LAPIS

Chaoran Chen

Oct 21, 2022

TUTORIALS

LAPIS (Lightweight API for Sequences) is an open web application programming interface (API) allowing easy querying of SARS-CoV-2 sequencing data using web links. The core features are:

- Filter sequences by metadata or mutations
- Aggregate data by any metadata field you like
- Get the full metadata
- Get the sequences as FASTA (aligned or unaligned)
- Responses can be formatted as JSON and as CSV

Two instances are publicly available:

- For monkeypox (data from GenBank): https://mpox-lapis.genspectrum.org (*reference*)
- For SARS-CoV-2 (data from GenBank): https://lapis.cov-spectrum.org/open (*reference*)

If you are hosting an own public instance and would like it to be added to the list, please send us an email or submit a pull request to change this page.

CHAPTER

ONE

TUTORIALS FOR DEVELOPERS AND MAINTAINERS

1.1 Setup LAPIS for monkeypox

In this tutorial, you will setup a LAPIS instance for monkeypox. You will learn:

- how import data into LAPIS' database
- how to start the API server

Note: This setup is only for testing/demonstration purposes and not recommended for productive use.

1.1.1 Requirements

- A working installation of Docker and basic familiarity with using Docker

1.1.2 Data download and alignment

To get started, LAPIS needs some metadata, the unaligned sequences and, if available, aligned sequences. For monkeypox, example data is available in Nextstrain's monkeypox repository. Please download the `metadata.tsv` and `sequences.fasta`. They are already correctly formatted.

Next, the sequences should be aligned. It is possible to start LAPIS without an alignment (in that case, create an empty `aligned.fasta`) but in order to filter and aggregate by mutations, an alignment is needed. You can use Nextclade version 2 to obtain an alignment. Open the website, select "Monkeypox" as the pathogen, select the sequences.fasta file that you downloaded and click an the "Run" button. It can take a few moments until the sequences are analyzed. Once it has finished, you can click on the download symbol on the top right and download the `nextclade.aligned.fasta` file. Rename the file to `aligned.fasta`.

Place the files metadata.tsv, sequences.fata and aligned.fasta into the same directory.

```
/path/to/data
    |-- metadata.tsv
    |-- sequences.fasta
    |-- aligned.fasta
```

1.1.3 Initialize database

LAPIS uses a PostgreSQL database (version 14+). An easy option to set up a database is to use the postges Docker image. First, download the SQL scripts in from here and place them in the same directory. Open `01_users.sql` and set the passwords for the users.

```
/path/to/db-scripts
    |-- 01_users.sql
    |-- 02_init.sql
    |-- 03_transform_and_merge.sql
```

Execute the following command to start the database. Change the password for the `postgres` user in the command.

```
docker network create -d bridge lapis_network

docker run -d \
    --name lapis_db \
    --net lapis_network \
    -e POSTGRES_PASSWORD=<missing> \
    -v /path/to/db-scripts:/docker-entrypoint-initdb.d \
    postgres:14
```

1.1.4 Data import

First, create a file `lapis-proc-config.yml` with the following content (please fill in the password):

```
default:
  vineyard:
    host: lapis_db
    port: 5432
    dbname: postgres
    username: lapis_proc
    password: <missing>
    schema: public
  workdir: /data
  maxNumberWorkers: 20
```

Then, execute the following command:

```
docker run --rm \
    --name lapis_proc \
    --net lapis_network \
    --entrypoint java \
    -v /path/to/lapis-proc-config.yml:/app/lapis-config.yml \
    -v /path/to/data:/data \
    ghcr.io/cevo-public/lapis-server:br-mpox \
        -jar /app/lapis.jar \
        --config /app/lapis-config.yml \
        Lapis --update-data load-mpox,transform-mpox,switch-in-staging
```

1.1.5 API server

To start the API server, create a file `lapis-api-config.yml` with the following content (please fill in the password):

```
default:
  vineyard:
    host: lapis_db
    port: 5432
    dbname: postgres
    username: lapis_api
    password: <missing>
    schema: public
  cacheEnabled: false
  redisHost:
  redisPort:
  apiOpennessLevel: OPEN
```

Then, execute the following command:

```
docker run --rm \
    --name lapis_api \
    --net lapis_network \
    -v /path/to/lapis-api-config.yml:/app/lapis-config.yml \
    -p 127.0.0.1:2345:2345 \
    ghcr.io/cevo-public/lapis-server:br-mpox
```

Wait half a minute and then open http://localhost:2345/v1/sample/aggregated in your browser.

