rnaseqflow Documentation

Release 0.2.0

Justin Palpant

March 30, 2016

Contents

CHAPTER 1

Overview

1.1 Introduction

rnaseqflow is an open-source Python package written to make preprocessing RNAseq files more convenient. It continues to be developed actively and is in the early stages of development. It provides the ability to automate and pipeline several operations that would otherwise be performed manually.

rnaseqflow currently supports the following operations:

- Discovery of files using recursive directory search with extension matching
- Merging files using intelligent filename pattern matching
- Trimming adapter sequences using fastq-mcf, either in single- or paired-end mode

These operations may be chained together, with one operation acting on the files found or created by the previous operation, to create a complete preprocessing workflow.

rnaseqflow is composed of two parts: a *command line interface* and a *Python package*. See the respective pages for information on how to use rnaseqflow.

rnaseqflow is constantly expanding and being developed, with more operations to be supported. To request support for a specific operation, request a feature, or report a bug, please create an issue at the GitHub repository below.

GitHub repository: jarvis-lab-rnaseq-flow

CircleCI build status:

TravisCI build status:

1.2 Installation

You can checkout the latest development version from GitHub with the following command:

```
git clone https://github.com/jpalpant/rnaseqflow
```

You can also install rnaseqflow with pip:

```
pip install -U rnaseqflow
```

1.3 Changelog

Version 0.2.1

- Continuous integration!
- This documentation!
- Creation of a paired-end fastq-mcf WorkflowStage

Version 0.2.0

- Complete overhaul of program structure
- Modular Workflow creation
- Addition of command line arguments
- Removal of PyQT dependency
- Addition of unit testing

Version 0.1.1

- Added timestamps to logging and additional parameters to fastq-mcf

Version 0.1.0

- Initial alpha release

1.4 Copyright and License

CHAPTER 2

Command Line Interface

While rnaseqflow is a Python package, most users will have need only of the command line entry point installed by setuptools, call with:

```
$ rnaseqflow
```

General help is available with the "–help" argument:

```
$ rnaseqflow --help
```

Which displays the following output:

```
usage: rnaseqflow [--help [{all,stages}]]
                  [--logging {debug,info,warning,error,critical}] [--version]
                  [--stages [STAGES [STAGES ...]]] [--root ROOT] [--ext EXT]
                  [--blocksize BLOCKSIZE] [--adapters ADAPTERS]
                  [--fastq FASTQ] [--fastq_args FASTQ_ARGS] [--quiet]

Preprocess RNAseq files.

optional arguments:
  --help [{all,stages}]
                        Display this help or detailed help on a topic
  --logging {debug,info,warning,error,critical}
                        Logging level (default: info)
  --version             show program's version number and exit
  --stages [STAGES [STAGES ...]]
                        Add stages
  --root ROOT           The root directory to be searched for RNAseq files
  --ext EXT             The file extension to search for
  --blocksize BLOCKSIZE
                        The size of the copy block (in kB) for merge
                        operations
  --adapters ADAPTERS   FastA adapters file to use
  --fastq FASTQ         Location of the fastq-mcf executable
  --fastq_args FASTQ_ARGS
                        Specify arguments to be passed to fastq-mcf
  --quiet               Silence extraneous console output
```

2.1 Sample Use Case

After an experiment, data may be delivered in the following folder structure:

```
/drive
   /experimentdata
      adapters.fasta
      /Sample_NIK1
         NIK1-2_TGACCA_L001_R1_001.fastq
         NIK1-2_TGACCA_L001_R1_002.fastq
         NIK1-2_TGACCA_L001_R2_001.fastq
         NIK1-2_TGACCA_L001_R2_002.fastq
      /Sample_NIK2
         NIK2-1_TGACCA_L001_R1_001.fastq
         NIK2-1_TGACCA_L001_R1_002.fastq
         NIK2-1_TGACCA_L001_R2_001.fastq
         NIK2-1_TGACCA_L001_R2_002.fastq
      ...
```

Where each _001, _002, etc. represent the nth part of a large fastq.

The standard workflow would be to rejoin these files by concatenating the relevant parts together, and then calling a program to trim adapter sequences from the trimmed files. It may also be desired that the files be trimmed based on read quality and read length.

