

Package ‘wiggplotr’

October 16, 2018

Title Make read coverage plots from BigWig files

Version 1.4.0

Author Kaur Alasoo [aut, cre]

Maintainer Kaur Alasoo <kaur.alasoo@gmail.com>

Description Tools to visualise read coverage from sequencing experiments together with genomic annotations (genes, transcripts, peaks). Introns of long transcripts can be rescaled to a fixed length for better visualisation of exonic read coverage.

Depends R (>= 3.4)

Imports dplyr, ggplot2 (>= 2.2.0), GenomicRanges, rtracklayer, cowplot, assertthat, purrr, S4Vectors, IRanges, GenomeInfoDb

License Apache License 2.0

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, biomaRt, GenomicFeatures, testthat, ensemblDb, EnsDb.Hsapiens.v86, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg38.knownGene, AnnotationDbi, AnnotationFilter

VignetteBuilder knitr

biocViews Coverage, RNASeq, ChIPSeq, Sequencing, Visualization, GeneExpression, Transcription, AlternativeSplicing

git_url <https://git.bioconductor.org/packages/wiggplotr>

git_branch RELEASE_3_7

git_last_commit 2756080

git_last_commit_date 2018-04-30

Date/Publication 2018-10-15

R topics documented:

getGenotypePalette	2
makeManhattanPlot	2
ncoa7_cdss	3
ncoa7_exons	3

ncoa7_metadata	4
pasteFactors	4
plotCoverage	5
plotCoverageFromEnsemblDb	7
plotCoverageFromUCSC	7
plotTranscripts	8
plotTranscriptsFromEnsemblDb	9
plotTranscriptsFromUCSC	10
wiggleplotr	11

Index 12

getGenotypePalette	<i>Returns a three-colour palette suitable for visualising read coverage stratified by genotype</i>
--------------------	---

Description

Returns a three-colour palette suitable for visualising read coverage stratified by genotype

Usage

```
getGenotypePalette()
```

Value

Vector of three colours.

Examples

```
getGenotypePalette()
```

makeManhattanPlot	<i>Make a Manhattan plot of p-values</i>
-------------------	--

Description

The Manhattan plots is compatible with wiggleplotr read coverage and transcript structure plots. Can be appended to those using the cowplot::plot_grid() function.

Usage

```
makeManhattanPlot(pvalues_df, region_coords, color_R2 = FALSE,
  data_track = TRUE)
```

Arguments

pvalues_df	Data frame of association p-values (required columns: track_id, p_nominal, pos)
region_coords	Start and end coordinates of the region to plot.
color_R2	Color the points according to R2 from the lead variant. Require R2 column in the pvalues_df data frame.
data_track	If TRUE, then remove all information from x-axis. Makes it easy to append to read coverage or transcript structure plots using cowplot::plot_grid().

Value

ggplot2 object

Examples

```
data = dplyr::data_frame(track_id = "GWAS", pos = sample(c(1:1000), 200), p_nominal = runif(200, min = 0.00001, max = 1))
makeManhattanPlot(data, c(1,1000), data_track = FALSE)
```

ncoa7_cdss

Coding sequences from 9 protein coding transcripts of NCOA7

Description

A dataset containing start and end coordinates of coding sequences (CDS) from nine protein coding transcripts of NCOA7.

Usage

```
ncoa7_cdss
```

Format

A GRangesList object with 9 elements:

element CDS start and end coordinates for a single transcript (GRanges object) ...

Source

<http://www.ensembl.org/>

ncoa7_exons

Exons from 9 protein coding transcripts of NCOA7

Description

A dataset containing start and end coordinates of exons from nine protein coding transcripts of NCOA7.

Usage

```
ncoa7_exons
```

Format

A GRangesList object with 9 elements:

element Exon start and end coordinates for a single transcript (GRanges object) ...

Source

<http://www.ensembl.org/>

ncoa7_metadata	<i>Gene metadata for NCOA7</i>
----------------	--------------------------------

Description

A a list of transcripts for NCOA7.

Usage

```
ncoa7_metadata
```

Format

A data.frame object with 4 columns:

transcript_id Ensembl transcript id.

gene_id Ensembl gene id.

gene_name Human readable gene name.

strand Strand of the transcript (either +1 or -1). ...

