Using branchpointer for annotation of intronic human splicing branchpoints

Beth Signal

May 19, 2021

Contents

1 Introduction . . . 1
2 Preparation . . . 2
2.1 Download genome annotations . . . 2
2.2 Read in exon annotations . . . 3
3 Branchpoint annotations in intronic regions . . . 3
3.1 Read query and calculate location attributes . . . 3
3.1.1 Using bedtools with a genome .fa file . . . 5
3.2 Predict branchpoint probabilities . . . 5
4 Effects of SNPs on branchpoint annotations . . . 7
4.1 Read query and calculate location attributes . . . 7
4.2 Predict branchpoint probabilities . . . 8
5 Performance . . . 11
5.1 Example run times . . . 11
6 Session info . . . 12

1 Introduction

The spliceosome mediates the formation of an intron lariat though inter-action between the 5' splice site and branchpoint (Will and Luhrmann, 2011). A subsequent reaction at the 3' SS then removes the intron lariat, producing a spliced RNA product. Mapping of branchpoints generally requires sequencing of the intron lariat following cDNA synthesis (Gao et al., 2008; Taggart et al., 2012). However, intron lariats are rapidly de-branched and degraded, and much less abundant than the spliced RNA, resulting in the poor recovery of elements using sequencing. Most recently, Mercer et al. (2015) employed a targeted sequencing approach to identify 59,359 branchpoints in 17.4% of annotated human gene introns. Whilst this constituted the largest annotation to date, the identification of branchpoints was restricted to highly-expressed genes with sufficient sequence coverage.

To address this limitation, and expand branchpoint annotations across the human genome, we have developed a machine-learning based model of branchpoints trained with this empirical annotation (Signal et al., 2016). This model requires only genomic sequence and exon annotations, and exhibits no discernible bias to gene type or expression, and can be applied using the R package, branchpointer. Aberrant splicing is known to lead to many human diseases (Singh and Cooper, 2012), however prediction of intronic variant effects have been typically limited to splice site alterations (McLaren et al., 2016; Wang et al., 2010). Therefore, in addition to annotation of branchpoints, branchpointer allows users to assess the effects of intronic mutations on branchpoint architecture.

Gao,K. et al. (2008) Human branch point consensus sequence is yUnAy. Nucleic Acids Res., 36, 2257–67.

McLaren,W. et al. (2016) The Ensembl Variant Effect Predictor. Genome Biol., 17, 122.

Mercer,T.R. et al. (2015) Genome-wide discovery of human splicing branchpoints. Genome Res., 25, 290–303.

Signal,B. et al. (2016) Machine-learning annotation of human splicing branchpoints. BioRxiv. doi: 10.1101/094003.

Singh,R.K. and Cooper,T.A. (2012) Pre-mRNA splicing in disease and therapeutics. Trends Mol. Med., 18, 472–482.

Taggart,A.J. et al. (2012) Large-scale mapping of branchpoints in human pre-mRNA transcripts in vivo. Nat. Struct. Mol. Biol., 19, 719–21.

Wang,K. et al. (2010) ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data. Nucleic Acids Res., 38, e164.

Will,C.L. and Luhrmann,R. (2011) Spliceosome structure and function. Cold Spring Harb. Perspect. Biol., 3, a003707.

2 Preparation

2.1 Download genome annotations

Branchpointer requires a genome annotation GTF file for branchpoint annotation. We will be using the GENCODE annotation (http://www.gencodegenes.org/releases/current.html) as an example, although others and custom annotations can be used.

