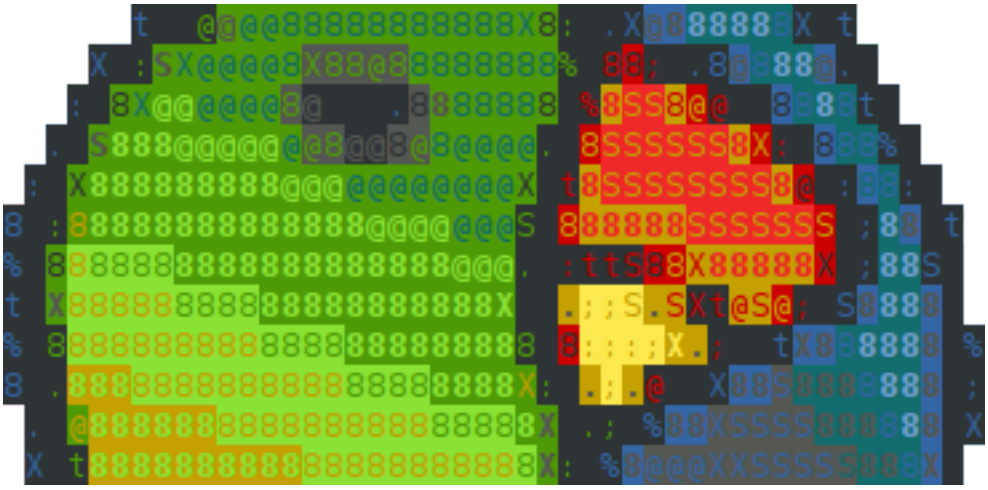

Moa Documentation

Release 0.10.12

Mark Fiers

January 04, 2012

CONTENTS

Lightweight, command line, workflows for bioinformatics

Moa aims to assist a bioinformatician to organize, document, share, execute and repeat workflows in a command line environment without losing flexibility, and, at all times giving the user full access to all aspects of the workflow (see also *Goals*).

NOTE: both the software and the manual are under development. Things might change.

CHAPTER

ONE

TABLE OF CONTENTS:

1.1 Goals

Moa aims to assist in achieving the following for a bioinformatics project:

- *Organized*:

  Moa facilitates project organization by allowing at only one *job* per directory, and, by having all configuration, templates, data, and intermediate data available as files in this directory structure.

- *Documented*:

  Moa provides the possibility to add a title, description and changelogs to each job.

- *Reproducible*

  By having all templates and configuration copied into a workflow - the workflow does never change (unless the user wants it to), even if templates in the repository change. Moreover, all templates are easy to find & inspect so it is always clear what happened.

- *Reusable & Shareable*:

  Moa provides reusable templates. New templates are easy to create, adapt and share. Workflows can be archived and reused with different data.

- *Flexible*:

  Moa provides a good number of hooks to insert custom code into a workflow, making that code part of the workflow. This ensures maximum flexibility.

1.2 Introduction

These days, generating massive amounts of data is an everyday element of biological research; and almost all projects have a bioinformatics components. Such embedded bioinformatics work commonly consists of chaining a number of 3rd party tools together, often with some data manipulation in between the steps. It is important to have such projects properly organized, particularly when a projects grows bigger.

There are many different ways to organize bioinformatics projects. Many bioinformaticians use the command line or tailor made scripts to organize and automate their work. This approach has obvious advantages, most importantly flexibility. Potential downsides to scripting are that a project easily becomes disorganized and untraceable unless measures are taken.

Moa aims to assist in organizing, automating and maintaining a command line bioinformatics project without loss of flexibility.

1.2.1 Example

The best way to understand how Moa can help you to achieve this is by an example. A Moa workflow consists of separate Moa jobs. A workflow is typically organised as a directory tree, where the structure of the tree reflects the structure of the project. So, Starting a Moa project starts with outlining a directory structure to contain the workflow:

```
$ mkdir test.project && cd test.project
$ mkdir 00.proteins

( copy or link some protein sequences into 00.proteins )

$ mkdir 10.blast
$ cd 10.blast
```

An important feature of Moa is that each separate analysis step is contained within a separate directory. Two Moa jobs never share a directory. This forces a Moa user to break a workflow down to atomic parts, which is typically beneficial to the organization and coherence of a workflow. The order of steps is easily ordered by prefixing directory names with a number. Note that these prefixes are not enforced by Moa; any alphabetical organization would work as well. Once a directory is created, a Moa job can be created:

```
$ moa new blast -t "demo run"
```

All interaction with Moa is done through a single command: *moa*. It is, at all times, possible to get help on the use of the *moa* command by invoking *moa –help*. The command above creates a *BLAST* job titled “demo run” in the current directory. All Moa related files are stored in a (hidden) sub-directory names *.moa* (have a look!). A Moa job consists, amongst others, of a configuration file and a number of template files. All template files are copied into the *.moa* directory. This ensures that a workflow remains the same over time, even if the templates are updated (*moa refresh* would update a template to the latest version).

Another topic in which Moa tries to help is by embedding (some) documentation. In the above command line the *-t* parameter sets a mandatory project title (a job won’t execute without a title).

Obviously, telling a Moa job to do a BLAST analysis is not enough, some variables will need to be set:

```
$ moa set db=/data/blast/db/nr
```

A few things could be noted here. Important is that you do not use spaces around the = sign. If you want to define a parameter with spaces, use quotes (*key=”value with spaces”*), and be aware of bash interpretation. A safe way of entering complex parameters is by running *moa set db* and Moa will query you the value.

Another point is that Moa does not give you a response. You can check the current job configuration using *moa show*, which would at this moment result in something resembling:

```
db     L /data/blast/db/nr
input  E (undefined)
jobid  L blast
title  L demo run
```

Note the variable *db* and *title*, which were set earlier. If you run *show -a*, more parameters will be revealed, amongst which is *program*. We will now set two more variables:

```
$ moa set program=blastp
$ moa set input=../00.proteins/*.fasta
```

The last statement defines the input files to blast. Once all is set you can actually run the BLAST analysis with:

```
$ moa run
```

Now Moa performs the BLAST analysis on the input files. The output can be found in the *out* subdirectory. As an extra, the Moa *blast* template generates a *blast_report* file with simple one line report for the best five hits of each query sequence. If you, for example, would like to check for the presence of dicer genes in your query set, you could *grep* this file:

```
$ grep -i dicer blast_report
```

Command line operation of data files can be very powerful, and this would be a typical operation for a command line bioinformatician. Moa lets you capture this and thus make it a part of the pipeline. Try:

```
$ moa set postcommand
```

and, at the prompt enter:

```
postcommand:
> grep -i dicer blast_report > dicer.out
```

If you now rerun *moa*, the BLAST job will not be repeated, but the *postcommand* will be executed and a *dicer.out* file will be generated. (note, there is also a *precommand*)

1.3 Installation

1.3.1 Prerequisites

Moa is developed and tested on Ubuntu and RHEL and is expected to operate without much problems on all modern Linux distributions. Moa has the following prerequisites (and a large number more for all templates). The version numbers are an indication, not strict prerequisites. Other, even older, versions might work.

- Gnu Make (3.81)
- Git (1.6). Necessary either to download the Moa software from github, or, to make use of the integrated version control.
- **Python (2.6). Moa is not tested with other** versions of Python
- **Bash (4.1.2). Many of the** embedded scripts expect the Bash shell.
- **Gnu Make Standard Library (GSML). A** set of standard routines for Gnu Make. GSML is distributed together with Moa.
- A number of support scripts & templates depend on Biopython. Consider installing it before starting to use Moa.

- *Python-dev*: the Python development package. A few prerequisites installed by easy_install try to compile C libraries, and need this. Although all of them have backup, python only, alteratives; from a performace perspective it is probably smart to have this installed:

  ```
  sudo apt-get install python-dev
  ```

- *python-yaml*: Again - this is not really necessary, but will improve performace:

  ```
  sudo apt-get install python-yaml
  ```

- Python easy_install is the preferred way to install Moa and a number of further prerequisites.

1.3.2 Installing Moa using easy_install

Easy:

```
sudo easy_install moa
```

The commandline will install moa and a number of other python libraries

There is a number of other prerequisites Moa requires the following modules to be installed:

- pyyaml
- Jinja2
- Ruffus
- gitpython
- Yaco
- fist
- **'unittest2 http://pypi.python.org/pypi/unittest2'_**
- **'lockfile http://pypi.python.org/pypi/lockfile'_**

These can be installed using install Moa:

```
easy_install-2.6 moa
```

Not part of the list of prerequisites are the following libraries, which you'll only need if you are planning to run the web interface:

- ElementTree
- Markdown

Note - these can be installed using easy_install:

```
$ sudo easy_install-2.6 ElementTree
$ sudo easy_install-2.6 Markdown
```

1.3.3 Bioinformatics tools

Each of the wrapped tools requires the tools to be present. Usually, Moa expects all tools to be present & executable on the system PATH. The standard Moa distribution comes with wrappers for:

- Blast

- BWA
- Bowtie
- Soap

and many more

1.3.4 Installation from source

Moa is hosted on and can be installed from github:

```
cd ~
git clone git://github.com/mfiers/Moa.git moa
```

1.3.5 Configuration

Configuration of Moa is simple, and can be done by sourcing the *moainit* script:

```
. ~/moa/bin/moainit
```

(Note the dot!, alternatively use: `source ~/moa/bin/moainit`)

It is probably a good idea to add this line to your `~/.bashrc` for future sessions.

Moa should now work, try *moa –help* or, for a more extensive test: *moa unittest*

If your default python version is NOT *python2.6* or *python2.7* there are a few options that you can pursue:

- change the hashbang of the *moa* script
- define an alias in your *~/.bashrc*: *alias moa='python2.6 moa'*
- create a symlink to python2.6 in your ~/bin directory and make sure that that is first in your path.

1.3.6 Installing the web interface

Note - this is a little experimental - you will need to experiment a little to get it working. Start with installing apache2.

Then - assuming that: * Your Moa work directory is under /home/moa/work * Your Moa is installed in /opt/moa Create a file in */etc/apache2/conf.d/moa.conf* with the following approximate contents:

```
Alias /moa/data /home/moa/work
<Directory /home/moa/work>
   Options +Indexes +FollowSymLinks
   Order allow,deny
   Allow from all

   SetEnv MOADATAROOT /home/moa/work
   SetEnv MOAWEBROOT /moa/data

   IndexOptions FoldersFirst SuppressRules HTMLTable IconHeight=24 SuppressHTMLPreamble

   HeaderName /moa/cgi/indexHeader.cgi
   ReadmeName /moa/html/indexFooter.html
```

```
</Directory>

ScriptAlias /moa/cgi/ /opt/moa/www/cgi/
<Directory /opt/moa/www/cgi/>
    AddType text/html .cgi
    Order allow,deny
    Allow from all
    SetEnv MOABASE /opt/moa
</Directory>

Alias /moa/html/ /opt/moa/www/html/
<Directory /opt/moa/www/html>
    Order allow,deny
    Allow from all
    Options +Indexes
</Directory>
```

You might want to check the shebang of */opt/moa/www/cgi/indexHeader.cgi* depending on your system configuration. Restart apache and it should work

1.4 Three core templates

Moa comes with a list of templates (see *templates*). The three most important, flexible templates of these that allow you to embed custom code (called *process*) in your project are:

simple:

> Simply executes *process* as a bash one-liner

map:

> Takes a set of in- and output files and executes the custom commands for each in- and output file (using the Jinja2 template language).

reduce:

> Takes a set of input files and a single output file and executes the custom commands with all input file, generating the output files.

Since *simple*, *map* and *reduce* have proven to be quite central to how Moa operates they come with their own shortcut commands (*moa simple*, *moa map* and *moa reduce*). These command query the user directly for the parameters instead of having to define this manually.

For example, a *simple* job:

```
$ mkdir simple_test && cd simple_test
$ moa simple -t 'Generate some files'
process:
> for x in `seq 1 5`; do touch test.$x; done
$ moa run
$ ls
test.1  test.2  test.3  test.4  test.5
```

Note that you can make your *process* as complicated as you like. Alternatively, you can write a script that you call from *process*.

A map job would work like this:

```
$ mkdir ../map_test && cd ../map_test
$ moa map -t 'Map some files'
process:
> echo {{ input }} ; echo {{ input }} > {{ output }}
input:
> ../simple_test/test.*
output:
> ./out.*
$ moa run
../simple_test/test.3
../simple_test/test.1
../simple_test/test.5
../simple_test/test.2
../simple_test/test.
Moa: Success executing "run" (<1 sec)
$ ls
out.1  out.2  out.3  out.4  out.5
$ cat out.1
../simple_test/test.1
```

Moa tracks which input file generates which outputfile. So, if you would like to repeat one of the jobs - you'll need to delete the output file & rerun *moa*:

```
$ rm out.3
$ moa run
../simple_test/test.3
Moa: Success executing "run" (<1 sec)
```

And a *reduce* example:

```
$ mkdir ../reduce_test && cd ../reduce_test
$ moa reduce -t 'Reduce some files'
process:
> echo {{ " ".join(input) }} >> {{ output }}
input:
> ../map_test/out.*
output:
> ./reduce_out
$ moa run
Moa: Success executing "run" (<1 sec)
$ ls
reduce_out
$ cat reduce_out
../map_test/out.1 ../map_test/out.3 ../map_test/out.4 ../map_test/out.5 ../map_test/out.
```

NOTE: both the software and the manual are under development. Expect things to change.

1.5 How to write a template

A MOA template is made up of a `.moa file` and a `.jinja2` (or `.mk`) file.

The `.moa` file mainly contains input-output file sets and parameter options used for the bash command(s). Some of these options have default values which the user can change while constructing the job.

The `.jinja2` file includes information to structure the command(s). It is written in jinja, which is a templating language for python and is simple to write and easy to understand.

These files are used by the backend, currently *ruffus*, that manages file set and parameter dependencies to make pipelines and render commands to the bash prompt. Initially, *GNU make* was the backend used. It is very powerful but some of its limitations and its complexity led to including *ruffus* as an option for the backend as well.

