MyGene.info R Client

Adam Mark, Ryan Thompson, Chunlei Wu

October 24, 2023

Contents

1 Overview ............................................. 2
2 Gene Annotation Service .............................. 2
   2.1 getGene ........................................... 2
   2.2 getGenes ......................................... 2
3 Gene Query Service ................................... 3
   3.1 query ............................................... 3
   3.2 queryMany ...................................... 4
4 makeTxDbFromMyGene. ............................... 5
5 Tutorial, ID mapping ................................. 6
   5.1 Mapping gene symbols to Entrez gene ids ...... 6
   5.2 Mapping gene symbols to Ensembl gene ids .... 7
   5.3 When an input has no matching gene .......... 8
   5.4 When input ids are not just symbols .......... 8
   5.5 When an input id has multiple matching genes .. 9
   5.6 Can I convert a very large list of ids? ....... 11
6 References ............................................ 11
1 Overview

MyGene.Info provides simple-to-use REST web services to query/retrieve gene annotation data. It’s designed with simplicity and performance emphasized. mygene is an easy-to-use R wrapper to access MyGene.Info services.

2 Gene Annotation Service

2.1 getGene

- Use `getGene`, the wrapper for GET query of "/gene/<geneid>" service, to return the gene object for the given geneid.

```r
> gene <- getGene("1017", fields="all")
> length(gene)
[1] 1
> gene["name"]
[[1]]
NULL
> gene["taxid"]
[[1]]
NULL
> gene["uniprot"]
[[1]]
NULL
> gene["refseq"]
[[1]]
NULL
```

2.2 getGenes

- Use `getGenes`, the wrapper for POST query of "/gene" service, to return the list of gene objects for the given character vector of geneids.

```r
> getGenes(c("1017","1018","ENSG000000148795"))
DataFrame with 3 rows and 7 columns
```
MyGene.info R Client

<table>
<thead>
<tr>
<th>query</th>
<th>_id</th>
<th>X_version</th>
<th>entrezgene</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;character&gt;</td>
<td>&lt;character&gt;</td>
<td>&lt;integer&gt;</td>
<td>&lt;character&gt;</td>
<td>&lt;character&gt;</td>
</tr>
<tr>
<td>1</td>
<td>1017</td>
<td>1017</td>
<td>4</td>
<td>1017 cyclin dependent kinase 2</td>
</tr>
<tr>
<td>2</td>
<td>1018</td>
<td>1018</td>
<td>4</td>
<td>1018 cyclin dependent kinase 2</td>
</tr>
<tr>
<td>3</td>
<td>ENSG00000148795</td>
<td>1586</td>
<td>3</td>
<td>1586 cytochrome P450 family 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>symbol</th>
<th>taxid</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDK2</td>
<td>9606</td>
</tr>
<tr>
<td>CDK3</td>
<td>9606</td>
</tr>
<tr>
<td>CYP17A1</td>
<td>9606</td>
</tr>
</tbody>
</table>

3 Gene Query Service

3.1 query

- Use `query`, a wrapper for GET query of "/query?q=<query>" service, to return the query result.

```r
> query(q="cdk2", size=5)

$took
[1] 9

$total
[1] 1442

$max_score
[1] 88.55403

$hits
     _id _score entrezgene name                        symbol taxid
  1 1017 88.55403 1017 cyclin dependent kinase 2    CDK2 9606
  2 12566 74.27112 12566 cyclin dependent kinase 2   Cdk2 10090
  3 5880545 63.78744 5880545 CDK2 EDI_169580 370354
  4 362817 62.84480 362817 cyclin dependent kinase 2 Cdk2 10116
  5 143384 62.07590 143384 CDK2 associated cullin domain 1 CACUL1 9606
```

```r
> query(q="NM_013993")

$took
[1] 6
```
$total
[1] 1

$max_score
[1] 1.672538

$hits

<table>
<thead>
<tr>
<th>_id</th>
<th>_score</th>
<th>entrezgene_name</th>
<th>symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>780</td>
<td>1.672538</td>
<td>discoidin domain receptor tyrosine kinase 1</td>
<td>DDR1</td>
</tr>
</tbody>
</table>

