

Apscale manual

Last updated: 08.08.2025, apscale version 4.1.1

Author: Dominik Buchner

Table of contents

1. Introduction	3
2. Installation	3
3. Usage.....	4
3.1. Scalability and resource requirements	4
3.2. Creating a project.....	5
3.3. Configuring the settings	5
3.4. Adding data.....	7
3.5. Running a module – Linear workflow	7
3.5.1. Running the full pipeline	7
3.5.2. Paired-end merging	8
3.5.3. Primer trimming	8
3.5.4. Quality filtering	8
3.5.5. Dereplication	10
3.6. Running a module – Modular workflow.....	10
3.6.1. Denoising	11
3.6.2. Swarm clustering	12
3.6.3. Replicate merging.....	12
3.6.4. Negative control filtering.....	13
3.6.5. Read table generation and sequence grouping	14
3.7. Working with the read data store	14
3.7.1. Apscale analyze	14
3.7.2. Adding sample metadata	15
3.7.3. Adding sequence metadata	17
3.7.4. Correcting species names via GBIF.....	18
3.7.5. Validating species occurrence via GBIF	20
3.7.6. Uploading datasets to ENA.....	22
3.7.7. Exporting read tables	23

1. Introduction

Apscale is a metabarcoding pipeline that handles the most common tasks in metabarcoding pipelines like paired-end merging, primer trimming, quality filtering, denoising, swarm and threshold-based clustering as well as basic data handling operations such as replicate merging and the removal of reads found in the negative controls. It uses a simple command line interface and is configured via a single configuration file. To add metadata to the dataset, a simple, browser-based interface has been introduced in version 4.0. Apscale automatically uses the available resources on the machine it runs on while still providing the option to use less if desired. All modules can be run on their own or as a comprehensive workflow.

Several different programs are called within the workflow. These include vsearch (Rognes et al. 2016), cutadapt (Martin 2011) and swarm (Mahé et al. 2021). Please cite those accordingly, when using apscale. A DuckDB backend was introduced in version 4.1. Please also cite the authors of DuckDB, they build an amazing tool that makes the current developments in Apscale possible.

Apscale (Buchner et al. 2022) has also been published and we are happy if we are cited too.

2. Installation

Apscale can be installed on all common operating systems (Windows, Linux, MacOS). Apscale requires Python 3.11 or higher and can be easily installed via pip in any command line:

```
pip install apscale
```

To update apscale run:

```
pip install --upgrade apscale
```

Apscale calls vsearch as well as swarm for multiple modules. Both programs should be installed and be in PATH to be executed from anywhere on the system.

Check the vsearch and swarm Github pages for further info:

<https://github.com/torognes/vsearch>

<https://github.com/torognes/swarm>

To check if every is correctly set up, please type this into your command line:

```
vsearch --version
```

```
swarm --version
```

It should return messages similar to these:

```
vsearch v2.30.0_win_x86_64, 127.9GB RAM, 24 cores
https://github.com/torognes/vsearch

Rognes T, Flouris T, Nichols B, Quince C, Mahe F (2016)
VSEARCH: a versatile open source tool for metagenomics
PeerJ 4:e2584 doi: 10.7717/peerj.2584 https://doi.org/10.7717/peerj.2584

Compiled with support for gzip-compressed files, and the library is loaded.
zlib version 1.2.13, compile flags 65
Compiled with support for bzip2-compressed files, and the library is loaded.

Swarm 3.1.5
Copyright (C) 2012–2024 Torbjorn Rognes and Frederic Mahe
https://github.com/torognes/swarm

Mahe F, Rognes T, Quince C, de Vargas C, Dunthorn M (2014)
Swarm: robust and fast clustering method for amplicon-based studies
PeerJ 2:e593 https://doi.org/10.7717/peerj.593

Mahe F, Rognes T, Quince C, de Vargas C, Dunthorn M (2015)
Swarm v2: highly-scalable and high-resolution amplicon clustering
PeerJ 3:e1420 https://doi.org/10.7717/peerj.1420

Mahe F, Czech L, Stamatakis A, Quince C, de Vargas C, Dunthorn M, Rognes T (2022)
Swarm v3: towards tera-scale amplicon clustering
Bioinformatics 38:1, 267–269 https://doi.org/10.1093/bioinformatics/btab493
```

Further dependencies – cutadapt:

Apscale also calls cutadapt with the primer trimming module. Cutadapt should be downloaded and installed automatically with the Apscale installation. To check this, type:

```
cutadapt --version
```

and it should return the version number, for example:

```
5.1
```

3. Usage

3.1. Scalability and resource requirements

The main strength of the Apscale pipeline lies in its ability to analyze large to massive datasets with modest resource requirements. In particular, the later modules—such as the generation of read tables—can effortlessly handle millions of distinct sequences across thousands of samples. While Apscale runs on virtually any system, we recommend a minimum of 16 GB of RAM for processing large datasets. Apscale is parallelized by default, automatically utilizing all available CPU cores. Additionally, when memory is insufficient, Apscale will spill data to disk. Although this may slow down processing, it ensures that even the most memory-intensive tasks can be completed.

