IR events detection and interpretation using FakIR

Lucile Broseus

Introduction

Abstract

Intron retention (IR) occurs when an intron in a pre-mRNA remains in the mature mRNA. An increasing body of literature has demonstrated a major role for IR in numerous biological functions and in disease. Although experimental technologies used to study other forms of mRNA splicing can also be used to investigate IR, a specialized downstream computational analysis is optimal for IR discovery and analysis. Here we provide a practical guide to detecting and analyzing IR using second and third generation RNA sequencing data. We also indicate code lines for predicting (in-silico) the potential impact of IR on gene expression and protein output.

Summary

Package Installation

The R package *FakIR* can be installed from my GitHub account by copy-pasting the following code line:

```
devtools::install_github("lbroseus/FakIR")
```

To start with, load *FakIR*:

```
suppressPackageStartupMessages( library(FakIR) )
```

We will also make use of the packages *dplyr* and *ggplot2*, available on the CRAN, to explore our IR results:

```
if( !("dplyr" %in% installed.packages()) ) install.packages( "dplyr" )

suppressPackageStartupMessages( library(dplyr) )

if( !("ggplot2" %in% installed.packages()) ) install.packages( "ggplot2" )

suppressPackageStartupMessages( library(ggplot2) )
```

External data

For illustrating how to use this package to detect intron retention events, we provide extracts (chromosome 2) of all standard input files.

Once you have installed *FakIR*, these files can be found in folder:

```
system.file("extdata", package = "FakIR")
```

You will find extracts from several public reference data for *Homo Sapiens*.

1. **A reference transcriptome**. An extraction of the annotated transcripts mapping on the chromosome 2, downloaded from *Ensembl*. We will make use of the release: *Homo_sapiens.GRCh38.97*:

```
gtf <- system.file("extdata", "Homo_sapiens.GRCh38.97.chr2.gtf", package = "FakIR")
```

2. **TSS inferred from CAGE data**.

An extraction of chromosome 2 of ENCODE/RIKEN CAGE data for the MCF7 cell line; we will make use of them to subset aligned full-length long read from a bam file (cf: Extraction of full-length reads).

```
cageFile <- system.file("extdata", "GSM849364_hg38_wgEncodeRikenCageMcf7CellPapTssGencV7.chr2.gtf",
                        package = "FakIR")
```

More and complete public CAGE data for several cell lines can also be found at <>.

Additionaly, we provide extracts from our own in-house matched Second and Third generation sequencing experiments, realised on the human cell line *MCF10A*.
The full raw datasets can be downloaded from *GEO*, under accession number *GSE126638* and were used and described in described in reference *3* (Supplementary Materials).

MCF10A data available in *FakIR* package consists in:

3. **Second generation sequencing data**.

You will find a typical *IRFinder* output file on chromosome 2 from an Illumina RNA-seq experiment:

```
irfinderFile <- system.file("extdata", "MCF10A_chr2.IRFinder-IR-dir.txt", package = "FakIR")
```

This file will be used to detect IR events from short read RNA-seq data (cf: Extraction of Intron Retention events). For more information as to how to generate such a file, please refer to Prerequisite: compute intron coverage information with IRFinder.

4. **Third generation sequencing data**.

We provide a **bam file** of direct-RNA-Nanopore alignments on the chromosome 2, performed using *Minimap2* (reference *4*):

```
bamFile <- system.file("extdata", "MCF10A_chr2.dont.bam", package = "FakIR")
```

Additionaly, using full-length long reads from this bam file, we performed a transcript assembly that we will use especially in Prediction of Premature Termination Codons in assembled transcripts.
You can find this assembly in:

```
assembly.gtf <- system.file("extdata", "MCF10A_chr2.dont_assembly.gtf", package = "FakIR")
```

BSGenome: representation of reference genomes in R

When it comes to retrieve transcript sequence, many R functions make use of reference genome represented as a *BSGenome* object.

You will need to identify, install and import a convenient reference to reproduce the following analyses on you own data. This can be done by adapting the code below to your organism and version of interest:

```
# Display available genomes and identify the one that fits your data:
BSgenome::available.genomes()

# Specify the reference of your choice:
yourGenomeChoice <- "BSgenome.Hsapiens.UCSC.hg38"

# Download its representation in R:
if (interactive()) {
    if (!require("BiocManager"))
        install.packages("BiocManager")
    BiocManager::install( yourGenomeChoice )
}

# Import into R:
BSgenome <- getBSgenome(genome = yourGenomeChoice, masked=FALSE)
```

For our toy dataset extracted from experiments carried on the human cell line MCF10A, we will import the hg38 reference genome:

```
BSgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
```

IR events detection: from Second Generation RNA-seq data

Prerequisite: compute intron coverage information with IRFinder

We will start from computations output using *IRFinder* (cf: https://github.com/williamritchie/IRFinder for more details and reference *1*).