Using rnaseqflow, the program would be executed with the following command:

```
$ rnaseqflow
Stages not given with --stages argument
The following WorkflowStages are available:
1: FindFiles - Find files recursively in a folder
2: MergeSplitFiles - Merge files by the identifying sequence and direction
3.0: FastQMCFTrimSolo - Trim adapter sequences from files using fastq-mcf one file at a time
3.1: FastQMCFTrimPairs - Trim adapter sequences from files using fastq-mcf in paired-end mode
Use "--help stages" for more details

Enter space separated stage specifiers (e.g. "1 2 3"): 1 2 3.1
No root directory provided with --root
Please enter a directory to use as the root folder: /drive/experimentdata
No file extension provided with --ext
Please provide a file extension (e.g. .fastq, .fastq.gz): .fastq
No blocksize for file copy operations given with --blocksize
Please provide a blocksize in kB (e.g. 1024): 1024
fasta adapter file not yet specified with --adapters
Please specify the .fasta adapter file location: /drive/experimentdata/adapters.fasta
No fastq arguments provided to --fastq_args
Provide an optional argument string for fastq here (e.g. "-q 30 -x 0.5"): "-q 30 -l 50 -x 0.5"
```

The program execution would begin by searching for fastq files within the root folder. With those files found it would create a folder /drive/experimentdata/merged into which it would put the concatenated files. It would take those concatenated files and pass them in forward-reverse pairs to fastq-mcf, putting the output in another folder /drive/experimentdata/trimmed.

Executing this command would require that fastq-mcf be installed and available on the system path.

It would also be possible to run this command without any interaction by using many command line arguments

```
$ rnaseqflow --stages 1 2 3.1 --root /drive/experimentdata
  --adapters /drive/experimentdata/adapters.fasta --ext .fastq --blocksize 1024
  --fastq_args "-q 30 -l 50 -x 0.5"
```

2.2 Arguments

–help Display information about the program.

```
$ rnaseqflow --help all
$ rnaseqflow --help
$ rnaseqflow --help stages
```

–help all is identical to **–help**. Details on **–help stages** are found below

–logging Followed by one of the arguments listed, to set the console log level

```
$ rnaseqflow --logging debug
```

–version Display's the version of the program

```
$ rnaseqflow --version
```

rnaseqflow 0.2.1

–stages Asks for one or more stage specifiers which determine the actual workflow to be carried out. Stage specifiers should space-delimited. No default. See *–help stages* for more information.

```
$rnaseqflow --stages 1 2
```

Finds the stages with the specifiers '1' and '2' (if they exist) These stages are then chained together and executed in sequence Any informatino needed by these stages not passed at the command line will be requested

–root Should be followed by a complete path to the directory in which all operations should be carried out. No default.

```
$rnaseqflow --root /Users/myname/Documents/rnaseqdatafolder
```

–ext Should be followed by a file extension (with the dot, e.g. '.fastq') which will be used for all operations. No default.

```
$rnaseqflow --ext .fastq
```

–blocksize Should be followed by an integer number of kilobytes; specifies the blocksize for use in file operations, such as file concatenation. No default.

```
$rnaseqflow --blocksize 1024
```

–adapters Should be followed by the complete path to the FASTA adapter file to be used by all stages. No default.

```
$rnaseqflow --adapters /Users/myname/Documents/rnaseqdatafolder/myadapters.fasta
```

–fastq_args Should be followed by a quoted string to pass directly to fastq-mcf, if fastq-mcf will be used. No default.

```
$rnaseqflow --fastq_args "-q 30 -l 50"
```

This will make sure that when fastq-mcf is invoked is is invoked with these arguments. Do not use this argument if fastq-mcf will not be used in your program.

–quiet Does not need to be followed by anything; if true, attempts to silence as much console output as possible. Does not affect output from logging, which is controlled with the **–logging** argument. Default is not quiet.

```
$rnaseqflow --quiet
```

If an argument is needed by any part of the workflow specified with the **–stages** argument and it is not provided, or if it has been provided incorrectly, the user will be asked to provide that argument before the program begins.

