
DICAST

Release 1.0

A. Fenn, O. Tsoy, A. Dietrich, T. Faro, F. Rößler

Jan 05, 2023

GET STARTED

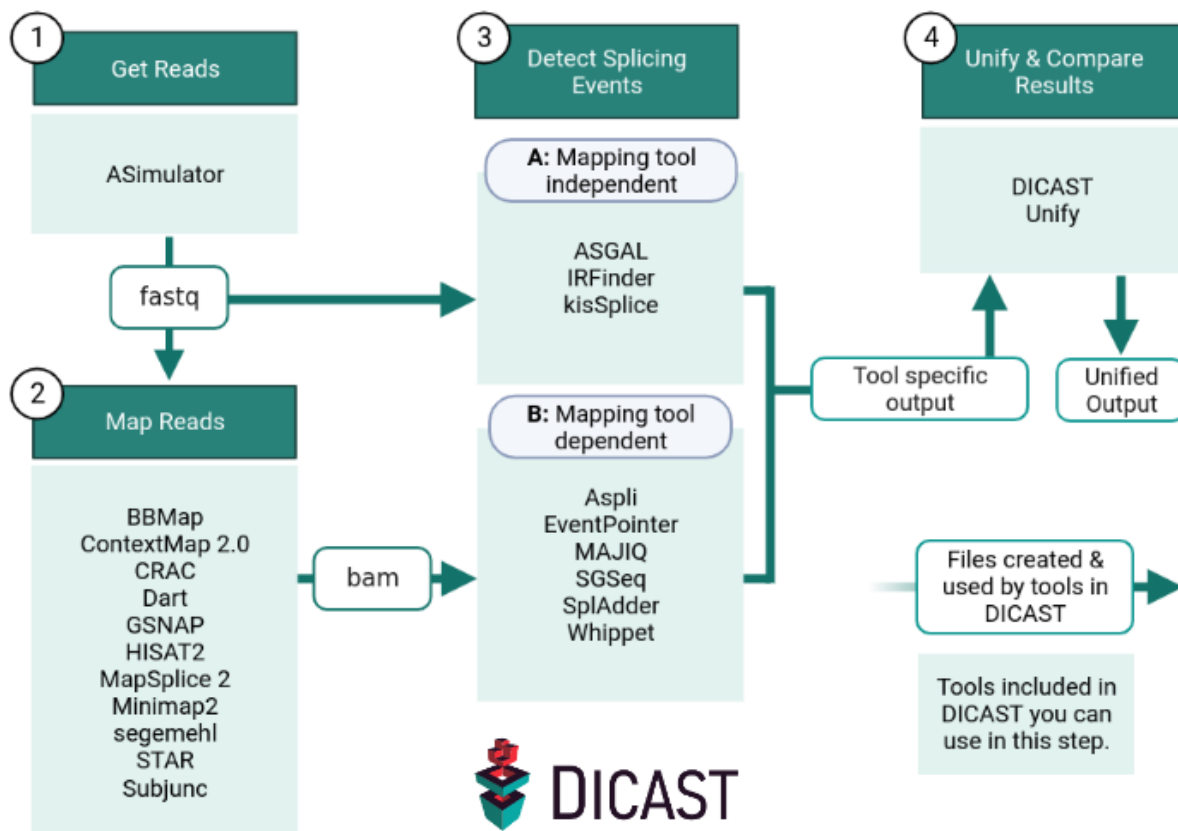
1	What is DICAST?	3
2	When should I use DICAST?	5
3	How do I cite DICAST?	7
3.1	Setup	7
3.1.1	Clone Git Repository	7
3.1.2	Directory Structure	8
3.1.3	Setup docker	10
3.1.4	Install Snakemake	12
3.2	Configuring DICAST	12
3.2.1	Read the full reference here	13
3.2.2	Most frequent changes to configurations:	20
3.2.3	Basic Parameters	21
3.2.4	Input Parameters	21
3.3	Run your analysis	21
3.3.1	Run DICAST via Command Line Interface	21
3.3.2	Run one specific tool via Docker	22
3.3.3	Run DICAST with a graphical user interface (deprecated)	23
3.3.4	DICAST Outputs	27
3.3.5	Workflow	28
3.4	General Information	30
3.5	Mapping tools	32
3.5.1	Mapping Input Files	32
3.5.2	Parameters	33
3.6	Splicing tools	49
3.6.1	Splicing Input Files	49
3.6.2	Parameters	51
3.7	Examples	59
3.7.1	Running multiple mapping tools (E.g., STAR, HISAT2 and bbmap)	60
3.7.2	Running multiple alternative splicing event detection tools (E.g., MAJIQ and Whippet)	61
3.8	FAQ	62
3.9	Uninstalling DICAST	62
3.10	About	62
3.10.1	Development	62
3.10.2	Citation	62
3.10.3	Acknowledgments	63
3.10.4	Contact us	63

The DICAST pipeline was initially designed to benchmark alternative splicing (AS) tools based on simulated “ground truth” reads produced by ASimulator. Here we provide a pipeline for running several mapping and AS event detection tools and evaluate and compare the results. DICAST however is not only suitable for simulated data, but you can also use it for real data.

We hope for this to be an open collaborative effort to represent and benchmark your tools. Please feel free to reach out to us, should your tool already be here and you have some edits to suggest. Also please reach out to us, if your Alternative splicing tool isn't here and you'd like it to be.

To provide a fair baseline while maintaining easy usability, per default we run the tools with their default variables. If you feel like this is not doing your tool justice please contact us. The default parameters can be changed by editing the ENTRYPOINT.sh scripts of each tool.

The tools included here are the most widely used and well maintained among AS event detection tools. If you would like to include your tool in the pipeline please let us know. We hope this collection can be a starting point for future benchmarking approaches and quality control.



WHAT IS DICAST?

DICAST is a collection of alternative splicing event detection tools for analyzing RNA-Seq data. DICAST runs on [Snakemake](#) pipelines and relies on [Docker](#) based containerization. For easy installation and maintenance, we provide docker containers for every integrated tool at [Dockerhub](#)

DICAST can be run as a complete pipeline, starting with simulating RNA-Seq data with ASimulator, mapping the reads to a fasta reference, get information about AS with one or multiple tools and finally visualize and compare the results from different tools with DICAST unify.

Alternatively, you can run one of the same tools as a single docker container without snakemake.

DICAST is available for download on [Github](#).

WHEN SHOULD I USE DICAST?

- If you want to benchmark or compare different mapping and splicing tools with a genome and an annotation.
- If you want to analyze your data with one or more tools included in DICAST and find out which tools give you the events you're interested in.

HOW DO I CITE DICAST?

The preprint citing DICAST is [Alternative splicing analysis benchmark with DICAST](#), is available for review on [bioRxiv](#)

If you use DICAST please cite the preprint as:

Fenn, A.M., Tsoy, O., Faro, T., Roessler, F., Dietrich, A., Kersting, J., Louadi, Z., Lio, C.T., Voelker, U., Baumbach, J. and Kacprowski, T., 2022. Alternative splicing analysis benchmark with DICAST. [bioRxiv](#).

3.1 Setup

DICAST has two main dependencies, Conda and Docker. Setting it up completely however, needs you to build docker images, which takes around 2.5 hours, locally, so it is recommended to get a bit of a headstart. It is faster to just pull them. Either way, should you run DICAST once, the images are cached on your computer and is accessible for later use.

Please follow these steps carefully to set up your working environment.

3.1.1 Clone Git Repository

If you don't have `git`, please install `git` with the following [Install Git](#).

Warning: Where you choose to install this `git` may be limited by “where can docker mount”.

Directories that can be mounted as Docker's mounted volumes have permission based limitations. If you don't feel confident about this section, please talk to your administrator.

Docker permission limitation:

Docker's mounted volumes must hold the permissions: `drwxrwxr-x`. If you're on a linux file system this means that all parent folders of your working directory must be more permissive than `drwxrwsr-x`, except “/”.

This can be ensured with the command for each of the parent directory until “/”.

```
$ chmod a+rX,u+w,g+w <directory hosting the DICAST git>
```

If you have `sudo`, consider `/opt/DICAST/` as your working directory:

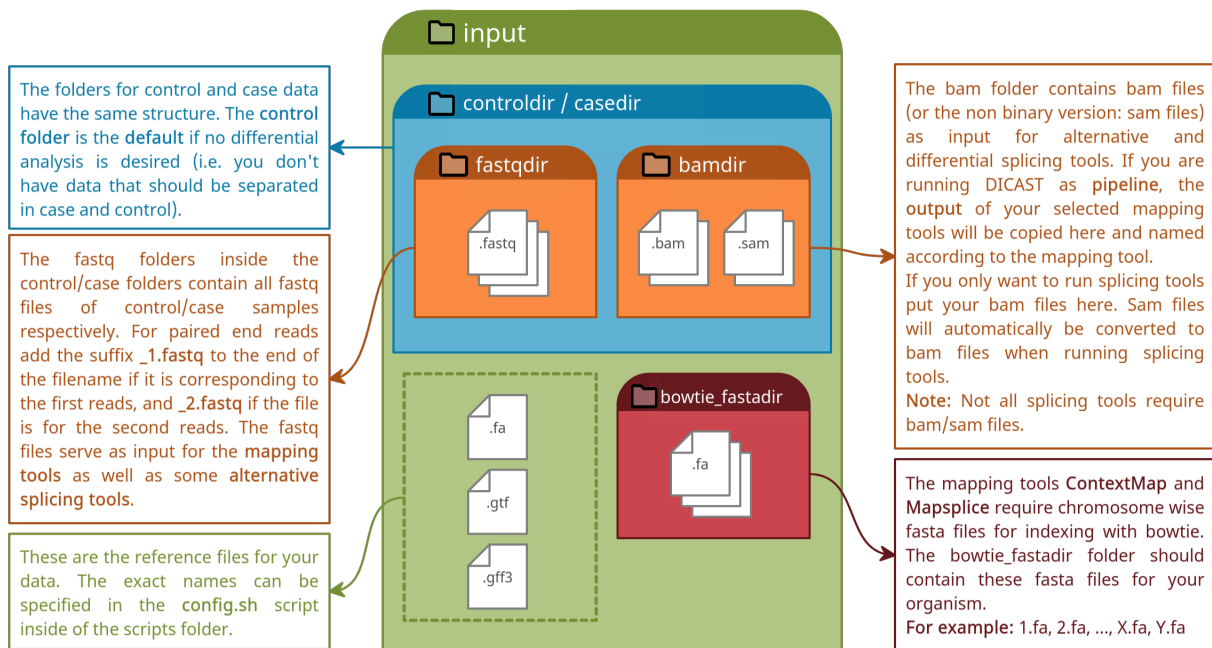
Cloning DICAST's git repository:

Clone our project repository to a directory of your choice. This directory will be considered the working directory for most of the commands listed in this documentation.

```
$ git clone https://github.com/CGAT-Group/DICAST.git
```

This will give you access to the necessary scripts and the *directory structure* hosts an example for the inputs in a directory called “sample_input”. The directory structure within this git is assumed by DICAST, so please don't modify directory names within this working directory.

3.1.2 Directory Structure



Note: Our pipeline allows to run many different tools in the same way. The scripts therefore rely on the directory structure specified here. Please don't rename any directories that are listed here within the git. An output directory is created with your first run. This directory may be renamed.

Example Tree Structure

This is an example for the tree structure when running the pipeline for alternative splicing. Please note that **you only need a .fa and a .gtf file** if you start your analysis with ASimulator, since it will create .fastq files for you. However, some tools need specific input files. Please refer to the respective *tool documentation* for further information.

```
input
├── casedir
│   ├── bamdir
│   └── fastqdir
└── controldir
```

(continues on next page)

(continued from previous page)

```
— bamdir
  — example_bbmap.sam
  — example_contextmap.sam
  — example_crac.sam
  — example_dart.sam
  — example_gsnap.sam
  — example_hisat.sam
  — example_mapsplice.sam
  — example_minimap.sam
  — example_segemehl.sam
  — example_starAligned.out.sam
  — example_subjunc.sam
— fastqdir
  — example_1.fastq
  — example_2.fastq
— bowtie_fastadir
  — 10.fa
  — 11.fa
  — 12.fa
  — 13.fa
  — 14.fa
  — 15.fa
  — 16.fa
  — 17.fa
  — 18.fa
  — 19.fa
  — 1.fa
  — 20.fa
  — 21.fa
  — 22.fa
  — 2.fa
  — 3.fa
  — 4.fa
  — 5.fa
  — 6.fa
  — 7.fa
  — 8.fa
  — 9.fa
  — MT.fa
  — X.fa
  — Y.fa
— example.fa
— example.gff
— example.gtf
```

Input files

The `sample_input` is a template of what the files should look like. Let's however compare this with a real world example. DICAST is not intended to be limited to any specific organism, but for examples, we go with the assembly you can download at [NCBI](#) *homo sapiens*.

