PARANOiD

Patrick Barth

Sep 26, 2023

CONTENTS:

CHAPTER

ONE

PARANOID INPUTS

Detailed description of all input files

1.1 Reads

`FASTQ` file containing all reads. Each read is represented by 4 lines:

1. **Sequence identifier** and optional description. Starts with a `@`
2. Actual **nucleotide sequence** of the read
3. **Delimiter line**. Starts with a +
4. **Quality values** of nucleotide sequence (line2). Must contain same number of symbols as line 2

Example:

```
1. @SEQ_ID
2. GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
3. +
4. !''((((***+))%%%++)(%%%%).1**-+*''))**55CCF>>>>>>CCCCCCC65
```

1.2 Barcodes

`TSV` file containing experiment names and the corresponding barcode sequences. Reads from the input `FASTQ` file are split according to the detected barcode sequence and assigned to the appropriate experiment. This results in one `FASTQ` file per experiment.
When choosing the option to merge replicates *--merge_replicates* the experiment names have to be chosen appropriately indicating which experiments belong together. In order to do that the appendix `_rep_<number>` has to be added to the experiment names, exchanging `<number>` with the replicate number.
Barcode files consist of 2 columns separated by a single `tab`:

1. experiment name
2. barcode sequence present in reads

Experiment names should only consist of Letters `{a-zA-Z}`, numbers `{1-9}` and underscores `_`. Any whitespaces (e.g. space, tab) will result in errors and thus the termination of the pipeline execution. The length of the barcode sequence is dependant on the protocol used an can be adapted via *--barcode_pattern*.

Example:

```
knockdown_N_rep_1    TGATAG
knockdown_N_rep_2    AGTGGA
knockdown_N_rep_3    GCTCGA
mock_N_rep_1    TAAGTA
mock_N_rep_2    GCAGTC
mock_N_rep_3    CCTAGG
```

1.3 Reference

`FASTA` file containing nucleotide data of interest. Is used to align reads to and thus find the location of cross-link sites. Can contain genomic or transcriptomic sequences of an organism or completely artificial sequences. Every sequence consists of at least 2 lines: 1. Header 2-n. Nucleotide sequence The header starts with a > and is followed by a description of the sequence The sequence consists of nucleotides `{ACGTN}` and can span an arbitrary amount of lines

Example:

```
>NW_024429180.1 Mesocricetus auratus isolate SY011 unplaced genomic scaffold
AACTCTGTTGtaaaaaggctttcccacattcattcCATTCATAAGGTTTCTGTACATTATGGATTCTTTCATGCCTTTTA
AGATGATTATGATATACATAGACTTTAACACCTCAAGAATAttcaggtttctctccagtatgacaATTTGGTCTAATTAT
AAAGAAGAATCAGATATTAAGGTTTTATCACTGTTTACACTCATGCTGTTCCCCTTCATTAAGGTTGGTTTGGATCTTTG
AATATACCTGGGTTCCTATAGTCTCCACCATCACATCTTTATGGAGATTCTTCTGGGAGGGATCCAGCAAATCCCACTCT
...
```

1.4 Annotation

`GFF` or `GTF` file. Contains annotation information belonging to the reference used in the input. Describes features and their positions. PARANOiD does not rely on the annotation for it's analysis, however it is highly recommended to provid it when working with splicing capable organisms (*--domain eu*) as annotation files typically contain information about intron-exon structures which highly improve the mapping capability. Furthermore, providing an annotation file enables the *RNA subtype analysis*. Consists of several header lines followed by one line feature. Header lines start with a `#` and contain general information about the annotation.

Feature lines consist of 9 columns which are separated by tabs:

1. **seqname**: name of the chromosome or scaffold on which the feature is located
2. **source**: name of the program that generated this feature or the source
3. **feature type**: type of the current feature - e.g. exon - intron - CDS - mRNA - 3_prime_UTR - transcript - 5_prime_UTR
4. **start**: Start position of the feature (1-based)
5. **end**: End position of the feature (1-based)
6. **score**: Float point value (can also simply be a `.`)
7. **strand**: Stradn on which the feature is present. + for forward; - for reverse
8. **frame**: Indicates which base of the feature is actually the first base of a codon. 0 -> the first base of the feature is the first abse of a codon; 1 -> the second base of the feature is the first base of a codon .... (can slo simply be a `.`)
9. **attributes**: Semicolon separated list of tag-value pairs providing additional information

Example:

```
##gff-version 3
#!gff-spec-version 1.21
#!processor NCBI annotwriter
#!genome-build BCM_Maur_2.0
#!genome-build-accession NCBI_Assembly:GCF_017639785.1
#!annotation-source NCBI Mesocricetus auratus Annotation Release 103
##sequence-region NW_024429180.1 1 52462669
##species https://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=10036
NW_024429180.1      RefSeq  region  1       52462669        .       +       .     ␣
↪ ID=NW_024429180.1:1..52462669;Dbxref=taxon:10036;Name=Unknown;chromosome=Unknown;
↪dev-stage=adult;gbkey=Src;genome=genomic;isolate=SY011;mol_type=genomic DNA;sex=female;
↪tissue-type=liver
NW_024429180.1      Gnomon  pseudogene      37366   38359   .       +       .  ␣
↪    ID=gene-LOC101842720;Dbxref=GeneID:101842720;Name=LOC101842720;gbkey=Gene;
↪gene=LOC101842720;gene_biotype=pseudogene;pseudo=true
```

CHAPTER

TWO

BASIC USAGE OF PARANOID

Description on how to the minimal set of PARANOiD. This means running it with only a *FASTQ file* containing reads, a *reference genome* and a *barcode file*. Generated outputs include *alignments*, *cross-link sites* in 3 different file types, an overview of the *peak height distribution*, :ref:` processing statistics <output-statistics>`, *strand distributions* and an *IGV session* which can be loaded directly into thr IGV to visualize first results. A directory with the name output (*unless stated otherwise*) containing all results will be generated. All parts marked with <> are files that need to be specified by the user

```
nextflow /path/to/directory/PARANOiD.nf --reads <read-file> --reference <reference-file>␣
↪--barcodes <barcode-file> --omit_peak_calling --omit_peak_distance --omit_sequence_
↪extraction
```

CHAPTER

THREE

EXAMPLE RUN

This page shows the minimal execution of PARANOiD on example data.

