apscale_blast2 — User Manual

Local BLAST-based taxonomic assignment for metabarcoding
(v1.1.2)

Álvaro Fueyo and Till Macher

Contents

1 Introduction

apscale_blast2 is a practical tool for **local, reproducible taxonomic assignment** using **NCBI BLAST+** and curated reference databases. It was inspired by apscale_blast, and it is optimised for **local BLAST-based assignment** with an emphasis on **reproducibility** and **transparent handling of**

ambiguity.

1.1 What apscale_blast2 is *not*

It helps to be explicit about scope:

- It is **not** a full read-processing pipeline (no demultiplexing, trimming, denoising).
- It is **not** a machine-learning classifier (no naïve Bayes, no model training).
- It will not solve an incomplete or biased reference database, but it will make the resulting uncertainty visible and reproducible.

2 Requirements

apscale_blast2 is a Python tool that relies on external BLAST executables.

2.1 Python

- **Python >= 3.10**
- Download: https://www.python.org/downloads/

2.2 BLAST+ (hard requirement)

- **NCBI BLAST+ >= 2.17.0** must be installed and available on your `PATH`.
- apscale_blast2 checks this at startup.

Quick sanity check:

```
blastn -version
makeblastdb -version
```

If these commands do not work in your terminal, fix BLAST+ installation before continuing.

2.3 Installing BLAST+ and adding it to your PATH

apscale_blast2 requires **NCBI BLAST+** (at least `blastn` and `makeblastdb`). Installation differs slightly by operating system:

2.3.1 Windows

1. Download BLAST+ from NCBI (installer or ZIP).
2. Locate the `bin` directory (often `C:\Program Files\NCBI\blast-<version>\bin`).
3. Add the `bin` directory to your **PATH**:
   - Start menu → search **Environment Variables** → *Edit the system environment variables*
   - **Environment Variables...** → select **Path** → **Edit** → **New**

- Paste the `...\bin` path and confirm.

4. Open a *new* terminal and test:

```
blastn -version
makeblastdb -version
```

2.3.2 macOS

Two common options:

- **Homebrew** (recommended if you use brew):

```
brew install blast
```

- **NCBI download**: install BLAST+ and ensure the `bin` directory is on your PATH.

Test:

```
blastn -version
makeblastdb -version
```

2.3.3 Linux

Common options:

- **Conda / mamba**:

```
conda install -c bioconda blast
```

- **System packages** (availability varies by distro), or the NCBI download.

Test:

```
blastn -version
makeblastdb -version
```

If the test commands work in your terminal, apscale_blast2 should detect BLAST+ automatically.

3 Installation

3.1 Install directly from GitHub

For most users, the easiest installation is:

```
pip install "git+https://github.com/AlvaroFueyo/apscale_blast2.git"
```

3.2 Editable install (recommended for development)

If you plan to modify the code or test local changes:

```
git clone https://github.com/AlvaroFueyo/apscale_blast2.git
cd apscale_blast2

python -m venv venv
# Linux/macOS:
source venv/bin/activate
# Windows PowerShell:
# venv\Scripts\Activate.ps1

pip install -U pip
pip install -e .
```

4 Your first run (wizard mode)

If you run `apscale_blast2` **without** non-interactive database arguments, it starts in interactive **wizard** mode:

```
apscale_blast2
```

The wizard is intentionally straightforward:

1. Select a folder containing FASTA files (`.fa`, `.fasta`, `.fna`, non-recursive).
2. Choose the BLAST search mode (`megablast` or `blastn`).
3. For each FASTA, select a database (or build/install one).
4. Review outputs in `raw_blast/` and `taxonomy/`.

4.1 Pre-select the FASTA folder

You can skip the first prompt by providing the FASTA directory:

```
apscale_blast2 --fastas path/to/fasta_dir
```

4.2 Choosing BLAST mode: `megablast` vs `blastn`

BLAST offers different "tasks" that trade off speed and sensitivity. In apscale_blast2 you typically choose between `megablast` and `blastn`.

4.2.1 `megablast` (fast, for very similar sequences)

`megablast` is tuned for queries that are expected to be **highly similar** to the reference database (typical DNA barcodes and amplicons within the same broad taxonomic scope). It is faster mainly because it uses a **larger word size** by default (commonly **28**), which reduces the number of seed matches BLAST needs to evaluate.

In practice, use `megablast` when: - you expect close matches (same marker, comparable length), - you want higher throughput, - and you mainly care about the best, near-identical hits.

4.2.2 `blastn` (more sensitive, for more divergent sequences)

`blastn` is a more general nucleotide search. It is typically **more sensitive** because it uses a **smaller word size** by default (commonly **11**), which increases the chances of seeding alignments even when there are mismatches or when queries are shorter/poorer quality. The trade-off is speed.

