Package 'mSigSpectra'

September 27, 2026

Type Package

Title Build Mutational-Spectrum Catalogs from Variant Call Format Files

Version 0.1.3

Description Reads variant call format (VCF) files in a caller-agnostic way, annotates variants with flanking sequence context and transcriptional strand, and builds mutational-spectrum catalogs of single base substitutions (SBS), doublet base substitutions (DBS), and small insertions and deletions (indels, ID) at several resolutions (SBS96, SBS192, SBS1536, DBS78, DBS136, DBS144, ID83, ID89, ID166, ID476) and in both counts and density representations. Successor to the numerical core of the 'ICAMS' package with plotting, 'shiny', and portable document format (PDF) reporting removed; plotting is provided separately by 'mSigPlot'. Used in the preparation of Rozen et al. (2026) <doi:10.5281/zenodo.18451842>.

License GPL-3

URL https://github.com/steverozen/mSigSpectra, https://steverozen.github.io/mSigSpectra/

BugReports https://github.com/steverozen/mSigSpectra/issues

Encoding UTF-8

LazyData true

Language en-US

Imports BSgenome, data.table, fastrc, GenomeInfoDb, GenomicRanges, IRanges, Rcpp, S4Vectors, stats, stringi, tools

Depends R (>= 4.1)

LinkingTo Rcpp

Suggests BSgenome.Hsapiens.1000genomes.hs37d5, BSgenome.Hsapiens.UCSC.hg38, BSgenome.Mmusculus.UCSC.mm10, knitr, mSigPlot, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Steve Rozen [aut, cre],
Nanhai Jiang [aut],
Arnoud Boot [aut],
Mo Liu [aut],
Yang Wu [aut],
Mi Ni Huang [aut],
Jia Geng Chang [aut]

Maintainer Steve Rozen <steverozen@pm.me>

Repository CRAN

Date/Publication 2026-09-27 16:40:24 UTC

Contents

`add_seq_context` *Add flanking sequence context to a VCF data frame*

Description

Extracts `seq_context_width` bases upstream and downstream of each variant position from the reference genome and attaches them as a new column named `seq.<N>bases` where `N = 2 * seq_context_width + 1`.

Usage

```
add_seq_context(df, ref_genome, seq_context_width = 10, name_of_vcf = NULL)
```

Arguments

`df`	A VCF as a data frame / data.table with columns `CHROM` and `POS`.
`ref_genome`	A BSgenome object or a character identifier accepted by `normalize_genome_arg()`.
`seq_context_width`	Number of flanking bases on each side (default 10, producing a 21-base window).
`name_of_vcf`	Optional VCF name used in warning/error messages.

Value

`df` with an added character column `seq.<N>bases`.

`add_transcript_strand` *Annotate a VCF data frame with transcript strand information*

Description

For each variant, finds overlapping transcripts in `trans_ranges` via `GenomicRanges::findOverlaps()` with `type = "within"`, and appends columns `trans.start.pos`, `trans.end.pos`, `trans.strand`, `trans.Ensembl.gene.ID`, `trans.gene.symbol`, plus `bothstrand` (TRUE if the variant falls on transcripts from both strands) and `count` (number of overlapping transcripts).

Usage

```
add_transcript_strand(df, ref_genome, trans_ranges = NULL, name_of_vcf = NULL)
```

Arguments

df	A VCF as a data frame / data.table with columns CHROM, POS, ALT.
ref_genome	A BSgenome object or a character identifier accepted by normalize_genome_arg().
trans_ranges	Optional data.table of transcript ranges with columns chrom, start, end, strand, Ensembl.gene.ID, gene.symbol. If NULL, the shipped table for ref_genome is used.
name_of_vcf	Optional VCF name used in warning/error messages.

Details

If trans_ranges is NULL and no shipped table is available for the given ref_genome, returns df as a data.table unchanged (no strand columns added). If df has zero rows, it is returned unchanged.

Value

A data.table with the annotation columns added (left-join semantics: variants outside any transcript have NA values).

---

all.abundance *K-mer abundances for density calculations*

Description

Nested list of k-mer counts keyed by BSgenome.Hsapiens.1000genomes.hs37d5 / BSgenome.Hsapiens.UCSC.hg38 / BSgenome.Mmusculus.UCSC.mm10, then by exome / transcript / genome, then by catalog size as a string ("78", "96", "136", "144", "192", "1536"). Each leaf value is a named integer vector: k-mer → count.

Usage

```
all.abundance
```

Format

An object of class list of length 3.

Details

Used to convert counts to density (mutations per megabase of context). See the planned transform_catalog() function (not yet ported).

Examples

```
all.abundance$BSgenome.Hsapiens.UCSC.hg38$transcript$`144`[1:5]
```

`annotate_id_vcf` *Annotate an in-memory ID (indel) VCF with sequence context, transcript strand, and COSMIC / Koh indel categories*

Description

Port of ICAMS's `AnnotateIDVCF()`. Justifies each indel against the reference genome, optionally annotates transcript strand, and categorizes each justified indel into the COSMIC 83, Koh 89, and Koh 476 schemes.

