DROP

May 07, 2021

Contents:

DROP is intended to help researchers use RNA-Seq data in order to detect genes with aberrant expression, aberrant splicing and mono-allelic expression. It consists of three 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`:

We recommend using a dedicated conda environment (here: `drop_env`) for installing drop.

```
conda 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

```
snakemake -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*).

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 >= 1.7
  - bcftools >= 1.7
  - GATK >= 4.0.4
  - 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

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
  - 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.

2.1.1 Global parameters

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 subdirectories 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

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

Parameter	Type	Description	Default/Examples
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

Parameter	Type	Description	Default/Examples
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 dictionary

Parameter	Type	Description	Default/Examples
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`

2.2 Creating the sample annotation table

For a detailed explanation of the columns of the sample annotation, please refer to Box 3 of the DROP manuscript.

Each row of the sample annotation table corresponds to a unique pair of RNA and DNA samples derived from the same individual. An RNA assay can belong to one or more DNA assays, and vice-versa. If so, they must be specified in different rows. The required columns are `RNA_ID`, `RNA_BAM_FILE` and `DROP_GROUP`, plus other module-specific ones (see DROP manuscript).

The following columns describe the RNA-seq experimental setup: `PAIRED_END`, `STRAND`, `COUNT_MODE` and `COUNT_OVERLAPS`. They affect the counting procedures of the aberrant expression and splicing modules. For a detailed explanation, refer to the documentation of HTSeq.

To run the MAE module, the columns `DNA_ID` and `DNA_VCF_FILE` are needed.

In case external counts are included, add a new row for each sample from those files (or a subset if not all samples are needed). Add the columns: `GENE_COUNTS_FILE`, `GENE_ANNOTATON`, `SPLIT_COUNTS_FILE` and `NON_SPLIT_COUNTS_FILE`. See examples below.

In case RNA-seq BAM files belong to different genome assemblies (eg, ncbi, ucsc), multiple reference genome fasta files can be specified. Add a column called *GENOME* that contains, for each sample, the key from the *genome* parameter in the config file that matches its genome assembly (eg, ncbi or ucsc).

The sample annotation file must be saved in the tab-separated values (tsv) format. The column order does not matter. Also, it does not matter where it is stored, as the path is specified in the config file. Here we provide some examples on how to deal with certain situations. For simplicity, we do not include all possible columns in the examples.

2.2.1 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

2.2.2 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

2.2.3 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

2.2.4 External count matrices

In case counts from external matrices are to be integrated into the analysis, the file must be specified in the GENE_COUNTS_FILE column. A new row must be added for each sample from the count matrix that should be included in the analysis. An RNA_BAM_FILE must not be specified. The DROP_GROUP of the local and external samples that are to be analyzed together must be the same. Similarly, the GENE_ANNOTATION of the external counts and the key of the *geneAnnotation* parameter from the config file must match.

RNA_ID	DNA_ID	DROP_GROUP	RNA_BAM_FILE	GENE_COUNTS_FILE	GENE_ANNOTATION
S10R	S10G	BLOOD	/path/to/S10R.BAM		
EXT-1R		BLOOD		/path/to/externalCounts.tsv.gz	gencode34
EXT-2R		BLOOD		/path/to/externalCounts.tsv.gz	gencode34

2.3 Files to download

Two different 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.

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 (see *Other DROP versions*). 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 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 -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 exeucted, run

```
snakemake -nr
```

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

```
snakemake -nq
```

Calling `snakemake` without any parameters will execute the whole workflow.

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
```

If the `--cores` flag is not specified, snakemake will use a single core by default.

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

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. 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

License

MIT License

CHAPTER 5

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 6

Quickstart

DROP is available on bioconda. We recommend using a dedicated conda environment. (installation time: ~ 10min)

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

Expected runtime: 25 min

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