

# Package ‘pram’

September 23, 2023

**Title** Pooling RNA-seq datasets for assembling transcript models

**Version** 1.17.0

**Description** Publicly available RNA-seq data is routinely used for retrospective analysis to elucidate new biology. Novel transcript discovery enabled by large collections of RNA-seq datasets has emerged as one of such analysis. To increase the power of transcript discovery from large collections of RNA-seq datasets, we developed a new R package named Pooling RNA-seq and Assembling Models (PRAM), which builds transcript models in intergenic regions from pooled RNA-seq datasets. This package includes functions for defining intergenic regions, extracting and pooling related RNA-seq alignments, predicting, selected, and evaluating transcript models.

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/pliu55/pram>

**BugReports** <https://github.com/pliu55/pram/issues>

**Depends** R (>= 3.6)

**Imports** methods, BiocParallel, tools, utils, data.table (>= 1.11.8), GenomicAlignments (>= 1.16.0), rtracklayer (>= 1.40.6), BiocGenerics (>= 0.26.0), GenomeInfoDb (>= 1.16.0), GenomicRanges (>= 1.32.0), IRanges (>= 2.14.12), Rsamtools (>= 1.32.3), S4Vectors (>= 0.18.3)

**RoxygenNote** 6.1.0

**Suggests** testthat, BiocStyle, knitr, rmarkdown

**Collate** 'Param.R' 'Transcript.R' 'buildModel.R' 'defIlgRanges.R' 'evalModel.R' 'prepIlgBam.R' 'runPRAM.R' 'selModel.R' 'util.R'

**VignetteBuilder** knitr

**biocViews** Software, Technology, Sequencing, RNASeq, BiologicalQuestion, GenePrediction, GenomeAnnotation, ResearchField, Transcriptomics

**SystemsRequirements** buildModel() and runPRAM() functions require external software Cufflinks, StringTie, and/or TACO. For details, please see the 'Required external software' section in vignette's 'Building transcript models: buildModel()'.  
**git\_url** <https://git.bioconductor.org/packages/pram>  
**git\_branch** devel  
**git\_last\_commit** 9dc2845  
**git\_last\_commit\_date** 2023-04-25  
**Date/Publication** 2023-09-22  
**Author** Peng Liu [aut, cre],  
 Colin N. Dewey [aut],  
 Sündüz Keleş [aut]  
**Maintainer** Peng Liu <pliu55.wisc@gmail.com>

## R topics documented:

buildModel	2
defIgRanges	4
evalModel	5
prepIgBam	7
runPRAM	8
selModel	9

<b>Index</b>	<b>11</b>
--------------	-----------

---

buildModel	<i>Build transcript models from aligned RNA-seq data</i>
------------	--

---

### Description

Build transcript models from aligned RNA-seq data

### Usage

```
buildModel(in_bamv, out_gtf, method = "plcf", nthreads = 1,
  tmpdir = NULL, keep_tmpdir = FALSE, cufflinks = "",
  stringtie = "", taco = "", cufflinks_ref_fa = "")
```

### Arguments

in_bamv	A character vector of input BAM file(s). If mode 'cf' or 'st' is used, only one input RNA-seq BAM file is allowed. Currently, PRAM only supports strand-specific paired-end data with the first mate on the right-most of transcript coordinate, i.e., 'fr-firststrand' by Cufflinks's definition.
out_gtf	An output GTF file of predicted transcript models

