Apscale manual

Last updated: 08.08.2025, apscale version 4.1.0

Author: Dominik Buchner

Table of contents

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

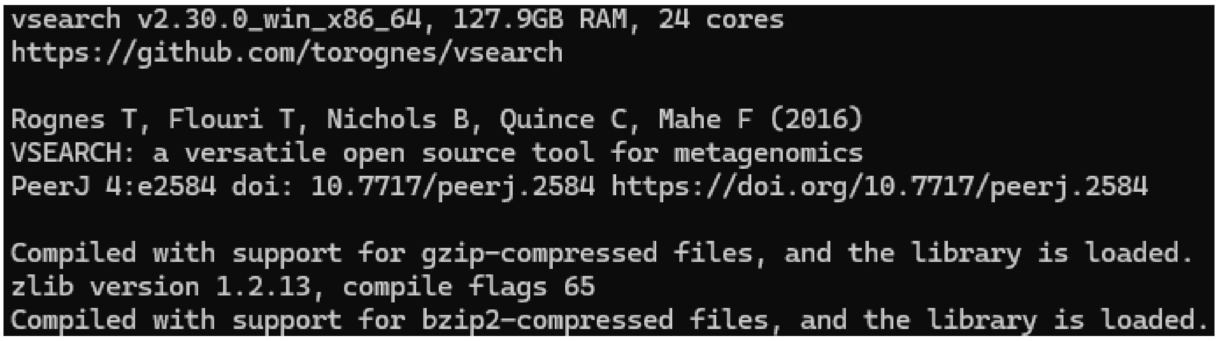
https://github.com/torognes/swarm

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

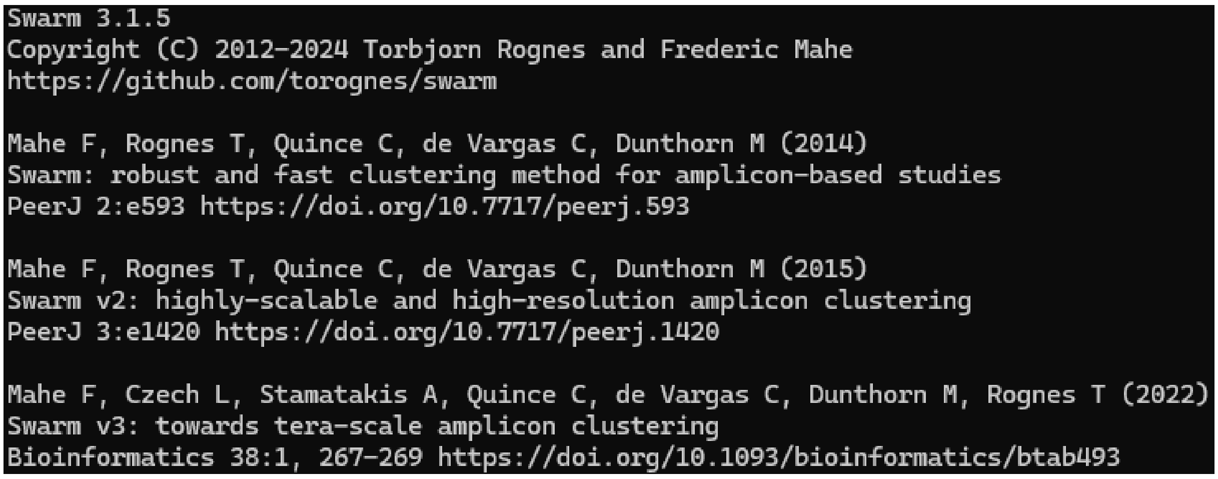
```
vsearch --version
```

```
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, Flouri 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.
```



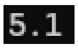
```
Swarm 3.1.5
Copyright (C) 2012-2024 Torbjorn Rognes and Frederic Mahe
https://github.com/torognes/swarm

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.

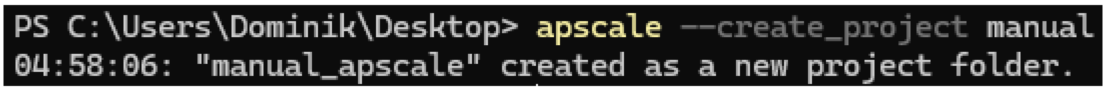
3.2. Creating a project

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.

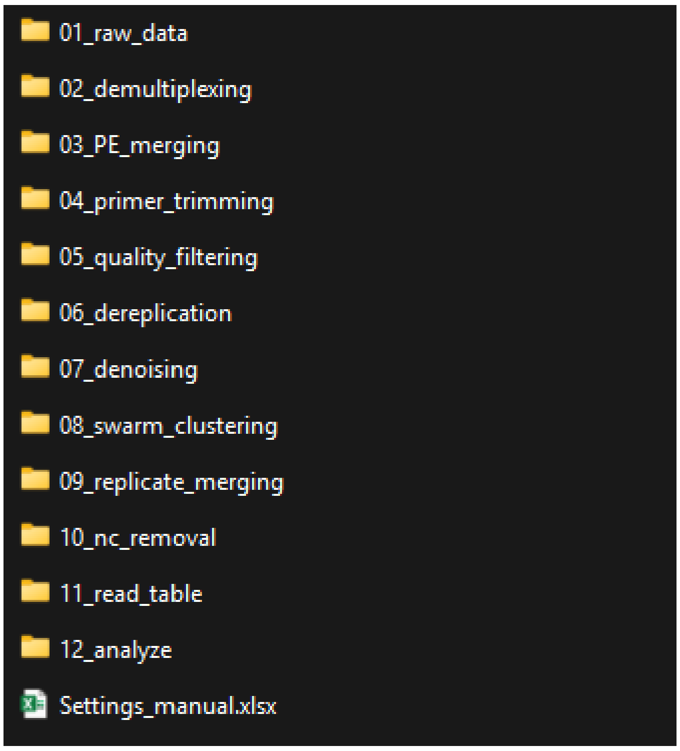
To create a new project, run:

```
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.
```



```
01_raw_data
02_demultiplexing
03_PE_merging
04_primer_trimming
05_quality_filtering
06_dereplication
07_denoising
08_swarm_clustering
09_replicate_merging
10_nc_removal
11_read_table
12_analyze
Settings_manual.xlsx
```

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]
```

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 reagion
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)