Source

<http://www.ensembl.org/>

pasteFactors	<i>Paste two factors together and preserved their joint order.</i>
--------------	--

Description

Paste two factors together and preserved their joint order.

Usage

```
pasteFactors(factor1, factor2)
```

Arguments

factor1 First factor

factor2 Second factor

Value

Factors factor1 and factor2 pasted together.

plotCoverage	<i>Plot read coverage across genomic regions</i>
--------------	--

Description

Also supports rescaling introns to constant length. Does not work on Windows, because rtracklayer cannot read BigWig files on Windows.

Usage

```
plotCoverage(exons, cdss = NULL, transcript_annotiations = NULL, track_data,
  rescale_introns = TRUE, new_intron_length = 50, flanking_length = c(50,
  50), plot_fraction = 0.1, heights = c(0.75, 0.25), alpha = 1,
  fill_palette = c("#a1dab4", "#41b6c4", "#225ea8"), mean_only = TRUE,
  connect_exons = TRUE, transcript_label = TRUE,
  return_subplots_list = FALSE, region_coords = NULL,
  coverage_type = "area")
```

Arguments

exons	list of GRanges objects, each object containing exons for one transcript. The list must have names that correspond to transcript_id column in transcript_annotiations data.frame.
cdss	list of GRanges objects, each object containing the coding regions (CDS) of a single transcript. The list must have names that correspond to transcript_id column in transcript_annotiations data.frame. If cdss is not specified then exons list will be used for both arguments. (default: NULL).
transcript_annotiations	Data frame with at least three columns: transcript_id, gene_name, strand. Used to construct transcript labels. (default: NULL)
track_data	data.frame with the metadata for the bigWig read coverage files. Must contain the following columns: <ul style="list-style-type: none"> • sample_id - unique id for each sample. • track_id - if multiple samples (bigWig files) have the same track_id they will be overlaid on the same plot, track_id is also used as the facet label on the right. • bigWig - path to the bigWig file. • scaling_factor - normalisation factor for each sample, useful if different samples sequenced to different depth and bigWig files not normalised for that. • colour_group - additional column to group samples into, is used as the colour of the coverage track.
rescale_introns	Specifies if the introns should be scaled to fixed length or not. (default: TRUE)
new_intron_length	length (bp) of introns after scaling. (default: 50)
flanking_length	Lengths of the flanking regions upstream and downstream of the gene. (default: c(50,50))

plot_fraction	Size of the random sub-sample of points used to plot coverage (between 0 and 1). Smaller values make plotting significantly faster. (default: 0.1)
heights	Specifies the proportion of the height that is dedicated to coverage plots (first value) relative to transcript annotations (second value). (default: c(0.75,0.25))
alpha	Transparency (alpha) value for the read coverage tracks. Useful to set to something < 1 when overlaying multiple tracks (see track_id). (default: 1)
fill_palette	Vector of fill colours used for the coverage tracks. Length must be equal to the number of unique values in track_data\$colour_group column.
mean_only	Plot only mean coverage within each combination of track_id and colour_group values. Useful for example for plotting mean coverage stratified by genotype (which is specified in the colour_group column) (default: TRUE).
connect_exons	Print lines that connect exons together. Set to FALSE when plotting peaks (default: TRUE).
transcript_label	If TRUE then transcript labels are printed above each transcript. (default: TRUE).
return_subplots_list	Instead of a joint plot return a list of subplots that can be joined together manually.
region_coords	Start and end coordinates of the region to plot, overrides flanking_length parameter.
coverage_type	Specifies if the read coverage is represented by either 'line', 'area' or 'both'. The 'both' option tends to give better results for wide regions. (default: area).

Value

Either object from cow_plot::plot_grid() function or a list of subplots (if return_subplots_list == TRUE)

Examples

```
require("dplyr")
require("GenomicRanges")
sample_data = dplyr::data_frame(sample_id = c("aibt_A", "aibt_C", "bima_A", "bima_C"),
  condition = factor(c("Naive", "LPS", "Naive", "LPS"), levels = c("Naive", "LPS")),
  scaling_factor = 1) %>%
  dplyr::mutate(bigWig = system.file("extdata", paste0(sample_id, ".str2.bw"), package = "wiggleplotr"))

track_data = dplyr::mutate(sample_data, track_id = condition, colour_group = condition)

selected_transcripts = c("ENST00000438495", "ENST00000392477") #Plot only two transcripts of the gens
## Not run:
plotCoverage(ncoa7_exons[selected_transcripts], ncoa7_cdss[selected_transcripts],
  ncoa7_metadata, track_data,
  heights = c(2,1), fill_palette = getGenotypePalette())

## End(Not run)
```

`plotCoverageFromEnsemblDb`*Plot read coverage directly from ensemblDb object.*

Description

A wrapper around the `plotCoverage` function. See the documentation for ([plotCoverage](#)) for more information.

