Package: rvarsim (via r-universe)

June 10, 2026

Title R/Bioconductor Variant Simulator with HGVS Notation

Version 0.99.1

Description Simulates all possible single nucleotide variants (SNVs) across MANE Select transcripts including coding sequence (CDS), untranslated regions (UTRs), and canonical splice sites. Outputs variants in HGVS notation. Provides a comprehensive HGVS toolkit: parsing, syntactic and semantic validation, normalization (3' shifting, common affix trimming), format conversion (HGVS <-> VCF <-> SPDI), transcription mapping (genomic <-> coding), translation to protein consequence, backtranslation from protein to coding variants, variant extraction from sequence alignment, and liftover between genome assemblies.

License MIT + file LICENSE

URL https://github.com/liu-sun/rvarsim

BugReports https://github.com/liu-sun/rvarsim/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Imports BiocGenerics, GenomeInfoDb, GenomicFeatures, GenomicRanges, IRanges, S4Vectors, Biostrings, BSgenome, ensembldb

Suggests AnnotationHub, BSgenome.Hsapiens.UCSC.hg38, EnsDb.Hsapiens.v86, VariantAnnotation, rtracklayer, BiocStyle, testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

biocViews Genetics, VariantAnnotation, SNP, Sequencing, Transcription, VariantDetection, Alignment, Normalization

Config/roxygen2/version 8.0.0

Config/pak/sysreqs
make libbz2-dev liblzma-dev libpng-dev libxml2-dev libssl-dev xz-utils zlib1g-dev

Repository https://biocstaging.r-universe.dev

Date/Publication 2026-06-10 07:08:19 UTC

RemoteUrl https://github.com/BiocStaging/rvarsim

RemoteRef HEAD

RemoteSha 7b0d289dedbea6448b59f9f74ead186b79b02e60

Contents

`backtranslate_hgvs` *Infer transcript variants from protein consequences*

Description

Given an observed protein (p.) variant, infers all possible coding (c.) variants that could produce it by enumerating possible codon changes in the genetic code.

Usage

```
backtranslate_hgvs(hgvs_p, txdb, bsgenome, genetic_code = NULL)
```

Arguments

hgvs_p	Character vector of HGVS p. notation strings (e.g., "p.Arg215Gly").
txdb	A TxDb or EnsDb object.
bsgenome	A BSgenome object for retrieving the reference coding sequence.
genetic_code	Genetic code table (default: standard code).

Details

For example, p.Arg215Gly could be caused by any of the 6 AGA/Gly codon pair changes. This function enumerates all possibilities.

Value

A list of character vectors, where each element is a vector of possible HGVS c. notation strings for the corresponding protein variant.

Examples

```
# The backtranslate function requires a TxDb and BSgenome with
# matching seqlevels (use same reference for both):
if (requireNamespace("EnsDb.Hsapiens.v86", quietly = TRUE) &&
    requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly = TRUE)) {
    cat("Ready for backtranslation")
}
```

---

extract_hgvs *Generate HGVS descriptions from reference and observed sequences*

Description

Aligns a reference and observed sequence, identifies differences (substitutions, insertions, deletions), and generates HGVS variant descriptions.

Usage

```
extract_hgvs(
  ref_seq,
  obs_seq,
  accession,
  notation = "c",
  start_pos = 1L,
  algorithm = c("needleman", "smith")
)
```

Arguments

ref_seq	Character string: reference sequence.
obs_seq	Character string: observed (variant) sequence.
accession	Accession number for the HGVS output (e.g., "NM_000546.6").
notation	HGVS notation prefix. Default "c".
start_pos	Starting position offset. For c. notation, 1 = first base of CDS. For g. notation, the genomic position of the first base in the reference sequence.
algorithm	Alignment algorithm: "needleman" (global) or "smith" (local). Default "needleman".

Value

A character vector of HGVS variant description strings, one per detected variant. Returns "=" if sequences are identical.