The easiest way to write a moa template is to edit an existing template to suit your requirements. This involves understanding the parts of an existing template.

The `bwa_aln` template is used as an example below. Just as a background, the *bwa aln* command takes a FASTQ file as input and aligns it to a reference genome that was previously indexed. The output is a .sai file with the alignments.

The `bwa_aln.moa` file has some main components:

- *Backend*

  ```
  backend: ruff
  ```

  This is 'ruff' which means that ruffus is used in the python script at a lower level to read the template .moa and .jinja2 file, and render the corresponding commands to the bash prompt.

- *Commands*

  ```
  commands:
    run:
      mode: map
      help:  run bwa aln
    clean:
      mode: simple
      help: Remove all job data, not the Moa job itself, note that this must be imple
  ```

  This indicates the function names that you will later define. In the example above, there are 2 commands- run and clean, so `moa run` or `moa clean` on the command prompt in the job directory would execute these functions.

- *Filesets*

  ```
  filesets:
    input:
      category: input
      extension: fq
      help: Fastq input files
      glob: '*'
      optional: false
      type: set
    output:
      category: output
      dir: .
      extension: sai
      glob: '{{ input_glob }}'
      source: input
      type: map
  ```

  Like the name, each filesets refer to a set of files in a single directory. The bwa_aln template shows 2 filesets: `input` and `output`.

  - *Category*: is essentially used to separate input from output.

  - *Extension*: refers to the type of file(s) required or generated.
  - *Glob*: searches for files with a specified pattern. Moa, by default (glob= *) automatically processes all files of the specified input extension in the directory specified. By specifying a glob, Moa will only process those files whose name pattern matches what is in the glob.
  - *Type*: refers to the data type of the fileset or parameter.

    A fileset can either be of `set` or `map` type. The type `set` refers to a simple set of files in a directory. The type `map` refers to a set of files that are linked to what their `source` value is. In the above code, the output fileset is mapped to the input fileset.
  - *Dir*: the directory of the output fileset is '.', which means that the output files will be placed in the current working directory.

- *Parameter category order*

```
parameter_category_order:
  - ''
  - input
  - system
  - advanced
```

- *Parameters*

```
mismatch_penalty:
  category: ''
  default: 3
  help: mismatch penalty
  optional: true
  type: integer
```

  They are the variables/options that specify a command.

  - *Category*:
  - *Default*: is the value that is used by default if not changed by the user.
  - *Optional*: specifies if it is necessary for the user to fill in a value for the variable. If `optional` is false, the user has to indicate a value for the parameter in order to execute the job.
  - *Type*: specifies the data type of the variable eg. integer, string, boolean.

- *Moa_id*

```
moa_id: bwa_aln
```

  is supposed to be the same as the filename. Ideally something descriptive (eg. bwa_aln). This is used to later link to the other template file.

The other template file is ''bwa_aln.jinja2" which is written in jinja, a templating language for python. *Note that the jinja2 file name is the same as the moa file name.*

Important features of the bwa_aln.jinja2 file are:

- The three hash's (###) specify the start of a function and are followed by the function name. In our bwa_aln example, we have defined 2 funtions: `run` and `clean`.

```
### run
```

- This defination is followed by a set of commands which you would want to be executed when you type `moa run` or `moa_clean` in the bwa_aln job directory. The commands in our example file look the same as what you would put in the command prompt but the values of the parameters are bought from the .moa file and hence it's value is replaced by the parameter name.

```
bwa aln {{db}}                                  \
    -n {{edit_dist_missing_prob}}               \
    .                                           \
    .                                           \
    .                                           \
    {{ input }}                                 \
    -f {{ output}}
```

- It is also possible to add if-else statements or other computing blocks in accordance with the design language.

```
{% if color_space %} -c {% endif %}
```

1.6 Command reference

1.6.1 moa !

Moa-fy the last (bash) command issued

Usage:

```
moa !
```

1.6.2 moa adhoc

Create an adhoc analysis

Usage:

```
moa adhoc -t "title" -- echo "do something"
```

1.6.3 moa archive

Archive a job,

1.6.4 moa blog

record a short note

Usage:

```
moa blog
```

1.6.5 moa cp

Copy a moa job

1.6.6 moa err

Returns stderr of the last moa run

1.6.7 moa files

Show an overview of the files for this job

1.6.8 moa gitadd

Add the current job to the git repository

1.6.9 moa gitlog

display a nicely formatted git log

1.6.10 moa gittag

Tag the current version

1.6.11 moa help

Display help for a template

1.6.12 moa kill

Kill a job

1.6.13 moa list

Print a list of all known templates

1.6.14 moa lock

Lock this job - prevent execution

1.6.15 moa log

Show the logs for this job

1.6.16 moa map

Create a “map” adhoc analysis

Usage:

```
moa map -t "title" -- echo "do something"
```

1.6.17 moa mv

Rename/renumber/move a job

1.6.18 moa new

Create a new Moa job

1.6.19 moa out

Returns stdout of the last moa run

1.6.20 moa pack

pack a moa tree

1.6.21 moa pause

Pause a job

1.6.22 moa postcommand

Run the postcommand

Usage:

```
moa postcommand
```

1.6.23 moa precommand

Run the precommand

Usage:

```
moa pprecommand
```

1.6.24 moa reduce

Create a “reduce” adhoc analysis

Usage:

```
moa reduce -t "title" -- echo "do something"
```

1.6.25 moa refresh

Reload the template

1.6.26 moa resume

Resume a job

1.6.27 moa set

Set, change or remove variables

Usage:

```
moa set [KEY] [KEY=VALUE]
```

1.6.28 moa show

Show configuration

Usage:

```
moa show
```

1.6.29 moa simple

Create a “simple” adhoc analysis

Usage:

```
moa simple -t "title" -- echo "do something"
```

1.6.30 moa status

Show the state of the current job

1.6.31 moa test

Test the currennt configuration

1.6.32 moa tree

display a directory tree

1.6.33 moa unittest

Run Moa unittests

1.6.34 moa unlock

Unlock this job

1.6.35 moa unpack

unpack a moa tree

1.6.36 moa unset

Remove a variable

Usage:

```
moa unset KEY
```

1.6.37 moa version

Print the moa version

1.7 Templates

Contents:

1.7.1 abyss_pe

Run Abysspe

Commands

clean Remove all job data

run Execute abysspe in paired-end mode

Filesets

fq_forward fastq input files directory - forward

fq_reverse fastq input files directory - reverse

type: *map*
source: *fq_forward*
category: *input*
optional: *True*
pattern: */*_2.fq*

output soap denovo output file

type: *single*
category: *output*
optional: *True*
pattern: *{}*

Parameters

joinpairs number of pairs needed to consider joining two contigs

type: *integer*
default: *10*
optional: *True*

kmer kmer size

type: *integer*
default: *31*
optional: *True*

threads no threads to use

type: *integer*
default: *3*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Mon, 21 Nov 2011 12:47:16

Modification date Mon, 21 Nov 2011 12:47:22

1.7.2 abyss_se

Run Abysspe

Commands

clean Remove all job data

run Execute abyss se

Filesets

input fastq input files directory

output soap denovo output file

type: *single*
category: *output*
optional: *True*
pattern: *{}*

Parameters

kmer kmer size

type: *integer*
default: *31*
optional: *True*

threads no threads to use

type: *integer*
default: *3*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Mon, 21 Nov 2011 12:47:16

Modification date Mon, 21 Nov 2011 12:47:22

1.7.3 adhoc

Execute an ad hoc analysis

The *adhoc* template assists in running one-liners - possibly on a set of input files

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

input Input files for adhoc

Parameters

mode

operation mode: *seq*, sequential: process the input files one by one; *par*, parallel: process the input files in parallel (use with *-j*); *all*: process all input files at once (use *$^* in *adhoc_process*) and *simple*: Ignore input files, just execute *adhoc_process* once.

type: *set*
default: *simple*
optional: *True*

name_sed A sed expression which can be used to derive the output file name for each input file (excluding the path). The sed expression is executed for each input file name, and the result is available as $t in the $(adhoc_process) statement. Make sure that you use single quotes when specifying this on the command line

type: *string*
default: *s/a/a/*
optional: *True*

output_dir Output subdirectory

type: *directory*
default: .
optional: *True*

process Command to execute for each input file. The path to the input file is available as $< and the output file as $t. (it is not mandatory to use both parameters, for example "cat $< > output" would concatenate all files into one big file

type: *string*
default: *echo "needs a sensbile command"*
optional: *True*

touch use touch files to track if input files have changed.

type: *set*
default: *T*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.4 bamextract

bamextract

Extract a region from a BAM file

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Extract a region from a BAM file

Filesets

bam BAM input

type: *single*
category: *input*
optional: *False*
pattern: *{}*

regions List with regions to extract (id seqid start stop)

type: *single*
category: *input*
optional: *True*
pattern: *{}*

Parameters

flank flanking region to extract

type: *integer*
default: *100*
optional: *{}*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.5 bartab

Bartab

BARTAB - a tool to process sff files

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run

Parameters

extra_parameters extra parameters to feed bartab

type: *string*
default: ‘‘
optional: *True*

forward_primer remove forward primer

type: *string*
default: ‘‘
optional: *True*

in input file for bartab

type: *file*
default: ‘‘
optional: *False*

map A file mapping barcodes to metadata

type: *file*
default: ‘‘
optional: *True*

min_length minimun acceptable sequence length

type: *integer*
default: *50*
optional: *True*

out base output name

type: *integer*
default: *bartab*
optional: *True*

qin Quality scores for the input fasta file

type: *file*
default: ‘‘
optional: *True*

reverse_primer remove reverse primer

type: *string*
default: ‘‘
optional: *True*

trim Trim barcode

type: *set*
default: *T*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.6 bdbb

Bidirectional best BLAST hit

Discover the bidirectional best blast hit between two sets of sequences

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run generate a list of bidirectional best blast hits between two databases of sequences

Filesets

input_a First multi fasta input set

type: *single*
category: *input*
optional: *False*
pattern: **/*.fasta*

input_b Second multi fasta input set

type: *single*
category: *input*
optional: *False*
pattern: **/*.fasta*

output List of bidirectional best blasts hits

type: *map*
source: *input_a*
category: *output*
optional: *True*
pattern: **/*.list*

Parameters

eval e value cutoff

type: *float*
default: *1e-10*
optional: *True*

extract Extract the identified sequences from the input fasta files

type: *boolean*
default: *False*
optional: *True*

nothreads Threads to run blast with with

type: *integer*
default: *4*
optional: *True*

protein Is this a protein set

type: *boolean*
default: *False*
optional: *True*

tblastx If this is a nucleotide set, use tblastx?? (otherwise use blastn)

type: *boolean*
default: *F*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date unknown

1.7.7 bfast_aln

Generate bam format alignments using bfast

Commands

clean Remove all job data, not the Moa job itself

run run bfast match, localalign, postprocess commands

Filesets

fa_input fasta input file

fq_input fastq input files

output_aln

type: *map*
source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.aln*

output_bam

type: *map*

source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

Parameters

algorithm_colour_space true -> colour space, false -> NT space

type: *boolean*
default: *False*
optional: *True*

avg_mism_qual Specifies the average mismatch quality

type: *integer*
default: *10*
optional: *True*

extra_params_localalign Any extra parameters for the localalign command

type: *string*
default: ‘‘
optional: *True*

extra_params_match Any extra parameters for the match command

type: *string*
default: ‘‘
optional: *True*

extra_params_postprocess Any extra parameters for the postprocess command

type: *string*
default: ‘‘
optional: *True*

min_mapping_qual Specifies to remove low mapping quality alignments

type: *integer*
default: *-2147483648*
optional: *True*

min_norm_score Specifies to remove low (alignment) scoring alignments

type: *integer*
default: *-2147483648*
optional: *True*

output_format 0 - BAF, 1 - SAM

type: *integer*
default: *1*
optional: *True*

paired_opp_strands Specifies that paired reads are on opposite strands

type: *boolean*
default: *False*
optional: *True*

pairing_std_dev Specifies the pairing distance standard deviation to examine when recuing

type: *float*
default: *2.0*
optional: *True*

print_params print program parameters

type: *boolean*
default: *False*
optional: *True*

thread_num Specifies the number of threads to use

type: *integer*

default: *1*
optional: *True*

timing_information specifies output timing information

type: *boolean*
default: *True*
optional: *True*

ungapped_aln Do ungapped local alignment

type: *boolean*
default: *False*
optional: *True*

ungapped_pairing_rescue Specifies that ungapped pairing rescue should be performed

type: *boolean*
default: *False*
optional: *True*

unpaired_reads True value specifies that pairing should not be performed

type: *boolean*
default: *False*
optional: *True*

usage_summary Display usage summary (help)

type: *boolean*
default: *False*
optional: *True*

which_strand 0 - consider both strands, 1 - forwards strand only, 2 - reverse strand only

type: *integer*
default: *0*

optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Feb 15 10:06:48 2011

Modification date unknown

1.7.8 bfast_db

Generate db index files for aligning reads with bfast

Commands

clean Remove all job data, not the Moa job itself

run run bfast fasta2brg and index commands

Filesets

fa_input fasta input file

Parameters

algorithm_colour_space true -> colour space, false -> NT space

type: *boolean*
default: *False*
optional: *True*

depth The depth of the splitting(d). The index will be split into 4^d parts.

type: *integer*
default: *0*
optional: *True*

extra_params Any extra parameters

type: *string*
default: ‘‘

optional: *True*

hash_width The hash width for the index (recommended from manual = 14)

type: *integer*
default: *{}*
optional: *False*

index_num Specifies this is the ith index you are creating

type: *integer*
default: *1*
optional: *True*

mask The mask or spaced seed to use.

type: *string*
default: *{}*
optional: *False*

print_params print program parameters

type: *boolean*
default: *False*
optional: *True*

thread_num Specifies the number of threads to use

type: *integer*
default: *1*
optional: *True*

timing_information specifies output timing information

type: *boolean*
default: *True*
optional: *True*

usage_summary Display usage summary (help)

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Feb 15 10:06:48 2011

Modification date unknown

1.7.9 blast

Basic Local Alignment Tool

Wraps BLAST [[Alt90]], probably the most popular similarity search tool in bioinformatics.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

report Generate a text BLAST report.

run Running BLAST takes an input directory, determines what sequences are present and executes BLAST on each of these. Moa BLAST is configured to create XML output (as opposed to the standard text based output) in the out directory. The output XML is subsequently converted to GFF3 by the custom blast2gff script (using BioPython). Additionally, a simple text report is created.

