Package 'ProteoDisco'

November 12, 2026

Type Package

Title Generation of customized protein variant databases from genomic variants, splice-junctions and manual sequences

Version 1.18.0

Date 2025-07-22

biocViews Software, Proteomics, RNASeq, SNP, Sequencing, VariantAnnotation, DataImport

URL https://github.com/ErasmusMC-CCBC/ProteoDisco

BugReports https://github.com/ErasmusMC-CCBC/ProteoDisco/issues

Description ProteoDisco is an R package to facilitate proteogenomics studies. It houses functions to create customized (variant) protein databases based on user-submitted genomic variants, splice-junctions, fusion genes and manual transcript sequences. The flexible workflow can be adopted to suit a myriad of research and experimental settings.

License GPL-3

LazyData FALSE

Depends R (>= 4.1.0),

Imports BiocGenerics (>= 0.38.0), BiocParallel (>= 1.26.0), Biostrings (>= 2.60.1), checkmate (>= 2.0.0), cleaver (>= 1.30.0), dplyr (>= 1.0.6), GenomeInfoDb (>= 1.28.0), GenomicFeatures (>= 1.44.0), GenomicRanges (>= 1.44.0), IRanges (>= 2.26.0), methods (>= 4.1.0), ParallelLogger (>= 2.0.1), plyr (>= 1.8.6), rlang (>= 0.4.11), S4Vectors (>= 0.30.0), Seqinfo, tibble (>= 3.1.2), tidyr (>= 1.1.3), VariantAnnotation (>= 1.36.0), XVector (>= 0.32.0),

RoxygenNote 7.1.2

Suggests AnnotationDbi (>= 1.54.1), BSgenome (>= 1.60.0), BSgenome.Hsapiens.UCSC.hg19 (>= 1.4.3), BiocStyle (>= 2.20.1), DelayedArray (>= 0.18.0), devtools (>= 2.4.2), knitr (>= 1.33), matrixStats (>= 0.59.0), markdown (>= 1.1), org.Hs.eg.db (>= 3.13.0), purrr (>= 0.3.4), RCurl (>= 1.98.1.3), readr (>= 1.4.0), ggplot2 (>= 3.3.5), rmarkdown (>= 2.9), rtracklayer (>=

1.52.0), seqinr (>= 4.2.8), stringr (>= 1.4.0), reshape2 (>= 1.4.4), scales (>= 1.1.1), testthat (>= 3.0.3), TxDb.Hsapiens.UCSC.hg19.knownGene (>= 3.2.2)

VignetteBuilder knitr

Encoding UTF-8

git_url https://git.bioconductor.org/packages/ProteoDisco

git_branch RELEASE_3_23

git_last_commit 4b444b0

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Job van Riet [cre],
Wesley van de Geer [aut],
Harmen van de Werken [ths]

Maintainer Job van Riet <jobvriet@gmail.com>

Contents

.checkReferenceAnchor . . . 3
checkProteotypicFragments . . . 3
exportProteoDiscography . . . 5
generateJunctionModels . . . 6
generateProteoDiscography . . . 8
getDiscography . . . 9
importGenomicVariants . . . 10
importGenomicVariants.MAF . . . 11
importGenomicVariants.VCF . . . 12
importSpliceJunctions . . . 14
importTranscriptSequences . . . 16
incorporateGenomicVariants . . . 17
mutantTranscripts . . . 19
ProteoDiscography-class . . . 19
ProteoDiscographyExample.hg19 . . . 20
setGenomicSequences . . . 21
setMutantTranscripts . . . 22
setTxDb . . . 23

Index **24**

.checkReferenceAnchor *Checks if mutations and reference genome correspond*

Description

Overlaps the reference bases and positions with the given genome and check if all positions overlap and reports non-overlapping mutations.

Usage

```
.checkReferenceAnchor(
  mutations,
  genomeSeqs,
  ignoreNonMatch = FALSE,
  threads = 1
)
```

Arguments

`mutations`	(VRanges): Mutations to incorporate within transcripts.
`genomeSeqs`	(BSgenome or DNAStringSet): Reference genome sequences.
`ignoreNonMatch`	(logical): Should non-matching reference anchors be ignored? These mutations will be removed.
`threads`	(integer): Number of threads.

Value

(logical): Returns TRUE/FALSE if per given mutation based on matching position and base with the reference genome.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

checkProteotypicFragments *Determine proteotypic fragments within the translated CDS*

Description

Proteotypic fragments are checked against the input TxDb (and additional peptide sequences) to infer whether peptide sequences contain one or multiple proteotypic fragment(s) after cleavage by a selected protease.

Usage

```
checkProteotypicFragments(
  x,
  enzymUsed = "trypsin",
  missedCleavages = 0,
  additionalPeptides = NULL,
  checkWithinMutantSeqs = FALSE
)

## S4 method for signature 'ProteoDiscography'
checkProteotypicFragments(
  x,
  enzymUsed = "Trypsin",
  missedCleavages = 0,
  additionalPeptides = NULL,
  checkWithinMutantSeqs = FALSE
)
```

Arguments

x	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
enzymUsed	(character): Preferred proteasome used in cleaving the peptide sequences, e.g. trypsin.
missedCleavages	(integer): Number of subsequent missed cleavages.
additionalPeptides	(AAStringSet): Additional peptide sequences against which to check proteotypic fragments, besides the input TxDb. If left empty, only the (translated) input TxDb will be checked.
checkWithinMutantSeqs	(logical): Should proteotypic fragments also not be present within another mutant peptide-sequence?

Value

ProteoDiscography with additional information added to the transcript sequence information.

ProteoDiscography with an additional column specifying the number of proteotypic fragments per record.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
# Import example ProteoDiscography (hg19)
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

# Results will now contain additional information about proteotypic fragments.
# With a large TxDb, this can take a while.
# ProteoDiscography.hg19 <- ProteoDisco::checkProteotypicFragments(ProteoDiscographyExample.hg19)
```

`exportProteoDiscography` *Export the generated (mutant) peptides sequences to a FASTA database.*

Description

Exports (mutant) peptide sequences to FASTA using several fields as identifier. In addition, users can specify to only export (mutant) sequences containing at a min. number of proteotypic fragments.

Usage

```
exportProteoDiscography(
  ProteoDiscography,
  outFile = NULL,
  minProteotypicFragments = NULL,
  aggregateSamples = TRUE
)
```

Arguments

`ProteoDiscography` (ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.

`outFile` (character): Filepath to output FASTA, will append the samplenames if aggregateSamples is FALSE. If left NULL, it will return an AAStringSet with the given records.

`minProteotypicFragments` (integer): Only output mutant protein-isoforms from incorporated genomic variants with at least this many proteotypic fragments (if ‘checkProteotypicFragments()‘ was performed).

`aggregateSamples` (logical): Should samples be aggregated into the same output database (TRUE) or should a seperate FASTA-file be generated per sample (FALSE).

Value

Writes a FASTA file containing the mutant protein isoforms if outFile is given. Otherwise, will return an AAStringSet.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
# Import example ProteoDiscography (hg19)
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

# Export peptide sequences to FASTA file. Optionally, only export those with at least a min.
# number of proteotypic fragments.
exportProteoDiscography(ProteoDiscographyExample.hg19, outFile = 'out.fasta')
```

---

generateJunctionModels

Generate putative transcript-models derived from splice-junctions.

Description

Generates splicing-isoforms using the supplied splice-junctions (SJ) within the ProteoDiscography.

Supplied junctions will be transformed into novel gene-models based on the nearest (or overlapping) exon with the TxDb of the given ProteoDiscography; strand-information is taken into account. If no strand-information is given, the nearest (or overlapping) known exon is assigned.

It will derive the putative gene-model based on the assigned exons as annotated within the TxDb. E.g., if matched with the second exon of geneA (+) and the fourth exon of geneB (+) it will generate the following gene-model:

geneA-Exon1, geneA-Exon2, geneB-Exon4, geneB-Exon5, ...

Users can also specify the max. search-window (in bp) in which the nearest canonical exonic boundary should fall.

Usage

```
generateJunctionModels(
  ProteoDiscography,
  maxDistance = 150,
  maxCrypticSize = 75,
  skipCanonical = TRUE,
  threads = 1
)
```

Arguments

ProteoDiscography	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
maxDistance	(integer): Max. distance (>=0 bp) from splice-junction to nearest (or overlapping) exon in bp. Setting this to high numbers will (erroneously) assign a distant exon as 'neighboring' resulting in unlikely models. If the SJ is beyond this distance, it will be assigned as an cryptic exon and the length of this exon is set with the [maxCrypticSize] parameter.
maxCrypticSize	(integer): The max. extension of bp (respective to orientation) beyond the SJ if it is assigned as a cryptic exon. I.e., the nr. of bp that will be used in determining the putative transcript sequence for each cryptic junction.
skipCanonical	(logical): Should canonical exon-exon junctions be skipped (TRUE) or generated (FALSE)?
threads	(integer): Number of threads.

Value

ProteoDiscography with derived splice-isoforms.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)

# Import splice-junctions (even spanning different chromosomes) based on our format.
testSJ <- readr::read_tsv(system.file('extdata', 'validationSetSJ_hg19.txt', package = 'ProteoDisco'))

# Add custom SJ to ProteoDiscography.
ProteoDiscography.hg19 <- ProteoDisco::importSpliceJunctions(
  ProteoDiscography = ProteoDiscography.hg19,
  inputSpliceJunctions = testSJ
)

# Generate junction-models from non-canonical splice-junctions.
ProteoDiscography.hg19 <- ProteoDisco::generateJunctionModels(
  ProteoDiscography = ProteoDiscography.hg19,
  # Max. distance from a known exon-boundary before introducing a novel exon.
 # If an adjacent exon is found within this distance, it will shorten or elongate that exon towards the SJ.
  maxDistance = 150,
 # Should we skip known exon-exon junctions (in which both the acceptor and donor are located on known adjacent exon
  skipCanonical = TRUE,
```

```
  # Perform on multiple threads (optional)
  threads = 1
)
```

---

generateProteoDiscography *Generate the annotation database (ProteoDiscography)*

Description

Generates a database containing the genomic and transcriptomic annotations which will be used downstream to integrate genomic variants and discover mutant protein sequences with ProteoDisco.

By default, only the common seqlevels between the TxDb and genomeSeqs will be kept..

Usage

```
generateProteoDiscography(
  TxDb,
  genomeSeqs,
  useOnlySharedSeqlevels = TRUE,
  geneticCode = "Standard"
)
```

Arguments

TxDb	(TxDb): TxDb object containing the genomic and transcriptomic annotations.
genomeSeqs	(DNAStringSet or BSgenome): Genomic sequence of the respective genome.
useOnlySharedSeqlevels	(logical): Should only shared seqlevels between the TxDb and genomeSeqs be kept?
geneticCode	(character): Which codon-table should be used? Default is set to Standard, check GENETIC_CODE_TABLE for additional options.

Value

ProteoDiscography

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
# Generate a ProteoDiscography using existing TxDb and annotations.
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)
```

`getDiscography` *Retrieve imported genomic variants, splice-junctions and manual sequences.*

Description

Retrieve imported genomic variants, splice-junctions and manual sequences.

Usage

```
getDiscography(x)

## S4 method for signature 'ProteoDiscography'
getDiscography(x)
```

Arguments

x	(ProteoDiscography): ProteoDiscography object.

Value

Return a list of imported records, per category (genomic variants, splice-junctions and manual sequences).

Examples

```
# Import example ProteoDiscography (hg19)
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

# Retrieve the imported records.
getDiscography(ProteoDiscographyExample.hg19)
```

`importGenomicVariants` *Import genomic variants into the ProteoDiscography*

Description

Imports genomic variants (SNV, MNV and InDels) present within the supplied VCF/MAF files into the ProteoDiscography as a VRanges. This genomic variants can later be incorporated within transcript sequences at a later stage.

Usage

```
importGenomicVariants(
  ProteoDiscography,
  files,
  samplenames = NULL,
  removeExisting = FALSE,
  overwriteDuplicateSamples = TRUE,
  performAnchorCheck = TRUE,
  ignoreNonMatch = FALSE,
  threads = 1
)
```

Arguments

`ProteoDiscography`	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
`files`	(character): Path(s) to VCF or MAF files.
`samplenames`	(character): Descriptive samplename(s) of the VCF files in the same order as input VCF file(s), if NULL the basename of the file will be used instead.
`removeExisting`	(logical): Should previous mutations within the ProteoDiscography be removed?
`overwriteDuplicateSamples`	(logical): Replace duplicate samples (TRUE) or throw an error if duplicate samples are found.
`performAnchorCheck`	(logical): Should the reference anchor be check for consistency with the given genomic sequences?
`ignoreNonMatch`	(logical): Should non-matching reference anchors be ignored? These mutations will be removed prior to appending.
`threads`	(integer): Number of threads.

Value

ProteoDiscography with additional imported SNVs, MNVs and InDels.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)

# Supply the ProteoDiscography with genomic variants to incorporate in downstream analysis. This can be one or multi
# Additional manual sequences and exon-exon mapping (i.e., splice junctions) can also be given as shown in the secti
ProteoDiscography.hg19 <- ProteoDisco::importGenomicVariants(
  ProteoDiscography = ProteoDiscography.hg19,
  # Provide the VCF / MAF files, if more then one supply a vector of files and corresponding samplenames.
  files = system.file('extdata', 'validationSet_hg19.vcf', package = 'ProteoDisco'),
  # We can replace the original samples within the VCF with nicer names.
  samplenames = 'Validation Set (GRCh37)',
  # Number of threads used for parallelization.
  # We run samples sequentially and parallelize within (variant-wise multi-threading).
  threads = 1,
  # To increase import-speed for this example, do not check for validity of the reference anchor with the given refer
  performAnchorCheck = FALSE
)
```

---

importGenomicVariants.MAF

Import MAF files into a ProteoDiscography.

Description

Import MAF files into a ProteoDiscography.

Usage

```
importGenomicVariants.MAF(
  ProteoDiscography,
  files,
  performAnchorCheck = TRUE,
  ignoreNonMatch = FALSE,
  samplenames = NULL,
  threads = 1
)
```

Arguments

ProteoDiscography	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
files	(character): Path(s) to VCF or MAF files.
performAnchorCheck	(logical): Should the reference anchor be check for consistency with the given genomic sequences?
ignoreNonMatch	(logical): Should non-matching reference anchors be ignored? These mutations will be removed prior to appending.
samplenames	(character): Descriptive samplename(s) of the VCF files in the same order as input VCF file(s), if NULL the basename of the file will be used instead.
threads	(integer): Number of threads.

Value

ProteoDiscography with additional imported SNVs, MNVs and InDels.

Examples

```
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)

# Supply the ProteoDiscography with genomic variants to incorporate in downstream analysis. This can be one or multi
# Additional manual sequences and exon-exon mapping (i.e., splice junctions) can also be given as shown in the secti
ProteoDiscography.hg19 <- ProteoDisco::importGenomicVariants(
  ProteoDiscography = ProteoDiscography.hg19,
 # Provide the VCF / MAF files, if more then one supply a vector of files and corresponding samplenames.
  files = system.file('extdata', 'validationSet_hg19.vcf', package = 'ProteoDisco'),
  # We can replace the original samples within the VCF with nicer names.
  samplenames = 'Validation Set (GRCh37)',
  # Number of threads used for parallelization.
  # We run samples sequentially and parallelize within (variant-wise multi-threading).
  threads = 1,
 # To increase import-speed for this example, do not check for validity of the reference anchor with the given refer
  performAnchorCheck = FALSE
)
```

---

importGenomicVariants.VCF

Import VCF files into a ProteoDiscography.

Description

Import VCF files into a ProteoDiscography.

Usage

```
importGenomicVariants.VCF(
  ProteoDiscography,
  files,
  samplenames = NULL,
  performAnchorCheck = TRUE,
  ignoreNonMatch = FALSE,
  threads = 1
)
```

Arguments

ProteoDiscography	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
files	(character): Path(s) to VCF or MAF files.
samplenames	(character): Descriptive samplename(s) of the VCF files in the same order as input VCF file(s), if NULL the basename of the file will be used instead.
performAnchorCheck	(logical): Should the reference anchor be check for consistency with the given genomic sequences?
ignoreNonMatch	(logical): Should non-matching reference anchors be ignored? These mutations will be removed prior to appending.
threads	(integer): Number of threads.

Value

ProteoDiscography with additional imported SNVs, MNVs and InDels.

Examples

```
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)

# Supply the ProteoDiscography with genomic variants to incorporate in downstream analysis. This can be one or multi
# Additional manual sequences and exon-exon mapping (i.e., splice junctions) can also be given as shown in the secti
ProteoDiscography.hg19 <- ProteoDisco::importGenomicVariants(
  ProteoDiscography = ProteoDiscography.hg19,
 # Provide the VCF / MAF files, if more then one supply a vector of files and corresponding samplenames.
  files = system.file('extdata', 'validationSet_hg19.vcf', package = 'ProteoDisco'),
  # We can replace the original samples within the VCF with nicer names.
  samplenames = 'Validation Set (GRCh37)',
  # Number of threads used for parallelization.
  # We run samples sequentially and parallelize within (variant-wise multi-threading).
  threads = 1,
 # To increase import-speed for this example, do not check for validity of the reference anchor with the given refer
  performAnchorCheck = FALSE
```

```
)
```

`importSpliceJunctions` *Import splice-junctions into the ProteoDiscograpy.*

Description

Generates putative gene-models based on supplied genomic coordinates of splice-junctions.

Input should be a tibble containing the following columns:

- junctionA: Genomic coordinates of the 5'-junction. (format: chr:start:strand, i.e.: chr1:100:+)
- junctionB: Genomic coordinates of the 3'-junction. (format: chr:end:strand, i.e.: chr1:150:+)
- sample: Names of the samples. (character, optional)
- identifier: The identifier which will be used in downstream analysis. (character, optional)

Common splice-junction formats (BED and SJ.out.tab (STAR)) can also be supplied and are converted into the correct DataFrame.

By utilizing two separate junction-sites, interchromosomal trans-splicing or chimeric transcripts from genomic fusions (e.g., resulting from the BCR/ABL1 fusion-gene) can also be handled.

Usage

```
importSpliceJunctions(
  ProteoDiscography,
  inputSpliceJunctions,
  isTopHat = TRUE,
  samples = NULL,
  aggregateSamples = FALSE,
  removeExisting = FALSE,
  overwriteDuplicateSamples = FALSE
)
```

Arguments

`ProteoDiscography`	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
`inputSpliceJunctions`	(tibble): Tibble containing the splice-junctions.
`isTopHat`	(logical): Are the imported (.BED) files from TopHat? If so, the start-end of the SJ are corrected for max. overhang.
`samples`	(character): Preferred names for the samples if BED / TAB files are supplied, default is derived from filepath.

`aggregateSamples`
: (logical): Should splice-junctions from multiple samples be aggregated? Or should sample-specific models be generated? If genomic variants are to be incorporated within the derived splice-transcripts, the names of samples need to be match.

`removeExisting`
: (logical): Should existing entries be removed?

`overwriteDuplicateSamples`
: (logical): Should duplicate samples be overwritten?

Value

ProteoDiscography with imported splice-junctions.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)

# Import from file.
ProteoDiscography.hg19 <- ProteoDisco::importSpliceJunctions(
  ProteoDiscography = ProteoDiscography.hg19,
 inputSpliceJunctions = system.file('extdata', 'spliceJunctions_pyQUILTS_chr22.bed', package = 'ProteoDisco'),
  # (Optional) Rename samples.
  samples = 'pyQUILTS',
  # Specify that the given BED files are obtained from TopHat.
  # Chromosomal coordinates from TopHat require additional formatting.
  isTopHat = TRUE,
)

# Or, import splice-junctions (even spanning different chromosomes) based on our format.
testSJ <- readr::read_tsv(system.file('extdata', 'validationSetSJ_hg19.txt', package = 'ProteoDisco'))

# Add custom SJ to ProteoDiscography.
ProteoDiscography.hg19 <- ProteoDisco::importSpliceJunctions(
  ProteoDiscography = ProteoDiscography.hg19,
  inputSpliceJunctions = testSJ,
  # Append to existing SJ-input.
  removeExisting = FALSE
)
```

`importTranscriptSequences` *Incorporate manual transcript sequences into the ProteoDiscography*

Description

The given DataFrame (transcripts) should include the following columns:

- sample: Names of the samples. (character)
- identifier: The identifier which will be used in downstream analysis. (character)
- Tx.SequenceMut: The mutant mRNA sequence. (DNAStringSet)
- gene: Name of the gene. (character, optional)

The supplied mutant sequences do not need to have a respective record within the used TxDb as these are handled as independent sequences.

Usage

```
importTranscriptSequences(
  ProteoDiscography,
  transcripts,
  removeExisting = FALSE,
  overwriteDuplicateSamples = FALSE
)
```

Arguments

`ProteoDiscography`	(ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.
`transcripts`	(DataFrame): DataFrame containing metadata and the transcript sequences.
`removeExisting`	(logical): Should previous mutations within the ProteoDiscography be removed?
`overwriteDuplicateSamples`	(logical): Replace duplicate samples (TRUE) or throw an error if duplicate samples are found.

Value

ProteoDiscography with added mutant transcript sequences from manual input.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
ProteoDiscography.hg19 <- ProteoDisco::generateProteoDiscography(
  TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens.UCSC.hg19.knownGene,
  genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19
)

manualSeq <- S4Vectors::DataFrame(
  sample = rep('Example', 2),
  identifier = c('Example 1', 'Example 2'),
  gene = c('GeneA', 'GeneB'),
 Tx.SequenceMut = Biostrings::DNAStringSet(list(Biostrings::DNAString('ATCGGGCCCGACGTT'), Biostrings::DNAStrin
)

# Add to ProteoDiscography.
ProteoDiscography.hg19 <- ProteoDisco::importTranscriptSequences(
  ProteoDiscography.hg19,
  transcripts = manualSeq
)
```

---

incorporateGenomicVariants

Incorporate genomic events into their overlapping exonic sequences

---

Description

`incorporateMutations` Incorporates SNV, MNV and InDels present in the ProteoDiscography on the transcripts.

Usage

```
incorporateGenomicVariants(
  ProteoDiscography,
  aggregateSamples = FALSE,
  aggregateWithinExon = TRUE,
  aggregateWithinTranscript = TRUE,
  ignoreOverlappingMutations = TRUE,
  threads = 1
)
```

Arguments

`ProteoDiscography`
: (ProteoDiscography): ProteoDiscography object which stores the annotation and genomic sequences.

`aggregateSamples`
: (logical): Should genomic mutations from different samples be incorporated within the same transcript?

aggregateWithinExon
: (logical): Should multiple mutations within the same exon be aggregated (TRUE) or should each mutation per exon produce a separate mutant transcript?

aggregateWithinTranscript
: (logical): Should multiple mutant exons within the same transcript be aggregated?

ignoreOverlappingMutations
: (logical): Incorporate first mutation (by order) and ignore subsequent overlapping mutations (and provide a warning) or stop the incorporation (if set to TRUE). If aggregateWithinExon is set to FALSE, each mutations are added into separate exon to produce seperate transcripts.

threads
: (integer): Number of threads.

Value

ProteoDiscography with mutant transcript sequences containing SNVs, MNVs and InDels.

Author(s)

Job van Riet <j.vanriet@erasmusmc.nl>

Wesley van de Geer <w.vandegeer@erasmusmc.nl>

Examples

```
# Import example ProteoDiscography (hg19)
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

# Incorporate the genomic variants.
ProteoDiscographyExample.hg19 <- ProteoDisco::incorporateGenomicVariants(
  ProteoDiscography = ProteoDiscographyExample.hg19,
  # Do not aggregate samples and generate mutant transcripts from the mutations per sample.
  aggregateSamples = FALSE,
  # If there are multiple mutations within the same exon (CDS), place them on the same mutant CDS sequence.
  aggregateWithinExon = TRUE,
  # Aggregate multiple mutant exons (CDS) within the same transcripts instead of incorporating one at a time.
  aggregateWithinTranscript = TRUE,
  # If there are overlapping mutations on the same coding position, retain only the first of the overlapping mutation
  # If set to FALSE, throw an error and specify which CDS had overlapping mutations.
  ignoreOverlappingMutations = TRUE,
  # Number of threads.
  threads = 1
)
```

mutantTranscripts *Adds mutant transcript sequences to the ProteoDiscography in the appropriate slot*

Description

Adds mutant transcript sequences to the ProteoDiscography in the appropriate slot

Usage

```
mutantTranscripts(x)
```

Arguments

x	(ProteoDiscography): ProteoDiscography.

Value

Return all incorporated mutant transcripts (list of DataFrame).

Examples

```
# Import example ProteoDiscography (hg19)
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

# Retrieve all generated mutant transcripts.
mutantTranscripts(ProteoDiscographyExample.hg19)
```

ProteoDiscography-class *ProteoDiscography*

Description

An S4 object containing the reference genome sequences and gene-model annotations (TxDB).

This also stores genomic variants and splice-junctions from which mutant transcript sequences can be generated.

Slots

`TxDb` (TxDb): TxDb object containing the genomic and transcript annotations.

`genomeSeqs` (DNAStringSet or BSgenome): Genomic sequence of the respective genome.

`input.genomicVariants` (VRangesList): Imported genomic variants (SNV, MNV and InDels).

`input.spliceJunctions` (DataFrame): Imported splice-junctions.

`input.manualSequences` (DataFrame): Imported manual sequences.

`mutantTranscripts.genomicVariants` (tibble): Generated mutant mRNA sequences from genomic variants.

`mutantTranscripts.spliceJunctions` (tibble): Generated mutant mRNA sequences from splice-junctions.

`mutantTranscripts.manualSequences` (tibble): Processed mutant mRNA sequences from manual input.

`GENETIC_CODE` (GENETIC_CODE_TABLE): The genetic code table to be used during translation.

`metadata` (data.frame): Supplied data.frame.

Author(s)

Job van Riet

`ProteoDiscographyExample.hg19` *Example ProteoDiscography.*

Description

A ProteoDiscography object based on validated genomic variants (hg19) using existing transcript annotations (hg19).

Usage

```
ProteoDiscographyExample.hg19
```

Format

An object of class `ProteoDiscography` of length 1.

Details

Generated using the following commands: # Use existing hg19 annotations. ProteoDiscographyExample.hg19 <- ProteoDisco::generateProteoDiscography(TxDb = TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb.Hsapiens genomeSeqs = BSgenome.Hsapiens.UCSC.hg19::BSgenome.Hsapiens.UCSC.hg19)

Add mutations from hg19 validation set. ProteoDiscographyExample.hg19 <- ProteoDisco::importGenomicVariants(ProteoDiscography = ProteoDiscographyExample.hg19, files = system.file('extdata', 'validationSet_hg19.vcf', package = 'ProteoDisco'), samplenames = 'Validation Set (GRCh37)', threads = 1)

Incorporate genomic mutations. ProteoDiscographyExample.hg19 <- ProteoDisco::incorporateGenomicVariants(ProteoDiscography = ProteoDiscographyExample.hg19, aggregateSamples = FALSE, aggregateWithinExon = TRUE, aggregateWithinTranscript = FALSE, ignoreOverlappingMutations = TRUE, threads = 1)

Source

https://cancer.sanger.ac.uk/cosmic

`setGenomicSequences` *Change the underlying genomic sequences of a ProteoDiscography object.*

Description

Change the underlying genomic sequences of a ProteoDiscography object.

Usage

```
setGenomicSequences(x, genomeSeqs)

## S4 method for signature 'ProteoDiscography'
setGenomicSequences(x, genomeSeqs)
```

Arguments

`x`	(ProteoDiscography): ProteoDiscography object.
`genomeSeqs`	(DNAStringSet or BSgenome): Genomic sequence of the respective genome.

Value

ProteoDiscography with updated genomic sequences.

Examples

```
# Import example ProteoDiscography (hg19) and re-link genomic sequences.
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

summary(ProteoDiscographyExample.hg19)
```

`setMutantTranscripts` *Adds mutant transcript sequences to the ProteoDiscography in the appropriate slot*

Description

Adds mutant transcript sequences to the ProteoDiscography in the appropriate slot

Usage

```
setMutantTranscripts(x, transcripts, slotType)

## S4 method for signature 'ProteoDiscography'
setMutantTranscripts(x, transcripts, slotType)

## S4 method for signature 'ProteoDiscography'
mutantTranscripts(x)
```

Arguments

`x`	(ProteoDiscography): The ProteoDiscography for which the slot will be edited.
`transcripts`	(DataFrame): Transcripts to be used in the slot.
`slotType`	(character): Implemented slot to be edited.

Value

ProteoDiscography with updated records.

Examples

```
# From a ProteoDiscography with imported and incorporated records, take only the first 10 records.
# ProteoDisco::setMutantTranscripts(ProteoDiscography)$genomicVariants[1:10], slotType = 'genomicVariants')
# ProteoDisco::setMutantTranscripts(ProteoDiscography)$spliceJunctions[1:10], slotType = 'spliceJunctions')
# ProteoDisco::setMutantTranscripts(ProteoDiscography)$manualSequences[1:10], slotType = 'manualSequences')

# Import example ProteoDiscography (hg19)
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb
ProteoDiscographyExample.hg19 <- setGenomicSequences(ProteoDiscographyExample.hg19, BSgenome.Hsapiens.UCSC.hg19

# Only keep the first ten records.
ProteoDiscographyExample.hg19 <- ProteoDisco::setMutantTranscripts(
  x = ProteoDiscographyExample.hg19,
 transcripts = ProteoDisco::mutantTranscripts(ProteoDiscographyExample.hg19)$genomicVariants[1:10,],
  slotType = 'genomicVariants'
)
```

setTxDb *Change the underlying TxDb of a ProteoDiscography object.*

Description

Change the underlying TxDb of a ProteoDiscography object.

Usage

```
setTxDb(x, TxDb)

## S4 method for signature 'ProteoDiscography'
setTxDb(x, TxDb)
```

Arguments

x	(ProteoDiscography): ProteoDiscography object.
TxDb	(TxDb): TxDb object containing the genomic and transcript annotations.

Value

ProteoDiscography with updated TxDb.

Examples

```
# Import example ProteoDiscography (hg19) and re-link TxDb.
data('ProteoDiscographyExample.hg19', package = 'ProteoDisco')
ProteoDiscographyExample.hg19 <- setTxDb(ProteoDiscographyExample.hg19, TxDb.Hsapiens.UCSC.hg19.knownGene::TxDb

summary(ProteoDiscographyExample.hg19)
```

Index

∗ **datasets**
ProteoDiscographyExample.hg19, 20
∗ **methods**
.checkReferenceAnchor, 3
checkProteotypicFragments, 3
exportProteoDiscography, 5
generateJunctionModels, 6
generateProteoDiscography, 8
importGenomicVariants, 10
importSpliceJunctions, 14
importTranscriptSequences, 16
incorporateGenomicVariants, 17
.checkReferenceAnchor, 3

BSgenome, *3*, *8*, *20, 21*

checkProteotypicFragments, 3
checkProteotypicFragments,ProteoDiscography-method
*(*checkProteotypicFragments*)*, 3

DataFrame, *16*
DNAStringSet, *3*, *8*, *20, 21*

exportProteoDiscography, 5

generateJunctionModels, 6
generateProteoDiscography, 8
GENETIC_CODE_TABLE, *8*, *20*
getDiscography, 9
getDiscography,ProteoDiscography-method
*(*getDiscography*)*, 9

importGenomicVariants, 10
importGenomicVariants.MAF, 11
importGenomicVariants.VCF, 12
importSpliceJunctions, 14
importTranscriptSequences, 16
incorporateGenomicVariants, 17

mutantTranscripts, 19
mutantTranscripts,ProteoDiscography-method
*(*setMutantTranscripts*)*, 22

ProteoDiscography, *5*, *7*
ProteoDiscography-class, 19
ProteoDiscographyExample.hg19, 20

setGenomicSequences, 21
setGenomicSequences,ProteoDiscography-method
*(*setGenomicSequences*)*, 21
setMutantTranscripts, 22
setMutantTranscripts,ProteoDiscography-method
*(*setMutantTranscripts*)*, 22
setTxDb, 23
setTxDb,ProteoDiscography-method
*(*setTxDb*)*, 23

tibble, *14*, *20*
TxDb, *8*, *20*, *23*

VRanges, *3*
VRangesList, *20*