2.3 Stages

The **–help stages** argument will display information similar to the following

```
$rnaseqflow --help stages
The following WorkflowStages are available:
1: FindFiles
    Find files recursively in a folder

    Input:
        No input is required for this WorkflowStage
    Output:
        A flat set of file path strings
    Args used:
        * --root: the folder in which to start the search
        * --ext: the file extention to search for

2: MergeSplitFiles
    Merge files by the identifying sequence and direction

    Input:
        An iterable of file names to be grouped and merged
    Output:
        A flat set of merged filenames
    Args used:
       * --root: the folder where merged files will be placed
       * --ext: the file extention to be used for the output files
       * --blocksize: number of kilobytes to use as a copy block size

3.0: FastQMCFTrimSolo
    Trim adapter sequences from files using fastq-mcf one file at a time

    Input:
        A flat set of files to be passed into fastq-mcf file-by-file
    Output:
        A flat set of trimmed file names
    Args used:
       * --root: the folder where trimmed files will be placed
       * --adapters: the filepath of the fasta adapters file
       * --fastq: the location of the fastq-mcf executable
       * --fastq_args: a string of arguments to pass directly to fastq-mcf
       * --quiet: silence fastq-mcf's output if given


3.1: FastQMCFTrimPairs
    Trim adapter sequences from files using fastq-mcf in paired-end mode

    Input:
        A flat set of files to be passed into fastq-mcf in pairs
    Output:
        A flat set of trimmed file names
    Args used:
       * --root: the folder where trimmed files will be placed
       * --adapters: the filepath of the fasta adapters file
       * --fastq: the location of the fastq-mcf executable
       * --fastq_args: a string of arguments to pass directly to fastq-mcf
       * --quiet: silence fastq-mcf's output if given
```

In each case, the information about the stage is structured as follows:

- First, the stage's specifier or *spec*, followed by a colon and the stage name.
- Second, a short description of the stage's function
- Third, a description of what the stage produces as output, and what it must receive as input. This is useful when chaining stages together - make sure each stage's output is compatible with the input of the next stage
- Fourth, a list of arguments that the stage will require. They need not be provided at the command line, but they can be. If they are not, the user will be asked to provide them before the workflow begins to execute.

The stage's specifier is what is to be provided to the **–stages** argument when the executable is called. With the specifiers above one could call the following command

```
$rnaseqflow --stages 1 2 3.1
```

to create a workflow with will recursively find files with a given extension, merge any split files found by the first stage using the logic in the MergeSplitFiles stage, and trim adapters from the merged files using fastq-mcf passing in merged files one at a time.

Since no other arguments are provided, the user will be asked to provide all arguments needed by these stages, such as a file extension, a root directory, an adapter file, etc.

Note: Make sure to use the specifiers given by your console's output from **–help stages**, not the specifiers here. The specifiers in your installation may be different than in those used here. The **–help stages** argument attempts to intelligently find all possible available stages.

The stage name will be visible in logging statements from that stage.

Python API/Reference Docs

CHAPTER 3

Modules

rnaseqflow is composed of three modules:

3.1 __main__

Provides the entry point for the executable in the function **main()** and argument parsing in **opts()**

3.1.1 function main()

rnaseqflow.__main__.**main**()

This method is installed as a console script entry point by setuptools

It uses the command line arguments specified by opts() to generate a Workflow object and adds to it several WorkflowStages.

If the needed command line arguments are not passed, the user is asked to enter them.

The generated Workflow object is then executed with run()

3.1.2 function opts()

rnaseqflow.__main__.**opts**()

3.2 workflow

Provides several classes that are used to execute a series of preprocessing steps on RNAseq data

3.2.1 class Workflow

class rnaseqflow.workflow.**Workflow**

Execute a simple series of steps used to preprocess RNAseq files

append(*item*)

Add a WorkflowStage to the workflow

Parameters **item** (WorkflowStage) – the WorkflowStage to insert

insert (*idx*, *item*)
Insert a WorkflowStage into the workflow

Parameters

- **idx** (*int*) – list index for insertion
- **item** (WorkflowStage) – the WorkflowStage to insert

logger = <logging.Logger object>
log4j-style class logger

run ()
Allows the user to select a directory and processes all files within that directory

This function is the primary function of the Workflow class. All other functions are written as support for this function, at the moment

3.2.2 class WorkflowStage

class rnaseqflow.workflow.**WorkflowStage**
Interface for a stage of a Workflow

Subclasses must override the run method, which takes and verifies arbitrary input, processes it, and returns some output

They must also provide a .spec property which is a short string to be used to select the specific WorkflowStage from many options. These should not overlap, but at the moment no checking is done to see if they do.

logger = <logging.Logger object>
log4j-style class logger

classmethod longhelp ()
Create a long help text with full docstrings for each subclass of WorkflowStage

Subclasses are found using cliutils.all_subclasses

run (*stage_input*)
Attempt to process the provided input according to the rules of the subclass