Filesets

db Blast database

type: *single*
category: *prerequisite*
optional: *False*
pattern: */*

input Directory with the input files for BLAST, in Fasta format

outgff

type: *map*

source: *input*
category: *output*
optional: *True*
pattern: *gff/*.gff*

output

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *out/*.out*

Parameters

eval e value cutoff

type: *float*
default: *1e-10*
optional: *True*

gff_blasthit (T,**F**) - export an extra blasthit feature to the created gff, grouping all hsp (match) features.

type: *set*
default: *F*
optional: *True*

gff_source source field to use in the gff

type: *string*
default: *BLAST*
optional: *True*

nohits number of hits to report

type: *integer*
default: *50*
optional: *True*

nothreads threads to run blast with (note the overlap with the Make -j parameter)

type: *integer*
default: *2*
optional: *True*

program blast program to use (default: blastn)

type: *set*
default: *blastn*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.10 blastdb

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Takes either a set of fasta files or a single multi-fasta input file and creates a BLAST database.

Filesets

dbname

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./db*

input The file with all input FASTA sequences for the blastdb.

type: *single*
category: *input*
optional: *False*
pattern: **/*.fasta*

Parameters

protein Protein database? (T)rue) or not (F)alse (default: F)

type: *set*
default: *F*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Tue, 03 Jan 2012 15:00:23

1.7.11 blat

Blat

Run BLAT on an set of input files (query) vs a database.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

db type of the database (dna, prot or dnax)

type: *set*
default: ‘‘
optional: *False*

db_id_list a sorted list of db ids and descriptions, enhances the report generated

type: *file*
default: ‘‘
optional: *True*

db_type type of the database (dna, prot or dnax)

type: *set*
default: *dna*
optional: *True*

eval evalue cutoff to select the reported hits on (defaults to 1e-15)

type: *float*
default: *1e-10*
optional: *True*

gff_source Source field for the generated GFF files

type: *string*
default: ‘‘
optional: *False*

input_dir source field in the generated gff

type: *directory*
default: ‘‘
optional: *False*

input_extension extension of the input files

type: *string*
default: *fasta*
optional: *True*

input_file input query file. If this variable is not defined, the combination of blat_input_dir and blat_input_extension is used to find a list of input files

type: *file*
default: ‘‘
optional: *False*

query_type type of the query (dna, rna, prot, dnax or rnax)

type: *set*
default: *dna*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.12 bowtie

Bowtie

Run BOWTIE on an set of input files (query) vs a database index.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template

run *no help defined*

Filesets

input Fasta/fastq input files for bowtie

output Output files

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.bam*

Parameters

db The (basename of the) bowtie database to use.

type: *string*
default: *{}*
optional: *False*

extra_params extra parameters to feed bowtie

type: *string*
default: ‘‘
optional: *True*

input_format Format of the input files

type: *set*
default: *fastq*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.13 bowtie_pe

Run BOWTIE on an set of input files (query) vs a database index.

Commands

clean Remove all job data, not the Moa job itself

finish finish up

report Create a report on the results

run Execute soapdenovo in paired-end mode

Filesets

db The (basename of the) bowtie database to use.

type: *single*
category: *prerequisite*
optional: *False*
pattern: *../20.bowtiedb/db*

fq_forward_input fastq input files directory - forward

fq_reverse_input fastq input files directory - reverse

type: *map*
source: *fq_forward_input*
category: *input*
optional: *True*
pattern: **/*_2.fq*

output Bam output file

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

Parameters

extra_params extra parameters to feed to bowtie

type: *string*
default: ‘‘
optional: *True*

input_format Format of the input files

type: *set*
default: *fastq*
optional: *True*

max_insertsize Maximum allowed insertsize

type: *integer*
default: *250*
optional: *True*

min_insertsize Minimum allowed insertsize

type: *integer*
default: *1*
optional: *True*

orientation orientation of the reads, allowed values are fr, rf, ff

type: *{}*
default: *fr*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.14 bowtie_se

Run BOWTIE on an set of input files (query) vs a database index.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template

run *no help defined*

Filesets

fq_input fastq input files directory

output Bam output file

type: *map*
source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

Parameters

ebwt_base The (basename of the) bowtie database to use.

type: *string*
default: *{}*
optional: *False*

extra_params extra parameters to feed to bowtie

type: *string*
default: ‘‘
optional: *True*

input_format Format of the input files

type: *set*
default: *fastq*
optional: *True*

output_format Format of the output file

type: *set*
default: *bam*
optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.15 bowtiedb

Bowtie index builder

Builds a bowtie index from a reference sequence

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Create the bowtie database

Filesets

input Input fasta file for the bowtie database

output database name to create

type: *single*
category: *output*
optional: *{}*
pattern: *db*

Parameters

extra_params any option parameters

type: *string*
default: ‘‘
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Dec 09 07:56:48 2010

1.7.16 bwa_aln

Use BWA to align a set of fastq reads against a db

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run bwa aln

Filesets

input Fastq input files

output

type: *map*
source: *input*
category: *output*

optional: *{}*
pattern: *./*.sai*

Parameters

best_hits_stop stop searching when there are >INT equally best hits

type: *integer*
default: *{}*
optional: *True*

color_space input sequences are in the color space

type: *boolean*
default: *False*
optional: *True*

db bwa database to align against

type: *string*
default: *{}*
optional: *False*

edit_dist_missing_prob max

type: *float*
default: *{}*
optional: *True*

gap_ext_max

type: *integer*
default: *{}*
optional: *True*

gap_ext_penalty gap extension penalty

type: *integer*
default: *{}*
optional: *True*

gap_open_penalty gap open penalty

type: *integer*
default: *{}*
optional: *True*

gap_opens_max maximum number or fraction of gap opens

type: *integer*
default: *{}*
optional: *True*

log_gap_penalty_del log-scaled gap penalty for long deletions

type: *boolean*
default: *{}*
optional: *True*

max_ext_long_del maximum occurrences for extending a long deletion

type: *integer*
default: *{}*
optional: *True*

max_queue_entry maximum entries in the queue

type: *integer*
default: *{}*
optional: *True*

mismatch_penalty mismatch penalty

type: *integer*
default: *{}*
optional: *True*

no_indel_from_ends do not put an indel within INT bp towards the ends

type: *integer*
default: *{}*
optional: *True*

non_iterative non-iterative mode search for all n-difference hits (slow)

type: *boolean*
default: *False*
optional: *True*

quality_step quality threshold for read trimming down to 35bp

type: *integer*
default: *{}*
optional: *True*

seed_len Seed length

type: *integer*
default: *{}*
optional: *True*

seed_max_diff Maximum differences in the seed

type: *integer*
default: *{}*
optional: *True*

thread_num number of threads

type: *integer*
default: *{}*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Nov 10 07:56:48 2010

Modification date unknown

1.7.17 bwa_index

Bwa index builder

Builds a bwa index from a reference sequence

Commands

clean Remove all job data

run Create the index

Parameters

algorithm Algorithm for constructing BWT index. Available options are 'is' and 'bwtsw'

type: *string*
default: *is*
optional: *True*

color_space input sequences are in the color space

type: *boolean*
default: *False*
optional: *True*

input_fasta input fasta file for the database

type: *file*
default: *{}*
optional: *False*

prefix Name of the bwa index to create

type: *string*
default: *{}*
optional: *False*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.18 bwa_sampe

Generate alignments in SAM format given paired end reads

Commands

clean Remove all job data, not the Moa job itself

run run bwa sampe

Filesets

fq_forward_input fastq input files directory - forward

fq_reverse_input fastq input files directory - reverse

type: *map*
source: *fq_forward_input*
category: *input*
optional: *True*
pattern: **/*_2.fq*

output_bam

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

sai_forward_input sai input files - forward

type: *map*
source: *fq_forward_input*
category: *input*
optional: *False*
pattern: **/*_1.sai*

sai_reverse_input sai input files - reverse files

type: *map*
source: *sai_forward_input*
category: *input*
optional: *True*
pattern: **/*_2.sai*

Parameters

db bwa database to align against

type: *string*
default: *{}*
optional: *False*

disable_insert_size disable insert size estimate (force -s)

type: *boolean*
default: *False*
optional: *True*

disable_SW disable Smith-Waterman for the unmapped mate

type: *boolean*
default: *False*
optional: *True*

max_aln_out maximum hits to output for paired reads

type: *integer*
default: *3*
optional: *True*

max_insert_size maximum insert size

type: *integer*
default: *500*

optional: *True*

max_occ_read maximum occurrences for one end

type: *integer*
default: *{}*
optional: *True*

max_out_discordant_pairs maximum hits to output for discordant pairs

type: *integer*
default: *{}*
optional: *True*

preload_index preload index into memory (for base-space reads only)

type: *boolean*
default: *False*
optional: *True*

prior_chimeric_rate prior of chimeric rate (lower bound)

type: *integer*
default: *{}*
optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Nov 25 17:06:48 2010

Modification date unknown

1.7.19 bwa_samse

Generate alignments in SAM format given single end reads, using both 'bwa samse'.

Commands

clean Remove all job data, not the Moa job itself

run run bwa samse

Filesets

fq_input fastq input file

output_bam output bam file

type: *map*
source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

sai_input sai input directory - filenames must correspond to the fastq input files

type: *map*
source: *fq_input*
category: *input*
optional: *False*
pattern: **/*.sai*

Parameters

db bwa database to align against

type: *string*
default: ‘‘
optional: *False*

max_aln_out Maximum number of alignments to output in the XA tag for reads paired properly

type: *integer*
default: *3*
optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Nov 25 17:06:48 2010

Modification date unknown

1.7.20 cdsmatrix

CdsMatrix

Predicts (prokaryotic) using glimmer3.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Generate a matrix of CDS's

Filesets

input Directory with the cds files for Glimmer3

output Output blast files

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *./*.out*

reference reference multi fasta file

type: *single*
category: *prerequisite*
optional: *{}*
pattern: **/*.fasta*

table table files

type: *map*
source: *input*
category: *output*

optional: *True*
pattern: *./*.tab*

Parameters

cutoff score cutoff value - disregards hits below this score

type: *{}*
default: *100*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Thu, 21 Jul 2011 20:31:10 +1200

1.7.21 cleanFasta

clean Fasta

Convert files to unix format and convert all characters that are not an A,C,G,T or N to N.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Cleanup of a FASTA file (in place!)

Parameters

cf_input_dir Directory with the sequences to run cleanfasta on

type: *directory*
default: ‘‘
optional: *False*

cf_input_extension input file extension

type: *string*
default: *fasta*

optional: *True*

sed_command

type: *string*
default: */^>/!s/[^ACGTNacgtn]/N/g*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.22 clustalgroup

clustalw

Run clustalw on two sets of sequences

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run clustalw

Parameters

cwg_input_dir This set of sequences to run clustalw on

type: *directory*
default: ‘‘
optional: *False*

cwg_input_extension Input file extension

type: *string*
default: *fasta*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.23 clustalpair

clustalw

Run clustalw on two sets of sequences

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run clustalw

Parameters

input_dir_a This set is compared to the sequences in input_dir_b. only a forward comparison is made (a against b, not the other way round)

type: *directory*
default: ‘‘
optional: *False*

input_dir_b The set to compare against

type: *directory*
default: ‘‘
optional: *False*

input_extension Extension of the input files

type: *string*
default: *fasta*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.24 clustalw

clustalw

Run clustalw on two sets of sequences

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run clustalw

Parameters

input_dir_a This set is compared to the sequences in input_dir_b.

type: *directory*
default: ‘‘
optional: *False*

input_dir_b The set to compare against. Only a forward comparison is made (a against b, not the other way round)

type: *directory*
default: ‘‘
optional: *False*

input_extension Extension of the input files

type: *string*
default: *fasta*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.25 concatenate

Concatenate

Concatenate a set of fasta files into one.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

input_dir Directory with the input data

type: *directory*
default: ‘‘
optional: *False*

input_extension Extension of the input files

type: *string*
default: *fasta*
optional: *True*

name name of the file, the outputfile will become ./name.fasta

type: *string*
default: ‘‘
optional: *False*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.26 dottup

EMBOSS Dottup

Use dottup (from EMBOSS) to compare two sets of sequences

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Run dottup

Parameters

input_dir_a This set is compared to the sequences in input_dir_b.

type: *directory*
default: ‘‘
optional: *False*

input_dir_b The set to compare against

type: *directory*
default: ‘‘
optional: *True*

input_extension Extension of the dottup input files

type: *string*
default: *fasta*
optional: *True*

wordsize Wordsize used to discover similarities between sequences

type: *integer*
default: *8*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.27 empty

empty

Do nothing...

