



**HAL**  
open science

# NSDPY: A python package to download DNA sequences from NCBI

Raphaël Hebert, Emese Meglécz

## ► To cite this version:

Raphaël Hebert, Emese Meglécz. NSDPY: A python package to download DNA sequences from NCBI. SoftwareX, 2022, 18, pp.101038. 10.1016/j.softx.2022.101038 . hal-03611377

**HAL Id: hal-03611377**

**<https://amu.hal.science/hal-03611377v1>**

Submitted on 22 Jul 2024

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

1 **NSDPY: A python package to download DNA sequences from NCBI**

2 **Raphael Hebert and Emese Meglécz**

3 raphaelhebert18@gmail.com, emese.meglecz@imbe.fr

4 **Aix Marseille Univ, Avignon Univ, CNRS, IRD, IMBE, Marseille, France**

5 Corresponding author:

6 Emese Meglécz

7 emese.meglecz@imbe.fr

8 Aix Marseille Univ, Avignon Univ, CNRS, IRD, IMBE

9 Chemin de la batterie des Lions

10 13007 Marseille FRANCE

11

12 **Abstract.**

13 Downloading large batches of DNA sequences can be useful to create custom databases containing for  
14 example sequences of a particular genomic region or a group of organisms. These sequences can be  
15 found on NCBI databases and accessed via a web browser (GUI) or directly via NCBI API. While the GUI is  
16 user-friendly, it lacks certain functionalities. On the other extreme, the use of the API is flexible but  
17 requires coding knowledge. NSDPY is a python package that combines flexibility and ease of use to  
18 download large amount of DNA sequences and includes several taxonomic or filtering options like batch  
19 downloading sequences for a list of taxa, downloading sequences including taxonomic lineage or  
20 filtering CDS sequences for a specific gene. NSDPY is available on PyPI, it is written to minimize  
21 dependencies on other packages and to be used directly from the terminal by simple command lines so  
22 that most users can use it without prior coding experience.

23

24 **Keywords:**

25 NCBI; Batch Download; CDS; Python; PyPI; Taxonomy

26

27 **Table 1 Code Metadata**

	<b>Code metadata description</b>	
C1	Current code version	<i>v.0.2.3</i>
C2	Permanent link to code/repository used of this code version	<i><a href="https://github.com/RaphaelHebert/nsdpy">https://github.com/RaphaelHebert/nsdpy</a></i>
C3	Code Ocean compute capsule	
C4	Legal Code License	<i>MIT</i>
C5	Code versioning system used	<i>Git</i>
C6	Software code languages, tools, and services used	<i>python</i>
C7	Compilation requirements, operating environments & dependencies	<i>Python 3.8+; Windows, Linux, Mac OS</i>
C8	If available Link to developer documentation/manual	<i><a href="https://nsdpy.readthedocs.io/en/latest/">https://nsdpy.readthedocs.io/en/latest/</a></i>
C9	Support email for questions	<i>raphaelhebert18@gmail.com</i>

28

## 29 **1. Motivation and Significance**

30 Downloading large batches of DNA sequences is becoming a common routine for molecular biologists  
31 for a wide range of purposes such as creating custom databases for example for taxonomic assignments  
32 in DNA metabarcoding projects[1,2], accessing sequences for phylogenetic studies[3], in silico PCR  
33 primer design and testing[4].

34 The National Center for Biotechnology Information (NCBI) hosts a large number of databases. One of the  
35 most frequently used and largest is the NCBI nucleotide database[5]. It is composed of GenBank, which  
36 contains primary nucleotide sequences from a large range of organisms and RefSeq, which is a non-  
37 redundant, well-annotated set of reference sequences. There are different ways to download large  
38 quantities of sequences from NCBI databases. The simplest and most straightforward way is using the  
39 Entrez search engine[6] via the NCBI website (NCBI GUI). It is an easy-to-use and versatile tool, since  
40 complex queries can be easily constructed. Sequences can be downloaded in different formats (fasta,  
41 GenBank, INSDSeqXML, TinySeqXML) but only formats relatively complicated to parse (GenBank and  
42 INSDSeqXML) contain both the sequence and the taxonomic lineage, thus it is necessary to have coding  
43 knowledge to obtain the file format desired by the users. Furthermore, the NCBI GUI may timeout if  
44 many sequences should be downloaded or if the connection is unstable; thus it is not well adapted for  
45 mass downloading.

