Package ‘factR’

May 17, 2024

Title  Functional Annotation of Custom Transcriptomes

Version  1.6.0

Description  factR contain tools to process and interact with custom-assembled transcriptomes (GTF). At its core, factR constructs CDS information on custom transcripts and subsequently predicts its functional output. In addition, factR has tools capable of plotting transcripts, correcting chromosome and gene information and shortlisting new transcripts.

Depends  R (>= 4.2)

biocViews  AlternativeSplicing, FunctionalPrediction, GenePrediction

Imports  BiocGenerics (>= 0.46), Biostrings (>= 2.68), GenomeInfoDb (>= 1.36), dplyr (>= 1.1), GenomicFeatures (>= 1.52), GenomicRanges (>= 1.52), IRanges (>= 2.34), purrr (>= 1.0), rtracklayer (>= 1.60), tidyr (>= 1.3), methods (>= 4.3), BiocParallel (>= 1.34), S4Vectors (>= 0.38), data.table (>= 1.14), rlang (>= 1.1), tibble (>= 3.2), wiggleplotr (>= 1.24), RCurl (>= 1.98), XML (>= 3.99), drawProteins (>= 1.20), ggplot2 (>= 3.4), stringr (>= 1.5), pbapply (>= 1.7), stats (>= 4.3), utils (>= 4.3), graphics (>= 4.3), crayon (>= 1.5)

License  file LICENSE

Encoding  UTF-8

ByteCompile  true

RoxygenNote  7.2.1

Suggests  AnnotationHub (>= 2.22), BSgenome (>= 1.58), BSgenome.Mmusculus.UCSC.mm10, testthat, knitr, markdown, markdown, zeallot, rmdformats, bio3d (>= 2.4), signalHsmm (>= 1.5), tidyverse (>= 1.3), covr, patchwork

VignetteBuilder  knitr

LazyData  FALSE

BiocType  Software

URL  https://fursham-h.github.io/factR/

git_url  https://git.bioconductor.org/packages/factR
git_branch  RELEASE_3_19

git_last_commit  f704331

git_last_commit_date  2024-04-30

Repository  Bioconductor 3.19

Date/Publication  2024-05-17

Author  Fursham Hamid [aut, cre]

Maintainer  Fursham Hamid <fursham.h@gmail.com>

Contents

buildCDS .......................................................... 3
chrom_matched_query_gtf ..................................... 4
domains.known .................................................. 4
domains.out ..................................................... 5
filtereach ......................................................... 5
has_consistentSeqlevels ..................................... 6
importFASTA ....................................................... 7
importGTF ........................................................ 7
matchChromosomes ............................................. 8
matched_query_gtf ............................................. 9
matchGeneInfo ................................................... 10
mutateeach ......................................................... 11
new_query_gtf .................................................... 12
predictDomains ................................................ 12
predictNMD ....................................................... 13
query_cds ........................................................ 15
query_exons ...................................................... 16
query_gtf ........................................................ 16
ref_cds ........................................................... 17
ref_exons ........................................................ 17
ref_gtf ............................................................ 18
sorteach .......................................................... 19
subsetNewTranscripts ....................................... 20
trimTranscripts ................................................ 21
viewTranscripts ................................................ 22

Index  24
**buildCDS**

*Reference-guided construction of CDS on GTF object*

**Description**

`buildCDS()` is designed to construct CDS information on transcripts from query GTF object.

**Usage**

`buildCDS(query, ref, fasta)`

**Arguments**

- `query` GRanges object containing query GTF data.
- `ref` GRanges object containing reference GTF data.
- `fasta` BSgenome or Biostrings object containing genomic sequence

**Details**

The `buildCDS()` function will first search for known reference mRNAs in `query` and annotate its CDS information. For the remaining transcripts, `buildCDS()` will search for a putative translation start site using a database of annotated ATG codons from `ref`. Transcripts containing an open-reading frame will be assigned the newly-determined CDS information.

**Value**

GRanges object containing query exon entries and newly-constructed CDS information

**Author(s)**

Fursham Hamid

**Examples**

```r
# Load genome and datasets
library(BSgenome.Mmusculus.UCSC.mm10)
data(matched_query_gtf, ref_gtf)

# Build CDS
buildCDS(matched_query_gtf, ref_gtf, Mmusculus)
```
chrom_matched_query_gtf

Chromosome matched version of "query_gtf"

Description

query_gtf data which have been corrected for its seqlevels

Usage

data(chrom_matched_query_gtf)

Format

A GRanges object with 56 ranges and 3 metadata columns:

- **ranges**: Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type**: Entry type; transcript or exon
- **transcript_id**: ID given to transcripts
- **gene_id**: ID given to gene origin of transcripts
- **gene_name**: Name given to gene origin of transcripts ...

domains.known

Example output of predictDomains()

Description

Output dataframe from predictDomains() function. mRNAs from GENCODE mouse annotation was predicted for putative domain families.

