

# CRISPRroots-1.3: User Manual

Giulia I. Corsi, Veerendra P. Gadekar, Jan Gorodkin and Stefan E. Seemann

Center for non-coding RNA in Technology and Health, Department of Veterinary and Animal Sciences,  
University of Copenhagen, Thorvaldsensvej 57, 1871 Frederiksberg, Denmark

## Contents

<b>1</b>	<b>Getting started</b>	<b>2</b>
1.1	The CRISPRroots pipeline . . . . .	2
1.2	Software dependencies . . . . .	2
1.3	Pipeline overview . . . . .	2
<b>2</b>	<b>Input files</b>	<b>2</b>
2.1	Samples table . . . . .	2
2.2	Sequencing data . . . . .	3
2.3	gRNA data . . . . .	3
2.4	Config file . . . . .	3
2.4.1	Project parameters . . . . .	3
2.4.2	Reference files . . . . .	4
2.4.3	Execution parameters . . . . .	4
<b>3</b>	<b>Pipeline usage</b>	<b>6</b>
3.1	Basic usage . . . . .	6
3.2	Advanced usage . . . . .	6
3.3	Usage in a computer cluster . . . . .	7
3.4	Data resources . . . . .	8
3.5	Test folder . . . . .	8
<b>4</b>	<b>Output files</b>	<b>8</b>
4.1	Report . . . . .	9
4.1.1	Candidate off-targets . . . . .	9
4.1.2	On-target knockin . . . . .	10
4.1.3	On-target knockout . . . . .	11
4.1.4	DESeq2 differential expression . . . . .	11
4.1.5	Multiqc sample statistics . . . . .	11
4.1.6	Mapping statistics . . . . .	11
4.1.7	eSNP-Karyotyping . . . . .	11
4.2	Results . . . . .	11
4.2.1	pre-processing (preproc) . . . . .	11
4.2.2	0_utils . . . . .	12
4.2.3	1_star_align2pass . . . . .	12
4.2.4	2_sortaligned . . . . .	12
4.2.5	2-1_RSeQC_libtype . . . . .	12
4.2.6	3_picard_sortaligned . . . . .	12
4.2.7	4_GATK_dedupSplit . . . . .	13
4.2.8	5_chromosome_wise_split_bam . . . . .	13
4.2.9	6_GATK_variants, 6-5_CRISPRoff . . . . .	13

4.2.10	7_GATK_mutect2_chromosome_wise . . . . .	13
4.2.11	7-1_GATK_mutect2_merged . . . . .	13
4.2.12	Output files from step 8 . . . . .	13
4.2.13	9_featurecounts_quantification . . . . .	14
4.2.14	10_bedops_vcf2bed . . . . .	14
4.2.15	11_VariantBasedScreening . . . . .	14
4.2.16	12-0_OffTargetCutPos . . . . .	14
4.2.17	12-1_DESeq2 . . . . .	14
4.2.18	12-2_DeGeneCoords . . . . .	14
4.2.19	12-3_intersect_degenes_offtargets . . . . .	14
4.2.20	12-4_CollapseCoordinatesGenesOff . . . . .	15
4.2.21	12-5_ExpressionBasedScreening . . . . .	15
4.2.22	13_flags . . . . .	15
4.2.23	logs . . . . .	15
<b>5</b>	<b>Frequently Asked Questions (FAQ)</b>	<b>15</b>
<b>6</b>	<b>Important remarks</b>	<b>16</b>
6.1	Variants with “*” . . . . .	16
<b>7</b>	<b>Citation</b>	<b>16</b>

## 1 Getting started

### 1.1 The CRISPRroots pipeline

The CRISPR/Cas9 genome editing tool can be used to study genomic variants and gene knockouts. By combining CRISPR/Cas9 mediated editing with transcriptomic analyses it is possible to measure the effects of genome alterations on gene expression. In such experiments it is crucial to understand not only if the editing was successful but also if the observed differential gene expression is the result of intended genome edits and not that of unwanted off-target effects. Potential off-targets sites that CRISPR/Cas9 may have edited need therefore to be verified by sequencing. However, a CRISPR/Cas9 gRNA may have hundreds (or thousands) potential off-target binding sites in a genome. How to decide which of them should be validated with higher priority? The RNA-seq data that can be sequenced as part of a CRISPR/Cas9 experiment contains information about the sequence and expression level of potential off-target sites/genes located in transcribed regions. Here we present **CRISPRroots**, a method that combines CRISPR/Cas9 and guide RNA binding properties, gene expression changes, and sequence variants between edited and non-edited cells, to discover and rank potential off-targets. The method is described in the corresponding publication [1]. **CRISPRroots: CRISPR–Cas9-mediated edits with accompanying RNA-seq data assessed for on-target and off-target sites.**

### 1.2 Software dependencies

The pipeline requires **Snakemake** v.7.28.3 or higher, **Apptainer** v.1.1.8 or higher, and the **CRISPRroots** docker container available for download in Docker Hub (`docker pull gcorsi1993/crisprroots`). All other software requirements are satisfied by **CRISPRroots** container. The pipeline was tested in a x86\_64 GNU/Linux environment with Ubuntu v.18.04.1 installed.

### 1.3 Pipeline overview

The **CRISPRroots** pipeline comprises several modules for the complete analysis of RNAseq data from CRISPR/Cas9-mediated genome editing experiments. These modules are: (1) RNA-seq read processing and mapping; (2) Somatic variant calling; (3) Variant-based off-target screening; (4) Differential gene expression; (5) Assessment of on-target knockins and knockouts; (6) gRNA off-target prediction; (7) Expression-based off-target screening. A complete description of the pipeline can be found in [1]. The basic output of **CRISPRroots** includes reports on the reads quality and mapping rates, on the status of on-target edits/knockout genes, on potential off-targets and their effects on the transcriptome, and on the differentially expressed genes. A complete description of the pipeline's output is provided below. Before executing the pipeline it is necessary to prepare the following input files (described in the next section):

- a table describing the samples
- a folder with sequencing data
- a *fasta* file with the CRISPR/Cas9 gRNA
- a configuration file

Once these are ready, the pipeline can be executed with a single line of command. Examples of execution are provided below. The reference files necessary to execute the pipeline on the human genome are provided together with the pipeline (see 3.4). A test folder with all the necessary instructions to launch the pipeline is also provided (see 3.5).

## 2 Input files

### 2.1 Samples table

The samples table is a tab-separated table in which the IDs and conditions of each sample are defined. An example of such table is given in Table 1.