Examples

```
ref <- "ATGCGTACGTAG"
obs <- "ATGCATACCTAG"
extract_hgvs(ref, obs, "NM_000546.6", "c", 1)
```

---

fetch_mane_txdb *Fetch MANE Select transcripts*

Description

Retrieves MANE Select transcripts using AnnotationHub and ensembldb. MANE Select transcripts are a single representative transcript per protein-coding gene, jointly curated by NCBI and EMBL-EBI.

Usage

```
fetch_mane_txdb(
  txdb = NULL,
  ah_id = NULL,
  species = "Homo sapiens",
  assembly = "GRCh38"
)
```

Arguments

txdb	An EnsDb or TxDb object. If NULL, the function fetches one from AnnotationHub.
ah_id	Optional AnnotationHub ID (e.g. "AH113665") for a specific EnsDb release. Overrides automatic lookup.
species	Species identifier. Default "Homo sapiens".
assembly	Genome assembly name. Default "GRCh38".

Value

An EnsDb object containing only MANE Select transcripts, or the original TxDb/EnsDb if filtering is not possible.

Examples

```
# Auto-fetch requires AnnotationHub (internet):
if (requireNamespace("AnnotationHub", quietly = TRUE) &&
    requireNamespace("EnsDb.Hsapiens.v86", quietly = TRUE)) {
    cat("Ready for MANE transcript fetching")
}
```

---

format_hgvs *Format variants in HGVS notation*

Description

Converts a variants data.frame (from generate_variants) into HGVS-compliant coding DNA notation.

Usage

```
format_hgvs(variants, use_transcript_version = FALSE)
```

Arguments

variants	A data.frame from generate_variants.
use_transcript_version	Logical. If TRUE, appends transcript version numbers (e.g., NM_000546.6) when the transcript ID is a RefSeq accession. Default: FALSE.

Details

HGVS conventions used:

- CDS: NM_.*:c.[pos][ref]>[alt] (e.g., c.215C>G)
- 5' UTR: c.-[pos][ref]>[alt] (e.g., c.-14G>A)
- 3' UTR: c.*[pos][ref]>[alt] (e.g., c.*32T>C)
- Splice donor: c.[boundary]+[offset][ref]>[alt] (e.g., c.453+1G>T)
- Splice acceptor: c.[boundary]-[offset][ref]>[alt] (e.g., c.612-2A>G)

Value

A data.frame with all original columns plus:

hgvs_c HGVS coding DNA notation string.

hgvs_g HGVS genomic notation string (e.g., NC_000001.11:g.123456A>G).

Examples

```
# Self-contained format demo:
df <- data.frame(
    transcript_id = "NM_TEST",
    region = c("cds", "five_utr"),
    genomic_pos = c(215L, 30L),
    genomic_ref = c("C", "G"),
    genomic_alt = c("G", "A"),
    cds_pos = c(215L, -14L),
    exon_boundary = c(NA_integer_, NA_integer_),
    offset = c(NA_integer_, NA_integer_),
    stringsAsFactors = FALSE
)
format_hgvs(df)
```

---

generate_variants *Generate all possible single nucleotide variants*

Description

For each position across CDS, UTR, and splice site regions, extracts the reference base from the genome and generates the three possible alternative alleles (A, C, G, T minus the reference).

Usage

```
generate_variants(
  tx_struct,
  bsgenome,
  regions = c("cds", "five_utr", "three_utr", "splice_site"),
  transcript_ids = NULL
)
```

Arguments

`tx_struct`	A `transcript_structure` object from `get_transcript_structure`.
`bsgenome`	A `BSgenome` object (e.g., `BSgenome.Hsapiens.UCSC.hg38`) providing the reference genome sequence.
`regions`	Character vector specifying which region types to include. Options: `"cds"`, `"five_utr"`, `"three_utr"`, `"splice_site"`. Default: all four.
`transcript_ids`	Optional character vector of transcript IDs to limit generation. `NULL` means all transcripts in the structure.

