LncRAnalyzer: A Pipeline for identification of lncRNAs and Novel Protein Coding Transcripts (NPCTs) using RNA-Seq

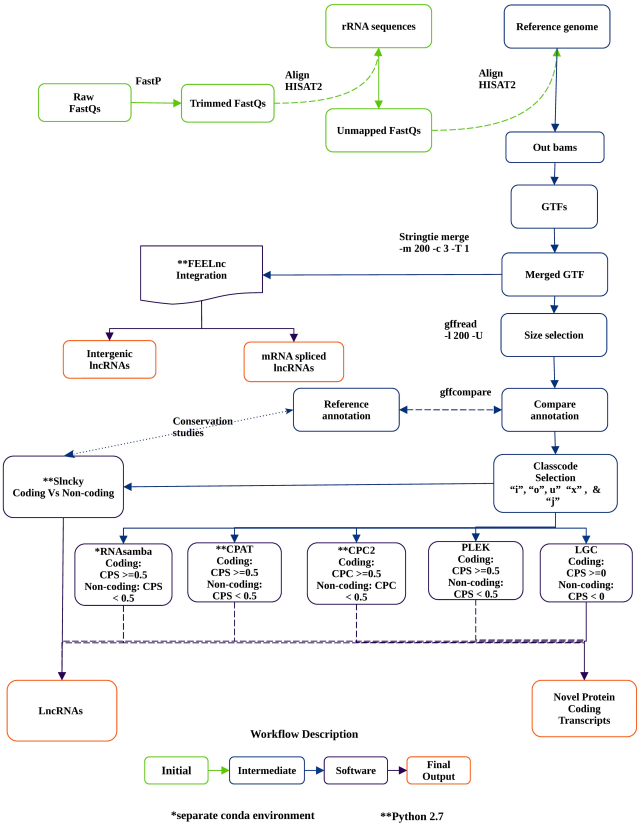
https://gitlab.com/nikhilshinde0909/LncRAnalyzer

LncRAnalyzer

A pipeline for lncRNAs and Novel Protein Coding Transcripts (NPCTs) identification using RNA-Seq

Introduction

LncRAnalyzer is a comprehensive workflow to identify lncRNAs and Novel Protein Coding Transcripts (NPCT) using RNA-Seq. The pipeline contains several steps including quality control, read alignment to reference genome, reference-guided transcript assembly, merge annotations, annotation comparison, class code selection, and retrieval of transcripts in FASTA format. The putative class code selected transcripts will be further evaluated for their coding potentials, features, and protein domain homologies using CPC2, CPAT, PLEK (Time-consuming), RNAsamba, LncFinder, LGC, and PfamScan. The final lncRNAs and NPCTs will be selected based on coding potentials, features, and protein domain homologies. Additionally, if LiftOver files for the organism and related species is provided; this pipeline also performs cross-species lncRNA conservation analysis using Slncky. We also integrated the FEELnc plugin to report the mRNA spliced and intergenic lncRNAs in given RNA-seq samples. For NPCTs, further functional annotations is needed which includes peptide sequences prediction using TransDecoder followed by homology searches using Pfamscan, BLASTP, and BLASTX. The entire workflow is automated using Bpipe and could be implemented in multiple working environment such as Conda, Docker and Singularity.



Implementation

Conda environment

1. To execute the steps in the pipeline, download the latest release of LncRAnalyzer to your local system with the following command

   ```
   git clone https://gitlab.com/nikhilshinde0909/LncRAnalyzer.git
   ```

2. Install latest Miniforge or update existing Mambaforge/Anaconda environment with required software tools as follows

   ```
   bash install_pipeline_tools.sh\
   ```

3. Change directory to LncRAnalyzer.

   ```
   cd LncRAnalyzer
   ```

   This will generate tools.groovy with configured paths for tools

4. Pipeline is ready to execute, prepare your inputs in working directory and data.groovy file

   ```
   Working directory
   mkdir data
   ├── data
   ├── SRR975551_1.fastq.gz
   |       ├── SRR975552_1.fastq.gz
   |       └── (and other fastq.gz files)
   |       ├── SRR975551_2.fastq.gz
   |       ├── SRR975552_2.fastq.gz
   |       └── (and other fastq.gz files)
   |       └── hg38.rRNA.fasta
   |       └── hg38.genome.fasta
   |       └── hg38.annotation.gtf
   |       └── (and other files)
   └── data.groovy
   ```

   Copy your RNA-seq reads (*.fastq.gz), rRNA sequences (*.fa), reference genomes (*.fa), related sp. reference genome (*.fa), annotations (*.gtf) and liftover files in data directory; create file data.txt in the same by using data_template.txt and add paths for raw fastq.gz, rRNA sequences, reference genome, rel sp. reference genome, annotations and liftover files in the same.