Create or move to a working directory where these files can be stored. Note that GTFs can be large files (over 1GB) when uncompressed.

```
wget ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/release_26/gencode.v26.annotation.gtf.gz

gunzip gencode.v26.annotation.gtf.gz
```

branchpointer requires either a BSGenome object, or a genome .fa file for sequence retrieval. The genome must correspond to the gtf used – i.e. gencodev26 uses GRCh38.

load a *BSGenome*:

```
library(BSgenome.Hsapiens.UCSC.hg38)
g <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38
```

or download .fa:

```
wget ftp://ftp.sanger.ac.uk/pub/gencode/Gencode_human/release_26/GRCh38.primary_assembly.genome.fa.gz
GRCh38.primary_assembly.genome.fa.gz
```

2.2 Read in exon annotations

Start by loading branchpointer.

```
library(branchpointer)
```

gtfToExons will generate an exon annotation GRanges object from a gtf. To load in the gtf downloaded from the preparation section:

```
exons <- gtfToExons("gencode.v26.annotation.gtf")
```

We will load in a small gtf from the package data for the following examples.

```
smallExons <- system.file("extdata","gencode.v26.annotation.small.gtf",
                          package = "branchpointer")
exons <- gtfToExons(smallExons)
```

3 Branchpoint annotations in intronic regions

3.1 Read query and calculate location attributes

Query regions must contain a branchpoint window - that is the region located at -18 to -44 from the 3' splice site. Each region given will be treated as only one query, and associated with the closest 3' exon. To cover multiple 3'exons, please provide branchpointer with separate region queries. For known regions, queries can be supplied as a table with the following formatting:

```
queryIntronFile <- system.file("extdata","intron_example.txt",
                               package = "branchpointer")
queryIntronTable <- read.delim(queryIntronFile)
head(queryIntronTable)
```

```
##             id chromosome    start      end strand
## 1 BRCA1_intron      chr17 43045820 43045846      -
## 2 BRCA2_intron      chr13 32346783 32346809      +
```

Query files can be read into branchpointer using readQueryFile(), and location information retrieved for these sequences:

```
queryIntron <- readQueryFile(queryIntronFile,
                             queryType = "region",
                             exons = exons)
head(queryIntron)
```

```
## GRanges object with 2 ranges and 6 metadata columns:
##       seqnames            ranges strand |           id to_3prime to_5prime
##          <Rle>         <IRanges>  <Rle> |  <character> <numeric> <numeric>
##   [1]    chr17 43045820-43045846      - | BRCA1_intron        18      1823
##   [2]    chr13 32346783-32346809      + | BRCA2_intron        18      2156
```

```
##       same_gene       exon_3prime       exon_5prime
##       <logical>       <character>       <character>
##   [1]      TRUE ENSE00001814242.1 ENSE00003687053.1
##   [2]      TRUE ENSE00000939171.1 ENSE00003753873.1
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

Alternatively, to generate branchpoint window region queries by from the gtf annotation, the exon object can be used:

Note that when searching for genes, transcripts, or exons, the ids used must be in the same format as in the annotation file (i.e. ENSG00000XXXXXX.X, ENST00000XXXXXX.X, ENSE00000XXXXXX.X). If you are unsure of an id, aliases can typically be found through ensembl (ensembl.org), or through a biomaRt query.

```
queryIntronFromGTF <- makeBranchpointWindowForExons("ENSE00000939171.1",
                                                    idType = "exon_id",
                                                    exons = exons)
head(queryIntronFromGTF)
```

```
## GRanges object with 1 range and 12 metadata columns:
##       seqnames            ranges strand |            gene_id      gene_type
##          <Rle>         <IRanges>  <Rle> |        <character>    <character>
##   [1]    chr13 32346783-32346809      + | ENSG00000139618.14 protein_coding
##          transcript_id transcript_type          exon_id exon_number
##            <character>     <character>       <character> <character>
##   [1] ENST00000380152.7  protein_coding ENSE00000939171.1          13
##       to_3prime to_5prime same_gene       exon_3prime       exon_5prime
##       <numeric> <integer> <logical>       <character>       <character>
##   [1]        18      2156      TRUE ENSE00000939171.1 ENSE00000939169.1
##                       id
##             <character>
##   [1] ENSE00000939171.1
##   -------
##   seqinfo: 4 sequences from an unspecified genome; no seqlengths
```

```
# for multiple ids:
queryIntronFromGTF <- makeBranchpointWindowForExons(c("ENSE00000939171.1",
                                                      "ENSE00001814242.1"),
                                                    idType = "exon_id",
                                                    exons = exons)
head(queryIntronFromGTF)
```

