Quick user guide

NGOMICS-WF, a Bioinformatic Workflow Tool for Batch Omics Data Analysis using Linux Cluster or Computer Environment

Contact: Weizhong Li, wli@jcvi.org or liwz@sdsc.edu

NGOMICS-wf is a workflow tool for batch omics data analysis. This workflow tool has been used in my group in several projects for many years. This tool can assist researchers with basic experiences in Linux system and Linux computer cluster to configure and run workflows for various types of omics data sets. The software package is available from https://github.com/weizhongli/ngomicswf. In addition to the workflow tool, several pre-configured workflows that were tested and published are also available for analysis of metagenomic, metatranscriptomic, RNA-seq and 16S data.

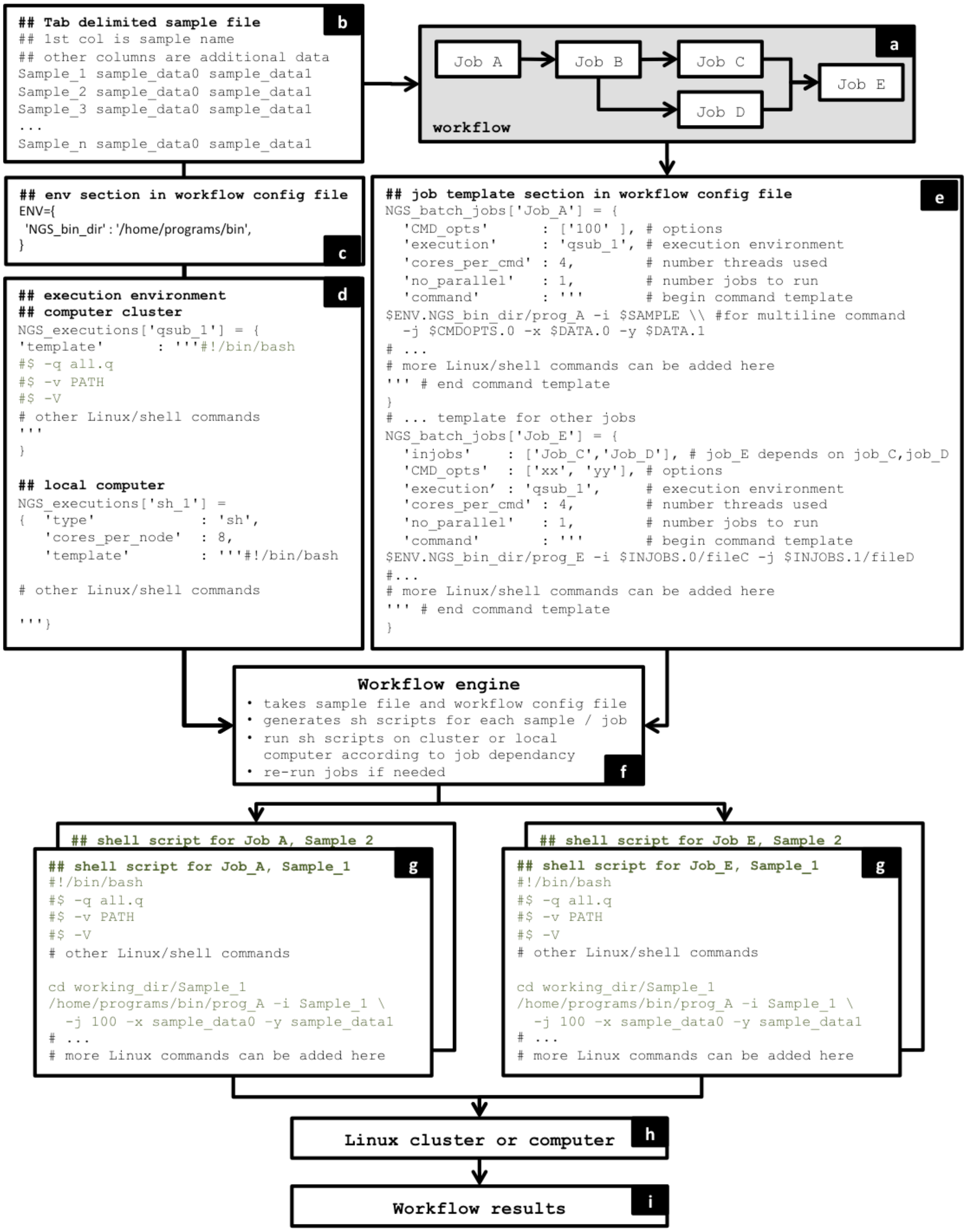
Installation and requirement

NGOMICS-wf is written in Python. Python 2.7 is required. No additional Python packages are needed. The tool works under Linux standalone computer or Linux cluster with a queue system. It has been tested with Sun Grid Engine (SGE), now is Open Grid Engine (OGE). But lots of other queue systems have very similar set of commands as SGE, so NGOMICS-wf should work with minor configuration.

