ANNOgesic Documentation

Release 1.1.14

Sung-Huan Yu

Jan 24, 2023

Contents

CHAPTER 1

Table of content

1.1 Required tools or databases

Each module has different required tools or databases. **For running a module, the users just need to install the specific tools for it.** For examples, running TSS prediction only needs TSSpredator.

Please also check our Docker image. Via Docker image, all the required tools and environment can be installed and set automaticaly. Moreover, Singularity can be used for building and running Docker image without root permission.

All the used tools are eithor compatible with the ISC license of ANNOgesic or open source.

1.1.1 Tools

Basic requirement:

Python : version higher or equal to 3.4.

BioPython: version higher or equal to 1.65.

Wget: version higher or equal to 1.17.1.

Matplotlib : version higher or equal to 1.5.0.

NumPy : version higher or equal to 1.9.2.

Annotation transfer:

BioPerl: version higher or equal to 1.6.1.

RATT : version higher or equal to 1.64. Please be attention, before you start to run RATT (annotation transfer), run `source $PAGIT_HOME/sourceme.pagit` first. It will modify the path for execute RATT. If you run ANNOgesic through Docker container, you can skip this step.

SNP calling:

Samtools : version higher or equal to 1.3.1 (using htslib 1.3.1).

Bcftools : version higher or equal to 1.3.1 (using htslib 1.3.1).

TSS and PS prediction:

TSSpredator : version higher or equal to 1.06.

TSS and PS parameter optimization:

TSSpredator : version higher or equal to 1.06.

sRNA detection:

Blast+ : version higher or equal to 2.2.28+.

ViennaRNA : version higher or equal to 2.3.2. RNAfold, mountain.pl and relplot.pl are needed for sRNA prediction.

Terminator detection:

TranstermHP : version higher or equal to 2.09.

ViennaRNA : version higher or equal to 2.3.2. RNAfold is needed for terminator prediction.

Promoter search:

MEME : version higher or equal to 4.11.1.

GLAM2 : version higher or equal to 4.11.1.

MPICH : version higher or equal to 3.2. It is for parallel version of promoter detection.

sRNA target prediction:

ViennaRNA : version higher or equal to 2.3.2.

RNAup, RNAplex, RNAplfold are required for executing many modules of ANNOgesic.

IntaRNA: version higher or equal to 2.0.4.

Circular RNA detection:

Samtools : version higher or equal to 1.3.1 (using htslib 1.3.1).

Segemehl : version higher or equal to 0.1.9. When you install Segemehl, please type 'make all' instead of 'make' after running configure. Otherwise, the testrealign.x won't appear.

Riboswitch and RNA thermometer identification:

Infernal : version higher or equal to 1.1.1.

CRISPR detection:

CRT: version higher or equal to 1.2.

Subcellular localization prediction:

Psortb : version higher or equal to 3.0.

Protein-protein interaction detection:

Networkx : version higher or equal to 1.10.

Generating screenshots of IGV:

IGV

Colorization of screenshots:

ImageMagick : version higher or equal to 6.9.0-0.

1.1.2 Databases

sRNA detection:

BSRD (A suggestion for sRNA prediction and can be found here.)

nr database

Riboswitch and RNA thermometer prediction:

Rfam (Only including Rfam IDs of riboswitches and RNA thermometers – the datasets can be found here.)

GO term identification:

idmapping_selected.tab from Uniprot

goslim.obo

go.obo

Protein-protein interaction detection:

species.v${VERSION}.txt from STRING (${VERSION} represents the version number.)

1.2 Installation

There are three ways to install ANNOgesic. Please refer to the following sections. ANNOgesic can only work when the requirements are installed properly. If you install ANNOgesic through source code or `pip3`, please install the pre-required tools by yourself.

1.2.1 Github

All the source code including a run script (contains all the commands which are presented in tutorial) of ANNOgesic can be retrieve from our Git repository. Using the following commands can clone the source code easily.

```
$ git clone https://github.com/Sung-Huan/ANNOgesic.git
```

or

```
$ git clone git@github.com:Sung-Huan/ANNOgesic.git
```

In order to make ANNOgesic runnable, we should create a soft link of `annogesiclib` in `bin`.

```
$ cd ANNOgesic/bin
$ ln -s ../annogesiclib .
```

1.2.2 Docker

Some modules of ANNOgesic need third-party tools. In order to avoid all the possible issue caused by the dependencies, a Docker image is provided. For the details of Docker image, please check here.

For using Docker image, please use one of the following commands:

1. You can simply pull the Docker image as following

```
$ docker pull silasysh/annogesic
```

2. Alternatively, you can build the image via Dockerfile. Please Download the Dockerfile from our Git repository. Then switch to the folder which Dockerfile are located. For the following commands, please refer to here.

If you want to check other commands of Docker, please refer to here.

1.2.3 Singularity

Singularity is another way to install ANNOgesic via Docker image without root permission.

```
$ singularity build \
    annogesic.img \
    docker://silasysh/annogesic:latest
```

After building Singularity image of ANNOgesic, the user just needs to put the following line before the command that needs to be executed.

```
singularity exec -B $STORAGE_PATH annogesic.img
```

Please put the storage path of your home directory to `$STORAGE_PATH`. `df` can be used to check the storage system.

1.2.4 pip3

ANNOgesic is also hosted in PyPI server. Thus, it can be simply installed via `pip3`.

```
$ pip3 install ANNOgesic
$ pip3 install ANNOgesic --upgrade
```

You can also install ANNOgesic without root permission.

```
$ pip3 install --user ANNOgesic
$ pip3 install ANNOgesic --user --upgrade
```

1.2.5 Install Dependencies

If the user want to install ANNOgesic via source code, `get_package_database.sh` can provide a way to install tools and download database automatically. The required versions of the tools will be shown on the screen as well.

1.3 Docker image

Docker is a platform for distributing package. It is light and easy to manage. `ANNOgesic` includes a `Dockerfile` which is for build up a environment and install all required tools for running `ANNOgesic`.

Two ways can be used to build or pull Docker image:

1. You can simply pull the Docker image by running

```
$ docker pull silasysh/annogesic
```

2. Alternatively, you can build the image by `Dockerfile`. Please go to the folder where `Dockerfile` are located. Then type

```
$ sudo docker build -t="annogesic" .
```

It will build up an image called annogesic. You can see the images by typing `docker images`

Based on different ways of installing docker image of ANNOgesic, the name of the docker image will be different. Pulling from DockerHub is:

```
REPOSITORY          TAG                 IMAGE ID            CREATED             ␣
↪VIRTUAL SIZE
silasysh/annogesic  latest              d35f555694ad        3 days ago          2.782␣
↪GB
ubuntu              14.04               d2a0ecffe6fa        11 days ago         188.4␣
↪MB
```

Building Docker image by `Dockerfile` is:

```
REPOSITORY          TAG                 IMAGE ID            CREATED             ␣
↪VIRTUAL SIZE
annogesic           latest              d35f555694ad        3 days ago          2.782␣
↪GB
ubuntu              14.04               d2a0ecffe6fa        11 days ago         188.4␣
↪MB
```

Then we can use the image to create a container for running `ANNOgesic`. Now, we used `silasysh/annogesic` to represent Docker image. If you built Docker image by yourself, please replace `silasysh/annogesic` by `annogesic`. Please type

```
$ docker run -t -i silasysh/annogesic bash
```

Then you will jump into the container.

```
root@c9de31fcd7e3:~ ls
ANNOgesic
```

If you want to mount the files from your host to the container, just add `-v` to the command.

```
$ docker run -t -i -v /host/path/target:/file/path/within/container silasysh/
↪annogesic bash
```

The paths should be absolute path. If we go to `root` in container. We can see the file.

If you want to copy the files from container to host, you can use `cp`.

```
$ docker cp <containerId>:/file/path/within/container /host/path/target
```

If you have no root permission for running Docker, Singularity is another way to build up the image without root permission.

```
$ singularity build \
    annogesic.img \
    docker://silasysh/annogesic:latest
```

After building Singularity image of ANNOgesic, the user just needs to put the following line before the command that needs to be executed.

```
singularity exec -B $STORAGE_PATH annogesic.img
```

Please put the storage path of your home directory to `$STORAGE_PATH`. `df` can be used to check the storage system.

1.4 ANNOgesic's subcommands

In general, the subcommands need at least one argument - the analysis folder. If it is not given, ANNOgesic assumes the current folder as the analysis folder. All the default settings were tested on several different organisms and produced in general good results. A user can perform an analysis rather easily which those default values and step-wise adapt the parameters to optimize the results if needed.

1.4.1 The format of filename

In order to recognize file types as well as relation of genome name and features, please use following principle of filename designation:

The genome filenames should be the same as the annotation file designation, i.e. `NC_007795.fa, NC_007795.gff, NC_007795.ptt, NC_007795.rnt, NC_007705.gbk`.

For all input gff files for ANNOgesic please use following format: `$GENOME_$FEATURE.gff`. For an example, `NC_007795_TSS.gff, NC_007795_transcript.gff`.

Possible feature names are:

Feature name	meaning
TSS	transcription starting site
processing	processing site
transcript	transcript
sRNA	small RNA
sORF	small open reading frame
term	terminator
5UTR	5'UTR
3UTR	3'UTR
circRNA	circular RNA
riboswitch	riboswitch
RNA_thermometer	RNA_thermometer
CRISPR	CRISPR

Please avoid `|` in the filename and genome name of Gff3 files or fasta file.

1.4.2 The input format of libraries for running ANNOgesic

Some `ANNOgesic` modules require certain library information. Please use following format:

`$LIBRARY_FILENAME:$LIBRARY_TYPE:$CONDITION:$REPLICATE:$STRAND`

`$LIBRARY_FILENAME` means the `.wig` file.

`$LIBRARY_TYPE` can be `tex` (TEX+) or `notex` (TEX-) or `frag` (fragmented/conventional).

`$CONDITION` is the index of conditions. Please use 1, 2, 3, ... to represent different conditions.

`$REPLICATE` is the index of replicates. Please use a, b, c, ... to represent different replicates.

`$STRAND` is the strand of wiggle file. Please use + or -.

All libraries should be seperated by colons and listed in one line, like shown in the example above, i.e. for TEX +/- treated libraries:

```
TSB_OD_0.2_TEX_reverse.wig:tex:1:a:- \
TSB_OD_0.5_TEX_reverse.wig:tex:2:a:- \
TSB_OD_0.2_TEX_forward.wig:tex:1:a:+ \
TSB_OD_0.5_TEX_forward.wig:tex:2:a:+ \
TSB_OD_0.2_reverse.wig:notex:1:a:- \
TSB_OD_0.5_reverse.wig:notex:2:a:- \
TSB_OD_0.2_forward.wig:notex:1:a:+ \
TSB_OD_0.5_forward.wig:notex:2:a:+
```

or for fragmented libraries (RNA-Seq generated after transcript fragmentation):

```
fragmented_forward.wig:frag:1:a:+ fragmented_reverse.wig:frag:1:a:-
```

If only conventional RNA-seq data without fragmentation or TEX treated can be provided, it can still be assigned to fragmented libraries. However, it may influence the results.

1.4.3 log file for storing the details of each module

If the user needs to check the details of the running of each module, `log.txt` can be found in the output folder of each module. These files include the full paths of the external tools, the required versions of the tools, the running commands of the tools and the full paths of the output files.

1.4.4 create (create analysis folder)

`create` generates the folders for analysis. Once created, please move the required files into the corresponding folders.

The folders are following:

input: Stores all input files.

> **BAMs:** For `.bam` files. `BAMs_map_related_genomes` is for the `.bam` files which are mapped on closely related genomes of the reference genomes. `BAMs_map_reference_genomes` is for the `.bam` files which are mapped on the reference genomes.
>
> **databases:** For all databases.
>
> **manual_TSSs:** If the manual detected transcription starting sites (TSSs) can be provided, it can be stored here for running `TSS_optimization` or merging the automatic predicted ones and manual detected ones. Please use gff3 format.
>
> **manual_processing_sites:** It is similar to `manual_TSS`, but for processing sites.
>
> **mutation_table:** If the mutation table between the closely ralated genomes and reference genomes is provided, please put the file here. Please check the section of *update_genome_fasta (update reference genome fasta file)* for the format of mutation table.
>
> **reads:** For running `circrna` with mapping reads by ANNOgesic, please put the reads here. `.bzip2` and `.gzip` as input is accepted.
>
> **references:** For annotation files and fasta files. If they can be downloaded from NCBI, the files can also be obtained via running *get_input_files (download required files)*.
>
> **riboswitch_ID_file:** For storing the file which contains all the Rfam IDs of riboswitches. For format details, please check the section of *riboswitch_thermometer (riboswitch and RNA thermometer detection)*.
>
> **RNA_thermometer_ID_file:** For storing the file which contains all the Rfam IDs of RNA thermometer. For format details, please check the section of *riboswitch_thermometer (riboswitch and RNA thermometer detection)*.

wigs: For wiggle files. Based on the methods of RNA-Seq, wiggle files can be stored in `fragment` (fragmented/conventional libraries) or `tex_notex` (TEX +/- treated libraries).

output: Stores all output files.

- **Arguments**

```
usage: annogesic create --project_path PROJECT_PATH

optional arguments:
  -h, --help            show this help message and exit

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Name/path of the project
```

1.4.5 get_input_files (download required files)

`get_input_files` is the subcommand for downloading required files (fasta, annotation files) from NCBI. Therefore, the web address of the reference genome in NCBI needs to be assigned. For an example, ftp://ftp.ncbi.nlm.nih.gov/genomes/all/GCF_000013425.1_ASM1342v1 Then, the user can assign the file type for download.

- **Reqired information**

FTP source: The IP of NCBI.

- **Basic arguments**

```
usage: annogesic get_input_files --project_path PROJECT_PATH
                                 --ftp_path FTP_PATH [--ref_fasta]
                                 [--ref_gff] [--ref_gbk]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --ftp_path FTP_PATH, -F FTP_PATH
                        Path of folder on the NCBI FTP server where the
                        required files are located.
  --ref_fasta, -f       Download fasta files of the reference. Default is
                        False.
  --ref_gff, -g         Download gff files of the reference. Default is False.
  --ref_gbk, -k         Download genbank files of the reference. Default is
                        False.
```

- **Additional arguments**

```
additional arguments:
  --ref_ptt, -p         Download ptt files of the reference. Default is False.
  --ref_rnt, -r         Download rnt files of the reference. Default is False.
  --convert_embl, -e    Convert gbk to embl files of the reference. Default is
                        False.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files will be stored in `$ANNOgesic_folder/input/reference`

Output folder names are following:

fasta: Fasta files.

annotation: Annotation files.

1.4.6 update_genome_fasta (update reference genome fasta file)

If fasta files of the reference genomes do not exist, `update_genome_fasta` can update fasta files from the closely related genomes to our reference genomes via searching the mutations. Therefore, a table of mutation information is required. For the format of the table, please check mutation table. Titles of the columns are presented on the top and they need to start with #. The format is basically the same as VCF format.

If no mutation information is provided, `snp` can be used for detecting mutations. (one module of `ANNOgesic`). Please check the section of *snp (SNP calling)*.

- **Required files**

Fasta files of reference genome

Mutation table: Contains the information of mutations between related and reference genomes. For an example, please check mutation table.

- **Arguments**

```
usage: annogesic update_genome_fasta [-h] --project_path PROJECT_PATH
                                      --related_fasta_files RELATED_FASTA_FILES
                                      [RELATED_FASTA_FILES ...]
                                      --mutation_table MUTATION_TABLE
                                      --updated_seq_name UPDATED_SEQ_NAME

optional arguments:
  -h, --help            show this help message and exit

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --related_fasta_files RELATED_FASTA_FILES [RELATED_FASTA_FILES ...], -c RELATED_
↪FASTA_FILES [RELATED_FASTA_FILES ...]
                        Path of the genome fasta files of the closely related
                        species.
  --mutation_table MUTATION_TABLE, -m MUTATION_TABLE
                        Path of the mutation table which stores the mutation
                        information between the reference genome and genome of
                        the closely related species. For an example check
                        https://github.com/Sung-
                        Huan/ANNOgesic/blob/master/tutorial_data/mutation.csv
  --updated_seq_name UPDATED_SEQ_NAME, -u UPDATED_SEQ_NAME
                        The file name of the updated sequence. The output
                        fasta file name will be --updated_seq_name.fa.
```

- **Output files**

Fasta files of updated genome: The updated fasta files are stored in `$ANNOgesic_folder/output/updated_references/fasta_files`.

1.4.7 annotation_transfer (annotation transfer)

`annotation transfer` is the subcommand for transferring the annotation from the closely related genomes to the reference genomes. To achieve this, RATT is integrated in ANNOgesic. The higher similarity between closely related genomes and reference genomes are, the more precise the performance is. Before running `annotation transfer`, please run `source $PAGIT_HOME/sourceme.pagit` first. it will modify the path for executing RATT. If you use Dockerfile to execute ANNOgesic, the path modification can be skipped. If the error message related to 'defined(@array)' shows, please check *FAQ*.

- **Required tools**

RATT.

- **Required files**

Annotation files of the closely related genomes: Genbank/embl files of the closely related genomes.

Fasta files of the closely related genomes

Fasta files of the updated genomes

- **Basic arguments**

```
usage: annogesic annotation_transfer --project_path PROJECT_PATH
                                     --compare_pair COMPARE_PAIR
                                     [COMPARE_PAIR ...]
                                     [--related_embl_files RELATED_EMBL_FILES [RELATED_
↪EMBL_FILES ...]]
                                     [--related_gbk_files RELATED_GBK_FILES [RELATED_GBK_
↪FILES ...]]
                                     --related_fasta_files RELATED_FASTA_FILES
                                     [RELATED_FASTA_FILES ...]
                                     --target_fasta_files TARGET_FASTA_FILES
                                     [TARGET_FASTA_FILES ...]
                                     [--additional arguments]

optional arguments:
  -h, --help            show this help message and exit

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --compare_pair COMPARE_PAIR [COMPARE_PAIR ...], -p COMPARE_PAIR [COMPARE_PAIR ...]
                        Please assign the sequence identifier of genome pairs,
                        e.g. NC_007795:NEW_NC_007795. The related genome is
                        NC_007795 and the target genome is NEW_NC_007795. The
                        assigned names are the headers of the fasta file
                        (start with ">"), not the filename of fasta file. If
                        the headers contain space or '|', only the string from
                        '>' to the first space or '|' will be considered as
                        the name for --compare_pair (normally this part is the
                        accession number). If multiple sequences need to be
                        assigned, please use spaces to separate them.
  --related_embl_files RELATED_EMBL_FILES [RELATED_EMBL_FILES ...], -ce RELATED_EMBL_
↪FILES [RELATED_EMBL_FILES ...]
                        The paths of the embl files of the related species. If
                        --related_embl_files is assigned, --related_gbk_files
                        is not needed.
  --related_gbk_files RELATED_GBK_FILES [RELATED_GBK_FILES ...], -cg RELATED_GBK_
↪FILES [RELATED_GBK_FILES ...]
```

(continues on next page)

(continued from previous page)

```
                        The paths of the genbank files of the related species.
                        The genbank can be ended by .gbk, .gbff or .gb. If
                        --related_gbk_files is assigned, --related_embl_files
                        is not needed.
  --related_fasta_files RELATED_FASTA_FILES [RELATED_FASTA_FILES ...], -cf RELATED_
↪FASTA_FILES [RELATED_FASTA_FILES ...]
                        The paths of the fasta files of the related species.
  --target_fasta_files TARGET_FASTA_FILES [TARGET_FASTA_FILES ...], -tf TARGET_FASTA_
↪FILES [TARGET_FASTA_FILES ...]
                        The paths of the target fasta files.
```

- **Additional arguments**

```
additional arguments:
  --ratt_path RATT_PATH
                        Path of the start.ratt.sh file of RATT folder. Default
                        is start.ratt.sh.
  --element ELEMENT, -e ELEMENT
                        --element will become the prefix of all output
                        file.Default is "chromosome".
  --transfer_type TRANSFER_TYPE, -t TRANSFER_TYPE
                        The transfer type for running RATT. (For the details,
                        please refer to the manual of RATT.) Default is
                        Strain.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files from RATT will be stored in `$ANNOgesic_folder/output/annotation_transfer`.

Annotation files (`.gff`, `.ptt`, `.rnt`) will be stored in `$ANNOgesic_folder/output/updated_references/annotations`.

1.4.8 snp (SNP calling)

`snp` can analyze the alignment files and fasta files to detect mutations by running Samtools and Bcftools. There are multiple programs which can be applied to detect mutations (with BAQ, without BAQ and extend BAQ) and there are multiple flag options to set filters (QUAL, DP, DP4, etc.). Moreover, `snp` can also be used for generating the fasta files of reference genomes if it is necessary.

- **Required tools**

Samtools.

Bcftools.

- **Required files**

BAM files: BAM files from fragmented/conventional libraries or TEX +/- treated libraries both can be accepted. For assigning the files, please follow the format – `$SET_NAME:$BAMFILE1,$BAMFILE2,...`. For an example, the user has four bam files of one genome. Then the input will be `set1:sample1.bam,sample2.bam,sample3.bam,sample4.bam`.

Fasta files of the closely related genomes or **Fasta files of the reference genomes**

- **Basic arguments**

```
usage: annogesic snp [-h] --project_path PROJECT_PATH --bam_type
                     {related_genome,reference_genome} --program
                     {with_BAQ,without_BAQ,extend_BAQ}
                     [{with_BAQ,without_BAQ,extend_BAQ} ...] --fasta_files
                     FASTA_FILES [FASTA_FILES ...] --bam_files BAM_FILES
                     [BAM_FILES ...] [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --bam_type {related_genome,reference_genome}, -t {related_genome,reference_genome}
                        If the BAM files are produced by mapping to a related
                        genome, please assign "related_genome". the mutations
                        between the related genome and the refernce genome can
                        be detected for generating sequence of the query
                        genome. If the BAM files are produced by mapping to
                        the reference genome, please assign
                        "reference_genome". The mutations of reference genome
                        can be detected.
  --program {with_BAQ,without_BAQ,extend_BAQ} [{with_BAQ,without_BAQ,extend_BAQ} ...],
↪ -p {with_BAQ,without_BAQ,extend_BAQ} [{with_BAQ,without_BAQ,extend_BAQ} ...]
                        The program for detecting SNPs: "with_BAQ",
                        "without_BAQ", "extend_BAQ". Multi-programs can be
                        executed at the same time (separated by spaces). For
                        example, with_BAQ without_BAQ extend_BAQ.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --bam_files BAM_FILES [BAM_FILES ...], -b BAM_FILES [BAM_FILES ...]
                        Path of input BAM files. Required format:
                        $SET_NAME:$BAM1,$BAM2,... . If multiple sets need to
                        be assigned, please use spaces to separate them.
```

- **Additional arguments**

```
additional arguments:
  --samtools_path SAMTOOLS_PATH
                        Path of samtools.
  --bcftools_path BCFTOOLS_PATH
                        Path of bcftools.
  --quality QUALITY, -q QUALITY
                        The minimum quality score of a mutation. Default is
                        40.
  --read_depth_range READ_DEPTH_RANGE, -d READ_DEPTH_RANGE
                        The range of read depth for a mutation. The format is
                        $MIN,$MAX. It can be assigned by different types: 1.
                        real number ("r"), 2. times of the number of bam files
                        (counted from --bam_files) ("n") or 3. times of the
                        average read depth ("a"). For example, n_10,a_3 is
                        assigned. If the average read depth is 70 and 4 bam
                        files are provided, n_10 will be 40 and a_3 will be
                        140 (average read depth * 3). Based on the same
                        example, if this value is r_10,a_3, the minimum read
                        depth will become exact 10 reads. If "none" is
                        assigned, read depth will not be considered. Default
                        is n_10,none.
  --ploidy {haploid,diploid}, -pl {haploid,diploid}
                        The query genome is haploid or diploid. Default is
```

(continues on next page)

(continued from previous page)

```
                        haploid.
  --rg_tag, -R          For one BAM file which includes multiple samples
                        (opposite of --ignore-RG in samtools). Default is
                        False.
  --caller {c,m}, -c {c,m}
                        The types of caller - consensus-caller or
                        multiallelic-caller. For details, please check
                        documentation of bcftools. "c" represents consensus-
                        caller. "m" represents multiallelic-caller. Default is
                        m.
  --dp4_cutoff DP4_CUTOFF, -D DP4_CUTOFF
                        The cutoff of DP4. The format is
                        $MIN_SNP_READS_NUMBER:$MIN_SNP_READS_RATIO.
                        $MIN_SNP_READS_NUMBER can be assigned by three types:
                        1. fix number ("r"), 2. times of the number of bam
                        files (counted from --bam_files) ("n") or 3. times of
                        average read depth ("a"). For example, n_10,0.8. If
                        the BAM files are 4, it means the minimum mapped reads
                        of a SNP is 40 (4 * 10), and the minimum ratio of
                        mapped read of a SNP (mapped reads of a SNP / total
                        reads) is 0.8. Default is n_10,0.8.
  --indel_fraction INDEL_FRACTION, -if INDEL_FRACTION
                        The minimum IDV and IMF which supports for insertion
                        of deletion. The minimum IDV can be assigned by
                        different types: 1. fix number ("r"), 2. times of the
                        number of bam files (assigned by --bam_files) ("n") or
                        3. times of the average read depth ("a"). The input
                        format is $MIN_IDF:$MIN_IMF. For example, The value is
                        n_10,0.8 and 4 BAM files are assigned. The minimum IDV
                        is 40, and the minimum IMF is 0.8. Default is
                        n_10,0.8.
  --filter_tag_info FILTER_TAG_INFO [FILTER_TAG_INFO ...], -ft FILTER_TAG_INFO␣
↪[FILTER_TAG_INFO ...]
                        For using more filters to improve the detection.
                        Please assign 1. the name of a tag, 2. bigger ("b") or
                        smaller ("s") and 3. the value of the filter. For
                        example, "RPB_b0.1,MQ0F_s0" means that RPB should be
                        bigger than 0.1 and MQ0F should be smaller than 0.
                        Default is RPB_b0.1,MQSB_b0.1,MQB_b0.1,BQB_b0.1.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

If `bam_type` is `related_genome`, the results will be stored in `$ANNOgesic/output/SNP_calling/compare_related_and_reference_genomes`. If `bam_type` is `reference_genome`, the results will be stored in `$ANNOgesic/output/SNP_calling/mutations_of_reference_genomes`.

The output folders and results are following:

SNP_raw_output: Stores output tables which be only considered read depth and QUAL.

VCF Table (only consider read depth and QUAL): Filename is `$GENOME_$PROGRAM_$SET.vcf`.

SNP_table: Stores two types of output tables

VCF Table (consider all filters): Filename is `$GENOME_$PROGRAM_$SET_best.vcf`.

Index of fasta files:: Filename is `$GENOME_$PROGRAM_$SET_seq_reference.csv`. The meaning of this file is like following example:

```
Staphylococcus_aureus_HG003     1632629 .       AaA     AA      57      .
Staphylococcus_aureus_HG003     1632630 .       aA      a       57      .
Staphylococcus_aureus_HG003     1499572 .       T       TT,TTTTT        43.8525 .
```

The example contains `position conflict` and `mutation conflict`. As a result, the conflicts will affect the other mutation's positions. Therefore, it will generate four different fasta files. The first two lines are `position conflict`, and the last line is `mutation conflict`. `$GENOME_$PROGRAM_$SET_seq_reference.csv` is the index for these four fasta files.