The most basic form of this table is made of two columns: "Sample\_ID" and "Condition". The column "Sample\_ID" contains the name of the samples, without suffixes (eg. R1/R2 for paired-end sequencing

and the file format). These will be defined in the config file under the Project parameters (see option “`sample_suffix`”, `sample_suffix_R1`, `sample_suffix_R2`). Note that paired-end sequenced samples appear only once in the table. The “Condition” column takes two possible values: “Original” (the non-edited wild-type) or “Edited”. If necessary, additional informative columns to be included (*e.g.* “Time”). These will be taken into account in the DESeq2 design formula.

Sample_ID	Condition	Time
Astrocytes_wt_rep1	Original	Week5
Astrocytes_wt_rep2	Original	Week5
Astrocytes_wt_rep3	Original	Week5
Astrocytes_wt_rep4	Original	Week10
Astrocytes_wt_rep5	Original	Week10
Astrocytes_wt_rep6	Original	Week10
Astrocytes_APOE_PM_rep1	Edited	Week5
Astrocytes_APOE_PM_rep2	Edited	Week5
Astrocytes_APOE_PM_rep3	Edited	Week5
Astrocytes_APOE_PM_rep4	Edited	Week10
Astrocytes_APOE_PM_rep5	Edited	Week10
Astrocytes_APOE_PM_rep6	Edited	Week10

Tab. 1: Example of samples table for a 6-replicates experiment in which an *APOE* heterozygous mutation was introduced in wild type astrocyte cell lines, harvested after 5 or 10 weeks.

## 2.2 Sequencing data

The pipeline takes as input raw sequencing data in *fastq* format, zipped with *gzip*. Both single-end and paired-end data are accepted. The data needs to be in a single folder, to be specified in the config file. The name of the samples needs to be consistent with that in the samples table, plus the suffix (eg. R1/R2 for paired-end sequencing and the file format).

## 2.3 gRNA data

The sequence of the gRNA and the protospacer adjacent motif (PAM) of its target, in *fasta* format. Simply, a two-lines text file as follows:

```
>gRNA_plus_PAM
GCAGCGGGCCAGCGTGTGAGGG
```

## 2.4 Config file

The configuration file (*config.yaml*) is a *yaml* file divided in 3 sections:

1. Project parameters
2. Reference files
3. Tool parameters

An example of configuration file is given in the CRISPRroots folder under **resources** and in the test folder.

### 2.4.1 Project parameters

- **CRISPRroots**: Path to the CRISPRroots pipeline folder.
- **results\_folder**: Path to the folder in which intermediate results are placed. If not existing, the folder will be created. See 4.2 Results

- **report\_folder**: Path to the folder in which reports are placed (assessment of on-target , possible off-targets , genome integrity, tables with summary statistics on pre-processing and processing, differential expression results). If not existing, the folder will be created. See 4.1 Report
- **samples\_folder**: Path to the folder containing all samples in *fastq* format, zipped with *gzip*.
- **samples\_table**: Path to a tab-separated table specifying the IDs of each sample and their "Condition", either "Original" (the non-edited wild-type) or "Edited". If necessary, additional informative columns to be included in the DESeq2 formula can be specified (*e.g.* "Time"). An example of this table is shown in **Table 1**. Note that the file suffix/format (*e.g.* *fastq.gz*) is not present. Paired-end sequenced samples are not duplicated in the table; R1 and R2 are identified by their suffixes (see option **sample\_suffix\_R1** and **sample\_suffix\_R2**).
- **sample\_suffix**, **sample\_suffix\_R1**, **sample\_suffix\_R2**: Suffix of *fastq* reads files. For single end reads only one sample suffix is required. Given the example in **Table 1**, a suffix could be **sample\_suffix**: ".fq.gz" for the file *Astrocytes\_wt\_rep1.fq.gz* in the samples folder. For paired end reads, suffixes for both sets of reads need to be specified as **sample\_suffix\_R1**: "\_R1.fq.gz" and **sample\_suffix\_R2**: "\_R2.fq.gz" for files *Astrocytes\_wt\_rep1\_R1.fq.gz* and *Astrocytes\_wt\_rep1\_R2.fq.gz*
- **gRNA\_with\_PAM\_fasta**: path to a *fasta* file containing the gRNA + on-target PAM sequence.
- **sequencing**: Type of sequencing, either "paired" for paired end reads or "single" for single end reads.
- **single\_chr**: List of chromosomes without a homolog (*e.g.* ["chrX", "chrY"] in male human)
- **variased\_genome**: Set to "yes" if the analyses should be performed on a variant-aware version of the genome, in which short variants discovered from the RNA-seq data are introduced in the reference sequence, "no" otherwise. If set to "no", a CRISPROff output file needs to be specified in **crisproff\_output**.

#### 2.4.2 Reference files

- **picard\_reference**: Path to a the *fasta* reference genome. A Picard index generated with Picard's *CreateSequenceDictionary* should be present in the same folder.
- **repeatmasked\_regions**: Path to a RepeatMasker *.bed* annotation file. This file is used to flag the variant coordinates that overlap to the repeat-masked regions in the final results. Please refer to the readme file in the resource directory for more details on downloading this data.
- **STAR\_indexed\_transcriptome**: Path to the genome indexed with STAR-2.6.1d.
- **common\_variants**: Path to *.vcf* file of known variants. Please refer to the readme file in the resource directory for more details on downloading this data.
- **annotations\_gtf**: Path to gene features annotated in *gtf* format.
- **ssu\_rrna**, **lsu\_rrna**: Path to file in *fasta* format zipped with *gzip* containing sequences of ribosomal RNA belonging, respectively, to the small and the large ribosomal subunits. These files are used in the pre-processing step for the removal of rRNA reads if present.
- **RSeQC\_gene\_model**: Annotations in *bed* format, required by RSeQC for library type estimation.

#### 2.4.3 Execution parameters

The parameters of different tools used in the pipeline are briefly described below. Please also refer to their respective manuals for additional information.

- **Singularity**: Path to the CRISPRroots singularity environment
- **Cutadapt**: **adapter**, **adapter\_R1**, **adapter\_R2**: Path to a *fasta* file containing adapter sequences. Two files, **adapter\_R1** and **adapter\_R2** should be given for paired end reads.
- **Cutadapt**: **pair\_filter**: Only for paired end reads, can be set to "any" (discard both reads in a pair if any of them satisfies one of the filtering criteria), "both" (discard a read pair only if both reads satisfy one of the filtering criteria) or "first" (ignore the second read during filtering).
- **Cutadapt**: **phread\_score**: Threshold used for quality trimming. Default: "30".
- **Cutadapt**: **min\_length**: Reads shorter than this threshold are discarded. We suggest to set it

to approx. "90%" of the original read length.