In practice, use `blastn` when: - you expect more divergence from references, - queries may be shorter/fragmented, - or you suspect `megablast` is missing reasonable candidates.

4.2.3 A practical rule of thumb

- Start with **`megablast`** for standard barcode/amplicon runs.
- Switch to **`blastn`** if you see too many apparent no-hits or if you expect higher divergence.

In addition, if you run with the legacy flag scheme (`--flag-scheme apscale`), apscale_blast2 forces `blastn` to replicate historical APSCALE-like defaults.

5 Non-interactive CLI usage

Wizard mode is great for exploration; scripting mode is for reproducible pipelines.

5.1 Use one database for all FASTA files

```
apscale_blast2 --fastas /path/to/fastas --db-for-all /path/to/db_folder --threads 8
```

5.2 Use a CSV mapping FASTA basenames to databases

Create a CSV with columns `fasta,db` (FASTA column must match the file basename, e.g. `sampleA.fasta`):

```
fasta,db
sampleA.fasta,/path/to/db_A
sampleB.fasta,/path/to/db_B
```

Run:

```
apscale_blast2 --fastas /path/to/fastas --db-map mapping.csv
```

5.3 Useful database utilities

Print where apscale_blast2 stores per-user databases:

```
apscale_blast2 --print-db-home
```

Open that folder in your file explorer:

```
apscale_blast2 --open-db-home
```

6 How databases are handled

BLAST databases are easy to break accidentally (missing index files, wrong prefix, unclear taxonomy mapping). apscale_blast2 aims to make database selection and provenance explicit.

6.1 Database home (DB home)

By default, apscale_blast2 maintains a **per-user database store** ("DB home") where it installs and auto-discovers databases.

You can override the DB home:

```
apscale_blast2 --db-home /custom/db_home
```

You can also add an extra directory to scan for databases (in addition to DB home):

```
apscale_blast2 --dbs /extra/db_directory
```

6.2 What is `db_prefix`?

When BLAST runs, it does not point to "a FASTA file". It points to a **database prefix**: a path without extensions that BLAST uses to locate its index files (`.nal` or `.nin/.nsq/.nhr`, etc.).

apscale_blast2 records both:

- `db_path`: the database folder you selected
- `db_prefix`: the exact prefix passed to BLAST with `-db <prefix>`

You will find both in the run sidecar file (`taxonomy/<sample>.runinfo.txt`), which is meant to make runs reproducible even when database folders contain multiple candidate indices.

6.3 Building and installing databases (wizard)

When you select **Build and install a new database (recipe)**, the wizard offers several builders, including:

- **MIDORI2**
- **UNITE**
- **SILVA**
- **PR2**
- **trnL**
- **DiatBarcode**
- **Precompiled BLAST bundle** (`.zip` containing indices + taxonomy mapping)

Each builder ultimately creates (inside DB home) a database folder such as:

```
<db_home>/db_<name>/
  db/...
  db_taxonomy.parquet.snappy
  apscale_blast2_defaults.json  (optional; created when you edit defaults in the wizard)
  ...
```

The important pieces are:

- **BLAST indices** (created by `makeblastdb`)
- **a taxonomy mapping table** (used to map BLAST accessions/IDs to ranks)

6.4 Precompiled databases

If you have a precompiled database bundle (`.zip`) that already contains indices + taxonomy mapping, the wizard can install it directly, avoiding any local formatting work.

6.5 Per-database defaults: `apscale_blast2_defaults.json`

apscale_blast2 supports per-database "preferred settings" to reduce repetition in the wizard. If present, the wizard loads:

- identity/coverage thresholds
- BLAST task
- max target sequences
- (and other parameters shown in the wizard prompts)

The file is stored inside the database folder as:

`<db_folder>/apscale_blast2_defaults.json`

If it does not exist, global defaults are used. If you edit values in the wizard, the file is created automatically.

7 What happens during a run (pipeline overview)

Under the hood, apscale_blast2 follows a simple pattern:

1. **Split** each FASTA into subsets (`--subset-size`) to keep BLAST calls manageable.
2. **Run BLAST** on each subset (optionally with multiple workers).
3. **Merge** subset results into a single hit table.
4. **Map taxonomy** using the database taxmap.
5. **Apply filtering and assignment logic** (including ambiguity handling).
6. **Write outputs** (raw hits + taxonomy table + run metadata sidecar).
7. **Clean up** temporary subset files unless `--keep-tsv` is enabled.

The splitting step makes runs more robust on large FASTAs and enables controlled parallelism.