$taxid
1 9606

3.2 queryMany

- Use `queryMany`, a wrapper for POST query of "/query" service, to return the batch query result.

```r
> queryMany(c('1053_at', '117_at', '121_at', '1255_g_at', '1294_at'),
+ scopes="reporter", species="human")

Finished
Pass returnall=TRUE to return lists of duplicate or missing query terms.

DataFrame with 6 rows and 7 columns

<table>
<thead>
<tr>
<th>query</th>
<th>_id</th>
<th>_score</th>
<th>entrezgene</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1053_at</td>
<td>5982</td>
<td>18.7276</td>
<td>5982 replication factor C..</td>
<td></td>
</tr>
<tr>
<td>117_at</td>
<td>3310</td>
<td>19.5433</td>
<td>3310 heat shock protein f..</td>
<td></td>
</tr>
<tr>
<td>121_at</td>
<td>7849</td>
<td>19.3550</td>
<td>7849 paired box 8</td>
<td></td>
</tr>
<tr>
<td>1255_g_at</td>
<td>2978</td>
<td>20.1858</td>
<td>2978 guanylate cyclase ac..</td>
<td></td>
</tr>
<tr>
<td>1294_at</td>
<td>100847079</td>
<td>19.1850</td>
<td>100847079 microRNA 5193</td>
<td></td>
</tr>
<tr>
<td>1294_at</td>
<td>7318</td>
<td>19.1850</td>
<td>7318 ubiquitin like modif..</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>symbol</th>
<th>taxid</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC2</td>
<td>9606</td>
</tr>
<tr>
<td>HSPA6</td>
<td>9606</td>
</tr>
<tr>
<td>PAX8</td>
<td>9606</td>
</tr>
<tr>
<td>GUCA1A</td>
<td>9606</td>
</tr>
<tr>
<td>MIR5193</td>
<td>9606</td>
</tr>
<tr>
<td>UBA7</td>
<td>9606</td>
</tr>
</tbody>
</table>
makeTxDbFromMyGene

TxDb is a container for storing transcript annotations. makeTxDbFromMyGene allows the user to make a TxDb object in the Genomic Features package from a mygene "exons" query using a default mygene object.

```r
> xli <- c('CCDC83',
   + 'MAST3',
   + 'RPL11',
   + 'ZDHHC20',
   + 'LUC7L3',
   + 'SNORD49A',
   + 'CTSH',
   + 'ACOT8')
> txdb <- makeTxDbFromMyGene(xli,
   + scopes="symbol", species="human")
> transcripts(txdb)
```

**GRanges object with 17 ranges and 2 metadata columns:**

```
seqnames ranges strand | tx_id tx_name
--- --- --- | --- ---
[1] 11 85855382-85920013 + | 1 NM_001286159
[2] 11 85855382-85920013 + | 2 NM_173556
[3] 19 18097777-18151686 + | 3 NM_015016
[4] 1 23691805-23696835 + | 4 NM_000975
[5] 1 23691778-23696426 + | 5 NM_001199802
... ... ... | ... ...
[13] 17 50719602-50756215 + | 13 NM_0016424
[14] 17 16440035-16440106 + | 14 NR_002744
[15] 15 78921749-78945098 - | 15 NM_001319137
[16] 15 78921859-78945046 - | 16 NM_004390
[17] 20 45841720-45857392 - | 17 NM_005469
```

seqinfo: 7 sequences from an unspecified genome; no seqlengths

makeTxDbFromMyGene invokes either the query or queryMany method and passes the response to construct a TxDb object. See ?TxDb for methods to utilize and access transcript annotations.
5 Tutorial, ID mapping

ID mapping is a very common, often not fun, task for every bioinformatician. Supposedly you have a list of gene symbols or reporter ids from an upstream analysis, and then your next analysis requires to use gene ids (e.g. Entrez gene ids or Ensembl gene ids). So you want to convert that list of gene symbols or reporter ids to corresponding gene ids.

Here we want to show you how to do ID mapping quickly and easily.

5.1 Mapping gene symbols to Entrez gene ids

Suppose xli is a list of gene symbols you want to convert to entrez gene ids:

```r
> xli <- c('DDX26B',
+ 'CCDC83',
+ 'MAST3',
+ 'FLOT1',
+ 'RPL11',
+ 'ZDHHC20',
+ 'LUC7L3',
+ 'SNORD49A',
+ 'CTSH',
+ 'ACOT8')
```

You can then call `queryMany` method, telling it your input is `symbol`, and you want `entrezgene` (Entrez gene ids) back.