First, you need to build a specific genome index and indicate.
Then, *IRFinder* will call the aligner *STAR* and perform several computations based on the read coverage of introns.

Eventually, *IRFinder* will output several indicators (in a file named *IRFinder-IR-dir.txt*), which are used to filter the best supported IR events (cf: reference *2*).

Here is an example of typical command lines for *Homo Sapiens* data, using the *hg38* reference genome and the release 97 of *Ensembl* human transcriptome (see also for examples with versions and genomes: http://mimirna.centenary.org.au/irfinder/genomes.html):

```
#Step 1: Build a genome index:

bin/IRFinder -m BuildRef -r REF/Human-hg38-release97 \
    -e REF/extra-input-files/RNA.SpikeIn.ERCC.fasta.gz \
    -R REF/extra-input-files/Human_hg38_nonPolyA_ROI.bed \
    ftp://ftp.ensembl.org/pub/release-97/gtf/homo_sapiens/Homo_sapiens.GRCh38.97.gtf.gz
```

```
#Step 2: align (paired-end) reads with STAR and analyse intron coverage:

bin/IRFinder -r REF/Human-hg38-release97  -d irfinder READ_1.fastq READ_2.fastq
```

Extraction of Intron Retention events

IRFinder outputs a file named *IRFinder-IR-dir.txt* containing a series of variables describing coverage properties of annotated introns. These are used to decide whether it shows convincing evidence of retention. For illustration purposes, we provide as External data, IRFinder results obtained from an Illumina RNA-seq experiment carried out on the MCF10A cell line:

```
irfinderFile <- system.file("extdata", "MCF10A_chr2.IRFinder-IR-dir.txt", package = "FakIR")
```

Let's have a look at variables computed by *IRFinder*:

```
irfinder_results <- read.table(irfinderFile, header = T, sep = "\t")

head(irfinder_results, 10)
#>    Chr  Start    End                          Name Null Strand ExcludedBases
#> 1    2  41627  42808 FAM110C/ENSG00000184731/clean    0      -            10
#> 2    2  41627  45439 FAM110C/ENSG00000184731/clean    0      -           164
#> 3    2  41627  46806 FAM110C/ENSG00000184731/clean    0      -          1240
#> 4    2 219001 224863  SH3YL1/ENSG00000035115/clean    0      -           614
#> 5    2 219001 229965  SH3YL1/ENSG00000035115/clean    0      -          1013
#> 6    2 222954 224867  SH3YL1/ENSG00000035115/clean    0      -            14
#> 7    2 224920 229965  SH3YL1/ENSG00000035115/clean    0      -           342
#> 8    2 231191 233100  SH3YL1/ENSG00000035115/clean    0      -          10
#> 9    2 231191 234159  SH3YL1/ENSG00000035115/clean    0      -         149
#> 10   2 233229 234159  SH3YL1/ENSG00000035115/clean    0      -          10
#>     Coverage IntronDepth IntronDepth25Percentile IntronDepth50Percentile
#> 1  0.4440650    0.543710                       0                       0
#> 2  0.4764250    0.550685                       0                       0
#> 3  0.4628080    0.502219                       0                       0
#> 4  0.2707700    0.000000                       0                       0
#> 5  0.1679230    0.000000                       0                       0
#> 6  0.1258560    0.000000                       0                       0
#> 7  0.0531576    0.000000                       0                       0
#> 8  0.5629280    1.151120                       0                       1
#> 9  0.4569000    0.476528                       0                       0
#> 10 0.2380430    0.000000                       0                       0
#>    IntronDepth75Percentile ExonToIntronReadsLeft ExonToIntronReadsRight
#> 1                        2                     0                      0
#> 2                        2                     0                      0
#> 3                        2                     0                      0
#> 4                        1                     9                      0
#> 5                        0                     9                      0
#> 6                        0                     3                      0
#> 7                        0                     0                      0
#> 8                        3                     0                      0
#> 9                        2                     0                      0
#> 10                       0                     1                      0
#>    IntronDepthFirst50bp IntronDepthLast50bp SpliceLeft SpliceRight SpliceExact
#> 1                   0.0                   0         32           2           2
#> 2                   0.0                   0         32          21          20
#> 3                   0.0                   3         32          11           9
#> 4                   8.5                   0        220          94          94
#> 5                   8.5                   0        220         202         122
#> 6                   3.0                   0          0           0           0
#> 7                   0.0                   0         80         202          80
#> 8                   1.0                   0        122         106         106
#> 9                   1.0                   0        122         141          16
#> 10                  0.0                   0        125         141         125
#>        IRratio     Warnings
#> 1  0.013687100     LowCover
#> 2  0.014669900 MinorIsoform
#> 3  0.014256600     LowCover
#> 4  0.001229260 MinorIsoform
#> 5  0.000762703 MinorIsoform
#> 6  0.000000000     LowCover
#> 7  0.000263087 MinorIsoform
#> 8  0.009347190            -
#> 9  0.003229960 MinorIsoform
#> 10 0.001685410            -
```

