ImmuneDB Documentation

Release 0.29.9

Aaron Rosenfeld

Sep 14, 2020

Getting Started

ImmuneDB is a database-backed system to analyze and store large amounts (terabytes) of high-throughput B-cell receptor (BCR) and T-cell receptor (TCR) data. Although it can be used as a stand-alone package for comprehensive repertoire profiling, ImmuneDB excels at acting as a central data store and interface between other tools such as IgBLAST, the Immcantation Framework, MiXCR, and VDJtools via AIRR compliant importing and exporting routines.

CHAPTER 1

Feature Highlights

- **Relational storage of repertoire data**: Sequences, annotations, clones, lineages, and statistics are all stored in a relational database to promote consistent formatting and easy querying.
- **Consolidated metadata**: Custom study, experiment, and replicate metadata is stored alongside your sequencing data in a non-redundant format to avoid inconsistencies and errors over the life of your study.
- **Web interface**: ImmuneDB provides a built-in web interface for interactive exploration of data.
- **Interoperability**: With AIRR compliant input and output methods, ImmuneDB can interface with other software in the AIRR ecosystem. Other output formats include Change-O and VDJtools.
- **Proven reliability**: ImmuneDB is used by multiple labs to manage terabytes of data comprised of billions of sequences in dozens of projects.

CHAPTER 2

Quick Start

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

2.1 Installing with Docker (recommended)

2.1.1 Pulling the Docker Image

With Docker installed, run the following command to pull the newest version of the ImmuneDB Docker image:

```
$ docker pull arosenfeld/immunedb:v0.29.9
```

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:v0.29.9
```

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/configs`: The recommended directory to store ImmuneDB configurations generated by `immunedb_admin create`.
- `/share/mysql_data`: The location MySQL (specifically MariaDB) will store its data.
- `/example`: A set of example input data to familiarize yourself with ImmuneDB.

2.1.3 Running ImmuneDB

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 process 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. All `immunedb_*` commands have a `--help` flag which will show all arguments and their descriptions. It is recommended for each command you run the help flag to see options not listed in this documentation.

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:v0.29.9
```

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 /share/configs
```

This creates a new database named `example_db` and stores its configuration in `/share/configs/example_db.json`.

2.3.3 Identifying or Importing Sequences

Data can be added to the new ImmuneDB database either by importing annotated sequencing data in AIRR format, or via a built-in gene assignment method based on Zhang, et al., 2015.

For this example, there are two input FASTQ files in `/example/fastq` along with an associated metadata file. These will be used regardless of the method you choose. There are also germline files for human and mouse included.

Option 1: Importing from AIRR Files (Recommended)

First, IgBLAST needs to be run on the input files. A small wrapper script is provided in the Docker container. It takes three parameters: a species, a locus (in uppercase), an input directory with FASTA/FASTQ files, and an output directory:

```
$ run_igblast.sh human IGH /example/fastq /example/airr
```

To import these files, run the `immunedb_import` command:

Listing 3: Run in Docker

```
$ immunedb_import /share/configs/example_db.json airr \
    /root/germlines/igblast/human/IGHV.gapped.fasta \
    /root/germlines/igblast/human/IGHJ.gapped.fasta \
    /example/airr
```

Option 2: Annotating FASTA/FASTQ Files via Anchoring

Alternatively, if you'd prefer to use the built-in annotation method on FASTA/FASTQ files, you can use the `immunedb_identify` command. Note this method is more sensitive to high mutation rates in the regions flanking the CDR3.

Listing 4: Run in Docker

```
$ immunedb_identify /share/configs/example_db.json \
    /root/germlines/anchor/human/IGHV.gapped.fasta \
    /root/germlines/anchor/human/IGHJ.gapped.fasta \
    /example
```

2.3.4 Sequence Collapsing

After data are imported or annotated on a sample-level basis, ImmuneDB determines the subject-level unique sequences; that is, the set of unique sequences across all samples in each subject. Because sequences may contain the ambiguous `N` symbol, the process is not trivial string equality checking. It is implemented in the `immunedb_collapse` command.

To collapse sequences, run:

Listing 5: Run in Docker

