

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.

```
> 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
```

```
[1] "NP_001277159.1"

$strtranslation[[2]]$rna
[1] "NM_001290230.1"

$strtranslation[[3]]
$strtranslation[[3]]$protein
[1] "NP_001789.2"

$strtranslation[[3]]$rna
[1] "NM_001798.4"

$strtranslation[[4]]
$strtranslation[[4]]$protein
[1] "XP_011536034.1"

$strtranslation[[4]]$rna
[1] "XM_011537732.1"
```

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
```

	<code>_id</code>	<code>X_score</code>	<code>entrezgene</code>	<code>name</code>
	<code><character></code>	<code><numeric></code>	<code><integer></code>	<code><character></code>
1	1017	19.85752	1017	cyclin dependent kinase 2
2	1018	21.85669	1018	cyclin dependent kinase 3
3	1586	21.46224	1586	cytochrome P450 family 17 subfamily A member 1
	<code>symbol</code>	<code>taxid</code>	<code>query</code>	
	<code><character></code>	<code><integer></code>	<code><character></code>	
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.

```
> 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] 3.847771
```

```
$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.

```
> 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
```

	_id	X_score	entrezgene	name
	<character>	<numeric>	<integer>	<character>
1	5982	20.67265	5982	replication factor C subunit 2
2	3310	12.91964	3310	heat shock protein family A (Hsp70) member 6
3	7849	12.92045	7849	paired box 8
4	2978	10.33636	2978	guanylate cyclase activator 1A
5	7318	21.86343	7318	ubiquitin like modifier activating enzyme 7
6	100847079	21.85680	100847079	microRNA 5193

	symbol	taxid	query
	<character>	<integer>	<character>
1	RFC2	9606	1053_at
2	HSPA6	9606	117_at
3	PAX8	9606	121_at
4	GUCA1A	9606	1255_g_at
5	UBA7	9606	1294_at
6	MIR5193	9606	1294_at

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('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:

	seqnames	ranges	strand	tx_id	tx_name
	<Rle>	<IRanges>	<Rle>	<integer>	<character>
[1]	11	[85855100, 85920020]	+	1	NM_001286159
[2]	11	[85855100, 85920020]	+	2	NM_173556
[3]	19	[18097792, 18151689]	+	3	NM_015016
[4]	1	[23691778, 23696426]	+	4	NM_000975
[5]	1	[23691778, 23696426]	+	5	NM_001199802

```

...      ...      ...      ...
[13]      17 [50719564, 50752711]      + |      13      NM_016424
[14]      17 [16440035, 16440106]      + |      14      NR_002744
[15]      15 [78921749, 78945098]      - |      15      NM_001319137
[16]      15 [78921749, 78945098]      - |      16      NM_004390
[17]      20 [45841720, 45857409]      - |      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:

```

> 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

```

notfound      query      _id      X_score      entrezgene

```

	<logical>	<character>	<character>	<numeric>	<integer>
1	TRUE	DDX26B	NA	NA	NA
2	NA	CCDC83	220047	96.21670	220047
3	NA	MAST3	23031	96.56346	23031
4	NA	FLOT1	10211	98.29359	10211
5	NA	RPL11	6135	91.59504	6135
6	NA	ZDHHC20	253832	95.85616	253832
7	NA	LUC7L3	51747	93.68782	51747
8	NA	SNORD49A	26800	114.40928	26800
9	NA	CTSH	1512	96.56941	1512
10	NA	ACOT8	10005	93.95580	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	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
4	c("ENSG00000236271", "ENSG00000230143", "ENSG00000224740")
5	ENSG00000142676
6	ENSG00000180776
7	ENSG00000108848
8	ENSG00000277370

```

9                               ENSG00000103811
10                              ENSG00000101473
> out$ensembl[[4]]$gene
[1] "ENSG00000236271" "ENSG00000230143" "ENSG00000224740" "ENSG00000232280"
[5] "ENSG00000223654" "ENSG00000137312" "ENSG00000206379" "ENSG00000206480"

```

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

	notfound	query	_id	X_score	entrezgene
	<logical>	<character>	<character>	<numeric>	<integer>
1	TRUE	DDX26B	NA	NA	NA
2	NA	CCDC83	220047	96.21670	220047
3	NA	MAST3	23031	96.56346	23031
4	NA	FLOT1	10211	98.29359	10211
5	NA	RPL11	6135	91.59504	6135
6	TRUE	Gm10494	NA	NA	NA

5.4 When input ids are not just symbols

```

> 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>	<integer>	<character>
1	DDX26B	TRUE	NA	NA	NA	NA
2	CCDC83	NA	220047	77.268326	220047	Q8IWF9
3	MAST3	NA	23031	77.846650	23031	O60307
4	FLOT1	NA	10211	80.660700	10211	O75955
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 HOYDV3

3 V9GYVO

4 A2AB09, A2AB10, A2AB11, ...

5 Q5VVC8, Q5VVC9, Q5VVD0

6

7 A0A024RCJ0, A0A024RCL1, A0A024RCQ1, ...

8

9 A0A0A0MTF5, A6PVH5, 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

```
> 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	O60307
4	FLOT1	NA	10211	80.660700	10211	O75955
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 AOA024RCJ0,AOA024RCL1,AOA024RCQ1,...
8
9 AOA0A0MTF5,A6PVH5,B2R9P6
```

```
$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