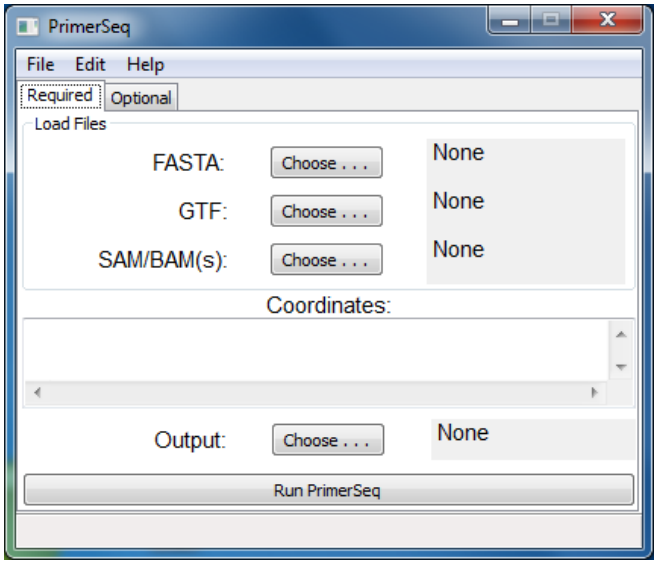
Content

Overview

PrimerSeq designs RT-PCR primers that evaluate alternative splicing events by incorporating RNA-Seq data. PrimerSeq is particularly advantageous for designing a large number of primers for validating alternative splicing events found in RNA-Seq data. PrimerSeq incorporates RNA-Seq data in the design process to weight exons by their read counts. Essentially, the RNA-Seq data allows primers to be placed using actually expressed transcripts. This could be for your particular cell line or experimental condition, rather than using annotations that incorporate transcripts that are not expressed for your data. Alternatively, you can design primers that are always on constitutive exons. PrimerSeq does not limit the use of gene annotations and can be used for a wide array of species. For detailed examples, please visit the website at http://primerseq.sourceforge.net/.

Fig. 1 PrimerSeq provides a Graphical User Interface (GUI) to allow convenient access by a wide range of users. PrimerSeq is available on the Windows operating system.



Input

Users must specify target exon(s), gene annotation (GTF), genome sequence (FASTA), and RNA-Seq mapped reads (BAM or SAM, optional). The following sections detail technical aspects of the input. For information regarding loading the input into PrimerSeq, please see the getting started web page at http://primerseq.sourceforge.net/getting_started.html.

Exon Coordinates

Users specify the target exon(s) as a list of (+|-)chr:start-end in 0-based format (e.g. -chr17:73969705-73969866).

Gene Annotation

PrimerSeq uses GTF files to define gene annotations. The GTF file defines the exons (nodes) and the minimum number of splice junctions (edges) in the splice graph (*see Splice Graph*). More junctions may be found based on the RNA-Seq data if the user allows this option. PrimerSeq ignores lines in the GTF that are not "exon" features. Users can download GTFs known to work from the PrimerSeq SourceForge website, download GTFs from UCSC or Ensembl, or use GTF output from transcript assemblers. However, unsorted GTFs or invalid gene IDs in GTF files can cause issues if not corrected.

Valid Gene IDs

If the GTF has valid gene IDs, then the splice graph can be constructed using transcripts in the annotation that all have the same gene ID. Thus it is easy to know which exons (nodes) and splice junctions (edges) are included in the splice graph (*see Splice Graph*).

Invalid Gene IDs

Splice junctions and exons are included in the pertinent splice graph if they are weakly connected to the target exon. This could possibly include annotated transcripts that are not from the same gene. GTFs with invalid gene IDs are often obtained from the UCSC Table Browser.

Sorting

The GTF file must be sorted by, in order, contig name (chromosome), gene name, transcript name, start position, and end position. PrimerSeq provides an option to sort the GTF although the memory requirements may be too high for some desktops. Selected pre-sorted GTFs can be obtained from the download pages at http://primerseq.sourceforge.net/downloads.html.

Genomic Sequence

PrimerSeq uses the FASTA file format as input for genomic sequences. It is more convenient if the FASTA file contains the entire genome of the species of interest, although no restriction is applied. The chromosome names in the FASTA should match the chromosome names of the target exons. Pygr is used to extract sequences in the middle of the FASTA (https://code.google.com/p/pygr/). After pygr indexes the FASTA once, sequence lookups occur rapidly without heavy memory overhead.

Aligned Reads

One or multiple SAM or BAM files can be specified as input to PrimerSeq. If the file does not end with ".sorted.bam" then the file is converted to a sorted BAM file with a ".sorted.bam" extension. The SAM-JDK is used to perform this conversion (http://picard.sourceforge.net/). The SAM-JDK is also used to extract mapped reads in BAM files by indexing the BAM, which only occurs once.

Splice Graph

Building the splice graph consists of constructing a weighted directed acyclic graph (DAG) $G = (V, E)$ where nodes are exons and edges connect the exons. Exons (nodes) are defined by a unique pair of start and end positions. We used NetworkX, a python graph library, to implement the splice graph (Hagberg, *et al.*, 2008).

