Pooling RNA-seq and Assembling Models

Peng Liu, Colin N. Dewey, and Sündüz Keleş

2018-11-06

Contents

Introduction

Pooling RNA-seq and Assembling Models (**PRAM**) is an **R** package that utilizes multiple RNA-seq datasets to predict transcript models. The workflow of PRAM contains four steps, which is shown in the figure below with function names and associated key parameters. In later sections of this vignette, we will describe each function in details.

Installation

Use the following **R** command on **Linux** or **macOS**

```
devtools::install_github('pliu55/pram')
```

Quick start

PRAM provides a function `runPRAM()` to let you run through the whole workflow.

For a given gene annotation and RNA-seq alignments, you can predict transcript models in intergenic genomic regions:

```
runPRAM(in_gtf, in_bamv, out_gtf)
```

- `in_gtf`: an input GTF file defining genomic coordinates of existing genes. Required to have an attribute of **gene_id** in the ninth column.
- `in_bamv`: a vector of input BAM file(s) containing RNA-seq alignments. Currently, PRAM only supports strand-specific paired-end RNA-seq with the first mate on the right-most of transcript coordinate, i.e., 'fr-firststrand' by Cufflinks definition.
- `out_gtf`: an output GTF file of predicted transcript models

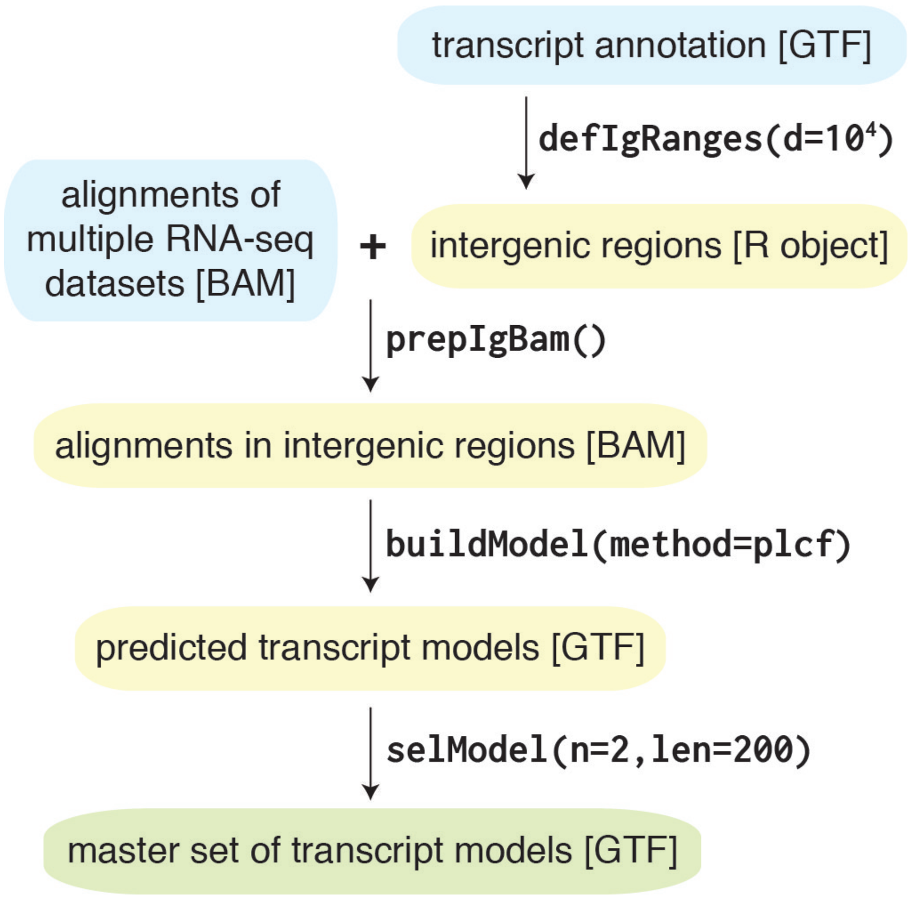
transcript annotation [GTF]
defIgRanges(d=10⁴)
alignments of multiple RNA-seq datasets [BAM]
+
intergenic regions [R object]
prepIgBam()
alignments in intergenic regions [BAM]
buildModel(method=plcf)
predicted transcript models [GTF]
selModel(n=2,len=200)
master set of transcript models [GTF]

Figure 1: PRAM workflow

Examples

PRAM has included input examples files in its `extdata/demo/` folder. The table below provides a quick summary of all the example files.