3.1 Download test data

The example files consist of 2 different experiments and can be downloaded from Zenodo via following link: https://zenodo.org/record/7733740

Alternatively they can be downloaded using the CLI with following commands:

```
# RVFV sample:
curl "https://zenodo.org/record/7733740/files/barcodes-RVFV.tsv" -o barcodes-RVFV.tsv
curl "https://zenodo.org/record/7733740/files/virion-reads-M-fragment-only.fastq.gz" -o␣
↪virion-reads-M-fragment-only.fastq.gz
curl "https://zenodo.org/record/7733740/files/reference_RVFV.fasta.gz" -o reference_RVFV.
↪fasta.gz
gzip -d reference_RVFV.fasta.gz
gzip -d virion-reads-M-fragment-only.fastq.gz

# BHK sample:
curl "https://zenodo.org/record/7733740/files/barcodes-BHK.tsv" -o barcodes-BHK.tsv
curl "https://zenodo.org/record/7733740/files/BHK-reads-M-fragment-only.fastq.gz" -o BHK-
↪reads-M-fragment-only.fastq.gz
curl "https://zenodo.org/record/7733740/files/reference_RVFV.fasta.gz" -o reference_RVFV.
↪fasta.gz
gzip -d reference_RVFV.fasta.gz
gzip -d BHK-reads-M-fragment-only.fastq.gz
```

3.2 Run PARANOiD on test data

To automatically download and then execute PARANOiD the following commands can be used:

```
# RVFV sample:
nextflow run patrick-barth/PARANOiD -r main --reads virion-reads-M-fragment-only.fastq --
↪reference reference_RVFV.fasta --barcodes barcodes-RVFV.tsv --output output-RVFV --
↪omit_peak_calling --omit_peak_distance --omit_sequence_extraction  -profile podman

# BHK sample:
```

(continues on next page)

(continued from previous page)

```
nextflow run patrick-barth/PARANOiD -r main --reads BHK-reads-M-fragment-only.fastq --
↪reference reference_RVFV.fasta --barcodes barcodes-BHK.tsv --output output-BHK --omit_
↪peak_calling --omit_peak_distance --omit_sequence_extraction  -profile podman
```

In case the resource To manually download and execute PARANOiD following commands can be used:

```
git clone git@github.com:patrick-barth/PARANOID.git

# RVFV sample:
nextflow PARANOID/main.nf --reads virion-reads-M-fragment-only.fastq --reference␣
↪reference_RVFV.fasta --barcodes barcodes-RVFV.tsv --output output-RVFV --omit_peak_
↪calling --omit_peak_distance --omit_sequence_extraction  -profile podman

# BHK sample:
nextflow PARANOID/main.nf --reads BHK-reads-M-fragment-only.fastq --reference reference_
↪RVFV.fasta --barcodes barcodes-BHK.tsv --output output-BHK --omit_peak_calling --omit_
↪peak_distance --omit_sequence_extraction  -profile podman
```

If another container execution system is to be used then *podman* can be displaced with *singularity* or *docker* as described *here*. If the jobs are supposed to be distributed to a cluster the distribution system can be added to the profile argument as described *here*. Please note that without distributing jobs to a cluster all processes will be calculated locally. This currently uses a minimum of 8 cores and 100 GB memory which can exceed the available resources of typical computers. In this case resource usage can be adapted in the config file.

3.3 Output

The minimal execution of PARANOiD only includes the *basic analysis* and should provide the following outputs if executed correctly:

1. *Directory containing alignments*
2. *Raw cross-link sites*
3. *Execution metrics*
4. *An IGV session*
5. *Distribution of peak heights*
6. *The reference sequence used for the run*
7. *Statistics and reports of the run and several processes*
8. *Strand distributions*

CHAPTER

FOUR

PARANOID PARAMETERS

Explanation of all PARANOiD parameters

4.1 --reads

Essential parameter!
States *file* containing reads obtained by iCLIP experiments.
Expects a `FASTQ` file.

Usage:

```
--reads /path/to/input-file.fastq
```

4.2 --barcodes

Essential parameter!
States *file* containing barcode sequences and experiment names. Necessary to split reads and allocate them to their experiment.
Expects a `TSV` file.

Usage:

```
--barcodes /path/to/barcodes.tsv
```

4.3 --reference

Essential parameter!
States *reference genome* used to align reads to and thus to determine the location of cross-link sites.
Expects a `FASTA` file.

Usage:

```
--reference /path/to/reference.fasta
```

4.4 --annotation

States *annotation file* used for the *RNA subtype analysius*.
Expects a `GFF` or `GTF` file.

Usage:

```
--annotation /path/to/annotation.gff
```

4.5 --merge_replicates

Merges replicates into a single representatiove form. In order to do so experiment names need to named in a particular manner which is further explained in the *barcodes section*.