Usage

data(domains.known)

Format

A data.frame with 85780 rows and 5 columns:

- **transcript**: Transcript ID of protein-coding RNAs
- **description**: Name of domain families
- **eval**: E-value score
- **begin**: Start position of domain in protein
- **end**: End position of domain in protein ...
domains.out

Example output of predictDomains()

Description
Output dataframe from predictDomains() function.

Usage
data(domains.out)

Format
A data.frame with 14880 rows and 5 columns:

- transcript: Transcript ID of protein-coding RNAs
- description: Name of domain families
- eval: E-value score
- begin: Start position of domain in protein
- end: End position of domain in protein ...

filtereach

Internally filter each element of a GenomicRangesList

Description
Internally filter each element of a GenomicRangesList

Usage
filtereach(x, ...)

Arguments

- x: GRangesList object
- ...: Logical conditions to filter each element in the GRanges by. Multiple conditions can be provided as comma-delimited inputs

Value
Filtered GRangesList object

Author(s)
Fursham Hamid
Examples

```r
# Load dataset
data(query_exons)

# select first element of each GRangesList item
filter_each(query_exons, dplyr::row_number() == 1)
```

```r
has_consistentSeqlevels

Test consistency of chromosome naming styles (aka seqlevels; e.g. "chr1" vs "1") across multiple objects
```

Description

This function will determine if all input ranges objects have the same chromosome naming convention. Input objects can be GenomicRanges, BSgenome or Biostrings object with seqlevel information.

Usage

```r
has_consistentSeqlevels(..., verbose = TRUE)
```

Arguments

- `...` Two or more objects with seqlevels information
- `verbose` Whether to print out message

Value

Logical value as to whether all objects have consistent seqlevel styles

Author(s)

Fursham Hamid

Examples

```r
## EXAMPLE USING TOY DATASET
require(GenomicRanges)

# Create toy GRanges objects
gr1 <- GRanges("1", IRanges(start = c(1, 101), width = c(20, 20)), "+")
gr2 <- GRanges("chr1", IRanges(start = c(1, 101), width = c(20, 20)), "+")

# Test for seqlevels consistency
has_consistentSeqlevels(gr1, gr2)
```
## importFASTA

Import FASTA file into R

### Description

This function is a wrapper to Biostrings::readDNAStringSet() function to import FASTA genome sequence file and simultaneously convert long chromosome names (e.g. 1 dna:chromosome chromosome:GRCm38:1:1:195471971:1 REF) to short names (e.g. 1)

### Usage

```r
importFASTA(con)
```

### Arguments

- **con**  
  Path to FASTA file

### Value

Imported DNAStringSet object

### Author(s)

Fursham Hamid

---

## importGTF

Import GTF file into R

### Description

This function loads GTF files into R and converts it into

a wrapper to rtracklayer::import() function to conveniently import GTF file into R as a GenomicRanges object.

### Usage

```r
importGTF(con)
```
matchChromosomes

**Arguments**

- **con** Path to GTF file

**Value**

Imported GenomicRanges object in GTF format

**Author(s)**

Fursham Hamid

**Examples**

```r
gtf <- system.file("extdata", "sc_merged_sample.gtf.gz", package = "factR")
importGTF(gtf)
```

---

**matchChromosomes**

*Match seqlevels of input GRanges to reference GRanges or BioString objects*

**Description**

A convenient wrapper to match seqlevels of a query GRanges object to a reference object that contain seqlevels information. Reference can be a GRanges, GRangesList, BioString or DNAString object. Seqlevels which fail to match will be dropped.

