DROP

Jun 23, 2022

Contents:

1 Installation **3**
- 1.1 Initialize a project . . . 3
- 1.2 Other DROP versions . . . 4
- 1.3 Prerequisites . . . 4

2 Preparing the Input Data **7**
- 2.1 Config file . . . 7
- 2.2 Files to download . . . 13
- 2.3 Example of RNA replicates . . . 13
- 2.4 Advanced options . . . 14

3 Pipeline Commands **15**
- 3.1 Dry run . . . 15
- 3.2 Parallelizing jobs . . . 15
- 3.3 Executing subworkflows . . . 16
- 3.4 Rerunning the pipeline . . . 16

4 Results and Output of DROP **19**
- 4.1 Aberrant Expression . . . 19
- 4.2 Aberrant Splicing . . . 20
- 4.3 Mono-allelic Expression . . . 21

5 License **23**

6 Help **25**

7 Quickstart **27**

DROP is intended to help researchers use RNA-Seq data in order to detect genes with aberrant expression, aberrant splicing, mono-allelic expression, and RNA-Seq variant calling. It consists of 4 independent modules for each of those strategies. After installing DROP, the user needs to fill in the config file and sample annotation table (*Preparing the Input Data*). Then, DROP can be executed in multiple ways (*Pipeline Commands*).

CHAPTER 1

Installation

DROP is available on bioconda . In case the conda channel priority is set to `strict`, it should be reset to `flexible`:

```
conda config --set channel_priority true
```

We recommend using a dedicated conda environment (here: `drop_env`) for installing drop. For installing, use *mamba* instead of *conda* as it provides more reliable and faster dependency solving.

```
mamba create -n drop_env -c conda-forge -c bioconda drop
```

Installation time: ~ 10min

Test whether the pipeline runs through by setting up the demo dataset in an empty directory (e.g. `~/drop_demo`).

```
mkdir ~/drop_demo
cd ~/drop_demo

# demo will download the necessary data and pipeline files
drop demo
```

The pipeline can be run using snakemake commands Run time: ~25min

```
snakemake --cores 1 -n # dryrun
snakemake --cores 1
```

1.1 Initialize a project

The demo project can be modified to be used for a new project. Alternatively, a new DROP project can be set up using `drop init`.

```
cd <path/to/project>
drop init
```

This will create an empty `config.yaml` file that needs to be filled according to the project data. You also need to prepare a sample annotation file. Go to *Preparing the Input Data* for more details.

1.2 Other DROP versions

The developer version of DROP can be found in the repository under the branch `dev`. Make sure that the *Prerequisites* are installed, preferably in a conda environment. Then install DROP from github using `pip`.

```
pip install git+https://github.com/gagneurlab/drop.git@dev
```

Alternatively, you can clone the desired branch of the repository and install from directory.

```
git clone -b dev https://github.com/gagneurlab/drop.git
pip install ./drop
```

If the package needs to be updated frequently, it is more useful to use the `-e` option of ``pip`. Any new update pulled from the repository will be available without reinstall. Note, that this requires an explicit call to update any existing project (*Updating DROP*).

```
pip install -e ./drop

# update project directory
cd <path/to/project>
drop update
```

1.3 Prerequisites

The easiest way to ensure that all dependencies are installed is to install the bioconda package, as described above. Once the environment is set up and installation was successful, other versions of drop can be installed with `pip`, overwriting the conda version of `DROP` (see *Other DROP versions*).

1.3.1 Installation without conda

Alternatively, DROP can be installed without `conda`. In this case the following dependencies must be met:

- Programming languages:
  - python >= 3.6 and pip >= 19.1
  - R >= 3.6, <=4.0.2 and corresponding bioconductor version
- Commandline tools:
  - GNU bc
  - GNU wget
  - tabix
  - samtools >= 1.9
  - bcftools >= 1.9
  - GATK >= 4.1.8
  - graphviz

- pandoc

Note: If you are using an already existing R installation, make sure that the R and bioconductor versions match. Otherwise, use the newest versions of R and bioconductor.