7.1 Concurrency: `--threads` vs `--workers`

Two knobs affect performance:

- `--threads`: BLAST threads per BLAST call. `0` means "auto", implemented as `max(1, cpu-2)`.
- `--workers`: how many subset BLAST jobs run concurrently.

A safe starting point is to keep `--workers 1` and tune `--threads`. Multiple workers can oversubscribe CPUs quickly (each worker may spawn a multi-thread BLAST process).

8 Filtering, trimming, and assignment logic

This is the part that most directly affects your ecological interpretation, so it is worth understanding.

8.1 Early (BLAST-side) filters

apscale_blast2 applies filters at BLAST execution time to avoid generating hits that will be discarded anyway:

- **maximum E-value** (`--max-evalue`) is passed to BLAST via `-evalue`.
- **minimum query coverage** (`--min-qcov`) is passed via `-qcov_hsp_perc` (HSP coverage).
- **minimum percent identity** (`--min-pident`) is, by default, passed via `-perc_identity` (see below).

8.1.1 Inline identity filter (default) vs post-filter only

By default, apscale_blast2 passes `--min-pident` into BLAST using `-perc_identity`. This is controlled by:

- `--inline-perc-identity` (default behaviour)
- `--no-inline-perc-identity` (do not pass `-perc_identity`; apply identity filtering only after BLAST)

Most users can keep the default. The "no inline" mode is mainly for debugging or niche BLAST behaviours.

8.2 Soft query coverage preference: `--prefer-qcov`

Hard filters remove hits. Sometimes you want something gentler: prefer high-coverage hits when available, but avoid losing assignments when nothing meets the preference.

apscale_blast2 implements this as:

- if at least one hit for a query has `query_coverage >= prefer_qcov`, keep only those hits;
- otherwise, keep all hits for that query (already filtered by hard filters).

You can disable the preference entirely by setting it to `0`:

```
apscale_blast2 --prefer-qcov 0
```

8.3 Similarity thresholds by rank: `--thresholds`

Many metabarcoding projects apply similarity thresholds to decide how deep a taxonomic label should go. apscale_blast2 uses:

`--thresholds "97,95,90,87,85"`

interpreted as thresholds for:

- Species, Genus, Family, Order, Class

(APSCALE defaults.)

If a hit does not meet the threshold for a given rank, ranks below that level are trimmed away. This avoids reporting unsupported deep assignments.

8.4 Hit selection order (mode 1)

Before thresholds and flags are applied, apscale_blast2 reduces the candidate set based on a consistent selection rule.

By default it uses **mode 1**:

- highest similarity first,
- then lowest E-value as a tie-breaker.

This matches the "similarity-first" approach used in classic APSCALE workflows.

9 Ambiguity flags and assignment schemes

apscale_blast2 supports **two** assignment/flagging schemes:

- `--flag-scheme apscale2` (default)
- `--flag-scheme apscale` (legacy APSCALE-like behaviour)

9.1 `--flag-scheme apscale2` (default): MRCA-based trimming

The default scheme is designed for curated local databases (often de-duplicated, with consistent taxonomy), where "dominant taxon" heuristics are not necessarily informative.

After filtering and similarity trimming:

1. the tool deduplicates by taxonomy profile,
2. checks how many unique taxa remain,
3. if more than one remains, it trims the reported taxonomy to the **MRCA** (Most Recent Common Ancestor),
4. stores the surviving candidates in **`Ambiguous taxa`**,
5. labels the situation with a flag:

- `Fl1` indicates two or more **species** remain (trimming to MRCA, typically genus),
- `Fl2` indicates two or more **genera** remain (trimming higher),
- and so on for higher ranks.

In other words, the number increases as ambiguity climbs up the taxonomy.

9.2 `--flag-scheme apscale` (legacy): classic APSCALE decision tree (F1–F4)

This mode aims to replicate the behaviour described for APSCALE / APSCALE-GUI, including dominant-species logic. When enabled, apscale_blast2 also restores several legacy defaults:

- forces `task = blastn`
- disables both query-coverage filters (`--min-qcov` and `--prefer-qcov` effectively become `0`)
- sets `--max-target-seqs` to `20`

Use legacy mode when you need continuity with older APSCALE-derived outputs.

10 Outputs

apscale_blast2 writes outputs **outside** your FASTA folder: it creates sibling folders in the **parent directory** of the FASTA directory.

Example:

```
project/
  fastas/                 <-- you point --fastas here
    sampleA.fasta
    sampleB.fasta
  raw_blast/              <-- created by apscale_blast2
  taxonomy/               <-- created by apscale_blast2
```

10.1 Raw BLAST hits table (`raw_blast/`)

For each FASTA `<base>.fasta`, apscale_blast2 writes:

- `raw_blast/<base>_raw_blast.xlsx` (default)

- or `raw_blast/<base>_raw_blast.parquet.snappy` (with `--output-format parquet`)

This table includes BLAST hit metrics plus mapped taxonomy and a few QC fields (e.g. mismatches and gap opens).