Usage

```
annotate_id_vcf(
  vcf,
  ref_genome,
  trans_ranges = NULL,
  name_of_vcf = NULL,
  suppress_discarded_variants_warnings = TRUE,
  explain_indels = 1,
  context_width_multiplier = 20L,
  add_transcript_ranges = TRUE
)
```

Arguments

`vcf`	An in-memory ID VCF as a `data.frame` / `data.table`. Expects a "context base" to the left of the indel, e.g. `REF = "ACG"`, `ALT = "A"` (deletion of `CG`) or `REF = "A"`, `ALT = "ACC"` (insertion of `CC`).
`ref_genome`	A BSgenome object or a character alias (e.g. `"GRCh37"`, `"GRCh38"`, `"mm10"`).
`trans_ranges`	Optional transcript-ranges `data.table`. If `NULL` and `add_transcript_ranges = TRUE`, the shipped table is used when available for `ref_genome`.
`name_of_vcf`	Optional name for the VCF, used only in warning / error messages.
`suppress_discarded_variants_warnings`	If `TRUE`, do not warn when variants that cannot be processed are discarded.
`explain_indels`	If `0`, do not explain. If `1`, emit messages for indels whose position was left-shifted by justification. If `2`, emit messages for all indels.
`context_width_multiplier`	Multiplier used to guess how much flanking sequence (per side) is needed to categorize each indel.
`add_transcript_ranges`	If `TRUE`, annotate transcript strand when a transcript-ranges table is available.

Value

A list with:

- `annotated.vcf`: the input VCF with new columns `seq.context`, `seq.context.width`, `pos_shift`, `COSMIC_83`, `Koh_89`, `Koh_476`, and (when transcript ranges are available) `trans.strand` / `bothstrand`.
- `discarded.variants`: rows that could not be justified, or `NULL` if none were discarded.

`annotate_sbs_or_dbs_vcf` *Annotate an SBS or DBS VCF with flanking sequence context and transcript strand*

Description

Adds flanking sequence context via `add_seq_context()` and, when a transcript-ranges table is available, transcript strand via `add_transcript_strand()`. The same pipeline is appropriate for SBS and DBS; DBS-specific validation (e.g. no `N` in the tetranucleotide context) happens downstream in `vcf_to_dbs_catalog()`.

Usage

```
annotate_sbs_or_dbs_vcf(
  vcf,
  ref_genome,
  trans_ranges = NULL,
  seq_context_width = 10L,
  name_of_vcf = NULL
)
```

Arguments

`vcf`	A VCF as a data.frame / data.table with `CHROM`, `POS`, `REF`, `ALT` columns. For SBS, `REF` and `ALT` are single bases; for DBS, both are two bases.
`ref_genome`	A BSgenome object or a character alias accepted by `normalize_genome_arg()`.
`trans_ranges`	Optional transcript-ranges `data.table`. If `NULL`, the shipped table is used when available for `ref_genome`.
`seq_context_width`	Width (per side) of the flanking-sequence window (default 10 → 21-base window).
`name_of_vcf`	Optional VCF name used in warnings.

Value

A list with:

- `annotated.vcf`: the input VCF with new columns `seq.<N>bases` and (when transcript ranges are available) `trans.strand` / `bothstrand`.
- `discarded.variants`: currently always `NULL` for this path (included for symmetry with `annotate_id_vcf()`).

`annot_vcf_to_476_catalog`

Convert an annotated indel VCF to a Koh 476-category catalog

Description

Take an annotated indel VCF data frame (with columns `Koh_476` and `R` as produced by indel classification functions) and produce a single-column data frame of mutation counts in the 476-category Koh classification scheme.

Usage

```
annot_vcf_to_476_catalog(
  annot_vcf,
  sample_id = "no_sample_id_provided",
  FILTER_PASS = TRUE,
  do_message = FALSE,
  clip_le_9 = TRUE
)
```

Arguments

`annot_vcf`	A data frame with at least columns `CHROM`, `POS`, `ALT`, `Koh_476`, and `R` (repeat count). If `FILTER_PASS` is `TRUE`, a `FILTER` column is also required.
`sample_id`	A character string used as the column name in the returned data frame.
`FILTER_PASS`	If `TRUE`, retain only rows where the `FILTER` column equals `"PASS"`.
`do_message`	If `TRUE`, emit diagnostic messages showing row counts at each processing step.
`clip_le_9`	Only keep variants with "R" <= 9, to approximate PCAWG indel calling.

Details

The function:

1. Optionally filters to PASS variants.
2. Removes duplicate positions (warns if ALT alleles differ).
3. Collapses single-base indels with repeat count ≥ 9 into an `"R(9,)"` bin.
4. Tallies counts per Koh 476 category and returns a data frame with one row per category (using `catalog_row_order()$ID476`).

Value

A single-column data frame with 476 rows (one per Koh category) and integer mutation counts. Row names are the Koh 476 category strings; the column name is `sample_id`.

