RNA-Seq Analysis Snakemake Workflow (RASflow)

Xiaokang Zhang
Email: zhxiaokang@gmail.com

July 6, 2020

Contents

1 Introduction **2**

2 Installation **2**
2.1 For Linux . . . 2
2.1.1 Install Miniconda . . . 2
2.1.2 Install RASflow . . . 2
2.1.3 Some practical tips for Conda . . . 3
2.2 For Mac . . . 3
2.2.1 Install Docker . . . 3
2.2.2 Start Docker engine . . . 3
2.2.3 Create the container for RASflow . . . 3
2.2.4 Some practical tips for Docker . . . 4
2.3 For Windows . . . 4
2.3.1 In the Linux way . . . 4
2.3.2 In the Mac way . . . 4
2.4 (Optional) Switch the branch . . . 5

3 Customized setting up **5**
3.1 Description of your data . . . 5
3.2 Customize the workflow . . . 5
3.3 Setting parameters for quality control and trimming . . . 6
3.4 Setting parameters for quantification of transcripts or genes . . . 6
3.4.1 Mapping to transcriptome . . . 6
3.4.2 Mapping to genome . . . 7
3.5 Setting parameters for Differential Expression Analysis . . . 7
3.6 Setting parameters for visualization . . . 8

4 Run the analysis **8**
4.1 Quality control and trimming 8
4.2 Quantification of transcripts or genes 9
4.3 Differential Expression Analysis (DEA) 9
4.4 Visualization of DEA results 10

5 Visualize the workflow **10**

6 Extend the workflow **10**

7 Reference **11**

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 For Linux

2.1.1 Install Miniconda

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

2.1.2 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:

```
$ conda activate rasflow
```

2.1.3 Some practical tips for Conda

To exit the environment, inside the environment (you will see "(rasflow)" at the beginning of the command line), run the following command:

```
$ conda deactivate
```

To list all the environments on your machine:

```
$ conda info --envs
```

2.2 For Mac

2.2.1 Install Docker

Please refer to https://docs.docker.com/docker-for-mac/install/.

2.2.2 Start Docker engine

To use Docker service, you need to keep the Docker Desktop that you just installed running.

2.2.3 Create the container for RASflow

The following command will pull the RASflow image from the Docker Hub cloud and create a container (named "rasflow") from the image:

```
$ docker container run --name=rasflow -it zhxiaokang/rasflow
```

The above command will get you into the container after the job is done. Everything required by RASflow is already set up in the container (including Miniconda, git, Conda environment called "rasflow"). To run RASflow, you need to activate the environment:

```
# conda activate rasflow
```

Then you will see "(rasflow)" at the beginning of your command line:

```
(rasflow) root@CONTAINER_ID:/#
```

RASflow is at the home directory (/root).

N.B. In the container, you are at the root mode, that's why you see the # tag before your command. But don't panic, you're doing everything inside the container, so you won't mess up your operating system.

2.2.4 Some practical tips for Docker

If you did everything correctly, with this command, you should see the conatainer "rasflow" listed:

```
$ docker container ls --all
```

To stop a container, if you're inside the container, just *exit*; from another Terminal, the following command will stop the container:

```
$ docker stop rasflow
```

To restart the container, use the command:

```
$ docker container start -i rasflow
```

To copy files between host machine and container (execute the commands on host, not inside container):

```
$ docker cp FILE_OR_FOLDER_ON_HOST CONTAINER_ID:/CONTAINER_DIRECTORY
or
$ docker cp CONTAINER_ID:FILE_OR_FOLDER_IN_CONTAINER HOST_DIRECTORY
```

2.3 For Windows

Some of the required tools are not available for Windows, so RASflow currently can not be run on Windows directly. 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

Then you have two options to run RASflow:

2.3.1 In the Linux way

You can almost treat the WSL as a Linux system, so in WSL, you can follow the instructions in section For Linux.

2.3.2 In the Mac way

If you don't want to bother to install the required software and to set up the Conda environment, you can do it in the Mac way: through Docker. Docker for Windows can be found here: https://docs.docker.com/docker-for-windows/install/. Then you follow the instructions in section For Mac

2.4 (Optional) Switch the branch

You are on branch "master" by default, which is the stable version with all the basic and main functions. But there are also some other branches with some particular functions which may not be useful for the majority of users. So in most cases, you can skip this subsection and continue with "master" branch, unless you find the other branches useful for you.

So far, there is a branch called "bwa" where the aligner is `BWA` instead of `HISAT2`. To switch to that branch, run the following command:

```
$ git checkout bwa
```

And install `BWA` with Conda:

```
$ conda install -c bioconda bwa
```

3 Customized setting up

3.1 Description of your data

Modify the metafile describing the data configs/metadata.tsv

There are three columns: ‘‘sample", ‘‘group", and ‘‘subject". 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. To be noted, if the sequencing is pair-end, even though there are 2 FASTQ files for each sample, only write the sample ID (without ‘‘R1" and ‘‘R2" which are usually used to specify the 2 FASTQ files for each sample) in the 1st column.

The 3rd column is needed when the experiment is designed in a pair-wise way and paired-test is required in downstream differential expression analysis. Otherwise, leave it empty.

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 for the parameter "TRANS".

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.

Or you can also map the reads to a reference transcriptome from another public database other than ENSEMBL, or even to a homemade transcriptome.

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[1]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":

- `DEATOOL` indicates the package to be used for DEA: either edgeR [1] or DESeq2 [2]. DESeq2 is recommended for transcriptome-based and DEA because it provides specially a function `DESeqDataSetFromTximport` to import output from package `tximport`[3] which is used in transcriptome-based pipeline as explained here.

