Package ‘hoardeR’

February 12, 2019

Type Package

Title Collect and Retrieve Annotation Data for Various Genomic Data
Using Different Webservices

Version 0.9.4-2

Date 2019-02-12

Author Daniel Fischer [aut, cre],
Anu Sironen [aut]

Maintainer Daniel Fischer <daniel.fischer@luke.fi>

Depends R (>= 3.2)

biocViews

Imports bamsignals (>= 1.10.0), Biostrings (>= 2.46.0), data.table (>= 1.11.4), GenomicRanges (>= 1.30.3), GenomicTools.fileHandler (>= 0.1.4), httr (>= 1.3.1), IRanges (>= 2.12.0), KernSmooth (>= 2.23-15), knitr (>= 1.20), MASS (>= 7.3-31), R.utils (>= 2.6.0), RCurl (>= 1.95), rmarkdown (>= 1.10), Rsamtools (>= 1.30.0), S4Vectors (>= 0.16.0), seqinr (>= 1.0-2), stringr (>= 1.3.1), XML (>= 3.98-1.1)

Description Cross-species identification of novel gene candidates using the NCBI web service is provided. Further, sets of miRNA target genes can be identified by using the targetscan.org API.

VignetteBuilder knitr

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-12 08:23:21 UTC

R topics documented:

hoardeR-package .................................................. 2
blastSeq ........................................................... 3
coverageDensity .................................................... 4
findSpecies ......................................................... 5
hoardeR-package

Collect and Retrieve Annotation Data for Various Genomic Data Using Different Web Services.

Details

- Package: hoardeR
- Type: Package
- Version: 0.9.4-2
- Date: 2019-02-12
- License: GPL
- LazyLoad: yes

Author(s)

Daniel Fischer, Anu Sironen
Maintainer: Daniel Fischer <daniel.fischer@luke.fi>

blastSeq

Sending Genomic Sequences to NCBI Blast service

Description

This function sends genomic sequences to the NCBI Blast service.

Usage

blastSeq(seq, n_blast=20, delay_req=3, delay_rid=60, email=NULL, xmlFolder=NULL, logFolder=NULL, keepInMemory=FALSE, database="refseq_genomes", verbose=TRUE, createLog=TRUE)

Arguments

seq
The fasta sequence that should be blasted (String).

n_blast
Amount of parallel blast requests, in case seq is a vector.

delay_req
Seconds between the single Blast requests.

delay_rid
Seconds between the single result requests.

email
User email, required information from NCBI (String).

xmlFolder
Path to the result folder.

logFolder
Path to the log folder.

keepInMemory
Logical, shall the results be kept in the memory.

database
The NCBI database to use.

verbose
Shall the program give extensive feedback.

createLog
Create log files, needed for continuing a crashed program.

Details

This function sends fasta sequences to the NCBI blast service. The defaults for the delays are required by NCBI and must not be smaller than the default values. Also, NCBI asks the user to provide an email address.

The input seq can be a vector of strings. In that case the sequences are one after another processed. The option n_blast sets then the upper threshold of how many blast requests are send to the NCBI Blast service at a time and kept running there parallel. It is here in the users obligation not to misuse the service with too many parallel requests.

The xmlFolder parameter specifies the folder to where the XML results will be stored. In case the folder does not exist, R will create it.

In case the option keepInMemory is set to TRUE the Blast results will be kept in memory, otherwise they will be just written to the HDD. Especially if many sequences are send to the blast service it is recommended to drop the result from the memory, meaning to set the option keepInMemory=FALSE. The option keepInMemory=TRUE is currently still under development and should not be used.
coverageDensity

If log files should be written (createLog=TRUE) a log path should be given in logPath. However, if a xmlPath is given and the option createLog=TRUE is set, then the log folder will be automatically created in the parental folder of the xmlFolder and is called logs.

Value

An xml file that contains the the NCBI result.

Author(s)

Daniel Fischer

Examples

```r
## Not run:
blastSeq("ACGTGCATCGACTAGCTACGACTACGACTATC", email="my.name@somewhere.com")

## End(Not run)
```

---

### coverageDensity

* Calculation of the coverage density *

**Description**

Calculates the coverage density.