- **Cutadapt:** other: String with additional Cutadapt parameters. Default: "--trim-n".
- **BBDuck:** mcf: Fraction of the read bases to be covered by reference kmers to be considered as match. Default = "0.5".
- **BBDuck:** K: Size of the Kmers. Default = "31"
- **BBDuck:** MAX\_MEM: Max amount of memory in GB to be used. Default = "-Xmx8g".
- **STAR:** threads: Number of threads to be used. Default = "12".
- **Featurecounts:** libtype: Integer indicating the strandness of the reads: 0=unstranded, 1=stranded, 2=reversely stranded; refers to the first reads in paired-end sequencing. In case it is unknown, the results of RSeQC could be inferred (see examples below).
- **DESeq2:** formula: Design formula. Note that only the comparison of Edited (nominator) vs Original (denominator) samples will be performed during differential expression. Default: ~ Condition
- **BCF\_consensus:** heterozygous\_keep: Select "A" to keep the alternative allele (variant to the reference) or "R" for the reference one in the presence of heterozygous mutations.
- **Mutect2:** base\_quality\_score\_threshold: Base qualities below this threshold will be changed to a minimum of 6 in Mutect2. Default = 30.
- **Mutect2:** callable\_depth: Minimum depth for an event to be considered. Default = 10
- **Mutect2:** min\_base\_quality\_score: Minimum base quality to consider a base for calling. Default = 10.
- **Mutect2:** num\_threads: Number of threads used to launch Mutect2 with GNU parallel.
- **Liftover:** min\_match: minimum ratio of bases that must remap. Default = 0.95.
- **HaplotypeCaller:** ploidy: ploidy of the samples. Default (human) = 2.
- **Endonuclease:** cut\_position: Distance from the protospacer adjacent motif 5' end at which the endonuclease cleaves the DNA. Default (SpCas9): -3.
- **Endonuclease:** gRNA\_sequence: Sequence of the gRNA, without PAM. *e.g.* "TGTATTTATACAGAACCACC".
- **Endonuclease:** binding\_site\_seq: List of possible binding sites for the endonuclease. Default (SpCas9): ["GG", "GA", "AG"]. Note that the ambiguous "N" nucleotide usually reported in the PAM sequence is absent.
- **Endonuclease:** binding\_sites\_ratios: List of weights associated to each of the PAMs defined in binding\_site\_seq. Default (SpCas9): [1.0, 0.8, 0.9].
- **Endonuclease:** binding\_site\_distance: Distance in nucleotides between the binding site (*i.e.* GG in the canonical SpCas9) and the 3' end of the gRNA-DNA duplex on the PAM's strand. In the case of SpCas9 this distance corresponds to the "N" in the "NGG" canonical PAM, thus is equal to 1. Default (SpCas9): 1.
- **Endonuclease:** extend\_binding: Extend the size of the region in which the optimal gRNA binding site is searched. Default (SpCas9): 2.
- **Endonuclease:** eng\_threshold: Threshold of maximum binding energy in *kcal/mol*. Default : 0.0.
- **Endonuclease:** seed\_region: Size of the seed region. Default (SpCas9): 10.
- **Endonuclease:** max\_mm\_seed: Maximum number of mismatches or bulges tolerated in the seed. Default: 1.
- **Edits:** type: "KI" (knockin) or "KO" (knockout).
- **Edits:** position: List of genomic coordinates (1-based) at which single base on-target edits are expected, *e.g.* ["chr14:73173676", "chr14:73173674"]. In knockout experiments use the cleavage position.
- **Edits:** mutant: List of mutated nucleotides (*e.g.* ["T", "C"]). The list must be in the same order and of the same length as option position. In knockout experiments, use "N".
- **Edits:** splice\_donor, splice\_acceptor: Lists of binary values specifying, for each position, if it corresponds to a splice donor/acceptor (1) or not (0). The lists must be in the same order and of the same length as option position. Example: [0, 0].
- **Edits:** intron: List of binary values specifying, for each position, if it is harbored within an intron in the target gene (1) or not (0). The list must be in the same order and of the same

length as option `position`. Example: `[0, 0]`.

- **Edits**: `KO`: List of gene IDs of the knockout genes (as in the annotations). The gene IDs must be present in the annotation file specified in option `annotations_gtf`. For knockin editing experiments, please use the gene ID of any gene overlapping the edited sites. Example: `["ENSG000000087263.17"]`.
- **VariantBasedScreening**: `expand_search`: Expand the PAM search up to  $n$  nucleotides from the cut site. Default: 2.
- **ExpressionBasedScreening**: `len_promoter`: Length of the upstream region of a gene to be considered as promoter. Default: 1000.
- **ExpressionBasedScreening**: `crisproff`: Flag, activate if the off-target results are obtained from CRISPRoff. Default: active.
- **CRISPRoff**: `crisproff_output`: (Optional) path to pre-computed CRISPRoff output.
- **CRISPRoff**: `webserver`: (Optional) Flag, activate if the off-target results are obtained from the CRISPRoff webserver instead of the CRISPRoff tool.

### 3 Pipeline usage

#### 3.1 Basic usage

After completing the configuration file, the pipeline can be executed from within the same folder containing the `config.yaml` (in the test dataset this is the subfolder `QPRT_DEL268T_chr16_10M-40M`) as follows:

```
$ cd QPRT_DEL268T_chr16_10M-40M # example with test dataset
$ snakemake -s <path_to_CRISPRroots>/run.smk --cores <int> --use-singularity --singularity-args
"--bind <global_path_to_QPRT_DEL268T_chr16_10M-40M>:<global_path_to_QPRT_DEL268T_chr16_10M-40M>
--bind <global_path_to_resources>:<global_path_to_resources>
--bind <global_path_to_CRISPRroots-1.3>:<global_path_to_CRISPRroots-1.3>" --dry-run

$ snakemake -s <path_to_CRISPRroots>/run.smk --cores <int> --use-singularity --singularity-args
"--bind <global_path_to_QPRT_DEL268T_chr16_10M-40M>:<global_path_to_QPRT_DEL268T_chr16_10M-40M>
--bind <global_path_to_resources>:<global_path_to_resources>
--bind <global_path_to_CRISPRroots-1.3>:<global_path_to_CRISPRroots-1.3>"
```

We suggest to run first a dryrun (`--dryrun`), which displays what passages will be executed without actually starting them. In the commands above, `--cores` specifies the maximum number of cores used in parallel by Snakemake (can be set to `all` instead of an integer). Remember to bind required folders to your singularity, as in the example, such that singularity has visibility in the folders and the results will be stored on your system after the singularity terminates the execution. If you need to first discover the library type of your sequencing data, first run the pipeline with `get_lib_type` as target rule (see instructions below).