Table 1: `runPRAM()`'s input example files.

input argument	file name(s)
`in_gtf`	in.gtf
`in_bamv`	SZP.bam, TLC.bam

You can access example files by `system.file()` in **R**, e.g. for the argument `in_gtf`, you can access its example file by

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

Below shows usage of `runPRAM()` with example input files:

```
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')

runPRAM(in_gtf, in_bamv, pred_out_gtf)
```

Define intergenic genomic ranges: `defIgRanges()`

To predict intergenic transcripts, we must first define intergenic regions by `defIgRanges()`. This function requires a GTF file containing known gene annotation supplied for its `in_gtf` argument. This GTF file should contain an attribue of **gene_id** in its ninth column. We provided an example input GTF file in PRAM package: `extdata/gtf/defIGRanges_in.gtf`.

In addition to gene annotation, `defIgRanges()` also requires user to provide chromosome sizes so that it would know the maximum genomic ranges. You can provide one of the following arguments:

- `chromgrs`: a GRanges object, or
- `genome`: a genome name, currently supported ones are: **hg19**, **hg38**, **mm9**, and **mm10**, or
- `fchromsize`: a UCSC genome browser-style size file, e.g. hg19.

By default, `defIgRanges()` will define intergenic ranges as regions 10 kb away from any known genes. You can change it by the `radius` argument.

Example

```
defIgRanges( system.file('extdata/gtf/defIgRanges_in.gtf', package='pram'),
             genome = 'hg38' )
```

Prepare input RNA-seq alignments: `prepIgBam()`

Once intergenic regions were defined, `prepIgBam()` will extract corresponding RNA-seq alignments from input BAM files. In this way, transcript models predicted at later stage will solely from intergenic regions. Also, with fewer RNA-seq alignments, model prediction will run faster.

Three input arguments are required by `prepIgBam()`:

- `finbam`: an input RNA-seq BAM file sorted by genomic coordinate. Currently, we only support strand-specific paired-end RNA-seq data with the first mate on the right-most of transcript coordinate, i.e. 'fr-firststrand' by Cufflinks's definition.
- `iggrs`: a GRanges object to define intergenic regions.
- `foutbam`: an output BAM file.

Example

```
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)
```

Build transcript models: `buildModel()`

`buildModel()` predict transcript models from RNA-seq BAM file(s). This function requires two arguments:

- `in_bamv`: a vector of input BAM file(s)
- `out_gtf`: an output GTF file containing predicted transcript models

Transcript prediction methods

`buildModel()` has implemented seven transcript prediction methods. You can specify it by the `method` argument with one of the keywords: **plcf**, **plst**, **cfmg**, **cftc**, **stmg**, **cf**, and **st**. The first five denote meta-assembly methods that utilize multiple RNA-seq datasets to predict a single set of transcript models. The last two represent methods that predict transcript models from a single RNA-seq dataset.

The table below compares prediction steps for these seven methods. By default, `buildModel()` uses **plcf** to predict transcript models.

Table 2: Prediction steps of the seven `buildModel()` methods

method	meta-assembly	preparing RNA-seq input	building transcripts	assembling transcripts
plcf	yes	pooling alignments	Cufflinks	no
plst	yes	pooling alignments	StringTie	no
cfmg	yes	no	Cufflinks	Cuffmerge
cftc	yes	no	Cufflinks	TACO
stmg	yes	no	StringTie	StringTie-merge
cf	no	no	Cufflinks	no
st	no	no	StringTie	no

Required external software

Depending on your specified prediction method, `buildModel()` requires external software: Cufflinks, StringTie and/or TACO, to build and/or assemble transcript models. You can either specify the software location using the `cufflinks`, `stringtie`, and `taco` arguments in `buildModel()`, or simply leave these three arugments undefined and let PRAM download them for you automatically. The table below summarized software versions `buildModel()` would download when required software was not specified. Please note that, for **macOS**, pre-compiled Cufflinks binary versions 2.2.1 and 2.2.0 appear to have an issue on processing BAM files, therefore we recommend to use version 2.1.1 instead.

Table 3: `buildModel()`-required software and recommended version

software	Linux binary	macOS binary	required by
Cufflinks, Cuffmerge	v2.2.1	v2.1.1	**plcf**, **cfmg**, **cftc**, and **cf**
StringTie, StringTie-merge	v1.3.3b	v1.3.3b	**plst**, **stmg**, and **st**
TACO	v0.7.0	v0.7.0	**cftc**

Example

```
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')

buildModel(fbams, foutgtf, method='plst')
```

Select transcript models: `selModel()`

Once transcript models were built, you may want to select a subset of them by their genomic features. `selModel()` was developed for this purpose. It allows you to select transcript models by their total number of exons and total length of exons and introns.

`selModel()` requires two arguments:

- `fin_gtf`: input GTF file containing to-be-selected transcript models. This file is required to have **transcript_id** attribute in the ninth column.
- `fout_gtf`: output GTF file containing selected transcript models.

By default: `selModel()` will select transcript models with >= 2 exons and >= 200 bp total length of exons and introns. You can change the default using the `min_n_exon` and `min_tr_len` arguments.

Example

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

fout_gtf = tempfile(fileext='.gtf')

selModel(fin_gtf, fout_gtf)
```