- Exons, $v \in V$, are obtained from GTF annotation
- Exons are naturally topologically sorted by chromosome position
- Junctions, $e \in E$, are found from the GTF or both GTF and RNA-Seq data
- Junction counts are encoded in edge weights w

The user decides whether junctions are found either from the GTF or by both the GTF and RNA-Seq data. Users can override the default value for the minimum number of junction reads to define a novel splice junction and the minimum number of reads to assign a splice junction that is known from the annotation. This feature is useful for very low read counts. PrimerSeq does not use dummy source nodes or sink nodes like DiffSplice (Hu, *et al.*, 2012) since the goal of PrimerSeq is to find real exons that can be used for primer design.

Alternative Splicing Modules (ASM)

Overview

To find regions where alternative splicing occurs, PrimerSeq uses the biconnected components (BCC) algorithm (Sacomoto, *et al.*, 2012). The splice graph is treated as an undirected graph for the purpose of finding the biconnected components. Only the ASM subgraph that includes the user's target exon is used for further analysis. Previous use of the idea of Alternative Splicing Modules can be seen in the software DiffSplice. The BCC algorithm is implemented in many graph libraries and thus provides an *easy* to use tool for finding ASMs.

Biconnected Components (BCC)

An undirected graph is biconnected if the number of connected components does not increase even after removing a vertex and any edges incident to that vertex. The biconnected components of a graph are the maximal subset of vertices such that the biconnected property holds within each component. Note by technical conditions, every vertex in a graph is a part of at least one biconnected component and cut vertices are defined as being vertices that are a part of more than one biconnected component. Cut vertices if removed from the graph will cause the graph to no longer be connected. More precisely, there will be an increase in the number of distinct connected components. In terms of the splice graph, cut vertices are effectively "constitutive" exons because they cannot be skipped. Despite all vertices being a part of a biconnected component, useful information can be obtained since not all vertices are a part of trivial biconnected components. Trivial biconnected components consist of components that are less than three exons. Since meaningful AS events only occur with three exons, these trivial

biconnected components are filtered from the biconnected components list. This leaves only complex biconnected components with multiple paths through the biconnected component. By definition of biconnected components, all vertices (exons) that are not the first or last exon of a biconnected component must be skippable. First and last exons of a biconnected component, as defined here, have no predecessor nodes and no successor nodes, respectively, in the biconnected component. Note, including the first and last node in the ASM differs from the convention of ASMs in DiffSplice. For further details about BCCs, please see *Equivalence of BCC and DiffSplice ASM*.

Biconnected Components Algorithm

The biconnected components algorithm uses Depth-first Search (DFS) with two stored variables to find the biconnected components in an undirected graph.

1. DFS depth of v
2. Lowest depth of all neighboring nodes of all descendant of v

The DFS based approach results in linear time complexity, $O(|V| + |E|)$. We use a python BCC algorithm implementation from the NetworkX graph library (Hagberg, *et al.*, 2008).

Equivalence of BCC and DiffSplice ASM

DiffSplice utilizes the property of dominance in a DAG to define where several exons could alternatively be included, i.e., ASM. Their methodology relies on a source node which has an edge to all first exons in a transcript and a sink node where every last exon in a transcript has an edge to the sink node. Thus every exon (node) has a path from a source node and to a sink node. DiffSplice describes formally an ASM as an induced subgraph H of a graph G with the following properties.

1. Single entry
2. Single exit
3. Alternative paths
4. Minimal

Using the BCC algorithm on the same splice graph with source and sink nodes, we would identify exactly the same ASMs as the dominance-based approach used by DiffSplice. In an effort to show equivalence we will show that the BCC algorithm satisfies all of the above criteria specified by DiffSplice. We will first introduce some terms used by DiffSplice.

Single entry – all nodes come from a single node, the single entry node (possibly source node)

Single exit – all edges go to a single node, the single exit node (possibly sink node)

Post-dominate – If all paths from a vertex $k \in V$ to a single exit node always includes a vertex $v \in V$ then v is said to post-dominate k.

Pre-dominate – If all paths from a single entry node to a vertex $k \in V$ always includes a vertex $v \in V$ then v is said to pre-dominate k.

Immediate pre/post-dominator – a vertex $v \in V$ that pre/post-dominates a vertex $k \in V$ but does not pre/post-dominate any other pre/post-dominators of k.

1. **Single entry**

First, suppose the BCC algorithm identified a component, an induced subgraph H, with two entry nodes. Since there is a source node, there must be a single node that immediately pre-dominates the two proposed entry nodes, possibly the source node in the original DAG. Thus at minimum there are two distinct paths which lead to the entry nodes from the immediate pre-dominator. This contradicts the maximal subset of vertices property of BCCs since at minimum the immediate pre-dominator node should be included in the BCC since its inclusion allows two alternative paths into the proposed BCC and removing individually the immediate pre-dominator or the two proposed entry nodes as a vertex leaves the BCC as still connected. Therefore with source and sink nodes defined, a BCC cannot have more than one entry node.