Commands

Parameters

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Mon Apr 04 16:02:58 2011

Modification date Mon Apr 04 16:03:18 2011

1.7.28 fasta2gff

GFF from FASTA

Derive GFF from a FASTA file, usually to accompany the Sequence for upload to a generic genome browser database.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

f2g_gffsource Source to be used in the gff

type: *string*
default: ‘‘
optional: *False*

f2g_input_dir Directory with the input fasta files

type: *directory*
default: ‘‘
optional: *False*

f2g_input_extension glob pattern of the fasta files (default: *.fasta)

type: *string*
default: *fasta*
optional: *True*

f2g_options options to be passed to the fasta2gff script

type: *string*
default: ‘‘
optional: *True*

f2g_output_dir Directory with the output gff

type: *directory*
default: *./gff*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.29 fastainfo

gather information on a set of fasta files

gather info on a set of input files

Commands

finish create a report

run generate info on each of the input sequences

Filesets

input "fastainfo" input files

output "fastainfo" raw output files

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *stats/*.out*

stats "fastainfo" collect stat files

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *stats/*.stat*

Parameters

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Mon, 11 Jul 2011 15:15:20

Modification date Mon, 11 Jul 2011 15:15:12

1.7.30 fastqc

Run FastQC for fastq QC

Run FastQC on a set a fastq files - quality assessment

Commands

finish Run Fastqc

finish delegates execution to: **report**

report Generate a simple fastqc report

run *no help defined*

Filesets

input fastqc input files'

touch touch files - track if a file has been processed - do not touch this unless you know what you're doing.

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *./*.touch*

Parameters

output_dir output directory for the fastQC report

type: *dir*
default: .
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Thu, 28 Apr 2011 09:27:17 +1200

Modification date Thu, 28 Apr 2011 14:19:04 +1200

1.7.31 fastx_clipper

run fastx_clipper

Commands

clean Remove all job data, not the Moa job itself

run run fastx_clipper

Filesets

input fastq input files directory

output

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.fq*

Parameters

adaptor ADAPTER string. default is CCTTAAGG (dummy adapter).

type: *string*
default: *CCTTAAGG*
optional: *True*

adaptor_and_bases Keep the adapter and N bases after it.

type: *integer*
default: *0*
optional: *True*

compress_output Compress output with GZIP.

type: *boolean*
default: *False*
optional: *True*

debug_output DEBUG output.

type: *boolean*
default: *False*
optional: *True*

help help screen

type: *boolean*
default: *False*
optional: *True*

keep_unknown_nuc_seq keep sequences with unknown (N) nucleotides. default is to discard such sequences.

type: *boolean*
default: *False*
optional: *True*

out_adaptor_only_seq Report Adapter-Only sequences.

type: *boolean*
default: *False*
optional: *True*

rm_clipped_seq Discard clipped sequences (i.e. - keep only sequences which did not contained the adapter).

type: *boolean*
default: *False*
optional: *True*

rm_non_clipped_seq Discard non-clipped sequences (i.e. - keep only sequences which contained the adapter).

type: *boolean*
default: *False*
optional: *True*

rm_short_seq discard sequences shorter than N nucleotides. default is 5.

type: *integer*
default: *5*
optional: *True*

verbose Verbose - report number of sequences. If [-o] is specified, report will be printed to STDOUT. If [-o] is not specified (and output goes to STDOUT), report will be printed to STDERR.

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Dec 06 17:06:48 2010

Modification date unknown

1.7.32 fastx_qual_stats

run fastx_quality_stats, fastq_quality_boxplot_graph.sh and fastx_nucleotide_distribution_graph.sh

Commands

clean Remove all job data, not the Moa job itself

run run fastx_quality_stats, fastq_quality_boxplot_graph.sh and fastx_nucleotide_distribution_graph.sh

Filesets

boxplot_output

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.png*

input fastq input files directory

nuc_distr_output

type: *map*
source: *input*
category: *output*

optional: *{}*
pattern: *./*.png*

qual_output

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.txt*

Parameters

gen_postScript_file Generate PostScript (.PS) file. Default is PNG image.

type: *boolean*
default: *False*
optional: *True*

graph_title Title - will be plotted on the graph.

type: *string*
default: *{{ input_glob }}*
optional: *True*

help help screen

type: *boolean*
default: *False*
optional: *True*

new_out_format New output format (with more information per nucleotide/cycle)

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Dec 03 17:06:48 2010

Modification date unknown

1.7.33 gather

gather files

gather a set of files and create hardlinks to. Hardlinks have as advantage that updates are noticed via the timestamp. Hence, make recognizes them.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run gather files

Parameters

g_input_dir list of directories with the input files

type: *directory*
default: ‘‘
optional: *False*

g_input_pattern glob pattern to download

type: *string*
default: *
optional: *True*

g_limit limit the number of files gathered (with the most recent files first, defaults to 1mln)

type: *integer*
default: *1000000*
optional: *True*

g_name_sed SED expression to be executed on each file name - allows you to change file names

type: *string*
default: *s/a/a/*
optional: *True*

g_output_dir Output subdirectory, defaults to .

type: *directory*
default: .
optional: *True*

g_parallel allow parallel execution (T) or not (**F**). If for example concatenating to one single file, you should not have multiple threads.

type: *set*
default: *F*
optional: *True*

g_powerclean Do brute force cleaning (T/F). Remove all files, except moa.mk & Makefile when calling make clean. Defaults to F.

type: *set*
default: *F*
optional: *True*

g_process Command to process the files. If undefined, hardlink the files.

type: *string*
default: *ln -f $$< $$(g_target)*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.34 genemarks

geneMarkS

predict genes using geneMarkS

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

input Directory with the input files for Genemarks

Parameters

gff_source source field to use in the gff. Defaults to "geneMarkS"

type: *string*
default: *genemarkS*
optional: *True*

matrix the matrix to use

type: *file*
default: ''
optional: *True*

miscellaneous

Backend ruff

Author

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.35 getorf

Getorf

Predicts open reading frames using the EMBOSS [[emboss]] getorf tool.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

gff

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./gff/*.gff*

input Input files for getorf

output

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./out/*.out*

Parameters

circular Is the sequence linear?

type: *set*
default: *N*
optional: *True*

find What to output? 0: Translation between stop codons, 1: Translation between start & stop codon, 2: Nucleotide sequence between stop codons; 3: Nucleotide sequence between start and stop codons. Default: 3

type: *set*
default: *3*
optional: *True*

gff_source source field to use in the gff.

type: *string*
default: *getorf*
optional: *True*

maxsize maximal nucleotide size of the predicted ORF.

type: *integer*
default: *1000000*
optional: *True*

minsize minimal nucleotide size of the predicted ORF.

type: *integer*
default: *30*
optional: *True*

table Genetic code to use: 0 Standard; 1 Standard with alternative initiation codons; 2 Vertebrate Mitochondrial; 3 Yeast Mitochondrial; 4 Mold, Protozoan, Coelenterate Mitochondrial and Mycoplasma/Spiroplasma; 5 Invertebrate Mitochondrial; 6 Ciliate Macronuclear and Dasycladacean; 9 Echinoderm Mitochondrial; 10 Euplotid Nuclear; 11 Bacterial; 12 Alternative Yeast Nuclear; 13 Ascidian Mitochondrial; 14 Flatworm Mitochondrial; 15 Blepharisma Macronuclear; 16 Chlorophycean Mitochondrial; 21 Trematode Mitochondrial; 22 Scenedesmus obliquus; 23 Thraustochytrium Mitochondrial.

type: *set*
default: *11*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.36 glimmer3

Glimmer3

Predicts (prokaryotic) using glimmer3.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Glimmer3 is a open reading frame discovery program from the EMBOSS [[emboss]] package. It takes a set of input sequences and predicts all open reading frames. Additionally, this template converts the default output (predicted protein sequences) to GFF3.

Filesets

cds CDS output files from glimmer3

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *cds/*.fasta*

gff GFF output files from glimmer3

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *gff/*.gff*

input Directory with the input files for Glimmer3

output Raw output files from glimmer3

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *out/*.g3*

pep peptide output files from glimmer3

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *pep/*.fasta*

Parameters

gene_len Minimum gene length (glimmer3 -g/–gene_len)

type: *integer*
default: *110*
optional: *True*

gff_source source field to use in the gff. Defaults to "glimmer3"

type: *string*
default: *glimmer3*
optional: *True*

max_overlap Maximum overlap, see the glimmer documentation for the -o or –max_olap parameter

type: *integer*
default: *50*
optional: *True*

stop_codons stop codons

type: *{}*
default: *tag,tga,taa,nnn,tnn,ann,gnn,cnn*
optional: *True*

treshold treshold for calling a gene a gene (glimmer3 -t)

type: *integer*
default: *30*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.37 gmap

Gmap

Run GMAP on an set of input files (query) vs a database index.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

align

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./align/*.align*

genepred

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./genepred/*.genepred*

gff

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./gff/*.gff*

gff_invert

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./gff/*.invert.gff*

input Sequences to map

raw

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./raw/*.raw*

Parameters

db Gmap db

type: *file*
default: ‘‘
optional: *False*

extra_parameters extra parameters to feed to gmap

type: *string*
default: ‘‘
optional: *True*

gff_source Source field to use in the output GFF

type: *string*
default: *gmap*
optional: *True*

invert_gff Invert the GFF (T/*F*)

type: *set*
default: *T*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.38 gmapdb

gmapdb index builder

Builds gmapdb index from a reference sequence

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

input The reference sequence to build a gmap database with.

type: *single*
category: *input*
optional: *False*
pattern: */*.fasta*

Parameters

name Name of the gmap index to create

type: *string*
default: *gmapdb*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.39 gsMapper

GSMapper

Run the Roche GS Reference mapper

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

annotation Gene annotation file in the UCSC GenePred format

type: *file*
default: ‘‘
optional: *True*

min_overlap_ident Minimum identity length in the assembly step

type: *integer*
default: *90*
optional: *True*

min_overlap_len Minimum overlap length in the assembly step

type: *integer*
default: *40*
optional: *True*

name Name identifying this mapping in the output gff

type: *string*
default: ‘‘
optional: *False*

reference_fasta A multifasta file with the reference sequence(s)with the library id.

type: *file*
default: ‘‘
optional: *True*

sfffile SFF files with reads to map against the reference sequences

type: *file*
default: ‘‘
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.40 h_blast

Hadoop Blast

Runs BLAST on a hadoop cluster

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Similar to a normal blast, but now running on an hadoop cluster

Parameters

db Location of the blast database

type: *file*
default: ‘‘
optional: *False*

eval e value cutoff

type: *float*
default: *1e-10*
optional: *True*

hadoop_base location of the hadoop installation

type: *directory*
default: ‘‘
optional: *False*

hdfs_base htfs://SERVER:PORT for the hdfs filesystem, defaults to “hdfs://localhost:9000”

type: *string*
default: *hdfs://localhost:9000*
optional: *True*

input_dir location of the hadoop installation

type: *directory*
default: ‘‘
optional: *False*

input_extension input file extension

type: *string*
default: *fasta*
optional: *True*

nohits number of hits to report

type: *integer*
default: *50*
optional: *True*

nothreads threads to run blast with (note the overlap with the Make -j parameter)

type: *integer*
default: *1*
optional: *True*

program blast program to use (default: blastn)

type: *set*
default: *blastn*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.41 hagfish

Run hagfish_extract & hagfish_combine

Run the preparatory steps for hagfish

Commands

circos convert to circos histogram data

clean remove all Hagfish files

combine *no help defined*

report *no help defined*

run Run hagfish

Filesets

input "hagfish" input files

output "hagfish" output files

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *./touch/*.touch*

Parameters

circosbinsize Binsize for generating circos formatted histograms

type: *int*
default: *{}*
optional: *True*

max_ok Maximal acceptable insert size for an aligned pair. If omitted, hagfish will make an estimate

type: *int*
default: *0*
optional: *True*

min_ok Minimal acceptable insert size for an aligned pair. If omitted, hagfish will make an estimate

type: *int*
default: *0*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Tue Mar 29 16:34:19 2011

Modification date Thu, 19 May 2011 20:49:04 +1200

1.7.42 kanga

use kanga to align short reads to a reference genome

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run kanga

Filesets

input_fasta Fasta input file

output output files

type: *map*
source: *rds_input*
category: *output*
optional: *True*
pattern: *./*.sam*

output_bam output files

type: *map*
source: *rds_input*
category: *output*
optional: *True*
pattern: *./*.bam*

output_log output log file

type: *map*
source: *rds_input*
category: *output*
optional: *{}*
pattern: *./*.log.txt*

rds_input rds (preprocessed) input files

sfx_input sfx array lookup file

Parameters

color_space process for colorspace (SOLiD)

type: *boolean*
default: *False*
optional: *True*

extra_params any extra parameters

type: *string*
default: ‘‘
optional: *True*

help print this help and exit

type: *boolean*
default: *False*
optional: *True*

max_Ns maximum number of intermediate N’s in reads before treating read as unalignable

type: *integer*
default: *1*
optional: *True*

max_pair_len accept paired end alignments with apparent length of at most this

type: *integer*
default: *300*
optional: *True*

min_pair_len accept paired end alignments with apparent length of at least this

type: *integer*
default: *100*
optional: *True*

no_multireads do not accept multiple reads aligning to the same loci

type: *boolean*
default: *False*
optional: *True*

out_format 0 - CSV loci only, 1 - CSV loci + match sequence, 2 - CSV loci + read sequence, 3 - CSV loci + read + match sequence, 4 - UCSC BED, 5 - SAM format

type: *integer*
default: *0*
optional: *True*

pe_mode 0 - none, 1 - paired ends with recover orphan ends, 2 - paired end no orphan recovery

type: *integer*
default: *0*
optional: *True*

quality fastq quality scoring- 0 - sanger, 1m - Illumina 1.3+, 2 - Solexa < 1.3, 3 - Ignore quality

type: *integer*
default: *3*
optional: *True*

thread_num number of processing threads (0 sets threads to number of CPU cores)

type: *integer*
default: *0*
optional: *True*

trim3 trim this number of bases from 3' end of reads when loading raw reads

type: *integer*
default: *0*
optional: *True*

trim5 trim this number of bases from 5' end of reads when loading raw reads

type: *integer*
default: *0*
optional: *True*

version print version information and exit

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Nov 10 07:56:48 2010

Modification date unknown

1.7.43 kangar_pe

use kangar to pre process raw fq reads

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run kangar

Filesets

fq_forward_input fastq input files - forward - containing the 5' end

fq_reverse_input fastq input files directory - reverse - containing the 3' end

type: *map*
source: *fq_forward_input*
category: *input*
optional: *True*
pattern: */*_2.fq*

output_log output log file

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*
pattern: *./*.log.txt*

rds_output output rds file

type: *map*
source: *fq_forward_input*
category: *output*
optional: *True*
pattern: *./*.rds*