46 The use of the NCBI API [7] is a powerful and versatile way for downloading sequences and libraries like  
47 Biopython [8] or the rentrez package in R [9] are also available to ease its use. However, they require  
48 programming skills and thus their use is quite complex for inexperienced users. The NCBI Mass  
49 Sequence Downloader [10] is an easy-to-use option that can be used in graphical or command-line  
50 mode and it is specifically designed for mass downloading. However, it lacks some useful options such as  
51 retrieving sequences with their taxonomic lineages and sorting these sequences according to them.  
52 The NSDPY (NCBI Sequence Downloader) python package aims to provide a solution to these problems  
53 while keeping the tool simple to use. NSDPY automatically downloads sequences based on an NCBI  
54 query sentence, eventually combined with a list of taxa of interest. It provides several options to  
55 retrieve taxonomic identifiers (TaxID) and taxonomic lineages and sorting sequences according to a  
56 given taxonomic level is also possible. Further options allow downloading Coding DNA Sequences (CDS)  
57 instead of the original fasta files and filter the results using regular expressions (a suite of characters  
58 that specifies a search pattern in text) to retain only relevant sequences if the original query cannot be  
59 sufficiently precise. The download is handled through the e-utilities to be able to handle lost connection  
60 and timeout. The script is written in python 3.8 and special care was taken to avoid unnecessary  
61 dependencies to limit maintenance issues and allow easy installation and use.

62

## 63 **2. Software Description**

### 64 *2.1. Software specifications*

65 NSDPY is a python package written in python 3.8. It uses the e-utilities tool [7] to access information in  
66 NCBI databases. NSDPY can be easily installed using PyPI (<https://pypi.org/project/nsdpy/>) or can be  
67 cloned directly from GitHub (<https://github.com/RaphaelHebert/nsdpy>) and run as a Python script. It  
68 can be freely modified since it is available under the MIT licence.

69 The package can be installed and run directly from a terminal so it can be directly included in a pipeline  
70 and does not require any prior knowledge of Python. Alternatively, it can be run from a Google Colab  
71 notebook to avoid using the user's own computer and connection resources and have the results  
72 available directly on Google Drive. This approach is not suitable for very large downloads as the Google  
73 Colab resources are available for a limited time and Google Drive storage limits the amounts of data that  
74 can be stored. However, it is very convenient to test, modify and run the code without having to install  
75 any third-party software.

76 The main functionality of NSDPY is the mass download of sequences from NCBI. To handle eventual  
77 timeouts or lost connection, NSDPY keeps track of the downloaded sequences and the sequences that  
78 are still to be downloaded. If the connection is lost, the process will wait for the connection to be re-  
79 established and resumes the download from the last correctly downloaded accession.  
80

## 81 *2.2. Software Functionalities*

82 NSDPY requires a query sentence just as the NCBI GUI searches[11]. The users can provide complex  
83 queries that combine several different search terms (e.g. MatK[Title], chloroplast[Filter], Viridiplantae  
84 [Organism]) linked by Boolean operators (and, or, not) and structured by parentheses. The basic usage  
85 of NSDPY is to download all relevant fasta files corresponding to the query and write them in one fasta  
86 file as a result. This basic usage can be completed by several options.

87 The default output format is a fasta file containing accession number and the definition field of the  
88 accession. However, the tsv option (-t, --tsv) produces a text file with tab-separated columns containing  
89 accession numbers (SequenceID), NCBI's numerical taxonomic identifiers (TaxID), sequence lengths and  
90 the sequences themselves. This file is particularly easy to filter for users without programming skills.

91 There are several taxonomy related options. The taxids option (-T, --taxids) prompts NSDPY to write a  
92 text file with SequenceIDs and TaxIDs. The information option (-i, --information) modifies the fasta  
93 definition line to write taxon name, SequenceID, TaxID and taxonomic lineage to the output fasta  
94 definition lines. It also completes the tsv file with the taxonomic lineage when the tsv option is evoked.  
95 At last, the downloaded sequences can be sorted to different fasta files according to a selected  
96 taxonomic level using the following mutually exclusive options: -k (kingdom), -p (phylum), -s (the lowest  
97 taxonomic level in the lineage) or preparing one file for each of the taxa specified by the user with the -l  
98 option (e.g. -l Crocodylia Lepidosauria Testudines).