At first invocation, all necessary R packages will be installed with the first pipeline call. As this is a lengthy process, it might be desirable to install them in advance, if a local copy of the repository exists.

```
# optional
Rscript <path/to/drop/repo>/drop/installRPackages.R drop/requirementsR.txt
```

CHAPTER 2

Preparing the Input Data

The input files of DROP are:

- BAM files from RNA-seq (and their respective index files)
- VCF files from either WES or WGS (and their respective index files). Only used for the MAE module
- a configuration file containing the different parameters
- a sample annotation file
- a gene annotation file (gtf)
- a reference genome file (fasta, and its respective index)

For more details see the Materials section of the DROP manuscript.

2.1 Config file

The config file is in YAML format. It is composed of general and module-specific parameters. In *YAML*, a variable can be of the following types: boolean, string, numeric, list and dictionary. They are declared by writing the variable name followed by a colon, a space, and the value, for example:

```
# A boolean is binary and can be true of false
boolean_var: true    # or false

# A string is a text. Quotation marks are not needed.
string_var: whatever text

# A numeric can be an integer or a real number. A dot separates a decimal.
numeric_var: 0.05

# A list is a collection of elements of the same type.
list_var:
  - element_1   # elements are indented
```

(continues on next page)

(continued from previous page)

```
  - element_2

# A dictionary contains key-value pairs. It is a collection of multiple
#    elements where the key is a string and the value any type.
dictionary_var:
  key_1: value_1
  key_2: value_2
```

Now we describe the different parameters needed in DROP. When providing a path to a file or directory, please provide the *full system path*.

2.1.1 Global parameters

These parameters are applied to multiple modules and as a result should be consistent throughout the data you are analyzing

Parameter	Type	Description	Default/Examples
projectTitle	character	Title of the project to be displayed on the rendered HTML output	`Project 1`
htmlOutputPath	character	Full path of the folder where the HTML files are rendered	`/data/project1/htmlOutput`
indexWithFolderName	boolean	If true, the basename of the project directory will be used as prefix for the index.html file	`true`
genomeAssembly	character	Either hg19/hs37d5 or hg38/GRCh38, depending on the genome assembly used for mapping	`/data/project1`
sampleAnnotation	character	Full path of the sample annotation table	`/data/project1/sample_annotation.tsv`
root	character	Full path of the folder where the sub-directories processed_data and processed_results will be created containing DROP's output files.	`/data/project1`
genome	character	Full path of a human reference genome fasta file	`/path/to/hg19.fa`
genome	dictionary	(Optional) Multiple fasta files can be specified when RNA-seq BAM files belong to different genome. assemblies (eg, ncbi, ucsc).	`ncbi: /path/to/hg19_ncbi.fa` `ucsc: /path/to/hg19_ucsc.fa`
geneAnnotation	dictionary	A key-value list of the annotation name (key) and the full path to the GTF file (value). More than one annotation file can be provided.	`anno1: /path/to/gtf1.gtf` `anno2: /path/to/gtf2.gtf`
hpoFile	character	Full path of the file containing HPO terms. If `null` (default), it reads it from our webserver. Refer to *files-to-download*	`/path/to/hpo_file.tsv`

tools	dictionary	A key-value list of different commands (key) and the command (value) to run them	`gatkCmd: gatk`
			`bcftoolsCmd: bcftools`
			`samtoolsCmd: samtools`

2.1.2 Export counts dictionary

These parameters are directly used by the `exportCounts` snakemake command. This section is used to designate which aberrant expression and aberrant splicing groups should be exported into datasets that can be shared. To avoid sharing sensitive data, only the canonical annotations as described by *geneAnnotations* are exported. Only the groups excluded by *excludeGroups* are not exported.

