ImmuneDB Documentation

Release 0.30.0

Aaron Rosenfeld

Jul 12, 2019

Getting Started

ImmuneDB is a system to efficiently storage and analyze high-throughput B- and T-cell sequencing data. It provides V- and J-gene identification, clonal assignment, lineage construction, selection pressure calculation, and thorough exporting functionality.

It also provides an intuitive and useful web interface a demo of which you can see here.

CHAPTER 1

Quick Start

To get started immediately, please see the *Docker installation instructions*.

CHAPTER 2

More Information

ImmuneDB is comprised of two GitHub repositories: Python analysis tools (arosenfeld/immunedb) and a web interface (arosenfeld/immunedb-frontend)

The system aims to:

- **Reduce ad-hoc scripting:** Data analysis performed on an ad-hoc basis with custom scripts and data formats is error-prone and leads to inconsistencies. ImmuneDB provides a standardized analysis platform, performing many common tasks automatically.
- **Minimize flat-files:** Flat files are currently the standard method of data exchange in the biological sciences. There are a myriad of drawbacks when using these including a lack of referential integrity, unclear provenance, and non-standardized formats.

  ImmuneDB attempts to reduce the need for flat files, through the use of an industry-leading database, MySQL. When data must be exchanged as a flat-file, many export options, including FASTA and tab-delineation, are available.
- **Interoperate with existing tools:** ImmuneDB integrates tools from other researchers to provide features such as lineage construction, genotyping, and selection pressure calculations. Further, ImmuneDB can import and export in a variety of common formats, making it compatible with the larger AIRR ecosystem of tools.

2.1 Installing with Docker (recommended)

2.1.1 Pulling the Docker Image

With Docker installed the following command to get the newest version of the ImmuneDB Docker image:

```
$ docker pull arosenfeld/immunedb:v0.30.0
```

2.1.2 Running the Container

To start a shell session within the container run:

```
$ docker run -v $HOME/immunedb_share:/share \
    -p 8080:8080 -it arosenfeld/immunedb:v0.30.0
```

This will start a shell with ImmuneDB and accessory scripts pre-installed as well as create a shared directory between the host and Docker container. Files placed in the host's `$HOME/immunedb_share` directory and it will appear in `/share` within the Docker container (and vice versa). Note `$HOME` on macOS is generally `/Users/your_username/` and on Linux it is generally `/home/your_username`.

Additionally, MySQL stores its data in `/share/mysql_data` so databases will persist across multiple container invocations.

The location of important files are:

- `/root/germlines`: IMGT aligned germlines for IGH, TCRA, and TCRB.
- `/apps/bowtie2/bowtie2`: The local-alignment tool Bowtie2. This file is in the container's `$PATH`.
- `/share/mysql_data`: The location MySQL (specifically MariaDB) will store its data.
- `/example`: A set of example input data to familiarize yourself with ImmuneDB

Next Steps

Once the Docker container is running, you should continue by testing out the *example pipeline*.

2.2 Installing Locally (advanced)

This section details how to set ImmuneDB up locally on a machine. This is a more complicated process than using the *Docker method* but may be useful if you plan on running ImmuneDB remotely on a server rather than locally.

2.2.1 Dependency Installation

MySQL

ImmuneDB utilizes MySQL as its underlying data store. We recommend using its drop-in replacement, MariaDB. Please consult their website and your operating systems package manager for installation instructions.

R (optional)

Baseline can optionally be used to calculate selection pressure on clones. This requires R to be installed along with the ade4 package. Installation is platform dependent.

The newest version of Baseline can be downloaded here. The path to the main script will be needed for clone statistics generation as described in *Statistics Generation*.

For genotyping, TIgGER must also be installed.

Bowtie2 (optional)

Bowtie2 can be used to locally align sequences which cannot be aligned using the built-in anchor method.

Clearcut (optional)

Clearcut can be used to generate lineage trees for clones. After downloading and compiling per the instructions, note the path to the `clearcut` executable which will be required for generating trees in *Clone Trees (Optional)*.

