BOSC 2020

Building better bioinformatics tools with batteries included

Peter Georgeson, Anna Syme, Jessica Chung, Michael Milton, Harriet Dashnow, Andrew Lonsdale, Clare Sloggett, **Bernard Pope** *

* Victorian Health and Medical Research Fellow
Melbourne Bioinformatics
The University of Melbourne, Australia
bjpope@unimelb.edu.au

Scientific software crisis?

Research Evaluation **24** (2015) pp. 454–470
Advance Access published on 27 July 2015
doi:10.1093/reseval/rvv014

Understanding the scientific software ecosystem and its impact: Current and future measures

James Howison[1,*], Ewa Deelman[2], Michael J. McLennan[3], Rafael Ferreira da Silva[2] and James D. Herbsleb[4]

How Do Scientists Develop and Use Scientific Software?

Jo Erskine Hannay
Dept. of Software Engineering
Simula Research Laboratory
Dept. of Informatics, Univ. of Oslo
johannay@simula.no

Carolyn MacLeod
Dept. of Computer Science
University of Toronto
cmacleod@cs.utoronto.ca

Janice Singer
Software Engineering Group
National Research Council of Canada

Hans Petter Langtangen
Center for Biomedical Computing

OPEN ACCESS Freely available online
PLoS COMPUTATIONAL BIOLOGY
Perspectives

Scientific Software Development Is Not an Oxymoron

Susan M. Baxter*, Steven W. Day, Jacquelyn S. Fetrow, Stepha

BRIEFINGS IN BIOINFORMATICS. VOL 12. NO 3. 288–300
Advance Access published on 28 January 2011
doi:10.1093/bib/bbq084

Case studies in reproducibility

Torsten Hothorn and Friedrich Leisch

Submitted: 3rd September 2010; Received (in revised form): 29th November 2010

focus developing scientific software

Dealing with Risk in Scientific Software Development

Rebecca Sanders, *Queen's University*
Diane Kelly, *Royal Military College of Canada*

ICCGI 2013 : The Eighth International Multi-Conference on Computing in the Global Information Technology

Lack of Software Engineering Practices in the Development of Bioinformatics Software

Dhawal Verma, Jon Gesell, Harvey Siy, Mansour Zand
Department of Computer Science
University of Nebraska at Omaha
Omaha, Nebraska 68182
Email: {dverma,jgesell,hsiy,zand}@unomaha.edu

POLICYFORUM
COMPUTATIONAL SCIENCE

Troubling Trends in Scientific Software Use

"Blind trust" is dangerous when choosing software to support research.

Lucas N. Joppa,[1*] Greg McInerny,[12] Richard Harper,[1] Lara Salido,[3] Kenji Takeda,[1] Kenton O'Hara,[1] David Gavaghan,[2] Stephen Emmott[1]

Toward effective software solutions for big biology

To the Editor:
Leading scientists tell us that the problem of large data and data integration, referred to as 'big data', is acute and hurting research. Recently, Snijder *et al.*[1] suggested a culture change in which scientists would aim

development that need to be resolved. Biologists are not formally trained for software engineering, so much of the bioinformatics software available today has been developed by PhD biologists in relative isolation on the back of funded experimental

News Feature | Published: 13 October 2010

Computational science: ...Error

Zeeya Merali

Nature **467**, 775–777(2010) | Cite this article

...why scientific programming does not compute.

Help is at hand!