Parameter	Type	Description	Default/Examples
geneAnnotations	list	key(s) from the `geneAnnotation` parameter, whose counts should be exported	`- gencode34`
excludeGroups	list	aberrant expression and aberrant splicing groups whose counts should not be exported. If `null` all groups are exported.	`- group1`

2.1.3 Aberrant expression dictionary

These parameters are directly used by the `aberrantExpression` snakemake command. Aberrant expression groups must have at least `10` samples per group. To use external counts please see the `Using External Counts` section.

Parameter	Type	Description	Default/Examples
run	boolean	If true, the module will be run. If false, it will be ignored.	`true`
groups	list	DROP groups that should be executed in this module. If not specified or `null` all groups are used.	`- group1` `- group2`
minIds	numeric	A positive number indicating the minimum number of samples that a group needs in order to be analyzed. We recommend at least 50.	`1`
fpkmCutoff	numeric	A positive number indicating the minimum FPKM per gene that 5% of the samples should have. If a gene has less it is filtered out.	`1 # suggested by OUTRIDER`
implementation	character	Either 'autoencoder', 'pca' or 'peer'. Methods to remove sample covariation in OUTRIDER.	`autoencoder`
zScoreCutoff	numeric	A non-negative number. Z scores (in absolute value) greater than this cutoff are considered as outliers.	`0`
padjCutoff	numeric	A number between (0, 1] indicating the maximum FDR an event can have in order to be considered an outlier.	`0.05`
maxTestedDimensionProportion	numeric	An integer that controls the maximum value that the encoding dimension can take. Refer to *advanced-options*.	`3`

2.1.4 Aberrant splicing dictionary

These parameters are directly used by the `aberrantSplicing` snakemake command. Aberrant splicing groups must have at least `10` samples per group. To use external counts please see the `Using External Counts` section.

Parameter	Type	Description	Default/Examples
run	boolean	If true, the module will be run. If false, it will be ignored.	`true`
groups	list	Same as in aberrant expression.	`# see aberrant expression example`
minIds	numeric	Same as in aberrant expression.	`1`
recount	boolean	If true, it forces samples to be recounted.	`false`
longRead	boolean	Set to true only if counting Nanopore or PacBio long reads.	`false`
keepNonStandardChrs	boolean	Set to true if non standard chromosomes are to be kept for further analysis.	`true`
filter	boolean	If false, no filter is applied. We recommend filtering.	`true`
minExpressionInOneSample	numeric	The minimal read count in at least one sample required for an intron to pass the filter.	`20`
minDeltaPsi	numeric	The minimal variation (in delta psi) required for an intron to pass the filter.	`0.05`
implementation	character	Either 'PCA' or 'PCA-BB-Decoder'. Methods to remove sample covariation in FRASER.	`PCA`
deltaPsiCutoff	numeric	A non-negative number. Delta psi values greater than this cutoff are considered as outliers.	`0.3 # suggested by FRASER`
padjCutoff	numeric	Same as in aberrant expression.	`0.1`
maxTestedDimensionProportion	numeric	Same as in aberrant expression.	`6`

2.1.5 Mono-allelic expression (MAE) dictionary

These parameters are directly used by the `mae` snakemake command. MAE groups are not bound by a minimum number of samples, but require additional information in the sample annotation table.

Parameter	Type	Description	Default/Examples
run	boolean	If true, the module will be run. If false, it will be ignored.	`true`
groups	list	Same as in aberrant expression.	`# see aberrant expression example`
gatkIgnoreHeaderCheck	boolean	If true (recommended), it ignores the header warnings of a VCF file when performing the allelic counts	`true`
padjCutoff	numeric	Same as in aberrant expression.	`0.05`
allelicRatioCutoff	numeric	A number between [0.5, 1) indicating the maximum allelic ratio allele1/(allele1+allele2) for the test to be significant.	`0.8`
addAF	boolean	Whether or not to add the allele frequencies from gnomAD	`true`
maxAF	numeric	Maximum allele frequency (of the minor allele) cut-off. Variants with AF equal or below this number are considered rare.	`0.001`
maxVarFreqCohort	numeric	Maximum variant frequency among the cohort.	`0.05`
qcVcf	character	Full path to the vcf file used for VCF-BAM matching. Refer to *files-to-download*.	`/path/to/qc_vcf.vcf.gz`
qcGroups	list	Same as "groups", but for the VCF-BAM matching	`# see aberrant expression example`