**Usage**

```r
coverageDensity(folder, chr=c(1:22,"X","Y","MT"), chr.length=NULL, posneg=FALSE, verbose=TRUE, use.sqrt=FALSE, kernel.package="slideWindowSum", step.size=50000, window.size=100000, bw=100)
```

**Arguments**

- **folder**: folder with bam files
- **chr**: Chromosome names to be plotted.
- **chr.length**: Length of chromosome
- **posneg**: Logical, plot pos and neg strand
- **verbose**: Logical, verbose output
- **use.sqrt**: Logical, apply sqrt transformation
- **kernel.package**: Class of kernel smoother
- **step.size**: Step size in bases
- **window.size**: Window size in bases
- **bw**: Bandwidth parameter
findSpecies

Details
This function calculates the coverage of bam-files

Author(s)
Daniel Fischer

---

findSpecies  

Search in the species Object.

Description
This function output rows from the species object that contain a certain string.

Usage
findSpecies(string)

Arguments
- string: Search string.

Details
This function output rows from the species object that contain a certain string. It uses the grepl function to identify the corresponding rows.

Value
A data.frame.

Author(s)
Daniel Fischer

See Also
species, grepl

Examples
findSpecies("cattle")
getAnnotation: Downloading or Importing of Annotation Data

Description

This function downloads (if needed) the annotation file from a given species from NCBI and loads it into the namespace.

Usage

```
getAnnotation(species=NULL, assembly=NULL, annotationFolder=NULL, type="gff3", verbose)
```

Arguments

- **species**: The scientific name of the species (String).
- **assembly**: The NCBI assembly version.
- **annotationFolder**: The folder where the file will be stored.
- **type**: The file extension/format of the annotation file.
- **verbose**: Logical, if function gives feedback.

Details

This function downloads for a given species the annotation file, as provided from NCBI. The main parameters basically define the URL, where the file is located. The file is then downloaded into the folder, provided in `annotationFolder` and then imported to the namespace.

If a file has been downloaded previously, it will be loaded directly from that folder. In case the user wants to use an annotation that is not provided by NCBI, the corresponding files can also be placed into the same folder, following the naming scheme as suggested from the function and the function will load it from there.

Value

A `data.table` with the annotation information.

Author(s)

Daniel Fischer
getEnsgInfo

Examples

```r
## Not run:
susScrofa <- getAnnotation(species = "Sus scrofa",
annotationFolder="/home/user/annotation")

homoSapiens <- getAnnotation(species = "Homo sapiens",
annotationFolder="/home/user/annotation")

## End(Not run)
```

getEnsgInfo

Retrieves Gene Information From the NCBI Database.

Description

This function retrieves for a given Ensembl Number the corresponding information from the NCBI database.

Usage

```r
getEnsgInfo(ensg)
```

Arguments

- `ensg` Ensembl ID (String).

Details

This function retrieves for a given Ensembl Number the corresponding information from the NCBI database. The object `ensg` can also be a vector of Ensembl IDs.

Value

A matrix with information.

Author(s)

Daniel Fischer

Examples

```r
## Not run:
ensg <- c("ENSG00000174482", "ENSG00000113494")
getEnsgInfo(ensg)

## End(Not run)
```
getFastaFromBed  

*Get fasta information based on locations in bed-format*

**Description**

For a given fasta and a bed file this function can extract the nucleotide sequences and stores them as fasta file.

**Usage**

```r
getFastaFromBed(bed, species=NULL, assembly=NULL, fastaFolder=NULL, verbose=TRUE, export=NULL, filename=NULL)
```

**Arguments**

- `bed`  The location in bed format, see details.
- `species`  Define the species.
- `assembly`  Assembly identifier.
- `fastaFolder`  Location of the fasta files.
- `verbose`  Logical, should informative status updates be given.
- `export`  Foldername.
- `filename`  Filename to store the FA object.

**Details**

Function expects as an input a `data.frame` in bed format. This means, the first column should contain the chromosome, the second the start-coordinates, the third the end-coordinates. The forth column contains the ID of the loci.

If a standard species is used (as defined in the `species` data frame), the function automatically downloads the required files from NCBI, takes the loci and extracts then the nucleotide sequences from it. If the corresponding assembly is not available from NCBI an own fasta file can be provided. For that the fa-file needs to be in the fastaFolder and follow the same naming system as the NCBI files are labelled. In that case, the function suggests the correct filename for an unknown assembly.

The export function, specifies then a folder to where the fasta file should be stored. If no filename is provided, the filename is then the object name passed to the `bed` function.

