Run manual

Required data
In order to run DROP some data needs to be acquired, some data will be necessary in every run others are optional:

Required:
Gene annotation table(.gtf), can be downloaded at https://www.gencodegenes.org/
Reference genome file, fasta format

Semi-required(at least one of these data formats have to be used):
RNA sequencing data, .BAM files
WGS or WES data, VCF files
Count data, count matrix in tsv format
qc_vcf_1000G_{genome_build}.vcf.gz(for BAM-VCF matching)*
high quality VCF file(for RNA variant calling module)*

Optional:
HPO file for phenotype terms*

*all these files can be found and downloaded form
https://www.cmm.in.tum.de/public/paper/drop_analysis/resource/

Sample annotation
First up, create a sample annotation table. This is a tab separated value(.tsv) file containing all the data samples to be used in the DROP run. This file has multiple headers, they are as follows:

Sample annotation	
RNA_ID	Identifier for the RNA seq sample
RNA_BAM_FILE	Absolute path to RNA bam file
DROP_GROUP	Group sample belongs to
DNA_ID	Identifier for DNA sample
DNA_VCF_FILE	Absolute path to VCF file
Sample information	
PAIRED_END	Boolean describing if the sample comes from paired-end sequencing
STRAND	Yes, no or reverse if the sample is strand specific. Reverse if strand specific and the first read in the pair is opposite strand to the feature and the second read on the same strand

COUNT_MODE	Count mode from the HTSeq module, either IntersectionStrict, Union or IntersectionNotEmpty
COUNT_OVERLAPS	Boolean specifying whether reads overlapping different regions are allowed and counted
Optional modifiers	
INDIVIDUAL_ID	Identifier for an individual(patient) when dealing with replicates
HPO_TERMS	Comma-separated phenotypes encoded as HPO terms
External count settings	
GENE_COUNT_FILE	Absolute path to count data file
GENE_ANNOTATION	Gene annotation key as given in config file
SPLIT_COUNTS_FILE	Path to directory containing split counts
NON_SPLIT_COUNTS_FILE	Path to directory containing non-split counts

While most headers are required to be in the table they don't need a value. As an example on how to fill in the sample annotation table the table from the demo set has been added to this repository. All sample information should be according to the way the samples were sequenced. Two fields to keep in mind are the groups and count mode. In the DROP_GROUP field the possibility is given to have some samples only go through specific parts of the pipeline and be fitted in the autoencoder against specific other files from the group. For example if some samples need to be put through OUTRIDER and FRASER and some only through FRASER you can give the first samples the group names: OUTRIDER and FRASER and the second group FRASER and after specifying this in the config file the samples will go through the right processes.

Count mode is specified for the HTSeq tool(HTSeq link) that is used in DROP and the value determines the way counts in the RNA files are linked to a given gene, we recommend 'Union' from own testing but feel free to look at the documentation at https://htseq.readthedocs.io/en/release_0.11.1/count.html and make your own assessment.

The DNA and RNA IDs are used to identify the used data sample. When using external count data the RNA ID can't be a self given ID, it has to match a sample header in the count file. These headers are most of the time the column headers, with the row headers being the genes. This way you specify in the ID column which sample you want to use from the count file, and thus the row needs to contain the sample ID and the file this ID comes from.

Config file

The config.yaml file is where the user can change the configuration of DROP. A default has been provided which only has to be filed in by the user, with all recommended values already in place, but these values can ofcourse always be edited to the experiment. The first fields are for naming the project and telling DROP the locations of necessary files like the sample annotation table, gene annotation file, reference genome and HPO file.

Next up is the export count section, the count data of the selected groups will be sent to DROP for further testing and algorithm fitting. Make sure no sensitive BAM files are sent.

Next are the run modules, aberrant expression, aberrant splicing, MAE and RNA variant calling. The user can select which modules to run by setting the run value of these modules to either true or false. Then select the groups that have to be run by each module under the groups tab. Aberrant expression and splicing have the implementation field, this determines what will be used for the variation in the data. These are set on the autoencoder by default and it is very much recommended to use this setting as it has been shown to be the best performing method. The other options are PCA and PEER. Furthermore all the modules have a cutoff field to determine what are outliers, these have standard values based on testing and literature but can be adjusted for a more or less strict test.

When MAE or RNA variant calling are used some extra files are needed so check these paths in the module, default values for these files can be found on the DROP repository*.

Singularity container

For this manual we consider that the user uses the singularity container found in this repository, if other installs are used the documentation will be the same by leaving the singularity commands out.

The first step when using DROP is to initiate the pipeline, this step only has to be done at installation, after this all required files will be in the directory.

“”
sudo singularity exec drop.sif drop init
“”

After this your directory should contain a Scripts folder, default config.yaml file, a readme, a Snakefile and a .drop directory(ctrl+h in file editor, ls -a in command line). Now DROP can be used.

To make sure DROP will work you can use the sampleAnnotation command to check if all files are correctly placed in the config file and sample annotation table. This command will only check if the given paths and names are correct so if there are problems with the files itself this command will not find them, but it’s a good way to check if DROP has been set up correctly:

“”
sudo singularity exec drop.sif snakemake sampleAnnotation -c1
“”

This will either return a message saying 100% of the steps were done correctly in which case the config and sample annotation were set up correctly. If not an error message be presented telling what went wrong. The log of this step can be found afterwards in the .snakemake directory under the log folder.

Now everything should be set up so the pipeline can be run. For this you can use the following command:

“”
sudo singularity exec drop.sif snakemake - -cores n
“”

Where n is the amount of cores to be used. There are also possibilities to run the pipeline separately by specifying the module to use, for example:

“”
sudo singularity exec drop.sif snakemake aberrantExpression - -cores 10
“”

Will run the aberrant expression module with 10 cores, but we recommend using the full pipeline as otherwise the overview with the volcano plots will not be generated. In this way if only specific modules have to be run you can check out the others in the config file.

If at any point a run is interrupted the pipeline will be locked, you can unlock the pipeline again by using the command:

“”
sudo singularity exec drop.sif snakemake unlock
“”

The container has a default run option that will run the pipeline with 8 cores, this can be used with:

“”
sudo singularity run drop.sif
“”

Interpreting the results:

Running the container on gearshift

When running the pipeline on the gearshift cluster of the UMCG some other commands need to be added to the container. On the cluster there are no sudo privileges and so the container will by default get onto your home directory. In order to run it in another directory the –mount command (https://apptainer.org/user-docs/master/bind_paths_and_mounts.html) has to be used, the standard container has directories for /groups and /apps built in but if

other directories are to be used specify this in the definition file for the container *conda_environment.def*. So in this case for an init the following command has to be used:

“””
singularity –mount type=bind,src=/groups/,dst=/groups/ –mount type=bind,src=/apps/,dst=/apps/ exec drop init
“””

For any other code this same mount command has to be used.

In order to run DROP as a job the only command that has to be added to the regular batch file is:

“””
singularity –mount type=bind,src=/groups/,dst=/groups/ –mount type=bind,src=/apps/,dst=/apps/ exec snakemake –cores n
“””

Where n is the amount of cores used(same as specified in the batch file). We recommend using 4GBs of memory per core used in this batch file.