# Package 'transmogR'

March 7, 2025

**Type** Package

**Title** Modify a set of reference sequences using a set of variants

**Version** 1.2.0

**Description** transmogR provides the tools needed to crate a new reference
genome or reference transcriptome, using a set of variants. Variants can
be any combination of SNPs, Insertions and Deletions. The intended
use-case is to enable creation of variant-modified reference
transcriptomes for incorporation into transcriptomic pseudo-alignment
workflows, such as salmon.

**License** GPL-3

**Encoding** UTF-8

**URL** https://github.com/smped/transmogR

**BugReports** https://github.com/smped/transmogR/issues

**Depends** Biostrings, GenomicRanges

**Imports** BSgenome, dplyr, GenomeInfoDb, GenomicFeatures, ggplot2 (>=
3.5.0), IRanges, jsonlite, matrixStats, methods, parallel,
rlang, scales, stats, S4Vectors, SummarizedExperiment,
VariantAnnotation, vroom

**Suggests** BiocStyle, BSgenome.Hsapiens.UCSC.hg38, ComplexUpset,
extraChIPs, InteractionSet, knitr, rmarkdown, rtracklayer,
testthat (>= 3.0.0)

**biocViews** Alignment, GenomicVariation, Sequencing,
TranscriptomeVariant

**BiocType** Software

**VignetteBuilder** knitr

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

**Config/testthat/edition** 3

**git_url** https://git.bioconductor.org/packages/transmogR

**git_branch** RELEASE_3_20

**git_last_commit** a83ed92

**git_last_commit_date** 2024-10-29

**Repository** Bioconductor 3.20

**Author** Stevie Pederson [aut, cre] (<<https://orcid.org/0000-0001-8197-3303>>)

**Maintainer** Stevie Pederson <stephen.pederson.au@gmail.com>

# Contents

---

transmogR-package | *transmogR: Create a variant-modified reference transcriptome*

---

### Description

The package transmogR has been designed for creation of a variant-modified reference transcriptome

### Details

The package transmogR provides two primary functions for modifying complete transcriptomes or genomes:

- transmogrify() for incorporating the supplied variants into transcriptomic sequences, and
- genomogrify() for incorporating the supplied variants into genomic sequences, ideally to be passed as decoy sequences to a tool such as salmon.

The main functions rely on lower-level functions such as:

- owl() which over-writes letters (i.e. SNPs) within a sequence, and
- indelcator() which incorporates InDels into an individual sequence

Additional utility functions are provided which allow characterisation and exploration of any set of variants:

- overlapsByVar() counts the variants which overlap sets of GenomicRanges, first splitting the variants into SNV, Insertions and Deletions
- parY() returns the pseudo-autosomal regions for a chosen genome build as a GenomicRanges object
- upsetVarByCol() produces an UpSet plot counting how many unique IDs are impacted by a set o variants. IDs can represent any column in the supplied ranges, such as gene_id or transcript_id
- varTypes() classifies a set of variants into SNV, Insertions of Deletions

## Author(s)

Stevie Pederson

## See Also

Useful links:

- <https://github.com/smped/transmogR>
- Report bugs at <https://github.com/smped/transmogR/issues>

---

| digestSalmon | *Parse the output from salmon* |

---

## Description

Parse transcript counts and additional data from salmon

## Usage

```
digestSalmon(
  paths,
  max_sets = 2L,
  aux_dir = "aux_info",
  name_fun = basename,
  verbose = TRUE,
  length_as_assay = FALSE,
  ...
)
```

## Arguments

| | |
|---|---|
| paths | Vector of file paths to directories containing salmon results |
| max_sets | The maximum number of indexes permitted |
| aux_dir | Subdirectory where bootstraps and meta_info.json are stored |
| name_fun | Function applied to paths to provide colnames in the returned object. Set to NULL or c() to disable. |
| verbose | Print progress messages |
| length_as_assay | |
| | Output transcript lengths as an assay. May be required if using separate reference transcriptomes for different samples |
| ... | Not used |

## Details

This function is based heavily on edgeR::catchSalmon() with some important exceptions:

1. A SummarizedExperiment object is returned
2. Differing numbers of transcripts are allowed between samples

The second point is intended for the scenario where some samples may have been aligned to a full reference, with remaining samples aligned to a partially masked reference (e.g. chrY). This will lead to differing numbers of transcripts within each salmon index, however, common estimates of overdispersions are required for scaling transcript-level counts. By default, the function will error if >2 different sets of transcripts are detected, however this can be modified using the max_sets argument.