Value

A `data.frame` with columns:

transcript_id Transcript identifier.

region Region type: "cds", "five_utr", "three_utr", "splice_donor", "splice_acceptor".

genomic_pos Genomic position (1-based).

genomic_ref Reference allele at this position.

genomic_alt Alternative allele.

cds_pos Position relative to CDS start (1 = A of ATG; negative = 5'UTR; NA for splice sites described by boundary).

exon_boundary For splice sites: the nearest coding coordinate (NA for other regions).

offset For splice sites: the intronic offset (+1, +2, -1, -2; NA otherwise).

Examples

```
if (requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly = TRUE)) {
    cat("Generate variants with BSgenome.Hsapiens.UCSC.hg38")
}
```

`get_transcript_structure` *Extract transcript structure for variant simulation*

Description

Extracts the coding sequence (CDS), 5' UTR, 3' UTR, and canonical splice site positions for one or more transcripts from a TxDb/EnsDb.

Usage

```
get_transcript_structure(txdb, transcript_ids = NULL)
```

Arguments

`txdb`	A `TxDb` or `EnsDb` object.
`transcript_ids`	Optional character vector of transcript IDs to limit extraction. If `NULL`, all transcripts are used.

Details

Canonical splice sites are defined as the first and last two positions of each intron (±1, ±2 from exon boundaries).

Value

A list with components:

cds A `GRangesList` of CDS exons by transcript.

five_utr A `GRangesList` of 5' UTR regions by transcript.

three_utr A `GRangesList` of 3' UTR regions by transcript.

exons A `GRangesList` of all exons by transcript.

splice_sites A `GRanges` of splice site positions (donor +1,+2 and acceptor -1,-2).

cds_start Named integer vector of CDS start positions (genomic coordinate of ATG).

cds_end Named integer vector of CDS end positions (genomic coordinate of stop codon last base).

strand Named character vector of strand ("+" or "-").

Examples

```
library(EnsDb.Hsapiens.v86)
mane <- fetch_mane_txdb(EnsDb.Hsapiens.v86)
struct <- get_transcript_structure(mane, transcript_ids = NULL)
```

---

hgvs_conversion *Convert HGVS to and from other variant formats*

Description

Converts between HGVS notation and common variant representation formats: VCF (Variant Call Format), SPDI (Sequence Position Deletion Insertion), and VRS (Variation Representation Specification, basic support).

Value

A `data.frame` or character vector, depending on the conversion function used.

VCF format

VCF uses 1-based positions: CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO. Conversion from HGVS c. notation requires transcript-to-genomic mapping.

SPDI format

NCBI's SPDI uses 0-based interbase coordinates: sequence_id:position:deleted_sequence:inserted_sequence

hgvs_to_spdi *Convert HGVS to SPDI format*

Description

SPDI uses 0-based interbase coordinates: seqid:position:deletion:insertion

Usage

```
hgvs_to_spdi(hgvs_strings, assembly = "GRCh38")
```

Arguments

hgvs_strings	Character vector of HGVS g. notation strings.
assembly	Genome assembly name.

Value

Character vector of SPDI strings.

Examples

```
hgvs_to_spdi("NC_000001.11:g.123456A>G")
```

hgvs_to_vcf *Convert HGVS to VCF data.frame*

Description

Convert HGVS to VCF data.frame

Usage

```
hgvs_to_vcf(hgvs_strings, txdb = NULL, bsgenome = NULL)
```

Arguments

hgvs_strings	Character vector of HGVS variant descriptions.
txdb	Optional TxDb for mapping c. to genomic coordinates. Required for transcript-level variants.
bsgenome	Optional BSgenome for chromosome naming.

Value

A data.frame with VCF columns: CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO.