`example.fa` : refers to a reference genome `.fna` or `fa`.

`example.gff`: refers to a reference annotation `.gff` or `.gff3`.

`example.gtf`: refers to a reference annotation `.gtf`.

Warning: All files, including references and fastq files must be unzipped, as most of the tools within dicast require them in unzipped form.

Note: If you work with the human genome or would like to just test if DICAST is installed well, check out the script at `initializing-dicast.sh` and execute it with the command `bash initializing-dicast.sh`, to populate your input directory with relevant human references.

Note: `casedir`, is currently unsupported. DICAST was built originally with a design that included tools for differential analysis. It maintains the directory structure in order to expand, to cover differential tools in the future.

`example_*.sam`: DICAST can map your fastq files for you with a mapper of your choice, the results of such mapping will be found here in this directory `bamdir`. If you have already mapped bam/sam files, place them in the `input/control_dir/bamdir`, for DICAST to start from here.

3.1.3 Setup docker

1. Get docker-compose for your system (sudo required)

This requires you to have administrative rights on your computer, or talk to your system administrator about getting `docker` & `docker-compose` on your system and giving you rights to use the `docker` user group.

A. Download Docker

Follow the Docker Engine installation manual for getting Docker, your first step. : <https://docs.docker.com/engine/install/>

B. Post-install Docker steps

To run DICAST in user mode entirely, please fulfill this [post-install](#) step to run docker as a non-root user.

C. Install docker-compose

This while closely related, docker-compose is the last of DICAST's docker dependencies. Follow the installation manual from docker-compose. <https://docs.docker.com/compose/install/#install-compose>

Basic Docker commands

Should you have docker configured on your system, you shouldn't run into a permission error for the following commands.

```
$ docker images
```

to list docker images in on your computer.

```
$ docker ps
```

to list all running containers only.

```
$ docker ps -a
```

to list all running and stopped containers.

```
$ docker --version
```

We support Docker version 19 and above.

```
$ docker-compose --version
```

2. Pull docker images (Not needed with snakemake)

Should the tool you intended to run, not build locally, it's also possible to pull them from DICAST's dockerhub repository at: <https://hub.docker.com/repository/docker/dicastproj/dicast>

```
$ docker pull dicastproj/dicast:tagname
```

3. Build docker images (For Developers)

While the steps described in this section are handled by DICAST's graphical interface, it can also be accessed via command line, for more control.

Build all images

If you intend to use multiple dockers at once you can use our snakemake pipeline, which will take care of building the docker images. If you want to build the dockers manually, we provide a `docker-compose.yml` file which will let you build them yourself. You can use the command the following command to build all images.

```
$ docker-compose -f scripts/Snakemake/docker-compose.yml build
```

If you'd like to edit DICAST's docker-compose file, see the [docker-compose Manual](#).

Build one image

If you only want to build one specific docker image, run the following command to first build some core essential containers:

```
$ docker-compose -f scripts/Snakemake/docker-compose.yml build base conda bowtie star
```

And if you want to build any of the other tools, use the following command:

```
$ docker-compose -f scripts/Snakemake/docker-compose.yml build <tool>
```

Where <tool> needs to be replaced with one or more of the following tools:

bbmap, contextmap, crac, dart, gsnap, hisat, mapssplice, minimap, segemehl, star, subunc, asgal, aspli, eventpointer, irfinder, majiq, sgseq, spladder, whippet

4. Other helpful commands

To gracefully stop a running docker container (If perhaps snakemake's process had to be killed):

```
$ docker stop <docker-container-name/ID>
```

Remove an image (to save space, after your analysis):

```
$ docker rmi -f <image id>
```

3.1.4 Install Snakemake

snakemake is the pipe-lining software that enables your DICAST runs. You can set up snakemake in a conda environment.

If you have never worked with conda before you might want to get conda first: <https://conda.io/projects/conda/en/latest/user-guide/install/index.html>

```
$ # create conda environment from .yaml file with snakemake in it.
$ conda env create -f scripts/snakemake/dicast-snakemake.yaml
$
$ # if you want to use DICAST, activate the "dicast-snakemake" environment
$ conda activate dicast-snakemake
```

If you want to learn more about snakemake, you can check out the snakemake documentation: [snakemake](#).

3.2 Configuring DICAST

Before running DICAST, please take some time to configure it for your first run.

DICAST is best *run with the GUI*, which automates the configuration of *scripts/snakemake/snakemake_config.yaml*, *scripts/config.sh* & *scripts/asevent_config.sh*, for a quick run of DICAST without simulated data, on your experiments.

DICAST can be run *via CLI*, however, this feature is currently in development.

If you'd like to modify the **Simulated dataset**, please modify *scripts/ASimulatorR_config.R* (See *ASimulatorR Parameters*)

If you'd like to run DICAST with just one tool *via docker*, then the files you need to modify are: *scripts/config.sh*, *scripts/asevent_config.sh*

It's recommended to take a closer look at the config files on disk before your first run.

The following files are all the configuration files found in DICAST:

```
scripts/snakemake/snakemake_config.yaml,
scripts/ASimulatoR_config.R,
scripts/config.sh &
scripts/asevent_config.sh.
```

3.2.1 Read the full reference here

Snakemake parameters

NEEDS edit Found in file: *scripts/snakemake/snakemake_config.yaml*

These parameters are either set in the GUI, or if you're running DICAST via cli, these parameters determine properties your DICAST run.

Possible_overwrite_acknowledge:

When running DICAST first, an *output* directory is created. When your run is finished, please rename the output directory, if you want to save this output; so that you don't overwrite outputs with the second run of DICAST.

ASimulatoR files such as *src/ASimulatoR/out/event_annotation.tsv* are also overwritten between runs, if ASimulatoR is run again.

true or *false*

true: DICAST runs uninterrupted.

false: DICAST run is interrupted until *true*

ASimulatoR:

do: *True / False*

Run ASimulatoR with the configs as stored in file: *:scripts/snakemake/snakemake_config*

(See *ASimulatoR config*)

Mapping_tools:

What_tools_to_run: '<insert name of mapping tools to run, separated by spaces>'

pick one of the following *bbmap contextmap crac dart gsnap hisat mapssplice minimap segemehl star subjunc*

Example: to run all tools: 'bbmap contextmap crac dart gsnap hisat mapssplice minimap segemehl star subjunc'

Example: to some two tools: 'minimap star'

Example: to run one tool: 'star'

Alternative_splicing_detection_tools:

What_tools_to_run: '<insert name of Alternative Splicing tools to run, separated by spaces>'

pick one of the following *asgal aspli eventpointer irfinder majiq sgseq spladder whippet*

Example: to run all tools: 'asgal aspli eventpointer irfinder majiq sgseq spladder whippet'

Example: to some two tools: 'eventpointer whippet'

Example: to run one tool: ‘whippet’

ASimulatoR parameters

Found in file: *scripts/ASimulatoR_config.R*

Parameters are also explained in the following git [github/biomedbigdata/ASimulatoR](https://github.com/biomedbigdata/ASimulatoR)

ncores

Number of cores used by ASimulatoR

Within dicast, the max number of cores supplied to Snakemake, is as much is used within DICAST’s pipeline. (See *Snakemake parameters*)

multi_events_per_exon

T or F

Should each exon be treated as a target for only one Alternative Splicing event or would you like to see events like Multiple Exon Skipping events, Alternative Last/First Exon + Exon Skipping events?

probs_as_freq

T or F

Default:

F: if probs_as_freq was FALSE, a random number would be drawn for each event-superset combination and only if it was smaller than 1/9 the AS event would be created

T: The exon supersets are partitioned corresponding to the event_prob parameter.

error_rate

Default: 0.001

In the uniform error model, probability that the sequencer records the wrong nucleotide at any given base.

readlen

Read Length

Default is 76

max_genes

define the number of genes you want to work with. If you want all exons, do not specify this parameter or set it to NULL

seq_depth

Sequencing depth of simulated experiment

num_reps

define, how many groups and samples per group you analyze. Here we create a small experiment with two groups with one sample per group:

as_events

make a list in R with the following set or a subset of the following:

`c('es', 'mes', 'ir', 'a3', 'a5', 'afe', 'ale', 'mee')`

as_combs

Combinations of AS events desired in the simulated dataset.

event_probs

Event probabilities of AS events within the simulated dataset

Tool core parameters

Found in file: *scripts/config.sh*

Note:

If a parameter is recommended as a default. It's for the snakemake workflow to work smooth.
Parameters with this value will be marked with: `recommended to leave at default`

Warning: If a parameter exists in the config files but isn't listed in this reference, please don't change the default on this parameter.

Basic parameters**ncores**

Number of cores or threads that each tool will use. Note when using a snakemake pipeline: the resulting number of cores used is a result of multiplication of ncores and snakemake -j parameter.

Default: 16

workdir recommended to leave at default

Name of the base directory inside the Docker.

Default: /MOUNT

outdir recommended to leave at default

Name of the output directory; should be named after the specific tool that was used (use the `$tool` variable for that).

Default: `$workdir/output/${tool:-unspecific}-output`

read_length

Length of the reads inside the fastq files.

Default: 76

Input Directories

inputdir recommended to leave at default

Base input directory.

Default: \$workdir/input

controlfolder recommended to leave at default

Directory for all needed input files when no differential comparison. Directory for control sample input files when running differential AS event detection.

Default: \$inputdir/controldir

casefolder recommended to leave at default

Directory only for case sample input files in case of differential AS event detection.

Default: \$inputdir/casedir

fastqdir recommended to leave at default

Directory for fastq files.

Currently same as 'controlfastq'

Default: \$controlfolder/fastqdir

bamdir recommended to leave at default

Directory for bam files.

Currently same as 'controlbam'

Default: \$controlfolder/bamdir

samdir recommended to leave at default

Directory for sam files.

Default: \$controlfolder/bamdir

fastadir

Directory for the reference genome file

Default: \$inputdir

gtfdir

Directory for the annotation file file.

Default: \$inputdir

gffdir

Directory for gff file

Default: \$inputdir

Tool specific parameters

bowtie_fastadir

Some tools require chromosome-wise fasta-inputs

Default: `$inputdir/fasta_chromosomes/`

Index Parameters

recompute_index

Force recompute the index even if the index with \$indexname already exists.

Default: `false`

indexname

Basename of the index (without eg. `.1.bt2` for bowtie index).

Default: `${fastaname}_index`

star_index

Folder containing a star index built with the `$gtf` and `$fasta` files (see below), used by: IRFinder, KisSplice, rMATS

Default: `$workdir/index/star_index`

indexdir

Directory of the index.

Default: `$workdir/index/${tool:-unspecific}_index`

ASimulatoR Parameters

asimulator_gtf

Name of the file in the input directory used by ASimulatoR to generate new transcripts

Example: `Homo_sapiens.GRCh38.105.gtf`

Input Parameters

fastaname

Name of the genome reference file (fasta format) inside `$fastadir`.

Example: `Homo_sapiens.GRCh38.dna.primary_assembly.fa`

gtfname

Name of annotation reference file inside `$gffdir`.

Example: `splicing_variants.gtf`

gffname

Name of gff reference file inside `$gffdir`.

Example: `splicing_variants.gff3`

Note: There should be no need to edit `fasta`, `gtf` and `gff` since they just combine other parameters.

fasta

Full path to the reference genome file.

Default: `${fastadir:-unspecific}/${fastaname}`

gtf

Full path to the annotation file.

Default: `${gtfdir:-unspecific}/${gtfname}`

gff

Full path to the gff file.

