MyGene.info R Client

Adam Mark, Ryan Thompson, Chunlei Wu

April 30, 2018

Contents

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] "A0A024RB10" "G3V5T9"     "E7ESI2"     "A0A024RB77" "B4DDL9"
[6] "G3V317"

> gene[[1]]$refseq

$genomic
[1] "NC_000012.12" "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_011537732.2"

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

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


$translation[[2]]
$translation[[2]]$protein
[1] "XP_011536034.1"

$translation[[2]]$rna
[1] "XM_011537732.2"


$translation[[3]]
$translation[[3]]$protein
[1] "NP_001277159.1"

$translation[[3]]$rna
[1] "NM_001290230.1"


$translation[[4]]
$translation[[4]]$protein
[1] "NP_001789.2"

$translation[[4]]$rna
[1] "NM_001798.4"
```

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_score entrezgene
      <character> <character> <numeric>  <integer>
1            1017        1017      1.55       1017
2            1018        1018      1.55       1018
3 ENSG00000148795        1586 17.969587       1586
                                        name      symbol     taxid
                                 <character> <character> <integer>
1                     cyclin dependent kinase 2        CDK2      9606
2                     cyclin dependent kinase 3        CDK3      9606
3 cytochrome P450 family 17 subfamily A member 1     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)

$max_score
[1] 445.6019

$took
[1] 10

$total
[1] 633

$hits
        _id   _score entrezgene                     name symbol taxid
1      1017 445.6019       1017 cyclin dependent kinase 2   CDK2  9606
2     12566 369.7370      12566 cyclin-dependent kinase 2   Cdk2 10090
3    362817 308.2000     362817 cyclin dependent kinase 2   Cdk2 10116
4 102398457 299.5158  102398457 cyclin dependent kinase 2   CDK2 89462
5 101544122 299.5158  101544122 cyclin dependent kinase 2   CDK2 42254
```

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

$max_score
[1] 6.261984

$took
[1] 15

$total
[1] 1

$hits
  _id   _score entrezgene                                          name symbol
1 780 6.261984        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
  <character> <character> <numeric>  <integer>
1     1053_at        5982 12.347883       5982
2      117_at        3310 12.069609       3310
3      121_at        7849  8.410426       7849
4   1255_g_at        2978 10.530668       2978
5     1294_at   100847079 12.643581  100847079
6     1294_at        7318 11.782185       7318
                                          name      symbol     taxid
                                   <character> <character> <integer>
1               replication factor C subunit 2        RFC2      9606
2 heat shock protein family A (Hsp70) member 6       HSPA6      9606
3                                paired box 8        PAX8      9606
```

```
4              guanylate cyclase activator 1A      GUCA1A      9606
5                                microRNA 5193     MIR5193      9606
6  ubiquitin like modifier activating enzyme 7        UBA7      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('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 23691778-23696835      + |         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
         query  notfound         _id   X_score entrezgene
   <character> <logical> <character> <numeric>  <integer>
1       DDX26B      TRUE          NA        NA         NA
2       CCDC83        NA      220047  88.78046     220047
```

```
3        MAST3        NA       23031  88.80842      23031
4        FLOT1        NA       10211  90.58115      10211
5        RPL11        NA        6135  83.34004       6135
6      ZDHHC20        NA      253832  89.45738     253832
7       LUC7L3        NA       51747 86.386856      51747
8     SNORD49A        NA       26800  106.4899      26800
9         CTSH        NA        1512 87.342766       1512
10       ACOT8        NA       10005 85.504456      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  88.80547
3        MAST3        NA       23031 88.803375
4        FLOT1        NA       10211 90.975204
5        RPL11        NA        6135  83.49154
6      ZDHHC20        NA      253832  89.44044
7       LUC7L3        NA       51747  86.38847
8     SNORD49A        NA       26800 106.49342
9         CTSH        NA        1512 87.351295
10       ACOT8        NA       10005 85.513306



1
2
3
4  list(gene = c("ENSG00000224740", "ENSG00000206379", "ENSG00000232280", "ENSG00000206480"
5
6
7
```

```
8
9
10
```

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

```
[1] "ENSG00000224740" "ENSG00000206379" "ENSG00000232280" "ENSG00000206480"
[5] "ENSG00000236271" "ENSG00000137312" "ENSG00000230143" "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>  <integer>
1      DDX26B      TRUE          NA        NA         NA
2      CCDC83        NA      220047 88.799255     220047
3       MAST3        NA       23031  88.80868      23031
4       FLOT1        NA       10211  90.57044      10211
5       RPL11        NA        6135 83.337746       6135
6     Gm10494      TRUE          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  88.80547     220047             Q8IWF9
3       MAST3        NA       23031 88.803375      23031             O60307
4       FLOT1        NA       10211 90.975204      10211             O75955
5       RPL11        NA        6135  83.49154       6135             P62913
6     Gm10494      TRUE          NA        NA         NA                 NA
7   1007_s_at        NA   100616237  9.134518  100616237                 NA
8   1007_s_at        NA         780  8.687636        780             Q08345
9    AK125780        NA        2978  9.943614       2978             P43080



1
2
3
4
5
6
7
8 c("A0A024RCL1", "A0A024RCQ1", "A0A024RCJ0", "A0A0A0MSX3", "Q96T62", "Q96T61", "A2ABM8",
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>  <integer>        <character>
1      DDX26B      TRUE          NA        NA         NA                 NA
2      CCDC83        NA      220047  88.79741     220047             Q8IWF9
3       MAST3        NA       23031  88.81223      23031             O60307
4       FLOT1        NA       10211  90.57044      10211             O75955
5       RPL11        NA        6135 83.331505       6135             P62913
6     Gm10494      TRUE          NA        NA         NA                 NA
7   1007_s_at        NA   100616237  9.211814  100616237                 NA
8   1007_s_at        NA         780  8.649236        780             Q08345
9    AK125780        NA        2978  9.966042       2978             P43080


1
2
3
4
5
6
7
8 c("A0A024RCL1", "A0A024RCQ1", "A0A024RCJ0", "A0A0A0MSX3", "Q96T62", "Q96T61", "A2ABM8",
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