Usage

```
plotCoverageFromEnsemblDb(ensemldb, gene_names, transcript_ids = NULL, ...)
```

Arguments

<code>ensemldb</code>	ensemldb object.
<code>gene_names</code>	List of gene names to be plotted.
<code>transcript_ids</code>	Optional list of transcript ids to be plotted.
<code>...</code>	Additional parameters to be passed to <code>plotCoverage</code> .

Value

ggplot2 object

Examples

```
require("EnsDb.Hsapiens.v86")
require("dplyr")
require("GenomicRanges")
sample_data = dplyr::data_frame(sample_id = c("aigt_A", "aigt_C", "bima_A", "bima_C"),
  condition = factor(c("Naive", "LPS", "Naive", "LPS"), levels = c("Naive", "LPS")),
  scaling_factor = 1) %>%
  dplyr::mutate(bigWig = system.file("extdata", paste0(sample_id, ".str2.bw"), package = "wigglyplotr"))

track_data = dplyr::mutate(sample_data, track_id = condition, colour_group = condition)
## Not run:
plotCoverageFromEnsemblDb(EnsDb.Hsapiens.v86, "NCOA7", transcript_ids = c("ENST00000438495", "ENST000003924"),
  track_data, heights = c(2,1), fill_palette = getGenotypePalette())

## End(Not run)
```

`plotCoverageFromUCSC` *Plot read coverage directly from UCSC OrgDb and TxDb objects.*

Description

A wrapper around the `plotCoverage` function. See the documentation for ([plotCoverage](#)) for more information.

Usage

```
plotCoverageFromUCSC(orgdb, txdb, gene_names, transcript_ids = NULL, ...)
```

Arguments

```
orgdb          UCSC OrgDb object.
txdb           UCSC TxDb object.
gene_names     List of gene names to be plotted.
transcript_ids Optional list of transcript ids to be plotted.
...           Additional parameters to be passed to plotCoverage.
```

Value

```
ggplot2 object
```

Examples

```
require("dplyr")
require("GenomicRanges")
require("org.Hs.eg.db")
require("TxDb.Hsapiens.UCSC.hg38.knownGene")

orgdb = org.Hs.eg.db
txdb = TxDb.Hsapiens.UCSC.hg38.knownGene

sample_data = dplyr::data_frame(sample_id = c("aigt_A", "aigt_C", "bima_A", "bima_C"),
  condition = factor(c("Naive", "LPS", "Naive", "LPS"), levels = c("Naive", "LPS")),
  scaling_factor = 1) %>%
  dplyr::mutate(bigWig = system.file("extdata", paste0(sample_id, ".str2.bw"), package = "wiggleplotr"))

track_data = dplyr::mutate(sample_data, track_id = condition, colour_group = condition)
## Not run:
##Note: This example does not work, because UCSC and Ensembl use different chromosome names
plotCoverageFromUCSC(orgdb, txdb, "NCOA7", transcript_ids = c("uc003qae.5", "uc063rdt.2"),
  track_data, heights = c(2,1), fill_palette = getGenotypePalette())

## End(Not run)
```

plotTranscripts

Quickly plot transcript structure without read coverage tracks

Description

Quickly plot transcript structure without read coverage tracks

Usage

```
plotTranscripts(exons, cdss = NULL, transcript_annotations = NULL,
  rescale_introns = TRUE, new_intron_length = 50, flanking_length = c(50,
  50), connect_exons = TRUE, transcript_label = TRUE,
  region_coords = NULL)
```


Arguments

exons	list of GRanges objects, each object containing exons for one transcript. The list must have names that correspond to transcript_id column in transcript_annotations data.frame.
cdss	list of GRanges objects, each object containing the coding regions (CDS) of a single transcript. The list must have names that correspond to transcript_id column in transcript_annotations data.frame. If cdss is not specified then exons list will be used for both arguments. (default: NULL)
transcript_annotations	Data frame with at least three columns: transcript_id, gene_name, strand. Used to construct transcript labels. (default: NULL)
rescale_introns	Specifies if the introns should be scaled to fixed length or not. (default: TRUE)
new_intron_length	length (bp) of introns after scaling. (default: 50)
flanking_length	Lengths of the flanking regions upstream and downstream of the gene. (default: c(50,50))
connect_exons	Print lines that connect exons together. Set to FALSE when plotting peaks (default: TRUE).
transcript_label	If TRUE then transcript labels are printed above each transcript. (default: TRUE).
region_coords	Start and end coordinates of the region to plot, overrides flanking_length parameter.