```
## GRanges object with 2 ranges and 12 metadata columns:
##       seqnames            ranges strand |            gene_id      gene_type
##          <Rle>         <IRanges>  <Rle> |        <character>    <character>
##   [1]    chr13 32346783-32346809      + | ENSG00000139618.14 protein_coding
##   [2]    chr17 43045820-43045846      - | ENSG00000012048.20 protein_coding
##          transcript_id transcript_type          exon_id exon_number
##            <character>     <character>       <character> <character>
##   [1] ENST00000380152.7  protein_coding ENSE00000939171.1          13
##   [2] ENST00000357654.7  protein_coding ENSE00001814242.1          23
##       to_3prime to_5prime same_gene       exon_3prime       exon_5prime
```

```
##       <numeric> <integer> <logical>       <character>       <character>
##   [1]        18      2156      TRUE ENSE00000939171.1 ENSE00000939169.1
##   [2]        18      1823      TRUE ENSE00001814242.1 ENSE00003687053.1
##                       id
##             <character>
##   [1] ENSE00000939171.1
##   [2] ENSE00001814242.1
##   -------
##   seqinfo: 4 sequences from an unspecified genome; no seqlengths
```

3.1.1 Using bedtools with a genome .fa file

During the prediction step, if a BSGenome object is not specified, sequences covering each site +/- 250 nt can be retrieved using bedtools. The absolute location of the bedtools binary must be provided for calls from within R. To find the location of your installed bedtools binary, using the command line type:

```
which bedtools
```

If chromosome names in the .fa genome file do not match those in the query (i.e chr1 in query, 1 in .fa), the argument rm_chr should be set to FALSE.

3.2 Predict branchpoint probabilities

Branchpoint probability scores can now be evaluated using the branchpointer model. This will generate a new GRanges object with a row for each site (of 27) in branchpoint window regions. If a SNP query type is provided (See next section), this will also perform an in silico mutation of the sequence.

We recommend use of the cut-off probability 0.52 to distinguish branchpoints and non-branchpoint sites. U2 binding energy can be used as a measurement of branchpoint strength when the probability score is above the cut-off.

All features required for the model to predict branchpoint probability are contained within the output object, along with the score and U2 binding energy.

```
branchpointPredictionsIntron <- predictBranchpoints(queryIntron,
                                                    queryType = "region",
                                                    BSgenome = g)
head(branchpointPredictionsIntron)
```