Parameters **stage_input** (*object*) – an arbitrary input to be processed, usually a list of file names or file-like objects. The subclass must typecheck the input as necessary, and define what input it takes

Returns the results of the subclass's processing

classmethod shorthelp ()
Create a short help text with one line for each subclass of WorkflowStage

Subclasses are found using cliutils.all_subclasses

spec
Abstract class property, override with @classmethod

Used by the help method to specify available WorkflowItems

3.2.3 class FindFiles

class rnaseqflow.workflow.**FindFiles** (*args*)
Bases: *rnaseqflow.workflow.WorkflowStage*

Find files recursively in a folder

Input: No input is required for this WorkflowStage

Output: A flat set of file path strings

Args used:

- –root: the folder in which to start the search
- –ext: the file extention to search for

`logger` **= <logging.Logger object>**
log4j-style class-logger

`run` (*stage_input*)
Run the recursive file finding stage

Parameters **`stage_input`** (*`object, None`*) – not used, only for the interface

Returns A flat set of files found with the correct extension

Return type set(str)

`spec` **= '1'**
FindFiles uses '1' as its specifier

3.2.4 `class MergeSplitFiles`

class `rnaseqflow.workflow.`**`MergeSplitFiles`**(*args*)
Bases: *`rnaseqflow.workflow.WorkflowStage`*

Merge files by the identifying sequence and direction

Input: An iterable of file names to be grouped and merged

Output: A flat set of merged filenames

Args used:

- –root: the folder where merged files will be placed
- –ext: the file extention to be used for the output files
- –blocksize: number of kilobytes to use as a copy block size

static **`_get_direction_id`**(*filename*)
Gets the direction identifier from an RNAseq filename

A direction identifier is either R1 or R2, indicating a forward or a backwards read, respectively.

Parameters **`filename`** (*`str`*) – the base filename to be processed

Returns the file's direction ID, R1 or R2

Return type string

static **`_get_part_num`**(*filename*)
Returns an integer indicating the file part number of the selected RNAseq file

RNAseq files, due to their size, are split into many smaller files, each of which is given a three digit file part number (e.g. 001, 010). This method returns that part number as an integer.

This requires that there only be one sequence of three digits in the filename

Parameters **`filename`** (*`str`*) – the base filename to be processed

Returns the file's part number

Return type int

static `_get_sequence_id`(*filename*)

Gets the six-letter RNA sequence that identifies the RNAseq file

Returns a six character string that is the ID, or an empty string if no identifying sequence is found.

Parameters **`filename`** (`str`) – the base filename to be processed

Returns the file's sequence ID, six characters of ACTG

Return type string

`_organize_files`(*files*)

Organizes a list of paths by sequence_id, part number, and direction

Uses regular expressions to find the six-character sequence ID, the three character integer part number, and the direction (R1 or R2)

Parameters **`files`** (`iterable(str)`) – filenames to be organized

Returns organized files in a dictionary mapping the sequence ID and direction to the files that have that ID, sorted in ascending part number

Return type dict(tuple:list)

`logger` = **<logging.Logger object>**

log4j-style class-logger

`run`(*stage_input*)

Run the merge files operation

Creates a directory merged under the root directory and fills it with files concatenated from individual parts of large RNAseq data files

Files are grouped and ordered by searching the file basename for a sequence identifier like AACTAG, a direction like R1, and a part number formatted 001

Parameters **`stage_input`** (`iterable(str)`) – file names to be organized and merged

Returns a set of organized files

Return type set(str)

`spec` = **'2'**

MergeSplitFiles uses '2' as its specifier

3.2.5 `class FastQMCFTrimSolo`

class `rnaseqflow.workflow.`**`FastQMCFTrimSolo`**(*args*)

Bases: *`rnaseqflow.workflow.WorkflowStage`*

Trim adapter sequences from files using fastq-mcf one file at a time

Input: A flat set of files to be passed into fastq-mcf file-by-file

Output: A flat set of trimmed file names

Args used:

- –root: the folder where trimmed files will be placed
- –adapters: the filepath of the fasta adapters file
- –fastq: the location of the fastq-mcf executable

- –fastq_args: a string of arguments to pass directly to fastq-mcf
- –quiet: silence fastq-mcf's output if given

`logger` **= <logging.Logger object>**
log4j-style class-logger

`run` (*stage_input*)
Trim files one at a time using fastq-mcf

Parameters **`stage_input`** (*`iterable(str)`*) – filenames to be processed

Returns a set of filenames holding the processed files

Return type set(str)

