How to use the inparanoid packages

Marc Carlson

Introduction

The inparanoid packages are based upon the gene to gene orthology groupings as determined by the inparanoid algorithm. More details on this algorithm can be found at the inparanoid website: (`http://inparanoid.sbc.su.se/cgi-bin/index.cgi`). A nice brief description of the inparanoid method is given by the FAQ at their website: "The InParanoid program uses the pairwise similarity scores, calculated using NCBI-Blast, between two complete proteomes for constructing orthology groups. An orthology group is initially composed of two so-called seed orthologs that are found by two-way best hits between two proteomes. More sequences are added to the group if there are sequences in the two proteomes that are closer to the correpsonding seed ortholog than to any sequence in the other proteome. These members of an orthology group are called inparalogs. A confidence value is provided for each inparalog that shows how closely related it is to its seed ortholog."

The Inparanoid algorithm has been run on 35 species so far. We provide packages for five of these species, and within the packages for these five supported species, we provide the mappings between that species and all 35 of the Inparanoid species. The packages are named as follows:

hom.Hs.inp.db for human mappings to the other 35 species
hom.Mm.inp.db for mouse mappings to the other 35 species
hom.Rn.inp.db for rat mappings to the other 35 species
hom.Dm.inp.db for fly mappings to the other 35 species
hom.Sc.inp.db for yeast mappings to the other 35 species

This vignette will discuss the different information contained within these packages and how to use these packages to get information about the genes that are most likely the orthologous match for a particular gene.

Contents of the packages

Within each inparanoid package there is a small database that contains tables that map relationships between genes of one species and genes of another. For the database that is inside of the human package (hom.Hs.inp.db) the tables will be named according to the "other" species that they will map to. As an example within the context of the human package, the mus_musculus table will map relationships between humans and mouse. This particular table will be the

exact same table that will be inside of the mouse package (hom.Mm.inp.db), but in that context the table will have the name homo_sapiens to denote that in this other context is provides a mapping relationship between mouse and humans.

In addition to this database, there are also a series of prefabricated mappings that can be used to get the information we anticipate most users will want. Specifically, the reciprocal best matches or the inparanoid "seed pairs". These are the matches that make up most of the inparanoid tables, and which are intended to indicate the best matches between one gene and another in a species match-up. So if you have a human gene and you want to know what the likely equivalent of that gene in mouse is, you can use the appropriate mapping to quickly get that information. A quick look at the mappings that are available when you load the human package will show you these:

```
> library("hom.Hs.inp.db")
> ls("package:hom.Hs.inp.db")

  [1] "hom.Hs.inp"           "hom.Hs.inp.db"
  [3] "hom.Hs.inpACYPI"      "hom.Hs.inpAEDAE"
  [5] "hom.Hs.inpANOGA"      "hom.Hs.inpAPIME"
  [7] "hom.Hs.inpARATH"      "hom.Hs.inpASPFU"
  [9] "hom.Hs.inpBATDE"      "hom.Hs.inpBOMMO"
 [11] "hom.Hs.inpBOSTA"      "hom.Hs.inpBRAFL"
 [13] "hom.Hs.inpBRUMA"      "hom.Hs.inpCAEBR"
 [15] "hom.Hs.inpCAEBRE"     "hom.Hs.inpCAEEL"
 [17] "hom.Hs.inpCAEJA"      "hom.Hs.inpCAERE"
 [19] "hom.Hs.inpCANAL"      "hom.Hs.inpCANFA"
 [21] "hom.Hs.inpCANGL"      "hom.Hs.inpCAPSP"
 [23] "hom.Hs.inpCAVPO"      "hom.Hs.inpCHLRE"
 [25] "hom.Hs.inpCIOIN"      "hom.Hs.inpCIOSA"
 [27] "hom.Hs.inpCOCIM"      "hom.Hs.inpCOPCI"
 [29] "hom.Hs.inpCRYHO"      "hom.Hs.inpCRYNE"
 [31] "hom.Hs.inpCRYPA"      "hom.Hs.inpCULPI"
 [33] "hom.Hs.inpCYAME"      "hom.Hs.inpDANRE"
 [35] "hom.Hs.inpDAPPU"      "hom.Hs.inpDEBHA"
 [37] "hom.Hs.inpDICDI"      "hom.Hs.inpDROAN"
 [39] "hom.Hs.inpDROGR"      "hom.Hs.inpDROME"
 [41] "hom.Hs.inpDROMO"      "hom.Hs.inpDROPS"
 [43] "hom.Hs.inpDROVI"      "hom.Hs.inpDROWI"
 [45] "hom.Hs.inpENTHI"      "hom.Hs.inpEQUCA"
 [47] "hom.Hs.inpESCCO"      "hom.Hs.inpFUSGR"
 [49] "hom.Hs.inpGALGA"      "hom.Hs.inpGASAC"
 [51] "hom.Hs.inpGIALA"      "hom.Hs.inpHELRO"
 [53] "hom.Hs.inpIXOSC"      "hom.Hs.inpKLULA"
 [55] "hom.Hs.inpLEIMA"      "hom.Hs.inpLOTGI"
 [57] "hom.Hs.inpMACMU"      "hom.Hs.inpMAGGR"
 [59] "hom.Hs.inpMAPCOUNTS" "hom.Hs.inpMONBR"
```

```
 [61] "hom.Hs.inpMONDO"     "hom.Hs.inpMUSMU"
 [63] "hom.Hs.inpNASVI"     "hom.Hs.inpNEMVE"
 [65] "hom.Hs.inpNEUCR"     "hom.Hs.inpORGANISM"
 [67] "hom.Hs.inpORNAN"     "hom.Hs.inpORYLA"
 [69] "hom.Hs.inpORYSA"     "hom.Hs.inpOSTTA"
 [71] "hom.Hs.inpPANTR"     "hom.Hs.inpPEDPA"
 [73] "hom.Hs.inpPHYPA"     "hom.Hs.inpPHYRA"
 [75] "hom.Hs.inpPHYSO"     "hom.Hs.inpPLAFA"
 [77] "hom.Hs.inpPLAVI"     "hom.Hs.inpPONPY"
 [79] "hom.Hs.inpPOPTR"     "hom.Hs.inpPRIPA"
 [81] "hom.Hs.inpPUCGR"     "hom.Hs.inpRATNO"
 [83] "hom.Hs.inpRHIOR"     "hom.Hs.inpSACCE"
 [85] "hom.Hs.inpSCHMA"     "hom.Hs.inpSCHPO"
 [87] "hom.Hs.inpSCLSC"     "hom.Hs.inpSORBI"
 [89] "hom.Hs.inpSTANO"     "hom.Hs.inpSTRPU"
 [91] "hom.Hs.inpTAKRU"     "hom.Hs.inpTETNI"
 [93] "hom.Hs.inpTETTH"     "hom.Hs.inpTHAPS"
 [95] "hom.Hs.inpTHEAN"     "hom.Hs.inpTHEPA"
 [97] "hom.Hs.inpTRIAD"     "hom.Hs.inpTRICA"
 [99] "hom.Hs.inpTRIVA"     "hom.Hs.inpTRYCR"
[101] "hom.Hs.inpUSTMA"     "hom.Hs.inpXENTR"
[103] "hom.Hs.inpYARLI"     "hom.Hs.inp_dbInfo"
[105] "hom.Hs.inp_dbconn"   "hom.Hs.inp_dbfile"
[107] "hom.Hs.inp_dbschema"
```

