Package ‘chimeraviz’

January 10, 2018

Type Package
Title Visualization tools for gene fusions
Version 1.4.0
Description chimeraviz manages data from fusion gene finders and provides useful visualization tools.
License Artistic-2.0
LazyData TRUE
Imports methods, readr, grid, Rsamtools, GenomeInfoDb, GenomicAlignments, RColorBrewer, graphics, AnnotationDbi, RCircos, org.Hs.eg.db, rmarkdown, graph, Rgraphviz, DT, plyr, dplyr, BiocStyle, ArgumentCheck
Depends Biostrings, GenomicRanges, IRanges, Gviz, S4Vectors, ensembldb, AnnotationFilter
Suggests testthat, roxygen2, devtools, knitr
SystemRequirements bowtie, samtools, and egrep are required for some functionalities
RoxygenNote 6.0.1
VignetteBuilder knitr
biocViews Infrastructure, Alignment
Encoding UTF-8
URL https://github.com/stianlagstad/chimeraviz
BugReports https://github.com/stianlagstad/chimeraviz/issues
NeedsCompilation no
Author Stian Lågstad [aut, cre], Sen Zhao [ctb], Andreas M. Hoff [ctb], Bjarne Johannessen [ctb], Ole Christian Lingjærde [ctb], Rolf Skotheim [ctb]
Maintainer Stian Lågstad <stianlagstad@gmail.com>
R topics documented:

- addFusionReadsAlignment ........................................ 3
- chimeraviz .................................................. 4
- chimeraviz-internals-fusionsToGeneLabelData ................ 4
- chimeraviz-internals-fusionsToLinkData ..................... 4
- chimeraviz-internals-scaleListToInterval ................. 5
- createFusionReport ......................................... 6
- decideTranscriptCategory .................................. 6
- downShift ................................................... 8
- downstreamPartnerGene ...................................... 8
- fetchReadsFromFastq ......................................... 9
- Fusion-class .................................................. 10
- fusionSpanningReadsCount ................................ 11
- fusionSplitReadsCount ..................................... 11
- fusionToDataFrame .......................................... 12
- getEnsemblIds .............................................. 13
- getFusionByChromosome .................................... 14
- getFusionByGeneName ....................................... 14
- getFusionById .............................................. 15
- getTranscriptsEnsemblDb .................................. 16
- importDefuse ............................................... 17
- importEricscript ........................................... 17
- importFunctionNonUCSC .................................... 18
- importFusioncatcher ....................................... 19
- importFusionmap ........................................... 19
- importInfusion .............................................. 20
- importJaffa ................................................. 21
- importPrada .................................................. 21
- importSoapfuse ............................................. 22
- importStarfusion .......................................... 23
- PartnerGene-class .......................................... 23
- partnerGeneEnsemblId ..................................... 24
- partnerGeneJunctionSequence ................................ 25
- plotCircle ................................................... 26
- plotFusion .................................................... 27
- plotFusionReads ............................................. 29
- plotFusionTranscript ....................................... 30
- plotFusionTranscriptsGraph ................................ 32
- plotFusionTranscriptWithProteinDomain .................... 33
- plotTranscripts ............................................. 35
- raw_cytobandhg19 ............................................ 37
- raw_cytobandhg38 ............................................ 37
- raw_defuse .................................................. 38
- raw_ericscript ................................................ 38
- raw_fusion5267proteindomains ................................ 39
- raw_fusion5267reads ........................................ 39
- raw_fusion5267readsBedGraph ................................ 39
- raw_fusioncatcher .......................................... 40
- raw_fusionmap .............................................. 40
- raw_Homo_sapiens.GRCh37.74 .................................. 40
- raw_infusion ............................................... 41
**addFusionReadsAlignment**

Add fusion reads alignment to fusion object

**Description**

This function lets you add a fusion read alignment file to a fusion object. If you’ve mapped the reads supporting a fusion against the fusion junction sequence, and have the resulting bamfile, use this function to add the information (as a Gviz::GAlignmentPairs object) to the fusion object.

**Usage**

`addFusionReadsAlignment(fusion, bamfile)`

**Arguments**

- **fusion**: The fusion object to add a genomic alignment to.
- **bamfile**: The bam file containing the fusion reads plotted to the fusion sequence.

**Value**

An updated fusion object with fusion@fusionReadsAlignment set.

**Examples**

```r
# Load data
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
# Find the specific fusion we have aligned reads for
fusion <- getFusionById(fusions, 5267)
# Get reference to the bamfile with the alignment data
bamfile5267 <- system.file(
  "extdata",
  "5267readsAligned.bam",
  package="chimeraviz")
# Add the bam file of aligned fusion reads to the fusion object
fusion <- addFusionReadsAlignment(fusion, bamfile5267)
```
chimeraviz: A package for working with and visualizing fusion genes.

Description
chimeraviz manages data from fusion gene finders and provides useful visualization tools.

chimeraviz-internals-fusionsToGeneLabelData
Create gene label data for RCircos from the given fusions.

Description
This function takes a list of Fusion objects and creates a data frame in the format that RCircos.Gene.Name.Plot() expects for gene label data.

Usage
.fusionsToGeneLabelData(fusionList)

Arguments
fusionList: A list of Fusion objects.

Value
A data frame with fusion gene label data compatible with RCircos.Gene.Name.Plot()

# @examples # Apparently examples shouldn’t be set on private functions defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz") fusions <- importDefuse(defuse833ke, "hg19", 3) labelData <- chimeraviz::.fusionsToGeneLabelData(fusions) # This labelData can be used with RCircos.Gene.Connector.Plot() and RCircos.Gene.Name.Plot()

chimeraviz-internals-fusionsToLinkData
Create link data for RCircos from the given fusions.

Description
This function takes a list of Fusion objects and creates a data frame in the format that RCircos::RCircos.Link.Plot() expects for link data.

Usage
.fusionsToLinkData(fusionList, minLinkWidth = 1, maxLinkWidth = 10)
chimeraviz-internals-scaleListToInterval

**Arguments**

- `fusionList` A list of Fusion objects.
- `minLinkWidth` The minimum link line width. Default = 1
- `maxLinkWidth` The maximum link line width. Default = 10

**Value**

A data frame with fusion link data compatible with `RCircos::RCircos.Link.Plot()`

```r
# @examples # Apparently examples shouldn't be set on private functions
defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz") fusions <- importDefuse(defuse833ke, "hg19", 3) linkData <- chimeraviz::.fusionsToLinkData(fusions) # This linkData can be used with `RCircos::RCircos.Link.Plot()`
```

---

**chimeraviz-internals-scaleListToInterval**

*Scale a vector of numeric values to an interval.*

**Description**

This function takes a vector of numeric values as well as an interval `[newMin, newMax]` that the numeric values will be scaled (normalized) to.

**Usage**

```r
.scaleListToInterval(theList, newMin, newMax)
```

**Arguments**

- `theList` A vector of numeric values.
- `newMin` Minimum value for the new interval.
- `newMax` Maximum value for the new interval.

**Value**

A data frame with fusion link data compatible with `RCircos::RCircos.Link.Plot()`

```r
# @examples # Apparently examples shouldn't be set on private functions
list012 <- c(0, 1, 2)
.scaleListToInterval(list012, 1, 3) # [1] 1 2 3
```
createFusionReport  Create a Fusion Report

Description

This function will create a html report with an overplot and a sortable, searchable table with the fusion data.