`Default: false`

Usage:

```
--merge_replicates
```

4.6 --correlation_analysis

Only applies when *replicate merging* is chosen. Does a correlation analysis of replicates to show their similarity (and thus if they should be merged at all). Can cause problems with large reference genomes due to excessive RAM usage.

`Default: false`

Usage:

```
--correlation_analysis
```

4.7 --barcode_pattern

Adapt barcode patterns to different protocols. Default protocol is iCLIP2. `N` s represent the random barcode and `X` s the experimental barcode

Usage (default):

```
--barcode_pattern NNNNNXXXXXXNNNN
```

Example for iCLIP1

```
--barcode_pattern NNNXXXXNN
```

4.8 --domain

Choose between bowtie2 and STAR to be used to align reads to the reference sequence. Bowtie2 should be used for prokarytic organisms or transcript sequences while STAR should be used for eukaryotic organisms (or rather all splicing capable organisms) as STAR is splicing aware. If using STAR for splicing capable organisms it is highly recommended to provide an *annotation file* file besides the reference.

Options:
pro -> Bowtie2 (default)
eu -> STAR

Usage (default):

```
--domain pro
```

4.9 --max_alignments

Maximum number of alignments the mapping tool provides per read. It is not guaranteed that this many alignments are found per read. If you want to find as many alignments as possible please use the parameter *--report_all_alignments*

Usage (default):

```
--max_alignments 1
```

4.10 --report_all_alignments

If used the mapping tools will report all alignments rather than a few. Overwrites the option *--max_alignments*

Usage:

```
--report_all_alignments
```

4.11 --output

Specify directory to which output generated by PARANOiD will be written.

Usage (default):

```
--output ./output
```

4.12 --min_length

Specify minimum length a read needs to have after adapter removal to persist. Reads that become shorter during adapter removal will be filtered out.

Usage (default):

```
--min_length 30
```

4.13 --min_qual

Minimum quality for bases. All bases below that quality are cut off. The quality score (also known as Phred quality score) describes the certainty of correctness of the base and is typically calculated as follows with e being the error probability: $Q - Score = -10log_{10}(e)$

Phred Quality score	Error probability	Accuracy
10	10%	90%
20	1%	99%
30	0.1%	99.9%
40	0.01%	99.99%

Usage (default):

```
--min_qual 20
```

4.14 --min_percent_qual_filter

Percentage of nucleotides that need to have a quality score above the chosen *minimum base quality*. Reads with less nucleotides above the desired quality will be removed.

Usage (default):

```
--min_percent_qual_filter 90
```

4.15 --barcode_mismatches

Number of mismatches allowed within the experimental barcode to still assign a read to an experiment. Typically, experimental barcodes should be designed with a v of at least 3 to each other in order to allow one mismatch.

Usage (default):

```
--barcode_mismatches 1
```

4.16 --mapq

Minimum alignment quality (mapq score) an alignment needs to retain. The meaning of different scores is dependant on the aligner chosen via *--domain*. All alignments with a mapq score below will be removed after the alignment step. Please note that these are just a short overview of the meaning of MAPQ scores and that they can be more complex than shown here when going into details. the MAPQ score can be found in alignment files (SAM/BAM/CRAM) in column 5.

Usage (default):

```
--mapq 2
```

4.16.1 Score meanings for Bowtie2 (--domain pro)

Apart from the description in the table a higher MAPQ score means less allowed mismatches (with difference of the base quality a mismatched nucleotide has)

MAPQ score	Description
0	All mappable reads
1	Multimapped reads that have the same alignment quality at different positions
2-39	Mulitmapped reads that have one specific alignment with a better score than the other potential positions
40	Reads mappable to only one position
42	Reads mappable to only one position with an almost perfect alignment. Best MAPQ score in Bowtie2 alignments

More information can be found here

4.16.2 Score meanings for STAR (--domain eu)

MAPQ score	Description
0	Maps to 10 or more positions
1	Maps to 4-9 positions
2	Maps to 3 positions
3	Maps to 2 positions
255	Reads mappable to only one position. Best MAPQ score in STAR alignments.

The mapping quality MAPQ (column 5) is 255 for uniquely mapping reads, and
$MAPQscore = int(-10log_{10}(1 - 1/[\text{number of positions the read maps to}]))$ for multi-mapping reads. This scheme is same as the one used by TopHat [...]

Source: Bowtie2 manual

4.17 --map_to_transcripts

Should be used when transcripts are given as reference instead of a reference genome. Returns the transcripts with most hits from each sample. More information can be found *here*

`Default: false`

Usage:

```
--map_to_transcripts
```

4.18 --number_top_transcripts

The number of transcripts with most hits that are selected from each sample if parameter *--map_to_transcripts* was used. As the amount is chosen from each sample the total number of transcripts can excede this number.

Usage (default):

```
--number_top_transcripts 10
```

4.19 --omit_peak_calling

If specified *peak calling* will not be performed. Will be performed by default.

Usage:

```
--omit_peak_calling
```

4.20 --peak_calling_for_high_coverage

Only has an effect if *peak calling* is performed. Proteins covering the whole reference genome can cause problems for PureCLIP causing it to throw an error. From our experience the parameters added by this argument can help PureCLIP with performing it's analysis. Adds following arguments to the PureCLIP execution: `-mtc 5000 -mtc2 5000 -ld`

Usage:

```
--peak_calling_for_high_coverage
```

4.21 --peak_calling_regions

Only has an effect if *peak calling* is performed. If specified peak regions instead of single peaks will be returned by PureCLIP.