`annot_vcf_to_83_catalog`

Convert an annotated indel VCF to a COSMIC 83-category catalog

---

Description

Take an annotated indel VCF data frame (with column `COSMIC_83` as produced by indel classification functions) and produce a single-column data frame of mutation counts in the 83-category COSMIC ID classification scheme.

Usage

```
annot_vcf_to_83_catalog(
  annot_vcf,
  sample_id = "no_sample_id_provided",
  FILTER_PASS = TRUE,
  do_message = FALSE,
  clip_le_9 = TRUE
)
```

Arguments

`annot_vcf`	A data frame with at least columns `CHROM`, `POS`, `ALT`, and `COSMIC_83`. If `FILTER_PASS` is `TRUE`, a `FILTER` column is also required.
`sample_id`	A character string used as the column name in the returned data frame.
`FILTER_PASS`	If `TRUE`, retain only rows where the `FILTER` column equals `"PASS"`.
`do_message`	If `TRUE`, emit diagnostic messages showing row counts at each processing step.
`clip_le_9`	Only keep variants with "R" <= 9, to approximate PCAWG indel calling.

Details

The function:

1. Optionally filters to PASS variants.
2. Removes duplicate positions (warns if ALT alleles differ).
3. Tallies counts per COSMIC 83 category and returns a data frame with one row per category (using `catalog_row_order()$ID`).

Value

A single-column data frame with 83 rows (one per COSMIC ID category) and integer mutation counts. Row names are the COSMIC 83 category strings; the column name is `sample_id`.

annot_vcf_to_89_catalog *Convert an annotated indel VCF to a Koh 89-category catalog*

Description

Take an annotated indel VCF data frame (with column `Koh_89` as produced by indel classification functions) and produce a single-column data frame of mutation counts in the 89-category Koh classification scheme.

Usage

```
annot_vcf_to_89_catalog(
  annot_vcf,
  sample_id = "no_sample_id_provided",
  FILTER_PASS = TRUE,
  do_message = FALSE,
  clip_le_9 = TRUE
)
```

Arguments

`annot_vcf`	A data frame with at least columns `CHROM`, `POS`, `ALT`, and `Koh_89`. If `FILTER_PASS` is `TRUE`, a `FILTER` column is also required.
`sample_id`	A character string used as the column name in the returned data frame.
`FILTER_PASS`	If `TRUE`, retain only rows where the `FILTER` column equals `"PASS"`.
`do_message`	If `TRUE`, emit diagnostic messages showing row counts at each processing step.
`clip_le_9`	Only keep variants with "R" <= 9, to approximate PCAWG indel calling.

Details

The function:

1. Optionally filters to PASS variants.
2. Removes duplicate positions (warns if ALT alleles differ).
3. Tallies counts per Koh 89 category and returns a data frame with one row per category (using `catalog_row_order()$ID89`).

Value

A single-column data frame with 89 rows (one per Koh category) and integer mutation counts. Row names are the Koh 89 category strings; the column name is `sample_id`.

as_catalog *Turn a numeric matrix into a mutational-spectrum catalog*

Description

Attaches the standard catalog attributes (type, counts_or_density, ref_genome, region, abundance) to x and returns it. **No S3 class is set** — mSigSpectra catalogs are plain matrices with attributes; functions key off attr(x, "type") via explicit checks.

Usage

```
as_catalog(
  x,
  type = NULL,
  ref_genome = NULL,
  region = "unknown",
  abundance = NULL,
  counts_or_density = "counts",
  infer_rownames = FALSE
)
```

Arguments

x	A numeric matrix, data.frame coercible to numeric, or a named numeric vector (converted to a one-column matrix with the vector names as rownames).
type	Optional catalog type identifier ("SBS96", "DBS78", "ID83", etc.). Inferred from nrow(x) if NULL.
ref_genome	Optional BSgenome object or alias; recorded as an attribute and used for abundance lookup.
region	One of "genome", "exome", "transcript", "unknown". For stranded catalogs (SBS192, DBS144) "genome" is silently promoted to "transcript".
abundance	Optional named numeric vector of k-mer counts. If NULL, inferred from shipped all.abundance keyed on ref_genome and region.
counts_or_density	One of "counts", "density", "counts.signature", "density.signature".
infer_rownames	If TRUE and x has no rownames, the canonical rownames for the catalog type are attached (**assuming the row order is already correct**). If FALSE, x must already have the canonical rownames.

Value

x as a numeric matrix with attributes set.

Examples

```
m <- matrix(
  1, nrow = 96, ncol = 1,
  dimnames = list(catalog_row_order()$SBS96, "sample1")
)
cat96 <- as_catalog(m)
attr(cat96, "type")   # "SBS96"
attr(cat96, "region") # "unknown"
```

`catalog_attrs` *Report the attributes of an mSigSpectra catalog*

Description

Report the attributes of an mSigSpectra catalog

Usage

```
catalog_attrs(x)
```

Arguments

`x`	A catalog.

Value

A named list with elements `type`, `counts_or_density`, `ref_genome`, `region`, `abundance`.

`catalog_row_order` *Return catalog row orders for all supported catalog types*

Description

Returns a named list containing the canonical row ordering for each catalog type. These are used for validation and ordering of mutation catalogs.