method	<p>A character string defining PRAM's model building method. Current available methods are:</p> <ul style="list-style-type: none"> <li>• plcf: pooling + cufflinks</li> <li>• plst: pooling + stringtie</li> <li>• cfmg: cufflinks + cuffmerge</li> <li>• stmg: stringtie + merging</li> <li>• cftc: cufflinks + taco</li> <li>• cf: cufflinks</li> <li>• st: stringtie</li> </ul> <p>Default: 'plcf'</p>
nthreads	An integer defining the number of threads to-be-used. Default: 1
tmpdir	A character string defining the full name of a folder for saving temporary files. If not tmpdir is give, PRAM will use R's tempdir().
keep_tmpdir	Whether to keep temporary files afterwards. Default: False
cufflinks	<p>Cufflinks executable. Required by mode 'plcf', 'cfmg', and 'cf'. For mode 'cfmg', executable files of Cuffmerge, Cuffcompare, and gtf_to_sam from the Cufflinks suite are assumed to be under the same folder as Cufflinks. All the executables are available to download for Linux <a href="http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.2.1.Linux_x86_64.tar.gz">http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.2.1.Linux_x86_64.tar.gz</a> and MacOS <a href="http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.1.1.OSX_x86_64.tar.gz">http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.1.1.OSX_x86_64.tar.gz</a>. Souce code can be obtained from <a href="http://cole-trapnell-lab.github.io/cufflinks/">http://cole-trapnell-lab.github.io/cufflinks/</a>. Default: "</p>
stringtie	<p>StringTie executable file. Required by mode 'plst', 'stmg', and 'st'. Executable can be downloaded for Linux <a href="http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.Linux_x86_64.tar.gz">http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.Linux_x86_64.tar.gz</a> and MacOS <a href="http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.OSX_x86_64.tar.gz">http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.OSX_x86_64.tar.gz</a>. Souce code can be obtained from <a href="https://ccb.jhu.edu/software/stringtie/">https://ccb.jhu.edu/software/stringtie/</a>. Default: "</p>
taco	<p>TACO executable file. Required by mode 'cftc'. Executable can be downloaded for Linux <a href="https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.Linux_x86_64.tar.gz">https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.Linux_x86_64.tar.gz</a> and MacOS <a href="https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.OSX_x86_64.tar.gz">https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.OSX_x86_64.tar.gz</a>. Souce code can be obtained from <a href="https://tacorna.github.io">https://tacorna.github.io</a>. Default: "</p>
cufflinks_ref_fa	<p>Genome reference fasta file for Cufflinks. If supplied, will be used for cufflinks's '-frag-bias-correct' and cuffmerge's '-ref-sequence' options. Default: "</p>

**Value**

None

**Examples**

```
fbams = c( system.file('extdata/bam/CMPRep1.sortedByCoord.clean.bam',
                      package='pram'),
           system.file('extdata/bam/CMPRep2.sortedByCoord.clean.bam',
                      package='pram') )

foutgtf = tempfile(fileext='.gtf')

## assuming the stringtie binary is in folder /usr/local/stringtie-1.3.3/
## you can run buildModel() by the following example
##
# buildModel(fbams, foutgtf, method='plst',
#            stringtie='/usr/local/stringtie-1.3.3/stringtie')
```

---

<code>defIgRanges</code>	<i>Define intergenic genomic regions</i>
--------------------------	--

---

**Description**

Define intergenic genomic regions

**Usage**

```
defIgRanges(in_gtg, chromgrs, genome = NULL, fchromsize = NULL,
            radius = 10000, feat = "exon", chroms = NULL)
```

**Arguments**

<code>in_gtg</code>	An input GTF file for defining genomic coordinates of existing genes. Required to have 'gene_id' in the attribute column (column 9)
<code>chromgrs</code>	A GRanges object defining chromosome sizes.
<code>genome</code>	Version of the genome. Will be used when 'chromgrs' is missing. Currently supported ones are: <ul style="list-style-type: none"> <li>• hg19</li> <li>• hg38</li> <li>• mm9</li> <li>• mm10</li> </ul> <p>All the above genomes have sizes for all chromosomes including random and alt ones. Default: NULL</p>
<code>fchromsize</code>	Name of a file defining chromosome sizes. Will be used when 'chromgrs' and 'genome' are missing. It can be downloaded from UCSC, e.g. for hg19, <a href="http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/chromInfo.txt.gz">http://hgdownload.cse.ucsc.edu/goldenpath/hg19/database/chromInfo.txt.gz</a> Required to have at least two tab-delimited columns without any header: <ol style="list-style-type: none"> <li>1. chromosome name, e.g. chr1</li> </ol>

	2. chromosome length, e.g. 249250621
	Both uncompressed and gzipped files are supported. Default: NULL
radius	Region length (bp) of gene's upstream and downstream to be excluded from intergenic region. Default: 10,000
feat	Feature in the GTF file (column 3) to-be-used for defining genic region. Default: exon
chroms	A vector of chromosomes names to define intergenic regions. e.g. c('chr10', 'chr11') Default: NULL

**Value**

a GRanges object of intergenic regions

**Examples**

```
fgtf = system.file('extdata/gtf/defIgRanges_in.gtf', package='pram')

defIgRanges(fgtf, genome='hg38')
```

