# RNA-Seq Analysis Snakemake Workflow (RASflow)

Xiaokang Zhang
Email: zhxiaokang@gmail.com

January 14, 2020

## Contents

# 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 are specified as is shown in the environment configuration file, the work is ensured to be reproducible.

# 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

Some of the required tools are not available for Windows, so RASflow currently can only be run on Linux and Mac. For Windows users, we highly recommend Windows Subsystem for Linux (WSL). It has been tested that RASflow can run smoothly on WSL. To install it, refer to https://docs.microsoft.com/en-us/windows/wsl/install-win10

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:

```
$ conda activate rasflow
```

# 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 Customize the workflow

The configuration file for RASflow is configs/config_main.yaml, where you will find all the settings for the whole workflow. To start, you are asked to give a name of the analysis (”PROJECT”), which will be extremely helpful when you need to do multiple analyses with different settings or different data sets.

In the next section, you can customize what sub-workflows you need RASflow to run for you.

Following that, you are required to set some ‘‘Shared parameters for some or all of the sub-workflows”.

- `KEY` If your fastq files are stored remotely instead of locally, normally they can be accessed with help of a key file. If they are stored locally, just leave it blank.
- `READSPATH` indicates where the fastq files are located.
- `METAFILE` is the metafile you made in last step.
- `END` specifies whether the sequencing is paired-end or single-end. If it is paired-end, there should be two fastq files for each sample.
- `NCORE` tells RASflow how many cores it can use. Use command ‘‘getconf _NPROCESSORS_ONLN” to check the number of cores/CPU on your machine.
- `OUTPUTPATH` is the directory you want to store the intermediate outputs. They are normally big files such as BAM files.
- `FINALOUTPUT` is the directory you want to store the final outputs which are most important and normally small files.

## 3.3 Setting parameters for quality control and trimming

You want to check the quality of the raw RNA-Seq data before you start the analysis. Set ‘‘QC” to ‘‘yes”, and all the other parameters needed to be set have already been set in the shared section.

Read the quality control report (output/[PROJECT]/fastqc/report_quality_control.html). If the quality of your data is OK, then you can continue with quantification or alignment and feature count, therefore set ‘‘TRIMMED” to ‘‘no”; Otherwise,

trimming is needed so set ‘‘TRIMMED” to ‘‘yes”. In both cases, remember to set ‘‘QC” to ‘‘no”.

After trimming, the trimmed read files will be stored under data/output/trim/ and they will be used for downstream analysis such as alignment. Another quality control will be run on the trimmed reads and the report can be found at output/[PROJECT]/fastqc_after_trimming/report_quality_control_after_trimming.html. Theoretically the quality should be better than the raw reads.

## 3.4 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.

If ‘‘REFERENCE” was set to ‘‘transcriptome”, you need to modify the parameters in section ‘‘Configuration for quantification using transcriptome”. On the other hand, if ‘‘REFERENCE” was set to ‘‘genome”, you need to modify the parameters in section ‘‘Configuration for alignment to genome and feature count”.

### 3.4.1 Mapping to transcriptome

If you choose to map to the reference transcriptome, specify 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.

### 3.4.2 Mapping to genome

If you choose to map the reads to the reference genome, 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.

## 3.5 Setting parameters for Differential Expression Analysis

If you want to do Differential Expression Analysis (DEA) after quantification, set the parameter ‘‘DEA” to ‘‘yes”. Besides of that, there are several more parameters to be set in section ‘‘Configuration for DEA”:

- `PAIR` indicates whether the experiment was designed in a pair-wise way. If yes, it’s better to do paired test during DEA, to make full use of this information. But 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.

- `CONTROL & TREAT` 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.

- `FILTER: yesOrNo & cpm` 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 "cpm" above the threshold across all samples in two compared groups will be kept.

Above parameters are for both transcriptome and genome used as mapping reference. If transcriptome was used as reference, the following two parameters also need to be specified.

- `GENE_LEVEL` specifies 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".

- `EnsemblDataSet` If "GENE_LEVEL" was 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.

### 3.6 Setting parameters for visualization

After DEA, you can visualize the results by Volcano plot and Heatmap. Set the parameter "VISUALIZE" to "yes".

## 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. With "QC" set to "yes" in config file, you start the program with:

```
$ python main.py
```

After quality control is done, you will be asked to turn it off. To do that, simply set "QC" to "no" in config file. You will also be asked to check the QC report which can be found at output/[PROJECT]/fastqc. There are the reports for each sample generated by FastQC). There is a report for each sample and here are the quality control reports of the example data given in the repository: output/test/fastqc. Explanation about how to interpret the report can be found here. The reports are also summarized by MultiQC [1]. The summary report can be found at output/[PROJECT]/fastqc/report_quality_control.html.