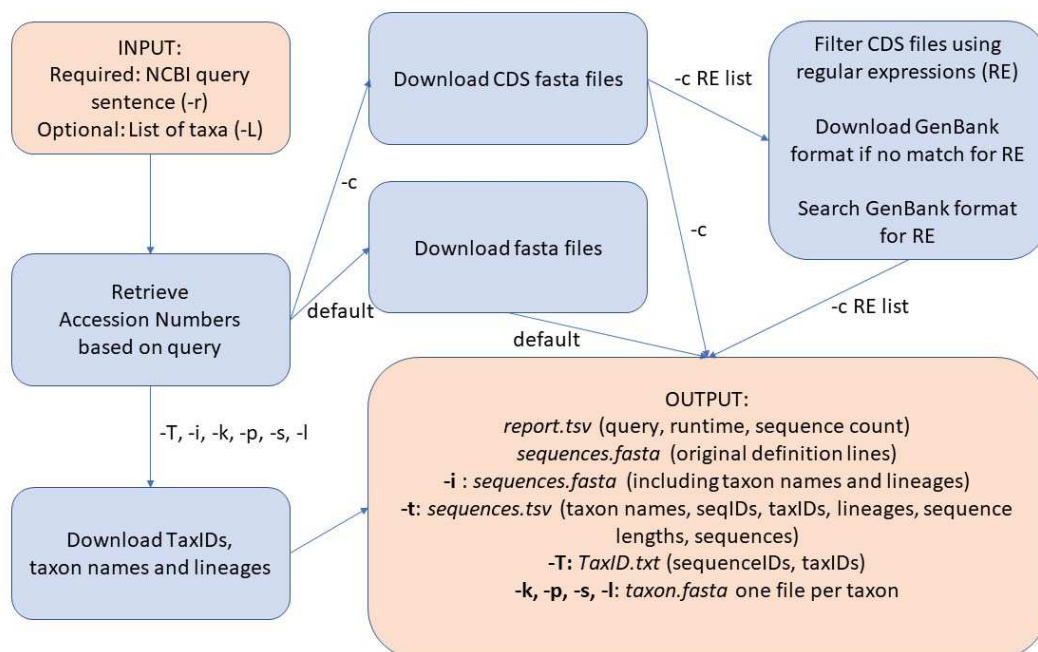
99 When a large list of taxa should be included in the input query sentence, the user can provide an input  
100 file containing the list of taxa using the -L option. The query will return all sequences that belong to any  
101 of the taxa listed in the input file and satisfy the criteria of the query sentence.

102 In case of coding sequences, the users can choose to download the Coding DNA Sequences (CDS) fasta  
103 files (-c option), instead of the complete fasta file. This helps to eliminate non-coding sequences and  
104 obtain the coding sequences in the correct orientation. Furthermore, the -c option can be followed by a  
105 list of regular expressions, to retain only coding sequences with at least one of the regular expressions in  
106 the definition line. This feature is particularly helpful to isolate coding sequences from long sequences

107 such as mitogenomes. Occasionally, due to incomplete annotations, the CDS fasta file might not contain  
 108 any of the regular expressions. In these cases, NSDPY downloads the accessions in GenBank format, and  
 109 tries to identify the desired coding sequences based on other fields such as note, gene synonym or  
 110 product.

111 The workflow is illustrated in Fig. 1. The input is an NCBI Entrez query sentence, or optionally a text file  
 112 with the list of taxa of interest. The main output is a fasta file with a downloaded and eventually filtered  
 113 sequences with original or modified identification lines with taxonomic information included (-i option).  
 114 Additionally, sequences can also be written into a tsv file together with their metadata (SeqId, TaxID,  
 115 taxonomic lineage, sequence length, sequence). A summary file (report.tsv) contains the query  
 116 sentence, runtime, and information on the run.

117



118

119 Fig. 1. Workflow of NSDPY. RE: Regular Expression.

120

### 121 3. Illustrative Examples

122 To illustrate the pros and cons of the script we ran it with three queries that illustrate typical uses of the  
 123 script and use the main functionalities of NSDPY. We used queries that may be used to retrieve the  
 124 cytochrome oxidase I (COI) sequences to make a custom database. For these examples, we installed  
 125 NSDPY as a package from PyPI and ran the commands directly from the terminal.

