Flair

Alison Tang, Cameron Soulette, Jeltje van Baren, Kevyn Hart, Eva

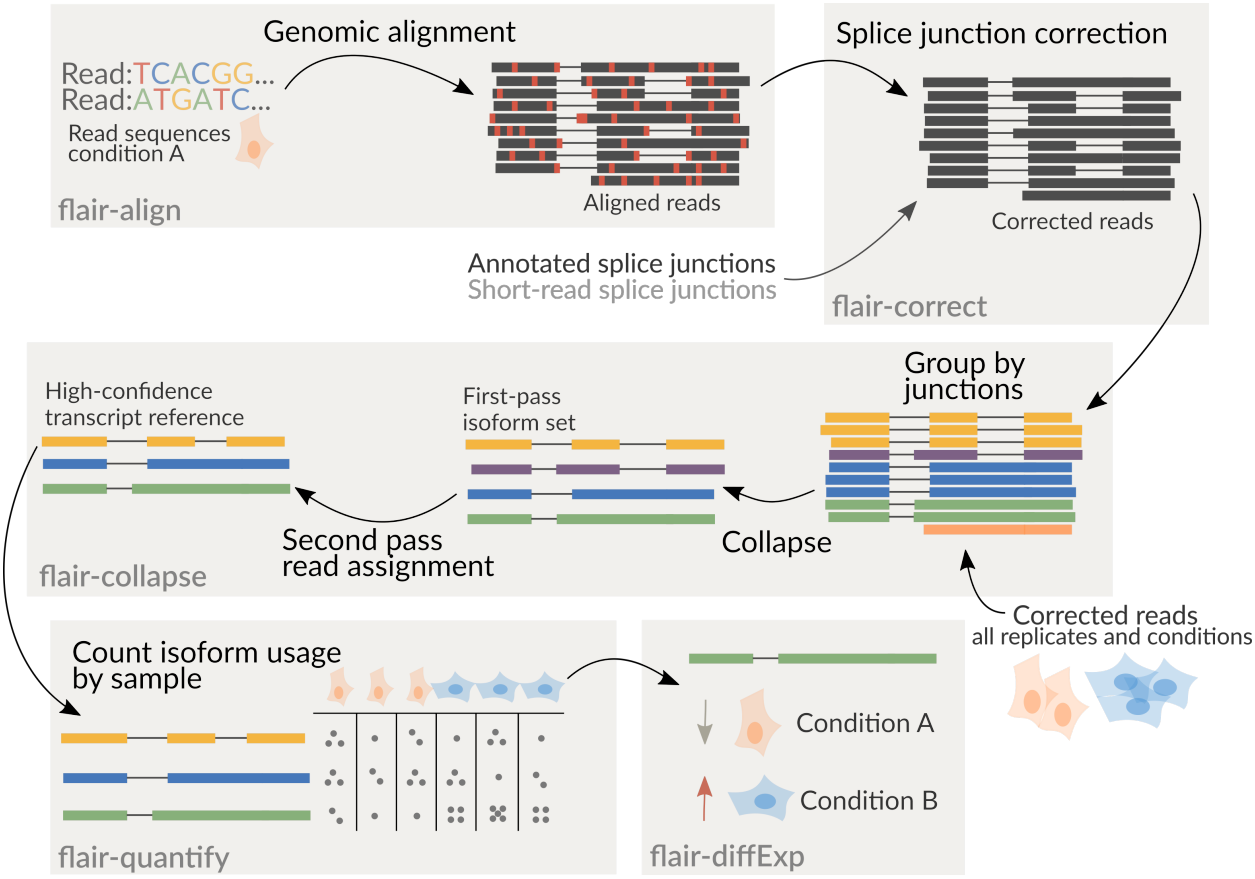
Sep 13, 2023

CONTENTS:

New: Flair can now be conda installed using

```
conda create -n flair -c conda-forge -c bioconda flair
conda activate flair
```

FLAIR can be run optionally with short-read data to help increase splice site accuracy of the long read splice junctions. FLAIR uses multiple alignment steps and splice site filters to increase confidence in the set of isoforms defined from noisy data. FLAIR was designed to be able to sense subtle splicing changes in nanopore data from Tang et al. (2020). Please read for more description of the methods.



It is recommended to combine all samples together prior to running flair-collapse for isoform assembly by concatenating corrected read `psl` or `bed` files together. Following the creation of an isoform reference from flair-collapse, consequent steps will assign reads from each sample individually to isoforms of the combined assembly for downstream analyses.

It is also good to note that `bed12` and `psl` can be converted using kentUtils bedToPsl or pslToBed, or using `bed_to_psl` and `psl_to_bed` provided in flair's `/bin` directory.

CHAPTER

ONE

INSTALLING FLAIR

The easiest way to install Flair and all of its dependencies is via conda:

```
conda create -n flair -c conda-forge -c bioconda flair
conda activate flair
flair [align/correct/...]
```

For other methods, please see the Other environments section

CHAPTER

TWO

MODULES

`flair` is a wrapper script with modules for running various processing scripts located in `src/flair`. Modules are assumed to be run in order (align, correct, collapse), but can be run separately.

2.1 flair align

```
usage: flair align -g genome.fa -r <reads.fq>|<reads.fa> [options]
```

This module aligns reads to the genome using minimap2, and converts the SAM output to BED12. Aligned reads in BED12 format can be visualized in IGV or the UCSC Genome browser.

Outputs

- `flair.aligned.bam`
- `flair.aligned.bam.bai`
- `flair.aligned.bed`

2.1.1 Options

Required arguments

```
--reads     Raw reads in fasta or fastq format. This argument accepts multiple
            (comma/space separated) files.

At least one of the following arguments is required:
--genome    Reference genome in fasta format. Flair will minimap index this file
            unless there already is a .mmi file in the same location.
--mm_index  If there already is a .mmi index for the genome it can be supplied
            directly using this option.
```

Optional arguments

```
--help              Show all options.
--output            Name base for output files (default: flair.aligned). You can supply
                    an output directory (e.g. output/flair_aligned) but it has to exist;
                    Flair will not create it. If you run the same command twice, Flair
                    will overwrite the files without warning.
--threads           Number of processors to use (default 4).
--junction_bed      Annotated isoforms/junctions bed file for splice site-guided
                    minimap2 genomic alignment.
--nvrna             Use native-RNA specific alignment parameters for minimap2 (-u f -k␣
↪14)
--quality           Minimum MAPQ score of read alignment to the genome. The default is 1,
                    which is the lowest possible score.
-N                  Retain at most INT secondary alignments from minimap2 (default 0).␣
↪Please
                    proceed with caution, changing this setting is only useful if you␣
↪know
                    there are closely related homologs elsewhere in the genome. It will
                    likely decrease the quality of Flair's final results.
--quiet             Dont print progress statements.
```

2.1.2 Notes

If you're using human sequences, the best reference genome is GCA_000001405.15_GRCh38_no_alt_analysis_set as described in this helpful blog post by Heng Li

If your input sequences are Oxford nanopore reads, please use Pychopper before running Flair.