If the quality if OK, you can skip the trimming and continue with quantification; if not, do trimming firstly. To do that, set "TRIMMED" to "yes" and re-run the program.

## 4.2 Quantification of transcripts or genes

Make sure that you set the mapping reference correctly in config file ("REFERENCE"). "QC" should be set to "no", and "TRIMMED" should be set accordingly. When it's all set, re-run the program by:

```
$ python main.py
```

If the transcriptome was used as reference, for each sample, you will find the quantification of transcripts output/[PROJECT]/trans/quant. The quantification file is named `quant.sf` which contains 5 columns: `Name, Length, EffectiveLength, TPM, NumReads`. The quantification is done by Salmon [2], and the explanation can be found here: Salmon. A summary of the quantification quality is give by MultiQC: output/[PROJECT]/trans/report_quantify.html.

If the genome was used as reference, for each sample, you will find the raw counts of genes output/[PROJECT]/genome/countFile. The count file is named `[sample]_count.tsv` which includes two columns: gene ID and the count. RASflow also applies Qualimap 2 [3] to evaluate the alignment quality. There is a report for each sample output/[PROJECT]/genome/alignmentQC. The report file is named `qualimapReport.html` and the explanation can be found here: Qualimap2. A summary of alignment and feature count is given by MultiQC: output/[PROJECT]/genome/report_align_count.html

## 4.3 Differential Expression Analysis (DEA)

If "DEA" was already set to "yes" before mapping / quantification, DEA will be run immediately after that.

If the transcriptome was used as reference, transcript-level DEA will surely be done. Results can be found at output/[PROJECT]/trans/dea/DEA/transcript-level. Besides of that, the user can also choose to do gene-level DEA (`GENE_LEVEL` in config_main.yaml needs to be set to "TRUE"). Results can be found at

output/[PROJECT]/trans/dea/DEA/gene-level. In both cases, for each comparison, there are two files named `dea_[control]_[treat].tsv` and `deg_[control]_[treat].tsv` which contains 6 columns: `genes, logFC, logCPM, LR, PValue, FDR`. The differential analysis is done by edgeR [4]. The explanation of output can be found here: edgeR.

If the genome was used as reference, the DEA results can be found here: output/[PROJECT]/genome/dea/DEA. The count files of samples are also merged together based on the group: output/[PROJECT]/genome/dea/countGroup. The file `[group]_count.tsv` contains the raw counts and file `[group]_gene_norm.tsv` contains the normalized counts which is done by edgeR. Trimmed mean of M values (TMM) [5] is used as the normalisation method.

### 4.4 Visualization of DEA results

To make the test run fast, the fastq and genome files provided as example are fake data, so no genes will show up as differentially expressed. To overcome that, we provide another toy data in data/example/dea for testing visualization. To test visualization on the toy data, simply run:

```
$ nice -5 snakemake -s workflow/visualize_test.rules 2>&1 | tee logs/
    log_visualize_test.txt
```

The results of the example data can be found here: data/example/dea/visualization.

But if you are serious and want to run on your real data, simply run the program as usual with "VISUALIZE" set to "yes". And the results will be in the same folder of your DEA results.

## 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
```

## 6 Reference

[1] P. Ewels, M. Magnusson, S. Lundin, and M. Käller, "MultiQC: summarize analysis results for multiple tools and samples in a single report," Bioinformatics, vol. 32, no. 19, pp. 3047–3048, Oct. 2016.

[2] R. Patro, G. Duggal, M. I. Love, R. A. Irizarry, and C. Kingsford, "Salmon provides fast and bias-aware quantification of transcript expression," Nat. Methods, vol. 14, no. 4, pp. 417–419, 2017.
[3] K. Okonechnikov, A. Conesa, and F. García-Alcalde, "Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data," Bioinformatics, vol. 32, no. 2, p. btv566, Oct. 2015.
[4] M. D. Robinson, D. J. McCarthy, and G. K. Smyth, "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data," Bioinformatics, vol. 26, no. 1, pp. 139–140, Jan. 2010.
[5] M. D. Robinson and A. Oshlack, "A scaling normalization method for differential expression analysis of RNA-seq data," Genome Biol., vol. 11, no. 3, p. R25, Mar. 2010.