053
```

```
# Accedemos a la información sobre su estado de conservación
eidos_cons = eidosapi::eidos_conservation_by_id(
  taxon_id = eidos_results$idtaxon
)

# Mostramos solo algunas columnas básicas:
eidos_cons[c("idtaxon", "anio", "categoriaconservacion", "aplicaa")]
```

	idtaxon	anio	categoriaconservacion	aplicaa
1	14053	2004	VU (Vulnerable)	España
2	14053	2018	LC (Preocupación menor)	Mundial
3	14053	2020	VU (Vulnerable)	Mundial
4	14053	2021	VU (Vulnerable)	Península

Así podemos saber que a nivel mundial en 2018 se le otorgó la categoría Preocupación menor (LC), pero esta fue modificada en 2020 a Vulnerable (VU), categoría que también se aplicaría a nivel de la Península Ibérica y de España desde los años 2021 y 2004 respectivamente.

Estado legal

Siguiendo este mismo procedimiento podríamos acceder al estado legal de una especie con la función `eidos_legal_status_by_id`. Esto nos permitiría saber qué categoría de conservación tiene la especie, si aparece en alguno de los anexos de la Directiva Hábitats, qué normas rigen esas categorías o el ámbito geográfico de las mismas.

```
# Buscamos el identificador por nombre:
eidos_results = eidosapi::eidos_taxon_by_name(
  taxon_list = "Larus audouinii"
)

# Accedemos a la información sobre su estado de conservación:
eidos_legal = eidosapi::eidos_legal_status_by_id(
  taxon_id = eidos_results$idtaxon
)

# Mostramos solo las parte porque las normas aparecen
# con su nombre completo y dificultan la visualización:
eidos_legal[1:2,
  c("idtaxon", "estadolegal", "ambito")]
```

	idtaxon	estadolegal	ambito
1	14053	Anexo I Internacional	
2	14053	Vulnerable	Autonómico

Información taxonómica

Y también podríamos volver a recuperar la información taxonómica del taxón si así lo deseásemos con la función `eidos_taxon_by_id`.

```
# Buscamos el identificador por nombre:
eidos_results = eidosapi::eidos_taxon_by_name(
  taxon_list = "Larus audouinii"
)

# Accedemos a la información sobre su estado de conservación:
eidos_taxo = eidosapi::eidos_taxon_by_id(
  taxon_id = eidos_results$idtaxon
)

eidos_taxo[c("nameid", "name", "nametype", "acceptednameid")]
```

	nameid	name	nametype
1	14053	Larus audouinii Payraudeau, 1826	Aceptado/válido
2	79314	Ichthyæetus audouinii (Payraudeau, 1826)	Sinónimo

	acceptednameid
1	14053
2	14053

Busqueda de especies con errores en la nomenclatura

Un problema común a la hora de trabajar con datos de especies son los errores de escritura como omitir letras o confundirlas con otras. El paquete **eidosapi** incluye la función `eidos_fuzzy_names` que, haciendo uso de lógica difusa gracias al paquete **fuzzyjoin** (Robinson, 2025), permite buscar en la base de datos de EIDOS los nombres que más se acerquen a la información que hayamos aportado. La función solo permite contrastar los nombres que aparezcan en la Lista patrón de las especies silvestres presentes en España (LP), y requiere que antes de emplearla descargemos la LP. Para facilitar esa tarea contamos con la función `eidos_clean_checklist`. En el caso de que no la hayamos descargado o se nos haya olvidado incluirla como argumento, la función `eidos_fuzzy_names` devolverá un error que nos avisará. Podemos comprobar un caso básico de uso con algunos nombres mal escritos.

```
# Creamos la tabla con la información que queremos contrastar:
taxa_list = data.frame(genus = c("Vorderea", "Alytes"),
                       species = c("pyrenaica", "cisternasi"))

# Obtendremos un error si no incluimos la LP como argumento:
eidosapi::eidos_fuzzy_names(taxa_list = taxa_list)
```

```
Error in eidosapi::eidos_fuzzy_names(taxa_list = taxa_list): Checklist missing.
Please run eidos_clean_checklist() and
include result in argument checklist
```

```
# O si no la hemos descargado previamente:
eidosapi::eidos_fuzzy_names(taxa_list = taxa_list,
                           checklist = checklist)
```

```
Error: objeto 'checklist' no encontrado
```

```
# Descargamos la LP y la guardamos en un objeto en el entorno de
# trabajo de R. Podríamos incluir la función eidos_clean_checklist
# directamente como argumento aunque no se recomienda porque, si
# fuésemos a realizar varias búsquedas con eidos_fuzzy_names el
# proceso se ralentizaría al tener que descargar la LP múltiples
# veces:
checklist = eidosapi::eidos_clean_checklist()
```

```
Downloading checklist and formatting, please wait...
```

```
# Ya podemos usar la función eidos_fuzzy_names:
eidos_result = eidosapi::eidos_fuzzy_names(taxa_list = taxa_list,
                                           checklist = checklist)

# Podemos comprobar que ha encontrado una coincidencia y
# el nombre aceptado:
eidos_result[c("idtaxon", "taxon_clean",
               "ScientificName", "WithoutAutorship")]
```

	idtaxon	taxon_clean	ScientificName	WithoutAutorship
1	2700	Borderea pyrenaica	Dioscorea pyrenaica Gren.	Dioscorea pyrenaica
2	10909	Alytes cisternasii	Alytes cisternasii Boscá, 1879	Alytes cisternasii

Afinamiento de búsquedas

Un problema que puede surgir a la hora de emplear esta función es que devuelva varias posibilidades muy dispares para un mismo taxón. Para solventar esto, la función incluye la posibilidad de incluir información taxonómica adicional (reino, filo, clase, orden y familia) en la tabla de datos. Se pueden incluir varias restricciones a la vez, por ejemplo clase y familia; y si queremos buscar varios taxa no es necesario aportar esta información adicional para todas ellas, bastará con poner *NA* en las celdas correspondientes. Para comprobar su utilidad, vamos a buscar información sobre el alcaudón real *Lanius meridionalis*, un ave, que coincide estrechamente con *Lasius meridionalis*, una hormiga.