3.2. Creating a project

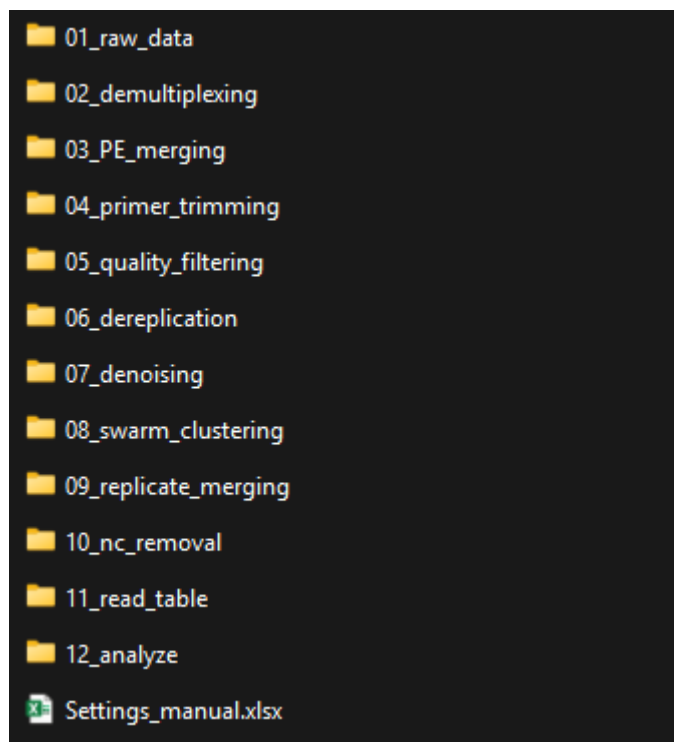
Apscale is organized into projects. A project is a directory that contains subdirectories for the different steps of the pipeline, along with a settings file. All data generated by Apscale will be saved in the corresponding subdirectory within the project's file system.

To create a new project, run:

```
apscale --create_project PROJECT_NAME
```

This command will generate a folder structure similar to the one shown below. If only a project name is provided, Apscale will create the project in the current working directory. To specify a different location, include the full path in the project name:

```
PS C:\Users\Dominik\Desktop> apscale --create_project manual
04:58:06: "manual_apscale" created as a new project folder.
```



3.3. Configuring the settings

All settings required by Apscale are configured within the project's settings file. Each processing step has its own sheet in the document. The first tab you'll see when opening the settings file is "0_general_settings". In this tab, you can define how many CPU cores Apscale should use for processing. By default, it uses the total number of available cores minus two.

You can also set the gzip compression level here. Apscale compresses all output files to conserve disk space. The default compression level is 6, which is suitable for most use cases. If you need to save more space, you can increase it to 9—this will produce smaller files but may slow down processing.

Most settings come with default values, except for those in the “04_primer_trimming” and “05_quality_filtering” modules. For details on the required input, please refer to the corresponding sections.

Setting	Default	Possible values	Effect
cores to use	Available cores – 2	1 – available cores	Manages parallelization in Apscale
compression level	6	1 – 9	Controls the gzip compression level. Lower values produce larger files but allow for faster processing; higher values result in smaller files at the cost of increased processing time.

3.4. Adding data

Apscale processes demultiplexed paired-end sequencing data. It requires one pair of files per sample, with any consistent naming pattern (e.g., sample1_f1.fastq.gz and sample1_r1.fastq.gz). Input files do not need to be unzipped—Apscale can handle gzip-compressed files directly.

If your data is already demultiplexed, place the read files in the 02_demultiplexing/data directory. If additional demultiplexing is required (e.g., when using inline barcodes), we recommend storing the raw data in 01_raw_data/data and writing the output of your demultiplexing script to 02_demultiplexing/data. This ensures that all data remains organized within the project's directory structure.

Note that Apscale does not perform demultiplexing itself, as there are many different tagging and barcoding schemes. However, we provide a dedicated tool for this purpose called `demultiplexer2`, available at: <https://github.com/DominikBuchner/demultiplexer2>.

3.5. Running a module – Linear workflow

The steps in the following section will be executed for all projects in a linear sequence. The later modules are modular and can be run or skipped in any combination, depending on the user's objectives. If you are already in the project directory, all commands follow the same logic. Specifying the project path is optional and only necessary if you are running the command from outside the project directory.

```
apscale --COMMAND [project_path]
```

Each processing step also generates a logfile that records detailed information about the step's execution..