If your reads are already aligned, you can convert the sorted `bam` output to `bed12` using `bam2Bed12` to supply for flair-correct. This step smoothes gaps in the alignment.

nvrna settings: See minimap2's manual for details.

quality: More info on MAPQ scores

2.2 flair correct

```
usage: flair correct -q query.bed12 [-f annotation.gtf]|[-j introns.tab] -g genome.fa␣
↪[options]
```

This module corrects misaligned splice sites using genome annotations and/or short-read splice junctions.

Outputs

- `<prefix>_all_corrected.bed` for use in subsequent steps
- `<prefix>_all_inconsistent.bed` rejected alignments
- `<prefix>_cannot_verify.bed` (only if the) chromosome is not found in annotation

2.2.1 Options

Required arguments

```
--query             Uncorrected bed12 file, e.g. output of flair align.
--genome            Reference genome in fasta format.

At least one of the following arguments is required:
--shortread         Bed format splice junctions from short-read sequencing. You can
                    generate these from SAM format files using the junctions_from_sam
                    program that comes with Flair.
--gtf               GTF annotation file.
```

Optional arguments

```
--help              Show all options
--output            Name base for output files (default: flair). You can supply an
                    output directory (e.g. output/flair) but it has to exist; Flair
                    will not create it. If you run the same command twice, Flair will
                    overwrite the files without warning.
--threads           Number of processors to use (default 4).
--nvrna             Specify this flag to make the strand of a read consistent with
                    the input annotation during correction.
--ss_window         Window size for correcting splice sites (default 15).
--print_check       Print err.txt with step checking.
```

2.2.2 Notes

Make sure that the genome annotation and genome sequences are compatible (if the genome sequence contains the 'chr' prefix, the annotations must too).

Please do use GTF instead of GFF; annotations should not split single exons into multiple entries.

2.3 flair collapse

```
usage: flair collapse -g genome.fa -q <query.bed> -r <reads.fq>/<reads.fa> [options]
```

Defines high-confidence isoforms from corrected reads. As FLAIR does not use annotations to collapse isoforms, FLAIR will pick the name of a read that shares the same splice junction chain as the isoform to be the isoform name. It is recommended to still provide an annotation with `--gtf`, which is used to rename FLAIR isoforms that match isoforms in existing annotation according to the transcript_id field in the gtf.

Intermediate files generated by this step are removed by default, but can be retained for debugging purposes by supplying the argument `--keep_intermediate` and optionally supplying a directory to keep those files with `--temp_dir`.

If there are multiple samples to be compared, the flair-corrected read `bed` files should be concatenated prior to running flair-collapse. In addition, all raw read fastq/fasta files should either be specified after `--reads` with space/comma separators or concatenated into a single file.

Please note: Flair collapse is not yet capable of dealing with large (>1G) input bed files. If you find that Flair needs a lot of memory you may want to split the input bed file by chromosome and run these separately. We do intend to improve this.

Outputs

- `isoforms.bed`
- `isoforms.gtf`
- `isoforms.fa`

If an annotation file is provided, the isoforms ID format will contain the transcript id, underscore, and then the gene id, so it would look like `ENST*_ENSG*` if you're working with the GENCODE human annotation.

If multiple TSSs/TESs are allowed (toggle with `--max_ends` or `--no_redundant`), then a `-1` or higher will be appended to the end of the isoform name for the isoforms that have identical splice junction chains and differ only by their TSS/TES.

For the gene field, the gene that is assigned to the isoform is based on whichever annotated gene has the greatest number of splice junctions shared with the isoform. If there are no genes in the annotation which can be assigned to the isoform, a genomic coordinate is used (e.g. `chr*:100000`).

2.3.1 Options

Required arguments

```
--query     Bed file of aligned/corrected reads
--genome    FastA of reference genome
--reads     FastA/FastQ files of raw reads, can specify multiple files
```

Optional arguments

```
--help              Show all options.
--output            Name base for output files (default: flair.collapse).
                    You can supply an output directory (e.g. output/flair_collapse)
--threads           Number of processors to use (default: 4).
--gtf               GTF annotation file, used for renaming FLAIR isoforms to
                    annotated isoforms and adjusting TSS/TESs.
--generate_map      Specify this argument to generate a txt file of read-isoform
                    assignments (default: not specified).
--annotation_reliant        Specify transcript fasta that corresponds to transcripts
                    in the gtf to run annotation-reliant flair collapse; to ask flair
                    to make transcript sequences given the gtf and genome fa, use
                    --annotation_reliant generate.
```

Options for read support

```
--support           Minimum number of supporting reads for an isoform; if s < 1,
                    it will be treated as a percentage of expression of the gene
                    (default: 3).
--stringent         Specify if all supporting reads need to be full-length (80%
                    coverage and spanning 25 bp of the first and last exons).
--check_splice      Enforce coverage of 4 out of 6 bp around each splice site and
```

(continues on next page)

(continued from previous page)

```
                    no insertions greater than 3 bp at the splice site. Please note:
                    If you want to use --annotation_reliant as well, set it to
                    generate instead of providing an input transcripts fasta file,
                    otherwise flair may fail to match the transcript IDs.
                    Alternatively you can create a correctly formatted transcript
                    fasta file using gtf_to_psl
--trust_ends        Specify if reads are generated from a long read method with
                    minimal fragmentation.
--quality           Minimum MAPQ of read assignment to an isoform (default: 1).
```

Variant options

```
--longshot_bam      BAM file from Longshot containing haplotype information for each␣
↪read.
--longshot_vcf      VCF file from Longshot.
```