22nd Conference on Software Engineering Education and Training

Software Engineering Education for Bioinformatics

Medha Umarji, Carolyn Seaman, A. Gunes Koru
Department of Information Systems, University of Maryland Baltimore County Baltimore, MD 21250 USA
{medha1, gkoru, cseaman}@umbc.edu

Hongfang Liu
Department of Biostatistics Bioinformatics and Biomathematics Georgetown University Medical Center Washington, DC 20007 USA
hl224@georgetown.edu

OPEN ACCESS Freely available online — PLOS | BIOLOGY

Community Page

Best Practices for Scientific Computing

Greg Wilson[1]*, D. A. Aruliah[2], C. Titus Brown[3], Neil P. Chue Hong[4], Matt Davis[5], Steven H. D. Haddock[7], Kathryn D. Huff[8], Ian M. Mitchell[9], Mark D. Plumbley[10], Ethan P. White[12], Paul Wilson[13]

frontiers in GENETICS — OPINION ARTICLE published: 02 July 2014 doi: 10.3389/fgene.2014.00199

On best practices in the development of bioinformatics software

Felipe da Veiga Leprevost[1,2]*, Valmir C. Barbosa[3], Eduardo L. Francisco[2], Yasset Perez-Riverol[4] and

and Protein Engineering, Carlos Chagas Institute - Fiocruz, Curitiba, Brazil
uritiba, Brazil
Computer Science Program, COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
pean Molecular Biology Laboratory, European Bioinformatics Institute, Cambridge, UK
prevost.com.br

Biophys Rev (2015) 7:343–352
DOI 10.1007/s12551-015-0177-3

REVIEW

How to test bioinformatics software?

Amir Hossein Kamali[1,2] · Eleni Giannoulatou[1,3] · Tsong Yueh Chen[4] · Michael A. Charleston[5] · Alistair L. McEwan[2] · Joshua W. K. Ho[1,3]

PLOS | COMPUTATIONAL BIOLOGY

EDITORIAL

Ten Simple Rules for Taking Advantage of Git and GitHub

Yasset Perez-Riverol[1]*, Laurent Gatto[2], Rui Wang[1], Timo Sachsenberg[3], Julian Uszkoreit[4], Felipe da Veiga Leprevost[5], Christian Fufezan[6], Tobias Ternent[1], Stephen J. Eglen[7], Daniel S. Katz[8], Tom J. Pollard[9], Alexander Konovalov[10], Robert M. Flight[11], Kai Blin[12], Juan Antonio Vizcaíno[1]*

Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software

PLOS | COMPUTATIONAL BIOLOGY

EDITORIAL

Ten simple rules for biologists learning to program

Maureen A. Carey[1], Jason A. Papin[2]*

1 Department of Microbiology, Immunology, and Cancer Biology, University of Virginia School of Medicine, Charlottesville, Virginia, United States of America, 2 Department of Biomedical Engineering, University of Virginia, Charlottesville, Virginia, United States of America

PLOS | COMPUTATIONAL BIOLOGY

EDITORIAL

Ten simple rules for documenting scientific software

Benjamin D. Lee*

School of Engineering and Applied Sciences, Harvard University, Cambridge, Massachusetts, United States of America

COMPUTATIONAL BIOLOGY

EDITORIAL

Ten simple rules for making research software more robust

Morgan Taschuk[1]*, Greg Wilson[2]

1 Genome Sequence Informatics, Ontario Institute for Cancer Research, Toronto, Ontario, Canad
2 Software Carpentry Foundation, Austin, Texas, United States of America

Seemann *GigaScience* 2013, **2**:15
http://www.gigasciencejournal.com/content/2/1/15

(GIGA)n SCIENCE

COMMENTARY — Open Access

Ten recommendations for creating usable bioinformatics command line software

Torsten Seemann[1,2]

OXFORD

Briefings in Bioinformatics, 19(4), 2018, 693–699

doi: 10.1093/bib/bbw134
Advance Access Publication Date: 14 January 2017
Paper

Top considerations for creating bioinformatics software documentation

Mehran Karimzadeh and Michael M. Hoffman

PLOS | COMPUTATIONAL BIOLOGY

EDITORIAL

Ten Simple Rules for Developing Usable Software in Computational Biology

Markus List[1]*, Peter Ebert[1,2]*, Felipe Albrecht[1,2]

1 Computational Biology and Applied Algorithmics, Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany, 2 Graduate School of Computer Science, Saarland Informatics Campus, Saarbrücken, Germany

Help is at hand!