3.5.1. Running the full pipeline

If the settings are fully configured and you want to run the complete analysis, all pipeline steps can be executed with a single command. This will run the entire pipeline using the defined settings. For more control, individual steps can also be run independently.

```
apscale --run_apscale [project_path]
```

3.5.2. Paired-end merging

The first step performed by Apscale is merging paired-end reads using vsearch. The default settings are fairly relaxed to merge the largest possible portion of reads, as quality filtering is handled in later steps.

To run this module, use the following command:

```
apscale --pe_merging [project_path]
```

Setting	Default	Possible values	Effect
maxdiffpct	25	0-100	Maximum percentage of non-matching nucleotides allowed in the overlap region
maxdiffs	199		Maximum number of non-matching nucleotides allowed in the overlap region
minovlen	5		Minimum overlap between the merged reads

3.5.3. Primer trimming

The next step performed by Apscale is primer trimming, which removes the primers used for target amplification since they do not contain biologically relevant information.

To run this module, use the following command:

```
apscale --primer_trimming [project_path]
```

Setting	Default	Possible values	Effect
P5 Primer (5' - 3')	-	-	Primer that can be found at the P5 end of the reads. Usually, the forward primer
P7 Primer (5' - 3')	-	-	Primer that can be found at the P7 end of the reads. Usually, the forward primer
anchoring	False	True / False	Whether the read starts directly with the primer sequence or not. If set to False, Apscale will, for example, locate the primer sequence even if it appears after an inline tag.

3.5.4. Quality filtering

After primer trimming, Apscale performs quality filtering. This step filters out reads with an expected error higher than the threshold defined in the settings, as well as sequences whose lengths fall outside the specified target range. Typically, we use a tolerance of ± 10 bases around the target length to allow for some biological variation while removing artifacts such as primer dimers.

To run this module, use the following command:

```
apscale --quality_filtering [project_path]
```

Setting	Default	Possible values	Effect
maxEE	1	0 - infinity	Maximum number of expected errors allowed per read before it is filtered out. For example, a value of 1 means that all reads with an expected error greater than or equal to 1 will be removed.
min length	-	-	Minimum length of the sequence
max length	-	-	Maximum length of the sequence

3.5.5. Dereplication

Before running the modular workflow, the reads from all samples must be dereplicated. This step does not alter the data itself but optimizes how it is stored. The output is a FASTA file containing all unique sequences with size annotations (e.g., >seq_1;size=100).

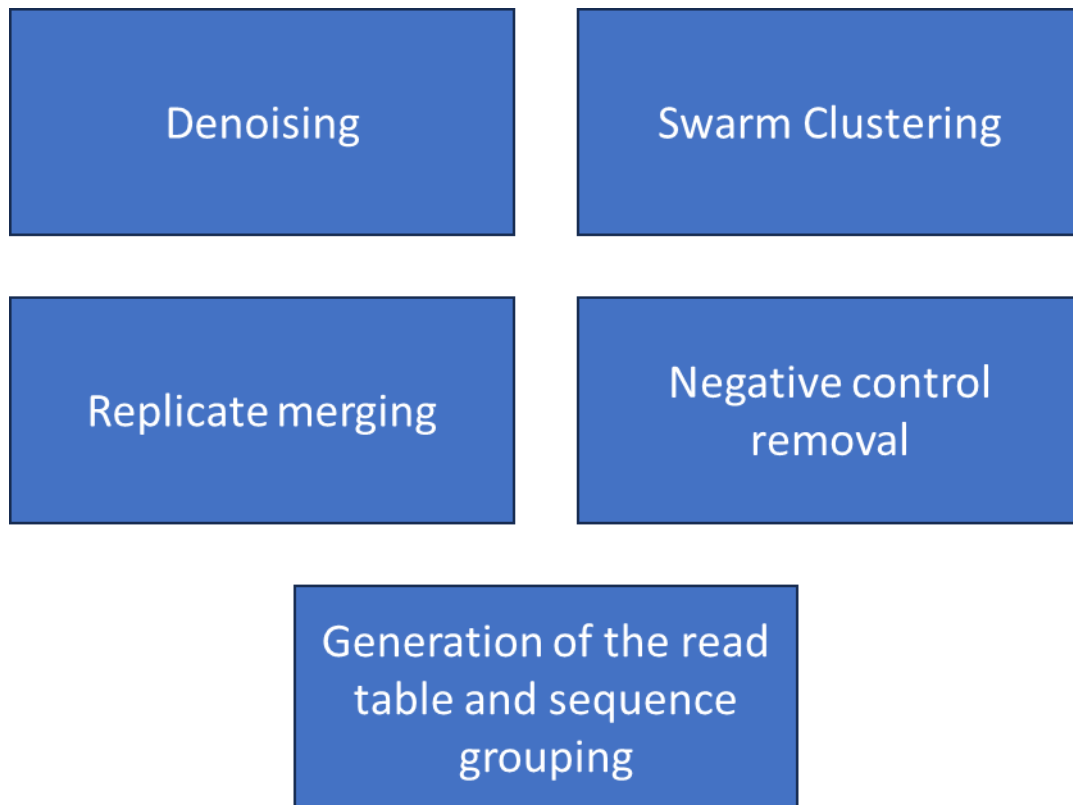
To run this module, use the following command:

```
apscale --dereplication [project_path]
```