Usage

```
catalog_row_order()
```

Details

Row names use a compact 4-letter format for SBS types: e.g. `ACAA` encodes the trinucleotide context as `<5' base><ref><3' base><alt>`. SBS288 row names add a strand prefix: `T:ACAA` (transcribed), `U:ACAA` (untranscribed), `N:ACAA` (non-transcribed).

Value

A named list with elements: SBS96, SBS192, SBS288, SBS1536, DBS78, DBS136, DBS144, ID (83-category COSMIC indels), ID166, ID89, ID476.

Examples

```
cro <- catalog_row_order()
head(cro$SBS96)
length(cro$DBS78)
```

---

categorize_1_justified_indel

Given a indel and its sequence context, categorize it

Description

This function is primarily for internal use, but we export it to document the underlying logic.

Usage

```
categorize_1_justified_indel(
  context,
  ins_or_del,
  ins_or_del_seq,
  pos,
  chrom = NULL,
  genomic_pos = NULL
)
```

Arguments

context	The sequence surrounding the indel PRIOR to the insertion or deletion.
ins_or_del	A single character, with "i" denoting an insertion and "d" denoting a deletion.
ins_or_del_seq	The sequence that was inserted or deleted.
pos	For deletions, the 1-based position of the start of the deleted sequence; for insertions, the position immediately to the right of where the insertion occurs.
chrom	Optional chromosome name; used only to enrich error messages when the deletion sequence does not match the context.
genomic_pos	Optional genomic position; used only to enrich error messages when the deletion sequence does not match the context.

Details

See https://github.com/steverozen/ICAMS/blob/v3.0.9-branch/data-raw/PCAWG7_indel_classification_2021_09_03.xlsx for additional information on deletion mutation classification.

This function first handles deletions in homopolymers, then handles deletions in simple repeats with longer repeat units (e.g. CACACACA), and if the deletion is not in a simple repeat, looks for microhomology.

Value

A string that is the canonical representation of the given deletion type. Return NA and raise a warning if there is an un-normalized representation of the deletion of a repeat unit. See FindDelMH for details. (This seems to be very rare.)

Examples

```
simplify = function(ll) unlist(ll[c("COSMIC_83", "Koh_89", "Koh_476")])
categorize_1_justified_indel("GGAAAGG", "d", ins_or_del_seq = "A", pos = 3) # "DEL:T:1:2"
categorize_1_justified_indel("GGAAAGG", "d", ins_or_del_seq = "A", pos = 4) # "DEL:T:1:2"
simplify(
  categorize_1_justified_indel("TTATT", "d", ins_or_del_seq = "A", pos = 3))
simplify(
  categorize_1_justified_indel("TTATATAT", "d", ins_or_del_seq = "TATA", pos = 2))
simplify(
  categorize_1_justified_indel("TTATATAT", "d", ins_or_del_seq = "TATAT", pos = 2))
```

---

cbind_catalogs *Combine catalogs across samples (column-bind)*

Description

Checks that all input catalogs share the same type, counts_or_density, ref_genome, and region, then cbinds their matrices and re-applies the shared attributes.

Usage

```
cbind_catalogs(catalogs)
```

Arguments

catalogs	A list of catalogs.

Value

A single catalog with ncol equal to the sum of the input ncols.

change_476_type_ids_to_open_intervals

Change 476-type indel category identifiers to use right-open repeat intervals

Description

Replaces bounded repeat-count suffixes like `:R(X,9)` with right-open equivalents like `:R(X,)`, making the classification agnostic as to whether repeats longer than 9 were discarded from the input data.

Usage

```
change_476_type_ids_to_open_intervals(type_476_indel_type_identifiers)
```

Arguments

`type_476_indel_type_identifiers`
: Character vector of 476-type indel category identifiers, e.g. as returned by `categorize_indels_in_vcf()`.

Value

Character vector the same length as the input, with `:R(X,9)` at the end of each string replaced by `:R(X,)`.

Examples

```
change_476_type_ids_to_open_intervals(
  c("Del(C):Ins(C):R(5,9)", "Del(T):R(3,5)", "Ins(C):R(5,9)")
)
```

change_89_type_ids_to_open_intervals

Change 89-type indel category identifiers to use right-open repeat intervals

Description

Replaces bounded repeat-count suffixes like `R(5,9)` with right-open equivalents like `R(5,)`, making the classification agnostic as to whether repeats longer than 9 were discarded from the input data.

Usage

```
change_89_type_ids_to_open_intervals(type_89_indel_type_identifiers)
```

Arguments

type_89_indel_type_identifiers
: Character vector of 89-type indel category identifiers.

Value

Character vector the same length as the input, with bounded repeat suffixes replaced by right-open equivalents.