**Usage**

```r
matchChromosomes(x, to)
```

**Arguments**

- **x** GRanges object with seqnames to change
- **to** GRanges object from which seqnames is referenced

**Value**

Corrected input GRanges

**Author(s)**

Fursham Hamid
Examples

```r
# EXAMPLE USING TOY DATASET

require(GenomicRanges)

# Create toy GRanges objects
gr1 <- GRanges("1", IRanges(start = c(1, 101), width = c(20, 20)), "+")
gr2 <- GRanges("chr1", IRanges(start = c(1, 101), width = c(20, 20)), "+")

# Match Ensembl-style chromosomes from gr1 to UCSC-style gr2
matchChromosomes(gr1, gr2)

# Possible to match chromosomes from GRanges object to a Biostrings
# object containing seqlevels
x0 <- c("chr2" = "CTCACCACTAT", "chr3" = "TGTCAGTCGA")
dna <- Biostrings::DNAStringSet(x0)

# Match gr1 to dna
matchChromosomes(gr1, dna)
```

Description

query_gtf data which have been corrected for its seqlevels and gene_ids

Usage

data(matched_query_gtf)

Format

A GRanges object with 56 ranges and 6 metadata columns:

- **ranges**: Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type**: Entry type; transcript or exon
- **transcript_id**: ID given to transcripts
- **gene_id**: Matched gene_id
- **old_gene_id**: Original gene_id
- **match_level**: Level of matching performed
- **gene_name**: Name of gene...
matchGeneInfo

Match gene metadata from query GTF to a reference GTF

Description

'matchGeneInfo()' matches and corrects Gene IDs from a query GTF object to a reference GTF

Usage

matchGeneInfo(query, ref, primary_gene_id = NULL, secondary_gene_id = NULL)

Arguments

query  Query GTF imported as GRanges object
ref    Reference GTF as GRanges object
primary_gene_id  Character name of the primary gene id metadata in query GTF. Input to this argument is typically 'gene_id'
secondary_gene_id  Character name of the secondary gene id in query file. Example of input to this argument is 'ref_gene_id'

Details

The default approach to this correction relies on finding overlaps between transcripts in query with transcripts in reference. Using this method alone could result in false positive matches (19 percent false positives). To improve this, users have the option to invoke two additional layers of matching. 
(1) Matching by ENSEMBL Gene_IDs. If both query and reference transcript annotations contain Ensembl-style Gene IDs, this program will try to match both IDs in a less stringent manner. This correction can be invoked by providing the 'primary_gene_id' argument
(2) Matching by secondary Gene_IDs. Depending on the transcript assembly program, GTF/GFF3 annotations may contain additional comments on the transcript information. This may include a distinct secondary Gene ID annotation that potentially matches with the reference. To invoke this correction, provide 'primary_gene_id' and 'secondary_gene_id' arguments. To determine if your transcript assembly contain possible secondary Gene IDs, import query GTF file using 'import-GTF()' and check its metadata columns

Value

Gene_id-matched query GRanges

Author(s)

Fursham Hamid
mutateeach

Examples

## EXAMPLE USING SAMPLE DATASET
# Load datasets
data(chrom_matched_query_gtf, ref_gtf)

# Run matching function
matchGeneInfo(chrom_matched_query_gtf, ref_gtf)

---

**mutateeach**

*Internally create or transform metadata of a GenomicRangesList*

Description

Internally create or transform metadata of a GenomicRangesList

Usage

mutateeach(x, ...)

Arguments

- **x**
  - GRangesList object

- **...**
  - Name-value pairs of expressions. The name of each argument will be the name of a new metadata column, and the value will be its corresponding value.

Value

Transformed GRangesList object

Author(s)

Fursham Hamid

Examples

# Load dataset
data(query_exons)

# Create chr:start-end id for each entry
mutateeach(query_exons, id = paste0(seqnames, ":", start, ":", end))
new_query_gtf  
*Query data containing CDS information*

**Description**

matched_query_gtf data that has undergone buildCDS function and containing CDS features

**Usage**

data(new_query_gtf)

**Format**

A GRanges object with 105 ranges and 7 metadata columns:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ranges</td>
<td>Chromosome, start, end, and strand info of 4 transcripts and its exons</td>
</tr>
<tr>
<td>type</td>
<td>Entry type; transcript or exon</td>
</tr>
<tr>
<td>transcript_id</td>
<td>ID given to transcripts</td>
</tr>
<tr>
<td>gene_id</td>
<td>Matched gene_id</td>
</tr>
<tr>
<td>old_gene_id</td>
<td>Original gene_id</td>
</tr>
<tr>
<td>match_level</td>
<td>Level of matching performed</td>
</tr>
<tr>
<td>gene_name</td>
<td>Name of gene</td>
</tr>
<tr>
<td>phase</td>
<td>Phase of open-reading frame</td>
</tr>
</tbody>
</table>

predictDomains  
*Predict protein domain families from coding transcripts*

**Description**

Predict protein domain families from coding transcripts

**Usage**

predictDomains(x, fasta, ..., plot = FALSE, progress_bar = FALSE, ncores = 4)