Setting	Default	Possible values	Effect
minimum sequence abundance	1	1 - infinity	Minimum abundance required for a sequence to be retained in the dataset. For example, a value of 2 will remove all singletons (sequences that appear only once)

3.6. Running a module – Modular workflow

After dereplication, Apscale has been modularized starting with version 4.0.0, meaning that all subsequent modules are optional. These steps can be freely combined, skipped, or disabled entirely, depending on your specific analysis goals. Each module uses the output of the previously executed step as its input, ensuring a flexible and customizable workflow.



3.6.1. Denoising

The denoising module performs sequence denoising using vsearch, processing each sample file individually. Pooling is intentionally avoided to ensure that the resulting sequences remain independent of the overall dataset size. This design choice guarantees that previously processed data remains unaffected when new samples are added and the project is reanalyzed. During denoising, Apscale automatically assigns unique identifiers to each sequence using the SHA3-256 hashing algorithm.

Several threshold types are available to control which reads are considered for denoising. By default, an absolute threshold is applied (`minsize = 4`), meaning that only sequences with an abundance of four or more are retained—effectively removing a substantial amount of low-abundance noise. Alternatively, a relative threshold can be used to retain only those sequences that represent a defined percentage of the sample's total read count (e.g., 0.01%).

Since both absolute and relative thresholds are inherently arbitrary, we introduced a third option in version 4.0.0: power law–based filtering. Read abundance distributions typically follow a power law, where a few sequences are highly abundant (true biological signals) and many are rare (a mixture of real low-abundance taxa, sequencing noise, and PCR artifacts). This filtering method fits a power law model to each sample's read distribution and sets the threshold at the point where the observed distribution deviates from the expected power law curve. The underlying assumption is that this inflection marks a shift in the signal-to-noise ratio, with noise becoming dominant. This approach results in a data-driven, rather than arbitrary, threshold for denoising.

To run this module, use the following command:

```
apscale --denoising [project_path]
```

Setting	Default	Possible values	Effect
perform denoising	True	True / False	Enables or disables the denoising module
alpha	2	1 – infinity	alpha value for the denoising algorithm
threshold type	absolute	absolute / relative / power law	Filtering strategy to use to determine the abundance cutoff
size threshold [absolute nr / %]	4	1 – infinity (absolute) 0 – infinity (relative, as %)	This can be set as either an absolute number (e.g., 4) or a percentage when using relative filtering. Power law-based filtering does not require a predefined threshold.

3.6.2. Swarm clustering

Alternatively, files can be clustered individually using the Swarm algorithm with $d=1$ and the fastidious option enabled by default (see Swarm [GitHub](#) for details). We consider Swarm clustering as an alternative to denoising; when retaining only the chosen centroid sequences, the results are generally quite similar. Additionally, the output from the denoising module can be further clustered with Swarm if desired.

Setting	Default	Possible values	Effect
perform swarm clustering	False	True / False	Enables or disables the swarm clustering module

3.6.3. Replicate merging

This module merges replicates of the same sample, provided they follow a consistent naming pattern. It supports an unlimited number of replication levels (e.g., sample replicates, extraction replicates, PCR replicates). The replicate type should be indicated by a common delimiter in the file names and **must also be included in file names without replicates** to ensure reliable processing. See the example below.

To run this module, use the following command:

```
apscale --replicate_merging [project_path]
```

Setting	Default	Possible values	Effect
perform replicate merging	True	True / False	Enables or disables the replicate merging module
replicate delimiter	–	Any character	Delimiter that indicates the replicate
minimum replicate presence	2	1 to infinity	Minimum number of replicates in which a sequence must be present to be retained

Example:

- Extraction replicates:

In our lab, we usually extract DNA from each sample twice and only retain sequences that are found in both replicates. Samples are named with suffixes indicating the extraction replicate, such as Sample1_A and Sample1_B, Sample2_A and Sample2_B, and so on. If we want to merge these replicates, the settings would be configured accordingly to keep only sequences present in both replicates:

perform replicate merging	replicate delimiter	minimum replicate presence
True	_	2

- PCR replicates, extraction replicates, field replicates:

Consider a more complex scenario like eDNA sampling. Sometimes it takes two filters to process 1 liter of water, so some samples have one filter while others have two. This can be indicated in the sample name as sample1_filter1, sample1_filter2, etc. On top of that, extraction replicates are performed as described above, resulting in sample names like sample1_filter1_A and sample1_filter1_B. Additionally, there may be multiple PCR replicates for each extraction replicate, indicated by another delimiter. For example, with four PCR replicates per extraction replicate, sample names might look like this: sample1_filter1_A_1, sample1_filter1_A_2, sample1_filter1_A_3, sample1_filter1_A_4, sample1_filter1_B_1, sample1_filter1_B_2, sample1_filter1_B_3, sample1_filter1_B_4.