```
1       1632629 1       1499572:TT      Staphylococcus_aureus_HG003
1       1632629 2       1499572:TTTTT   Staphylococcus_aureus_HG003
2       1632630 1       1499572:TT      Staphylococcus_aureus_HG003
2       1632630 2       1499572:TTTTT   Staphylococcus_aureus_HG003
```

The first column is the index of the `position conflict`. The second column is the selected position. The third one is the index of the `mutations conflict`. The fourth one is the selected position and nucleotides. The last column is the genome name.

Potential fasta files: Filename is `$FASTANAME_$SET_$STRIANNAME_$INDEXofPOSITIONCONNFLICT_$INDEXofMUTATI` `fa`, and it is stored in `$ANNOgesic/output/SNP_calling/$BAM_TYPE/seqs`. Based on the example in **Index of fasta files**, `Staphylococcus_aureus_HG003_set1_Staphylococcus_aureus_HG003_1_1.fa` will be generated based on the first line of `$GENOME_$PROGRAM_seq_reference.csv`. `Staphylococcus_aureus_HG003_set1_Staphylococcus_aureus_HG003_1_2.fa` and will be generated based on the second line of `$GENOME_$PROGRAM_seq_reference.csv` and so forth.

statistics: Stores the statistic files and figures, ex: the distribution of SNPs based on QUAL.

1.4.9 tss_ps (TSS and processing site prediction)

`tss_ps` can generate the TSS and processing sites via running TSSpredator. Since the parameters can affect the results strongly, `optimize_tss_ps` can obtain the optimized parameters of TSSpredator. Please check the section *optimize_tss_ps (optimization of TSS and processing site detection)* for details.

- **Required tools**

TSSpredator.

- **Required files**

Wiggle files of TEX +/-: Please check the section *The input format of libraries for running ANNOgesic* for assigning correct format.

Fasta files of the reference genomes

GFF files of the reference genomes

- **Optional input files**

Gff files of the manual detected TSSs: If gff file of the manual detected TSSs can be provided, `tss_ps` can merge the manual detected TSSs and TSSpredator predicted ones.

Gff files of transcripts: If comparing TSSs with transcripts is required, gff files of the transcripts need to be assigned. For the transcripts, please check the section *transcript (transcript detection)*.

- **Basic arguments**

```
usage: annogesic tss_ps --project_path PROJECT_PATH --program {TSS,PS}
                        --fasta_files FASTA_FILES [FASTA_FILES ...]
                        --annotation_files ANNOTATION_FILES
                        [ANNOTATION_FILES ...] --tex_notex_libs TEX_NOTEX_LIBS
                        [TEX_NOTEX_LIBS ...]
                        --condition_names CONDITION_NAMES
                        [CONDITION_NAMES ...]
                        [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --program {TSS,PS}, -p {TSS,PS}
                        The feature to predict. Please assign "TSS" or "PS".
                        Default is "TSS".
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs, etc.
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
                        TEX+/- wig files for TSSpredator. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_tex_forward.wig:tex:1:a:+
                        my_lib_tex_reverse.wig:tex:1:a:-.
  --replicate_tex REPLICATE_TEX [REPLICATE_TEX ...], -rt REPLICATE_TEX [REPLICATE_TEX␣
↪...]
                        This value is the minimal number of replicates that a
                        TSS has to be detected in. The format is
                        $NUMBERofCONDITION_$NUMBERofREPLICATE. If different
                        --replicate_tex values need to be assigned to
                        different conditions, please use spaces to separate
                        them. For example, 1_2 2_2 3_3 means that
                        --replicate_tex is 2 in number 1 and number 2
                        conditions. In number 3 condition, --replicate_tex is
                        3. For assigning the same --replicate_tex to all
                        conditions, just use like all_1 (--replicate_tex is 1
                        in all conditions).
  --condition_names CONDITION_NAMES [CONDITION_NAMES ...], -cn CONDITION_NAMES␣
↪[CONDITION_NAMES ...]
                        The output prefix of all conditions. If multiple
                        conditions need to be assigned, please use spaces to
                        separate them. For an example, prefix_condition1
                        prefix_condition2.
  --output_id OUTPUT_ID, -oi OUTPUT_ID
                        The tag of attributes will be used for TSS
                        classification. Default is locus_tag.
```

- **Additional arguments**

```
additional arguments:
  --tsspredator_path TSSPREDATOR_PATH
```

(continues on next page)

(continued from previous page)

```
                        Path of TSSpredator. Default is
                        /usr/local/bin/TSSpredator.jar
  --auto_load_optimized_parameters AUTO_LOAD_OPTIMIZED_PARAMETERS, -ap AUTO_LOAD_
↪OPTIMIZED_PARAMETERS
                        If optimize_tss_ps has been done before,
                        --auto_load_optimized_parameters can automatically
                        load the optimized parameters for the detection by
                        specifying the folder of "optimized_TSSpredator"
                        containing all output of "optimize_tss_ps".If the
                        parameters need to be assigned manually, please do not
                        turn on this function and assign the parameters to
                        --parameter_sets. Default is None.
  --genome_order GENOME_ORDER [GENOME_ORDER ...]
                        It only works when --auto_load_optimized_parameters is
                        off. It is the order for applying the parameters
                        (--height, --height_reduction, --factor,
                        --factor_reduction, --base_height,
                        --enrichment_factor, and --processing_factor) of
                        TSSpedator to genomes, ex: if --genome_order is
                        "NC_002505.1 NC_002506.1" and --height is "0.3 0.5",
                        0.3 will be used as --height to NC_002505.1 and 0.5
                        will be used to NC_002506.1. Default is applying one
                        parameter set to all genomes.
  --height HEIGHT [HEIGHT ...], -he HEIGHT [HEIGHT ...]
                        It only works when --auto_load_optimized_parameters is
                        off. This value relates to the minimal number of read
                        starts at a certain genomic position to be considered
                        as a TSS candidate. If using different --height to
                        multiple genomes is required, please use spaces to
                        split the input values. The order should match
                        --genome_order. For the details, please check
                        --geonme_order. Default is 0.3.
  --height_reduction HEIGHT_REDUCTION [HEIGHT_REDUCTION ...], -rh HEIGHT_REDUCTION␣
↪[HEIGHT_REDUCTION ...]
                        It only works when --auto_load_optimized_parameters is
                        off. When comparing different genomes/conditions and
                        the step height threshold is reached in at least one
                        genome/condition, the threshold is reduced for the
                        other genomes/conditions by the value set here. This
                        value must be smaller than the step height threshold.
                        If using different --height_reduction to multiple
                        genomes is required, please use spaces to split the
                        input values. The order should match --genome_order.
                        For the details, please check --geonme_order. Default
                        is 0.2.
  --factor FACTOR [FACTOR ...], -fa FACTOR [FACTOR ...]
                        It only works when --auto_load_optimized_parameters is
                        off. The minimal factor by which the TSS height has to
                        exceed the local expression background. If using
                        different --factor to multiple genomes is required,
                        please use spaces to split the input values. The order
                        should match --genome_order. For the details, please
                        check --geonme_order. Default is 2.0.
  --factor_reduction FACTOR_REDUCTION [FACTOR_REDUCTION ...], -rf FACTOR_REDUCTION␣
↪[FACTOR_REDUCTION ...]
                        It only works when --auto_load_optimized_parameters is
                        off. When comparing different genomes/conditions and
```

(continues on next page)

(continued from previous page)

```
                        the step factor threshold is reached in at least one
                        genome/condition, the threshold is reduced for the
                        other genomes/conditions by the value set here. This
                        value must be smaller than the step factor threshold.
                        If using different --factor_reduction to multiple
                        genomes is required, please use spaces to split the
                        input values. The order should match --genome_order.
                        For the details, please check --geonme_order. Default
                        is 0.5.
  --enrichment_factor ENRICHMENT_FACTOR [ENRICHMENT_FACTOR ...], -ef ENRICHMENT_
↪FACTOR [ENRICHMENT_FACTOR ...]
                        It only works when --auto_load_optimized_parameters is
                        off. The minimal enrichment factor. If using different
                        --enrichment_factor to multiple genomes is required,
                        please use spaces to split the input values. The order
                        should match --genome_order. For the details, please
                        check --geonme_order. Default is 2.0.
  --processing_factor PROCESSING_FACTOR [PROCESSING_FACTOR ...], -pf PROCESSING_
↪FACTOR [PROCESSING_FACTOR ...]
                        It only works when --auto_load_optimized_parameters is
                        off. The minimal processing factor. If the value for
                        the untreated library is higher than the treated
                        library the positionsis considered as a processing
                        site and not annotated as detected. If using different
                        --processing_factor to multiple genomes is required,
                        please use spaces to split the input values. The order
                        should match --genome_order. For the details, please
                        check --geonme_order. Default is 1.5.
  --base_height BASE_HEIGHT [BASE_HEIGHT ...], -bh BASE_HEIGHT [BASE_HEIGHT ...]
                        It only works when --auto_load_optimized_parameters is
                        off. The minimal number of reads should be mapped on
                        TSS/PS. If using different --base_height to multiple
                        genomes is required, please use spaces to split the
                        input values. The order should match --genome_order.
                        For the details, please check --geonme_order. Default
                        is 0.0.
  --utr_length UTR_LENGTH, -u UTR_LENGTH
                        The length of UTRs. Default is 300.
  --tolerance TOLERANCE, -to TOLERANCE
                        The 5'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated TSSs (--compare_transcript_files is
                        provided). Default is 5.
  --cluster CLUSTER, -c CLUSTER
                        This value defines the maximal distance (nucleotides)
                        between TSS candidates have to be clustered together.
                        If the distance between these multiple TSSs is smaller
                        or equal to this value, only one of them will be
                        printed out. Default is 2.
  --manual_files MANUAL_FILES [MANUAL_FILES ...], -m MANUAL_FILES [MANUAL_FILES ...]
                        If gff files of the manual checked TSS are provided,
                        this function will merge manual checked ones and
                        TSSpredator predicted ones. Please assign the path of
                        manual-checked TSS gff files.
  --curated_sequence_length CURATED_SEQUENCE_LENGTH [CURATED_SEQUENCE_LENGTH ...], -
↪le CURATED_SEQUENCE_LENGTH [CURATED_SEQUENCE_LENGTH ...]
                        The length of the sequence used for the manual set of
```

(continues on next page)

(continued from previous page)

```
                        TSS/PS. This value is required to calculate the
                        accurracy. If the whole genome was used write "all".
                        Otherwise use the name of the reference sequence in
                        the folowing format: $GENOME:SLENGTH. Multiple entries
                        are accepted. For an example, test.gff contains two
                        sequences s1 and s2. For s1 100 kb were checked while
                        for s2 the whole sequence was curated. The value of
                        this argument would be s1:100000 s2:all. Per default
                        all the full length of all sequences will be used.
  --validate_gene, -v   Using TSS candidates to validate genes in annotation
                        file. it will be store in statistics folder. Default
                        is False.
  --compare_transcript_files COMPARE_TRANSCRIPT_FILES [COMPARE_TRANSCRIPT_FILES ...],␣
↪-ta COMPARE_TRANSCRIPT_FILES [COMPARE_TRANSCRIPT_FILES ...]
                        If the paths of transcript gff files are provided,
                        this function will compare TSS and transcript to
                        obtain the overlap information. Default is False.
  --re_check_orphan, -ro
                        If there is no information of gene or locus_tag in
                        genome annotation gff file, all TSSs will be assigned
                        to orphan TSSs by TSSpredator. The function can
                        compare TSSs with CDSs to classify the TSS correctly.
                        Default is False.
  --remove_overlap_feature, -of
                        If a processing site and a TSS are overlaping, keep
                        "TSS", The predicted feature (based on --program) will
                        be removed. Default is False.
  --compare_overlap_gff COMPARE_OVERLAP_GFF [COMPARE_OVERLAP_GFF ...], -rg COMPARE_
↪OVERLAP_GFF [COMPARE_OVERLAP_GFF ...]
                        If --overlap_feature is "TSS" or "PS",
                        --reference_gff_files need to be assigned. For TSS
                        prediction, please assign the path of processing site.
                        For processing site prediction, please assign the path
                        of TSS. Don't use this flag if --overlap_feature is
                        "both".
  --remove_low_expression REMOVE_LOW_EXPRESSION, -rl REMOVE_LOW_EXPRESSION
                        For removing the low expressed TSS by comparing the
                        manual detected TSSs and predicted ones. Please assign
                        the manual-checked TSS in gff format.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

The results of TSS are stored in `$ANNOgesic/output/TSSs`, and the results of processing site are stored in `$ANNOgesic/output/processing_sites`.

The output folders are following:

MasterTables: MasterTable from TSSpredator.

statistics: Statistic files.

Venn Figures of TSS types: Filename is `TSS_venn_$GENOME.png`.

TSS types with corresponding amounts: Table is `stat_TSS_class_$GENOME.csv`, and Figure is `TSS_class_$GENOME.png`.

Conditions with corresponding amounts: `stat_TSS_libs_$GENOME.csv` stores all combination

of conditions with corresponding amounts. `TSSstatistics.tsv` stores the number of TSS which can be detected or missing in each condition.

Comparing TSSs with other features: `stat_compare_TSS_transcript_$GENOME.csv` is for comparing TSSs with transcripts. `stat_gene_vali_$GENOME.csv` is for comparing TSS with genome annotations like CDSs.

Comparing manual detected TSSs and predicted TSSs: In `stat_compare_TSSpredator_manual_$GENOME.csv`, the accuracy of TSS prediction can be found.

configs: Configuration files for running TSSpredator.

gffs: Output gff files of TSSs. Some useful information can be found in the tags of the attributes within the TSS gff files. Based on this information, we can know the details of the specific TSS. The tags are as following:

method: Stores the information that this TSS is detected by manual detection or TSSpredator.

type: TSS type of this TSS. It could be Primary, Secondary, Internal, Antisense or Orphan.

utr_length: UTR length of this TSS.

associated_gene: Which genes are associated with this TSS.

Parent: Presents the parent transcripts of this TSS, if the user has compared TSS with the transcript.

libs: Shows in which libraries the TSS can be detected.

1.4.10 transcript (transcript detection)

`transcript` can detect transcripts based on the coverage. Most of the transcript assembly tools are focus on eukaryotic transcript. Due to this, we constructed a subcommand which is based on the nucleotide coverage data, given gene annotations and several parameters that can be set by the user.

- **Required files**

Wiggle files of fragmented/conventional libraries or TEX+/- treated libraries: For importing the information about libraries, please check the section *The input format of libraries for running ANNOgesic*.

- **Optional input files**

TSS gff files: If the user wants to compare transcripts with TSSs, TSS gff files are required. Please check the section *tss_ps (TSS and processing site prediction)*.

Gff files of genome anntations containing CDSs, tRNAs, rRNAs, etc: If the user wants to compare transcripts with genome annotations or modify transcript based on genome annotations like CDSs, tRNAs, rRNAs, genome annotation gff files are required. There are four options for modification of transcripts:

merge_overlap: If multiple transcripts overlap the same gene, they will be merged as one complete transcript.

extend_3end: If the transcript starts at the upstream of the gene and ends within the gene, the end point of the transcript will be extended to the end point of gene.

extend_5end: If the transcript starts within the gene and ends at the downstream of gene, the starting point of the transcript will be extended to the starting point of the gene.

within_extend_ends: If the transcript is within the gene, the two ends of the transcript will be extended to the two ends of gene.

none: Transcripts will not be modified by the genome annotations

- **Basic arguments**

```
usage: annogesic transcript --project_path PROJECT_PATH
                        [--annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...]]
                        [--modify_transcript {merge_overlap,extend_3end,extend_5end,
↪within_extend_ends,none} [{merge_overlap,extend_3end,extend_5end,within_extend_ends,
↪none} ...]]
                        [--tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...]]
                        [--frag_libs FRAG_LIBS [FRAG_LIBS ...]]
                        [--replicate_tex REPLICATE_TEX [REPLICATE_TEX ...]]
                        [--replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...]]
                        [--tex_notex {1,2}] [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs, etc. TSS gff files and terminator
                        gff files need to be separately assigned to
                        --tss_files and --terminator_files, respectively.
  --modify_transcript {merge_overlap,extend_3end,extend_5end,within_extend_ends,none}␣
↪[{merge_overlap,extend_3end,extend_5end,within_extend_ends,none} ...], -mt {merge_
↪overlap,extend_3end,extend_5end,within_extend_ends,none} [{merge_overlap,extend_
↪3end,extend_5end,within_extend_ends,none} ...]
                        If --annotation_files is provided, the post-
                        modification of transcript based on genome annotations
                        can be assigned. There are five opetions. 1.
                        "merge_overlap": if multiple transcripts overlap with
                        the same gene, they will be merged as one complete
                        transcript. 2. "extend_3end": if a transcript starts
                        at the upstream of a gene and ends within the gene,
                        the end point of the transcript will be extended to
                        the end point of the gene. 3. "extend_5end": if a
                        transcript starts within a gene and ends at the
                        downstream of gene, the starting point of the
                        transcript will be extended to the starting point of
                        the gene. 4. "within_extend_ends": if a transcript is
                        within a gene, the two ends of the transcript will be
                        extended to the two ends of gene. 5. "none": the
                        transcript will not be modified by the genome
                        annotations. For using mutliple modifications, please
                        separated them by spaces. Default is merge_overlapped.
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
                        TEX+/- wig files. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_tex_forward.wig:tex:1:a:+
                        my_lib_tex_reverse.wig:tex:1:a:-.
  --frag_libs FRAG_LIBS [FRAG_LIBS ...], -fl FRAG_LIBS [FRAG_LIBS ...]
                        Wig files of RNA-Seq with transcript fragmented. The
                        format is: wig_file_path:frag:condition_id(integer):re
                        plicate_id(alphabet):strand(+ or -). If multiple wig
                        files need to be assigned, please use spaces to
                        separate the wig files. For example,
```

(continues on next page)

(continued from previous page)

```
                        my_lib_frag_forward.wig:frag:1:a:+
                        my_lib_frag_reverse.wig:frag:1:a:-.
  --replicate_tex REPLICATE_TEX [REPLICATE_TEX ...], -rt REPLICATE_TEX [REPLICATE_TEX␣
↪...]
                        This value is the minimal number of replicates that a
                        transcript has to be detected in. The format is
                        $NUMBERofCONDITION_$NUMBERofREPLICATE. If different
                        --replicate_tex values need to be assigned to
                        different conditions, please use spaces to separate
                        them. For example, 1_2 2_2 3_3 means that
                        --replicate_tex is 2 in number 1 and number 2
                        conditions. In number 3 condition, --replicate_tex is
                        3. For assigning the same --replicate_tex to all
                        conditions, just use like all_1 (--replicate_tex is 1
                        in all conditions).
  --replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...], -rf REPLICATE_FRAG [REPLICATE_
↪FRAG ...]
                        Similar to --replicates_tex. This value is for
                        fragmented (or conventional) libraries.
  --tex_notex {1,2}, -te {1,2}
                        The value is for TEX+/- libraries to decide the
                        transcript should be detected in both (TEX+ and TEX-,
                        assigned by 2) or can be detected in only one library
                        (TEX+ or TEX-, assigned by 1). Please assign 1 or 2.
                        Default is 1.
```

- **Additional arguments**

```
additional arguments:
  --length LENGTH, -l LENGTH
                        The minimum length of the transcript after modifying
                        base on genome annotation. Default is 20.
  --height HEIGHT, -he HEIGHT
                        The minimum coverage of the transcript. If --tex_notex
                        is 2, the average coverage of TEX+ and TEX- libraries
                        should be higher than this value. The default is 10.
  --width WIDTH, -w WIDTH
                        The minimum length of the transcript without modifying
                        by genome annotation. The default is 20.
  --tolerance TOLERANCE, -t TOLERANCE
                        The number of nucleotides which coveraes can drop
                        below the --height in a transcript. The default is 5.
  --tolerance_coverage TOLERANCE_COVERAGE, -tc TOLERANCE_COVERAGE
                        The minimum coverage of the nucleotides which match
                        the situation of --tolerance, Default is 0.
  --tss_files TSS_FILES [TSS_FILES ...], -ct TSS_FILES [TSS_FILES ...]
                        Paths of TSS files for comparing transcripts and TSSs.
  --compare_feature_genome COMPARE_FEATURE_GENOME [COMPARE_FEATURE_GENOME ...], -cf␣
↪COMPARE_FEATURE_GENOME [COMPARE_FEATURE_GENOME ...]
                        If --compare_genome_annotation is provided, please
                        assign the feature for comparing. Multiple features
                        can be separated by spaces. Default is None.
  --tss_tolerance TSS_TOLERANCE, -tt TSS_TOLERANCE
                        The 5'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated TSSs (--tss_files is provided). Default is
                        5.
```

(continues on next page)

(continued from previous page)

```
  --terminator_files TERMINATOR_FILES [TERMINATOR_FILES ...], -e TERMINATOR_FILES␣
↪[TERMINATOR_FILES ...]
                        Paths of terminator gff files for comparing
                        transcripts and terminators. Default is None.
  --terminator_tolerance TERMINATOR_TOLERANCE, -et TERMINATOR_TOLERANCE
                        The 3'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated terminators. Default is 30.
  --max_length_distribution MAX_LENGTH_DISTRIBUTION, -mb MAX_LENGTH_DISTRIBUTION
                        For generating the figure of distribution of
                        transcript length, please assign the maximum length.
                        Default is 2000.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/transcripts`.

The generated output folders are as following:

tables: Table of transcript with more details. The meanings of the columns in the table are following:

> **Genome:** Genome name.
>
> **Name:** Transcript name in the gff file.
>
> **Start:** Starting point of this transcript.
>
> **End:** End point of this transcript.
>
> **Strand:** Strand of this transcript.
>
> **Detect_lib_type:** This transcript can be detected in fragmented/conventional or TEX+/- libraries.
>
> **Associated_gene:** Which genes are associated with this transcript.
>
> **Associated_tss:** Which TSSs are located on this transcript.
>
> **Associated_term:** Which terminators are associated with this transcript.
>
> **Avg_coverage:$LIB_NAME:** Stores the average coverage information of the libraries about this transcript.

statistics: Stores statistic files.

> **Comparing transcript with other features:** `stat_compare_transcript_genome_$GENOMENAME.csv` is for comparing transcript with genome annotation like CDSs, `stat_compare_transcript_TSS_$GENOMENAME.csv` is for comparing transcript with TSS, and `stat_compare_transcript_terminator_$GENOMENAME.csv` is for comparing transcript with terminator.
>
> **Figure of the distribution of transcript length:** `$GENOME_length_all.png` is for analyzing of all transcript length. `$GENOME_length_less_$LENGTH.png` is for the analyzing of the assigned length.

gffs: Stores gff files of transcripts. Some useful information can be found in the tags of the attributes within the transcript gff file. Based on this information, we can know the details of the specific transcript. The tags are as following:

compare_$FEATURE: State of overlap between transcripts and features (If `--compare_feature_genome` and `--annotation_files` are assigned). The value may be `cover`, `right_shift`, `left_shift`, `within` or `no_related`.

associated_tss: Shows which TSSs are located on this transcript (If `--tss_files` is assigned).

associated_term: Shows which terminators are located on this transcript (If `--terminator_files` is assigned).

associated_$FEATURE: Shows that the features are located on this transcript (If `--compare_feature_genome` and `--annotation_files` are assigned).

detect_lib: This transcript is detected by Tex-treated libraries or fragmented/conventional libraries.

best_avg_coverage: The average coverage of the highest expressed library within this transcript.

1.4.11 terminator (terminator detection)

`terminator` will predict the rho-independent terminators. `ANNOgesic` combines the results of two methods in order to get more reliable candidates. The first method is using TranstermHP. The other one detects the specific secondary structure between converging pairs of transcripts and CDSs. `ANNOgesic` can check the coverages in order to generate the terminators which have coverage significant decrease.

- **Required tools**

TranstermHP

RNAfold of ViennaRNA.

- **Required files**

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

Fasta files of the genome sequences

Wiggle files of TEX +/- treated libraries or fragmented/conventional libraries

Gff files of the transcripts: Please check the section *transcript (transcript detection)*.

- **Basic arguments**

```
usage: annogesic terminator --project_path PROJECT_PATH --fasta_files
                            FASTA_FILES [FASTA_FILES ...] --annotation_files
                            ANNOTATION_FILES [ANNOTATION_FILES ...]
                            --transcript_files TRANSCRIPT_FILES
                            [TRANSCRIPT_FILES ...]
                            [--tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...]]
                            [--frag_libs FRAG_LIBS [FRAG_LIBS ...]]
                            [--tex_notex {1,2}]
                            [--replicate_tex REPLICATE_TEX [REPLICATE_TEX ...]]
                            [--replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...]]
                            [--additional arguments]


basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
```

(continues on next page)

(continued from previous page)

```
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs, etc. Transcript gff files and sRNA
                        gff files need to be separately assigned to
                        --transcript_files and --srna_files, respectively.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
                        Paths of the transcript gff files.
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
                        TEX+/- wig files. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_tex_forward.wig:tex:1:a:+
                        my_lib_tex_reverse.wig:tex:1:a:-.
  --frag_libs FRAG_LIBS [FRAG_LIBS ...], -fl FRAG_LIBS [FRAG_LIBS ...]
                        Wig files of RNA-Seq with transcript fragmented. The
                        format is: wig_file_path:frag:condition_id(integer):re
                        plicate_id(alphabet):strand(+ or -). If multiple wig
                        files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_frag_forward.wig:frag:1:a:+
                        my_lib_frag_reverse.wig:frag:1:a:-.
  --tex_notex {1,2}, -te {1,2}
                        The value is for TEX+/- libraries to decide the
                        terminator should be detected in both (TEX+ and TEX-,
                        assigned by 2) or can be detected in only one library
                        (TEX+ or TEX-, assigned by 1). Please assign 1 or 2.
                        Default is 1.
  --replicate_tex REPLICATE_TEX [REPLICATE_TEX ...], -rt REPLICATE_TEX [REPLICATE_TEX␣
↪...]
                        This value is the minimal number of replicates that a
                        terminator has to be detected in. The format is
                        $NUMBERofCONDITION_$NUMBERofREPLICATE. If different
                        --replicate_tex values need to be assigned to
                        different conditions, please use spaces to separate
                        them. For example, 1_2 2_2 3_3 means that
                        --replicate_tex is 2 in number 1 and number 2
                        conditions. In number 3 condition, --replicate_tex is
                        3. For assigning the same --replicate_tex to all
                        conditions, just use like all_1 (--replicate_tex is 1
                        in all conditions).
  --replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...], -rf REPLICATE_FRAG [REPLICATE_
↪FRAG ...]
                        Similar to --replicates_tex. This value is for
                        fragmented (or conventional) libraries.
```

- **Additional arguments**

```
additional arguments:
  --transterm_path TRANSTERM_PATH
                        Path of "transterm" in TransTermHP.
  --expterm_path EXPTERM_PATH
                        Path of expterm.dat for TransTermHP. Default is
                        /usr/local/bin/expterm.dat
  --rnafold_path RNAFOLD_PATH
```

(continues on next page)

(continued from previous page)

```
                        Path of RNAfold of Vienna package.
  --srna_files SRNA_FILES [SRNA_FILES ...], -sr SRNA_FILES [SRNA_FILES ...]
                        Paths of sRNA gff files if sRNA information need to be
                        considered as well.
  --decrease DECREASE, -d DECREASE
                        The maximum ratio -- (lowest coverage / highest
                        coverage) within (or nearby) the terminator. If the
                        ratio is smaller than --decrease, the candidate will
                        be considered as highly-confidence terminator. Default
                        is 0.5.
  --tolerance_detect_coverage TOLERANCE_DETECT_COVERAGE, -tc TOLERANCE_DETECT_COVERAGE
                        The extended region (nucleotides) of the terminators
                        for detecting coverage significant drop. For example,
                        the location of terminator is 300-400, and
                        --tolerance_detect_coverage is 30. If the coverage
                        decrease is detected within 270-430, this candidate is
                        still considered as the terminator which have coverage
                        dramatic decrease. Default is 30.
  --tolerance_within_transcript TOLERANCE_WITHIN_TRANSCRIPT, -tut TOLERANCE_WITHIN_
↪TRANSCRIPT
                        If the candidates are within transcript and the
                        distance (nucleotides) between the end of transcript
                        and terminator is within this value, the candidate
                        will be considered as a terminator. Otherwise, it will
                        be removed. Default is 30.
  --tolerance_downstream_transcript TOLERANCE_DOWNSTREAM_TRANSCRIPT, -tdt TOLERANCE_
↪DOWNSTREAM_TRANSCRIPT
                        The meaning is similar to
                        --tolerance_within_transcript. This value is for the
                        candidates which are at the downstream of transcript.
                        Default is 30.
  --tolerance_within_gene TOLERANCE_WITHIN_GENE, -twg TOLERANCE_WITHIN_GENE
                        The meaning is similar to
                        --tolerance_within_transcript. This value is for gene
                        in stead of transcript. Default is 10.
  --tolerance_downstream_gene TOLERANCE_DOWNSTREAM_GENE, -tdg TOLERANCE_DOWNSTREAM_
↪GENE
                        The meaning is similar to
                        --tolerance_downstream_transcript. This value is for
                        gene in stead of transcript. Default is 310.
  --highest_coverage HIGHEST_COVERAGE, -hc HIGHEST_COVERAGE
                        The minimum value of the highest coverage of
                        terminator. The low expressed terminator are not
                        included in "best_candidates", but are still in
                        "all_candidates". Default is 10.
  --window_size WINDOW_SIZE, -wz WINDOW_SIZE
                        Window size for searching secondary structure in
                        intergenic region. Default is 100 nts.
  --window_shift WINDOW_SHIFT, -ws WINDOW_SHIFT
                        The number of nucleotides for window shift. Default is
                        20 nts.
  --min_loop_length MIN_LOOP_LENGTH, -ml MIN_LOOP_LENGTH
                        The minimum loop length of terminator. Default is 3
                        nts.
  --max_loop_length MAX_LOOP_LENGTH, -Ml MAX_LOOP_LENGTH
                        The maximum loop length of terminator. Default is 10
                        nts.
```

(continues on next page)

(continued from previous page)

```
  --min_stem_length MIN_STEM_LENGTH, -ms MIN_STEM_LENGTH
                        The minimum stem length of terminator. Default is 4
                        nts.
  --max_stem_length MAX_STEM_LENGTH, -Ms MAX_STEM_LENGTH
                        The maximum stem length of terminator. Default is 20
                        nts.
  --miss_rate MISS_RATE, -mr MISS_RATE
                        The percentage of nucleotides which can be no pair in
                        the stem. Default is 0.25.
  --min_u_tail MIN_U_TAIL, -mu MIN_U_TAIL
                        The minimum number of U in poly U-tail of terminator.
                        Default is 5.
  --mutation_u_tail MUTATION_U_TAIL, -uu MUTATION_U_TAIL
                        The number of nts which are not U can be tolerated.
                        Default is 2.
  --keep_multi_term, -kp
                        Sometimes, one gene is associated with multiple
                        terminators In default, it will only keep the highly-
                        confidence one. This flag can keep all terminators
                        which are associated with the same gene. Default is
                        False.
optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/terminators`.

The output folders are as following:

statistics: Stores statistic files.

> **Terminator detection method with corresponding amounts:** Filename is `stat_$GENOME.csv`.
>
> **Comparing terminators with transcripts:** Based on different types of terminators, the files are `stat_compare_terminator_transcript_$GENOME_all_candidates.csv`, `stat_comparison_terminator_transcript_$GENOME_best.csv` and `stat_comparison_terminator_transcript_$GENOME_express.csv`

transtermhp_results: Store any output of TranstermHP.

gffs: Store gff files of terminators.

There are four different sub-folders for storing different gff files.

> **all_candidates:** Stores all terminators which `ANNOgesic` can detect.
>
> **expressed_candidates:** Stores the terminators revealing gene expression.
>
> **best_candidates:** Stores the terminators which reveal gene expression and show dramatic decrease of its coverage.
>
> **non_expressed_candidates:** Stores the terminators which has no gene expression.

Some useful information can be found in the tags of the attributes within the terminator gff file. Based on this information, we can know the details of the specific terminator. The tags are as following:

> **method:** By which method the terminator is detected.
>
> **coverage_decrease:** The terminators coverage reveals dramatic decrease or not.
>
> **express:** The terminator reveals gene expression or not.

diff_coverage: This value shows the library which reveals strongest coverage decreasing.

associated_gene: Which genes are associated with this terminator.

Parent: This tag presents the parent transcript of the terminator.

tables: Stores tables of terminators with more details.

There are four different sub-folders for storing different tables.

all_candidates: Stores all terminators which `ANNOgesic` can detect.

express_candidates: Stores the terminators revealing gene expression.

best_candidates: Stores the terminators which reveal gene expression and show dramatic decrease of its coverage.

non_expressed_candidates: Stores the terminators which has no gene expression.

The meanings of the columns are as following:

Genome: Genome name.

Name: Name of this terminator in the gff file.

Start: Staring point of this terminator.

End: End point of this terminator.

Strand: Strand of this terminator.

Detect: This terminator is detected by which method.

Associated_gene: Which genes are associated with this terminator.

Associated_transcript: The parent transcript of this terminator.

Coverage_decrease: This terminator shows dramatic decrease of its coverage or not.

Coverage_$LIB_NAME: Shows the coverage information of the libraries about this terminator. `high` means the highest coverage of the libraries, `low` means the lowest coverage of the libraries, and `diff` represents the difference between `high` and `low`. If `No_coverage_decreasing` is showed, it means this terminator reveal gene expression but no coverage decrease. If `NA` is showed, it means that this terminator has no gene expression.

1.4.12 utr (UTR detection)

`utr` can compare TSSs, CDSs/tRNAs/sRNAs, transcripts and terminators to generate 5'UTR and 3'UTR. 5'UTRs are based on detecting the regions between TSSs and CDSs/tRNAs/sRNAs. 3'UTRs are based on detecting the regions between the end of the transcripts and CDSs/tRNAs/sRNAs. If the input gff files of TSSs are not computed by ANNOgesic, please use `--tss_source` to classify TSSs for the analysis.

- **Required files**

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

Gff files of the TSSs: Please check the section *tss_ps (TSS and processing site prediction)*.

Gff files of the transcripts: Please check the section *transcript (transcript detection)*.

