CelebrationNight

Scripts for transcriptome analysis.

- This project include some experiment coding for testing purpose.
- Use it at your own risk.
- I will try my best to complete and polish this documentation.

Type of process

CoFRIL

- **Co**nfig **F**irst and **R**un **I**t **L**ater

Definition	Naming
1. Pack process code into python class (**.py** file with "lib" prefix)	
2. Run *-setConfig.py first to configure dependent variables (Initialization)	*-setConfig.py
	template: "*-configExample.py"
3. Run related python class in *-run.py	*-run.py
	template: "*-exampleRun.py"

ReRa

- **Re**lay **Ra**ce

Definition
1. Use script language to control the workflow
2. Use executable binary or script language or both to process information

RuDi

- **Ru**n **Di**rectly

Definition
1. Declare sample-dependent variables on the beginning of script
2. Derive downstream variables
3. Run code directly without class

Configuration

libConfig

For general-purpose storage of configuration

- *Coming soon*

cf-Branch

For experiment design

- *Coming soon*

cf-Command

For version management of executable

- *Coming soon*

cf-Parameter

For adjustment

- *Coming soon*

Database conversion (Preparation)

Extract information from Genome annotation

- *Coming soon*

Reconstruct information from functional annotation database

- *Coming soon*

Processing Stage (General usage)

Stage 01 - FASTQ Quality Report

List	Detail
Codename	**01-fq-fastqc**
Usage	Quality reports
Type	ReRa
Binary	FastQC [1]
Language	Shell script & Python 3
Input	NGS reads, FASTQ format
Output	HTML reports

- **Command:**
  - `bin/FastQC/fastqc -f fastq -o [largeData/04-hisat2/species/speciesDatabase-trimQ30/report]`

Stage 02 - HISAT2 index

List	Detail
Codename	**02-hisat2-index**
Usage	Build HISAT2 index
Type	CoFRIL
Class	libHISAT.indexer() indexer
Binary	hisat2-build from HISAT2 [2]
Input	Genome sequences, FASTA format
Output	HISAT2 index of genome

Stage 03 - Trim

List	Detail
Codename	**03-trim**
Usage	Trim FASTQ files
Type	CoFRIL
Class	libTrim.trimmer() indexer
Binary	Trimmomatic [3]
Input	raw NGS reads, FASTQ format
Output	Trimmed reads, FASTQ format

Stage 04 - HISAT2

List	Detail
Codename	**04-hisat2**
Usage	Alignment and mapping
Type	CoFRIL
Class	libHISAT.aligner() aligner
Binary	hisat2 from HISAT2 [2]
	samtools from SAMtools [4]
Input	HISAT2 index of genome **(02-hisat2-index)**
	Trimmed reads, FASTQ format **(03-trim)**
Output	Alignments, BAM format

Stage 05 - gffRead

List	Detail
Codename	**05-gr-gffRead**
Usage	Convert genome into transcriptome
Type	CoFRIL
Class	libCuffdiff.converter() converter
Binary	gffread [5]
Input	Genome annotation, GFF3 format
Output	Transcriptome annotation, GTF2 format

Stage 06 - Transcripts extraction

List	Detail
Codename	**06-fn**
Usage	Transcriptome extractor
Type	ReRa
Binary	gffread [5]
Language	Shell script & Python 3
Input	Genome sequences, FASTA format
	Transcriptome annotation, GTF2 format **(05-gr-gffRead)**
Output	Transcripts, FASTA format
	Transcripts table, TSV format