There are <u>lots</u> of recommendations in the literature about how to address this problem.

But they are spread over many papers, and only partially overlap.

Advice is useful, but action is better.

What is Bionitio?

- A **tool** for starting new bioinformatics software projects following recommended best practices
- Supports 12 different programming languages

What is Bionitio?

- One command starts a new bioinformatic project:

```
bionitio-boot.sh -n skynet -i python
```

What is Bionitio?

- One command starts a new bioinformatic project:

```
bionitio-boot.sh -n skynet -i python
```

new project name

What is Bionitio?

- One command starts a new bioinformatic project:

```
bionitio-boot.sh -n skynet -i python
```

programming language

What is Bionitio?

- One command starts a new bioinformatic project:

```
bionitio-boot.sh -n skynet -i python
```

programming language

choices: C, C++, C# , Clojure, Java, Javascript, Haskell, Perl 5, Python 3, R, Ruby, Rust

What is Bionitio?

- The result is a functional software artefact:

1. working example of best practices
2. template for making new tools

What does the new project do?

```
$ skynet -h
usage: skynet [-h] [--minlen N] [--version] [--log LOG_FILE]
               [FASTA_FILE [FASTA_FILE ...]]

Read one or more FASTA files, compute simple stats for each file

positional arguments:
  FASTA_FILE      Input FASTA files

optional arguments:
  -h, --help      show this help message and exit
  --minlen N      Minimum length sequence to include in stats (default 0)
  --version       show program's version number and exit
  --log LOG_FILE  record program progress in LOG_FILE
```

What does the new project do?

```
$ skynet file1.fasta file2.fasta

FILENAME      NUMSEQ TOTAL  MIN  AVG  MAX

file1.fasta   1       237    237  237  237

file2.fasta   2       357    120  178  237
```