- **Optional input files**

Gff files of the terminators: If the information of terminators is needed, the gff files of terminators are required. Please check the section *terminator (terminator detection)*.

- **Basic Arguments**

```
usage: annogesic utr --project_path PROJECT_PATH --annotation_files
                     ANNOTATION_FILES [ANNOTATION_FILES ...] --tss_files
                     TSS_FILES [TSS_FILES ...] --transcript_files
                     TRANSCRIPT_FILES [TRANSCRIPT_FILES ...]
                     [--terminator_files TERMINATOR_FILES [TERMINATOR_FILES ...]]
                     [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs, etc. Gff files of TSSs,
                        terminators, and transcripts need to be separately
                        assigned to --tss_files, terminator_files, and
                        transcript_files, respectively.
  --tss_files TSS_FILES [TSS_FILES ...], -t TSS_FILES [TSS_FILES ...]
                        Paths of the TSS files.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
                        Paths of the transcript gff files.
  --terminator_files TERMINATOR_FILES [TERMINATOR_FILES ...], -e TERMINATOR_FILES␣
↪[TERMINATOR_FILES ...]
                        If the paths of terminator files are assigned,
                        terminator will be used to detect 3'UTR.
```

- **Additional arguments**

```
additional arguments:
  --tss_source, -s      The TSS gff file is generated by ANNOgesic or not.
                        Default is True (from ANNOgesic).
  --base_5utr {both,transcript,TSS}, -b5 {both,transcript,TSS}
                        The information for detection of 5'UTR. It can be
                        "TSS" or "transcript" or "both". Default is both.
  --utr_length UTR_LENGTH, -l UTR_LENGTH
                        The maximum UTR length. Default is 300.
  --base_3utr {both,transcript,terminator}, -b3 {both,transcript,terminator}
                        please assign the information for detection of 3'UTR.
                        It can be "transcript" or "terminator" or "both".
                        Default is transcript.
  --terminator_tolerance TERMINATOR_TOLERANCE, -et TERMINATOR_TOLERANCE
                        The 3'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated terminators. Default is 30.
  --tolerance_3utr TOLERANCE_3UTR, -t3 TOLERANCE_3UTR
                        The length of 3'UTR can be extended or withdrew by
                        this value (nucleotides). It only works when
                        transcript information is provided. Default is 10.
  --tolerance_5utr TOLERANCE_5UTR, -t5 TOLERANCE_5UTR
                        The length of 5'UTR can be extended or withdrew by
                        this value (nucleotides). It only works when
                        transcript information is provided. Default is 5.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files of 5'UTRs are stored in `$ANNOgesic/output/UTRs/5UTRs`.

Output files of 3'UTRs are stored in `$ANNOgesic/output/UTRs/3UTRs`.

The output folders are as following:

gffs: Stores gff files of the 5'UTR/3'UTR. Some useful information can be found in the tags of the attributes within the UTR gff file. Based on this information, we can know the details of the specific UTR. The tags are as following:

length: UTR length.

associated_cds: Which CDSs/rRNAs/tRNAs are associated with this UTR.

associated_gene: Which genes are associated with this UTR.

Parent: Shows the parent transcript of this UTR.

associated_tss: Which TSSs are associated with this 5'UTR.

tss_type: What types of TSSs are associated with this 5'UTR.

associated_term: Which terminators are associated with this 3'UTR.

statiatics: `$GFFNAME_$GENOME_$UTRTYPE_length.png` is the distribution of the UTR length.

1.4.13 srna (sRNA detection)

`srna` can predict different types of sRNAs. For intergenic and antisense sRNA, it is detected via comparison of the transcripts and annotation profiles, as well as coverage files. For UTR-derived sRNA, the detection is based on the TSSs, processing sites, transcripts, genome annotations and coverage files. Further filters like folding free energy change, BLAST to nr database and sRNA database can be set as well.

- **Required files**

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

Gff files of the transcripts: Please check the section *transcript (transcript detection)*.

Wiggle files of the fragmented/conventional or TEX+/- libraries: Please check the section *The input format of libraries for running ANNOgesic*.

- **Optional input files**

Gff files of the TSSs: If you want to detect the UTR-derived sRNAs, it is necessary to input TSS information. If you don't want to detect UTR-derived sRNAs, TSS information still can be provided as a filter. We strongly recommend input this file. please check the section *tss_ps (TSS and processing site prediction)*.

Gff files of processing sites: For checking the sRNAs which end with processing sites. Moreover, Some 3'UTR-derived and interCDS-derived sRNA candidates start from processing sites not TSSs. If you don't want to detect UTR-derived sRNAs, This information still can be provided to increase the accuracy, especially for some long non-coding regions. We strongly recommend input this file if you want to detect UTR-derived sRNAs. Please check the section *tss_ps (TSS and processing site prediction)*.

Promoter tables: Information of the promoter motifs can be used for prioritizing sRNA candidates via promoters and sRNA coverage. The format should be as following:

Genome	TSS_position	TSS_strand	Motif
NC_000915.1	237118	-	MOTIF_1
NC_000915.1	729009	-	MOTIF_1

First irow is header of the table, the last column is the name of promoter motif. If subcommand `promoter` was implemented before, the table will be generated automatically. Please refer to the section *promoter (Promoter motif detection)*.

- **Filters with the corresponding input files and tools**

There are some filters which can improve the prediction. The user can assign the information to remove false positive. If the information is not assigned to be a filter, it still can input to the module. Then, the information will be shown in the output files, but this information is not considered as a filter. For an example, if terminator association is not assigned to be a filter, the user still can specify the path of terminator gff files. The associated terminators will be shown in output gff files and tables, but the sRNA candidates which are not associated with terminators will still be included. Following is the filter names with the required files and tools.

Secondary structure: Remove the false positive by checking the folding energy change of secondary structure.

Required tools:

ViennaRNA

Required files:

Fasta files of genome sequences

TSS: Remove the candidates which are not associated with TSSs.

Required files:

Gff files of TSSs

Searching sRNA candidate in sRNA database: If homology of this sRNA candidate can be found in sRNA database, this candidate will be included to the result without considering other filters. `--blast_score_srna` and `--blast_e_srna` can be used for adjustment of the prediction.

Required tools:

Blast+

Required files:

sRNA database: Such as BSRD. Format of the header should be `$ID|$GENOME|$SRNANAME`. For an example, `srn_4840|S._aureus_NCTC8325|RsaOV` The ID is srn_4840, the strain of this sRNA is S._aureus_NCTC8325 and the name of sRNA is RsaOV. If the format of the header is not correct, an error or non-sense results will occur. If you want to use BSRD with proper headers, you can download it from our Git repository easily.

Searching sRNA candidate in nr database: If homologs of this sRNA candidates can be found in nr database and the hit numbers are more than `--cutoff_nr_hit`, this candidates will be removed. `--blast_score_nr` and `--blast_e_nr` can be used for adjustment of the prediction.

Required tools:

Blast+

Required files:

nr database: The file can be download from nr database.

Terminator: Remove the candidates which are not associated with terminators.

Required files:

Gff files of the terminators: Please check the section *terminator (terminator detection)*.

sORF: Remove the candidates which overlap sORF.

Required files:

Gff files of the sORFs: Please check the section *sorf (sORF detection)*.

Promoter: Remove the candidates which are not associated with promoter motif.

Required files:

Tables of the promoters: Please check the Promoter Tables of this section and the section *promoter (Promoter motif detection)*.

- **Basic arguments**

```
usage: annogesic srna --project_path PROJECT_PATH [--utr_derived_srna]
                      [--filter_info {tss,sec_str,blast_nr,blast_srna,sorf,term,
↪promoter,none} [{tss,sec_str,blast_nr,blast_srna,sorf,term,promoter,none} ...]]
                      --transcript_files TRANSCRIPT_FILES
                      [TRANSCRIPT_FILES ...] --annotation_files
                      ANNOTATION_FILES [ANNOTATION_FILES ...]
                      [--tss_files TSS_FILES [TSS_FILES ...]]
                      [--fasta_files FASTA_FILES [FASTA_FILES ...]]
                      [--compute_sec_structures]
                      [--srna_database_path SRNA_DATABASE_PATH]
                      [--nr_database_path NR_DATABASE_PATH]
                      [--tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...]]
                      [--frag_libs FRAG_LIBS [FRAG_LIBS ...]]
                      [--tex_notex {1,2}]
                      [--replicate_tex REPLICATE_TEX [REPLICATE_TEX ...]]
                      [--replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...]]
                      [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --utr_derived_srna, -u
                        Assign to detect UTR-derived sRNA. Default is False.
  --filter_info {tss,sec_str,blast_nr,blast_srna,sorf,term,promoter,none} [{tss,sec_
↪str,blast_nr,blast_srna,sorf,term,promoter,none} ...], -d {tss,sec_str,blast_nr,
↪blast_srna,sorf,term,promoter,none} [{tss,sec_str,blast_nr,blast_srna,sorf,term,
↪promoter,none} ...]
                        The filters for improving the sRNA detection: 1. tss
                        (sRNA has to start with a TSS), 2. sec_str (free
                        energy change of secondary structure (normalized by
                        length) has to be smaller than --cutoff_energy), 3.
                        blast_nr (the number of the homologs in the non-
                        redundant database has to be below the --cutoff_nr_hit
                        ), 4. blast_srna (as long as the homologs can be found
                        in the sRNA database, the candidates will be included
                        to the best candidtes without considering other
                        filters), 5. sorf (sRNA must not overlap with sORFs),
                        6. term (sRNA has to be associated with a terminator),
                        7. promoter (sRNA has to be associated with a promoter
                        motif). For using multiple filters, please separated
                        them by spaces. If blast_srna was assigned, the
                        headers of sequences in sRNA database should be
                        $ID|$GENOME|$SRNANAME. "tss sec_str blast_nr
                        blast_srna" are recommended to be used. If "none" is
                        assigned, no filters are applied. Default is tss
                        sec_str blast_nr blast_srna.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
```

(continues on next page)

(continued from previous page)

```
                        Paths of the transcript files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs, etc. Gff files of transcripts,
                        TSSs, terminators, processing sites, and sORFs need to
                        be separately assigned to --transcript_files,
                        --tss_files, --terminator_files,
                        --processing_site_files, and --sorf_files,
                        respectively.
  --tss_files TSS_FILES [TSS_FILES ...], -t TSS_FILES [TSS_FILES ...]
                        Paths of TSS gff files. For detecting UTR-derived sRNA
                        or "tss" in --filter_info, TSS gff files MUST be
                        provided.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        paths of fasta files of reference genome.
  --compute_sec_structures, -cs
                        Computing secondary structures of sRNAs. Default is
                        False.
  --srna_database_path SRNA_DATABASE_PATH, -sd SRNA_DATABASE_PATH
                        Path of sRNA database with proper headers of
                        sequences. Format of the header should be
                        $ID|$GENOME|$NAME. Please check
                        https://github.com/Sung-Huan/ANNOgesic/blob/master/dat
                        abase/sRNA_database_BSRD.fa
  --nr_database_path NR_DATABASE_PATH, -nd NR_DATABASE_PATH
                        Path of nr database
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
                        TEX+/- wig files. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_tex_forward.wig:tex:1:a:+
                        my_lib_tex_reverse.wig:tex:1:a:-.
  --frag_libs FRAG_LIBS [FRAG_LIBS ...], -fl FRAG_LIBS [FRAG_LIBS ...]
                        Wig files of RNA-Seq with fragmented transcripts. The
                        format is: wig_file_path:frag:condition_id(integer):re
                        plicate_id(alphabet):strand(+ or -). If multiple wig
                        files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_frag_forward.wig:frag:1:a:+
                        my_lib_frag_reverse.wig:frag:1:a:-.
  --tex_notex {1,2}, -te {1,2}
                        If TEX+/- libraries are assigned, a sRNA should be
                        detected in both (TEX+ and TEX-, assigned by 2) or
                        needs to be detected in only one library (TEX+ or
                        TEX-, assigned by 1). Default is 2.
  --replicate_tex REPLICATE_TEX [REPLICATE_TEX ...], -rt REPLICATE_TEX [REPLICATE_TEX␣
↪...]
                        This value is the minimal number of replicates that a
                        sRNA has to be detected in. The format is
                        $NUMBERofCONDITION_$NUMBERofREPLICATE. If different
                        --replicate_tex values need to be assigned to
                        different conditions, please use spaces to separate
                        them. For example, 1_2 2_2 3_3 means that
```

(continues on next page)

(continued from previous page)

```
                        --replicate_tex is 2 in number 1 and number 2
                        conditions. In number 3 condition, --replicate_tex is
                        3. For assigning the same --replicate_tex to all
                        conditions, use all_1 (--replicate_tex is 1 in all
                        conditions).
  --replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...], -rf REPLICATE_FRAG [REPLICATE_
↪FRAG ...]
                        Similar to --replicates_tex. This value is for
                        libraries with fragmented transcripts.
```

- **Additional argument**

```
additional arguments:
  --rnafold_path RNAFOLD_PATH
                        Path of RNAfold of the Vienna package
  --relplot_path RELPLOT_PATH
                        Path of relplot.pl of the Vienna package.
  --mountain_path MOUNTAIN_PATH
                        Path of mountain.pl of the Vienna package.
  --blastn_path BLASTN_PATH
                        Path of blastn of the BLAST+ package.
  --blastx_path BLASTX_PATH
                        Path of blastx of the BLAST+ package.
  --makeblastdb_path MAKEBLASTDB_PATH
                        Path of makeblastdb of the BLAST+ package.
  --processing_site_files PROCESSING_SITE_FILES [PROCESSING_SITE_FILES ...], -p␣
↪PROCESSING_SITE_FILES [PROCESSING_SITE_FILES ...]
                        Paths of the processing site gff files. It can improve
                        the detection of UTR-derived sRNAs.
  --terminator_files TERMINATOR_FILES [TERMINATOR_FILES ...], -e TERMINATOR_FILES␣
↪[TERMINATOR_FILES ...]
                        Paths of the terminator gff files.
  --promoter_tables PROMOTER_TABLES [PROMOTER_TABLES ...], -pt PROMOTER_TABLES␣
↪[PROMOTER_TABLES ...]
                        If promoter tables can be provided, please assign the
                        paths of promoter tables, The format of the table is
                        $GENOME $TSS_POSITION $TSS_STRAND $PROMOTER_NAME.
  --promoter_names PROMOTER_NAMES [PROMOTER_NAMES ...], -pn PROMOTER_NAMES [PROMOTER_
↪NAMES ...]
                        If --promoter_tables is provided, please assign the
                        promoter name (the last column of promoter table). For
                        multiple promoters, please put spaces between the
                        promoters. Default is None.
  --sorf_files SORF_FILES [SORF_FILES ...], -O SORF_FILES [SORF_FILES ...]
                        If comparison between sRNAs and sORFs needs to be
                        done, Please assign the paths of sORF gff files
  --parallel_blast PARALLEL_BLAST, -pb PARALLEL_BLAST
                        The number of parallel jobs. Default is 10.
  --tss_source, -ts     The TSS gff files are generated from ANNOgesic or not.
                        Default is True (from ANNOgesic).
  --tss_intergenic_antisense_tolerance TSS_INTERGENIC_ANTISENSE_TOLERANCE, -tit TSS_
↪INTERGENIC_ANTISENSE_TOLERANCE
                        The 5'ends of intergenic and antisense sRNA candidates
                        will be extended or withdrew by this value
                        (nucleotides) for searching the associated TSSs.
                        Default is 3.
  --tss_5utr_tolerance TSS_5UTR_TOLERANCE, -t5 TSS_5UTR_TOLERANCE
```

(continues on next page)

(continued from previous page)

```
                        The 5'ends of 5'UTR-derived sRNAs will be extended or
                        withdrew by this value (nucleotides) for searching the
                        associated TSSs. The input type can be percentage
                        ("p") or the real amount of reads ("n"). For example,
                        p_0.05 means this value is 5 percent of the length of
                        5'UTR. n_10 means this value is 10 nts. Default is
                        n_3.
  --tss_3utr_tolerance TSS_3UTR_TOLERANCE, -t3 TSS_3UTR_TOLERANCE
                        Similar to --tss_5utr_tolerance. This value is for
                        3'UTR-derived sRNAs. Default is p_0.04.
  --tss_intercds_tolerance TSS_INTERCDS_TOLERANCE, -tc TSS_INTERCDS_TOLERANCE
                        Similar to --tss_5utr_tolerance. This value is for
                        interCDS-derived sRNAs. Default is p_0.04.
  --terminator_tolerance_in_srna TERMINATOR_TOLERANCE_IN_SRNA, -eti TERMINATOR_
↪TOLERANCE_IN_SRNA
                        The 3'ends of sRNA candidates will be withdrew by this
                        value (nucleotides) for searching the associated
                        terminators which are within sRNAs. Default is 30.
  --terminator_tolerance_out_srna TERMINATOR_TOLERANCE_OUT_SRNA, -eto TERMINATOR_
↪TOLERANCE_OUT_SRNA
                        The 3'ends of sRNA candidates will be extended by this
                        value (nucleotides) for searching the associated
                        terminators which are behind of sRNAs. Default is 30.
  --min_length MIN_LENGTH, -lm MIN_LENGTH
                        The minimum sRNA length. Default is 30.
  --max_length MAX_LENGTH, -lM MAX_LENGTH
                        The maximum sRNA length. Default is 500.
  --min_intergenic_tex_coverage MIN_INTERGENIC_TEX_COVERAGE, -it MIN_INTERGENIC_TEX_
↪COVERAGE
                        The minimum average coverage of intergenic sRNAs for
                        TEX+ libraries. This value is based on different types
                        of TSSs. The order of numbers is
                        "Primary,Secondary,Internal,Antisense,Orphan". For
                        example, 0,0,0,50,10 means that antisense TSS (minimum
                        coverage is 50) and orphan TSS (minimum coverage is
                        10) are used for sRNA prediction. The other types of
                        TSSs will not be used for the detection (assign to 0).
                        If TSS information is not provided, the lowest value
                        would be the general cutoff for the prediction.
                        Default is 0,0,0,40,20.
  --min_intergenic_notex_coverage MIN_INTERGENIC_NOTEX_COVERAGE, -in MIN_INTERGENIC_
↪NOTEX_COVERAGE
                        Similar to --min_intergenic_tex_coverage. This value
                        is for TEX- libraries. Default is 0,0,0,30,10.
  --min_intergenic_fragmented_coverage MIN_INTERGENIC_FRAGMENTED_COVERAGE, -if MIN_
↪INTERGENIC_FRAGMENTED_COVERAGE
                        Similar to --min_intergenic_tex_coverage. This value
                        is for fragmented (or conventional) libraries. Default
                        is 400,200,0,50,20.
  --min_complete_5utr_transcript_coverage MIN_COMPLETE_5UTR_TRANSCRIPT_COVERAGE, -ib␣
↪MIN_COMPLETE_5UTR_TRANSCRIPT_COVERAGE
                        Several primary/secondary TSSs are also associated
                        with a complete transcript containing no
                        CDSs/tRNA/rRNA in 5'UTR of the following CDS which is
                        located in another transcript. In order to detect the
                        sRNA candidates in these transcripts, please assign
                        the minimum average coverage of the sRNA candidates.
```

(continues on next page)

(continued from previous page)

```
                        The format is $TEX,$NOTEX,$FRAG. For example,
                        200,100,100 means that the minimum average coverage is
                        200 for TEX+ libraries, 100 for TEX- and fragmented
                        (or conventional) libraries. Default is 30,20,30.
  --min_antisense_tex_coverage MIN_ANTISENSE_TEX_COVERAGE, -at MIN_ANTISENSE_TEX_
↪COVERAGE
                        Similar to --min_intergenic_tex_coverage. This value
                        is for antisense in stead of intergenic. Default is
                        0,0,0,40,20.
  --min_antisense_notex_coverage MIN_ANTISENSE_NOTEX_COVERAGE, -an MIN_ANTISENSE_
↪NOTEX_COVERAGE
                        Similar to --min_intergenic_notex_coverage. This value
                        is for antisense in stead of intergenic. Default is
                        0,0,0,30,10.
  --min_antisense_fragmented_coverage MIN_ANTISENSE_FRAGMENTED_COVERAGE, -af MIN_
↪ANTISENSE_FRAGMENTED_COVERAGE
                        Similar to --min_intergenic_fragmented_coverage. This
                        value is for antisense in stead of intergenic. Default
                        is 400,200,0,50,20.
  --min_utr_tex_coverage MIN_UTR_TEX_COVERAGE, -ut MIN_UTR_TEX_COVERAGE
                        The minimum average coverage of UTR-derived sRNA
                        candidates in TEX+ libraries. The input can be
                        assigned by the percentile ("p") or real number of
                        coverage ("n"). The order of numbers is
                        "5'UTR,3'UTR,interCDS". For example,
                        "p_0.7,p_0.5,p_0.5" means that 70 percentile of all
                        5'UTR coverages is used for the minimum coverage of
                        5'UTR-derived sRNA, median of all 3'UTR and interCDS
                        coverages is used for minimum coverage of 3'UTR and
                        interCDS-derived sRNA. Default is p_0.8,p_0.6,p_0.7.
  --min_utr_notex_coverage MIN_UTR_NOTEX_COVERAGE, -un MIN_UTR_NOTEX_COVERAGE
                        Similar to --min_utr_tex_coverage. This value is for
                        TEX- libraries. Default is p_0.7,p_0.5,p_0.6.
  --min_utr_fragmented_coverage MIN_UTR_FRAGMENTED_COVERAGE, -uf MIN_UTR_FRAGMENTED_
↪COVERAGE
                        Similar to --min_utr_tex_coverage. This value is for
                        fragmented (or conventional) libraries. Default is
                        p_0.7,p_0.5,p_0.6.
  --min_all_utr_coverage MIN_ALL_UTR_COVERAGE, -mu MIN_ALL_UTR_COVERAGE
                        The minimum coverage of UTR-derived sRNAs. The
                        coverage of UTR-derived sRNAs should not only exceed
                        the --min_utr_TEX_coverage, --min_utr_noTEX_coverage
                        and --min_utr_fragmented_coverage, but also this
                        value. Default is 50.
  --cutoff_energy CUTOFF_ENERGY, -ce CUTOFF_ENERGY
                        If "sec_str" is included in --filter_info, please
                        assign the maximum folding energy change (normalized
                        by length of gene). Default is -0.05.
  --mountain_plot, -m   Generating mountain plot of sRNA candidate. Default is
                        False.
  --nr_format, -nf      Format nr database. Default is False.
  --srna_format, -sf    Format sRNA database. Default is False.
  --decrease_intergenic_antisense DECREASE_INTERGENIC_ANTISENSE, -di DECREASE_
↪INTERGENIC_ANTISENSE
                        This value is for detecting the coverage decrease in
                        intergenic/antisense transcript. If the length of
                        intergenic transcript is longer than the --max_length,
```

(continues on next page)

(continued from previous page)

```
                        it will searching the coverages of the transcript. If
                        the ratio -- (the lowest coverage / the highest
                        coverage) of the transcript is smaller than this value
                        and the length is within a given range, the transcript
                        will be considered as a sRNA as well.Default is 0.1.
  --decrease_utr DECREASE_UTR, -du DECREASE_UTR
                        Similar to --decrease_intergenic_antisense. This value
                        is for UTR-derived sRNA. Default is 0.05.
  --tolerance_intergenic_antisense TOLERANCE_INTERGENIC_ANTISENSE, -ti TOLERANCE_
↪INTERGENIC_ANTISENSE
                        The 5'ends and 3'ends of intergenic and antisense
                        sRNAs will be extended by this value (nucleotides) for
                        detecting the significant coverage decrease. (please
                        check --decrease_intergenic_antisense). For example,
                        the location of intergenic sRNA is 300-400, and
                        --tolerance_intergenic_antisense is 30. The searching
                        region is 270-430. Default is 10.
  --tolerance_utr TOLERANCE_UTR, -tu TOLERANCE_UTR
                        Similar to --tolerance_intergenic_antisense. This is
                        for UTR-derived sRNAs. Default is 10.
  --cutoff_nr_hit CUTOFF_NR_HIT, -cn CUTOFF_NR_HIT
                        The maximum hits number in nr database. Default is 0.
  --blast_e_nr BLAST_E_NR, -en BLAST_E_NR
                        The maximum e-value for searching in nr database.
                        Default is 0.0001.
  --blast_e_srna BLAST_E_SRNA, -es BLAST_E_SRNA
                        The maximum e-value for searching in sRNA database.
                        Default is 0.0001.
  --blast_score_srna BLAST_SCORE_SRNA, -bs BLAST_SCORE_SRNA
                        The minimum score for searching in sRNA database.
                        Default is 40.
  --blast_score_nr BLAST_SCORE_NR, -bn BLAST_SCORE_NR
                        The minimum score for searching in nr database.
                        Default is 40.
  --detect_srna_in_cds, -ds
                        Searching sRNA in CDS (e.g. the genome annotation is
                        not correct). More sRNA candidates which overlap with
                        CDS will be detected. Beware, this argument may cause
                        many false positives due to without considering the
                        locations of genes and CDSs. Moreover, the rank of
                        sRNA candidates will be influenced as well.Default is
                        False.
  --overlap_percent_cds OVERLAP_PERCENT_CDS, -oc OVERLAP_PERCENT_CDS
                        The maximum ratio of overlapping between CDS and sRNA
                        candidates. It only works if --detect_srna_in_cds is
                        true. Default is 0.5
  --search_poly_u SEARCH_POLY_U, -sp SEARCH_POLY_U
                        The tolerance length for searching poly U tail of
                        sRNA. If this value is assigned by 0, the 3'end of
                        sRNA will not be extended by searching poly U tail.
                        Default is 15.
  --min_u_poly_u MIN_U_POLY_U, -np MIN_U_POLY_U
                        The minimum number of U that poly U tail should
                        contain. Default is 5.
  --mutation_poly_u MUTATION_POLY_U, -mp MUTATION_POLY_U
                        The minimum number of nts which are not U can be
                        tolerated. Default is 2.
```

(continues on next page)

(continued from previous page)

```
  --ignore_hypothetical_protein, -ih
                        For ignoring hypothetical proteins in the genome
                        annotation file. Default is False.
  --ranking_time_promoter RANKING_TIME_PROMOTER, -rp RANKING_TIME_PROMOTER
                        If --promoter_tables is provided, the information of
                        promoter can be use for ranking sRNA candidates. The
                        ranking score is --ranking_time_promoter * average
                        coverage. For example, a sRNA candidate which is
                        associated with a promoter and its average coverage is
                        10. If --ranking_time_promoter is 2, the ranking score
                        will be 20 (2*10). For the candidate which are not
                        associated with a promoter, the
                        --ranking_time_promoter will be 1. Therefore,
                        --ranking_time_promoter can not be smaller than 1.
                        Default is 2.
  --exclude_srna_in_annotation_file, -ea
                        For excluding the sRNAs which are already annotated in
                        --annotation_files. Default is False.
optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/sRNAs`. the output folders and files are following:

sRNA_2d_$GENOME: The secondary structures of all sRNA candidates.

sRNA_seq_$GENOME: The sequences of all sRNA candidates.

blast_results_and_misc: Stores the results of blast.

nr_blast_$GENOME.txt: output of BLAST for the nr database.

sRNA_blast_$GENOME.txt: output of BLAST for the sRNA database.

figs: Stores the figures about secondary structures of sRNAs.

mountain_plots: Stores mountain plots of the sRNA candidates. Filename is as `srna10_NC_009839.1_335339_335435_+_mountain.pdf`. `srna10`, `NC_009839.1`, `335339`, `335435`, `+` are ID of sRNA gff file, genome name, starting point, end point and strand, respectively.

sec_plots: Stores the secondary structure plots of sRNA candidates. Filename of is as `srna10_NC_009839.1_335339_335435_+_rss.ps`. `srna10`, `NC_009839.1`, `335339`, `335435`, `+` are ID of sRNA gff file, genome name, starting point, end point and strand, respectively.

dot_plots: Stores the dot plots of sRNA candidates. Filename of dot plot is as `srna10_NC_009839.1_335339_335435_+_dp.ps`. `srna10`, `NC_009839.1`, `335339`, `335435`, `+` are ID of sRNA gff file, genome name, starting point, end point and strand, respectively.

statistics: Stores statistics files. `stat_$GENOME_sRNA_blast.csv` is the analysis result of BLAST for sRNA databases. `stat_sRNA_class_Staphylococcus_aureus_HG003.csv` is the classification of sRNA candidates.

TSS_classes: If the TSSs are not computed by ANNOgesic, `TSS_classes` will be generated for classification of TSS. TSS gff files with TSS types will be stored here.

tables: Stores sRNA tables with more details. There are also some sub-folders:

for_classes: Stores the results based on different sRNA classes. The information of sRNA classes can be found in `stat_sRNA_class_$GENOME.csv`.

best_candidates: Stores the best results of sRNAs after filtering.

all_candidates: Stores all candidates without filtering.

The meanings of the columns are as following:

Rank: Ranking number of this sRNA.

Genome: Genome name.

Name: sRNA Name which is shown in gff file. The sRNA name is generated from the BLAST search.

Start: Starting point of this sRNA.

End: End point of this sRNA.

Strand: Strand of this sRNA.

Start_with_TSS/Cleavage_site: This sRNA starts with which TSS or cleavage site.

End_with_cleavage: If the sRNA ends with a cleavage site, the information of this cleavage site will be showed here.

Candidates: Position of this sRNA.

Lib_type: This sRNA is detected by TEX+/- or fragmented/conventional library.

Best_avg_coverage: Based on coverage of all libraries, The best average coverage of this sRNA will be showed here.

Normalized_secondary_energy_change(by_length): Secondary folding energy change (normalized by length) of this sRNA.

sRNA_types: Shows the sRNA type.

Conflict_sORF: If this sRNA overlaps sORF, the overlapped sORF will be showed here.

nr_hit_number: The hit numbers of this sRNA in nr database.

sRNA_hit_number: The hit numbers of this sRNA in sRNA database.

nr_hit_top3|ID|e-value|score: The top 3 hits of this sRNA in nr database will be showed here. The information includes protein name, ID, e-value, and score.

sRNA_hit|e-value|score: If the homology of this sRNA can be found in sRNA database, the information will be showed here. The information includes sRNA name, e-value, and score.

Overlap_CDS_forward: If the sRNA overlaps genomic features, the information of the overlapped features will be showed here (for forward strand).

Overlap_nts_forward: If the sRNA overlaps genomic features, the length and percentage of the overlapping regions will be showed here (for forward strand).

Overlap_CDS_reverse: If the sRNA overlaps genomic features, the information of the overlapped features will be showed here (for reverse strand).

Overlap_nts_reverse: If the sRNA overlaps genomic features, the length and percentage of the overlapping regions will be showed here (for reverse strand).

End_with_terminator: The terminator which is associated with this sRNA.

Associated_promoter: The promoter which is associated with this sRNA.

sRNA_length: sRNA length.