- **Command:**

  -
    ```
    bin/cufflinks/gffread\
    -g userData/dbgs-GenomeSequence/speciesDatabase/speciesDatabase.fn\
    -w userData/06-gr-exportTranscript/speciesTreatment/speciesDatabase-trimQ30-transcript.fn\
    userData/05-gr-transcriptomeConstruction/speciesTreatment/speciesDatabase-trimQ30-final.gtf
    ```

Stage 07 - Transcript abundances estimation

List	Detail
Codename	**07-cd-CuffDiff**
Usage	Estimate transcript abundances
Type	CoFRIL
Class	libCuffdiff.differ() [differ]
Binary	cuffdiff from Cufflinks [6]
Input	Alignments, BAM format **(04-hisat2)**
	Transcriptome annotation, GTF2 format **(05-gr-gffRead)**
Output	Abundances/Expression profile, DIFF format (TSV format)

Stage 08 - Homologous annotation

List	Detail
Codename	**08-an**
Usage	an1. Extract info of transcript-gene relation
	an2. Link transcript ID with gene ID and homologous ID
	(cont.) Further annotate with functional annotation databases
Type	RuDi
Language	Python 3
Input	Abundances/Expression profile, DIFF format (TSV format) **(07-cd-CuffDiff)**
	Genome annotation, JSON format **(dbga-GenomeAnnotation)**
	Homolog , JSON format **(dbga-GenomeAnnotation)**
	GO Terms, JSON format **(dbgo-GOdatabase)**
	KEGG pathways, JSON format **(dbkg-KEGG-hirTree)**
	Other databases...
Output	Homologous annotations, JSONs format (various files)

Stage 09 - Transcriptome information summarizer

List	Detail
Codename	**09-cd**
Usage	cd1. Convert the results of cuffdiff into SQLite3 form
	cd2. Annotate and seperate information into...
	(cont.) sample-orientation Expression tables
Type	RuDi
Language	Python 3
Input	Abundances/Expression profile, DIFF format (TSV format) **(07-cd-CuffDiff)**
	Homologous annotations, JSONs format **(08-an)**
Output	Annotated abundances/expression profile, SQLite3 format
	Annotated abundances/expression profile, TSV format

Stage 10 - Differential Expression Analysis (Haven't release)

List	Detail
Codename	**10-grouping**
Usage	Group and calculate the count of Differential Expressed Genes (DEGs) and Specifically Expressed Genes (SEGs)
Type	RuDi
Language	Python 3
Input	Annotated abundances/expression profile, TSV format **(09-cd)**
	Homologous annotations, JSONs format **(08-an)**
Output	DEA result files, JSON format
	DEA result files, TSV format
	DEA count record, LOG format (TXT format)

Stage 11 - Gene set enrichment analysis (Haven't release)

List	Detail
Codename	**11-ea-enrichAnaly**
Usage	Compare and calculate the ratio of count of DEGs or SEGs
Type	RuDi
Language	Python 3
Input	DEA result files, JSON format **(10-grouping)**
	Annotated abundances/expression profile, SQLite3 format **(09-cd)**
	Homologous annotations, JSONs format **(08-an)**
Output	GSEA result files, JSON format
	GSEA result files, TSV format

Stage 12 - Fisher's Exact Test and visualization (Haven't release)

List	Detail
Codename	**12-ft**
Usage	ft1. Do Fisher's Exact Test
	ft2. Visualization
Type	RuDi
Language	Python 3
Input	GSEA result files, JSON format **(11-ea-enrichAnaly)**
	Homologous annotations, JSONs format **(08-an)**
Output	FET result files, PNG format
	FET result files, SVG format
	FET result files, TSV format

Python Library (Module & Classes)

libHISAT

- Original Command:
    - libHISAT.indexer()
        - `hisat2-build \` **#Building HISAT2 Index** `-p [THREAD] <Path and Name of GENOME File> \` `< prefix of HISAT2-build genome index (path+header)>`
    - libHISAT.aligner()
        - `hisat2 \` **#Aligning and mapping** `-q [--dta/--dta-cufflinks] --phred <phred> -p <thread> \` `-x <prefix of HISAT2-build genome index> \` `-1 <forward fastq files of samples> \` `-2 <reverse fastq files of samples> \` `-S <output SAM files> \`
        - `samtools view -o <out.bam> -Su <in.sam>` **#Convert SAM to BAM**
        - `samtools sort -o <out-sorted.bam> <in.bam>` **#Sorting BAM for decreasing file size**

libTrim

- Original Command:
    - libTrim.trimmer()
        - `java -jar <bin>/trimmomatic-0.35.jar PE \` **#Pair-End** `-phred33 -threads <threads> \` `input_forward.fq.gz input_reverse.fq.gz \` `output_forward_paired.fq.gz output_forward_unpaired.fq.gz \` `output_reverse_paired.fq.gz output_reverse_unpaired.fq.gz \` `<ILLUMINACLIP> <LEADING> \` `<TRAILING> <SLIDINGWINDOW> <MINLEN>`
        - `java -jar <bin>/trimmomatic-0.35.jar SE \` **#Single-End** `-phred33 -threads <threads> \` `input.fq.gz output.fq.gz \` `<ILLUMINACLIP> <LEADING> \` `<TRAILING> <SLIDINGWINDOW> <MINLEN>`

libCuffdiff

- Original Command:
  - **libCuffdiff.differ()**
    - `cuffdiff \``-p -o ` `-L <label1,label2,…,labelN> <transcripts.gtf> \` `[[sample1_replicate1.sam,…] …… […,sampleN_replicateM.sam]]`
  - **libCuffdiff.converter()**
    - `bin/cufflinks/gffread <inputFile> -T -o <outputFile>`

References

1. [FastQC] https://www.bioinformatics.babraham.ac.uk/projects/fastqc/
   - **Andrews, S.** (2018). FastQC: a quality control tool for high throughput sequence data (Babraham Bioinformatics, Babraham Institute, Cambridge, United Kingdom).
2. [HISAT2] https://doi.org/10.1038/nprot.2016.095 (Article)
   - **Pertea, M., Kim, D., Pertea, G.M., Leek, J.T., and Salzberg, S.L.** (2016). Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown. Nat Protoc **11,** 1650-1667.
   - https://ccb.jhu.edu/software/hisat2/manual.shtml (Documentation & Binary)
3. [Trimmomatic] https://doi.org/10.1093/bioinformatics/btu170 (Article)
   - **Bolger, A.M., Lohse, M., and Usadel, B.** (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. Bioinformatics **30,** 2114-2120.
   - http://www.usadellab.org/cms/?page=trimmomatic (Documentation & Binary)
4. [SAMtools] https://www.htslib.org/doc/samtools.html
5. [gffread] https://ccb.jhu.edu/software/stringtie/gff.shtml
6. [Cuffdiff] https://doi.org/10.1038/nbt.2450 (Article)
   - **Trapnell, C., Hendrickson, D.G., Sauvageau, M., Goff, L., Rinn, J.L., and Pachter, L.** (2013). Differential analysis of gene regulation at transcript resolution with RNA-seq. Nat Biotechnol **31,** 46-53.
   - https://cole-trapnell-lab.github.io/cufflinks/manual/ (Documentation)
   - https://cole-trapnell-lab.github.io/cufflinks/install/ (Binary)

Processing Stage (Distinctive usage)

The scripts under this catalogue may lose function as their development didn't stick to the current coding style.

Stage 04 - HISAT2 Summariser

List	Detail
Codename	**04-hs**
Usage	Analyse HISAT2 result
Type	CoFRIL
Class	libHISAT.summariser()
Binary	samtools from SAMtools
Input	**04-hisat2**

Stage 07 - Comparing Genomic Annotation

List	Detail
Codename	**07-cg**
Usage	Get information of isoform under each gene model
Type	RuDi
Input	**07-st**