Package ‘hgvsParseR’

April 25, 2018

Title Parse HGVS variant descriptor strings

Version 0.0.0.9000

Description Parses genetic variant descriptor scripts in the HGVS format.

Depends R (>= 3.1.2)

License GPL

Encoding UTF-8

LazyData true

Suggests testthat

RoxygenNote 6.0.1

R topics documented:

`new.hgvs.builder.c` *Coding Sequence HGVS Builder*

Description

A constructor for a CDS (=coding sequence) HGVS builder object. The object contains a collection of functions for building CDS HGVS strings. The resulting object encapsulates the following functions:

- substitution(pos,ancestral,variant,posOffset=0) CDS substitution variants. pos = position (integer); ancestral = ancestral nucleotide [ACGT]; variant = variant nucleotide [ACGT]; posOffset = offset from the position when crossing exon-intron borders (integer, defaults to 0)

- deletion(start,stop,startOffset=0,stopOffset=0) CDS deletion. start = start position (integer); stop = stop position (integer); startOffset = offset from the start position when crossing exon-intron borders (integer, defaults to 0); stopOffset = offset from the stop position when crossing exon-intron borders (integer, defaults to 0)
- inversion(start,stop,startOffset=0,stopOffset=0) CDS inversion. start = start position (integer); stop = stop position (integer); startOffset = offset from the start position when crossing exon-intron borders (integer, defaults to 0); stopOffset = offset from the stop position when crossing exon-intron borders (integer, defaults to 0)
- duplication(start,stop,startOffset=0,stopOffset=0) CDS duplication. start = start position (integer); stop = stop position (integer); startOffset = offset from the start position when crossing exon-intron borders (integer, defaults to 0); stopOffset = offset from the stop position when crossing exon-intron borders (integer, defaults to 0)
- insertion(start,variant,startOffset=0) CDS insertion. start = position immediately preceeding the insertion (integer); seq = inserted nucleotide sequence [ACGT]+ ; startOffset = offset from the start position when crossing exon-intron borders (integer, defaults to 0)
- delins(start,stop,variant,startOffset=0,stopOffset=0) CDS deletion and insertion. start = start position (integer); stop = stop position relative to the reference (integer); seq = inserted nucleotide sequence [ACGT]+ ; startOffset = offset from the start position when crossing exon-intron borders (integer, defaults to 0); stopOffset = offset from the stop position when crossing exon-intron borders (integer, defaults to 0)
- cis(...) Multi-variant phased in cis. Parameters are coding HGVS strings for the corresponding single mutants
- trans(...) Multi-variant phased in trans. Parameters are coding HGVS strings for the corresponding single mutants
- nophase(...) Multi-variant with unknown phasing. Parameters are coding HGVS strings for the corresponding single mutants

Usage

```
new.hgvs.builder.c()
```

Value

A `hgvs.builder.c` object with functions for building coding HGVS strings. The individual functions return single-element character vectors containing these strings.

Examples

```
builder <- new.hgvs.builder.c()
string1 <- builder$substitution(123,"A","G",posOffset=2)
string2 <- builder$delins(123,129,"ATTG")
string3 <- with(builder,cis(substitution(123,"A","C"),substitution(231,"G","A")))
```

`new.hgvs.builder.g` *Genomic HGVS Builder*

Description

A constructor for a genomic-level HGVS builder object. The object contains a collection of functions for building genomic HGVS strings.

Usage

```
new.hgvs.builder.g()
```

Details

The resulting object encapsulates the following functions:

- substitution(pos,ancestral,variant) Genomic substitution variants. pos = position (integer); ancestral = ancestral nucleotide [ACGT]; variant = variant nucleotide [ACGT]
- deletion(start,stop) Genomic deletion. start = start position (integer); stop = stop position (integer)
- inversion(start,stop) Genomic inversion. start = start position (integer); stop = stop position (integer)
- duplication(start,stop) Genomic duplication. start = start position (integer); stop = stop position (integer)
- insertion(start,variant) Genomic insertion. start = position immediately preceeding the insertion (integer); seq = inserted nucleotide sequence [ACGT]+
- delins(start,stop,variant) Genomic deletion and insertion. start = start position (integer); stop = stop position relative to the reference (integer); seq = inserted nucleotide sequence [ACGT]+
- cis(...) Multi-variant phased in cis. Parameters are genomic HGVS strings for the corresponding single mutants
- trans(...) Multi-variant phased in trans. Parameters are genomic HGVS strings for the corresponding single mutants
- nophase(...) Multi-variant with unknown phasing. Parameters are genomic HGVS strings for the corresponding single mutants

Value

A `hgvs.builder.g` object with functions for building genomic HGVS strings. The individual functions return single-element character vectors containing these strings.