```r
> queryMany(xli, scopes="symbol", fields="entrezgene", species="human")
```

Finished

Pass `returnall=TRUE` to return lists of duplicate or missing query terms.

Dataframe with 10 rows and 5 columns

<table>
<thead>
<tr>
<th>query</th>
<th>notfound</th>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDX26B</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>CCDC83</td>
<td>NA</td>
<td>220047</td>
<td>17.8989</td>
<td>220047</td>
</tr>
<tr>
<td>MAST3</td>
<td>NA</td>
<td>23031</td>
<td>18.0033</td>
<td>23031</td>
</tr>
<tr>
<td>FLOT1</td>
<td>NA</td>
<td>10211</td>
<td>18.2866</td>
<td>10211</td>
</tr>
<tr>
<td>RPL11</td>
<td>NA</td>
<td>6135</td>
<td>16.5730</td>
<td>6135</td>
</tr>
<tr>
<td>ZDHHC20</td>
<td>NA</td>
<td>253832</td>
<td>18.0896</td>
<td>253832</td>
</tr>
<tr>
<td>LUC7L3</td>
<td>NA</td>
<td>51747</td>
<td>17.6133</td>
<td>51747</td>
</tr>
<tr>
<td>SNORD49A</td>
<td>NA</td>
<td>26800</td>
<td>22.0672</td>
<td>26800</td>
</tr>
</tbody>
</table>
### 5.2 Mapping gene symbols to Ensembl gene ids

Now if you want Ensembl gene ids back:

```r
> out <- queryMany(xli, scopes="symbol", fields="ensembl.gene", species="human")
```

Finished

Pass returnall=TRUE to return lists of duplicate or missing query terms.

```r
> out
```

DataFrame with 10 rows and 5 columns

<table>
<thead>
<tr>
<th>query</th>
<th>notfound</th>
<th>_id</th>
<th>X_score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDX26B</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>CCDC83</td>
<td>NA</td>
<td>220047</td>
<td>17.8989</td>
</tr>
<tr>
<td>MAST3</td>
<td>NA</td>
<td>23031</td>
<td>18.0048</td>
</tr>
<tr>
<td>FLOT1</td>
<td>NA</td>
<td>10211</td>
<td>18.2773</td>
</tr>
<tr>
<td>RPL11</td>
<td>NA</td>
<td>6135</td>
<td>16.5740</td>
</tr>
<tr>
<td>ZDHHC20</td>
<td>NA</td>
<td>253832</td>
<td>18.0919</td>
</tr>
<tr>
<td>LUC7L3</td>
<td>NA</td>
<td>51747</td>
<td>17.6099</td>
</tr>
<tr>
<td>SNORD49A</td>
<td>NA</td>
<td>26800</td>
<td>22.0580</td>
</tr>
<tr>
<td>CTSH</td>
<td>NA</td>
<td>1512</td>
<td>17.6049</td>
</tr>
<tr>
<td>ACOT8</td>
<td>NA</td>
<td>10005</td>
<td>17.5923</td>
</tr>
</tbody>
</table>

```r
> out$ensembl[[4]]$gene
```

```
[1] "ENSG00000150676" "ENSG00000099308"
[5] "ENSG00000236710" "ENSG00000236271"
5.3 When an input has no matching gene

In case that an input id has no matching gene, you will be notified from the output. The returned list for this query term contains `notfound` value as True.

```r
> xli <- c('DDX26B',
+ 'CCDC83',
+ 'MAST3',
+ 'FLOT1',
+ 'RPL11',
+ 'Gm10494')
> queryMany(xli, scopes="symbol", fields="entrezgene", species="human")

Finished
Pass returnall=TRUE to return lists of duplicate or missing query terms.
```

<table>
<thead>
<tr>
<th>query</th>
<th>notfound</th>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDX26B</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>CCDC83</td>
<td>NA</td>
<td>220047</td>
<td>17.8966</td>
<td>220047</td>
</tr>
<tr>
<td>MAST3</td>
<td>NA</td>
<td>23031</td>
<td>18.0036</td>
<td>23031</td>
</tr>
<tr>
<td>FLOT1</td>
<td>NA</td>
<td>10211</td>
<td>18.2887</td>
<td>10211</td>
</tr>
<tr>
<td>RPL11</td>
<td>NA</td>
<td>6135</td>
<td>16.5740</td>
<td>6135</td>
</tr>
<tr>
<td>Gm10494</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

