MyGene.info R Client

Adam Mark, Ryan Thompson, Chunlei Wu

October 17, 2016

Contents

1 Overview 1

2 Gene Annotation Service 2
  2.1 getGene 2
  2.2 getGenes 3

3 Gene Query Service 3
  3.1 query 4
  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? 10

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[[1]]$name
[1] "cyclin dependent kinase 2"
> gene[[1]]$taxid
[1] 9606
> gene[[1]]$uniprot
$`Swiss-Prot`
[1] "P24941"
$TrEMBL
[1] "AOA024RB10" "AOA024RB77" "B4DDL9" "E7ESI2" "G3V317" "G3V5T9"
> gene[[1]]$refseq
$genomic
[1] "NC_000012.12" "NC_018923.2" "NG_034014.1"

$protein
[1] "NP_001277159.1" "NP_001789.2" "NP_439892.2" "XP_011536034.1"

$rna
[1] "NM_001290230.1" "NM_001798.4" "NM_052827.3" "XM_011537732.1"

$translation
$translation[[1]]
$translation[[1]]$protein
[1] "NP_439892.2"

$translation[[1]]$rna
[1] "NM_052827.3"

$translation[[2]]
$translation[[2]]$protein
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","ENSG00000148795"))
```

DataFrame with 3 rows and 7 columns

<table>
<thead>
<tr>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>cyclin dependent kinase 2</td>
</tr>
<tr>
<td>1017</td>
<td>19.85752</td>
<td>1017</td>
<td>cyclin dependent kinase 2</td>
</tr>
<tr>
<td>1018</td>
<td>21.85669</td>
<td>1018</td>
<td>cyclin dependent kinase 3</td>
</tr>
<tr>
<td>1586</td>
<td>21.46224</td>
<td>1586</td>
<td>cytochrome P450 family 17 subfamily A member 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>symbol</th>
<th>taxid</th>
<th>query</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDK2</td>
<td>9606</td>
<td>1017</td>
</tr>
<tr>
<td>CDK3</td>
<td>9606</td>
<td>1018</td>
</tr>
<tr>
<td>CYP17A1</td>
<td>9606</td>
<td>ENSG00000148795</td>
</tr>
</tbody>
</table>
### 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)
```

```
$hits
   _id  _score entrezgene             name symbol taxid
  1  1017  399.76840 1017 cyclin dependent kinase 2 CDK2 9606
  2 12566  287.59793 12566 cyclin-dependent kinase 2 Cdk2 10090
  3 362817 243.35211 362817 cyclin dependent kinase 2 Cdk2 10116
  4 143384  21.79327 143384 CDK2 associated cullin domain 1 CACUL1 9606
  5  52004  20.91099  52004 CDK2-associated protein 2 Cdk2ap2 10090

$total
[1] 21

$max_score
[1] 399.7684

$took
[1] 4

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

```
$took
[1] 6

$max_score
[1] 3.84777

$total
[1] 1

$hits
   _id  _score entrezgene             name symbol taxid
  1  780  3.847771  780 discoidin domain receptor tyrosine kinase 1 DDR1 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
### 4 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
taxdb <- makeTxDbFromMyGene(xli, scopes="symbol", species="human")
```

> transcripts(txdb)

GRanges object with 17 ranges and 2 metadata columns:

<table>
<thead>
<tr>
<th>seqnames</th>
<th>ranges</th>
<th>strand</th>
<th>tx_id</th>
<th>tx_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>[85855100, 85920020]</td>
<td>+</td>
<td>1</td>
<td>NM_001286159</td>
</tr>
<tr>
<td>[2]</td>
<td>[85855100, 85920020]</td>
<td>+</td>
<td>2</td>
<td>NM_173556</td>
</tr>
<tr>
<td>[3]</td>
<td>[18097792, 18151689]</td>
<td>+</td>
<td>3</td>
<td>NM_015016</td>
</tr>
<tr>
<td>[4]</td>
<td>[23691778, 23696426]</td>
<td>+</td>
<td>4</td>
<td>NM_000975</td>
</tr>
<tr>
<td>[5]</td>
<td>[23691778, 23696426]</td>
<td>+</td>
<td>5</td>
<td>NM_001199802</td>
</tr>
</tbody>
</table>
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.

Data Frame with 10 rows and 5 columns

<table>
<thead>
<tr>
<th>notfound</th>
<th>query</th>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
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

query notfound _id X_score

<character> <logical> <character> <numeric>
1 DDX26B TRUE NA NA
2 CCDC83 NA 220047 96.21670
3 MAST3 NA 23031 96.56346
4 FLOT1 NA 10211 98.29359
5 RPL11 NA 6135 91.59504
6 ZDHHC20 NA 253832 95.85616
7 LUC7L3 NA 51747 93.68782
8 SNORD49A NA 26800 114.40928
9 CTSH NA 1512 96.56941
10 ACOT8 NA 10005 93.95580

ensembl

<list>
1
2 ENSG00000150676
3 ENSG00000099308
c("ENSG00000236271", "ENSG00000230143", "ENSG00000224740")
5 ENSG00000142676
6 ENSG00000180776
7 ENSG00000108848
8 ENSG00000277370
```
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")
```

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