For more information on the Longshot variant caller, see its github page

Transcript starts and ends

```
--end_window        Window size for comparing transcripts starts (TSS) and ends
                    (TES) (default: 100).
--promoters         Promoter regions bed file to identify full-length reads.
--3prime_regions    TES regions bed file to identify full-length reads.
--no_redundant      <none,longest,best_only> (default: none). For each unique
                    splice junction chain, report options include:
                            - none          best TSSs/TESs chosen for each unique
                                            set of splice junctions
                            - longest       single TSS/TES chosen to maximize length
                            - best_only     single most supported TSS/TES
--isoformtss        When specified, TSS/TES for each isoform will be determined
                    from supporting reads for individual isoforms (default: not
                    specified, determined at the gene level).
--no_gtf_end_adjustment     Do not use TSS/TES from the input gtf to adjust
                    isoform TSSs/TESs. Instead, each isoform will be determined
                    from supporting reads.
--max_ends          Maximum number of TSS/TES picked per isoform (default: 2).
--filter            Report options include:
                            - nosubset      any isoforms that are a proper set of
                                            another isoform are removed
                            - default       subset isoforms are removed based on support
                            - comprehensive default set + all subset isoforms
                            - ginormous     comprehensive set + single exon subset
                                            isoforms
```

Other options

```
--temp_dir          Directory for temporary files. use "./" to indicate current
                    directory (default: python tempfile directory).
--keep_intermediate         Specify if intermediate and temporary files are to
                    be kept for debugging. Intermediate files include:
                    promoter-supported reads file, read assignments to
                    firstpass isoforms.
```

(continues on next page)

(continued from previous page)

```
--fusion_dist       Minimium distance between separate read alignments on the
                    same chromosome to be considered a fusion, otherwise no reads
                    will be assumed to be fusions.
--mm2_args          Additional minimap2 arguments when aligning reads first-pass
                    transcripts; separate args by commas, e.g. --mm2_args=-I8g,--MD.
--quiet             Suppress progress statements from being printed.
--annotated_bed     BED file of annotated isoforms, required by --annotation_reliant.
                    If this file is not provided, flair collapse will generate the
                    bedfile from the gtf. Eventually this argument will be removed.
--range             Interval for which to collapse isoforms, formatted
                    chromosome:coord1-coord2 or tab-delimited; if a range is specified,
                    then the --reads argument must be a BAM file and --query must be
                    a sorted, bgzip-ed bed file.
```

2.3.2 Suggested uses

Human

```
flair collapse -g genome.fa --gtf gene_annotations.gtf -q reads.flair_all_corrected.bed -
↪r reads.fastq
--stringent --check_splice --generate_map --annotation_reliant generate
```

For novel isoform discovery in organisms with more unspliced transcripts and more overlapping genes, we recommend using a combination of options to capture more transcripts. For example:

Yeast

```
flair collapse -g genome.fa --gtf gene_annotations.gtf -q reads.flair_all_corrected.bed -
↪r reads.fastq
--stringent --no_gtf_end_adjustment --check_splice --generate_map --trust_ends
```

Note that if you are doing direct-RNA, this command will likely call degradation products as isoforms. If you want to avoid this this we recommend using –annotation-reliant.

2.4 flair quantify

```
usage: flair quantify -r reads_manifest.tsv -i isoforms.fa [options]
```

Output

Isoform-by-sample counts file that can be used in the flair_diffExp and flair_diffSplice programs.

2.4.1 Options

Required arguments

```
--isoforms          Fasta of Flair collapsed isoforms
--reads_manifest    Tab delimited file containing sample id, condition, batch,
                    reads.fq, where reads.fq is the path to the sample fastq file.
```

Reads manifest example:

```
sample1      condition1      batch1  mydata/sample1.fq
sample2      condition1      batch1  mydata/sample2.fq
sample3      condition1      batch1  mydata/sample3.fq
sample4      condition2      batch1  mydata/sample4.fq
sample5      condition2      batch1  mydata/sample5.fq
sample6      condition2      batch1  mydata/sample6.fq
```

Note: Do **not** use underscores in the first three fields, see below for details.

Optional arguments

```
--help              Show all options
--output            Name base for output files (default: flair.quantify). You
                    can supply an output directory (e.g. output/flair_quantify).
--threads           Number of processors to use (default 4).
--temp_dir          Directory to put temporary files. use ./ to indicate current
                    directory (default: python tempfile directory).
--sample_id_only    Only use sample id in output header instead of a concatenation
                    of id, condition, and batch.
--quality           Minimum MAPQ of read assignment to an isoform (default 1).
--trust_ends        Specify if reads are generated from a long read method with
                    minimal fragmentation.
--generate_map      Create read-to-isoform assignment files for each sample.
--isoform_bed       isoform .bed file, must be specified if --stringent or
                    --check-splice is specified.
--stringent         Supporting reads must cover 80% of their isoform and extend
                    at least 25 nt into the first and last exons. If those exons
                    are themselves shorter than 25 nt, the requirement becomes
                    'must start within 4 nt from the start' or 'end within 4 nt
                    from the end'.
--check_splice      Enforces coverage of 4 out of 6 bp around each splice site
                    and no insertions greater than 3 bp at the splice site.
```

2.4.2 Other info

Unless `--sample_id_only` is specified, the output counts file concatenates id, condition and batch info for each sample. flair_diffExp and flair_diffSplice expect this information.

```
id   sample1_condition1_batch1  sample2_condition1_batch1  sample3_condition1_batch1 ␣
↪sample4_condition2_batch1  sample5_condition2_batch1  sample6_condition2_batch1
ENST00000225792.10_ENSG00000108654.15   21.0    12.0    10.0    10.0    14.0    13.0
ENST00000256078.9_ENSG00000133703.12    7.0     6.0     7.0     15.0    12.0    7.0
```

2.5 flair_diffExp

IMPORTANT NOTE: diffExp and diffSplice are not currently part of the main flair code. Instead they are supplied as separate programs named flair_diffExp and flair_diffSplice. They take the same inputs as before.

```
usage: flair_diffExp -q counts_matrix.tsv --out_dir out_dir [options]
```

This module performs differential *expression* and differential *usage* analyses between **exactly two** conditions with 3 or more replicates. It does so by running these R packages:

- DESeq2 on genes and isoforms. This tests for differential expression.
- DRIMSeq is used on isoforms only and tests for differential usage. This is done by testing if the ratio of isoforms changes between conditions.

If you do not have replicates you can use the *diff_iso_usage* standalone script.

If you have more than two sample condtions, either split your counts matrix ahead of time or run DESeq2 and DRIMSeq yourself.

Outputs

After the run, the output directory (`--out_dir`) contains the following, where COND1 and COND2 are the names of the sample groups.

- `genes_deseq2_MCF7_v_A549.tsv` Filtered differential gene expression table.
- `genes_deseq2_QCplots_MCF7_v_A549.pdf` QC plots, see the DESeq2 manual for details.
- `isoforms_deseq2_MCF7_v_A549.tsv` Filtered differential isoform expression table.
- `isoforms_deseq2_QCplots_MCF7_v_A549.pdf` QC plots
- `isoforms_drimseq_MCF7_v_A549.tsv` Filtered differential isoform usage table
- `workdir` Temporary files including unfiltered output files.

2.5.1 Options

Required arguments

```
--counts_matrix     Tab-delimited isoform count matrix from flair quantify
--out_dir           Output directory for tables and plots.
```

Optional arguments

```
--help              Show this help message and exit
--threads           Number of threads for parallel DRIMSeq.
--exp_thresh        Read count expression threshold. Isoforms in which both
                    conditions contain fewer than E reads are filtered out (Default E=10)
--out_dir_force     Specify this argument to force overwriting of files in
                    an existing output directory
```

2.5.2 Notes

DESeq2 and DRIMSeq are optimized for short read experiments and expect many reads for each expressed gene. Lower coverage (as expected when using long reads) will tend to result in false positives.

For instance, look at this counts table with two groups (s and v) of three samples each:

```
gene   s1    s2      s3      v1      v2      v3
   A    1     0       2       0       4       2
   B  100    99     101     100     104     102
```

Gene A has an average expression of 1 in group s, and 2 in group v but the total variation in read count is 0-4. The same variation is true for gene B, but it will not be considered differentially expressed.

Flair does not remove low count genes as long as they are expressed in all samples of at least one group so please be careful when interpreting results.

Results tables are filtered and reordered by p-value so that only p<0.05 differential genes/isoforms remain. Unfiltered tables can be found in `workdir`

Code requirements

This module requires python modules and R packages that are not necessary for other Flair modules (except diffSplice).

If you are not using the docker container or the conda installed version of Flair you may have to install these separately:

1. python modules: pandas, numpy, rpy2
2. DESeq2
3. ggplot2
4. qqman
5. DRIMSeq
6. stageR

2.6 flair diffSplice

IMPORTANT NOTE: diffExp and diffSplice are not currently part of the main flair code. Instead they are supplied as separate programs named flair_diffExp and flair_diffSplice. They take the same inputs as before.

```
usage: flair_diffSplice -i isoforms.bed -q counts_matrix.tsv [options]
```

This module calls alternative splicing (AS) events from isoforms. Currently supports the following AS events:

- intron retention (ir)
- alternative 3' splicing (alt3)
- alternative 5' splicing (alt5)
- cassette exons (es)

If there are 3 or more samples per condition, then you can run with `--test` and DRIMSeq will be used to calculate differential usage of the alternative splicing events between two conditions. See below for more DRIMSeq-specific arguments.

If conditions were sequenced without replicates, then the diffSplice output files can be input to the *diffsplice_fishers_exact* script for statistical testing instead.

Outputs

After the run, the output directory (`--out_dir`) contains the following tab separated files:

- `diffsplice.alt3.events.quant.tsv`
- `diffsplice.alt5.events.quant.tsv`
- `diffsplice.es.events.quant.tsv`
- `diffsplice.ir.events.quant.tsv`

If DRIMSeq was run (where `A` and `B` are conditionA and conditionB, see below):

- `drimseq_alt3_A_v_B.tsv`
- `drimseq_alt5_A_v_B.tsv`
- `drimseq_es_A_v_B.tsv`
- `drimseq_ir_A_v_B.tsv`
- `workdir` Temporary files including unfiltered output files.

2.6.1 Options

Required arguments

```
--isoforms          Isoforms in bed format from Flair collapse.
--counts_matrix     Tab-delimited isoform count matrix from Flair quantify.
--out_dir           Output directory for tables and plots.
```

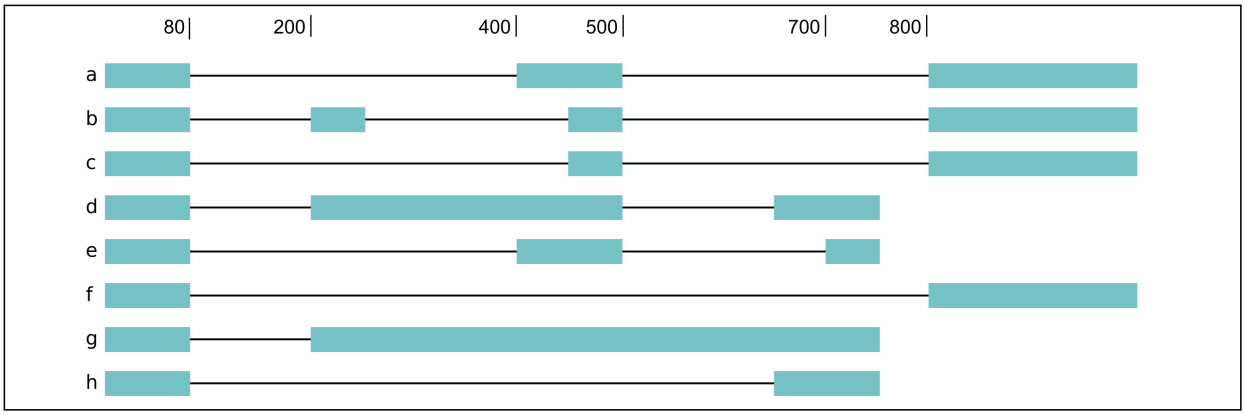
Optional arguments

```
--help              Show all options.
--threads           Number of processors to use (default 4).
--test              Run DRIMSeq statistical testing.
--drim1             The minimum number of samples that have coverage over an
                    AS event inclusion/exclusion for DRIMSeq testing; events
                    with too few samples are filtered out and not tested (6).
--drim2             The minimum number of samples expressing the inclusion of
                    an AS event; events with too few samples are filtered out
                    and not tested (3).
--drim3             The minimum number of reads covering an AS event
                    inclusion/exclusion for DRIMSeq testing, events with too
                    few samples are filtered out and not tested (15).
--drim4             The minimum number of reads covering an AS event inclusion
                    for DRIMSeq testing, events with too few samples are
                    filtered out and not tested (5).
--batch             If specified with --test, DRIMSeq will perform batch correction.
--conditionA        Specify one condition corresponding to samples in the
                    counts_matrix to be compared against condition2; by default,
                    the first two unique conditions are used. This implies --test.
--conditionB        Specify another condition corresponding to samples in the
                    counts_matrix to be compared against conditionA.
--out_dir_force     Specify this argument to force overwriting of files in an
                    existing output directory
```

2.6.2 Notes

Results tables are filtered and reordered by p-value so that only p<0.05 differential genes/isoforms remain. Unfiltered tables can be found in `workdir`

For a complex splicing example, please note the 2 alternative 3' SS, 3 intron retention, and 4 exon skipping events in the following set of isoforms that `flair diffSplice` would call and the isoforms that are considered to include or exclude the each event:




```
a3ss_feature_id     coordinate                  sample1 sample2 ... isoform_ids
inclusion_chr1:80   chr1:80-400_chr1:80-450    75.0    35.0    ... a,e
exclusion_chr1:80   chr1:80-400_chr1:80-450    3.0     13.0    ... c
```

(continues on next page)

(continued from previous page)

```
inclusion_chr1:500  chr1:500-650_chr1:500-700   4.0     18.0    ... d
exclusion_chr1:500  chr1:500-650_chr1:500-700   70.0    17.0    ... e
```

```
ir_feature_id           coordinate      sample1 sample2 ... isoform_ids
inclusion_chr1:500-650  chr1:500-650    46.0    13.0    ... g
exclusion_chr1:500-650  chr1:500-650    4.0     18.0    ... d
inclusion_chr1:500-700  chr1:500-700    46.0    13.0    ... g
exclusion_chr1:500-700  chr1:500-700    70.0    17.0    ... e
inclusion_chr1:250-450  chr1:250-450    50.0    31.0    ... d,g
exclusion_chr1:250-450  chr1:250-450    80.0    17.0    ... b
```

```
es_feature_id           coordinate      sample1 sample2 ... isoform_ids
inclusion_chr1:450-500  chr1:450-500    83.0    30.0    ... b,c
exclusion_chr1:450-500  chr1:450-500    56.0    15.0    ... f
inclusion_chr1:200-250  chr1:200-250    80.0    17.0    ... b
exclusion_chr1:200-250  chr1:200-250    3.0     13.0    ... c
inclusion_chr1:200-500  chr1:200-500    4.0     18.0    ... d
exclusion_chr1:200-500  chr1:200-500    22.0    15.0    ... h
inclusion_chr1:400-500  chr1:400-500    75.0    35.0    ... e,a
exclusion_chr1:400-500  chr1:400-500    56.0    15.0    ... f
```

CHAPTER

THREE

ADDITIONAL PROGRAMS

When you `conda install` flair, the following helper programs will be in your $PATH:

3.1 diff_iso_usage

```
usage: diff_iso_usage counts_matrix colname1 colname2 diff_isos.txt
```

Requires four positional arguments to identify and calculate significance of alternative isoform usage between two samples using Fisher's exact tests: (1) counts_matrix.tsv from flair-quantify, (2) the name of the column of the first sample, (3) the name of the column of the second sample, (4) `txt` output filename containing the p-value associated with differential isoform usage for each isoform. The more differentially used the isoforms are between the first and second condition, the lower the p-value.

Output file format columns are as follows:

- gene name
- isoform name
- p-value
- sample1 isoform count
- sample2 isoform count
- sample1 alternative isoforms for gene count
- sample2 alternative isoforms for gene count

3.2 diffsplice_fishers_exact

```
usage: diffsplice_fishers_exact events.quant.tsv colname1 colname2 out.fishers.tsv
```

Identifies and calculates the significance of alternative splicing events between two samples without replicates using Fisher's exact tests. Requires four positional arguments: (1) flair-diffSplice `tsv` of alternative splicing calls for a splicing event type, (2) the name of the column of the first sample, (3) the name of the column of the second sample, and (4) `tsv` output filename containing the p-values from Fisher's exact tests of each event.

Output

The output file contains the original columns with an additional column containing the p-values appended.

3.3 fasta_seq_lengths

```
usage: fasta_seq_lengths fasta outfilename [outfilename2]
```

3.4 junctions_from_sam

Usage: junctions_from_sam [options]

```
Options:
  -h, --help            show this help message and exit
  -s SAM_FILE           SAM/BAM file of read alignments to junctions and
                        the genome. More than one file can be listed,
                        but comma-delimited, e.g file_1.bam,file_2.bam
  --unique              Only keeps uniquely aligned reads. Looks at NH
                        tag to be 1 for this information.
  -n NAME               Name prefixed used for output BED file.
                        Default=junctions_from_sam
  -l READ_LENGTH        Expected read length if all reads should be of
                        the same length
  -c CONFIDENCE_SCORE   The mininmum entropy score a junction
                        has to have in order to be considered
                        confident. The entropy score =
                        -Shannon Entropy. Default=1.0
  -j FORCED_JUNCTIONS   File containing intron coordinates
                        that correspond to junctions that will be
                        kept regardless of the confidence score.
  -v                    Will run the program with junction strand ambiguity
                        messages
```

3.5 mark_intron_retention

```
usage: mark_intron_retention in.psl|in.bed out_isoforms.psl out_introns.txt
```

Assumes the psl has the correct strand information

Requires three positional arguments to identify intron retentions in isoforms:

- `psl` of isoforms
- `psl` output filename
- `txt` output filename for coordinates of introns found.

Outputs

- an extended `psl` with an additional column containing either values 0 or 1 classifying the isoform as either spliced or intron-retaining, respectively
- `txt` file of intron retentions with format `isoform name chromosome intron 5' coordinate intron 3' coordinate`.

Note: A psl or bed file with more additional columns will not be displayed in the UCSC genome browser, but can be displayed in IGV.

3.6 mark_productivity

```
usage: mark_productivity reads.psl annotation.gtf genome.fa > reads.productivity.psl
```

3.7 normalize_counts_matrix

```
usage: normalize_counts_matrix matrix outmatrix [cpm/uq/median] [gtf]
```

Gtf if normalization by protein coding gene counts only

3.8 plot_isoform_usage

```
plot_isoform_usage <isoforms.psl>|<isoforms.bed> counts_matrix.tsv gene_name
```

Visualization script for FLAIR isoform structures and the percent usage of each isoform in each sample for a given gene. If you supply the isoforms.bed file from running `predictProductivity`, then isoforms will be filled according to the predicted productivity (solid for `PRO`, hatched for `PTC`, faded for `NGO` or `NST`). The gene name supplied should correspond to a gene name in your isoform file and counts file.

The script will produce two images, one of the isoform models and another of the usage proportions.

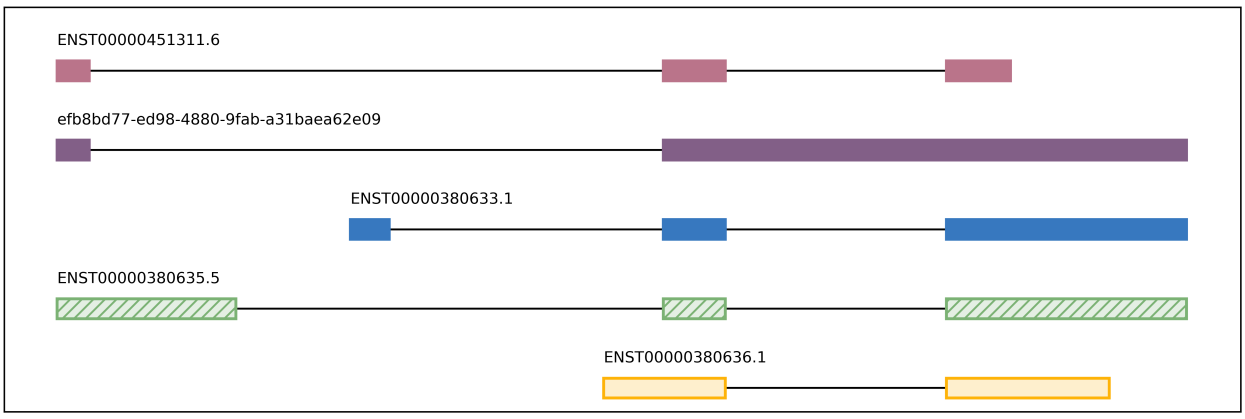
The most highly expressed isoforms across all the samples will be plotted.

The minor isoforms are aggregated into a gray bar. You can toggle min_reads or color_palette to plot more isoforms. Run with –help for options

Outputs

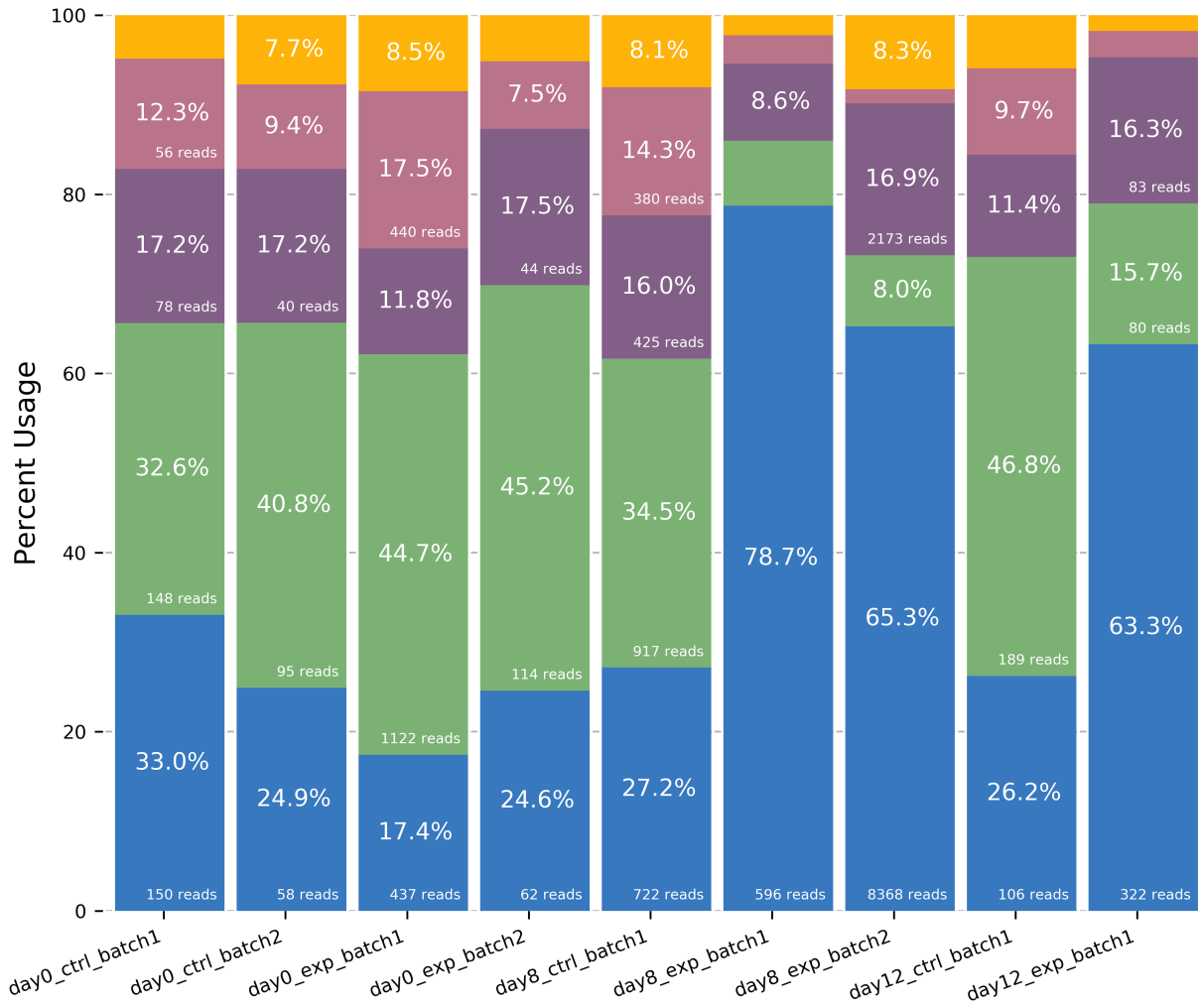
- gene_name_isoforms.png of isoform structures
- gene_name_usage.png of isoform usage by sample

For example:




```
positional arguments:
  isoforms              isoforms in psl/bed format
  counts_matrix         genomic sequence
```

(continues on next page)


Percent Usage
100
80
60
40
20
0
12.3%
56 reads
17.2%
78 reads
32.6%
148 reads
33.0%
150 reads
7.7%
9.4%
17.2%
40 reads
40.8%
95 reads
24.9%
58 reads
8.5%
17.5%
440 reads
11.8%
44.7%
1122 reads
17.4%
437 reads
7.5%
17.5%
44 reads
45.2%
114 reads
24.6%
62 reads
8.1%
14.3%
380 reads
16.0%
425 reads
34.5%
917 reads
27.2%
722 reads
8.6%
78.7%
596 reads
8.3%
16.9%
2173 reads
8.0%
65.3%
8368 reads
9.7%
11.4%
46.8%
189 reads
26.2%
106 reads
16.3%
83 reads
15.7%
80 reads
63.3%
322 reads
day0_ctrl_batch1
day0_ctrl_batch2
day0_exp_batch1
day0_exp_batch2
day8_ctrl_batch1
day8_exp_batch1
day8_exp_batch2
day12_ctrl_batch1
day12_exp_batch1

(continued from previous page)

```
  gene_name             Name of gene, must correspond with the gene names in
                        the isoform and counts matrix files

options:
  -h, --help            show this help message and exit
  -o O                  prefix used for output files (default=gene_name)
  --min_reads MIN_READS
                        minimum number of total supporting reads for an
                        isoform to be visualized (default=6)
  -v VCF, --vcf VCF     VCF containing the isoform names that include each
                        variant in the last sample column
  --palette PALETTE     provide a palette file if you would like to visualize
                        more than 7 isoforms at once or change the palette
                        used. each line contains a hex color for each isoform
```

3.9 predictProductivity

```
usage: predictProductivity -i isoforms.bed -f genome.fa -g annotations.gtf
```

Annotated start codons from the annotation are used to identify the longest ORF for each isoform for predicting isoform productivity. Requires three arguments to classify isoforms according to productivity: (1) isoforms in `psl` or `bed` format, (2) `gtf` genome annotation, (3) `fasta` genome sequences. Bedtools must be in your `$PATH` for predictProductivity to run properly.

Output

Outputs a bed file with either the values `PRO` (productive), `PTC` (premature termination codon, i.e. unproductive), `NGO` (no start codon), or `NST` (has start codon but no stop codon) appended to the end of the isoform name. When isoforms are visualized in the UCSC genome browser or IGV, the isoforms will be colored accordingly and have thicker exons to denote the coding region.

```
options:
  -h, --help            show this help message and exit
  -i INPUT_ISOFORMS, --input_isoforms INPUT_ISOFORMS
                        Input collapsed isoforms in psl or bed12 format.
  -g GTF, --gtf GTF     Gencode annotation file.
  -f GENOME_FASTA, --genome_fasta GENOME_FASTA
                        Fasta file containing transcript sequences.
  --quiet               Do not display progress
  --append_column       Append prediction as an additional column in file
  --firstTIS            Defined ORFs by the first annotated TIS.
  --longestORF          Defined ORFs by the longest open reading frame.
```

CHAPTER

FOUR

FILE CONVERSION SCRIPTS

4.1 bam2Bed12

```
usage: bam2Bed12 -i sorted.aligned.bam
options:
  -h, --help            show this help message and exit
  -i INPUT_BAM, --input_bam Input bam file.
  --keep_supplementary  Keep supplementary alignments
```

A tool to convert minimap2 BAM to Bed12.

4.2 bed_to_psl

```
usage: bed_to_psl chromsizes bedfile pslfile
```

chromsizes is a tab separated file of chromosome sizes, needed to make the `psl` file genome browser compatible. Here is one for GRCh38/hg38.

4.3 psl_to_bed

```
usage: psl_to_bed in.psl out.bed
```

4.4 sam_to_map

```
usage: sam_to_map sam outfile
```

CHAPTER

FIVE

FLAIR2 CAPABILITIES

FLAIR2 has an updated isoform detection algorithm and an added feature of variant-aware isoform detection.

5.1 Performance increases

FLAIR2 run with the `--annotation_reliant` argument invokes an alignment of the reads to an annotated transcriptome first, followed by novel isoform detection. This can be run with or without `--check_splice`, which enforces higher quality matching specifically around each splice site for read-to-isoform assignment steps.

```
flair collapse --check_splice --annotation_reliant generate -f annotation.gtf -g genome.
↪fa -r reads.fa -q corrected.bed [options]
```

If you are running collapse with the same transcript reference multiple times, you can specify the previously generated transcript sequence file to the `--annotation_reliant` argument instead.

5.2 Variant integration

FLAIR has two modalities for phasing variants to discover variant-aware transcript models. The first uses phasing information from longshot, which is comprised of a phase set determined for each read and a set of variants corresponding to each phase set. For the second modality, FLAIR can approach phasing variants that is agnostic to ploidy, which may be worthy of exploration if working with RNA edits and potential cancer-related aneuploidies: 1) given variant calls, FLAIR tabulates the most frequent combinations of variants present in each isoform from the supporting read sequences; 2) from the isoform-defining collapse step, FLAIR generates a set of reads assigned to each isoform; so 3) isoforms that have sufficient read support for a collection of mismatches are determined. This latter method of phasing focuses solely on frequency of groups of mismatches that co-occur within reads and does not use ploidy information to refine haplotypes, allowing for the generation of multiple haplotypes within a gene and transcript model.

5.2.1 Longshot

Longshot provides phased read outputs, which can be supplied to flair-collapse via `--longshot_vcf` and `--longshot_bam`. The outputs of collapse are the following: 1) isoform models as a bed file, 2) the subset of variants from the longshot vcf that were used, and 3) isoform sequences with variants as a fasta file. The isoform models and variants can be viewed by aligning the isoform sequences and using IGV or other visualization tools. .. code:: sh

> longshot –bam flair.aligned.bam –ref genome.fa –out longshot.vcf –out_bam longshot.bam –min_allele_qual 3 -F samtools index longshot.bam
>
> flair collapse -r reads.fa -q corrected.bed -g genome.fa –longshot_vcf longshot.vcf –longshot_bam flair.longshot.bam [options]

minimap2 -ax splice –secondary=no genome.fa flair.collapse.isoforms.fa > flair.collapse.isoforms.fa.sam samtools sort flair.collapse.isoforms.fa.sam -o flair.collapse.isoforms.fa.bam samtools index flair.collapse.isoforms.fa.bam

5.2.2 Any vcf

FLAIR2 can also take a vcf agnostic to the variant caller and spike variants in given any isoform model file. If enough supporting reads for an individual isoform contain the same pattern of variants, then FLAIR will create an additional isoform with _PSX appended to the name. Flair-collapse needs to be run with `--generate_map`. .. code:: sh

flair collapse -r reads.fa -q corrected.bed -g genome.fa –generate_map [options]

assign_variants_to_transcripts –bam flair.aligned.bam -i flair.collapse.isoforms.bed -v variants.vcf –map flair.collapse.isoform.read.map.txt –bed_out out.bed –map_out out.map > out.vcf

psl_to_sequence out.bed genome.fa out.fa -v out.vcf

minimap2 -ax splice –secondary=no genome.fa out.fa > out.fa.sam samtools sort out.fa.sam -o out.fa.bam samtools index out.fa.bam

CHAPTER

SIX

OTHER WAYS TO RUN FLAIR MODULES

For convenience, multiple FLAIR modules can be run in the same command. In place of a single module name, multiple module numbers can be specified (module numbers: align=1, correct=2, collapse=3, collapse-range=3.5, quantify=4, diffExp=5, diffSplice=6). All arguments for the modules that will be run must be provided. For example, to run the align, correct, and collapse modules, the command might look like:

```
flair 123 -r reads.fa -g genome.fa -f annotation.gtf -o flair.output --temp_dir temp_
↪flair [optional arguments]
```

A beta version of the collapse module, called collapse-range, has been developed. The corrected reads are divided into many independent regions, which are then subject to isoform calling separately and parallelized over the number of threads specified. This dramatically decreases the memory footprint of intermediate files and increases the speed in which the module runs without altering the final isoforms. This version can be invoked by specifying collapse-range as the module (or 3.5 if using numbers). An additional program, bedPartition, needs to be in your $PATH.

```
flair collapse-range -r reads.bam -q query.bed -g genome.fa -f annotation.gtf -o flair.
↪output --temp_dir temp_flair [optional arguments]
```

If you would prefer not to use python's multiprocessing module, a bash script has also been provided (`run_flair_collapse_ranges.sh`) that runs collapse-range for the user that parallelizes using GNU parallel, which you can alter as they see fit for their system.

CHAPTER

SEVEN

OTHER ENVIRONMENTS

The easiest way to install Flair and all of its dependencies is via conda:

```
conda create -n flair -c conda-forge -c bioconda flair
conda activate flair
flair [align/correct/...]
```

It is also possible to get the full Flair setup as a docker image:

```
docker pull brookslab/flair:latest
docker run -w /usr/data -v [your_path_to_data]:/usr/data brookslab/flair:latest flair␣
↪[align/correct/...]
```

7.1 Other methods (not recommended)

Flair consists of six modules. The first three are `align`, `correct`, and `collapse`. They are the most used, so we will refer to them here as basic Flair.

The other three modules are `quantify`, `flair_diffExp`, and `flair_diffSplice`. Together with basic Flair these are called full Flair. These three additional modules have more dependencies than basic Flair so if you don't need them, you will not need the modules listed under 5.

There are other ways to install Flair:

- `pip install flair-brookslab` will install basic Flair and all necessary python modules (see below)
- Download the latest release
- Use git to check out the current flair repository

7.1.1 Requirements

1. Bedtools
2. samtools
3. minimap2

If you do not use `pip install` or `conda env create`, you will also need:

4. python v3.6+ and python modules:
    - numpy=1.9.*
    - tqdm

- ncls
- pybedtools
- mappy
- pysam=v0.8.4+

5. full Flair additional python modules:

- Cython
- pandas
- rpy2=2.9.*
- R
- r-ggplot2=2.2.1
- r-qqman
- bioconductor-deseq2
- bioconductor-drimseq
- bioconductor-stager
- matplotlib
- seaborn

7.1.2 Pip install

`pip install flair-brookslab` will put the latest Flair release in your `$PATH`, as well as the helper scripts discussed in this manual. It also installs all python modules needed to run basic Flair. If you want to use full Flair, install the packages listed under point 5 in the list above.

7.1.3 Download the latest release

Navigate to the Flair release page and select one of the source code files under Assets. Exctract the file and navigate to the resulting *flair* directory. Add Flair and the helper scripts to your `$PATH` for instance (in Linux) with `export PATH=$(pwd)/bin:$PATH`.

Make sure to (*pip*) install the python modules listed above. If you have conda, you can create a basic Flair environment using

```
conda env create -f misc/flair_basic_conda_env.yaml
```

7.1.4 Download the latest code

Check out the current Flair repository from github. Please be aware that while this may have the latest bug fixes, it's quite possible that new bugs were introduced. This method is only useful if you have reported a problem and a Flair developer lets you know it has been fixed.

Once you have cloned the repository, navigate to the */flair* directory. Follow the steps as described under Download the latest release.

CHAPTER

EIGHT

TESTING FLAIR

Prerequisites:

- flair and flair scripts are in your `$PATH` (see below)
- You have a copy of the `flair/test` directory (e.g. `git clone git@github.com:BrooksLabUCSC/flair.git`)
- GNU make

Flair is in your `$PATH` if you used `conda install -c conda-forge -c bioconda flair`.

If you downloaded the latest release from github or cloned the flair repository:

```
export PATH=/path/to/flair/src/flair:/path/to/flair/bin:$PATH
```

Move to the `flair/test` directory, then run `make test`.

If this is the first time, make will download some sequences from the UCSC Genome Browser download page and store them as test_input/genome.fa.

`make test` tests all six flair modules and two helper programs. You can also test them individually using:

- `make test-align`
- `make test-correct`
- `make test-collapse`
- `make test-quantify`
- `make test-diffexp`
- `make test-diffsplice`
- `make test-predict-productivity`
- `make test-diff-iso-usage`

`make` outputs a lot of information. If a test fails, it will stop with an error and not run any additional tests. Errors look like this:

```
make:  *** [makefile:71:  test-predict-productivity] Error 2
```

You can usually find more information in the lines preceding the error. If you cannot figure out the problem, please create a ticket.

CHAPTER

NINE

EXAMPLE FILES

We have provided the following example files here:

- `star.firstpass.gm12878.junctions.3.tab`, a file of splice junctions observed from short read sequencing of GM18278 that can be used in the correction step with `-j`. Junctions with fewer than 3 uniquely mapping reads have been filtered out.
- `promoter.gencode.v27.20.bed`, promoter regions determined from ENCODE promoter chromatin states for GM12878 and 20 bp around annotated TSS in GENCODE v27. Can be supplied to flair-collapse with `-p` to build the initial firstpass set with only reads with start positions falling within these regions

Other downloads:

- Native RNA Pass reads Running these 10 million nanopore reads from `fastq` through flair align, correct, and collapse modules to assembled isoforms with 8 threads requires ~3.5 hours (includes ~2.5 hours of minimap2 alignment)
- NanoSim_Wrapper.py, a wrapper script written for simulating nanopore transcriptome data using Nanosim

CHAPTER

TEN

FAQ

10.1 1. Flair collapse uses too much memory, what can I do?

Flair's memory requirements increase with larger input files. If your bed file is over 1 Gigabyte, consider splitting it by chromosome and then running separately on each file.

CHAPTER

ELEVEN

CITE FLAIR

If you use or discuss FLAIR, please cite the following paper:

Tang, A.D., Soulette, C.M., van Baren, M.J. et al. Full-length transcript characterization of SF3B1 mutation in chronic lymphocytic leukemia reveals downregulation of retained introns. Nat Commun 11, 1438 (2020).

CHAPTER

TWELVE

INDICES AND TABLES

- genindex
- modindex
- search