```
$ immunedb_collapse /share/configs/example_db.json
```

2.3.5 Clonal Assignment

After collapsing unique sequences across each subject they can be grouped into clones which are aggregations of sequences likely deriving from a common progenitor cell.

ImmuneDB offers two clonal inference methods, *similarity* and *cluster*. The *cluster* method is recommended and documented here as it more robust than *similarity*.

For both methods, clones are inferred in two steps: grouping sequences and then merging similar clones. Both steps are run together with the `immunedb_clones` command

By default, only sequences with a subject-level copy number greater than 1 are included in clones. This can be changed with the `--min-copy` parameter.

In the first step of clonal inference, sequences meeting the above copy number criteria are hierarchically clustered together such that any two sequences in a clone must (1) have the same CDR3 length and (2) share at least 85% amino-acid similarity in the CDR3. The similarity can be changed with `--min-similarity X` parameter where X is the minimum similarity between 0 and 1. If nucleotide similarity should be used, `--level nt` can be passed.

Note: For T-cells it is recommended the `--min-similarity 1` parameter be set but the `--level` parameter by left at the default amino-acid setting. Using both `--min-similarity 1 --level nt` may lead to the creation of spurious clones due to sequencing error. Only pass both if you're quite certain your sequencing error has been eliminated (e.g. by barcoding).

After this step is complete, sequences have been assigned to clones. In some cases clones may have the same CDR3 nucleotide sequence but different gene calls. This can indeed occur biologically but frequently due to mutation and sequencing error causing incorrect gene calls.

To rectify this, a second step in clonal inference is to collapse merge clones that have the same CDR3 nucleotide sequences. In cases where this occurs, the highest copy clone absorbs the lower copy clones. This second step can be configured in two ways via the `--reduce-difference` flag. Setting it to a negative number (e.g. `--reduce-difference -1`) disables the step entirely. Setting it to a positive number (e.g. `--reduce-difference 2`) will alter the step's behavior to combine clones differing by at most that number of nucleotides. The default value is 0, so only clones with exactly the same CDR3 nucleotide sequences will be combined.

Listing 6: Run in Docker

```
$ immunedb_clones /share/configs/example_db.json cluster
```

2.3.6 Statistics Generation

Two sets of statistics can be calculated in ImmuneDB:

- **Clone Statistics:** For each clone and sample combination, statistics on the clone's size, mutation level, and top copy sequence
- **Sample Statistics:** Distribution of sequence and clone features on a per-sample basis, including gene usage, mutation level, copy number, CDR3 length.

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 /share/configs/example_db.json
$ immunedb_sample_stats /share/configs/example_db.json
```

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 /share/configs/example_db.json \
    /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.

There are many parameters that can be changed for tree construction:

- `--min-seq-copies` (default 0): The minimum number copy number required for a sequence to be included in the tree.
- `--min-seq-samples` (default 0): The minimum number samples in which a sequence must appear for it to be included in the tree.
- `--min-mut-copies` (default 0): The minimum number of copies in which a mutation must occur to be included in the tree.
- `--min-mut-samples` (default 0): The minimum number of samples in which a mutation must occur to be included in the tree.
- `--exclude-stops` (default `False`): Exclude sequences with a stop codon.
- `--full-seq` (default `False`): By default only the V-region of each sequence (the portion 5' of the CDR3) is included in the tree construction. Setting this flag will use the entire sequence.

Generally we recommend using `--min-seq-copies 2`.

Listing 9: Run in Docker

```
$ immunedb_clone_trees /share/configs/example_db.json --min-seq-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 database name.

2.3.10 Next Steps

Now that the basic workflow has been covered, instructions to run ImmuneDB on your own data can be found at *Running the Pipeline on Your Data*.

2.4 Running the Pipeline on Your Data

This page describes how to run the ImmuneDB pipeline on your own BCR/TCR 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 or AIRR-formatted IgBLAST output 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. Replace `PATH_TO_SEQUENCES` with the path to your sequencing data.

Listing 10: Run on Host

```
$ mkdir -p $HOME/immunedb_share/input
$ cp PATH_TO_SEQUENCES $HOME/immunedb_share/input
```

2.4.2 Running IgBLAST (optional)

Note: If your data is already in AIRR-compliant IgBLAST format or you are planning on using the built in anchoring method, you can skip this step.

The following command will run IgBLAST on your files. Valid values for species and locus are:

- `SPECIES`:`human`, `mouse`
- `LOCUS`: `IGH`, `IGL`, `IGK`, `TRA`, `TRB`,

```
$ run_igblast.sh SPECIES LOCUS /share/input /share/input
```

For consistency with the commands in the rest of this tutorial, we'll move the new IgBLAST output files to `/share/input` and move the FASTA/FASTQ files to `/share/sequences`.