```
## GRanges object with 6 ranges and 31 metadata columns:
##       seqnames            ranges strand |           id to_3prime to_5prime
##          <Rle>         <IRanges>  <Rle> |  <character> <numeric> <numeric>
##   [1]    chr17 43045820-43045846      - | BRCA1_intron        18      1823
##   [2]    chr13 32346783-32346809      + | BRCA2_intron        18      2156
##   [3]    chr17 43045820-43045846      - | BRCA1_intron        18      1823
##   [4]    chr13 32346783-32346809      + | BRCA2_intron        18      2156
##   [5]    chr17 43045820-43045846      - | BRCA1_intron        18      1823
##   [6]    chr13 32346783-32346809      + | BRCA2_intron        18      2156
##       same_gene       exon_3prime       exon_5prime                   seq
##       <logical>       <character>       <character>           <character>
##   [1]      TRUE ENSE00001814242.1 ENSE00003687053.1 TCCAGGAGAATGAATTGACA..
```

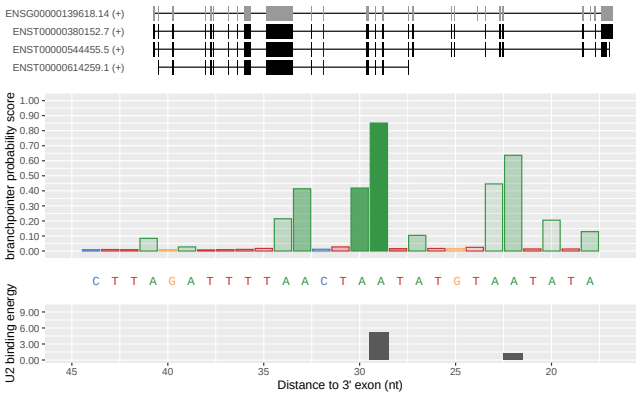
```
##   [2]      TRUE ENSE00000939171.1 ENSE00003753873.1 TATTCTCTTAGATTTTAACT..
##   [3]      TRUE ENSE00001814242.1 ENSE00003687053.1 TCCAGGAGAATGAATTGACA..
##   [4]      TRUE ENSE00000939171.1 ENSE00003753873.1 TATTCTCTTAGATTTTAACT..
##   [5]      TRUE ENSE00001814242.1 ENSE00003687053.1 TCCAGGAGAATGAATTGACA..
##   [6]      TRUE ENSE00000939171.1 ENSE00003753873.1 TATTCTCTTAGATTTTAACT..
##           status to_3prime_point to_5prime_point test_site seq_pos0
##      <character>       <integer>       <numeric> <numeric> <factor>
##   [1]         REF              44            1797  43045846        A
##   [2]         REF              44            2130  32346783        C
##   [3]         REF              43            1798  43045845        G
##   [4]         REF              43            2131  32346784        T
##   [5]         REF              42            1799  43045844        A
##   [6]         REF              42            2132  32346785        T
##       seq_pos1 seq_pos2 seq_pos3 seq_pos4 seq_pos5 seq_neg1 seq_neg2
##       <factor> <factor> <factor> <factor> <factor> <factor> <factor>
##   [1]        G        A        A        T        G        G        G
##   [2]        T        T        A        G        A        T        C
##   [3]        A        A        T        G        A        A        G
##   [4]        T        A        G        A        T        C        T
##   [5]        A        T        G        A        A        G        A
##   [6]        A        G        A        T        T        T        C
##       seq_neg3 seq_neg4 seq_neg5 canon_hit1 canon_hit2 canon_hit3 canon_hit4
##       <factor> <factor> <factor>  <numeric>  <numeric>  <numeric>  <numeric>
##   [1]        A        C        C         42         53         62         81
##   [2]        T        T        A          3         42         54         61
##   [3]        G        A        C         41         52         61         80
##   [4]        C        T        T          2         41         53         60
##   [5]        G        G        A         40         51         60         79
##   [6]        T        C        T          1         40         52         59
##       canon_hit5 ppt_start ppt_run_length branchpoint_prob U2_binding_energy
##        <numeric> <numeric>      <numeric>        <numeric>         <numeric>
##   [1]         83        19             24       0.03166614               0.5
##   [2]         87        16              7       0.00860353               0.5
##   [3]         82        18             24       0.00868235               0.1
##   [4]         86        15              7       0.00991773               1.2
##   [5]         81        17             24       0.02976173               1.4
##   [6]         85        14              7       0.00904757               1.2
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

The window scores can be plotted using plotBranchpointWindow(), with optional plots for gene and isoform structure. The main panel displays the probability scores of each site within the branchpoint window. The opacity of the bars is representative of relative U2 binding energy (darker = stronger), and the lower panel shows U2 binding energy for all sites above the provided probability cutoff.

BRCA2 intron (ENSE00000939169.1 - ENSE00000939171.1):

```
plotBranchpointWindow(queryIntron$id[2],
                      branchpointPredictionsIntron,
                      probabilityCutoff = 0.52,
                      plotMutated = FALSE,
```

```
                        plotStructure = TRUE,
                        exons = exons)
```



4 Effects of SNPs on branchpoint annotations

In addition to locating branchpoints in intronic windows, branchpointer can be used to evaluate the local effects of SNPs on branchpoints. The general workflow is the same as for annotation of intronic windows, however queryType="SNP" must be used.