CHAPTER

TWO

DATA VERSIONS

Distinguishing data versions is an important technique in LAPIS to ensure consistency and correctness of down-stream analyses. It is not for versioning: apart from some cached values, LAPIS only maintains the most recent data and no older versions.

Problem:

Knowing the data version is relevant when you need the results from more than one request and need to ensure that the results are generated from the exact same underlying dataset.

Here is an example of a potential problem: Imagine you would like to obtain the proportion of sequences from Oceania in the whole dataset. To calculate it, you could make two API requests: the first to get the number of sequences from Oceania and the second for the total number of available sequences. However, an error arises if the dataset gets updated between the two requests and new sequences (including some from Oceania) are added. Then, when you divide the number from the first request by the number of the second request to obtain the proportion, the result would be too small and neither reflect the correct proportion in the previous dataset nor the proportion in the new dataset.

Solution:

Every server response contains the version number of the data. It is provided in the HTTP response header `lapis-data-version`. For JSON responses, it is additionally provided in the `dataVersion` field. It is recommended to compare the data versions of the different responses. If they are not the same, the data should be re-fetched.

The data version is the Unix timestamp of the moment when the dataset was put in place.

CHAPTER

THREE

DATE HANDLING

Note: Partial dates are currently only supported by the monkeypox instance. For SARS-CoV-2, the `year` and `month` fields do not exist.

The `date` field returns and the `dateFrom` and `dateTo` parameters expect a string formatted as YYYY-MM-DD (e.g., 2022-05-29). There are however samples for which we do not know the exact date but only a partial date: e.g., only the year or the year and the month. In those cases, the `date` is considered as unknown and will return a `null`. That means that the query `dateFrom=2022-01-01` will not return samples for which we do not know the exact date but only that it is from May 2022.

To support partial dates, LAPIS additionally has the fields `year` and `month`. They are returned by the `details` endpoint and can be used as an aggregation field (e.g., `fields=year,month` is possible). Further, LAPIS offers `yearFrom`, `yearTo`, `yearMonthFrom` and `yearMonthTo` filters. `yearMonth` has to be formatted as YYYY-MM. For example, the queries `yearFrom=2022` and `yearMonthFrom=2022-05` will include all samples from May 2022.

3.1 Background

Why is the query `dateFrom=2022-01-01` not returning samples from May 2022 that don't have an exact date? The reason is that the following (desirable) property would be violated:

```
For t0 < t1:

aggregated(dateFrom=t0)
= aggregated(dateFrom=t0,dateTo=t1) + aggregated(dateFrom=t1+1)
= sum(aggregated(dateFrom=t0,fields=date))
```

CHAPTER

FOUR

MUTATION FILTERS

It is possible to filter for amino acid and nucleotide bases/mutations. Multiple mutations can be provided by specifying a comma-separated list.

A nucleotide mutation has the format *<position><base>*. A "base" can be one of the four nucleotides *A*, *T*, *C*, and *G*. It can also be - for deletion and *N* for unknown.

An amino acid mutation has the format *<gene>:<position><base>*. The following genes are available: E, M, N, ORF1a, ORF1b, ORF3a, ORF6, ORF7a, ORF7b, ORF8, ORF9b, S. A "base" can be one of the 20 amino acid codes. It can also be - for deletion and *X* for unknown.

The *<base>* can be omitted to filter for any mutation. You can write a . for the *<base>* to filter for sequences for which it is confirmed that no mutation occurred, i.e., has the same base as the reference genome at the specified position.

CHAPTER

FIVE

PANGO LINEAGE QUERY

Pango lineage names inherit the hierarchical nature of genetic lineages. For example, B.1.1 is a sub-lineage of B.1. More information about the pango nomenclature can be found on the website of the Pango network.

With the `pangoLineage` filter and in *variant queries*, it is possible to not only filter for a very specific lineage but also to include its sub-lineages. To include sub-lineages, add a `*` at the end. For example, writing B.1.351 will only give samples of B.1.351. Writing B.1.351* or B.1.351.* (there is no difference between these two options) will return B.1.351, B.1.351.1, B.1.351.2, etc.