IR events calling, using advised parameters (see reference *1* and *2*), can be achieved directly (or adjusted) from a standard *IRFinder-IR-dir.txt* file as follows:

```
# Main variables from which IR events are inferred:
minIRratio <- 0.1
minSpliceRead <- 3
```

```
minExonToIntronReads <- 1
minCoverage <- 0.1

# Import the file IRFinder-IR-dir.txt and extract filter IR events:
ir_events <- filterIRevents(irfinderFile = irfinderFile,
                            minIRratio = minIRratio,
                            minSpliceRead = minSpliceRead,
                            minExonToIntronReads = minExonToIntronReads,
                            minCoverage = minCoverage)
```

About other IR detection approaches

Other softwares have been developed to compute intron coverage information (eg: KMA and iRead). You can use them (or your own filtering method) instead of IRFinder to select credible IR events, and still apply the following workflow.
At this step, all you need is to feed the variable *ir_events* with the genomic intervals of your retained introns as a *GRanges* object or as a data.frame that can be coerced to a *GRanges* object (typically with columns chr/seqnames, start, end, strand).

Differential Analysis of IR levels: from Second Generation RNA-seq data

Though it would deserve more precautions (and benchmarking), most approaches that have been proposed so far simply consist in applying existing well-known methods for differential analysis (eg: DESeq2, edgeR, DEXSeq) to intronic features (see reference *2* for a discussion).
FakIR includes a wrapper to perform end-to-end differential analysis of introns through DEXSeq.
Please refer to the DEXSeq vignette (https://www.bioconductor.org/packages/release/bioc/vignettes/DEXSeq/inst/doc/DEXSeq.html) and to the paper (reference *8*) for more information about the method in itself.

To carry out differential analysis of IR-levels using FakIR, you will need:

1. a reference transcriptome in a gtf file (from which to extract intron genomic coordinates);
2. bam files for all samples to be considered in the differential analysis;
3. to know how samples should be grouped together (!)
4. and the library type of your samples (ie: whether they are single or paired-end).

So first, indicate a gtf file for your reference transcriptome. Please adapt the path.

```
gtf <- "Your_favourite_organism_versionXXX.gtf"
```

Then, you must provide the paths to your input bam files and indicate how your samples should be grouped together in the differential test (a "target"). You may also have this in a separate text file and import it into R as a data.frame.

Here is a toy example:

```
files <- c("sample1.WT.bam", "sample2.WT.bam", "sample3.WT.bam",
           "sample1.KO.bam", "sample2.KO.bam", "sample3.KO.bam")

conditions <- c("WT", "WT", "WT",
                "KO", "KO", "KO")

libType <- "single-end" # choose between "single-end" or "paired-end"

target <- data.frame(file = files, condition = conditions)

target
#>             file condition
#> 1 sample1.WT.bam        WT
#> 2 sample2.WT.bam        WT
#> 3 sample3.WT.bam        WT
#> 4 sample1.KO.bam        KO
#> 5 sample2.KO.bam        KO
#> 6 sample3.KO.bam        KO
```

Now, we are ready to run DEXSeq computations through our wrapper function:

```
idxd <- DEXSeq4Introns(gtf, target, libType, verbose = TRUE)
```

This will run intron read counting, estimation of size factors, estimation of dispersions, the differential test and estimation of intron fold changes. It may take a while…

```
table ( idxd$padj < 0.1 )
```

IR events detection: from Third Generation RNA-seq data

Analysis of long read alignments

The first step for a reference-based analysis is to define *independent* intron intervals from a reference transcriptome (see ref 2 for more details on intron definition).
Here, we use the annotation *Homo_sapiens.GRCh38.97* provided by *Ensembl*.

```
intronGR <- createIntronGR(gtf = gtf)
```