```
DataFrame with 6 rows and 5 columns

<table>
<thead>
<tr>
<th>notfound</th>
<th>query</th>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>DDX26B</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>CCDC83</td>
<td>220047</td>
<td>96.21670</td>
<td>220047</td>
</tr>
<tr>
<td>NA</td>
<td>MAST3</td>
<td>23031</td>
<td>96.56346</td>
<td>23031</td>
</tr>
<tr>
<td>NA</td>
<td>FLOT1</td>
<td>10211</td>
<td>98.29359</td>
<td>10211</td>
</tr>
<tr>
<td>NA</td>
<td>RPL11</td>
<td>6135</td>
<td>91.59504</td>
<td>6135</td>
</tr>
<tr>
<td>TRUE</td>
<td>Gm10494</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:

```r
> 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.

```r
> out
```

DataFrame 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>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;character&gt;</td>
<td>&lt;logical&gt;</td>
<td>&lt;character&gt;</td>
<td>&lt;numeric&gt;</td>
<td>&lt;integer&gt;</td>
<td>&lt;character&gt;</td>
</tr>
<tr>
<td>1 DDX26B</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>2 CCDC83</td>
<td>NA</td>
<td>220047</td>
<td>77.268326</td>
<td>220047</td>
<td>Q8IWF9</td>
</tr>
<tr>
<td>3 MAST3</td>
<td>NA</td>
<td>23031</td>
<td>77.846650</td>
<td>23031</td>
<td>060307</td>
</tr>
<tr>
<td>4 FLOT1</td>
<td>NA</td>
<td>10211</td>
<td>80.660700</td>
<td>10211</td>
<td>075955</td>
</tr>
<tr>
<td>5 RPL11</td>
<td>NA</td>
<td>6135</td>
<td>70.038470</td>
<td>6135</td>
<td>P62913</td>
</tr>
<tr>
<td>6 Gm10494</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>7 1007_s_at</td>
<td>NA</td>
<td>780</td>
<td>12.923538</td>
<td>780</td>
<td>Q08345</td>
</tr>
<tr>
<td>8 1007_s_at</td>
<td>NA</td>
<td>100616237</td>
<td>12.919736</td>
<td>100616237</td>
<td>NA</td>
</tr>
<tr>
<td>9 AK125780</td>
<td>NA</td>
<td>2978</td>
<td>5.168179</td>
<td>2978</td>
<td>P43080</td>
</tr>
</tbody>
</table>

uniprot.TrEMBL

<table>
<thead>
<tr>
<th>&lt;list&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>H0YDV3</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>V9GYV0</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>A2AB09,A2AB10,A2AB11,...</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>Q5VVC8,Q5VVC9,Q5VVD0</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>A0A024RCJ0,A0A024RCL1,A0A024RCQ1,...</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>A0A0AOMTF5,A6PVH5,B2R9P6</td>
</tr>
</tbody>
</table>

```r
> 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
> queryMany(xli, scopes=c("symbol", "reporter", "accession"),
+     fields=c("entrezgene", "uniprot"), species='human', returnall=TRUE)

Finished
$response

DataFrame with 9 rows and 7 columns

query notfound _id X_score entrezgene uniprot.Swiss.Prot
<character> <logical> <character> <numeric> <integer> <character>
1 DDX26B TRUE NA NA NA NA
2 CCDC83 NA 220047 77.268326 220047 Q8IWF9
3 MAST3 NA 23031 77.846650 23031 Q60307
4 FLT1 NA 10211 80.660700 10211 Q75955
5 RPL11 NA 6135 70.038470 6135 P62913
6 Gm10494 TRUE NA NA NA NA
7 1007_s_at NA 780 12.923538 780 Q08345
8 1007_s_at NA 100616237 12.919736 100616237 NA
9 AK125780 NA 2978 5.168179 2978 P43080

uniprot.TrEMBL
<list>
1
2
3
4
5
6
7
8
9

$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., 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