The SummarizedExperiment object returned will also contain multiple assays, as described below

## Value

A SummarizedExperiment object containing assays for counts, scaledCounts, TPM and effective-Length. The scaledCounts assay contains counts divided by overdispersions. rowData in the returned object will also include transcript-lengths along with the overdispersion estimates used to return the scaled counts.

---

| genomogrify | *Mogrify a genome using a set of variants* |

---

## Description

Use a set of SNPS, insertions and deletions to modify a reference genome

## Usage

```
genomogrify(x, var, ...)

## S4 method for signature 'XStringSet,GRanges'
genomogrify(
  x,
  var,
  alt_col = "ALT",
  mask = GRanges(),
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
  var_sep = "_",
  verbose = TRUE,
  ...
)

## S4 method for signature 'BSgenome,GRanges'
genomogrify(
  x,
  var,
  alt_col = "ALT",
  mask = GRanges(),
  names,
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
  var_sep = "_",
```

```
  verbose = TRUE,
  ...
)

## S4 method for signature 'BSgenome,VcfFile'
genomogrify(
  x,
  var,
  alt_col = "ALT",
  mask = GRanges(),
  names,
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
  var_sep = "_",
  which,
  verbose = TRUE,
  ...
)

## S4 method for signature 'XStringSet,VcfFile'
genomogrify(
  x,
  var,
  alt_col = "ALT",
  mask = GRanges(),
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
  var_sep = "_",
  which,
  verbose = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| x | A DNAStringSet or BSgenome |
| var | GRanges object containing the variants, or a [VariantAnnotation::VcfFile](#) |
| ... | Passed to [parallel::mclapply](#) |
| alt_col | The name of the column with var containing alternate bases |
| mask | Optional GRanges object defining regions to be masked with an 'N' |
| tag | Optional tag to add to all sequence names which were modified |
| sep | Separator to place between seqnames names & tag |
| var_tags | logical(1) Add tags indicating which type of variant were incorporated, with 's', 'i' and 'd' representing SNPs, Insertions and Deletions respectively |
| var_sep | Separator between any previous tags and variant tags |
| verbose | logical(1) Print progress messages while running |
| names | Sequence names to be mogrified |
| which | GRanges object passed to [VariantAnnotation::ScanVcfParam](#) if using a VCF directly |

## Details

This function is designed to create a variant-modified reference genome, intended to be included as a set of decoys when using salmon in selective alignment mode. Sequence lengths will change if InDels are included and any coordinate-based information will be lost on the output of this function.

Tags are able to be added to any modified sequence to assist identifying any changes that have been made to a sequence.

## Value

XStringSet with variant modified sequences

## Examples

```
library(GenomicRanges)
dna <- DNAStringSet(c(chr1 = "ACGT", chr2 = "AATTT"))
var <- GRanges(c("chr1:1", "chr1:3", "chr2:1-3"))
var$ALT <- c("C", "GG", "A")
dna
genomogrify(dna, var)
genomogrify(dna, var, tag = "mod")
genomogrify(dna, var, var_tags = TRUE)
genomogrify(dna, var, mask = GRanges("chr2:1-5"), var_tags = TRUE)
```

---

indelcator                    *Substitute InDels into one or more sequences*

---

## Description

Modify one or more sequences to include Insertions or Deletions

## Usage

```
indelcator(x, indels, ...)

