RNA-Seq Analysis Snakemake Workflow (RASflow)

Xiaokang Zhang
Email: zhxiaokang@gmail.com

December 9, 2019

Contents

1 Introduction **1**

2 Installation **2**
2.1 Install Miniconda 2
2.2 Install RASflow 2

3 Customized setting up **3**
3.1 Description of your data 3
3.2 Setting parameters for quality control and trimming 3
3.3 Setting parameters for quantification of transcripts or genes . . . 3
3.3.1 Mapping to transcriptome 3
3.3.2 Mapping to genome 3
3.4 Setting parameters for Differential Expression Analysis 4
3.5 Setting parameters for visualization 5

4 Run the analysis **5**
4.1 Quality control and trimming 5
4.2 Quantification of transcripts or genes 5
4.3 Differential Expression Analysis (DEA) 6
4.4 Visualization of DEA results 6

5 Visualize the workflow **6**

1 Introduction

RASflow (RNA-Seq Analysis Snakemake Workflow) allows a customized automatic RNA-Seq analysis requiring minimum programming knowledge.

This workflow is managed by Snakemake. As stated on its webpage:

The Snakemake workflow management system is a tool to create reproducible and scalable data analyses. Workflows are described via a human readable, Python based language. They can be seamlessly scaled to server, cluster, grid and cloud environments, without the need to modify the workflow definition. Finally, Snakemake workflows can entail a description of required software, which will be automatically deployed to any execution environment.

By setting up the environment, all the required tools will be installed with only one command. Besides of that, since the versions of all tools can be specified, the work is ensured to be reproducible. But many essential tools such samtools are not available for Windows on Anaconda Cloud, so this workflow only works on Linux and macOS now.

2 Installation

2.1 Install Miniconda

Download *Miniconda* installer from here: https://docs.conda.io/en/latest/miniconda.html
Install it to your laptop or server.

2.2 Install RASflow

Download the repository from here: https://github.com/zhxiaokang/RASflow/archive/master.zip and unzip it.

If you have Git installed on your machine, a better way to download the repository is to use git:

```
$ git clone https://github.com/zhxiaokang/RASflow.git
```

With that, you can easily keep yourself updated with the latest version using the command `git pull`.

Open your terminal in the directory of the repository *RASflow* and run the following command to set up the environment:

```
$ conda env create -n rasflow -f env.yaml
```

Then activate the environment. For Linux and Mac:

```
$ conda activate rasflow
```

For Windows:
Very sorry for Windows users, but some of required tools are not available for Windows, so RASflow currently can not be run on Windows.

3 Customized setting up

3.1 Description of your data

Modify the metafile describing the data configs/metadata.tsv

There are two columns: "sample" and "group". Fill the 1st column with the sample ID and specify the corresponding group in the 2nd column. Make sure that the sample ID here agrees with the file name of the fastq files.

3.2 Setting parameters for quality control and trimming

You want to check the quality of the RNA-Seq data you have before start your analysis. Indicate where the fastq files are and where you want to store the output in the config file configs/config_quality_control.yaml.

Read the report. If the quality of your data is OK, then you can continue with alignment and feature count. Otherwise, trimming is needed. The config file for trimming is configs/config_trim.yaml. One important parameter is "END" which indicates whether the sequencing is single-end or paired-end.

3.3 Setting parameters for quantification of transcripts or genes

The next step of the analysis workflow is to get the expression table which can either be transcripts or genes.

3.3.1 Mapping to transcriptome

If you choose to map to the reference transcriptome, then the config file is configs/config_quantify_trans.yaml.

Modify the parameters accordingly. Most of them are the directories where you store your FASTQ files, the meta file describing the experiment setting up, the output directory, the reference transcriptome file. If you don't have the reference transcriptome file at hand, you can download them from public databases, such as ENSEMBL: https://www.ensembl.org/info/data/ftp/index.html. Specify whether your sequencing data is single-end or pair-end. Adjust the number of cores ("NCORE") you want to use for the workflow based on your laptop or the server you are using. If you do not know how many cores are available on the machine, you can use the following command to get the answer:

```
$ getconf _NPROCESSORS_ONLN
```

3.3.2 Mapping to genome

If you choose to map to the reference genome, then the config file is configs/config_align_count_genome.yaml.

Refer to the previous section, the setting of the config file is very similar to the one of transcriptome. But instead of using transcriptome as mapping reference,

you need to provide the genome and also annotation file which can also be found on ENSEMBL. One thing to double check is the parameter "ATTRIBUTE" which can be found in the annotation file. It is usually "gene_id", but there may be special cases. The aligner used by RASflow is HISAT2 which requires low memory from the machine. The default tool for feature count is featureCounts, but you can also choose htseq-count.

3.4 Setting parameters for Differential Expression Analysis

If you want to do Differential Expression Analysis (DEA) after quantification, you need to continue with the next step of the workflow: DEA. The config file for this step is configs/config_dea_trans.yaml or configs/config_dea_genome.yaml, which depends on the previous step whether you mapped the sequences to transcriptome or genome.