Hint: Snakemake allows to visualize the jobs as a graph (directed acyclic graph, or DAG), highlighting the jobs completed and those to be run in different ways. To create an *svg* plot of your DAG, run the following command:

```
$ snakemake -s <path_to_CRISPRroots>/run.smk --dag | dot -Tsvg > dag.svg
```

To learn more about the DAG and the visualization of jobs please visit the Snakemake tutorial at <https://snakemake.readthedocs.io/en/stable/tutorial/basics.html> (*Step 4: Indexing read alignments and visualizing the DAG of jobs*).

#### 3.2 Advanced usage

To execute only a part of the pipeline, the corresponding target rule can be specified as shown here:

```
$ snakemake -s <path_to_CRISPRroots>/run.smk --cores <int> --use-singularity --singularity-args
"--bind <global_path_to_QPRT_DEL268T_chr16_10M-40M>:<global_path_to_QPRT_DEL268T_chr16_10M-40M>
--bind <global_path_to_resources>:<global_path_to_resources>
--bind <global_path_to_CRISPRroots-1.3>:<global_path_to_CRISPRroots-1.3>" preproc_and_map
```

The rule `preproc_and_map` only executes the pre-processing and mapping steps.

Relevant predefined target rules include:

- `preproc_and_map`: Executes all pre-processing and mapping steps. Output in:  
Mapped reads: `<path_to_results_folder>/2_sortaligned/<sample name>/Aligned.Sorted.bam`  
Preprocessing and fastQC/multiQC quality reports: `<path_to_results_folder>/preproc/`
- `variants_to_genome`: Executes all of the rules necessary to produce files containing filtered variants (*vcf*) between each sample and the reference genome. Output in:  
`<path_to_results_folder>/6_GATK_variants/<sample name>/variants_filtered.vcf`
- `on_target_check`: Executes the on-target editing assessment. Output in:  
`<path_to_report_folder>/on_target_knockin.xlsx <path_to_report_folder>/on_target_knockout.xlsx`
- `get_variated_genome`: Produces a variant-aware version of the reference genome, in which variants discovered from the RNA-seq are introduced in the reference sequence. Output in:  
`<path_to_results_folder>/6_GATK_variants/variased_genome.fa`
- `get_lib_type`: Assesses the library type with RSeQC. Output in:  
`<path_to_results_folder>/2-1_RSeQC_libtype/`
- `preproc_and_map`: Runs the reads pre-processing and mapping.  
Mapping output in: `<path_to_results_folder>/2_sortaligned/`  
Mapping statistics in: `<path_to_report_folder>/report/mapping_stats.xlsx`  
Pre-processing results in: `<path_to_results_folder>/preproc/`  
Pre-processing statistics in: `<path_to_report_folder>/report/multiqc_samples_stats.xlsx`

Note: In the test dataset, `<path_to_results_folder>` and `<path_to_report_folder>` correspond to the subfolders *results* and *report* that will be automatically generated inside the folder *CRISPR-roots-test-dataset/QPRT-DEL268T.chr16-10M-40M* by the pipeline.

To avoid removing output files defined as temporary (*e.g.* partially processed reads) while executing only a part of the pipeline use the option `--notemp`. Otherwise, temporary files (eg. *.bam* files) are removed and will need to be recreated if required by a subsequent execution of the pipeline.

Please consult the Snakemake manual at <https://snakemake.readthedocs.io> for further instructions on how to run a Snakemake pipeline or type `snakemake -h` for help in the command line.

### 3.3 Usage in a computer cluster

An example of how to set up CRISPRroots to run it with the Slurm Workload Manager is given in the test directory as `cluster_run.sh`. Configurations are given in `cluster_config.yaml`. To test the script on your system, first you need to fill in the fields:

- `<set_global_path_to_run.smk>` : path to the `run.smk` script in the CRISPRroots-1.3 folder
- `--singularity-args "<set_singularity_args_eg._binds>"` : all your singularity arguments, such as bindings to the folders containing data, resources, and code

Once the script is complete, you can run it simply as `./cluster_run.sh`. You can add the name of a target rule to run only a part of the pipeline `./cluster_run.sh [target_rule]`.



### 3.4 Data resources

A data folder that contains all of the required reference files for launching the pipeline on human samples (indexed genome, merged GENCODE v.33 GRCh38 and FANTOM-CAT v.1.0.0 annotations, repeat masked regions, known SNPs...) is provided at <https://rth.dk/resources/crispr/crisprroots>. Please see the README file included in the folder for details.

### 3.5 Test folder

A test folder can be downloaded at [https://rth.dk/resources/crispr/crisprroots/downloads/CRISPRroots\\_test\\_dataset-1.3.tar.gz](https://rth.dk/resources/crispr/crisprroots/downloads/CRISPRroots_test_dataset-1.3.tar.gz). This folder contains the required project directory structure and input files as discussed in Project parameters:

- A sub-folder, `QPRT_DEL268T_chr16_10M-40M`, with:
  - sub-folder `SAMPLES` with reads from the GEO bioproject GSE113734 (del268T\_rep1-3 and eCtrl\_rep1-3 samples) mapping between 10M to 40M bases in chr16 (data published by Haslinger *et al.*, *Mol Autism* (2018)).
  - file `config.yaml`
  - file `cluster_config.yaml`, configurations for the workload manager system (Slurm as example).
  - file `cluster_run.sh`, example of how to run on a computer cluster (Slurm)
  - file `gRNA_plus_PAM.fa`, that contains the gRNA used by Haslinger *et al.*
  - table `samples_table.tsv` describing the test samples
  - file `GCAGTTGAGTTGGGTAAATATGG.CRISPROff.tsv`, results of running CRISPROff on the example gRNA (using the CRISPROff webserver at <https://rth.dk/resources/crispr/crisproff/>).
  - folder `pre-computed` with pre-computed results
  - folder `templates`: contains a copy of the templates for `config.yaml` and `cluster_run.sh` in case mistakes happen while updating the files.
- A `resources` folder that includes a subset of the resources in the reference files dedicated to chr16.
- The script `make_config.py` to automatically setup the paths in the config file.