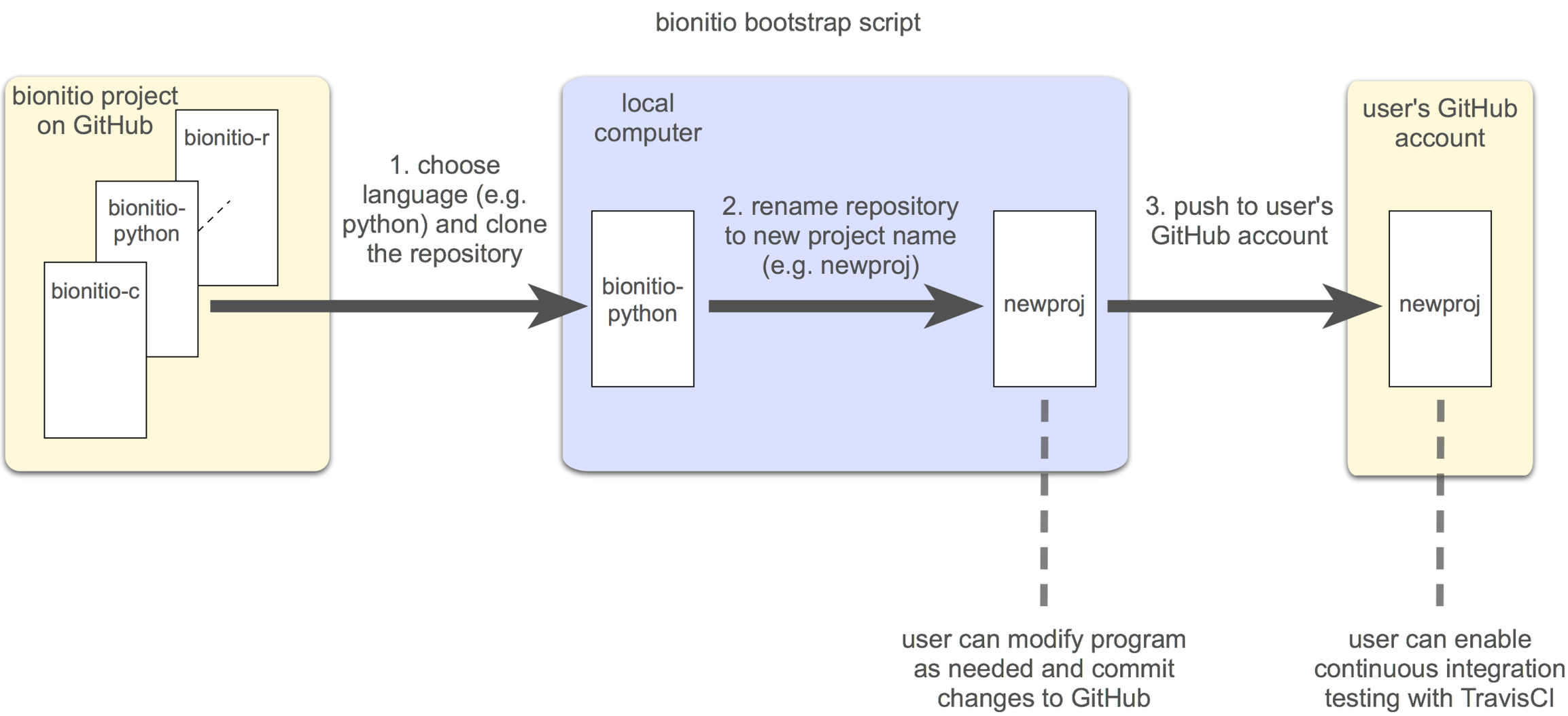
Key features built into the template

- Command-line argument parsing
- Progress logging
- Defined exit status values
- Test suite
- Version number
- Software packaging and a Docker container
- Standard open source license
- Documentation
- Revision control with Git and (optionally) GitHub
- Wrapper for the Common Workflow Language

Create a new project

- `bionitio-boot.sh` creates new projects
- It can be run from:
  - Docker
  - from GitHub, via curl, or downloaded manually

Create a new project



Create a new project

```
$ URL=https://git.io/bionitio-boot

$ curl -sSfL $URL | bash -s -- -i python -n skynet
```

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Git repository (contents abbreviated for the sake of this slide)

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Files for continuous integration testing with Travis CI

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Support for a Docker container

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

A standard open source license. Defaults to MIT, but you can choose a different option with the -c flag.

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

User documentation explaining how to install and use the program.

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Testing scripts and data

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Python-specific installation scripts and data.

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Source code

Contents of new project directory

```
$ tree -a skynet
skynet
├── .git
│   ├── COMMIT_EDITMSG
│   ├── HEAD
│   ... abbreviated ...
├── .travis
│   ├── install-dependencies.sh
│   └── unit-test.sh
├── .travis.yml
├── Dockerfile
├── LICENSE
├── README.md
├── functional_tests
│   ├── skynet-test.sh
│   └── test_data
│       ├── empty_file
│       ├── empty_file.expected
│       ├── no_header
│       ├── one_sequence.fasta
│       ├── one_sequence.fasta.expected
│       ├── single_greater_than.fasta
│       ├── two_sequence.fasta
│       ├── two_sequence.fasta.expected
│       ├── two_sequence.fasta.minlen_1000.expected
│       ├── two_sequence.fasta.minlen_200.expected
│       └── two_sequence.fasta.minlen_200.stdin.expected
├── requirements-dev.txt
├── setup.py
├── skynet
│   ├── .gitignore
│   ├── __init__.py
│   ├── skynet.py
│   └── skynet_test.py
└── skynet.cwl
```

Support for the Common Workflow Language

Automatically create a GitHub remote

```
$ URL=https://git.io/bionitio-boot

$ curl -sSfL $URL | bash -s -- -i python -n skynet \
    -g cyberdyne \
    -a 'Miles Bennett Dyson' \
    -e 'miles@cyberdyne.com'
```