## S4 method for signature 'XString,GRanges'
indelcator(x, indels, exons, alt_col = "ALT", ...)

## S4 method for signature 'DNAStringSet,GRanges'
indelcator(x, indels, alt_col = "ALT", mc.cores = 1, verbose = TRUE, ...)

## S4 method for signature 'BSgenome,GRanges'
indelcator(x, indels, alt_col = "ALT", mc.cores = 1, names, ...)
```

## Arguments

| | |
|---|---|
| x | Sequence of class XString |
| indels | GRanges object with InDel locations and the alternate allele |
| ... | Passed to parallel::mclapply |
| exons | GRanges object containing exon structure for x |

| | |
|---|---|
| alt_col | Column containing the alternate allele |
| mc.cores | Number of cores to use when calling parallel::mclapply internally |
| verbose | logical(1) Print all messages |
| names | passed to BSgenome::getSeq when x is a BSgenome object |

## Details

This is a lower-level function relied on by both transmogrify() and genomogrify().

Takes an Biostrings::XString or Biostrings::XStringSet object and modifies the sequence to incorporate InDels. The expected types of data determine the behaviour, with the following expectations describing how the function will incorporate data

| Input Data Type | Exons Required | Use Case | Returned |
|---|---|---|---|
| XString | Y | Modify a Reference Transcriptome | XString |
| DNAStringSet | N | Modify a Reference Genome | DNAStringSet |
| BSgenome | N | Modify a Reference Genome | DNAStringSet |

## Value

A DNAStringSet or XString object (See Details)

## See Also

transmogrify() genomogrify()

## Examples

```
## Start with a DNAStringSet
library(GenomicRanges)
seq <- DNAStringSet(c(seq1 = "AATCTGCGC"))
## Define an Insertion
var <- GRanges("seq1:1")
var$ALT <- "AAA"
seq
indelcator(seq, var)

## To modify a single transcript
library(GenomicFeatures)
ex <- GRanges(c("seq1:1-3:+", "seq1:7-9:+"))
orig <- extractTranscriptSeqs(seq, GRangesList(tx1 = ex))[["tx1"]]
orig
indelcator(orig, var, exons = ex)
```

---

overlapsByVar *Count overlaps by variant type*

---

### Description

Count how many variants of each type overlap ranges

### Usage

```
overlapsByVar(x, var, ...)

## S4 method for signature 'GRangesList,GRanges'
overlapsByVar(x, var, alt_col = "ALT", ...)

## S4 method for signature 'GRanges,GRanges'
overlapsByVar(x, var, alt_col = "ALT", ...)
```

### Arguments

| | |
|---|---|
| x | A GRangesList with features of interest |
| var | A Granges object with variants of interest |
| ... | Passed to [rowSums](#) |
| alt_col | The column within mcols(var) which contains the alternate allele |

### Details

Taking any GRanges or GRangesList, count how many of each variant type overlap a region.

### Value

A vector or matrix

### Examples

```
library(rtracklayer)
library(VariantAnnotation)
gtf <- import.gff(
    system.file("extdata/gencode.v44.subset.gtf.gz", package = "transmogR")
)
grl <- splitAsList(gtf, gtf$type)
vcf <- system.file("extdata/1000GP_subset.vcf.gz", package = "transmogR")
var <- rowRanges(readVcf(vcf, param = ScanVcfParam(fixed = "ALT")))
overlapsByVar(grl, var)
```

---

owl                           *OverWrite Letters in an XStringSet*

---

### Description

OverWrite Letters (e.g. SNPs) in an XStringSet

### Usage

```
owl(seq, snps, ...)

## S4 method for signature 'XStringSet,GRanges'
owl(seq, snps, alt_col = "ALT", ...)

## S4 method for signature 'BSgenome,GRanges'
owl(seq, snps, alt_col = "ALT", names, ...)
```

### Arguments

| | |
|---|---|
| seq | A BSgenome, DNAStringSet, RNAStringSet or other XStringSet. |
| snps | A GRanges object with SNP positions and a column containing the alternate allele |
| ... | Passed to Biostrings::replaceLetterAt() |
| alt_col | Column name in the mcols element of snps containing the alternate allele |
| names | Sequence names to operate on |

### Details

This is a lower-level function called by transmogrify() and genomogrify(), but able to be called by the user if needed

Note that when providing a BSgenome object, this will first be coerced to a DNAStringSet which can be time consuming.

### Value

An object of the same class as the original object, but with SNPs inserted at the supplied positions

### Examples

```
seq <- DNAStringSet(c(chr1 = "AAGC"))
snps <- GRanges("chr1:2")
snps$ALT <- "G"
snps
seq
owl(seq, snps)
```

---

parY *Get the PAR-Y Regions From a Seqinfo Object*

---

### Description

Define the Pseudo-Autosomal Regions from a Seqinfo Object

### Usage

```
parY(x, ...)

## S4 method for signature 'Seqinfo'
parY(x, ...)

## S4 method for signature 'character'
parY(x, prefix = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | A Seqinfo object or any of named build. If passing a character vector, `match.arg()` will be used to match the build. |
| ... | Not used |
| prefix | Optional prefix to place before chromosome names. Can only be NULL, "" or "chr" |

### Details

Using a seqinfo object based on either hg38, hg19, CHM13.v2 or their variations, create a GRanges object with the Pseudo-Autosomal Regions from the Y chromosome for that build. The length of the Y chromosome on the seqinfo object is used to determine the correct genome build when passing a Seqinfo object. Otherwise

An additional mcols column called PAR will indicate PAR1 and PAR2

### Value

A GenomicRanges object

### Examples

```
library(GenomeInfoDb)
sq <- Seqinfo(
    seqnames = "chrY", seqlengths = 59373566, genome = "hg19_only_chrY"
)
parY(sq)

## PAR regions for CHM13 are also available
sq <- Seqinfo(
    seqnames = "chrY", seqlengths = 62460029, genome = "CHM13"
)
parY(sq)

## Or just call by name
```

```
parY("GRCh38", prefix = "chr")
```

---

sjFromExons                 *Obtain Splice-Junctions from Exons and Transcripts*

---

#### Description

Using GRanges defining exons and transcripts, find the splice-junctions

#### Usage

```
sjFromExons(
  x,
  rank_col = c("exon_number", "exon_rank"),
  tx_col = c("transcript_id", "tx_id"),
  extra_cols = "all",
  don_len = 8,
  acc_len = 5,
  as = c("GRanges", "GInteractions"),
  ...
)
```

#### Arguments

| | |
|---|---|
| x | GRanges object with exons and transcripts. A column indicating the position (or rank) of each exon within the transcript must be included. |
| rank_col | The column containing the position of each exons within the transcript |
| tx_col | The column containing unique transcript-level identifiers |
| extra_cols | Can be a vector of column names to return beyond rank_col and tx_col. By default all columns are returned (extra_cols = "all"). |
| don_len, acc_len | |
| | Length of donor and acceptor sites respectively |
| as | Return as a set of GenomicRanges, or with each splice junction annotated as a GenomicInteraction |
| ... | Not used |

#### Details

A canonical splice junction consists of a donor site and an acceptor site at each end of an intron, with a branching site somewhere wthin the intron. Canonical donor sites are 8nt long with the the first two bases being exonic and the next 6 being derived form intronic sequences. Canonical acceptor sites are 5nt long with the first four bases being intronic and the final base being the first base of the next exon.

This functions uses each set of exons within a transcript to identify both donor and acceptor sites. Branch sites are not identified.

## Value

A GRanges object with requested columns, and an additional column, 'site', annotating each region as a donor or acceptor site.

Alternatively, by specifying as = "GInteractions", the junctions can be returned with each splice junction annotated as a GenomicInteraction. This can make the set of junctions easier to interpret for a given transcript.

## Examples

```
library(rtracklayer)
gtf_cols <- c(
  "transcript_id", "transcript_name", "gene_id", "gene_name", "exon_number"
)
gtf <- import.gff(
    system.file("extdata/gencode.v44.subset.gtf.gz", package = "transmogR"),
    feature.type = "exon", colnames = gtf_cols
)
sj <- sjFromExons(gtf)
sj

## Or to simplify shared splice junctions across multiple transcripts
library(extraChIPs, quietly = TRUE)
chopMC(sj)

## Splice Junctions can also be returned as a GInteractions object with
## anchorOne as the donor & anchorTwo as the acceptor sites
sjFromExons(gtf, as = "GInteractions")
```

---

|          transmogrify          |          *Mogrify a transcriptome using a set of variants*          |
|---|---|

---

## Description

Use a set of SNPs, insertions and deletions to modify a reference transcriptome

## Usage

```
transmogrify(x, var, exons, ...)

## S4 method for signature 'XStringSet,GRanges,GRanges'
transmogrify(
  x,
  var,
  exons,
  alt_col = "ALT",
  trans_col = "transcript_id",
  omit_ranges = NULL,
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
```

```
  var_sep = "_",
  verbose = TRUE,
  mc.cores = 1,
  ...
)

## S4 method for signature 'BSgenome,GRanges,GRanges'
transmogrify(
  x,
  var,
  exons,
  alt_col = "ALT",
  trans_col = "transcript_id",
  omit_ranges = NULL,
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
  var_sep = "_",
  verbose = TRUE,
  mc.cores = 1,
  ...
)

## S4 method for signature 'BSgenome,VcfFile,GRanges'
transmogrify(
  x,
  var,
  exons,
  alt_col = "ALT",
  trans_col = "transcript_id",
  omit_ranges = NULL,
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
  var_sep = "_",
  verbose = TRUE,
  mc.cores = 1,
  which,
  ...
)

## S4 method for signature 'XStringSet,VcfFile,GRanges'
transmogrify(
  x,
  var,
  exons,
  alt_col = "ALT",
  trans_col = "transcript_id",
  omit_ranges = NULL,
  tag = NULL,
  sep = "_",
  var_tags = FALSE,
```

```
    var_sep = "_",
    verbose = TRUE,
    mc.cores = 1,
    which,
    ...
)
```

## Arguments

| x | Reference genome as either a DNAStringSet or BSgenome |
|---|---|
| var | GRanges object containing the variants |
| exons | GRanges object with ranges representing exons |
| ... | Passed to parallel::mclapply |
| alt_col | Column from var containing alternate bases |
| trans_col | Column from 'exons' containing the transcript_id |
| omit_ranges | GRanges object containing ranges to omit, such as PAR-Y regions, for example |
| tag | Optional tag to add to all sequence names which were modified |
| sep | Separator to place between seqnames names & tag |
| var_tags | logical(1) Add tags indicating which type of variant were incorporated, with 's', 'i' and 'd' representing SNPs, Insertions and Deletions respectively |
| var_sep | Separator between any previous tags and variant tags |
| verbose | logical(1) Include informative messages, or operate silently |
| mc.cores | Number of cores to be used when multi-threading via parallel::mclapply |
| which | GRanges object passed to VariantAnnotation::ScanVcfParam if using a VCF directly |

## Details

Produce a set of variant modified transcript sequences from a standard reference genome. Supported variants are SNPs, Insertions and Deletions

Ranges needing to be masked, such as the Y-chromosome, or Y-PAR can be provided.

**It should be noted that this is a time consuming process** Inclusion of a large set of insertions and deletions across an entire transcriptome can involve individually modifying many thousands of transcripts, which can be a computationally demanding task. Whilst this can be parallelised using an appropriate number of cores, this may also prove taxing for lower power laptops, and pre-emptively closing memory hungry programs such as Slack, or internet browers may be prudent.

## Value

An XStringSet

## Examples

```
library(GenomicRanges)
library(GenomicFeatures)
seq <- DNAStringSet(c(chr1 = "ACGTAAATGG"))
exons <- GRanges(c("chr1:1-3:-", "chr1:7-9:-"))
exons$transcript_id <- c("trans1")
```

```
# When using extractTranscriptSeqs -stranded exons need to be sorted by end
exons <- sort(exons, decreasing = TRUE, by = ~end)
exons
trByExon <- splitAsList(exons, exons$transcript_id)

# Check the sequences
seq
extractTranscriptSeqs(seq, trByExon)

# Define some variants
var <- GRanges(c("chr1:2", "chr1:8"))
var$ALT <- c("A", "GGG")

# Include the variants adding tags to indicate a SNP and indel
# The exons GRanges object will be split by transcript internally
transmogrify(seq, var, exons, var_tags = TRUE)
```

---

upsetVarByCol                 *Show Variants by Impacted Columns*

---

### Description

Produce an UpSet plot showing unique values from a given column

### Usage

```
upsetVarByCol(
  gr,
  var,
  alt_col = "ALT",
  mcol = "transcript_id",
  ...,
  intersection_args = list(),
  intersection_lab = "Intersection Size",
  set_geom = geom_bar(width = 0.6),
  set_expand = 0.2,
  set_counts = TRUE,
  hjust_counts = 1.1,
  set_lab = "Set Size",
  title
)
```

### Arguments

| | |
|---|---|
| gr | GRanges object with ranges representing a key feature such as exons |
| var | GRanges object with variants in a given column |
| alt_col | Column within var containing the alternate allele |
| mcol | The column within gr to summarise results by |
| ... | Passed to [ComplexUpset::upset](ComplexUpset::upset) |

intersection_args

        See [ComplexUpset::intersection_size](#) for possible values

intersection_lab

        Y-axis label for the intersection panel

set_geom      Passed to [ComplexUpset::upset_set_size](#)

set_expand     Expand the set-size axis by this amount

set_counts     logical(1) Show counts on set sizes

hjust_counts   Horizontal adjustment of counts, if being shown

set_lab       X-axis label for the set-sizes panel

title          Summary title to show above the intersection panel. Can be hidden by setting to NULL

## Details

Take a set of variants, classify them as SNV, Insertion and Deletion, then using a GRanges object, produce an UpSet plot showing impacted values from a given column

## Value

An UpSet plot

## See Also

[ComplexUpset::upset](#)

## Examples

```
library(rtracklayer)
library(VariantAnnotation)
gtf <- import.gff(
    system.file("extdata/gencode.v44.subset.gtf.gz", package = "transmogR"),
    feature.type = "exon"
)
vcf <- system.file("extdata/1000GP_subset.vcf.gz", package = "transmogR")
var <- rowRanges(readVcf(vcf, param = ScanVcfParam(fixed = "ALT")))
upsetVarByCol(gtf, var)
```

varTypes                  *Identify SNVs, Insertions and Deletions*

## Description

Identify SNVs, Insertions and Deletions within a GRanges object

## Usage

```
varTypes(x, alt_col = "ALT", ...)
```

## Arguments

| | |
|---|---|
| x | GenomicRanges object |
| alt_col | Name of the column with mcols(x) which contains the alternate allele. Can be an XStringSetList, XStringSet or character |
| ... | Not used |

## Details

Using the width of the reference and alternate alleles, classify each range as an SNV, Insertion or Deletion.

- SNVs are expected to have REF & ALT widths of 1
- Insertions are expected to have ALT longer than REF
- Deletions are expected to have ALT shorter than REF

These are relatively permissive criteria

## Value

Character vector

## Examples

```
# Load the example VCF and classify ranges
library(VariantAnnotation)
f <- system.file("extdata/1000GP_subset.vcf.gz", package = "transmogR")
vcf <- readVcf(f)
gr <- rowRanges(vcf)
type <- varTypes(gr)
table(type)
gr[type != "SNV"]
```

# Index