Default: `${gffdir:-unspecific}/${gffname}`

Basic Mapping Parameters**outname recommended to leave at default**

Base name of the output files. They will usually be prefixed with the fastq file name and suffixed with `.sam`.

Default: `$tool` (the name of the tool creating the output files)

Warning: Something broke while changing the config file? Make sure there is no space between the variable, the equal sign and the value.

Since these files are bash scripts, it is important to mind the syntax rules. E.g., there can't be a whitespace before and after `"=`".

For example: | Wrong: `workdir = "dockers/"` | Right: `workdir="dockers/"`

Alternative splicing tools parameters

Found in file: *scripts/asevent_config.sh*

This config file sets parameters that are specific to AS event detection tools only.

Basic Parameters

transcript

Fasta file for gene transcripts.

Default: `$fasta` (set in `config.sh`, see the *General Parameters*)

star_alignment_files

Path to the folder containing star alignment files (*.SJ.)

Default: `$workdir/output/star-output`

Note:

We support only paired RNA-Seq - fastq files have to be in pairs.

Set the suffixes parameters (including the file extension) for all fastq pairs (e.g. `_1.fastq` and `_2.fastq`).

fastqpair1suffix

Suffix for the first file of the fastq pair.

Example: `_1.fastq`

fastqpair2suffix

Suffix for the second file of the fastq pair.

Example: `_2.fastq`

use_bam_input_files

Determines what kind of input to use: 1 for bam files, 0 for fastq files.

Default: 0

combine_events

Events such as Multiple Exon Skipping should be represented as such, instead of individual exon skipping events.

Default: 1

Warning: Something broke while changing the config file? Make sure there is no space between the variable, the equal sign and the value.

Since these files are bash scripts, it is important to mind the syntax rules. E.g., there can't be a white-space before and after `"="`.

For example: | Wrong: workdir = “dockers/” | Right: workdir=”dockers/”

3.2.2 Most frequent changes to configurations:

Note: The following explanations assume that you use our directory structure as described in *Directory Structure*.

1. *scripts/snakemake/snakemake_config.yaml*

The following are the snakemake parameters that you’re most likely to change for a *CLI run*:

Possible_overwrite_acknowledge:

do: false

change to true. This is set to false after every run to prevent overwriting of output files

Mapping_tools:

What_tools_to_run: ‘<insert name of mapping tools to run, separated by spaces>’

pick one of the following *bbmap contextmap crac dart gsnap hisat mapsplice minimap segemehl star subjunc*

Example: to some two tools: ‘minimap star’

Example: to run all tools: ‘bbmap contextmap crac dart gsnap hisat mapsplice minimap segemehl star subjunc’

Example: to run one tool: ‘star’

Alternative_splicing_detection_tools:

What_tools_to_run: ‘<insert name of Alternative Splicing tools to run, separated by spaces>’

pick one of the following *asgal aspli eventpointer irfinder majiq sgseq spladder whippet*

Example: to run all tools: ‘asgal aspli eventpointer irfinder majiq sgseq spladder whippet’

Example: to some two tools: ‘eventpointer whippet’

Example: to run one tool: ‘whippet’

2. *scripts/config.sh*

The following are basic parameters that you are most likely to change on the GUI and in the file *scripts/config.sh*.

Warning: Since these files are bash scripts, it is important to mind the syntax rules. E.g., there can’t be a whitespace before and after “=”.

3.2.3 Basic Parameters

ncores

if you want to use more cores for each tool within snakemake. (not the same as total cores available for `snakemake -j 2`)

3.2.4 Input Parameters

asimulator_gtf

the genome gtf annotation that you use to simulate the data. Default: 'Homo_sapiens.GRCh38.105.gtf'.

fastaname

the genome reference file. Default: 'Homo_sapiens.GRCh38.dna.primary_assembly.fa'.

gtfname

the genome gtf annotation that you use for mapping and alternative splicing analysis. If you're using ASimulatoR, leave this as ASimulatoR.gtf.

gffname

the genome gff3 annotation that you use for mapping and alternative splicing analysis. If you're using ASimulatoR, leave this as ASimulatoR.gff3.

The reference genome, annotation file and gff3 files could be downloaded from [Ensembl](#).

3.3 Run your analysis

Note: This guide page assumes that you have followed all four pages of the *setup* carefully and that you have *configured DICAST*.

3.3.1 Run DICAST via Command Line Interface

In this section we will explain how to use DICAST to run a whole pipeline on the terminal alone.

Make sure you followed the steps described in the *setup* section carefully.

Change `config.sh` according to your run (see *How to change your config.sh file*)

Before getting started make sure to activate the snakemake conda environment:

```
$ conda activate dicast-snakemake
```

Note: Snakemake is set up to run all tools in parallel, meaning if the pipeline is run unrestricted, it will use all available cores. Be sure to set the number of cores to limit the resources available to Snakemake (explained below).

To run the snakemake pipeline:

- Go to /path/to/DICAST/scripts/snakemake
- Edit snakemake_config.yaml to list the tools you want to run in the corresponding lines.
- Use a snakemake command. For example:

See `snakemake -h`

E.g.:

```
$ snakemake -j 2 -d /opt/DICAST/ -s /path/to/DICAST/scripts/snakemake/Snakefile-cli --configfile /path/
```

Important arguments:

Argument	Explanation
<code>-j, --cores <n></code>	Set the number of cores used by the pipeline.
<code>-s, --snakefile <file></code>	Set the path to the Snakemake file.
<code>-d, --directory <path></code>	Set the path to the working directory (containing the input, scripts folder etc.)
<code>--configfile <file></code>	Set the path to the configuration file which specifies what part of the pipeline to run.

For more information, see the [Snakemake documentation](#).

Warning: This feature has been depreciated and needs another Snakefile, without the `Possible_overwrite_acknowledge` rule. We keep this documentation here, for future support and to show you how the tool works under the hood.

Troubleshooting

- Check log files under `output/<tool>-output/logs/`
- If a run was canceled/exited unexpectedly and the directory is still locked, try running the same snakemake command again with `--unlock` or remove the files from `working_directory/.snakemake/locks/`
- Snakemake itself creates some logging files, they can be found in `working_directory/.snakemake/log/`

3.3.2 Run one specific tool via Docker

Change `config.sh` according to your run (see [How to change your config.sh file](#))

If you have already built the image with `<tool>:<tag>` (see the [docker setup](#)) you can run the following command to run the image and start the tool:

```
$ docker run -v <your mounted folder>:/MOUNT --user $(id -u):$(id -g) <tool>:<tag>
$
$ # Examples:
$ # If you are using our directory structure for your input and are in the dockers directory:
$ # Add the --rm flag to remove container, after run.
$
$ docker run -v $(pwd):/MOUNT --user $(id -u):$(id -g) --rm gsnap:0.1
```

Troubleshooting

- Check Snakemake Output to see which rule failed.
- if the rule that failed was named after a tool, check log files under `output/<tool>-output/logs/` to see where the error was.

3.3.3 Run DICAST with a graphical user interface (deprecated)

Make sure you are in the working directory and that listing your directory looks like the directory structure mentioned in *directory structure*. To run dicast, activate the `dicast-snakemake` conda environment:

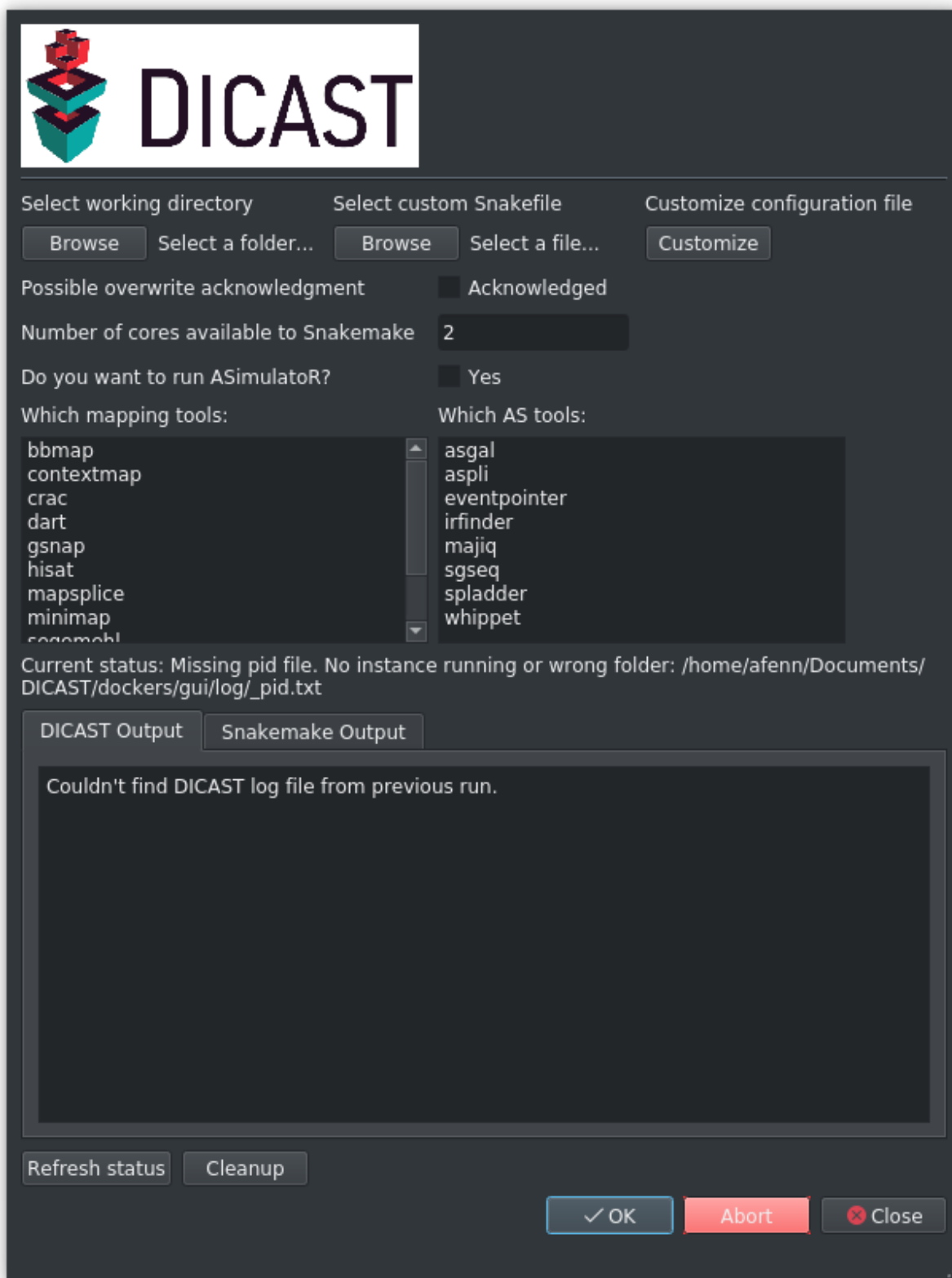
```
$ conda activate dicast-snakemake
```

Your prompt should show you `(dicast-snakemake)`, to show you your conda environment. If so, start DICAST with the following command.

```
$ python gui/dicast.py
```

Warning: DICAST is set to Run ASimulatoR with default values, however, should you wish to tune simulated dataset to your investigative questions, please modify the file `scripts/ASimulatoR_config.R` (See *ASimulatoR Parameters*)

Note: The graphical user interface assumes a X11 rendering system. If you're using `ssh`, please use the `ssh -X` flag, to allow X11 forwarding. If you're not on a Linux machine, locally, find out how to host an X11 daemon for yourself. If you're on a Mac, this could mean installing `Xquartz` on your local machine. If you're running DICAST on your local linux machine, the output of `echo $DISPLAY` should read `:0`; this suggests that you have X11 forwarded correctly to your local machine.