Usage:

```
--peak_calling_regions
```

4.22 --peak_calling_regions_width

Only has an effect if *peak calling regions* are stated. Changes the width of peak calling regions returned by PureCLIP.

Usage (default):

```
--peak_calling_regions_width 8
```

4.23 --gene_id

Only has an effect if an *annotation file* is provided and thus the *RNA subtype analysis* performed.
Wording of the tag that describes the gene ID. Is found in the last column of annotation files, typically as the first tag-value pair.
The column looks similar to this:
`ID=gene-LOC101842720;Dbxref=GeneID:101842720;Name=LOC101842720;gbkey=Gene;gene=LOC101842720;gene_biotype=pseudogene;pseudo=true`
In this case the tag necessary is *ID*.

Usage (default):

```
--gene_id ID
```

4.24 --color_barplot

Color of barplots returned by PARANOiD. Affects graphs generated by *peak height distribution*, *RNA subtype analysis* and the experimental barcode distribution. Color is staded via a hexadecimal color code. If unsure which code translates to which color several websites can help to pick the correct one. Example

Usage (default):

```
--color_barplot #69b3a2
```

4.25 --rna_subtypes

Only has an effect if an *annotation file* is provided and thus the *RNA subtype analysis* performed. RNA subtypes/regions that shall be included in the *RNA subtype analysis*. RNA subtypes need to be separated by a , and should appear in the *annotation file* within the **feature type** column (3rd column). If both requirements are not met the analysis will either not be performed correctly or be aborted. If not sure which RNA subtypes are included within your annotation file you can use the script *featuretypes-from-gtfgff.awk*. Additionally, users should beware not to choose subtypes/regions that are in a hierarchical relationship to each other as they can cover the same regions and thus make affected peaks appear as **ambiguous**. Inormation about the hierarchical structure of RNA subtypes/regions can be obtained here.

Usage (default):

```
--rna_subtypes 3_prime_UTR,transcript,5_prime_UTR
```

4.26 --omit_peak_distance

Omits the *peak distance analysis*

Usage:

```
--omit_peak_distance
```

4.27 --distance

Max distance used for the *peak distance analysis*.

Usage (default):

```
--distance 30
```

4.28 --percentile

Peak percentiles for *peak distance analysis* and *sequence extraction/motif analysis*. Only peaks with a value above this threshold are considered while all peaks below are omitted as background noise. A percentile of 90 means that only top 10% of peaks are used.

Usage (default):

```
--percentile 90
```

4.29 --omit_sequence_extraction

Omits the *motif detection*

Usage:

```
--omit_sequence_extraction
```

4.30 --seq_len

Only applies when *motif detection* is performed. Length in nucleotides to each side of a peak that is extracted from the *reference*. A value of 20 will lead to sequences of 41 nucleotides being extracted. (20nt upstream;cross-link nt;20nt downstream)

Usage (default):

```
--seq_len 20
```

4.31 --omit_cl_nucleotide

Only applies when *motif detection* is performed. The nucleotide directly at the cross-linking position will be substituted with an **N** when extracting sequences. Can improve the motif detection since iCLIP tends to have a bias towards **U** when cross-linking which can influence the motif search.

Usage:

```
--omit_cl_nucleotide
```

4.32 --omit_cl_width

Only applies when *motif detection* is performed and the *cl nucleotide is omitted*. Omits nucleotides on both sides of the cross-linking position with an **N** to avoid potential uridine-polymers which can negatively influence the motif search. The number determines the amount of nucleotides on both sides that are to be replaced.

Usage (default):

```
--omit_cl_width 0
```

4.33 --remove_overlaps

Only applies when *motif detection* is performed. Removes cross-link sites with lower peak values if their extracted sequence would overlap with the sequence from another cross-link site. This can be done to avoid doubled sequences during motif detection.

Usage:

```
--remove_overlaps
```

4.34 --max_motif_num

Only applies when *motif detection* is performed. Maximum number of motifs that is reported by streme.

Usage (default):

```
--max_motif_num 50
```

4.35 --min_motif_width

Only applies when *motif detection* is performed. Minimum length of motifs reported by streme. Cannot be lower than 3

Usage (default):

```
--min_motif_width 8
```

4.36 --max_motif_width

Only applies when *motif detection* is performed. Maximum length of motifs reported by streme. Cannot be higher than 30

Usage (default):

```
--max_motif_width 15
```

CHAPTER

FIVE

INCLUDED ANALYSES

Short overview of all analyses implemented in PARANOiD

5.1 Basic analysis

The basic analysis of PARANOiD includes the preprocessing of *FASTQ files*, demultiplexing, aligning reads to a reference and calculating the cross-linking position based on the alignment. These positions are then translated into *WIG, BIGWIG and BEDGRAPH files* and given as output to the user. Additionally, a distribution of the *peak height* and the *stradnedness* are given as output together with a *statistics overview* of important proceses. Lastly, an *XML file* is generated that can be importet by the *IGV <https://software.broadinstitute.org/software/igv/>* to automatically visualize results generated by PARANOiD. The preprocessing involves adapter removal, quality filtering, splitting reads according to their experimental barcode and removing the whole barcode. The alignmentcan be done via *2 different alignment tools (Bowtie2 or STAR)* and is followed by a deduplication step in which PCR duplicates are removed. Finally, alignments are filtered via the *MAPQ score* and cross-linking positions are calculated for each alignment. *2 WIG, BIGWIG and BEDGRAPH files* are generated for each sample - one for forward and one for reverse alignments.

```
Associated parameters (preprocessing)
--barcode_pattern           States composition of random and experimental barcodes
--barcode_mismatches        Number of mismatches allowed mismatches within experimental␣
↪barcode to still align it to it's sample
--min_length                Minimum length of reads necessary to retain after adapter␣
↪removal
--min_qual                  Minimum quality a base needs to retain
--min_percent_qual_filter   Percentage of bases above the quality threshold necessary to␣
↪retain the read

Associated parameters (alignment & cross link site determination)
--domain                  States if Bowtie2 or STAR is being used as aligner
--mapq                    Minimum MAPQ score for alignments necessary to retain
--max_alignments          Maximum number of alignments provided by the mapping tool
--report_all_alignments   Reports all possible alignments (might be filtered out later␣
↪on)
```