5.4 When input ids are not just symbols

```r
> xli <- c('DDX26B',
+ 'CCDC83',
+ 'MAST3',
+ 'FLOT1',
+ 'RPL11',
+ 'Gm10494',
+ '1007_s_at',
+ 'AK125780')

Above id list contains symbols, reporters and accession numbers, and supposedly we want to get back both Entrez gene ids and uniprot ids. Parameters `scopes`, `fields`, `species` are all flexible enough to support multiple values, either a list or a comma-separated string:
> out <- queryMany(xli, scopes=c("symbol", "reporter","accession"),
+                  fields=c("entrezgene","uniprot"), species="human")

Finished
Pass returnall=TRUE to return lists of duplicate or missing query terms.

> out

Dataframe with 9 rows and 7 columns
query notfound _id X_score entrezgene uniprot.Swiss.Prot
<character> <logical> <character> <numeric> <character> <character>
1 DDX26B TRUE NA NA NA NA
2 CCDC83 NA 220047 17.8976 220047 Q8IWF9
3 MAST3 NA 23031 18.0033 23031 O60307
4 FLOT1 NA 10211 18.3021 10211 075955
5 RPL11 NA 6135 16.5719 6135 P62913
6 Gm10494 TRUE NA NA NA NA
7 1007_s_at NA 100616237 18.9420 100616237 NA
8 1007_s_at NA 780 18.9420 780 Q08345
9 AK125780 NA 118142757 21.5310 118142757 P43080

uniprot.TrEMBL
<list>
1
2 H0YDV3
3 A0A8V8TLL8,A0A994J700,V9GYV0,...
4 A2ABJ5,Q5ST80,A2AB10,...
5 A0A2R8Y447,Q5VVD0,Q5VVC8
6
7
8 A0A024RCQ1,A0A024RCL1,A0A0A0MSX3,...
9 A0A712V6E2,B2R9P6

> out$uniprot.Swiss.Prot[[5]]
[1] "P62913"

5.5 When an input id has multiple matching genes

From the previous result, you may have noticed that query term 1007_s_at
matches two genes. In that case, you will be notified from the output, and the
returned result will include both matching genes.

By passing returnall=TRUE, you will get both duplicate or missing query terms
```r
> queryMany(xli, scopes=c("symbol", "reporter", "accession"),
+           fields=c("entrezgene", "uniprot"), species='human', returnall=TRUE)

Finished
$response

Data Frame with 9 rows and 7 columns

<table>
<thead>
<tr>
<th>query</th>
<th>notfound</th>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
<th>uniprot.Swiss.Prot</th>
<th>uniprot.TrEMBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDX26B</td>
<td>TRUE</td>
<td>NA</td>
<td>17.8966</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>CCDC83</td>
<td>NA</td>
<td>220047</td>
<td>18.0033</td>
<td>220047</td>
<td>Q8IWF9</td>
<td></td>
</tr>
<tr>
<td>MAST3</td>
<td>NA</td>
<td>23031</td>
<td>18.2731</td>
<td>23031</td>
<td>O60307</td>
<td></td>
</tr>
<tr>
<td>FLOT1</td>
<td>NA</td>
<td>10211</td>
<td>18.5007</td>
<td>10211</td>
<td>O75955</td>
<td></td>
</tr>
<tr>
<td>RPL11</td>
<td>NA</td>
<td>6135</td>
<td>16.5907</td>
<td>6135</td>
<td>P62913</td>
<td></td>
</tr>
<tr>
<td>Gm10494</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>1007_s_at</td>
<td>NA</td>
<td>100616237</td>
<td>19.0089</td>
<td>100616237</td>
<td>NA</td>
<td></td>
</tr>
<tr>
<td>1007_s_at</td>
<td>NA</td>
<td>780</td>
<td>19.0089</td>
<td>780</td>
<td>Q08345</td>
<td></td>
</tr>
<tr>
<td>AK125780</td>
<td>NA</td>
<td>118142757</td>
<td>21.5310</td>
<td>118142757</td>
<td>P43080</td>
<td></td>
</tr>
</tbody>
</table>

uniprot.TrEMBL

<list>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>H0YDV3</td>
</tr>
<tr>
<td>2</td>
<td>A0A8V8TLL8,A0A994J700,V9GYV0,...</td>
</tr>
<tr>
<td>3</td>
<td>A2ABJ5,Q5ST80,A2AB10,...</td>
</tr>
<tr>
<td>4</td>
<td>A0A2R8Y447,Q5VVD0,Q5VVC8</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>A0A024RCQ1,A0A024RCL1,A0A0A0MSX3,...</td>
</tr>
<tr>
<td>9</td>
<td>A0A7I2V6E2,B2R9P6</td>
</tr>
</tbody>
</table>

$duplicates

X1007_s_at
1 2

$missing
[1] "DDX26B"  "Gm10494"
```

The returned result above contains **out** for mapping output, **missing** for missing query terms (a list), and **dup** for query terms with multiple matches (including the number of matches).
5.6 Can I convert a very large list of ids?

Yes, you can. If you pass an id list (i.e., \texttt{xli} above) larger than 1000 ids, we will do the id mapping in-batch with 1000 ids at a time, and then concatenate the results all together for you. So, from the user-end, it's exactly the same as passing a shorter list. You don't need to worry about saturating our backend servers. Large lists, however, may take a while longer to query, so please wait patiently.

6 References