Examples

```
change_89_type_ids_to_open_intervals(
  c("Ins(2,):R(5,9)", "Ins(C):R(7,9)", "[Del(T):R(8,9)]",
    "[Ins(T):R(8,9)]", "Del(2,):U(1,2):R(5,9)", "Del(3,):U(3,):R(3,9)")
)
```

---

check_and_remove_discarded_variants

Check a VCF for common variant-level problems and remove the offenders

---

Description

Removes:

- Rows with identical REF and ALT.
- Stray `#CHROM` header repeats.
- Duplicated (CHROM, POS, REF, ALT) rows (keeping one copy).
- Multiple-ALT rows (comma-separated ALT field).
- Non-standard chromosome names (or those outside `chr_names_to_process` when supplied).
- Substitutions of length > 2 (e.g. `ACT>TGA`).
- Complex indels (`REF[1] != ALT[1]`).
- Wrong DBS rows where REF and ALT share a base at the same position.
- Variants whose REF base is not in `{A, C, G, T}`.

Usage

```
check_and_remove_discarded_variants(
  vcf,
  name_of_vcf = NULL,
  chr_names_to_process = NULL
)
```

Arguments

vcf	A VCF as a data.frame / data.table.
name_of_vcf	Optional name, used in warning messages.
chr_names_to_process	Optional character vector of chromosome names to keep (overrides the default non-standard-contig filter).

Value

A list with element df, the retained rows of vcf (same class as the input), and, only when at least one row was removed, element discarded.variants, a data.table of the removed rows with an added character column discarded.reason explaining why each row was discarded.

---

collapse_catalog *Collapse a higher-resolution catalog to a lower-resolution one*

Description

Supports:

- SBS1536 -> SBS96: collapse pentanucleotide → trinucleotide contexts.
- SBS192 -> SBS96: sum across the two transcript strands.
- DBS144 -> DBS78: sum across the two transcript strands.

Usage

```
collapse_catalog(catalog, to = c("SBS96", "DBS78"))
```

Arguments

catalog	An mSigSpectra catalog.
to	Target catalog type ("SBS96", "DBS78").

Value

A new catalog with the collapsed rows.

infer_trans_ranges *Infer transcript ranges for a reference genome*

Description

If `trans_ranges` is supplied, it is returned unchanged. Otherwise, returns the shipped transcript-ranges table for the given reference genome (GRCh37 / GRCh38 / GRCm38).

Usage

```
infer_trans_ranges(ref_genome, trans_ranges = NULL)
```

Arguments

`ref_genome`	A BSgenome object or a character identifier accepted by `normalize_genome_arg()`.
`trans_ranges`	Optional user-supplied transcript ranges.

Value

A `data.table` of transcript ranges, or `NULL` if no shipped table is available and none was supplied.

is_catalog *Check whether an object looks like an mSigSpectra catalog*

Description

Returns `TRUE` if `x` is a numeric matrix with the five catalog attributes (`type`, `counts_or_density`, `ref_genome`, `region`, `abundance`) and canonical rownames for its type.

Usage

```
is_catalog(x)
```

Arguments

`x`	Any R object.

Value

A single logical value: `TRUE` if `x` is a numeric matrix carrying the catalog attributes with the canonical row names for its `type`, otherwise `FALSE`.

`justify_id_vcf` *Add sequence context and transcript information to an in-memory ID (insertion/deletion) VCF, and confirm that they match the given reference genome*

Description

Add sequence context and transcript information to an in-memory ID (insertion/deletion) VCF, and confirm that they match the given reference genome

Usage

```
justify_id_vcf(
  ID.vcf,
  ref.genome,
  name.of.VCF = NULL,
  suppress.discarded.variants.warnings = TRUE,
  explain_indels = 1,
  context_width_multiplier = 20L
)
```

Arguments

`ID.vcf`	An in-memory ID (insertion/deletion) VCF as a `data.frame`. This function expects that there is a "context base" to the left, for example REF = ACG, ALT = A (deletion of CG) or REF = A, ALT = ACC (insertion of CC).
`ref.genome`	Can be a string or a BSgenome. If a string, it should a well-known name for reference genome in BSgenome
`name.of.VCF`	Name of the VCF file.
`suppress.discarded.variants.warnings`	If TRUE, do warn when variants that cannot be processed are discarded.
`explain_indels`	If 0, do not explain, if 1, explain indels that were justified (via messages), if 2, generate messages for all indels.
`context_width_multiplier`	Used to guess how much sequence on each side of an indel is needed to categorize it.

Value

A list of elements:

- `annotated.vcf`: The original VCF data frame in which `POS`, `REF` and `ALT` with new columns added to the input data frame:
  - `seq.context`: The sequence embedding the variant.
  - `seq.context.width`: The width of `seq.context` to the left of the indel
  - `pos_shift`

- `discarded.variants`: **Non-NULL only if** there are variants that were excluded from the analysis. See the added extra column `discarded.reason` for more details.

Examples

```
# See tests/testthat/test_indel_classification.R for an end-to-end example
# against a real Strelka ID VCF.
```

`justify_indel` *Move the notional position of a deletion or insertion as far left as possible.*

Description

short_string should be generated by a deletion in long-str at (1-based) position pos. If supplied, the expected_delta should be the deleted sequence. If expected_delta is not NULL, the function checks this and aborts if this is not the case.

