PanRNA Manual

Miguel Morard // Javier Alonso-del-Real

June 2019

Contents

iata
Instituto de Agroquímica
y Tecnología de Alimentos

VNIVERSITAT DE VALÈNCIA

1 PanRNA Workflow

PanRNA is a python written pipeline to map and count reads of an RNA-seq experiment. We divide the pipeline in two parts, see figure 1. The first one, we call Basic Run, is the the most common task on RNA-seq which only map the reads to a reference genome and return counts files for each sample. These can be used for differential expression analysis. The second part, the full PanRNA run, includes an extra task meant to avoid loosing information due to a lack in the reference genome annotation or due to the presence of genes in the samples that are not in the reference genome used.

One of the difficulties of RNA-seq, specially with microbial eukaryotes, is that what is not in the reference is not seen which makes analysis of non-laboratory strains hard. Another problem is that their genomes are compacts and so makes the assembly of the transcriptome quite difficult. That is what made us develop PanRNA. PanRNA was first thought to be used with a *pangenome* reference (a multi-fasta file containing all the coding sequences known in a specific species). Even using this approach we found that some important genes were not always reported. We therefore implemented the search for non-reference transcripts part of the pipeline. Using a pangenome plus the search for non-annotated genes has the advantage to consider a wider vision of the genome which considers divergence in gene content between the reference and the studied transcriptome.

Even if we implemented PanRNA for this purpose it is possible to use it to perform a standard RNA-seq run with a complete genome. PanRNA allows to perform the overall mapping and counting in a simple python command. It also accelerates the counting process as the htseq-count process was parallelized.

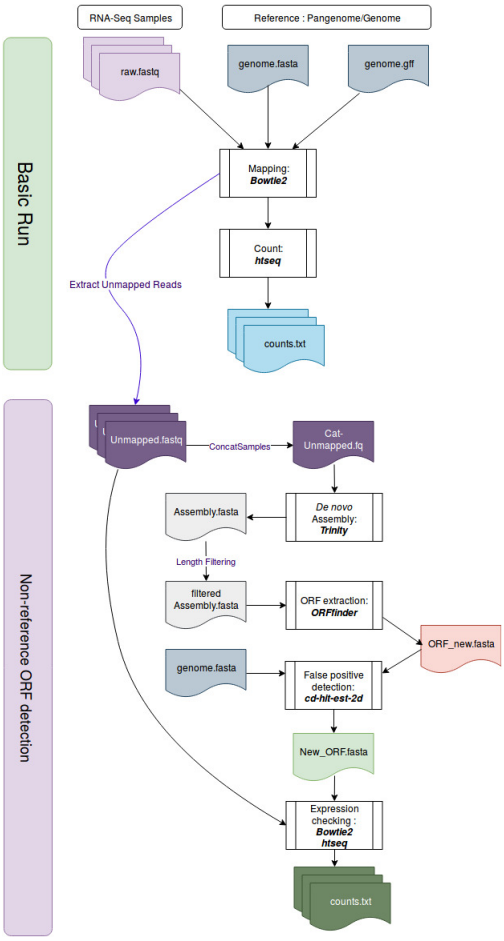

Figure 1: PanRNA Workflow

2 Installation

2.1 Dependencies

PanRNA needs several programs to run. Here is the list of which are needed and the version used in the test. If you will use another version, take care that the parameters used are given in the same way.

Program	version tested
bowtie2	v-2.3.2
samtools	v-1.4.1
htseq-count	v-0.9.0
Trinity	v-2.6.6
seqtk	v-1.3-r106
ORFfinder	v-0.4.3
CD-HIT	v-4.8.1

2.2 Download, Instalation and citation

PanRNA is freely available on github : URL Once downloaded make sure that all the dependencies in the table of section 2.1 are in your path. If you use PanRNA in your publication please cite : CITATION For questions about running PanRNA or its installation please send an email to : miguel.morard@uv.es

3 Fast RUN

To run PanRNA you just need to have a *fasta* formatted reference genome or pangenome, a *gff* file of the annotation of this genome and a text file with the name of the *fastq* files of the RNA-seq experiment. For more explanation of the input files see section 4. Here you will find the basic command to run PanRNA depending on the analysis you want to perform and your data.

3.1 Basic Run Paired-End

Run the Basic Run (no new ORF detection) with paired-end reads. Your *fqFiles.txt* might have a sample per line. Each line might contain the name of R1 and R2 file separated by a tabulation.

```
$ python ./panrna.py --ref pangenome.fasta --gff pangenome.gff --fq fqFiles.txt --out PREFIX
```

Your resuts will be in a folder cold *PREFIX*. The count file for each of the samples in fq files listed in *fqFiles.txt* will be in the *PREFIX/counts* folder.

3.2 Basic Run Single-End

Run the Basic Run (no new ORF detection) with single-end reads. Each sample is a line in the *fqFiles.txt file*. You will only have a file name per line.

```
$ python ./panrna.py --ref pangenome.fasta --gff pangenome.gff --fq fqFiles.txt --out PREFIX --single
```

Your resuts will be in a folder cold *PREFIX*. The count file for each of the samples in fq files listed in *fqFiles.txt* will be in the *PREFIX/counts* folder.

3.3 Complete Run paired-end

Using the same parameters as in the basic paired-end run, here you can run the complete pipeline which include assembly of the unmapped reads and detection of possible new ORFs.

```
$ python ./panrna.py --ref pangenome.fasta --gff pangenome.gff --fq fqFiles.txt --out PREFIX --umap
```

Your resuts will be in a folder cold *PREFIX*. The count file for each of the samples in fq files listed in *fqFiles.txt* will be in the *PREFIX/counts* folder. The results of the last part of the pipeline will be in the *PREFIX/Trinity_umap* folder. For more information see section 5.