126

#### 127 3.1. First example

128 We used the following command to isolate the COI sequences from complete mitochondrial sequences:

129  
130 `nsdpy -r "(mitochondrion[Title] AND complete[Title])" -c 'cox[li]' 'co[i1]' -`  
131 `v`

132  
133 In this query, the request (-r) helped to identify complete mitochondrial sequences. Since we are  
134 interested in extracting COI sequences from these mitogenomes, we used a -c (--cds) option to  
135 download the fasta files containing the CDS of the mitochondria, instead of the fasta files containing the  
136 complete records. Furthermore, a list of regular expressions was given after the -c option to filter the  
137 CDS file and to extract sequences with annotation corresponding to the provided regular expressions.  
138 When no CDS fasta file was available, or no COI gene found in it, the script downloaded and searched  
139 the accession in GenBank format (.gb). The annotation in GenBank files may be organized differently  
140 depending on the authors: in most cases, COI is annotated in the gene field (e.g. KY052129.1) while in  
141 others in the note (e.g. NC\_046833.1, AY191994.1, EU273708.1), gene\_synonym (e.g. AB475096.1, ) or  
142 product (e.g. CP014249.1) fields.

143 Note that the -v option stands for verbose. It just allows the user to see how many percent of the  
144 sequences have been downloaded.

145 In short, in this query, the request helped to download complete mitochondrial sequences which would  
146 not have been found by a keyword search for COI gene and adding the regular expression helped to  
147 filter the sequences to isolate the desired gene.

148 The results are as follows:

- 149 • 98 398 coding sequences including 97 149 sequences identified from the fasta files containing  
150 the CDS and 1 249 sequences extracted from the GenBank files, where the COI gene was not  
151 available in the fasta file.
- 152 • 6 301 accession numbers in the notfound.txt file, where the COI gene could not be identified  
153 (e.g. MH500775.1).
- 154 • 252 accession numbers in the duplicates.txt file with accession numbers that returned more  
155 than one coding sequence for the COI. For example, the accession MK562756.1 has two genes  
156 annotated as COX1 in its GenBank file.

### 157 158 3.2. Second example

159 We used the following command to download all COI sequences using keywords present in the title  
160 fields and sort the sequences by kingdom:

161 `nsdpy -r "(COX1[Title] OR COXI[Title] OR CO1[Title] OR COI[Title])" -k -T`

162 The request (-r) combines a series of abbreviations frequently used to refer to the cytochrome oxidase  
163 subunit1 gene. The search is limited to the title field of the accessions, to avoid false positives.

164 Automatically sorting the sequences according to given taxonomic levels can help users in post-  
165 processing the sequences. The kingdom option (-k) sorts the sequences according to the kingdoms that  
166 are pre-recorded in the script. The -T (or --taxids) option produces a text file with all the accession  
167 numbers returned by NCBI for a user's query and the corresponding NCBI taxonomic IDs.

168 The results are as follows:

- 169 • METAZOA.fasta file containing 3 637 305 sequences

- 171 • PLANTAE.fasta file containing 25 987 sequences
- 172 • FUNGI.fasta file containing 2 089 sequences
- 173 • OTHERS.fasta file containing 30 371 sequences with lineage that does not correspond to
- 174 any of the above three kingdoms (e.g. MZ152022.1 which is a heterokont).
- 175 • The no\_taxid.fasta file contains 31 581 sequences for which no taxonomic information
- 176 could be found. In some cases, the taxonomic identifiers available in the downloaded files are
- 177 out of date, and not easily accessible from the taxonomy database.
- 178 • The TaxIDs.txt file contains all accession numbers and their TaxIDs in two, tabulation
- 179 separated columns.