Examples

```
# VCF conversion:
vcf <- hgvs_to_vcf("NC_000001.11:g.123456A>G")

# SPDI round-trip:
hgvs_to_spdi("NC_000001.11:g.123456A>G")
```

`is_valid_hgvs` *Check if an HGVS string is valid (convenience wrapper)*

Description

Check if an HGVS string is valid (convenience wrapper)

Usage

```
is_valid_hgvs(hgvs_string, ...)
```

Arguments

`hgvs_string`	A single HGVS variant string.
`...`	Additional arguments passed to `validate_hgvs`.

Value

`TRUE` if both syntactic and semantic checks pass.

Examples

```
is_valid_hgvs("NM_000546.6:c.215C>G")
```

`liftover_c_hgvs` *Lift over a c. variant by converting to genomic, lifting, then converting back*

Description

Lift over a c. variant by converting to genomic, lifting, then converting back

Usage

```
liftover_c_hgvs(
  hgvs_c,
  source_txdb,
  source_bsgenome,
  chain_file,
  target_txdb,
  target_bsgenome
)
```

Arguments

hgvs_c	A HGVS c. notation string.
source_txdb	TxDb for the source assembly.
source_bsgenome	BSgenome for the source assembly.
chain_file	Path to chain file.
target_txdb	TxDb for the target assembly.
target_bsgenome	BSgenome for the target assembly.

Value

HGVS c. notation string in the target assembly, or NA.

Examples

```
if (requireNamespace("rtracklayer", quietly = TRUE)) {
    cat("rtracklayer available for c. liftover")
}
```

---

liftover_hgvs *Map variants between aligned genomic sequences*

Description

Converts HGVS variants between genome assemblies (e.g., hg19 to hg38) using chain files via the `rtracklayer` package. Supports both g. and c. notation (c. requires TxDb for the target assembly).

Usage

```
liftover_hgvs(hgvs_strings, chain_file, target_bsgenome = NULL)
```

Arguments

hgvs_strings	Character vector of HGVS variant descriptions.
chain_file	Path to a UCSC chain file (e.g., `"hg19ToHg38.over.chain"`).
target_bsgenome	A `BSgenome` object for the target assembly (used to resolve reference alleles in the target).

Value

Character vector of HGVS strings mapped to the target assembly, or `NA` for variants that cannot be mapped.

Examples

```
# Liftover requires rtracklayer and a chain file:
if (requireNamespace("rtracklayer", quietly = TRUE)) {
    cat("rtracklayer available for liftover")
}
```

`normalize_hgvs` *Normalize HGVS variant descriptions*

Description

Applies HGVS normalization rules: 3' shifting for indels in repetitive regions, rewriting insertions as duplications when applicable, and trimming common prefixes/suffixes.

Usage

```
normalize_hgvs(hgvs_strings, bsgenome = NULL, txdb = NULL)
```

Arguments

`hgvs_strings`	Character vector of HGVS variant descriptions.
`bsgenome`	A `BSgenome` object providing the reference sequence for 3'-shifting context.
`txdb`	Optional `TxDb` for transcript context when normalizing c. notation variants.

Details

Normlization ensures a single canonical representation for each variant, which is critical for comparison and deduplication.

Value

A character vector of normalized HGVS strings, or the original string if normalization cannot be applied.

Examples

```
# Normalize an HGVS string:
normalize_hgvs("NM_000546.6:c.215C>G")
```

parse_hgvs *Parse HGVS variant descriptions*

Description

Parses HGVS-formatted variant strings into structured R list objects. Supports all major HGVS notation types: substitution (>), deletion (del), insertion (ins), duplication (dup), inversion (inv), deletion- insertion (delins), repeat, and frameshift variants across c., g., p., n., m., and r. notation prefixes.

Usage

```
parse_hgvs(hgvs_strings, strict = FALSE)
```

Arguments

hgvs_strings	Character vector of HGVS variant descriptions.
strict	Logical. If TRUE, raises errors on parse failures. If FALSE, returns NA for unparseable strings with a warning. Default: FALSE.

