

Genomon Pipeline Development Guideline

1. Source code structure

```
|-- db
|-- helpers
|-- perl
|-- python
|-- R
|-- resource
|-- samples
|-- shell
```

- perl, python, R, shell ディレクトリには、パイプラインで実際に使用されるスクリプトが配置されています。
- samples ディレクトリには、YAML ファイルのサンプルが配置されています。
- resource ディレクトリには、パイプラインで実行されるシェルスクリプトが、パイソンの文字列の形で格納されています。またその他の、文字リソースとして使用されている文字列が格納されています。
- helpers ディレクトリには、パイソンのプログラムが格納されています。

解析に使用するスクリプトを新たに加える場合は、このディレクトリにファイルを配置し、resource/genomon_rc.py の script_files に記述を加えてください。genomon_rc.py の script_files は、実行時に必要なスクリプトを result/.../script ディレクトリにコピーするファイルのリストを記述しています。

2. YAML files

サンプル YAML ファイルは、Github の Genomon の samples の中に、または、下記にあります。

/home/eigos/GenomonProj/test or Github:Genomon/samples

2つのタイプのファイルがあり、それぞれ別々に作成して指定することができます。

2.1. job description yaml file

Job description file には、解析のデータ、解析方法、などを記述します。

RNA-Seq データ解析の例)

RNA-Seq の例)

```

project_root:          /home/eigos/Data/GenomonProj/genomon
sample_name:          small_rna
sample_date:          20150409
analysis_date:        today

input_file_dir:        /home/eigos/Data/GenomonProj/input/small
input_file_type:       single_fastq
file_name:             d_100_*.fastq.bz2
file_ext:              .fastq.bz2
fastq_filter:          False # True or False

directory_permission:  owner    # owner, group, all

tasks:
    RNA:
        - tophat2
        - cufflinks
        - cummeRbund

#
#   HGC Shirokane super computer specific parameters
#
cmd_options:
    tophat2:            '-l 1job,s_vmem=2G,mem_req=2'
    cufflinks:          '-l 1job,s_vmem=2G,mem_req=2'
    cummeRbund:         '-l 1job,s_vmem=2G,mem_req=2'
qsub_cmd: "qsub -sync yes -now no {cmd_options} {cmd}"
drmaa_native: "-shell y {cmd_options}"

```

新たに機能を加える場合は、

- 1) 入力ファイルのフォーマットをしてい。
- 2) 'tasks:'に新たにパイプラインの機能を追加する。
- 3) 'cmd_options:'に必要なメモリ量を指定する。

2.2. analysis parameter yaml file

Analysis parameter file には、解析を実行する時のパラメーターを指定します。指定されたパラメーターは、解析のスクリプトに渡されます。

パラメータ指定の例)

```

fisher_mutation_call:
    min_depth:          9
    max_indel:          2
    max_distance:       5
    base_quality:       15
    map_quality:        30
    mismatch_rate:      0.07

```

上記のように、job description yaml file に指定した、それぞれの処理でのパラメータを指定できます。

実際にパイプラインの中で値を取得するには、下記のように utility function を使用します。

```
Geno.job.get_param( 'fisher_mutation_call', 'max_indel' )
```

2.3. system configuration file

3.Utility functions

3.1. パラメーター取得の関数

```
Geno.job.get_job( 'cmd_options' )
```

```
Geno.job.get_param( 'fisher_mutation_call', 'max_indel' )
```

```
Geno.conf.get( 'SOFTWARE', 'samtools' )
```

3.2. ジョブ実行の関数

```
Geno.RT.run_arrayjob(  
    shell_script_full_path,  
    Geno.job.get_job( 'cmd_options' )[ function_name ],  
    id_start = 1,  
    id_end = wgs_res.interval_num )
```

```
Geno.RT.runtask(  
    shell_script_full_path,  
    Geno.job.get_job( 'cmd_options' )[ function_name ] )
```

3.3. 実行結果の保存関数

```
Geno.status.save_status( function_name, output_file, return_code )
```

```
Geno.status.check_exit_status( process_name, output_file )
```