**Arguments**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>Can be a GRanges object containing 'CDS' features in GTF format</td>
</tr>
<tr>
<td>fasta</td>
<td>BSgenome or Biostrings object containing genomic sequence</td>
</tr>
<tr>
<td>...</td>
<td>Logical conditions to pass to dplyr::filter to subset transcripts for analysis. Variables are metadata information found in 'x' and multiple conditions can be provided delimited by comma. Example: transcript_id == &quot;transcript1&quot;</td>
</tr>
</tbody>
</table>
predictNMD

predictNMD(x, ..., cds = NULL, NMD_threshold = 50, progress_bar = TRUE)

**Description**

Predict NMD sensitivity on mRNA transcripts

**Usage**

predictNMD(x, ..., cds = NULL, NMD_threshold = 50, progress_bar = TRUE)

**Value**

Dataframe containing protein features for each cds entry
predictNMD

Arguments

x
Can be a GRanges object containing exon and CDS transcript features in GTF format.
Can be a GRangesList object containing exon features for a list of transcripts. If so, ‘cds’ argument have to be provided.
Can be a GRanges object containing exon features for a transcript. If so, ‘cds’ argument have to be provided.

... Logical conditions to pass to dplyr::filter to subset transcripts for analysis. Variables are metadata information found in ‘x’ and multiple conditions can be provided delimited by comma. Example: transcript_id == “transcript1”

cds
If ‘x’ is a GRangesList object, ‘cds’ has to be a GRangesList containing CDS features for the list of transcripts in ‘x’. List names in ‘x’ and ‘cds’ have to match.
If ‘x’ is a GRanges object, ‘cds’ has to be a GRanges containing CDS features for the transcript in ‘x’.

NMD_threshold Minimum distance of stop_codon to last exon junction (EJ) which triggers NMD. Default = 50bp

progress_bar Whether to display progress Default = TRUE

Value

Dataframe with prediction of NMD sensitivity and NMD features:
is_NMD: logical value in predicting transcript sensitivity to NMD
stop_to_lastEJ: Integer value of the number of bases between the first base of the stop_codon to the last base of EJ. A positive value indicates that the last EJ is downstream of the stop_codon.
um_of_down_EJs: Number of EJs downstream of the stop_codon.
‘3_UTR_length’: Length of 3’ UTR

Author(s)

Fursham Hamid

Examples

```r
# Load datasets
data(new_query_gtf, query_exons, query_cds)

# Using GTF GRanges as input
predictNMD(new_query_gtf)

# Using GTF GRanges as input
predictNMD(new_query_gtf, transcript_id == "transcript1")
```
query_cds
	ranscript_id %in% c("transcript1", "transcript3")

## Using exon and CDS GRangesLists as input
predictNMD(query_exons, cds = query_cds)
predictNMD(query_exons, cds = query_cds, transcript_id == "transcript3")

## Using exon and CDS GRanges as input
predictNMD(query_exons[[3]], cds = query_cds[[3]])

## ---------------------------------------------------------------------
## EXAMPLE USING TRANSCRIPT ANNOTATION
## ---------------------------------------------------------------------
library(AnnotationHub)

## Retrieve GRCm38 transcript annotation
ah <- AnnotationHub()
GRCm38_gtf <- ah["AH60127"]

## Run tool on specific gene family
predictNMD(GRCm38_gtf, gene_name == "Ptbp1")

query_cds

CDS from 4 transcripts entries of the same gene

Description
A dataset containing coordinates of CDS from 4 transcripts of mouse Ptbp1. Transcript names and
gene IDs have been modified

Usage
data(query_cds)

Format
A GRangesList object with 4 elements:

- **ranges** Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type** Entry type; transcript or exon
- **transcript_id** ID given to transcripts
- **phase** Phase of open-reading frame
- **built_from** Method by which CDS was built...
**query_exons**

**Description**

A dataset containing coordinates of exons from 4 transcripts of mouse Ptbp1. Transcript names and gene IDs have been modified.

**Usage**

data(query_exons)

**Format**

A GRangesList object with 4 elements:

- **ranges** Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type** Entry type; transcript or exon
- **transcript_id** ID given to transcripts
- **gene_id** Matched gene_id
- **old_gene_id** Original gene_id
- **match_level** Level of matching performed
- **gene_name** Name of gene ...