   Note: Please refer data.groovy template in LncRAnalyzer dir

5. If you don't have a reference genome, annotations, and rRNA sequence information; you can download the same with the script provided with the pipeline as follows

   ```
   python check_ensembl.py org_name
   eg. python find_species_in_ensembl.py Sorghum
   > sbicolor
   python ensembl.py org_name_in_ensembl
   eg. python download_datasets_ensembl.py sbicolor
   > Ensembl version 56 <- download the datasets
   ```

6. Similarly, if you don't have liftover files for conservation analysis then you can generate it through genome alignments of reference and query species genomes as follows

   ```
   python Liftover.py <threads> <genome> <org_name> <genome_related_species> <rel_sp_name>
   <params_distance>
   ```

   eg.

   ```
   python Liftover.py 16 Sorghum_bicolor.dna.toplevel.fa Sbicolor Zea_mays.dna.toplevel.fa Zmays near
   ```

   We also provide an additional script which will take ensembl gtf and produce bed files to run Slncky as follows

   ```
   python ensembl_gtf2bed.py <ensembl_gtf>  <output_prefix>
   ```

   eg.

   ```
   python ensembl_gtf2bed.py Sorghum_bicolor.58.gtf Sorghum_bicolor
   ```

   This will produce protein-coding, non-coding, mirRNA, and snoRNA bed files for Slncky.

7. The pipeline is ready for execution, run the following command to print help

   ```
   bpipe run ~/Path_to_LncRAnalyzer/Main.groovy –help
   ```

8. Search for supporting species and configure it in the data.groovy

   ```
   bpipe run ~/Path_to_LncRAnalyzer/Main.groovy --supporting_species
   ```

9. Execute pipeline with RNA-seq datasets as follows

   ```
   bpipe run -n ${threads} ~/Path_to_LncRAnalyzer/Main.groovy data.groovy
   ```

Docker environment

1. Build a docker image from the docker file

   ```
   docker build -t nikhilshinde0909/lncranalyzer .
   ```

2. Run the following commands and check LncRAnalyzer and tools.groovy has been created and configured with the proper paths

   ```
   docker run --rm -it nikhilshinde0909/lncranalyzer bash
   cd LncRAnalyzer/
   cat tools.groovy
   exit
   ```

3. Prepare data and data.groovy in your working directory

   ```
   Working directory
   mkdir data
   ├── data
   ├── SRR975551_1.fastq.gz
   |       ├── SRR975552_1.fastq.gz
   |       └── (and other fastq.gz files)
   |       ├── SRR975551_2.fastq.gz
   |       ├── SRR975552_2.fastq.gz
   |       └── (and other fastq.gz files)
   |       └── hg38.rRNA.fasta
   |       └── hg38.genome.fasta
   ```

```
|       └── hg38.annotation.gtf
|       └── (and other files)
└── data.groovy
```

4. Download the required reference genomes and annotations, and then prepare the Slncky annotations and LiftOver files as outlined in the Conda environment section (steps 5 and 6).

5. Run the LncRNAlyzer using docker in your working directory as follows

```
docker run \
-v $(pwd)/data:/pipeline/data \
-v $(pwd)/data.groovy:/pipeline/data.groovy \
nikhilshinde0909/lncranalyzer bpipe run -n 16 /pipeline/LncRAnalyzer/Main.groovy /pipeline/data.groovy
```

6. Export your results to local as follows

```
# list containers
docker ps -a
# Copy data
docker cp container_id:/pipeline ${path to copy results}
```

Note: Please take a look at the LncRAnalyzer documentation to work with pre-built LncRAnalyzer Docker images.