2.1.6 RNA Variant Calling dictionary

Variant Calling originating from RNA-seq data may be useful for researchers who do not have access to variant calls from genomic data. While variant calling from WES and WGS technologies may be more traditional (and reliable), we have found that variant calling from RNA-Seq data can provide additional evidence for the underlying causes of aberrant expression or splicing. The RNA variant calling process uses information from multiple samples (as designated by the `groups` variable) to improve the variant calling process. However, the larger the group size, the more costly the computation is in terms of time and resources. When building the sample annotation table, take this into account. For the most accurate variant calls include many samples in each `DROP_GROUP` group, but in order to speed up computation, separate samples into many groups.

groups list groups that should be executed in this module. If not specified or `null` all groups are used. `- group1`

`- group2`

highQualityVCFs list Filepaths where each item in the list is path to a vcf file. Each vcf file describes known high quality variants, which are used to recalibrate sequencing scores. Refer to *files-to-download* `- known_indels.vcf`

`- known_SNPs.vcf`

dbSNP character Location of the dbSNP `.vcf` file. This improves both recalibrating sequencing scores, as well as variant calling precision. Refer to *files-to-download* `path/to/dbSNP.vcf` repeat_mask character Location of the RepeatMask `.bed` file. Refer to *files-to-download* `path/to/RepeatMask.bed` minAlt numeric Integer describing the minimum required reads that support the alternative allele. We recommend a minimum of 3 if further filtering on your own. 10 otherwise. `3` hcArgs character String describing additional arguments for GATK haplocaller. For expert tuning. `""`

RNA_ID DNA_ID DROP_GROUP RNA_BAM_FILE GENE_COUNTS_FILE GENE_ANNOTATION SPLICE_COUNTS_DIR `======` `======` `================` `================` `============================` `==============` `========================` S10R S10G

BLOOD_AE,BLOOD_AS /path/to/S10R.BAM EXT-1R BLOOD_AE /path/to/externalCounts.tsv.gz gencode34 EXT-2R BLOOD_AE,BLOOD_AS /path/to/externalCounts.tsv.gz gencode34 /path/to/externalCountDir EXT-3R BLOOD_AS /path/to/externalCountDir ====== ====== ================ ================ ============================ ============== ========================

2.2 Files to download

The following files can be downloaded from our public repository.

1. VCF file containing different positions to be used to match DNA with RNA files. The file name is `qc_vcf_1000G_{genome_build}.vcf.gz`. One file is available for each genome build (hg19/hs37d5 and hg38/GRCh38). Download it together with the corresponding `.tbi` file. Indicate the full path to the vcf file in the `qcVcf` key in the mono-allelic expression dictionary. This file is only needed for the MAE module. Otherwise, write `null` in the `qcVcf` key.

2. Text file containing the relations between genes and phenotypes encoded as HPO terms. The file name is `hpo_genes.tsv.gz`. Download it and indicate the full path to it in the `hpoFile` key. The file is only needed in case HPO terms are specified in the sample annotation. Otherwise, write `null` in the `hpoFile` key.

