Multiscale Complex Genomics

Documentation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676556.

Tool integration - step by step

Step 1

1. **Wrap your application code**
   1.1. Set up your tool repository
   1.2. Create your tool and pipeline
   1.3. Test your tool
   1.4. Integrate with COMPSs
   1.5. Documentation

Step 2

2. **Register MuGVRE tool**
   2.1. Submit MuGVRE tool
   2.2. Document MuGVRE tool

Documentation

- We use ReadTheDocs.org (RTD) to host the documentation.
- RTD uses Sphinx to generate the documentation

```
pip install sphinx
```

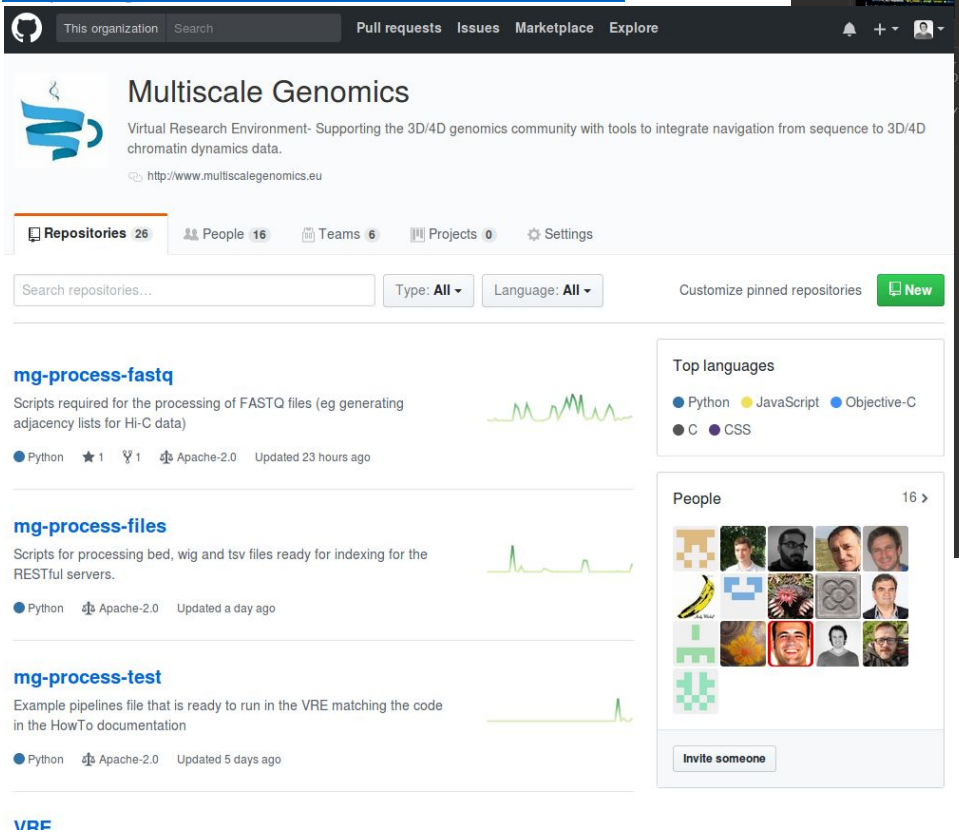
- In the docs directory this has been set up already making it easier to test building your documentation.