Avg_coverage:$LIB_NAME: Shows the average coverage information of the libraries about this sRNA.

gffs: Stores gff files of the sRNA. There are also some sub-folders:

for_classes: Stores the results based on different sRNA classes.

best_candidates: Stores the best results of sRNAs after filtering.

all_candidates: Stores all candidates without filtering.

Some useful information can be found in the tags of the attributes within the sRNA gff file. Based on this information, we can know the details of the specific sRNA. The tags are as following:

Name: The sRNA name is generated from the BLAST search.

sRNA_type: This sRNA is from 5'UTR, 3'UTR, interCDS, intergenic, antisense or within CDS.

with_TSS: Which TSSs are related to this sRNA.

sORF: Which sORFs overlap this sRNA.

sRNA_hit: Blast hit numbers of this sRNA in sRNA database.

nr_hit: Blast hit numbers of this sRNA in nr database.

2d_energy: Normalized (by the length of sRNA) folding energy change of the sRNA secondary structure.

with_term: Terminators which are associated with the sRNA candidate.

end_cleavage: If this sRNA ends with a cleavage site, information of the cleavage site will be showed here.

overlap_cds: This sRNA overlaps CDS or not.

overlap_percentage: If this sRNA overlap CDS. The percentage of the overlap between CDS and sRNA will be showed here.

promoter: Promoters which are associated with the sRNA.

If the amount of sRNA candidates are too many for the user, please check the FAQ Q9 to do the further filtering.

1.4.14 sorf (sORF detection)

`sorf` can detect sORF based on searching ribosome binding sites (Shine-Dalgarno sequence), start codons and stop codons within the non-coding expressed regions. Since these regions may be sRNAs or sORFs, Comparison between sORFs and sRNAs can be done by this subcommand as well. If multiple sORFs overlap with each other, this subcommand will merge them to be one sORF. Therefore, one region may contain more than one sORF. Position of the start codon which listed in output table is assigned by the first nucleotide. The position of stop codon is assigned by the last nucleotide. Moreover, one region may contain different frame shifts. Ex: (200, 202, 203) are the positions of three start codons and (241, 243) are two stop codons in a small transcript. There are three sORF candidates (200-241, 203-241 and 202-243).

- **Required files**

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc Gff files of the transcripts: Please check the section *transcript (transcript detection)*.

Wiggle files of TEX+/- or fragmented/conventional libraries: Please refer to the section *The input format of libraries for running ANNOgesic*.

fasta files of the genome sequences

- **Optional input files**

Gff files of the TSSs: For checking the sORFs start from TSS or not. We strongly recommend to input this file. Please check the section *tss_ps (TSS and processing site prediction)*.

Gff files of sRNAs: For checking the overlap of sRNAs and sORFs. Please check the section *srna (sRNA detection)*.

- **Basic arguments**

```
usage: annogesic sorf --project_path PROJECT_PATH [--utr_derived_sorf]
                      --fasta_files FASTA_FILES [FASTA_FILES ...]
                      --transcript_files TRANSCRIPT_FILES
                      [TRANSCRIPT_FILES ...] --annotation_files
                      ANNOTATION_FILES [ANNOTATION_FILES ...]
                      [--tss_files TSS_FILES [TSS_FILES ...]]
                      [--tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...]]
                      [--frag_libs FRAG_LIBS [FRAG_LIBS ...]]
                      [--tex_notex {1,2}]
                      [--replicate_tex REPLICATE_TEX [REPLICATE_TEX ...]]
                      [--replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...]]
                      [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --utr_derived_sorf, -u
                        Detect UTR-derived sORF. Default is False.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the fasta files of the reference genome.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
                        Paths of the transcript gff files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the the genome annotation gff files
                        containing CDSs, tRNAs, rRNAs, etc. Gff files of
                        transcripts, sRNAs, and TSSs need to be separately
                        assigned to --transcript_files, --srna_files, and
                        --tss_files, respectively.
  --tss_files TSS_FILES [TSS_FILES ...], -t TSS_FILES [TSS_FILES ...]
                        Paths of TSS gff files.
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
                        TEX+/- wig files. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_tex_forward.wig:tex:1:a:+
                        my_lib_tex_reverse.wig:tex:1:a:-.
  --frag_libs FRAG_LIBS [FRAG_LIBS ...], -fl FRAG_LIBS [FRAG_LIBS ...]
                        Wig files of RNA-Seq with fragmented transcripts. The
                        format is: wig_file_path:frag:condition_id(integer):re
                        plicate_id(alphabet):strand(+ or -). If multiple wig
                        files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_frag_forward.wig:frag:1:a:+
                        my_lib_frag_reverse.wig:frag:1:a:-.
  --tex_notex {1,2}, -te {1,2}
                        If the TEX+/- libraries are provided, this value is
                        that a sORF should be detected in both (TEX+ and TEX-,
                        assigned by 2) or can be detected in only one library
                        (TEX+ or TEX-, assigned by 1). Please assign 1 or 2.
                        Default is 2.
  --replicate_tex REPLICATE_TEX [REPLICATE_TEX ...], -rt REPLICATE_TEX [REPLICATE_TEX␣
↪...]
```

(continues on next page)

(continued from previous page)

```
                        This value is the minimal number of replicates that a
                        sORF has to be detected in. The format is
                        $NUMBERofCONDITION_$NUMBERofREPLICATE. If different
                        --replicate_tex values need to be assigned to
                        different conditions, please use spaces to separate
                        them. For example, 1_2 2_2 3_3 means that
                        --replicate_tex is 2 in number 1 and number 2
                        conditions. In number 3 condition, --replicate_tex is
                        3. For assigning the same --replicate_tex to all
                        conditions, just use like all_1 (--replicate_tex is 1
                        in all conditions).
  --replicate_frag REPLICATE_FRAG [REPLICATE_FRAG ...], -rf REPLICATE_FRAG [REPLICATE_
↪FRAG ...]
                        Similar to --replicates_tex. This value is for
                        fragmented (or conventional) libraries.
```

- **Additional arguments**

```
additional arguments:
  --srna_files SRNA_FILES [SRNA_FILES ...], -s SRNA_FILES [SRNA_FILES ...]
                        Paths of the sRNA gff files for comparing sORF and
                        sRNA to detect the overlapping.
  --contain_multi_stop, -ms
                        If --contain_multi_stop is True, the outpu files will
                        only contain the non-annotated transcripts (potential
                        sORFs) which have single stop codon (but may still
                        contain multiple sORFs from different reading frames).
                        If --contain_multi_stop is False, the output files
                        will assign the longest open reading frame which may
                        contain multiple stop codons to be sORFs. Default is
                        False.
  --utr_length UTR_LENGTH, -ul UTR_LENGTH
                        The utr length for comparing TSS with sORF. The
                        default number is 300.
  --min_length MIN_LENGTH, -ml MIN_LENGTH
                        The minimum nucleotide length of sORF. Default is 30.
  --max_length MAX_LENGTH, -Ml MAX_LENGTH
                        The maximum nucleotide length of sORF. Default is 300.
  --cutoff_intergenic_coverage CUTOFF_INTERGENIC_COVERAGE, -ci CUTOFF_INTERGENIC_
↪COVERAGE
                        The minimum coverage of intergenic sORF candidates.
                        Default is 10.
  --cutoff_antisense_coverage CUTOFF_ANTISENSE_COVERAGE, -ai CUTOFF_ANTISENSE_COVERAGE
                        The minimum coverage of antisense sORF candidates.
                        Default is 10.
  --cutoff_5utr_coverage CUTOFF_5UTR_COVERAGE, -cu5 CUTOFF_5UTR_COVERAGE
                        The minimum coverage for 5'UTR derived sORF
                        candidates. This value can be assigned by percentile
                        ("p") or the amount of reads ("n"). For example, p_0.5
                        means that the coverage of sORF candidates should be
                        higher than the 50 percentile of all 5'UTR
                        transcripts. n_10 means that the coverage of sORF
                        candidates should be higher than 10 reads. Default is
                        p_0.5.
  --cutoff_3utr_coverage CUTOFF_3UTR_COVERAGE, -cu3 CUTOFF_3UTR_COVERAGE
                        Similar to --cutoff_5utr_coverage. This value is for
                        3'UTRs. Default is p_0.5.
```

(continues on next page)

(continued from previous page)

```
  --cutoff_intercds_coverage CUTOFF_INTERCDS_COVERAGE, -cuf CUTOFF_INTERCDS_COVERAGE
                        Similar to --cutoff_5utr_coverage. This value is for
                        interCDS. Default is p_0.5.
  --cutoff_base_coverage CUTOFF_BASE_COVERAGE, -cub CUTOFF_BASE_COVERAGE
                        The general minimum coverage of all sORF candidates.
                        All candidates should exceed this value. Default is
                        10.
  --start_codon START_CODON [START_CODON ...], -ac START_CODON [START_CODON ...]
                        The types of start coden. If multiple types of start
                        codon need to be assigned, please use spaces to
                        separate them. Default is ATG.
  --stop_codon STOP_CODON [STOP_CODON ...], -oc STOP_CODON [STOP_CODON ...]
                        The types of stop codon. If multiple types of stop
                        codon need to be assigned, please use spaces to
                        separate them. Default is TTA TAG TGA.
  --min_rbs_distance MIN_RBS_DISTANCE, -mr MIN_RBS_DISTANCE
                        The minimum distance (nucleotides) between the
                        ribosome binding site (Shine-Dalgarno sequence) and
                        the start codon. Default is 3.
  --max_rbs_distance MAX_RBS_DISTANCE, -Mr MAX_RBS_DISTANCE
                        The maximum distance (nucleotides) between the
                        ribosome binding site (Shine-Dalgarno sequence) and
                        the start codon. Default is 15.
  --rbs_not_after_tss, -at
                        Include the sORFs which are not associated with
                        ribosome binding site to the high-confidence sORF
                        list. Default is False.
  --rbs_seq RBS_SEQ [RBS_SEQ ...], -rs RBS_SEQ [RBS_SEQ ...]
                        The sequence of ribosome binding site. If multiple
                        sequences need to be assigned, please use space to
                        split them. Default is AGGAGG.
  --tolerance_rbs TOLERANCE_RBS, -tr TOLERANCE_RBS
                        The number of nucleotides of ribosome binding sites
                        allowed to be different from AGGAGG. Default is 2.
  --tolerance_3end TOLERANCE_3END, -t3 TOLERANCE_3END
                        The number of nucleotides can be extended from the end
                        of transcript for searching stop codon. Default is 30.
  --tolerance_5end TOLERANCE_5END, -t5 TOLERANCE_5END
                        The number of nucleotides can be extended from the
                        starting point of transcript for searching start
                        codon. Default is 5.
  --print_all_combination, -pa
                        For printing all combinations of multiple start and
                        stop codons. Default is False.
  --best_no_srna, -bs   Excluding the sORFs which overlap with sRNAs to highly
                        confidence sORF list. Default is False.
  --best_no_tss, -bt    Excluding the sORFs which do not start with TSS to
                        highly confidence sORF list. Default is False.
  --ignore_hypothetical_protein IGNORE_HYPOTHETICAL_PROTEIN, -ih IGNORE_HYPOTHETICAL_
↪PROTEIN
                        For ignoring hypothetical protein in the genome
                        annotation file. Default is False.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/sORFs`.

statistics: Stores statistic files.

tables: Stores tables of the sORFs with more details. There are also some sub-folders:

best_candidates: Stores the best results of sORFs after filtering.

all_candidates: Stores all candidates without filtering.

The meaning of each column is as following:

Genome: Genome name.

Name: the sORF name which is also shown in gff file.

Start: Starting point of this sORF.

End: End point of this sORF.

Strand: Strand of this sORF.

Type: sORF type.

TSS: TSSs which are associated with this sORF.

Ribosome_binding_site: Ribosome binding site (Shine-Dalgarno sequence) of this sORF.

All_start_points: Positions of all start codons which can be found in the region of this sORF.

All_stop_points: Positions of all stop codons which can be found in the region of this sORF.

Conflict_sRNA: If this sORF overlaps sRNA, the overlapped sRNA will be showed here.

Frame_shift: If there are sORF candidates which can be found by frame shift, the number of frame shift will be showed here. `1` means there are some candidates can be found by frame shift once. `2` means there are some candidates can be found by frame shift twice.

Lib_type: This sORF can be detected in TEX+/- or fragmented/conventional libraries.

Best_avg_coverage: Based on coverage of all libraries, The best average coverage of this sORF will be showed here.

Seq: Sequence of this sORF.

Avg_coverage:$LIB_NAME: Shows the average coverage information of the libraries about this sORF.

gffs: Stores gff files of the sORFs. There are also some sub-folders:

best_candidates: Stores the best results of sORFs after filtering.

all_candidates: Stores all candidates without filtering.

Some useful information can be found in the tags of the attributes within the sORF gff file. Based on this information, we can know the details of the specific sORF. The tags are as following:

start_TSS: Shows this sORF starts with which TSS.

with_TSS: Which TSSs are associated with this sORF.

sORF_type: Type of the sORF (5'UTR, 3'UTR, interCDS, intergenic, antisense or within CDS).

sRNA: Which sRNAs are overlap with this sORF.

rbs: Ribosome binding sites (Shine-Dalgarno sequences) of this sORF.

frame_shift: The number of frame shifts in the regions.

1.4.15 promoter (Promoter motif detection)

`promoter` can scan the upstream of TSSs to discover the promoter motifs. We integrated MEME (for ungapped motifs) and GLAM2 (for gapped motifs) to predict the promoters. Based on the tool, HTML files can be generated for visualization. If the input TSS gff file is not computed by ANNOgesic, please add `--tss_source` to classify TSSs for predicting promoter motifs.

- **Required tools**

MEME.

GLAM2

MPICH (if parallel runs are required)

- **Required files**

Fasta files of the genome sequences

Gff files of the TSSs: If the input TSS gff file is not generated by ANNOgesic, the libraries and wiggle files are necessary. Please refer to the *The input format of libraries for running ANNOgesic* in order to assign the correct format. And for the details of TSS, please check the section *tss_ps (TSS and processing site prediction)*.

- **Basic arguments**

```
usage: annogesic promoter [--program {meme,glam2,both}] --project_path
                          PROJECT_PATH --fasta_files FASTA_FILES
                          [FASTA_FILES ...] --tss_files TSS_FILES
                          [TSS_FILES ...]
                          [--use_tss_type USE_TSS_TYPE [USE_TSS_TYPE ...]]
                          [--motif_width MOTIF_WIDTH [MOTIF_WIDTH ...]]
                          [--num_motifs NUM_MOTIFS]
                          [--nt_before_tss NT_BEFORE_TSS]
                          [--additional arguments]

basic arguments:
  --program {meme,glam2,both}, -p {meme,glam2,both}
                        Please choose the program -- meme, glam2 or both.
                        Default is both
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --tss_files TSS_FILES [TSS_FILES ...], -t TSS_FILES [TSS_FILES ...]
                        Paths of the TSS gff files.
  --use_tss_type USE_TSS_TYPE [USE_TSS_TYPE ...], -u USE_TSS_TYPE [USE_TSS_TYPE ...]
                    The types of TSSs for generating promoter information.
                    The options are: 1. all TSSs, 2. primary TSSs, 3.
                    secondary TSSs, 4. internal TSSs, 5. antisense TSSs,
                    6. orphan TSSs, 7. all TSSs without orphan ones.
                    Multi-choices are allowed (split by space), ex: if 1 2
                    3 are assigned, the prediction will run three times
                    for all TSSs, primary TSSs and secondary TSSs. Default
                    is 1.
  --motif_width MOTIF_WIDTH [MOTIF_WIDTH ...], -w MOTIF_WIDTH [MOTIF_WIDTH ...]
                        Length of the motifs to detects. For a range insert
                        "-" between two values. Moreover, if multiple lengths
                        need to be assigned, please use spaces to separate
                        them. For an example, 50 2-10 means that the lengths
                        of motifs are 50 and within 2 to 10. The number should
```

(continues on next page)

(continued from previous page)

```
                        be less or equal than --nt_before_TSS. Default is 50.
  --num_motifs NUM_MOTIFS, -n NUM_MOTIFS
                        The number of motifs. Default is 10.
  --nt_before_tss NT_BEFORE_TSS, -b NT_BEFORE_TSS
                        The number of nucleotides upstream of the TSSs for
                        promoter prediction. Default is 50.
```

- **Additional arguments**

```
additional arguments:
  --meme_path MEME_PATH
                        path of MEME.
  --glam2_path GLAM2_PATH
                        path of GLAM2.
  --e_value E_VALUE, -e E_VALUE
                        The maximum e value for running MEME. Default is 0.05.
  --end_run END_RUN, -er END_RUN
                        If the result of GLAM2 is not improved after running
                        this number of iteration, it will be ended. Default is
                        10000.
  --parallels PARALLELS, -pl PARALLELS
                        The number of parallel jobs.
  --tss_source, -s      The TSS gff files are generated from ANNOgesic or not.
                        Default is True (from ANNOgesic)
  --tex_libs TEX_LIBS [TEX_LIBS ...], -tl TEX_LIBS [TEX_LIBS ...]
                        If --tss_source is False, please assign the name of
                        the TEX+/- library. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For an example,
                        $WIG_PATH_1:tex:1:a:+ $WIG_PATH_2:tex:1:a:-.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        If --tss_source is False, please assign the paths of
                        the genome annotation gff files containing CDSs,
                        tRNAs, rRNAs, etc.
  --combine_all, -c     Generate global promoter motifs across all reference
                        sequences in the TSS files. Default is False.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/promoters`. The output folders are following:

allfasta: If `--combine_all` is True, it will combine all TSS files in `--tss_files` to generate promoter motifs. The results will be stored in this folder.

fasta_classes: The fasta files of different TSS types.

$GENOME_NAME: Stores output of MEME and GLAM2 based on different TSS types. Two sub-folders are under this folder.

MEME: Stores the results of MEME.

GLAM2: Stores the results of GLAM2.

The format of folders which under these two folders is `promoter_motifs_$FILENAME_$GENOME_$TSSTYPE_$PROMOTE`
ex: `promoter_motifs_NC_000915.1_allgenome_internal_45_nt`. `NC_000915.1`, `allgenome`, `primary` and `45_nt` are gff filename, genome name, TSS type and upstream nucleotides of TSS, respectively. If genome name is `allgenome`, this means that the result is generated by the information of all genomes of gff files. If there is only one genome in the gff file, the genome name will be assigned as `allgenome` as well. Several files are stored in the sub-folder:

Figures of the promoter motifs: Contains EPS and PNG files.

Details of the promoter motifs: Contains HTML file, XML file and TXT file. These files include the TSS information.

Promoter tables: `meme.csv` or `glam2.csv` is the promoter table which also includes the TSS information. Moreover, it can used as an input for sRNA detection (`srna`). Please check the section `srna`.

TSS_classes: If the TSSs are not computed by ANNOgesic, `TSS_classes` will be generated for classification of TSS. TSS gff files with TSS types will be stored here.

1.4.16 operon (Operon detection)

`operon` will search operons and sub-operons based on TSSs, transcripts, and genes. If the transcripts are not associated with genes, they would not be counted as operons.

- **Required files**

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

Gff files of the transcripts: Please check the section *transcript (transcript detection)*.

- **Optional input files**

Gff files of the TSSs: We strongly recommend to input this file for detecting sub-operon. Please check the section *tss_ps (TSS and processing site prediction)*.

Gff files of the terminators: Please check the section *terminator (terminator detection)*.

- **Basic arguments**

```
usage: annogesic operon --project_path PROJECT_PATH
                        [--tss_files TSS_FILES [TSS_FILES ...]]
                        --annotation_files ANNOTATION_FILES
                        [ANNOTATION_FILES ...] --transcript_files
                        TRANSCRIPT_FILES [TRANSCRIPT_FILES ...]
                        [--terminator_files TERMINATOR_FILES [TERMINATOR_FILES ...]]
                        [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --tss_files TSS_FILES [TSS_FILES ...], -t TSS_FILES [TSS_FILES ...]
                        Paths of the TSS gff files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs etc. Gff files of TSSs,
                        transcripts, terminators, need to be separately
                        assigned to --tss_files, --transcript_files,
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
```

(continues on next page)

(continued from previous page)

```
                        Paths of the transcript gff files.
  --terminator_files TERMINATOR_FILES [TERMINATOR_FILES ...], -e TERMINATOR_FILES␣
↪[TERMINATOR_FILES ...]
                        Paths of terminator gff files.
```

- **Additional arguments**

```
additional arguments:
  --tss_tolerance TSS_TOLERANCE, -tt TSS_TOLERANCE
                        The 5'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated TSSs. Default is 5.
  --terminator_tolerance TERMINATOR_TOLERANCE, -et TERMINATOR_TOLERANCE
                        The 3'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated terminators. Default is 30.
  --min_length MIN_LENGTH, -l MIN_LENGTH
                        The minimum length of operon. Default is 20.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/operons`. The output folders are as following:

gffs: The gff files of operons. **associated_gene** shows the genes located in the operon.

tables: The tables of operons which store all information of operons and sub-operons.

The meaning of each column is as following:

Operon_ID: Operon ID.

Genome: Genome name.

Operon_position: Starting point and end point of the operon.

Strand: Strand of the operon.

Number_of_suboperon: The amount of sub-operons in this operon region.

Position_of_suboperon: Starting point and end point of the sub-operons.

Start_with_TSS: This operon starts with TSS or not.

Number_of_TSS: The number of the TSSs which are located on this operon.

Terminated_with_terminator: This operon ends with TSS or not.

Number_of_terminator: The number of the terminators which are associated with this operon.

Number_of_gene_associated_suboperon: The number of the genes which are associated with the sub-operon.

Number_of_gene_associated_operon: The number of the genes which are associated with the operon.

Associated_genes_with_suboperon: Locus tag of the genes which are associated with the sub-operon.

Associated_genes_with_whole_operon: Locus tag of the genes which are associated with the operon.

statistics: Stores statistic file which includes the number of sub-operons, monocistronic operon, polycistronic operon, etc.

1.4.17 circrna (circular RNA detection)

`circrna` can detect the potential circular RNAs via Segemehl. Moreover, the false positive can be removed by checking genome annotation files and quality of splicing site detection. The user can assign reads for detecting circular RNAs or assign BAM files to skip mapping. BE CAREFUL, BAM files must be mapped by Segemehl with `--splits` or `circrna` can't find the proper candidates.

- **Required tools**

segemehl.x and testrealign.x in Segemehl. For generating testrealign.x, please refer to *Required tools or databases*.

- **Required files**

Fasta files of reads or BAM files: If you want to use BAM files directly, they should be mapped by Segemehl with `--splits`. The input format is `$SET_NAME:$READ1,$READ2,...` or `$SET_NAME:$BAM1,$BAM2,...`. For an example, `set1:read1.fa,read2.fa` means these two fasta files need to be computed together. If your BAM files are generated by mapping reads on multiple reference genomes, testrealign.x may not be able to handle them.

Fasta files of the genome annotations

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

- **Basic Arguments**

```
usage: annogesic circrna --project_path PROJECT_PATH
                         [--read_files READ_FILES [READ_FILES ...]]
                         [--bam_files BAM_FILES [BAM_FILES ...]] --fasta_files
                         FASTA_FILES [FASTA_FILES ...] --annotation_files
                         ANNOTATION_FILES [ANNOTATION_FILES ...]
                         [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --read_files READ_FILES [READ_FILES ...], -rp READ_FILES [READ_FILES ...]
                        Paths of read fasta or fastq files. ANNOgesic will map
                        the reads via segemehl (with -S). Required format:
                        $SET_NAME:$READ1,$READ2,... If multiple data sets need
                        to be assigned, please separated them by spaces. The
                        read files compressed by bz2 or gz files can be
                        accepted as well. For using BAM files, please check
                        --bam_files.
  --bam_files BAM_FILES [BAM_FILES ...], -b BAM_FILES [BAM_FILES ...]
                        Path of input BAM files. Required format:
                        $SET_NAME:$BAM1,$BAM2,... . BAM files need to be
                        generated using the mapper segemehl with the parameter
                        "-S". If multiple data sets need to be assigned,
                        please separated them by spaces. For using fasta files
                        of reads, please check --read_files.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs, tRNAs, rRNAs, etc.
```

- **Additional arguments**

```
additional arguments:
  --segemehl_path SEGEMEHL_PATH
                        Path of segemehl.x in segemehl package.
  --testrealign_path TESTREALIGN_PATH
                        Path of testrealign.x in segemehl package.
  --samtools_path SAMTOOLS_PATH
                        Path of samtools.
  --parallels PARALLELS, -p PARALLELS
                        The number of parallels runs for mapping if
                        --read_files is assigned. Default is 10.
  --support_reads SUPPORT_READS, -s SUPPORT_READS
                        The minimum reads for supporting a circular RNA.
                        Default is 10.
  --start_ratio START_RATIO, -sr START_RATIO
                        The minimum ratio -- (reads of circRNA / all reads) at
                        the starting points of candidates. Default is 0.5.
  --end_ratio END_RATIO, -er END_RATIO
                        Similar to --start_ratio. This value is for the end
                        points of candidates. Default is 0.5.
  --ignore_hypothetical_protein IGNORE_HYPOTHETICAL_PROTEIN, -ih IGNORE_HYPOTHETICAL_
↪PROTEIN
                        For ignoring hypothetical protein in the genome
                        annotation file. Default is False.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/circRNAs`. The output folders are following:

segemehl_alignment_files: If read files are assigned, the folder is for results of mapping.

segemehl_splice_results: The results of splicing detection. For understanding the splicing tables, please refer to Segemehl.

gffs: Stores gff files of the circular RNAs. `$GENOME_$SET_circRNA_best.gff` is gff files for the best results after checking genome annotation and quality of splicing. `$GENOME_$SET_circRNA_all.gff` is for all candidates without filtering. Some useful information can be found in the tags of the attributes within the circular RNA gff file. Based on this information, we can know the details of the specific circular RNA. The tags are as following:

support_reads: The number of reads which support the circular RNA.

read_at_start: (supported reads / total reads) at starting point of the circular RNA.

read_at_end: (supported reads / total reads) at end point of the circular RNA.

conflict: The circular RNA overlap genome annotation or not.

method: The circular RNA is detected by which method.

circRNA_tables: Stores tables of the circular RNAs with more details. The meaning of each column is as following:

Genome: Genome name.

Strand: Strand of the circular RNA.

Start: Starting point of the circular RNA.

End: End point of the circular RNA.

Annotation_overlap: If there is a genome annotation (like a CDS) which overlap this circular RNA, the overlapped feature will be showed here.

Supported_reads: The number of reads which support the circular RNA.

Supported_reads/Reads_at_start: (supported reads / total reads) at starting point of the circular RNA.

Supported_reads/Reads_at_end: (supported reads / total reads) at end point of the circular RNA.

1.4.18 go_term (GO term retrieving)

`go_term` can retrieve the information of Gene Ontology from Uniprot. Some analyses of GO terms can be done as well.

- **Required files**

Uniprot mapping table: idmapping_selected.tab from Uniprot.

GOslim file: goslim.obo.

GO file: go.obo.

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

- **Optional input files**

Gff files of the transcripts: For detecting the GO terms only based on expressed CDSs. Please check the section *transcript (transcript detection)*.

- **Arguments**

```
usage: annogesic go_term [-h] --project_path PROJECT_PATH --annotation_files
                         ANNOTATION_FILES [ANNOTATION_FILES ...]
                         [--transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...]]
                         --uniprot_id UNIPROT_ID --go_obo GO_OBO --goslim_obo
                         GOSLIM_OBO

optional arguments:
  -h, --help            show this help message and exit

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containg
                        CDSs.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
                        Paths of the transcript gff files which can be used to
                        retrieve GO terms based on expressed CDS and all CDS.
  --uniprot_id UNIPROT_ID, -u UNIPROT_ID
                        Path of the UniProt ID mapping database.
  --go_obo GO_OBO, -go GO_OBO
                        Path of go.obo.
  --goslim_obo GOSLIM_OBO, -gs GOSLIM_OBO
                        Path of goslim.obo.
```

- **Output files**

Output files are stored in `ANNOgesic/output/GO_terms`. If gff files of the transcript are assigned, two sub-folders will be generated. Results of the expressed genes are stored in `ANNOgesic/output/GO_term/expressed_CDSs` and results of all CDSs are stored in `ANNOgesic/output/GO_term/all_CDSs`.

GO_term_results: Stores tables of the GO terms information. The meaning of each column is as following:

Genome: Genome name.

Strand: Strand of the gene.

Start: Starting point of this CDS.

End: End point of this CDS.

Protein_id Protein ID of this CDS.

GO_term GO term of This CDS.

statistics: Stores statistic files and figures.

GO term with corresponding amount: Format of the filename is `stat_$GENOME.csv`.

Figures of the three GO term classes: All figures are stored in the sub-folder - `figs`. `$GENOME_biological_process.png`, `$GENOME_cellular_component.png`, `$GENOME_molecular_function.png` and `$GENOME_three_roots.png` are the figures of biological process, cellular component, molecular_function and three roots of GO term classes, respectively.

1.4.19 srna_target (sRNA target prediction)

`srna_target` can search potential targets of the sRNA via different programs (RNAup or RNAplex or both). We recommend running with both programs. `srna_target` can also compare the results of both programs and provide the best ones.

- **Required tools**

ViennaRNA . IntaRNA.

- **Required files**

Gff files of the genome annotations containing CDSs

Gff files of the sRNAs: Please check the section *srna (sRNA detection)*.