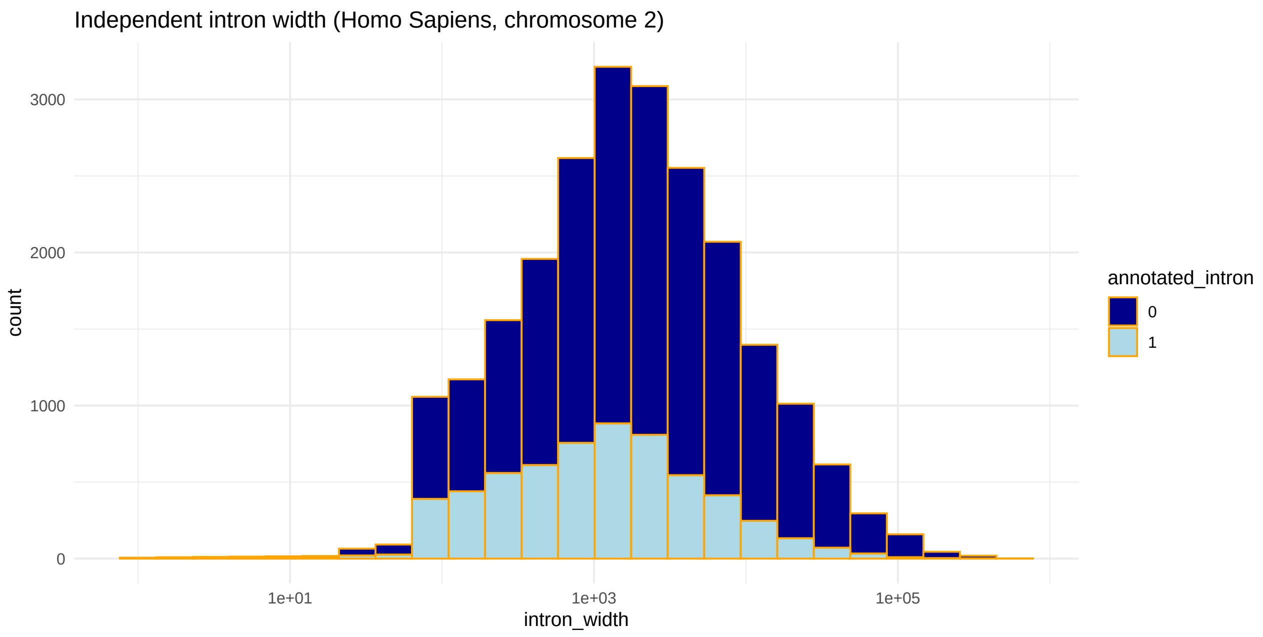
This generates a *GRanges* object with *independent* intron intervals as well as their gene of origin:

```
intronGR
#> GRanges object with 23056 ranges and 2 metadata columns:
#>           seqnames              ranges strand |         gene_id
#>              <Rle>           <IRanges>  <Rle> |     <character>
#>       [1]        2       199902-200163      + | ENSG00000227061
#>       [2]        2       202606-202915      + | ENSG00000227061
#>       [3]        2       202989-209484      + | ENSG00000227061
#>       [4]        2       264499-264722      + | ENSG00000143727
#>       [5]        2       265008-265204      + | ENSG00000143727
#>       ...      ...                 ...    ... .             ...
#>   [23052]        2 242047764-242048057      - | ENSG00000232002
#>   [23053]        2 242048284-242058047      - | ENSG00000232002
#>   [23054]        2 242058885-242059119      - | ENSG00000232002
#>   [23055]        2 242059194-242060296      - | ENSG00000232002
#>   [23056]         2 242060520-242084095      - | ENSG00000232002
#>           annotated_intron
#>                   <factor>
#>       [1]                0
#>       [2]                0
#>       [3]                0
#>       [4]                0
#>       [5]                1
#>       ...              ...
#>   [23052]                0
#>   [23053]                0
#>   [23054]                0
#>   [23055]                0
#>   [23056]                0
#>   -------
#>   seqinfo: 1 sequence from an unspecified genome; no seqlengths
```

Introns that are already part of an reference transcript (transcripts annotated as *retained_intron*) can be identified by the value 1 in column *annotated_intron*:

```
table( intronGR$annotated_intron )
#> 
#>     0     1 
#> 17080  5976
```

```
data.frame(intronGR) %>% mutate(intron_width = end - start + 1) %>%
                         ggplot(aes(intron_width, fill = annotated_intron)) +
                         geom_histogram(bins = 25, color = "orange") + scale_x_log10() +
                         theme_minimal() + scale_fill_manual(values = c("darkblue", "lightblue")) +
                         ggtitle("Independent intron width (Homo Sapiens, chromosome 2)")
```



We then compare long read genome alignments (bam file) to intron intervals using:

```r
IR_events <- callIRevents(bamFile, intronGR)
```

This will create a *data.frame* crossing long reads and the introns they overlap with, along with some properties of the events:
1. **intron_fracoverlap**: the proportion of the intron that is contained in the long read (taking into account splicing and possible gaps);
2. **intron_startOverlap**: indicates whether the 5' exon-intron junction is contained in the long read;
3. **intron_endOverlap**: indicates whether the 3' exon-intron junction is contained in the long read.