**Value**

An `fa` object containing the nucleotide sequences in fasta format.

**Author(s)**

Daniel Fischer
getGeneLocation

Examples

## Not run:

```r
myBed <- data.frame(chr=c(1,2),
                     start=c(235265,12356742),
                     end=c(435265,12386742),
                     gene=c("LOC1", "LOC2"))

myFA <- getFastaFromBed(myBed, species="Homo sapiens", fastaFolder="/home/user/fasta/", export=TRUE)
```

## End(Not run)

### getGeneLocation

#### Extracting Gene Locations

**Description**

This function extracts the gene locations from an imported gtf file.

**Usage**

```r
getGeneLocation(gtf)
```

**Arguments**

- `gtf`: An imported gtf object.

**Details**

This function extracts the information from an imported gtf object.

**Value**

A matrix.

**Author(s)**

Daniel Fischer

**Examples**

## Not run:

```r
getGeneLocation(gtf)
```

## End(Not run)
getGeneSeq

Extracting a gene sequence from NCBI database.

Description

This function retrieves a gene sequence from the NCBI database.

Usage

getGeneSeq(chr, start, end, organism)

Arguments

chr  Chromosome number, numeric/string
start Start position, numeric
end   End position, numeric
organism Name of the organism, string

Details

Extracting a gene sequence from NCBI database. For a list of available organism, visit http://genome.ucsc.edu/cgi-bin/das/dsn. All id="." field are available.

Value

A string that contains the genomic sequence.

Author(s)

Daniel Fischer

Examples

## Not run:
# Extracting for Sus Scrofa, build version 3:
getGeneSeq(1,2134,14532,"susScr3")
getGeneSeq(10,1233312,1233350,"hg38")

## End(Not run)
getSequenceFromNCBI

Extracts a sequence from the NCBI webpage

Description
Retrieve a sequence from the NCBI webpage

Usage
getSequenceFromNCBI(id, file=NULL)

Arguments

<table>
<thead>
<tr>
<th>id</th>
<th>The gene identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>File name to where the sequence shall be stored</td>
</tr>
</tbody>
</table>

Details
This function extracts the sequence for a given identifier and then stores, if requested the sequence to the HDD.

Author(s)
Daniel Fischer

intersectXMLAnnot

Intersect XML object with annotation object

Description
For a annotation object this function intersects the loci of it with the output of the tableSpecies function.

Usage
intersectXMLAnnot(tabSpecies, annot, level="gene", flanking=NULL)

Arguments

| tabSpecies | The table with locations from tableSpecies. |
| annot      | The annotation object. |
| level      | The level of intersection. |
| flanking   | Allowed flanking space for intersection. |
**plotCoverage**

Details

Function expects as an input table from `tableSpecies` with the option `locations=TRUE`. Further, it needs an annotation object, as provided by the `getAnnotation` function. With that it intersects then the loci on the level as specified in `level`. Currently only "gene" is supported.

The `flanking` option allows for flanking space up- and down-stream of the genes. This is especially then useful if the novel gene candidates are in the extension of known genes (e.g. responsible for regulation or if they are novel exons.)

Value

A table with intersection loci.

Author(s)

Daniel Fischer

Examples

```r
## Not run:

pigHits <- tableSpecies(xmls, species="Sus scrofa", locations = TRUE)
ssannot <- getAnnotation(species = "Sus scrofa", annotationFolder="/home/user/annotation")
pigInter <- list()
for(i in 1:nrow(pigHits)){
  pigInter[[i]] <- intersectXMLAnnot(pigHits[i,], ssannot)
}
## End(Not run)
```

---

**Description**

Plots a coverage density object.

**Usage**

```r
plotCoverage(x, use.sqrt=TRUE)
```

**Arguments**

- `x` A coverage density object
- `use.sqrt` Logical, use sqrt scale?
Details

This function plots the coverage of bam-files

Author(s)

Daniel Fischer

plotHit

Visualization of a cross-species hit

Description

For each cross-species hit the function plots the similarity within that area together with an optional annotation and coverage track.