Parameters

extra_params any extra parameters

type: *string*
default: ''
optional: *True*

help print this help and exit

type: *boolean*
default: *False*

optional: *True*

mode processing mode 0 - single end create, 1 - paired end create, 2 - output statistics 3 - dump as fasta

type: *integer*
default: *0*
optional: *True*

quality fastq quality scoring- 0 - sanger, 1m - Illumina 1.3+, 2 - Solexa < 1.3, 3 - Ignore quality

type: *integer*
default: *3*
optional: *True*

reads_num limit number of reads (or dumps) in each input file to this many, 0 if no limit

type: *integer*
default: *0*
optional: *True*

rm_duplicates remove duplicate reads retaining only one

type: *boolean*
default: *False*
optional: *True*

trim3 trim this number of bases from 3' end of sequence

type: *integer*
default: *0*
optional: *True*

trim5 trim this number of bases from 5' end of sequence

type: *integer*
default: *0*
optional: *True*

version print version information and exit

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Nov 10 07:56:48 2010

Modification date unknown

1.7.44 kangar_se

use kangar to pre process raw fq single end reads

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run kangar

Filesets

fq_input fastq input files - forward - containing the 5' end

output_log output log file

type: *map*
source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.log.txt*

rds_output output rds file

type: *map*
source: *fq_input*
category: *output*
optional: *True*
pattern: *./*.rds*

Parameters

extra_params any extra parameters

type: *string*
default: ‘‘
optional: *True*

help print this help and exit

type: *boolean*
default: *False*
optional: *True*

mode processing mode 0 - single end create, 1 - paired end create, 2 - output statistics 3 - dump as fasta

type: *integer*
default: *0*
optional: *True*

quality fastq quality scoring- 0 - sanger, 1m - Illumina 1.3+, 2 - Solexa < 1.3, 3 - Ignore quality

type: *integer*
default: *3*
optional: *True*

reads_num limit number of reads (or dumps) in each input file to this many, 0 if no limit

type: *integer*
default: *0*
optional: *True*

rm_duplicates remove duplicate reads retaining only one

type: *boolean*
default: *False*
optional: *True*

trim3 trim this number of bases from 3' end of sequence

type: *integer*
default: *0*
optional: *True*

trim5 trim this number of bases from 5' end of sequence

type: *integer*
default: *0*
optional: *True*

version print version information and exit

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Nov 10 07:56:48 2010

Modification date unknown

1.7.45 kangax

use kangax to create the suffix array lookup database for the reference genome

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run run kangax

Filesets

input_fasta Fasta input file

output_log output log file

type: *map*
source: *input_fasta*
category: *output*
optional: *{}*
pattern: *./*.log.txt*

output_sfx output suffix array lookup

type: *map*
source: *input_fasta*
category: *output*
optional: *{}*
pattern: *./*.sfx*

Parameters

block_seq_len generated suffix blocks to hold at most this length (MB) concatenated sequences

type: *integer*
default: *3300*
optional: *True*

color_space generate for colorspace (SOLiD)

type: *boolean*
default: *False*
optional: *True*

extra_params any extra parameters

type: *string*
default: ‘‘
optional: *True*

help print this help and exit

type: *boolean*
default: *False*
optional: *True*

reference_species reference species

type: *string*
default: ‘‘
optional: *False*

target_dep generate target file only if missing or older than any independent source files

type: *boolean*
default: *False*
optional: *True*

version print version information and exit

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Nov 10 07:56:48 2010

Modification date unknown

1.7.46 lftp

lftp

Use LFTP to download files. This template has two modi, one is set lftp_mode to mirror data, in which case both lftp_url and lftp_pattern (default *) are used. The other modus is lftp_mode=get, when one file defined by lftp_url is downloaded. In the mirror mode it is possible to download only those files that are newer as the files already downloaded by using the lftp_timestamp parameter

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run execute the download

Parameters

dos2unix Run dos2unix to prevent problems with possible dos text files

type: *set*
default: *F*
optional: *True*

get_name target name of the file to download

type: *string*
default: ‘‘
optional: *True*

lftp_output_dir subdir to create & write all output to. If not defined, data will be downloaded to directory containing the Makefile

type: *directory*
default: .
optional: *True*

lock Lock this job after running. This means that you will have to manually unlock the job before lftp actually reruns. This is a good choice if your downloading large datasets or have a slow connection

type: *set*
default: *T*
optional: *True*

mode Mode of operation - mirror or get. Mirror enables timestamping. Get just gets a single file. If using get, consider setting depend_lftp_timestamp to F. When using get, the full url should be in lftp_url. lftp_pattern is ignored. Defaults to mirror.

type: *set*
default: *get*
optional: *True*

noclean set of files not to be deleted by the powerclean

type: *string*

default: *moa.mk Makefile*
optional: *True*

pass password for the remote site, note that this can be defined on the commandline using: make lftp_pass=PASSWORD

type: *password*
default: ‘‘
optional: *True*

pattern glob pattern to download

type: *string*
default: *‘*’*
optional: *True*

powerclean Do brute force cleaning (T/F). Remove all files, except moa.mk & Makefile when calling make clean. Defaults to F.

type: *set*
default: *F*
optional: *True*

timestamp Depend on lftp to decide if a file needs updating, else a touchfile is created that you need to delete or touch before updating (T/*F*)

type: *set*
default: *F*
optional: *True*

url The base url to download from

type: *string*
default: ‘‘
optional: *True*

user username for the remote site

type: *string*
default: ‘‘
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.47 map

Execute a “map” ad-hoc analysis

Execute one command, on a number of input files.

Commands

run *no help defined*

Filesets

input “map” input files

output “map” output files

type: *map*
source: *input*
category: *output*
optional: *True*
pattern: *./**

Parameters

process The command to execute

type: *string*
default: *True*
optional: *False*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Tue Mar 29 16:34:19 2011

Modification date Wed Mar 30 06:02:01 2011

1.7.48 maq_fasta2bfa

Convert fasta to bfa

> Converts a FASTA file to MAQ format for use with a BFA a maq_fasta2bfa index from a reference sequence

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

bfa

> *type*: *map*
> *source*: *input*
> *category*: *output*
> *optional*: *{}*
> *pattern*: *./bfa/*.bfa*

input input FASTA files

Parameters

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.49 maq_fastq2bfq

Convert FASTQ to BFQ

> Converts a FASTQ file to MAQ BFQ format.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

bfq

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./bfq/*.bfq*

input input FASTA files

Parameters

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.50 maq_match_pair

MAQ paired ends mapper

Map paired ends to a reference sequence using MAQ

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

forward_suffix Suffix of each forward filename - recognize forward files this way. Note this is not a regular extension, no . is assumed between the filename & suffix

type: *string*
default: *_f.bfq*
optional: *True*

maxdist max outer distance for a (non RF) readpair. This applies to illumina matepairs - i.e. short inserts

type: *integer*
default: *250*
optional: *True*

read_dir directory containing the forward reads

type: *string*
default: ‘‘
optional: *False*

reference Reference bfa file to map the reads to

type: *string*
default: ‘‘
optional: *False*

reverse_suffix suffix of reverse files

type: *string*
default: *_r.bfq*
optional: *True*

RF_maxdist max outer distance for an RF readpair (corresponds to the -A parameter). This applies to long insert illumina pairs

type: *integer*
default: *15000*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.51 maq_pe

Generate alignments in SAM format given paired end reads using Maq.

Commands

clean Remove all job data, not the Moa job itself

run run maq's fasta2bfa, fastq2bfq and map.

Filesets

bam_output bam alignment output file

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

bfa_output BFA Index name

type: *single*
category: *other*
optional: *{}*
pattern: *{}*

bfq_forward_output bfq files - forward files

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*
pattern: *./*_1.bfq*

bfq_reverse_output bfq files - reverse files

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*

pattern: *./*_2.bfq*

fa_input directory with reference fasta file name

fq_forward_input fastq input files directory - forward files

fq_reverse_input fastq input files directory - reverse files

type: *map*
source: *fq_forward_input*
category: *input*
optional: *{}*
pattern: **/*_2.fq*

map_output maq map output files

type: *map*
source: *fq_forward_input*
category: *output*
optional: *{}*
pattern: *./*.map*

Parameters

disable_sw disable Smith-Waterman alignment

type: *boolean*
default: *False*
optional: *True*

extra_parameters Any extra parameters

type: *string*
default: ‘‘
optional: *True*

first_read_len length of the first read (<=127)s

type: *integer*
default: *0*

optional: *True*

match_in_colorspace match in the colorspace

type: *boolean*
default: *False*
optional: *True*

max_dist_read_pairs max distance between two paired reads s

type: *integer*
default: *250*
optional: *True*

max_dist_RF_read_pairs max distance between two RF paired reads s

type: *integer*
default: *0*
optional: *True*

max_mismatch_qual_sum maximum allowed sum of qualities of mismatches

type: *integer*
default: *70*
optional: *True*

max_num_hits_out max number of hits to output. >512 for all 01 hits.

type: *integer*
default: *250*
optional: *True*

num_mismatch_24bp number of mismatches in the first 24bp

type: *integer*
default: *2*
optional: *True*

read_ref_diff_rate rate of difference between reads and references

type: *float*
default: *0.001*
optional: *True*

sec_read_len length of the second read (<=127)s

type: *integer*
default: *0*
optional: *True*

trim_all_reads trim all reads (usually not recommended)

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Dec 03 17:06:48 2010

Modification date unknown

1.7.52 maq_se

Generate alignments in SAM format given single end reads using Maq.

Commands

clean Remove all job data, not the Moa job itself

run run maq's fasta2bfa, fastq2bfq and map.

Filesets

bam_output bam alignment output file

type: *map*

source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.bam*

bfa_output BFA Index name

type: *single*
category: *other*
optional: *{}*
pattern: *{}*

bfq_output bfq files - forward files

type: *map*
source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.bfq*

fa_input directory with reference fasta file name

fq_input fastq input files

map_output maq map output files

type: *map*
source: *fq_input*
category: *output*
optional: *{}*
pattern: *./*.map*

Parameters

disable_sw disable Smith-Waterman alignment

type: *boolean*
default: *False*
optional: *True*

extra_parameters other parameters

type: *string*
default: ‘‘
optional: *True*

match_in_colorspace match in the colorspace

type: *boolean*
default: *False*
optional: *True*

max_mismatch_qual_sum maximum allowed sum of qualities of mismatches

type: *integer*
default: *70*
optional: *True*

max_num_hits_out number of mismatches in the first 24bp

type: *integer*
default: *250*
optional: *True*

num_mismatch_24bp number of mismatches in the first 24bp

type: *integer*
default: *2*
optional: *True*

read_ref_diff_rate rate of difference between reads and references

type: *float*
default: *0.001*
optional: *True*

trim_all_reads trim all reads (usually not recommended)

type: *boolean*

default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers, Yogini Idnani

Creation date Wed Dec 02 17:06:48 2010

Modification date unknown

1.7.53 moatest

Unittest template

Not to be used - is used by unitmoatests

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Do nothing - no need to call this.

Parameters

test_opt test variable

type: *string*
default: *konijntje*
optional: *True*

txt test variable

type: *string*
default: ‘‘
optional: *False*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.54 mummer

mummer

Run mummer between two sequences

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Run mummer

Filesets

input Set 1 input fasta files

reference Set 1 input fasta files

Parameters

base base name for all generated files

type: *{}*
default: *out*
optional: *True*

breaklen Set the distance an alignment extension will attempt to extend poor scoring regions before giving up (default 200)

type: *integer*
default: *200*
optional: *True*

genomecenter genome center - used in the AGP file

type: *{}*
default: *pflnz*
optional: *True*

gff_source GFF source field

type: *{}*
default: *mumscaff*

optional: *True*

linker linker sequence for the merged output sequence

type: *{}*
default: *NNNNNNCTAGCTAGCATGNNNNNN*
optional: *True*

matchmode use all matching fragments (max) or only unique matchers (mum)

type: *set*
default: *mum*
optional: *True*

mum_plot_raw plot an alternative visualization where mummer does not attempt to put the sequences in the correct order

type: *boolean*
default: *False*
optional: *True*

organism Organism name - used in the AGP file

type: *{}*
default: ‘‘
optional: *True*

taxid Taxonomy id - used in the AGP file

type: *{}*
default: ‘‘
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.55 ncbi

Download data from NCBI

Download a set of sequences from NCBI based on a query string *ncbi_query* and database *ncbi_db*. This tempate will run only **once**, after a succesful run it creates a lock file that you need to remove to rerun

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Download from NCBI

Parameters

db NCBI database

type: *string*
default: *nuccore*
optional: *True*

query NCBI query (for example txid9397[Organism%3Aexp])

type: *string*
default: ‘‘
optional: *True*

rename_sequence try to rename the sequence - note, this does not work if you are downloading more than one sequence

type: *boolean*
default: *False*
optional: *True*

sequence_name Name of the file to write the downloaded sequences to. Use ‘from_dir’ to have the sequence name extracted from the directory name

type: *string*
default: *out*

optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.56 newbler

Newbler

Run a simple, out of the box, newbler assembly. As an extra feature, this template automatically creates uniquely named links to the two main output fasta files (454AllContigs.fna, 454LargeContigs.fna). This is convenient for subsequence gather steps. The links are named after the directory.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

input input SFF files

Parameters

largecontig_cutoff min length of a contig in 454LargeContigs.fna

type: *integer*
default: ‘‘
optional: *True*

library_name A library identifier for this assembly. This is used to create an extra fasta file, named using this variable, that contain the generated contigs with their ids prepended with the library id.

type: *string*
default: *$(shell echo ‘basename $(CURDIR)* | sed “s/[////]//g”)‘
optional: *True*

mid_configuration Mid configuration file to use

type: *file*
default: ‘‘
optional: *True*

mids mids to use for this assembly

type: *string*
default: ‘‘
optional: *True*

min_identity Minimal overalp identity used during assembly

type: *integer*
default: ‘‘
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.57 newjobtest

Execute a “simple” ad hoc analysis

Execute one command, No in or output files are tracked by Moa.