4 Input Files

To perform the analisis you will need a *fasta* file of your reference, a *gff file* and a text file containing the path to the different *fastq* file for each sample. For more information about the format of the genome and annotation format see htseq-count documentation. Here you can see how should be the text file with the location of fastq files.

Paired-end files:
Each line corresponds to a sample and the R1 and R2 fastq files are separated by a tab. Files can be in fastq format or gzip compressed fastq files.

```
Sample1.R1.fastq.gz          Sample1.R2.fastq.gz
Sample2.R1.fastq             Sample2.R2.fastq
../folderY/Sample3.R1.fastq.gz          ../folderY/Sample3.R2.fastq.gz
```

Single-end files:
Each line corresponds to a sample.

```
Sample1.fastq.gz
Sample2.fastq
../folderY/Sample3.fastq.gz
```

PanRNA is meant to work with a pangenome so each sequence in the fasta file might be the CDS sequence and the gff file correspond each sequence name and the position 0 to the end of the sequence. It is also allowed to use a complete genome and a gff file locating genes in the genome.

5 Output Files

When running the complete panRNA pipeline all the results will be in the folder you named after the *–out* parameter. In this folder you will find the following folders which contains different data :

- **/alignments**

  This folder contains all the bam files of the alignments for each sample.

- **/counts**

  This folder contains the htseq-count results for each sample *.counts.table* extension.

- **/Trinity_umap**

  Here are all the results of the end of the pipeline with Trinity assembly of the unmapped reads and ORF detection. It has the following folders :

  - **/alignments_Umap**
    In this folder you will find the bam files containing the reads that did not map to the reference and the id of these.
  - **/reads_Umap**
    Here will be placed the fastq files containing only reads that did not map to the references and that will be used to perform the assembly. There will be one file per sample and a concatenated file (*allNm.*.fastq.gz*) containing all the unmapped reads from all the samples in the experiment.
  - **/trinity_out_dir**
    This folder contains the results of Trinity (see Trinity manual for more details). The Trinity scaffolds are named *Trinity.fasta*. Then theses scaffolds are filtered depending on length parameters and the filtered scaffolds are named *Trinity.filtered.fasta*. ORFfinder is run on these and all the ORF found on these scaffolds are in the file *Trinity.filtered.CDS.fasta*. These are then clustered to the reference to remove false positive. The potentially new CDS are then in the file *Trinity.filtered.CDS.clustered.fasta*.
    Trinity.filtered.CDS.clustered.gff is generated to perform the htseq-count.

- **/alignments_trinity**
  The mapping of the unmapped reads on the potentially new CDS file will be in this folder.
- **/counts_trinity**
  The counts of the unmapped reads for each sample on the potentially new CDS can be found here.

---

6 Advanced Options

Most of the arguments that can be modified in each program in the pipeline can also be modified through PanRNA. Here you can find all the arguments that we allow to change. You can also find this information calling PanRNA with -h argument.

6.1 Arguments

optional arguments:
-h, –help show this help message and exit
–out OUT Output name for the analysis, **required**
–umap Set this option if you want to run Trinity on unmap reads
–single Set this option if you have single-end reads
–ref REF Reference PanGenome or Genome, **required**
–fq FQ Tab delimited file with paired fastq file names
–p P Number of threads to run bowtie2 and samtools, default 1
–index Set this option if you do NOT want to index reference
–gff GFF Gff file describing pangenome or genome, **required**
–a A Minimum quality of the reads [0-10], default 10
–idattr IDATTR Feature in gff defining gene name, default Name
–t T Feature to consider from gff, default CDS
–m M htseq-count sensibility, default intersection-nonempty
–Proc PROC Number of processes to run htseq in parallell, default 1
–stranded STRANDED whether the data is from a strand-specific assay <yes/no/reverse>, default: no
–max_memory MAX_MEMORY Maximum RAM allowed for Trinity, default 10G
–CPU CPU Maximum CPU allowed for Trinity, default 5
–filt FILT Minimal length for Trinity scaffold filter, default 800
–ml ML Minimal length of the ORF (nt) that will detect ORFfinder, default 600
–c C sequence identity threshold, this is the default cd- hit's global sequence identity calculated as: number of identical amino acids or bases in alignment divided by the full length of the shorter sequence, default 0.8

–**s2 S2** length difference cutoff by default, seqs in db1 >= seqs in db2 in a same cluster if set to 0.9, seqs in db1 may just >= 90difference cutoff by default, seqs in db1 >= seqs in db2 in a same cluster if set to 0.9, seqs in db1 may just >= 90

–**i I** Fasta file with all CDS sequences annotated in REF (in case of using complete genome), default REF

6.2 Advanced run example

Even if PanRNA was meant to be run on a pangenome, you can run it on a complete genome but you will then have to extract all the CDS annotated in your gff file. If the purpose is to find unanotated genes in the reference we would recommend to run PanRNA on these CDS instead of the complete genome. But you still can run it on the complete genome and provide the CDS in a separate file.

```
$ python ./panrna.py --ref genome.fasta --gff genome.gff --fq fqFiles.txt \
--out PREFIX  --umap --p 6 --Proc 6 \
--i CDS-in-gff.fasta
```

This command will run the complete analysis (–umap option). It will use 6 threads for bowtie2 and samtools (–p) and the samples will be sent to htseq-count in groups of 6 at a time, so using 6 threads, (–Proc). To perform the clustering of the detected new ORF in the Trinity results it will use the extracted CDS file (CDS-in-gff.fasta) instead of the complete genome (genome.fasta).