Dually, pos is the 1-based position in short string in front o which one can make an insertion to get long_str.

This function moves pos as far to left as possible so that a deletion at that position still results in an edit of long_str to generate short_str.

Usage

```
justify_indel(long_str, short_str, pos, expected_delta = NULL)
```

Arguments

`long_str`	A single character string
`short_str`	A single character string.
`pos`	An integer; see the description
`expected_delta`	A single string; see the description

Value

A list with elements

- leftmost_pos
- del_string
- del_string_plus del_string plus one base to the left the position of that base to the left is leftmost_pos - 1
- edge_warning See the third example. We are not sure how far to the left we could have moved the indel if we had more context on the left.
- error if non-NULL we did not see the expected value for expected_delta

Examples

```
justify_indel("CAAAG", "CAAG", pos = 2, expected_delta = "A")
justify_indel("CAAAG", "CAAG", pos = 3, expected_delta = "A")
justify_indel("CACAG", "CAG", pos = 3, expected_delta = "CA")
```

---

read_catalog *Read a mutational-spectrum catalog from a file*

Description

Auto-detects the file format from its first row. Recognized formats:

- **ICAMS native** CSV — first N columns are header columns (mutation type, context, etc), remaining columns are per-sample counts / densities.
- **SigProfiler** TSV / CSV — single header column with bracketed mutation labels like `A[C>A]A`, `AA[C>A]AA`, or PCAWG indel codes like `1:Del:C:0`.
- **COSMIC** CSV — SBS96 uses the ICAMS-external row-header format; SBS192 uses a stranded variant of the same; DBS78 / ID83 also share their SigProfiler-style layout.

Usage

```
read_catalog(
  file,
  ref_genome = NULL,
  region = "unknown",
  counts_or_density = "counts",
  format = c("auto", "ICAMS", "SigProfiler", "COSMIC")
)
```

Arguments

`file`	Path to the catalog file.
`ref_genome`	Optional BSgenome object or alias; stored as attribute.
`region`	One of `"genome"`, `"exome"`, `"transcript"`, `"unknown"`.
`counts_or_density`	One of `"counts"`, `"density"`, `"counts.signature"`, `"density.signature"`.
`format`	`"auto"` (default), `"ICAMS"`, `"SigProfiler"`, or `"COSMIC"`. In `"auto"` mode the format is inferred from the file's first data row.

Details

The matrix is reordered to the canonical rownames for its type and wrapped in a catalog via `as_catalog()`.

Value

A catalog matrix with attributes (see `as_catalog()`).

read_vcf *Read a VCF file into a data.table, caller-agnostically*

Description

Reads the *body* of a VCF file (lines after #CHROM) into a data.table. The resulting table has whatever columns the VCF has (CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, and optionally FORMAT plus one or more sample columns), with #CHROM renamed to CHROM.

Usage

```
read_vcf(file, filter = TRUE, name_of_vcf = NULL)
```

Arguments

file	Path or URL to the VCF file.
filter	Controls which rows are kept based on the FILTER column: • TRUE (default): keep rows where FILTER %in% c("PASS", ".", "") — the union of passing values across common callers. • FALSE or NULL: keep all rows. • A character vector: keep rows where FILTER %in% filter. Rows with no FILTER column are always kept; a warning is emitted when filter is non-trivial in that case.
name_of_vcf	Optional name for the VCF, used only for warning / error messages. Defaults to the filename with extension stripped.

Details

Caller-agnostic. read_vcf() does not know or care which variant caller produced the VCF. It does not parse FORMAT/sample columns and does not extract VAF or read depth. The only caller-dependent semantics is the default value of the filter argument (see below).

Uses data.table::fread() with check.names=FALSE, na.strings = ""``, fill = TRUE‘, to parse the VCF body. Handles uncompressed and gzipped files; does not handle bgzipped/tabix.

Value

A data.table with one row per variant. The name of the first column is 'CHROM', not '#CHROM'. Other column names

read_vcfs *Read multiple VCF files*

Description

Thin batch wrapper around `read_vcf()`.

Usage

```
read_vcfs(files, ..., names_of_vcfs = NULL)
```

Arguments

`files`	Character vector of file paths / URLs.
`...`	Passed through to `read_vcf()`.
`names_of_vcfs`	Optional character vector of names (same length as `files`). Defaults to stripping extensions from basenames.

Value

A named list of `data.tables`, one per input file.

segment_simple_cpp *Segment a single indel using Rcpp interface*

Description

This function provides direct C++ interface for indel segmentation without calling an external binary via system2.

