Quick Guide for BioIDMapper Package

Xiaoyong Sun[†‡*]

February 2, 2010

[†]Binformatics and Computational Biology Program, [‡]Department of Statistics
Iowa State University, Ames, Iowa 50010, USA

Contents

[*]sunx1@iastate.edu

1 Introduction

Many new databases aiming at genes and proteins are developed as more and more species are sequenced. It becomes tedious job about how to navigate among different data resources, map various IDs, and analyze separate biological knowledge. Current popular databases include Entrez Gene, UniProt, Gene Ontology, EMBL, OMIM, PubMed, KEGG, etc. Based on NCBI, UniProt, KEGG and other web services, BioIDMapper can facilitate mapping between different databases, integrate various ID systems and provide a full view from gene level, mRNA level and functional level regarding one specific ID. This package is based on NCBI and UniProt websites, utilizing two packages: XML and RCurl.

From version 2.0, users have options to utilize a graphical framework to manage ID mapping. The graphical user interace requires installation of GTK and additional R libraries. The detailed example is in section: Graphical user interface.

2 Installation

BioIDMapper needs to install the following programs and R packages:

1. Install RCurl
a. Install curl (version 7.14.0 or higher) http://curl.haxx.se
b. Install R package, RCurl: *install.packages("Rcurl")*

2. Install XML (Please see XML Installation notes if you have problems)
a. Install libxml2 (>= 2.6.3)
b. Install R package, XML: *install.packages("XML")*

3. Install RGtk2 (Please see RGtk2 Installation notes if you have problems)
a. Install GTK+ (>= 2.8.0)
b. Install R package, RGtk2: *install.packages("RGtk2")*

4. Install gWidgets
a. Install R package, gWidgets: *install.packages("gWidgets")*

5. Install gWidgetsRGtk2
a. Install R package, gWidgetsRGtk2: *install.packages("gWidgetsRGtk2")*

6. Install lattice
a. Install R package, lattice: *install.packages("lattice")*

3 BioID mapping table

In this package, the following 59 BioIDs can be translated to each other.

Note:

(1)"GI number" shows in both NCBI and UniPort databases, and it serves as bridge between two databases.
(2) "Biokey number" is the "currency" connecting functions.

Biokey number	**BioIDs**	**Sources**
1	GI number	NCBI
2	Pubmed id	NCBI
3	GEO id	NCBI
4	OMIM id	NCBI
5	SNP id	NCBI
6	UniGene cluster id	NCBI
7	UniSTS id	NCBI
8	Popset id	NCBI
9	MMDB id	NCBI
10	3D SDI id	NCBI
11	PSSM id	NCBI
12	TAXID	NCBI
13	Genome id	NCBI
14	PubChem Compound id	NCBI
15	PubChem Substance id	NCBI
16	PubChem BioAssay id	NCBI
17	NNNNNN	Boundary
18	GI number	UniProt
19	UniProtKB Accession	UniProt
20	UniProtKB id	UniProt
21	PIR Accession	UniProt
22	Enzyme Commission	UniProt
23	GO id	UniProt
24	Entrez Gene id	UniProt
25	EMBL id	UniProt
26	ENSEMBL id	UniProt
27	UniGene id	UniProt
28	TAIR id	UniProt
29	TIGR id	UniProt
30	KEGG id	UniProt

Biokey number	BioIDs	Sources
31	NCBI Taxon id	UniProt
32	OMIM id	UniProt
33	Ecogene id	UniProt
34	Flybase id	UniProt
35	GENEDB_SPOMBE id	UniProt
36	GERMONLINE id	UniProt
37	GRAMENE id	UniProt
38	HIV id	UniProt
39	IPI	UniProt
40	PDB id	UniProt
41	REBASE id	UniProt
42	Refseq Accession	UniProt
43	SGD id	UniProt
44	TRANSFAC id	UniProt
45	WORMPEP id	UniProt
46	UniRef100 id	UniProt
47	UniRef90 id	UniProt
48	UniRef50 id	UniProt
49	InterPro id	UniProt
50	Medline id	UniProt
51	PFAM id	UniProt
52	PIRSF id	UniProt
53	PRINTS id	UniProt
54	PRODOM id	UniProt
55	PROSITE id	UniProt
56	PMID	UniProt
57	SMART id	UniProt
58	TAXGRPID	UniProt
59	TIGRFAMs id	UniProt
60	TRANSFAC id	UniProt

4 Function description

This package includes one standard mapping table displayed by bio.type()function, and three features: mapping, linking and data analysis. Mapping feature is implemented by bio.convert() function; linking feature involves bio.link() function, and data analysis is done by 2 functions: bio.sum() and bio.select().