These two config files are almost the same, except that in config_dea_trans.yaml, you need to specify whether you want to do gene-level DEA. If you are using homemade transcriptome reference (for example de novo assembly), you can of course only do transcript-level DEA; if you use the transcriptome from a database other than ENSEMBL, RASflow doesn't provide gene-level DEA for now. So the parameter "GENE_LEVEL" can only be set to "FALSE". If you use a transcriptome from ENSEMBL, you can choose to do both transcript- and gene-level DEA. "GENE_LEVEL" therefore needs to be set to "TRUE". You also need to set the parameter "EnsemblDataSet" which specifies the ENSEMBL data set for your organism. You can find this in the look-up table config/EnsemblDataSet_look_up_table.csv.

RASflow uses edgeR to do DEA since it is capable of doing paired test. Make sure to specify the parameter "PAIR" as "TRUE" or "FALSE" which depends on the experiment design. If you are going to do paired test, also make sure that you indicate the "subject" which the sample belongs to in the meta file metadata.tsv; if you are going to do a simple test, just ignore the column "subject" since the workflow will not use that information anyway.

RASflow only considers pairwise comparison, meaning that in each DEA, it only compares two groups. If you have more than two groups, say you have one control group (named "control"), but two treated groups (named "treat1" and "treat2"), divide them into two pairwise comparisons: control group versus each of the two treated groups. The the parameter "CONTROL" will be ["control", "control"], and the parameter "TREAT" will be ["treat1", "treat2"]. Make sure that the group names agree with what you fill in the column "group" in the meta file.

You can also specify whether filtering is required before doing differential analysis. If the choice is "TRUE", only the transcripts/genes who have at least one "tpm" above the threshold across all samples in two compared groups will be kept.

3.5 Setting parameters for visualization

After DEA, you can visualize the results by Volcano plot and Heatmap. Set the parameters in the config file configs/config_visualize.yaml

The "DEAFILE" includes the statistics for all transcripts or genes and the "DEGFILE" only includes the significantly differentially expressed transcripts or genes. "NORMFILES" are the files of normalized count tables, for both control and treat groups. Specify the names of control and treat groups with "NAME_CONTROL" and "NAME_TREAT".

4 Run the analysis

4.1 Quality control and trimming

Make sure that you have activated the environment as stated in section Install RASflow. If you have done that correctly, you should see "(rasflow)" at the beginning of your command line. Then you start with workflow/quality_control.rules:

```
$ nice -5 snakemake -s workflow/quality_control.rules 2>&1 | tee logs/
    log_quality_control.log
```

- `nice -5` makes you nice to other users with whom you share the server. The smaller the number is, the higher priority you are requiring (ranging from -20 to 19 and the default is 0 if you do not use "nice")
- `snakemake -s` specifies the Snakemake file you want to execute
- `2>&1 | tee logs/log_quality_control.log` allows the log information to be printed on the screen and also stored in a log file for your convenience to check the processing history

Check the report from quality control. If the quality if OK, you can skip the trimming and continue with quantification; if not, do trimming firstly:

```
$ nice -5 snakemake -s workflow/trim.rules 2>&1 | tee logs/log_trim.log
```

4.2 Quantification of transcripts or genes

The useful file for this step is workflow/quantify_trans.rules or workflow/align_count_genome.rules, depending on whether you want to map the sequences to transcriptome or genome.

Before starting the program, edit the section of rule "getReads" in that Snakemake file. RASflow assumes that you have your fastq files locally, if not, adjust the commands in this rule to tell RASflow where to find the fastq files. If trimming is done, also make sure that the naming format of the fastq files is correct in rule "getReads".

When you are ready, start the analysis using the following command:

```
$ nice -5 snakemake -s workflow/quantify_trans.rules 2>&1 | tee logs/
    log_quantify_trans.log
```

or

```
$ nice -5 snakemake -s workflow/align_count_genome.rules 2>&1 | tee logs/
    log_align_count_genome.log
```

4.3 Differential Expression Analysis (DEA)

When the previous workflow has been finished, you will get the count matrices with which DEA can be done. The useful file for this step is workflow/dea_trans.rules or workflow/dea_genome.rules. Start DEA with the command:

```
$ nice -5 snakemake -s workflow/dea_trans.rules 2>&1 | tee logs/
    log_dea_trans.log
```

or

```
$ nice -5 snakemake -s workflow/dea_genome.rules 2>&1 | tee logs/
    log_dea_genome.log
```

4.4 Visualization of DEA results

After setting up the config file, simply run the following command to visualize the results of DEA:

```
$ nice -5 snakemake -s workflow/visualize.rules 2>&1 | tee logs/
    log_visualize.log
```

5 Visualize the workflow

This section will tell you how to visualize your workflow using a directed acyclic graph (DAG). You can visualize the gerenal workflow (using `--rulegraph`) or with details (using `--dag`):

```
$ nice -5 snakemake --rulegraph -s workflow/align_count_genome.rules |
    dot -Tsvg > workflow/ruleDag_align_count_genome.svg

$ nice -5 snakemake --dag -s workflow/align_count_genome.rules | dot -
    Tsvg > workflow/dag_align_count_genome.svg
```