10.2 Taxonomy table (`taxonomy/`)

For each FASTA, apscale_blast2 writes:

- `taxonomy/<base>_taxonomy.xlsx` (default)
- or `taxonomy/<base>_taxonomy.parquet.snappy`

Columns include:

- `unique ID`
- taxonomy ranks (`Kingdom` to `Species`)
- `Similarity`, `query_coverage`, `evalue`
- `Flag`
- `Ambiguous taxa`

If a query has no hits after filtering, ranks are filled with `NoMatch` and metrics are set to safe defaults.

10.3 Reproducibility sidecar (`.runinfo.txt`)

Alongside the taxonomy output, apscale_blast2 writes:

- `taxonomy/<base>.runinfo.txt`

It records the effective configuration used in that run (task, thresholds, filters, flag scheme, database path/prefix, etc.). This keeps result tables clean while still making runs reproducible.

10.4 Excel vs Parquet

Excel is convenient for manual review. Parquet is ideal for large projects and programmatic analysis:

- smaller on disk
- faster to read/write
- robust data types

If you choose Parquet output, you can load it with Python:

```
import pandas as pd

df = pd.read_parquet("taxonomy/sampleA_taxonomy.parquet.snappy")
```

11 Practical examples

11.1 A conservative, fast run for typical barcodes

```
apscale_blast2 \
  --fastas ./fastas \
  --db-for-all /path/to/my_db \
  --task megablast \
  --threads 0 \
  --max-target-seqs 30
```

11.2 A run that writes Parquet for large projects

```
apscale_blast2 \
  --fastas ./fastas \
  --db-for-all /path/to/my_db \
  --output-format parquet
```

11.3 Prefer coverage, but do not hard-filter coverage

Sometimes you want to avoid discarding low-coverage hits, but still prefer high-coverage ones if present:

```
apscale_blast2 \
  --fastas ./fastas \
  --db-for-all /path/to/my_db \
  --min-qcov 0 \
  --prefer-qcov 75
```

12 Troubleshooting

12.1 "BLAST+ not found" or version errors

Symptoms: - The tool exits early complaining about BLAST executables or BLAST version.

Fix: - Ensure `blastn` and `makeblastdb` are installed and on your `PATH`. - Confirm `blastn -version` reports >= **2.17.0**.

12.2 "No databases installed yet ..."

This is informational, not an error. It means your DB home is empty.

Use the wizard to build/install a database, or point `--db-for-all` / `--db-map` to an existing database folder.

12.3 "Invalid database" / missing index files

BLAST databases require index files. If a database folder is incomplete, validate/build it again.

Common causes: - copying only the FASTA but not the BLAST indices, - interrupted `makeblastdb` run, - wrong prefix/path.

12.4 Excel is slow or files are huge

Switch to Parquet output:

```
apscale_blast2 --output-format parquet
```

12.5 Temporary files are not deleted

By default, temporary subset files are cleaned up. If you used:

- `--keep-tsv`

then temporary intermediate files are preserved for debugging.

13 Good practice for reproducibility

If you plan to publish or to re-run projects later:

- keep the `taxonomy/*.runinfo.txt` files together with results,
- record which database bundle/release you used (or keep a copy),
- consider versioning your DB folders (e.g. include a date in the DB name),
- cite the reference database sources when reporting results.

14 References

The repository README includes recommended citations for BLAST+ and common reference databases used by the wizard builders. For convenience, DOI links are included below (and release-specific DOIs should be preferred when a database provides them).

- Camacho et al. (2009) BLAST+: Architecture and applications. *BMC Bioinformatics* 10:421. https://doi.org/10.1186/1471-2105-10-421
- Leray et al. (2022) MIDORI2: a collection of quality-controlled sequences for taxonomic assignment. https://doi.org/10.1111/1755-0998.13644
- Nilsson et al. (2019) The UNITE database for molecular identification of fungi. https://doi.org/10.1093/nar/gky1022
- Quast et al. (2013) The SILVA ribosomal RNA gene database project. https://doi.org/10.1093/nar/gks1219
- Guillou et al. (2013) The Protist Ribosomal Reference database (PR2). https://doi.org/10.1093/nar/gks1160

- Rimet et al. (2019) The Diat.barcode reference library for diatoms. https://doi.org/10.1111/1755-0998.13018

For the most up-to-date reference list, see the repository: https://github.com/AlvaroFueyo/apscale_blast2