2.2.2 ImmuneDB Installation

It is recommended that ImmuneDB be installed within a *venv*, creating an isolated environment from the rest of the system.

To create a virtual environment and activate it run:

```
$ python3 -m venv immunedb
$ source immunedb/bin/activate
```

Then install ImmuneDB:

```
$ pip install immunedb
```

2.2.3 Web Interface Installation

Please refer to the ImmuneDB Frontend installation instructions.

2.3 Running the Example Pipeline

This page serves to familiarize new users with the basic flow of running the ImmuneDB pipeline. Example input FASTQ files are provided which contain human B-cell heavy chain sequences.

Commands are listed as either being run in either the Docker container or on the host.

To begin, run the Docker container *as documented*:

Listing 1: Run on Host

```
$ docker run -v $HOME/immunedb_share:/share \
    -p 8080:8080 -it arosenfeld/immunedb:v0.29.0
```

2.3.1 Metadata Specification

Before ImmuneDB can be run, metadata must be specified for each input file. For this example, one has already been created for you. To learn how to create a metadata file for your own data, see *Creating a Metadata Sheet*.

2.3.2 ImmuneDB Instance Creation

Next, we create a database for the data with:

Listing 2: Run in Docker

```
$ immunedb_admin create example_db
```

This creates a new database named `example_db`.

2.3.3 Identifying or Importing Sequences

The first step of the pipeline is to annotate sequences and store the resulting data in the newly created database. To do so, the `immunedb_identify` is used. It requires that V and J germline sequences be specified in two separate FASTA files. The Docker image provides Human & Mouse IGH, TRA, and TRB germlines in `$HOME/germlines`.

For this example, there are two provided input files in `/example` along with the requisite `metadata.tsv` file which you can view with:

Listing 3: Run in Docker

```
$ ls /example
```

Given this, run the `immunedb_identify` command:

Listing 4: Run in Docker

```
$ immunedb_identify example_db \
     /root/germlines/imgt_human_ighv.fasta \
     /root/germlines/imgt_human_ighj.fasta \
     /example
```

2.3.4 Sequence Collapsing

ImmuneDB determines the uniqueness of a sequence both at the sample and subject level. For the latter, `immunedb_collapse` is used to find sequences that are the same except at positions that have an `N`. Thus, the sequences `ATNN` and `ANCN` would be collapsed.

To collapse sequences, run:

Listing 5: Run in Docker

```
$ immunedb_collapse example_db
```

2.3.5 Clonal Assignment

After sequences are assigned V and J genes, they can be clustered into clones based on CDR3 Amino Acid similarity with the `immunedb_clones` command. This takes a number of arguments which should be read before use.

There are three ways to create clones: based on CDR3 AA similarity, T-cell exact CDR3 NT identity, and a lineage based method. For this example we'll use the similarity based method with default parameters:

Listing 6: Run in Docker

```
$ immunedb_clones example_db similarity
```

This will create clones where all sequences in a clone will have the same V-gene, J-gene, and (by default) 85% CDR3 AA identity.

2.3.6 Statistics Generation

Two sets of statistics can be calculated in ImmuneDB:

- **Clone Statistics:** For each clone and sample combination, how many unique and total sequences appear as well as the mutations from the germline.

- **Sample Statistics:** Distribution of sequence and clone features on a per-sample basis, including V and J usage, nucleotides matching the germline, copy number, V length, and CDR3 length. It calculates all of these with and without outliers, and including and excluding partial reads.

These are calculated with the `immunedb_clone_stats` and `immunedb_sample_stats` commands and must be run in that order.

Listing 7: Run in Docker

```
$ immunedb_clone_stats example_db
$ immunedb_sample_stats example_db
```

2.3.7 Selection Pressure (Optional)

Warning: Selection pressure calculations are time-consuming, so you can skip this step if time is limited.

Selection pressure of clones can be calculated with Baseline. To do so run:

Listing 8: Run in Docker

```
$ immunedb_clone_pressure example_db \
    /apps/baseline/Baseline_Main.r
```

Note, this process is relatively slow and may take some time to complete.

