

# Package ‘branchpointer’

May 25, 2026

**Type** Package

**Title** Prediction of intronic splicing branchpoints

**Version** 1.38.0

**Date** 2025-07-22

**Author** Beth Signal

**Maintainer** Beth Signal <b.signal@garvan.org.au>

**Description** Predicts branchpoint probability for sites in intronic branchpoint windows. Queries can be supplied as intronic regions; or to evaluate the effects of mutations, SNPs.

**License** BSD\_3\_clause + file LICENSE

**LazyData** FALSE

**Depends** caret, R(>= 3.4)

**Imports** plyr, kernlab, gbm, stringr, cowplot, ggplot2, biomaRt,  
Biostrings, parallel, utils, stats,  
BSgenome.Hsapiens.UCSC.hg38, rtracklayer, GenomicRanges,  
Seqinfo, IRanges, S4Vectors, data.table

**Suggests** knitr, BiocStyle

**RoxygenNote** 6.0.1

**VignetteBuilder** knitr

**biocViews** Software, GenomeAnnotation, GenomicVariation,  
MotifAnnotation

**git\_url** <https://git.bioconductor.org/packages/branchpointer>

**git\_branch** RELEASE\_3\_23

**git\_last\_commit** 8b735ec

**git\_last\_commit\_date** 2026-04-28

**Repository** Bioconductor 3.23

**Date/Publication** 2026-05-25

## Contents

exonsToIntrons . . . . .	2
getBranchpointSequence . . . . .	3
getCanonical3SS . . . . .	4
getExonDists . . . . .	4
getPPT . . . . .	5
getQueryLoc . . . . .	5
gtfToExons . . . . .	6
makeBranchpointWindowForExons . . . . .	7
makeBranchpointWindowForSNP . . . . .	8
plotBranchpointWindow . . . . .	9
plotStructure . . . . .	10
predictBranchpoints . . . . .	10
predictionsToSummary . . . . .	11
readQueryFile . . . . .	12
<b>Index</b>	<b>14</b>

---

exonsToIntrons	<i>Convert exon annotation GRanges to intron locations</i>
----------------	--

---

### Description

Converts exon annotation to intron locations overlapping the branchpoint region for exculsion of non-branchpoint region SNPs Returns a character vector of chromosome locations

### Usage

```
exonsToIntrons(exons, maxDist = 50)
```

### Arguments

exons	GRanges containing exon co-ordinates. Should be produced by gtfToExons()
maxDist	Maximum distance from the 3' exon to create the branchpoint region.

### Value

GRanges containing intron window co-ordinates

### Author(s)

Beth Signal

---

`getBranchpointSequence`

*Get branchpoint sequence features Gets intronic sequence covering the branchpoint window and extracts predictive features*

---

### Description

Get branchpoint sequence features Gets intronic sequence covering the branchpoint window and extracts predictive features

### Usage

```
getBranchpointSequence(query, uniqueId = "test", queryType,  
  workingDirectory = ".", genome = NA, bedtoolsLocation = NA,  
  BSgenome = NULL, useParallel = FALSE, cores = 1, rmChr = FALSE)
```

### Arguments

<code>query</code>	branchpointer query GenomicRanges
<code>uniqueId</code>	unique string identifier for intermediate .bed and .fa files.
<code>queryType</code>	type of branchpointer query. "SNP" or "region".
<code>workingDirectory</code>	directory where intermediate .bed and .fa are located
<code>genome</code>	.fa genome file location
<code>bedtoolsLocation</code>	bedtools binary location (which bedtools)
<code>BSgenome</code>	BSgenome object
<code>useParallel</code>	use parallelisation to speed up code?
<code>cores</code>	number of cores to use in parallelisation (default = 1)
<code>rmChr</code>	remove "chr" before chromosome names before writing bed file. Required if genome sequence names do not contain "chr"

### Value

GenomicRanges with all features required to predict branchpoint probability scores

### Author(s)

Beth Signal

---

getCanonical3SS	<i>Get locations of the first five AG 3' splice site motifs</i>
-----------------	---

---

**Description**

Takes a variable length vector of distances to the AG motif, sorts and returns the first five. If there are less than five elements in the vector, returns the sorted vector and fills the remainder of the values with 300.

