FlipFlop: Fast Lasso-based Isoform Prediction as a Flow Problem

Elsa Bernard Laurent Jacob Julien Mairal Jean-Philippe Vert

April 5, 2016

Abstract

FlipFlop implements a fast method for *de novo* transcript discovery and abundance estimation from RNA-Seq data. It differs from Cufflinks by simultaneously performing the transcript and quantitation tasks using a penalized maximum likelihood approach, which leads to improved precision/recall. Other softwares taking this approach have an exponential complexity in the number of exons in the gene. We use a novel algorithm based on network flow formalism, which gives us a polynomial runtime. In practice, *FlipFlop* was shown to outperform penalized maximum likelihood based softwares in terms of speed and to perform transcript discovery in less than 1/2 second even for large genes.

1 Introduction

Over the past decade, quantitation of mRNA molecules in a cell population has become a popular approach to study the effect of several factors on cellular activity. Typical applications include the detection of genes whose expression varies between two or more populations of samples (differential analysis), classification of samples based on gene expression (**?**), and clustering, which consists of identifying a grouping structure in a sample set (**?**). While probe-based DNA microarray technologies only allow to quantitate mRNA molecules whose sequence is known in advance, the recent development of deep sequencing has removed this restriction. More precisely, RNA-Seq technologies (**?**) allow the sequencing of cDNA molecules obtained by reverse transcription of RNA molecules present in the cell. Consequently, any transcript can be sequenced and therefore quantitated, even though its sequence might not be available a priori for designing a specific probe. In addition to facilitating the study of non-coding parts of known genomes and organisms whose genome has not been sequenced (**?**), RNA-Seq technologies facilitate the quantitation of alternatively spliced genes. Genes in eukaryote cells indeed contain a succession of exon and intron sequences. Transcription results in a pre-mRNA molecule from which most introns are removed and some exons are retained during a processing step called RNA splicing. It is

estimated that more than 95% of multiexonic genes are subject to alternative splicing (**?**): the set of exons retained during splicing can vary, resulting for the same gene in different versions of the mRNA, referred to as transcripts or isoforms. Identification and quantification of isoforms present in a sample is of outmost interest because different isoforms can later be translated as different proteins. Detection of isoforms whose presence or quantity varies between samples may lead to new biomarkers and highlight novel biological processes invisible at the gene level.

Sequencing technologies are well suited to transcript quantitation as the read density observed along the different exons of a gene provide information on which alternatively spliced mRNAs were expressed in the sample, and in which proportions. Since the read length is typically smaller than the mRNA molecule of a transcript, identifying and quantifying the transcripts is however difficult: an observed read mapping to a particular exon may come from an mRNA molecule of any transcript containing this exon. Some methods consider that the set of expressed isoforms (**?**) or a candidate superset (**??**) is known in advance, in which case the only problem is to estimate their expression. However little is known in practice about the possible isoforms of genes, and restricting oneself to isoforms that have been described in the literature may lead to missing new ones.

Two main paradigms have been used so far to estimate expression at the transcript level while allowing de novo transcript discovery. On the one hand, the Cufflinks software package (**?**) proceeds in two separate steps to identify expressed isoforms and estimate their abundances. It first estimates the list of alternatively spliced transcripts by building a small set of isoforms containing all observed exons and exon junctions. In a second step, the expression of each transcript is quantified by likelihood maximization given the list of transcripts. Identification and quantification are therefore done independently. On the other hand, a second family of methods (**?????**) jointly estimates the set of transcripts and their expression using a penalized likelihood approach. These methods model the likelihood of the expression of all possible transcripts, possibly after some preselection, and the penalty encourages sparse solutions that have a few expressed transcripts.

The two-step approach of Cufflinks (**?**) is reasonably fast, but does not exploit the observed read density along the gene, which can be a valuable information to identify the set of transcripts. This is indeed a conclusion drawn experimentally using methods from the second paradigm (see **?????**).

To summarize, the first paradigm is fast but can be less statistically powerful than the second one in some cases, and the second paradigm should always be powerful but becomes untractable for genes with many exons. The contribution of this paper is to allow methods of the second family to run efficiently without prefiltering the set of isoform candidates, although they solve a non-smooth optimization problem over an exponential number of variables. To do so, we show that the penalized likelihood maximization can be reformulated as a convex cost network flow problem, which can be solved efficiently (**???**).

For more detail about the statistical model and method, see **?** and references therein.

2 Software features

FlipFlop takes aligned reads in `sam` format and offers the following functionalities:

Transcript discovery *FlipFlop* estimates the set of transcripts which are most likely to be expressed according to the model described in **?**.

Abundance estimation The implemented method simultaneously estimates the abundance of the expressed transcripts in FPKM.

3 Case studies

We now show on a simple one gene example how *FlipFlop* can be used to estimate which transcripts are expressed in an RNA-Seq experiment, and what are the transcript abundances.

