gbrs Documentation

Release 0.1.5

Kwangbom "KB" Choi, Ph.D.

February 14, 2017

Contents

1 GBRS **3**
- 1.1 Overview . . . 3
- 1.2 References . . . 3

2 Installation **5**

3 Usage **7**
- 3.1 To use GBRS in command line . . . 7
- 3.2 To use GBRS in a project . . . 8

4 Contributing **9**
- 4.1 Types of Contributions . . . 9
- 4.2 Get Started! . . . 10
- 4.3 Pull Request Guidelines . . . 10
- 4.4 Tips . . . 11

5 Credits **13**
- 5.1 Development Lead . . . 13
- 5.2 Contributors . . . 13

6 History **15**
- 6.1 0.1.5 (05/04/2016) . . . 15
- 6.2 0.1.4 (05/04/2016) . . . 15
- 6.3 0.1.2 (04/27/2016) . . . 15
- 6.4 0.1.1 (04/27/2016) . . . 15
- 6.5 0.1.0 (01/22/2016) . . . 15

7 Indices and tables **17**

Contents:

CHAPTER 1

GBRS

1.1 Overview

GBRS is a suite of tools for reconstructing genomes using RNA-Seq data from multiparent population and quantifying allele specific expression. Although we tested it with mouse models only, GBRS should work for any multiparent populations. If you are interested in using it for other multiparent population model, please contact Kwangbom "KB" Choi <kb.choi@jax.org> at The Jackson Laboratory. For the Diversity Outbred (DO https://www.jax.org/strain/009376), Collaborative Cross (CC) mice, or F1 hybrids of CC's (CCRIX), required data files are available at ftp://churchill-lab.jax.org/pub/software/GBRS/.

- Free software: GPLv3 license
- Documentation: https://gbrs.readthedocs.org.

1.2 References

Manuscript in preparation (Expected in 2016)

CHAPTER 2

Installation

Although GBRS is available at PyPI (https://pypi.python.org/pypi/gbrs/) 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. GBRS 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 gbrs jupyter
$ source activate gbrs
```

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

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

Then, make a folder to store GBRS specific data and set the following environment variable. You may want to add the second line (export command) to your shell rc file (e.g., .bashrc or .bash_profile). For example,

```
$ mkdir /home/<user>/gbrs
$ export GBRS_DATA=/home/<user>/gbrs
```

(For DO, CC, or CCRIX) Download data files to $GBRS_DATA folder:

```
$ cd $GBRS_DATA
$ wget ftp://churchill-lab.jax.org/pub/software/GBRS/R84-REL1505/\* .
$ tar xzf gbrs.hybridized.targets.bowtie-index.tar.gz
```

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

```
(gbrs) $ source deactivate
```

CHAPTER 3

Usage

3.1 To use GBRS in command line

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

```
source activate gbrs
```

The first step is to align our RNA-Seq reads against pooled transcriptome of all founder strains:

```
bowtie -q -a --best --strata --sam -v 3 ${GBRS_DIR}/bowtie.transcriptome ${FASTQ} \
    | samtools view -bS - > ${BAM_FILE}
```

Before quantifying multiway allele specificity, bam file should be converted into emase format:

```
gbrs bam2emase -i ${BAM_FILE} \
               -m ${GBRS_DATA}/ref.transcripts.info \
               -s ${COMMA_SEPARATED_LIST_OF_HAPLOTYPE_CODES} \
               -o ${EMASE_FILE}
```

We can compress EMASE format alignment incidence matrix:

```
gbrs compress -i ${EMASE_FILE} \
              -o ${COMPRESSED_EMASE_FILE}
```

If storage space is tight, you may want to delete ${BAM_FILE} or ${EMASE_FILE} at this point since ${COMPRESSED_EMASE_FILE} has all the information the following steps would need. If you want to merge emase format files in order to, for example, pool technical replicates, you run 'compress' once more listing files you want to merge with commas:

```
gbrs compress -i ${COMPRESSED_EMASE_FILE1},${COMPRESSED_EMASE_FILE2},... \
              -o ${MERGED_COMPRESSED_EMASE_FILE}
```

and use ${MERGED_COMPRESSED_EMASE_FILE} in the following steps. Now we are ready to quantify multiway allele specificity:

```
gbrs quantify -i ${COMPRESSED_EMASE_FILE} \
              -g ${GBRS_DATA}/ref.gene2transcripts.tsv \
              -L ${GBRS_DATA}/gbrs.hybridized.targets.info \
              -M 4 --report-alignment-counts
```

Then, we reconstruct the genome based upon gene-level TPM quantities (assuming the sample is a female from the 20th generation Diversity Outbred mice population)

```
gbrs reconstruct -e gbrs.quantified.multiway.genes.tpm \
                 -t ${GBRS_DATA}/tranprob.DO.G20.F.npz \
                 -x ${GBRS_DATA}/avecs.npz \
                 -g ${GBRS_DATA}/ref.gene_ids.ordered.npz
```

We can now quantify allele-specific expressions on diploid transcriptome:

```
gbrs quantify -i ${COMPRESSED_EMASE_FILE} \
              -G gbrs.reconstructed.genotypes.tsv \
              -g ${GBRS_DATA}/ref.gene2transcripts.tsv \
              -L ${GBRS_DATA}/gbrs.hybridized.targets.info \
              -M 4 --report-alignment-counts
```

Genotype probabilities are on a grid of genes. For eQTL mapping or plotting genome reconstruction, we may want to interpolate probability on a decently-spaced grid of the reference genome.:

```
gbrs interpolate -i gbrs.reconstructed.genoprobs.npz \
                 -g ${GBRS_DATA}/ref.genome_grid.64k.txt \
                 -p ${GBRS_DATA}/ref.gene_pos.ordered.npz \
                 -o gbrs.interpolated.genoprobs.npz
```

To plot a reconstructed genome:

```
gbrs plot -i gbrs.interpolated.genoprobs.npz \
          -o gbrs.plotted.genome.pdf \
          -n ${SAMPLE_ID}
```

3.2 To use GBRS in a project

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

```
import gbrs
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

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

4.1.2 Fix Bugs

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

4.1.3 Implement Features

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

4.1.4 Write Documentation

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

4.1.5 Submit Feedback

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

4.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/gbrs.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 gbrs
$ cd gbrs/
$ 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 gbrs 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.

4.3 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, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/kbchoi-jax/gbrs/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

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

CHAPTER 5

Credits

5.1 Development Lead

- Kwangbom "KB" Choi, Ph. D. <kb.choi@jax.org>, The Jackson Laboratory

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.1.5 (05/04/2016)

- `compress` subcommand also merges emase files
- `reconstruct` does not require `ref.gene_ids.ordered.npz` file anymore

6.2 0.1.4 (05/04/2016)

- Uploaded to Anaconda Cloud

6.3 0.1.2 (04/27/2016)

- Setup Travis CI automated building

6.4 0.1.1 (04/27/2016)

- Moved to github churchill-lab space
- First release onto PyPI

6.5 0.1.0 (01/22/2016)

- Started a private project on bitbucket.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search