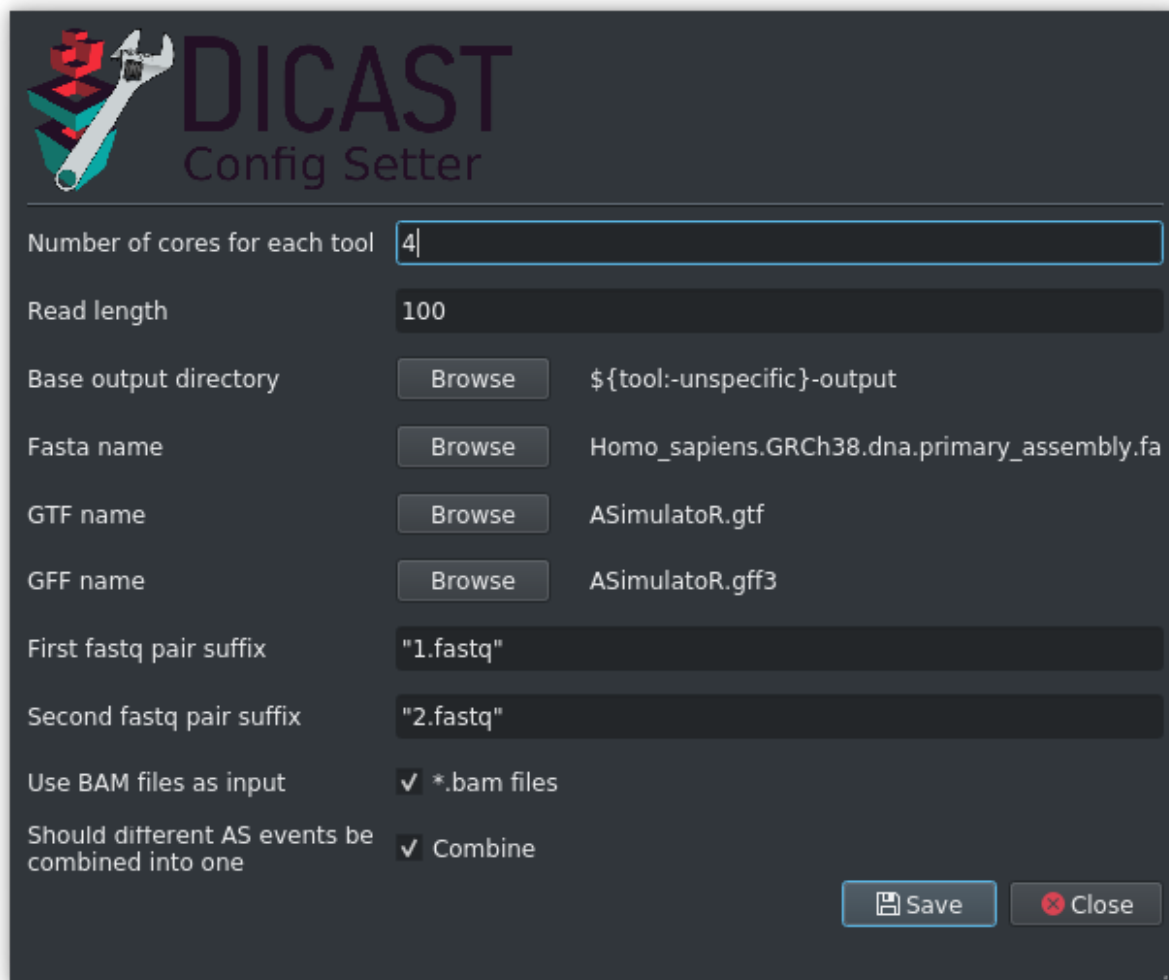
Options	Explanation
Select working directory	The working directory is where you've hosted your DICAST git and it looks like the described <i>directory structure</i> needed to run DICAST. This acts as the root directory for the project.
Select custom Snakefile	By default, DICAST stores its Snakefile under <code>scripts/snakemake/Snakefile</code> . Moving the snakefile is not advised.
Possible overwrite acknowledgement	DICAST writes all outputs to output directory, generated by DICAST. If such a directory exists, maybe rename it, before starting a new run, or you'll lose data. ASimulatoR files such as <code>src/ASimulatoR/out/event_annotation.tsv</code> are also overwritten between runs.
Number of cores available to Snakemake	Total number of cores given to DICAST. Minimum advised: 2
Do you want to run ASimulatoR?	ASimulatoR comes as a part of DICAST. ASimulatoR can be configured by modifying file at <code>src/ASimulatoR/runASimulatoR.R</code>
Referesh status	Is useful, if you connect to DICAST and a previous run is still continuing. DICAST catches previous run and updates your progress.
Abort	Interrupt a running DICAST execution.
Clean up	Cleans up intermediate snakemake files from incomplete runs, use after Abort.
Which Mapping/AS tools:	Select the tools you want to run within DICAST and click okay.
OK	Starts a DICAST run.
Close	Shuts down the GUI, leaving a running session of DICAST, you can connect to next time you start the gui.

DICAST will continue to run, and you can safely close the GUI. Re-opening the GUI connects you back to a running instance of DICAST, if it isn't finished already.

Custom Configuration file:

With this, custom configure your dicast run

Options	Explanation
Number of cored for each tools	Usually seen as <code>-ncores</code> for many tools. gives the option of efficient parallelization when possible.
Read length	An option seen by quite a few tools. supported length <200. Not every tool has been tested with varying read lengths.
Fasta name	Browse to select your reference genome
GTF name	Browse to select your reference annotation, if you're using a real dataset. If you're using ASimulatoR, leave this as <code>ASimulatoR.gtf</code>
GFF name	Browse to select your reference annotation, if you're using a real dataset. If you're using ASimulatoR, leave this as <code>ASimulatoR.gff3</code>
First fastq pair suffix	How do you differentiate paired end reads? how does each fastq file end? "1.fastq"?
Second fastq pair suffix	How do you differentiate paired end reads? how does each fastq file end? "2.fastq"?
Base output directory	Please leave this at default
Use BAM files as inputs	Some tools give you the option of starting from fastq files or from mapped files.
Should different AS events be combined into one	Multiple exon skipping, could be considered one or many AS events. How do you prefer reporting them?



The image shows a software window titled "DICAST Config Setter". The window has a dark grey background. At the top left is a logo consisting of a stylized wrench and a gear. To the right of the logo, the text "DICAST" is written in a large, bold, sans-serif font, and "Config Setter" is written below it in a smaller, regular font. Below the title bar, there are several configuration options, each with a label on the left and a corresponding input field or control on the right. The options are: "Number of cores for each tool" with a text input field containing the number "4"; "Read length" with a text input field containing "100"; "Base output directory" with a "Browse" button and a text field containing "\${tool:-unspecific}-output"; "Fasta name" with a "Browse" button and a text field containing "Homo_sapiens.GRCh38.dna.primary_assembly.fa"; "GTF name" with a "Browse" button and a text field containing "ASimulatorR.gtf"; "GFF name" with a "Browse" button and a text field containing "ASimulatorR.gff3"; "First fastq pair suffix" with a text input field containing "\"1.fastq\""; "Second fastq pair suffix" with a text input field containing "\"2.fastq\""; "Use BAM files as input" with a checked checkbox and the text "*.bam files"; and "Should different AS events be combined into one" with a checked checkbox and the text "Combine". At the bottom right of the window, there are two buttons: "Save" with a floppy disk icon and "Close" with a red 'X' icon.

DICAST
Config Setter

Number of cores for each tool

Read length

Base output directory \${tool:-unspecific}-output

Fasta name Homo_sapiens.GRCh38.dna.primary_assembly.fa

GTF name ASimulatorR.gtf

GFF name ASimulatorR.gff3

First fastq pair suffix

Second fastq pair suffix

Use BAM files as input ☒ *.bam files

Should different AS events be combined into one ☒ Combine

Note: the close button warns you about losing changes even if you saved them. This is a bug, and will be corrected soon.

Troubleshooting

- Check Snakemake Output to see which rule failed.
- if the rule that failed was named after a tool, check log files under `output/<tool>-output/logs/` to see where the error was.

Warning: Aborting a run: Once the dockers begin, they're not under DICAST's control to abort. If you really want to interrupt DICAST, also check for running containers `docker ps` and stop/ kill running containers with `docker stop <container-name>`. Also use the clean up function to clean up an interrupted run.

Interrupting a DICAST run

If you want to interrupt a DICAST run. Click on the **Abort** button and then click on the **Clean up** button. DICAST unfortunately doesn't show you that this is a required step, so until **Clean up** is clicked upon, your next run will not start. Your configurations should stay as you set them last. Click on **Acknowledge overwrite** checkbox and you're all set for the next run.

3.3.4 DICAST Outputs

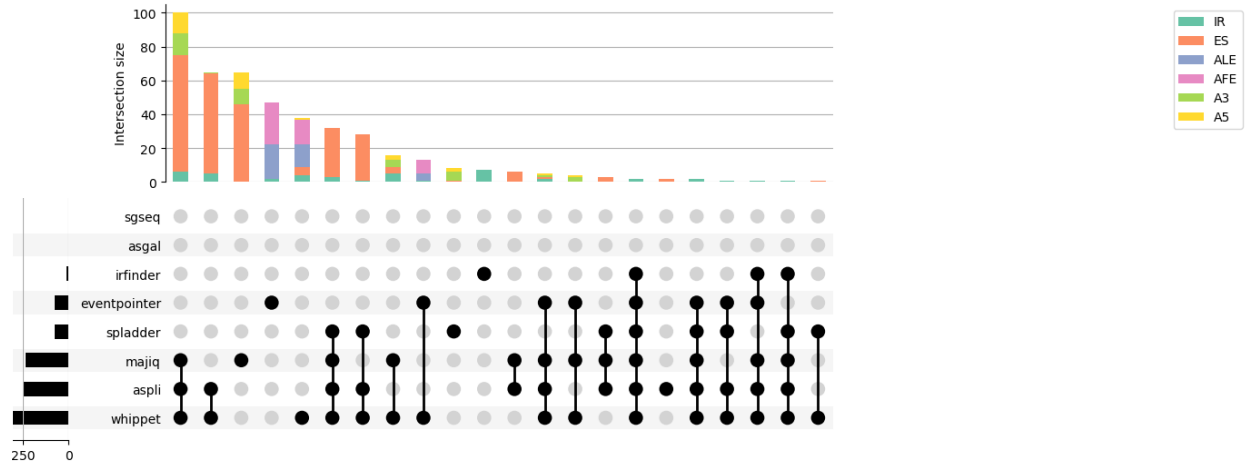
DICAST provides outputs as you would expect them from each Alternative Splicing tool within `output/<astoolname>-output/<Fastq-filename>_output`.

DICAST also provides a `output/<astoolname>-output/<Fastq-filename>_output_dicast_unified` output format for each tool. This is a simple tsv file that hosts all the events found from each tool. We used this to unify the outputs needed to build each of the plots outputted by DICAST.

```
$ output/
$   |— <astoolname>-output
$   |   |— logs
$   |   |— <Fastq-filename>_output
$   |   |— <Fastq-filename>_output_<astoolname>_dicast_unified
$   |— plots
$   |   |— <Fastq-filename>
$   |       |— <mapping tool>-name
$   |       |   |— A3_compare.png
$   |       |   |— A5_compare.png
$   |       |   |— AFE_compare.png
$   |       |   |— ALE_compare.png
$   |       |   |— ES_compare.png
$   |       |   |— IR_compare.png
$   |       |   |— MEE_compare.png
$   |       |   |— MES_compare.png
$   |       |   |— overall_compare.png
$   |       |— unmapped
$   |       |   |— A3_compare.png
$   |       |   |— A5_compare.png
```

```
$ AFE_compare.png
$ ALE_compare.png
$ ES_compare.png
$ IR_compare.png
$ MEE_compare.png
$ MES_compare.png
$ overall_compare.png
```

DICAST also outputs an UpSet plot for each Fastq-filename-mapping_tool combination.



This plot shows the events that were found in common by tools and shows you which tools found these events as well.

