MyGene.info R Client

Adam Mark and Chunlei Wu

May 28, 2014

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   `mg.getgene`

- Use `mg.getgene`, the wrapper for GET query of "/gene/<geneid>" service, to return the gene object for the given geneid.

```
> mg.getgene("1017", fields=c("name","symbol","taxid","entrezgene"))

 chr [1:4] "name" "symbol" "taxid" "entrezgene"
$`__need_your_help`
[1] "We are working on a grant based on MyGene.info, please send us an email at cwu@scripps

$`_id`
[1] "1017"

$entrezgene
[1] 1017

$name
[1] "cyclin-dependent kinase 2"

$symbol
[1] "CDK2"

$taxid
[1] 9606
```

2.2   `mg.getgenes`

- Use `mg.getgenes`, the wrapper for POST query of "/gene" service, to return the list of gene objects for the given character vector of geneids.

```
> mg.getgenes(c("1017","1018","ENSG00000148795"), species="human")

 chr "human"
 chr "symbol,name,taxid,entrezgene"
[[1]]
[[1]]$name
[1] "cyclin-dependent kinase 2"

[[1]]$symbol
[1] "CDK2"

[[1]]$taxid
[1] 9606
```

```
[[1]]$entrezgene
[1] 1017

[[1]]$query
[1] "1017"

[[1]]$`_id`
[1] "1017"


[[2]]
[[2]]$name
[1] "cyclin-dependent kinase 3"

[[2]]$symbol
[1] "CDK3"

[[2]]$taxid
[1] 9606

[[2]]$entrezgene
[1] 1018

[[2]]$query
[1] "1018"

[[2]]$`_id`
[1] "1018"


[[3]]
[[3]]$name
[1] "cytochrome P450, family 17, subfamily A, polypeptide 1"

[[3]]$symbol
[1] "CYP17A1"

[[3]]$taxid
[1] 9606

[[3]]$entrezgene
[1] 1586

[[3]]$query
```

```
[1] "ENSG00000148795"

[[3]]$`_id`
[1] "1586"
```

3   Gene Query Service

3.1   `mg.query`

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

```
> mg.query("cdk2", size=5)

$`__need_your_help`
[1] "We are working on a grant based on MyGene.info, please send us an email at cwu@scripps

$hits
  entrezgene                                name    _score  symbol     _id taxid
1       1017          cyclin-dependent kinase 2 380.14430    CDK2    1017  9606
2      12566          cyclin-dependent kinase 2 339.55472    Cdk2   12566 10090
3     362817          cyclin dependent kinase 2 269.77985    Cdk2  362817 10116
4      52004          CDK2-associated protein 2  20.42308 Cdk2ap2   52004 10090
5     143384 CDK2-associated, cullin domain 1  19.47708  CACUL1  143384  9606

$max_score
[1] 380.1443

$took
[1] 7

$total
[1] 26

> mg.query("symbol:cdk2", species="human")

$`__need_your_help`
[1] "We are working on a grant based on MyGene.info, please send us an email at cwu@scripps

$hits
  entrezgene                      name  _score symbol  _id taxid
1       1017 cyclin-dependent kinase 2 88.02286   CDK2 1017  9606

$max_score
[1] 88.02286
```

```
$took
[1] 4


$total
[1] 1
```

3.2 `mg.querymany`

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

```
> mg.querymany(c("1017", "695"), scopes="entrezgene", species="human")

Finished
Pass returnall=TRUE to return lists of duplicate or missing query terms.
DataFrame with 2 rows and 6 columns
                                          name          symbol        taxid     entrezgene
                              <CharacterList> <CharacterList> <NumericList> <NumericList>
1              cyclin-dependent kinase 2                CDK2          9606           1017
2 Bruton agammaglobulinemia tyrosine kinase                 BTK          9606            695
          query            X_id
  <CharacterList> <CharacterList>
1            1017            1017
2             695             695


> mg.querymany(c("1017","695"), scopes="entrezgene", species="9606", returnall=TRUE)

Finished
$out
DataFrame with 2 rows and 6 columns
                                          name          symbol        taxid     entrezgene
                              <CharacterList> <CharacterList> <NumericList> <NumericList>
1              cyclin-dependent kinase 2                CDK2          9606           1017
2 Bruton agammaglobulinemia tyrosine kinase                 BTK          9606            695
          query            X_id
  <CharacterList> <CharacterList>
1            1017            1017
2             695             695


$dup
list()


$missing
list()
```

4 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.

