

# Package ‘svaNUMT’

July 26, 2025

**Type** Package

**Title** NUMT detection from structural variant calls

**Version** 1.15.1

**Date** 2025-07-22

**Description** svaNUMT contains functions for detecting NUMT events from structural variant calls.

It takes structural variant calls in GRanges of breakend notation and identifies NUMTs by nuclear-mitochondrial breakend junctions. The main function reports candidate NUMTs if there is a pair of valid insertion sites found on the nuclear genome within a certain distance threshold. The candidate NUMTs are reported by events.

**License** GPL-3 + file LICENSE

**Depends** GenomicRanges, rtracklayer, VariantAnnotation, StructuralVariantAnnotation, BiocGenerics, Biostrings, R (>= 4.0)

**Imports** assertthat, stringr, dplyr, methods, rlang, S4Vectors, Seqinfo, GenomeInfoDb, GenomicFeatures, pwalgn

**Suggests** TxDb.Hsapiens.UCSC.hg19.knownGene, BSgenome.Hsapiens.UCSC.hg19, ggplot2, devtools, testthat (>= 2.1.0), roxygen2, knitr, readr, plyranges, circlize, IRanges, SummarizedExperiment, rmarkdown

**RoxygenNote** 7.1.2

**Encoding** UTF-8

**VignetteBuilder** knitr

**biocViews** DataImport, Sequencing, Annotation, Genetics, VariantAnnotation

**BugReports** <https://github.com/PapenfussLab/svaNUMT/issues>

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

**git\_branch** devel

**git\_last\_commit** 5cfc9e8

**git\_last\_commit\_date** 2025-07-22

**Repository** Bioconductor 3.22

**Date/Publication** 2025-07-25

**Author** Ruining Dong [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-1433-0484>>)

**Maintainer** Ruining Dong <lnyidrn@gmail.com>

Contents

.mtLen . . . . .	2
numtDetect . . . . .	2
numtDetect_insseq . . . . .	4
numtDetect_known . . . . .	4
numtDetect_MT . . . . .	5
seqAlignment.score . . . . .	5
svaNUMT . . . . .	6
<b>Index</b>	<b>7</b>

---

.mtLen	<i>Calculating MT sequence length.</i>
--------	--

---

Description

Calculating MT sequence length.

Usage

.mtLen(bnd.start, bnd.end, chrM.len)

Arguments

- |           |                                       |
|-----------|---------------------------------------|
| bnd.start | starting breakend of the MT sequence. |
| bnd.end   | ending breakend of the MT sequence.   |
| chrM.len  | length of the reference MT genome.    |

Details

This function calculate the length of MT sequence length with BND notations.

Value

The length of the MT sequence. When the candidate MT BNDs can't be linked as one sequence, the returned value is NA.

---

numtDetect	<i>Detecting nuclear mitochondria fusion events.</i>
------------	--

---

Description

Detecting nuclear mitochondria fusion events.

**Usage**

```
numtDetect(
  gr,
  numtS,
  genomeMT,
  max_ins_dist = 10,
  maxgap_numtS = 10,
  min_len = 20,
  min.Align = 0.8
)
```

**Arguments**

<code>gr</code>	A GRanges object
<code>numtS</code>	A GRanges object of known NUMT sites.
<code>genomeMT</code>	A genome object of the mitochondria.
<code>max_ins_dist</code>	The maximum distance allowed on the reference genome between the paired insertion sites. Only intra-chromosomal NUMT events are supported. Default value is 10.
<code>maxgap_numtS</code>	The maximum distance allowed between the insertion sequence loci and known NUMTs.
<code>min_len</code>	The minimum length allowed of the insertion sequences. Default value is 20.
<code>min.Align</code>	The minimum alignment score allowed between the insertion sequence and MT genome.

**Details**

Nuclear mitochondrial fusion (NUMT) is a common event found in human genomes. This function searches for NUMT events by identifying breakpoints supporting the fusion of nuclear chromosome and mitochondrial genome. Only BND notations are supported at the current stage. Possible linked nuclear insertion sites are reported by chromosome in GRanges format.

**Value**

A nested list of GRanges objects of candidate NUMTs.