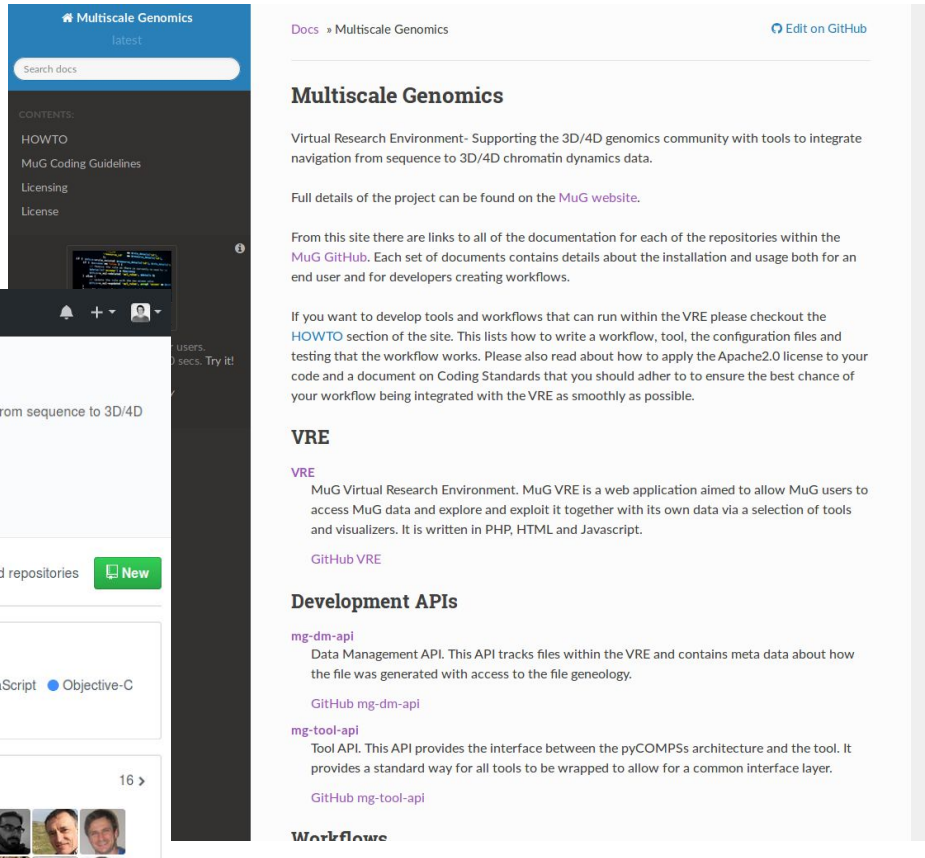
```
cd docs
make html
```

Documentation - GitHub to RTD

MuG is an organisation on GitHub

https://github.com/Multiscale-Genomics





All documentation is pushed to ReadTheDocs

https://multiscale-genomics.readthedocs.io/

What Should be documented

- All Classes (Pipelines and Tools)
- All methods (Pipelines and Tools)
- Installation procedures
  - Including required software and how to install it
- How test data was generated
- Architectural Design Record (ADR)

Class and Method Documentation Blocks

```
"""
Function to run MACS2 for peak calling on aligned sequence files and
normalised against a provided background set of alignments.

Parameters
----------
name : str
       Name to be used to identify the files
bam_file : str
       Location of the aligned FASTQ files as a bam file
bai_file : str
       Location of the bam index file
narrowpeak : str
       Location of the output narrowpeak file
summits_bed : str
       Location of the output summits bed file
broadpeak : str
       Location of the output broadpeak file
gappedpeak : str
       Location of the output gappedpeak file

Returns
-------
narrowPeak : file
       BED6+4 file - ideal for transcription factor binding site
       identification
summitPeak : file
       BED4+1 file - Contains the peak summit locations for everypeak
broadPeak : file
       BED6+3 file - ideal for histone binding site identification
gappedPeak : file
       BED12+3 file - Contains a merged set of the broad and narrow peak
       files

Definitions defined for each of these files have come from the MACS2
documentation described in the docs at https://github.com/taoliu/MACS
"""
```

Template docs

```
docs/
    _static/
        style.css
    _templates/
        layout.html
    conf.py
    index.rst
    install.rst
    license.rst
    Makefile
    pipelines.rst
    requirements.txt
    tool.rst
```

conf.py

This defines the configuration parameters for Sphinx along with Mocking modules that are not required for the building of the documentation

- Uninstalled modules need to be mocked
- Project, author and copyright need updating
- Document page titles need updating to be module specific

Makefile

Update the SPHINXPROJ line to match the correct project name

index.rst

Defines and linkes all of the pages within the documentation

Also defines the indexes and search capabilities that are to be provided.