180

### 181 3.3. Third example

182 We used the following command to download mitochondrial (mitochondrion[Filter]) sequences of four  
 183 Lepidopteran genera (Parnassius[ORGN] OR Pieris[ORGN] OR Melithaea[ORGN] OR Iphiclidides[ORGN])  
 184 published in NCBI nucleotide database since 2010 ("2010"[Publication Date] : "3000"[Publication Date]),  
 185 with a barcode present in the keyword field (barcode[Keyword]):

186

```
187 nsdpy -r "(Parnassius[ORGN] OR Pieris[ORGN] OR Melithaea[ORGN] OR
188 Iphiclidides[ORGN]) AND COI[Title] AND (2010[Publication Date] :
189 3000[Publication Date]) AND barcode[Keyword] AND mitochondrion[Filter]" -i -t
```

190

191 The request (-r) combines a series of search terms, each of them limited to a particular field (e.g.  
 192 Organism, Publication Date) of the GenBank record. Note the use of OR to link the different names of  
 193 the genera, to obtain sequences from all of them, and the use of parentheses to correctly structure the  
 194 query. The -i option replaces the original definition lines to give taxonomically informative format more  
 195 adapted for phylogenetic studies: taxon\_name-SeqID | TaxID | Taxonomic lineage with comma-  
 196 separated values. The -t option produces a tsv file with the following fields: Taxon name, SequenceID,  
 197 TaxID, Taxonomic lineage, Sequence length, Sequence.

198 Note, the exact same result can be obtained by the following command:

199

```
200 nsdpy -r "COI[Title] AND (2010[Publication Date] : 3000[Publication Date])
201 AND barcode[Keyword] AND mitochondrion[Filter]" -i -t -L genus_list.txt
```

202

203 The genus\_list.txt is a text file with the four Lepidopteran genera of interest (one genus name in each  
 204 line). We used just a short list, in this example, but it can contain thousands of taxa of any taxonomic  
 205 level.

206

207 The command has downloaded 1447 sequences form 55 taxa (genus, species, or subspecies).

208

## 209 4. Impact

210

211 High throughput sequencing (HTS) has gone through a spectacular development in the last 20 years.  
 212 Sequencing throughput increased, while sequencing cost decreased exponentially[12]. This has

213 revolutionized molecular biology and now HTS is routinely used in very diverse fields of biology such as  
214 medical research[13–15], transcriptome analyses[16], genome structure[17], genomic variation and  
215 biodiversity studies [18–20]. Many of these fields require the use of custom databases, but  
216 metabarcoding is one of the fields where the need of custom databases is most pressing[21].  
217 Metabarcoding has become a well-established method for biodiversity surveys[22]. The number of  
218 metabarcoding studies has dramatically increased in the past ten years. Google Scholar returned 124  
219 studies with metabarcoding as a keyword published in 2011, while this number is about 5500 for 2021.  
220 One of the key aspects of metabarcoding is the taxonomic assignment of the sequences, which relies on  
221 a reference database as complete as possible. The NCBI nucleotide database contains nearly 500 million  
222 sequence records[5] and this number is steadily increasing. This provides an excellent resource for  
223 taxonomic assignment due to its large taxonomic coverage. However, due to its size, direct comparison  
224 of sequences to this database is impractical. Therefore, there is a need to establish custom databases of  
225 molecular markers and/or taxonomic groups and to maintain them up to date[21,23–25].  
226 The download of large amounts of DNA sequences is inevitable when constructing custom databases.  
227 NSDPY is a practical tool to automate the download, and to sort and filter the sequences. Accessing  
228 taxonomic information is particularly important for creating custom databases for biodiversity studies  
229 and particularly for metabarcoding projects. Precise taxonomic information (TaxID, Lineage) is seldom  
230 included in standard downloading formats, or if present, further processing is necessary that requires  
231 some programming knowledge. Thus, the ability to download the taxonomic information and sort the  
232 sequences to different files according to a given taxonomic level can save a considerable amount of time  
233 and helps to avoid sorting errors. Another time-saving option is the possibility to download sequences  
234 from a long list of taxa with a single command.  
235 The possibility of filtering the sequences by a gene name (provided as a regular expression) is a useful  
236 feature when downloading whole organelle sequences such as mitochondria or selecting different genes  
237 from long sequences. This feature is very useful, since it allows users to obtain a set of high-quality  
238 sequences where the gene is complete and correctly oriented. When searching for genes of organelles,  
239 these high-quality sequences cannot be easily accessed by classic keyword searches. Thus, NSDPY fills a  
240 gap and helps users to easily obtain complete sequences of a gene from organelle genomes.  
241 With NSDPY, if a disconnection occurs while downloading, the script will wait for the connection to be  
242 re-established to resume the download thus avoids restarting the procedure and losing time.  
243 This software package is meant to be used to download large amount of DNA sequences from NCBI  
244 databases to generate custom reference databases. The software can be included in bioinformatic  
245 pipelines. The automation of gathering the sequences with their taxonomic information, sorting, and  
246 filtering the sequences without having to learn how to code can drastically improve the workflow of  
247 users. For the users with coding experience, the script can be easily modified to be adapted to other  
248 specific needs.