4.1 Read query and calculate location attributes

Query SNPs should be located nearby a branchpoint window to have any potential effects on branchpoint architecture SNP queries can be supplied as a table formatted as follows:

```
querySNPFile <- system.file("extdata","SNP_example.txt",
                            package = "branchpointer")
querySNPTable <- read.delim(querySNPFile)
head(querySNPTable)
```

```
##            id chromosome chrom_start strand ref_allele alt_allele
## 1 rs786205083       chr2    71590178      +          A          G
## 2 rs587776767      chr11     2165787      -          A          T
```

When reading in exceptionally large numbers of SNPs, it is recommended to set filter = TRUE. This adds a pre-filtering step which removes any SNPs not located in an intron or 50nt from a 3' exon.

Each SNP will be associated with the closest 3' exon. If SNPs are distal from branchpoint windows, the max_dist argument will remove any greater than the specified distance. Filtering prior to exon associations can therefore speed up processing in instances where it is unknown if the majority of SNPs fall nearby branchpoint windows.

```
querySNP <- readQueryFile(querySNPFile,
                          queryType = "SNP",
                          exons = exons,
                          filter = TRUE)
head(querySNP)
```

```
## GRanges object with 2 ranges and 8 metadata columns:
##       seqnames    ranges strand |              id  ref_allele  alt_allele
##          <Rle> <IRanges>  <Rle> |     <character> <character> <character>
##   [1]     chr2  71590178      + | rs786205083_pos           A           G
##   [2]    chr11   2165787      - | rs587776767_neg           A           T
##       to_3prime to_5prime same_gene       exon_3prime       exon_5prime
##       <numeric> <numeric> <logical>       <character>       <character>
##   [1]        33       492      TRUE ENSE00003642866.1 ENSE00003663865.1
##   [2]        24        66      TRUE ENSE00003550033.1 ENSE00001878270.1
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

Queries can be provided as stranded or unstranded. In the case of unstranded queries, any value except "+" or "-" will cause branchpointer to run on both strands.

Alternatively, appropriate attributes can be pulled from biomart when a list of refsnp ids is provided:

```
library(biomaRt)
mart <- useMart("ENSEMBL_MART_SNP", dataset="hsapiens_snp",host="www.ensembl.org")
querySNP <- makeBranchpointWindowForSNP(c("rs587776767","rs786205083"),
                                        mart.snp = mart,
                                        exons = exons,
                                        filter  = FALSE)
head(querySNP)
```

```
## GRanges object with 2 ranges and 8 metadata columns:
##       seqnames    ranges strand |              id  ref_allele  alt_allele
##          <Rle> <IRanges>  <Rle> |     <character> <character> <character>
##   [1]     chr2  71590178      + | rs786205083_pos           A           G
##   [2]    chr11   2165787      - | rs587776767_neg           A           T
##       to_3prime to_5prime same_gene       exon_3prime       exon_5prime
##       <numeric> <numeric> <logical>       <character>       <character>
##   [1]        33       492      TRUE ENSE00003642866.1 ENSE00003663865.1
##   [2]        24        66      TRUE ENSE00003550033.1 ENSE00001878270.1
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

By default, all SNPs retrieved will be unstranded, and hence further processing will be done on both strands

4.2 Predict branchpoint probabilities

Using a .fa and bedtools:

```
branchpointPredictionsSNP <- predictBranchpoints(querySNP,
                                        queryType = "SNP",
                                        genome = "GRCh38.primary_assembly.genome.fa",
                                        bedtoolsLocation="/Apps/bedtools2/bin/bedtools")
```