3.4. シェルスクリプトの作成

RNA-Seq tophat2 の例)

```
        shell_script_full_path =
make_script_file_name( function_name, Geno )
        shell_script_file = open( shell_script_full_path, 'w' )
        shell_script_file.write( rna_res.tophat2.format(
                                log = Geno.dir[ 'log' ],
                                ref_fa =
Geno.conf.get( 'REFERENCE', 'ref_fasta' ),
                                input_fastq = input_file,
                                output_file = output_file,
                                ref_gtf =
Geno.conf.get( 'REFERENCE', 'ref_gtf' ),
                                bowtie2_database =
Geno.conf.get( 'REFERENCE', 'bowtie2_db' ),
                                bowtie_path = bowtie_path,
                                tophat2 =
Geno.conf.get( 'SOFTWARE', 'tophat2' )
                                )
                                )
        shell_script_file.close()
```

4. Pipeline example

5. Execution and Debug

実行方法 1)

```
run.sh    [Job description YAML file]\
          [Analysis parameter YAML file]
```

実行方法 2) python debugger を使う。

```
run.sh    [Job description YAML file]\
          [Analysis parameter YAML file]\
          '-m pdb'
```

パラメーター)

```
$ >python ./genomon.py
usage: genomon.py [-h] [--verbose [VERBOSE]] [--version] [-L FILE]
                  [-T JOBNAME] [-j N] [--use_threads] [-n]
                  [--touch_files_only] [--recreate_database]
                  [--checksum_file_name FILE] [--flowchart FILE]
                  [--key_legend_in_graph] [--draw_graph_horizontally]
```

```
 [--flowchart_format FORMAT] [--forced_tasks JOBNAME]
 [-s CONFIG_FILE] [-f JOB_FILE] [-p PARAM_FILE] [-m] [-d]
 [-l]
```

Genome Analysis Pipeline

optional arguments:

-h, --help show this help message and exit

Common options:

--verbose [VERBOSE], -v [VERBOSE]
Print more verbose messages for each additional
verbose level.
--version show program's version number and exit
-L FILE, --log_file FILE
Name and path of log file

pipeline arguments:

-T JOBNAME, --target_tasks JOBNAME
Target task(s) of pipeline.
-j N, --jobs N Allow N jobs (commands) to run simultaneously.
--use_threads Use multiple threads rather than processes. Needs
--jobs N with N > 1
-n, --just_print Don't actually run any commands; just print the
pipeline.
--touch_files_only Don't actually run the pipeline; just 'touch' the
output for each task to make them appear up to date.
--recreate_database Don't actually run the pipeline; just recreate the
checksum database.
--checksum_file_name FILE
Path of the checksum file.
--flowchart FILE Don't run any commands; just print pipeline as a
flowchart.
--key_legend_in_graph
Print out legend and key for dependency graph.
--draw_graph_horizontally
Draw horizontal dependency graph.
--flowchart_format FORMAT
format of dependency graph file. Can be 'pdf', 'svg',
'svgz' (Structured Vector Graphics), 'pdf', 'png',
'jpg' (bitmap graphics) etc
--forced_tasks JOBNAME
Task(s) which will be included even if they are up to
date.

genomon:

Genomon options

-s CONFIG_FILE, --config_file CONFIG_FILE
Genomon pipeline configuration file
-f JOB_FILE, --job_file JOB_FILE
Genomon pipeline job configuration file
-p PARAM_FILE, --param_file PARAM_FILE
Genomon pipeline analysis parameter file
-m, --mpi Use MPI job submission
-d, --drmaa Use DRMAA job submission
-l, --abpath Use absolute path in scripts

genomon.py を実行するには、下記の環境変数を指定しなければなりません。

export

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/package/python2.7/current/lib
export
PYTHONPATH=$PYTHONPATH:/home/w3varann/.local/lib/python2.7/site-packages
```

run.sh で使用されているパラメーターは、下記に示したとおりです。--config_file、--job_file、--param_file は、必ず指定しなければなりません。

```
    --config_file
/home/eigos/Data/GenomonProj/test/genomon.cfg \
    --job_file $1 \
    --param_file $2 \
    --jobs 10 \
    --verbose 10
```