qarrayrun Documentation

Release 1.1.0

Steve Davis

Apr 28, 2020

Contents

1 qarrayrun **3**
- 1.1 Features . . . 3
- 1.2 Citing qarrayrun . . . 3
- 1.3 License . . . 4

2 Installation **5**
- 2.1 Upgrading qarrayrun . . . 5
- 2.2 Uninstalling qarrayrun . . . 5

3 Usage **7**
- 3.1 Quick test . . . 7
- 3.2 Use with qsub . . . 7
- 3.3 Language . . . 8

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

5 Credits **13**
- 5.1 Development Lead . . . 13
- 5.2 CFSAN Bioinformatics Team . . . 13
- 5.3 External Contributors . . . 13

6 History **15**
- 6.1 1.1.0 (2020-04-28) . . . 15
- 6.2 1.0.1 (2018-12-11) . . . 15
- 6.3 1.0.0 (2018-11-29) . . . 15

7 Indices and tables **17**

Contents:

CHAPTER 1

qarrayrun

A helper tool for running array jobs on an HPC computational node.

The qarrayrun package was developed by the United States Food and Drug Administration, Center for Food Safety and Applied Nutrition.

This script executes a single slot of an array job on an HPC compute node. It is intended to be used with Sun Grid Engine, SLURM, or Torque job schedulers. It assumes every instance of the job array runs the same command but with different arguments. This script performs the work of looking-up the arguments in a text file and substituting those arguments into the command to be executed.

- Free software
- Documentation: https://qarrayrun.readthedocs.io
- Source Code: https://github.com/CFSAN-Biostatistics/qarrayrun
- PyPI Distribution: https://pypi.python.org/pypi/qarrayrun

1.1 Features

- Executes a single slot of an array job on an HPC compute node
- Simple parameter lookup language
- Supports execution in a subshell when needed

1.2 Citing qarrayrun

To cite qarrayrun, please reference the qarrayrun GitHub repository:

https://github.com/CFSAN-Biostatistics/qarrayrun

1.3 License

See the LICENSE file included in the qarrayrun distribution.

CHAPTER 2

Installation

At the command line:

```
$ pip install --user qarrayrun
```

Update your .bashrc file with the path to user-installed python packages:

```
export PATH=~/.local/bin:$PATH
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv qarrayrun
$ pip install qarrayrun
```

2.1 Upgrading qarrayrun

If you previously installed with pip, you can upgrade to the newest version from the command line:

```
$ pip install --user --upgrade qarrayrun
```

2.2 Uninstalling qarrayrun

If you installed with pip, you can uninstall from the command line:

```
$ pip uninstall qarrayrun
```

CHAPTER 3

Usage

3.1 Quick test

To try qarrayrun:

```
# Create a file with array job parameters, one line per task.
$ echo A B C > file
$ echo X Y Z >> file

# Store the task number in an envirnment variable.
# Normally, this is done automatically by the HPC job execution engine.
$ export SGE_TASK_ID=1

# Execute qarrayrun
$ qarrayrun SGE_TASK_ID file "echo {3} {2} {1}"
C B A

# Repeat for task 2
$ export SGE_TASK_ID=2
$ qarrayrun SGE_TASK_ID file "echo {3} {2} {1}"
Z Y X

# You can reuse any parameter more than once and create more complex command lines
$ qarrayrun SGE_TASK_ID file "echo {1}/{1}/{2}.{2}/{3}{3}.txt"
X/X/Y.Y/ZZ.txt
```

3.2 Use with qsub

To use qarrayrun with qsub, pipe the qarrayrun command line into qsub, and tell qsub how many tasks are in the job. The HPC job execution engine will automatically set the task number in an environment variable and execute qarrayrun on compute nodes.

Follow these examples:

```
# Sort some txt files, writing the sorted output to new files
ls *.txt > files.txt
echo 'qarrayrun SGE_TASK_ID files.txt sort -o sorted.{1} {1}' | qsub -t 1-$(cat files.
↪txt | wc -l) -cwd -j y -V -o log

# Your input file might have multiple columns, use {2} for the 2nd column
# Sort the largest files first
ls *.txt | xargs -n 1 wc -c | sort -n -r > files.txt
echo 'qarrayrun SGE_TASK_ID files.txt sort -o sorted.{2} {2}' | qsub -t 1-$(cat files.
↪txt | wc -l) -cwd -j y -V -o log

# Use the --shell option and quote your pipeline when you need shell redirection
# Remove blanks before sorting files
ls *.txt > files.txt
echo 'qarrayrun --shell SGE_TASK_ID files.txt "cat {1} | tr -d [:blank:] | sort >␣
↪sorted.{1}"' | qsub -t 1-$(cat files.txt | wc -l) -cwd -j y -V -o log
```

3.3 Language

As you can see from the examples above, qarrayrun has a very simple language for extracting parameters from a file and substituting the parameters into a command line.

Parameters in the array job parameter file are whitespace-separated. The parameters can have any meaning you want – numbers, strings, file names, directory names, etc.

The substitution language is just a number inside curly braces.

`{1}` is the first parameter found in the array job parameter file.

`{2}` is the second parameter found in the array job parameter file.

`{3}` is the third parameter found in the array job parameter file.

`{1000}` is the 1000th parameter found in the array job parameter file. There is no limit to the number of parameters per line in the array job parameter file.

3.3.1 Variable number of parameters

Sometimes, the lines in the array job parameter file may have varying numbers of parameters. If you specify a substitution placeholder with a parameter number higher than the number of parameters on the line, it will be silently ignored and replaced by an empty string. This lets you pass a variable number of parameters to other programs.

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/CFSAN-Biostatistics/qarrayrun/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

qarrayrun could always use more documentation, whether as part of the official qarrayrun 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/CFSAN-Biostatistics/qarrayrun/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 *qarrayrun* for local development.

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

```
$ git clone git@github.com:your_name_here/qarrayrun.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 qarrayrun
$ cd qarrayrun/
$ pip install sphinx_rtd_theme    # the documentation uses the ReadTheDocs theme
$ pip install pytest
$ 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 qarrayrun tests
$ pytest -v
$ tox
```

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

6. Update the documentation and review the changes locally with sphinx:

```
$ cd docs
$ sphinx-build -b html . ./_build
$ xdg-open _build/index.html
```

7. 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
```

8. 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 2.7, 3.4, 3.5, 3.6, and for PyPy.

4.4 Tips

To run a subset of tests:

```
$ pytest -v tests/test_qarrayrun.py
```

CHAPTER 5

Credits

5.1 Development Lead

- Steve Davis

5.2 CFSAN Bioinformatics Team

- Steve Davis

5.3 External Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 1.1.0 (2020-04-28)

- Update command line help to mention SLURM support.

6.2 1.0.1 (2018-12-11)

- Fix the text in usage example.

6.3 1.0.0 (2018-11-29)

- First public release.

CHAPTER 7

Indices and tables

- genindex
- search