Automatically create a GitHub remote

Same as before

```
$ URL=https://git.io/bionitio-boot

$ curl -sSfL $URL | bash -s -- -i python -n skynet \
    -g cyberdyne \
    -a 'Miles Bennett Dyson' \
    -e 'miles@cyberdyne.com'
```

Automatically create a GitHub remote

```
$ URL=https://git.io/bionitio-boot

$ curl -sSfL $URL | bash -s -- -i python -n skynet \
    -g cyberdyne \
    -a 'Miles Bennett Dyson' \
    -e 'miles@cyberdyne.com'
```

GitHub username, author name, email address

Automatically create a GitHub remote

bjpop / skynet

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki | Security | Insights | Settings

No description, website, or topics provided. Edit

Manage topics

1 commit | 1 branch | 0 packages | 0 releases | 0 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

BIONITIO Initial commit of skynet; starting from bionitio (python) — Latest commit 9362c08 2 days ago

.travis	Initial commit of skynet; starting from bionitio (python)	2 days ago
functional_tests	Initial commit of skynet; starting from bionitio (python)	2 days ago
skynet	Initial commit of skynet; starting from bionitio (python)	2 days ago
.travis.yml	Initial commit of skynet; starting from bionitio (python)	2 days ago
Dockerfile	Initial commit of skynet; starting from bionitio (python)	2 days ago
LICENSE	Initial commit of skynet; starting from bionitio (python)	2 days ago
README.md	Initial commit of skynet; starting from bionitio (python)	2 days ago
requirements-dev.txt	Initial commit of skynet; starting from bionitio (python)	2 days ago
setup.py	Initial commit of skynet; starting from bionitio (python)	2 days ago
skynet.cwl	Initial commit of skynet; starting from bionitio (python)	2 days ago

README.md

build unknown

Overview

This program reads one or more input FASTA files. For each file it computes a variety of statistics, and then prints a summary of the statistics as output.

In the examples below, `$` indicates the command line prompt.

Licence

This program is released as open source software under the terms of MIT License.