The variables *intron_startOverlap* and *intron_endOverlap* can help ensure an intron overlap is an actual IR events, rather than an intronic alternative Transcription Start Site (cf: reference *6*).

Here is an overview of the result:

```r
head(IR_events)
#>                              read_name qwidth  chr   start     end strand
#> 1 1072a86c-67ee-4a66-9c45-c74a6fc28c4c    786 chr2  669183  677383      -
#> 2 8fbf23fc-d804-406a-b0e5-34810a58f3b5    674 chr2  669238  677404      -
#> 3 78b1928e-2a89-4fd0-91da-55ee6c0e7467   2402 chr2 3380934 3479565      +
#> 4 74a323f3-e65c-460a-9d20-3381e0d06df9   1567 chr2 3465652 3479555      +
#> 5 6af5fc43-e025-4f35-a8d4-87ffc23dbc1f   3497 chr2 3575304 3580893      +
#> 6 6af5fc43-e025-4f35-a8d4-87ffc23dbc1f   3497 chr2 3575304 3580893      +
#>           gene_id intron_chr intron_start intron_end intron_strand
#> 1 ENSG00000151353       chr2       669676     669756             -
#> 2 ENSG00000151353       chr2       676705     677288             -
#> 3 ENSG00000171853       chr2      3379877    3387619             +
#> 4 ENSG00000171853       chr2      3477796    3478845             +
#> 5 ENSG00000171863       chr2      3577775    3580109             +
#> 6 ENSG00000171863       chr2      3580476    3580804             +
#>   intron_fracoverlap intron_startOverlap intron_endOverlap
#> 1         0.90123457                TRUE              TRUE
#> 2         0.09589041                TRUE             FALSE
#> 3         0.01924319               FALSE             FALSE
#> 4         1.00000000                TRUE              TRUE
#> 5         1.00000000                TRUE              TRUE
#> 6         1.00000000                TRUE              TRUE
```

Each line corresponds to a long read and one intron overlap. Long read overlapping several introns will then give rise to several lines in the data.frame.
IR event calling can then be easily performed. See the following for some examples.

How to use long read coverage information

Extract reliable IR events

You can extract complete intron retention events:

```r
full_IR_events <- IR_events %>%
                  filter(intron_fracoverlap == 1 & intron_startOverlap & intron_endOverlap)
```

Count full and partial IR events

```
retained_introns <- IR_events %>%
                    filter(intron_fracoverlap == 1 & intron_startOverlap & intron_endOverlap) %>%
                    group_by(intron_chr, intron_start, intron_end, intron_strand, gene_id) %>%
                    summarise(read_count = n()) %>%
                    data.frame()
#> `summarise()` regrouping output by 'intron_chr', 'intron_start', 'intron_end', 'intron_strand' (over
```

Examine intron-retaining long reads

In order to extract long reads containing at least one intron at its full extent:

```
IR_reads <- full_IR_events %>%
            group_by(read_name) %>%
            summarise(nbfullRI = n()) %>%
            data.frame()
#> `summarise()` ungrouping output (override with `.groups` argument)
```

IR transcript reconstruction: from Third Generation RNA-seq data

Typical transcriptome assembly from long reads

Here is the general *StringTie2* command line to assemble transcripts from a long read RNA-seq data set(cf: https://github.com/gpertea/stringtie):

```
stringtie -L -G Homo_sapiens.GRCh38.97.gtf -o assembly.gtf long_reads.bam
```

Several filtering steps can be considered to improve the assembly of (Intron-Retaining) transcripts:

1. The extraction of full-length long reads to mitigate the number of false positive shorter transcripts
2. The extraction of long read overlapping an intron.