3.1 Loading the library and the data

We load the *FlipFlop* package by typing or pasting the following codes in R command line:

```
> library(flipflop)
```

A `.sam` data file can be loaded by the following command:

```
> data.file <- system.file(file.path('extdata', 'vignette-sam.txt'), package='flipflop')
```

These toy data correspond to the alignments of 1000 single-end reads of 125 base-pair long against the hg19 reference genome, available on the UCSC genome browser [1]. The reads have been simulated with the RNASeqReadSimulator [2] from two annotated human transcripts (see reference ID uc001alm.1 and uc001aln.3 in the UCSC genome browser).

In a general context `data.file` should simply be the path to the `.sam` alignement file.

FlipFlop pre-processing of the reads (exctracting exon boundaries, junctions and associated counts), is based on the `processsam` function from the `isolasso` software. More information about the `isolasso` software and the `processsam` options can be found at the following link: `http://alumni.cs.ucr.edu/~liw/isolasso.html`.

Note that the `.sam` file has to be sorted according to chromosome name and starting position. In Unix or Mac systems, it can be done with the command `sort -k 3,3 -k 4,4n in.sam > in.sorted.sam`.

[1] http://genome.ucsc.edu

[2] http://alumni.cs.ucr.edu/ liw/rnaseqreadsimulator.html

3.2 Estimation

In order to estimate the set of expressed isoforms and their abundances, we run the `flipflop` function on the `.sam` file. By default the reads are considered as single-end and no annotation file is necessary. If you have paired-end reads you can use the option `paired=TRUE` and give the mean fragment size (option `frag`) and standard deviation (option `std`) of your RNA-seq library.

3.2.1 without annotation

```
> # The minimum number of clustered reads
> # to consider a cluster of reads as a gene (default 40):
> min.read <- 50
> # The minimim number of spanning reads
> # to consider a valid junction
> min.junc=10
> # The maximum number of isoforms given
> # during regularization path (default 10):
> max.iso <- 7
> ff.res <- flipflop(data.file=data.file,
+                    out.file='FlipFlop_output.gtf',
+                    minReadNum=min.read,
+                    minJuncCount=min.junc,
+                    max_isoforms=max.iso)

> names(ff.res)

[1] "transcripts"     "abundancesFPKM"  "expected.counts" "timer"
```

The `flipflop` function outputs a list whose important features are lists `transcripts`, `abundancesFPKM` and `expected.counts`. Each element of these lists corresponds to a different gene in the `sam` file.

The `transcripts` list is a `GRangesList` object from the *GenomicRanges* package (**?**). More information concerning manipulations of this object can be found in (**?**). Each element of the list is a `GRanges` object that describes the structure of the transcripts that are found to be expressed. Rows of the object correspond to exons. On the left hand side each exon is described by the gene name, the chromosome, its genomic position on the chromosome and the strand. Transcripts are described on the right hand side. Every transcript is a binary vector where an exon is labelled by 1 if it is included in the transcript. Elements of `abundancesFPKM` are vector whose length is the number of isoforms listed in the `transcripts` object. Each element of the vector is the estimated abundance in FPKM of the corresponding transcript. `expected.counts` has the same structure whereas

it corresponds to the expected fragment counts for each transcript (ie the expected number of mapped fragments by transcript).

```
> transcripts <- ff.res$transcripts[[1]]
> abundancesFPKM <- ff.res$abundancesFPKM[[1]]
> expected.counts <- ff.res$expected.counts[[1]]
> print(transcripts)

GRanges object with 7 ranges and 3 metadata columns:
        seqnames             ranges strand | read.count transcript.V1
           <Rle>          <IRanges>  <Rle> |  <numeric>     <numeric>
  Inst1     chr1 [4715106, 4715515]      + |        147             1
  Inst1     chr1 [4771960, 4772760]      + |        382             1
  Inst1     chr1 [4829913, 4830001]      + |         85             1
  Inst1     chr1 [4832340, 4832586]      + |        137             1
  Inst1     chr1 [4834487, 4834619]      + |         88             1
  Inst1     chr1 [4837461, 4837845]      + |         41             0
  Inst1     chr1 [4842605, 4843851]      + |        340             1
        transcript.V2
            <numeric>
  Inst1             1
  Inst1             1
  Inst1             1
  Inst1             1
  Inst1             1
  Inst1             1
  Inst1             0
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths

> print(abundancesFPKM)

         [,1]     [,2]
[1,] 278294.3 113947.9

> print(expected.counts)

NULL
```

Our example `sam` file contains a gene with 7 exons. Two transcripts were found to be expressed, with respective abundances 278210.2 and 114046.3 FPKM. The first of the expressed isoforms contains all exons except the exon 6, the second isoform does not contain exon 7.

The output is also stored in a standart `gtf` format file. For more details about the GTF format visit `http://mblab.wustl.edu/GTF2.html`. In the so-called attributes column, `FPKM` corresponds to abundances in FPKM unit while `EXP-COUNT` corresponds to the expected fragment counts.

3.2.2 with annotation

The `flipflop` function allows as well the use of an annotated transcript file in `bed` format to settle a priori the exon boundaries. More precisely, the `bed` file must be a `bed12` file with 12 columns. It also has to be sorted according to chromosome name and starting position of isoforms.

