

# MyGene.info R Client

*Adam Mark, Ryan Thompson, Chunlei Wu*

October 27, 2020

## Contents

1	Overview . . . . .	2
2	Gene Annotation Service . . . . .	2
2.1	<code>getGene</code> . . . . .	2
2.2	<code>getGenes</code> . . . . .	2
3	Gene Query Service . . . . .	3
3.1	<code>query</code> . . . . .	3
3.2	<code>queryMany</code> . . . . .	4
4	<code>makeTxDbFromMyGene</code> . . . . .	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 . . . . .	10
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.

```
> 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.

```
> getGenes(c("1017", "1018", "ENSG00000148795"))  
  
DataFrame with 3 rows and 7 columns  
      query      _id X_version entrezgene      name  
      <character> <character> <integer> <character> <character>  
1      1017      1017         6      1017 cyclin dependent kin..  
2      1018      1018         1      1018 cyclin dependent kin..  
3 ENSG00000148795 1586         2      1586 cytochrome P450 fami..  
      symbol      taxid  
      <character> <integer>  
1      CDK2      9606  
2      CDK3      9606  
3      CYP17A1    9606
```

## 3 Gene Query Service

### 3.1 query

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

```
> query(q="cdk2", size=5)  
  
$took  
[1] 9  
  
$total  
[1] 963  
  
$max_score  
[1] 440.2821  
  
$hits  
      _id  _score entrezgene      name symbol taxid  
1      1017 440.2821      1017 cyclin dependent kinase 2  CDK2  9606  
2      12566 372.7855      12566 cyclin-dependent kinase 2  Cdk2 10090  
3      362817 312.4583      362817 cyclin dependent kinase 2  Cdk2 10116  
4 102732832 286.7581 102732832 cyclin dependent kinase 2  CDK2  9713  
5      711002 286.7581      711002 cyclin dependent kinase 2  CDK2  9544
```

## MyGene.info R Client

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

$took
[1] 8

$total
[1] 1

$max_score
[1] 3.753742

$hits
  _id  _score entrezgene          name symbol
1 780 3.753742      780 discoidin domain receptor tyrosine kinase 1  DDR1
  taxid
1 9606
```

### 3.2 queryMany

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

```
> 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
  query      _id  X_score  entrezgene          name
<character> <character> <numeric> <character>      <character>
1 1053_at    5982  13.6180      5982 replication factor C..
2 117_at     3310  13.5981      3310 heat shock protein f..
3 121_at     7849  12.3177      7849          paired box 8
4 1255_g_at  2978  12.8262      2978 guanylate cyclase ac..
5 1294_at    7318  13.6581      7318 ubiquitin like modif..
6 1294_at  100847079  13.6162  100847079          microRNA 5193
  symbol      taxid
<character> <integer>
1  RFC2      9606
2  HSPA6     9606
3  PAX8      9606
```

```

4      GUCA1A      9606
5      UBA7        9606
6      MIR5193     9606

```

## 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.

```

> 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
      <Rle>        <IRanges> <Rle> | <integer> <character>
 [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_016424
 [14]      17 16440035-16440106   + |        14  NR_002744
 [15]      15 78921749-78945098   - |        15 NM_001319137
 [16]      15 78921059-78945046   - |        16  NM_004390
 [17]      20 45841720-45857392   - |        17  NM_005469
-----
seqinfo: 7 sequences from an unspecified genome; no seqlengths

```

## MyGene.info R Client

`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:

```
> 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.

