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
$\text{\textdollar}\text{Swiss-Prot}$
[1] "P24941"

$\text{\textdollar}\text{TrEMBL}$
[1] "A0A024RB10" "A0A024RB77" "B4DDL9" "E7ESI2" "G3V317" "G3V5T9"
> gene[[1]]$refseq
$\text{genomic}$
[1] "NC_000012.12" "NC_018923.2" "NG_034014.1"

$\text{protein}$
[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"

$\text{translation}$
$\text{translation}[[1]]$
$\text{translation}[[1]]$protein
[1] "XP_011536034.1"

$\text{translation}[[1]]$rna
[1] "XM_011537732.1"

$\text{translation}[[2]]$
$\text{translation}[[2]]$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

_id  X_score  entrezgene  name                                          
<character> <numeric>  <integer>  <character>
1  1017    20.46760   1017 cyclin dependent kinase 2
2  1018    21.28927   1018 cyclin dependent kinase 3
3  1586    22.42408   1586 cytochrome P450 family 17 subfamily A member 1

symbol  taxid  query
<character>  <integer>  <character>
1  CDK2     9606  1017
2  CDK3     9606  1018
3  CYP17A1   9606  ENSG00000148795
```

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)

$max_score
[1] 414.2571

$took
[1] 6

$total
[1] 32

$hits
  _id   _score entrezgene          name   symbol taxid
1 1017  414.25710  1017 cyclin dependent kinase 2 CDK2  9606
2 12566 307.99966  12566 cyclin-dependent kinase 2 Cdk2 10090
3 362817 260.61510 362817 cyclin dependent kinase 2 Cdk2 10116
4 52004  20.81025  52004  CDK2-associated protein 2 Cdk2ap2 10090
5 143384 20.70036 143384  CDK2 associated cullin domain 1 CACUL1  9606

> query(q="NM_013993")

$max_score
[1] 4.1818

$took
[1] 15

$total
[1] 1

$hits
  _id   _score entrezgene          name   symbol taxid
1  780   4.1818  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
<table>
<thead>
<tr>
<th>_id</th>
<th>X_score</th>
<th>entrezgene</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5982</td>
<td>20.46712</td>
<td>5982</td>
<td>replication factor C subunit 2</td>
</tr>
<tr>
<td>3310</td>
<td>12.79256</td>
<td>3310</td>
<td>heat shock protein family A (Hsp70) member 6</td>
</tr>
<tr>
<td>7849</td>
<td>12.78363</td>
<td>7849</td>
<td>paired box 8</td>
</tr>
<tr>
<td>2978</td>
<td>10.22690</td>
<td>2978</td>
<td>guanylate cyclase activator 1A</td>
</tr>
<tr>
<td>7318</td>
<td>22.47452</td>
<td>7318</td>
<td>ubiquitin like modifier activating enzyme 7</td>
</tr>
<tr>
<td>100847079</td>
<td>20.45381</td>
<td>100847079</td>
<td>microRNA 5193</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>symbol</th>
<th>taxid</th>
<th>query</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFC2</td>
<td>9606</td>
<td>1053_at</td>
</tr>
<tr>
<td>HSPA6</td>
<td>9606</td>
<td>117_at</td>
</tr>
<tr>
<td>PAX8</td>
<td>9606</td>
<td>121_at</td>
</tr>
<tr>
<td>GUCA1A</td>
<td>9606</td>
<td>1255_g_at</td>
</tr>
<tr>
<td>UBA7</td>
<td>9606</td>
<td>1294_at</td>
</tr>
<tr>
<td>MIR5193</td>
<td>9606</td>
<td>1294_at</td>
</tr>
</tbody>
</table>

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
> xli <- c('DDX26B',
+            '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:

<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>&lt;Rle&gt;</td>
<td>&lt;IRanges&gt;</td>
<td>&lt;Rle&gt;</td>
<td>&lt;integer&gt;</td>
<td>&lt;character&gt;</td>
</tr>
<tr>
<td>[1] 11 [85855100, 85920020] +</td>
<td>1 NM_001286159</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

DataFrame 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>
5.2 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.

> out

Dataframe with 10 rows and 5 columns

<table>
<thead>
<tr>
<th>notfound</th>
<th>query</th>
<th>_id</th>
<th>X_score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>DDX26B</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>CCDC83</td>
<td>220047</td>
<td>97.63600</td>
</tr>
<tr>
<td>NA</td>
<td>MAST3</td>
<td>23031</td>
<td>99.67186</td>
</tr>
<tr>
<td>NA</td>
<td>FLOT1</td>
<td>10211</td>
<td>100.06795</td>
</tr>
<tr>
<td>NA</td>
<td>RPL11</td>
<td>6135</td>
<td>91.97031</td>
</tr>
<tr>
<td>NA</td>
<td>ZDHHC20</td>
<td>253832</td>
<td>99.29120</td>
</tr>
<tr>
<td>NA</td>
<td>LUC7L3</td>
<td>51747</td>
<td>95.53487</td>
</tr>
<tr>
<td>NA</td>
<td>SNORD49A</td>
<td>26800</td>
<td>114.37310</td>
</tr>
<tr>
<td>NA</td>
<td>CTSH</td>
<td>1512</td>
<td>97.06035</td>
</tr>
<tr>
<td>NA</td>
<td>ACOT8</td>
<td>10005</td>
<td>95.55949</td>
</tr>
</tbody>
</table>

ensembl

<table>
<thead>
<tr>
<th>&lt;list&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENSG00000150676</td>
</tr>
<tr>
<td>ENSG00000180776</td>
</tr>
<tr>
<td>ENSG00000108848</td>
</tr>
<tr>
<td>ENSG000001214508</td>
</tr>
<tr>
<td>ENSG00000236271,ENSG00000224740,ENSG00000137312</td>
</tr>
<tr>
<td>ENSG0000000142676</td>
</tr>
<tr>
<td>ENSG0000000180776</td>
</tr>
<tr>
<td>ENSG0000000108848</td>
</tr>
<tr>
<td>ENSG0000000277370</td>
</tr>
</tbody>
</table>
```
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.
DataFrame with 6 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>97.63600</td>
<td>220047</td>
</tr>
<tr>
<td>MAST3</td>
<td>NA</td>
<td>23031</td>
<td>99.67186</td>
<td>23031</td>
</tr>
<tr>
<td>FLOT1</td>
<td>NA</td>
<td>10211</td>
<td>100.06795</td>
<td>10211</td>
</tr>
<tr>
<td>RPL11</td>
<td>NA</td>
<td>6135</td>
<td>91.97031</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:

```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