Usage

```
segment_simple_cpp(ins_or_del, string, context)
```

Arguments

`ins_or_del`	Character indicating insertion ("i") or deletion ("d")
`string`	A single indel sequence to segment (character scalar)
`context`	A single flanking context sequence (character scalar)

Value

A list with the following elements:

`unit`	Repeat unit sequence (character)
`unit_length`	Unit length (integer)
`internal_rep`	Internal repeat region (character)
`internal_reps`	Internal repeat count (integer)
`spacer`	Spacer sequence (character)
`spacer_length`	Spacer length (integer)
`prime3_rep`	3' flanking repeat region (character)
`prime3_reps`	3' flanking repeat count (integer)
`original_reps`	Original repeat count (integer)

`seg_simple` *Segment a single indel sequence using Rcpp interface*

Description

Segment a single indel sequence using Rcpp interface

Usage

```
seg_simple(ins_or_del, string, context)
```

Arguments

`ins_or_del`	Character indicating insertion ("i") or deletion ("d")
`string`	A single indel sequence to segment (character scalar)
`context`	A single flanking context sequence (character scalar)

Details

This function segments an indel sequence by finding the optimal repeat unit that best explains the sequence structure. The algorithm tries all possible repeat unit sizes and selects the best segmentation based on:

- Highest 3' flanking repeat count (most important)
- Highest internal repeat count
- Lowest spacer length (prefer clean repeats)
- Lowest unit length (prefer simpler units)

Value

A list with the following elements:

`unit`	Repeat unit sequence (character)
`unit_length`	Unit length (integer)
`internal_rep`	Internal repeat region (character)
`internal_reps`	Internal repeat count (integer)
`spacer`	Spacer sequence (character)
`spacer_length`	Spacer length (integer)
`prime3_rep`	3' flanking repeat region (character)
`prime3_reps`	3' flanking repeat count (integer)
`original_reps`	Original repeat count (integer)

Examples

```
# Simple AT repeat
result <- seg_simple("d", "ATATAT", "ATATGG")
print(result)

# CG repeat
result <- seg_simple("d", "CGCGCG", "CGCGAA")
print(result)
```

---

split_vcf *Split a mixed-mutation VCF into SBS / DBS / ID sub-tables*

Description

Classifies each row by `REF`/`ALT` length alone:

- SBS (single base substitution): `nchar(REF) == 1 && nchar(ALT) == 1`
- DBS (double base substitution): `nchar(REF) == 2 && nchar(ALT) == 2`
- ID (indel): `nchar(REF) != nchar(ALT)`
- Other (e.g. 3+bp substitutions): discarded with a recorded reason.

Usage

```
split_vcf(vcf, name_of_vcf = NULL)
```

Arguments

`vcf`	A VCF as a data.frame / data.table with at least `REF` and `ALT` columns.
`name_of_vcf`	Optional VCF name used in warning / error messages.

Details

This is **caller-agnostic**. In particular, mSigSpectra does *not* merge adjacent SBSs into DBSs based on VAF similarity (which ICAMS did via `SplitOneVCF` for Strelka-style VCFs). Users who want that behavior should apply it as a post-processing step with their own VAF column.

Value

A list with elements

- `SBS`: `data.table` of SBS rows.
- `DBS`: `data.table` of DBS rows.
- `ID`: `data.table` of indel rows.
- `discarded`: `data.table` of rows that did not fit any of the above, with a `discarded.reason` column — `NULL` **when no rows were discarded**.

`subset_catalog` *Subset a catalog while preserving attributes*

Description

Base `[` on a matrix drops attributes other than `dim` / `dimnames`. Use `subset_catalog()` when you want to keep the catalog's type / ref_genome / region / abundance metadata through a subset operation.

Usage

```
subset_catalog(x, rows = NULL, cols = NULL)
```

Arguments

`x`	A catalog.
`rows, cols`	Numeric, logical, or character indices (see base::Extract). If `NULL`, all rows / columns are kept.

Value

A catalog (numeric matrix) containing the selected rows and columns, with the `type`, `counts_or_density`, `ref_genome`, and `region` attributes of `x` preserved. The `abundance` attribute is preserved only when all rows are kept, because it is not meaningful for a subset of mutation types.

trans.ranges *Transcript ranges for transcriptional strand annotation*

Description

Precomputed transcript ranges (one row per gene) used by `add_transcript_strand()` to determine the coding strand for each variant. Sources: GENCODE v30 (human) and vM21 (mouse). Only genes with CCDS IDs are retained.

Usage

```
trans.ranges.GRCh37

trans.ranges.GRCh38

trans.ranges.GRCm38
```

Format

A data.table::data.table with columns `chrom`, `start`, `end`, `strand`, `Ensembl.gene.ID`, `gene.symbol`. One-based coordinates.

An object of class `data.table` (inherits from `data.frame`) with 19083 rows and 6 columns.

An object of class `data.table` (inherits from `data.frame`) with 19096 rows and 6 columns.

An object of class `data.table` (inherits from `data.frame`) with 20325 rows and 6 columns.

Source

https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/GRCh37_mapping/gencode.v30lift37.annotation.gff3.gz

https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_human/release_30/gencode.v30.annotation.gff3.gz

https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M21/gencode.vM21.annotation.gff3.gz

transform_catalog *Transform a catalog between counts and density*

Description

Converts between counts (raw mutation counts per category) and density (mutations per megabase of context) representations, and between catalog <-> signature (column-normalized) forms. The transformation is expressed by multiplying each category's count by `target_abundance[source.n.mer] / source_abundance[source.n.mer]`, where `source.n.mer` is the reference context encoded in the row name (first 3 characters for SBS96/192, 5 for SBS1536, 2 for DBS78/144, 4 for DBS136).