An official pango lineage name can only have at most three number components. A sub-lineage of a lineage with a maximal-length name (e.g., B.1.617.2) will get an alias. A list of aliases can be found here. B.1.617.2 has the alias AY so that AY.1 would be a sub-lineage of B.1.617.2. LAPIS is aware of aliases. Filtering B.1.617.2* will include every lineage that starts with AY. It is further possible to search for B.1.617.2.1 which will then return the same results as AY.1.

CHAPTER

SIX

RESPONSE FORMAT

Most endpoints can return data either as JSON or CSV. The default is JSON. To get a CSV, specify the query parameter `dataFormat=csv`.

Responses returned in JSON have three top level attributes:

- "info" - data about the API itself
- "errors" - an array (hopefully empty!) of things that went wrong
- "data" - the actual response data

Example:


```
{
  "info":{"apiVersion":1,"dataVersion":1653160874,"deprecationDate":null,"deprecationInfo
↪":null,"acknowledgement":null},
  "errors":[],
  "data":[{"count":84}]
}
```


Genomic sequences (`fasta` and `fasta-aligned` endpoints) are returned in the FASTA format.

Note: Every response, independent of the data format, contains the data version as it is a very important information. See the *data versions page* for details.

CHAPTER

SEVEN

VARIANT QUERY

LAPIS offers a special query language to specify variants. A variant query can be used to filter sequences and be passed to the server through the query parameter `variantQuery`. It is not allowed to use the `variantQuery` parameter alongside other variant-defining parameters (e.g., `pangoLineage` or `aaMutations`). Don't forget to encode/escape the query correctly (in JavaScript, this can be done with the "encodeURIComponent()" function)!

The formal specification of the query language is available here as an ANTLR v4 grammar. In following, we provide an informal description and examples.

The query language understands Boolean logic. Expressions can be connected with `&` (and), `|` (or) and `!` (not). Parentheses `(` and `)` can be used to define the order of the operations. Further, there is a special syntax to match sequences for which at least or exactly `n` out of a list of expressions are fulfilled.

7.1 Examples

Get the sequences with the nucleotide mutation 300G, without a deletion at position 400 and either the AA change S:123T or the AA change S:234A:

```
300G & !400- & (S:123T | S:234A)
```

Get the sequences with at least 3 out of five mutations/deletions:

```
[3-of: 123A, 234T, S:345-, ORF1a:456K, ORF7:567-]
```

Get the sequences that fulfill exactly 2 out of 4 conditions:

```
[exactly-2-of: 123A & 234T, !234T, S:345- | S:346-, [2-of: 222T, 333G, 444A, 555C]]
```

For , it is also possible to use pango lineage queries (either called by pangolin or by Nextclade) and filter by Nextstrain clades:

```
BA.5* | nextcladePangoLineage:BA.5* | nextstrainClade:22B
```

CHAPTER

EIGHT

REFERENCE: MONKEYPOX INSTANCE

The API for monkeypox uses all monkeypox data on NCBI GenBank and from authors who shared them directly with us. The sequences are aligned with Nextclade.

8.1 Overview

The API has the following endpoints related to samples. These endpoints provide different types of data:

- `/sample/aggregated` - to get summary data aggregated across samples
- `/sample/details` - to get per-sample metadata
- `/sample/contributors` - to get author names of the samples
- `/sample/nuc-mutations` - to get the common nucleotide mutations (shared by at least 5% of the sequences)
- `/sample/fasta` - to get original (unaligned) sequences
- `/sample/fasta-aligned` - to get aligned sequences

The API returns a response (data) based on a query to one of the endpoints. You can view a response in your browser, or use the data programmatically.

8.1.1 Query Format

To query an endpoint, use the web link with prefix `https://mpox-lapis.genspectrum.org/v1` and the suffix for the relevant endpoint. In the examples, we only show the suffixes to keep things simple, but a click takes you to the full link in your browser.

Query example: Get the total number of available sequences: /sample/aggregated

8.1.2 Response Format

See *Response format*

8.2 Filters

We can adapt the query to filter to only samples of interest. The syntax for adding filters is `<attribute1>=<valueA>&<attribute2>=<valueB>`.

All **sample** endpoints can be filtered by the following attributes:

- dateFrom (see *Date handling*)
- dateTo
- yearFrom
- yearTo
- yearMonthFrom
- yearMonthTo
- dateSubmittedFrom
- dateSubmittedTo
- region
- country
- division
- host
- variantQuery (see *Variant query*)
- clade
- lineage
- nucMutations

The endpoints `details`, `contributors`, `nuc-mutations`, `fasta`, and `fasta-aligned` can additionally be filtered by these attributes:

- sraAccession
- strain

To determine which values are available for each attribute, see the example in section "Aggregation".