Usage

createFusionReport(fusions, outputFilename, quiet = TRUE)

Arguments

- fusions: A list of Fusion objects.
- outputFilename: Output html-file filename.
- quiet: Parameter passed to rmarkdown::render() to toggle its output.

Value

Creates a html report with an overplot and a sortable, searchable table with the fusion data.

Examples

```r
# Load data
defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 3)

# Temporary file to store the report
outputFilename <- tempfile(pattern = "fusionReport", fileext = "html", tmpdir = tempdir())

# Create report
createFusionReport(fusions, outputFilename)
```

decideTranscriptCategory

Retrieves transcripts for partner genes in a Fusion object using Ensembldb

Description

This function will check where in the transcript (the GRanges object) the fusion breakpoint is located, and return either "exonBoundary", "withinExon", "withinIntron", or "intergenic".
**Usage**

```
decideTranscriptCategory(gr, fusion)
```

**Arguments**

- `gr` The GRanges object containing the transcript to be checked.
- `fusion` The fusion object used to check the transcript.

**Value**

Either "exonBoundary", "withinExon", "withinIntron", or "intergenic" depending on where in the transcript the breakpoint hits.

**Examples**

```r
# Load fusion data and choose a fusion object:
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Create edb object
edbSqliteFile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)
# Get all exons for all transcripts in the genes in the fusion transcript
allTranscripts <- ensembldb::exonsBy(
 edb,
  filter = list(
    AnnotationFilter::GeneIdFilter(
      list(
        partnerGeneEnsemblId(upstreamPartnerGene(fusion)),
        partnerGeneEnsemblId(downstreamPartnerGene(fusion))))),
  columns = c(
    "gene_id",
    "gene_name",
    "tx_id",
    "tx_cds_seq_start",
    "tx_cds_seq_end",
    "exon_id"))
# Extract one of the GRanges objects
gr <- allTranscripts[[1]]
# Check where in the transcript the fusion breakpoint hits
decideTranscriptCategory(gr, fusion)
# "exonBoundary"
# Check another case
gr <- allTranscripts[[3]]
decideTranscriptCategory(gr, fusion)
# "withinIntron"
```
downShift  
 Remove introns and shift exons leftward

Description
This function takes a GRanges object and moves each IRanges object within next to each other
starting at 1. This effectively removes the introns from the GRanges object.

Usage

downShift(transcript)

Arguments

transcript  The GRanges object to remove introns from.

Value

A GRanges object with introns removed.

Examples

# Create a simple GRanges object:
gr <- IRanges::IRanges(
  start = c(13, 40, 100),
  end = c(20, 53, 110))
# Downshift it and see the introns are removed:
downShift(gr)

downstreamPartnerGene  
Get the downstream fusion partner gene

Description
This getter retrieves the downstream PartnerGene object.
This sets the downstream PartnerGene object of a Fusion object

Usage

downstreamPartnerGene(x)

## S4 method for signature 'Fusion'
downstreamPartnerGene(x)

downstreamPartnerGene(object) <- value

## S4 replacement method for signature 'Fusion'
downstreamPartnerGene(object) <- value
**fetchReadsFromFastq**

Fetch reads from fastq files

**Description**

This function will fetch read sequences from fastq files and put them into new fastq files.

**Usage**

```r
fetchReadsFromFastq(reads, fastqFileIn1, fastqFileIn2, fastqFileOut1, fastqFileOut2)
```

**Arguments**

- **reads** List of read IDs that is to be fetched.
- **fastqFileIn1** First fastq file to search in.
- **fastqFileIn2** Second fastq file to search in.
- **fastqFileOut1** First fastq file with results.
- **fastqFileOut2** Second fastq file with results.
Details

Note: This function runs (read only) bash commands on your system. Therefore the function will only work on a unix system.

Value

The files fastqFileOut1 and fastqFileOut2 populated with the specified reads.

Examples

```r
## Not run:
# fastq files that has the supporting reads
fastq1 <- system.file("extdata", "reads.1.fq", package="chimeraviz")
fastq2 <- system.file("extdata", "reads.2.fq", package="chimeraviz")
# Which read ids to extract
reads <- c(
  "13422259", "19375605", "29755061",
  "31632876", "32141428", "33857245")
# Extract the actual reads and put them in the tmp files "fastqFileOut1" and
# "fastqFileOut2"
fastqFileOut1 <- tempfile(pattern = "fq1", tmpdir = tempdir())
fastqFileOut2 <- tempfile(pattern = "fq2", tmpdir = tempdir())
fetchReadsFromFastq(reads, fastq1, fastq2,
  fastqFileOut1, fastqFileOut2)
# We now have the reads supporting fusion 5267 in the two files.
## End(Not run)
```

Fusion-class

An S4 class to represent a fusion event.

Description

The Fusion class represents a fusion event, holding data imported from a fusion tool.

Slots

- id  A unique id representing a fusion event. For deFuse data this will be the cluster id.
- fusionTool Name of the fusion tool.
- genomeVersion Name of the genome used to map reads.
- spanningReadsCount The number of spanning reads supporting the fusion.
- splitReadsCount The number of split reads supporting the fusion.
- fusionReadsAlignment A Gviz::AlignmentsTrack object holding the fusion reads aligned to the fusion sequence.
- geneA A PartnerGene object holding information of the upstream fusion partner gene.
- geneB A PartnerGene object holding information of the downstream fusion partner gene.
- inframe A logical value indicating whether or not the downstream fusion partner gene is inframe or not. Not all fusion-finders report this.
fusionToolSpecificData  A list that will hold fields of importance for a specific fusion finder. This field is used because many fusion-finders report important values that are hard to fit into a standardized format. Examples of values that are added to this list is probability from deFuse and EricScore from EricScript.

fusionSpanningReadsCount

*Get the spanning reads count from a Fusion object*

Description

This getter retrieves the spanning reads count from a Fusion object

Usage

```r
fusionSpanningReadsCount(x)
```

## S4 method for signature 'Fusion'
fusionSpanningReadsCount(x)

Arguments

- `x` The Fusion object you wish to retrieve the spanning reads count for.

Value

The Fusion spanning reads count.

Examples

```r
# Load data
defuseData <- system.file("extdata", "defuse_833ke_results.filterd.tsv", package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the spanning reads count
fusionSpanningReadsCount(fusion)
```

fusionSplitReadsCount

*Get the split reads count from a Fusion object*

Description

This getter retrieves the split reads count from a Fusion object
Usage
fusionSplitReadsCount(x)

## S4 method for signature 'Fusion'
fusionSplitReadsCount(x)

Arguments
x
The Fusion object you wish to retrieve the split reads count for.

Value
The Fusion split reads count.

Examples
# Load data
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz"
)fusion <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the split reads count
fusionSplitReadsCount(fusion)

fusionToDataFrame

Coerce Fusion object to data.frame

Description
This function is used in createFusionReport() to convert Fusion objects to a data.frame-format.

Usage
fusionToDataFrame(fusion)

Arguments
fusion
The Fusion object to coerce.

Value
A data.frame with the fusion object.

See Also
createFusionReport
Examples

# Load data
defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
# Find the fusion object to create a data frame from
fusion <- getFusionById(fusions, 5267)
# Create the data frame
dfFusion <- fusionToDataFrame(fusion)

getEnsemblIds

Get ensembl ids for a fusion object

Description

This function will get the ensembl ids from the org.Hs.eg.db package given the gene names of the fusion event.

Usage

getEnsemblIds(fusion)

Arguments

fusion  The Fusion object we want to get ensembl ids for.

Value

The Fusion object with Ensembl ids set.

Examples

# Import the filtered defuse results
defuse833keFiltered <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833keFiltered, "hg19", 1)
# Get a specific fusion
fusion <- getFusionById(fusions, 5267)
# See the ensembl ids:
partnerGeneEnsemblId(upstreamPartnerGene(fusion))
# [1] "ENSG00000180198"
partnerGeneEnsemblId(downstreamPartnerGene(fusion))
# [1] "ENSG00000162639"
# Reset the fusion objects ensembl ids
partnerGeneEnsemblId(upstreamPartnerGene(fusion)) <- ""
partnerGeneEnsemblId(downstreamPartnerGene(fusion)) <- ""
# Get the ensembl ids
fusion <- getEnsemblIds(fusion)
getFusionByGeneName

Find fusions that includes the given gene.

Description
Helper function to retrieve the Fusion objects that has geneName as one of the partner genes.

Usage
getFusionByGeneName(fusionList, geneName)

getFusionByChromosome
Find fusions that involves genes in the given chromosome.

Description
Helper function to retrieve the Fusion objects that involves genes in the given chromosome name.

Usage
getFusionByChromosome(fusionList, chr)

Arguments
- fusionList: A list of Fusion objects.
- chr: The chromosome name we’re looking for fusions in.

Value
A list of Fusion objects.

Examples
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz"
)fusions <- importDefuse(defuse833ke, "hg19", 1)
length(getFusionByChromosome(fusions, "chr1"))
# [1] 1
**getFusionById**

**Arguments**

- `fusionList` A list of Fusion objects.
- `geneName` The gene name we’re looking for.

**Details**

Note: `getFusionByGeneName(fusionList, "MT")` will match both MT-ND5 and MT-ND4.

**Value**

A list of Fusion objects.

**Examples**

```r
defuse833ke <- system.file(  
  "extdata",  
  "defuse_833ke_results.filtered.tsv",  
  package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
length(getFusionByGeneName(fusions, "RCC1"))
# [1] 1
```

---

**getFusionById**  
*Find a specific fusion object in a list by id*

**Description**

Helper function to retrieve the Fusion object with the given id.

**Usage**

`getFusionById(fusionList, id)`

**Arguments**

- `fusionList` A list of Fusion objects.
- `id` The id (e.g. the cluster_id from a deFuse run) we’re looking for.

**Value**

A Fusion object.

**Examples**

```r
defuse833ke <- system.file(  
  "extdata",  
  "defuse_833ke_results.filtered.tsv",  
  package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# This should be the Fusion object:
```
getTranscriptsEnsemblDb

Retrieves transcripts for partner genes in a Fusion object using EnsemblDb

Description

This function will retrieve transcripts for both genes in a fusion. It will check all transcripts and decide for each transcript if the fusion breakpoint happens at 1) an exon boundary, 2) within an exon, or 3) within an intron. This is done because fusions happening at exon boundaries are more likely to produce biologically interesting gene products. The function returns an updated Fusion object, where the fusion@geneA@transcriptsX slots are set with transcript information.

Usage

getTranscriptsEnsemblDb(fusion, edb)

Arguments

fusion The fusion object to find transcripts for.
edb The edb object used to fetch data from.

Value

An updated fusion object with transcript data stored.

Examples

# Load fusion data and choose a fusion object:
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Create edb object
edbSqliteFile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)
# Add transcripts data to fusion object
fusion <- getTranscriptsEnsemblDb(fusion, edb)
importDefuse

# The transcripts are now accessible through fusion@geneA@transcripts and
# fusion@geneB@transcripts.

importDefuse  

Import results from a deFuse run into a list of Fusion objects.

Description

A function that imports the results from a deFuse run, typically from a results.filtered.tsv file, into
a list of Fusion objects.

Usage

importDefuse(filename, genomeVersion, limit)

Arguments

filename
Filename for the deFuse results .tsv file.

genomeVersion
Which genome was used in mapping (hg19, hg38, etc.).

limit
A limit on how many lines to read.

Value

A list of Fusion objects.

Examples

defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz"
)fusions <- importDefuse(defuse833ke, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.

importEricscript

Import results from a EricScript run into a list of Fusion objects.

Description

A function that imports the results from a EricScript run into a list of Fusion objects.

Usage

importEricscript(filename, genomeVersion, limit)

**Arguments**

- `genomeVersion`: Which genome was used in mapping (hg19, hg38, etc.).
- `limit`: A limit on how many lines to read.

**Value**

A list of Fusion objects.

**Examples**

```r
ericscriptData <- system.file(
  "extdata",
  "ericscript_SRR1657556.results.total.tsv",
  package = "chimeraviz")
fusions <- importEricscript(ericscriptData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

---

**importFunctionNonUCSC**  
**Alternative import function for Gviz::AlignmentsTrack**

**Description**

This alternative import function for use with Gviz::AlignmentsTrack imports a bamfile with non-UCSC chromosome names.

**Usage**

```r
importFunctionNonUCSC(file, selection)
```

**Arguments**

- `file`: The bamfile.
- `selection`: Which regions to get from the bamfile.

**Value**

A GRanges object with coverage data for the selection.
importFusioncatcher  Import results from a Fusioncatcher run into a list of Fusion objects.

Description
A function that imports the results from a Fusioncatcher run, typically from a final-list-candidate-fusion-genes.txt file, into a list of Fusion objects.

Usage
importFusioncatcher(filename, genomeVersion, limit)

Arguments
filename Filename for the Fusioncatcher final-list-candidate-fusion-genes.txt results file.
genomeVersion Which genome was used in mapping (hg19, hg38, etc.).
limit A limit on how many lines to read.

Value
A list of Fusion objects.

Examples
fusioncatcher833ke <- system.file(
  "extdata",
  "fusioncatcher_833ke_final-list-candidate-fusion-genes.txt",
  package = "chimeraviz")
fusions <- importFusioncatcher(fusioncatcher833ke, "hg38", 3)
# This should import a list of 3 fusions described in Fusion objects.

importFusionmap  Import results from a FusionMap run into a list of Fusion objects.

Description
A function that imports the results from a FusionMap run, typically from a InputFastq.FusionReport.txt file, into a list of Fusion objects.

Usage
importFusionmap(filename, genomeVersion, limit)

Arguments
filename Filename for the FusionMap PairedEndFusionReport.txt results file.
genomeVersion Which genome was used in mapping (hg19, hg38, etc.).
limit A limit on how many lines to read.
importInfusion

Value
A list of Fusion objects.

Examples
fusionmapData <- system.file(
  "extdata",
  "FusionMap_01_TestDataset_InputFastq.FusionReport.txt",
  package = "chimeraviz")
fusions <- importFusionmap(fusionmapData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.

importInfusion Import results from an InFusion run into a list of Fusion objects.

Description
A function that imports the results from an InFusion run into a list of Fusion objects.

Usage
importInfusion(filename, genomeVersion, limit)

Arguments
filename Filename for the jaffa_results.csv file.
genomeVersion Which genome was used in mapping (hg19, hg38, etc.).
limit A limit on how many lines to read.

Value
A list of Fusion objects.

Examples
infusionData <- system.file(
  "extdata",
  "infusion_fusions.txt",
  package = "chimeraviz")
fusions <- importInfusion(infusionData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
importJaffa

Description
A function that imports the results from a JAFFA run, typically from a jaffa_results.csv file, into a list of Fusion objects.

Usage
importJaffa(filename, genomeVersion, limit)

Arguments
- genomeVersion: Which genome was used in mapping (hg19, hg38, etc.).
- limit: A limit on how many lines to read.

Value
A list of Fusion objects.

Examples
jaffaData <- system.file("extdata", "jaffa_results.csv", package = "chimeraviz")
fusions <- importJaffa(jaffaData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.

importPrada

Description
A function that imports the results from a PRADA run into a list of Fusion objects.

Usage
importPrada(filename, genomeVersion, limit)

Arguments
- filename: Filename for the PRADA results file.
- genomeVersion: Which genome was used in mapping (hg19, hg38, etc.).
- limit: A limit on how many lines to read.
importSoapfuse

Value
A list of Fusion objects.

Examples
pradaData <- system.file(
  "extdata",
  "PRADA.acc.fusion fq.TAF.tsv",
  package = "chimeraviz")
fusions <- importPrada(pradaData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.

importSoapfuse

Import results from a SOAPfuse run into a list of Fusion objects.

Description
A function that imports the results from a SOAPfuse run, typically from a final.Fusion.specific.for.genes file, into a list of Fusion objects.

Usage
importSoapfuse(filename, genomeVersion, limit)

Arguments
filename Filename for the SOAPfuse final-list-candidate-fusion-genes.txt results file.
genomeVersion Which genome was used in mapping (hg19, hg38, etc.).
limit A limit on how many lines to read.

Value
A list of Fusion objects.

Examples
soapfuse833ke <- system.file(
  "extdata",
  "soapfuse_833ke_final.Fusion.specific.for.genes",
  package = "chimeraviz")
fusions <- importSoapfuse(soapfuse833ke, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
importStarfusion

Import results from a STAR-Fusion run into a list of Fusion objects.

**Description**

A function that imports the results from a STAR-Fusion run, typically from a star-fusion.fusion_candidates.final.abridged file, into a list of Fusion objects.

**Usage**

importStarfusion(filename, genomeVersion, limit)

**Arguments**

- **filename**: Filename for the STAR-Fusion star-fusion.fusion_candidates.final.abridged results file.
- **genomeVersion**: Which genome was used in mapping (hg19, hg38, etc.).
- **limit**: A limit on how many lines to read.

**Value**

A list of Fusion objects.

**Examples**

```
starfusionData <- system.file(
  "extdata",
  "star-fusion.fusion_candidates.final.abridged.txt",
  package = "chimeraviz")
fusions <- importStarfusion(starfusionData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

**PartnerGene-class**

An S4 class to represent a gene partner in a fusion

**Description**

The PartnerGene class represents one of the genes in a fusion event.

**Slots**

- **name**: Character containing name of the gene.
- **ensemblId**: Character containing ensembl id for the gene.
- **chromosome**: Character containing chromosome name.
- **breakpoint**: Numeric containing the fusion breakpoint.
- **strand**: Character containing gene strand.
junctionSequence BiostRings::DNAString containing the sequence right before/after the fusion breakpoint.

transcripts GenomicRanges::GRangesList containing three GenomicRanges::Granges() objects, one for each "transcript type". The transcript types are: 1) Transcripts where the fusion breakpoint hits an exon boundary, 2) transcripts where the fusion breakpoint is within an exon, 3) transcripts where the fusion breakpoint is within an intron.

**partnerGeneEnsemblId**  
*Get the Ensembl ID from a PartnerGene object*

**Description**

This getter retrieves the Ensembl ID from a PartnerGene object

This sets the Ensembl ID of a PartnerGene object.

**Usage**

```r
partnerGeneEnsemblId(x)
```

```r
## S4 method for signature 'PartnerGene'
partnerGeneEnsemblId(x)
```

```r
partnerGeneEnsemblId(object) <- value
```

```r
## S4 replacement method for signature 'PartnerGene'
partnerGeneEnsemblId(object) <- value
```

**Arguments**

- **x** The PartnerGene object you wish to retrieve the Ensembl ID for.
- **object** The PartnerGene object you wish to set a new Ensembl ID for.
- **value** The new Ensembl ID.

**Value**

The upstream fusion partner gene Ensembl ID.

**Examples**

```r
# Load data
defuseData <- system.file(  
  "extdata",  
  "defuse_833ke_results.filtered.tsv",  
  package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the Ensembl ID from the upstream fusion partner gene
partnerGeneEnsemblId(upstreamPartnerGene(fusion))
```

```r
# Load data
defuseData <- system.file(  
"extdata",  
"defuse_833ke_results.filtered.tsv",  
package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the Ensembl ID from the upstream fusion partner gene
partnerGeneEnsemblId(upstreamPartnerGene(fusion))
```
partnerGeneJunctionSequence

Get the junction sequence from a PartnerGene object

Description

This getter retrieves the junction sequence from a PartnerGene object

Usage

partnerGeneJunctionSequence(x)

## S4 method for signature 'PartnerGene'
partnerGeneJunctionSequence(x)

Arguments

x

The PartnerGene object you wish to retrieve the junction sequence for.

Value

The upstream fusion partner gene junction sequence.

Examples

# Load data
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Set the downstream PartnerGene object to be the same as the upstream PartnerGene object
partnerGeneEnsemblId(upstreamPartnerGene(fusion)) <- "test"

# Get the junction sequence from the upstream fusion partner gene
partnerGeneJunctionSequence(upstreamPartnerGene(fusion))
Create a circle plot of the given fusions.

Description

This function takes a list of Fusion objects and creates a circle plot indicating which chromosomes the fusion genes in the list consists of.

Usage

plotCircle(fusionList)

Arguments

fusionList A list of Fusion objects.

Details

Note that only a limited number of gene names can be shown in the circle plot due to the limited resolution of the plot. RCircos will automatically limit the number of gene names shown if there are too many. Also note that fusions involving mitochondrial DNA will not be shown in this plot.

Value

Creates a circle plot.

Examples

defuse833ke <- system.file("extdata", "defuse_833ke_results_filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 3)
# Temporary file to store the plot
pngFilename <- tempfile(pattern = "circlePlot", fileext = ".png", tmpdir = tempdir())
# Open device
png(pngFilename, width = 1000, height = 750)
# Plot!
plotCircle(fusions)
# Close device
dev.off()
plotFusion

Plot a fusion event with transcripts, coverage and ideograms.

Description

This function creates a plot with information about transcripts, coverage, location and more.

Usage

plotFusion(fusion, edb = NULL, bamfile = NULL,
            whichTranscripts = "exonBoundary", ylim = c(0, 1000), nonUCSC = TRUE,
            reduceTranscripts = FALSE, bedgraphfile = NULL)

plotFusionSeparate(fusion, edb, bamfile = NULL,
                   whichTranscripts = "exonBoundary", ylim = c(0, 1000), nonUCSC = TRUE,
                   reduceTranscripts = FALSE, bedgraphfile = NULL)

plotFusionTogether(fusion, edb, bamfile = NULL,
                    whichTranscripts = "exonBoundary", ylim = c(0, 1000), nonUCSC = TRUE,
                    reduceTranscripts = FALSE, bedgraphfile = NULL)

Arguments

fusion The Fusion object to plot.
edb The ensembldb object that will be used to fetch data.
bamfile The bamfile with RNA-seq data.
whichTranscripts This character vector decides which transcripts are to be plotted. Can be "exonBoundary", "withinExon", "withinIntron", "intergenic", or a character vector with specific transcript ids. Default value is "exonBoundary".
ylim Limits for the coverage y-axis.
nonUCSC Boolean indicating whether or not the bamfile used has UCSC- styled chromosome names (i.e. with the "chr" prefix). Setting this to true lets you use a bamfile with chromosome names like "1" and "X", instead of "chr1" and "chrX".
reduceTranscripts Boolean indicating whether or not to reduce all transcripts into a single transcript for each partner gene.
bedgraphfile A bedGraph file to use instead of the bamfile to plot coverage.

Details

plotFusion() will dispatch to either plotFusionSeparate() or plotFusionTogether(). plotFusionSeparate() will plot the fusion gene partners in separate graphs shown next to each other, while plotFusionTogether() will plot the fusion gene partners in the same graph with the same x-axis. plotFusion() will dispatch to plotFusionTogether() if the fusion gene partners are on the same strand, same chromosome and are close together (\(\leq 50,000\) bp apart).

Value

Creates a fusion plot.
Examples

# Load data and example fusion event
defuse833ke <- system.file(  
    "extdata",  
    "defuse_833ke_results.filtered.tsv",  
    package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)

# Load edb
edbSqlitefile <- system.file(  
    "extdata",  
    "Homo_sapiens.GRCh37.74.sqlite",  
    package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)

# bamfile with reads in the regions of this fusion event
bamfile5267 <- system.file(  
    "extdata",  
    "fusion5267and11759reads.bam",  
    package="chimeraviz")

# Temporary file to store the plot
pngFilename <- tempfile(  
    pattern = "fusionPlot",  
    fileext = ".png",  
    tmpdir = tempdir())

# Open device
png(pngFilename, width = 1000, height = 750)

# Plot!
plotFusion(  
    fusion = fusion,  
    bamfile = bamfile5267,  
    edb = edb,  
    nonUCSC = TRUE)

# Close device
dev.off()

# Example using a .bedGraph file instead of a .bam file:
# Load data and example fusion event
defuse833ke <- system.file(  
    "extdata",  
    "defuse_833ke_results.filtered.tsv",  
    package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)

# Load edb
edbSqlitefile <- system.file(  
    "extdata",  
    "Homo_sapiens.GRCh37.74.sqlite",  
    package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)

# bedgraphfile with coverage data from the regions of this fusion event
bedgraphfile <- system.file(  
    "extdata",  
    "fusion5267and11759reads.bedGraph",  
    package="chimeraviz")

# Temporary file to store the plot
pngFilename <- tempfile(  
    pattern = "fusionPlot",  
    fileext = ".png",  
    tmpdir = tempdir())

# Open device
png(pngFilename, width = 1000, height = 750)

# Plot!
plotFusion(  
    fusion = fusion,  
    bamfile = bamfile5267,  
    edb = edb,  
    nonUCSC = TRUE)

# Close device
dev.off()
plotFusionReads

pattern = "fusionPlot",
fileext = ".png",
tmpdir = tempdir())
# Open device
png(pngFilename, width = 1000, height = 750)
# Plot!
plotFusion(
    fusion = fusion,
    bedgraphfile = bedgraphfile,
    edb = edb,
    nonUCSC = TRUE)
# Close device
dev.off()

plotFusionReads Create a plot of the reads supporting the given fusion.

Description

This function takes a Fusion object and plots the reads supporting the fusion on top of the fusion sequence (fusion@junctionSequence), provided that addFusionReadsAlignment() has been run earlier in order to add fusion reads alignment data to the fusion object.

Usage

plotFusionReads(fusion, showAllNucleotides = TRUE, nucleotideAmount = 10)

Arguments

fusion The Fusion object to plot.
showAllNucleotides Boolean indicating whether or not to show all nucleotides. If FALSE, then only nucleotideAmount amount of nucleotides will be shown on each end of the fusion junction. If TRUE, then the whole fusion junction sequence will be shown.
nucleotideAmount The number of nucleotides to show on each end of the fusion junction sequence. Defaults to 10. Only applicable if showAllNucleotides is set to TRUE.

Details

Note that the package used for plotting, Gviz, is strict on chromosome names. If the plot produced doesn’t show the reads, the problem might be solved by naming the fusion sequence "chrNA".

Value

Creates a fusion reads plot.

See Also

addFusionReadsAlignment
Examples

# Load data
defuseData <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
# Find the specific fusion we have aligned reads for
fusion <- getFusionById(fusions, 5267)
# Add the bam file of aligned fusion reads to the fusion object
fusion <- addFusionReadsAlignment(fusion, bamfile)
# Temporary file to store the plot
pngFilename <- tempfile(pattern = "fusionPlot", fileext = ".png", tmpdir = tempdir())
# Calculate image size based on supporting reads and length of junction sequence.
imageWidth <- (nchar(partnerGeneJunctionSequence(upstreamPartnerGene(fusion))) +
               nchar(partnerGeneJunctionSequence(downstreamPartnerGene(fusion)))) * 15
imageHeight <- (fusionSplitReadsCount(fusion)+fusionSpanningReadsCount(fusion)) * 20
# Open device
png(pngFilename, width = imageWidth, height = imageHeight)
# Now we can plot
plotFusionReads(fusion)
# Close device
dev.off()

plotFusionTranscript  Plot possible fusion transcripts based on annotation.

Description

This function takes a fusion object and an ensembldb object and plots the reduced version of the fusion transcript. This transcript consist of the "mashed together" version of all possible fusion transcripts based on known annotations. See plotFusionTranscripts() for the not-reduced version of this plot. If a bamfile is specified, the fusion transcript will be plotted with coverage information.

Usage

plotFusionTranscript(fusion, edb = NULL, bamfile = NULL,
                      whichTranscripts = "exonBoundary", bedgraphfile = NULL)

Arguments

fusion  The Fusion object to plot.
edb     The edb object that will be used to fetch data.
bamfile The bamfile with RNA-seq data.
whichTranscripts

This character vector decides which transcripts are to be plotted. Can be "exonBoundary", "withinExon", "withinIntron", "intergenic", or a character vector with specific transcript ids. Default value is "exonBoundary".

bedgraphfile

A bedGraph file to use instead of the bamfile to plot coverage.

Details

Note that the transcript database used (the edb object) must have the same seqnames as any bamfile used. Otherwise the coverage data will be wrong.

Value

Creates a fusion transcript plot.

Examples

# Load data and example fusion event
defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)

# Load edb
edbSqliteFile <- system.file("extdata", "Homo_sapiens.GRCh37.74.sqlite", package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)

# bamfile with reads in the regions of this fusion event
bamfile5267 <- system.file("extdata", "fusion5267and11759reads.bam", package="chimeraviz")

# Temporary file to store the plot
pngFilename <- tempfile(pattern = "fusionPlot", fileext = ".png", tmpdir = tempdir())

# Open device
png(pngFilename, width = 500, height = 500)

# Plot!
plotFusionTranscript(
  fusion = fusion,
  bamfile = bamfile5267,
  edb = edb)

# Close device
dev.off()

# Example using a .bedGraph file instead of a .bam file:
# Load data and example fusion event
defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
plotFusionTranscriptsGraph

Graph plot of possible fusion transcripts.

Description

This function takes a fusion object and a TranscriptDb object and plots a graph showing the possible fusion transcripts.

Usage

plotFusionTranscriptsGraph(fusion, edb = NULL, 
whichTranscripts = "exonBoundary", rankdir = "TB")

Arguments

fusion The Fusion object to plot.
edb The edb object that will be used to fetch data.
whichTranscripts This character vector decides which transcripts are to be plotted. Can be "exonBoundary", "withinExon", "withinIntron", "intergenic", or a character vector with specific transcript ids. Default value is "exonBoundary".
rankdir Choose whether the graph should be plotted from left to right ("LR"), or from top to bottom ("TB"). This parameter is given to Rgraphviz::plot().
plotFusionTranscriptWithProteinDomain

Value

Creates a fusion transcripts graph plot.

Examples

```r
# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Load edb
edbSqliteFile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)
# Temporary file to store the plot
pngFilename <- tempfile(
  pattern = "fusionPlot",
  fileext = ".png",
  tmpdir = tempdir())
# Open device
png(pngFilename, width = 500, height = 500)
# Plot!
plotFusionTranscriptsGraph(
  fusion = fusion,
  edb = edb)
# Close device
dev.off()
```

plotFusionTranscriptWithProteinDomain

Plot a specific fusion transcript with protein domain annotations

Description

This function takes a fusion object, an ensembldb object, a bedfile with protein domain data and two specific transcript ids. The function plots the specific fusion transcript along with annotations of protein domains. If a bamfile is specified, the fusion transcript will be plotted with coverage information.

Usage

```r
plotFusionTranscriptWithProteinDomain(fusion, edb = NULL, bamfile = NULL,
  bedfile = NULL, geneATranscript = "", geneBtranscript = "",
  plotDownstreamProteinDomainsIfFusionIsOutOfFrame = FALSE)
```
plotFusionTranscriptWithProteinDomain

Arguments

- **fusion**: The Fusion object to plot.
- **edb**: The edb object that will be used to fetch data.
- **bamfile**: The bamfile with RNA-seq data.
- **bedfile**: The bedfile with protein domain data.
- **geneAtranscript**: The transcript id for the upstream gene.
- **geneBtranscript**: The transcript id for the downstream gene.
- **plotDownstreamProteinDomainsIfFusionIsOutOfFrame**: Setting this to true makes the function plot protein domains in the downstream gene even though the fusion is out of frame.

Details

Note that the transcript database used (the edb object) must have the same seqnames as any bamfile used. Otherwise the coverage data will be wrong.

Value

Creates a fusion transcript plot with annotations of protein domains.

Examples

```r
# Load data and example fusion event
defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Select transcripts
geneAtranscript <- "ENST00000434290"
geneBtranscript <- "ENST00000370031"
# Load edb
edbSqliteFile <- system.file("extdata", "Homo_sapiens.GRCh37.74.sqlite", package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)
# bamfile with reads in the regions of this fusion event
bamfile5267 <- system.file("extdata", "fusion5267and11759reads.bam", package="chimeraviz")
# bedfile with protein domains for the transcripts in this example
bedfile5267 <- system.file("extdata", "protein_domains_5267.bed", package="chimeraviz")
# Temporary file to store the plot
pngFilename <- tempfile(pattern = "fusionPlot", fileext = ".png")
```
```
plotTranscripts

tmpdir = tempdir()
# Open device
png(pngFilename, width = 500, height = 500)
# Plot!
plotFusionTranscriptWithProteinDomain(
  fusion = fusion,
  edb = edb,
  bamfile = bamfile5267,
  bedfile = bedfile,
  geneAtranscript = geneAtranscript,
  geneBtranscript = geneBtranscript,
  plotDownstreamProteinDomainsIfFusionIsOutOfFrame = TRUE)
# Close device
dev.off()
```

---

**plotTranscripts**  
*Plot transcripts for each partner gene in a fusion event.*

**Description**

This function takes a fusion object and an ensemblldb object and plots transcripts for both genes, showing which parts of each genes are included in the fusion event. If the bamfile parameter is set, then the coverage is plotted beneath the transcripts.

**Usage**

```r
plotTranscripts(fusion, edb = NULL, bamfile = NULL,  
                  whichTranscripts = "exonBoundary", nonUCSC = TRUE, ylim = c(0, 1000),  
                  reduceTranscripts = FALSE, bedgraphfile = NULL)
```

**Arguments**

- `fusion`  
The Fusion object to plot.
- `edb`  
The edb object that will be used to fetch data.
- `bamfile`  
The bamfile with RNA-seq data.
- `whichTranscripts`  
  This character vector decides which transcripts are to be plotted. Can be "exonBoundary", "withinExon", "withinIntron", "intergenic", or a character vector with specific transcript ids. Default value is "exonBoundary".
- `nonUCSC`  
  Boolean indicating whether or not the bamfile used has UCSC- styled chromosome names (i.e. with the "chr" prefix). Setting this to true lets you use a bamfile with chromosome names like "1" and "X", instead of "chr1" and "chrX".
- `ylim`  
  Limits for the coverage y-axis.
- `reduceTranscripts`  
  Boolean indicating whether or not to reduce all transcripts into a single transcript for each partner gene.
- `bedgraphfile`  
  A bedGraph file to use instead of the bamfile to plot coverage.

**Value**

Creates a fusion transcripts plot.
Examples

# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz"
)fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Load edb
edbSqlitefile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz"
)edb <- ensembldb::EnsDb(edbSqliteFile)
# bamfile with reads in the regions of this fusion event
bamfile5267 <- system.file(
  "extdata",
  "fusion5267and11759reads.bam",
  package="chimeraviz"
)# Temporary file to store the plot
pngFilename <- tempfile(
  pattern = "fusionPlot",
  fileext = ".png",
  tmpdir = tempdir())
# Open device
gray(pngFilename, width = 500, height = 500)
# Plot!
plotTranscripts(
  fusion = fusion,
  edb = edb,
  bamfile = bamfile5267,
  nonUCSC = TRUE)
# Close device
dev.off()

# Example using a .bedGraph file instead of a .bam file:
# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz"
)fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Load edb
edbSqlitefile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz"
)edb <- ensembldb::EnsDb(edbSqliteFile)
# bedgraphfile with coverage data from the regions of this fusion event
bedgraphfile <- system.file(
  "extdata",
  "fusion5267and11759reads.bedGraph",
  package="chimeraviz"
)# Temporary file to store the plot
pngFilename <- tempfile(
raw_cytobandhg19

```r
pattern = "fusionPlot",
fileext = ".png",
tmpdir = tempdir()
# Open device
png(pngFilename, width = 500, height = 500)
# Plot!
plotTranscripts(
  fusion = fusion,
  edb = edb,
  bedgraphfile = bedgraphfile,
  nonUCSC = TRUE)
# Close device
dev.off()
```

---

**raw_cytobandhg19**  
*Cytdoband information HG19*

### Description


UCSC.HG19.Human.CytoBandIdeogram.txt

This data is used with RCircos in plotCircle().

---

**raw_cytobandhg38**  
*Cytdoband information HG138*

### Description


UCSC.HG38.Human.CytoBandIdeogram.txt

This data is used with RCircos in plotCircle().

### Details

All _alt or _random entries has been manually removed, as has the chrM entry.
Description

Documentation for the deFuse example data.

defuse_833ke_results.filtered.tsv

This file has the results from a run of deFuse-0.7.0 on the 833ke cell line. The program was ran with the standard configuration, but with the parameter span_count_threshold=5 instead of the standard 3. The resulting results.filtered.tsv file was then manually filtered to only include 17 fusion events in the interest of saving computing time for tests and examples. The original results contained 171 fusion events.

reads_supporting_defuse_fusion_5267.*.fq

These two files, reads_supporting_defuse_fusion_5267.1.fq and reads_supporting_defuse_fusion_5267.2.fq, contains the reads that support the fusion event with cluster_id 5267.

5267readsAligned.bam

The bamfile 5267readsAligned.bam and the 5267readsAligned.bam.bai index file contains the reads supporting the fusion event with cluster_id 5267 aligned to the fusion sequence. It is used with plotFusionReads().

Description

Documentation for the EricScript example data.

ericScript_SRR1657556.results.total.tsv

This is example data thankfully provided by EricScript author Matteo Benelli.
protein_domains_5267.bed

This file is an excerpt from a larger file that we created by: - downloading domain name annotation from Pfam database (PfamA version 31) and domain region annotation from Ensembl database through BioMart API - switching the domain coordinates in the protein level to these in transcript level.

fusion5267and11759reads.bam

This file is the result of running these commands:

```
samtools view -b original_bamfile.bam "1:28831455-28866812" "1:109189912-109205148" "12:8608225-8677832" > fusion5267and11759reads.bam
samtools index fusion5267and11759reads.bam
```

where we extract the reads mapping to the region where we know the fusions with cluster_id=5267 and cluster_id=11759 is.

The original_bamfile.bam is from a study of the 833KE cell line by Andreas M. Hoff et al., documented in the paper [Identification of Novel Fusion Genes in Testicular Germ Cell Tumors](http://cancerres.aacrjournals.org/content/76/1/108.full).

fusion5267and11759reads.bedGraph

This file is the result of running this command:

```
bedtools genomecov -ibam fusion5267and11759reads.bam -bg > fusion5267and11759reads.bam.bedGraph
```

fusion5267and11759reads.bam has its own documentation entry for how it was created.
**raw_fusioncatcher**  
Fusioncatcher data

**Description**

Documentation for the Fusioncatcher example data.

**fusioncatcher_833ke_final-list-candidate-fusiongenes.txt**

This file has the results from a run of Fusioncatcher-0.99.3e on the 833ke cell line. The program was ran with the standard configuration file and with the parameters "-p 8 -z –keep-preliminary".

---

**raw_fusionmap**  
FusionMap data

**Description**

Documentation for the FusionMap example data.

**FusionMap_01_TestDataset_InputFastq_FusionReport.txt**

This is example data provided with the FusionMap version released 2015-03-31.

---

**raw_Homo_sapiens.GRCh37.74**

**Homo_sapiens.GRCh37.74_subset.gtf**

**Description**

Homo_sapiens.GRCh37.74_subset.gtf

**Homo_sapiens.GRCh37.74_subset.gtf**

The Homo_sapiens.GRCh37.74.gtf file is a subset version of the Ensembl Homo_sapiens.GRCh37.74.gtf file, located here: ftp://ftp.ensembl.org/pub/release-74/gtf/homo_sapiens. This gtf file contains transcripts for the partner genes in two of the fusion transcripts from the deFuse example data provided with this package: The fusion transcript with cluster_id=5267, and the fusion transcript with cluster_id=11759.

The file is the result of running this command:

```
# grep "ENST00000373832|ENST00000373833|ENST00000373834|ENST00000373835|ENST00000000000411533|ENST00000000000411534|ENST00000000000411535|ENST00000000000411536|ENST00000000000411537" Homo_sapiens.GRCh37.74.gtf > Homo_sapiens.GRCh37.74_subset.gtf
```

The transcript names given in the command above are all transcripts available for the genes CLEC6A, CLEC4D, HENMT1, and RCC1 in Ensembl version 74.

**Homo_sapiens.GRCh37.74.sqlite**

The Homo_sapiens.GRCh37.74.sqlite file is the sqlite database that the Ensembldb package creates from the corresponding gtf file. It was created using this command:

```
# ensDbFromGtf( # gtf = "Homo_sapiens.GRCh37.74_subset.gtf", # organism = "Homo_sapiens", # genomeVersion = "GRCh37", # version = 74)
```
raw_infusion  

**Description**  
Documentation for the InFusion example data.

**infusion_fusions.txt**  
This is example data from the InFusion getting started page located at https://bitbucket.org/kokonech/infusion/wiki/Getting

raw_jaffa  

**Description**  
Documentation for the JAFFA example data.

**jaffa_results.csv**  
This is example data from the described JAFFA example run documented at https://github.com/Oshlack/JAFFA/wiki/Example.

raw_prada  

**Description**  
Documentation for the PRADA example data.

**PRADA.acc.fusion.fq.TAF.tsv**  
This is example data thankfully provided by PRADA authors Siyuan Zheng and Roeland Verhaak.

raw_soapfuse  

**Description**  
Documentation for the SOAPfuse example data.

**soapfuse_833ke_final.Fusion.specific.for.genes**  
This file has the results from a run of soapfuse-1.26 on the 833ke cell line. The program was ran with the standard configuration file.
**selectTranscript**

```
raw_starfusion   STAR-Fusion data
```

**Description**

Documentation for the STAR-Fusion example data.

**star-fusion.fusion_candidates.final.abridged.txt**

This example data was retrieved from the STAR-Fusion github page June 2nd 2017.

**selectTranscript**

**Select which transcript to use (for plotting) for a GenePartner object**

**Description**

This function takes a GenePartner object and creates a transcript data.frame with transcript information, including only the transcripts given by the parameter whichTranscripts.

**Usage**

```
selectTranscript(genePartner, whichTranscripts = "exonBoundary")
```

**Arguments**

- `genePartner` The GenePartner object to select a transcript for.
- `whichTranscripts` This character vector decides which transcripts are to be plotted. Can be "exonBoundary", "withinExon", "withinIntron", "intergenic", or a character vector with specific transcript ids. Default value is "exonBoundary".

**Details**

`selectTranscript()` selects which transcript to create by this prioritization:

1. Exon boundary transcripts. 2. Within exon transcripts. 3. Within intron transcripts. 4. Intergenic transcripts.

**Value**

A data.frame with transcript data.
show,Fusion-method

Examples

# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- importDefuse(defuse833ke, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Load edb
edbSqliteFile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz")
edb <- ensmbldb::EnsDb(edbSqliteFile)
# Get transcripts
fusion <- getTranscriptsEnsembldb(fusion, edb)
# Select transcript
transcriptsA <- selectTranscript(upstreamPartnerGene(fusion))

---

show,Fusion-method

Show method for the Fusion class.

Description

Show method for the Fusion class.

Usage

## S4 method for signature 'Fusion'
show(object)

Arguments

object A Fusion object

Value

Shows information about a Fusion object.

---

show,PartnerGene-method

Show method for the PartnerGene class.

Description

Show method for the PartnerGene class.

Usage

## S4 method for signature 'PartnerGene'
show(object)
**splitOnUtrAndAddFeature**

**Arguments**

object A PartnerGene object

**Value**

Shows information about a PartnerGene object.

---

**splitOnUtrAndAddFeature**

*Split GRanges object based on cds*

**Description**

This function will look for ranges (exons) in the GRanges object that has the coding DNA sequence starting or stopping within it. If found, these exons are split, and each exon in the GRanges object will be tagged as either "protein_coding", "5utr", or "3utr". The returned GRanges object will have feature values set in mcols(gr)$feature reflecting this.

**Usage**

`splitOnUtrAndAddFeature(gr)`

**Arguments**

gr The GRanges object we want to split and tag with feature info.

**Value**

An updated GRanges object with feature values set.

**Examples**

```r
# Load fusion data and choose a fusion object:
defuseData <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- getFusionById(fusions, 5267)
# Create edb object
edbSqliteFile <- system.file("extdata", "Homo_sapiens.GRCh37.74.sqlite", package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)
# Get all exons for all transcripts in the genes in the fusion transcript
allTranscripts <- ensembldb::exonsBy(
edb, filter = list(
  AnnotationFilter::GeneIdFilter(
    list(
      partnerGeneEnsemblId(upstreamPartnerGene(fusion)),
      partnerGeneEnsemblId(downstreamPartnerGene(fusion)))
    )))
```
columns = c(
  "gene_id",
  "gene_name",
  "tx_id",
  "tx_cds_seq_start",
  "tx_cds_seq_end",
  "exon_id")

# Extract one of the GRanges objects
gr <- allTranscripts[[1]]

# Check how many ranges there are here
length(gr)
# Should be 9 ranges

# Split the ranges containing the cds start/stop positions and add feature
# values:
gr <- splitOnUtrAndAddFeature(gr)

# Check the length again
length(gr)
# Should be 11 now, as the range containing the cds_strat position and the
# range containing the cds_stop position has been split into separate ranges

---

### upstreamPartnerGene

*Get the upstream fusion partner gene*

#### Description

This getter retrieves the upstream PartnerGene object.

This sets the upstream PartnerGene object of a Fusion object

#### Usage

```r
upstreamPartnerGene(x)
```

```r
## S4 method for signature 'Fusion'
upstreamPartnerGene(x)
```

```r
upstreamPartnerGene(object) <- value
```

```r
## S4 replacement method for signature 'Fusion'
upstreamPartnerGene(object) <- value
```

#### Arguments

- **x**
  - The Fusion object you wish to retrieve the upstream PartnerGene object for.

- **object**
  - The Fusion object you wish to set a new upstream PartnerGene object for.

- **value**
  - The new PartnerGene object.

#### Value

The upstream PartnerGene object.
Examples

# Load data
defuseData <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the upstream fusion partner gene
upstreamPartnerGene(fusion)

# Load data
defuseData <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Set the upstream PartnerGene object to be the same as the downstream PartnerGene object
upstreamPartnerGene(fusion) <- downstreamPartnerGene(fusion)

writeFusionReference  Write fusion junction sequence to a fasta file

Description

This function will write the fusion sequence to a fasta file, using Biostring::writeXStringSet().

Usage

writeFusionReference(fusion, filename)

Arguments

  fusion  The Fusion object we want to create a fasta file from.
  filename The filename to write to.

Value

Writes the fusion junction sequence to the given filename.

Examples

# Import the filtered defuse results
defuse833keFiltered <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz")
fusions <- importDefuse(defuse833keFiltered, "hg19", 1)
# Get a specific fusion
fusion <- getFusionById(fusions, 5267)
# Create temporary file to hold the fusion sequence
fastaFileOut <- tempfile(pattern = "fusionSequence", tmpdir = tempdir())
# Write fusion sequence to file
writeFusionReference(fusion, fastaFileOut)
Index

.fusionsToGeneLabelData
  (chimeraviz-internals-fusionsToGeneLabelData)
  4
.fusionsToLinkData
  (chimeraviz-internals-fusionsToLinkData)
  4
.scaleListToInterval
  (chimeraviz-internals-scaleListToInterval)
  5

addFusionReadsAlignment, 3
chimeraviz, 4
chimeraviz-internals-fusionsToGeneLabelData, 4
chimeraviz-internals-fusionsToLinkData, 4
chimeraviz-internals-scaleListToInterval, 5
chimeraviz-package (chimeraviz), 4
createFusionReport, 6
decideTranscriptCategory, 6
downShift, 8
downstreamPartnerGene, 8
downstreamPartnerGene, Fusion-method
downstreamPartnerGene, Fusion-method
downstreamPartnerGene<-
downstreamPartnerGene<-, Fusion-method
fetchReadsFromFastq, 9
Fusion (Fusion-class), 10
Fusion-class, 10
fusionSpanningReadsCount, 11
fusionSpanningReadsCount, Fusion-method
fusionSplitReadsCount, 11
fusionSplitReadsCount, Fusion-method
fusionToDataFrame, 12
getEnsemblIds, 13
getFusionByChromosome, 14
getFusionByGeneName, 14
getGeneById, 15
getTranscriptsEnsembldb, 16
importDefuse, 17
importEricscript, 17
importFunctionNonUCSC, 18
importFusioncatcher, 19
importFusionmap, 19
importInfusion, 20
importJaffa, 21
importPrada, 21
importSoapfuse, 22
importStarfusion, 23
PartnerGene (PartnerGene-class), 23
PartnerGene-class, 23
partnerGeneEnsemblId, 24
partnerGeneEnsemblId, PartnerGene-method
partnerGeneEnsemblId, PartnerGene-method
partnerGeneEnsemblId<-
partnerGeneEnsemblId<-, PartnerGene-method
partnerGeneEnsemblId<-, PartnerGene-method
partnerGeneJunctionSequence, 25
partnerGeneJunctionSequence, PartnerGene-method
partnerGeneJunctionSequence, PartnerGene-method
plotCircle, 26
plotFusion, 27
plotFusionReads, 29
plotFusionSeparate (plotFusion), 27
plotFusionTogether (plotFusion), 27
plotFusionTranscript, 30
plotFusionTranscriptsGraph, 32
plotFusionTranscriptsWithProteinDomain, 33
plotTranscripts, 35
raw_cytobandhg19, 37
raw_cytobandhg38, 37
raw_defuse, 38
raw_ericsscript, 38
raw fusion5267proteindomains, 39
raw_fusion5267reads, 39
raw_fusion5267readsBedGraph, 39
raw_fusioncatcher, 40
raw_fusionmap, 40
raw_Homo_sapiens.GRCh37.74, 40
raw_infusion, 41
raw_jaffa, 41
raw_prada, 41
raw_soapfuse, 41
raw_starfusion, 42
selectTranscript, 42
show, Fusion-method, 43
show, PartnerGene-method, 43
splitOnUtrAndAddFeature, 44
upstreamPartnerGene, 45
upstreamPartnerGene, Fusion-method
(upstreamPartnerGene), 45
upstreamPartnerGene<-
(upstreamPartnerGene), 45
upstreamPartnerGene<-, Fusion-method
(upstreamPartnerGene), 45
writeFusionReference, 46