---

**query_gtf**

**Description**

A dataset containing coordinates of transcript and exons from 4 transcripts of mouse Ptbp1. Transcript names and gene IDs have been modified to demonstrate de novo origin of GTF.

**Usage**

data(query_gtf)

**Format**

A GRanges object with 56 ranges and 3 metadata columns:

- **ranges** Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type** Entry type; transcript or exon
- **transcript_id** Name or ID given to transcripts
- **gene_id** Name or ID given to gene origin of transcripts ...

---
**ref_cds**  

**Source**  

http://www.ensembl.org/

---

**Description**  

A dataset containing coordinates of CDS from 2 reference transcripts of mouse Ptbp1.

**Usage**  

data(ref_cds)

**Format**  

A GRangesList object with 2 elements:  

- **ranges**  Chromosome, start, end, and strand info of 4 transcripts and its exons  
- **type**  Entry type: transcript or exon  
- **phase**  Phase of open-reading frame  
- **gene_id**  Matched gene_id  
- **gene_name**  Name of gene  
- **transcript_id**  ID given to transcripts ...

**Source**  

https://www.gencodegenes.org/

---

**ref_exons**  

**Description**  

A dataset containing coordinates of exons from 2 reference transcripts of mouse Ptbp1.

**Usage**  

data(ref_exons)
Format

A GRangesList object with 2 elements:

- **ranges**: Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type**: Entry type; transcript or exon
- **phase**: Phase of open-reading frame
- **gene_id**: Matched gene_id
- **gene_name**: Name of gene
- **transcript_id**: ID given to transcripts ...

Source

[https://www.gencodegenes.org/](https://www.gencodegenes.org/)

---

**ref_gtf**

*Imported GTF file containing 2 reference transcript entries of the same gene*

Description

A dataset containing coordinates of transcript and exons from 2 reference transcripts of mouse Ptbp1.

Usage

```r
data(ref_gtf)
```

Format

A GRanges object with 64 ranges and 5 metadata columns:

- **ranges**: Chromosome, start, end, and strand info of 4 transcripts and its exons
- **type**: Entry type; transcript or exon
- **phase**: Phase of open-reading frame
- **gene_id**: Matched gene_id
- **gene_name**: Name of gene
- **transcript_id**: ID given to transcripts ...

Source

[https://www.gencodegenes.org/](https://www.gencodegenes.org/)
sorteach

**Internally sort each element of a GenomicRangesList**

**Description**

Internally sort each element of a GenomicRangesList

**Usage**

sorteach(x, ...)

**Arguments**

x  
GRangesList object

...  
Comma separated list of unquoted variable names to sort by. Variables are names of metadata columns found in GRangesList object. Use desc() to sort a variable in descending order. Input can be 'exonorder' to sort each element in exon order

**Value**

Sorted GRangesList object

**Author(s)**

Fursham Hamid

**Examples**

# Load dataset
data(query_exons)

# sort elements in each GRangesList in descending coordinate order
query_exons_desc <- sorteach(query_exons, dplyr::desc(start))

# sort elements in each GRangesList in its order in transcript
query_exons_exonorder <- sorteach(query_exons_desc, exonorder)

# test similarity of query_exons and query_exons_exonorder
identical(query_exons, query_exons_exonorder)
subsetNewTranscripts  \textit{Shortlist GTF GRanges object for new transcripts}

\textbf{Description}

‘subsetNewTranscripts()’ will retain transcripts in ‘query’ that are distinct from those in ‘ref’

\textbf{Usage}

\begin{verbatim}
subsetNewTranscripts(query, ref, refine.by = c("none", "intron", "cds"))
\end{verbatim}

\textbf{Arguments}

- \texttt{query}  \texttt{GRanges} object containing query GTF data.
- \texttt{ref}  \texttt{GRanges} object containing reference GTF data.
- \texttt{refine.by}  Whether to refine the selection process by removing query transcripts with similar introns or CDS structure to reference. Default input is "none", and can be changed to "intron" or "cds" respectively.

\textbf{Details}

‘subsetNewTranscripts()’ will compare query and reference GTF GRanges and return query transcripts with different exon structures from reference transcripts. Transcriptome assemblers may sometime extend 5' and 3' ends of known transcripts based on experimental data. These annotated transcripts can be removed by inputting "intron" to the refine.by argument. This will further compare and remove transcripts of identical intron structures. Alternatively, transcripts with unique CDS coordinates can be selected by typing "cds" to the refine.by argument.