A `.bed` annotation file can be loaded by the following command:

```
> annot.file <- system.file(file.path('extdata', 'vignette-annot.bed.txt'),
+                           package='flipflop')

> ff.res.annot <- flipflop(data.file=data.file,
+                     out.file='FlipFlop_output.gtf',
+                     annot.file=annot.file)

> transcripts.annot <- ff.res.annot$transcripts[[1]]
> print(transcripts.annot)

GRanges object with 13 ranges and 3 metadata columns:
        seqnames             ranges strand   | read.count transcript.V1
           <Rle>          <IRanges>  <Rle>   |  <numeric>     <numeric>
  Inst1     chr1 [4715105, 4715515]      +   |        147             1
  Inst1     chr1 [4715515, 4771960]      +   |          0             0
  Inst1     chr1 [4771960, 4772760]      +   |        382             1
  Inst1     chr1 [4772760, 4829913]      +   |          0             0
  Inst1     chr1 [4829913, 4830001]      +   |         85             1
    ...      ...                 ...    ... ...        ...           ...
  Inst1     chr1 [4834487, 4834619]      +   |         88             1
  Inst1     chr1 [4834619, 4837461]      +   |          0             0
  Inst1     chr1 [4837461, 4837855]      +   |         42             0
  Inst1     chr1 [4837855, 4842605]      +   |          0             0
  Inst1     chr1 [4842605, 4843851]      +   |        339             1
        transcript.V2
            <numeric>
  Inst1             1
  Inst1             0
  Inst1             1
```

```
  Inst1             0
  Inst1             1
    ...           ...
  Inst1             1
  Inst1             0
  Inst1             1
  Inst1             0
  Inst1             0
  -------
  seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Two transcripts are again found to be expressed. The number of exons and the positions on the genome are not the same as in the previous example because boundaries now include exons and introns from the annotation.

3.3 Read the output GTF file

The transcripts which are found to be expressed are stored in a `gtf` format file. The transcript information from the GTF file can be easily extracted using the `makeTxDbFromGFF` function from the *GenomicFeatures* package (**?**).

The following example shows (when the *GenomicFeatures* is installed) how to create a `TxDb` object from the GTF file, and several accessor functions allow to manipulate it. More information can be found in (**?**). For instance the `exonsBy` function extracts the list of exons for each gene or transcript:

```
> if(require(GenomicFeatures)){
+    txdb <- makeTxDbFromGFF(file='FlipFlop_output.gtf', format='gtf')
+    # List of exons for each transcript:
+    exonsBy(txdb, by='tx')
+ }

GRangesList object of length 2:
$1
GRanges object with 6 ranges and 3 metadata columns:
      seqnames             ranges strand |   exon_id   exon_name exon_rank
         <Rle>          <IRanges>  <Rle> | <integer> <character> <integer>
  [1]     chr1 [4715105, 4715514]      + |         1        <NA>         1
  [2]     chr1 [4771960, 4772759]      + |         2        <NA>         2
  [3]     chr1 [4829913, 4830000]      + |         3        <NA>         3
  [4]     chr1 [4832340, 4832585]      + |         4        <NA>         4
  [5]     chr1 [4834487, 4834618]      + |         5        <NA>         5
  [6]     chr1 [4837461, 4837854]      + |         6        <NA>         6
```

```
$2
GRanges object with 6 ranges and 3 metadata columns:
      seqnames             ranges strand | exon_id exon_name exon_rank
  [1]     chr1 [4715105, 4715514]      + |       1      <NA>         1
  [2]     chr1 [4771960, 4772759]      + |       2      <NA>         2
  [3]     chr1 [4829913, 4830000]      + |       3      <NA>         3
  [4]     chr1 [4832340, 4832585]      + |       4      <NA>         4
  [5]     chr1 [4834487, 4834618]      + |       5      <NA>         5
  [6]     chr1 [4842605, 4843851]      + |       7      <NA>         6

-------
seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

4 Session Information

```
R version 3.2.4 (2016-03-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] stats4    parallel  stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
[1] GenomicFeatures_1.22.13 AnnotationDbi_1.32.3    Biobase_2.30.0
[4] GenomicRanges_1.22.4    GenomeInfoDb_1.6.3      IRanges_2.4.8
[7] S4Vectors_0.8.11        BiocGenerics_0.16.1     flipflop_1.8.2

loaded via a namespace (and not attached):
 [1] XVector_0.10.0             GenomicAlignments_1.6.3
 [3] zlibbioc_1.16.0            BiocParallel_1.4.3
 [5] lattice_0.20-33            tools_3.2.4
 [7] SummarizedExperiment_1.0.2 grid_3.2.4
 [9] DBI_0.3.1                  lambda.r_1.1.7
[11] futile.logger_1.4.1        Matrix_1.2-4
[13] rtracklayer_1.30.4         futile.options_1.0.0
```

```
[15] bitops_1.0-6               RCurl_1.95-4.8
[17] biomaRt_2.26.1             RSQLite_1.0.0
[19] Biostrings_2.38.4          Rsamtools_1.22.0
[21] XML_3.98-1.4
```