BanzaiDB Documentation

Release 0.3.0

Mitchell Stanton-Cook

Jul 19, 2017

Contents

BanzaiDB is a tool for pairing Microbial Genomics Next Generation Sequencing (NGS) analysis with a NoSQL database. We use the RethinkDB NoSQL database.

BanzaiDB:

- initalises the NoSQL database and associated tables,
- populates the database with results of NGS experiments/analysis and,
- provides a set of query functions to wrangle with the data stored within the database.

CHAPTER 1

BanzaiDB documentation contents

User documentation

Learn how to initalise a database, populate the database and query it!

Installing BanzaiDB

There are a number of ways to install BanzaiDB once you have RethinkDB and the RethinkDB python driver installed. See the following for instructions on the installation of RethinkDB and the RethinkDB python driver. We recommend installing the python driver with the protobuf library that uses a C++ backend. There is a script in the root of the BanzaiDB git repository that can achieve this task.

Option 1a (with root/admin):

```
$ pip install BanzaiDB
```

Option 1b (as a standard user):

```
$ pip install BanzaiDB --user
```

You'll need to have git installed for the following alternative install options.

Option 2a (with root/admin & git):

```
$ cd ~/
$ git clone git://github.com/mscook/BanzaiDB.git
$ cd BanzaiDB
$ sudo python setup.py install
```

Option 2b (standard user & git) **replacing INSTALL/HERE with appropriate**:

```
$ cd ~/
$ git clone git://github.com/mscook/BanzaiDB.git
$ cd BanzaiDB
```

```
$ echo 'export PYTHONPATH=$PYTHONPATH:~/INSTALL/HERE/lib/python2.7/site-packages' >> ~
↪/.bashrc
$ echo 'export PATH=$PATH:~/INSTALL/HERE/bin' >> ~/.bashrc
$ source ~/.bashrc
$ python setup.py install --prefix=~/INSTALL/HERE/BanzaiDB/
```

If the install went correctly:

```
$ which BanzaiDB
/INSTALLED/HERE/bin/BanzaiDB
$ BanzaiDB -h
```

Please regularly check back to make sure you're running the most recent BanzaiDB version. You can upgrade like this:

If installed using option 1x:

```
$ pip install --upgrade BanzaiDB
$ # or
$ pip install --upgrade BanzaiDB --user
```

If installed using option 2x:

```
$ cd ~/BanzaiDB
$ git pull
$ sudo python setup.py install
$ or
$ cd ~/BanzaiDB
$ git pull
$ echo 'export PYTHONPATH=$PYTHONPATH:~/INSTALL/HERE/lib/python2.7/site-packages' >> ~
↪/.bashrc
$ echo 'export PATH=$PATH:~/INSTALL/HERE/bin' >> ~/.bashrc
$ source ~/.bashrc
$ python setup.py install --prefix=~/INSTALL/HERE/BanzaiDB/
```

Initialising your first BanzaiDB database

Initialising a BanzaiDB involves having BanzaiDB connect to your RethinkDB instance. Once connected BanzaiDB creates a database and initialises some default tables.

Prerequisites

Please ensure the following has been met: # You have installed BanzaiDB, RethinkDB & the RethinkDB python driver # You have a RethinkDB instance running

Telling BanzaiDB about your RethinkDB instance

The file ~/.BanzaiDB.cfg (~/ = /home/$USER/) is used to provide information to Banzai DB about your RethinkDB instance.

An example would be:

```
db_host = 127.0.0.1
db_name = my-cool-project
```

db_host: will always be 127.0.0.1 unless you have installed your RethinkDB instance on a remote server.

db_name: will typically reflect the short running title of the project

Getting study metadata into BanzaiDB

Here we will walk you through populating the metadata table into a freshly initalised BanzaiDB database.

> **Warning:** We expect that in the near future certain table headers will be required.

Absolute requirements

1. The study metadata **must be in** CSV or XLS[1] format,
2. The first row of the CSV or XLS **contains table headers**,
3. **There should be no missing data (empty cells)**. Please use *null* to represent missing cell data. Unknown may also be used. *null* should be used when it's absolutely known that the information does not exist, while *unknown* is used when it's actually unknown if the data may exist,
4. The strain identifier **must be provided**. The header for the strain identifier should be **StrainID**. If this is not the case please note the header as you'll need it when populating the table,

User considerations

1. We will guess the type (i.e string, number etc.). If some numbers are really meant to be strings please note the correct type for each header element as it will be needed when populating the table. See the following table of relationship between Python and JSON types:
2. If your table headers contain spaces they will be replaced by '_'

Python	JSON
dict	object
list, tuple	array
str, unicode	string
int, long, float	number
True	true
False	false
None	null

API documentation

Explore the available methods within BanzaiDB.

BanzaiDB

BanzaiDB package

[1] Metadata provided in XLS in converted to CSVKit.

Subpackages

BanzaiDB.fabfile package

Submodules

BanzaiDB.fabfile.variants module

Module contents

Submodules

BanzaiDB.banzaidb module

BanzaiDB.config module

BanzaiDB.converters module

BanzaiDB.core module

BanzaiDB.database module

BanzaiDB.errors module

BanzaiDB.fetch module

BanzaiDB.imaging module

BanzaiDB.misc module

BanzaiDB.parsers module

Module contents

Developer documentation

Learn how to contribute to the development of BanzaiDB.