4.1 Retrieve mapping table

4.1.1 bio.type

```
bio.type <-function(type2id)
```

Show all Biokey numbers, and biological types that this package can handle. Presently 59 biological types are included in the package.

It takes one parameters:
a. type2id

Biokey number or BioIDs (biological types) from BioID mapping table;
If no argument is used, return BioID mapping table for all biokey numbers;
If argument is Biokey number from bio.type(), return the corresponding biological type;
If argument is biological type, return the corresponding Biokey number.

```
>bio.type()

      Biokey number BioIDs                  Sources
 [1,] "1"           "GI number"             "NCBI"
 [2,] "2"           "Pubmed id"             "NCBI"
 [3,] "3"           "GEO id"                "NCBI"
 [4,] "4"           "OMIM id"               "NCBI"
 [5,] "5"           "SNP id"                "NCBI"
 [6,] "6"           "UniGene cluster id"    "NCBI"
 [7,] "7"           "UniSTS id"             "NCBI"
 [8,] "8"           "Popset id"             "NCBI"
 [9,] "9"           "MMDB id"               "NCBI"
[10,] "10"          "3D SDI id"             "NCBI"
.....
```

```
> library(BioIDMapper)
> bio.type(5)

[1] "SNP id"

> bio.type("SNP id")

[1] 5
```

4.2 Mapping feature

4.2.1 bio.convert

```
bio.convert <-function(id_list, from, to)
```

This is the main interface for mapping ids. It takes three parameters:
a. id_list: id list you want to map;
b. from: Biokey number of source type; bio.type() will show all the Biokey numbers for biological types.
c. to: Biokey number of destination type; bio.type() will show all the Biokey numbers for biological types.

```
> data(glist)
> myMap <- bio.convert(glist, 1, 5)
```

4.3 Linking feature

4.3.1 bio.link

```
bio.link <-function(id, to)
```

This is the main interface for linking to external data sources. It will start web browser, and link that id to external data source. It takes two parameters:

a. id: id you want to link;
b. to: The corresponding Biokey number of external biology types you want to link; bio.type() will show all Biokey number for biological types.

Note:
"id" should match "to". For example, id: "27242148" is "SNP id"; to: 5 is Biokey number for "SNP id".

```
>bio.link("27242148", 5)
```

4.4 Data analysis feature

This feature is to analyze result from mapping function: bio.convert().

4.4.1 bio.sum

```
bio.sum <- function(result_matrix, start_idList, option)
```

Summary the result after mapping. It takes three parameters:
a. result_matrix: result matrix from bio.convert() function
b. start_idList: the orginial id list you want to map
c. option: a logical value. If TRUE, all summary results are returned. If FALSE, only basic summary is returned. default value is FALSE

```
> data(glist)
> myMap <- bio.convert(glist, 1, 5)
> bio.sum(myMap)
```

```
----------------------------------------------------------------------

MAPPING SCHEMA:
239 GI number  are mapped to  479 snp
----------------------------------------------------------------------

$`Summary for result`
             GI number snp
maping_Total       239 479
```

```
> mySum <- bio.sum(myMap, glist, FALSE)
```

```
----------------------------------------------------------------------

MAPPING SCHEMA:
239 GI number  are mapped to  479 snp

MAPPING PERCENTAGE:
100.00% GI number  are mapped to  snp
----------------------------------------------------------------------
```

4.4.2 bio.select

```
bio.select <- function(myid, result_matrix, colno)
```

Show mapping result for one id. It takes three parameters:
a. myid: id you are interested
b. result_matrix: result matrix from bio.convert() function
c. colno: the column number of result_matrix that contains id you are interested.

```
> data(glist)
> myMap <- bio.convert(glist, 1, 5)
> bio.select(myMap, 1, "41386735")

    GI number  snp
684 "41386735" "41386735"
```

4.5 Graphical user interface

From BioIDMapper version 2.0, users will have a graphical framework for managing ID mapping. The graphical user interface provides a convenient, flexiable environment to convert, subset, link and summarize target IDs.

To utilize this framework, users have to install GTK library and four R libraries: RGtk2, gWidgets, gWidgetsRGtk2 and lattice.