```
notfound  query  _id  X_score entrezgene uniprot.Swiss.Prot
<logical> <character> <character> <numeric> <integer> <character>
1        TRUE DDX26B NA        NA       NA       NA
2           NA CCDC83 220047  77.574100  220047     Q8IWF9
3           NA  MAST3 230313  80.841225  230313     O60307
4           NA  FLOT1 102115  81.485954  102115     075955
5           NA  RPL11  61356  68.832990   61356       P62913
6        TRUE  Gm10494 NA        NA       NA       NA
7           NA 1007_s_at 100616237 12.792385 100616237     NA
8           NA 1007_s_at    7802  12.792251    7802       Q08345
9           NA AK125780  29782  5.113452   29782       P43080
```

uniprot.TrEMBL

```
<list>
```

1
2
3
4
5
6
7
8
9

> 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

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DDX26B</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>CCDC83</td>
<td>NA</td>
<td>220047</td>
<td>77.574100</td>
<td>220047</td>
<td>Q8IWF9</td>
</tr>
<tr>
<td>MAST3</td>
<td>NA</td>
<td>23031</td>
<td>80.841225</td>
<td>23031</td>
<td>O60307</td>
</tr>
<tr>
<td>FLQT1</td>
<td>NA</td>
<td>10211</td>
<td>81.485954</td>
<td>10211</td>
<td>O75955</td>
</tr>
<tr>
<td>RPL11</td>
<td>NA</td>
<td>6135</td>
<td>68.832990</td>
<td>6135</td>
<td>P62913</td>
</tr>
<tr>
<td>Gm10494</td>
<td>TRUE</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>1007_s_at</td>
<td>NA</td>
<td>100616237</td>
<td>12.792385</td>
<td>100616237</td>
<td>NA</td>
</tr>
<tr>
<td>1007_s_at</td>
<td>NA</td>
<td>780</td>
<td>12.792251</td>
<td>780</td>
<td>Q08345</td>
</tr>
<tr>
<td>AK125780</td>
<td>NA</td>
<td>2978</td>
<td>5.113452</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>H0YDV3</td>
</tr>
<tr>
<td>V9GYV0</td>
</tr>
<tr>
<td>A2AB09,A2AB10,A2AB11,...</td>
</tr>
<tr>
<td>Q5VVC8,Q5VVC9,Q5VVD0</td>
</tr>
<tr>
<td>AOA024RCJ0, AOA024RCJ1, AOA024RCQ1,...</td>
</tr>
<tr>
<td>AOA0A0MTF5,A6PVH5,B2R9P6</td>
</tr>
</tbody>
</table>

$duplicates

<table>
<thead>
<tr>
<th>X1007_s_at</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2</td>
</tr>
</tbody>
</table>

$missing

[1] "DDX26B" "Gm10494"

The returned result above contains:
- out for mapping output,
- missing for missing query terms (a list),
- 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