If you create subdirectories then there should be a new index.rst with matching toctree element.

Tool.rst (pipelines.rst)

Links to each of the tool (pipeline) classes. Each tool should have its own section.

At the build time all of the methods are listed and documentation from the documentation blocks are formatted and displayed

```
Tools
=====

.. automodule:: mg_process_macs2.tool

   MACS2
   -----
   .. autoclass:: mg_process_macs2.tool.macs2.Macs2
     :members:
```

Tool.rst (pipelines.rst)

MuG - MACS2 Pipelines

latest

Search docs

TABLE OF CONTENTS

Docs » Tools

Edit on GitHub

Tools

MACS2

class `mg_process_macs2.tool.macs2.Macs2`(*configuration=None*) [source]

Tool for peak calling for ChIP-seq data

static `get_macs2_params`(*params*) [source]

Function to handle to extraction of commandline parameters and formatting them for use in the aligner for BWA ALN

Parameters: **params** (*dict*) –

Returns: **list** – List of lists with each list is the parameter and the matching value

Return type: list

`macs2_peak_calling`(***kwargs*) [source]

Function to run MACS2 for peak calling on aligned sequence files and normalised against a provided background set of alignments.

Parameters:

- **name** (*str*) – Name to be used to identify the files
- **bam_file** (*str*) – Location of the aligned FASTQ files as a bam file
- **bai_file** (*str*) – Location of the bam index file
- **bam_file_bgd** (*str*) – Location of the aligned FASTQ files as a bam file representing background values for the cell
- **bai_file_bgd** (*str*) – Location of the background bam index file
- **narrowpeak** (*str*) – Location of the output narrowpeak file
- **summits_bed** (*str*) – Location of the output summits bed file
- **broadpeak** (*str*) – Location of the output broadpeak file
- **gappedpeak** (*str*) – Location of the output gappedpeak file
- **chromosome** (*str*) – If the tool is to be run over a single chromosome the matching chromosome name should be specified. If None then the whole bam file is analysed

Multiscale Complex Genomics

Installation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676556.

Preparing your module for pip installation

To make it easy for users to install your code Python uses pip for quick and simple installation

pip allows you to define all the python packages that are required by you tool

Installation is as easy as:

```
pip install -e .
pip install -r requirements.txt
```

pip installation - setup.py

```
from setuptools import setup, find_packages

setup(
    name='mg_process_macs2',
    packages=find_packages(),
    include_package_data=True,
    install_requires=[
        'pytest', 'pylint', 'pysam', 'macs2', 'ConfigParser'
    ],
    setup_requires=[
        'pytest-runner',
    ],
    tests_require=[
        'pytest',
    ],
)
```

pip installation - requirements.txt

If you have modules that are not available in pip then they should go in the requirements.txt file:

```
git+https://github.com/Multiscale-Genomics/mg-tool-api.git
git+https://github.com/Multiscale-Genomics/mg-common.git
```

Documenting the Installation procedure

Within the docs/install.rst file you should fully describe the installation procedure going from a base installation of Ubuntu linux to a fully functional pipeline.

This should include:

- All apt installation commands
- Commands for the retrieval of external software, and its compilation
- Commands for the installation of your python module.

Multiscale Complex Genomics

Linting and Testing

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676556.

Linting

- We use the PEP8 style guidelines to make sure that we are all developing code with a consistent format.
- PEP8 is enforced using pylint are part of the testing
- pylintrc and .flake8 files are provided to help with the development of code
- pylint can be integrated into many editors (eg Sublime) making sure that issues are flagged up sooner.
- MuG Coding Style Guidelines:
  - http://multiscale-genomics.readthedocs.io/en/latest/coding_standards.html

Testing

Tests should be written to ensure that the code behaves as expected.

This is important when future changes are made to the code to avoid breaking the functionality.

pytest is used for running all tests.