Extraction of full-length reads

First, we specify the (custom and reference) data we will need to identify alignments that (may) correspond to full-length reads:

```
# Long read splice-aware alignments on a reference genome (eg: output by Minimap2):
alignments <- system.file("extdata", "MCF10A_chr2.dont.bam", package = "FakIR")

# Reference transcriptome in a gtf file:
gtf <- system.file("extdata", "Homo_sapiens.GRCh38.97.chr2.gtf", package = "FakIR")

# TSS called from CAGE data carried on a comparable cell line (here MCF7), provided by ENCODE:
cageFile <-  system.file("extdata", "GSM849364_hg38_wgEncodeRikenCageMcf7CellPapTssGencV7.chr2.gtf",
                         package = "FakIR")
```

```
# Name for the output bam file, that will contain only full-length reads alignments:
saveFile <- "fl_long_reads.bam"  # Or replace with a convenient path
```

```
extractFullLengthReads(alignments, gtf, cageFile, saveFile, verbose = TRUE)
```

Extraction of IR-transcripts

So as to avoid ignoring rare IR-transcripts, you may first extract potential intron-retaining transcripts (by analysing genome alignments) and then perform reconstruction from those reads only with specific parameters (eg: no minimum isoform fraction).

```
gtf <- system.file("extdata", "Homo_sapiens.GRCh38.97.chr2.gtf", package = "FakIR")

bamFile <- system.file("extdata", "MCF10A_chr2.dont.bam", package = "FakIR")

saveFile <- "ir_long_reads.bam"
# Or replace with a convenient path
```

```
extractIRreads(alignments, gtf = gtf, saveFile = saveFile, intronMinoverlap = 5, keepSecondaryAlignment
```

This function will first identify intron intervals from the input reference transcriptome (gtf) by calling the function [‘FakIR::createIntronGR()’] and will then select all long reads whose spliced alignment overlaps an intron.
Here, as a bam file name is given in *saveFile*, these alignments will be saved in a (new) bam file.

You may also perform you own identification or subsetting of intron intervals and specify them in a GRanges object. The following lines of code are equivalent to the one above:

```
# Define intronic intervals from a reference transcriptome
# Or feed intronGR with custom intron interval in a GRanges
intronGR <- createIntronGR( gtf )

extractIRreads(alignments, intronGR = intronGR, saveFile, intronMinoverlap = 5, keepSecondaryAlignments
```

You can then assemble only intron-overlapping long reads with *StringTie2* (for example) using:

```
stringtie -L -G Homo_sapiens.GRCh38.97.gtf -f 0 -o assembly.gtf ir_long_reads.bam
```

IR events interpretation: prediction of IR-NMD coupling

We present several *in-silico* approaches to identify genes or transcripts that may be regulated through IR-NMD degradation.
They can be split into two families, depending on whether you know the full structure and sequence of expressed IR-isoforms (eg: assembled from full-length long reads), or only the genomic interval of retained introns (eg: inferred from short reads or truncated long reads).

1. **From intron genomic intervals**: Detect stop codons in retained introns, Assume hypothetical IR-transcripts using a transcriptome annotation

2. **From full-length IR-isoforms**:
   Prediction of Premature Termination Codons in assembled transcripts

Detect stop codons in retained introns

We will need IR events (eg: selected as above) and reference genome sequences to extract intron sequences:

```
BSgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
```

```
ir_events <- countStopCodons(ir_events, BSgenome, stopCodons = c("TAA", "TAG", "TGA"))
```

This will add three columns **frame1**, **frame2**, **frame3** to the *data.frame* of selected IR events, giving the number of stop codons found in the intron sequence according to the three possible "reading frames":

```
ir_events %>% dplyr::select(seqnames, start, end, strand , frame1, frame2, frame3) %>% head(15)
#>    seqnames    start      end strand frame1 frame2 frame3
#> 1         2 11783922 11784885      +     38     15     14
#> 2         2 19898035 19901029      -    183     87     69
#> 3         2 23783028 23786026      -    195     91     64
#> 4         2 24768135 24768220      +      4      2      3
#> 5         2 24820451 24820723      -     10      6      3
#> 6         2 26793893 26800928      +    316    136    100
#> 7         2 27133337 27133531      -      8      6      1
#> 8         2 27133721 27134286      -     21     12      8
#> 9         2 27316186 27317077      -     24     13      6
#> 10        2 27378371 27378466      -      4      2      1
#> 11        2 27454162 27454353      -      6      2      1
#> 12        2 43225752 43226264      -      4      2      1
#> 13        2 43291768 43292103      -     18      7      3
#> 14        2 61185618 61186445      +     48     27     17
#> 15        2 72887736 72888313      +     19      9      7
```

The conclusions that can be drawn from this simple screen are limited.
Nonetheless, it may allow a quick selection of interesting targets:

1. **NMD-targets**. Introns whose retention would insert stop codons in all three reading frame (if part of the CDS and if the transcript were exported to the cytoplasm) are sign of IR-NMD degradation. Here, they represent most of the IR events:

```
ir_events %>% filter(frame1*frame2*frame3 > 0) %>% nrow()
#> [1] 91

ir_events %>% filter(frame1*frame2*frame3 > 0) %>%
              dplyr::select(seqnames, start, end, start, gene_name, frame1, frame2, frame3) %>%
              head(15)
#>    seqnames    start      end gene_name frame1 frame2 frame3
#> 1         2 11783922 11784885     LPIN1     38     15     14
#> 2         2 19898035 19901029     TTC32    183     87     69
#> 3         2 23783028 23786026    ATAD2B    195     91     64
#> 4         2 24768135 24768220     NCOA1      4      2      3
#> 5         2 24820451 24820723     ADCY3     10      6      3
#> 6         2 26793893 26800928     CENPA    316    136    100
#> 7         2 27133337 27133531      PREB      8      6      1
#> 8          2 27133721 27134286      PREB     21      12       8
#> 9          2 27316186 27317077     MPV17     24      13       6
#> 10         2 27378371 27378466    ZNF513      4       2       1
#> 11         2 27454162 27454353    IFT172      6       2       1
#> 12         2 43225752 43226264   ZFP36L2      4       2       1
#> 13         2 43291768 43292103     THADA     18       7       3
#> 14         2 61185618 61186445    AHSA2P     48      27      17
#> 15         2 72887736 72888313       SPR     19       9       7
```

2. **Translated IR**. On the opposite, introns whose retention would include no PTC into the transcript (whatever the rest of its exonic structure) may be indicate that IR-transcripts are actively translated (if exported to the cytoplasm):

```r
ir_events %>% filter(frame1+frame2+frame3 == 0)
```

There is no such case in out toy data set.

3. **Ambiguous cases**. Other cases are ambiguous and could only be identified with the knowledge of the full transcript structure:

```r
ir_events %>% filter(frame1+frame2+frame3 != 0 & frame1*frame2*frame3 == 0) %>%
              dplyr::select(seqnames, start, end, start, gene_id, frame1, frame2, frame3)
#>    seqnames     start       end         gene_id frame1 frame2 frame3
#> 39        2 130370623 130370704 ENSG00000072135      1      0      0
#> 90        2 240596521 240596680 ENSG00000142330      2      0      1
```

Assume hypothetical IR-transcripts using a transcriptome annotation

Once isolated IR events have been selected without knowing the complete transcript structure (eg: from short read data IR events detection: from Second Generation RNA-seq data, or truncated long read IR events detection: from Third Generation RNA-seq data), one can create a "hypothetical" annotation for IR-transcripts by taking advantage of annotated transcripts that could contain them.

Thus, first, we will need to specify suitable reference transcripts (here downloaded from *Ensembl*):

```r
gtf <- system.file("extdata", "Homo_sapiens.GRCh38.97.chr2.gtf", package = "FakIR")
```

Then, we identify reference transcripts that are compatible with the detected IR events and create corresponding IR-transcripts assembly:

```r
# Where to save IR-transcripts assembly:
outfile <- paste(system.file("extdata",  package = "FakIR"), "MCF10A.hypothetical_ir_transcripts.gtf", 

findCompatibleTranscripts(ir_events, gtf, transcript_biotypes = "protein_coding", outfile = outfile)
```

Prediction of Premature Termination Codons in assembled transcripts

There are several approaches to infer whether a given transcript harbours a Premature Termination Codon (PTC).
They are more or less accurate, and depend on the amount of data available to you.

1. Translation Initiation Sites are not easy to determine

2. A single transcript may have several open reading frames

We suggest a simple procedure that makes use of functions implemented in the Bioconductor package *IsoformSwitchAnalyzer*.

IsoformSwitchAnalyzeR can be installed via Bioconductor by:

```
install.packages("BiocManager")
BiocManager::install()

BiocManager::install("IsoformSwitchAnalyzeR")
```

ORFs and PTC prediction from complete transcript sequences

Once installed, load *IsoformSwitchAnalyzeR* into your current R session:

```
suppressPackageStartupMessages( require(IsoformSwitchAnalyzeR) )
```

In this example, we will import our own assembly of full-length MCF10A, provided as external data.
You may replace *assembly.gtf* with a path to your own assembly:

```
assembly.gtf <- paste(system.file("extdata",  package = "FakIR"), "MCF10A_chr2.dont_assembly.gtf", sep
```

Now, import the transcriptome assembly as *switchAnalyzeRlist* object:

```
mySwitchList <- IsoformSwitchAnalyzeR::importGTF(pathToGTF = assembly.gtf)
#> importing GTF (this may take a while)
#> converting GTF to switchAnalyzeRlist
#> Warning in IsoformSwitchAnalyzeR::importGTF(pathToGTF = assembly.gtf): No CDS
#> was found in the GTF file. Please make sure the GTF file have the CDS "feature"
#> annotation. Adding NAs instead
```

Additionaly, we will need to specify a correct reference genome from which to retrieve transcripts sequences.
Many reference genomes are made available by Bioconductor.