What you will notice when you look at these is that these mappings all have the format of "hom.Hs.inpXXXXX". This indicates that they are mappings from the human package to their respective organisms give by a 5 character code. Because a simple 2 letter species abbreviation is too short to avoid redundancy when 35 species mappings are available, we have adopted the inparanoid style of species abbreviations for these mappings. This means that the 1st three letters designate the genus, and the 2nd two designate the species. And so for example, "MUSMU" is short for mus musculus, "DROME" is short for drosophila melanogaster etc. One thign to note about these mappings is that because they are maps, the data will have been formatted into a map form. In most cases this detail will be completely irrelevant since most seed pairs map 1:1. But some seed pairs map one to many or many to many. In these cases, it is important to remember that the map format will display the data as a 1:1 or 1:many relationship only. No data has been lost in this transformation, but it is a good idea to keep in mind that a tiny proportion of your keys may in fact map to the same lists of values when using maps like this.

You can of course look at the contents of a mapping in the usual way:

```
> as.list(hom.Hs.inpMUSMU[1:4])

$ENSP00000364178
[1] "ENSMUSP00000097561"
```

```
$ENSP00000356224
[1] "ENSMUSP00000051825"

$ENSP00000386259
[1] "ENSMUSP00000074773"

$ENSP00000271588
[1] "ENSMUSP00000074340"
```

When you do this you will probably notice that most of the seed mappings are 1:1. But you might also notice that the IDs might not be the kinds of IDs that you normally use.

The IDs in these mapping are the ones that were used by inparanoid in their initial comparisons, but it is probably not a serious problem if the inparanoid IDs are not the ones that you might have initially wanted. For the mainstream organisms such as mouse, human, rat, yeast, and fly, we also provide the needed data in the organism level packages so that you can map back from inparanoid to a more familiar set of IDs. Here is an example of how you can chain annotation packages together to start with a common gene symbol for human (MSX2), and then work over to the equivalent information in mouse (Msx2). An important caveat to this is that the organism level packages are entrez gene centric. This means that to extract meaningful information from them, it is always necessary to map through an entrez gene ID. In the example that follows we will show how you can take the human gene symbol MSX2, and then use one of the human organism mapping package to get its entrez gene ID, which can then be used to retrieve the ensemble protein ID that is needed to use the inparanoid data. We can then use the inparanoid mapping to get a homologous mouse ID to the ensemble protein ID which is returned as a jackson lab ID. Finally, the Jackson Lab ID can be mapped back to a mouse symbol by using the mouse organism mapping package.

```
> # load the organism annotation data for human
> library(org.Hs.eg.db)
> # get the entrex gene ID for gene symbol "MSX2"
> mget("MSX2", org.Hs.egSYMBOL2EG)

$MSX2
[1] "4488"

> # get the ensembl protein ID for the entrez gene ID "4488"
> mget("4488", org.Hs.egENSEMBLPROT)

$`4488`
[1] "ENSP00000239243" "ENSP00000427425"

> # use the inparanoid package to get the mouse gene that is considered
> # equivalent to ensembl protein ID "ENSP00000239243"
> mget("ENSP00000239243", hom.Hs.inpMUSMU)
```

```
$ENSP00000239243
[1] "ENSMUSP00000021922"
```

```
> # load the organism annotation data for mouse
> library(org.Mm.eg.db)
> # get the entrez gene ID for Jackson labs ID "MGI:97169"
> mget("MGI:97169", org.Mm.egMGI2EG)
```

```
$`MGI:97169`
[1] "17702"
```

```
> # finally get the gene symbol for entrez gene ID "17702" from mouse
> mget("17702", org.Mm.egSYMBOL)
```

```
$`17702`
[1] "Msx2"
```