Usage

```
transform_catalog(
  catalog,
  target_ref_genome = NULL,
  target_region = NULL,
  target_counts_or_density = NULL,
  target_abundance = NULL
)
```

Arguments

catalog	An mSigSpectra catalog (see as_catalog()) with attributes type, counts_or_density, ref_genome, region, abundance.
target_ref_genome, target_region, target_counts_or_density	Target attributes. If NULL, the catalog's current attribute is used.
target_abundance	Optional target abundance vector. If NULL, inferred from target_ref_genome + target_region.

Value

A new catalog (plain matrix with attributes).

---

vcf_to_dbs_catalog *Build a DBS mutational-spectrum catalog from an annotated DBS VCF*

Description

Build a DBS mutational-spectrum catalog from an annotated DBS VCF

Usage

```
vcf_to_dbs_catalog(
  annotated_vcf,
  type = c("DBS78", "DBS136", "DBS144"),
  ref_genome = NULL,
  region = "unknown",
  sample_name = "count"
)
```

Arguments

annotated_vcf	A DBS VCF annotated by annotate_sbs_or_dbs_vcf(). May be the bare annotated data.table or the full list(annotated.vcf, discarded.variants) returned by the annotator. Must contain a seq.<N>bases column; for type = "DBS144" also requires trans.strand / bothstrand.

type	One of "DBS78", "DBS136", "DBS144".
ref_genome	Optional BSgenome object or alias; recorded on the output catalog.
region	One of "genome", "exome", "transcript", "unknown".
sample_name	Column name for the single-sample catalog matrix.

Value

A single-column numeric matrix with catalog attributes (see as_catalog()).

---

vcf_to_id_catalog *Build an ID (indel) mutational-spectrum catalog from an annotated ID VCF*

Description

Turns an indel-annotated VCF (with COSMIC_83 / Koh_89 / Koh_476 columns as produced by annotate_id_vcf()) into a count matrix for the requested ID classification scheme.

Usage

```
vcf_to_id_catalog(
  annotated_vcf,
  type = c("ID83", "ID89", "ID476"),
  ref_genome = NULL,
  region = "unknown",
  sample_name = "count",
  FILTER_PASS = TRUE,
  clip_le_9 = TRUE
)
```

Arguments

annotated_vcf	An ID VCF annotated by annotate_id_vcf(). May be the bare annotated data.table or the full list(annotated.vcf, discarded.variants) returned by the annotator. Must contain the categorization column corresponding to type.
type	One of "ID83", "ID89", "ID476".
ref_genome	Optional BSgenome object or alias; recorded on the output catalog.
region	One of "genome", "exome", "transcript", "unknown".
sample_name	Column name for the single-sample catalog matrix.
FILTER_PASS	If TRUE, retain only rows where the VCF FILTER column is "PASS".
clip_le_9	If TRUE, drop variants with repeat count R > 9, approximating PCAWG indel calling.

Value

A single-column numeric matrix with catalog attributes (see as_catalog()).

vcf_to_sbs_catalog *Build an SBS mutational-spectrum catalog from an annotated SBS VCF*

Description

Returns a single catalog matrix of the requested `type`. Intermediate matrices for the other SBS resolutions are still computed (cheap) but not returned, keeping the public API focused on "one call, one catalog type".

Usage

```
vcf_to_sbs_catalog(
  annotated_vcf,
  type = c("SBS96", "SBS192", "SBS1536"),
  ref_genome = NULL,
  region = "unknown",
  sample_name = "count"
)
```

Arguments

`annotated_vcf`	An SBS VCF annotated by `annotate_sbs_or_dbs_vcf()`. May be the bare annotated `data.table` or the full `list(annotated.vcf, discarded.variants)` returned by the annotator. Must contain a `seq.<N>bases` column; for `type = "SBS192"` also requires `trans.strand` / `bothstrand`.
`type`	One of `"SBS96"`, `"SBS192"`, `"SBS1536"`.
`ref_genome`	Optional BSgenome object or alias; recorded on the output catalog.
`region`	One of `"genome"`, `"exome"`, `"transcript"`, `"unknown"`.
`sample_name`	Column name for the single-sample catalog matrix.

Value

A single-column numeric matrix with catalog attributes (see `as_catalog()`).

write_catalog *Write a mutational-spectrum catalog to a file*

Description

Writes `catalog` in ICAMS-native CSV format (the only format currently supported for writing). The row-header columns that precede the sample columns are taken from the shipped `catalog.row.headers` object for the catalog's type.

Usage

```
write_catalog(
  catalog,
  file,
  format = c("ICAMS", "SigProfiler", "COSMIC"),
  sep = ","
)
```

Arguments

`catalog`	A catalog (see `as_catalog()`).
`file`	Output path.
`format`	Output format. Currently only `"ICAMS"` is supported; `"SigProfiler"` and `"COSMIC"` error with an informative message.
`sep`	Column separator. Defaults to `","` for ICAMS format.

Value

`file` invisibly.

Index