\textbf{Value}

Filtered GRanges GTF object

\textbf{Author(s)}

Fursham Hamid

\textbf{Examples}

\begin{verbatim}
# Load dataset
data(matched_query_gtf, ref_gtf)

# Shortlist new transcripts
subsetNewTranscripts(matched_query_gtf, ref_gtf)
\end{verbatim}
trimTranscripts

Resize 5’ and 3’ ends of a transcript GenomicRanges

Description

Resize 5’ and 3’ ends of a transcript GenomicRanges

Usage

trimTranscripts(x, start = 0, end = 0)

Arguments

x

GRanges or GRangesList object containing exon coordinates for each transcript

start

Number of bases to trim from the start of transcript. Providing a negative value
will extend the transcript instead. If ‘x’ is a GRanges object, ‘start’ is a single
integer. If ‘x’ is a GRangesList, ‘start’ can be a single integer or a list of integers
of the same length as ‘x’

end

Number of bases to trim from the end of transcript. Providing a negative value
will extend the transcript instead. If ‘x’ is a GRanges object, ‘end’ is a single
integer. If ‘x’ is a GRangesList, ‘end’ can be a single integer or a list of integers
of the same length as ‘x’

Value

Trimmed GenomicRanges object

Author(s)

Fursham Hamid

Examples

library(GenomicRanges)
gr1 <- GRanges(
    seqnames = "chr1", strand = c("+", "+", "+"),
    ranges = IRanges(
        start = c(1, 500, 1000),
        end = c(100, 600, 1100)
    )
)

trimTranscripts(gr1, 20, 80)
trimTranscripts(gr1, 110, 150)
viewTranscripts  

Plot transcripts directly from GTF.

Description

A wrapper around wiggleplotr’s plotTranscripts function. See the documentation for (plotTranscripts) for more information.

Usage

viewTranscripts(x, ..., rescale_introns = FALSE, ncol = 1)

Arguments

x  
GRanges object containing transcript annotation in GTF format

...  
Character value of features to plot. Multiple features can be plotted by entering comma-delimited values. Features will be extracted from metadata gene_name, gene_id and transcript_id of the GTF. Can also be a conditional statement to filter values from variables in the GTF (e.g. gene_name == "Ptbp1")

rescale_introns  
Specifies if the introns should be scaled to fixed length or not. (default: FALSE)

ncol  
Number of columns to patch the output plots (default: 1)

Value

ggplot2 object. If multiple genes are detected, plots will be combined using patchwork

Author(s)

Fursham Hamid

Examples

```r
## EXAMPLE USING SAMPLE DATASET
## -------------------------------------------
# Load datasets
data(query_gtf, ref_gtf)

viewTranscripts(query_gtf)
viewTranscripts(query_gtf, "transcript1")
viewTranscripts(ref_gtf)

## EXAMPLE USING TRANSCRIPT ANNOTATION
## -------------------------------------------
library(AnnotationHub)
```
## Retrieve GRCm38 transcript annotation
ah <- AnnotationHub()
GRCm38_gtf <- ah["AH60127"]

## Plot transcripts from Ptbp1 gene
viewTranscripts(GRCm38_gtf, "Ptbp1")

# Plot transcripts from Ptbp1 and Ptbp2 genes
viewTranscripts(GRCm38_gtf, "Ptbp1", "Ptbp2")
## Index

### *datasets*
- chrom_matched_query_gtf, 4
- domains.known, 4
- domains.out, 5
- matched_query_gtf, 9
- new_query_gtf, 12
- query_cds, 15
- query_exons, 16
- query_gtf, 16
- ref_cds, 17
- ref_exons, 17
- ref_gtf, 18

- buildCDS, 3
- chrom_matched_query_gtf, 4
- domains.known, 4
- domains.out, 5
- filtereach, 5
- has_consistentSeqlevels, 6
- importFASTA, 7
- importGTF, 7
- matchChromosomes, 8
- matched_query_gtf, 9
- matchGeneInfo, 10
- mutateeach, 11
- new_query_gtf, 12
- plotTranscripts, 22
- predictDomains, 12
- predictNMD, 13
- query_cds, 15
- query_exons, 16
- query_gtf, 16
- ref_cds, 17
- ref_exons, 17
- ref_gtf, 18
- sorteach, 19
- subsetNewTranscripts, 20
- trimTranscripts, 21
- viewTranscripts, 22