g2gtools Documentation

Release 0.1.31

Kwangbom "KB" Choi, The Jackson Laboratory

February 17, 2017

Contents

Contents:

CHAPTER 1

g2gtools

g2gtools creates custom genomes by incorporating (phased) SNPs and indels into reference genome, extracts regions of interest, e.g., exons or transcripts, from custom genomes, and converts coordinates of files (bam, gtf, bed) between two genomes.

- Free software: GNU General Public License v3 (GPLv3)
- Documentation: https://g2gtools.readthedocs.org.

Reference

Manuscript in preparation (expected in 2017)

CHAPTER 2

Installation

Although g2gtools is available at PyPI (https://pypi.python.org/pypi/g2gtools/) for 'pip install' or 'easy_install', we highly recommend using Anaconda distribution of python (https://www.continuum.io/downloads) to install all the dependencies without issues. The most recent version of g2gtools is also available on Anaconda Cloud, so add the following channels if you have not already:

```
$ conda config --add channels r
$ conda config --add channels bioconda
```

To avoid conflicts among dependencies, we highly recommend using conda virtual environment:

```
$ conda create -n g2gtools jupyter
$ source activate g2gtools
```

Once g2gtools virtual environment is created and activated, your shell prompt will show '(g2gtools)' at the beginning to specify what virtual environment you are currently in. Now please type the following and install g2gtools:

```
(g2gtools) $ conda install -c kbchoi g2gtools
```

That's all! We note that you can go out from g2gtools virtual environment anytime once you are done using g2gtools:

```
(g2gtools) $ source deactivate
```

CHAPTER 3

Usage

To use g2gtools in command line

Note: We will assume you installed g2gtools in its own conda virtual environment. First of all, you have to "activate" the virtual environment by doing the following:

```
source activate g2gtools
```

To create a custom genome, we need the following information:

```
${REF}          reference genome in fasta format
${STRAIN}       strain name (usually a column name in the vcf file), e.g., CAST_EiJ
${VCF_INDELS}   vcf file for indels
${VCF_SNPS}     vcf file for snps
${GTF}          gene annotation file in gtf format
```

First of all, we need to create a chain file for mapping bases between two genomes. In this case, between reference and some other strain, like CAST_EiJ:

```
$ g2gtools vcf2chain -f ${REF} -i ${VCF_INDELS} -s ${STRAIN} -o ${STRAIN}/REF-to-${STRAIN}.chain
```

Then we patch snps onto reference genome:

```
$ g2gtools patch -i ${REF} -s ${STRAIN} -v ${VCF_SNPS} -o ${STRAIN}/${STRAIN}.patched.fa
```

We incorporate indels onto the snp-patched genome:

```
$ g2gtools transform -i ${STRAIN}/${STRAIN}.patched.fa -c ${STRAIN}/REF-to-${STRAIN}.chain -o ${STRA
```

Create custom gene annotation with respect to the new custom genome. We also create custom annotation database (so we can extract from custom genome) in the following steps:

```
$ g2gtools convert -c ${STRAIN}/REF-to-${STRAIN}.chain -i ${GTF} -f gtf -o ${STRAIN}/${STRAIN}.gtf
$ g2gtools gtf2db -i ${STRAIN}/${STRAIN}.gtf -o ${STRAIN}/${STRAIN}.gtf.db
```

We can also extract regions of interest from the custom genome. For example,:

```
$ g2gtools extract --transcripts -i ${STRAIN}/${STRAIN}.fa -db ${STRAIN}/${STRAIN}.gtf.db > ${STRAIN
$ g2gtools extract --genes -i ${STRAIN}/${STRAIN}.fa -db ${STRAIN}/${STRAIN}.gtf.db > ${STRAIN}/${STR
$ g2gtools extract --exons -i ${STRAIN}/${STRAIN}.fa -db ${STRAIN}/${STRAIN}.gtf.db > ${STRAIN}/${STR
```

If snp-patched genome is not of interest, we can remove it:

```
$ rm ${STRAIN}/${STRAIN}.patched.fa
$ rm ${STRAIN}/${STRAIN}.patched.fa.fai
```

To use g2gtools in a project

All the features are available as a python module. You can simply:

```
import g2gtools
```

CHAPTER 4

Credits

Development Lead

- Software design: **Kwangbom "KB" Choi, Ph. D.**, The Jackson Laboratory <kb.choi@jax.org>
- Software engineering: **Matthew Vincent**, The Jackson Laboratory <matt.vincent@jax.org>

Contributors

- Narayanan Raghupathy, The Jackson Laboratory <Narayanan.Raghupathy@jax.org>
- Anuj Srivastava, The Jackson Laboratory <Anuj.Srivastava@jax.org>

CHAPTER 5

History

0.1.29 (05/17/2016)

- Fixed parsing issue with GATK-generated VCF files

0.1.27 (05/04/2016)

- Uploaded the package to Anaconda Cloud
- Fixed Travis CI fail

0.1.23 (02/15/2016)

- Setup Travis CI for automated building

0.1.22 (02/15/2016)

- Updated documentation

0.1.20 (02/03/2016)

- Released to public with documentation at g2gtools.readthedocs.org

0.1.0 (02/09/2015)

- Started github repository

CHAPTER 6

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/churchill-lab/g2gtools/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "feature" is open to whoever wants to implement it.

Write Documentation

g2gtools could always use more documentation, whether as part of the official g2gtools docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/churchill-lab/g2gtools/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *g2gtools* for local development.

1. Fork the *g2gtools* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/g2gtools.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv g2gtools
$ cd g2gtools/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 g2gtools tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/churchill-lab/g2gtools/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_g2gtools
```

CHAPTER 7

Indices and tables

- genindex
- modindex