5.2 Merge replicates

Merges several replicates into a single representative version which can be used for publications, posters or presentations. This version shows the mean hit count for every position. Additionally, a correlation analysis is performed to give the user an evaluation of the sample similarity and therefore a rationale for this analysis. The correlation is performed on raw cross-link sites (or on significant ones in case *peak calling* is used) via the Pearson correlation. Is deactivated by default.

```
Associated parameters:
--merge_replicates          Merges replicates according to the name in the barcode file
--correlation_analysis      Does a correlation analysis for merged replicates
```

5.3 RNA subtypes

Analysis to determine if the protein of interest is prone to bind to specific RNA subtypes or regions. As this is determined via the *annotation file* only subtypes included there can be determined (shown in column 3). To see which RNA subtypes are included in the annotation file a *script* was added. When choosing RNA subtypes one has to be careful not to use subtypes that are hierarchically higher or lower to each other as these will at least partially cover the same reference regions making hits in these regions ambiguous. The SO ontologies can be used to get an overview of the official hierarchical structures of annotation files. Is activated when an *annotation file* is provided.

```
Associated parameters
--gene_id               Tag for the gene ID used within the annotation file
--color_barplot         Color bars within the barplot generated by this analysis
--rna_subtypes          RNA subtypes/regions used for this analysis
```

5.4 Transcript analysis

Analysis to show if specific RNAs are more prone to interact with the the protein of interest. If choosing this analysis a file containing all RNAs of interest should be used as input reference instead of the genome. Here all RNAs of interest (or artificial RNAs present in the sample) can be combined to a single fasta file. If the general transcriptome of an organism shall be examined, they can often be accessed next to the genome and annotation of the organism. If not a FASTA file containing the transcripts can be generated as follows (needs the genome and an annotation file):

''' gffread -w output_transcripts.fa -g input_reference_genome.fa input_annotation.gff3 '''

```
Associated parameters
--map_to_transcripts            Activates transcript analysis
--number_top_transcripts        Amount of transcripts with most hits per sample that␣
↪are offered as output
```

5.5 Peak calling

Results obtained from analyzed iCLIP experiments typically contain a fair amount of background noise (signal not caused by the actual protein-RNA interaction). This can be due to the reverse transcription not terminating when encountering an aminoacid or by a covalent binding of the protein of interest with an RNA just because their were in close proximity. Peak calling is supposed to filter out this background noise and thus reduce the amount of false positive signal. PARANOiD employs PureCLIP for its peak calling process. PureCLIP uses a hidden Markov model to divide the reference into 4 different states based on the peak distribution. Additionally, identified peaks in close proximity can be merged into binding regions.

```
Associated parameters:
--omit_peak_calling                          Omits peak calling analysis
--peak_calling_for_high_coverage             Adds parameters to PureCLIP which can␣
↪allow it's succesful execution for high coverage samples
--peak_calling_regions                       Allows merging several cross link sites in␣
↪close proximity to a cross link region
--peak_calling_regions_width                 Sets the width until which cross link␣
↪sites in close proximity are allowed to be merged
```

5.6 Motif detection

Protein binding sites are often determined by protein-specific RNA motifs. These motifs are typically found at or in close proximity to cross-linking sites. To identify these motifs the motif detection was implemented. Background noise is being filtered out by using only the top percentiles of peaks (by default only the top 10% are used) in the same manner as in the *peak distance analysis*. Sequences around all peaks above the threshold are extracted and provided as output. All extracted sequences are then used for motif detection via streme, which offers several enriched sequences.

```
Associated parameters:
--omit_sequence_extraction          Omits the sequence extraction and motif detection
--percentile                        Sets threshold for peak values used for this␣
↪analysis using percentiles
--seq_len                           Nucleotides extracted from each side of a cross link␣
↪site
--omit_cl_nucleotide                Omits the nucleotide at the cross link position
--omit_cl_width                     Omits the nucleotides surrounding the cross link␣
↪position
--remove_overlaps                   Removes overlapping sequences
--max_motif_num                     Maximum number of motifs generated
--min_motif_width                   Minimum width allowed for motifs
--max_motif_width                   Maximum width allowed for motifs
```

5.7 Peak distance analysis

Some proteins bind to long stretches of RNA instead of certain motif-dependent RNA subregions. This is, for example, the case with the Nucleocapsid (N) protein of several virus species which bind to a distinct number of nucleotides per N protein while packaging the viral RNA. The peak distance analysis was implemented to detect such periodical RNA-protein interactions by determining the occurences of distances between peaks. Background noise is being filtered out by using only the top percentiles of peaks (by default only the top 10% are used) in the same manner as in the *motif detection*. Then, going through every peak above the threshold, the distances to all other peaks above this threshold, which are within a certain distance (by default 30 nt) are measured, summarized and provided as a TSV file and visualized as a plot.

```
Associated parameters:
--omit_peak_distance     Omits the peak distance analysis
--percentile             Sets threshold for peak values used for this analysis using␣
↪percentiles
```