Usage

plotHit(hits, flanking=1, window=NULL, annot=TRUE, coverage=FALSE, smoothPara=NULL, diagonal=0.25, verbose=TRUE, output=FALSE, hitSpecies=NULL, hitSpeciesAssembly=NULL, origSpecies=NULL, origSpeciesAssembly=NULL, fastaFolder=NULL, origAnnot=NULL, hitAnnot=NULL, nTick=5, which=NULL, figureFolder=NULL, figurePrefix=NULL, indexOffset=0, bamFolder=NULL, bamFiles=NULL, groupIndex=NULL, groupColor=NULL, countWindow=NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hits</td>
<td>The hit object to be plotted.</td>
</tr>
<tr>
<td>flanking</td>
<td>Allowed flanking site in Mb.</td>
</tr>
<tr>
<td>window</td>
<td>Moving window size of similarity measure</td>
</tr>
<tr>
<td>annot</td>
<td>Logical, add annotation track</td>
</tr>
<tr>
<td>coverage</td>
<td>Logical, add coverage track</td>
</tr>
<tr>
<td>smoothPara</td>
<td>Smoothing parameter for coverage</td>
</tr>
<tr>
<td>diagonal</td>
<td>Threshold for allowed diagonal similarity</td>
</tr>
<tr>
<td>verbose</td>
<td>Logical, shall the function give status updates</td>
</tr>
<tr>
<td>output</td>
<td>Logical, shall numerical results be given</td>
</tr>
<tr>
<td>hitSpecies</td>
<td>Scientific identifier of the hit species</td>
</tr>
<tr>
<td>hitSpeciesAssembly</td>
<td>Version of the hit species assembly</td>
</tr>
<tr>
<td>origSpecies</td>
<td>Scientific name of the original species</td>
</tr>
<tr>
<td>origSpeciesAssembly</td>
<td>Version of the original species</td>
</tr>
<tr>
<td>fastaFolder</td>
<td>Location of the fasta files</td>
</tr>
</tbody>
</table>
origAnnot: Annotation object of the original species
hitAnnot: Annotation object of the hit species
nTick: Number of ticks on the annotation track
which: Which hits should be plotted
figureFolder: Folder where figures should be stored
figurePrefix: Prefix of the figure filenames
indexOffset: Offset of the running index of the filenames
bamFolder: Folder with the bam-files
bamFiles: Filenames of the bam-files
groupIndex: Index of subgroups in the bamfiles
groupColor: Vector with colors, one for each subgroup
countWindow: Window size to count the reads from bam-files.

Details

This function is the workhorse of hoardeR and visualizes the findings of the blast and intersection runs. It is really flexible to handle the hits and hence there are many different options. The required options are hits, hitSpecies, origSpecies and fastaFolder.

The hit object is an object as provided by intersectXMLAnnot and contains all intersections of interest (=intersections that are in close proximity of a gene in the hit species). Naturally the hit and the original species have to be specified as well as the folder, where the required fasta files are stored, or to where they should be downloaded. If the species are the default species from Ensembl (as can be seen in the data.frame species), the annotation and assembly will be automatically downloaded to the specified location on the harddrive. Changes from that version can be adjusted with the the hitSpeciesAssembly and origSpeciesAssembly options, but the filenames have still to match the convention, as they are provided by NCBI.

If in addition to the similarity also a coverage track should be added, the option coverage has to be set to TRUE. The option smoothPara sets then the level of smoothing of the coverage. By default no smoothing will be applied.

In case an annotation track is requested (annot=TRUE), the annotation objects need to be provided to the origAnnot and hitAnnot options.

The option diagonal defines the minimum level of similarity so that a (diagonal) match will be plotted. The colors are then towards green for total similarity and towards red for total disagree, based on a nucleotide mismatch matrix.

If the option verbose=TRUE is set, the function gives a verbose output while running. Further, if output=TRUE then, in addition to the figure also a data.frame with the numerical results is provided.

In case that hits contains more than one hit, the plotHit function plots for each hit a figure. In that case a folder should be provided to where the figures should be stored, this can be done with the figureFolder and figurePrefix options. In case only asserted hits of hits shall be plotted, they can be selected with the which option.

The function can also plot a coverage track over the similarity. For that, the option coverage=TRUE has to be set and a folder that contains the necessary bam-files has to be specified in bamFolder. By default all bam files in that folder are used, if only a subset is requested, the filenames can be
specified in `bamFiles`. In case several bam-files are given, the average coverage at each loci is used. Further, if the data contains subgroups (e.g., case/control), the vector `groupIndex` gives the group labels. Naturally its length should be similar to `bamFiles` (or similar to the total amount of files in the bam-folder). In case that more than one group is plotted in the coverage track, their colors can be defined in `groupColor`. Of course, this vector has to be as long as the number of groups are defined. The option `countWindow` controls the moving window length in which the number of counts is calculated. The default is the same length as the hit.