In the final analysis, depending on the research goal, you might only be interested in sequences from sample1, with all replicates properly merged. The filtering would be configured as follows:

perform replicate merging	replicate delimiter	minimum replicate presence
True	_	1
True	_	2
True	_	1

First, keep any sequence present in at least one of the PCR replicates. Second, keep only sequences found in both extraction replicates. Finally, keep sequences found in at least one of the filters used to process the sample volume.

3.6.4. Negative control filtering

This module automatically subtracts the maximum number of reads found in the negative controls from the corresponding sequences. Negative controls must be identifiable by a common prefix, which defaults to NC_. Negative control filtering is typically performed after replicate merging, since replicate merging helps remove background noise introduced in the lab. Reads remaining in the negative controls after merging are considered potential true contaminants.

To run this module, use the following command:

```
apscale --nc_removal [project_path]
```

Setting	Default	Possible values	Effect
perform nc removal	True	True / False	Enables or disables the nc removal module
negative control prefix	NC_	Any characters	Prefix that defines the negative controls

3.6.5. Read table generation and sequence grouping

This module generates the read table and performs threshold-based sequence grouping, similar to classical OTU clustering. Apscale always outputs both sequences (ESVs) and sequence groups (OTUs). The read table is saved in Parquet format and, if the dataset contains fewer than 1,000,000 distinct sequences, also in Excel format.

Additionally, this module creates a “read data store,” a DuckDB (<https://duckdb.org/>) database that contains comprehensive information about sequences, groups, samples, and read counts. The read data store efficiently handles even very large datasets—potentially billions of sequences—at high speed, without requiring the entire dataset to be loaded into memory. This makes it especially useful for scaling up analyses.

The read data store is extensively used in the Apscale analyze module, which is described in the following chapter.

To run this module, use the following command:

```
apscale --generate_read_table [project_path]
```

Setting	Default	Possible values	Effect
generate read table	True	True / False	Enables or disables the read table generation module
sequence group threshold	0.97	0 - 1	Defines the clustering threshold

3.7. Working with the read data store

The read data store enables users to perform a variety of useful operations on large datasets. Users can add metadata for samples and sequences, while the analyze module automatically infers corresponding metadata for sequence groups. Additionally, users can harmonize and validate taxonomy via GBIF, upload studies to ENA, and export read tables with custom filters.

3.7.1. Apscale analyze

The Apscale analyze module is implemented via streamlit (<https://streamlit.io/>) which generates an interface in the browser to directly interact with the read data store.

To run this module, use the following command:

```
apscale --analyze [project_path]
```

This will open the default browser and display the interface. An overview windows will give you basic information about the current project.


3.7.2. Adding sample metadata

This module allows users to add sample metadata to the read data store. The input must be a table containing sample identifiers in one column and any number of additional columns with metadata (e.g. geographical location, habitat, sampler, sampling date, etc.). Metadata can be provided in Excel, Parquet, or CSV format and added via drag and drop.

Add sample metadata

Metadata can be added by uploading a table that holds all metadata information. [?](#)

Metadata table upload:

 Drag and drop file here
Limit 200MB per file • PARQUET, SNAPPY, CSV, XLSX

Browse files

Example sample metadata:

sample_id	lat	lon	type	sampling_date	sampler
Rh_103_04	47.6147	8.223	river	17.04.2025	lisa
Rh_126_05	47.5554	7.9923	river	18.04.2025	lisa
Rh_168_06	47.5866	7.5866	forest	19.04.2025	lisa
Rh_195_08	47.76382	7.533869	forest	20.04.2025	lisa
Rh_226_09	48.06484	7.573503	forest	21.04.2025	dominik
Rh_258_10	48.31677	7.730412	forest	22.04.2025	dominik
Rh_28_01	47.6754	8.8285	river	23.04.2025	dominik
Rh_63_02	47.60489	8.596649	river	24.04.2025	dominik
Rh_98_03	47.6087	8.2679	river	25.04.2025	dominik

Once the metadata has been uploaded, the module will automatically try to infer the columns and corresponding datatypes:

Select and modify metadata

Select the sample identifier column:

sample_id

✓ The provided identifier matches the samples in the read store!

Please select the correct datatype for each column.

sample_id	✓ Include	Data type string ×
lat	✓ Include	Data type floating point... ×
lon	✓ Include	Data type floating point... ×
type	✓ Include	Data type string ×
sampling_date	✓ Include	Data type timestamp ×
sampler	✓ Include	Data type string ×

Currently Apscale supports strings, integers, floating point number, timestamps and Boolean values. Once the sample metadata is set up accordingly, the button “Save to read data store” will add the data to the database. Once this is done, the module will display a preview of the metadata in the read data store.