---

evalModel

*Evaluate transcript model*


---

**Description**

Evaluate trascript model's precision and recall on exon nucleotides, splice junctions, and splice patterns by comparing them to transcript targets

**Usage**

```
evalModel(model_exons, target_exons)

## S4 method for signature 'GRanges,GRanges'
evalModel(model_exons, target_exons)

## S4 method for signature 'character,character'
evalModel(model_exons, target_exons)

## S4 method for signature 'data.table,data.table'
evalModel(model_exons, target_exons)

## S4 method for signature 'character,data.table'
evalModel(model_exons, target_exons)
```

**Arguments**

`model_exons`      genomic coordinates for transcript model exons  
`target_exons`     genomic coordinates for transcript target exons

**Value**

a data table of precision, recall, number of true positive, false negative, false positive for all three evaluated features

**Methods (by class)**

- `model_exons = GRanges, target_exons = GRanges`: Both **model\_exons** and **target\_exons** are `GRanges` objects to define genomic coordinates of exons. Required to have a meta-data column named 'trid' to define each exon's transcript ID.
- `model_exons = character, target_exons = character`: Both **model\_exons** and **target\_exons** are GTF files with full names. Each GTF file is required to have a 'transcript\_id' tag in column 9.
- `model_exons = data.table, target_exons = data.table`: Both **model\_exons** and **target\_exons** are `data.table` objects to define exon genomic coordinates. Required to have the following columns:
  - `chrom`: exon's chromosome, e.g. 'chr8'
  - `start`: exon's start position
  - `end`: exon's end position
  - `strand`: exon's strand, '+' or '-'
  - `trid`: exon's transcript ID
- `model_exons = character, target_exons = data.table`: The **model\_exons** is a GTF file with full name and **target\_exons** is a `data.table` object. Requirements for GTF and `data.table` are the same as above

**Examples**

```
fmdl = system.file('extdata/benchmark/plcf.tsv', package='pram')
ftgt = system.file('extdata/benchmark/tgt.tsv', package='pram')

mdltdt = data.table::fread(fmdl, header=TRUE, sep="\t")
tgttdt = data.table::fread(ftgt, header=TRUE, sep="\t")

evalModel(mdltdt, tgttdt)
```

---

`prepIgBam`*Extract alignments in intergenic regions from BAM files*

---

**Description**

Extract alignments in intergenic regions from BAM files

**Usage**

```
prepIgBam(finbam, iggrs, foutbam, max_uni_n_dup_aln = 10,  
          max_mul_n_dup_aln = 10)
```

**Arguments**

<code>finbam</code>	Full name of an input RNA-seq BAM file. Currently, PRAM only supports strand-specific paired-end data with the first mate on the right-most of transcript coordinate, i.e., 'fr-firststrand' by Cufflinks's definition
<code>iggrs</code>	A GenomicRanges object defining intergenic regions
<code>foutbam</code>	Full name of an output BAM file to save all alignment fell into intergenic regions
<code>max_uni_n_dup_aln</code>	Maximum number of uniquely mapped fragments to report per each alignment. Default: 10
<code>max_mul_n_dup_aln</code>	Maximum number of multi-mapping fragments to report per each alignment. Default: 10

**Value**

None

**Examples**

```
finbam = system.file('extdata/bam/CMPRep2.sortedByCoord.raw.bam',  
                    package='pram')  
  
iggrs = GenomicRanges::GRanges('chr10:77236000-77247000:+')  
  
foutbam = tempfile(fileext='.bam')  
  
prepIgBam(finbam, iggrs, foutbam)
```