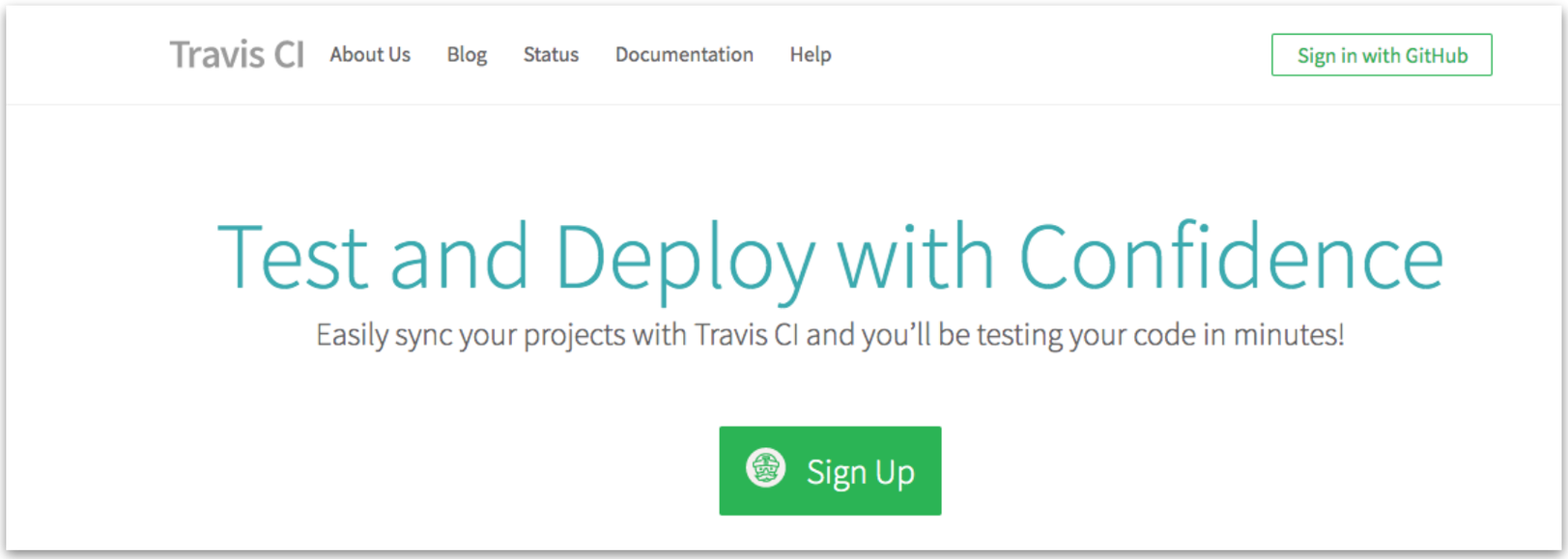
Installing

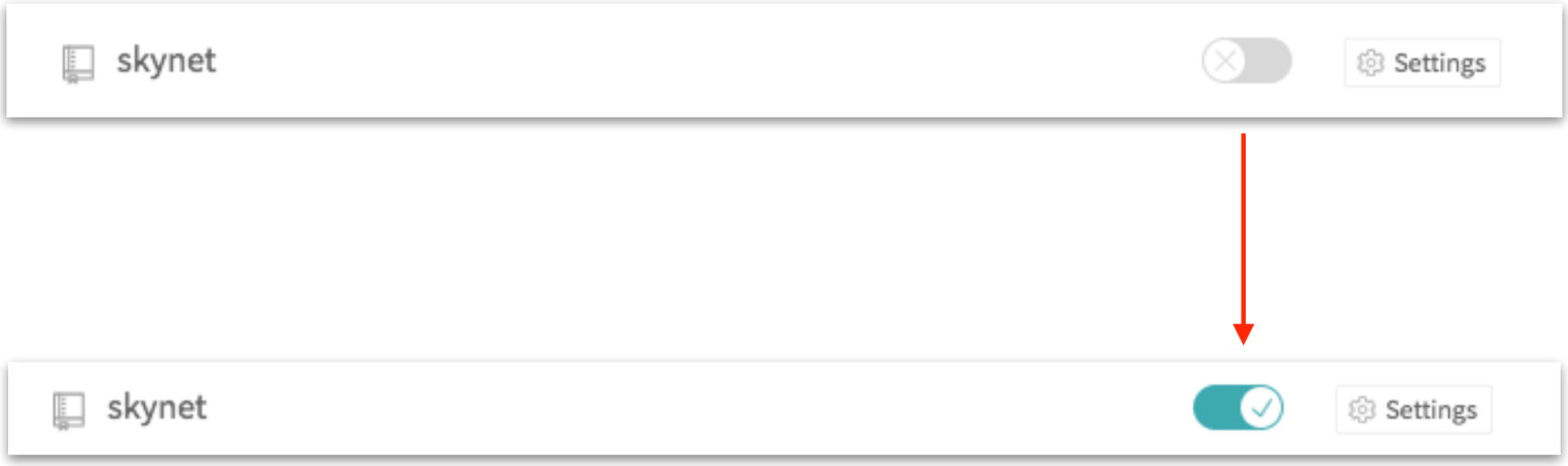
You can install skynet directly from the source code or build and run it from within Docker container.

license is easy to find

README.md provides user documentation

Integration with Travis CI





Integration with Travis CI

bjpop / skynet build passing

Current Branches Build History Pull Requests More options

master Trivial change #1 passed Restart build

Commit 62940ff
Compare 72fd5fb..62940ff
Branch master

Ran for 1 min 49 sec
3 minutes ago

Bernie Pope

Python: 3.5
AMD64

Modify the program to suit your own needs

- Once a new project has been created we expect users to modify it to suit their own purposes.
- This might involve rewriting large parts of the code, documentation and test suite.
- However, this can be done incrementally.
- The important point is that they are starting from a working project that already has batteries included.
- It is much easier *and faster* to modify an existing project than to start from scratch.