4.5.1 Additional installation for GUI

1. install GTK
For Windows, you can download the GTK Developer's Pack from
http://gladewin32.sourceforge.net/

For Unix, you can fetch the source files for the different libraries from
ftp://ftp.gtk.org/pub/gtk/v2.8/

2. install additional R libraries: RGtk2, gWidgets, gWidgetsRGtk2 and lattice

4.5.2 Features for GUI

In graphical user interface of BioIDMapper, there are 9 features, including open, preferences, convert, subset, connect, evaluate, plot, save, and quit. 1. open
Users can utilize this option to input IDs for conversion.

2. preferences
This option helps user to pick destination IDs for conversion. It should be chosen before the next step: convert.

3. convert
This function is similar to bio.convert. It utilizes the target IDs from "open",

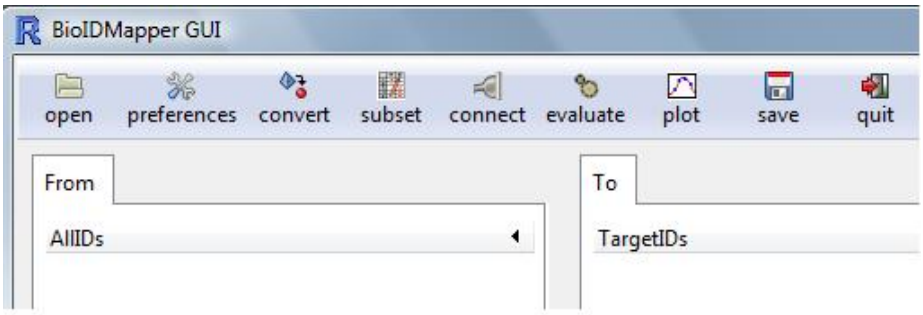

Figure 1: Menu for BioIDMapper

and the destination IDs from "preferences". Users can pick more than 1 destination IDs, and a process bar will mark the process of conversion.

4. subset
This function is similar to bio.select, which is utilized for subsetting the current data in the main panel.

5. connect
This function is similar to bio.link, which links to extern sources for certain ID.

6. evaluate
This function is similar to simple function of bio.sum, which summarize the mapping results.

7. plot
This function is similar to complex function of bio.sum, which summarize the results with detailed tables and graphs.

8. save
This function helps user to save either mapping results, subset results or summaries in txt format.

9. quit
Exit from the program.

5 Demonstration

To illustrate how to use this package, three examples are used, including mapping within NCBI, UniProt respectively, and mapping between NCBI and UniProt, to show how to translate different biological ids .

5.1 Examples for mapping within NCBI

Let's assume that you have 500 Genbank gi numbers, and you are interested in related snp ids.

First, you can find Biokey number from bio.type() function,

```
> bio.type()

      Biokey number BioIDs                  Sources
 [1,] "1"           "GI number"            "NCBI"
 [2,] "2"           "Pubmed id"            "NCBI"
 [3,] "3"           "GEO id"               "NCBI"
 [4,] "4"           "OMIM id"              "NCBI"
 [5,] "5"           "SNP id"               "NCBI"
 [6,] "6"           "UniGene cluster id"   "NCBI"
 [7,] "7"           "UniSTS id"            "NCBI"
 [8,] "8"           "Popset id"            "NCBI"
 [9,] "9"           "MMDB id"              "NCBI"
[10,] "10"          "3D SDI id"            "NCBI"
.....
```

Second, you can use the bio.convert() to map from one type of id to the other type of id:

```
> data(glist)
> myMap <- bio.convert(glist, 1, 5)
```

In addition, you can analyze the mapping result with bio.sum() function

```
>bio.sum(myMap)

          [,1]      [,2]
          "protein" "snp"
mappingNo "38"      "272"
```

Also, you can select the id you are interested from result:

```
>bio.select(myMap, 1, "200529")
```

Finally, you can check the detailed information about snp with id: "27242138" using bio.link(), and it will give you more detailed information from web browser.

```
>bio.link("27242138", 5)
```