Value

A list of parsed HGVS objects, each with components:

type Variant type: "substitution", "deletion", "insertion", "duplication", "inversion", "delins", "frameshift", "repeat", or "unknown".

accession Accession number (e.g., "NM_000546.6").

notation Notation prefix: "c", "g", "p", "n", "m", "r".

reference Reference allele/sequence.

alternate Alternate allele/sequence.

position List with start, end, offset_start, offset_end, is_utr, utr_offset.

raw Original input string.

Examples

```
parse_hgvs(c("NM_000546.6:c.215C>G", "NC_000001.11:g.123456delA"))
```

`simulate_variants` *Simulate all possible SNVs for MANE Select transcripts*

Description

Main entry point for the variant simulation pipeline. Fetches MANE Select transcripts, extracts transcript structure, generates all possible single nucleotide variants across CDS, UTR, and canonical splice site regions, and formats them in HGVS notation.

Usage

```
simulate_variants(
  txdb = NULL,
  bsgenome = NULL,
  transcript_ids = NULL,
  regions = c("cds", "five_utr", "three_utr", "splice_site"),
  use_messages = TRUE
)
```

Arguments

`txdb`	An `EnsDb` or `TxDb` object. If `NULL`, fetches from AnnotationHub (requires internet).
`bsgenome`	A `BSgenome` object providing the reference genome sequence (e.g., `BSgenome.Hsapiens.UCSC.hg38`).
`transcript_ids`	Optional character vector of transcript IDs to simulate. `NULL` processes all MANE Select transcripts. Limiting is recommended for interactive use due to the large number of variants (~10,000 per transcript).
`regions`	Character vector. Which regions to simulate. Options: `"cds"` (default), `"five_utr"`, `"three_utr"`, `"splice_site"`.
`use_messages`	Logical. If `TRUE`, prints progress messages. Default: `TRUE`.

Value

A `data.frame` with columns:

transcript_id Transcript identifier.

region Region type.

genomic_pos Genomic position (1-based).

genomic_ref Reference allele.

genomic_alt Alternative allele.

cds_pos CDS-relative position (NA for splice sites).

exon_boundary Exon boundary coordinate (splice sites only).

offset Intronic offset (splice sites only).

hgvs_c HGVS coding DNA notation.

hgvs_g HGVS genomic notation.

See Also

fetch_mane_txdb for obtaining MANE Select transcripts. get_transcript_structure for transcript structure extraction. generate_variants for the variant generation step. format_hgvs for HGVS formatting details.

Examples

```
# Check validity of HGVS strings:
is_valid_hgvs("NM_000546.6:c.215C>G")

# Simulate requires matching TxDb + BSgenome with same seqlevels style:
# library(EnsDb.Hsapiens.v86)
# library(BSgenome.Hsapiens.UCSC.hg38)
# simulate_variants(EnsDb.Hsapiens.v86, BSgenome.Hsapiens.UCSC.hg38,
#                   transcript_ids = "ENST00000357654")
```

---

spdi_to_hgvs *Convert SPDI to HGVS g. notation*

Description

Convert SPDI to HGVS g. notation

Usage

```
spdi_to_hgvs(spdi_strings)
```

Arguments

spdi_strings	Character vector of SPDI strings.

Value

Character vector of HGVS g. notation strings.

Examples

```
spdi_to_hgvs("NC_000001.11:123455:A:G")
```

`transcribe_hgvs` *Map variants between genome and transcript notation*

Description

Converts HGVS variants between genomic (g.) and coding (c.) notation using a TxDb/EnsDb to resolve exon/CDS structure. Handles intronic offsets, UTR positions, and splice boundary coordinates.