Configure and run workflow

The NGOMICS-wf workflow tool is illustrated in the Figure 1. Given an example workflow or pipeline (Fig. 1a) and a list of samples that need to run through this workflow. Three steps are needed before running the workflow.

1. Create a sample file (Fig. 1b). Each line in this file contain a sample, starting with sample name at beginning, optionally followed by a list of parameters for the sample, all delimited by space or TAB.
2. Create a wokflow config file (Fig. 1c, 1d, 1e). This file defines some local variables and setting, compute resources and the actual command to run the workflow. This config file needs to be written in Python, but you don't have to know Python to write the config file. You can use the pre-configured workflow config files as template to implement yours.
    a. Local variables (Fig. 1c). As shown in the figure, you can define a set of local variables, which can be used in actual job scripts.
    b. Local computer resources (Fig. 1d). You can define multiple computing environments to run your jobs in the pipeline. Computing environment can be on local computer or on computer cluster. You can define separate computing environments for different queues on a computer cluster.
    c. Job commands (Fig. 1e). For each job in the pipeline, you will need to define a shell script template and to declare the input, dependent jobs, computing resources and so on. There are some special variables (the ones with $ sign) in the job command template. These variables will be replaced by local variables, job parameters, job name, sample name etc according to a set of rules. Based on the template, the workflow engine (Fig. 1f) will generate shell script for each job sample pair (Fig. 1g). Rules for variables replacement are:



```
## Tab delimited sample file
## 1st col is sample name
## other columns are additional data
Sample_1 sample_data0 sample_data1
Sample_2 sample_data0 sample_data1
Sample_3 sample_data0 sample_data1
...
Sample_n sample_data0 sample_data1
```

```
## env section in workflow config file
ENV={
 'NGS_bin_dir' : '/home/programs/bin',
}
```

```
## execution environment
## computer cluster
NGS_executions['qsub_1'] = {
'template'      : '''#!/bin/bash
#$ -q all.q
#$ -v PATH
#$ -V
# other Linux/shell commands
'''
}

## local computer
NGS_executions['sh_1'] =
{  'type'             : 'sh',
   'cores_per_node'   : 8,
   'template'         : '''#!/bin/bash

# other Linux/shell commands

'''}
```

```
## job template section in workflow config file
NGS_batch_jobs['Job_A'] = {
  'CMD_opts'      : ['100' ], # options
  'execution'     : 'qsub_1', # execution environment
  'cores_per_cmd' : 4,        # number threads used
  'no_parallel'   : 1,        # number jobs to run
  'command'       : '''      # begin command template
$ENV.NGS_bin_dir/prog_A -i $SAMPLE \\ #for multiline command
  -j $CMDOPTS.0 -x $DATA.0 -y $DATA.1
# ...
# more Linux/shell commands can be added here
''' # end command template
}
# ... template for other jobs
NGS_batch_jobs['Job_E'] = {
  'injobs'    : ['Job_C','Job_D'], # job_E depends on job_C,job_D
  'CMD_opts'  : ['xx', 'yy'], # options
  'execution' : 'qsub_1',     # execution environment
  'cores_per_cmd' : 4,        # number threads used
  'no_parallel'   : 1,        # number jobs to run
  'command'       : '''      # begin command template
$ENV.NGS_bin_dir/prog_E -i $INJOBS.0/fileC -j $INJOBS.1/fileD
#...
# more Linux/shell commands can be added here
''' # end command template
}
```

Workflow engine
- takes sample file and workflow config file
- generates sh scripts for each sample / job
- run sh scripts on cluster or local computer according to job dependancy
- re-run jobs if needed

```
## shell script for Job A, Sample 2
```

```
## shell script for Job_A, Sample_1
#!/bin/bash
#$ -q all.q
#$ -v PATH
#$ -V
# other Linux/shell commands

cd working_dir/Sample_1
/home/programs/bin/prog_A -i Sample_1 \
  -j 100 -x sample_data0 -y sample_data1
# ...
# more Linux commands can be added here
```

```
## shell script for Job E, Sample 2
```

```
## shell script for Job_E, Sample_1
#!/bin/bash
#$ -q all.q
#$ -v PATH
#$ -V
# other Linux/shell commands

cd working_dir/Sample_1
/home/programs/bin/prog_A -i Sample_1 \
  -j 100 -x sample_data0 -y sample_data1
# ...
# more Linux commands can be added here
```

Linux cluster or computer

Workflow results

Figure 1. Schema of ngomics-WF

i. $SAMPLE will be replaced by sample name
ii. $DATA.0 will be replaced by first parameter in sample file, and so on
iii. $JOB will be replaced by job name
iv. $INJOBS.0 will be replaced by the name of the 1st dependent job in injobs entry and so on
v. $CMD_opts.0 will be replaced by the 1st item in the CMD_opts entry, and so on
vi. $ENV.var_name will be replaced by variables defined in Fig. 1c