Singularity environment

1. Build a Singularity image from the Singularity file

```
sudo singularity build lncranalyzer.sif Singularity
```

2. Run the following commands and check LncRAnalyzer and tools.groovy has been created and configured with the proper paths

```
singularity exec lncranalyzer.sif
bash cd /pipeline/LncRAnalyzer/
cat tools.groovy
exit
```

3. Prepare data and data.groovy in your working directory

```
Working directory
mkdir data
├── data
├── SRR975551_1.fastq.gz
|       ├── SRR975552_1.fastq.gz
|       └── (and other fastq.gz files)
|       ├── SRR975551_2.fastq.gz
|       ├── SRR975552_2.fastq.gz
|       └── (and other fastq.gz files)
|       └── hg38.rRNA.fasta
|       └── hg38.genome.fasta
|       └── hg38.annotation.gtf
|       └── (and other files)
└── data.groovy
```

4. Download the required reference genomes and annotations, and then prepare the Slncky annotations and LiftOver files as outlined in the Conda environment section (steps 5 and 6).

5. Run the LncRNAlyzer using singularity in your working directory as follows

```
singularity exec -B $(pwd)/data:/pipeline/data \
-B $(pwd)/data.groovy:/pipeline/data.groovy \
~/LncRAnalyzer/lncranalyzer.sif \
bpipe run -n 16 /pipeline/LncRAnalyzer/Main.groovy /pipeline/data.groovy
```

6. Enjoy your results in the working directory

   Note: Please look at the LncRAnalyzer documentation to work with pre-built LncRAnalyzer Singularity images.

Working with pre-built LncRAnalyzer docker images

1. Pull the docker image from Docker Hub or GitLab

```
# Pull Docker image from DockerHub
docker pull nikhilshinde0909/lncranalyzer:latest

or
# Pull Docker image from GitLAB
docker pull registry.gitlab.com/nikhilshinde0909/lncranalyzer:latest
```

2. Run the following commands and check LncRAnalyzer and tool.groovy has been created is configured with the proper paths, supporting species and help

```
# Run DockerHub container
docker run --rm -it nikhilshinde0909/lncranalyzer bash

or
# Run GitLab container
docker run --rm -it registry.gitlab.com/nikhilshinde0909/lncranalyzer bash

# check paths
cd LncRAnalyzer/
cat tools.groovy

# print help and supporting species
bpipe run Main.groovy --help
bpipe run Main.groovy --supporting_species

# exit
exit
```

3. Prepare data and data.groovy in your working directory

```
working directory
mkdir data
├── data
|   ├── SRR975551_1.fastq.gz
|   ├── SRR975552_1.fastq.gz
|   └── (and other fastq.gz files)
|   ├── SRR975551_2.fastq.gz
|   ├── SRR975552_2.fastq.gz
|   └── (and other fastq.gz files)
|   └── hg38.rRNA.fasta
|   └── hg38.genome.fasta
|   └── hg38.annotation.gtf
|   └── (and other files)
└── data.groovy
```

Note: Please refer data.groovy template in LncRAnalyzer dir

4. If you don't have a reference genome, annotations, and rRNA sequence information; you can download the

same with the script provided with the pipeline as follows

```
python check_ensembl.py org_name
eg. python find_species_in_ensembl.py Sorghum
> sbicolor
python ensembl.py org_name_in_ensembl
eg. python download_datasets_ensembl.py sbicolor
> Ensembl version 56 <- download the datasets
```

5. Similarly, if you don't have LiftOver files for conservation analysis then you can generate it through genome alignments of reference and query species genomes as follows

```
python Liftover.py <threads> <genome> <org_name> <genome_related_species> <rel_sp_name>
 <params_distance>
eg.
python Liftover.py 16 Sorghum_bicolor.dna.toplevel.fa Sbicolor Zea_mays.dna.toplevel.fa Zmays near
```

6. We also provide an additional script which will take ensembl *.gtf and produce bed files to run slncky as follows

```
python ensembl_gtf2bed.py <ensembl_gtf> <output_prefix>
eg.
python ensembl_gtf2bed.py Sorghum_bicolor.58.gtf Sorghum_bicolor
```

This will produce protein-coding, non-coding, miRNA, and snoRNA bed files for Slncky.