Using a BSgenome:

```
#for query SNPs
branchpointPredictionsSNP <- predictBranchpoints(querySNP,
                                      queryType = "SNP",
                                      BSgenome = g)
head(branchpointPredictionsSNP)

## GRanges object with 6 ranges and 33 metadata columns:
##       seqnames    ranges strand |             id  ref_allele  alt_allele
##          <Rle> <IRanges>  <Rle> |     <character> <character> <character>
##   [1]     chr2  71590178      + | rs786205083_pos           A           G
##   [2]    chr11   2165787      - | rs587776767_neg           A           T
##   [3]     chr2  71590178      + | rs786205083_pos           A           G
##   [4]    chr11   2165787      - | rs587776767_neg           A           T
##   [5]     chr2  71590178      + | rs786205083_pos           A           G
##   [6]    chr11   2165787      - | rs587776767_neg           A           T
##       to_3prime to_5prime same_gene       exon_3prime       exon_5prime
##       <numeric> <numeric> <logical>       <character>       <character>
##   [1]        33       492      TRUE ENSE00003642866.1 ENSE00003663865.1
##   [2]        24        66      TRUE ENSE00003550033.1 ENSE00001878270.1
##   [3]        33       492      TRUE ENSE00003642866.1 ENSE00003663865.1
##   [4]        24        66      TRUE ENSE00003550033.1 ENSE00001878270.1
##   [5]        33       492      TRUE ENSE00003642866.1 ENSE00003663865.1
##   [6]        24        66      TRUE ENSE00003550033.1 ENSE00001878270.1
##                          seq      status to_3prime_point to_5prime_point
##                  <character> <character>       <integer>       <numeric>
##   [1] AACCACTCCAGCCACTCACT..         REF              44             481
##   [2] CCGGTGGGCGGCAGCTGTCT..         REF              44              46
##   [3] AACCACTCCAGCCACTCACT..         REF              43             482
##   [4] CCGGTGGGCGGCAGCTGTCT..         REF              43              47
##   [5] AACCACTCCAGCCACTCACT..         REF              42             483
##   [6] CCGGTGGGCGGCAGCTGTCT..         REF              42              48
##       test_site seq_pos0 seq_pos1 seq_pos2 seq_pos3 seq_pos4 seq_pos5
##       <numeric> <factor> <factor> <factor> <factor> <factor> <factor>
##   [1]  71590167        T        C        C        A        G        C
##   [2]   2165807        G        G        C        G        G        C
##   [3]  71590168        C        C        A        G        C        C
##   [4]   2165806        G        C        G        G        C        A
##   [5]  71590169        C        A        G        C        C        A
##   [6]   2165805        C        G        G        C        A        G
##       seq_neg1 seq_neg2 seq_neg3 seq_neg4 seq_neg5 canon_hit1 canon_hit2
##       <factor> <factor> <factor> <factor> <factor>  <numeric>  <numeric>
##   [1]        C        A        C        C        A          3         42
##   [2]        G        T        G        G        C          6         42
##   [3]        T        C        A        C        C          2         41
##   [4]        G        G        T        G        G          5         41
##   [5]        C        T        C        A        C          1         40
##   [6]        G        G        G        T        G          4         40
##       canon_hit3 canon_hit4 canon_hit5 ppt_start ppt_run_length
##        <numeric>  <numeric>  <numeric> <numeric>      <numeric>
##   [1]         80        107        124        19             19
##   [2]         63         78         81        18              7
```

```
##   [3]         79        106        123        18             19
##   [4]         62         77         80        18              7
##   [5]         78        105        122        17             19
##   [6]         61         76         79        18              7
##       branchpoint_prob U2_binding_energy
##              <numeric>         <numeric>
##   [1]       0.00779342               1.4
##   [2]       0.00761277               0.0
##   [3]       0.01090087               1.7
##   [4]       0.00815408               0.2
##   [5]       0.01019025               2.0
##   [6]       0.00868430               0.8
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

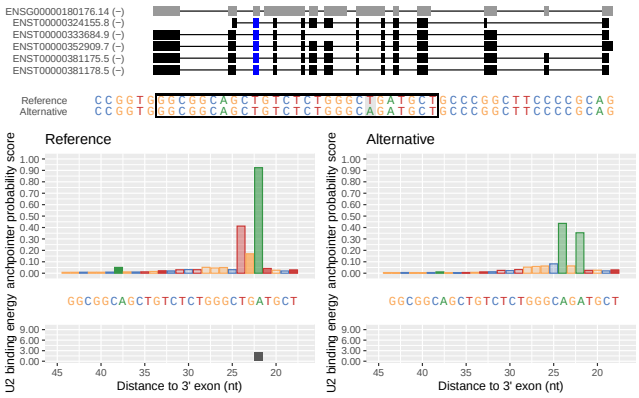
```
#to summarise effects:
querySNPSummary <- predictionsToSummary(querySNP,branchpointPredictionsSNP)
head(querySNPSummary)
```