---

runPRAM	<i>Predict intergenic transcript models from RNA-seq</i>
---------	--

---

**Description**

Predict intergenic transcript models from RNA-seq

**Usage**

```
runPRAM(in_gtf, in_bamv, out_gtf, method, cufflinks = "",
        stringtie = "", taco = "")
```

**Arguments**

in_gtf	An input GTF file for defining genomic coordinates of existing genes. Required to have 'gene_id' in the attribute column (column 9)
in_bamv	A character vector of input BAM file(s). If mode 'cf' or 'st' is used, only one input RNA-seq BAM file is allowed. Currently, PRAM only supports strand-specific paired-end data with the first mate on the right-most of transcript coordinate, i.e., 'fr-firststrand' by Cufflinks's definition.
out_gtf	An output GTF file of predicted transcript models
method	A character string defining PRAM's model building method. Current available methods are: <ul style="list-style-type: none"> <li>• plcf: pooling + cufflinks</li> <li>• plst: pooling + stringtie</li> <li>• cfmg: cufflinks + cuffmerge</li> <li>• stmg: stringtie + merging</li> <li>• cftc: cufflinks + taco</li> <li>• cf: cufflinks</li> <li>• st: stringtie</li> </ul> Default: 'plcf'
cufflinks	Cufflinks executable. Required by mode 'plcf', 'cfmg', and 'cf'. For mode 'cfmg', executable files of Cuffmerge, Cuffcompare, and gtf_to_sam from the Cufflinks suite are assumed to be under the same folder as Cufflinks. All the executables are available to download for Linux <a href="http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.2.1.Linux_x86_64.tar.gz">http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.2.1.Linux_x86_64.tar.gz</a> and MacOS <a href="http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.1.1.OSX_x86_64.tar.gz">http://cole-trapnell-lab.github.io/cufflinks/assets/downloads/cufflinks-2.1.1.OSX_x86_64.tar.gz</a> . Source code can be obtained from <a href="http://cole-trapnell-lab.github.io/cufflinks/">http://cole-trapnell-lab.github.io/cufflinks/</a> . Default: ""
stringtie	StringTie executable file. Required by mode 'plst', 'stmg', and 'st'. Executable can be downloaded for Linux <a href="http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.Linux_x86_64.tar.gz">http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.Linux_x86_64.tar.gz</a> and MacOS <a href="http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.OSX_x86_64.tar.gz">http://ccb.jhu.edu/software/stringtie/dl/stringtie-1.3.3b.OSX_x86_64.tar.gz</a> . Source



code can be obtained from <https://ccb.jhu.edu/software/stringtie/>. Default: ”

taco TACO executable file. Required by mode 'cftc'. Executable can be downloaded for Linux [https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.Linux\\_x86\\_64.tar.gz](https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.Linux_x86_64.tar.gz) and MacOS [https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.OSX\\_x86\\_64.tar.gz](https://github.com/tacorna/taco/releases/download/v0.7.0/taco-v0.7.0.OSX_x86_64.tar.gz). Source code can be obtained from <https://tacorna.github.io>. Default: ”

**Value**

None

**Examples**

```
in_gtf = system.file('extdata/demo/in.gtf', package='pram')

in_bamv = c(system.file('extdata/demo/SZP.bam', package='pram'),
            system.file('extdata/demo/TLC.bam', package='pram') )

pred_out_gtf = tempfile(fileext='.gtf')

## assuming the stringtie binary is in folder /usr/local/stringtie-1.3.3/
## you can run runPRAM() by the following example
##
# runPRAM(in_gtf, in_bamv, pred_out_gtf, method='plst',
#         stringtie='/usr/local/stringtie-1.3.3/stringtie')
```

selModel

*Select transcript models***Description**

Select transcript models

**Usage**

```
selModel(fin_gtf, fout_gtf, min_n_exon = 2, min_tr_len = 200,
        info_keys = c("transcript_id"))
```

**Arguments**

fin\_gtf Character of an input GTF file that contains transcript models. Required to have 'transcript\_id' in the attribute column (column 9)

fout\_gtf Character of an output GTF file that contains selected transcript models

min\_n\_exon Minimum number of exons a transcript model required to have Default: 2

<code>min_tr_len</code>	Minimum length (bp) of exon(s) and intron(s) a transcript model required to have Default: 200
<code>info_keys</code>	A vector of characters defining the attributes in input GTF file's column 9 to be saved in the output GTF file. 'transcript_id' will always be saved. Default: c('transcript_id')

**Value**

None

**Examples**

```
fin_gtf = system.file('extdata/gtf/selModel_in.gtf', package='pram')  
fout_gtf = tempfile(fileext='.gtf')  
  
selModel(fin_gtf, fout_gtf)
```

# Index

`buildModel`, [2](#)

`defIgRanges`, [4](#)

`evalModel`, [5](#)

`evalModel`, `character`, `character-method`  
(`evalModel`), [5](#)

`evalModel`, `character`, `data.table-method`  
(`evalModel`), [5](#)

`evalModel`, `data.table`, `data.table-method`  
(`evalModel`), [5](#)

`evalModel`, `GRanges`, `GRanges-method`  
(`evalModel`), [5](#)

`prepIgBam`, [7](#)

`runPRAM`, [8](#)

`selModel`, [9](#)