8.2.1 Mutation filters

See *Mutation filters*

8.3 Aggregation

Above, we used the `/sample/aggregated` endpoint to get the total counts of sequences with or without filters. Using the query parameter `fields`, we can group the samples and get the counts per group. For example, we can use it to get the number of samples per country. We can also use it to list the available values for each attribute.

`fields` accepts a comma-separated list. The following values are available:

- date (see *Date handling*)
- year

- month
- dateSubmitted
- region
- country
- division
- host
- clade
- lineage

CHAPTER

NINE

REFERENCE: SARS-COV-2 INSTANCE

The API for SARS-CoV-2 uses all SARS-CoV-2 data on NCBI GenBank The sequences were pre-processed by Nextstrain.

9.1 Overview

The API has the following endpoints related to samples. These endpoints provide different types of data:

- `/sample/aggregated` - to get summary data aggregated across samples
- `/sample/details` - to get per-sample metadata
- `/sample/contributors` - to get author names of the samples
- `/sample/aa-mutations` - to get the common amino acid mutations
- `/sample/nuc-mutations` - to get the common nucleotide mutations
- `/sample/fasta` - to get original (unaligned) sequences
- `/sample/fasta-aligned` - to get aligned sequences

The API returns a response (data) based on a query to one of the endpoints. You can view a response in your browser, or use the data programmatically.

9.1.1 Query Format

To query an endpoint, use the web link with prefix `https://lapis.cov-spectrum.org/open/v1` and the suffix for the relevant endpoint. In the examples, we only show the suffixes to keep things simple, but a click takes you to the full link in your browser.

Query example: Get the total number of available sequences: /sample/aggregated

9.1.2 Response Format

See *Response format*

9.2 Filters

We can adapt the query to filter to only samples of interest. The syntax for adding filters is `<attribute1>=<valueA>&<attribute2>=<valueB>`.

All **sample** endpoints can be filtered by the following attributes:

- dateFrom
- dateTo
- dateSubmittedFrom
- dateSubmittedTo
- region
- country
- division
- location
- regionExposure
- countryExposure
- divisionExposure
- ageFrom
- ageTo
- sex
- host
- samplingStrategy
- variantQuery (see *Variant query*)
- pangoLineage
- nextcladePangoLineage
- nextstrainClade
- gisaidClade
- submittingLab
- originatingLab
- nucMutations
- aaMutations
- nextcladeQcOverallScoreFrom
- nextcladeQcOverallScoreTo
- nextcladeQcMissingDataScoreFrom

- nextcladeQcMissingDataScoreTo
- nextcladeQcMixedSitesScoreFrom
- nextcladeQcMixedSitesScoreTo
- nextcladeQcPrivateMutationsScoreFrom
- nextcladeQcPrivateMutationsScoreTo
- nextcladeQcSnpClustersScoreFrom
- nextcladeQcSnpClustersScoreTo
- nextcladeQcFrameShiftsScoreFrom
- nextcladeQcFrameShiftsScoreTo
- nextcladeQcStopCodonsScoreFrom
- nextcladeQcStopCodonsScoreTo

The endpoints `details`, `contributors`, `nuc-mutations`, `fasta`, and `fasta-aligned` can additionally be filtered by these attributes:

- genbankAccession
- sraAccession
- gisaidEpiIsl
- strain

To determine which values are available for each attribute, see the example in section "Aggregation".

9.2.1 Mutation filters

See *Mutation filters*

9.2.2 Pango lineage filter

See *Pango lineage query*

9.3 Aggregation

Above, we used the `/sample/aggregated` endpoint to get the total counts of sequences with or without filters. Using the query parameter `fields`, we can group the samples and get the counts per group. For example, we can use it to get the number of samples per country. We can also use it to list the available values for each attribute.

`fields` accepts a comma-separated list. The following values are available:

- date
- dateSubmitted
- region
- country
- division
- location

- regionExposure
- countryExposure
- divisionExposure
- age
- sex
- host
- samplingStrategy
- pangoLineage
- nextcladePangoLineage
- nextstrainClade
- gisaidClade
- submittingLab
- originatingLab