2.3.8 Clone Trees (Optional)

Lineage trees for clones is generated with the `immunedb_clone_trees` command. The only currently supported method is neighbor-joining as provided by Clearcut.

Among others, the `--min-mut-copies` parameter allows for mutations to be omitted if they have not occurred at least a specified number of times. This can be useful to correct for sequencing error.

Listing 9: Run in Docker

```
$ immunedb_clone_trees example_db --min-mut-copies 2
```

2.3.9 Web Interface

ImmuneDB has a web interface to interact with a database instance. The Docker container automatically makes this available at http://localhost:8080/frontend/example_db

When you create more databases, simply replace *example_db* with the proper databse name.

2.4 Running the Pipeline on Your Data

This page describes how to run the ImmuneDB pipeline on raw FASTA/FASTQ data. It is assumed that you've previously tried the *example pipeline* and understand the basics of running commands in the Docker container.

Like in the example, each code block has a header saying if the command should be run on the host or in the Docker container.

2.4.1 Copying Your Sequence Data Into Docker

Unlike in the *example pipeline* where sequencing data was provided, you'll need to copy your own FASTA/FASTQ sequencing data into the Docker container.

To do so, on the host, we create a new directory in the shared directory into which we'll copy your sequencing data. Here we're calling it `sequences` but you'll probably want to choose a more descriptive name:

Listing 10: Run on Host

```
$ mkdir $HOME/immunedb_share/sequences
$ cp PATH_TO_SEQUENCES $HOME/immunedb_share/sequences
```

2.4.2 Creating a Metadata Sheet

Next, we'll use the `immunedb_metadata` command to create a template metadata file for your sequencing data. In the Docker container run:

Listing 11: Run in Docker

```
$ cd /share/sequences
$ immunedb_metadata --use-filenames
```

This creates a `metadata.tsv` file in `/share/sequences` in Docker or `$HOME/immunedb_share/sequences` on the host.

The `--use-filenames` flag is optional, and simply populates the `sample_name` field with the file names stripped of their `.fasta` or `.fastq` extension.

Editing the Metadata Sheet

On the host open the `$HOME/immunedb_share/sequences` file in Excel or your favorite spreadsheet editor. The headers included in the file are **required**. You may add additional headers as necessary for your dataset (e.g. `tissue`, `cell_subset`, `timepoint`) so long as they follow the following rules:

- The headers must all be unique
- Each header may only contain *lowercase* letters, numbers, and underscores
- Each header must begin with a (lowercase) character
- Each header must not exceed 32 characters in length
- The *values* within each column cannot exceed 64 characters in length

---

Note: When data is missing or not necessary in a field, leave it blank or set to `NA`, `N/A`, `NULL`, or `None` (case-insensitive).

2.4.3 Running the Pipeline

Much of the rest of the pipeline follows from the example pipeline's *instance creation step*. To start, create a database. Here we'll call it `my_db` but you'll probably want to give it a more descriptive name:

Listing 12: Run in Docker

```
$ immunedb_admin create my_db
```

Then we'll identify the sequences. For this process the germline genes must be specified. The germlines provided as FASTA files in the Docker image are:

- `imgt_human_ighv` & `imgt_human_ighj`: Human B-cell heavy chains
- `imgt_human_trav` & `imgt_human_traj`: Human T-cell α chains
- `imgt_human_trbv` & `imgt_human_trbj`: Human T-cell β chains
- `imgt_mouse_ighv` & `imgt_mouse_ighj`: Mouse B-cell heavy chains

For this segment we'll assume human B-cell heavy chains, but the process is the same for any dataset:

Listing 13: Run in Docker

```
$ immunedb_identify my_db \
      /root/germlines/imgt_human_ighv.fasta \
      /root/germlines/imgt_human_ighj.fasta \
      /share/sequences
$ immunedb_collapse my_db
```

Then we assign clones. For B-cells we recommend:

Listing 14: Run in Docker

```
$ immunedb_clones my_db similarity
```

For T-cells we recommend:

Listing 15: Run in Docker

```
$ immunedb_clones my_db similarity --level nt \
      --min-similarity 1
```

If you have a mixed dataset, you can assign clones in different ways, filtering on V-gene type. For example:

Listing 16: Run in Docker

```
$ immunedb_clones my_db similarity --gene IGHV
$ immunedb_clones my_db similarity --gene TCRB \
      --level nt --min-similarity 1
```

The last required step is to generate aggregate statistics:

Listing 17: Run in Docker

```
 $ immunedb_clone_stats my_db
 $ immunedb_sample_stats my_db
```

For B-cells, you might want to generate lineages too. The following excludes mutations that only occur once. `immunedb_clone_trees` has many other parameters for filtering which you can view with the `--help` flag:

Listing 18: Run in Docker

```
$  immunedb_clone_trees my_db --min-mut-copies 2
```

Selection pressure can be run with the following. This process is quite time-consuming, even for small datasets:

Listing 19: Run in Docker

```
$ immunedb_clone_pressure my_db \
     /apps/baseline/Baseline_Main.r
```

Finally, the data should be available at http://localhost:8080/frontend/my_db.

2.5 Python API

Note: This section is currently incomplete. We're working to fill out the details of the Python API as soon as possible.

2.5.1 Configuration

The `immunedb.common.config` module provides methods to initialize a connection to a new or existing database.

Most programs using ImmuneDB will start with code similar to:

```
import immunedb.common.config as config


parser = config.get_base_arg_parser('Some description of the program')
# ... add any additional arguments to the parser ...
args = parser.parse_args()

session = config.init_db(args.db_config)
```

When this script is run, it will require at least one argument which is the path to a database configuration (as generated with `immunedb_admin`). Using that, a `Session` object will be made, connected to the associated database.

One can also directly specify the path to a configuration directly.

```
import immunedb.common.config as config


session = config.init_db('path/to/config')
```

Alternatively a dictionary with the same information can be passed:

```
import immunedb.common.config as config


session = config.init_db({
    'host': '...',
    'database': '...',
    'username': '...',
    'password': '...',
})
```

Returned will be a `Session` object which can be used to interact with the database.

2.5.2 Using the Session

ImmuneDB is built using SQLAlchemy as a MySQL abstraction layer. Simply put, instead of writing SQL, the database is queried using Python constructs. Full documentation on using the session can be found in SQLAlchemy's documentation.

Once a session is created, the models listed below can be queried.

2.5.3 Example Queries

Below are some example queries that demonstrate how to use the ImmuneDB API.

Clone CDR3s

Get all clones with a given V-gene and print their CDR3 AA sequences.

Input

```
import immunedb.common.config as config
from immunedb.common.models import Clone

session = config.init_db(...)

for clone in session.query(Clone).filter(Clone.v_gene == 'IGHV3-30'):
    print('clone {} has AAs {}'.format(clone.id, clone.cdr3_aa))
```

Output

```
clone 37884 has AAs CARGYSSSYFDYW
clone 37886 has AAs CARSRTSLSIYGVVPTGDFDSW
clone 37885 has AAs CARNGLNTVSGVVISPKYWLDPW
clone 37887 has AAs CARDLFRGVDFYYYGMDVW
```

Clone Frequency

Determine how many sequences appear in each sample belonging to clone 1234.

Note the `CloneStats` model has one entry for each clone/sample combination plus one where the `sample_id` field is `null` which represents the overall clone.