**Usage**

```
getCanonical3SS(ag)
```

**Arguments**

ag	Vector of distances to the AG splice site motif.
----	--

**Value**

Locations of the first five AG dinucleotides

**Author(s)**

Beth Signal

---

getExonDists	<i>Get the closest 3' and 5' exons</i>
--------------	--

---

**Description**

Finds the closest annotated exons from a genomic co-ordinate. Returns the distance to the 3' exon, distance to the 5' exon, ids of the 3' and 5' exon, and if the exons are from the same parent gene

**Usage**

```
getExonDists(query, exons, queryType)
```

**Arguments**

query	GenomicRangesquery
exons	GenomicRanges containing exon co-ordinates. Should be produced by gtfToExons()
queryType	type of query. "SNP" or "region"

**Value**

GenomicRanges with distance to the closest 3' and 5' exons, whether these exons are part of the same gene (i.e. is the location intronic), and the identifiers for the 3' and 5' exons.

**Author(s)**

Beth Signal

---

getPPT	<i>Get the best polypyrimidine tract</i>
--------	--

---

**Description**

Takes a query genomic sequence, finds all potential polypyrimidine tracts (PPTs) between the test site and the annotated 3' exon. Returns the distance to the start of the longest PPT, and its length.

**Usage**

```
getPPT(attributes)
```

**Arguments**

attributes      query attributes GenomicRanges

**Value**

distance to the start of the longest PPT, and its length

**Author(s)**

Beth Signal

---

getQueryLoc	<i>Find the closest 3' and 5' exons to a branchpointer query</i>
-------------	--

---

**Description**

Finds the closest annotated exons from genomic co-ordinates in a branchpointer query GRanges

**Usage**

```
getQueryLoc(query, queryType, maxDist = 50, filter = TRUE, exons)
```

**Arguments**

query	branchpointer query GenomicRanges must have chromosome at position 2, genomic co-ordinate at position 3, and strand at position 4.
queryType	type of query file ("SNP" or "region")
maxDist	maximum distance a SNP can be from an annotated 3' exon.
filter	remove SNP queries prior to finding nearest exons.
exons	data.frame containing exon co-ordinates. Should be produced by gtfToExons()

**Value**

GenomicRanges with the query and its location relative to the 3' and 5' exons

**Author(s)**

Beth Signal

---

gtfToExons	<i>Convert GTF file to exon location file</i>
------------	---

---

**Description**

Converts a GTF annotation to exon locations

**Usage**

```
gtfToExons(gtf)
```

**Arguments**

gtf	file containing the gtf annotation.
-----	-------------------------------------

**Value**

exon annotation GRanges

**Author(s)**

Beth Signal

**Examples**

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

---

`makeBranchpointWindowForExons`*Make branchpoint window regions*

---

**Description**

Generate branchpoint window regions corresponding to annotated exon(s) within a queried gene, transcript or exon id

**Usage**

```
makeBranchpointWindowForExons(id, idType, exons, forceClosestExon = FALSE)
```

**Arguments**

<code>id</code>	identifier(s) for the query gene/transcript/exon id
<code>idType</code>	type of id to match in the exon annotation file ("gene_id", "transcript_id", or "exon_id")
<code>exons</code>	GRanges containing exon co-ordinates.
<code>forceClosestExon</code>	Force branchpointer to find the closest exon and not the exon annotated as 5' to the query

**Value**

Granges with formatted query

**Author(s)**

Beth Signal

**Examples**

```
smallExons <- system.file("extdata", "gencode.v26.annotation.small.gtf", package = "branchpointer")
exons <- gtfToExons(smallExons)
windowquery <- makeBranchpointWindowForExons("ENSG00000139618.14", "gene_id", exons)
windowquery <- makeBranchpointWindowForExons("ENST00000357654.7", "transcript_id", exons)
windowquery <- makeBranchpointWindowForExons("ENSE000003518965.1", "exon_id", exons)
```

---

`makeBranchpointWindowForSNP`*Makes a branchpointer formatted GRanges object from refsnps ids*

---

### Description

Searches Biomart for refsnps ids, and pulls genomic location and sequence identity information  
Reformats alleles so each query has only one alternative allele

### Usage

```
makeBranchpointWindowForSNP(refSNP, mart.snp, exons, maxDist = 50,  
  filter = TRUE)
```

### Arguments

<code>refSNP</code>	Vector of refsnps ids
<code>mart.snp</code>	biomaRt mart object specifying the BioMart database and dataset to be used
<code>exons</code>	GRanges containing exon co-ordinates. Should be produced by <code>gtfToExons()</code>
<code>maxDist</code>	maximum distance a SNP can be from an annotated 3' exon.
<code>filter</code>	remove SNP queries prior to finding nearest exons?