**Value**

Optional, a table with intersection loci.

**Author(s)**

Daniel Fischer

**Examples**

```r
## Not run:
pigInter.flank <- list()
for(i in 1:nrow(pigHits)){
pigInter.flank[[i]] <- intersectXMLAnnot(pigHits[i,], ssannot, flanking=100)
}

# Basic usage:
plotHit(hits=pigInter.flank,
        flanking=100,
        hitSpecies = "Sus scrofa",
        origSpecies = "Bos taurus",
        fastaFolder = "/home/user/fasta/",
        figureFolder = "/home/user/figures/"
        )

# Annotation tracks added:
plotHit(hits=pigInter.flank,
        flanking=100,
        hitSpecies = "Sus scrofa",
        origSpecies = "Bos taurus",
        fastaFolder = "/home/user/fasta/",
        figureFolder = "/home/user/figures/",
        origAnnot=btannot,
        hitAnnot=ssannot
        )

# Annotation and coverage added:
plotHit(hits=pigInter.flank,
        flanking=100,
        hitSpecies = "Sus scrofa",
        origSpecies = "Bos taurus",
        fastaFolder = "/home/daniel/fasta/",
        figureFolder = "/home/user/figures/",
        origAnnot=btannot,
        hitAnnot=ssannot
        coverage=TRUE,
        bamFolder = "/home/users/bams/"
        )
```
species

## End(Not run)

---

```r
print.fa
```

**Description**

Prints an `fa` object.

**Usage**

```r
## S3 method for class 'fa'
print(x, n=2, seq.out=50, ...)
```

**Arguments**

- `x`: Object of class `fa`.
- `n`: Amount of elements to be displayed, numeric.
- `seq.out`: Length of each element to be displayed, numeric.
- `...`: Additional parameters.

**Details**

The `print` function displays an `fa` object. By default just the first two elements with their first 50 bases are displayed. To display the full sequence, set `seq.out=NULL`.

**Author(s)**

Daniel Fischer

---

### species

**Available species at NCBI**

**Description**

This is a list of all organisms/species that are provided by NCBI and hence could end up in the Blast run. Further, it defines the default versions of the assemblies that will be downloaded if no further version is specified in `plotHit`, `getAnnotation` or `getFastaFromBed`.

**Format**

A data frame with 348 species.
subDose

Source

As downloaded on 05.10.2016 from

Examples

data(species)
summary(species)

---

subDose  Rewrite the Dose File from a Beagle Output

Description

This function takes a Dose Beagle output and rewrites the output.

Usage

subDose(file=NULL, vmmk=NULL, out=NULL, removeInsertions=TRUE, verbose=TRUE)

Arguments

file  Location of the original Beagle file (String).
vmmk  Location of the Variant Map Master key (String).
out  Name and location of the output file (String).
verbose  The function gives feedback.
removeInsertions  All Indels will be removed.

Details

This function takes a Beagle Dose file and rewrites the alleles from numerical to character, based on the information provided in a variant map master key.

Value

A rewritten beagle phased file.

Author(s)

Daniel Fischer
Description

This function takes a Gprobs Beagle output and rewrites the output.

Usage

subGprobs(file=NULL, vmmk=NULL, out=NULL, chunkSize=100000, removeInsertions=TRUE, verbose = TRUE, writeOut=TRUE)

Arguments

- **file** Location of the original Beagle file (String).
- **vmmk** Location of the Variant Map Master key (String).
- **out** Name and location of the output file (String).
- **chunkSize** For large Beagle files, the chunk size.
- **removeInsertions** All Indels will be removed.
- **verbose** The function gives feedback.
- **writeOut** Logical, write the output back to the HDD.

Details

This function takes a Beagle Gprobs file and rewrites the alleles from numerical to character, based on the information provided in a variant map master key. For larger files the function can process the rewriting in chunks in order to save memory.

Value

A rewritten beagle Gprobs file.

Author(s)

Daniel Fischer
Rewrite the Phased File from a Beagle Output

Description
This function takes a phased Beagle output and rewrites the output.