BanzaiDB Developer HOWTO

In addition to what is described here, this document by Jeff Forcier and this talk from Carl Meyer provide wonderful footings for developing on/in open source projects.

Maintaining a consistent development environment

1) Ensure all development in performed within a virtualenv. A good way too bootstrap this is via virtualenv-burrito.

Execute the installation using:

```
$ curl -sL https://raw.githubusercontent.com/brainsik/virtualenv-burrito/master/
↪virtualenv-burrito.sh | $SHELL
```

2) Make a virtualenv called BanzaiDB:

```
$ mkvirtualenv BanzaiDB
```

3) Install autoenv:

```
$ git clone git://github.com/kennethreitz/autoenv.git ~/.autoenv
$ echo 'source ~/.autoenv/activate.sh' >> ~/.bashrc
```

Get the current code from GitHub

Something like this:

```
$ cd $PATH_WHERE_I_KEEP_MY_REPOS
$ git clone https://github.com/mscook/BanzaiDB.git
```

Install dependencies

Something like this:

```
$ cd BanzaiDB
$ # Assuming you installed autoenv -
$ # You'll want to say 'y' as this will activate the virtualenv each time you enter␣
↪the code directory
$ # Otherwise -
$ # workon BanzaiDB
$ pip install -r requirements.txt
$ pip install -r requirements-dev.txt
```

Familiarise yourself with the code

The BanzaiDB/BanzaiDB.py is the core module. It handles database insertion, deletion and updating.

For example:

```
$ ~/BanzaiDB/BanzaiDB$ python BanzaiDB.py -h
usage: BanzaiDB.py [-h] [-v] {init,populate,update,query} ...

BanzaiDB v 0.3.0 - Database for Banzai NGS pipeline tool (http://github.com/mscook/
↪BanzaiDB)

positional arguments:
  {init,populate,update,query}
                        Available commands:
    init                Initialise a DB
```

```
    populate            Populates a database with results of an experiment
    update              Updates a database with results from a new experiment
    query               List available or provide database query functions

optional arguments:
    -h, --help            show this help message and exit
    -v, --verbose         verbose output

Licence: ECL 2.0 by Mitchell Stanton-Cook <m.stantoncook@gmail.com>
```

Listing help on populate:

```
$ python BanzaiDB.py populate -h
usage: BanzaiDB.py populate [-h] {qc,mapping,assembly,ordering,annotation} run_path

positional arguments:
  {qc,mapping,assembly,ordering,annotation}
                    Populate the database with data from the given pipeline step
    run_path              Full path to a directory containing finished experiments␣
↪from a pipeline run

optional arguments:
    -h, --help            show this help message and exit
```

The fabfile (Fabric file) in fabfile directory contains query pre-written functions.

You can list them like this:

```
$ ~/BanzaiDB$ fab -l
Available commands:

    variants.get_variants_by_keyword           Return variants with a match in the␣
↪"Product" with the regular_expression
    variants.get_variants_in_range             Return all the variants in given␣
↪[start:end] range (inclusive of)
    variants.plot_variant_positions            Generate a PDF of SNP positions for␣
↪given strains using GenomeDiagram
    variants.strain_variant_stats              Print the number of variants and␣
↪variant classes for all strains
    variants.variant_hotspots                  Return the (default = 100) prevalent␣
↪variant positions
    variants.variant_positions_within_atleast  Return positions that have at least␣
↪this many variants
    variants.what_differentiates_strains       Provide variant positions that␣
↪differentiate two given sets of strains
```

Note: python BanzaiDB.py query simply calls the fabfile discussed above.

Development workflow

Use GitHub. You will have already cloned the BanzaiDB repo (if you followed instructions above). To make things easier, please fork (https://github.com/mscook/BanzaiDB/fork) and update your local copy to point to your fork.

Something like this:

```
$ # Assuming your fork is like this
$ # https://github.com/$YOUR_USERNAME/BanzaiDB/
```

```
$ vi .git/config
$ # Replace:
    $ # url = git@github.com:mscook/BanzaiDB.git
$ #  with:
$ # url = git@github.com:$YOUR_USERNAME/BanzaiDB.git
```

With this setup you will be able to push development changes to your fork and submit Pull Requests to the core BanzaiDB repo when you're happy.

Important Note: Upstream changes will not be synced to your fork by default. Please, before submitting a pull request please sync your fork with any upstream changes (specifically handle any merge conflicts). Info on syncing a fork can be found here.

Code style/testing/Continuous Integration

We try to make joining and/or modifying the BanzaiDB project simple.

General:

- As close to PEP8 as possible but I ain't no Saint. Just a long as it's clean and readable,
- Using standard lib UnitTest. There are convenience functions check_coverage.sh & tests/run_tests.sh respectively. We would prefer SMART test vs 100 % coverage.

In the master GitHub repository we use hooks that call:

- landscape.io (code QC)
- drone.io (continuous integration)
- ReadTheDocs (documentation building)

CHAPTER 2

Indices and tables

- genindex
- modindex
- search