When run with **ASimulatoR**, DICAST also outputs precision and recall plots for each Fastq-filename-mapping_tool combination.;

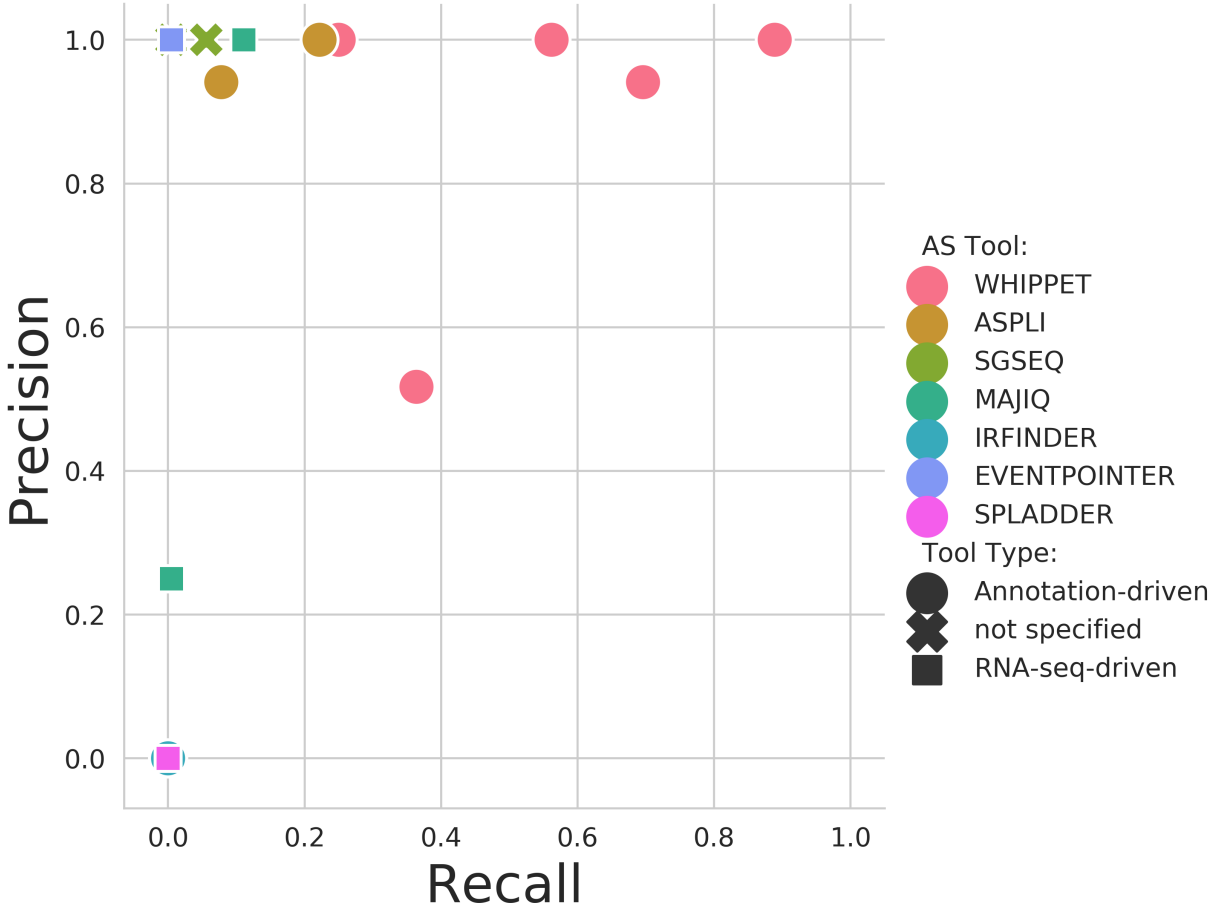
for all events

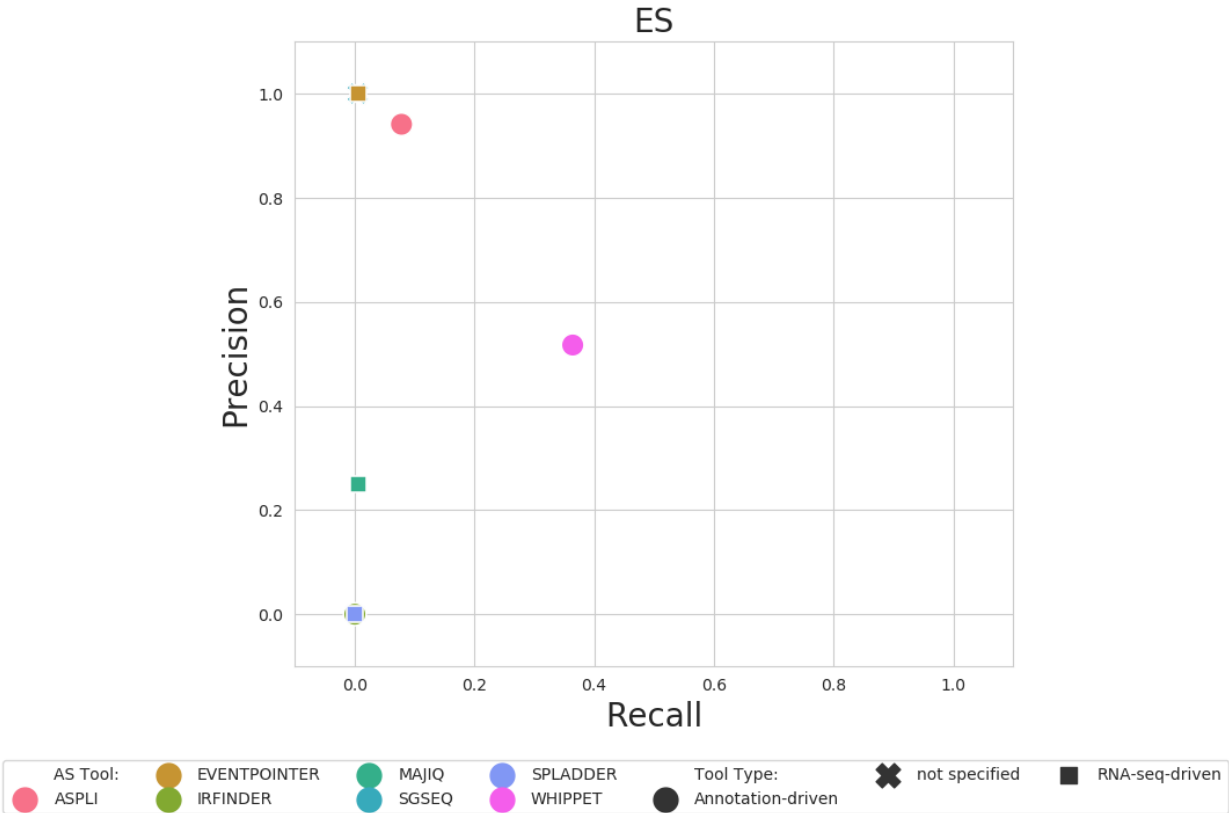
and for each event

3.3.5 Workflow

To run the entire pipeline, you need a reference genome file and a annotation file file of your organism. If you do not want to work with simulated data, you can enter each step with your own data. E.g. you can enter step 2 or 3A with fastq files from your own experiment or step 3B with bam files from your own mapping tool. Please note that not all mapping and splicing detection tools are compatible with each other and have different file requirements (e.g. reference genome, annotation file, gff). For further information, please refer to the tool-specific DICAST documentation.

Warning: We tried our best to unify the input that is required for all tools. This did not work for all tools. When a tool requires custom input you will see a warning like this on the concerning documentation page.



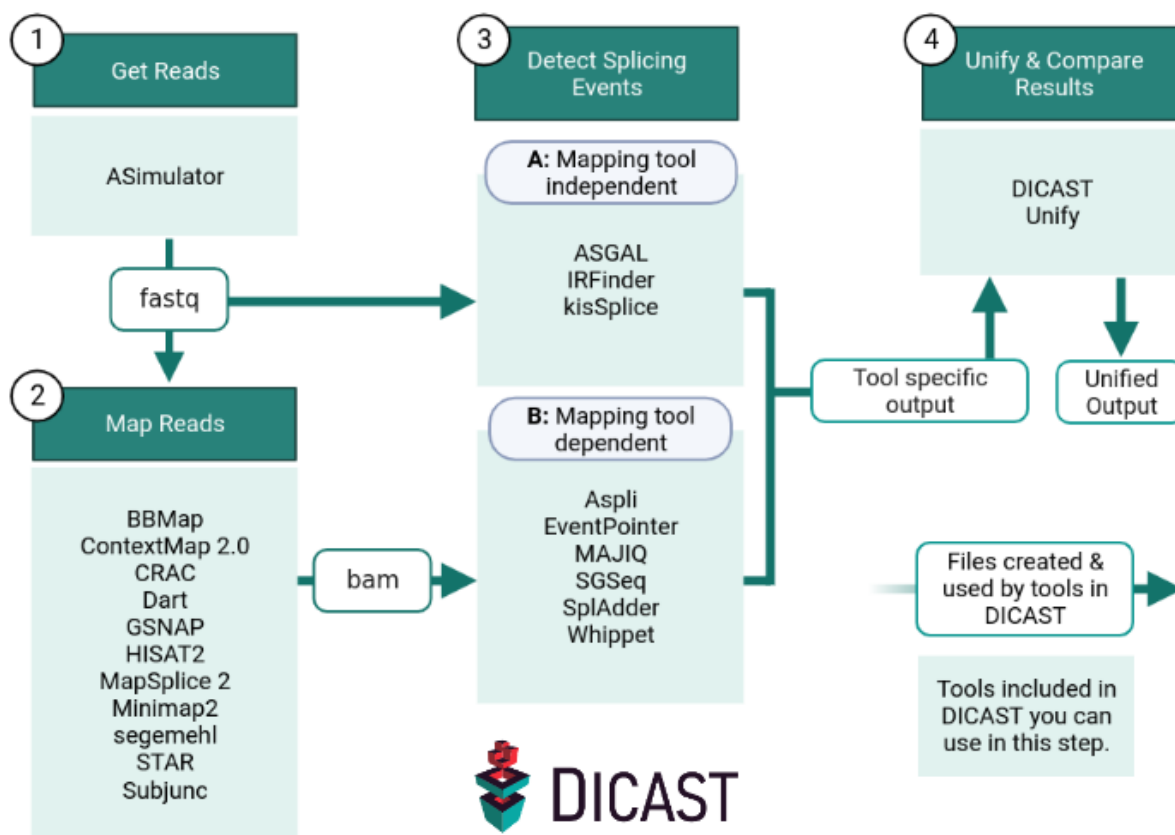


3.4 General Information

Not all mapping and splicing tools are compatible with each other. Please refer to the table below to see which tools you can use together successfully. `fastq` only tools do only work with fastq files and not with bam files and therefore don't depend on a mapping tool.

Table 1: Compatibility

	asgal	aspli	eventpointer	irfinder	majiq	sgseq	spladder	whippet
bbmap	fastq only	Yes	No	Yes	No	No	Yes	Yes
contextmap	fastq only	Yes	Yes	Yes	No	No	Yes	Yes
crac	fastq only	Yes	Yes	Yes	Yes	Yes	Yes	Yes
dart	fastq only	Yes	No	Yes	Yes	No	Yes	Yes
gsnap	fastq only	No	No	Yes	No	No	Yes	Yes
hisat	fastq only	Yes	Yes	Yes	Yes	Yes	Yes	Yes
mapsplice	fastq only	Yes	Yes	Yes	Yes	Yes	Yes	Yes
minimap	fastq only	No	No	Yes	Yes	No	Yes	Yes
segemehl	fastq only	Yes	Yes	Yes	Yes	Yes	Yes	Yes
star	fastq only	Yes	Yes	Yes	Yes	Yes	Yes	Yes
subjunc	fastq only	Yes	Yes	Yes	Yes	Yes	Yes	Yes



3.5 Mapping tools

Note: Most mapping tools need an index file of the reference genome for mapping. The computation of these index files can take a long time. Our mapping tool scripts check if there already is an index for the respective tool and only build it, if it is not found. If you face any index related errors, either set the parameter `$recompute_index=True` or delete the old index to recalculate it.

3.5.1 Mapping Input Files

Tip: The paths assume you are using our suggested *input structure*. Example input files you can find in our *examples section*.