- `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` You can also specify whether filtering is required during DEA. If the choice is "TRUE", low expressed transcripts/genes will be removed before doing statistical test.

[1] `BWA` is used in the branch `bwa`. Refer to subsection 2.3

Above parameters are for both transcriptome and genome used as mapping reference. If transcriptome was used as reference, transcript-level DEA will always be done, but you are also provided with the choice to do gene-level DEA on top of that, by specifying the parameter `GENA_LEVEL`.

- If `GENE_LEVEL` is set to "FALSE", the following 3 parameters can be skipped.
- If it is set to "TRUE", you need to specify them.
  - If you use a reference transcriptome downloaded from ENSEMBL, set `ENSEMBL` to "TRUE" and specify the ENSEMBL data set for your organism with parameter `EnsemblDataSet` (this can be found in the look-up table config/EnsemblDataSet_look_up_table.csv).
  - If you use a reference transcriptome downloaded from another database or it is homemade (for example, De Novo assembly), set `ENSEMBL` to "FALSE" and provide the tx2gene file which contains two tab-delimited columns: transcript ID (the one used in the reference transcriptome) and its corresponding gene ID.

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 [4]. 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 [5], 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 [6] 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.

The outputs of DEA include three types of tables: normalized quantification tables, some important statistics for the whole gene or transcript list, and the list of significantly differentially expressed genes or transcripts (with default threshold of $FDR < 0.05$).

The raw count is normalized based on Trimmed Mean of M values (TMM) [7] (if edgeR is used) or the median-of-ratios method [8] (if DESeq2 is used) when the reads are mapped to a genome. The raw and normalized counts can be found here: output/[PROJECT]/genome/dea/countGroup. The file `[group]_gene_count.tsv` contains the raw counts and file `[group]_gene_norm.tsv` contains the normalized counts. But if the reads are mapped to a transcriptome, the normalized values are estimated Transcripts Per Million (TPM) from Salmon and are scaled using the average transcript length over samples and then the library size by “tximport”. The raw and normalized abundance can be found here: output/[PROJECT]/trans/dea/countGroup. The file `[group]_trans_abundance.tsv` contains the raw abundance and file `[group]_trans_norm.tsv` contains the normalized abundance. If `GENE_LEVEL` was set to `TRUE`, gene-level abundance can also be found at the same directory.

If the transcriptome was used as reference, the results of transcript-level DEA can be found at output/[PROJECT]/trans/dea/DEA/transcript-level. Besides 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` if edgeR is used, and `baseMean, log2FoldChange, lfcSE, stat, pvalue, padj` if DESeq2 is used. The explanation of output can be found here: edgeR and DESeq2.

If the genome was used as reference, the DEA results can be found here: output/[PROJECT]/genome/dea/DEA.

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 Extend the workflow

It is possible to replace the tools used in RASflow and this can be easily done.

Firstly, install the new tool using Conda. The installation information can be found on Anaconda Cloud.

Secondly, go the corresponding Snakemake file (workflow/*.rules), locate the rule using this tool, and replace the current tool with the new one. But pay attention to the input and output in that rule, and make sure that the new tool's input and output agree with the context.

To be noted, if the workflow has already been run and the output has already been generated, remember to remove the output for that rule before testing the newly added tool, because Snakemake will skip running the rule if its output already exists.

If you feel that the tool will also be useful for many other users, we highly encourage you to request a commitment. We will then evaluate the necessity and potentially accept the commitment to make the new function available for everyone.

For users who want to add new tools but are not advanced programmers, please open an issue on the GitHub page issues, and we will consider adding it to RASflow.

7 Reference

References

[1] M. D. Robinson, D. J. McCarthy, and G. K. Smyth. "edgeR: a Bioconductor package for differential expression analysis of digital gene expression data". In: *Bioinformatics* 26.1 (Jan. 2010), pp. 139–140. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp616. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btp616.

[2] Michael I Love, Wolfgang Huber, and Simon Anders. "Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2". In: *Genome Biology* 15.12 (Dec. 2014), p. 550. ISSN: 1474-760X. DOI: 10.1186/s13059-014-0550-8. URL: http://genomebiology.biomedcentral.com/articles/10.1186/s13059-014-0550-8.

[3] Charlotte Soneson, Michael I Love, and Mark D Robinson. "Differential analyses for RNA-seq: transcript-level estimates improve gene-level inferences". In: *F1000Research* 4 (2015), p. 1521. DOI: 10.12688/f1000research.7563.2. URL: http://www.ncbi.nlm.nih.gov/pubmed/26925227.

[4] Philip Ewels et al. "MultiQC: summarize analysis results for multiple tools and samples in a single report". In: *Bioinformatics* 32.19 (Oct. 2016), pp. 3047–3048. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw354. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btw354.

[5] Rob Patro et al. "Salmon provides fast and bias-aware quantification of transcript expression". In: *Nature Methods* 14.4 (2017), pp. 417–419. ISSN: 1548-7091. DOI: 10.1038/nmeth.4197.

[6] Konstantin Okonechnikov, Ana Conesa, and Fernando García-Alcalde. "Qualimap 2: advanced multi-sample quality control for high-throughput sequencing data". In: *Bioinformatics* 32.2 (Oct. 2015), btv566. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv566. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv566.

[7] Mark D Robinson and Alicia Oshlack. "A scaling normalization method for differential expression analysis of RNA-seq data". In: *Genome Biology* 11.3 (Mar. 2010), R25. ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-3-r25. URL: http://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-3-r25.

[8] Simon Anders and Wolfgang Huber. "Differential expression analysis for sequence count data". In: *Genome Biology* 11.10 (Oct. 2010), R106. ISSN: 1474-760X. DOI: 10.1186/gb-2010-11-10-r106. URL: https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-10-r106.