Conclusion

- Bionitio takes a pragmatic approach to solving the scientific software crisis.
- We want to help bioinformaticians to develop good habits early on, and to use them all the time - even for small scripts.
- Bionitio both *illustrates* good practices and makes them *easy to use*.

Conclusion

- We have also found that Bionitio is a good tool for training beginner and intermediate bioinformatics software developers

More information


OXFORD

(GIGA)n SCIENCE

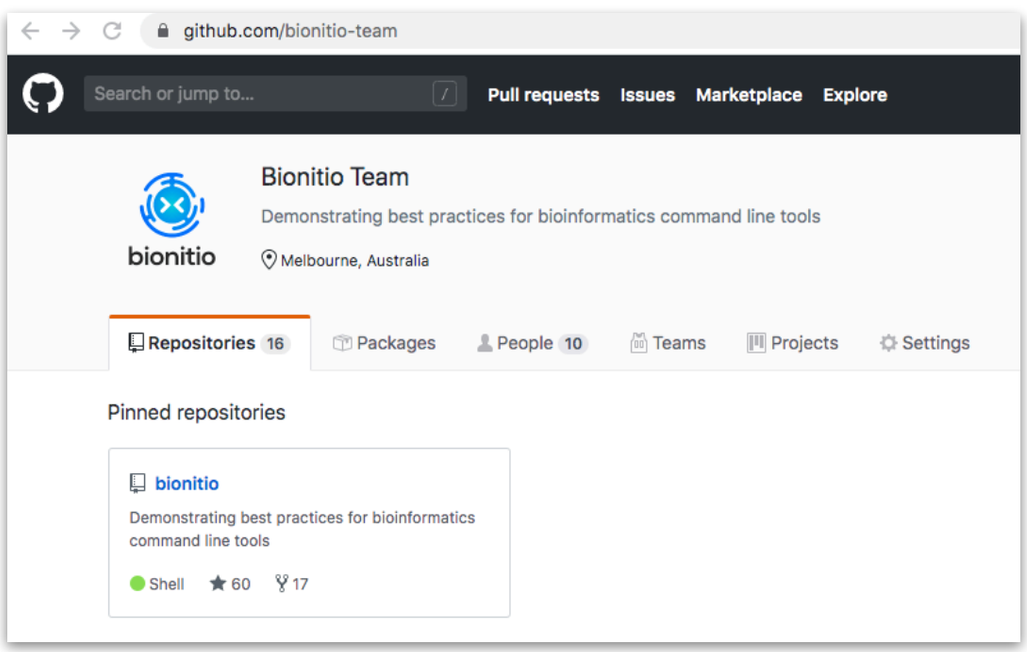
GigaScience, 8, 2019, 1–10

doi: 10.1093/gigascience/giz109
Technical Note

TECHNICAL NOTE

Bionitio: demonstrating and facilitating best practices for bioinformatics command-line software

Peter Georgeson[1,2,†], Anna Syme[1,3,†], Clare Sloggett[1], Jessica Chung[1], Harriet Dashnow[4,5], Michael Milton[1,6], Andrew Lonsdale[4,7], David Powell[8], Torsten Seemann[1,9] and Bernard Pope[1,2,10,*]



- https://github.com/bionitio-team/bionitio

Acknowledgements

Bionitio authors

- Peter Georgeson
- Anna Syme
- Clare Sloggett
- Jessica Chung
- Harriet Dashnow
- Michael Milton
- Andrew Lonsdale
- David Powell
- Torsten Seemann

Melbourne Bioinformatics

Department of Health and Human Services, Victoria