CHAPTER

SIX

PARANOID OUTPUTS

Explanation of all outputs generated by PARANOiD

6.1 Alignments

Directory that contains deduplicated alignments in `BAM` format together with an index file in `BAM.BAI` format. BAM files are compressed binary forms of SAM files. SAM/BAM files are tab separated and show one alignment per line. The information shown by the columns go as follows: 1. Read header 2. Bitwise FLAG 3. Name of reference sequence 4. Position of alignment (1-based) 5. MAPQ-score 6. CIGAR string 7. Name of mate read (shows * if information is not available) 8. Position of mate read (shows 0 if information is not available) 9. Length of alignment on the reference (shows 0 if information is not available) 10. Read sequence (shows * if information is not available) 11. Quality of read sequence (shows * if information is not available)

One of each is generated per sample.

Is included in the *basic analysis*.

Example:

```
NB501399:129:HLW7VAFX2:3:11409:5471:17963_AAGACACTG     272     1       14572   0       ␣
↪23M    *       0       0       CCACACAGTGCTGGTTCCGTCAC EEEEEEEEEEEAEEEEEEEEEE NH:i:7␣
↪ HI:i:4  AS:i:22 nM:i:0
NB501399:129:HLW7VAFX2:3:11604:9407:1314_TCTGCCCAC      272     1       14747   0       ␣
↪36M    *       0       0       CGGCAGAGGAGGGATGGAGTCTGACACGCGGGCAAA   ␣
↪EEEEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEE    NH:i:5  HI:i:4  AS:i:35 nM:i:0
NB501399:129:HLW7VAFX2:2:11201:6526:7382_TCCCCGACC      272     1       14847   0       ␣
↪40M    *       0       0       AGTGAGGGTGGTTGGTGGGAAACCCTGGTTCCCCCAGCCC        ␣
↪EEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEE        NH:i:6  HI:i:3  AS:i:39 nM:i:0
NB501399:129:HLW7VAFX2:1:11204:3841:14476_GCGATCCCG     272     1       14992   0       ␣
↪37M    *       0       0       GTTGAAGAGATCCGACATCAAGTGCCCACCTTGGCTC  ␣
↪EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE   NH:i:8  HI:i:5  AS:i:36 nM:i:0
NB501399:129:HLW7VAFX2:2:11204:16119:17944_CACACCCCG    272     1       14992   0       ␣
↪37M    *       0       0       GTTGAAGAGATCCGACATCAAGTGCCCACCTTGGCTC  ␣
↪EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE   NH:i:8  HI:i:5  AS:i:36 nM:i:0
NB501399:129:HLW7VAFX2:1:21211:6880:4260_CCACAACTC      272     1       15923   0       ␣
↪1S25M659N10M    *       0       0       GACCACTTCCCTGGGAGCTCCCTGGACTGAAGGAGA   ␣
↪AEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE    NH:i:7  HI:i:3  AS:i:35 nM:i:0
```

6.2 Cross link sites peak called

6.3 Raw cross link sites

Directory that contains unmodified cross-link sites with all background noise remaining. Cross-link sites are provided in 3 different formats, which are separated in one directory each; `WIG`, `BIGWIG` and `BEDGRAPH`. Each format represents the same data. Is included in the *basic analysis*.

6.3.1 WIG (Wiggle)

Format to represent genome-wide coverage that consists of one line per reference chromosome with the coverage listed below each in a tab separated manner. Column 1 represents the position while column 2 represents the coverage at the current position. For each sample 2 WIG files are generated - one representing cross-link events on the forward ond one on the reverse string which can be distinguished by the name. The amount of cross-link events on the reverse strand is displayed as negative.

```
variableStep chrom=reference_1 span=1
2815    1.0
3726    1.0
3895    1.0
6201    1.0
6367    1.0
variableStep chrom=reference_2 span=1
22  1.0
31  1.0
66  1.0
80  1.0
```

6.3.2 BIGWIG

An extension of the previously mentioned WIG format. While WIG uses plain text BIGWIG uses a binary format to store the data, reducing the file size. Therefore, accessing the data requires specialized software such as the IGV.

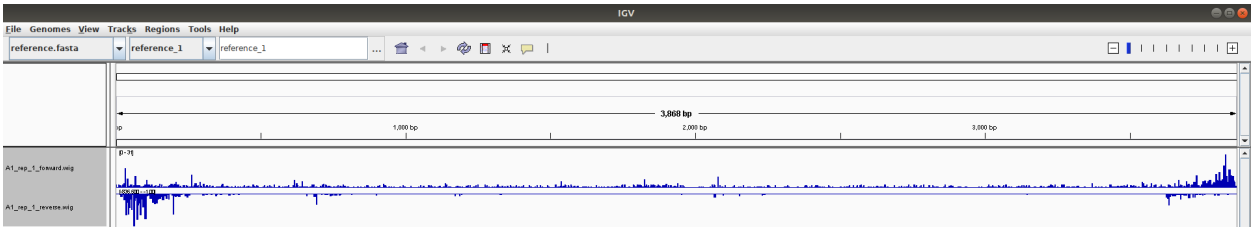
6.3.3 BEDGRAPH

Similar format to WIG or BIGWIG. BEDGRAPH files consist of 4 columns: 1. The chromosome name 2. The start position of the described events 3. The end position of the described events (for PARANOiD this is the position of the actual cross-link event) 4. Coverage of currently described event (negative for reverse strand)