You can find the required input files in the tool-specific documentation.

fastq

Fastq files for paired end mapping. The directories are separated in `controldir` and `casedir`. The `controldir` is the default folder for all analyses. The `casedir` is only used for differential splicing analysis.

```
input/fastq/controldir/*yourFastqFile1*_1.fastq
input/fastq/controldir/*yourFastqFile1*_2.fastq
input/fastq/controldir/*yourFastqFile2*_1.fastq
input/fastq/controldir/*yourFastqFile2*_2.fastq
. . .
```

fasta

The fasta reference for your organism. Mapping tools usually only need it for indexing (see tool specific documentation).

```
input/*yourFastaFile*.fa
```

gtf

annotation reference file.

```
input/*yourGTFfile*.gtf
```

bowtie_fastadir

Only needed by some tools. Chromosome-wise fasta files for your organism to build an index with bowtie.

```
input/bowtie_fastadir/1.fa
input/bowtie_fastadir/2.fa
input/bowtie_fastadir/3.fa
input/bowtie_fastadir/4.fa
. . .
input/bowtie_fastadir/X.fa
input/bowtie_fastadir/Y.fa
```

Optional: Index

Tool specific index file(s). If no index file is found in the index folder it will be built the first time you run the tool. **This might take some time.** If you want to provide your own index please make sure it is in the correct format and file names. Since the index is usually built based on the fasta reference we recommend to name the index based on the fasta reference (default). You can change the `indexname` variable in the config script.

```
index/*toolname*-index/*yourIndexBaseName*
```

3.5.2 Parameters

To provide a fair baseline while maintaining easy usability, per default we run the tools with their default variables. The default parameters can be changed by editing the ENTRYPOINT.sh scripts of each tool. The variables used by mapping ENTRYPOINT.sh scripts can be set in the `config.sh` and `mapping_config.sh` files in the `scripts` folder. For a usual analysis you should not need to change these parameters.

BBMap

BBMap uses a multi kmer seed and extend strategy for read mapping.

BBMap Factsheet

Toolname	<i>bbmap</i>
Version	<i>38.94</i>
License	BBTools Copyright (c) 2014

Required Files

- For indexing only: *fasta*
- For mapping: *fastq*

Compatible splicing tools

- *Aspli*
- *IRFinder*
- *SplAdder*
- *Whippet*

Links

- BBMap [manual](#)
- BBMap publication: [Long Read RNA-seq Mapper](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/$indexname` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/bbmap-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the *src/bbmap/ENTRYPOINT.sh* script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the BBMap [manual](#).

-in	Fastq filename of paired end read 1.	<code>-in *yourFastqFile1_*1.fastq</code>
-in2	Fastq filename of paired end read 2.	<code>-in2 *yourFastqFile1_*2.fastq</code>
-ref	Reference genome in fasta format.	<code>-ref \$fasta</code>
-path	Base name of the index folder and files.	<code>-path \$indexdir/\$indexname</code>
-intronlen	Length of introns.	<code>-intronlen 20</code>
-xstag	Add sam flags to improve compatibility with alternative splicing tools.	<code>-xstag us</code>
-outm	The path to the mapped output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.	<code>-outm \$outdir/\$controlfolder/*yourFastqFile1_ ↪*bbmap.sam</code>
-outu	The path to the unmapped output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.	<code>-outu \$outdir/\$controlfolder/*yourFastqFile1_*. ↪bbmap_unmapped.sam</code>

Known Issues

- issue
- another
- another

ContextMap 2.0

Warning: Make sure to put the file `jre-8u241-linux-i586.tar.gz` in the same folder as the ContextMap Dockerfile (`src/contextmap2/`). You can get the file here: <https://www.oracle.com/java/technologies/javase/javase8u211-later-archive-downloads.html>

ContextMap 2.0 Factsheet

Toolname	<i>contextmap</i>
Version	2.7.9
License	Artistic software License

Required Files

- For mapping: *fastq*, *bowtie_fastadir*

Compatible splicing tools

- *Aspli*
- *EventPointer*
- *IRFinder*
- *SplAdder*
- *Whippet*

Links

- ContextMap 2.0 [manual](#)
- ContextMap 2.0 publication: [ContextMap 2: fasta and accurate context-based RNA-seq mapping](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/$indexname` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/contextmap-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/contextmap/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the ContextMap 2.0 [manual](#).

-reads Comma separated list of file paths to reads in fastq format. One pair of fastq files for paired-end mapping

```
-reads *yourFastqFile1_*1.fastq,*yourFastqFile1_*2.
↪fastq
```

-aligner_name Used aligner (index tool). We use bowtie2.

```
-aligner_name bowtie2
```

-aligner_bin Path to the used aligner. If you use our docker you will not have to worry about it.

```
-aligner_bin /home/biodocker/bin/bowtie2
```

-indexer_bin Path to the indexing tool of the aligner.

```
-indexer_bin /home/biodocker/bin/bowtie2-build
```

-indices Comma separated list to your index files base names.

```
-indices *IndexChromosome1*,*IndexChromosomes2*,
↪*IndexChromosome3*, . . .
```

-genome Directory path with chromosome-wise fasta files.

```
-genome $bowtie_fastadir
```

CRAC

CRAC Factsheet

Toolname	<i>crac</i>
Version	2.5.2
License	GNU General Public License version 3

Required Files

- For mapping: *fastq, fasta*

Compatible splicing tools

- *Aspli*
- *EventPointer*
- *IRFinder*
- *MAJIQ*

- *SGSeq*
- *SplAdder*
- *Whippet*

Links

- CRAC [manual](#)
- CRAC publication: [CRAC: an integrated approach to the analysis of RNA-seq reads](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/$indexname.ssa` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/crac-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/crac/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the CRAC [manual](#).

-i	Base name of the index folder and files.
	<code>-i \$indexdir/\$indexname</code>
-k	Number of k-mers to be used. 22 is the recommended number for human genome.
	<code>-k 22</code>
-r	Space separated list of file paths to reads in fastq format. One pair of fastq files for paired-end mapping
	<code>-r *yourFastqFile1_*1.fastq *yourFastqFile1_*2. ↪fastq</code>
-o	The path to the mapped output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.
	<code>-o \$outdir/\$controlfolder/*yourFastqFile1_*crac.sam</code>
	—detailed-sam Return a detailed sam file as output.
--stranded	Reads are from a strand specific RNA-seq protocol.

Dart

Dart Factsheet

Toolname	<i>dart</i>
Version	<i>1.4.6</i>
License	GNU General Public License version 2

Required Files

- For mapping: *fastq, fasta*

Compatible splicing tools

- *Aspli*
- *IRFinder*
- *MAJIQ*
- *SplAdder*
- *Whippet*

Links

- Dart [manual](#)
- Dart publication: [DART: a fast and accurate RNA-seq mapper with a partitioning strategy](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/$indexname.sa` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/dart-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/dart/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the Dart [manual](#).

-i Base name of the index folder and files.

```
-i $indexdir/$indexname
```

-f Fastq filename of paired end read 1.

```
-f *yourFastqFile1_*1.fastq
```

-f2

Fastq filename of paired end read 2.

```
-f2 *yourFastqFile1_*2.fastq
```

-o

The path to the **mapped** output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
-o $outdir/$controlfolder/*yourFastqFile1_*dart.sam
```

GSNAP

GSNAP Factsheet

Toolname	<i>gsnap</i>
Version	2020-03-12
License	Apache Licence 2.0

Required Files

- For mapping: *fastq, fasta*

Compatible splicing tools

- *IRFinder*
- *SplAdder*
- *Whippet*

Links

- GSNAP [manual](#)
- GSNAP publication: [Fast and SNP-tolerant detection of complex variants and splicing in short reads](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/$indexname/$indexname.contig` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/gsnap-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/gsnap/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the GSNAP [manual](#).

--db Base name of the index folder and files.

```
--db $indexdir/$indexname
```

—dir Base folder of the index files.

```
--dir $indexdir
```

—output-file The path to the **mapped** output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
--output-file $outdir/$controlfolder/*yourFastqFile1_*gsnap.sam
```

—format Define output format (one of sam, m8).

```
--format sam
```

—force-xs-dir Add sam flags to improve compatibility with alternative splicing tools.

```
--force-xs-dir us
```

—nthreads Number of threads to be used during the computation

```
--nthreads $ncores
```

reads

After all other options call space separated list of file paths to reads in fastq format. One pair of fastq files for paired-end reads.

```
*yourFastqFile1_*1.fastq *yourFastqFile1_*2.fastq
```

HISAT2

HISAT2 Factsheet

Toolname	<i>hisat</i>
Version	<i>2.2.1</i>
License	GNU General Public License version 3

Required Files

- For mapping: *fastq, fasta, gtf*

Compatible splicing tools

- *Aspli*
- *EventPointer*

- *IRFinder*
- *MAJIQ*
- *SGSeq*
- *SplAdder*
- *Whippet*

Links

- [HISAT2 manual](#)
- HISAT2 publication: [Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/${indexname}.4.ht2` and `$indexdir/${gtfname}_splicesites.txt` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/hisat-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/hisat/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the [HISAT2 manual](#).

--x Base name of the index folder and files.

```
--x $indexdir/$indexname
```

-1 Fastq filename of paired end read 1.

```
-1 *yourFastqFile1_*1.fastq
```

-2 Fastq filename of paired end read 2.

```
-2 *yourFastqFile1_*2.fastq
```

-S The path to the **mapped** output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
-S $outdir/$controlfolder/*yourFastqFile1_*hisat.  
↪sam
```

—known-splicesite-infile Provide a list of known splice sites.

```
--known-splicesite-infile $indexdir/$indexname/splicesites.txt
```

-q Activate quiet mode so only error messages are printed.

MapSplice 2

MapSplice 2 Factsheet

Toolname	<i>mapsplice</i>
Version	<i>latest conda version</i>
License	MapSplice Copyright (C) 2012-2013

Required Files

- For mapping: *fastq*, *bowtie_fastadir*, *gtf*

Compatible splicing tools

- *Aspli*
- *EventPointer*
- *IRFinder*
- *MAJIQ*
- *SGSeq*
- *SplAdder*
- *Whippet*

Links

- MapSplice 2 [manual](#)
- MapSplice 2 publication: [MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if *\$indexdir/\$indexname.rev.2.ebwt* exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set *\$recompute_index=true* in *scripts/mapping_config.sh*. If you want to use your own precomputed index file copy it to *index/mapsplice-index/* and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/mapsplICE/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the MapSplice 2 [manual](#).

-c Directory path with chromosome-wise fasta files.

```
-c $bowtie_fastadir
```

-x Base name of the index folder and files.

```
-x $indexdir/$indexname
```

—gene-gtf The path to the gene annotation file in GTF format for annotation of fusion junctions.

```
--gene-gtf $gtf
```

-o The path to the directory for the **mapped** output in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
-o $outdir/$controlfolder/*yourFastqFile1_  
↪*mapsplICE
```

-p Number of threads to be used during the computation

```
-p $ncores
```

-1 Fastq filename of paired end read 1.

```
-1 *yourFastqFile1_*1.fastq
```

-2 Fastq filename of paired end read 2.

```
-2 *yourFastqFile1_*2.fastq
```

Minimap2

Minimap2 Factsheet

Toolname	<i>minimap</i>
Version	<i>latest conda version</i>
License	MIT Licence

Required Files

- For mapping: *fastq, fasta*

Compatible splicing tools

- *IRFinder*
- *MAJIQ*

- [SplAdder](#)
- [Whippet](#)

Links

- Minimap2 [manual](#)
- Minimap2 publication: [Minimap2: pairwise alignment for nucleotide sequences](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$indexdir/$indexname` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/minimap-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/minimap/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the Minimap2 [manual](#).

- a** Generate CIGAR and provide output in sam format.
- o** The path to the **mapped** output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
-o $outdir/$controlfolder/*yourFastqFile1_*minimap.  
↪sam
```

- t** Number of threads to be used during the computation

```
-t $ncores
```

index

Base name of the index folder and files.

```
$indexdir/$indexname
```

reads

After all other options call space separated list of file paths to reads in fastq format. One pair of fastq files for paired-end reads.

