Package 'chimeraviz'

February 27, 2024

Type Package

Title Visualization tools for gene fusions

Version 1.28.0

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

License Artistic-2.0

LazyData TRUE

Imports methods, grid, Rsamtools, GenomeInfoDb, GenomicAlignments, RColorBrewer, graphics, AnnotationDbi, RCircos, org.Hs.eg.db, org.Mm.eg.db, rmarkdown, graph, Rgraphviz, DT, plyr, dplyr, BiocStyle, checkmate, gtools, magick

Depends Biostrings, GenomicRanges, IRanges, Gviz, S4Vectors, ensembldb, AnnotationFilter, data.table

Suggests testthat, roxygen2, devtools, knitr, lintr

SystemRequirements bowtie, samtools, and egrep are required for some functionalities

RoxygenNote 7.1.1

VignetteBuilder knitr

biocViews Infrastructure, Alignment

Encoding UTF-8

URL https://github.com/stianlagstad/chimeraviz

BugReports https://github.com/stianlagstad/chimeraviz/issues

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

git_branch RELEASE_3_18

git_last_commit 2a2ebf3

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-02-27

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:

`add_fusion_reads_alignment` *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

```
add_fusion_reads_alignment(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@fusion_reads_alignment set.

Examples

```
# Load data
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuse833ke, "hg19", 1)
# Find the specific fusion we have aligned reads for
fusion <- get_fusion_by_id(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 <- add_fusion_reads_alignment(fusion, bamfile5267)
```

---

chimeraviz *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-fusions_to_gene_label_data *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

```
.fusions_to_gene_label_data(fusion_list)
```

Arguments

fusion_list	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 <- import_defuse(defuse833ke, "hg19", 3) labelData <- chimeraviz::.fusions_to_gene_label_data(fusions) # This labelData can be used with RCircos.Gene.Connector.Plot() and RCircos.Gene.Name.Plot()

---

chimeraviz-internals-fusions_to_link_data

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

```
.fusions_to_link_data(fusion_list, min_link_width = 1, max_link_widt = 10)
```

Arguments

fusion_list	A list of Fusion objects.
min_link_width	The minimum link line width. Default = 1
max_link_widt	The maximum link line width. Default = 10

Value

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

@examples # Apparently examples shouldn't be set on private functions defuse833ke <- system.file("extdata", "defuse_833ke_results.filtered.tsv", package="chimeraviz") fusions <- import_defuse(defuse833ke, "hg19", 3) linkData <- chimeraviz::.fusions_to_link_data(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 [new_min, new_max] that the numeric values will be scaled (normalized) to.

Usage

```
.scale_list_to_interval(the_list, new_min, new_max)
```

Arguments

the_list	A vector of numeric values.
new_min	Minimum value for the new interval.
new_max	Maximum value for the new interval.

Value

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

@examples # Apparently examples shouldn't be set on private functions list012 <- c(0,1,2) .scale_list_to_interval(list012, 1, 3) # [1] 1 2 3

create_fusion_report *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

```
create_fusion_report(fusions, output_filename, quiet = TRUE)
```

Arguments

fusions	A list of Fusion objects.
output_filename	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

```
# Load data
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuse833ke, "hg19", 3)
# Temporary file to store the report
random_filename <- paste0(
  paste0(sample(LETTERS, 5, replace = TRUE), collapse=''),
  ".png"
)
# Create report
create_fusion_report(fusions, random_filename)
# Delete the file
file.remove(random_filename)
```

---

decide_transcript_category

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

```
decide_transcript_category(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

```
# Load fusion data and choose a fusion object:
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuseData, "hg19", 1)
fusion <- get_fusion_by_id(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(
      c(
        partner_gene_ensembl_id(upstream_partner_gene(fusion)),
        partner_gene_ensembl_id(downstream_partner_gene(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
decide_transcript_category(gr, fusion)
# "exonBoundary"
# Check another case
gr <- allTranscripts[[3]]
decide_transcript_category(gr, fusion)
# "withinIntron"
```

---

downstream_partner_gene

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

```
downstream_partner_gene(x)

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

downstream_partner_gene(object) <- value

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

Arguments

x	The Fusion object you wish to retrieve the downstream PartnerGene object for.
object	The Fusion object you wish to set a new downstream PartnerGene object for.
value	The new PartnerGene object.

Value

The downstream PartnerGene object.

Examples

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

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

down_shift *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

```
down_shift(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:
down_shift(gr)
```

fetch_reads_from_fastq *Fetch reads from fastq files*

Description

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

Usage

```
fetch_reads_from_fastq(
  reads,
  fastq_file_in1,
  fastq_file_in2,
  fastq_file_out1,
  fastq_file_out2
)
```

Arguments

reads	List of read IDs that is to be fetched.
fastq_file_in1	First fastq file to search in.
fastq_file_in2	Second fastq file to seach in.
fastq_file_out1	First fastq file with results.
fastq_file_out2	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

```
## 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())
fetch_reads_from_fastq(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.

`fusion_tool` Name of the fusion tool.

`genome_version` Name of the genome used to map reads.

`spanning_reads_count` The number of spanning reads supporting the fusion.

`split_reads_count` The number of split reads supporting the fusion.

`fusion_reads_alignment` A Gviz::AlignmentsTrack object holding the fusion reads aligned to the fusion sequence.

`gene_upstream` A PartnerGene object holding information of the upstream fusion partner gene.

`gene_downstream` 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.

`fusion_tool_specific_data` 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.

`fusion_spanning_reads_count` *Get the spanning reads count from a Fusion object*

Description

This getter retrieves the spanning reads count from a Fusion object

Usage

```
fusion_spanning_reads_count(x)

## S4 method for signature 'Fusion'
fusion_spanning_reads_count(x)
```

Arguments

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

Value

The Fusion spanning reads count.

Examples

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

`fusion_split_reads_count` *Get the split reads count from a Fusion object*

Description

This getter retrieves the split reads count from a Fusion object

Usage

```
fusion_split_reads_count(x)

## S4 method for signature 'Fusion'
fusion_split_reads_count(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")
fusions <- import_defuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the split reads count
fusion_split_reads_count(fusion)
```

fusion_to_data_frame *Coerce Fusion object to data.frame*

Description

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

Usage

```
fusion_to_data_frame(fusion)
```

Arguments

fusion	The Fusion object to coerce.

Value

A data.frame with the fusion object.

See Also

create_fusion_report

Examples

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

get_ensembl_ids *Get ensembl ids for a fusion object*

Description

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

Usage

```
get_ensembl_ids(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 <- import_defuse(defuse833keFiltered, "hg19", 1)
# Get a specific fusion
fusion <- get_fusion_by_id(fusions, 5267)
# See the ensembl ids:
partner_gene_ensembl_id(upstream_partner_gene(fusion))
# [1] "ENSG00000180198"
partner_gene_ensembl_id(downstream_partner_gene(fusion))
# [1] "ENSG00000162639"
# Reset the fusion objects ensembl ids
partner_gene_ensembl_id(upstream_partner_gene(fusion)) <- ""
partner_gene_ensembl_id(downstream_partner_gene(fusion))  <- ""
# Get the ensembl ids
fusion <- get_ensembl_ids(fusion)
# See that we now have the same ensembl ids again:
partner_gene_ensembl_id(upstream_partner_gene(fusion))
# [1] "ENSG00000180198"
partner_gene_ensembl_id(downstream_partner_gene(fusion))
# [1] "ENSG00000162639"
```

---

get_fusion_by_chromosome

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

```
get_fusion_by_chromosome(fusion_list, chr)
```

Arguments

fusion_list	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 <- import_defuse(defuse833ke, "hg19", 1)
length(get_fusion_by_chromosome(fusions, "chr1"))
# [1] 1
```

---

get_fusion_by_gene_name

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

```
get_fusion_by_gene_name(fusion_list, gene_name)
```

Arguments

fusion_list	A list of Fusion objects.
gene_name	The gene name we're looking for.

Details

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

Value

A list of Fusion objects.

Examples

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

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

Description

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

Usage

```
get_fusion_by_id(fusion_list, id)
```

Arguments

fusion_list	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

```
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(fusions, 5267)
# This should be the Fusion object:
fusion
# [1] "Fusion object"
# [1] "id: 5267"
# [1] "Fusion tool: defuse"
# [1] "Genome version: hg19"
# [1] "Gene names: RCC1-HENMT1"
# [1] "Chromosomes: chr1-chr1"
# [1] "Strands: +,-"
```

get_transcripts_ensembl_db *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@gene_upstream@transcriptsX slots are set with transcript information.

Usage

```
get_transcripts_ensembl_db(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 <- import_defuse(defuseData, "hg19", 1)
fusion <- get_fusion_by_id(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 <- get_transcripts_ensembl_db(fusion, edb)
# The transcripts are now accessible through fusion@gene_upstream@transcripts and
# fusion@gene_downstream@transcripts .
```

---

import_aeron *Import results from an Aeron run into a list of Fusion objects.*

Description

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

Usage

```
import_aeron(
  filename_fusion_support,
  filename_fusion_transcript,
  genome_version,
  limit
)
```

Arguments

filename_fusion_support	Filename for the Aeron result file fusion_support..txt.
filename_fusion_transcript	Filename for the Aeron result file fusion_transcripts..txt.
genome_version	Which genome was used in mapping (hg19, hg38, etc.).
limit	A limit on how many lines to read.

Details

Note that the strands and breakpoint positions are not included in the result files from Aeron. These have to be retrieved manually, using the ensembl identifiers (which are included in the result files, and will be available in the Fusion objects after importing).

Value

A list of Fusion objects.

Examples

```
aeronfusionsupportfile <- system.file(
  "extdata",
  "aeron_fusion_support.txt",
  package="chimeraviz")
aeronfusiontranscriptfile <- system.file(
  "extdata",
  "aeron_fusion_transcripts.fa",
  package="chimeraviz")
fusions <- import_aeron(
  aeronfusionsupportfile,
  aeronfusiontranscriptfile,
  "hg19",
  3)
# This should import a list of 3 fusions described in Fusion objects.
```

import_chimpipe *Import results from a ChimPipe run into a list of Fusion objects.*

Description

A function that imports the results from a ChimPipe run, typically from a chimericJunctions_.txt file, into a list of Fusion objects.

Usage

```
import_chimpipe(filename, genome_version, limit)
```

Arguments

filename	Filename for the ChimPipe results.
genome_version	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

```
chimpipefile <- system.file(
  "extdata",
  "chimericJunctions_MCF-7.txt",
  package="chimeraviz")
fusions <- import_chimpipe(chimpipefile, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

import_defuse *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

```
import_defuse(filename, genome_version, limit)
```

Arguments

filename	Filename for the deFuse results .tsv file.
genome_version	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 <- import_defuse(defuse833ke, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

---

import_ericscript *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

```
import_ericscript(filename, genome_version, limit)
```

Arguments

filename	Filename for the EricScript results file.
genome_version	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

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

import_function_non_ucsc *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

```
import_function_non_ucsc(file, selection)
```

Arguments

file	The bamfile.
selection	Which regions to get from the bamfile.

Value

A GRanges object with coverage data for the selection.

import_fusioncatcher *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

```
import_fusioncatcher(filename, genome_version, limit)
```

Arguments

filename	Filename for the Fusioncatcher final-list-candidate-fusion-genes.txt results file.
genome_version	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 <- import_fusioncatcher(fusioncatcher833ke, "hg38", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

---

import_fusionmap *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

```
import_fusionmap(filename, genome_version, limit)
```

Arguments

filename	Filename for the FusionMap PairedEndFusionReport.txt results file.
genome_version	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

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

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

Description

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

Usage

```
import_infusion(filename, genome_version, limit)
```

Arguments

filename	Filename for the jaffa_results.csv file.
genome_version	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 <- import_infusion(infusionData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

import_jaffa *Import results from a JAFFA run into a list of Fusion objects.*

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

```
import_jaffa(filename, genome_version, limit)
```

Arguments

filename	Filename for the jaffa_results.csv file.
genome_version	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 <- import_jaffa(jaffaData, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

---

import_oncofuse *Import results from a oncofuse run into a list of Fusion objects.*

Description

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

Usage

```
import_oncofuse(filename, genome_version, limit)
```

Arguments

filename	Filename for the oncofuse results .tsv file.
genome_version	Which genome was used in mapping (hg19, hg38, etc.).
limit	A limit on how many lines to read.

Details

This import function was contributed by Lavinia G, ref https://github.com/stianlagstad/chimeraviz/issues/47#issuecomment-409773158

Value

A list of Fusion objects.

Examples

```
oncofuse833ke <- system.file(
  "extdata",
  "oncofuse.outfile",
  package="chimeraviz")
fusions <- import_oncofuse(oncofuse833ke, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

`import_prada` *Import results from a PRADA run into a list of Fusion objects.*

Description

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

Usage

```
import_prada(filename, genome_version, limit)
```

Arguments

`filename`	Filename for the PRADA results file.
`genome_version`	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

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

`import_soapfuse` *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

```
import_soapfuse(filename, genome_version, limit)
```

Arguments

`filename`	Filename for the SOAPfuse final-list-candidate-fusion-genes.txt results file.
`genome_version`	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 <- import_soapfuse(soapfuse833ke, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

`import_squid` *Import results from a SQUID run into a list of Fusion objects.*

Description

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

Usage

```
import_squid(filename, genome_version, limit)
```

Arguments

filename	Filename for the SQUID results.
genome_version	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

```
squidfile <- system.file(
  "extdata",
  "squid_hcc1954_sv.txt",
  package="chimeraviz")
fusions <- import_squid(squidfile, "hg19", 3)
# This should import a list of 3 fusions described in Fusion objects.
```

---

import_starfusion *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

```
import_starfusion(filename, genome_version, limit)
```

Arguments

filename	Filename for the STAR-Fusion star-fusion.fusion_candidates.final.abridged results file.
genome_version	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 <- import_starfusion(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.

`ensembl_id` Character containing ensembl id for the gene.

`chromosome` Character containing chromosome name.

`breakpoint` Numeric containing the fusion breakpoint.

`strand` Character containing gene strand.

`junction_sequence` 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.

---

partner_gene_ensembl_id *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

```
partner_gene_ensembl_id(x)

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

partner_gene_ensembl_id(object) <- value

## S4 replacement method for signature 'PartnerGene'
partner_gene_ensembl_id(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

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

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

partner_gene_junction_sequence *Get the junction sequence from a PartnerGene object*

Description

This getter retrieves the junction sequence from a PartnerGene object

Usage

```
partner_gene_junction_sequence(x)

## S4 method for signature 'PartnerGene'
partner_gene_junction_sequence(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 <- import_defuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the junction sequence from the upstream fusion partner gene
partner_gene_junction_sequence(upstream_partner_gene(fusion))
```

plot_circle *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

```
plot_circle(fusion_list)
```

Arguments

fusion_list	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.

Value

Creates a circle plot.

Examples

```
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(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!
plot_circle(fusions)
# Close device
dev.off()
```

---

plot_fusion *Plot a fusion event with transcripts, coverage and ideograms.*

Description

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

Usage

```
plot_fusion(
  fusion,
  edb = NULL,
  bamfile = NULL,
  which_transcripts = "exonBoundary",
  ylim = c(0, 1000),
```

```
  non_ucsc = TRUE,
  reduce_transcripts = FALSE,
  bedgraphfile = NULL
)

plot_fusion_separate(
  fusion,
  edb,
  bamfile = NULL,
  which_transcripts = "exonBoundary",
  ylim = c(0, 1000),
  non_ucsc = TRUE,
  reduce_transcripts = FALSE,
  bedgraphfile = NULL
)

plot_fusion_together(
  fusion,
  edb,
  bamfile = NULL,
  which_transcripts = "exonBoundary",
  ylim = c(0, 1000),
  non_ucsc = TRUE,
  reduce_transcripts = 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.
which_transcripts	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.
non_ucsc	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".
reduce_transcripts	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

plot_fusion() will dispatch to either plot_fusion_separate() or plot_fusion_together(). plot_fusion_separate() will plot the fusion gene partners in separate graphs shown next to each other, while plot_fusion_together() will plot the fusion gene partners in the same graph with the same x-axis. plot_fusion() will dispatch to plot_fusion_together() if the fusion gene partners are on the same strand, same chromosome and are close together (<=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 <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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!
plot_fusion(
  fusion = fusion,
  bamfile = bamfile5267,
  edb = edb,
  non_ucsc = 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 <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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!
plot_fusion(
  fusion = fusion,
  bedgraphfile = bedgraphfile,
  edb = edb,
  non_ucsc = TRUE)
# Close device
dev.off()
```

`plot_fusion_reads` *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@junction_sequence), provided that add_fusion_reads_alignment() has been run earlier in order to add fusion reads alignment data to the fusion object.

Usage

```
plot_fusion_reads(fusion, show_all_nucleotides = TRUE, nucleotide_amount = 10)
```

Arguments

`fusion`	The Fusion object to plot.

show_all_nucleotides
: Boolean indicating whether or not to show all nucleotides. If FALSE, then only nucleotide_amount amount of nucleotides will be shown on each end of the fusion junction. If TRUE, then the whole fusion junction sequence will be shown.

nucleotide_amount
: The number of nucleotides to show on each end of the fusion junction sequence. Defaults to 10. Only applicable if show_all_nucleotides 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

add_fusion_reads_alignment

Examples

```
# Load data
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuseData, "hg19", 1)
# Find the specific fusion we have aligned reads for
fusion <- get_fusion_by_id(fusions, 5267)
bamfile <- system.file(
  "extdata",
  "5267readsAligned.bam",
  package="chimeraviz")
# Add the bam file of aligned fusion reads to the fusion object
fusion <- add_fusion_reads_alignment(fusion, bamfile)
# Temporary file to store the plot
pngFilename <- tempfile(
  pattern = "fusionPlot",
  fileext = ".png",
  tmpdir = tempdir())
# Calculate image size based on supporting reads and lenght of junction
# sequence.
imageWidth <- (nchar(partner_gene_junction_sequence(upstream_partner_gene(fusion))) +
  nchar(partner_gene_junction_sequence(downstream_partner_gene(fusion)))) * 15
imageHeight <- (fusion_split_reads_count(fusion)+fusion_spanning_reads_count(fusion)) * 20
# Open device
png(pngFilename, width = imageWidth, height = imageHeight)
# Now we can plot
plot_fusion_reads(fusion)
# Close device
```

```
dev.off()
```

`plot_fusion_transcript` *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. If a bamfile is specified, the fusion transcript will be plotted with coverage information.

Usage

```
plot_fusion_transcript(
  fusion,
  edb = NULL,
  bamfile = NULL,
  which_transcripts = "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.
`which_transcripts`	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 <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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!
plot_fusion_transcript(
  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")
fusions <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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 = 500, height = 500)
# Plot!
plot_fusion_transcript(
  fusion = fusion,
  bamfile = bamfile5267,
  edb = edb)
# Close device
dev.off()
```

`plot_fusion_transcripts_graph` *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

```
plot_fusion_transcripts_graph(
  fusion,
  edb = NULL,
  which_transcripts = "exonBoundary",
  rankdir = "TB"
)
```

Arguments

`fusion`	The Fusion object to plot.
`edb`	The edb object that will be used to fetch data.
`which_transcripts`	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().

Value

Creates a fusion transcripts graph plot.

Examples

```
# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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!
plot_fusion_transcripts_graph(
  fusion = fusion,
  edb = edb)
# Close device
dev.off()
```

---

plot_fusion_transcript_with_protein_domain

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

```
plot_fusion_transcript_with_protein_domain(
  fusion,
  edb = NULL,
  bamfile = NULL,
  bedfile = NULL,
  gene_upstream_transcript = "",
  gene_downstream_transcript = "",
```

```
  plot_downstream_protein_domains_if_fusion_is_out_of_frame = FALSE
)
```

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.
gene_upstream_transcript	The transcript id for the upstream gene.
gene_downstream_transcript	The transcript id for the downstream gene.
plot_downstream_protein_domains_if_fusion_is_out_of_frame	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

```
# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(fusions, 5267)
# Select transcripts
gene_upstream_transcript <- "ENST00000434290"
gene_downstream_transcript <- "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
```

```
bedfile <- system.file(
  "extdata",
  "protein_domains_5267.bed",
  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!
plot_fusion_transcript_with_protein_domain(
  fusion = fusion,
  edb = edb,
  bamfile = bamfile5267,
  bedfile = bedfile,
  gene_upstream_transcript = gene_upstream_transcript,
  gene_downstream_transcript = gene_downstream_transcript,
  plot_downstream_protein_domains_if_fusion_is_out_of_frame = TRUE)
# Close device
dev.off()
```

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

Description

This function takes a fusion object and an ensembldb 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

```
plot_transcripts(
  fusion,
  edb = NULL,
  bamfile = NULL,
  which_transcripts = "exonBoundary",
  non_ucsc = TRUE,
  ylim = c(0, 1000),
  reduce_transcripts = 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.
which_transcripts	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".
non_ucsc	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.
reduce_transcripts	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 <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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!
```

```
plot_transcripts(
  fusion = fusion,
  edb = edb,
  bamfile = bamfile5267,
  non_ucsc = 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 <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(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 = 500, height = 500)
# Plot!
plot_transcripts(
  fusion = fusion,
  edb = edb,
  bedgraphfile = bedgraphfile,
  non_ucsc = TRUE)
# Close device
dev.off()
```

---

raw_chimpipe *ChimPipe data*

Description

Documentation for the ChimPipe example data.

Documentation for the SQUID example data.

chimericJunctions_MCF-7.txt

This is example data from the ChimPipe tutorial all-in-one package located at https://chimpipe.readthedocs.io/en/latest/tutoria
all-in-one-package It was downloaded March 10th 2020

squid_hcc1954_sv.txt

This is example data for SQUID provided on GitHub here: https://github.com/Kingsford-Group/squid/issues/20#issuecomme
598888472 It was downloaded March 17th 2020

---

raw_cytobandhg19 *Cytoband information HG19*

Description

Cytoband information for the HG19 assembly from UCSC. Downloaded from http://hgdownload.cse.ucsc.edu/goldenpath/hg

UCSC.HG19.Human.CytoBandIdeogram.txt

This data is used with RCircos in plot_circle().

---

raw_cytobandhg38 *Cytoband information HG138*

Description

Cytoband information for the HG38 assembly from UCSC. Downloaded from http://hgdownload.cse.ucsc.edu/goldenpath/hg

Details

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

UCSC.HG38.Human.CytoBandIdeogram.txt

This data is used with RCircos in plot_circle().

raw_defuse *deFuse data*

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 plot_fusion_reads().

raw_ericscript *EricScript data*

Description

Documentation for the EricScript example data.

ericscript_SRR1657556.results.total.tsv

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

raw_fusion5267proteindomains *protein_domains_5267 bed file*

Description

Documentation for the protein_domains_5267.bed file containing protein domains for the genes in the fusion with cluster_id=5267.

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.

raw_fusion5267reads *Fusion5267and11759 bamfile*

Description

Documentation for the fusion5267and11759reads.bam file containing reads mapped to the region where the genes in the fusions with cluster_id=5267 and cluster_id=11759 is.

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 fusion5267and11759reads.bam.b

where we extract the reads mapping to the region where we know the fusions with cluster_id=5267 and cluster_id=11759 from the deFuse example data 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/c

raw_fusion5267readsBedGraph *Fusion5267and11759 bedGraph file*

Description

Documentation for the fusion5267and11759reads.bedGraph file containing read count data from the regions of the fusion event with cluster_id=5267.

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-fusion-genes.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 "ENST00000373831\|ENST00000373832\|ENST00000373833\|ENST00000398958\|ENST00000411533\|ENST000
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 *InFusion data*

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 *JAFFA data*

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_oncofuse *oncofuse data*

Description

Documentation for the oncofuse example data.

oncofuse.outfile

The example output from oncofuse was kindly provided by Lavinia G here: https://github.com/stianlagstad/chimeraviz/issues/
409773158

raw_prada *PRADA data*

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 *SOAPfuse data*

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.

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 2.nd 2017.

select_transcript *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 which_transcripts

Usage

```
select_transcript(gene_partner, which_transcripts = "exonBoundary")
```

Arguments

gene_partner	The GenePartner object to select a transcript for.
which_transcripts	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

select_transcript() 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.

Examples

```
# Load data and example fusion event
defuse833ke <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuse833ke, "hg19", 1)
fusion <- get_fusion_by_id(fusions, 5267)
# Load edb
edbSqliteFile <- system.file(
  "extdata",
  "Homo_sapiens.GRCh37.74.sqlite",
  package="chimeraviz")
edb <- ensembldb::EnsDb(edbSqliteFile)
# Get transcripts
fusion <- get_transcripts_ensembl_db(fusion, edb)
# Select transcript
transcriptsA <- select_transcript(upstream_partner_gene(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)
```

Arguments

object	A PartnerGene object

Value

Shows information about a PartnerGene object.

split_on_utr_and_add_feature *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

```
split_on_utr_and_add_feature(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

```
# Load fusion data and choose a fusion object:
defuseData <- system.file(
  "extdata",
  "defuse_833ke_results.filtered.tsv",
  package="chimeraviz")
fusions <- import_defuse(defuseData, "hg19", 1)
fusion <- get_fusion_by_id(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(
      c(
        partner_gene_ensembl_id(upstream_partner_gene(fusion)),
        partner_gene_ensembl_id(downstream_partner_gene(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 <- split_on_utr_and_add_feature(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
```

`upstream_partner_gene` *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

```
upstream_partner_gene(x)

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

upstream_partner_gene(object) <- value

## S4 replacement method for signature 'Fusion'
upstream_partner_gene(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 <- import_defuse(defuseData, "hg19", 1)
fusion <- fusions[[1]]
# Get the upstream fusion partner gene
upstream_partner_gene(fusion)

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

write_fusion_reference *Write fusion junction sequence to a fasta file*

Description

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

Usage

```
write_fusion_reference(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 <- import_defuse(defuse833keFiltered, "hg19", 1)
# Get a specific fusion
fusion <- get_fusion_by_id(fusions, 5267)
# Create temporary file to hold the fusion sequence
fastaFileOut <- tempfile(pattern = "fusionSequence", tmpdir = tempdir())
# Write fusion sequence to file
write_fusion_reference(fusion, fastaFileOut)
```

Index