The previous example demonstrates how the inparanoid mappings can give you a shortcut to genes that are likely to be homologs. In addition, this example shows how you can tap into a lot of desirable information about whatever gene mappings you find by using the inparanoid package in conjunction with the organism annotation packages and passing through an entrez gene ID.

0.1 Use *hom.Xx.Inp.db* to explore other paralogous relationships among organisms

As mentioned earlier, each database has a table that contains all the information needed to make each of the standard mappings provided. But there is other information contained in these tables as well such as the inparalogs and their scores. This information can be accessed by doing some simple queries using the DBI interface.

As an example consider the following seed pair mapping:

```
> mget("ENSP00000301011", hom.Hs.inpMUSMU)
```

```
$ENSP00000301011
[1] "ENSMUSP00000017622"
```

What if we wanted to know about other possible mappings that were not seed mappings? To do this we could use the DBI interface. But in order to do that we 1st have to look at what the underlying table looks like. In order to that we will use the following 3 helper functions to extablish a connection to the database, list the tables contained by the database and list the fields within a table of interest:

```
> # make a connection to the human database
> mycon <- hom.Hs.inp_dbconn()
> # make a list of all the tables that are available in the DB
> head(dbListTables(mycon))
```

```
[1] "Acyrthosiphon_pisum"   "Aedes_aegypti"
[3] "Anopheles_gambiae"     "Apis_mellifera"
[5] "Arabidopsis_thaliana"  "Aspergillus_fumigatus"
```

```
> # make a list of the columns in the table of interest
> dbListFields(mycon, "mus_musculus")
```

```
[1] "inp_id"      "clust_id"    "species"     "score"
[5] "seed_status"
```

At this point we know the name of the table for the mouse data must be mus_musculus, and we also know the names of the colums that this table contains thanks to `dbListFields`. So now we have all the information we need to start querying the database directly. Lets begin by probing the database with a simple query for all of the infomation about the ensembl protein ID "ENSP00000301011":

```
> #make a query that will let us see which clust_id we need
> sql <- "SELECT * FROM mus_musculus WHERE inp_id = 'ENSP00000301011';"
> #retrieve the data
> dataOut <- dbGetQuery(mycon, sql)
> dataOut
```

```
           inp_id clust_id species score seed_status
1 ENSP00000301011     2084   HOMSA     1        100%
```

From the results of this query, we can see that this ID belongs to cluster ID # 1731. Inparanoid groups genes that are considered to be related into groupings that all share a common cluster ID. So now we can adjust our query very slightly so that we pull out ALL of the information about that entire group from the mus_musculus table:

```
> #make a query that will let us see all the data that is affiliated with a clust id
> sql <- "SELECT * FROM mus_musculus WHERE clust_id = '1731';"
> #retrieve the data
> dataOut <- dbGetQuery(mycon, sql)
> dataOut
```

```
              inp_id clust_id species score seed_status
1    ENSP00000273857     1731   HOMSA     1        100%
2 ENSMUSP00000005352     1731   MUSMU     1        100%
```

And there you have it, the complete inparanoid data about cluster_id 1731, including the member of that grouping that we used to find the group "ENSP00000301011". Because the database in this example is the human inparanoid database, the mus_musculus table shows us information about both mouse and human genes, and which groups they share. If for example you wanted to see a table about mouse and zebrafish homologs, you would have to go look at the zebrafish table contained in the mouse package.

1 Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 3.1.1 Patched (2014-09-25 r66681), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.28.0, Biobase 2.26.0, BiocGenerics 0.12.0, DBI 0.3.1, GO.db 3.0.0, GenomeInfoDb 1.2.0, IRanges 2.0.0, RSQLite 0.11.4, Rgraphviz 2.10.0, S4Vectors 0.4.0, XML 3.98-1.1, annotate 1.44.0, graph 1.44.0, hgu95av2.db 3.0.0, hom.Hs.inp.db 3.0.0, org.Hs.eg.db 3.0.0, org.Mm.eg.db 3.0.0, xtable 1.7-4
- Loaded via a namespace (and not attached): tools 3.1.1