```
*yourFastqFile1_*1.fastq *yourFastqFile1_*2.fastq
```


segemehl

segemehl Factsheet

Toolname	<i>segemehl</i>
Version	<i>0.3.4</i>
License	GNU General Public License version 3

Required Files

- For mapping: *fastq, fasta*

Compatible splicing tools

- *Aspli*
- *EventPointer*
- *IRFinder*
- *MAJIQ*
- *SGSeq*
- *SplAdder*
- *Whippet*

Links

- [segemehl manual](#)
- [segemehl publication: Fast Mapping of Short Sequences with Mismatches, Insertions and Deletions Using Index Structures](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if *\$indexdir/\$indexname* exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set *\$recompute_index=true* in *scripts/mapping_config.sh*. If you want to use your own precomputed index file copy it to *index/segemehl-index/* and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/segemehl/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the segemehl [manual](#).

-d	Reference genome in fasta format.	<code>-d \$fasta</code>
-q	Fastq filename of paired end read 1.	<code>-q *yourFastqFile1_*1.fastq</code>
-q	Fastq filename of paired end read 2.	<code>-q *yourFastqFile1_*2.fastq</code>
-i	Base name of the index folder and files.	<code>-i \$indexdir/\$indexname</code>
		—splits Use split reads alignment
-o	The path to the mapped output file in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.	<code>-o \$outdir/\$controlfolder/*yourFastqFile1_ ↪ *segemehl.sam</code>
-t	Number of threads to be used during the computation	<code>-t \$ncores</code>

STAR

STAR Factsheet

Toolname	<i>star</i>
Version	2.7.5
License	MIT Licence

Required Files

- For mapping: *fastq, fasta, gtf*

Compatible splicing tools

- *Aspli*
- *EventPointer*
- *IRFinder*
- *MAJIQ*

- *SGSeq*
- *SplAdder*
- *Whippet*

Links

- STAR [manual](#)
- STAR publication: [STAR: ultrafast universal RNA-seq aligner](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if `$star_index/$indexname/genomeParameters.txt` exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set `$recompute_index=true` in `scripts/mapping_config.sh`. If you want to use your own precomputed index file copy it to `index/star-index/` and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the `src/star/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the STAR [manual](#).

--sjdbGTFfile The path to the gene annotation file in GTF format for annotation of fusion junctions.

```
--sjdbGTFfile $gtf
```

—readFilesIn Space separated list of file paths to reads in fastq format. One pair of fastq files for paired-end mapping

```
--readFilesIn *yourFastqFile1_*1.fastq *yourFastqFile1_*2.fastq
```

—genomeDir Base name of the index folder and files.

```
--genomeDir $indexdir/$indexname
```

—outFileNamePrefix The path to the directory for the **mapped** output in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
--outFileNamePrefix $outdir/$controlfolder/*yourFastqFile1_*star
```

—runTreadN Number of threads to be used during the computation

```
--runTreadN $ncores
```

—twopassMode Basic 2-pass mapping, with all 1st pass junctions inserted into the genome indices on the fly

```
--twopassMode Basic
```

—outSAMstrandField Add strand derived from the intron motif.

```
--outSAMstrandField intronMotif
```

—outSAMattributes Add sam flags to improve compatibility with alternative splicing tools.

```
--outSAMattributes us
```

Subjunc

Subjunc Factsheet

Toolname	<i>subjunc</i>
Version	<i>2.0.0</i>
License	GNU General Public License version 3

Required Files

- For mapping: *fastq*, *fasta*

Compatible splicing tools

- *Aspli*
- *EventPointer*
- *IRFinder*
- *MAJIQ*
- *SGSeq*
- *SplAdder*
- *Whippet*

Links

- Subjunc [manual](#)
- Subjunc publication: [The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote](#)

Indexing

Note: Indexing might take some time but only has to be run once per fasta file. Make sure to reuse already computed indices if possible.

DICAST will check if *\$indexdir/\$indexname.reads* exists. If there is no index it will be automatically built. If you want to rebuild the index anyway set *\$recompute_index=true* in *scripts/mapping_config.sh*. If you want to use your own precomputed index file copy it to *index/subjunc-index/* and make sure the index is complete and named appropriately and according to the parameters set in the config files. We recommend including the name of the fasta file in the index name to avoid overwriting. Per default this is already the case and **no parameter changes are needed**.

Parameters

These are the default parameters set in the *src/subjunc/ENTRYPOINT.sh* script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the Subjunc [manual](#).

-i	Base name of the index folder and files.	<code>-i \$indexdir/\$indexname</code>
-r	Fastq filename of paired end read 1.	<code>-r *yourFastqFile1_*1.fastq</code>
-R	Fastq filename of paired end read 2.	<code>-R *yourFastqFile1_*2.fastq</code>
-o	The path to the directory for the mapped output in sam format. The output will be separated into case and control folder based on the basefolder of the according fastq file.	<code>-o \$outdir/\$controlfolder/*yourFastqFile1_*subjunc</code>
-T	Number of threads to be used during the computation	<code>-T \$ncores</code>

—SAMoutput Return a sam file as output.

3.6 Splicing tools

Warning: Currently **only alternative splicing event detection is supported**. Differential splicing tools are coming soon. The differential splicing function of tools which are able to compute both alternative and differential splicing the differential mode is still in beta.

For splicing tools we differentiate between alternative and differential splicing tools. Some tools are able to compute both. Differential splicing tools compute alternative splicing for two conditions (e.g. case and control) and the files should be separated as indicated by our input directory structure. For alternative splicing analysis “control” is the default.

3.6.1 Splicing Input Files

Tip: The paths assume you are using our suggested *input structure*. Example input files you can find in our *examples section*.

You can find the required input files in the tool-specific documentation.

fastq

Fastq files for pair 1 and 2 fastq files stored in `$fastqdir`, identified by the suffix `$fastqpair1suffix` and `$fastqpair2suffix` respectively. Not all splicing tools work with fastq files. The path variables can be

found in *scripts/config.sh* and *scripts/asevent_config.sh*. For differential splicing the files need to be separated in *controldir* and *casedir*

```
# Fastq file paths
# Assumed variable settings:
#   $fastqdir=input/fastq  ## in config.sh
#   $fastqpair1suffix="_1.fastq"  ## in asevent_config.sh
#   $fastqpair2suffix="_2.fastq"  ## in asevent_config.sh
# Replace the text between the stars "...*" with your file names

input/controldir/fastq/*yourFastqFile1*_1.fastq
input/controldir/fastq/*yourFastqFile1*_2.fastq
input/controldir/fastq/*yourFastqFile2*_1.fastq
input/controldir/fastq/*yourFastqFile2*_2.fastq
. . .
```

bam

Bam files created by a mapping tool of your choice. When DICAST is run as a pipeline, these will be created by the selected mapping tool(s).

```
input/controldir/fastq/*yourFastqFile1*_1.fastq
```

fasta:

The name of the reference fasta file. The path variable can be found in *scripts/config.sh*.

```
# Fasta files paths
# Replace the text between the stars "...*" with your file name

input/*yourFastaFile*.fa
```

transcript

The name of the fasta file for gene transcripts. The path variable can be found in *scripts/asevent_config.sh*.

```
# Assumed variable settings:
#   $inputdir=input  ## in config.sh

input/*yourTranscriptFasta*.fasta
```

gtf

Gene annotation file in GTF format.

```
# Replace the text between the stars "...*" with your file name

input/*yourGTFfile*.gtf
```

gff

Gene annotation file in GFF format.

```
# Replace the text between the stars "...*" with your file name

input/*yourGFFfile*.gff
```

3.6.2 Parameters

To provide a fair baseline while maintaining easy usability, per default we run the tools with their default variables. The default parameters can be changed by editing the ENTRYPOINT.sh scripts of each tool. The variables used by mapping ENTRYPOINT.sh scripts can be set in the `config.sh` and `asevent_config.sh` files in the `scripts` folder. For a usual analysis you should not need to change these parameters.

ASGAL

Warning: ASGAL requires the variables `$fastqpair1suffix` and `$fastqpair2suffix` to be set in the `scripts/asevent_config.sh` file.

ASGAL Factsheet

Toolname	<i>asgal</i>
Version	<i>1.1.6</i>
License	GNU General Public License version 3

Required Files

- *fastq* , *fasta* , *gtf*

Links

- ASGAL [manual](#)
- ASGAL publication: [ASGAL: aligning RNA-Seq data to a splicing graph to detect novel alternative splicing events](#)

Parameters

These are the default parameters set in the `src/asgal/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the [ASGAL manual](#).

--multi Set multi option.

-g Reference genome in fasta format.

`-g $fasta`

-a The path to the gene annotation file in GTF format for annotation of fusion junctions.

`-a $gtf`

-t Transcript file.

`-t $transcript`

-s Fastq filename of paired end read 1.

```
-s *yourFastqFile1_*1.fastq
```

-s2

Fastq filename of paired end read 2.

```
-s2 *yourFastqFile1_*2.fastq
```

-o

Output directory. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
-o $outdir
```

-@

Number of threads to be used during the computation

```
-@ $ncores
```

--allevents

Report all events, not only novel ones.

Aspli

Note: Aspli can be used to calculate differential splicing as well as only alternative-splicing events. If you want to perform differential analysis set `differential=1` in the `/scripts/asevent_config.sh` config file. Otherwise set `differential=0`.

Aspli Factsheet

Toolname	<i>aspli</i>
Version	<i>1.12.0</i>
License	GNU General Public License version 3

Required Files

- *gtf*, *bam*

Links

- Aspli [manual](#)
- Aspli publication: [ASpli: integrative analysis of splicing landscapes through RNA-Seq assays](#)

Note: Aspli is an R package. Therefore our ENTRYPOINT.sh script for Aspli calls an R script to run the tool. The parameters listed here are the parameters given to the R script.

Parameters

These are the default parameters set in the `src/aspli/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the Aspli [manual](#).

--gtf	The path to the gene annotation file in GTF format for annotation of fusion junctions.
	<code>--gtf \$gtf</code>
--cores	Number of threads to be used during the computation
	<code>--cores \$ncores</code>
--readLength	Length of reads.
	<code>--readLength \$read_length</code>
--out	Output directory. The output will be separated into case and control folder based on the basefolder of the according bam file. If you are running the DICAST pipeline to compare different mapping tools this will include the name of the mapping tool of the used bam file.
	<code>--out \$outdir</code>
--differential	1 to run differential analysis, 0 otherwise.
	<code>--differential \$differential</code>

EventPointer

Note: EventPointer can be used to calculate differential splicing as well as only alternative-splicing events. If you want to perform differential analysis set `differential=1` in the `/scripts/asevent_config.sh` config file. Otherwise set `differential=0`.

EventPointer Factsheet

Toolname	<i>eventpointer</i>
Version	<i>2.4.0</i>
License	<i>Artistic software License 2</i>

Required Files

- *gtf*, *bam*

Links

- EventPointer [manual](#)
- EventPointer publication: [EventPointer: an effective identification of alternative splicing events using junction arrays](#)

Note: EventPointer is an R package. Therefore our ENTRYPOINT.sh script for EventPointer calls an R script to run the tool. The parameters listed here are the parameters given to the R script.

Parameters

These are the default parameters set in the `src/eventpointer/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the EventPointer [manual](#).

--gtf The path to the gene annotation file in GTF format for annotation of fusion junctions.

```
--gtf $gtf
```

--cores Number of threads to be used during the computation

```
--cores $ncores
```

--out Output directory. The output will be separated into case and control folder based on the basefolder of the according bam file. If you are running the DICAST pipeline to compare different mapping tools this will include the name of the mapping tool of the used bam file.

```
--out $outdir
```

--bamfolder Location of bam files.

```
--bamfolder $controlfolder
```

--differential 1 to run differential analysis, 0 otherwise.

```
--differential $differential
```

IRFinder

IRFinder Factsheet

Toolname	<i>irfinder</i>
Version	<i>1.3.1</i>
License	MIT Licence

Required Files

- *fastq* , *fasta* , *gtf*

Links

- IRFinder [manual](#)
- IRFinder publication: [IRFinder: assessing the impact of intron retention on mammalian gene expression](#)

Note: IRFinder can use both fastq and bam files. To use bamfiles please set the parameter `$use_bam_input_files=1`, and `=0` to use fastq files in the `as_config.sh` script.

Parameters

These are the default parameters set in the `src/irfinder/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the IRFinder [manual](#).

-r Base folder of the index files.

```
-r $indexdir
```

-d Output directory. The output will be separated into case and control folder based on the basefolder of the according fastq file.

```
-d $outdir
```

reads

After all other options call space separated list of file paths to reads in fastq format. One pair of fastq files for paired-end reads.