Commands

run *no help defined*

Parameters

process The command to execute

type: *string*

default: *True*
optional: *False*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Tue Mar 29 16:34:19 2011

Modification date Wed Mar 30 06:02:01 2011

1.7.58 nstretch

Nstretch

Run NSTRETCH on an set of input files

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

input_dir input dir with the fasta files

type: *directory*
default: ‘‘
optional: *False*

input_extension extension of the input files

type: *string*
default: *fasta*
optional: *True*

len minimal number of Ns before its reported (default 10)

type: *integer*
default: *10*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.59 orthomcl

Run OrthoMCL

Execute one command, No in or output files are tracked by Moa.

Commands

run *no help defined*

Parameters

db Db name

type: *string*
default: *{}*
optional: *False*

eval Evalue cutoff for blast to use

type: *string*
default: *1e-5*
optional: *True*

group_prefix OrthoMCL prefix for group names

type: *string*
default: *g_*
optional: *True*

host Db Host

type: *localhost*

default: *{}*
optional: *True*

input_dir Input directory with compliant (read the manual) fasta files

type: *string*
default: *{}*
optional: *False*

login Db username

type: *string*
default: *None*
optional: *False*

mcl_i mcl -i value

type: *float*
default: *1.5*
optional: *True*

num_threads Number of threads to use

type: *integer*
default: *4*
optional: *True*

pass Db password

type: *string*
default: *None*
optional: *False*

port Db port

type: *integer*
default: *3306*

optional: *True*

prefix OrthoMCL prefix for the database tables

type: *string*
default: *ortho*
optional: *True*

vendor Db vendor

type: *string*
default: *mysql*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Tue Mar 29 16:34:19 2011

Modification date Wed Mar 30 06:02:01 2011

1.7.60 pregap

Pregap

Run Pregap. Note that running phrap could be a part of this.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

cloning_vector File containing the cloning vector

type: *file*
default: ‘‘
optional: *False*

ecoli_screenseq File containing ecoli screen sequences

type: *file*
default: ‘‘
optional: *False*

input_dir Directory with the input data

type: *string*
default: ‘‘
optional: *False*

input_pattern file name pattern

type: *string*
default: ‘‘
optional: *False*

quality_value_clip quality cutoff

type: *integer*
default: *10*
optional: *True*

repeat_masker_lib File with a repeatmasker library

type: *file*
default: ‘‘
optional: *False*

sequencing_vector File containing the sequencing vector

type: *file*
default: ‘‘
optional: *False*

template the template pregap config file to use. if not defined, Moa tries ./files/pregap.config.

type: *file*
default: *./files/pregap.config.*
optional: *True*

vector_primerfile File with the vector primers

type: *file*
default: ‘‘
optional: *False*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.61 project

Create a project

Create a new project, a placeholder for project settings, and used by several plugins.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run This template does not do anything - it is a project placeholder.

Parameters

description A description of what this project is supposed to achieve, how to use it, and what parameters are most important to set

type: *string*
default: ‘‘
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.62 reduce

Execute a "reduce" ad-hoc analysis

Execute one command, on a number of input files.

Commands

run *no help defined*

Filesets

input "reduce" input files

output "reduce" output files

type: *single*
category: *output*
optional: *True*
pattern: *./**

Parameters

process The command to execute

type: *string*
default: *True*
optional: *False*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Tue Mar 29 16:34:19 2011

Modification date Wed Mar 30 06:02:01 2011

1.7.63 sam2bam

Convert SAM to BAM using samtools

Converts a FASTQ file to MAQ BFQ format.

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Filesets

input input SAM files

output

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.bam*

Parameters

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.64 samtools_pileup

Print the alignment in the pileup format.

Commands

clean Remove all job data, not the Moa job itself

run run samtools pileup command

Filesets

fasta reference fasta file

type: *single*
category: *prerequisite*
optional: *True*
pattern: **/*.fasta*

input bam or sam files

output

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.pileup*

output_bam

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.sorted*

Parameters

cap_mapQ_at cap mapping quality at INT

type: *integer*
default: *60*
optional: *True*

extra_params any extra parameters

type: *string*
default: ‘‘
optional: *True*

filter_read_bits filtering reads with bits in INT

type: *integer*
default: *1796*
optional: *True*

input_is_SAM the input is in SAM

type: *boolean*
default: *False*
optional: *True*

num_haplotypes number of haplotypes in the sample (for -c/-g)

type: *integer*
default: *2*
optional: *True*

out_2nd_best output the 2nd best call and quality

type: *boolean*
default: *False*
optional: *True*

out_GLFv3_format output in the GLFv3 format (suppressing -c/-i/-s)

type: *boolean*
default: *False*
optional: *True*

out_maq_consensus output the maq consensus sequence

type: *boolean*
default: *False*
optional: *True*

phred_prob_indel phred prob. of an indel in sequencing/prep. (for -c/-g)

type: *integer*

default: *40*
optional: *True*

print_variants_only print variants only (for -c)

type: *boolean*
default: *False*
optional: *True*

prior_diff_haplotypes phred prob. of an indel in sequencing/prep. (for -c/-g)

type: *float*
default: *0.001*
optional: *True*

prior_indel_haplotypes number of haplotypes in the sample (for -c/-g)

type: *float*
default: *0.00015*
optional: *True*

show_lines_indels only show lines/consensus with indels

type: *boolean*
default: *False*
optional: *True*

simple_pileup_format simple (yet incomplete) pileup format

type: *boolean*
default: *False*
optional: *True*

theta_maq_model number of haplotypes in the sample (for -c/-g)

type: *float*
default: *0.85*

optional: *True*

use_SOAPsnp_model use the SOAPsnp model for SNP calling

type: *boolean*
default: *False*
optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Dec 15 17:06:48 2010

Modification date unknown

1.7.65 sffinfo

sffinfo

Roche sffinfor tool - extract information from sff files

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Use the Roche sffinfo tool to extract reads, quality scores, flowgrams and accession ids from one or more sff files

Filesets

accession

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.acc*

flowgram

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.flow*

input Sff input files

quality

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.qual*

sequence

type: *map*
source: *input*
category: *output*
optional: *{}*
pattern: *./*.reads*

Parameters

accessions Output the accessions

type: *set*
default: *T*
optional: *True*

flowgrams output the flowgrams

type: *set*
default: *F*
optional: *True*

quality Output quality scores

type: *set*
default: *T*
optional: *True*

sequences Output the sequences

type: *set*
default: *T*
optional: *True*

untrimmed output untrimmed sequences & qualities

type: *set*
default: *F*
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.66 simple

Execute a "simple" ad hoc analysis

Execute one command, No in or output files are tracked by Moa.

Commands

run *no help defined*

Parameters

process The command to execute

type: *string*
default: *True*
optional: *False*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Tue Mar 29 16:34:19 2011

Modification date Wed Mar 30 06:02:01 2011

1.7.67 soapdenovo_pe

Run Soapdenovo

Commands

clean Remove all job data

run Execute soapdenovo in paired-end mode

Filesets

fq_forward fastq input files directory - forward

fq_reverse fastq input files directory - reverse

type: *map*
source: *fq_forward*
category: *input*
optional: *True*
pattern: **/*_2.fq*

output soap denovo output file

type: *single*
category: *output*
optional: *True*
pattern: *{}*

Parameters

avg_insert library insert size

type: *integer*
default: *200*
optional: *{}*

executable which executable to use (SOAPdenovo-127mer, SOAPdenovo-31mer or SOAPdenovo-63mer)

type: *{}*
default: *SOAPdenovo-31mer*
optional: *True*

kmer kmer size

type: *integer*
default: *31*
optional: *True*

skip_config_file skip automatic config file generation - if you skip this, make sure that you have a soap.config configuration file in the current directory

type: *boolean*
default: *False*
optional: *True*

threads no threads to use

type: *integer*
default: *8*
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Mon, 21 Nov 2011 12:47:16

Modification date Mon, 21 Nov 2011 12:47:22

1.7.68 statsidx

Retrieve and print stats from BAM file to an index file

Commands

clean Remove all job data, not the Moa job itself

run run samtools idxstats

Filesets

input bam input files directory - forward files

output

type: *map*
source: *input*
category: *output*

optional: *{}*
pattern: *./*.index*

Parameters

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Dec 08 17:06:48 2010

Modification date unknown

1.7.69 sync

Sync directories

Create this directory in sync with another directory

Commands

run Sync!

Parameters

ignore ignore these names (space separated list)

type: *{}*
default: ‘‘
optional: *True*

original The local directory to use as a source. If the target (based on what is in the source) does not exists, this directory is copied. If the target exists - only the configuration is copied, and all directory contents are left alone. If this parameter is omitted, the directory with the most recently changed moa configuration.

type: *string*
default: *{}*
optional: *True*

source The directory to keep in sync with

type: *string*

default: *{}*
optional: *False*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Thu, 30 Jun 2011 21:26:19

Modification date Thu, 30 Jun 2011 21:25:53

1.7.70 unittest

Template used in testing - has no other purpose

Commands

clean Remove all job data

prepare prepare for the unittest

run Prepare & Run

run delegates execution to: **prepare, run2**

run2 actually run

Filesets

input_1 Input file set 1

input_2 Input file set 2

type: *map*
source: *input_1*
category: *input*
optional: *{}*
pattern: *in2/*_2.txt*

output output files

type: *map*
source: *input_1*
category: *output*
optional: *{}*
pattern: *./*.out*

Parameters

test_string Test string values

type: *string*
default: *{}*
optional: *True*

miscellaneous

Backend ruff

Author Yogini Idnani, Mark Fiers

Creation date Wed Nov 25 17:06:48 2010

Modification date unknown

1.7.71 varscan

Varscan

Run VARSCAN to detect snps

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run *no help defined*

Parameters

extra_params location of varscan.pl, defaults to /usr/lib/perl5/site_perl/5.8.8/varscan.pl

type: *string*
default: ‘‘
optional: *True*

input_file Varscan input alignments file

type: *file*
default: ‘‘
optional: *True*

output_name Base name of the output files

type: *string*
default: *out*
optional: *True*

perl_file the varscan (perl) executable

type: *file*
default: ‘‘
optional: *True*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.72 vpcr

VPCR

Virtual PCR, based on Bowtie

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Predict the fragments that would be generated by a PCR

Parameters

bowtie_db Location of the bowtie database used for the vpcr

type: *file*
default: ‘‘
optional: *True*

insert_max maximum insert size for a vpcr fragment

type: *integer*
default: *10000*
optional: *True*

insert_min minimal insert size for a fragment

type: *integer*
default: *10*
optional: *True*

primer_1 First primer to use

type: *string*
default: ‘‘
optional: *False*

primer_2 Second primer to use

type: *string*
default: ‘‘
optional: *False*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.73 vpcr_list

Virtual PCR, based on Bowtie

Commands

clean Remove all job data, not the Moa job itself, note that this must be implemented by the template.

run Predict the fragments that would be generated by a PCR

Parameters

bowtie_db Location of the bowtie database used for the vpcr

type: *file*

default: ‘‘
optional: *False*

insert_max maximum insert size for a vpcr fragment

type: *integer*
default: *10000*
optional: *True*

insert_min minimal insert size for a fragment

type: *integer*
default: *10*
optional: *True*

primer_list List of primers to check

type: *file*
default: *{}*
optional: *False*

miscellaneous

Backend gnumake

Author Mark Fiers

Creation date Wed Nov 10 07:56:48 2010

Modification date Wed Nov 10 07:56:48 2010

1.7.74 wget

wget

Use WGET to download files. This template has two modi, one is set wget_mode to mirror data, in which case both wget_url and wget_pattern (default *) are used. The other modus is wget_mode=get, when one file defined by wget_url is downloaded. In the mirror mode it is possible to download only those files that are newer as the files already downloaded by using the wget_timestamp parameter

Commands

run Download

Parameters

pass Password for the remote site (note - this is not very safe, the password will be stored in plan text

type: *password*
default: ‘‘
optional: *True*

url The url of the file to download

type: *string*
default: *{}*
optional: *False*

user Username for the remote site

type: *string*
default: ‘‘
optional: *True*

miscellaneous

Backend ruff

Author Mark Fiers

Creation date Thu, 02 Jun 2011 10:22:31 +1200

Modification date Thu, 02 Jun 2011 10:22:53 +1200

1.8 Moa API

1.8.1 moa.actor

‘Simple’ wrapper around subprocess to execute code

`moa.actor.`**`getLastStderr`**(*job*)
Get the last stderr

`moa.actor.`**`getLastStdout`**(*job*)
Get the last stdout

`moa.actor.`**`getRecentOutDir`**(*job*)
Return the most recent output directory

moa.actor.**simpleRunner**(*wd*, *cl*, *conf={}*, ***kwargs*)

Don't think - just run - here & now

what does this function do? - put env in the environment - Execute the commandline (in cl) - store stdout & stderr in log files - return the rc

1.8.2 moa.commands

Handle Moa commands (i.e. anything that you can run as *moa COMMAND* on the commandline

1.8.3 moa.job

class moa.job.**Job**(*wd*)

Class defining a single job

Note - in the moa system, there can be only one current job - many operations try to access the job in sysConf

```
>>> wd = tempfile.mkdtemp()
>>> job = Job(wd)
>>> assert(isinstance(job, Job))
>>> assert(job.template.name == 'nojob')
```

checkCommands(*command*)

Check command, and rearrange if there are delegates.

```
>>> job = newTestJob('unittest')
>>> assert(job.template.commands.run.delegate == ['prepare', 'run2'])
>>> assert(job.checkCommands('run2') == ['run2'])
>>> assert(job.checkCommands('run') == ['prepare', 'run2'])
>>> assert(job.checkCommands('prepare') == ['prepare'])
```