There should be:

- Tests for all tools and pipelines
- Test data (<module>/tests/data)
- Test config and input json files (<module>/tests/json)

Example Test

```
@pytest.mark.chipseq
def test_macs2():
    """
    Function to test MACS2
    """

    resource_path = os.path.join(os.path.dirname(__file__), "data/")

    input_files = {
        "bam": resource_path + "macs2.Human.DRR000150.22_aln_filtered.bam"
    }

    out_1 = resource_path + "macs2.Human.DRR000150.22_peaks.narrowPeak",
    out_2 = resource_path + "macs2.Human.DRR000150.22_peaks.summits.bed",
    out_3 = resource_path + "macs2.Human.DRR000150.22_peaks.broadPeak",
    out_4 = resource_path + "macs2.Human.DRR000150.22_peaks.gappedPeak"

    output_files = {
        "narrow_peak": out_1, "summits": out_2, "broad_peak": out_3,
        "gapped_peak": out_4
    }

    metadata = {
        "bam": Metadata(
            "data_chipseq", "fastq", [], None, {'assembly': 'test'}),
    }

    macs_handle = Macs2({"macs_nomodel_param": True})
    macs_handle.run(input_files, metadata, output_files)

    assert os.path.isfile(out_1 is True
    assert os.path.getsize(out_1 > 0
    assert os.path.isfile(out_2) is True
    assert os.path.getsize(out_2) > 0
```

Running pytest

The pytest.ini file defines the location of the pytest scripts. Edit the testpaths to match:

```
[pytest]
testpaths = mg_process_macs2/tests
norecursedirs = data tmp* env docs *.egg_info
```

As simple as:

```
pytest
```

pytest output

```
$ pytest
============================= test session starts
==============================
platform linux2 -- Python 2.7.12, pytest-3.7.0, py-1.5.4, pluggy-0.7.1
rootdir: /.../mg-process-macs2, inifile: pytest.ini
plugins: mock-1.10.0
collected 2 items


mg_process_macs2/tests/test_macs2.py .
    [ 50%]
mg_process_macs2/tests/test_pipeline_macs2.py .
    [100%]


========================== 2 passed in 0.59 seconds
==========================
```

Multiscale Complex Genomics

Continuous Integration

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676556.

Continuous Integration

- This is the process of sharing code with a mainline process multiple times a day.
  - Running code tests
  - Building documentation
  - Checking code quality

Continuous Integration

- This is the process of sharing code with a mainline process multiple times a day.
  - Running code tests — TravisCI
  - Building documentation — RTD
  - Checking code quality — LandscapeIO / TravisCI

TravisCI

- Requires a .travis.yml file. An example file is already included in the mg-process-test repo
- The file include all software and procedures for installation, environments the code should be tested in (py2.7 and py3.6) and the scripts that contain the tests.
  - This will help when writing your installation documentation
- There are 3 tests:
  - pytest - Runs all the pytest scripts
  - docs - Runs Sphinx
  - pylint - Runs pylint over the python code

TravisCI

Multiscale-Genomics / mg-process-macs2

build passing

Current | Branches | Build History | Pull Requests | More options

master Updates to the docs to be more descriptive on RTB a

#69 passed

Restart build

Commit b29576b

Compare 1961ed7..b29576b

Branch master

Mark McDowall authored and committed

Ran for 7 min 17 sec

Total time 23 min 45 sec

about 3 hours ago

Build Jobs

# 69.1	Python: 2.7	TESTENV=docs	2 min 38 sec	
# 69.2	Python: 3.6	TESTENV=docs	4 min 29 sec	
# 69.3	Python: 2.7	TESTENV=code	4 min 18 sec	
# 69.4	Python: 3.6	TESTENV=code	5 min 35 sec	
# 69.5	Python: 2.7	TESTENV=pylint	2 min 27 sec	
# 69.6	Python: 3.6	TESTENV=pylint	4 min 18 sec	

Travis For Docs and Linting

Travis is running the docs and linting to reduce the number of places that require monitoring if there is a problem

ReadTheDocs.org

RTD provides a nicely formatted version of your documentation.