Add sample metadata

Sample metadata is already present in the read data store.

Rows to display in preview

5

	sample_idx	sample	lat	lon	type	sampling_date	sampler
0	7	Rh_103_04	47.6147	8.223	river	2025-04-17 00:00:00	lisa
1	1	Rh_126_05	47.5554	7.9923	river	2025-04-18 00:00:00	lisa
2	8	Rh_168_06	47.5866	7.5866	forest	2025-04-19 00:00:00	lisa
3	4	Rh_195_08	47.7638	7.5339	forest	2025-04-20 00:00:00	lisa
4	6	Rh_226_09	48.0648	7.5735	forest	2025-04-21 00:00:00	dominik

Reset sample metadata

3.7.3. Adding sequence metadata

This module allows users to add sequence metadata to the read data store. The input must be a table containing sample identifiers in one column and any number of additional columns with metadata (e.g. geographical location, habitat, sampler, sampling date, etc.). Metadata can be provided in Excel, Parquet, or CSV format and added via drag and drop. The logic is the same as with the sample metadata. Upon saving the metadata will be added to the read data store.

3.7.4. Correcting species names via GBIF

This module can be used to harmonize species names with the GBIF taxonomic backbone, ensuring consistency and comparability across different studies. To run this module, the sequence metadata must include a column containing species names. The column can be selected via the dropdown menu and a preview will display all distinct species name found in this column.

Query the GBIF backbone to harmonize species names

Please select the column that holds the species names.

Species column name

species

There are **120 distinct values** in the selected column.

File preview

	species
0	<i>Acanthocyclops americanus</i>
1	<i>Achlya colorata</i>
2	<i>Achlya oligocantha</i>
3	<i>Amphichaeta leydigi</i>
4	<i>Amphichaeta raptisae</i>
5	<i>Ancylus fluviatilis</i>
6	<i>Aphanomyces laevis</i>
7	<i>Apis mellifera</i>
8	<i>Audouinella hermannii</i>
9	<i>Batrachospermum gelatinosum</i>

Query GBIF API

Once the button is clicked, Apscale will query the GBIF API and add a column named “gbif_taxonomy” to the sequence metadata..

Query the GBIF backbone to harmonize species names

GBIF taxonomy has already been added.

Rows to display in preview

- +

		flags	gbif_taxonomy
0	FX6893 BOLD:ACR1936	5	Orthocladus rivicola
1	AB5106	2	Diamesa tonsa
2	CB9778 BOLD:AEG3086 BOLD:AEG4629 BOLD:ACH7774 BOLD:AED8843 BOLD:	5	Polyarthra dolichoptera
3			Dinobryon bavaricum
4	FX6893 BOLD:ACR1936	5	Orthocladus rivicola
5	EQ8533	3	Cricotopus tremulus
6	CR1936	3 4	Orthocladus rivicola
7	DV8590 BOLD:ABW5947	2 5	Orthocladus thienemanni
8	EK3719 BOLD:ADV8590 BOLD:AAB3988	2 5	Orthocladus rivulorum
9	AE2233 BOLD:ACL2016 BOLD:ABA0395 BOLD:AAE2234 BOLD:ADL9379 BOLD:/	5	Keratella cochlearis

Reset GBIF taxonomy

3.7.5. Validating species occurrence via GBIF

This module validates taxonomy using GBIF occurrence data. It requires both GBIF taxonomy information and geographic coordinates (latitude and longitude) for each sample in the dataset. Apscale calculates a user-defined radius (default: 200 km) around the known occurrences of each sequence, which is then used to assess whether the taxonomic assignment is geographically plausible.

Validate species names via GBIF record search

This module needs sample metadata (lat, lon) for each sample.

This module needs harmonized GBIF taxonomy.

Latitude



Longitude



lat



lon



File preview

	sample_idx	sample	lat	lon
0	7	Rh_103_04	47.6147	8.223
1	1	Rh_126_05	47.5554	7.9923
2	8	Rh_168_06	47.5866	7.5866
3	4	Rh_195_08	47.7638	7.5339
4	6	Rh_226_09	48.0648	7.5735
5	5	Rh_258_10	48.3168	7.7304
6	9	Rh_28_01	47.6754	8.8285
7	2	Rh_63_02	47.6049	8.5966
8	3	Rh_98_03	47.6087	8.2679

Radius to check around sample

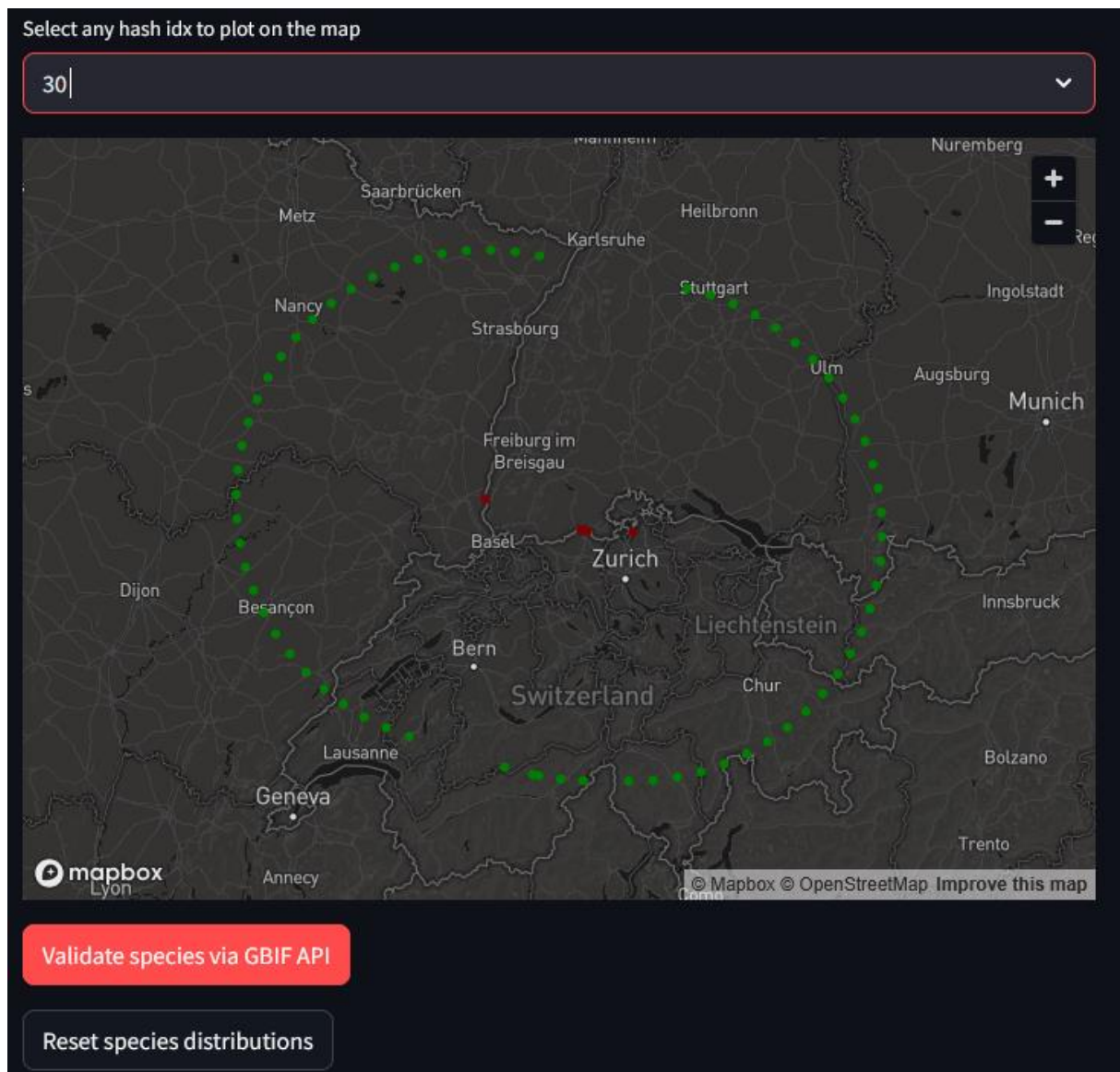
200

50

500

Compute species distributions

Once the distributions are computed, species occurrences across all samples will be displayed on the map as red points, while the computed distribution ranges will be shown as green points.



Once the button is clicked, Apscale will validate all species names by checking whether any GBIF records for each species fall within the computed distribution range. The result of the validation procedure will be saved to the sequence metadata as either plausible or implausible in a column called “gbif_validation”. After computation a preview will show the results.

Rows to display in preview

15

	sequence_idx	gbif_taxonomy	gbif_validation
0	1452	Orthocladus rivicola	plausible
1	1	Diamesa tonsa	implausible
2	1629	Polyarthra dolichoptera	plausible
3	2462	Dinobryon bavaricum	implausible
4	461	Orthocladus rivicola	plausible
5	2480	Cricotopus tremulus	implausible
6	296	Orthocladus rivicola	plausible
7	1635	Orthocladus thienemanni	implausible
8	2134	Orthocladus rivulorum	plausible
9	2265	Keratella cochlearis	plausible

Reset GBIF validation

3.7.6. Uploading datasets to ENA

This module will be added in version 4.2.

3.7.7. Exporting read tables

The final module allows users to export and filter read tables. An unlimited number of filters can be applied to both sequence metadata and sample metadata. Additionally, output tables can be split based on sample metadata fields, enabling parallelized downstream analysis.

The results are exported to the project folder under 12_analyze/data in both Parquet and CSV formats. Users can also select which columns to include in the output independently of any applied filters. Outputs will contain a table for sequences as well as for sequence groups.

Example:

Select filters for sequence metadata

Apply filter to:

phylum x

Keep sequences where phylum matches:

Annelida x Arthropoda x

Select filters for sample metadata

Apply filter to:

sampler x

Keep samples where sampler matches:

dominik x

Split output tables by sample metadata

Split the output table on these columns:

type x

Export read tables

Select columns to include in exported table

sequence_idx x hash x sequence x sequence_order x read_sum x

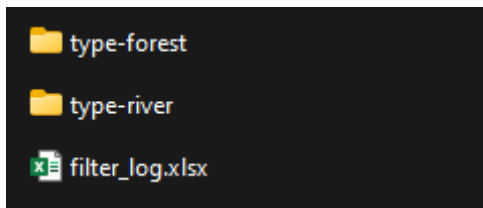
phylum x class x order x family x genus x species x pct_identity x

status x records x records_ratio x selected_level x BIN x flags x

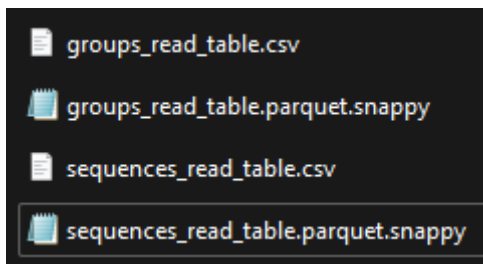
gbif_taxonomy x gbif_validation x

Export read tables

In this example, the output tables will only contain samples where the Phylum matches Annelida or Arthropoda and the sampler is Dominik. Read tables will be split by the type column and include the selected sequence metadata columns.



The output folder will contain two files (.parquet, .csv) for sequences as well as the sequence groups.



Apscale will create a log with the selected filters and one folder per “type” as the data is split by forest and river.

sequence_id	hash	phylum	class	order	family	genus	species	ct_identif	taxonomif	if_validation	Rh_226_09	Rh_258_10
454	50a635904	Arthropoc	Insecta	Diptera	Chironom	Orthocladius		99.5122			1249	161
1283	1a82b984f	Arthropoc	Insecta	Diptera	Chironom	Orthocladius		100			1628	492
1452	bb5aa9030	Arthropoc	Insecta	Diptera	Chironom	Orthoclad	Orthoclad	99.5122	Orthoclad	plausible	0	0
2458	8d6ead60c	Arthropoc	Collembola	Entomobryomorpha				86.17886			72959	128633
1	c1c083424	Arthropoc	Insecta	Diptera	Chironom	Diamesa	Diamesa t	100	Diamesa t	implausib	0	0
1284	c35c9b2c8	Arthropoc	Insecta	Diptera	Chironom	Orthocladius		100			0	412
1113	936f991b5	Arthropoc	Insecta	Diptera	Chironom	Orthocladius		100			0	11
1791	de7d19aa1	Arthropoc	Insecta	Diptera	Chironom	Orthocladius		100			0	0
606	4bffe3752	Arthropoc	Insecta	Neuroptera				88.34951			16465	23924
1286	dafa68528	Arthropoc	Arachnida	Araneae				86			43509	55234
461	8b41620b2	Arthropoc	Insecta	Diptera	Chironom	Orthoclad	Orthoclad	99.02439	Orthoclad	plausible	0	0
54	635cf1d5b	Arthropoc	Insecta	Hymenoptera				86.91589			128938	61711
138	e443663d5	Arthropoc	Insecta					81.28655			12592	11609
3	4784baa61	Arthropoc	Insecta					81.59204			1950	91
933	d8d690642	Annelida	Clitellata	Tubificida	Naididae	Chaetogaster		100			1676	262
1455	5f917f125	Annelida	Clitellata	Tubificida				87.27273			2376	1263
5	664b6f1e1	Arthropoc	Insecta					83.59375			0	0
1112	b929c02c1	Arthropoc	Insecta	Lepidoptera				85.14851			1445	131
637	9f87306ad	Arthropoc	Insecta	Hemiptera				87.12871			45533	84575
139	1c351b9f7	Arthropoc	Arachnida	Sarcoptiform	Ceratozetidae			90.09901			7415	912
2480	2d62a90cc	Arthropoc	Insecta	Diptera	Chironom	Cricotopus	Cricotopus	100	Cricotopus	implausib	0	0
296	d6516d55f	Arthropoc	Insecta	Diptera	Chironom	Orthoclad	Orthoclad	100	Orthoclad	plausible	0	0
1800	c440e1ea6	Arthropoc	Insecta					81.33333			142	374
1942	c2356bfa9	Arthropoc	Insecta					81.28655			8598	6579
1635	7e39753d2	Arthropoc	Insecta	Diptera	Chironom	Orthoclad	Orthoclad	99.5122	Orthoclad	implausib	0	0
771	4889b86b2	Arthropoc	Insecta					81.28655			7644	2633