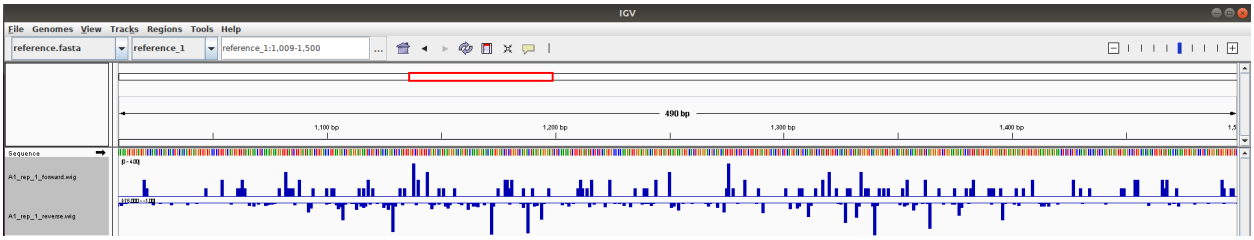
```
DQ375404.1  2814    2815    1
DQ375404.1  3725    3726    1
DQ375404.1  3894    3895    1
DQ375404.1  6200    6201    1
DQ375404.1  6366    6367    1
DQ380154.1  21      22      1
DQ380154.1  30      31      1
DQ380154.1  65      66      1
DQ380154.1  79      80      1
```

6.3.4 Visualization with IGV

All provided file types can be easily visualized via the Integrative Genomics Viewer (IGV). To do so first the reference sequences need to be loaded into IGV. This is done by clicking on the tab called *Genomes* - which is located on the top left corner - and then choose the origin of the reference genome.



The reference track can be used to zoom in allowing users to see cross-link sites more detailed.



6.4 Cross link sites merged

6.5 Execution metrics

Directory that contains general execution metrics of the workflow such as:

1. **container_information.txt**
   Container system used to execute the processes together with the containers that were used during the workflow
2. **execution_information.txt**

   Contains information necessary to reproduce the results such as

   a. Command used for the execution

   b. Directory of PARANOiD

   c. Config file used

   d. Profiles used

   e. Version od Nextflow and PARANOiD

   f. Execution directory
3. **parameter_information.txt**
   Contains all parameters used

Is included in the *basic analysis*.

6.6 IGV-session

An `XML` file that can be transferred directly to the IGV. This can be done by clicking on the *data* tab on the top left and then on *Open Session*. A window will open in which you can navigate to the output directory of PARANOiD and choose *igv-session.xml*. This will open a predefined IGV session that includes the reference, the cross-link sites of all samples (forward and reverse) and the alignment files of all samples. If the option *--merge_replicates* was chosen then only the merged cross-link sites are shown. Is included in the *basic analysis*.

6.7 Peak height distribution

Is included in the *basic analysis*.

6.8 Reference

The *reference sequence* provided as input.

Is included in the *basic analysis*.

6.9 Statistics

Is included in the *basic analysis*.

6.10 Strand distribution

Is included in the *basic analysis*.

CHAPTER

SEVEN

FILES PRESENT IN PARANOID

This is an overview of the files and directories that come with PARANOiD.

1. *bin*
2. *dockerfiles*
3. *docs*
4. *modules*
5. *build_docker.sh*
6. *featuretypes-from-gtfgff.awk*
7. *LICENSE*
8. *LICENSE.pybam*
9. *main.nf*
10. *PARANOiD-deprecated-DSL1.nf*
11. *pull_images.sh*
12. *README.md*

7.1 bin

Directory that mainly consists of custom scripts that are needed for several PARANOiD steps. This directory is only necessary if no containers are used to execute PARANOiD. Typically there is no need for users to interact with files in this directory.

7.2 dockerfiles

Directory that contains dockerfiles from which container images can be built if necessary. Images built from these dockerfiles can be used to generate containers for every step executed by PARANOiD (except PureCLIP). Typically there is no need for users to interact with files in this directory.

7.3 docs

Directory that contains files necessary to build and display this documentation. Typically there is no need for users to interact with files in this directory.

7.4 modules

Directory that contains all nextflow modules included by PARANOiD. These modules are a collection of processes that can be included in nextflow. Each process describes the implementation of a specific step together with the necessary and optional inputs and the generated outputs. Typically there is no need for users to interact with files in this directory.

7.5 build_docker.sh

Shell script that can be used to automatically build images from all docker files included in the *correspondent directory* and upload them to docker hub. Typically there is no need for users to interact with files in this directory.

7.6 featuretypes-from-gtfgff.awk

Short awk script that can be used to get all feature-types described within a gtf or gff file. Can be useful for the *RNA subtype analysis* as it needs the exact subtype names. Usage can be found *here*.

7.7 LICENSE

MIT copyright declaration. Basically says that PARANOiD can be used however you please. You can copy, change and publish this software or parts of it as long as it is under MIT copyright.

7.8 LICENSE.pybam

Apache copyright declaration which is only valid for pybam, which is used in the process of generating cross-link pile ups from bam files after the alignment. The Apache copyright allows you to use or change the software as much as you want, as long as you do it under the Apache copyright and make notices on all altered files.

7.9 main.nf

Nextflow script to *run* when starting a PARANOiD anaylsis. Uses processes described within the *modules directory* and connects them in the right order and with the correct logic to form the pipeline.