**Examples**

```
vcf.file <- system.file("extdata", "MT.vcf", package = "svaNUMT")
vcf <- VariantAnnotation::readVcf(vcf.file, "hg19")
gr <- breakpointRanges(vcf, nominalPosition=TRUE)
numtS <- readr::read_table(system.file("extdata", "numtS.txt", package = "svaNUMT"), col_names = FALSE)
colnames(numtS) <- c("bin", "seqnames", "start", "end", "name", "score", "strand")
numtS <- GRanges(numtS)
GenomeInfoDb::seqlevelsStyle(numtS) <- "NCBI"
genome <- BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
genomeMT <- genome$chrMT
numt.gr <- numtDetect(gr, numtS, genomeMT, max_ins_dist=20)
```

---

numtDetect_insseq	<i>Detecting nuclear mitochondria fusion events from unmapped insertion sequences.</i>
-------------------	--

---

### Description

Detecting nuclear mitochondria fusion events from unmapped insertion sequences.

### Usage

```
numtDetect_insseq(gr, genomeMT, min_len = 20, min.Align = 0.8)
```

### Arguments

gr	A GRanges object
genomeMT	A genome object of the mitochondria.
min_len	The minimum length allowed of the insertion sequences. Default value is 20.
min.Align	The minimum alignment score allowed between the insertion sequence and MT genome.

### Details

This function looks for NUMTs which the insertion MT sequences come from insertion sequences reported by SV callers.

### Value

A nested list of GRanges objects of candidate NUMTs.

---

numtDetect_known	<i>Detecting nuclear mitochondria fusion events from known NUMT sites.</i>
------------------	--

---

### Description

Detecting nuclear mitochondria fusion events from known NUMT sites.

### Usage

```
numtDetect_known(gr, numtS, max_ins_dist = 10, maxgap_numtS = 10)
```

### Arguments

gr	A GRanges object
numtS	A GRanges object of known NUMT sites.
max_ins_dist	The maximum distance allowed on the reference genome between the paired insertion sites. Only intra-chromosomal NUMT events are supported. Default value is 10.
maxgap_numtS	The maximum distance allowed between the insertion sequence loci and known NUMTs.

**Details**

This function looks for NUMTs which the insertion MT sequences come from known NUMT sites.

**Value**

A nested list of GRanges objects of candidate NUMTs.

---

numtDetect_MT	<i>Detecting nuclear mitochondria fusion events from breakpoints connected to MT reference genome.</i>
---------------	--

---

**Description**

Detecting nuclear mitochondria fusion events from breakpoints connected to MT reference genome.

**Usage**

```
numtDetect_MT(gr, max_ins_dist = 10)
```

**Arguments**

gr	A GRanges object
max_ins_dist	The maximum distance allowed on the reference genome between the paired insertion sites. Only intra-chromosomal NUMT events are supported. Default value is 10.

**Details**

This function looks for NUMTs which the insertion MT sequences come from known NUMT sites.

**Value**

A nested list of GRanges objects of candidate NUMTs.

---

seqAlignment.score	<i>Calculating the alignment score between a DNA sequence and target genome.</i>
--------------------	--

---

**Description**

Calculating the alignment score between a DNA sequence and target genome.

**Usage**

```
seqAlignment.score(seq, genome)
```

**Arguments**

seq	A string of DNA sequence.
genome	An XString of the target genome.

**Details**

This function calculates the alignment score between a DNA sequence and target genome.

**Value**

A alignment score between a DNA sequence and target genome.

---

svaNUMT	<i>svaNUMT: a package for NUMT detection</i>
---------	--

---

**Description**

svaNUMT contains functions for detecting NUMT events from structural variant calls. svaNUMT contains functions for detecting NUMT events from structural variant calls. It takes structural variant calls in GRanges of breakend notation and identifies NUMTs by nuclear-mitochondrial break-end junctions. The main function reports candidate NUMTs if there is a pair of valid insertion sites found on the nuclear genome within a certain distance threshold. The candidate NUMTs are reported by events.

**Details**

For more details on the features of StructuralVariantAnnotation, read the vignette: ‘browseVignettes(package = "svaNUMT")’

# Index

## \* **internal**

- .mtLen, [2](#)
- numtDetect\_insseq, [4](#)
- numtDetect\_known, [4](#)
- numtDetect\_MT, [5](#)
- seqAlignment.score, [5](#)
- .mtLen, [2](#)

- numtDetect, [2](#)
- numtDetect\_insseq, [4](#)
- numtDetect\_known, [4](#)
- numtDetect\_MT, [5](#)

- seqAlignment.score, [5](#)
- svaNUMT, [6](#)