The script `make_config.py` can be used to automatically set the paths to the data, resources, and code directories in the configuration file `config.yaml` located in:

`CRISPRroots_test_dataset/QPRT_DEL268T_chr16_10M-40M`

To run the script, you need python3. To setup the config file for the test dataset, run the following commands:

```
$ cd CRISPRroots_test_dataset
$ python3 make_config.py --CRISPRroots <path_to_CRISPRroots>
--singularity <path_to_singularity_environment>
```

Note: you can use the optional parameters `--data_directory` and `--resources_directory` to specify the location of the data and of the resources if these are different from: `<current_directory>/QPRT_DEL268T_chr16_10M-40M`, `<current_directory>/resources`, respectively, where `<current_directory>` is `CRISPRroots_test_dataset`. The config file contains the parameters defined for the execution of the various steps of the pipeline. A copy of the config file for the CRISPRroots test dataset is provided together with the CRISPRroots software package. This file can be used as template to create the configuration file for your own dataset.

## 4 Output files

The pipeline's output files are collected in two folders: **report** and **results**.

## 4.1 Report

This folder contains the main output, including the candidate off-targets and the knockin/knockout assessment. Results regarding differential expression and processing statistics (data quality and mapping) are also present. A description of the files and content is provided below.

### 4.1.1 Candidate off-targets

This file contains the results of the off-target assessment. It is divided in two excel tables (sheets), for the expression-based and the variant-based analysis, respectively. An additional excel sheet with information on how to interpret the results is also given.

The table named “Expression-based” contains candidate off-targets resulting from the expression-based analysis, *i.e.* predicted off-targets evaluated and sorted by considering their overlap to differentially expressed genes. The table named “Variant-based” contains candidate off-targets resulting from the variant-based analysis, *i.e.* possible off-targets related to a short genomic variant discovered from RNA-seq data. The two tables share the following columns:

- **ID**: A numerical unique identifier assigned to the candidate off-target
- **COORDINATES (1-based inclusive)**: genomic coordinates of the candidate off-target (plus context) in the reference genome or in the variant-aware genome, 1-based and inclusive (*i.e.* fully closed). The variant-aware genome is optionally created by introducing variants, including indels, discovered from the RNA-seq data in the reference sequence, therefore the coordinates will not match those of the original reference. The strand information refers to that of the protospacer adjacent motif (PAM) of the candidate off-target.
- **OFF-TARGET+CONTEXT**: sequence of the candidate off-target, plus the context (default 2 nt).
- **PAM**: sequence of the protospacer adjacent motif, PAM, of the candidate off-target site.
- **BINDING STRUCT(Q-T)**: most favorable binding pattern between the gRNA (query, Q) and the candidate off-target plus context (target, T). On the left of the underscore the gRNA pattern, on the right the DNA one. X=Base not used (possible only at left-end); |=Base pair; W=wobble base pair; B=bulge. Align the two binding patterns from the right to reconstruct the full interaction. The unused bases can be ignored. For instance, the following binding pattern:

```
XXXXX||WBB||B|W||||_XXXXXXXXXX|W|||W||||
```

can be read as follows:

```
gRNA binding pattern: XXXXX||WBB||B|W||||
DNA binding pattern: XXXXXXXXXXXX|W|||W||||
Optimal interaction:
  GGGCCAGCGTGTGA
  ||WBB||B|W||||
  ||W--||-|W||||
  CCT--TC-CGCAACT
```

- **FULL MATCH-MISMATCH PATTERN (Q-T)**: Most favorable binding pattern between the gRNA (query, Q) and the candidate off-target plus context (target, T) in which all bases are reported as matches/mismatches/bulges, even if not used to compute the binding energy. Hence, there is no “X” symbol for bases not used to compute the hybridization energy. This pattern is reported for compatibility with other off-target tools and to avoid confusion regarding the binding potential at the off-target (for cases with several “X”). Left of the underscore is the gRNA binding pattern right the DNA one. —=Base pair; W=wobble base pair; B=bulge.
- **DeltaG<sub>B</sub>**: Binding energy obtained from the model described in [5]:  $\Delta G_B = \Delta G_H - \Delta G_O - \Delta G_U$ .  $\Delta G_H$  is the RNA-DNA hybridization energy, weighted using positional weights measuring Cas9 influence.  $\Delta G_O$  is the DNA-DNA binding energy.  $\Delta G_U$  is the RNA self-folding energy (minimum free energy, mfe).
- **Repeatmask**: Flag, set to “Yes” if the binding site coordinates overlap a repeat-masked region.

- **GENOMIC FEATURES:** Information regarding genes that overlap the cleavage site, such as annotation source, gene ID, gene name. In the “Expression-based” table the following additional information are given: DE=outcome of differential expression (DE) analysis, see the description of column DE\_EVENTS for more info; L2FC=log (base 2) fold change; OR=mean normalized counts in unedited samples (original); ER=mean normalized counts in edited samples. If multiple genes overlap the cleavage site of the candidate off-target, the results are separated with a vertical bar. Genes are ordered in the same order as the DE\_EVENTS.
- **N. SEED MISMATCH:** number of mismatches, DNA bulges, or RNA bulges in the seed region.
- **N. MISMATCH/UNUSED:** number of mismatches, DNA bulges, RNA bulges, or unused bases pairs in the complete gRNA-target interaction.
- **RISK:** Degree of risk for the candidate off-target.
- **PASS:** YES if the candidate off-target passes the filters (mismatches number and on the minimum free energy), no otherwise.

The table “Expression-based” has the following additional columns:

- **DE\_EVENTS:** result of the differential expression (DE) analysis for the gene(s) overlapping the cleavage site of the candidate off-target. If multiple genes overlap a the site, the results are separated with a vertical bar. The field takes 4 possible values:
  1. DEDown: the candidate off-target overlaps a gene downregulated in edited cells vs controls
  2. DEUp: the candidate off-target overlaps a gene upregulated in edited cells vs controls
  3. NotDE, Exp: the candidate off-target overlaps an expressed gene that is not differentially expressed between edited cells and controls
  4. NotDE, NotExp: the candidate off-target overlaps a gene that is not expressed (a minimum average across cell lines of 10 reads, after normalization, is required)

The table “Variant-based” has the following additional columns:

- **EVENT:** genomic variant related to the candidate off-target, including the following information separated by vertical bars: the genomic coordinates (pos, this is relative to the reference genome!), the reference allele (ref), the alternative allele (alt).
- **dbSNP:** Flag, set to “Yes” if the variant is reported in dbSNP (or another provided dataset of known variants).
- **ALLELIC DEPTHS:** Allelic depths supporting each genotype for the ref and alt alleles in the order listed. 0 is the reference allele, while alternative alleles are numbered from 1. See VCF 4.2 format (GATK). Sample statistics are separated by a semicolon.