7.10 nextflow.config

Config file that is automatically used by PARANOiD (given that it is present in the same directory as the *main.nf script*). Consists of 3 parts:

7.10.1 Parameters

A list of all *parameters* usable when running PARANOiD together and their default values. Default parameters can be adapted by users to better suit their needs.

7.10.2 Profiles

Describes usage of *container executors* and *cluster distribution*. The specifications should work on most systems but there is a possibility that they need to be adapted if errors related to the profiles arise.

7.10.3 Resource allocations

Describes the computational resources that will be required to run each process. The current resource requirements are chosen in order to work for most datasets and might not be necessary for all datasets. In some cases they might even be set too low; it depends on the size of the *read file* and the *reference*. However, they can (and in some cases should) be adapted if the used system does not meet the required resources which are currently set to 8 cores and 100 GB RAM. If PARANOiD will be executed on a local computer with less resources available than necessary, the resource requirements can be adapted in this file. Lowering the required resources can also increase the computing speed as more processes are allowed to be run in parallel. In this case the file *nextflow.config* can be opened via a text editor and the relevant resource requirements changed. The most relevant processes will be 'build_index_STAR|mapping_STAR' as they require the highest amount of resources. When opening the config file the relevant entry looks like this:

```
withName: 'build_index_STAR|mapping_STAR' {
            cpus = 8
            memory = '100 GB'
            container = 'docker://pbarth/star:1.0'
    }
```

To change the required cores the number after **cpus = ** needs to be changed - to lower it to 4 cores it should be **cpus = 4**. To change the required memory the number after **memory = ** needs to be changed - to lower it to 50 GB it should be **memory = '50 GB'**.

7.11 PARANOiD-deprecated-DSL1.nf

An older version of PARANOiD that uses DSL1 instead of the later DSL2. Should not be used as it is already deprecated and will not receive any updates in future.

7.12 pull_images.sh

Shell script that can be used to download all images used to build containers by PARANOiD into a specific directory. Can be used as preparation if PARANOiD is supposed to be run without internet connection. Additional information on how t run the script can be found *here*.

7.13 README.md

Readme displayed on github. Typically there is no need for users to interact with this file.

CHAPTER

EIGHT

CONTAINER USAGE

PARANOiD offers the usage of docker containers via several different executors.
Currently supported are Docker, Podman and Singularity.
As containers are handled via Nextflow the download should start automatically when running the pipeline.
The default directory for images is within the work directory generated by Nextflow.
If problems occur while downloading images via Nextflow the script *pull_images.sh* can be used to preload them.
It is recommended to run PARANOiD with containers to ensure correct versioning of tools.
If several profiles are used (e.g. using singularity together with SLURM) they are separated by a single `,` (`-profile singularity,slurm`)

8.1 Docker

Uses Docker to run processes within containers

```
-profile docker
```

8.2 Podman

Uses Podman to run processes within containers

```
-profile podman
```

8.3 Singularity

Uses Singularity to run processes within containers

```
-profile singularity
```

CHAPTER

NINE

CLUSTER USAGE

PARANOiD supports the distribution of jobs to clusters via job schedulers.
Currently supported are SGE and SLURM, but the list can potentially expanded upon request.
Using job scheduling systems to distribute jobs can immensly shorten execution time.
If several profiles are used (e.g. using singularity together with SLURM) they are separated by a single `,` (`-profile singularity,slurm`)

9.1 SGE

Uses SGE to distribute jobs.

```
-profile sge
```

9.2 SLURM

Uses SLURM to distribute jobs.

```
-profile slurm
```

CHAPTER

TEN

SUPPLEMENTARY SCRIPTS FOR PARANOID

Supplementary scripts for PARANOiD

10.1 Determine valid RNA subtypes

Script added to determine valid RNA subtypes for the *RNA subtype analysis*.
Script name: featuretypes-from-gtfgff.awk

Usage:

```
featuretypes-from-gtfgff.awk /path/to/anntotation_file.gff
```

10.2 Pull images

Pulls images via singularity. Images are used to build the containers used by processes.
Should be used to pull all images before starting PARANOiD.

Usage:

```
pull_images.sh /path/to/PARANOiD/dockerfiles /path/to/image_directory
```

CHAPTER

ELEVEN

REQUIREMENTS

Requirements to run PARANOiD. Note that most are fulfilled by containers and it's thus recommended to use one of the options stated in the *container section*. All versions shown here ae fixed with PARANOiD version 1.0 and will only change with future updates.

11.1 Essential

Tool	Version	Note
Nextflow	23.04.1.5866	

11.2 Requirements when run without container

Tool	Version	Note
fastqc	0.11.9	
cutadapt	4.2	
trim_galore	0.6.7	
fastx_toolkit	0.0.14	
umi_tools	1.1.4	
python	3.11	
samtools	1.16.1	
bamtools	2.5.2	
wigToBigWig	2.9	
bigWigToBedGraph		
Bowtie2	2.5.1	Only when using *--domain pro* or running the *transcript analysis*
STAR	2.7.10b	Only when using *--domain eu*
subread	2.0.3	
pureCLIP	1.3.1	Only when using *peak calling*
multiqc	1.13	
pysam	0.19.1	
R	4.0.3	
optparse		R package
wig		R package
reshape2		R package
ggplot2		R package
numpy		python package
biopython		python package
gff3sort.pl	1.0.0	Only when providing an *annotation file*
bgzip	1.16	Only when providing an *annotation file*
tabix	1.16	Only when providing an *annotation file*
meme	5.4.1	Only when using *motif detection*

CHAPTER

TWELVE

FREQUENTLY ASKED QUESTIONS