Value

ggplot2 object

Examples

```
plotTranscripts(ncoa7_exons, ncoa7_cdss, ncoa7_metadata, rescale_introns = FALSE)
```

plotTranscriptsFromEnsemblDb

Plot transcripts directly from ensemblDb object.

Description

A wrapper around the plotTranscripts function. See the documentation for ([plotTranscripts](#)) for more information.

Usage

```
plotTranscriptsFromEnsemblDb(ensemldb, gene_names, transcript_ids = NULL,
  ...)
```

Arguments

ensemldb ensemblDb object.
 gene_names List of gene names to be plotted.
 transcript_ids Optional list of transcript ids to be plotted.
 ... Additional parameters to be passed to plotTranscripts

Value

ggplot2 object

Examples

```
require("EnsDb.Hsapiens.v86")
plotTranscriptsFromEnsemblDb(EnsDb.Hsapiens.v86, "NCOA7", transcript_ids = c("ENST00000438495", "ENST000003
```

```
plotTranscriptsFromUCSC
```

Plot transcripts directly from UCSC OrgDb and TxDb objects.

Description

A wrapper around the plotTranscripts function. See the documentation for ([plotTranscripts](#)) for more information. Note that this function is much slower than ([plotTranscripts](#)) or ([plotTranscriptsFromEnsemblDb](#)) functions, because individually extracting exon coordinates from txdb objects is quite inefficient.

Usage

```
plotTranscriptsFromUCSC(orgdb, txdb, gene_names, transcript_ids = NULL, ...)
```

Arguments

orgdb UCSC OrgDb object.
 txdb UCSC TxDb object.
 gene_names List of gene names to be plot.
 transcript_ids Optional list of transcript ids to be plot. (default = NULL)
 ... Additional parameters to be passed to plotTranscripts

Value

Transcript plot.

Examples

```
#Load OrgDb and TxDb objects with UCSC gene annotations
require("org.Hs.eg.db")
require("TxDb.Hsapiens.UCSC.hg38.knownGene")
orgdb = org.Hs.eg.db
txdb = TxDb.Hsapiens.UCSC.hg38.knownGene

plotTranscriptsFromUCSC(orgdb, txdb, "NCOA7", transcript_ids = c("uc003qae.5", "uc063rdt.2"))
```

*wiggleplotr**wiggleplotr*

Description

wiggleplotr package provides tools to visualise transcript annotations ([plotTranscripts](#)) and plot sequencing read coverage over annotated transcripts ([plotCoverage](#)).

Details

You can also use convenient wrapper functions ([plotTranscriptsFromEnsemblDb](#)), ([plotCoverageFromEnsemblDb](#)), ([plotTranscriptsFromUCSC](#)) and ([plotCoverageFromUCSC](#)).

To learn more about wiggleplotr, start with the vignette: `browseVignettes(package = "wiggleplotr")`

Index

*Topic **datasets**

ncoa7_cdss, [3](#)

ncoa7_exons, [3](#)

ncoa7_metadata, [4](#)

getGenotypePalette, [2](#)

makeManhattanPlot, [2](#)

ncoa7_cdss, [3](#)

ncoa7_exons, [3](#)

ncoa7_metadata, [4](#)

pasteFactors, [4](#)

plotCoverage, [5](#), [7](#), [11](#)

plotCoverageFromEnsemblDb, [7](#), [11](#)

plotCoverageFromUCSC, [7](#), [11](#)

plotTranscripts, [8](#), [9–11](#)

plotTranscriptsFromEnsemblDb, [9](#), [10](#), [11](#)

plotTranscriptsFromUCSC, [10](#), [11](#)

wiggleplotr, [11](#)

wiggleplotr-package (wiggleplotr), [11](#)