MyGene.info R Client

Adam Mark, Ryan Thompson, Chunlei Wu

May 4, 2016

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] 37

> gene$name

[1] "cyclin-dependent kinase 2"

> gene$taxid

[1] 9606

> gene$uniprot

$TrEMBL
[1] "A0A024RB10" "A0A024RB77" "B4DDL9"     "E7ESI2"     "G3V317"     "G3V5T9"

$`Swiss-Prot`
[1] "P24941"

> gene$refseq

$rna
[1] "NM_001290230" "NM_001798"    "NM_052827"    "XM_011537732"

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

$genomic
[1] "NC_000012"    "NC_018923"    "NG_034014"    "NT_029419"    "NW_004929384"
```

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 6 columns
           query                                        name entrezgene      symbol
     <character>                                 <character>  <integer> <character>
1           1017                  cyclin-dependent kinase 2       1017        CDK2
```

```
2            1018                        cyclin-dependent kinase 3       1018        CDK3
3 ENSG00000148795 cytochrome P450 family 17 subfamily A member 1       1586     CYP17A1
          _id     taxid
  <character> <integer>
1        1017      9606
2        1018      9606
3        1586      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)
```

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

$max_score
[1] 370.4161

$took
[1] 5

$total
[1] 22
```

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

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

$max_score
[1] 0.5258086

$took
[1] 3
```

```
$total
[1] 1
```

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 6 columns
       symbol         _id       query entrezgene     taxid
  <character> <character> <character>  <integer> <integer>
1        RFC2        5982     1053_at       5982      9606
2       HSPA6        3310      117_at       3310      9606
3        PAX8        7849      121_at       7849      9606
4      GUCA1A        2978   1255_g_at       2978      9606
5     MIR5193   100847079     1294_at  100847079      9606
6        UBA7        7318     1294_at       7318      9606
                                      name
                                <character>
1              replication factor C subunit 2
2 heat shock protein family A (Hsp70) member 6
3                             paired box 8
4              guanylate cyclase activator 1A
5                           microRNA 5193
6  ubiquitin like modifier activating enzyme 7
```

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 15 ranges and 2 metadata columns:
       seqnames               ranges strand |     tx_id      tx_name
          <Rle>            <IRanges>  <Rle> | <integer>  <character>
   [1]       11 [85855100, 85920020]      + |         1    NM_173556
   [2]       11 [85855100, 85920020]      + |         2 NM_001286159
   [3]       19 [18097792, 18151689]      + |         3    NM_015016
   [4]        1 [23691778, 23696426]      + |         4 NM_001199802
   [5]        1 [23691778, 23696426]      + |         5    NM_000975
   ...      ...                  ...    ... .       ...          ...
  [11]       17 [50719564, 50752711]      + |        11    NM_016424
  [12]       17 [16440035, 16440106]      + |        12    NR_002744
  [13]       15 [78921749, 78945098]      - |        13 NM_001319137
  [14]       15 [78921749, 78945098]      - |        14    NM_004390
  [15]       20 [45841720, 45857409]      - |        15    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 4 columns
    notfound       query         _id entrezgene
   <logical> <character> <character>  <integer>
1       TRUE      DDX26B          NA         NA
2         NA      CCDC83      220047     220047
3         NA       MAST3       23031      23031
4         NA       FLOT1       10211      10211
5         NA       RPL11        6135       6135
6         NA     ZDHHC20      253832     253832
7         NA      LUC7L3       51747      51747
8         NA    SNORD49A       26800      26800
9         NA        CTSH        1512       1512
10        NA       ACOT8       10005      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    ensembl.gene  ensembl
   <character> <logical> <character>     <character>   <List>
1       DDX26B      TRUE          NA              NA ########
2       CCDC83        NA      220047 ENSG00000150676 ########
3        MAST3        NA       23031 ENSG00000099308 ########
4        FLOT1        NA       10211              NA ########
5        RPL11        NA        6135 ENSG00000142676 ########
6      ZDHHC20        NA      253832 ENSG00000180776 ########
7       LUC7L3        NA       51747 ENSG00000108848 ########
```

```
8     SNORD49A         NA       26800 ENSG00000277370 ########
9         CTSH         NA        1512 ENSG00000103811 ########
10       ACOT8         NA       10005 ENSG00000101473 ########
```

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

```
[1] NA
```

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 4 columns
   notfound       query entrezgene         _id
  <logical> <character>  <integer> <character>
1      TRUE      DDX26B         NA          NA
2        NA      CCDC83     220047      220047
3        NA       MAST3      23031       23031
4        NA       FLOT1      10211       10211
5        NA       RPL11       6135        6135
6      TRUE     Gm10494         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 6 columns
        query  notfound entrezgene         _id uniprot.Swiss-Prot uniprot.TrEMBL
  <character> <logical>  <integer> <character>        <character>         <List>
1      DDX26B      TRUE         NA          NA                 NA       ########
2      CCDC83        NA     220047      220047             Q8IWF9       ########
3       MAST3        NA      23031       23031             O60307       ########
4       FLOT1        NA      10211       10211             O75955       ########
5       RPL11        NA       6135        6135             P62913       ########
6     Gm10494      TRUE         NA          NA                 NA       ########
7   1007_s_at        NA  100616237   100616237                 NA       ########
8   1007_s_at        NA        780         780             Q08345       ########
9    AK125780        NA       2978        2978             P43080       ########
```

```
> 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 6 columns
        query  notfound uniprot.Swiss-Prot uniprot.TrEMBL         _id entrezgene
  <character> <logical>        <character>         <List> <character>  <integer>
1      DDX26B      TRUE                 NA       ########          NA         NA
2      CCDC83        NA             Q8IWF9       ########      220047     220047
3       MAST3        NA             O60307       ########       23031      23031
4       FLOT1        NA             O75955       ########       10211      10211
```

```
5       RPL11        NA              P62913       ########         6135         6135
6     Gm10494      TRUE                  NA       ########           NA           NA
7    1007_s_at        NA                  NA       ########    100616237    100616237
8    1007_s_at        NA              Q08345       ########          780          780
9     AK125780        NA              P43080       ########         2978         2978

$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