Rows are coloured based on the RISK category and the PASS filter:

- **RED**=potential off-targets with RISK=CRITICAL and PASS=YES
- **YELLOW**=potential off-targets with RISK=MAJOR (any type) and PASS=YES
- **WHITE**=potential off-targets with RISK=MINOR or PASS=NO

The text is in gray if PASS=NO, black otherwise.

#### 4.1.2 On-target knockin

This table contains the results of the on-target knockin assessment. For each sample (rows) the columns report the number reads mapping to the reference (Ref), to the variant specified in the config file (A,T,G,C, or N), to skips (Skip) or to other events (Other, eg. indels). The pileup of the mapped reads is reported after a newline. Every column represents one of the edited positions specified in the config file. The column “Status” provides a verbal description of the editing status at each position (homozygous/heterozygous mutation, absence of mutation, absence of reads...).

### 4.1.3 On-target knockout

This table contains the results of the on-target knockout assessment. For each sample (rows) the column(s) named with the genome ID of the edited gene(s) report the number of normalized reads mapping to the gene(s), if it is up- or down-regulated, the fold change in log 2 scale (L2FC) and the associated Benjamini-Hochberg adjusted P-value (Wald Test).

### 4.1.4 DESeq2 differential expression

This table contains the result of the differential expression analysis carried out with DESeq2. The index column is “gene ID|gene name”. Additional columns are:

- baseMean: mean normalized counts
- log2FoldChange: fold change estimate in the comparison Edited vs Original (non-edited)
- lfcSE: standard error estimate of log2FoldChange
- stat: Wald statistic
- pvalue: Wald test P-value
- padj: Bonferroni-Hochberg P-adjusted value
- one column for each sample, with the normalized gene counts
- mean original: mean between the normalized gene counts of samples marked as “Original” (non-edited)
- mean edited: mean between the normalized gene counts of samples marked as “Edited”

### 4.1.5 Multiqc sample statistics

The table “multiqc\_sample\_stats” contains the number of reads that passed each of the following pre-processing steps: initial assessment, adapter removal and quality or length filtering, rRNA removal. In the case of pair-wise sequencing, two entries are present for each sample (R1, R2).

### 4.1.6 Mapping statistics

The table “mapping\_stats” contains a collection of log files produced during the alignment with STAR[3] (2 pass), reporting statistics such on the mapping speed, the mapping rate, the number of splices detected, indels, multimappers, chimeric bindings and so on. Each column contains the summary of one sample.

### 4.1.7 eSNP-Karyotyping

This module has been removed from the pipeline since version 1.3.

## 4.2 Results

The results folder contains the intermediate pipeline’s results (reads pre-processing, mapping, counting, differential expression, variant detection, off-targets search, evaluation of potential off-targets...). To diminish storage requirements some intermediate output files are removed once they are not required anymore by the pipeline (*eg.* intermediate read filtering steps). To avoid removing intermediate files, the pipeline can be launched with the `--notemp` option in **Snakemake**. Below, the output files that are not marked as temporary are underlined.

### 4.2.1 pre-processing (preproc)

The results of the pre-processing steps (filtering of raw reads) are stored under the results subfolder “preproc”. Here the following output folders are given:

- 0\_fastqc: FastQC reports of samples prior filtering (raw reads).
- 0-1\_multiqc: MultiQC summary reports of samples prior filtering (raw reads).
- 1\_cutadapt\_cleaning: reads filtered with Cutadapt (reads quality, length, other..).

- 1-1\_fastqc\_after\_cutadapt\_cleaning: FastQC reports of samples after filtering with Cutadapt (reads quality, length, other..).
- 1-2\_multiqc\_after\_cutadapt\_cleaning: MultiQC summary reports of samples after filtering with Cutadapt (reads quality, length, other..).
- 2\_bbduk\_rrna\_filter: reads filtered for residual rRNA with BBDuk. Files suffixes have the following meaning: matched\_LSU: reads that match the large ribosomal subunit, matched\_SSU: reads that match the small ribosomal subunit, filtered\_SSU: reads filtered from matches with the small ribosomal subunit. The file carrying the name of the sample without any suffix contains reads filtered from matches with both ribosomal subunits.
- 2-1\_fastqc\_after\_rRNA\_removal: FastQC reports of samples after rRNA removal.
- 2-2\_multiqc\_after\_rRNA\_removal: MultiQC summary reports of samples after rRNA removal.

#### 4.2.2 0\_utils

The folder “0\_utils” contains the following files:

- dict\_chroms\_lengths.pkl: pickle file containing the length of each chromosome in a dictionary
- edits.bed: genomic coordinates of the CRISPR/Cas9-mediated edits
- gRNA.fa: gRNA in fasta format
- guide\_RNA.fold: folding of the gRNA as produced by RNAfold [6].
- lifted\_annotations: annotations in gtf format lifted to the variant-aware reference genome (optional)
- lifted\_edits: genomic coordinates of the CRISPR/Cas9-mediated edits lifted to the variant-aware reference genome (optional)
- lifted\_repeatmask.bed: repeatmasked genomic regions lifted to the variant-aware reference genome (optional)
- lifted\_SNPdb.bed: SNPdb entries lifted to the variant-aware reference genome (optional)
- unlifted\_annotations: annotations in gtf format that failed to be lifted to the variant-aware reference genome (optional)
- unlifted\_edits: genomic coordinates of the CRISPR/Cas9-mediated edits that failed to be lifted to the variant-aware reference genome (optional)
- unlifted\_repeatmask.bed: repeatmasked genomic regions that failed to be lifted to the variant-aware reference genome (optional)
- unlifted\_SNPdb.bed: SNPdb entries that failed to be lifted to the variant-aware reference genome (optional)
- variated\_genome.2bit: reference genome (or variant-aware reference genome, optional) in 2bit format.

#### 4.2.3 1\_star\_align2pass

For each sample, the output of the alignment with STAR [3] (2 pass). This include the aligned reads in *bam* format in the file “<SampleName>.Aligned.out.bam” and log files. Please visit the repository of STAR for further information.

#### 4.2.4 2\_sortaligned

For each sample, the aligned reads in *bam* format under “<SampleName>/Aligned.Sorted.bam” sorted by coordinates and indexed with SAMtools.

#### 4.2.5 2-1\_RSeQC\_libtype

For each sample, the library type as estimated by RSeQC “infer\_experiment”.  
Files are named as: “<SampleName>\_libtype.txt”.