Fasta files of the genomes

- **Basic arguments**

```
usage: annogesic srna_target --project_path PROJECT_PATH
                             --annotation_files ANNOTATION_FILES
                             [ANNOTATION_FILES ...] --fasta_files FASTA_FILES
                             [FASTA_FILES ...] --srna_files SRNA_FILES
                             [SRNA_FILES ...]
                             [--query_srnas QUERY_SRNAS [QUERY_SRNAS ...]]
                             --program {RNAplex,RNAup,IntaRNA}
                             [{RNAplex,RNAup,IntaRNA} ...] [--top TOP]
                             [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        CDSs.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
```

(continues on next page)

(continued from previous page)

```
                        Paths of the genome fasta files.
  --srna_files SRNA_FILES [SRNA_FILES ...], -s SRNA_FILES [SRNA_FILES ...]
                        Paths of the sRNA gff files.
  --query_srnas QUERY_SRNAS [QUERY_SRNAS ...], -q QUERY_SRNAS [QUERY_SRNAS ...]
                        The query sRNA. The input format is
                        $GENOME:$START:$END:$STRAND. If multiple sRNAs need to
                        be assigned, please use spaces to separate them. For
                        example, NC_007795.1:200:534:+
                        NC_007795.1:6767:6900:-. If all sRNAs of the reference
                        genome need to be used, please assign "all". Default
                        is all.
  --program {RNAplex,RNAup,IntaRNA} [{RNAplex,RNAup,IntaRNA} ...], -p {RNAplex,RNAup,
↪IntaRNA} [{RNAplex,RNAup,IntaRNA} ...]
                        The program for detecting sRNA-mRNA interaction.
                        Please choose "RNAplex", "RNAup" or "IntaRNA". If
                        multiple programs need to be executed, please use
                        space to separate them.
  --top TOP, -t TOP     The ranking number of targets which will be included
                        to final output. The ranking is based on the binding
                        energy. Default is 50.
```

- **Additional arguments**

```
additional arguments:
  --rnaplfold_path RNAPLFOLD_PATH
                        Path of RNAplfold of the Vienna package.
  --rnaplex_path RNAPLEX_PATH
                        Path of RNAplex of the Vienna package.
  --rnaup_path RNAUP_PATH
                        Path of RNAup of the Vienna package.
  --intarna_path INTARNA_PATH
                        Path of IntaRNA.
  --interaction_length INTERACTION_LENGTH, -i INTERACTION_LENGTH
                        Maximum length of an interaction. Default is 30.
  --window_size_target_rnaplex WINDOW_SIZE_TARGET_RNAPLEX, -wt WINDOW_SIZE_TARGET_
↪RNAPLEX
                        The average of the pair probabilities over windows for
                        mRNA target. It is only applied for "RNAplex". Default
                        is 240.
  --span_target_rnaplex SPAN_TARGET_RNAPLEX, -st SPAN_TARGET_RNAPLEX
                        The maximum allowed separation of a base pair to span
                        for mRNA target. It is only applied for "RNAplex".
                        Default is 160.
  --window_size_srna_rnaplfold WINDOW_SIZE_SRNA_RNAPLFOLD, -ws WINDOW_SIZE_SRNA_
↪RNAPLFOLD
                        Similar to --window_size_target, but for sRNA. Default
                        is 30.
  --span_srna_rnaplfold SPAN_SRNA_RNAPLFOLD, -ss SPAN_SRNA_RNAPLFOLD
                        Similar to --span_target, but for sRNA. Default is 30.
  --unstructured_region_rnaplex_target UNSTRUCTURED_REGION_RNAPLEX_TARGET, -ut␣
↪UNSTRUCTURED_REGION_RNAPLEX_TARGET
                        Calculate the mean probability of the unpaired region
                        for mRNA target. It only works for "RNAplex". Default
                        is 30.
  --unstructured_region_rnaplex_srna UNSTRUCTURED_REGION_RNAPLEX_SRNA, -us␣
↪UNSTRUCTURED_REGION_RNAPLEX_SRNA
                        Similar to --unstructured_region_rnaplex_target, but
```

(continues on next page)

(continued from previous page)

```
                        for sRNA. Default is 30.
  --unstructured_region_rnaup UNSTRUCTURED_REGION_RNAUP, -uu UNSTRUCTURED_REGION_RNAUP
                        Compute the mean probability of unpaired region. It
                        only works for "RNAup". Default is 40.
  --energy_threshold_rnaplex ENERGY_THRESHOLD_RNAPLEX, -e ENERGY_THRESHOLD_RNAPLEX
                        The minimum energy for a duplex. It only works for
                        "RNAplex". Default is -8.
  --duplex_distance_rnaplex DUPLEX_DISTANCE_RNAPLEX, -d DUPLEX_DISTANCE_RNAPLEX
                        Distance between target 3'ends of two consecutive
                        duplexes. It works for "RNAplex". Default is 20.
  --parallels_rnaplex PARALLELS_RNAPLEX, -pp PARALLELS_RNAPLEX
                        The number of parallel jobs for running RNAplex.
                        Default is 5.
  --parallels_rnaup PARALLELS_RNAUP, -pu PARALLELS_RNAUP
                        The number of parallel jobs for running RNAup. Default
                        is 20.
  --parallels_intarna PARALLELS_INTARNA, -pi PARALLELS_INTARNA
                        The number of parallel jobs for running IntaRNA.
                        Default is 10.
  --continue_rnaup, -cr
                        For running RNAup based on the previous intermediate
                        results if the previous process stopped. Default is
                        False.
  --slide_window_size_srna_intarna SLIDE_WINDOW_SIZE_SRNA_INTARNA, -sw SLIDE_WINDOW_
↪SIZE_SRNA_INTARNA
                        The silding window size of sRNA sequences. 0 will use
                        the full sequence to execute IntaRNA. Default is 150.
  --max_loop_length_srna_intarna MAX_LOOP_LENGTH_SRNA_INTARNA, -ls MAX_LOOP_LENGTH_
↪SRNA_INTARNA
                        The maximal loop length of sRNA. If the value is
                        assigned by 0, --slide_window_size_srna_intarna will
                        be used for the maximal loop length of sRNA. Default
                        is 100.
  --slide_window_size_target_intarna SLIDE_WINDOW_SIZE_TARGET_INTARNA, -tw SLIDE_
↪WINDOW_SIZE_TARGET_INTARNA
                        The silding window size of target sequences. 0 will
                        use the full sequence to execute IntaRNA. Default is
                        150.
  --max_loop_length_target_intarna MAX_LOOP_LENGTH_TARGET_INTARNA, -lt MAX_LOOP_
↪LENGTH_TARGET_INTARNA
                        The maximal loop length of target. If the value is
                        assigned by 0, --slide_window_size_target_intarna will
                        be used for the maximal loop length of target. Default
                        is 100.
  --mode_intarna {H,E,M}, -mi {H,E,M}
                        The prediction mode of IntaRNA. 'H' is heuristic, 'M'
                        is exact with long computational time. 'E' is exact
                        with long computational time and high memory.
  --potential_target_start POTENTIAL_TARGET_START, -ps POTENTIAL_TARGET_START
                        Distance for the extraction of upstream nucleotides of
                        --target_feature. Default is 200.
  --potential_target_end POTENTIAL_TARGET_END, -pe POTENTIAL_TARGET_END
                        Distance for the extraction of downstream nucleotides
                        of --target_feature. Default is 150.
  --target_feature TARGET_FEATURE [TARGET_FEATURE ...], -tf TARGET_FEATURE [TARGET_
↪FEATURE ...]
                        The feature name of potential targets. If multiple
```

(continues on next page)

(continued from previous page)

```
                        features need to be assigned, please use spaces to
                        separate them. For example, CDS exon. Default is CDS.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/sRNA_targets`.

RNAplex_results: Stores all results of RNAplex. `$GENOME_RNAplex.txt` is raw results of RNAplex. `$GENOME_RNAplex_rank.csv` is the tables with details, and the targets are sorted by binding energy. The meaning of each column in `$GENOME_RNAplex_rank.csv` is as following:

sRNA: sRNA name which is shown in sRNA gff file.

Genome: Genome name.

sRNA_position: Starting point and end point of this sRNA.

sRNA_interacted_position_RNAplex: The interaction region of this sRNA.

sRNA_strand: Strand of this sRNA.

Target: Locus tag or gene name of the target mRNA.

Target_position: Starting point and end point of this mRNA.

Target_interacted_position_RNAplex: The interaction region of this mRNA.

Target_strand: Strand of this target mRNA.

Energy_RNAplex: Interaction energy change of this interaction.

Rank_RNAplex: Ranking of the interaction (the ranking is based on the binding energy).

RNAup_results: Stored all results of RNAup. `$GENOME_RNAup.txt` is raw results of RNAup. `$GENOME_RNAup_rank.csv` is the tables with details, and the targets are sorted by binding energy. The meaning of each column is similar to the table of RNAplex.

IntaRNA_results: Stored all results of IntaRNA. `$GENOME_IntaRNA.txt` is raw results of IntaRNA. `$GENOME_RNAup_rank.csv` is the tables with details, and the targets are sorted by binding energy. The meaning of each column is similar to the table of RNAplex.

merged_results: Store the results which are merged by the results of `RNAplex_results`, `RNAup_results`, and `IntaRNA_results`. `$GENOME_merge.csv` contains all candidates of the all assigned programs. `$GENOME_overlap.csv` contains the results which are top 50 (default) in the all assigned methods. The meaning of each column is similar to the table of RNAplex.

sRNA_seqs: Stores fasta sequences of the sRNAs.

target_seqs: Stores fasta sequences of the potential targets.

1.4.20 ppi_network (protein-protein interaction network detection)

`ppi_network` can retrieve the PPI data from STRING. Then using PIE to search the supported literature of the protein-protein interaction networks.

- **Required files**

Species table of STRING: species.v${VERSION}.txt from STRING.

Gff files of the genome annotations containing CDSs

- **Basic arguments**

```
usage: annogesic ppi_network --project_path PROJECT_PATH
                             --annotation_files ANNOTATION_FILES
                             [ANNOTATION_FILES ...] --species_string
                             SPECIES_STRING --query_strains QUERY_STRAINS
                             [QUERY_STRAINS ...] [--query QUERY [QUERY ...]]
                             [--without_strain_pubmed] [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff files containing
                        genes and CDSs with proper locus_tag items in the
                        attributes.
  --species_string SPECIES_STRING, -d SPECIES_STRING
                        Path of the species table of STRING
                        (species.$VERSION.txt).
  --query_strains QUERY_STRAINS [QUERY_STRAINS ...], -s QUERY_STRAINS [QUERY_STRAINS .
↪..]
                        The name of the input file of the query genomes.
                        Required format: $GFF_FILE:$STRAIN_IN_GFF:$STRAIN_IN_S
                        TRING:$STRAIN_FOR_PUBMED. $GFF_FILE is the name of the
                        gff file, $STRAIN_IN_GFF is the name/ID of the strain
                        in the gff file, $STRAIN_IN_STRING is the strain name
                        in species table of STRING (species.$VERSION.txt), and
                        $STRAIN_FOR_PUBMED is the strain name for searching in
                        Pubmed. If the strain is not available in STRING
                        database, it can be relaced by a related strain. For
                        example, Staphylococcus_aureus_HG003.gff:Staphylococcu
                        s_aureus_HG003:"Staphylococcus aureus NCTC
                        8325":"Staphylococcus aureus" (Staphylococcus aureus
                        NCTC 8325 is a related strain of HG003 since HG003 is
                        not available in STRING). If the assigned name is with
                        spaces, please use double quotes. For assigning
                        multiple strains, please separated them by spaces.
  --query QUERY [QUERY ...], -q QUERY [QUERY ...]
                        The query proteins. Required format:
                        $GENOME_NAME_OF_GFF:$START_POINT:$END_POINT:$STRAND.
                        For assigning multiple proteins, please use spaces to
                        separate them. For example,
                        Staphylococcus_aureus_HG003:345:456:+
                        Staphylococcus_aureus_HG003:2000:3211:-. For computing
                        all proteins in gff files, just type "all". Default is
                        all.
  --without_strain_pubmed, -n
                        For retrieving the literature from Pubmed only based
                        on protein name without assigning strains. Default is
                        False.
```

- **Additional arguments**

```
additional arguments:
  --score SCORE, -ps SCORE
                        The minimum PIE score for searching literature. The
                        value is from -1 (worst) to 1 (best). Default is 0.
```

(continues on next page)

(continued from previous page)

```
  --node_size NODE_SIZE, -ns NODE_SIZE
                        The size of nodes in figure, default is 4000.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/PPI_networks`. The output folders are as following:

best_results: Stores the results which the scores of PIE for supported literature are higher than `--score`.

all_results: Stores the results of all protein-protein interactions (including the low score(PIE) literature).

Under `best_results` and `all_results`, several files and folders are generated:

Results of searching literatures without assigning a specific strain: `$STRAIN_without_strain.csv`.

Results of searching literatures with assigning a specific strain: `$STRAIN_with_strain.csv`.

without_strain: Stores all interaction information which is searched without assigning a specific strain.

with_strain: Stores all interaction information which is searched with assigning a specific strain.

figures: Stores the protein-protein networks of the query proteins. Thickness represents how many literature can be found for the interactions. Solid line means that strong supported literature can be found. Dash-dot line means that the supported literature are very weak. Dot line means that no supported literature can be found. Color is the best score of the supported literature of the interactions.

1.4.21 localization (subcellular localization prediction)

`localization` can predict the subcellular localization of proteins. Some statistics and visualization files are provided as well.

- **Required tools**

Psortb.

- **Required files**

Gff files of the genome annotations containing CDSs

Fasta files of the genome sequences

- **Optional input files**

Gff files of the transcripts: For detecting subcellular localization only based on expressed CDSs. Please check the section *transcript (transcript detection)*.

- **Basic arguments**

```
usage: annogesic localization --project_path PROJECT_PATH
                              --annotation_files ANNOTATION_FILES
                              [ANNOTATION_FILES ...] --fasta_files FASTA_FILES
                              [FASTA_FILES ...]
                              [--transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES .
↪..]]
                              --bacteria_type {positive,negative}
                              [--additional arguments]
```

(continues on next page)

(continued from previous page)

```
basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of genome annotation gff files containing CDSs.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of genome fasta files.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
                        Paths of the transcript gff files for detecting the
                        subcellular localization based on expressed CDS and
                        all CDS.
  --bacteria_type {positive,negative}, -b {positive,negative}
                        The type of bacteria (Gram-positive or Gram-negative).
                        Please assign 'positive' or 'negative'.
```

- **Additional arguments**

```
additional arguments:
  --psortb_path PSORTB_PATH
                        Path of Psortb.
  --difference_multi DIFFERENCE_MULTI, -d DIFFERENCE_MULTI
                        For the protein which have multiple predicted
                        locations, if the difference of psorb scores is
                        smaller than this value, all locations will be printed
                        out. Default is 0.5. The maximum value is 10.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/subcellular_localization`. If gff files of the transcripts are assigned, two sub-folders will be generated. Results of the expressed CDSs are stored in `ANNOgesic/output/subcellular_localization/expressed_CDSs` and results of all CDS are stored in `ANNOgesic/output/subcellular_localization/all_CDSs`.

psortb_results: Stores the results of Psortb:

Raw output data of Psortb: Format of the filename is `$GENOME_raw.txt`.

Table of subcellular localization: Format of the filename is `$GENOME_table.csv`. The meaning of each column is as following:

Genome: Genome name.

Protein: Protein ID.

Strand: Strand of this protein.

Start: Starting point of this protein.

End: End point of this protein.

Location: Predicted subcellular localization of this protein.

Score: Psortb score.

statistics: Stores statistic files and figures.

Subcellular localization with corresponding amounts: Format of the filename is `stat_$GENOME_sublocal.csv`.

Figure of Subcellular localization with corresponding amounts: Format of the filename is `$FILENAME_$GENOME_sublocal.png`.

1.4.22 riboswitch_thermometer (riboswitch and RNA thermometer detection)

`riboswitch_thermometer` can search riboswitches and RNA thermometers between TSSsthe starting point of transcript was assigned if no TSS was detected) and its downstream CDSs, as well as associated ribosome binding sites (Shine-Dalgarno sequence). Then using Infernal to scan the region in Rfam.

- **Required tools**

Infernal.

- **Required files**

Rfam.

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

Gff files of the transcripts: Please check the section *transcript (transcript detection)*.

Fasta files of the genome sequences

Rfam ID files of the riboswitch or RNA thermometer: The file should contain Rfam IDs, name and description of riboswitches or RNA thermometers as following.

#Rfam_ID	Name	Description
RF00162	SAM	SAM riboswitch box leader
RF00059	TPP	TPP riboswitch THI element

All columns are separated by `tab`. You can also download riboswitch and RNA thermometer data from our Git repository.

- **Optional input files**

Gff files of the TSSs: For checking the ribosome binding site. We strongly recommend to input this file. Please check the section *tss_ps (TSS and processing site prediction)*.

- **Basic arguments**

```
usage: annogesic riboswitch_thermometer --project_path PROJECT_PATH
                                        [--program {riboswitch,thermometer,both}]
                                        [--riboswitch_id_file RIBOSWITCH_ID_FILE]
                                        [--rna_thermometer_id_file RNA_THERMOMETER_ID_
↪FILE]
                                        --annotation_files ANNOTATION_FILES
                                        [ANNOTATION_FILES ...]
                                        [--tss_files TSS_FILES [TSS_FILES ...]]
                                        --transcript_files TRANSCRIPT_FILES
                                        [TRANSCRIPT_FILES ...] --fasta_files
                                        FASTA_FILES [FASTA_FILES ...]
                                        --rfam_path RFAM_PATH
                                        [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
```

(continues on next page)

(continued from previous page)

```
                        Path of the project folder.
  --program {riboswitch,thermometer,both}, -p {riboswitch,thermometer,both}
                        Please choose the feature for the detection. The
                        options can be "riboswitch", "thermometer", "both".
                        Default is both.
  --riboswitch_id_file RIBOSWITCH_ID_FILE, -ri RIBOSWITCH_ID_FILE
                        Path of the file which contains the information of
                        riboswitches in Rfam. Required format of the file:
                        $RFAM_ID{tab}$RIBOSWITCH_NAME{tab}$DESCRIPTION. Please
                        check an example in https://github.com/Sung-Huan/ANNOg
                        esic/blob/master/database/Rfam_riboswitch_ID.csv
  --rna_thermometer_id_file RNA_THERMOMETER_ID_FILE, -ti RNA_THERMOMETER_ID_FILE
                        Same format as for -riboswitch_id_file, but for RNA
                        thermometers. Please check an example in
                        https://github.com/Sung-Huan/ANNOgesic/blob/master/dat
                        abase/Rfam_RNA_thermometer_ID.csv
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the annotation gff files.
  --tss_files TSS_FILES [TSS_FILES ...], -t TSS_FILES [TSS_FILES ...]
                        Paths of the TSS gff files.
  --transcript_files TRANSCRIPT_FILES [TRANSCRIPT_FILES ...], -a TRANSCRIPT_FILES␣
↪[TRANSCRIPT_FILES ...]
                        Paths of the transcript gff files.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --rfam_path RFAM_PATH, -R RFAM_PATH
                        Path of the Rfam CM database.
```

- **Additional arguments**

```
additional arguments:
  --cmscan_path CMSCAN_PATH, -cs CMSCAN_PATH
                        Path of cmscan in Infernal package.
  --cmpress_path CMPRESS_PATH, -cp CMPRESS_PATH
                        Path of cmpress in Infernal package.
  --utr_length UTR_LENGTH, -u UTR_LENGTH
                        The UTR length. Default is 300.
  --cutoff CUTOFF, -cf CUTOFF
                        The cutoff of the infernal search. The cutoff can be
                        assigned by e value (assigned by 'e') or score
                        (assigned by 's'). For example, 'e_0.001' represents
                        using e value as a cutoff and the maximum value is
                        0.001. 's_8' represents using score as a cutoff and
                        the minimum score is 8. Default is e_0.001.
  --output_all, -o      One query sequence may fit multiple riboswitches or
                        RNA thermometers. It can print multiple riboswitches
                        or RNA thermometers. Otherwise, only the highest
                        confident one will be printed. Default is False.
  --tolerance TOLERANCE, -to TOLERANCE
                        The 5'ends and 3'ends of potential riboswitches or RNA
                        thermometers will be extended by this value
                        (nucleotides) for extracting the sequences to search
                        in Rfam. Default is 10.
  --without_rbs, -wr    Running the prediction without considering ribosome
                        binding site.Default is False.
  --rbs_seq RBS_SEQ [RBS_SEQ ...], -rs RBS_SEQ [RBS_SEQ ...]
```

(continues on next page)

(continued from previous page)

```
                        The sequences of ribosome binding site. If the
                        multiple sequences needs to be assigned, please use
                        space to split them. Default is AGGAGG.
  --tolerance_rbs TOLERANCE_RBS, -tr TOLERANCE_RBS
                        The number of nucleotides of ribosome binding site
                        allow to be different with AGGAGG. Default is 2.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files of the riboswitches are stored in `$ANNOgesic/output/riboswitches` and output files of the RNA thermometers are stored in `$ANNOgesic/output/RNA_thermometers`.

Names of the output folders are as following:

scan_Rfam_results: Stores the results of searching to Rfam with `cmscan` (Infernal).

gffs: Stores gff files of riboswitches/RNA_thermometers. Some useful information can be found in the tags of the attributes within the gff file. Based on this information, we can know the details of the specific riboswitches/RNA_thermometers. The tags are as following:

rfam_id: Rfam ID of this riboswitch/RNA_thermometer.

e-value: E-value of searching this riboswitch/RNA_thermometer to Rfam.

score: Score of searching this riboswitch/RNA_thermometer to Rfam.

method: This riboswitch/RNA_thermometer is detected by which method.

tables: Stores tables of riboswiches/RNA_thermometers with more details. The meaning of each column in this table is as following:

ID: Riboswich/RNA_thermometer ID.

Genome: Genome name.

Strand: Strand of the riboswich/RNA_thermometer.

Associated_CDS: Downstream CDS of the riboswich/RNA_thermometer.

Start_genome: This riboswich/RNA_thermometer starts from which position of the genome.

End_genome: This riboswich/RNA_thermometer ends to which position of the genome.

Rfam: Rfam ID of this riboswich/RNA_thermometer.

E_value: E-value of searching this riboswitch/RNA_thermometer to Rfam.

Score: Score of searching this riboswitch/RNA_thermometer to Rfam.

Start_align: Position of this riboswich/RNA_thermometer can be aligned to the genome.

End_align: Position this riboswich/RNA_thermometer can be aligned to the genome.

statistics: Stores the file which contains the riboswich/RNA_thermometer with corresponding amount.

1.4.23 crispr (CRISPR detection)

`crispr` integrates CRISPR Recognition Tool (CRT) which can detect the repeat units and spacers of CRISPR. Moreover, the false positive can be removed by comparing candidates with genome annotations.

- **Required tools**

CRT.

- **Required files**

Fasta files of the genome sequences

- **Optional input files**

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc: This file can be used for removing false positive.

- **Basic arguments**

```
usage: annogesic crispr --project_path PROJECT_PATH --fasta_files
                        FASTA_FILES [FASTA_FILES ...]
                        [--annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...]]
                        [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the genome fasta files.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome gff files containing CDSs for
                        comparing CRISPRs and the genome annotation to remove
                        the false positives. Default is None.
```

- **Additional arguments**

```
additional arguments:
  --crt_path CRT_PATH   Path of CRT.jar. Default is /usr/local/bin/CRT.jar
  --window_size WINDOW_SIZE, -w WINDOW_SIZE
                        Length of the window size for searching CRISPR (range:
                        6-9). Default is 8.
  --min_number_repeats MIN_NUMBER_REPEATS, -mn MIN_NUMBER_REPEATS
                        Minimum number of repeats that a CRISPR must contain.
                        Default is 3.
  --min_length_repeat MIN_LENGTH_REPEAT, -ml MIN_LENGTH_REPEAT
                        Minimum length of CRISPR repeats. Default is 23.
  --Max_length_repeat MAX_LENGTH_REPEAT, -Ml MAX_LENGTH_REPEAT
                        Maximum length of CRISPR repeats. Default is 47.
  --min_length_spacer MIN_LENGTH_SPACER, -ms MIN_LENGTH_SPACER
                        Minimum length of CRISPR spacers. Default is 26.
  --Max_length_spacer MAX_LENGTH_SPACER, -Ms MAX_LENGTH_SPACER
                        Maximum length of CRISPR spacers. Default is 50.
  --ignore_hypothetical_protein, -in
                        To ignore hypothetical proteins. Default is inactive.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output files are stored in `$ANNOgesic/output/crisprs`. The folders which are generated by the subcommand are as following:

CRT_results: Stores the output of CRT.

gffs: Stores CRSIPR gff files. **Please be aware that CRISPR has no strand information (shows '.' in gff files)**. Two sub-folders are under this folder:

all_candidates: Stores gff files which contains all CRISPRs.

best_candidates: Stores gff files which contains the CRISPRs without overlapping genome annotation.

statistics: Stores statistic files.

1.4.24 optimize_tss_ps (optimization of TSS and processing site detection)

`optimize_tss_ps` can adapt the parameter set of TSSpredator. For running it, please manual detect TSSs around 200kb and find at least 50 TSSs (using gff format). If there are less than 50 TSSs within 200kb, please continue checking until 50 TSSs are detected. Then `optimize_tss_ps` can scan whole genome based on the manual detected set to get optimized parameters.

- **Required tools**

TSSpredator.

- **Required files**

Wiggle files of TEX +/-: Please check the section *The input format of libraries for running ANNOgesic*.

Fasta files of the genome sequences

Gff files of the genome annotations containing CDSs, tRNAs, rRNAs, etc

Gff files of the manual-detected TSSs

- **Basic arguments**

```
usage: annogesic optimize_tss_ps --project_path PROJECT_PATH
                                 [--program {TSS,PS}] --fasta_files
                                 FASTA_FILES [FASTA_FILES ...]
                                 --annotation_files ANNOTATION_FILES
                                 [ANNOTATION_FILES ...] --manual_files
                                 MANUAL_FILES [MANUAL_FILES ...]
                                 [--curated_sequence_length CURATED_SEQUENCE_LENGTH␣
↪[CURATED_SEQUENCE_LENGTH ...]]
                                 --tex_notex_libs TEX_NOTEX_LIBS
                                 [TEX_NOTEX_LIBS ...]
                                 [--replicate_tex REPLICATE_TEX [REPLICATE_TEX ...]]
                                 --condition_names CONDITION_NAMES
                                 [CONDITION_NAMES ...]
                                 [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --program {TSS,PS}, -p {TSS,PS}
                        The feature for optimization. Please assign "TSS" or
                        "PS". Default is TSS.
  --fasta_files FASTA_FILES [FASTA_FILES ...], -f FASTA_FILES [FASTA_FILES ...]
                        Paths of the fasta file of the reference genome.
  --annotation_files ANNOTATION_FILES [ANNOTATION_FILES ...], -g ANNOTATION_FILES␣
↪[ANNOTATION_FILES ...]
                        Paths of the genome annotation gff file containing
                        CDSs, tRNAs, rRNAs, etc.
  --manual_files MANUAL_FILES [MANUAL_FILES ...], -m MANUAL_FILES [MANUAL_FILES ...]
```

(continues on next page)

(continued from previous page)

```
                        Paths of the manual-checked TSS or PS in gff format.
                        It is used for benchmarking the prediction during the
                        optimization. The set should comprise roughly 50
                        TSS/PS or more.
  --curated_sequence_length CURATED_SEQUENCE_LENGTH [CURATED_SEQUENCE_LENGTH ...], -
↪le CURATED_SEQUENCE_LENGTH [CURATED_SEQUENCE_LENGTH ...]
                        The length of the sequence used for the manual set of
                        TSS/PS. This value is required to calculate the
                        accurracy. If the whole genome was used write "all".
                        Otherwise use the name of the reference sequence in
                        the folowing format: $GENOME:SLENGTH. Multiple entries
                        are accepted. For an example, test.gff contains two
                        sequences s1 and s2. For s1 100 kb were checked while
                        for s2 the whole sequence was curated. The value of
                        this argument would be s1:100000 s2:all. Per default
                        all the full length of all sequences will be used.
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
                        TEX+/- wig files for TSSpredator. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_tex_forward.wig:tex:1:a:+
                        my_lib_tex_reverse.wig:tex:1:a:-.
  --replicate_tex REPLICATE_TEX [REPLICATE_TEX ...], -rt REPLICATE_TEX [REPLICATE_TEX␣
↪...]
                        This value is the minimal number of replicates that a
                        TSS has to be detected in. The format is
                        $NUMBERofCONDITION_$NUMBERofREPLICATE. If different
                        --replicate_tex values need to be assigned to
                        different conditions, please use spaces to separate
                        them. For example, 1_2 2_2 3_3 means that
                        --replicate_tex is 2 in number 1 and number 2
                        conditions. In number 3 condition, --replicate_tex is
                        3. For assigning the same --replicate_tex to all
                        conditions, just use like all_1 (--replicate_tex is 1
                        in all conditions).
  --condition_names CONDITION_NAMES [CONDITION_NAMES ...], -cn CONDITION_NAMES␣
↪[CONDITION_NAMES ...]
                        The prefixs of output filename.If multiple conditions
                        need to be assigned, please use spaces to separate
                        them. For an example, prefix_condition1
                        prefix_condition2.
  --output_id OUTPUT_ID, -oi OUTPUT_ID
                        The tag of attributes will be used for TSS
                        classification. Default is locus_tag.
```

- **Additional arguments**

```
additional arguments:
  --tsspredator_path TSSPREDATOR_PATH
                        Path of TSSpredator. Default is
                        /usr/local/bin/TSSpredator.jar
  --max_height MAX_HEIGHT, -he MAX_HEIGHT
                        This value relates to the minimum number of read
                        starts at a certain genomic position to be considered
```

(continues on next page)

(continued from previous page)

```
                        as a TSS candidate. During optimization, --max_height
                        will be never larger than this value. Default is 2.5.
  --max_height_reduction MAX_HEIGHT_REDUCTION, -rh MAX_HEIGHT_REDUCTION
                        When comparing different genomes/conditions and the
                        step height threshold is reached in at least one
                        genome/condition, the threshold is reduced for the
                        other genomes/conditions by the value set here. This
                        value must be smaller than the step height threshold.
                        During optimization, --max_height_reduction will be
                        never larger than this value. Default is 2.4.
  --max_factor MAX_FACTOR, -fa MAX_FACTOR
                        The minimum factor by which the TSS height has to
                        exceed the local expression background. During
                        optimization, --max_factor will be never larger than
                        this value. Default is 10.
  --max_factor_reduction MAX_FACTOR_REDUCTION, -rf MAX_FACTOR_REDUCTION
                        When comparing different genomes/conditions and the
                        step factor threshold is reached in at least one
                        genome/condition, the threshold is reduced for the
                        other genomes/conditions by the value set here. This
                        value must be smaller than the step factor threshold.
                        During optimization, --max_factor_reduction will be
                        never larger than this value. Default is 9.9.
  --max_base_height MAX_BASE_HEIGHT, -bh MAX_BASE_HEIGHT
                        The minimum number of reads should be mapped on TSS.
                        During optimization, --max_base_height will be never
                        larger than this value. Default is 0.06.
  --max_enrichment_factor MAX_ENRICHMENT_FACTOR, -ef MAX_ENRICHMENT_FACTOR
                        The minimum enrichment factor. During optimization,
                        --max_enrichment_factor will be never larger than this
                        value. Default is 6.0.
  --max_processing_factor MAX_PROCESSING_FACTOR, -pf MAX_PROCESSING_FACTOR
                        The minimum processing factor. If the value for the
                        untreated library is higher than the treated library
                        the positionsis considered as a processing site and
                        not annotated as detected. During optimization,
                        --max_processing_factor will be never larger than this
                        value. Default is 6.0
  --utr_length UTR_LENGTH, -u UTR_LENGTH
                        The length of UTR. Default is 300.
  --cluster CLUSTER, -cu CLUSTER
                        This value defines the maximal distance (nucleotides)
                        between TSS candidates have to be clustered together.
                        Default is 2.
  --parallels PARALLELS, -c PARALLELS
                        Number of parallel threats for optimization. Default
                        is 4.
  --steps STEPS, -s STEPS
                        Number of tota runs for the optimization. Default is
                        4000 runs.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Based on the programs (TSS/processing site), Output files are stored in `$ANNOgesic/output/TSSs/optimized_TSSpredator` or `$ANNOgesic/output/processing_sites/`

optimized_TSSpredator. Two output files are as following:

stat_$GENOME.csv: Stores the information of every run. The first column is the number of run. The second column is the parameter set. he represents height; rh represents height reduction; fa means factor; rf means factor reduction; bh indicates base height; ef indicates enrichment factor; pf means processing factor. About the details of parameters, please refer to TSSpredator. For an example, he_2.0_rh_1.8_fa_4.4_rf_2.8_bh_0.08_ef_3.0_pf_2.6 means that height is 2.0, height reduction is 1.8, factor is 4.4, factor reduction is 2.8, base height is 0.08, enrichment factor is 3.0 and processing factor is 2.6. The third column is the number of true positive. The fourth column is true positive rate. The fifth columns is the number of false positive. The sixth is false positive rate. The seventh column is the number of false negatives. The eighth column is missing rate.

best_$GENOME.csv: Stores the best parameter set. The meanings of all columns are the same as stat_$GENOME.csv.