```
> 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
  query notfound _id X_score entrezgene
<character> <logical> <character> <numeric> <character>
1 DDX26B TRUE NA NA NA
2 CCDC83 NA 220047 89.3847 220047
```

3	MAST3	NA	23031	90.2213	23031
4	FLOT1	NA	10211	91.4342	10211
5	RPL11	NA	6135	82.9844	6135
6	ZDHHC20	NA	253832	89.6033	253832
7	LUC7L3	NA	51747	87.4169	51747
8	SNORD49A	NA	26800	109.0497	26800
9	CTSH	NA	1512	87.4712	1512
10	ACOT8	NA	10005	88.3713	10005

## 5.2 Mapping gene symbols to Ensembl gene ids

Now if you want Ensembl gene ids back:

```
> 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

	query	notfound	_id	X_score	
	<character>	<logical>	<character>	<numeric>	
1	DDX26B	TRUE	NA	NA	
2	CCDC83	NA	220047	89.3839	
3	MAST3	NA	23031	90.2213	
4	FLOT1	NA	10211	91.4342	
5	RPL11	NA	6135	82.9844	
6	ZDHHC20	NA	253832	89.6000	
7	LUC7L3	NA	51747	87.4210	
8	SNORD49A	NA	26800	109.0489	
9	CTSH	NA	1512	87.4756	
10	ACOT8	NA	10005	88.3665	

ensembl  
<list>

1	
2	ENSG00000150676
3	ENSG00000099308
4	ENSG00000230143, ENSG00000206379, ENSG00000232280
5	ENSG00000142676
6	ENSG00000180776
7	ENSG00000108848

```
8           ENSG00000277370
9           ENSG00000103811
10          ENSG00000101473

> out$ensembl[[4]]$gene

[1] "ENSG00000230143" "ENSG00000206379" "ENSG00000232280" "ENSG00000206480"
[5] "ENSG00000224740" "ENSG00000137312" "ENSG00000236271" "ENSG00000223654"
```

### 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`.

```
> 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

	query	notfound	_id	X_score	entrezgene
	<character>	<logical>	<character>	<numeric>	<character>
1	DDX26B	TRUE	NA	NA	NA
2	CCDC83	NA	220047	89.3806	220047
3	MAST3	NA	23031	90.2195	23031
4	FLOT1	NA	10211	91.4041	10211
5	RPL11	NA	6135	82.9532	6135
6	Gm10494	TRUE	NA	NA	NA

### 5.4 When input ids are not just symbols

```
> xli <- c('DDX26B',
+         'CCDC83',
+         'MAST3',
```



## MyGene.info R Client

```
+ 'FL0T1',
+ '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 89.3806 220047 Q8IWF9
3 MAST3 NA 23031 90.2195 23031 060307
4 FL0T1 NA 10211 91.4041 10211 075955
5 RPL11 NA 6135 82.9532 6135 P62913
6 Gm10494 TRUE NA NA NA NA
7 1007_s_at NA 100616237 13.6513 100616237 NA
8 1007_s_at NA 780 12.8613 780 Q08345
9 AK125780 NA 118142757 18.4859 118142757 NA
      uniprot.TrEMBL
      <list>
1
2 H0YDV3
3 V9GYV0
4 A2AB09, A2AB10, A2AB11, ...
5 A0A2R8Y447, Q5VVC8, Q5VVD0
6
7
8 A0A024RCJ0, A0A024RCL1, A0A024RCQ1, ...
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
  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  89.3839  220047  Q8IWF9
3   MAST3      NA      23031  90.2195  23031  060307
4   FL0T1      NA      10211  91.4041  10211  075955
5   RPL11      NA      6135  82.9532  6135  P62913
6   Gm10494    TRUE      NA      NA      NA      NA
7  1007_s_at    NA  100616237  13.6045  100616237  NA
8  1007_s_at    NA      780  12.8663  780  Q08345
9  AK125780    NA  118142757  18.4774  118142757  NA
      uniprot.TrEMBL
      <list>
1
2      H0YDV3
3      V9GYV0
4  A2AB09,A2AB10,A2AB11,...
5  A0A2R8Y447,Q5VVC8,Q5VVD0
6
7
8  A0A024RCJ0,A0A024RCL1,A0A024RCQ1,...
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

---

Wu C, MacLeod I, Su AI (2013) BioGPS and MyGene.info: organizing online, gene-centric information. *Nucl. Acids Res.* 41(D1): D561-D565. [help@mygene.info](mailto:help@mygene.info)