2. **Single exit**

Same logic as "1. Single entry" except order reversed for sink node.

3. **Alternative Paths**

The alternative paths criterion requires a simple filtering step of BCCs due to the formal definition of BCC as a subgraph that does not increase the number of connected components after removal of any single vertex. Graphs with less than three connected nodes cannot be separated into two non-connected graphs by removal of a node because the maximal number of connected components in any graph is $|V|$ (i.e. there are no edges in the graph). Therefore BCCs with a number of nodes less than 3 are removed. BCCs with at least three nodes must have alternative paths since the subgraph remains connected even after removing any vertex, implying there is another path from the single entry node to the single exit node.

4. **Minimal**

The 'minimal' condition in DiffSplice states that any node in the ASM (excluding the entry and exit node) cannot post-dominate the entry node or pre-dominate the exit node. Suppose for a BCC that a node other than the entry or exit node could either pre-dominate the exit node or post-dominate the entry node. Removal of that node, a cut vertex, would separate the BCC into a graph before the node and after the node as a result of the definition of pre-dominance or post-dominance. More precisely, this dominance criterion indicates that all paths from entry node to exit node must use a certain node which indicates that its removal would separate the entry node and exit node into two distinct connected graphs. This clearly violates the property of BCC and thus a BCC always follow the 'minimal' condition.

Generating Isoforms

Overview

Generation of isoforms depends on whether the user allows the finding of novel splice junctions from read support in the RNA-Seq data. If the user does not, the isoforms for only the ASM containing the target exon are exactly the isoforms defined in the GTF annotation. If the user allows novel splice junctions, novel generated isoforms for the ASM are added to the list of isoforms that are known from the annotation. It is an important note that the isoforms are only obtained in the region of the ASM and not for the full length of the gene.

Only GTF Splice Junctions

By default, isoforms are obtained from the GTF annotation.

GTF & Novel Splice Junctions

In addition to isoforms in the GTF annotation, novel isoforms are generated based on the novel junctions. All possible paths with in the ASM are considered and only isoforms (a path in the splice graph) that contain at least one novel junction are added to the list of known isoforms from the GTF.

Estimating Isoform Counts & Exon Inclusion

Overview

A common occurrence is that a splice junction occurs in more than one isoform. To avoid inaccurate estimates of isoforms, an EM algorithm was used to estimate the MLE for the probabilities, p, of a multinomial distribution. Specifically the probabilities, p, are the probability that a read came from a certain isoform (i.e. not normalized by length). The MLE estimate for probability of each isoform defines the expected sufficient statistic (i.e. expected read counts) by the equation $c_k = np_k$ where n is the total number of reads and k is an index for each isoform. Assuming expected inclusion counts for exons are binomially distributed, the MLE for exon inclusion, $\hat{\psi}$, is defined by the following equation:

$$\hat{\psi} = \frac{\sum_{i \in I} c_i / |E_i|}{\sum_{t \in T} c_t / |E_t|}$$

Where I denotes indexes corresponding to inclusion forms and T denotes indexes for all isoforms. $|E_k|$ denotes the number of junctions in isoform k. It is an important note that read counts are normalized by junction number after the EM algorithm as indicated by the above equation.

Method

PrimerSeq uses a previous multinomial EM algorithm to approximate inclusion levels (Xing, *et al.*, 2006). Primerseq does not use EST fragments like Xing *et al.* but rather we applied the fundamental algorithm to work with RNA-Seq data and the corresponding Splice Graph.

Choosing Flanking Exons

Overview

There are two methods for systematically selecting flanking exons to place primers on. The first method does not require estimation of exon inclusion level because the user defined that the flanking exons must be 100% included. The second method uses the $\hat{\psi}$ estimates of exons to determine which flanking exons are above a user-defined threshold.

1. **No-Estimation Method**

The cut vertices of the biconnected component are chosen as the flanking exons.

2. Estimation Method

The two closest flanking exons with $\hat{\psi}$ greater than the user-defined threshold are chosen.

Obtaining Designed Primers

Once the flanking exons are chosen, the sequences for the flanking exons and the target exon are obtained from the FASTA. The sequence and associated Primer3 parameters are used as input to Primer3 which then designs the primers. Information about the result is saved as a text file, displayed in the GUI, and viewable as a web page. For detailed user instructions please see the PrimerSeq website at http://primerseq.sourceforge.net.

References

Hagberg, A.A., Schult, D.A. and Swart, P.J. (2008) Exploring Network Structure, Dynamics, and Function using NetworkX. *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, pp. 11 - 15.

Hu, Y., *et al.* (2012) DiffSplice: the genome-wide detection of differential splicing events with RNA-seq. *Nucleic Acids Res.*

Sacomoto, G.A., *et al.* (2012) KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics*, **13 Suppl 6**, S5.

Xing, Y., *et al.* (2006) An expectation-maximization algorithm for probabilistic reconstructions of full-length isoforms from splice graphs. *Nucleic Acids Res*, **34**, 3150-3160.