1.4.25 screenshot (screenshot generation)

screenshot can generate batch files for producing screenshot of IGV. Generating screenshots can reduce the time for checking the results in genome browser. When the batch files is produced, the user just needs to open IGV, then presses tools on the top tags and choose run batch script. The program will automatically produce screenshots.

- **Required tools**

IGV.

- **Required files**

Gff files that the user wants to produce screenshots: All screenshots will be produced based on the positions of --main_gff. If comparing --main_gff with other features is required, please assign gff files of other features to --side_gffs.

Fasta files of the genomes

Wiggle files of TEX+/- or fragmented/conventional libraries: Please check the section The format of libraries for import to ANNOgesic.

- **Basic arguments**

```
usage: annogesic screenshot --project_path PROJECT_PATH --fasta_file
                            FASTA_FILE --main_gff MAIN_GFF
                            [--side_gffs SIDE_GFFS [SIDE_GFFS ...]]
                            [--tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...]]
                            [--frag_libs FRAG_LIBS [FRAG_LIBS ...]]
                            --output_folder OUTPUT_FOLDER
                            [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --fasta_file FASTA_FILE, -f FASTA_FILE
                        Path of the genome fasta file.
  --main_gff MAIN_GFF, -mg MAIN_GFF
                        Screenshots will be generated based on the positions
                        of genomic features in this gff file.
  --side_gffs SIDE_GFFS [SIDE_GFFS ...], -sg SIDE_GFFS [SIDE_GFFS ...]
                        The gff files of further genomic features (besides
                        --main_gff). For assigning multiple files, please use
                        spaces to separated them.
  --tex_notex_libs TEX_NOTEX_LIBS [TEX_NOTEX_LIBS ...], -tl TEX_NOTEX_LIBS [TEX_NOTEX_
↪LIBS ...]
```

(continues on next page)

(continued from previous page)

```
                        TEX+/- wig files. The format is:
                        wig_file_path:TEX+/-(tex or notex):condition_id(intege
                        r):replicate_id(alphabet):strand(+ or -). If multiple
                        wig files need to be assigned, please use spaces to
                        separate the wig files.
  --frag_libs FRAG_LIBS [FRAG_LIBS ...], -fl FRAG_LIBS [FRAG_LIBS ...]
                        Wig files of RNA-Seq of fragmented transcripts. The
                        format is: wig_file_path:frag:condition_id(integer):re
                        plicate_id(alphabet):strand(+ or -). If multiple wig
                        files need to be assigned, please use spaces to
                        separate the wig files. For example,
                        my_lib_frag_forward.wig:frag:1:a:+
                        my_lib_frag_reverse.wig:frag:1:a:-.
  --output_folder OUTPUT_FOLDER, -o OUTPUT_FOLDER
                        Path of the output folder. A sub-folder "screenshots"
                        in the --output_folder will be created to store the
                        results.
```

- **Additional arguments**

```
additional arguments:
  --height HEIGHT, -he HEIGHT
                        Height of the screenshot. Default is 1500.
  --present {expand,collapse,squish}, -p {expand,collapse,squish}
                        The presentation types (expand, collapse, or squish)
                        of the features in the screenshot. Default is expand.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Based on the paths of `--main_gff`, `screenshot` will generate a folder - `screenshots` under the folder of `--main_gff`. Output files will be stored in this folder. the output files and folders are as following:

forward.txt: Batch file of the forward strand for running on IGV.

reverse.txt: Batch file of reverse strand for running on IGV.

forward: Folder for storing screenshots of the forward strand.

reverse: Folder for storing screenshots of the reverse strand.

When batch files are executed on IGV, the screenshots will be automatically stored in the folder called `forward` and `reverse`. Format of the filenames will be `$GENOME:$START-$END.png`. For an example, `NC_007795:1051529-1051696.png` means the genome is NC_007795, the feature's start point is 1051529 and end point is 1051696.

1.4.26 colorize_screenshot_tracks (colorize the tracks of screenshots)

`colorize_screenshot_tracks` is a following procedure of `screenshot`. If numerous samples are included in one figure, Tracks will be difficult to check. `colorize_screenshot_tracks` can color the tracks based on TEX +/- libraries for improving the checking process.

- **Required tools**

ImageMagick.

- **Required files**

The screenshots folder: Please make sure the folders of `forward` and `reverse` exist in the folder of `screenshots`.

- **Basic arguments**

```
usage: annogesic colorize_screenshot_tracks --project_path PROJECT_PATH
                                            --screenshot_folder
                                            SCREENSHOT_FOLDER --track_number
                                            TRACK_NUMBER
                                            [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
                        Path of the project folder.
  --screenshot_folder SCREENSHOT_FOLDER, -f SCREENSHOT_FOLDER
                        The folder containing "screenshots" which are
                        generated from the subcommand "screenshot".
  --track_number TRACK_NUMBER, -t TRACK_NUMBER
                        The number of tracks.
```

- **Additional arguments**

```
additional arguments:
  --imagemagick_covert_path IMAGEMAGICK_COVERT_PATH, -m IMAGEMAGICK_COVERT_PATH
                        Path of "convert" in the ImageMagick package.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

The new screenshots will replace the previous ones automatically.

1.4.27 merge_features (merge all annotation features)

If merging multiple features of the annotation to one gff file is needed, `merge_features` can achieve this purpose. `merge_features` can merge all features (the user assigned) to one gff file, and search the parent transcript to each feature.

- **Required files**

Gff files of features that the user wants to merge: If transcript gff files can be provided, this module will search the parent transcripts to other input features. Moreover, if the genomic features from different gff files are overlapped, the module can select the data based on user-assigned source or keep all overlapping features to the final output.

- **Basic arguments**

```
usage: annogesic merge_features --project_path PROJECT_PATH
                                --output_prefix OUTPUT_PREFIX
                                [--transcript_file TRANSCRIPT_FILE]
                                [--other_features_files OTHER_FEATURES_FILES [OTHER_
↪FEATURES_FILES ...]]
                                [--source_for_overlapping SOURCE_FOR_OVERLAPPING␣
↪[SOURCE_FOR_OVERLAPPING ...]]
                                [--additional arguments]

basic arguments:
  --project_path PROJECT_PATH, -pj PROJECT_PATH
```

(continues on next page)

(continued from previous page)

```
                        Path of the project folder.
  --output_prefix OUTPUT_PREFIX, -op OUTPUT_PREFIX
                        The prefix name of output gff file. The file name will
                        be $OUTPUT_PREFIX_merge_features.gff.
  --transcript_file TRANSCRIPT_FILE, -a TRANSCRIPT_FILE
                        Path of transcript gff file. The parent transcripts
                        ("Parent" in gff attributes) of all features will be
                        generated.
  --other_features_files OTHER_FEATURES_FILES [OTHER_FEATURES_FILES ...], -of OTHER_
↪FEATURES_FILES [OTHER_FEATURES_FILES ...]
                        Paths of the gff files (besides transcript gff file).
                        For assigning multiple gff files, please use spaces to
                        separate them.
  --source_for_overlapping SOURCE_FOR_OVERLAPPING [SOURCE_FOR_OVERLAPPING ...], -s␣
↪SOURCE_FOR_OVERLAPPING [SOURCE_FOR_OVERLAPPING ...]
                        If the locations of features are overlapped, the
                        module will only keep the feature positions which
                        provided from --source_for_overlapping. For example,
                        if --source_for_overlapping is 'Ref' 'ANNOgesic', the
                        overlapping features from these two sources will be
                        both kept. However, if --source_for_overlapping is
                        'Ref', only the overlapping features from 'RefSeq will
                        be kept.' The value of --source_for_overlapping should
                        be the same as the second column of the gff file. if
                        all the sources of the overlapping features can not be
                        found in --source_for_overlapping, all the overlapping
                        features will be kept. Default is keeping all overlap
                        features.
```

- **Additional arguments**

```
additional arguments:
  --terminator_tolerance TERMINATOR_TOLERANCE, -et TERMINATOR_TOLERANCE
                        The 3'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated terminators. Default is 30.
  --tss_tolerance TSS_TOLERANCE, -tt TSS_TOLERANCE
                        The 5'ends of transcripts will be extended or withdrew
                        by this value (nucleotides) for searching the
                        associated TSSs. Default is 5.

optional arguments:
  -h, --help            show this help message and exit
```

- **Output files**

Output gff files are stored in `$ANNOgesic/output/merge_all_features`. The tag - `Parent` in the attributes of gff file shows the parent transcript.

1.5 Tutorial of ANNOgesic

This tutorial guids you through a small test case to show you how to use ANNOgesic's subcommands. It builds on a differential RNA-Seq data set from *Campylobacter jejuni* subsp. jejuni 81116 which can be downloaded from NCBI GEO and that was part of a work by Dugar et al.. As part of the tutorial several output files will be generated in different formats including CSV files (tabular separated plain text files), that can be opened in LibreOffice or Excel,

GFF3 files, plain text files and figures. For the viewing GFF3 files you can use genome browsers like IGB, IGV or Jbrowse.

Before we start, please check *The format of filename* and *The input format of libraries for running ANNOgesic*. All the parameters and details can be found in the section *ANNOgesic's subcommands*. Moreover, all the requirements are listed in the section *Required tools or databases*.

If a subcommand requires third-party softwares (e.g. TSSpredator in the subcommand `tss_ps`) make sure that path of the executable file is properly specified or is part of the environmental variable `$PATH`. Moreover, if `annogesic` (executable file of ANNOgesic) is not in your `$PATH`, please specify its full path.

1.5.1 Using Singularity (optional)

Using Singularity to build an image of ANNOgesic can install all required tools and environment automatically. If you would like to use Singularity to run ANNOgesic, please check the following commands.

First of all, we need to build up an image of ANNOgesic via Docker image.

```
$ singularity build \
    annogesic.img \
    docker://silasysh/annogesic:latest
```

Now, we can use Singularity to run ANNOgesic. You just need to add the following line before the command that you want to run.

```
singularity exec -B $STORAGE_PATH annogesic.img
```

Please put the storage path of your home directory to `$STORAGE_PATH`. `df` can be used to check the storage system.

For example, if you want to creat the analyzing folder called `ANNOgesic` and your storage path is `storage1`, you can use the following command to run ANNOgesic via Singularity.

```
singularity exec -B /storage1 annogesic.img \
   annogesic create -pj ANNOgesic
```

1.5.2 Run scripts for tutorial

We also provide a shell script which stores all commands that were used in this tutorial. You can run/comment the module by simply add/remove # in front of the module. The run script (file name is run.sh) can be found here. If you are using Singularity, the file name of run script is run_with_singularity.sh which can be found in the same folder.

1.5.3 Generating a project

First of all, we need to create a working folder (`--project_path`) by running `create`.

```
$ annogesic create --project_path ANNOgesic
```

This will create the following folder structure:

```
$ tree ANNOgesic
ANNOgesic
├── input
│   ├── BAMs
│   │   ├── BAMs_map_reference_genomes
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── fragment
│   │   │   └── tex_notex
│   │   └── BAMs_map_related_genomes
│   │       ├── fragment
│   │       └── tex_notex
│   ├── databases
│   ├── manual_processing_sites
│   ├── manual_TSSs
│   ├── mutation_tables
│   ├── reads
│   ├── references
│   │   ├── annotations
│   │   └── fasta_files
│   ├── riboswitch_ID_file
│   ├── RNA_thermometer_ID_file
│   └── wigs
│       ├── fragment
│       └── tex_notex
├── output
└── used_annogesic_version.txt

22 directories, 1 file
```

1.5.4 Retrieving the genome sequences and annotation files

For our test case, the genome and annotation data has to be retrieved NCBI.

ANNOgesic offers a convenient to do that and we download fasta files (`--ref_fasta`), gff files (`--ref_gff`), gbk files (`--ref_gbk`), ptt files (`--ref_ptt`), rnt files (`--ref_rnt`), and convert the the gff files to embl format (`--convert_embl`).

```
$ annogesic get_input_files \
    --ftp_path ftp://ftp.ncbi.nlm.nih.gov/genomes/refseq/bacteria/Campylobacter_
↪jejuni/latest_assembly_versions/GCF_000017905.1_ASM1790v1/ \
    --ref_gff --ref_fasta --ref_gbk --ref_ptt --ref_rnt --convert_embl \
    --project_path ANNOgesic
```

The file will be place in the following locations:

```
$ ls ANNOgesic/input/references/fasta_files/
NC_009839.1.fa
$ ls ANNOgesic/input/references/annotations/
NC_009839.1.embl  NC_009839.1.gbk  NC_009839.1.gff
```

Alternatively you can manually copy the file into these subfolders.

1.5.5 Retrieving wiggle and read files

We need to download reads in SRA format form GEO here.

```
$ wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByStudy/sra/SRP/SRP013/
↪SRP013869/SRR515254/SRR515254.sra
$ wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByStudy/sra/SRP/SRP013/
↪SRP013869/SRR515255/SRR515255.sra
```

(continues on next page)

(continued from previous page)

```
$ wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByStudy/sra/SRP/SRP013/
↪SRP013869/SRR515256/SRR515256.sra
$ wget ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByStudy/sra/SRP/SRP013/
↪SRP013869/SRR515257/SRR515257.sra
```

Then we can convert SRA files to Fasta or Fastq format for mapping by using `fastq-dump` of the the SRA toolkit.

```
$ fastq-dump --fasta SRR515254.sra
$ fastq-dump --fasta SRR515255.sra
$ fastq-dump --fasta SRR515256.sra
$ fastq-dump --fasta SRR515257.sra
$ mv *.fasta ANNOgesic/input/reads
$ rm SRR515254.sra SRR515255.sra SRR515256.sra SRR515257.sra
```

Then we have to download the coverage files in wiggle format.

```
$ wget -cP ANNOgesic/input/wigs/tex_notex ftp://ftp.ncbi.nlm.nih.gov/geo/samples/
↪GSM951nnn/GSM951380/suppl/GSM951380%5FLog%5F81116%5FR1%5Fminus%5FTEX%5Fin%5FNC
↪%5F009839%5Fminus.wig.gz
$ wget -cP ANNOgesic/input/wigs/tex_notex ftp://ftp.ncbi.nlm.nih.gov/geo/samples/
↪GSM951nnn/GSM951380/suppl/GSM951380%5FLog%5F81116%5FR1%5Fminus%5FTEX%5Fin%5FNC
↪%5F009839%5Fplus.wig.gz
$ wget -cP ANNOgesic/input/wigs/tex_notex ftp://ftp.ncbi.nlm.nih.gov/geo/samples/
↪GSM951nnn/GSM951381/suppl/GSM951381%5FLog%5F81116%5FR1%5Fplus%5FTEX%5Fin%5FNC
↪%5F009839%5Fminus.wig.gz
$ wget -cP ANNOgesic/input/wigs/tex_notex ftp://ftp.ncbi.nlm.nih.gov/geo/samples/
↪GSM951nnn/GSM951381/suppl/GSM951381%5FLog%5F81116%5FR1%5Fplus%5FTEX%5Fin%5FNC
↪%5F009839%5Fplus.wig.gz
$ cd ANNOgesic/input/wigs/tex_notex
$ gunzip GSM951380_Log_81116_R1_minus_TEX_in_NC_009839_minus.wig.gz \
         GSM951380_Log_81116_R1_minus_TEX_in_NC_009839_plus.wig.gz \
         GSM951381_Log_81116_R1_plus_TEX_in_NC_009839_minus.wig.gz \
         GSM951381_Log_81116_R1_plus_TEX_in_NC_009839_plus.wig.gz
$ cd ../../../../
```

If we check the wiggle files, we will find that the fasta filename (presented by "chrom") is not the same as fasta or annotation gff file.

```
$ head ANNOgesic/input/wigs/tex_notex/GSM951380_Log_81116_R1_minus_TEX_in_NC_009839_
↪minus.wig
  track type=wiggle_0 name="Log_81116_R1_minus_TEX_in_NC_009839"
  variableStep chrom=NC_009839 span=1
  7  -1.0
  8  -1.0
  9  -1.0
  10 -1.0
  11 -1.0
  12 -1.0
  13 -1.0
  14 -1.0
```

Our genome fasta file is NC_009839.1.fa. Thus "chrom" in wiggle file should be NC_009839.1 not NC_009839. We can use replace_seq_id.py from our Git repository to replace the genome name in wiggle files properly. If the genome names in your fasta, annotation, wiggle files are the same, you don't need to do this step.

```
$ wget https://raw.githubusercontent.com/Sung-Huan/ANNOgesic/master/tutorial_data/
↪replace_seq_id.py
$ python3 replace_seq_id.py -i ANNOgesic/input/wigs/tex_notex -n NC_009839.1
$ rm replace_seq_id.py
```

Since this is a tutorial, we only download one replicate to reduce the running time.

1.5.6 Mapping (optional)

ANNOgesic needs several input files, such as alignment files and wiggle files. In this tutorial, all the required files can be downloaded from public database. However, the users may have their own dataset. In order to generate these required files, we recommand the users to use READemption for mapping their reads. The following example is for mapping the read files of NC_009839.1 that we just retrieved from SRA.

First of all, we need to create an folder for the analysis.

```
$ reademption create READemption
```

Then, we need to move or copy the required input files to the corresponding folders.

```
$ cp ANNOgesic/input/reads/* READemption/input/reads
$ cp ANNOgesic/input/references/fasta_files/*.fa READemption/input/reference_sequences
$ cp ANNOgesic/input/references/annotations/*.gff READemption/input/annotations
```

Now, the mapping can be performed..

```
$ reademption \
    align \
    -r \
    -S \
    -a 95 \
    -l 12 \
    --poly_a_clipping \
    --progress \
    READemption
```

We can assign `-p` for running parallel, such as `-p 20`.

When the alignment is done, the alignemnt files in BAM format can be found in `READemption_analysis/output/align/alignments/`.

Now, we can generate coverage files in wiggle format.

```
$ reademption \
    coverage \
    READemption
```

`-p` can be assigned for `coverage` as well. The raw and normalized coverage files can be found in `READemption_analysis/output/coverage`.

For the details of using READemption, please check READemption

1.5.7 Improving the reference genome

If the fasta file of the target reference genome is not available, `update_genome_fasta` can generate it from a related genome.

Now, we assume that we need to generate fasta file of the reference genome. First of all, we need to find a closely related genome (fasta file and gff file can be found) of our target genome. Then, we need to generate a mutation table (please check the section *ANNOgesic's subcommands*) between these two genomes. When these files are available, we can run subcommand `update_genome_fasta` for getting fasta file of the target genome.

A simple example of the mutation table can be found in mutation table. Each column of the table is separated by tab.

```
$ wget https://raw.githubusercontent.com/Sung-Huan/ANNOgesic/master/tutorial_data/
↪mutation.csv
$ mv mutation.csv ANNOgesic/input/mutation_tables
```

We assume NC_009839.1.fa is a related genome of our target genome – NC_test.1 and test_case2. The fasta files of the new genomes (NC_test.1 and test_case2) will be generated in `ANNOgesic/output/updated_references/fasta_files`.

```
$ annogesic update_genome_fasta \
   --related_fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
   --mutation_table ANNOgesic/input/mutation_tables/mutation.csv \
   --updated_seq_name NC_test.1 \
   --project_path ANNOgesic
```

`--related_fasta_files` is path of the fasta file of closely related genome. When the running process is done, the following information will appear.

```
$ Updating the reference sequences
  Please use the new fasta file to remapping again.
```

Since the data (`ANNOgesic/output/updated_references/fasta_files`) that we generated is only a dummy data, we can ignore the information now. Otherwise, you have to re-map again in order to get the correct alignment and coverage files.

Now we can check the results.

```
$ head ANNOgesic/input/references/fasta_files/NC_009839.1.fa
>NC_009839.1
ATGAATCCAAATCAAATACTTGAAAATTTAAAAAAAGAATTAAGTGAAAACGAATACGAAAATTATATCGCTATCTTAAA
ATTTAACGAAAAACAAAGCAAAGCAGATTTTCTAGTCTTTAACGCTCCTAATGAGCTTTTAGCCAAATTCATACAAACAA
AATACGGTAAAAAAATTTCACATTTTTATGAAGTACAAAGCGGAAATAAAGCGAGCGTTTTGATACAAGCACAAAGTGCT
AAACAAAGTAGCAAAAGCACTAAAATCGATATCGCTCATATCAAGGCGCAAAGTACGATTTTAAATCCTTCTTTTACTTT
TGAAAGCTTTGTAGTGGGGGATTCTAACAAATACGCTTATGGAGCTTGTAAAGCTATCTCACAAAAAGACAAACTGGGAA
AACTTTATAATCCTATCTTTATCTATGGGCCTACAGGGCTTGGAAAAACGCACTTGCTTCAAGCTGTGGGAAATGCAAGT
TTGGAAATGGGAAAAAAAGTGATTTATGCTACGAGTGAAAATTTTATCAATGATTTTACTTCAAATTTAAAAAATGGCTC
TTTAGATAAATTTCACGAAAAATATAGAAATTGTGATGTTTTACTCATAGATGATGTGCAGTTTTTAGGAAAAACCGATA
AAATTCAAGAAGAATTTTTCTTTATATTTAATGAAATCAAAAATAACGATGGACAAATCATCATGACTTCAGACAATCCA
$ head ANNOgesic/output/updated_references/fasta_files/NC_test.1.fa
>NC_test.1
ATcAACCAAATCAAATACTTGAAAATTTAAAAAAAGAATTAAGTGAAAACGAATACGAAA
ATTATATCGCTATCTTAAAATTTAACGAAAAACAAAGCAAAGCAGATTTTCTAGTCTTTA
ACGCTCCTAATGAGCTTTTAGCCAAATTCATACAAACAAAATACGGTAAAAAAATTTCAC
ATTTTTATGAAGTACAAAGCGGAAATAAAGCGAGCGTTTTGATACAAGCACAAAGTGCTA
AACAAAGTAGCAAAAGCACTAAAATCGATATCGCTCATATCAAGGCGCAAAGTACGATTT
TAAATCCTTCTTTTACTTTTGAAAGCTTTGTAGTGGGGGATTCTAACAAATACGCTTATG
GAGCTTGTAAAGCTATCTCACAAAAAGACAAACTGGGAAAACTTTATAATCCTATCTTTA
TCTATGGGCCTACAGGGCTTGGAAAAACGCACTTGCTTCAAGCTGTGGGAAATGCAAGTT
TGGAAATGGGAAAAAAAGTGATTTATGCTACGAGTGAAAATTTTATCAATGATTTTACTT
```

We can see the third nucleotide of `NC_test.1.fa` is replaced from G to c. Moreover, The sixth nucleotide T is deleted.

If the mutation table can not be provided, we can also use subcommand `snp` to detect mutations and generate fasta files automatically. For `snp`, we will go through it later.

1.5.8 Generating annotation files

If the genome annotations of the target reference genome in GFF format is not available, `annotation_transfer` can generate it from a related genome.

Like the last section, we assume NC_009839.1.fa is a related genome of our target genome – NC_test.1 and test_case2.

Before we running this subcommand, we have to modify environment paths of RATT. If you run ANNOgesic in docker container, the path is already set. Otherwise, please check RATT to set your environment paths properly. Moreover, if the error message related to 'defined(@array)' occurs, please check *FAQ*.

After setting the environment, we can try it.

```
annogesic annotation_transfer \
    --related_embl_files ANNOgesic/input/references/annotations/NC_009839.1.embl \
    --related_fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --target_fasta_files ANNOgesic/output/updated_references/fasta_files/NC_test.1.fa␣
↪\
    --element chromosome \
    --transfer_type Strain \
    --compare_pair NC_009839.1:NC_test.1 \
    --project_path ANNOgesic
```

`--element` is prefix name of the output embl files. `--transfer_type` is a program of RATT. We use `Strain` because the similarity between two genomes is higher than 90% (please check RATT). In `--compare_pair`, the pairs of the genomes (NC_test.1 and test_case2) and their closely related genomes (NC_000915.1) are assigned. The annotation information in embl, GFF3, ptt, and rnt format will be stored in `ANNOgesic/output/updated_references/annotations`.

Once the transfer is done, we can see

```
$ ls ANNOgesic/output/updated_references/annotations/
NC_test.1.gff  NC_test.1.ptt  NC_test.1.rnt
$ ls ANNOgesic/output/annotation_transfer/
chromosome.NC_test.1.final.embl  log.txt  NC_test.1.gff  ratt_log.txt
```

In `ANNOgesic/output/updated_references/annotations`, we can find ptt, rnt and gff files. In `ANNOgesic/output/annotation_transfer`, we can find the output of RATT.

We already saw how to update genome fasta and annotation files. For the following subcommands, we will use `ANNOgesic/input/references/annotations/NC_009839.1.gff` and `ANNOgesic/input/references/fasta_files/NC_009839.1` as the reference genome.

1.5.9 TSS and processing site prediction and optimization

Before running following subcommands, we need to setup our libraries in a correct format. First, we set the paths of wig files.

```
WIG_FOLDER="ANNOgesic/input/wigs/tex_notex"
```

Then, we can setup our libraries – `$WIGFILE:$TEXorNOTEXorFRAG:CONDITION:REPLICATE:STRAND` (*The input format of libraries for running ANNOgesic*).

```
TEX_LIBS="$WIG_FOLDER/GSM951380_Log_81116_R1_minus_TEX_in_NC_009839_minus.
↪wig:notex:1:a:- \
          $WIG_FOLDER/GSM951381_Log_81116_R1_plus_TEX_in_NC_009839_minus.wig:tex:1:a:-
↪ \
          $WIG_FOLDER/GSM951380_Log_81116_R1_minus_TEX_in_NC_009839_plus.
↪wig:notex:1:a:+ \
          $WIG_FOLDER/GSM951381_Log_81116_R1_plus_TEX_in_NC_009839_plus.wig:tex:1:a:+"
```

Now, we can start to test other subcommands.

Before running `tss_ps`, we can use `optimize_tss_ps` to optimize the parameters (It is an optional step, but we highly recommand you to do it). The optimization requires a small set of the manual-detected TSSs in GFF3 format. In our experience, we recommend you to detect at least 50 TSSs and check more than 200kb of the genome.

For this test case, you can download the manual TSS file from our git repository.

```
$ wget -cP ANNOgesic/input/manual_TSSs/ https://raw.githubusercontent.com/Sung-Huan/
↪ANNOgesic/master/tutorial_data/NC_009839_manual_TSS.gff
```

Now, we have a manual-detected TSS gff file which is stored in `ANNOgesic/input/manual_TSSs`. we can try `optimize_tss_ps` right now (since we only check first 200kb, we set `--genome_lengths` as "NC_009839.1:200000" which means only first 200kb of NC_009839.1 is valid.).

```
$ annogesic optimize_tss_ps \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tex_notex_libs $TEX_LIBS \
    --condition_names TSS --steps 25 \
    --manual_files ANNOgesic/input/manual_TSSs/NC_009839_manual_TSS.gff \
    --curated_sequence_length NC_009839.1:200000 \
    --replicate_tex all_1 \
    --project_path ANNOgesic
```

`optimize_tss_ps` will compare manual-checked TSSs with predicted TSSs to search the optimized parameters. Results of the different parameters will be printed in the screen, and stored in `stat_NC_009839.1.csv` as well. We only set 25 runs for testing. For optimization of processing sites, we just need to change `--program` from TSS to PS. `--replicate_tex` means the minimum replicates that a TSS can be detected. `all_1` means that a TSS should be detected in at least one replicate in all conditions. `--replicate_tex` can be also assigned like `all_2` (a TSS should be detected in at least two replicates in all conditions) or `1_2 2_2 3_3` (in condition 1 and 2 (based on the setting of `--tex_notex_libs`, a TSS should be detected in at least two replicates, and a TSS should be predicted in three replicates in condition 3). Once the optimization is done, you can find several files.

```
$ ls ANNOgesic/output/TSSs/optimized_TSSpredator/
best_NC_009839.1.csv  log.txt  results_all_steps.txt   stat_NC_009839.1.csv
```

`best_NC_009839.1.csv` is for the results of the optimized parameters; `stat_NC_009839.1.csv` is for the results of each step.

Now, we assume the optimized parameters are following: height is 0.4, height_reduction is 0.1, factor is 1.7, factor_reduction is 0.2, base_height is 0.039, enrichment_factor is 1.1, processing_factor is 4.5. We can set these parameters for running `tss`.

```
$ annogesic tss_ps \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tex_notex_libs $TEX_LIBS \
    --condition_names test \
```

(continues on next page)

(continued from previous page)

```
    --height 0.4 \
    --height_reduction 0.1 \
    --factor 1.7 \
    --factor_reduction 0.2 \
    --base_height 0.039 \
    --enrichment_factor 1.1 \
    --processing_factor 4.5 \
    --validate_gene \
    --program TSS \
    --replicate_tex all_1 \
    --curated_sequence_length NC_009839.1:200000 \
    --manual_files ANNOgesic/input/manual_TSSs/NC_009839_manual_TSS.gff \
    --project_path ANNOgesic
```

We assigned the manual-checked TSS gff file to `--manual_files`. Therefore, the output gff file will contain the manual-detected TSSs and predicted TSSs. If we didn't assign it, Only the predicted TSSs will be included in output gff file.

A another way to run `tss_ps` with optimized parameters is using `--auto_load_optimized_parameters`. If `optimize_tss_ps` has been done before, assigning the final output folder of `optimize_tss_ps` to `--auto_load_optimized_parameters` will directly use the optimized parameters for the prediction.

```
$ annogesic tss_ps \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tex_notex_libs $TEX_LIBS \
    --condition_names test \
    --auto_load_optimized_parameters ANNOgesic/output/TSSs/optimized_TSSpredator \
    --validate_gene \
    --program TSS \
    --replicate_tex all_1 \
    --curated_sequence_length NC_009839.1:200000 \
    --manual_files ANNOgesic/input/manual_TSSs/NC_009839_manual_TSS.gff \
    --project_path ANNOgesic
```

If you did not run `optimize_tss_ps` before and just want to do TSS prediction with default parameters, `--height`, `--height_reduction`, `--factor`, `--factor_reduction`, `--base_height`, and `--enrichment_factor` do not need to be assigned. For assigning different parameters to multiple genomes, please check *tss_ps (TSS and processing site prediction)* of *ANNOgesic's subcommands* for the details (`--genome_order`).

```
$ annogesic tss_ps \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tex_notex_libs $TEX_LIBS \
    --condition_names test \
    --validate_gene \
    --program TSS \
    --replicate_tex all_1 \
    --curated_sequence_length NC_009839.1:200000 \
    --manual_files ANNOgesic/input/manual_TSSs/NC_009839_manual_TSS.gff \
    --project_path ANNOgesic
```

The output files are gff file, MasterTable and statistic files.

```
$ ls ANNOgesic/output/TSSs/
configs  gffs  MasterTables  log.txt  optimized_TSSpredator  screenshots  statistics
$ ls ANNOgesic/output/TSSs/configs/
config_NC_009839.1.ini
$ ls ANNOgesic/output/TSSs/gffs/
NC_009839.1_TSS.gff
$ ls ANNOgesic/output/TSSs/MasterTables/MasterTable_NC_009839.1/
AlignmentStatistics.tsv  err.txt  log.txt  MasterTable.tsv  superConsensus.fa
↪superTSS.gff  superTSStracks.gff  test_super.fa  test_super.gff  test_TSS.gff
$ ls ANNOgesic/output/TSSs/statistics/NC_009839.1/
stat_compare_TSSpredator_manual_NC_009839.1.csv  stat_TSS_class_NC_009839.1.csv  TSS_
↪class_NC_009839.1.png  TSS_venn_NC_009839.1.png
stat_gene_vali_NC_009839.1.csv                   stat_TSS_libs_NC_009839.1.csv
↪TSSstatistics.tsv
```

If we want to predict processing sites, the procedures are the same. We just need to change the program from TSS to PS (`--program`) and assign the proper parameter sets. We assume the best parameter sets are following: height is 0.2, height_reduction is 0.1, factor is 2.0, factor_reduction is 0.5, base_height is 0.009, enrichment_factor is 1.2, processing_factor is 1.5.

```
$ annogesic tss_ps \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tex_notex_libs $TEX_LIBS \
    --condition_names test \
    --height 0.2 \
    --height_reduction 0.1 \
    --factor 2.0 \
    --factor_reduction 0.5 \
    --base_height 0.009 \
    --enrichment_factor 1.2 \
    --processing_factor 1.5 \
    --replicate_tex all_1 \
    --program PS \
    --project_path ANNOgesic
```

The output files are following:

```
$ ls ANNOgesic/output/processing_sites/
configs  gffs  MasterTables  log.txt  statistics
$ ls ANNOgesic/output/processing_sites/configs/
config_NC_009839.1.ini
$ ls ANNOgesic/output/processing_sites/gffs/
NC_009839.1_processing.gff
$ ls ANNOgesic/output/processing_sites/MasterTables/MasterTable_NC_009839.1/
AlignmentStatistics.tsv  err.txt  log.txt  MasterTable.tsv  superConsensus.fa
↪superTSS.gff  superTSStracks.gff  test_super.fa  test_super.gff  test_TSS.gff
$ ls ANNOgesic/output/processing_sites/statistics/NC_009839.1/
processing_class_NC_009839.1.png  processing_venn_NC_009839.1.png  stat_processing_
↪class_NC_009839.1.csv  stat_processing_libs_NC_009839.1.csv  TSSstatistics.tsv
```

Since we use TSSpredator to detect processing site, the files in `ANNOgesic/output/processing_sites/MasterTables/MasterTable_NC_009839.1/` are for processing site not for TSS.

1.5.10 Performing transcript detection

Transcript detection is a basic procedure for detecting transcript boundary. we can use subcommand `transcript` to detect the transcript. Normally, we strongly recommend that the user should provide the libraries of RNA-Seq with transcript fragmented (`--frag_libs`) because dRNA-Seq focus on 5'end and usually loses some information of 3'end. However, we only use TEX +/- for testing since we have no fragmented libraries.

There are several options for modifying transcripts by comparing transcripts and genome annotations like CDSs (`--modify_transcript`). By assigning `--modify_transcript`, transcripts can be merged or extended based on the genome annotations. If you want to know the details, please check *transcript (transcript detection)*. Now, we use default setting to run this module:

```
$ annogesic transcript \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tex_notex_libs $TEX_LIBS \
    --replicate_tex all_1 \
    --compare_feature_genome gene CDS \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --project_path ANNOgesic
```

The output files are gff files, tables and statistic files.

```
$ ls ANNOgesic/output/transcripts/gffs
NC_009839.1_transcript.gff
$ ls ANNOgesic/output/transcripts/tables
NC_009839.1_transcript.csv
$ ls ANNOgesic/output/transcripts/statistics
NC_009839.1_length_all.png  NC_009839.1_length_less_2000.png  stat_compare_transcript_
↪TSS_NC_009839.1.csv  stat_compare_transcript_genome_NC_009839.1.csv
```

1.5.11 Prediction of terminator

We can run subcommand `terminator` to detect terminators. The command is like following:

```
$ annogesic terminator \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --tex_notex_libs $TEX_LIBS \
    --replicate_tex all_1 \
    --project_path ANNOgesic
```

Four different kinds of gff files and tables will be generated.

```
$ ls ANNOgesic/output/terminators/gffs/
all_candidates  best_candidates  expressed_candidates  non_expressed_candidates
$ ls ANNOgesic/output/terminators/tables
all_candidates  best_candidates  expressed_candidates  non_expressed_candidates
```

`all_candidates` is for all candidates; `expressed_candidates` is for the candidates which reveal gene expression; `best_candidates` is for the candidates which reveal gene expression and their coverages show significant decreasing; `non_expressed_candidates` is for the candidates which have no expression.

```
$ ls ANNOgesic/output/terminators/gffs/best_candidates
NC_009839.1_term.gff
$ ls ANNOgesic/output/terminators/gffs/expressed_candidates
```

(continues on next page)

(continued from previous page)

```
NC_009839.1_term.gff
$ ls ANNOgesic/output/terminators/gffs/all_candidates
NC_009839.1_term.gff
$ ls ANNOgesic/output/terminators/gffs/non_expressed_candidates
NC_009839.1_term.gff
$ ls ANNOgesic/output/terminators/tables/best_candidates
NC_009839.1_term.csv
$ ls ANNOgesic/output/terminators/tables/expressed_candidates
NC_009839.1_term.csv
$ ls ANNOgesic/output/terminators/tables/all_candidates
NC_009839.1_term.csv
$ ls ANNOgesic/output/terminators/tables/non_expressed_candidates
NC_009839.1_term.csv
```

In transtermhp folder, output files of TranstermHP can be found.

```
$ ls ANNOgesic/output/terminators/transtermhp_results/NC_009839.1
NC_009839.1_best_terminator_after_gene.bag  NC_009839.1_terminators.txt  NC_009839.1_
↪terminators_within_robust_tail-to-tail_regions.t2t
```

Moreover, statistic files are stored in `statistics`.

```
$ ls ANNOgesic/output/terminators/statistics/
stat_compare_terminator_transcript_NC_009839.1_all_candidates.csv   stat_compare_
↪terminator_transcript_NC_009839.1_expressed_candidates.csv
stat_compare_terminator_transcript_NC_009839.1_best_candidates.csv  stat_NC_009839.1.
↪csv
```

1.5.12 Generating UTR

Now, we have the information of TSSs, transcripts and terminators. We can detect the 5'UTRs and 3'UTRs by using subcommand `utr`.

```
$ annogesic utr \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --terminator_files ANNOgesic/output/terminators/gffs/best_candidates/NC_009839.1_
↪term.gff \
    --project_path ANNOgesic
```

If the TSS gff file is not generated by ANNOgesic, please add `--tss_source` which can classify TSSs for generating UTRs. Output gff files and statistic files will be stored in `ANNOgesic/output/UTRs/5UTRs` and `ANNOgesic/output/UTRs/3UTRs`.

```
$ ls ANNOgesic/output/UTRs/3UTRs
gffs/       statistics/
$ ls ANNOgesic/output/UTRs/5UTRs
gffs/       statistics/
$ ls ANNOgesic/output/UTRs/3UTRs/gffs
NC_009839.1_3UTR.gff
$ ls ANNOgesic/output/UTRs/5UTRs/gffs
NC_009839.1_5UTR.gff
$ ls ANNOgesic/output/UTRs/5UTRs/statistics
```

(continues on next page)

(continued from previous page)

```
NC_009839.1_all_5utr_length.png
$ ls ANNOgesic/output/UTRs/3UTRs/statistics
NC_009839.1_all_3utr_length.png
```

Now, we have all information for defining the transcript boundary.

1.5.13 Detecting operon and suboperon

We have TSSs, transcripts, terminators, CDSs, UTRs now. We can integrate all these feature to detect operons and suboperons by executing subcommand `operon`.

```
$ annogesic operon \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --terminator_files ANNOgesic/output/terminators/gffs/best_candidates/NC_009839.1_
↪term.gff \
    --project_path ANNOgesic
```

Three folders will be generated to store table and statistics files.

```
$ ls ANNOgesic/output/operons/
gffs  log.txt  statistics  tables
$ ls ANNOgesic/output/operons/gffs/
NC_009839.1_operon.gff
$ ls ANNOgesic/output/operons/tables/
NC_009839.1_operon.csv
$ ls ANNOgesic/output/operons/statistics/
stat_NC_009839.1_operon.csv
```

1.5.14 Promoter motif detection

As long as we have TSSs, we can use subcommand `promoter` to identify promoters. If the TSS gff files are not generated by `ANNOgesic`, please add `--tss_source` and corresponding genome annotation file containing CDSs, tRNAs, rRNAs, etc, (`--annotation_files`) in order to classify TSSs for detecting promoters. Now, let's try `promoter` (`--program` is assigned by "both" in default. If you want to only run MEME or GLAM2, please assign "meme" or "glam2" to `--program`), the process may take a while.

```
$ annogesic promoter \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --motif_width 45 2-10 \
    --project_path ANNOgesic
```

We defined the length of the motifs as `50` and `2-10`. `2-10` means the width can be from 2 to 10.

If the software for running MEME and GLAM2 in parallels is installed, `--parallels` can also be assigned for running MEME and GLAM2 in parallels.

```
$ annogesic promoter \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --motif_width 45 2-10 \
```

(continues on next page)

(continued from previous page)

```
    --parallels 10 \
    --project_path ANNOgesic
```

Based on different types of the TSSs and the length of the motif, numerous output files will be generated.

```
$ ls ANNOgesic/output/promoters/
fasta_classes  NC_009839.1   log.txt
$ ls ANNOgesic/output/promoters/fasta_classes/NC_009839.1
NC_009839.1_allgenome_all_types.fa  NC_009839.1_allgenome_internal.fa  NC_009839.1_
↪allgenome_primary.fa    NC_009839.1_allgenome_without_orphan.fa
NC_009839.1_allgenome_antisense.fa  NC_009839.1_allgenome_orphan.fa    NC_009839.1_
↪allgenome_secondary.fa
$ ls ANNOgesic/output/promoters/NC_009839.1
MEME GLAM2
$ ls ANNOgesic/output/promoters/NC_009839.1/MEME
promoter_motifs_NC_009839.1_allgenome_all_types_2-10_nt  promoter_motifs_NC_009839.1_
↪allgenome_all_types_45_nt
$ ls ANNOgesic/output/promoters/NC_009839.1/GLAM2
promoter_motifs_NC_009839.1_allgenome_all_types_2-10_nt  promoter_motifs_NC_009839.1_
↪allgenome_all_types_45_nt
$ ls ANNOgesic/output/promoters/NC_009839.1/MEME/promoter_motifs_NC_009839.1_
↪allgenome_all_types_45_nt/
logo1.eps  logo1.png  logo2.eps  logo2.png  logo3.eps  logo3.png  logo_rc1.eps  logo_
↪rc1.png  logo_rc2.eps  logo_rc2.png  logo_rc3.eps  logo_rc3.png  meme.csv  meme.
↪html  meme.txt  meme.xml
$ ls ANNOgesic/output/promoters/NC_009839.1/GLAM2/promoter_motifs_NC_009839.1_
↪allgenome_all_types_45_nt/
glam2.csv   glam2.txt   logo1.eps  logo2.png  logo4.eps  logo5.png  logo7.eps  logo8.
↪png  logo_ssc10.eps  logo_ssc1.png  logo_ssc3.eps  logo_ssc4.png  logo_ssc6.eps ␣
↪logo_ssc7.png  logo_ssc9.eps
glam2.html  logo10.eps  logo1.png  logo3.eps  logo4.png  logo6.eps  logo7.png  logo9.
↪eps  logo_ssc10.png  logo_ssc2.eps  logo_ssc3.png  logo_ssc5.eps  logo_ssc6.png ␣
↪logo_ssc8.eps  logo_ssc9.png
glam2.meme  logo10.png  logo2.eps  logo3.png  logo5.eps  logo6.png  logo8.eps  logo9.
↪png  logo_ssc1.eps   logo_ssc2.png  logo_ssc4.eps  logo_ssc5.png  logo_ssc7.eps ␣
↪logo_ssc8.png
```

If you want to detect promoters based on a specific type of TSSs, `fasta_classes` stores different types of TSSs. You can use these fasta files to run promoter detection again.

1.5.15 Prediction of sRNA and sORF

Based on transcripts, genome annotations and coverage information, sRNAs can be detected. Moreover, we have TSSs and processing sites which can be used for detecting UTR-derived sRNAs as well. Now, we can get sRNAs by running subcommand `srna`. Normally, we recommend that the user uses the libraries of RNA-Seq with transcript fragmented as well. Here, we only use TEX +/- for testing.

For running `srna`, we can apply several filters to improve the detection. These filters are `tss`, `sec_str`, `blast_nr`, `blast_srna`, `promoter`, `term`, `sorf`. Normally, `tss`, `sec_str`, `blast_nr`, `blast_srna` are recommended to be used.

Please be aware, filters are strict. For examples, if your filters include `term`, only the sRNAs which are associated with terminators will be included in the list of best candidates. If you want to include terminator information but do not use terminator as a filter, you can remove `term` in filters and still assign the path of terminator gff file. The results will include the sRNAs which are not associated with terminators, and terminator information can be shown and checked in the results as well. Many parameters can be used for adjustment the prediction, such as `blast_e_srna`,

blast_e_nr, blast_score_nr, blast_score_srna, etc. For details of the filters, please check the section *srna (sRNA detection)* in ANNOgesic subcommand.

Before running srna, we have to download sRNA database (we can use BSRD) and nr database (if you have not downloaded before). We can download fasta file of BSRD from our Git repository.

```
$ wget -cP ANNOgesic/input/databases/ https://raw.githubusercontent.com/Sung-Huan/
↪ANNOgesic/master/database/sRNA_database_BSRD.fa
```

If you have your sRNA database in other folders, please assign your path of databases to --srna_database_path (please check *srna (sRNA detection)* to modify the headers of your database). If your database was formatted before, you can remove --srna_format. In order to use the recommended filters, we have to download nr database (takes a while). If you already downloaded it, you can skip this step.

```
$ wget -cP ANNOgesic/input/databases/ ftp://ftp.ncbi.nih.gov/blast/db/FASTA/nr.gz
$ gunzip ANNOgesic/input/databases/nr.gz
$ mv ANNOgesic/input/databases/nr ANNOgesic/input/databases/nr.fa
```

If your nr database is in other folders, please assign your path to --nr_database_path. You can also remove --nr_format if your database is already formatted. Now, we can use the recommended filters to run srna, but it may takes a while.

```
$ annogesic srna \
    --filter_info tss blast_srna sec_str blast_nr \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --processing_site_files ANNOgesic/output/processing_sites/gffs/NC_009839.1_
↪processing.gff \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --terminator_files ANNOgesic/output/terminators/gffs/best_candidates/NC_009839.1_
↪term.gff \
    --promoter_tables ANNOgesic/output/promoters/NC_009839.1/MEME/promoter_motifs_NC_
↪009839.1_allgenome_all_types_45_nt/meme.csv \
    --promoter_names MOTIF_1 \
    --mountain_plot \
    --utr_derived_srna \
    --compute_sec_structures \
    --srna_format \
    --nr_format \
    --nr_database_path ANNOgesic/input/databases/nr \
    --srna_database_path ANNOgesic/input/databases/sRNA_database_BSRD \
    --tex_notex_libs $TEX_LIBS \
    --replicate_tex all_1 \
    --project_path ANNOgesic
```

If you have sORF information, you can also assign the path of sORF gff file to --sorf_files. Then, the comparison between sRNAs and sORFs can be executed.

Output files are following.

```
$ ls ANNOgesic/output/sRNAs/
blast_results_and_misc  figs  gffs  log.txt  sRNA_2d_NC_009839.1  sRNA_seq_NC_009839.
↪1  statistics  tables
```

blast_results_and_misc stores the results of blast; figs stores plots of sRNAs; statistics stores statistic files.

sRNA_2d_NC_009839.1 and sRNA_seq_NC_009839.1 are text files of sRNA sequences and secondary structures.

```
$ ls ANNOgesic/output/sRNAs/blast_results_and_misc/
nr_blast_NC_009839.1.txt  sRNA_blast_NC_009839.1.txt
$ ls ANNOgesic/output/sRNAs/figs/
dot_plots  mountain_plots  sec_plots
$ ls ANNOgesic/output/sRNAs/figs/mountain_plots/NC_009839.1/
srna0_NC_009839.1_36954_37044_-_mountain.pdf     srna25_NC_009839.1_854600_854673_-_
↪mountain.pdf    srna40_NC_009839.1_1091155_1091251_-_mountain.pdf  srna56_NC_009839.
↪1_1440826_1441414_+_mountain.pdf
srna10_NC_009839.1_248098_248257_-_mountain.pdf  srna26_NC_009839.1_879881_880088_-_
↪mountain.pdf    srna41_NC_009839.1_1097654_1097750_-_mountain.pdf  srna57_NC_009839.
↪1_1448211_1448306_+_mountain.pdf
...

$ ls ANNOgesic/output/sRNAs/figs/dot_plots/NC_009839.1/
srna0_NC_009839.1_36954_37044_-_dp.ps     srna25_NC_009839.1_854600_854673_-_dp.ps
↪srna40_NC_009839.1_1091155_1091251_-_dp.ps  srna56_NC_009839.1_1440826_1441414_+_dp.
↪ps
srna10_NC_009839.1_248098_248257_-_dp.ps  srna26_NC_009839.1_879881_880088_-_dp.ps
↪srna41_NC_009839.1_1097654_1097750_-_dp.ps  srna57_NC_009839.1_1448211_1448306_+_dp.
↪ps
...

$ ls ANNOgesic/output/sRNAs/figs/sec_plots/NC_009839.1/
rna0_NC_009839.1_36954_37044_-_rss.ps     srna25_NC_009839.1_854600_854673_-_rss.ps
↪ srna40_NC_009839.1_1091155_1091251_-_rss.ps  srna56_NC_009839.1_1440826_1441414_+_
↪rss.ps
srna10_NC_009839.1_248098_248257_-_rss.ps  srna26_NC_009839.1_879881_880088_-_rss.ps
↪  srna41_NC_009839.1_1097654_1097750_-_rss.ps  srna57_NC_009839.1_1448211_1448306_+_
↪rss.ps
...

$ ls ANNOgesic/output/sRNAs/statistics/
stat_NC_009839.1_sRNA_blast.csv  stat_sRNA_class_NC_009839.1.csv
```

In gffs and tables, three different folders are generated. all_candidates is for all candidates without filtering; best_candidates is for the candidates after filtering; for_classes is for different sRNA types based on stat_sRNA_class_NC_009839.1.csv.

```
$ ls ANNOgesic/output/sRNAs/gffs/
all_candidates  best_candidates  for_classes
$ ls ANNOgesic/output/sRNAs/tables/
all_candidates  best_candidates  for_classes
$ ls ANNOgesic/output/sRNAs/gffs/all_candidates/
NC_009839.1_sRNA.gff
$ ls ANNOgesic/output/sRNAs/tables/all_candidates/
NC_009839.1_sRNA.csv
$ ls ANNOgesic/output/sRNAs/gffs/best_candidates/
NC_009839.1_sRNA.gff
$ ls ANNOgesic/output/sRNAs/tables/best_candidates/
NC_009839.1_sRNA.csv
$ ls ANNOgesic/output/sRNAs/gffs/for_classes/NC_009839.1/
class_1_all.gff                                  class_1_class_2_class_7_all.
↪gff                 class_2_all.gff                                  class_3_all.
↪gff
class_1_class_2_all.gff                          class_1_class_3_all.gff
↪                    class_2_class_3_all.gff                          class_3_class
↪4_all.gff
```

(continues on next page)

(continued from previous page)

```
...

$ ls ANNOgesic/output/sRNAs/tables/for_classes/NC_009839.1/
class_1_all.csv                                    class_1_class_2_class_7_all.
↪csv                  class_2_all.csv                                  class_3_all.
↪csv
class_1_class_2_all.csv                            class_1_class_3_all.csv
↪                    class_2_class_3_all.csv                          class_3_class_
↪4_all.csv
...
```

If the amount of sRNA candidates are too many for the user, please check the FAQ Q9 to do the further filtering. As we know, expressed regions without annotations may be sORFs as well. In order to get information of sORFs, we can use subcommand `sorf`.

```
$ annogesic sorf \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --srna_files ANNOgesic/output/sRNAs/gffs/best_candidates/NC_009839.1_sRNA.gff \
    --tex_notex_libs $TEX_LIBS \
    --replicate_tex all_1 -u \
    --project_path ANNOgesic
```

For generating best candidates, some filters can be assigned (ex: with ribosome binding site (Shine-Dalgarno sequence), with TSS, without overlap with sRNA, etc.). After running `sorf`, gff files, statistic files and tables of the sORF will be generated. `all_candidates` stores the gff files and tables without filtering; `best_candidates` stores the gff_files and tables with filtering.

```
$ ls ANNOgesic/output/sORFs/gffs/all_candidates/
NC_009839.1_sORF.gff
$ ls ANNOgesic/output/sORFs/gffs/best_candidates/
NC_009839.1_sORF.gff
$ ls ANNOgesic/output/sORFs/tables/all_candidates/
NC_009839.1_sORF.csv
$ ls ANNOgesic/output/sORFs/tables/best_candidates/
NC_009839.1_sORF.csv
$ ls ANNOgesic/output/sORFs/statistics/
stat_NC_009839.1_sORF.csv
```

1.5.16 Performing sRNA target prediction

Now we have sRNA candidates. If we want to know targets of these sRNAs, we can use `srna_target`.

```
$annogesic srna_target \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --srna_files ANNOgesic/output/sRNAs/gffs/best_candidates/NC_009839.1_sRNA.gff \
    --query_srnas NC_009839.1:36954:37044:- \
    --mode_intarna H \
    --program RNAup IntaRNA RNAplex \
    --project_path ANNOgesic
```

For testing, we only assign one sRNA to do the prediction. You can also assign several sRNAs like `NC_009839.1:36954:37044:- NC_009839.1:75845:75990:+`. If you want to compute all sRNAs, you can assign `all` to `--query_srnas` (may take several days).

Several output folders will be generated.

```
$ ls ANNOgesic/output/sRNA_targets/
IntaRNA_results  merged_results  log.txt  RNAplex_results  RNAup_results  sRNA_seqs
↪target_seqs
```

`sRNA_seqs` and `target_seqs` are for sequences of the sRNAs and the potential targets.

```
$ ls ANNOgesic/output/sRNA_targets/sRNA_seqs
NC_009839.1_sRNA.fa
$ ls ANNOgesic/output/sRNA_targets/target_seqs
NC_009839.1_target.fa
```

`IntaRNA_results`, `RNAplex_results` and `RNAup_results` are for the output of IntaRNA, RNAplex and RNAup.

```
$ ls ANNOgesic/output/sRNA_targets/RNAplex_results/NC_009839.1/
NC_009839.1_RNAplex_rank.csv  NC_009839.1_RNAplex.txt
$ ls ANNOgesic/output/sRNA_targets/RNAup_results/NC_009839.1/
NC_009839.1_RNAup.log  NC_009839.1_RNAup_rank.csv  NC_009839.1_RNAup.txt
$ ls ANNOgesic/output/sRNA_targets/IntaRNA_results/NC_009839.1/
NC_009839.1_IntaRNA_rank.csv  NC_009839.1_IntaRNA.txt
```

`merged_results` is for the merged results of IntaRNA, RNAplex and RNAup. `NC_009839.1_merge.csv` contains all results of the assigned methods, and `NC_009839.1_overlap.csv` only stores candidates which are top 50 (default) in the assigned methods.

```
$ ls ANNOgesic/output/sRNA_targets/merged_results/NC_009839.1/
NC_009839.1_merge.csv  NC_009839.1_overlap.csv
```

1.5.17 Mapping and detecting of circular RNA

You may also be interested in circular RNAs. The subcommand `circrna` can help us to predict circular RNAs by using Segemehl. Since we didn't map reads before, we can also do mapping by running `circrna`. If you already mapped the reads by Segemehl with `--splits`, you can add path of the bam files to `--bam_files` directly. However, if you mapped the reads by other tools or you mapped the reads by Segemehl without `--splits`, Unfortunately, you have to re-map the reads(`--read_files`) again. You can assign the number of parallels (`--parallels`) for mapping.

Since we just want to test the subcommand. Thus, we can reduce the running time by selecting the subset of the reads (first 50000) for only testing.

```
$ head -n 50000 ANNOgesic/input/reads/SRR515254.fasta > ANNOgesic/input/reads/
↪SRR515254_50000.fasta
$ head -n 50000 ANNOgesic/input/reads/SRR515255.fasta > ANNOgesic/input/reads/
↪SRR515255_50000.fasta
$ head -n 50000 ANNOgesic/input/reads/SRR515256.fasta > ANNOgesic/input/reads/
↪SRR515256_50000.fasta
$ head -n 50000 ANNOgesic/input/reads/SRR515257.fasta > ANNOgesic/input/reads/
↪SRR515257_50000.fasta
$ rm ANNOgesic/input/reads/SRR515254.fasta
$ rm ANNOgesic/input/reads/SRR515255.fasta
```

(continues on next page)

(continued from previous page)

```
$ rm ANNOgesic/input/reads/SRR515256.fasta
$ rm ANNOgesic/input/reads/SRR515257.fasta
```

Then we setup the read files.

:: $READ_FILES=ANNOgesic/input/reads/SRR515254_50000.fasta,ANNOgesic/input/reads/SRR515255_50000.fasta,ANNOgesic/i

After that, we assign `all_samples:$READ_FILE` to `--read_files`. `all_sample` is the set name of read files. The all four read files will be compute together. Now, we can try `circrna`

```
$ annogesic circrna \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --parallels 10 \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --read_files all_samples:$READ_FILES \
    --project_path ANNOgesic
```

`testrealign.x` is not available, please refer to *Required tools or databases*.

Several output folders will be generated.

```
$ ls ANNOgesic/output/circRNAs/
circRNA_tables  gffs  log.txt  segemehl_alignment_files  segemehl_splice_results␣
↪statistics
```

`segemehl_alignment_files` and `segemehl_splice_results` are for output of Segemehl. `segemehl_alignment_files` stores Bam files of the alignment and `segemehl_splice_results` stores results of the splice detection.

```
$ ls ANNOgesic/output/circRNAs/segemehl_alignment_files/NC_009839.1/
SRR515254_50000_NC_009839.1.bam  SRR515256_50000_NC_009839.1.bam
SRR515255_50000_NC_009839.1.bam  SRR515257_50000_NC_009839.1.bam
$ ls ANNOgesic/output/circRNAs/segemehl_splice_results/NC_009839.1/
NC_009839.1_all_samples_splicesites.bed  NC_009839.1_all_samples_transrealigned.bed
```

Gff files, tables and statistic files are stored in `gffs`, `circRNA_tables` and `statistics`.

```
$ ls ANNOgesic/output/circRNAs/gffs/NC_009839.1/
NC_009839.1_all_samples_circRNA_all.gff  NC_009839.1_all_samples_circRNA_best.gff
$ ls ANNOgesic/output/circRNAs/circRNA_tables/NC_009839.1/
NC_009839.1_all_samples_circRNA_all.csv  NC_009839.1_all_samples_circRNA_best.csv
$ ls ANNOgesic/output/circRNAs/statistics/
stat_NC_009839.1_all_samples_circRNA.csv
```

`NC_009839.1_all_samples_circRNA_all.gff` and `NC_009839.1_all_samples_circRNA_all.csv` store all circular RNAs without filtering. `NC_009839.1_all_samples_circRNA_best.gff` and `NC_009839.1_all_samples_circRNA_best.csv` store the circular RNAs after filtering. In our case, there are some circular RNAs can be detected without filtering, but no one can exist after filtering.

1.5.18 SNP calling

If we want to know SNPs or mutations based on our RNA-Seq data, we can use `snp` to achieve this purpose. `snp` is compose of two parts. One part is for obtaining the differences between our reference genome and the closely related genome. If we have no fasta file of our reference genome, this part will be very useful. We just need to map the reads on the fasta file of the closely related genome. Then using `snp` can automatically detect differences between the closely related genome and our reference genome. Furthermore, potential fasta files of the refernce genome can be

generated automatically as well. The other part is for detecting SNPs or mutations of the reference genome if the fasta file of the reference genome can be provided. In this part, you can know real mutations of our reference genonme.

Before running the subcommand, BAM files are required. Since we already generated them via running `circrna`, we can just put them to the corresponding folder. Please remember that the mapping function of `circrna` is only basic one.