Parameters **commands** (*list of strings*) – The list of commands to check

Returns The checked list of commands

Return type list of strings

checkConfDir()

Check if the configuration directory exists. If not create it.

```
>>> job = newTestJob('unittest')
>>> confdir = os.path.join(job.wd, '.moa')
>>> assert(os.path.exists(confdir))
>>> import shutil
>>> shutil.rmtree(confdir)
>>> assert(os.path.exists(confdir) == False)
>>> job.checkConfDir()
>>> assert(os.path.exists(confdir))
```

defineOptions(*parser*)

Set command line options - deferred to the backends

```
>>> job = newTestJob('unittest')
>>> import optparse
>>> parser = optparse.OptionParser()
>>> job.defineOptions(parser)
```

execute(*verbose=False*, *silent=False*)
Execute *command* in the context of this job. Execution is alwasy deferred to the backend

#Note: Uncertain how to test verbose & silent

Parameters

- **verbose** (*Boolean*) – output lots of data
- **silent** (*Boolean*) – output nothing

getFiles()
Return all moa files - i.e. all files crucial to this job.

hasCommand(*command*)
Check if this job defines a certain command

> **Warning:** THIS METHOD DOES NOT WORK PROPERLY YET

```
>>> job = newTestJob('unittest')
>>> assert(job.hasCommand('run'))
>>> assert(not job.hasCommand('dummy'))
```

initialize()
Initialize a new job in the current wd

isMoa()
Check if this is a Moa directory - Currently, this needs to be overridden #weird; uncertain if this ever gets called

loadBackend()
load the backend

loadTemplate()
Load the template for this job, based on what configuration can be found

prepare()
Give this job a chance to prepare for execution - deferred to the backend.

```
>>> job = newTestJob('unittest')
>>> job.prepare()
```

refreshTemplate()
Reload the template into the local .moa/template.d directory

```
>>> job = newTestJob('unittest')
>>> templateFile = os.path.join(job.confDir, 'template.d', 'unittest.jinja2')
>>> assert(os.path.exists(templateFile))
>>> os.unlink(templateFile)
>>> assert(not os.path.exists(templateFile))
>>> job.refreshTemplate()
>>> assert(os.path.exists(templateFile))
```

setTemplate(*name*, *provider=None*)
Set a new template for this job

```
>>> job = newTestJob('unittest')
>>> job.setTemplate('adhoc')
>>> afile = os.path.join(job.confDir, 'template.d', 'adhoc.mk')
>>> assert(os.path.exists(afile))
```

simpleExecute(*commandList*)
Just 'execute' a template call

moa.job.**newJob**(*wd*, *template*, *title*, *parameters=[]*, *provider=None*)
Create a new job in the wd and return the proper job object currently only makefile jobs are supported - later we'll scan the template, and instantiate the proper job type

```
>>> wd = tempfile.mkdtemp()
>>> job = newJob(wd, template='blast', title='test')
>>> assert(isinstance(job, Job))
>>> assert(job.template.name == 'blast')
>>> assert(job.conf.title == 'test')
```

Parameters

- **wd** – Directory to create this job in, note that this directory must already exists
- **template** (*String*) – Template name for this job
- **parameters** (*list of (key, value) tuples*) – A list of parameters to set for this job

Return type instance of moa.job.Job

moa.job.**newTestJob**(*template*, *title='Test job'*, *provider=None*)
for testing purposes - creates a temporary directory and uses that to instantiate a job. This function returns the job object created

```
>>> job = newTestJob(template = 'adhoc', title='test title')
>>> assert(isinstance(job, Job))
>>> assert(os.path.exists(job.wd))
>>> assert(job.conf.title == 'test title')
>>> assert(os.path.exists(os.path.join(job.wd, '.moa')))
>>> assert(os.path.exists(os.path.join(job.wd, '.moa', 'template')))
>>> assert(job.template.name == 'adhoc')
```

Returns the created job

Return type instance of moa.job.Job

1.8.4 moa.jobConf

moa job configuration

class moa.jobConf.**JobConf**(*job*)
to distinguish between attributes of this object & proper job configuration parameters

doNotCheck
these fields are not be type-checked

doNotSave
these fields are not to be saved

isEmpty()
Check if the config is empty is empty

isPrivate(*k*)
Is this a private variable? can be locally defined or in the template definition

keys()
return a dict with all known parameters and values, either defined in the job configuration of the template

load(*confFile*, *delta=None*)
Load a configuration file

Parameters **delta** – if a value appears to be a relative path, try to correct for this. Currently this only works for files that exist. i.e.

private
these fields are private (i.e. not to be displayed by default)

save()
Save the conf to disk

setRecursiveVar(*k*, *v*)
Register a recursive variable

1.8.5 moa.sysConf

Store Moa wide configuration

1.8.6 moa.ui

communicate information to the user

1.8.7 moa.utils

A set of random utilities used by Moa

moa.utils.**deprecated**(*func*)
Decorator function to flag a function as deprecated

Parameters **func** – any function

moa.utils.**flog**(*f*)
A simple logger - uses the `moa.logger` code to log the calling function. Use as a decorator:

```
@moa.utils.flog
def any_function(*args);
    ...
```

This is for debugging purposes (obviously)

Parameters **func** – Any python function

moa.utils.**getMoaBase**()
Return MOABASE - the directory where Moa is installed. This function also sets an environment variable *MOABASE*

```
>>> d = getMoaBase()
>>> assert(os.path.isdir(d))
>>> assert(os.path.isfile(os.path.join(d, 'README')))
>>> assert(os.path.isdir(os.path.join(d, 'lib')))
```

Return type string (path)

moa.utils.**getProcessInfo**(*pid*)
Return some info on a process

moa.utils.**getResource**(*what*)
Gets a data file from the moa package.

There are two possible locations where any resource could be, either three dirs up, or only one. This depends on if this a pypi (one dir up) package or the git package (three dirs up)

moa.utils.**listResource**(*what*)
List a directory

moa.utils.**moaDirOrExit**(*job*)
Check if the job contains a proper Moa job, if not, exit with an error message and a non-zero exit code.

Parameters **job** – An instance of moa.job.Job

moa.utils.**simple_decorator**(*decorator*)
This decorator can be used to turn simple functions into well-behaved decorators, so long as the decorators are fairly simple. If a decorator expects a function and returns a function (no descriptors), and if it doesn't modify function attributes or docstring, then it is eligible to use this. Simply apply @simple_decorator to your decorator and it will automatically preserve the docstring and function attributes of functions to which it is applied.

Note; I got this code from somehwere, but forgot where exactly. This seems the most likely source:

http://svn.navi.cx/misc/trunk/djblets/djblets/util/decorators.py

1.8.8 moa.template

moa.template

Store information on a template. This module is also responsible for retrieving template information.

moa.template.**initTemplate**(**args*, ***kwargs*)

moa.template.**installTemplate**(*wd*, *tName*, *provider=None*)
Initialize the template - this means - try to figure out where the template came from & copy the template files into *job/.moa/template* & *job/.moa/template.d/extra*.

Currently all templates come from the moa repository. In the future, multiple sources must be possible

```
>>> import tempfile
>>> wd = tempfile.mkdtemp()
>>> installTemplate(wd, 'adhoc')
>>> templateFile = os.path.join(wd, '.moa', 'template')
>>> adhocFile = os.path.join(wd, '.moa', 'template.d', 'adhoc.mk')
>>> assert(os.path.exists(templateFile))
>>> assert(os.path.exists(adhocFile))
```

moa.template.**refresh**(*wd*)

Refresh the template - try to find out what the template is from {{wd}}/.moa/template.d/meta. If that doesn't work, revert to the default template. If default is not specified - exit with an error

```
>>> import tempfile
>>> wd = tempfile.mkdtemp()
>>> installTemplate(wd, 'adhoc')
>>> templateFile = os.path.join(wd, '.moa', 'template')
>>> adhocFile = os.path.join(wd, '.moa', 'template.d', 'adhoc.mk')
>>> os.unlink(adhocFile)
>>> os.unlink(templateFile)
>>> assert(not os.path.exists(templateFile))
>>> assert(not os.path.exists(adhocFile))
>>> refresh(wd)
>>> assert(os.path.exists(templateFile))
>>> assert(os.path.exists(adhocFile))
```

moa.template.template

Store information on a template. This module is also responsible for retrieving template information.

class moa.template.template.**Template**(*templateFile*)

Template extends Yaco

getRaw()

Return a Yaco representation of the yaml-template, without any of this Template processing. This is really useful when processing a template that needs to be written back to disk

```
>>> import moa.job
>>> job = moa.job.newTestJob(template='adhoc')
>>> raw = job.template.getRaw()
>>> assert(isinstance(raw, Yaco.Yaco))
>>> assert(raw.has_key('parameters'))
```

1.8.9 moa.template.provider

moa.provider.core

Provides templates from the Moa package.

1.8.10 moa.backend

Gnumake

moa.backend.gnumake.**load**(*job*)
Create & return the GnuMake backend

Ruff

Ruffus (and Jinja) Backend

members

1.8.11 moa.plugin

adhoc - create jobs from adhoc bash code

moa.plugin.adhoc.**createAdhoc**(*job*)
Creates an adhoc job.

moa.plugin.adhoc.**createMap**(*job*)
Create a 'map' adhoc job.

There are a number of ways this command can be used:

```
$ moa map -t 'a title' -- echo 'define a command'
```

Anything after – will be the executable command. If omitted, Moa will query the user for a command.

Moa will also query the user for input & output files. An example session:

```
$ moa map -t 'something intelligent'
process:
> echo 'processing {{ input }} {{ output }}'
input:
> ../10.input/*.txt
output:
> ./*.out
```

Assuming you have a number of text files in the *../10/input/* directory, you will see, upon running:

```
processing ../10.input/test.01.txt ./test.01.out
processing ../10.input/test.02.txt ./test.02.out
processing ../10.input/test.03.txt ./test.03.out
...
```

moa.plugin.adhoc.**createReduce**(*job*)
Create a 'reduce' adhoc job.

There are a number of ways this command can be used:

```
$ moa reduce -t 'a title' -- echo 'define a command'
```

Anything after – will be the executable command. If omitted, Moa will query the user for a command.

Moa will also query the user for input & output files. An example session:

```
$ moa map -t 'something intelligent'
process:
> echo 'processing {{ input }} {{ output }}'
input:
> ../10.input/*.txt
output:
> ./*.out
```

Assuming you have a number of text files in the *../10/input/* directory, you will see, upon running:

```
processing ../10.input/test.01.txt ./test.01.out
processing ../10.input/test.02.txt ./test.02.out
processing ../10.input/test.03.txt ./test.03.out
...
```

moa.plugin.adhoc.**createSimple**(*job*)
Create a 'simple' adhoc job. Simple meaning that no in or output files are tracked.

There are a number of ways this command can be used:

```
moa simple -t 'a title' -- echo 'define a command'
```

Anything after – will be the executable command. Note that bash will attempt to process the command line. A safer method is:

```
moa simple -t 'a title'
```

Moa will query you for a command to execute (the parameter *process*).

moa.plugin.adhoc.**exclamate**(*job*)
Set the 'process' parameter to the last issued command. If no moa job exists, create a 'simple'job.

moa.plugin.adhoc.**exclamateInJob**(*job*)
Reuse the last issued command: set it as the 'process' parameters in the current job

moa.plugin.adhoc.**exclamateNoJob**(*job*)
Create a "simple" job & set the last command to the 'process' parameter

configure - Configure jobs

Control job configuration

moa.plugin.configure.**configSet**(*job*)
This command can be used in a number of ways:

```
moa set PARAMETER_NAME=PARAMETER_VALUE
moa set PARAMETER_NAME='PARAMETER VALUE WITH SPACES'
moa set PARAMETER_NAME
```

In the first two forms, moa sets the parameter *PARAMETER_NAME* to the *PARAMETER_VALUE*. In the latter form, Moa will present the user with a prompt to enter a value. Note that the first two forms the full command lines will be processed by bash, which can either create complications or prove very useful. Take care to escape variables that you do not want to be expandend and use single quotes where you can.

moa.plugin.configure.**configShow**(*job*)
Show all parameters know to this job. Parameters in **bold** are specifically configured for this job (as opposed to those parameters that are set to their default value). Parameters in red are not configured, but need to be for the template to operate. Parameters in blue are not configured either, but are optional.

moa.plugin.configure.**configUnset**(*job*)
Remove a configured parameter from this job. In the parameter was defined by the job template, it reverts back to the default value. If it was an ad-hoc parameter, it is lost from the configuration.

moa.plugin.configure.**hook_defineCommands**()
Set the moa commands for this plugin

moa.plugin.configure.**hook_git_finish_set**()
Execute just after setting a parameter

extraCommands - Pre & Post commands

Allow execution of a bash oneline before & after job completion

moa.plugin.extraCommands.**hook_postRun**()
If defined, execute the postCommand

moa.plugin.extraCommands.**hook_preRun**()
If defined, execute the precommand

moa.plugin.extraCommands.**runPostCommand**(*d*)
Execute the *postcommand*

moa.plugin.extraCommands.**runPreCommand**(*d*)
Execute the *precommand*

fileset - define sets of in&output files

moa.plugin.fileset.**hook_defineCommands**()
Set the moa commands for this plugin

moa.plugin.fileset.**hook_preparefilesets**()
prepare all filesets - can be called as a plugin hook - if need be

moa.plugin.fileset.**showFiles**(*job*)
moa files - Display discovered & inferred files for this job

Usage:

```
moa files
```

Display a list of all files discovered (for input & prerequisite type filesets) and inferred from these for map type filesets.

help - generate help

moa.plugin.help.**pager**(*template*, *templateData*)
render the template & send it to the pager

moa.plugin.help.**templateHelp**(*job*)

moa.plugin.help.**welcome**(*job*)
print a welcome message

info - Job information

Print info on Moa jobs and Moa

moa.plugin.info.**hook_defineCommands**()
Set the moa commands for this plugin

moa.plugin.info.**rawCommands**(*job*)
(private) **moa raw_commands** - Print a list of all known commands

Usage:

```
moa raw_commands
```

Print a list of known Moa commands, both global, plugin defined commands as template specified ones. This command is mainly used by software interacting with Moa.

moa.plugin.info.**rawParameters**(*job*)
(private) **moa raw_parameters** - Print out a list of all known parameters

Usage:

```
moa raw_parameters
```

print a list of all defined or known parameters

moa.plugin.info.**status**(*job*)
moa status - print out a short status status message

Usage:

```
moa status
```

moa.plugin.info.**version**(*job*)
moa version - Print the moa version number

lock - Lock/Unlock moa jobs

logger - Log Moa activity

moa.plugin.logger.**niceRunTime**(*d*)
Nice representation of the run time d is time duration string

moa.plugin.logger.**showLog**(*job*)
moa lcog - show a log of the most recent moa calls

Usage:

```
moa log [LINES]
```

Shows a log of moa commands executed. Only commands with an impact on the pipeline are logged, such as *moa run* & *moa set*. The number of log entries to display can be controlled with the optional LINES parameter.

logo - Print a big, in your face, moa logo

moa.plugin.logo.**hook_preRun**()
Print the logo just before a moa run

moaGit - maintain a git repository with job information

moa.plugin.moaGit.**gitadd**(*job*)
Add a job to the git repository

moa.plugin.moaGit.**gitlog**(*job*)
Print a log to screen

moa.plugin.moaGit.**hook_finish**()
Handle all git changes post execution - actually defer to plugin specific calls

moa.plugin.moaGit.**hook_prepare_3**()
Register a function for submitting a job

moautil - Some extra utilities - copy/move jobs

moa.plugin.moautil.**archive**(*job*)
Archive a job, or tree with jobs for later execution.