3. For the `rnaVariantCalling` module known high quality variants are needed to calibrate variant and sequencing scores to be used in the `rnaVariantCalling` module in the `highQualityVCF` config parameter. These and the associated `.tbi` indexes can be downloaded for hg19 at our public repository and for hg38 through the Broad Institute's resource bundle.

hg19

- `Mills_and_1000G_gold_standard.indels.hg19.sites.chrPrefix.vcf.gz`
- `1000G_phase1.snps.high_confidence.hg19.sites.chrPrefix.vcf.gz`

hg38

- `Mills_and_1000G_gold_standard.indels.hg38.vcf.gz`
- `Homo_sapiens_assembly38.known_indels.vcf.gz`

We also recommend using the variants from dbSNP which is quite large. You can download them and their associated `.tbi` indexes from NCBI

- follow links for the current version (`human_9606/VCF/00-All.vcf.gz`) or older assemblies (eg. `human_9606_b151_GRCh37p13/VCF/00-All.vcf.gz`)

The repeat masker file is used to filter hard-to-call regions. In general, this removes false-positive calls, however, some targeted and known splicing defects lie within these repeat regions. Understand that this filter is labeled `Mask` in the result VCF files. You can download the repeat mask and associated `.idx` on our public repository. for the `repeat_mask` config parameter.

2.3 Example of RNA replicates

RNA_ID	DNA_ID	DROP_GROUP	RNA_BAM_FILE	DNA_VCF_FILE
S10R_B	S10G	BLOOD	/path/to/S10R_B.BAM	/path/to/S10G.vcf.gz
S10R_M	S10G	MUSCLE	/path/to/S10R_M.BAM	/path/to/S10G.vcf.gz

2.3.1 Example of DNA replicates

RNA_ID	DNA_ID	DROP_GROUP	RNA_BAM_FILE	DNA_VCF_FILE
S20R	S20E	WES	/path/to/S20R.BAM	/path/to/S20E.vcf.gz
S20R	S20G	WGS	/path/to/S20R.BAM	/path/to/S20G.vcf.gz

2.3.2 Example of a multi-sample vcf file

RNA_ID	DNA_ID	DROP_GROUP	RNA_BAM_FILE	DNA_VCF_FILE
S10R	S10G	WGS	/path/to/S10R.BAM	/path/to/multi_sample.vcf.gz
S20R	S20G	WGS	/path/to/S20R.BAM	/path/to/multi_sample.vcf.gz

2.4 Advanced options

A local copy of DROP can be edited and modified for uncovering potential issues or increasing outputs. For example, the user might want to add new plots to the `Summary` scripts, or add additional columns to the results tables. Also, the number of threads allowed for a computational step can be modified.

Note: DROP needs to be installed from a local directory using `pip install -e <path/to/drop-repo>` so that any changes in the code will be available in the next pipeline run Any changes made to the R code need to be updated with `drop update` in the project directory.

The aberrant expression and splicing modules use a denoising autoencoder to correct for sample covariation. This process reduces the fitting space to a dimension smaller than the number of samples N. The encoding dimension is optimized. We recommend the search space to be at most N/3 for the aberrant expression, and N/6 for the aberrant splicing case. Nevertheless, the user can specify the denominator with the parameter `maxTestedDimensionProportion`.

DROP allows that BAM files from RNA-seq from samples belonging to the same *DROP_GROUP* were aligned to different genome assemblies from the same build (eg, some to ucsc and others to ncbi, but all to either hg19 or hg38). If so, for the aberrant expression and splicing modules, no special configuration is needed. For the MAE and rnaVariantCalling module, the different fasta files must be specified as a dictionary in the *genome* parameter of the config file, and, for each sample, the corresponding key of the *genome* dictionary must be specified in the *GENOME* column of the sample annotation. In additon, DROP allows that BAM files from RNA-seq were aligned to one genome assembly (eg ucsc) and the corresponding VCF files from DNA sequencing to another genome assembly (eg ncbi). If so, the assembly of the reference genome fasta file must correspond to the one of the BAM file from RNA-seq.

CHAPTER 3

Pipeline Commands

DROP is Snakemake pipeline, so it is called with the `snakemake` command.

3.1 Dry run

Open a terminal in your project repository. Execute

```
snakemake --cores 1 -n
```

This will perform a *dry-run*, which means it will display all the steps (or rules) that need to be executed. To also display the reason why those rules need to be executed, run

```
snakemake --cores 1 -nr
```

Finally, a simplified dry-run can be achieved by executing

```
snakemake --cores 1 -nq
```

Calling `snakemake --cores 1` without any additional parameters will execute the whole workflow. Snakemake requires you to designate the number of cores when running the `snakemake` command.