It does provide some testing of the code and can accept a customised requirements.txt file if you want to provide specific versions of modules that aren't mocked.

To hook up with RTD just log in with your GitHub account and select the relevant repository.

ReadTheDocs.org

Read the Docs

markmcdowall

Import a Project

Search Read the Docs

Projects

mg-common	8 builds	passing
mg-dm-api	307 builds	passing
mg-process-fastq	492 builds	passing
mg-process-files	86 builds	passing
mg-process-macs2	29 builds	passing
mg-process-test	33 builds	passing
mg-rest-3d	38 builds	passing
mg-rest-adjacency	39 builds	passing
mg-rest-dm	66 builds	passing
mg-rest-file	11 builds	passing
mg-rest-service	20 builds	passing
mg-rest-util	5 builds	passing
mg-tool-api	30 builds	passing
multiscale-genomics	65 builds	passing

Support Read the Docs

Read the Docs depends on users like you to help us keep the site sustainable.

We now offer Read the Docs Gold to allow folks to support us. Gold subscriptions allow us to keep the site running, and improving all the time. If you find value in Read the Docs, please consider getting a subscription.

Become a Gold user!

Learn More

Check out the documentation for Read the Docs. It contains lots of information about how to get the most out of RTD.

Landscape.io

Provides a nice interface over the pylint and flake8 reports.

Highlights errors, warnings and code that could be written in a more succinct way.

Provides a quality score over the code and can help identify and prioritise code that needs fixing.

Landscape.io

- Overview
- Pull Requests
- master #837
  - Summary
  - Changes
  - Errors 1
  - Smells 11
  - Style 95
  - All Messages 107
  - All Files
  - Requirements
  - Configuration
- Blog and Updates
- FAQ
- Documentation
- Email Us
- Twitter

98%

Excellent!

No change compared to #822
SCORE STAYED THE SAME

0 new problems.
0 problems were fixed.

Information

Check ran using **Python version 2**. See the configuration documentation for information on how to change this.

Check ran: 11 Jul 2018, 12:31 p.m.
Check took: 1 min 40 sec
Commit Hash: a1a99c6 (master)
Committer: web-flow
Committed: 11 Jul 2018, 12:31 p.m.

Message:
Merge pull request #23 from Multiscale-Genomics/fix_gem_location

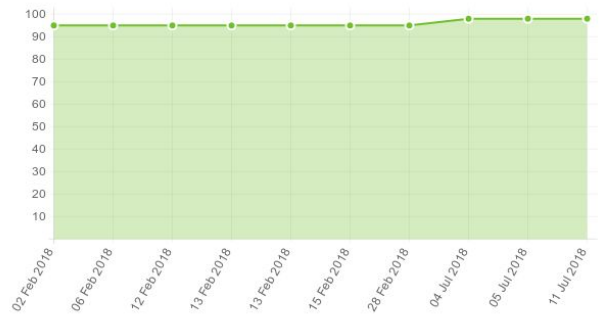
Fix gem location

Where?

Your check ran on machine: **solo**.

Previous: #822 Next

Trend



Worst Offenders

Score	Path
Excellent! (89%)	tool/tb_generate_tads.py
Excellent! (90%)	tool/tb_filter.py
Excellent! (90%)	process_hic.py
Excellent! (91%)	tool/tb_save_hdf5_matrix.py
Excellent! (91%)	tool/tb_parse_mapping.py
Excellent! (91%)	tool/tb_full_mapping.py
Excellent! (95%)	process_genome.py
Excellent! (95%)	scripts/ExtractRowsFromFASTQs.py
Excellent! (96%)	process_iNPS.py
Excellent! (96%)	process_mnaseseq.py

See all files

Badges

With all the checks you can then highlight this on your repository README.md pages and have them displayed in GitHub: