rmats-turbo / README.md

EricKutschera Initial commit rMATS turbo v4.1.0 bc29dc5 · 4 years ago

349 lines (279 loc) · 17.8 KB

rMATS turbo v4.1.0

About

rMATS turbo is the C/Cython version of rMATS (refer to http://rnaseq-mats.sourceforge.net). The major difference between rMATS turbo and rMATS is speed and space usage. rMATS turbo is 100 times faster and the output file is 1000 times smaller than rMATS. These advantages make analysis and storage of a large scale dataset easy and convenient.

	Counting part	Statistical part
Speed (C/Cython version vs Python version)	20~100 times faster (one thread)	300 times faster (6 threads)
Storage usage (C/Cython version vs Python version)	1000 times smaller	-

Table of contents

Dependencies

Tested with

- Python (either 2.7 or 3.6)
    - Cython (0.29.14)
    - numpy (1.16.5)
- BLAS, LAPACK
- GNU Scientific Library (GSL 2.5)
- GCC (5.4.0)
- gfortran (Fortran 77)
- CMake (3.15.4)
- PAIRADISE (optional)
- Samtools (optional)
- STAR (optional)

Build

If the required dependencies are already installed, then rMATS can be built with:

```
./build_rmats
```

And then run with:

```
python rmats.py {arguments}
```

The build_rmats script usage is:

```
./build_rmats [--conda] [--no-paired-model]

--conda: create a conda environment for Python and R dependencies
--no-paired-model: do not install dependencies for the paired model
```

With `--conda` build_rmats installs a conda environment that satisfies the required Python dependencies and also the R dependencies needed to use the paired model (PAIRADISE). The Python dependencies are listed in python_requirements.txt and the R dependencies are handled using install_r_deps.R after cloning the PAIRADISE git repo.

run_rmats is a wrapper to call rmats.py with the conda environment used by build_rmats. It also sources setup_environment.sh which can be modified to handle other setup that might be needed before running rmats (such as Environment Modules).

If rMATS was built with `./build_rmats --conda` then it should be run with:

```
./run_rmats {arguments}
```

Test

test_rmats creates a conda environment and uses run_rmats to run the automated tests in tests/

Usage

Examples

Starting with FASTQ files

Suppose there are 2 sample groups with 2 sets of paired read (R1, R2) FASTQ files per group.

- group 1 FASTQs
  - `/path/to/1_1.R1.fastq`
  - `/path/to/1_1.R2.fastq`
  - `/path/to/1_2.R1.fastq`
  - `/path/to/1_2.R2.fastq`
- group 2 FASTQs
  - `/path/to/2_1.R1.fastq`

  - `/path/to/2_1.R2.fastq`
  - `/path/to/2_2.R1.fastq`
  - `/path/to/2_2.R2.fastq`

Create txt files that will be used to pass this grouping of inputs to rMATS. The expected format is `:` to separate paired reads and `,` to separate replicates.

- `/path/to/s1.txt`

```
/path/to/1_1.R1.fastq:/path/to/1_1.R2.fastq,/path/to/1_2.R1.fastq:/path/to/1_2
```

- `/path/to/s2.txt`

```
/path/to/2_1.R1.fastq:/path/to/2_1.R2.fastq,/path/to/2_2.R1.fastq:/path/to/2_2
```

Details about the remaining arguments are discussed in All arguments

run rMATS on this input with:

```
python rmats.py --s1 /path/to/s1.txt --s2 /path/to/s2.txt --gtf
/path/to/the.gtf --bi /path/to/STAR_binary_index -t paired --readLength 50
--nthread 4 --od /path/to/output --tmp /path/to/tmp_output
```

rMATS will first process the FASTQ input into BAM files stored in the `--tmp` directory. Then the splicing analysis will be performed.

Starting with BAM files

Reads can be mapped independently of rMATS with any aligner and then the resulting BAM files can be used as input to rMATS.

Suppose there are 2 sample groups with 2 BAM files per group.

- group 1 BAMs
  - `/path/to/1_1.bam`
  - `/path/to/1_2.bam`
- group 2 BAMs
  - `/path/to/2_1.bam`
  - `/path/to/2_2.bam`

Create txt files that will be used to pass this grouping of inputs to rMATS. The expected format is `,` to separate replicates.

- `/path/to/b1.txt`

```
/path/to/1_1.bam,/path/to/1_2.bam
```

- `/path/to/b2.txt`

```
/path/to/2_1.bam,/path/to/2_2.bam
```

Details about the remaining arguments are discussed in All arguments

run rMATS on this input with:

```
python rmats.py --b1 /path/to/b1.txt --b2 /path/to/b2.txt --gtf
/path/to/the.gtf -t paired --readLength 50 --nthread 4 --od /path/to/output
--tmp /path/to/tmp_output
```

Running prep and post separately

rMATS analysis has two steps, prep and post. In the prep step, the input files are processed and a summary is saved to a `.rmats` file in the `--tmp` directory. That `.rmats` file tracks info from each BAM separately according to the full path of the BAM specified in the input `.txt` file. In the post step, one or more `.rmats` files are read and the final output files are created.

The `--task` argument allows the prep step of rMATS to be run independently for different subsets of input BAM files. Then the post step can be run on the independently generated `.rmats` files. This allows the computation to be run at different times and/or on different machines.

Suppose we have 8 BAMs and two machines that each have 4 CPU threads. Each machine can run the prep step on 4 BAMs concurrently. Then the post step can be run on one of the machines.

Split the BAMs into two groups. The assignment of BAMs to prep steps does not restrict the choice of `--b1` and `--b2` for a later post step.

- `/path/to/prep1.txt`

```
/path/to/1.bam,/path/to/2.bam,/path/to/3.bam,/path/to/4.bam
```

- `/path/to/prep2.txt`

```
/path/to/5.bam,/path/to/6.bam,/path/to/7.bam,/path/to/8.bam
```

On machine 1 run the prep step with prep1.txt:

```
python rmats.py --b1 /path/to/prep1.txt --gtf /path/to/the.gtf -t paired --readLength 50 --nthread 4 --od /path/to/output --tmp /path/to/tmp_output_prep_1 --task prep
```

On machine 2 run the prep step with prep2.txt:

```
python rmats.py --b1 /path/to/prep2.txt --gtf /path/to/the.gtf -t paired --readLength 50 --nthread 4 --od /path/to/output --tmp /path/to/tmp_output_prep_2 --task prep
```

Split the BAMs into two groups. This split is for statistically comparing the two groups and does not need to reflect the split used in the prep steps

- `/path/to/post1.txt`

```
/path/to/1.bam,/path/to/3.bam,/path/to/8.bam
```

- `/path/to/post2.txt`

```
/path/to/2.bam,/path/to/4.bam,/path/to/5.bam,/path/to/6.bam,/path/to/7.bam
```

Copy the `.rmats` files from the separate prep steps to a directory so that the post step can access all the prep data. The filenames have the format `{datetime}.rmats` and the filenames may conflict for prep steps run concurrently. The script cp_with_prefix.py is provided to disambiguate the `.rmats` filenames when copying to a shared directory:

```
python cp_with_prefix.py prep_1_ /path/to/tmp_output_post/ /path/to/tmp_output_prep_1/*.rmats
python cp_with_prefix.py prep_2_ /path/to/tmp_output_post/ /path/to/tmp_output_prep_2/*.rmats
```

On machine 1 run the post step:

```
python rmats.py --b1 /path/to/post1.txt --b2 /path/to/post2.txt --gtf
/path/to/the.gtf -t paired --readLength 50 --nthread 4 --od /path/to/output
--tmp /path/to/tmp_output_post --task post
```

Using the paired stats model

The default statistical model considers the samples to be unpaired. The `--paired-stats` flag can be used if each entry in `--b1` is matched with its pair in `--b2`. As an example, if there are three replicates where each replicate has paired "a" and "b" data, then b1.txt and b2.txt should look like:

- `/path/to/b1.txt`

```
/path/to/pair_1_a.bam,/path/to/pair_2_a.bam,/path/to/pair_3_a.bam
```

- `/path/to/b2.txt`

```
/path/to/pair_1_b.bam,/path/to/pair_2_b.bam,/path/to/pair_3_b.bam
```

The `--paired-stats` flag can then be given so that the paired statistical model is used instead of the default unpaired model.

Tips

- The statistical comparison between the two input sample groups can be skipped with `--statoff`. It is also possible to use a single sample group (only `--b1` or `--s1`) when using `--statoff`.
- A cluster environment can be utilized to run many prep steps concurrently and when the prep steps are finished a single post step can be run.
- When splitting the computation using `--task {prep, post}`, rMATS will consider all `.rmats` files in the `--tmp` directory when running the post step. The `.rmats` files from multiple prep steps can be copied to a shared location for running the post step. Replacing `--task post` in the command line that is going to be used for the post step with `--task inte` will perform an integrity check to verify that the BAM filenames in `--b1` and `--b2` match 1-to-1 with the BAM filenames recorded in the `.rmats` files in `--tmp`.
- The `.rmats` filenames from concurrently run prep steps may conflict. The script cp_with_prefix.py is provided to disambiguate the `.rmats` filenames when copying to

a shared directory.

- The full path of the BAM files given in `--b1` and `--b2` for the prep step must match the full paths given in the post step. Otherwise the lookup into the `.rmats` file(s) will fail. As an example, if the full `/path/to/1.bam` is used in the prep step, a relative path of just `1.bam` cannot be used in the post step.
- If analyzing a small data set, `--task both` can be used to perform the prep and post steps in a single run.
- `--novelSS` is an experimental feature that allows splicing events to be detected that involve an unannotated splice site.

All Arguments

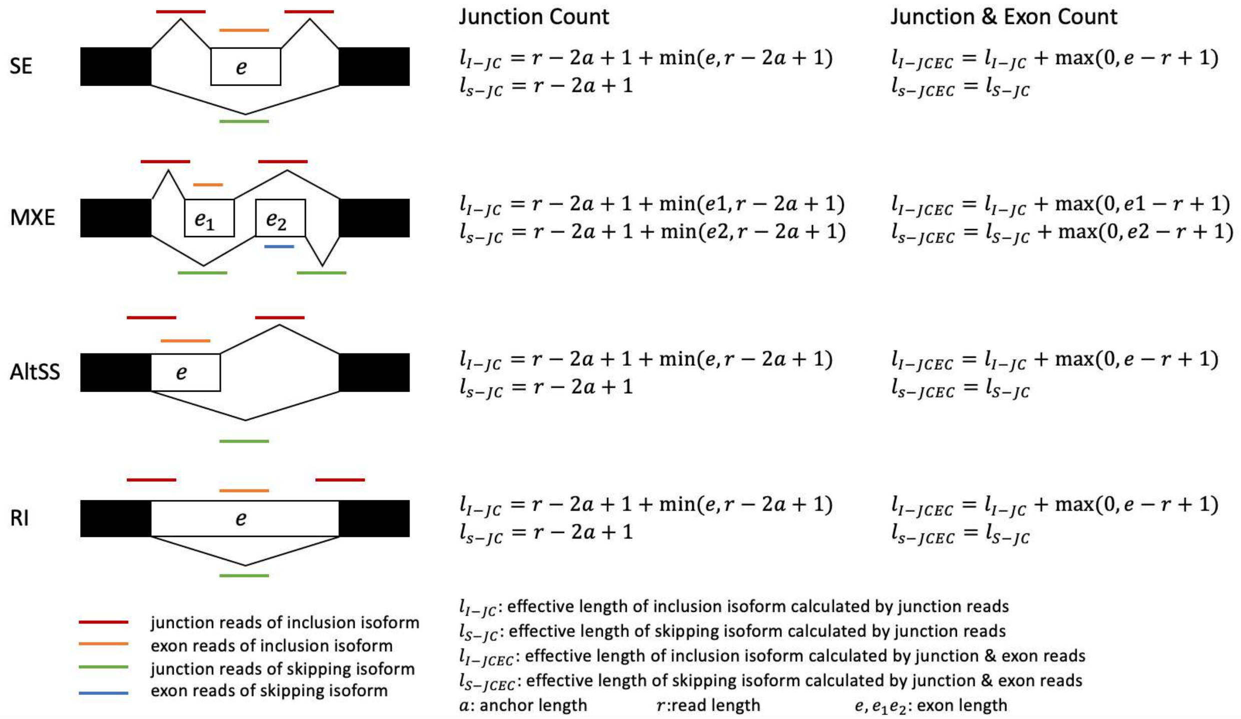
```
python rmats.py -h

usage: rmats.py [options]

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --gtf GTF             An annotation of genes and transcripts in GTF
format
  --b1 B1               A text file containing a comma separated list of
the
                        BAM files for sample_1. (Only if using BAM)
  --b2 B2               A text file containing a comma separated list of
the
                        BAM files for sample_2. (Only if using BAM)
  --s1 S1               A text file containing a comma separated list of
the
                        FASTQ files for sample_1. If using paired reads the
                        format is ":" to separate pairs and "," to separate
                        replicates. (Only if using fastq)
  --s2 S2               A text file containing a comma separated list of
the
                        FASTQ files for sample_2. If using paired reads the
                        format is ":" to separate pairs and "," to separate
                        replicates. (Only if using fastq)
  --od OD               The directory for final output
  --tmp TMP             The directory for intermediate output such as
".rmats"
                        files from the prep step
  -t {paired,single}    Type of read used in the analysis: either "paired"
for
                        paired-end data or "single" for single-end data.
                        Default: paired
                        The length of each read
  --variable-read-length
                        Allow reads with lengths that differ from --
readLength
                        to be processed. --readLength will still be used to
                        determine IncFormLen and SkipFormLen
  --anchorLength ANCHORLENGTH
                        The anchor length. Default is 1
  --tophatAnchor TOPHATANCHOR
                        The "anchor length" or "overhang length" used in
the
                        aligner. At least "anchor length" NT must be mapped
to
                        each end of a given junction. The default is 6.
(Only
                        if using fastq)
  --bi BINDEX           The directory name of the STAR binary indices (name
of
                        the directory that contains the SA file). (Only if
                        using fastq)
  --nthread NTHREAD     The number of threads. The optimal number of
threads
                        should be equal to the number of CPU cores.
Default: 1
  --tstat TSTAT         The number of threads for the statistical model.
                        Default: 1
  --cstat CSTAT         The cutoff splicing difference. The cutoff used in
the
                        null hypothesis test for differential splicing. The
                        default is 0.0001 for 0.01% difference. Valid: 0 <=
                        cutoff < 1. Does not apply to the paired stats
model
  --task {prep,post,both,inte}
                        Specify which step(s) of rMATS to run. Default:
both.
                        prep: preprocess BAMs and generate a .rmats file.
                        post: load .rmats file(s) into memory, detect and
                        count alternative splicing events, and calculate P
                        value (if not --statoff). both: prep + post. inte
                        (integrity): check that the BAM filenames recorded
by
                        the prep task(s) match the BAM filenames for the
                        current command line
  --statoff             Skip the statistical analysis
  --paired-stats        Use the paired stats model
  --novelSS             Enable detection of novel splice sites (unannotated
                        splice sites). Default is no detection of novel
splice
                        sites
  --mil MIL             Minimum Intron Length. Only impacts --novelSS
                        behavior. Default: 50
  --mel MEL             Maximum Exon Length. Only impacts --novelSS
behavior.
                        Default: 500
```

Output

Each alternative splicing event type has a corresponding set of output files. In the filename templates below `[AS_Event]` is replaced by one of [SE (skipped exon), MXE (mutually exclusive exons), A3SS (alternative 3' splice site), A5SS (alternative 5' splice site), RI (retained intron)] for the event specific filename.


Junction Count
Junction & Exon Count
SE
e
$l_{I-JC} = r - 2a + 1 + \min(e, r - 2a + 1)$
$l_{s-JC} = r - 2a + 1$
$l_{I-JCEC} = l_{I-JC} + \max(0, e - r + 1)$
$l_{s-JCEC} = l_{S-JC}$
MXE
e_1
e_2
$l_{I-JC} = r - 2a + 1 + \min(e1, r - 2a + 1)$
$l_{s-JC} = r - 2a + 1 + \min(e2, r - 2a + 1)$
$l_{I-JCEC} = l_{I-JC} + \max(0, e1 - r + 1)$
$l_{s-JCEC} = l_{s-JC} + \max(0, e2 - r + 1)$
AltSS
e
$l_{I-JC} = r - 2a + 1 + \min(e, r - 2a + 1)$
$l_{s-JC} = r - 2a + 1$
$l_{I-JCEC} = l_{I-JC} + \max(0, e - r + 1)$
$l_{s-JCEC} = l_{S-JC}$
RI
e
$l_{I-JC} = r - 2a + 1 + \min(e, r - 2a + 1)$
$l_{s-JC} = r - 2a + 1$
$l_{I-JCEC} = l_{I-JC} + \max(0, e - r + 1)$
$l_{s-JCEC} = l_{S-JC}$
🟥 junction reads of inclusion isoform
🟧 exon reads of inclusion isoform
🟩 junction reads of skipping isoform
🟦 exon reads of skipping isoform
l_{I-JC}: effective length of inclusion isoform calculated by junction reads
l_{S-JC}: effective length of skipping isoform calculated by junction reads
l_{I-JCEC}: effective length of inclusion isoform calculated by junction & exon reads
l_{S-JCEC}: effective length of skipping isoform calculated by junction & exon reads
a: anchor length
r:read length
$e, e_1 e_2$: exon length

`--od` contains the final output files:

- `[AS_Event].MATS.JC.txt`: Final output including only reads that span junctions defined by rmats (Junction Counts)
- `[AS_Event].MATS.JCEC.txt`: Final output including both reads that span junctions defined by rmats (Junction Counts) and reads that do not cross an exon boundary (Exon Counts)
- `fromGTF.[AS_Event].txt`: All identified alternative splicing (AS) events derived from GTF and RNA

- `fromGTF.novelJunction.[AS_Event].txt`: Alternative splicing (AS) events which were identified only after considering the RNA (as opposed to analyzing the GTF in isolation). This does not include events with an unannotated splice site.
- `fromGTF.novelSpliceSite.[AS_Event].txt`: This file contains only those events which include an unannotated splice site. Only relevant if `--novelSS` is enabled.
- `JC.raw.input.[AS_Event].txt`: Event counts including only reads that span junctions defined by rmats (Junction Counts)
- `JCEC.raw.input.[AS_Event].txt`: Event counts including both reads that span junctions defined by rmats (Junction Counts) and reads that do not cross an exon boundary (Exon Counts)
- Shared columns:
  - `ID`: rMATS event id
  - `GeneID`: Gene id
  - `geneSymbol`: Gene name
  - `chr`: Chromosome
  - `strand`: Strand of the gene
  - `IJC_SAMPLE_1`: Inclusion counts for sample 1. Replicates are comma separated
  - `SJC_SAMPLE_1`: Skipping counts for sample 1. Replicates are comma separated
  - `IJC_SAMPLE_2`: Inclusion counts for sample 2. Replicates are comma separated
  - `SJC_SAMPLE_2`: Skipping counts for sample 2. Replicates are comma separated
  - `IncFormLen`: Length of inclusion form, used for normalization
  - `SkipFormLen`: Length of skipping form, used for normalization
  - `PValue`: Significance of splicing difference between the two sample groups. (Only available if the statistical model is on)
  - `FDR`: False Discovery Rate calculated from p-value. (Only available if statistical model is on)
  - `IncLevel1`: Inclusion level for sample 1. Replicates are comma separated. Calculated from normalized counts
  - `IncLevel2`: Inclusion level for sample 2. Replicates are comma separated. Calculated from normalized counts
  - `IncLevelDifference`: average(IncLevel1) - average(IncLevel2)
- Event specific columns (event coordinates):
  - SE: `exonStart_0base` `exonEnd` `upstreamES` `upstreamEE` `downstreamES` `downstreamEE`
    - The inclusion form includes the target exon (`exonStart_0base`, `exonEnd`)
  - MXE: `1stExonStart_0base` `1stExonEnd` `2ndExonStart_0base` `2ndExonEnd` `upstreamES` `upstreamEE` `downstreamES` `downstreamEE`

        - If the strand is `+` then the inclusion form includes the 1st exon (`1stExonStart_0base`, `1stExonEnd`) and skips the 2nd exon
        - If the strand is `-` then the inclusion form includes the 2nd exon (`2ndExonStart_0base`, `2ndExonEnd`) and skips the 1st exon
    - A3SS, A5SS: `longExonStart_0base` `longExonEnd` `shortES` `shortEE` `flankingES` `flankingEE`
        - The inclusion form includes the long exon (`longExonStart_0base`, `longExonEnd`) instead of the short exon (`shortES` `shortEE`)
    - RI: `riExonStart_0base` `riExonEnd` `upstreamES` `upstreamEE` `downstreamES` `downstreamEE`
        - The inclusion form includes (retains) the intron (`upstreamEE`, `downstreamES`)

`--tmp` contains the intermediate files generated by the prep step:

- `{datetime}.rmats`: Summary generated from processing the BAM(s)
- `bam{sample_num}_{replicate_num}/Aligned.sortedByCoord.out.bam`: result of mapping input FASTQ files