```
*yourFastqFile1_1.fastq *yourFastqFile1_2.fastq
```

MAJIQ

Note: MAJIQ can be used to calculate differential splicing as well as only alternative-splicing events. If you want to perform differential analysis set `differential=1` in the `/scripts/asevent_config.sh` config file. Otherwise set `differential=0`.

MAJIQ Factsheet

Toolname	<i>majiq</i>
Version	2.3
License	Academic License

Required Files

- *gff*, *bam*

Links

- MAJIQ [manual](#)
- MAJIQ publication: [A new view of transcriptome complexity and regulation through the lens of local splicing variations](#)

Parameters

These are the default parameters set in the *src/majiq/ENTRYPOINT.sh* script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the MAJIQ [manual](#).

build reference

The path to the gene annotation file in GFF format.

`$gff`

-c MAJIQ config file (built based on DICAST config parameters in ENTRYPOINT.sh)

`-c $config`

-j Number of threads to be used during the computation

`-j $ncores`

-o Output directory with majiq build output.

`-o $outdir/$outdir_name/build`

maqiq psi

Run MAJIQ in psi mode with files built from gff as input.

-j Number of threads to be used during the computation

`-j $ncores`

-o Output directory with psi output. Used to build splicegraph with voila.

`-o $outdir/$outdir_name/psi`

-n Run with bam files as input.

`-n "BAM"`

SGSeq

SGSeq Factsheet

Toolname	<i>sgseq</i>
Version	<i>1.24.0</i>
License	Artistic software License 2

Required Files

- *gff*, *bam*

Links

- [SGSeq manual](#)

- SGSeq publication: [Prediction and Quantification of Splice Events from RNA-Seq Data](#)

Note: SGSeq is an R package. Therefore our ENTRYPOINT.sh script for SGSeq calls an R script to run the tool. The parameters listed here are the parameters given to the R script.

Parameters

These are the default parameters set in the `src/sgseq/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the SGSeq [manual](#).

--gtf	The path to the gene annotation file in GTF format for annotation of fusion junctions.
	<code>--gtf \$gtf</code>
--path_to_bam	Name of bamfile.
	<code>--path_to_bam \$controlfolder/*filename*.bam</code>
--out	Output directory. The output will be separated into case and control folder based on the basefolder of the according bam file. If you are running the DICAST pipeline to compare different mapping tools this will include the name of the mapping tool of the used bam file.
	<code>--out \$outdir</code>
--cores	Number of threads to be used during the computation
	<code>--cores \$ncores</code>

SplAdder

Note: SplAdder can be used to calculate differential splicing as well as only alternative-splicing events. If you want to perform differential analysis set `differential=1` in the `/scripts/asevent_config.sh` config file. Otherwise set `differential=0`.

SplAdder Factsheet

Toolname	<i>spladder</i>
Version	2.4.3
License	BSD License

Required Files

- *gtf* , *bam*

Links

- SplAdder [manual](#)
- SplAdder publication: [SplAdder: identification, quantification and testing of alternative splicing events from RNA-Seq data](#)

Parameters

These are the default parameters set in the `src/spladder/ENTRYPOINT.sh` script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the SplAdder [manual](#).

-b Name of bamfile.

```
-b $controlfolder/*filename*.bam
```

-o Output directory. The output will be separated into case and control folder based on the basefolder of the according bam file. If you are running the DICAST pipeline to compare different mapping tools this will include the name of the mapping tool of the used bam file.

```
-o $outdir
```

-a The path to the gene annotation file in GTF format for annotation of fusion junctions.

```
-a $gtf
```

--parallel Number of threads to be used during the computation

```
--parallel $cores
```

-n Length of reads.

```
-n $read_length
```

--output-txt-conf Output in txt format.

Whippet

Note: Whippet can be used to calculate differential splicing as well as only alternative-splicing events. If you want to perform differential analysis set `differential=1` in the `/scripts/asevent_config.sh` config file. Otherwise set `differential=0`.

Whippet Factsheet

Toolname	<i>whippet</i>
Version	<i>0.11.1</i>
License	MIT Licence

Required Files

- *fastq* , *fasta* , *gtf*

Links

- Whippet [manual](#)
- Whippet publication: [Efficient and Accurate Quantitative Profiling of Alternative Splicing Patterns of Any Complexity on a Laptop](#)

Parameters

These are the default parameters set in the *src/whippet/ENTRYPOINT.sh* script. If you want to change it you can do this in the ENTRYPOINT script directly. Please refer to the Whippet [manual](#).

--fasta Reference genome in fasta format.

```
--fasta $fasta
```

--gtf The path to the gene annotation file in GTF format for annotation of fusion junctions.

```
--gtf $gtf
```

--bam Name of bamfile.

```
--bam $controlfolder/*filename*.bam
```

-x Output directory for whippet index.

```
-x $outdir/*bamfilename*/graph
```

-o Output directory. The output will be separated into case and control folder based on the basefolder of the according bam file. If you are running the DICAST pipeline to compare different mapping tools this will include the name of the mapping tool of the used bam file.

```
-o $outdir
```

3.7 Examples

Here we provide the detailed description of possible workflows. We recommend to run analysis using a terminal multiplexer, e.g. tmux or screen.

3.7.1 Running multiple mapping tools (E.g., STAR, HISAT2 and bbmap)

1. Make sure you followed the steps described in the *setup* section carefully.
2. Before getting started make sure to activate the snakemake conda environment:

```
$ conda activate dicast-snakemake
```

3. Create the *input* folder:

```
$ cd /path/to/DICAST/  
$ mkdir input
```

4. Create the directory structure as in the *sample_output*:

```
$ cd input  
$ mkdir controldir  
$ cd controldir  
$ mkdir fastqdir
```

5. Download or copy the genome fasta file into the *input* folder. Don't forget to uncompress it. E.g.:

```
$ cd /path/to/DICAST/input  
$ wget http://ftp.ensembl.org/pub/release-105/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz  
$ gunzip Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
```

6. Download or copy the genome gtf annotation into the *input* folder. Don't forget to uncompress it. E.g.:

```
$ wget http://ftp.ensembl.org/pub/release-105/gtf/homo_sapiens/Homo_sapiens.GRCh38.105.gtf.gz  
$ gunzip Homo_sapiens.GRCh38.105.gtf.gz
```

7. Download or copy the fastq files you want to align into the */path/to/DICAST/input/controldir/fastqdir*. Note: we support only paired-end RNA-Seq - fastq files have to be in pairs.

8. Go to */path/to/DICAST/scripts* and edit *config.sh* according to your run (see *How to change your config.sh file*):

```
$ cd /path/to/DICAST/scripts  
$ nano config.sh
```

In the *config.sh* file edit the following lines:

```
read_length=76  
fastaname=Homo_sapiens.GRCh38.dna.primary_assembly.fa  
gtfname=Homo_sapiens.GRCh38.105.gtf
```

9. List the mapping tools you want to run:

```
$ cd /path/to/DICAST/scripts/snakemake/  
$ nano snakemake_config.yaml
```

In the *snakemake_config.yaml* file edit the following lines:

```
Mapping_tools:  
  What_tools_to_run: 'star, hisat, bbmap'
```

10. In the */path/to/DICAST/scripts/snakemake/* folder run:

```
$ snakemake -j 1 -d /path/to/DICAST/input -s Snakefile -c snakemake_config.yaml
```

This command will start the mapping tools indicated in the *snakemake_config.yaml* (E.g. STAR, HISAT2 and bbmap).

First, the pipeline will build all necessary dockers. Second, it will create a */path/to/DICAST/index* folder and put the results of indexing. Finally, the pipeline will create a */path/to/DICAST/output* folder with the alignment results inside the dedicated folders (e.g., star-output, hisat-output, bbmap-output).

3.7.2 Running multiple alternative splicing event detection tools (E.g., MAJIQ and Whippet)

1. Make sure you followed the steps described in the *setup* section carefully.
2. Before getting started make sure to activate the snakemake conda environment:

```
$ conda activate dicast-snakemake
```

3. Create the *input* folder:

```
$ cd /path/to/DICAST/  
$ mkdir input
```

4. Create the directory structure as in the *sample_output*:

```
$ cd input  
$ mkdir controldir  
$ cd controldir  
$ mkdir fastqdir  
$ mkdir bamdir
```

5. Download or copy the genome fasta file into the *input* folder. Don't forget to uncompress it. E.g.:

```
$ cd /path/to/DICAST/input  
$ wget http://ftp.ensembl.org/pub/release-105/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz  
$ gunzip Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz
```

6. Download or copy the genome annotation file into the *input* folder. Don't forget to uncompress it. E.g.:

```
$ wget http://ftp.ensembl.org/pub/release-105/gtf/homo_sapiens/Homo_sapiens.GRCh38.105.gtf.gz  
$ gunzip Homo_sapiens.GRCh38.105.gtf.gz
```

7. Download or copy the genome gff3 annotation into the *input* folder (for MAJIQ). Don't forget to uncompress it. E.g.:

```
$ wget http://ftp.ensembl.org/pub/release-105/gff3/homo_sapiens/Homo_sapiens.GRCh38.105.gff3.gz  
$ gunzip Homo_sapiens.GRCh38.105.gff3.gz
```

8. Download or copy the fastq files you want to use into the */path/to/DICAST/input/controldir/fastqdir*. Note: we support only paired-end RNA-Seq - fastq files have to be in pairs.

9. Download or copy the bam files you want to use into the */path/to/DICAST/input/controldir/bamdir*.

10. Go to */path/to/DICAST/scripts* and edit *config.sh* according to your run (see [How to change your config.sh file](#)):

```
$ cd /path/to/DICAST/scripts  
$ nano config.sh
```

In the *config.sh* file edit the following lines:

```
$ read_length=76  
$ fastaname=Homo_sapiens.GRCh38.dna.primary_assembly.fa  
$ gtfname=Homo_sapiens.GRCh38.105.gtf  
$ gffname=Homo_sapiens.GRCh38.105.gff3
```

11. List the mapping tools you want to run:

```
$ cd /path/to/DICAST/scripts/snakemake/  
$ nano snakemake_config.yaml
```

In the *snakemake_config.yaml* file edit the following lines:

```
Alternative_splicing_detection_tools:  
  What_tools_to_run: 'maji, whippet'
```

12. In the `/path/to/DICAST/scripts/snakemake/` folder run:

```
$ snakemake -j 1 -d /path/to/DICAST/input -s Snakefile -c snakemake_config.yaml
```

This command will start the mapping tools indicated in the `snakemake_config.yaml` (E.g. MAJIQ, Whippet).

First, the pipeline will build all necessary dockers. Second, the pipeline will create a `/path/to/DICAST/output` folder with the event detection results inside the dedicated folders (e.g., majiq-output, hisat-output, whippet-output).

3.8 FAQ

Here you will find more frequently asked questions soon.

Q: How do I contribute to DICAST?

A: The best way to reach us for code updates is via our [github](#)

Q: How do I resolve issue: *docker: Error response from daemon: error while creating mount source path..*

A: Check if the folder that docker is trying to mount has the following permissions:`drwxrwsr-x`. Grant them when needed with ``chmod a+rX,u+w,g+w``.

3.9 Uninstalling DICAST

In order to uninstall dicast, please execute the script with:

```
bash scripts/uninstall-dicast.sh
```

We hope DICAST served you well, please remember to cite DICAST, should you have found it useful.

3.10 About

3.10.1 Development

DICAST was jointly developed by the groups [Big Data in Biomedicine](#), [Computational Systems Medicine](#), and [Computational Genomics and Transcriptomics Group](#)

Maintainer: Amit Fenn With contributions from Tim Faro, Fanny Roessler, Johannes Kersting, Alexander Dietrich, Chit Tong Lio

3.10.2 Citation

If you use DICAST please cite the [preprint](#)

Fenn, A.M., Tsoy, O., Faro, T., Roessler, F., Dietrich, A., Kersting, J., Louadi, Z., Lio, C.T., Voelker, U., Baumbach, J. and Kacprowski, T., 2022. Alternative splicing analysis benchmark with DICAST. *bioRxiv*.

3.10.3 Acknowledgments

DICAST was created with funding from the BMBF Sys_CARE project

3.10.4 Contact us

Amit Fenn <amit.fenn@tum.de>

Olga Tsoy <olga.tsoy@uni-hamburg.de>

Markus List <markus.list@tum.de>

Tim Kacprowski <t.kacprowski@tu-braunschweig.de>