In our example, we will use the *BSgenome* object containing the UCSC reference genome *Homo_sapiens* hg38:

```
BSgenome <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
```

There exist several approaches for predicting Open Reading Frames. It is far from being straightforward.
You will find more details and discussion in the *IsoformSwitchAnalyzer* package:

```
vignette( "IsoformSwitchAnalyzeR" )
#> starting httpd help server ... done
```

As for our own MCF10A assembly we do not have any external information about the CDS of transcripts, we will to infer them from the sequences only (which is probably not the best method).

```
orfMethod <- "mostUpstream"

mySwitchList <- analyzeORF(switchAnalyzeRlist = mySwitchList,
                           genomeObject = BSgenome,
                           orfMethod = orfMethod,
                               quiet = TRUE)
```

Typical output of the analysis of Open Reading Frames by *IsoformSwitchAnalyzeR* can be found in the *data.frame*:

```
mySwitchList$orfAnalysis
```

In particular:

```
mySwitchList$orfAnalysis %>% dplyr::select(isoform_id,
                                           orfStartGenomic, orfEndGenomic,
                                           stopDistanceToLastJunction, PTC) %>%
                               head(15)
#>         isoform_id orfStartGenomic orfEndGenomic stopDistanceToLastJunction
#> 1  TCONS_00035327          264965        277301                        -75
#> 2  TCONS_00035328          264965        276988                         97
#> 3  TCONS_00035329          264965        277301                        -75
#> 4  TCONS_00035330          264965        272255                        218
#> 5  TCONS_00035331         3387624       3388742                      -1123
#> 6  TCONS_00035332         3387624       3479458                       -240
#> 7  TCONS_00035333         3387624       3479458                       -240
#> 8  TCONS_00035334         3387624       3479458                       -240
#> 9  TCONS_00035335         3575610       3580879                        -75
#> 10 TCONS_00035336         3575610       3575849                        -33
#> 11 TCONS_00035337         3575610       3580314                       -205
#> 12 TCONS_00035338         3575486       3575884                        364
#> 13 TCONS_00035339         3575610       3580879                        -75
#> 14 TCONS_00035340         3575610       3580879                        -75
#> 15 TCONS_00035341         3575486       3575884                          4
#>      PTC
#> 1  FALSE
#> 2   TRUE
#> 3  FALSE
#> 4   TRUE
#> 5  FALSE
#> 6  FALSE
#> 7  FALSE
#> 8  FALSE
#> 9  FALSE
#> 10 FALSE
#> 11 FALSE
#> 12  TRUE
#> 13 FALSE
#> 14 FALSE
#> 15 FALSE
```

We can now get a contigency table of assembled transcripts with or without a PTC:

```
table( mySwitchList$orfAnalysis$PTC )
#>
#> FALSE  TRUE
#>  3155  1487
```

References

1. *Middleton et al.* ***IRFinder:assessing the impact of intron retention on mammalian gene expression*** *Genome Biology. (2017)*
2. *Broseus and Ritchie* ***Challenges in detecting and quantifying intron retention from NGS data.*** *CSBJ. (2020)*
3. *Broseus et al.* ***TALC: Transcription Aware Long Read Correction.*** *BioRxiv. (2020)*
4. *Li.* ***Minimap2: pairwise alignment for nucleotide sequences.*** *Bioinformatics (2018)*
5. *Kovaka et al.* ***Transcriptome assembly from long-read RNA-seq alignments with StringTie2*** *Genome Biology (2019)*
6. *Vacik et al.* ***Alternative intronic promoters in development and disease.*** *Protoplasma (2017)*
7. *Vitting-Seerup et al.* ***IsoformSwitchAnalyzeR: Analysis of changes in genome-wide patterns of alternative splicing and its functional consequences.*** *Bioinformatics (2019)*
8. *Anders et al.* ***Detecting Differential Usage of Exons from RNA-seq Data.*** *Genome Research (2008)*