Examples

```
builder <- new.hgvs.builder.g()
string1 <- builder$substitution(123,"A","G")
string2 <- builder$delins(123,129,"ATTG")
string3 <- with(builder,cis(substitution(123,"A","C"),substitution(231,"G","A")))
```

`new.hgvs.builder.p` *Protein HGVS Builder*

Description

A constructor for a protein-level HGVS builder object. The object contains a collection of functions for building protein HGVS strings.

Usage

```
new.hgvs.builder.p(aacode = c(1, 3))
```

Details

The resulting object encapsulates the following functions:

- synonymous() A synonymous variant. No parameters required.
- synonymous(pos,ancestral) Unofficial (yet frequently used) version of synonymous variant syntax. pos = position (integer); ancestral = ancestral amino acid in one-letter or three-letter code.
- substitution(pos,ancestral,variant) AA substitution variants. pos = position (integer); ancestral = ancestral amino acid in one-letter or three-letter code; variant = variant amino acid in one-letter or three-letter code
- deletion(startPos,startAA,endPos,endAA) AA deletion. startPos = start position (integer); startAA = start amino acid in one-letter or three-letter code; endPos = stop position (integer); endAA = start amino acid in one-letter or three-letter code
- duplication(startPos,startAA,endPos,endAA) AA duplication. startPos = start position (integer); startAA = start amino acid in one-letter or three-letter code; endPos = stop position (integer); endAA = start amino acid in one-letter or three-letter code
- insertion(leftPos,leftAA,rightAA,seq) AA insertion. leftPos = position immediately preceeding the insertion (integer); leftAA = corresponding amino acid in one-letter or three-letter code; rightAA = amino acid to the right of the insertion, in one-letter or three-letter code; seq = inserted amino acid sequence, given as a character vector containing the individual one-letter or three-letter amino acid codes.
- delins(startPos,startAA,endPos,endAA,seq) AA deletion and insertion. startPos = start position (integer); startAA = start amino acid in one-letter or three-letter code; endPos = stop position (integer); endAA = start amino acid in one-letter or three-letter code; seq = inserted amino acid sequence, given as a character vector containing the individual one-letter or three-letter amino acid codes.
- frameshift(startPos,startAA,variantAA=NA,newStop=NA) Frameshift variant. startPos = start position (integer); startAA = start amino acid in one-letter or three-letter code; variantAA = amino acid replacing the start position in the frameshift sequence, given in one-letter or three-letter code, or `NA` to omit (default); newStop = the position of the nearest coding resulting from the frameshift, or `NA` to omit (default).
- cis(...) Multi-variant phased in cis. Parameters are coding HGVS strings for the corresponding single mutants. As phasing in trans would be nonsensical in a protein context, the `trans()` and `nophase()` methods are not provided here.

Value

A `hgvs.builder.g` object with functions for building genomic HGVS strings. The individual functions return single-element character vectors containing these strings.

Examples

```
builder <- new.hgvs.builder.g()
string1 <- builder$substitution(123,"R","K")
string2 <- builder$delins(123,"Arg",152,"Leu",c("Lys","Trp","Ser"))
string3 <- with(builder,cis(substitution(123,"R","K"),deletion(125,"S",152,"L")))
```

`parseHGVS` *HGVS Parser*

Description

Parses HGVS strings

Usage

```
parseHGVS(strings, aacode = c(NA, 1, 3))
```

Arguments

`strings`	A character vector containing the HGVS strings
`aacode`	allowed values: 1, 3, or NA. Determines whether 1-letter codes or 3-letter codes should be forced. NA uses input format.

Value

A `data.frame` with the following columns:

Examples

```
result <- parseHGVS(c("g.1318G>T","c.123_125inv","p.R123_L152del"))
```

Index