5.2 Examples for mapping within UniProt

Assume that we have 10 UniProt Accession numbers, and let's find the related PDB ids.
First, you need to find out related id number in BioIDMapper package using bio.type function:

```
> bio.type()
.....
[19,] "19"       "UniProtKB Accession"  "UniProt"
[20,] "20"       "UniProtKB id"         "UniProt"
[21,] "21"       "PIR Accession"        "UniProt"
[22,] "22"       "Enzyme Commission"    "UniProt"
```

```
[23,] "23"       "Go id"                "UniProt"
[24,] "24"       "Entrez Gene id"       "UniProt"
[25,] "25"       "EMBL id"              "UniProt"
[26,] "26"       "ENSEMBL id"           "UniProt"
[27,] "27"       "UniGene id"           "UniProt"
[28,] "28"       "TAIR id"              "UniProt"
[29,] "29"       "TIGR id"              "UniProt"
[30,] "30"       "KEGG id"              "UniProt"
[31,] "31"       "NCBI Taxon id"        "UniProt"
[32,] "32"       "OMIM id"              "UniProt"
[33,] "33"       "Ecogene id"           "UniProt"
[34,] "34"       "Flybase id"           "UniProt"
[35,] "35"       "GENEDB_SPOMBE id"     "UniProt"
[36,] "36"       "GERMONLINE id"        "UniProt"
[37,] "37"       "GRAMENE id"           "UniProt"
[38,] "38"       "HIV id"               "UniProt"
[39,] "39"       "IPI"                  "UniProt"
[40,] "40"       "PDB id"               "UniProt"
.....

> bio.type("PDB id")

[1] 40
```

Second, you can use the bio.convert() to map from one type of id to the other type of id:

```
> data(ulist)
> myMap <- bio.convert(ulist, 19, 40)
```

You can also utilize bio.sum(), bio.select(), and bio.link() tools to check the related information.

5.3 Examples for mapping between NCBI and UniProt

You can do the mapping between NCBI and Uniprot exactly as before. Currently the bridge is "GI number" between the mapping of NCBI and Uniprot.

If you are interested in translating UniProt Accession Number to SNP id,

```
> data(ulist)
> myMap <- bio.convert(ulist, 19, 5)
```

5.4 Examples for mapping using Graphical User Interface

GUI in BioIDMapper provides a convenient managing platform for users. Let's see how to map hundreds of Genbank gi numbers (data: glist).

1. Start GUI with function: bio.gui()

2. Input Source IDs:
click OPEN button on the toolbar. There are two parameters:

a. Read data from R environment: the data should already be attached and available.
b. Source ID type: this ID type should match the input data.

For this example, we input "glist" as data and choose "GI number" as source ID type.

3. Choose Destination IDs:
click PREFERENCE button on the toolbar. Select interested IDs and move them from left column to the right column by clicking moving button in the middle.

For this data, let's choose "Pubmed ID" and "PDB" ID as destination IDs for our data set. Click "Go to next step" to finish configuration.

4. Map or cross-referencing
click CONVERT button on the toolbar, and it will begin to map from source ID type to destination ID type. You can track working progress with a progress bar on top of the main GUI.

5. Connect
click CONNECT button on the toolbar, choose biological ID type and interested multiple IDs. This function will connect to external data sources for selected IDs.

6. Subset, evaluate, plot
Subset, connect, evaluate and plot buttons help user to subset interesting ID groups, evaluate mapping results, and display final mapping graphically. The final mapping is plotted as dot plot with X: Source IDs, and Y: frequency of source ID involved in the final mapping.

6 Case study

To illustrate how to use these features, "chicken.db" data package from bioconductor.org is utilized for demonstration. The following R code enable user to map data from Entrez Gene id to UniProtKB Accession number (in the package, 24 represents Entrez Gene id, and 19 represents UniProt Accession Number).

```
# load "chicken" data package from Bioconductor
>library(chicken.db)
>xx <- as.list(chickenENTREZID)

# collect multiple entries for one probe and delete "NA" entries.
>uxx <- unlist(xx)
>myList <- unique(as.matrix(uxx[!is.na(uxx)]))

# map the Entrez Gene id to UniProt Accession Number
```

```
>library(BioIDMapper)
>result <- bio.convert(myList, 24, 19)
```

Also, the result can be linked directly to the related database website for more detailed information about that specific ID.

```
>bio.link(result[2,2],19)
```

In addition, the data analysis module offers services to summarize mapping results.

```
>bio.sum(result, myList, F)
```

Let's look at how to analyze the results in graphical user interface. Before we start GUI, we need to have data in the R environment.

```
# load "chicken" data package from Bioconductor
>library(chicken.db)
>xx <- as.list(chickenENTREZID)

# collect multiple entries for one probe and delete "NA" entries.
>uxx <- unlist(xx)
>myList <- unique(as.matrix(uxx[!is.na(uxx)]))

# change IDs to data.frame format
>myList <- data.frame(myList[1:100])
```

Now we can start GUI,

```
>bio.gui()
```

In OPEN dialog, input "myList" as data and "Entrez Gene ID" as biological ID type. Then you can proceed to manage CONVERT (mapping to destination IDs), SUBSET, LINK, EVALUATE and PLOT operations.