This command stores only those files that are necessary for execution of this job, that is: templates & configuration. In & output files, and any other file are ignored. An exception to this are all files that start with 'moa.'

Usage:

```
moa archive
```

or:

```
moa archive [NAME]
```

an archive name can be omitted when the command is issued in a directory with a moa job, in which case the name is derived from the *jobid* parameter

It is possible to run this command recursively with the *-r* parameter - in which case all (moa job containing) subdirectories are included in the archive.

As an alternative application you can specify the *–template*.

moa.plugin.moautil.**moacp**(*job*)
Copy a moa job, or a tree with jobs.

moa cp copies only those files defining a job: the template files and the job configuration. Additionaly, all files in the moa directory that start with *moa.* (for example *moa.description* are copied as well. Data and log files are not copied!

The command has two modes of operation. The first is:

```
moa cp 10.from 20.to
```

copies the moa job in 10.from to a newly created 20.to directory. If the *20.to* directory already exists, a new directory is created in *20.to/10.from*. As an shortcut one can use:

```
moa cp 10.from 20
```

in which case the job will be copied to the *20.from* directory.

If the source (*10.from*) directory is not a Moa job, the command exits with an error.

The second mode of operation is recursive copying:

```
moa cp -r 10.from 20.to
```

in which case all subdirectories under 10.from are traversed and copied - if a directory contains a Moa job.

::TODO.. Warn for changing file & dir links

moa.plugin.moautil.**moamv**(*job*)
Renumber or rename a moa job..

newjob - Instantiate new jobs

moa.plugin.newjob.**newJob**(*job*)
moa new

Usage:

```
moa new TEMPLATE_NAME -t 'a descriptive title'
```

archive - pack / unpack Moa trees

moa.plugin.pack.**pack**(*job*)

parameterCheck - check parameters

moa.plugin.parameterCheck.**hook_defineCommands**()
Define the parameters test commands

moa.plugin.parameterCheck.**hook_promptSnippet**()
Function used by the prompt plugin to generate snippets for inlusion in the prompt

prompt - Moa BASH prompt enhancer

moa.plugin.prompt.**hook_defineCommands**()
Set the moa commands for this plugin

status - Job Status

Possible job states:

- waiting - not yet executed
- running - is currently being executed
- success - finished succesfully

- error - finished with an error
- interrupted - manual interruption

moa.plugin.status.**hook_defineCommands**()
Set the moa commands for this plugin

moa.plugin.status.**kill**(*job*)
See if a job is running, if so - kill it

moa.plugin.status.**pause**(*job*)
pause a running job

moa.plugin.status.**resume**(*job*)
pause a running job

moa.plugin.status.**status**(*job*)
moa status - print out a status status message

Usage:

```
moa status
```

template - information on templates

moa.plugin.template.**dumpTemplate**(*job*)
moa template_dump - Show raw template information

Usage:

```
moa template_dump [TEMPLATE_NAME]
```

Show the raw template sysConf.

moa.plugin.template.**hook_defineCommands**()
Set the moa commands for this plugin

moa.plugin.template.**listTemplates**(*job*)
moa list - Print a list of all known templates

Usage:

```
moa list
moa list -l
```

Print a list of all templates known to this moa installation. If the option '-l' is used, a short description for each tempalte is printed as well.

moa.plugin.template.**refresh**(*job*)
Refresh the template - i.e. reload the template from the central repository.

moa.plugin.template.**template**(*job*)
moa template - Print the template name of the current job

Usage:

```
moa template
```

moa.plugin.template.**templateSet**(*job*)
moa template_set - set a template parameter.

This only works for top level template parameters

test - Run unittests

twit - Tweet results

Use twitter to send a message upon job completion

moa.plugin.twit.**postRun**(*job*)
Send a tweet out upon completing the default run

1.8.12 Yaco

Yaco provides a *dict* like structure that can be serialized to & from yaml. Yaco objects behave as dictionaries but also allow attribute access (loosely based on this **'recipe < http://code.activestate.com/recipes/473786/>'_**). Sublevel dictionaries are automatically converted to Yaco objects, allowing sublevel attribute access, for example:

```
>>> x = Yaco()
>>> x.test = 1
>>> x.sub.test = 2
>>> x.sub.test
2
```

Note that sub-dictionaries do not need to be initialized. This has as a consequence that requesting uninitialized items automatically return an empty Yaco object (inherited from a dictionary).

Yaco can be found in the Python package index and is also part of the Moa source distribution

Autogenerating keys

An important feature (or annoyance) of Yaco is the auto generation of keys that are not present (yet). For example:

```
>>> x = Yaco()
>>> x.a.b.c.d = 1
>>> assert(x.a.b.c.d == 1)
```

works - *a*, *b* and *c* are assumed to be *Yaco* dictionaries and d is give value *1*. This makes populating data structures easy.

It might also generate some confusion when querying for keys in the Yaco structure - if a key does not exists, it automatically comes back as an empy *dict* or *Yaco* object (renders as *{}*). This means that if it is easy to check if a certain 'branch' of a Yaco datastructure exists:

```
>>> x = Yaco()
>>> assert (not x.a.b)
```

but now the following works as well:

```
>>> assert(x.has_key('a'))
>>> assert(x.a.has_key('b'))
```

So, a safe way to test a data structure, without introducing extra branches is:

```
>>> x = Yaco()
>>> assert(not x.has_key('a'))
```

Todo: Need to find a more elegant way of testing without introducing data structures

class Yaco.**Yaco**(*data={}*)

Rather loosely based on http://code.activestate.com/recipes/473786/ (r1)

```
>>> v= Yaco()
>>> v.a = 1
>>> assert(v.a == 1)
>>> assert(v['a'] == 1)
>>> v= Yaco({'a':1})
>>> assert(v.a == 1)
>>> assert(v['a'] == 1)
```

get_data()

Prepare & parse data for export

```
>>> y = Yaco()
>>> y.a = 1
>>> y.b = 2
>>> y._c = 3
>>> assert(y._c == 3)
>>> d = y.get_data()
>>> assert(d.has_key('a') == True)
>>> assert(d.has_key('b') == True)
>>> assert(d.has_key('_c') == False)
>>> y._private = ['b']
>>> d = y.get_data()
>>> assert(d.has_key('a') == True)
>>> assert(d.has_key('b') == False)
>>> assert(d.has_key('_c') == False)
```

load(*from_file*)

Load this dict from_file

```
>>> import yaml
>>> import tempfile
>>> tf = tempfile.NamedTemporaryFile(delete=False)
>>> tf.write(yaml.dump({'a' : [1,2,3, [1,2,3, {'d' : 4}]], 'b': 4, 'c': '5'}))
>>> tf.close()
>>> y = Yaco()
>>> y.load(tf.name)
>>> assert(y.a[3][3].d == 4)
```

pretty()

Return data as a pprint.pformatted string

save(*to_file*, *doNotSave=[]*)

simple()

return a simplified representation of this Yaco struct - remove Yaco from the equation - and all object reference. Leave only bool, float, str, lists, tuples and dicts

```
>>> x = Yaco()
>>> x.y.z = 1
>>> assert(isinstance(x.y, Yaco))
```

```
>>> s = x.simple()
>>> assert(s['y']['z'] == 1)
>>> assert(isinstance(s['y'], dict))
>>> assert(not isinstance(s['y'], Yaco))
```

update(*data*)

```
>>> v = Yaco({'a' : [1,2,3,{'b' : 12}]})
>>> assert(v.a[3].b == 12)

>>> v = Yaco({'a' : [1,2,3,[1,{'b' : 12}]]})
>>> assert(v.a[3][1].b == 12)
```

1.8.13 fist

Filesets

Handle & manipulate sets of files

This module aims at providing classes to handle and manipulate sets of files. Two simple examples are a simple set containing one file (`fist.fistSingle`) or a *glob* based set of files (`fist.fistFileset`). A more complicated example is `fistMapset` that maps another fileset based on a pattern.

Each fileset inherits from *list* - hence fist filesets behave as lists.

Future work should allow the definition of remote filesets (for example over http or ssh).

Each fist class is istantiated with a url defining the file(set). In the case of `fist.fistFileset` this url contains a globbing characters:

```
fs = fist.fistFileset('/tmp/*,txt')
```

This fileset object contains a list with all **.txt* files in */tmp*. Subsequently it is possible to map this set

class `fist.`**fistCore**(*url*)
Core class for all fist classes

class `fist.`**fistFileset**(*url*)
Most basic set of files - handle a set of files described by a single URI with wildcards, for example:

```
* `*.txt`
* `../*.txt`
* `file:///home/name/data/*.txt`

>>> f = fistFileset('*.txt')
>>> assert(f.path=='.')
>>> assert(f.glob=='*.txt')
>>> assert(f.path=='.')
>>> assert(f.glob=='*.txt')
>>> f = fistFileset('/tmp')
>>> assert(f.path=='/tmp')
>>> assert(f.glob=='*')
>>> f = fistFileset('/tmp/*.txt')
>>> assert(f.path=='/tmp')
>>> assert(f.glob=='*.txt')
>>> f = fistFileset('../*.txt')
```

```
>>> assert(f.path=='..')
>>> assert(f.glob=='*.txt')
>>> f = fistFileset(os.path.join(wd, 'in', '*.txt'))
>>> f.resolve()
>>> assert(len(f) == 100)
>>> f = fistFileset(os.path.join(wd, 'in', 'in1*.txt'))
>>> f.resolve()
>>> assert(len(f) == 10)
>>> f = fistFileset('~/*')
>>> f.resolve()
>>> assert(len(f) > 0)
```

class fist.**fistMapset**(*url*)

fistMapset

Map set - map a fileset based on a target uri

```
>>> f = fistFileset(os.path.join(wd, 'in', '*'))
>>> f.resolve()
>>> assert(len(f) == 100)
>>> ##
>>> ## Null mapping
>>> ##
>>> m = fistMapset('*/*')
>>> m.resolve(f)
>>> assert(len(m) == 100)
>>> assert(os.path.join(wd, 'in/in18.txt') in m)
>>> ##
>>> ## simple folder mapping
>>> ##
>>> m = fistMapset('out/*')
>>> m.resolve(f)
>>> assert(len(m) == 100)
>>> assert('out/in18.txt' in m)
>>> ##
>>> ## simple folder mapping
>>> ##
>>> m = fistMapset('./*')
>>> m.resolve(f)
>>> assert(len(m) == 100)
>>> assert('./in18.txt' in m)
>>> ##
>>> ## simple folder & mapping & extension append
>>> ##
>>> m = fistMapset('out/*.out')
>>> m.resolve(f)
>>> assert(len(m) == 100)
>>> assert('out/in18.txt.out' in m)
>>> ##
>>> ## New from fileset - now with a pattern defining the extension
>>> ##
>>> f = fistFileset(os.path.join(wd, 'in', '*.txt'))
>>> f.resolve()
>>> ##
>>> ## extension mapping
>>> ##
>>> m = fistMapset('out/*.out')
>>> m.resolve(f)
```

```
>>> assert(len(m) == 100)
>>> assert('out/in18.out' in m)
>>> ##
>>> ## New from fileset - now with a pattern defining file glob &
>>> ## extension
>>> ##
>>> f = fistFileset(os.path.join(wd, 'in', 'in*.txt'))
>>> f.resolve()
>>> ##
>>> ## more complex filename mapping
>>> ##
>>> m = fistMapset('out/test*.out')
>>> m.resolve(f)
>>> assert(len(m) == 100)
>>> assert('out/test18.out' in m)
>>> ##
>>> ## mapping keeping the extension the same
>>> ##
>>> m = fistMapset('out/test*.txt')
>>> m.resolve(f)
>>> assert(len(m) == 100)
>>> assert('out/test18.txt' in m)
```

resolve(*mapFrom*)
Resolve the mapped set based on a input fileSet

resolver(*mapFrom*, *list*)
map all files in the incoming list

class fist.**fistSingle**(*url*)
Represents a single file

init()
Assuming the url is a single file

CHAPTER

TWO

MORE INFORMATION

- Browse the Moa source at Github.
- Download a pdf version of the manual.

CHAPTER

THREE

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX

f

m

y

INDEX

T

U

V

W

Y