`spec` **= '3.0'**
FastQMCFTrimSolo uses '3.0' as its specifier

3.2.6 `class FastQMCFTrimPairs`

class `rnaseqflow.workflow.`**`FastQMCFTrimPairs`**(*args*)
Bases: *`rnaseqflow.workflow.WorkflowStage`*

Trim adapter sequences from files using fastq-mcf in paired-end mode

Input: A flat set of files to be passed into fastq-mcf in pairs

Output: A flat set of trimmed file names

Args used:

- –root: the folder where trimmed files will be placed
- –adapters: the filepath of the fasta adapters file
- –fastq: the location of the fastq-mcf executable
- –fastq_args: a string of arguments to pass directly to fastq-mcf
- –quiet: silence fastq-mcf's output if given

`_find_file_pairs` (*files*)
Finds pairs of forward and backward read files

Parameters **`files`** (*`iterable(str)`*) – filenames to be paired and trimmed

Returns pairs (f1, f2) that are paired files, forward and backward If a file f1 does not have a mate, f2 will be None, and the file will be trimmed without a mate

Return type set(tuple(str, str))

static **`_get_sequence_id`** (*filename*)
Gets the six-letter RNA sequence that identifies the RNAseq file

Returns a six character string that is the ID, or an empty string if no identifying sequence is found.

Parameters **`filename`** (*`str`*) – the base filename to be processed

Returns the file's sequence ID, six characters of ACTG

Return type string

`logger` **= <logging.Logger object>**
log4j-style class-logger

run (*stage_input*)
Trim files one at a time using fastq-mcf

Parameters **stage_input** (*iterable(str)*) – filenames to be processed

Returns a set of filenames holding the processed files

Return type set(str)

spec = '3.1'
FastQMCFTrimPairs uses '3.1' as its specifier

3.3 cliutils

Provides a class that can intelligently ask the user to provide arguments if the required arguments were not provided at the command line

3.3.1 class ArgFiller

class rnaseqflow.cliutils.**ArgFiller** (*args*)
An interactive method of filling in arguments not given at runtime

Code completion taken from https://gist.github.com/iamatypeofwalrus/5637895

_fill_adapters ()
Fill in self.args.adapters with a valid file path

_fill_blocksize ()
Fill in self.args.blocksize with a valid integer (in kB)

_fill_ext ()
Fill in self.args.ext with a file type extension

_fill_fastq ()
Fill in the fastq-mcf executable self.args.fastq with a default 'fastq-mcf'

_fill_fastq_args ()
Fill in self.args.fastq_args

_fill_quiet ()
Fill in the quiet argument self.args.quiet with default False

_fill_root ()
Fill in self.args.root with a valid root directory

classmethod _get_directory_input (*message*)
Ask the user to enter a directory path in the command line

Parameters **message** (*str*) – a message to display with raw_input

classmethod _get_filepath_input (*message*)
Ask the user to enter a file path in the command line

Parameters **message** (*str*) – a message to display with raw_input

classmethod _get_integer_input (*message*)
Ask the user to enter an integer in the command line

Parameters **message** (*str*) – a message to display with raw_input

`fill`(*args_needed*)
Add the needed arguments to self.args if they are not there

Asks the user for input for each of the missing arguments

Parameters **`args_needed`** (*`list (str)`*) – a list of attributes to ensure self.args contains

`pathCompleter`(*text*, *state*)
This is the tab completer for systems paths.

`set_path_complete`(*enable*)
Enable or disable readline pathcompletion

Parameters **`enable`** (*`bool`*) – enable or disable completion

3.3.2 function `all_subclasses`

`rnaseqflow.cliutils.`**`all_subclasses`**(*cls*)
Recursively generate all subclasses of cls

Parameters **`cls`** – a python class

Returns all subclasses of cls

Return type list(cls)

3.3.3 function `trim`

`rnaseqflow.cliutils.`**`trim`**(*docstring*)
Trim a **PEP 0257** docstring

Code taken directly from **PEP 0257#handling-docstring-indentation**

Parameters **`docstring`** (*`str`*) – a Python docstring

Returns the first line of the docstring

Return type str

3.3.4 function `firstline`

`rnaseqflow.cliutils.`**`firstline`**(*docstring*)
Extract and return only the first line of a **PEP 0257** docstring

Parameters **`docstring`** (*`str`*) – a Python docstring

Returns the first line of the docstring

Return type str

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index