Usage
```
subphased(file=NULL, vmmk = NULL, out=NULL, chunkSize=100000, verbose=TRUE,
          removeInsertions=TRUE)
```

Arguments
- **file**: Location of the original Beagle file (String).
- **vmmk**: Location of the Variant Map Master key (String).
- **out**: Name and location of the output file (String).
- **chunkSize**: For large Beagle files, the chunk size.
- **verbose**: The function gives feedback.
- **removeInsertions**: All Indels will be removed.

Details
This function takes a Beagle phased file and rewrites the alleles from numerical to character, based on the information provided in a variant map master key. For larger files the function can process the rewriting in chunks in order to save memory.

Value
A rewritten beagle phased file.

Author(s)
Daniel Fischer
tableSpecies

**Description**

Tables the species in xml file

**Usage**

```
tableSpecies(xml, species=NULL, type="chr", minOutput=TRUE, exclude="", locations=FALSE)
```

**Arguments**

- `xml`: The xml file.
- `species`: Restrict species to a certain set.
- `type`: Filter option.
- `minOutput`: Logical, should the output be minimal.
- `exclude`: Names of species to exclude.
- `locations`: Logical, shall the hit locations be given as well.
**Details**

Function provides a table of identified species. This table can e.g. be put into the `barplot` function to visualize the findings.

Further, if the option `locations` is set to `TRUE` the function not only tables the species, but also the individual locations of the hits. This output is required for the further steps. Hence, this function plays a important role in the identification pipeline.

Be default the option `type=“chr”` is set so that only hits in species will full genomes will be reported. Further, the species names are intersected with the species data frame and only those that appear there are reported.

**Value**

A table with the species from the XML file

**Author(s)**

Daniel Fischer

**Examples**

```r
## Not run:
tableSpecies(xmls)
pigHits <- tableSpecies(xmls, species=“Sus scrofa”, locations = TRUE)
## End(Not run)
```

---

**targetScan**

Retrieving miRNA target information from targetscan.org

**Description**

This function requests from the webpage targetscan.org the stored information for mirnas.

**Usage**

```r
targetScan(mirna=NULL, species=NULL, release="7.1", maxOut=NULL)
```

**Arguments**

- `mirna` The name of the mirna (String).
- `species` The species identifier, see details (String).
- `release` The release version of targetscan.org.
- `maxOut` The amount of target genes, default (NULL) is all.
Details

This function sends a miRNA name to the targetscan.org webpage, retrieves the information and gives it back as a data.frame. Options for species are "Human", "Mouse", "Rat", "Chimpanzee", "Rhesus", "Cow", "Dog", "Opossum", "Chicken", "Frog".

Value

A data.frame with the following columns

- Ortholog: The ortholog name of the target gene.
- geneName: The long description of the target gene.
- consSites: The total number of conserved sites.
- poorlySites: The total number of poorly conserved sites.

Author(s)

Daniel Fischer

References


Examples

```r
## Not run:
targetScan(mirna="miR-9-5p", species="Cow", maxOut=5)
## End(Not run)
```
Index

*Topic datasets
  species, 16

*Topic methods
  blastSeq, 3
  coverageDensity, 4
  findSpecies, 5
  getAnnotation, 6
  getEnsgInfo, 7
  getFastaFromBed, 8
  getGeneLocation, 9
  getGeneSeq, 10
  getSequenceFromNCBI, 11
  intersectXMLAnnot, 11
  plotCoverage, 12
  plotHit, 13
  print.fa, 16
  subDose, 17
  subGprobs, 18
  subPhased, 19
  summary.fa, 20
  tableSpecies, 20
  targetScan, 21

*Topic multivariate
  hoardeR-package, 2

*Topic print
  print.fa, 16
  summary.fa, 20

blastSeq, 3
coverageDensity, 4
findSpecies, 5
getAnnotation, 6
getEnsgInfo, 7
getFastaFromBed, 8
getGeneLocation, 9
getGeneSeq, 10
getSequenceFromNCBI, 11

hoardeR-package, 2
intersectXMLAnnot, 11
plotCoverage, 12
plotHit, 13
print.fa-method (print.fa), 16
print.fa, 16

R/hoardeR-package (hoardeR-package), 2
species, 5, 16
subDose, 17
subGprobs, 18
subPhased, 19
summary.fa-method (summary.fa), 20
summary.fa, 20

tableSpecies, 20
targetScan, 21