Input

```
import immunedb.common.config as config
from immunedb.common.models import CloneStats

session = config.init_db(...)
for stat in session.query(CloneStats).filter(
        CloneStats.clone_id == 1234).order_by(CloneStats.sample_id):
    print('clone {} has {} unique sequences and {} copies {}'.format(
        stat.clone_id,
        stat.unique_cnt,
        stat.total_cnt,
        ('in sample ' + stat.sample.name) if stat.sample else 'overall'))
```

Output

```
clone 1234 has 53 unique sequences and 1331 copies overall
clone 1234 has 27 unique sequences and 379 copies in sample sample1
clone 1234 has 27 unique sequences and 339 copies in sample sample3
clone 1234 has 24 unique sequences and 311 copies in sample sample4
clone 1234 has 28 unique sequences and 302 copies in sample sample10
```

V-gene Usage

This is a more complex query which gathers the V-gene usage of all sequences which are (a) in subject with ID 1, (b) associated with a clone, and (c) are unique to the subject, printing them from least to most frequent.

Input

```
import immunedb.common.config as config
from immunedb.common.models import Sequence, SequenceCollapse

session = config.init_db(...)

subject_unique_seqs = session.query(
    func.count(Sequence.seq_id).label('count'),
    Sequence.v_gene
).join(
    SequenceCollapse
).filter(
    Sequence.subject_id == 1,
    ~Sequence.clone_id.is_(None),
    SequenceCollapse.copy_number_in_subject > 0
).group_by(
    Sequence.v_gene
).order_by(
    'count'
)

for seq in subject_unique_seqs:
    print(seq.v_gene, seq.count)
```

Output

```
# ... output truncated ...
IGHV4-34 1128
IGHV1-2 1160
IGHV3-48 1169
IGHV4-39 1310
IGHV3-7 1345
IGHV3-30|3-30-5|3-33 1607
IGHV3-23|3-23D 1626
IGHV3-21 1878
```

2.5.4 Data Models

2.6 Directly Querying the Database

ImmuneDB is backed by a MySQL database that can be queried directly to gather information, bypassing the Python API.

2.6.1 Accessing the Database

There are many ways to access the database directly. The two introduced here are directly through MySQL or using `immunedb_sql` which simply wraps a call to MySQL.

With the `immunedb_sql` wrapper (recommended)

```
$ immunedb_sql PATH_TO_CONFIG
```

This is entirely equivalent to using `mysql` and will drop to the MySQL interpreter. You can also pass a query directly from the command line. For example:

```
$ immunedb_sql PATH_TO_CONFIG --query 'select * from samples'
```

Directly with MySQL

From the command line, you may access an ImmuneDB database `DATABASE` from user `USERNAME` with:

```
$ mysql -u USERNAME -p DATABASE
```

This will prompt for a password and then to the database. This method of access is useful for quickly querying the database. To save results of a query `QUERY` run the command:

```
$ mysql -u USERNAME -p DATABASE -e "QUERY" > output
```

2.7 Referencing ImmuneDB

If you use ImmuneDB, please cite the tool as:

Rosenfeld, A. M., Meng, W., Luning Prak, E. T., Hershberg, U., **ImmuneDB, a Novel Tool for the Analysis, Storage, and Dissemination of Immune Repertoire Sequencing Data**. Frontiers in Immunology **9** (2018).

ImmuneDB was originally announced previously in:

Rosenfeld, A. M., Meng, W., Luning Prak, E. T., Hershberg, U., **ImmuneDB: a system for the analysis and exploration of high-throughput adaptive immune receptor sequencing data**, Bioinformatics **33** (2016), no. 2, 292–293.

2.8 Related Publications & Software

Clumpiness: Schwartz, G.W., Shokoufandeh, A., Ontanon, S., Hershberg, U. (2016) **Using a novel clumpiness measure to unite data with metadata: finding common sequence patterns in immune receptor germline V genes**. Pattern Recognition Letters 74: 24-29

ConservedIdentification: Zhang, B., Meng, W., Luning Prak, E.T. and Hershberg U. (2015) **Discrimination of germline V genes at different sequencing lengths and mutational burdens: A new tool for identifying and evaluating the reliability of V gene assignment**. Journal of Immunological Methods 427: 105-116

Diversity: Schwartz, G.W., Hershberg, U. (2013) **Conserved variation: identifying patterns of stability and variability in BCR and TCR V genes with different diversity and richness metrics**. Phys Biol 10