First, we copy the bam files to `BAMs_map_reference_genomes`.

```
$ cp ANNOgesic/output/circRNAs/segemehl_alignment_files/NC_009839.1/SRR51525*␣
↪ANNOgesic/input/BAMs/BAMs_map_reference_genomes/tex_notex
```

Now, we can set our bam files

```
$ BAM_FILES=ANNOgesic/input/BAMs/BAMs_map_reference_genomes/tex_notex/SRR515254_50000_
↪NC_009839.1.bam,\
  ANNOgesic/input/BAMs/BAMs_map_reference_genomes/tex_notex/SRR515255_50000_NC_009839.
↪1.bam,\
  ANNOgesic/input/BAMs/BAMs_map_reference_genomes/tex_notex/SRR515256_50000_NC_009839.
↪1.bam,\
  ANNOgesic/input/BAMs/BAMs_map_reference_genomes/tex_notex/SRR515257_50000_NC_009839.
↪1.bam
```

Then we can run the subcommand with three programs – `extend_BAQ`, `with_BAQ` and `without_BAQ`. `all_sample:$BAM_FILES` for `--bam_files` means the set name of BAM files is "all_sample", and all four BAM files need to be computed together.

```
$ annogesic snp \
    --bam_type reference_genome \
    --program with_BAQ without_BAQ extend_BAQ \
    --bam_files all_samples:$BAM_FILES \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --project_path ANNOgesic
```

Two output folders will be generated, `compare_related_and_reference_genomes` is for the results of comparison between closely related genome and reference genome, `mutations_of_reference_genomes` is for results of detecting mutations of the reference genome.

```
$ ls ANNOgesic/output/SNP_calling/
compare_related_and_reference_genomes  mutations_of_reference_genomes  log.txt
```

Since we run `reference_genome`, the output folders are generated under `mutations_of_reference_genomes`.

```
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/
seqs  SNP_raw_outputs  SNP_tables  statistics
```

The output folders are compose of three parts - `extend_BAQ`, `with_BAQ` and `without_BAQ`.

```
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/seqs/
extend_BAQ/  with_BAQ/    without_BAQ/
```

In `seqs`, the potential sequences can be found.

```
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/seqs/with_BAQ/NC_
↪009839.1/
NC_009839.1_all_samples_NC_009839.1_1_1.fa
```

SNP_raw_outputs stores output of Samtools and Bcftools. SNP_tables stores results after filtering and the indices of potential sequence (potential sequences are stored in seqs). statistics stores the statistic files.

```
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/SNP_raw_outputs/NC_
↪009839.1/
NC_009839.1_extend_BAQ_all_samples.vcf  NC_009839.1_with_BAQ_all_samples.vcf  NC_
↪009839.1_without_BAQ_all_samples.vcf
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/SNP_tables/NC_009839.
↪1/
NC_009839.1_extend_BAQ_all_samples_best.vcf           NC_009839.1_with_BAQ_all_
↪samples_best.vcf           NC_009839.1_without_BAQ_all_samples_best.vcf
NC_009839.1_extend_BAQ_all_samples_seq_reference.csv  NC_009839.1_with_BAQ_all_
↪samples_seq_reference.csv  NC_009839.1_without_BAQ_all_samples_seq_reference.csv
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/statistics/
figs                                                  stat_NC_009839.1_with_BAQ_all_
↪samples_SNP_best.csv     stat_NC_009839.1_without_BAQ_all_samples_SNP_raw.csv
stat_NC_009839.1_extend_BAQ_all_samples_SNP_best.csv  stat_NC_009839.1_with_BAQ_all_
↪samples_SNP_raw.csv
stat_NC_009839.1_extend_BAQ_all_samples_SNP_raw.csv   stat_NC_009839.1_without_BAQ_
↪all_samples_SNP_best.csv
$ ls ANNOgesic/output/SNP_calling/mutations_of_reference_genomes/statistics/figs
NC_009839.1_extend_BAQ_all_samples_NC_009839.1_SNP_QUAL_best.png  NC_009839.1_with_
↪BAQ_all_samples_NC_009839.1_SNP_QUAL_best.png  NC_009839.1_without_BAQ_all_samples_
↪NC_009839.1_SNP_QUAL_best.png
NC_009839.1_extend_BAQ_all_samples_NC_009839.1_SNP_QUAL_raw.png   NC_009839.1_with_
↪BAQ_all_samples_NC_009839.1_SNP_QUAL_raw.png   NC_009839.1_without_BAQ_all_samples_
↪NC_009839.1_SNP_QUAL_raw.png
```

1.5.19 Mapping Gene ontology

Gene ontology is useful for understanding functions of gene products. go_term can search GO terms of the proteins in annotation files. Before running go_term, we need to prepare some databases. First, please download goslim.obo and go.obo and idmapping_selected.tab.

```
$ wget -cP ANNOgesic/input/databases http://www.geneontology.org/ontology/subsets/
↪goslim_generic.obo
$ wget -cP ANNOgesic/input/databases http://geneontology.org/ontology/go.obo
$ wget -cP ANNOgesic/input/databases ftp://ftp.uniprot.org/pub/databases/uniprot/
↪current_release/knowledgebase/idmapping/idmapping_selected.tab.gz
$ gunzip ANNOgesic/input/databases/idmapping_selected.tab.gz
```

Now, we have all required databases. We can also import information of the transcripts to generate results which only contain the expressed CDSs.

Let's try it.

```
$ annogesic go_term \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --go_obo ANNOgesic/input/databases/go.obo \
    --goslim_obo ANNOgesic/input/databases/goslim_generic.obo \
    --uniprot_id ANNOgesic/input/databases/idmapping_selected.tab \
    --project_path ANNOgesic
```

The results of go_term are stored in GO_term_results. The statistic files and figures are stored in statistics.

```
$ ls ANNOgesic/output/GO_terms/
all_CDSs  expressed_CDSs  log.txt
$ ls ANNOgesic/output/GO_terms/all_CDSs/
GO_term_results  statistics
$ ls ANNOgesic/output/GO_terms/all_CDSs/GO_term_results/NC_009839.1/
all_genomes_uniprot.csv
$ ls ANNOgesic/output/GO_terms/all_CDSs/statistics/NC_009839.1/
figs  stat_NC_009839.1.csv
$ ls ANNOgesic/output/GO_terms/all_CDSs/statistics/NC_009839.1/figs/
NC_009839.1_biological_process.png  NC_009839.1_cellular_component.png  NC_009839.1_
↪molecular_function.png  NC_009839.1_three_roots.png
```

1.5.20 Prediction of Subcellular localization

Subcellular localization is also a useful information for analysis of protein functions. For detecting subcellular localization, we can use the subcommand `localization`. We can also import information of the transcripts to generate results which only contain the expressed CDSs.

```
$ annogesic localization \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --bacteria_type negative \
    --project_path ANNOgesic
```

Two output folders will be generated. `psortb_results` stores the output files of Psortb. `statistics` stores statistic files and figures.

```
$ ls ANNOgesic/output/subcellular_localization/
all_CDSs  expressed_CDSs  log.txt
$ ls ANNOgesic/output/subcellular_localization/all_CDSs/
psortb_results  statistics
$ ls ANNOgesic/output/subcellular_localization/all_CDSs/psortb_results/NC_009839.1/
NC_009839.1_raw.txt  NC_009839.1_table.csv
$ ls ANNOgesic/output/subcellular_localization/all_CDSs/statistics/NC_009839.1/
NC_009839.1_NC_009839.1_sublocal.png  stat_NC_009839.1_sublocal.csv
```

1.5.21 Generating protein-protein interaction network

`ppi_network` can detect protein-protein interaction based on STRING (a database of protein-protein interaction) and searching the literatures by implementing PIE (text-mining for protein-protein interaction). Therefore, `ppi_network` can generate protein-protein interaction networks based on literatures.

Before running the subcommand, you need to download species.v{$VERSIO}.txt from STRING

```
$ wget -cP ANNOgesic/input/databases http://string-db.org/newstring_download/species.
↪v10.5.txt
```

Now, we can try the subcommand.

```
$ annogesic ppi_network \
    --query_strains NC_009839.1.gff:NC_009839.1:'Campylobacter jejuni 81176':
↪'Campylobacter jejuni' \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
```

(continues on next page)

(continued from previous page)

```
    --species_string ANNOgesic/input/databases/species.v10.5.txt \
    --query NC_009839.1:962231:963001:- NC_009839.1:123943:125151:+ \
    --without_strain_pubmed \
    --project_path ANNOgesic
```

We only detected for two proteins. If you want to detect for all proteins in gff files, you can easily assign `all` in `--query`.

Three output folders were generated.

```
$ ls ANNOgesic/output/PPI_networks/
all_results/  best_results/ figures/  log.txt
```

`all_results` is for all interactions without filtering. `best_results` is for the interactions with the high PIE score. `figures` is for figures of the protein-protein interaction networks. There are two subfolders - `with_strain` and `without_strain` in `figures`, `all_results`, and `best_results`. The two subfolders store all information of the interactions and PIE scores. `with_strain` is for results with assigning specific strain name for searching literatures. `without_strain` is for results without giving specific strain name for searching literatures.

```
$ ls ANNOgesic/output/PPI_networks/all_results/PPI_NC_009839.1/
NC_009839.1_without_strain.csv  NC_009839.1_with_strain.csv  without_strain  with_
↪strain
$ ls ANNOgesic/output/PPI_networks/best_results/PPI_NC_009839.1/
NC_009839.1_without_strain.csv  NC_009839.1_with_strain.csv  without_strain  with_
↪strain
$ ls ANNOgesic/output/PPI_networks/figures/PPI_NC_009839.1/
without_strain  with_strain
$ ls ANNOgesic/output/PPI_networks/all_results/PPI_NC_009839.1/with_strain/NC_009839.
↪1/
atpC_atpD.csv                    Cjejjejuni_010100005380_livH.csv                  ␣
↪ Cjejjejuni_010100005385_livF.csv  Cjejjejuni_010100005385_livM.csv  livH_livG.csv ␣
↪livM_livG.csv
Cjejjejuni_010100005380_livF.csv  Cjejjejuni_010100005380_livM.csv                ␣
↪ Cjejjejuni_010100005385_livG.csv  livG_livF.csv                    livH_livM.csv
Cjejjejuni_010100005380_livG.csv  Cjejjejuni_010100005385_Cjejjejuni_010100005380.csv␣
↪ Cjejjejuni_010100005385_livH.csv  livH_livF.csv                    livM_livF.csv
$ ls ANNOgesic/output/PPI_networks/all_results/PPI_NC_009839.1/without_strain/NC_
↪009839.1/
atpC_atpD.csv                    Cjejjejuni_010100005380_livH.csv                  ␣
↪ Cjejjejuni_010100005385_livF.csv  Cjejjejuni_010100005385_livM.csv  livH_livG.csv ␣
↪livM_livG.csv
Cjejjejuni_010100005380_livF.csv  Cjejjejuni_010100005380_livM.csv                ␣
↪ Cjejjejuni_010100005385_livG.csv  livG_livF.csv                    livH_livM.csv
Cjejjejuni_010100005380_livG.csv  Cjejjejuni_010100005385_Cjejjejuni_010100005380.csv␣
↪ Cjejjejuni_010100005385_livH.csv  livH_livF.csv                    livM_livF.csv
$ ls ANNOgesic/output/PPI_networks/best_results/PPI_NC_009839.1/without_strain/NC_
↪009839.1/
Cjejjejuni_010100005380_livF.csv  Cjejjejuni_010100005380_livH.csv  Cjejjejuni_
↪010100005385_livF.csv  Cjejjejuni_010100005385_livH.csv  livG_livF.csv  livH_livG.
↪csv  livM_livF.csv
Cjejjejuni_010100005380_livG.csv  Cjejjejuni_010100005380_livM.csv  Cjejjejuni_
↪010100005385_livG.csv  Cjejjejuni_010100005385_livM.csv  livH_livF.csv  livH_livM.
↪csv  livM_livG.csv
$ ls ANNOgesic/output/PPI_networks/best_results/PPI_NC_009839.1/with_strain/NC_009839.
↪1/
Cjejjejuni_010100005385_Cjejjejuni_010100005380.csv
```

(continues on next page)

(continued from previous page)

```
$ ls ANNOgesic/output/PPI_networks/figures/PPI_NC_009839.1/with_strain/NC_009839.1/
C8J_RS04960_livG.png
$ ls ANNOgesic/output/PPI_networks/figures/PPI_NC_009839.1/without_strain/NC_009839.1/
C8J_RS04960_livG.png
```

1.5.22 Generating riboswitch and RNA thermometer

If we want to detect riboswitches and RNA thermometers, we can use subcommand `riboswitch_thermometer`. Before running it, we need to get the information of known riboswitches and RNA thermometers in Rfam. The riboswitches and RNA thermometer files can be downloaded them from our Git repository.

```
$ wget -cP ANNOgesic/input/riboswitch_ID_file/ https://raw.githubusercontent.com/Sung-
↪Huan/ANNOgesic/master/database/Rfam_riboswitch_ID.csv
$ wget -cP ANNOgesic/input/RNA_thermometer_ID_file/ https://raw.githubusercontent.com/
↪Sung-Huan/ANNOgesic/master/database/Rfam_RNA_thermometer_ID.csv
```

We also need to download Rfam.

```
$ wget -cP ANNOgesic/input/databases ftp://ftp.ebi.ac.uk/pub/databases/Rfam/12.0/Rfam.
↪tar.gz
$ cd ANNOgesic/input/databases
$ tar -zxvf Rfam.tar.gz
$ rm Rfam.tar.gz
$ cd ../../../
```

Now we can try the subcommand.

```
$ annogesic riboswitch_thermometer \
    --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
    --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --riboswitch_id_file ANNOgesic/input/riboswitch_ID_file/Rfam_riboswitch_ID.csv \
    --rna_thermometer_id_file ANNOgesic/input/RNA_thermometer_ID_file/Rfam_RNA_
↪thermometer_ID.csv \
    --rfam_path ANNOgesic/input/databases/CMs/Rfam.cm \
    --transcript_files ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
    --tss_files ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --project_path ANNOgesic
```

Output files are following, `gffs` stores gff files of the riboswitchs / RNA_thermometers; `tables` stores tables of the riboswitchs / RNA_thermometers; `scan_Rfam_results` stores output files of scanning Rfam; `statistics` is for statistic files.

```
$ ls ANNOgesic/output/riboswitches/
gffs  log.txt  scan_Rfam_results  statistics  tables
$ ls ANNOgesic/output/riboswitches/gffs/
NC_009839.1_riboswitch.gff
$ ls ANNOgesic/output/riboswitches/scan_Rfam_results/NC_009839.1/
NC_009839.1_riboswitch_prescan.txt  NC_009839.1_riboswitch_scan.txt
$ ls ANNOgesic/output/riboswitches/tables/
NC_009839.1_riboswitch.csv
$ ls ANNOgesic/output/riboswitches/statistics/
stat_NC_009839.1_riboswitch.txt
$ ls ANNOgesic/output/RNA_thermometers/
gffs  log.txt  scan_Rfam_results  statistics  tables
```

(continues on next page)

(continued from previous page)

```
$ ls ANNOgesic/output/RNA_thermometers/gffs/
NC_009839.1_RNA_thermometer.gff
$ ls ANNOgesic/output/RNA_thermometers/scan_Rfam_results/NC_009839.1/
NC_009839.1_RNA_thermometer_prescan.txt  NC_009839.1_RNA_thermometer_scan.txt
$ ls ANNOgesic/output/RNA_thermometers/tables/
NC_009839.1_RNA_thermometer.csv
$ ls ANNOgesic/output/RNA_thermometers/statistics/
stat_NC_009839.1_RNA_thermometer.txt
```

1.5.23 Detection of CRISPR

CRISPR is an important features for research of immunology. `crispr` is a useful subcommand for CRISPR detection. Let's try it.

```
$ annogesic crispr \
   --annotation_files ANNOgesic/input/references/annotations/NC_009839.1.gff \
   --fasta_files ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
   --project_path ANNOgesic
```

Output are as following, `CRT_results` stores output of CRT; `gffs` stores gff files of the CRISPRs; `statistics` is for statistic files.

```
$ ls ANNOgesic/output/crisprs/
CRT_results  gffs  log.txt  statistics
$ ls ANNOgesic/output/crisprs/CRT_results
NC_009839.1.txt
$ ls ANNOgesic/output/crisprs/gffs
all_candidates  best_candidates
$ ls ANNOgesic/output/crisprs/gffs/all_candidates
NC_009839.1_CRISPR.gff
$ ls ANNOgesic/output/crisprs/gffs/best_candidates
NC_009839.1_CRISPR.gff
$ ls ANNOgesic/output/crisprs/statistics
NC_009839.1.csv
```

1.5.24 Merge all features to be one gff file

Now, we generated all features that ANNOgesic can provide. Sometimes, merging all features into one gff file is useful. `merge_features` is the subcommand to achieve this purpose. Moreover, `merge_features` can search parent transcript to each feature that we assigned.

Now let's do it. We merge all features that we have.

```
ALL_FEATURES="ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
              ANNOgesic/input/references/annotations/NC_009839.1.gff \
              ANNOgesic/output/UTRs/5UTRs/gffs/NC_009839.1_5UTR.gff \
              ANNOgesic/output/UTRs/3UTRs/gffs/NC_009839.1_3UTR.gff \
              ANNOgesic/output/terminators/gffs/best_candidates/NC_009839.1_term.gff \
              ANNOgesic/output/processing_sites/gffs/NC_009839.1_processing.gff \
              ANNOgesic/output/sRNAs/gffs/best_candidates/NC_009839.1_sRNA.gff \
              ANNOgesic/output/sORFs/gffs/best_candidates/NC_009839.1_sORF.gff \
              ANNOgesic/output/riboswitches/gffs/NC_009839.1_riboswitch.gff \
              ANNOgesic/output/RNA_thermometers/gffs/NC_009839.1_RNA_thermometer.gff \
              ANNOgesic/output/crisprs/gffs/best_candidates/NC_009839.1_CRISPR.gff"
```

```
$ annogesic merge_features \
   --transcript_file ANNOgesic/output/transcripts/gffs/NC_009839.1_transcript.gff \
   --other_features_files $ALL_FEATURES \
   --output_prefix NC_009839.1 \
   --source_for_overlapping RefSeq \
   --project_path ANNOgesic
```

In the tutorial, if duplicated features exist, only the data from RefSeq will be kept. Output gff file is stored in `merge_all_features`

```
$ ls ANNOgesic/output/merge_all_features/
NC_009839.1_merge_features.gff  log.txt
```

1.5.25 Producing the screenshots

It is a good idea if we can get screenshots of our interesting features. Then we can check them very quickly. Therefore, ANNOgesic provides a subcommand `screenshot` for generating screenshots.

Before we running it, we have to install IGV.

For testing, we use TSSs as main feature, sRNAs and CDSs as side features.

```
$ annogesic screenshot \
    --main_gff ANNOgesic/output/TSSs/gffs/NC_009839.1_TSS.gff \
    --side_gffs ANNOgesic/input/references/annotations/NC_009839.1.gff \
                ANNOgesic/output/sRNAs/gffs/best_candidates/NC_009839.1_sRNA.gff \
    --fasta_file ANNOgesic/input/references/fasta_files/NC_009839.1.fa \
    --output_folder ANNOgesic/output/TSSs \
    --tex_notex_libs $TEX_LIBS \
    --project_path ANNOgesic
```

Two txt files and two folders will be generated.

```
$ ls ANNOgesic/output/TSSs/screenshots/NC_009839.1/
forward/     forward.txt  reverse/     reverse.txt
```

`forward.txt` and `reverse.txt` are batch files for running in IGV. `forward` and `reverse` are the folders for storing screenshots.

Since there are numerous candidates, we only generate several ones for testing.

```
head -n 30 ANNOgesic/output/TSSs/screenshots/NC_009839.1/forward.txt > ANNOgesic/
↪output/TSSs/screenshots/NC_009839.1/forward_6_cases.txt
head -n 30 ANNOgesic/output/TSSs/screenshots/NC_009839.1/reverse.txt > ANNOgesic/
↪output/TSSs/screenshots/NC_009839.1/reverse_6_cases.txt
```

Now, please open IGV and follow the procedures: Tools -> Run Batch Script -> choose `forward_6_cases.txt`. Once it is done, please do it again for the reverse strand: Tools -> Run Batch Script -> choose `reverse_6_cases.txt`. If you want to generate the screenshots for all candidates, you can run `forward.txt` and `reverse.txt`. Please be careful, if you use Docker container, the path may be not correct.

As soon as the generation of the screenshots is done, we can see that there are several screenshots in `forward` and `reverse`.

```
$ ls ANNOgesic/output/TSSs/screenshots/NC_009839.1/forward
NC_009839.1:1396-1396.png  NC_009839.1:14812-14812.png  NC_009839.1:6676-6676.png  NC_
↪009839.1:6680-6680.png  NC_009839.1:8098-8098.png  NC_009839.1:9295-9295.png
```

(continues on next page)

(continued from previous page)

```
$ ls ANNOgesic/output/TSSs/screenshots/NC_009839.1/reverse
NC_009839.1:15670-15670.png  NC_009839.1:18053-18053.png  NC_009839.1:18360-18360.png␣
↪ NC_009839.1:2199-2199.png  NC_009839.1:4463-4463.png  NC_009839.1:856-856.png
```

1.5.26 Coloring the screenshots

If we have numerous libraries and we want to check TSSs, distinguishing the tracks of TEX+ and TEX- will be painful. Therefore, we provide a subcommand `colorize_screenshot_tracks` to colorize our screenshots based on the tracks.

```
$ annogesic colorize_screenshot_tracks \
    --track_number 2 \
    --screenshot_folder ANNOgesic/output/TSSs \
    --project_path ANNOgesic
```

The output filenames are the same as before. However, when we open the files of figures, the tracks are colorized.

```
$ ls ANNOgesic/output/TSSs/screenshots/NC_009839.1/forward
NC_009839.1:1396-1396.png  NC_009839.1:14812-14812.png  NC_009839.1:6676-6676.png  NC_
↪009839.1:6680-6680.png  NC_009839.1:8098-8098.png  NC_009839.1:9295-9295.png
$ ls ANNOgesic/output/TSSs/screenshots/NC_009839.1/reverse
NC_009839.1:15670-15670.png  NC_009839.1:18053-18053.png  NC_009839.1:18360-18360.png␣
↪ NC_009839.1:2199-2199.png  NC_009839.1:4463-4463.png  NC_009839.1:856-856.png
```

Now we already finished the first wonderful trip of ANNOgesic. Hopefully, you enjoy it!

1.6 FAQ

No matter which kinds of error messages occur, please always make sure **the seq IDs or strain names of all Gff files, fasta files, BAM files and wiggle files must be consistent.**

- **Q1:**

I got the error about complaining no module can be found as following:

```
Traceback (most recent call last):
  File "ANNOgesic.py", line 6, in <module>
    from annogesiclib.controller import Controller
ImportError: No module named annogesiclib.controller
```

- **A1:**

Switch to Python 3.3 or higher. The newer versions can handle this.

- **Q2:**

I try to install ANNOgesic in my own directory. However, when I execute ANNOgesic, an error comes out as following:

```
Traceback (most recent call last):
  File "bin/annogesic", line 7, in <module>
    from annogesiclib.controller import Controller
ImportError: No module named 'annogesiclib'
```

- **A2:**

Please generate a soft link of `annogesiclib` to `bin`.

```
$ cd ANNOgesic/bin
$ ln -s ../annogesiclib .
```

- **Q3:**

When I finished sRNA detection, there are some files in `sRNAs/gffs/for_classes` and `sRNAs/tables/for_classes`. What do the classes mean?

- **A3:**

Please check `stat_sRNA_class_$GENOME.csv` in `stat` folder. The information of classes can be found in it. The classes are based on the input information.

- **Q4:**

I got a sRNA gff file which only contain `##gff-version 3`. What does it mean?

- **A4:**

If you only got `##gff-version 3` in sRNA gff file, it means that ANNOgesic can not find any sRNA in this genome. This case can be applied to other genomic feature prediction.

- **Q5:**

How long does sRNA target prediction take?

- **A5**

It depends on the input data size. However, most of the sRNA target prediction tools need to spend around one hour for detecting targets of one sRNA. RNAplex is a fast one which can detect the targets of one sRNA in several minutes. If you want to reduce the running time, you can specify `--program RNAplex`. However, the accuracy may be influenced.

- **Q6:**

When I run sRNA target prediction, I found many files in `RNAplfold`. Are they necessary?

- **A6:**

These files are intermediate input for RNAplex. They will be deleted when RNAplex is done.

- **Q7:**

If I only have fragmented RNA-Seq libraries or conventional RNA-Seq libraries, what kinds of annotations can be generated? On the other hands, if only dRNA-Seq libraries are available, what kinds of annotations can be generated?

- **A7:**

Only fragmented RNA-Seq or conventional RNA-Seq libraries are available:

dRNA-Seq data (TEX+/-) are mainly used for TSS detection. Therefore, if the user has no dRNA-Seq data, the annotations which need TSS information may not be able to detect. The annotations which can be detected without dRNA-Seq libraries are: transcript, operon (sub-operon can not be detected), terminator, CRISPR, sRNA (the results may be influenced, and UTR-derived sRNA can not be detected), sRNA target, GO term, subcellular localization, circular RNA, SNP, protein-protein interaction, annotation transfer. For the details of sRNA detection, if TSSs can not be obtained, the selection of the best sRNA candidates can only considers the information like secondary structure folding energy, BLAST results, etc. Thus, the accuracy of sRNA detection (especially for the selection of the best sRNA candidates) will be influenced. For operon detection, since the TSS information is not available, the sub-operons can not be detected.

The other features (including UTR-derived sRNAs) which can not be detected precisely or at all by only fragmented or conventional RNA-Seq data require the information of TSS or ribosome-binding site.

Only dRNA-Seq libraries are available:

Actually, all the genomic features can be detected by only using dRNA-Seq data. However, dRNA-Seq usually does not able to detect transcript boundary. Thus, the genomic features which are related to transcript boundary will be influenced, such as transcripts, sRNAs, sORFs, operon, terminators, etc. Although the 3'end of transcripts may be not clear, the genomic feature detections of ANNOgesic still show high performances by comparing to the previously published data.

- **Q8:**

An error message related to 'defined(@array)' shows when I run annotation_transfer.

- **A8:**

This issue is caused by different version of Perl. RATT is written by the old version of Perl. Please do the following steps to fix this error.

1. Download RATT via PAGIT
2. Set environment path.
3. Run sed -i '244s/defined//' $RATT_FOLDER/main.ratt.pl && sed -i '19s/$PAGIT_HOME//usr/' $RATT_FOLDER/start.ratt.sh (Please specify the folder of your RATT to $RATT_FOLDER)

We also suggest the user running ANNOgesic via our Docker image. This issue was fixed in our Docker image.

- **Q9:**

After running sRNA prediction of ANNOgesic, I got a huge number of sRNA candidates. The amount is too many for me, how can I remove some low-confidence ones?

- **A9:**

Although the sRNA prediction of ANNOgesic was tested by RNA-Seq data from different strains of bacteria, the setting may not be able to apply to some other specific RNA-Seq data. If the amount of candidates are too many, it may be due to the low coverage rate of RNA-Seq data. Therefore, many short transcripts are genearted. There are several ways to further filted out the sRNAs.

- Using promoter and terminator for `--filter_info`: We suggest the user to put promoter information first.

If terminator information is still needed to add to `--filter_info`, please run `terminator` with sRNA files (`--srna_files`) first. Then using that terminator output file to run `srna`.

- Using ranking number: The output table of sRNAs contains the ranking number based on coverage and promoter information.

The user can select top ranking ones to do the following analyses.

- Adjusting coverage cutoff (not recommended): There are some coverage cutoffs can be modified by the user. The

default setting is tested based on some RNA-Seq data from different bacteria. Thus, changing these parameters may influence the accuracy of the prediction.

1.7 License

THE SOFTWARE IS PROVIDED “AS IS” AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

CHAPTER 2

Introduction

ANNOgesic is the swiss army knife for RNA-Seq based annotation of bacterial/archaeal genomes.

It is a modular, command-line tool that can integrate different types of RNA-Seq data based on dRNA-Seq (differential RNA-Seq) or RNA-Seq protocols that inclusde transcript fragmentation to generate high quality genome annotations. It can detect genes, CDSs/tRNAs/rRNAs, transcription starting sites (TSS) and processing sites, transcripts, terminators, untranslated regions (UTR) as well as small RNAs (sRNA), small open reading frames (sORF), circular RNAs, CRISPR related RNAs, riboswitches and RNA-thermometers. It can also perform RNA-RNA and protein-protein interactions prediction. Furthermore, it groups genes into operons and sub-operons and reveal promoter motifs. It can also allocate GO term and subcellular localization to genes. Several of ANNOgesic features are new implementations while other build on well known third-party tools for which it offers adaptive parameter-optimizations. Additionally, numerous visualization and statistics help the user to quickly evaluat feature predictions resulting from an ANNOgesic analysis. The tool was heavily tested with several RNA-Seq data set from bacterial as well as archaeal samples.

```
usage: annogesic [-h] [--version]
                 {create,get_input_files,update_genome_fasta,annotation_transfer,
                  tss_ps,optimize_tss_ps,terminator,transcript,utr,srna,sorf,
                  promoter,operon,circrna,go_term,srna_target,snp,ppi_network,
                  localization,riboswitch_thermometer,crispr,merge_features,
                  screenshot,colorize_screenshot_tracks}
                 ...

positional arguments:
  {create,get_input_files,update_genome_fasta,annotation_transfer,tss_ps,optimize_tss_
↪ps,
   terminator,transcript,utr,srna,sorf,promoter,operon,circrna,go_term,srna_target,
↪snp,
   ppi_network,localization,riboswitch_thermometer,crispr,merge_features,screenshot,
   colorize_screenshot_tracks}
                        commands
    create              Create a project
    get_input_files     Get required files. (i.e. annotation files, fasta
                        files)
    update_genome_fasta
                        Get fasta files of query genomes if the query
```

(continues on next page)

(continued from previous page)

```
                        sequences do not exist.
    annotation_transfer
                        Transfer the annotations from a closely related
                        species genome to a target genome.
    tss_ps              Detect TSSs or processing sites.
    optimize_tss_ps     Optimize TSSs or processing sites based on manual
                        detected ones.
    terminator          Detect rho-independent terminators.
    transcript          Detect transcripts based on coverage file.
    utr                 Detect 5'UTRs and 3'UTRs.
    srna                Detect intergenic, antisense and UTR-derived sRNAs.
    sorf                Detect expressed sORFs.
    promoter            Discover promoter motifs.
    operon              Detect operons and sub-operons.
    circrna             Detect circular RNAs.
    go_term             Extract GO terms from Uniprot.
    srna_target         Detect sRNA-mRNA interactions.
    snp                 Detect SNP/mutation and generate fasta file if
                        mutations were found.
    ppi_network         Detect protein-protein interactions suported by
                        literature.
    localization        Predict subcellular localization of proteins.
    riboswitch_thermometer
                        Predict riboswitches and RNA thermometers.
    crispr              Predict CRISPR related RNAs.
    merge_features      Merge all features to one gff file.
    screenshot          Generate screenshots for selected features using IGV.
    colorize_screenshot_tracks
                        Add color information to screenshots (e.g. useful for
                        dRNA-Seq based TSS and PS detection. It only works
                        after running "screenshot" (after running batch
                        script).

optional arguments:
  -h, --help            show this help message and exit
  --version, -v         show version
```

CHAPTER 3

Download

```
git clone git@github.com:Sung-Huan/ANNOgesic.git
```

or

```
pip3 install annogesic
```

Please check *Required tools or databases* for the required third-party tools.

CHAPTER 4

Source code

The source code of ANNOgesic can be found at Github.

CHAPTER 5

License

ISC (Internet Systems Consortium license ~ simplified BSD license) - see LICENSE

CHAPTER 6

Cite

SH Yu, J. Vogel, KU Förstner. 2018, GigaScience, DOI:10.1093/gigascience/giy096, PMID:30169674.

CHAPTER 7

Contact

For question and requests feel free to contact Sung-Huan Yu