249

## 250 **5. Conclusions**

251 In this paper, we described the implementation of NSDPY and its built-in features and provided  
252 examples of possible usages. As it is written under the MIT licence and can be easily customized, it is  
253 expected that NSDPY will be developed to adapt to more specific work. It meant to support the work of  
254 researchers who need to work with custom databases of DNA sequences.



255 **Conflict of Interest**

256 We wish to confirm that there are no known conflicts of interest associated with this publication and  
257 there has been no significant financial support for this work that could have influenced its outcome.

258

259 **Acknowledgements**

260 IMBE has covered the Article Processing Charges. We thank two anonymous reviewers for their constructive  
261 comments to improve NSDPY and the manuscript and Judith Nève for proofreading the manuscript.

262

263 **Author contributions**

264 Raphael Hebert: Conceptualization, Methodology, Software, Validation, Writing - Original Draft

265 Emese Meglécz: Supervision, Conceptualization, Validation, Writing- Reviewing and Editing

266

267 **References**

- 268 [1] V. Arranz, W.S. Pearman, J.D. Aguirre, L. Liggins, *Scientific Data* 7 (2020) 209.
- 269 [2] J.-N. Macher, T.-H. Macher, F. Leese, *Metabarcoding and Metagenomics* 1 (2017) e22262.
- 270 [3] M. Gouy, S. Guindon, O. Gascuel, *Molecular Biology and Evolution* 27 (2010) 221–224.
- 271 [4] V. Elbrecht, F. Leese, *Methods in Ecology and Evolution* 8 (2017) 622–626.
- 272 [5] E.W. Sayers, E.E. Bolton, J.R. Brister, K. Canese, J. Chan, D.C. Comeau, R. Connor, K. Funk, C. Kelly, S.  
273 Kim, T. Madej, A. Marchler-Bauer, C. Lanczycki, S. Lathrop, Z. Lu, F. Thibaud-Nissen, T. Murphy, L.  
274 Phan, Y. Skripchenko, T. Tse, J. Wang, R. Williams, B.W. Trawick, K.D. Pruitt, S.T. Sherry, *Nucleic  
275 Acids Research* 50 (2022) D20–D26.
- 276 [6] E.W. Sayers, T. Barrett, D.A. Benson, E. Bolton, S.H. Bryant, K. Canese, V. Chetvernin, D.M. Church,  
277 M. Dicuccio, S. Federhen, M. Feolo, I.M. Fingerman, L.Y. Geer, W. Helmberg, Y. Kapustin, S. Krasnov,  
278 D. Landsman, D.J. Lipman, Z. Lu, T.L. Madden, T. Madej, D.R. Maglott, A. Marchler-Bauer, V. Miller, I.  
279 Karsch-Mizrachi, J. Ostell, A. Panchenko, L. Phan, K.D. Pruitt, G.D. Schuler, E. Sequeira, S.T. Sherry,  
280 M. Shumway, K. Sirotkin, D. Slotta, A. Souvorov, G. Starchenko, T.A. Tatusova, L. Wagner, Y. Wang,  
281 W.J. Wilbur, E. Yaschenko, J. Ye, *Nucleic Acids Res.* 40 (2012) D13-25.
- 282 [7] J. Kans, *Entrez Direct: E-Utilities on the Unix Command Line*, National Center for Biotechnology  
283 Information (US), 2021.
- 284 [8] B. Chapman, J. Chang, *SIGBIO Newsl.* 20 (2000) 15–19.
- 285 [9] D.J. Winter, *Rentrez: An R Package for the NCBI EUtils API*, PeerJ Inc., 2017.
- 286 [10] F. Pina-Martins, O.S. Paulo, *SoftwareX* 5 (2016) 80–83.
- 287 [11] M.D. Bethesda, *Entrez Help*, National Center for Biotechnology Information (US), 2016.
- 288 [12] J.A. Reuter, D. Spacek, M.P. Snyder, *Mol Cell* 58 (2015) 586–597.
- 289 [13] W.W. Soon, M. Hariharann, M.P. Snyder, *Molecular Systems Biology* 9 (2013) 640.
- 290 [14] G. Lightbody, V. Haberland, F. Browne, L. Taggart, H. Zheng, E. Parkes, J.K. Blayney, *Briefings in  
291 Bioinformatics* 20 (2019) 1795–1811.
- 292 [15] C. Manzoni, D.A. Kia, J. Vandrovcova, J. Hardy, N.W. Wood, P.A. Lewis, R. Ferrari, *Brief Bioinform* 19  
293 (2018) 286–302.
- 294 [16] Z. Jiang, X. Zhou, R. Li, J.J. Michal, S. Zhang, M.V. Dodson, Z. Zhang, R.M. Harland, *Cell Mol Life Sci* 72  
295 (2015) 3425–3439.
- 296 [17] A.C. Brant, W. Tian, V. Majerciak, W. Yang, Z.-M. Zheng, *Cell & Bioscience* 11 (2021) 136.
- 297 [18] K.R. Andrews, J.M. Good, M.R. Miller, G. Luikart, P.A. Hohenlohe, *Nat Rev Genet* 17 (2016) 81–92.
- 298 [19] K.M. Ruppert, R.J. Kline, M.S. Rahman, *Global Ecology and Conservation* 17 (2019) e00547.
- 299 [20] P. Arribas, C. Andújar, M.I. Bidartondo, K. Bohmann, É. Coissac, S. Creer, J.R. deWaard, V. Elbrecht,  
300 G.F. Ficetola, M. Goberna, S. Kennedy, H. Krehenwinkel, F. Leese, V. Novotny, F. Ronquist, D.W. Yu,