4.1 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 `mg.querymany` method, telling it your input is `symbol`, and you want `entrezgene` (Entrez gene ids) back.

```
> mg.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 3 columns
```

	query	entrezgene	X_id
	\<CharacterList\>	\<NumericList\>	\<CharacterList\>
1	DDX26B	203522	203522
2	CCDC83	220047	220047
3	MAST3	23031	23031
4	FLOT1	10211	10211
5	RPL11	6135	6135
6	ZDHHC20	253832	253832
7	LUC7L3	51747	51747
8	SNORD49A	26800	26800
9	CTSH	1512	1512
10	ACOT8	10005	10005

4.2   Mapping gene symbols to Ensembl gene ids

Now if you want Ensembl gene ids back:

```
> out<-mg.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 3 columns
                                         ensembl.gene              X_id              query
                                      <CharacterList>   <CharacterList>   <CharacterList>
1                     ENSG00000165359,ENSG00000268630            203522             DDX26B
2                                     ENSG00000150676            220047             CCDC83
3                                     ENSG00000099308             23031              MAST3
4   ENSG00000137312,ENSG00000206379,ENSG00000206480,...           10211              FLOT1
5                                     ENSG00000142676              6135              RPL11
6                                     ENSG00000180776            253832            ZDHHC20
7                                     ENSG00000108848             51747             LUC7L3
8                     ENSG00000206956,ENSG00000175061             26800           SNORD49A
9                                     ENSG00000103811              1512               CTSH
10                                    ENSG00000101473             10005              ACOT8

> out$ensembl.gene[[4]]

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

4.3   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')
> mg.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 3 columns
          query     entrezgene               X_id
```

```
  <CharacterList> <NumericList> <CharacterList>
1           DDX26B        203522           203522
2           CCDC83        220047           220047
3            MAST3         23031            23031
4            FLOT1         10211            10211
5            RPL11          6135             6135
6          Gm10494
```

4.4   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<-mg.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 4 columns
            query    entrezgene  uniprot            X_id
  <CharacterList> <NumericList>   <List> <CharacterList>
1          DDX26B        203522 ########          203522
2          CCDC83        220047 ########          220047
3           MAST3         23031 ########           23031
4           FLOT1         10211 ########           10211
5           RPL11          6135 ########            6135
6         Gm10494               ########
7       1007_s_at     100616237 ########       100616237
8       1007_s_at           780 ########             780
9        AK125780          2978 ########            2978

> out$uniprot[[5]]
```

```
$`Swiss-Prot`
[1] "P62913"

$TrEMBL
[1] "Q5VVD0"
```

4.5   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, together with the mapping output, from the returned result:

```
> mg.querymany(xli, scopes=c("symbol","reporter","accession"),
+              fields=c("entrezgene","uniprot"), species="human", returnall=TRUE)

Finished
$out
DataFrame with 9 rows and 4 columns
              query     entrezgene   uniprot              X_id
    <CharacterList> <NumericList>    <List> <CharacterList>
1           DDX26B          203522 ########          203522
2           CCDC83          220047 ########          220047
3            MAST3           23031 ########           23031
4            FLOT1           10211 ########           10211
5            RPL11            6135 ########            6135
6           Gm10494                 ########
7         1007_s_at       100616237 ########       100616237
8         1007_s_at             780 ########             780
9         AK125780            2978 ########            2978

$dup
$dup$`1007_s_at`
[1] 2

$dup$`1007_s_at`
[1] 2

$dup$`1007_s_at`
[1] 2

$dup$`1007_s_at`
[1] 2

$dup$`1007_s_at`
```

```
[1] 2

$dup$`1007_s_at`
[1] 2


$missing
$missing[[1]]
[1] "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).

4.6   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.

5   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`