```
# Creamos la tabla con la información que queremos contrastar:
taxa_list = data.frame(genus = c("Lanius"),
                       species = c("meridionalis"))
checklist = eidosapi::eidos_clean_checklist()
```

Downloading checklist and formatting, please wait...

```
# Si realizamos la búsqueda con esta información obtendremos
# dos coincidencias:
eidos_fuzzy1 = eidosapi::eidos_fuzzy_names(
  taxa_list = taxa_list,
  checklist = checklist)
eidos_fuzzy1[c("supplied_taxon", "idtaxon", "ScientificName", "class")]
```

	supplied_taxon	idtaxon	ScientificName	class
1	Lanius meridionalis	16426	Lanius meridionalis Temminck, 1820	Aves
2	Lanius meridionalis	56234	Lasius meridionalis (Bondroit, 1920)	Insecta


```
# Podemos refinar la búsqueda añadiendo,
# por ejemplo, la clase a la que pertenece nuestro taxón de interés:
taxa_list = data.frame(class = "Aves",
                        genus = "Lanius",
                        species = "meridionalis")
eidos_fuzzy2 = eidosapi::eidos_fuzzy_names(
  taxa_list = taxa_list,
  checklist = checklist)
eidos_fuzzy2[c("supplied_taxon", "idtaxon", "ScientificName", "class")]
```

	supplied_taxon	idtaxon	ScientificName	class
1	Lanius meridionalis	16426	Lanius meridionalis Temminck, 1820	Aves

Cabe destacar que también podemos buscar las especies aportando un vector con los nombres que queramos en vez de una tabla. Si queremos aportar información adicional habrá que hacerlo también como un vector que se incluirá como un argumento en la función (kingdom, phylum, class, order y/o family).

Además de esta posibilidad, la función `eidos_fuzzy_names` cuenta con varios argumentos extra heredados de la función `stringdist_join` del paquete **fuzzyjoin** (Robinson, 2025) que controlan el método para estimar las diferencias entre el nombre que aportemos y los que aparecen en la lista (method), la diferencia máxima entre el nombre aportado y alguno en la LP (maxdist), si queremos que en el resultado final aparezca una columna con estas diferencias (distance_col) y el tipo de unión que queremos con la LP en función del nombre aportado (mode). El método por defecto es “osa” (*optimal string alignment*), con el cual una distancia de 1 equivaldría a que los dos nombres contrastados se diferenciarían en una letra o carácter (e.g. *Lanius* y *Lasius*). Se puede encontrar información adicional sobre este y el resto de los métodos disponibles en la documentación del paquete **fuzzyjoin**. Por defecto la función `eidos_fuzzy_names` usa una distancia de 2, pero esta asunción puede relajarse. En cuanto al tipo de unión, salvo que lo especifiquemos explícitamente la función devuelve solamente los registros de la LP que coincidan con alguno de los que hayamos aportado y aparezcan en ambas tablas (*inner join*), aunque también podemos obtener la LP completa incluyendo nuestras especies de interés (*full join*) y otras variantes de este tipo de uniones entre tablas (*anti*, *left* y *right*).

Funciones adicionales

El paquete **eidosapi** cuenta con varias funciones adicionales, enfocadas principalmente a descargar información asociada a la [Lista patrón de las especies silvestres presentes en España](#). Entre estas funciones está la ya mencionada `eidos_clean_checklist`, que descarga la LP con los sinónimos disponibles en un formato largo, para facilitar el uso de la función

El funcionamiento de `eidos_fuzzy_names`. `eidos_clean_checklist` depende de la función `eidos_tables`, que permite descargar las diferentes versiones disponibles de la LP (con y sin sinonimias, con normativas y categorías de protección y con pasarelas a otras bases de datos) así como otras tablas consultables en el siguiente [enlace](#). Estas se refieren a listas de regiones biogeográficas, comunidades autónomas, provincias, normativas legales y demás, que podrían ser de cierta utilidad.

```
# Accedemos a las tablas de comunidades autónomas y de regiones biogeográficas:
head(
  eidos_tables("comunidades_autonomas")
)
```

	identificador	nombre	nut1	nut2
1	90	AGE	90	ES90
2	61	Andalucía	6	ES61
3	24	Aragón	2	ES24
4	70	Canarias	7	ES70
5	13	Cantabria	1	ES13
6	42	Castilla-La Mancha	4	ES42

```
head(
  eidos_tables("regbiogeograf_termar")
)
```

	id	regbio	nombre	tipo
1	8	MAC	Macaronésica	TERRESTRE
2	6	MMED	Marina Mediterránea	MARINA
3	3	MATL	Marina Atlántica	MARINA
4	9	MMAC	Marina Macaronésica	MARINA
5	7	MMED	Marina Mediterránea	MARINA
6	4	MATL	Marina Atlántica	MARINA

Referencias

Cs'ardi, G., Hester, J., Wickham, H., Chang, W., Morgan, M., Tenenbaum, D. 2024. [remotes: R Package Installation from Remote Repositories, Including 'GitHub'](#).

Robinson, D. 2025. [fuzzyjoin: Join Tables Together on Inexact Matching](#).