301 L. Zinger, T.J. Creedy, E. Meramveliotakis, V. Noguerales, I. Overcast, H. Morlon, A.P. Vogler, A.  
302 Papadopoulou, B.C. Emerson, *Mol Ecol* 30 (2021) 1120–1135.

303 [21] H. Weigand, A.J. Beermann, F. Čiampor, F.O. Costa, Z. Csabai, S. Duarte, M.F. Geiger, M. Grabowski,  
304 F. Rimet, B. Rulik, M. Strand, N. Szucsich, A.M. Weigand, E. Willassen, S.A. Wyler, A. Bouchez, A.  
305 Borja, Z. Čiamporová-Zaťovičová, S. Ferreira, K.-D.B. Dijkstra, U. Eisendle, J. Freyhof, P. Gadawski, W.  
306 Graf, A. Haegerbaeumer, B.B. van der Hoorn, B. Japoshvili, L. Keresztes, E. Keskin, F. Leese, J.N.  
307 Macher, T. Mamos, G. Paz, V. Pešić, D.M. Pfannkuchen, M.A. Pfannkuchen, B.W. Price, B. Rinkevich,  
308 M.A.L. Teixeira, G. Várbíró, T. Ekrem, *Science of The Total Environment* 678 (2019) 499–524.

309 [22] K. Deiner, H.M. Bik, E. Mächler, M. Seymour, A. Lacoursière-Roussel, F. Altermatt, S. Creer, I. Bista,  
310 D.M. Lodge, N. de Vere, M.E. Pfrender, L. Bernatchez, *Molecular Ecology* 26 (2017) 5872–5895.

311 [23] R.T. Richardson, J. Bengtsson-Palme, M.M. Gardiner, R.M. Johnson, *PeerJ* 6 (2018) e5126.

312 [24] J. Decelle, S. Romac, R.F. Stern, E.M. Bendif, A. Zingone, S. Audic, M.D. Guiry, L. Guillou, D. Tessier, F.  
313 Le Gall, P. Gourvil, A.L. Dos Santos, I. Probert, D. Vaultot, C. de Vargas, R. Christen, *Molecular Ecology*  
314 *Resources* 15 (2015) 1435–1445.

315 [25] R.H. Nilsson, K.-H. Larsson, A.F.S. Taylor, J. Bengtsson-Palme, T.S. Jeppesen, D. Schigel, P. Kennedy,  
316 K. Picard, F.O. Glöckner, L. Tedersoo, I. Saar, U. Kõljalg, K. Abarenkov, *Nucleic Acids Research* 47  
317 (2019) D259–D264.

318