3.2 Parallelizing jobs

DROP's steps are computationally heavy, therefore it is a good idea to run them in parallel. Snakemake automatically determines the steps that can be parallelized. The user simply needs to specify the maximum number of cores that Snakemake can take, e.g. for 10 cores:

```
snakemake --cores 10
```

3.3 Executing subworkflows

Every single module can be called independently.

```
snakemake <subworkflow>
```

Subworkflow	Description
`aberrantExpression`	Aberrant expression pipeline
`aberrantSplicing`	Aberrant splicing pipeline
`mae`	Monoalleic expression pipeline
`rnaVariantCalling`	RNA Variant Calling pipeline

An example for calling the aberrant expression pipeline with 10 cores would be

```
snakemake aberrantExpression --cores 10
```

3.4 Rerunning the pipeline

When DROP is updated or jobs fail, the following commands can be used to rerun and troubleshoot.

3.4.1 Unlocking the pipeline

While running, Snakemake *locks* the directory. If, for a whatever reason, the pipeline was interrupted, the directory might be kept locked. Therefore, call

```
snakemake unlock
```

to unlock it. This will call snakemake's `unlock` command for every module

3.4.2 Updating DROP

Every time a project is initialized, a temporary folder `.drop` will be created in the project folder. If a new version of drop is installed, the `.drop` folder has to be updated for each project that has been initialized using an older version. *drop update* will also reset the local project's *Scripts/* directory to match the installed version, so be sure to save any additional scripts or analyses in another location.

To do this run:

```
drop update
```

3.4.3 Skipping recomputation of files

If snakemake is interrupted and restarted, it will continue with the last unsuccessful job in the job graph. If a script is updated with minor change, e.g. when calling `drop update`, all jobs of the modified script and its downstream steps will be rerun. However, in some cases one might want to keep the intermediate files instead and continue with the missing files. In order to do so, first execute

```
snakemake <rule> --touch
```

for whichever rule or module you want to continue the computation. The `--touch` command touches all output files required by the pipeline that have already been computed. Omitting the rule will lead to accessing the complete pipeline. Afterwards, use

```
snakemake unlock
```

to unlock the submodules, so that the jobs that need to be computed can be identified.

CHAPTER 4

Results and Output of DROP

DROP is intended to help researchers use RNA-Seq data in order to detect genes with aberrant expression, aberrant splicing and mono-allelic expression. By simplifying the workflow process we hope to provide easy-to-read HTML files and output files. This section explains the results files. The paths of the output files correspond to the ones from the demo (that can be run with the following code snippet):

```
#install drop
mamba create -n drop_env -c conda-forge -c bioconda drop
conda activate drop_env

mkdir drop_demo
cd drop_demo
drop demo

snakemake -c1
```

4.1 Aberrant Expression

4.1.1 HTML file

Looking at the resulting `Output/html/drop_demo_index.html` we can see the `AberrantExpression` tab at the top of the screen. The Overview tab contains links to the:

- **Counts Summaries for each aberrant expression group**
  - number of local and external samples
  - Mapped reads and size factors for each sample
  - histograms showing the mean count distribution with different conditions
  - expressed genes within each sample and as a dataset
- **Outrider Summaries for each aberrant expression group**

  - aberrantly expressed genes per sample
  - correlation between samples before and after the autoencoder
  - biological coefficient of variation
  - aberrant samples
  - results table
- **Files for each aberrant expression group**
  - **OUTRIDER datasets**
    - Follow the OUTRIDER vignette for individual OUTRIDER object file (ods) analysis.
  - **Results tables**
    - `results.tsv` this text file contains only the significant genes and samples that meet the cutoffs defined in the config file for `padjCutoff` and `zScoreCutoff`

4.1.2 Local result files

Additionally the `aberrantExpression` module creates the file `Output/processed_results/aberrant_expression/{annotation}/outrider/{drop_group}/OUTRIDER_results_all.Rds`. This file contains the entire OUTRIDER results table regardless of significance.