```
$ mkdir -p /share/sequences
$ mv /share/input/*.fast[aq] /share/sequences
```

2.4.3 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/input
$ immunedb_metadata --use-filenames
```

Note: This command expects the files to end in .fasta for FASTA, .fastq for FASTQ, or .tsv for AIRR.

This creates a `metadata.tsv` file in `/share/input` in Docker which is linked to `$HOME/immunedb_share/input` on the host.

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

Editing the Metadata Sheet

On the host open the metadata 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).

2.4.4 Pipeline Steps

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 /share/configs
```

Then we'll identify or import the sequences. For this process the germline genes must be specified. The germlines are provided FASTA files in the Docker image at `/root/germlines`.

Note: You can use your own germline files if you desire so long as they are IMGT gapped.

For this segment we'll assume human B-cell heavy chains, but the process is the same for any dataset. Depending on if you want to use IgBLAST input (recommended) or the built-in annotation method the command will be one of the following:

Option 1: Importing from IgBLAST output (recommended):

Listing 13: Run in Docker

```
$ immunedb_import /share/configs/example_db.json airr \
    /root/germlines/igblast/human/IGHV.gapped.fasta \
    /root/germlines/igblast/human/IGHJ.gapped.fasta \
    /share/input
```

Option 2: Using anchoring method:

Listing 14: Run in Docker

```
$ immunedb_identify /share/configs/my_db.json \
      /root/germlines/anchor/human/IGHV.gapped.fasta \
      /root/germlines/anchor/human/IGHJ.gapped.fasta \
      /share/input
```

After importing or identifying sequences, continue running the pipeline from here:

Listing 15: Run in Docker

```
$ immunedb_collapse /share/configs/my_db.json
```

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

Listing 16: Run in Docker

```
$ immunedb_clones /share/configs/my_db.json cluster
```

For T-cells we recommend:

Listing 17: Run in Docker

```
$ immunedb_clones /share/configs/my_db.json cluster --min-similarity 1
```

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

Listing 18: Run in Docker

```
$ immunedb_clones /share/configs/my_db.json cluster --gene IGHV
$ immunedb_clones /share/configs/my_db.json cluster --gene TCRB \
      --min-similarity 1
```

The last required step is to generate aggregate statistics:

Listing 19: Run in Docker

```
 $ immunedb_clone_stats /share/configs/my_db.json
 $ immunedb_sample_stats /share/configs/my_db.json
```

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 or at *Clone Trees (Optional)*.

Listing 20: Run in Docker

```
 $  immunedb_clone_trees /share/configs/my_db.json --min-seq-copies 2
```

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

Listing 21: Run in Docker

```
 $ immunedb_clone_pressure /share/configs/my_db.json \
      /apps/baseline/Baseline_Main.r
```

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

2.4.5 Analyzing Your Data

After all the above steps are complete, you should have a fully populated database, ready for analysis via *Exporting Data to Files*, *Querying with SQL*, and the *Python API*.

2.5 Modifying the Database

Databases can be modified in various ways using the `immunedb_modify` command.

2.5.1 Appending New Data

Adding new samples to a database is simply running the steps in *Running the Pipeline on Your Data* just on the new FASTA/FASTQ or AIRR files. Effort has been made to reduce the amount of information that needs to be recomputed when samples are added. However, after new samples are added all affected subjects will be entirely re-collapsed and clones will be recalculated.

2.5.2 Changing Metadata

Metadata specified when initially populating ImmuneDB via importing or identification can be updated in two steps. First, export the metadata currently in the database with:

```
$ immunedb_export PATH_TO_CONFIG samples --for-update
```

This will generate a `samples.tsv` file which can by modified. Headers and values can be changed, deleted, or added.

Note: Note that changing the subject of any sample will require steps after and including `immunedb_collapse` to be re-run.

After modifying the metadata, update the database with:

```
$ immunedb_modify PATH_TO_CONFIG update-metadata samples.tsv
```

2.5.3 Combining Samples

Warning: You cannot collapse samples from multiple subjects. If that functionality is desired, first modify the metadata to set the subject for each sample to be the same with `update-metadata`, and then run `combine-samples`.

One assumption ImmuneDB makes is that each sample is a *biological replicate* in that no one cell has its BCR/TCR sequence in more than one sample. If you have *technical replicates*, multiple independent sequencing runs of the same same biological replicate, they should be combined into one ImmuneDB-sample each. To do so, add a metadata field to the database as described in *Changing Metadata* where all technical replicates from the same biological replicate have the same value.

For example, if we have the following samples where each sample has two technical replicates:

sample	subject
biorep1_techrep1	S1
biorep1_techrep2	S1
biorep2_techrep1	S1
biorep2_techrep2	S1

You would update the metadata to be:

sample	subject	collapse
biorep1_techrep1	S1	first_sample
biorep1_techrep2	S1	first_sample
biorep2_techrep1	S1	second_sample
biorep2_techrep2	S1	second_sample

And then run:

```
$ immunedb_modify PATH_TO_CONFIG combine-samples collapse
```

This will result in the four replicates being collapsed into two, using the `collapse` field as the new name for each:

sample	subject
first_sample	S1
second_sample	S1

Note the header `collapse` can have any value you want so long as it's passed to `immunedb_modify`. Further, the values in that column can be arbitrary but will be used as the new name of the samples after collapsing.

2.5.4 Deleting Samples

The following command can be used to delete samples by ID:

```
$ immunedb_modify PATH_TO_CONFIG delete-samples [sample_ids]
```

Note that deleting samples will require the subject to be re-analyzed by running all pipeline steps after and including `immunedb_collapse`.

2.6 Running in the Background

After you have populated your ImmuneDB database(s), you may want to leave the frontend web service running in the background. To do so, you can start ImmuneDB in detached mode with the following:

```
$ docker run -v $HOME/immunedb_share:/share \
    -p 8080:8080 -e IMMUNEDB_DAEMON=1 -d=true \
    arosenfeld/immunedb:v0.29.9
```

If you want to stop the process in the future, get its process ID with

```
$ docker ps
```

And then run:

```
$ docker stop ID
```

2.7 Exporting Data to Files

You can use the `immunedb_export` command to export your data in a variety of formats.

2.7.1 Exporting Samples

To export samples statistics run the command:

```
$ immunedb_export PATH_TO_CONFIG samples
```

After completion, a TSV file `samples.tsv` will be written with the following headers, one line per sample:

Field	Description
`id`	Unique numeric sample identifier
`name`	Name given to the sample
`subject`	Subject from which the sample originated
`input_sequences`	Reads input into ImmuneDB
`identified`	Reads successfully annotated
`in_frame`	Reads in-frame
`stops`	Reads with stop codons
`functional`	Functional reads (in-frame and no stop codons)
`avg_clone_cdr3_num_nts`	Average clonal CDR3 length in nucleotides
`avg_clone_v_identity`	Average clonal V-region identity
`clones`	Total number of clones

2.7.2 Exporting Clones

In it's most basic form, the command to export clones is:

```
$ immunedb_export PATH_TO_CONFIG clones
```

This will generate one file per sample each with one line per clone having the fields below. Note that `intances`, `copies`, `avg_v_identity`, and `top_copy_seq` are for the clone in the context of that sample. That is, those fields may vary for the same clone in different samples.

Field	Description
clone_id	Database-wide unique clone identifier. This number can be used to track clones across samples.
subject	Subject in which the clone was found
v_gene	V-gene of the clone
j_gene	J-gene of the clone
functional	If the clone is in-frame and contains no stop in the consensus (T or F)
insertions	Insertions in the clone **(deprecated)**
deletions	Deletions in the clone **(deprecated)**
cdr3_nt	CDR3 nucleotide sequence
cdr3_num_nts	CDR3 nucleotide sequence length
cdr3_aa	CDR3 amino-acid sequence
uniques	Unique sequences in the clone **overall**
instances	Sequences instances in the clone in the associated sample
copies	Copies in the clone in the associated sample
germline	Clonal germline sequence
parent_id	Parent ID **(deprecated)**
avg_v_identity	Average V-gene identity to germline
top_copy_seq	Nucleotide sequence of top-copy sequence

The `--pool-on` parameter can be used to change how data is aggregated. By default it takes the value `sample` (as described above) but it also accepts, `subject`, or any custom metadata field(s).

For the purposes of illustration, assume we have samples with the associated metadata below.

sample	subject	tissue	subset
sample1	S1	blood	naive
sample2	S1	spleen	naive
sample3	S1	spleen	mature
sample4	S3	blood	native

Passing `--pool-on subject` will generate one file per subject with the clone information aggregated across all samples in that subject. Alternatively, passing `--pool-on tissue` will generate one file per subject/tissue combination. You can pass multiple metadata fields to the `--pool-on` parameter as well. For example `--pool-on tissue subset` will generate one file per subject/tissue/subset combination.

Two other common parameters are `--sample-ids` which restricts which samples to include in the export and `--format` which accepts `immunedb` (the default) or `vdjtools` for interoperability with the VDJtools suite.

2.7.3 Exporting Sequences

Sequences can be exported in Change-O and AIRR formats.

The basic command is:

```
$ immunedb_export PATH_TO_CONFIG sequences
```

This will generate one file per sample in Change-O format. To use AIRR format, specify `--format airr`. You can filter out sequences that were not assigned to a clone with the `--clones-only` flag.

2.7.4 Exporting Selection Pressure

If selection pressure was calculated with the `immunedb_clone_pressure` command, the results can be exported in TSV format, one row per clone/sample combination. Additionally, unless the `--filter samples` is passed, there will be one additional row per clone with a `All Samples` value for the sample which indicates the overall selection pressure on the clone.

For more information on interpreting the values see Uduman, et al, 2011 and Yaari, et al. 2012.

Field	Value
`clone_id`	Clone ID
`subject`	Subject to which the clone belongs
`sample`	Sample within which the selection pressure was calculated. If `All Samples` the overall selection pressure for the clone.
`threshold`	The threshold at which the selection pressure was calculated
`expected_REGION_TYPE`	The expected number of `TYPE` (`r` or `s`) mutations in `REGION` (`cdr` or `fwr`)
`observed_REGION_TYPE`	The observed number of `TYPE` (`r` or `s`) mutations in `REGION` (`cdr` or `fwr`)
`sigma_REGION`	The selection pressure in `REGION`
`sigma_REGION_clower`	The lower bound of the confidence interval of selection in `REGION`
`sigma_REGION_cupper`	The upper bound of the confidence interval of selection in `REGION`
`sigma_p_REGION`	The P-value of the selection in `REGION`

2.7.5 Exporting MySQL Data

The final method of exporting data is to dump the entire MySQL database to a file. This is meant to be a backup method rather than for downstream-analysis.

To backup run:

```
$ immunedb_admin backup PATH_TO_CONFIG BACKUP_PATH
```

To restore a backup run:

```
$ immunedb_admin restore PATH_TO_CONFIG BACKUP_PATH
```

2.8 Querying with SQL

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

2.8.1 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
```

2.9 Python API

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

2.9.1 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({
```

(continues on next page)

(continued from previous page)

```
    'host': '...',
    'database': '...',
    'username': '...',
    'password': '...',
})
```

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

2.9.2 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.

2.9.3 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
```

(continues on next page)

(continued from previous page)

```
IGHV4-39 1310
IGHV3-7 1345
IGHV3-30|3-30-5|3-33 1607
IGHV3-23|3-23D 1626
IGHV3-21 1878
```

2.9.4 Data Models

2.10 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.

2.11 Related Publications

Rosenfeld, A.M., Meng, W., Chen, D.Y., Zhang, B., Granot, T., Farber, D.L., Hershberg, U., and Prak, E.T.L., **Computational Evaluation of B-Cell Clone Sizes in Bulk Populations**. Frontiers in Immunology 9 (2018).

Vander Heiden, J.A., Marquez, S., Marthandan, N., Bukhari, S.A.C., Busse, C. E., Corrie, B., Hershberg, U., Kleinstein, S.H., Matsen, F.A., Ralph, D.K., Rosenfeld, A.M., Schramm, C.A., Christley, S., and Laserson, U., **AIRR Community Standardized Representations for Annotated Immune Repertoires**. Frontiers in Immunology 9 (2018).

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