### Value

formatted SNP query GRanges

### Author(s)

Beth Signal

### Examples

```
smallExons <- system.file("extdata", "gencode.v26.annotation.small.gtf", package = "branchpointer")  
exons <- gtfToExons(smallExons)  
  
mart.snp <- biomaRt::useMart("ENSEMBL_MART_SNP", dataset="hsapiens_snp", host="www.ensembl.org")  
query <- makeBranchpointWindowForSNP("rs587776767", mart.snp, exons)
```

---

plotBranchpointWindow *Plots branchpointer predictions*

---

## Description

Plots branchpointer predictions

## Usage

```
plotBranchpointWindow(queryName, predictions, probabilityCutoff = 0.52,  
  plotMutated = FALSE, plotStructure = TRUE, exons)
```

## Arguments

queryName	query id used to identify the SNP or region
predictions	Granges object generated by predictBranchpoints()
probabilityCutoff	probability score cutoff value for displaying U2 binding energy
plotMutated	plot alternative sequence predicitions alongside reference sequence predictions
plotStructure	plot structures for gene and 3' exon containing and skipping isoforms
exons	Granges containing exon co-ordinates. Should be produced by gtfToExons()

## Value

ggplot2 plot with branchpoint features in the specified intronic region

## Author(s)

Beth Signal

## Examples

```
smallExons <- system.file("extdata", "gencode.v26.annotation.small.gtf",  
  package = "branchpointer")  
exons <- gtfToExons(smallExons)  
g <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38  
  
querySNPFile <- system.file("extdata", "SNP_example.txt", package = "branchpointer")  
querySNP <- readQueryFile(querySNPFile, queryType = "SNP", exons = exons, filter = FALSE)  
predictionsSNP <- predictBranchpoints(querySNP, queryType = "SNP", BSgenome = g)  
plotBranchpointWindow(querySNP$id[1], predictionsSNP,  
  plotMutated = TRUE, exons = exons)
```

plotStructure *Plots transcript structures*

---

**Description**

Plots transcript structures

**Usage**

```
plotStructure(exonID, exons, keepTranscripts = "overlapping")
```

**Arguments**

exonID            id of the exon to plot  
exons            Granges containing exon co-ordinates.  
keepTranscripts        which transcripts to plot ("overlapping" or "withExon") "overlapping" will plot all transcripts overlapping the exon, whereas "withExon" will plot all transcripts containing the exon.

**Value**

ggplot2 plot transcript structures

**Author(s)**

Beth Signal

---

predictBranchpoints *Predict branchpoint probability scores*

---

**Description**

predicts branchpoint probability scores for each query site.

**Usage**

```
predictBranchpoints(query, uniqueId = "test", queryType,  
  workingDirectory = ".", genome = NA, bedtoolsLocation = NA,  
  BSgenome = NULL, useParallel = FALSE, cores = 1, rmChr = FALSE)
```

### Arguments

query	branchpointer query GenomicRanges
uniqueId	unique string identifier for intermediate .bed and .fa files.
queryType	type of branchpointer query. "SNP" or "region".
workingDirectory	directory where intermediate .bed and .fa are located
genome	.fa genome file location
bedtoolsLocation	bedtools binary location (which bedtools)
BSgenome	BSgenome object
useParallel	use parallelisation to speed up code?
cores	number of cores to use in parallelisation (default = 1)
rmChr	remove "chr" before chromosome names before writing bed file. Required if genome sequence names do not contain "chr"