```
## GRanges object with 2 ranges and 18 metadata columns:
##       seqnames    ranges strand |              id  ref_allele  alt_allele
##          <Rle> <IRanges>  <Rle> |     <character> <character> <character>
##   [1]     chr2  71590178      + | rs786205083_pos           A           G
##   [2]    chr11   2165787      - | rs587776767_neg           A           T
##       to_3prime to_5prime same_gene       exon_3prime       exon_5prime
##       <numeric> <numeric> <logical>       <character>       <character>
##   [1]        33       492      TRUE ENSE00003642866.1 ENSE00003663865.1
##   [2]        24        66      TRUE ENSE00003550033.1 ENSE00001878270.1
##       BP_num_REF BP_num_ALT deleted_n created_n dist_to_BP_REF
##        <numeric>  <numeric> <numeric> <numeric>      <numeric>
##   [1]          2          1         1         0              0
##   [2]          1          0         1         0              2
##       dist_to_BP_ALT max_prob_REF max_prob_ALT max_U2_REF max_U2_ALT
##            <numeric>    <numeric>    <numeric>  <numeric>  <numeric>
##   [1]              8     0.922265     0.561610        2.0        0.5
##   [2]             NA     0.923688     0.435892        2.5         NA
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

The window scores in the reference and alternative sequences can be visualised using plotBranchpointWindow().

rs587776767 in TH intron

```
plotBranchpointWindow(querySNP$id[2],
                      branchpointPredictionsSNP,
                      probabilityCutoff = 0.52,
                      plotMutated = TRUE,
                      plotStructure = TRUE,
                      exons = exons)
```



5 Performance

Branchpointer is vectorised where possible to decrease run times.

When reading in SNP queries, a prefiltering step can be applied by setting filter=TRUE. This step adds less than 10 seconds to readQueryFile(), and can reduce run times when the locations of SNPs are unknown [Figure 1].

Figure 1: Time taken for readQueryFile() on SNP query files

predictBranchpoints() is the rate limiting step, taking 55 seconds per 100 region queries. This can be lowered using parallelisation by setting useParallel=TRUE and specifying the number of cores [Figure 2].

Figure 2: Time taken for predictBranchpoints() on region queries

5.1 Example run times

```
# Step times for annotating branchpoints in introns:
gtfToExons()
# user  system elapsed
# 41.385   3.848  47.096

# Set 1. 294 lincRNA introns on chr22:
makeBranchpointWindowForExons()
# user  system elapsed
# 0.196   0.024   0.226
predictBranchpoints()
# user  system elapsed
# 208.934   4.157 225.849

# Set 2. 3693 protein coding exons on chr22:
makeBranchpointWindowForExons()
```

```
# user  system elapsed
# 0.245   0.013   0.261
predictBranchpoints()
# user   system  elapsed
# 2332.519   38.266 2482.032

# Step times for annotating branchpoints with SNPs:
# 29899 GWAS SNPS
readQueryFile(filter = TRUE)
# user  system elapsed
# 5.997   1.608   7.773
readQueryFile(filter = FALSE)
# user  system elapsed
# 1.744   0.427   2.339

# 298 filtered SNPS
predictBranchpoints()
# user  system elapsed
# 172.495   2.485 181.876

predictionsToSummary()
# user  system elapsed
# 0.057   0.003   0.061
```

Example scripts used to test times are found in inst/scripts, and were run on a 2.4GHz Macbook Pro with 8GB RAM