7. Run the LncRNAlyzer using docker in your working directory as follows

```
docker run \
 -v $(pwd)/data:/pipeline/data \
 -v $(pwd)/data.txt:/pipeline/data.txt \
 nikhilshinde0909/lncranalyzer bpipe run -n 16 /pipeline/LncRAnalyzer/Main.groovy
 /pipeline/data.groovy

or

docker run \
   -v $(pwd)/data:/pipeline/data \
   -v $(pwd)/data.txt:/pipeline/data.txt \
 registry.gitlab.com/nikhilshinde0909/lncranalyzer bpipe run -n 16 /pipeline/LncRAnalyzer/Main.groovy
 /pipeline/data.groovy
```

8. Export your results from the container as follows

```
# list containers
docker ps -a
# Copy data
docker cp container_id:/pipeline ${path to copy resuls}
```

Thanks for using LncRAnalyzer-Docker !

Working with pre-built LncRAnalyzer singularity images

1. Pull the singularity image from singularity Sylabs cloud [https://cloud.sylabs.io/library/nikhilshinde0909/rnaseq/lncranalyzer]

```
# Pull singularity image from Sylabs cloud
singularity pull library://nikhilshinde0909/rnaseq/lncranalyzer
```

   This will pull the lncranalyzer.sif file from Sylabs cloud

2. Run the following commands and check LncRAnalyzer and tool.groovy has been created and is configured with the proper paths, supporting species, and help

```
# Run singularity container
singularity run lncranalyzer.sif bash

# check paths
cd LncRAnalyzer/
cat tools.groovy

# print help and supporting species
bpipe run Main.groovy --help
bpipe run Main.groovy --supporting_species

# exit
exit
```

3. Prepare data and data.groovy in your working directory

```
working directory
mkdir data
├── data
|   ├── SRR975551_1.fastq.gz
|   ├── SRR975552_1.fastq.gz
|   └── (and other fastq.gz files)
|   ├── SRR975551_2.fastq.gz
|   ├── SRR975552_2.fastq.gz
|   └── (and other fastq.gz files)
|   └── hg38.rRNA.fasta
|   └── hg38.genome.fasta
|   └── hg38.annotation.gtf
|   └── (and other files)
└── data.groovy
```

   Note: Please refer data.groovy template in LncRAnalyzer dir

4. If you don't have a reference genome, annotations, and rRNA sequence information; you can download the same with the script provided with the pipeline as follows

```
python check_ensembl.py org_name
eg. python find_species_in_ensembl.py Sorghum
```

```
> sbicolor
python ensembl.py org_name_in_ensembl
eg. python download_datasets_ensembl.py sbicolor
> Ensembl version 56 <- download the datasets
```

5. Similarly, if you don't have LiftOver files for conservation analysis then you can generate it through genome alignments of reference and query species genomes as follows

```
python Liftover.py <threads> <genome> <org_name> <genome_related_species> <rel_sp_name>
 <params_distance>
eg.
python Liftover.py 16 Sorghum_bicolor.dna.toplevel.fa Sbicolor Zea_mays.dna.toplevel.fa Zmays near
```

6. We also provide an additional script which will take ensembl *.gtf and produce bed files to run slncky as follows

```
python ensembl_gtf2bed.py <ensembl_gtf> <output_prefix>
eg.
python ensembl_gtf2bed.py Sorghum_bicolor.58.gtf Sorghum_bicolor
```

   This will produce protein-coding, non-coding, miRNA, and snoRNA bed files for Slncky.

7. Run the LncRNAlyzer using singularity in your working directory as follows

```
singularity exec -B $(pwd)/data:/pipeline/data \
  -B $(pwd)/data.groovy:/pipeline/data.groovy \
  lncranalyzer.sif \
  bpipe run -n 16 /pipeline/LncRAnalyzer/Main.groovy /pipeline/data.groovy
```

8. Enjoy results in your working directory

Thanks for using LncRAnalyzer-singularity !

Performance

The performance of coding potential prediction using CPAT, CPC2, LGC, RNAsamba, and FEELnc was estimated with 50 RNA-Seq accessions of sorghum cultivar PR22 from past studies [https://doi.org/10.1186/s12864-019-5734- x]