4.2 Aberrant Splicing

4.2.1 HTML file

Looking at the resulting `Output/html/drop_demo_index.html` we can see the `AberrantSplicing` tab at the top of the screen. The Overview tab contains links to the:

- **Counting Summaries for each aberrant splicing group**
  - number of local and external samples
  - number introns/splice sites before and after merging
  - comparison of local and external mean counts
  - histograms showing the junction expression before and after filtering and variability
- **FRASER Summaries for each aberrant splicing group**
  - the number of samples, introns, and splice sites
  - correlation between samples before and after the autoencoder
  - results table
- **Files for each aberrant splicing group**
  - **FRASER datasets (fds)**
    - Follow the FRASER vignette for individual FRASER object file (fds) analysis.
  - **Results tables**
    - `results_per_junction.tsv` this text file contains only significant junctions that meet the cutoffs defined in the config file.

4.2.2 Local result files

Additionally the `aberrantSplicing` module creates the following file `Output/processed_results/aberrant_splicing/results/{annotation}/fraser/{drop_group}/results.tsv`. This text file contains only significant junctions that meet the cutoffs defined in the config file, aggregated at the gene level. Any sample/gene pair is represented by only the most significant junction.

4.3 Mono-allelic Expression

4.3.1 HTML file

Looking at the resulting `Output/html/drop_demo_index.html` we can see the `MonoallelicExpression` tab at the top of the screen. The Overview tab contains links to the:

- **Results for each mae group**
  - number of samples, genes, and mono-allelically expressed heterozygous SNVs
  - a cascade plot that shows additional filters
  - histogram of inner cohort frequency
  - summary of the cascade plot and results table
- **Files for each mae group**
  - **Allelic counts**
    - a directory containing the allelic counts of heterozygous variants
  - **Results data tables of each sample (.Rds)**
    - Rds objects containing the full results table regardless of MAE status
  - **Significant MAE results tables**
    - a link to the results file
    - Only contains significant MAE for the alternative allele results and results that pass the config file cutoffs
- **Quality Control**
  - **QC Overview**
    - For each mae group QC checks for DNA/RNA matching

4.3.2 Local result files

Additionally the `mae` module creates the following files:

- **`Output/processed_results/mae/{drop_group}/MAE_results_all_{annotation}.tsv.gz`**
  - this file contains the MAE results of all heterozygous SNVs regardless of significance
- **`Output/processed_results/mae/{drop_group}/MAE_results_{annotation}.tsv`**
  - this is the file linked in the HTML document and described above
- **`Output/processed_results/mae/{drop_group}/MAE_results_{annotation}_rare.tsv`**

- this file is a subset of `MAE_results_{annotation}.tsv` with only the variants that pass the allele frequency cutoffs. If `add_AF` is set to `true` in config file must meet minimum AF set by `max_AF`. Additionally, the inner-cohort frequency must meet the `maxVarFreqCohort` cutoff

CHAPTER 5

License

CHAPTER 6

Help

In case you have any issues, please open an issue on git.

You can also write an e-mail to yepez@in.tum.de or mumichae@in.tum.de

CHAPTER 7

Quickstart

DROP is available on bioconda. We recommend using a dedicated conda environment. (installation time: ~ 10min) Use *mamba* instead of *conda* as it provides more reliable and faster dependency solving.

```
mamba create -n drop -c conda-forge -c bioconda drop
```

Test installation with demo project

```
mkdir ~/drop_demo
cd ~/drop_demo
drop demo
```

The pipeline can be run using snakemake commands

```
snakemake --cores 1 -n # dryrun
snakemake --cores 1
```

Expected runtime: 25 min

For more information on different installation options, refer to *Installation*.