Usage

```
transcribe_hgvs(
  hgvs_strings,
  txdb,
  bsgenome,
  transcript_id = NULL,
  direction = c("c_to_g", "g_to_c", "auto")
)

c_to_g(hgvs_strings, txdb, bsgenome, transcript_id = NULL)

g_to_c(hgvs_strings, txdb, bsgenome, transcript_id = NULL)
```

Arguments

`hgvs_strings`	Character vector of HGVS variant descriptions.
`txdb`	A `TxDb` or `EnsDb` object providing transcript structure.
`bsgenome`	A `BSgenome` object for sequence context.
`transcript_id`	Optional transcript ID to use when disambiguating c. notation (required when multiple transcripts share the same accession).
`direction`	Direction of transcription: `"c_to_g"`, `"g_to_c"`, or `"auto"` (auto-detect from notation). Default: `"auto"`.

Value

A list of HGVS variant strings in the target notation.

Examples

```
# Transcribe needs a TxDb with matching BSgenome seqlevels:
if (requireNamespace("EnsDb.Hsapiens.v86", quietly = TRUE) &&
    requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly = TRUE)) {
    cat("Ready for transcription mapping")
}
```

translate_hgvs *Infer protein consequences from transcript variants*

Description

Translates CDS nucleotide variants to protein (p.) HGVS notation using the genetic code and CDS structure from a TxDb/EnsDb.

Usage

```
translate_hgvs(hgvs_strings, txdb, bsgenome, genetic_code = NULL)
```

Arguments

hgvs_strings	Character vector of HGVS c. notation strings.
txdb	A TxDb or EnsDb object.
bsgenome	A BSgenome object for retrieving coding sequence.
genetic_code	A named integer vector or genetic code table mapping codons to single-letter amino acids. If NULL (default), uses the standard genetic code.

Details

Supported consequence types:

- Missense: p.Arg215Gly (substitution causing amino acid change)
- Nonsense: p.Trp215Ter (premature stop codon)
- Silent: p.(=) (no amino acid change)
- Frameshift: p.Arg215LeufsTer5 (frameshift with altered reading frame)
- Inframe deletion: p.Lys215_Gly217del
- Inframe insertion: p.Lys215_Gly216insAla

Value

Character vector of HGVS p. notation strings, or NA for variants that cannot be translated (e.g., non-coding variants).

Examples

```
# Translate requires a TxDb with matching BSgenome:
if (requireNamespace("EnsDb.Hsapiens.v86", quietly = TRUE) &&
    requireNamespace("BSgenome.Hsapiens.UCSC.hg38", quietly = TRUE)) {
    cat("Ready for translation")
}
```

validate_hgvs *Validate HGVS variant descriptions*

Description

Performs syntactic and semantic validation of HGVS variant strings. Syntactic validation checks that the string follows correct HGVS grammar. Semantic validation verifies that positions are in range, reference alleles match expected sequence, and the variant is biologically plausible.

Usage

```
validate_hgvs(hgvs_strings, txdb = NULL, bsgenome = NULL)
```

Arguments

hgvs_strings	Character vector of HGVS variant descriptions.
txdb	Optional TxDb or EnsDb for transcript- context semantic validation (e.g., checking that CDS positions are within the actual CDS length).
bsgenome	Optional BSgenome for reference allele verification.

Value

A data.frame with columns:

input Original input string.

syntax_valid Logical: passes syntactic checks.

semantic_valid Logical: passes semantic checks.

syntax_error Error message from syntactic check (or NA).

semantic_error Error message from semantic check (or NA).

type Variant type if successfully parsed.

Examples

```
validate_hgvs(c("NM_000546.6:c.215C>G", "invalid string"))
```

vcf_to_hgvs *Convert VCF row to HGVS notation*

Description

Convert VCF row to HGVS notation

Usage

```
vcf_to_hgvs(vcf_df, assembly = "GRCh38")
```

Arguments

vcf_df	A data.frame with VCF columns: CHROM, POS, REF, ALT. May also contain ID, QUAL, FILTER.
assembly	Genome assembly name for creating NC_ accession (e.g., "GRCh38").

Value

Character vector of HGVS g. notation strings.

Examples

```
vcf <- data.frame(CHROM = "1", POS = 123456L, REF = "A", ALT = "G")
vcf_to_hgvs(vcf, "GRCh38")
```

Index