#### 4.2.6 3\_picard\_sortaligned

For each sample, the aligned reads in *bam* format sorted by queryname with Picard “SortSam”.

#### 4.2.7 4\_GATK\_dedupSplit

For each sample, the aligned reads in *bam* format processed with GATK “MarkDuplicates” (filename contains “Dedup”), which marks duplicate reads, and with GATK “SplitNCigarReads” (filename contains “Split”), which splits reads containing Ns in the cigar string.

#### 4.2.8 5\_chromosome\_wise\_splitted\_bam

For each sample, the *bam* alignments split by chromosome and indexed.

Files are named “<SampleName>/<SampleName>.<chromosome>.bam” (or *.bai*).

#### 4.2.9 6\_GATK\_variants, 6-5\_CRISPRoff

The folder 6\_GATK\_variants contains the results of the germline variant calling performed with the GATK HaplotypeCaller module and other related files. The former are stored in one folder for each non-edited sample and include *vcf* files of unfiltered variants, variants with the filters applied (the filename contains “filters”), and variants filtered by removing those variants that do not pass the filters (the filename contains “filtered”); NB: germline variants are only called in non-edited samples. The latter are the following:

- intersection\_original: output of bcftools isec, which intersects the variants called for different samples. The file 0000\_filtered.vcf.gz contains the results of the intersection used to make the variant-aware genome, in which the “\*” symbols are changed to “N”s (see also Section 6.1).
- variased\_genome.fa: a *fasta* file containing the sequence of the reference genome onto which the variants common to all unedited samples are introduced with bcftools consensus.
- chain\_liftover\_variased\_genome: chain file that can be used to lift genomic annotations relative to the reference genome to the new variant-aware reference. This is produced by bcftools consensus.
- variased\_genome.suf: suffix array of the (variant-aware) reference genome, built with Rsearch2.

The variant-aware genome is generated to improve the detection of potential off-target, which are searched with Rsearch2 allowing up to 6 mismatches between the gRNA and the DNA sequence, and evaluated for CRISPR/Cas9 binding potential with CRISPRoff. The output of this analysis is stored in the folder 6-5\_CRISPRoff. There, the subfolder Rsearch2 contains the output of the genome-wide search of binding sites for the gRNA used in the CRISPR/Cas9-mediated editing process. Other output included in 6-5\_CRISPRoff is the list of off-targets evaluated by CRISPRoff (gRNA\_ID.CRISPRoff.tsv) and two tables with energy-based binding properties of the gRNA (CRISPRparams.tsv) and its specificity to the on-target site calculated by taking into account the genome-wide off-target potential (CRISPRspec.tsv). See CRISPRoff for details [5].

#### 4.2.10 7\_GATK\_mutect2\_chromosome\_wise

For each chromosome, the output of the somatic variant calling analysis done with GATK “Mutect2” and of “FilterMutectCalls” (“filtered” in the filename). These include variants in vcf format, related indexes, and statistics on the output.

#### 4.2.11 7-1\_GATK\_mutect2\_merged

Filtered Mutect2 somatic variant calls from all chromosomes merged into a single *vcf* file (mutect2.vcf.gz), indexed.

#### 4.2.12 Output files from step 8

Step 8 of the pipeline is the knockin and knockout assessment of CRISPR/Cas9 mediated edits. The output of this analysis is in the report folder.

#### 4.2.13 9\_featurecounts\_quantification

Output of featureCounts, consisting of two tables: `feature_count.tsv`, which contains read counts of genes (rows) for each sample (columns); `feature_count.tsv.summary`, containing summary statistics of feature assignment.

#### 4.2.14 10\_bedops\_vcf2bed

Mutect2 variants in *bed* format (`mutect2_variants.bed`) and lifted to the variant-aware genome (optional, file `mutect2_lifted.bed`). Unlifted entries can be found in `mutect2_unlifted.bed`.

#### 4.2.15 11\_VariantBasedScreening

Raw results of the variant-based off-target search, including the evaluation of gRNA binding sites related to variants and the intersection of their corresponding cut sites with annotated genes. Please refer to the more complete table in the report folder.

#### 4.2.16 12-0\_OffTargetCutPos

Position of the cleavage site of the potential off-targets identified by CRISPRoff, in *bed* format. Additional columns in the bed file include the sequence of the off-target (PAM strand), a score representing the binding potential of the gRNA to this site, and the PAM.

#### 4.2.17 12-1\_DESeq2

This folder contains the results of the gene expression and differential expression analysis carried out with featureCounts and DESeq2, in various formats. The following files are provided:

- `feature_count.tsv`: read counts of genes as described in 4.2.13. Additional columns with genes descriptions are removed, and column headers are renamed with the samples names instead of the path to the alignment file.
- `diffexpr-results.tsv`: copy of the DESeq2 results table, in *tsv* format.
- `genes_DEDown.tsv`, `genes_DEUp.tsv`, `genes_NotDEExpressed.tsv`, `genes_NotDENotExpressed.tsv`: tables with differentially expressed genes of a certain category. Categories are defined by the expression and differential expression level of the genes. DEDown: expressed genes (basemean  $\geq 10$ ) downregulated ( $\log_2$  fold change  $< -0.05$ ,  $\text{padj} < 0.01$ ); DEUp: expressed genes (basemean  $\geq 10$ ) upregulated ( $\log_2$  fold change  $> 0.05$ ,  $\text{padj} < 0.01$ ); NotDEExpressed: not in DEDown or DEUp, basemean  $\geq 10$ ; NotDENotExpressed: basemean  $< 10$ .

#### 4.2.18 12-2\_DeGeneCoords

The file `genes_coordinates.bed` contains the coordinates of all genes and promoters and their expression and differential expression levels. Columns are as follows: chromosome; start position; end position; gene information (source, geneID, GeneName, Type (promoter or gene), differential expression category,  $\log_2$  fold change (L2FC), mean normalized reads in original (OR, *i.e.* non-edited) cells and in edited (ER) cells); placeholder; strand.

#### 4.2.19 12-3\_intersect\_degenes\_offtargets

The intersection between differentially expressed genes and predicted off-target cleavage coordinates obtained with BEDtools is in the file `genes_offtargets_intersect.bed`. Columns start with the description of the predicted off-target: chromosome; start position; end position; sequence; gRNA binding energy score; strand; PAM sequence; and continue with the description of the overlapping genomic feature (if any, otherwise -1 is used as placeholder): chromosome, start position, end position, gene information as in 4.2.18; placeholder; strand. The files `genes_coordinates_sorted.bed` and `off-target_cut_positions_sorted.bed` are as described in 4.2.16 and 4.2.18, but are sorted by coordinates.