### Value

GenomicRanges object with branchpoint probability scores for each site in query

### Author(s)

Beth Signal

### Examples

```
smallExons <- system.file("extdata", "gencode.v26.annotation.small.gtf",
package = "branchpointer")
exons <- gtfToExons(smallExons)
g <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38

querySNPFile <- system.file("extdata", "SNP_example.txt", package = "branchpointer")
querySNP <- readQueryFile(querySNPFile, queryType = "SNP", exons = exons, filter = FALSE)
predictionsSNP <- predictBranchpoints(querySNP, queryType = "SNP", BSgenome = g)
```

---

predictionsToSummary *Convert SNP branchpoint predictions across the branchpoint window to an intronic summary*

---

### Description

Takes predictions of branchpoint probabilities from a reference and alternative SNP and summarises the effect(s) of the SNP.

**Usage**

```
predictionsToSummary(query, predictions, probabilityCutoff = 0.52,
  probabilityChange = 0.15)
```

**Arguments**

query            query GRanges containing all SNP ids to be summarised

predictions    site-wide branchpoint probability predictions produced from predictBranchpoints()

probabilityCutoff  
Value to be used as the cutoff for discriminating branchpoint sites from non-branchpoint sites (default = 0.52)

probabilityChange  
Minimum probability score change required to call a branchpoint site as deleted or created by a SNP (default = 0.15)

**Value**

GRanges with summarised branchpoint changes occurring within the intron due to a SNP.

**Author(s)**

Beth Signal

**Examples**

```
smallExons <- system.file("extdata", "gencode.v26.annotation.small.gtf", package = "branchpointer")
exons <- gtfToExons(smallExons)
g <- BSgenome.Hsapiens.UCSC.hg38::BSgenome.Hsapiens.UCSC.hg38

querySNPFile <- system.file("extdata", "SNP_example.txt", package = "branchpointer")
querySNP <- readQueryFile(querySNPFile, queryType = "SNP", exons = exons, filter = FALSE)
predictionsSNP <- predictBranchpoints(querySNP, queryType = "SNP", BSgenome = g)

summarySNP <- predictionsToSummary(querySNP, predictionsSNP)
```

---

readQueryFile            *Read a query file*

---

**Description**

Reads and formats a manually generated query file, and finds relative locations of the closest annotated exons Converts unstranded SNPs to two entries for each strand. Checks for duplicate names and replaces if found.

**Usage**

```
readQueryFile(queryFile, queryType, exons, maxDist = 50, filter = TRUE)
```

**Arguments**

queryFile	tab delimited file containing query information. For intronic regions should be in the format: region id, chromosome name, region start, region end, strand. For SNP variants should be in the format: SNP id, chromosome name, SNP position, strand, reference allele (A/T/C/G), alternative allele (A/T/C/G)
queryType	type of query file ("SNP" or "region")
exons	GRanges containing exon co-ordinates. Should be produced by gtfToExons()
maxDist	maximum distance a SNP can be from an annotated 3' exon.
filter	remove SNP queries prior to finding finding nearest exons.

**Value**

Formatted query GRanges

**Author(s)**

Beth Signal

**Examples**

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

querySNPFile <- system.file("extdata", "SNP_example.txt", package = "branchpointer")
querySNP <- readQueryFile(querySNPFile, queryType = "SNP", exons)

queryIntronFile <- system.file("extdata", "intron_example.txt", package = "branchpointer")
queryIntron <- readQueryFile(queryIntronFile, queryType = "region", exons)
```

# Index

## \* **internal**

- exonsToIntrons, [2](#)
- getBranchpointSequence, [3](#)
- getCanonical3SS, [4](#)
- getExonDists, [4](#)
- getPPT, [5](#)
- getQueryLoc, [5](#)
- plotStructure, [10](#)

exonsToIntrons, [2](#)

getBranchpointSequence, [3](#)  
getCanonical3SS, [4](#)  
getExonDists, [4](#)  
getPPT, [5](#)  
getQueryLoc, [5](#)  
gtfToExons, [6](#)

makeBranchpointWindowForExons, [7](#)  
makeBranchpointWindowForSNP, [8](#)

plotBranchpointWindow, [9](#)  
plotStructure, [10](#)  
predictBranchpoints, [10](#)  
predictionsToSummary, [11](#)

readQueryFile, [12](#)