3. Setup a working directory. In the working directory, make separate directories for each sample, using sample name as the directory name. Put the input files (e.g. fastq files) inside each sample directory. So, you have a directory tree like this:

```
_working_dir____
                |___ sample_1 ___
                |                |___R1.fq.gz
                |                |___R2.fq.gz
                |
                |___ sample_2 ___
                |                |___R1.fq.gz
                |                |___R2.fq.gz
                |
                |___ other_samples_directories
                |
                |
                |___ NGS-samples (sample file)
```

After these three steps, you can run the workflow with command line within the working directory:
> nohup NG-Omics-WF.py -s NGS-samples -i NGS-WF-config.py &

Here

- NGS-samples is the filename of sample file (Fig. 1b)
- NGS-WF-config.py is the filename of workflow config file (Fig. 1c, d, e)

You can also run this command first
> NG-Omics-WF.py -s NGS-samples -i NGS-WF-config.py -J write-sh

Here, this command let workflow only write the shell scripts without executing them. The scripts will be in WF-sh directory. So you can check whether these shell scripts are write.

Additional details for configuration

Queue config: For execution on computer cluster, in workflow config file you can define the parameters of a queue (Fig. 1d and box below).

- qsub_exe: the command to call qsub, e.g. 'qsub' or '/opt/bin/qsub -myopt'
- command_name_opt etc: for SGE and many other queuing system. The opts for job name, error output and std output are '-N', '-e', '-o'. Usually you don't need to change
- template: you can put additional options allowed by your queuing system to define the queue name, env variables etc.

```
NGS_executions['qsub_1'] = {
  'type'                : 'qsub-pe',
  'qsub_exe'            : 'qsub',
  'cores_per_node'      : 32,
  'number_nodes'        : 64,
  'user'                : 'user_name',
  'command_name_opt'    : '-N',
  'command_err_opt'     : '-e',
  'command_out_opt'     : '-o',
  'template'            : '''#!/bin/bash
#$ -q all.q
#$ -v PATH
#$ -V

'''
}
```

Job command: As introduced earlier, each job in the pipeline is configured as in Fig. 1e and box below. And special variables (starting with $ sign) will be replaced, as described before. Here are some additional details:

- cores_per_cmd: this is the number of cores this job needs, if you run a program with 4 threads and this should be 4
- CMD_opts: you can define a list of parameters here. You can certainly put parameters in the template, but put in CMD_opts array will allow you to change them at run-time, with additional -T or -t option to NG-Omics-WF.py (see next section)
- No_parallel: how many instances you want to submit this job. If this value is >1, then the workflow engine will submit the script multiple times. Then you need to have logic built in your command so that multiple scripts can run simultaneously.

```
NGS_batch_jobs['Job_A'] = {
   'CMD_opts'      : ['100' ], # options
   'execution'     : 'qsub_1', # execution environment
   'cores_per_cmd' : 4,        # number threads used
   'no_parallel'   : 1,        # number jobs to run
   'command'       : '''       # begin command template
 $ENV.NGS_bin_dir/prog_A -i $SAMPLE \\ #for multiline command
   -j $CMDOPTS.0 -x $DATA.0 -y $DATA.1
 # ...
 # more Linux/shell commands can be added here
 ''' # end command template
 }
 # ... template for other jobs
 NGS_batch_jobs['Job_E'] = {
   'injobs'    : ['Job_C','Job_D'], # job_E depends on job_C,job_D
   'CMD_opts'  : ['xx', 'yy'], # options
   'execution’ : 'qsub_1',     # execution environment
   'cores_per_cmd' : 4,        # number threads used
   'no_parallel'   : 1,        # number jobs to run
   'command'       : '''       # begin command template
 $ENV.NGS_bin_dir/prog_E -i $INJOBS.0/fileC -j $INJOBS.1/fileD
 #...
 # more Linux/shell commands can be added here
 ''' # end command template
 }
```

Run-time parameter file: if you want to change command line parameters within each job, without change the workflow config file. You can use CMD_opts to define a list of default parameters in the workflow config file. For example:

```
NGS_batch_jobs['Job_E'] = {
  'injobs'    : ['Job_C','Job_D'],
  'CMD_opts'  : ['100', '200'],
  'command'   : '''       # begin command template

command -i $CMDOPTS.0 -j $CMDOPTS.1

 ''' # end command template
 }
```

Then the workflow engine will translate the command to

    Command -i 100 -j 200

Here, if you want to use different parameters without changing the workflow config file you can make another parameter file like below, that use 111 and 222 to replace 100 and 200.

```
## TAB delimited parameter file
## 1st col is the job name
## other columns are parameters to replace CMD_opts list
Job_E 111 222
```

Then run the workflow with -T parameter_file

Pre-configured workflows: These are examples (workflow-example folder) that can be used as template to implement your workflows.

More parameters: more up-to-date help information and usage can be printed with

    ./NGS-WF-config.py -h

Please refer to the full user's guide (being developed) for more advanced usage.