#### 4.2.20 12-4\_CollapseCoordinatesGenesOff

- `collapsed_genes_offtargets_coordinates.bed`: Collapsed version of the genes and off-targets intersection file described in 4.2.19, in which genomic features (genes or promoters) that overlap the same predicted off-target cleavage site are fused in the same row and separated by a vertical bar. Columns reporting the coordinates of these features are not present.
- `collapsed_genes_offtargets_coordinates_singlechr_filtered.bed`: filtered version of the file above in which potential off-targets laying on hemizygous chromosomes and overlapping an expressed gene are removed.

#### 4.2.21 12-5\_ExpressionBasedScreening

Raw results of the expression-based off-target search, in which potential gRNA binding sites are evaluated. Please refer to the more complete table in the report folder.

#### 4.2.22 13\_flags

Files in *bed* format containing the coordinates of the flags that are added to the off-target report tables to highlight the following properties:

- `frt.bed`: repeatmasked bases overlapping potential off-target in the expression-based analysis
- `frv.bed`: repeatmasked bases overlapping the coordinates of potential off-targets related to variants
- `fsv.bed`: SNPdb entries overlapping the coordinates of potential off-targets related to variants

#### 4.2.23 logs

Log files (standard output and error) of all tools executed as part of the pipeline.

## 5 Frequently Asked Questions (FAQ)

### Which potential off-targets should I validate?

We suggest to validate all the potential off-targets marked in red (CRITICAL) or yellow (MAJOR). If they are too many, you can restrict the validation to the CRITICAL potential off-targets only.

### There are too many CRITICAL potential off-targets. Can I filter the results further?

Ideally, you should validate all CRITICAL potential off-targets. However, in some cases they may be still too many, either because the off-target assessment was not performed optimally during the gRNA design phase or because only few gRNAs could be designed at the target region. Further filters you may want to apply are:

- Filter by “DeltaG\_B” column, which contains the gRNA-DNA binding energy (see Alkan et al., 2018). You can select for your validation the CRITICAL potential off-targets with lowest DeltaG\_B value.
- Filter by maximum number of mismatches/bulges/unused bases in column “N. MISMATCH/UNUSED”, hence selecting the potential off-targets with lowest number of mismatches/bulges/unused bases. You can also choose different thresholds for different PAMs (eg. up to n=8 mismatches/bulges/unused bases for NGG PAMs, up to n=6 for other PAMs).

### How to design primers for the validation?

You can use the script `<path_to_CRISPRroots>/scripts/get_offtarget_context_for_sequencing` to extract the DNA sequences around your potential off-targets, and use these sequences to design your primers. Note that you can filter the results further when launching the script (eg. set a maximum DeltaG\_B). The script runs in *python3* with *pandas* installed, and requires *bedtools* to be installed. The CRISPRroots docker/singularity environment provided can be used to run this script.

For instance, to obtain the DNA sequence plus 300nt left-right of the top 20 CRITICAL off-targets



highlighted by the Expression-based off-target analysis, eliminating those overlapping repeatmasked bases, you can run the script as follows:

```
$ <path_to_CRISPRroots>/scripts/get_offtarget_context_for_sequencing.py \  
-ot <path_to_report_folder>/candidate_off_targets.xlsx \  
-r CRITICAL \  
-t Expression-based \  
-n 300 \  
-g <path_to_results_folder>/6_GATK_variants/variased_genome.fa \  
-rk \  
-top 20 \  
-outf <path_to_output_folder>
```

Note that in this case the genome used to extract the DNA sequences is the “*variased\_genome.fa*”, as the variant-aware version of the genome was used to search for potential off-targets (option `variased_genome` was set to `yes`).

See `<path_to_CRISPRroots>/scripts/get_offtarget_context_for_sequencing.py --help` to visualize the full list of options for this script.

The output folder specified in `-outf <path_to_output_folder>` will contain the files:

- *candidate\_off\_targets\_Expression-based.fa*: *fasta* sequences of the potential off-targets+context (300nt left + potential. off-target + context as specified by option `extend_binding` + 300nt right)
- *candidate\_off\_targets\_Expression-based.bed*: coordinates of such sequences (in the given reference or variant-aware genome) in *bed* format

## 6 Important remarks

### 6.1 Variants with “\*”

Since CRISPRroots v.1.2 the generation of the variant-aware genome was modified by transforming “\*” variants into “N”s. When a “\*” in the called variants (*vcf*) is integrated in the *fasta* genome, then it is removed automatically by `faToTwoBit` during the generation of the *twobit* genome. This causes a shift of the bases positions in the corresponding chromosome. Hence, “\*” is changed to “N”. Note that this does not affect the results reported in the CRISPRroots original publication (Corsi et al., Nucleic Acids Research (2021)).

## 7 Citation

When using the CRISPRroots software please cite:

Corsi GI, Gadekar VP, Gorodkin J and Seemann SE, **CRISPRroots: on- and off-target assessment of RNA-seq data in CRISPR-Cas9 edited cells**. *Nucleic Acids Research*, in press (2021)

<http://dx.doi.org/10.1093/nar/gkab1131>.

## References

- [1] Corsi, G. I., Gadekar, V. P., Gorodkin, J., and Seemann, S. E. (2021) CRISPRroots: on- and off-target assessment of RNA-seq data in CRISPR-Cas9 edited cells. *Nucleic Acids Research*, *in press*,.
- [2] Weissbein, U., Schachter, M., Egli, D., and Benvenisty, N. (July, 2016) Analysis of chromosomal aberrations and recombination by allelic bias in RNA-Seq. *Nature Communications*, **7**(1).
- [3] Dobin, A., Davis, C. A., Schlesinger, F., Drenkow, J., Zaleski, C., Jha, S., Batut, P., Chaisson, M., and Gingeras, T. R. (October, 2012) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**(1), 15–21.
- [4] Kim, D., Pertea, G., Trapnell, C., Pimentel, H., Kelley, R., and Salzberg, S. L. (2013) TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions. *Genome Biology*, **14**(4), R36.
- [5] Alkan, F., Wenzel, A., Anthon, C., Havgaard, J. H., and Gorodkin, J. (October, 2018) CRISPR-Cas9 off-targeting assessment with nucleic acid duplex energy parameters. *Genome Biology*, **19**(1).
- [6] Lorenz, R., Bernhart, S. H., zu Siederdissen, C. H., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (November, 2011) ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, **6**(1).