6 Session info

```
sessionInfo()
```

```
## R version 4.1.0 RC (2021-05-10 r80283)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server x64 (build 17763)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=C
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] biomaRt_2.48.0                  branchpointer_1.18.0
```

```
##  [3] caret_6.0-88                     ggplot2_3.3.3
##  [5] lattice_0.20-44                  BSgenome.Hsapiens.UCSC.hg38_1.4.3
##  [7] BSgenome_1.60.0                  rtracklayer_1.52.0
##  [9] Biostrings_2.60.0                XVector_0.32.0
## [11] GenomicRanges_1.44.0             GenomeInfoDb_1.28.0
## [13] IRanges_2.26.0                   S4Vectors_0.30.0
## [15] BiocGenerics_0.38.0              knitr_1.33
##
## loaded via a namespace (and not attached):
##   [1] colorspace_2.0-1            rjson_0.2.20
##   [3] ellipsis_0.3.2              class_7.3-19
##   [5] farver_2.1.0                bit64_4.0.5
##   [7] AnnotationDbi_1.54.0        prodlim_2019.11.13
##   [9] fansi_0.4.2                 lubridate_1.7.10
##  [11] xml2_1.3.2                  codetools_0.2-18
##  [13] splines_4.1.0                cachem_1.0.5
##  [15] pROC_1.17.0.1                Rsamtools_2.8.0
##  [17] kernlab_0.9-29               dbplyr_2.1.1
##  [19] png_0.1-7                    BiocManager_1.30.15
##  [21] compiler_4.1.0               httr_1.4.2
##  [23] assertthat_0.2.1             Matrix_1.3-3
##  [25] fastmap_1.1.0                htmltools_0.5.1.1
##  [27] prettyunits_1.1.1            tools_4.1.0
##  [29] gtable_0.3.0                 glue_1.4.2
##  [31] GenomeInfoDbData_1.2.6       reshape2_1.4.4
##  [33] dplyr_1.0.6                  rappdirs_0.3.3
##  [35] Rcpp_1.0.6                   Biobase_2.52.0
##  [37] vctrs_0.3.8                  nlme_3.1-152
##  [39] iterators_1.0.13             timeDate_3043.102
##  [41] gower_0.2.2                  xfun_0.23
##  [43] stringr_1.4.0                lifecycle_1.0.0
##  [45] restfulr_0.0.13              XML_3.99-0.6
##  [47] zlibbioc_1.38.0              MASS_7.3-54
##  [49] scales_1.1.1                 ipred_0.9-11
##  [51] BiocStyle_2.20.0             hms_1.1.0
##  [53] MatrixGenerics_1.4.0         SummarizedExperiment_1.22.0
##  [55] yaml_2.2.1                   curl_4.3.1
##  [57] memoise_2.0.0                rpart_4.1-15
##  [59] stringi_1.6.2                RSQLite_2.2.7
##  [61] highr_0.9                    BiocIO_1.2.0
##  [63] foreach_1.5.1                filelock_1.0.2
##  [65] BiocParallel_1.26.0          lava_1.6.9
##  [67] rlang_0.4.11                 pkgconfig_2.0.3
##  [69] matrixStats_0.58.0           bitops_1.0-7
##  [71] evaluate_0.14                purrr_0.3.4
##  [73] labeling_0.4.2               GenomicAlignments_1.28.0
##  [75] recipes_0.1.16               cowplot_1.1.1
##  [77] bit_4.0.4                    tidyselect_1.1.1
##  [79] gbm_2.1.8                    plyr_1.8.6
##  [81] magrittr_2.0.1               R6_2.5.0
##  [83] generics_0.1.0               DelayedArray_0.18.0
```

```
##  [85] DBI_1.1.1                  pillar_1.6.1
##  [87] withr_2.4.2                survival_3.2-11
##  [89] KEGGREST_1.32.0            RCurl_1.98-1.3
##  [91] nnet_7.3-16                tibble_3.1.2
##  [93] crayon_1.4.1               utf8_1.2.1
##  [95] BiocFileCache_2.0.0        rmarkdown_2.8
##  [97] progress_1.2.2             grid_4.1.0
##  [99] data.table_1.14.0          blob_1.2.1
## [101] ModelMetrics_1.2.2.2       digest_0.6.27
## [103] munsell_0.5.0
```