# Bioconductor: annotation databases

**Thomas Lumley**
**Ken Rice**

UW Biostatistics

*Seattle, June 2008*

# Outline

One goal of Bioconductor is to provide efficient access inside R to the genome databases that are vital to interpreting associations.

We will look at a few of these

- RSNPper
- biomaRt
- goTools and GOstats.

The reason to have an R interface to these databases is to be able to analyze annotation data for many SNPs or RNA transcripts.

# RSNPper

This is an interface to the `SNPper` service, part of the Children's Hospital Informatics Program (CHIP) at Boston Children's Hospital.

There are five basic functions

- `geneInfo()` information on a gene: location, name, coding strand, id in various databases
- `geneLayout()` information on exon locations
- `geneSNPs()` known SNPs in a gene
- `SNPinfo()` location, alleles, amino acid alleles, dbSNP id.
- `itemsInRange()` genes, SNPs, or counts of SNPs in segment of chromosome.

# Example: Angiotensinogen

```
> geneInfo("AGT")
SNPper Gene metadata:
There are  1 entries.
Basic information:
     GENEID NAME CHROM STRAND                       PRODUCT
GENE   2375  AGT  chr1      - angiotensinogen preproprotein
     NSNPS  TX.START    TX.END CODSEQ.START CODSEQ.END
GENE   215 228904892 228916564    228905510  228913219
     LOCUSLINK   OMIM  UNIGENE SWISSPROT   MRNAACC   PROTACC
GENE       183 106150 Hs.19383    P01019 NM_000029 NP_000020
     REFSEQACC
GENE      NULL
SNPper info:
     SOURCE           VERSION                GENOME DBSNP
[1,] "*RPCSERV-NAME*" "$Revision: 1.1.1.1 $" "hg18" "125"
```

[Note that the output also includes build numbers for dbSNP and the Human Genome. The build has changed since we last taught this course.]

# Example: Angiotensinogen

The ID number for angiotensinogen is 2375, which is the key for other queries

```
> geneLayout(2375)
               ID             NAME            CHROM
             " "            "AGT"           "chr1"
TRANSCRIPT.START  CODINGSEQ.START   TRANSCRIPT.END
     "228904892"      "228905510"      "228916564"
   CODINGSEQ.END      exon1.start        exon1.end
     "228913219"      "228904892"      "228905698"
     exon2.start        exon2.end      exon3.start
     "228906562"      "228906706"      "228908302"
       exon3.end      exon4.start        exon4.end
     "228908569"      "228912364"      "228913222"
     exon5.start        exon5.end
     "228916455"      "228916564"
```

# Example: Angiotensinogen

```
attr(,"toolInfo")
                SOURCE                VERSION
      "*RPCSERV-NAME*" "$Revision: 1.1.1.1 $"
                GENOME                  DBSNP
                "hg18"                  "125"

> agtsnps<-geneSNPs(2375)
> length(agtsnps)
[1] 217
>  agtsnps[[1]]
DBSNPID  "rs3789657"
TSCID      " "
CHROMOSOME "chr1"
POSITION  "228894922"
ALLELES   "A/G/T"
ROLE      "Downstream"
```

# Example: Angiotensinogen

```
RELPOS     "18297"
AMINO      " "
AMINOPOS   " "
HUGO       "AGT"
LOCUSLINK  "183"
NAME       "angiotensinogen preproprotein"
MRNA       "NM_000029"

>  itemsInRange("genes", "chr1", "228900000", "228910000")[[1]][-3]
  NAME  CHROM  NSNPS
 "AGT" "chr1"  "215"
>   itemsInRange("countsnps", "chr1", "228900000", "228910000")
 total exonic nonsyn
    49     14      2
```

For some SNPs there is additional information available from the `SNPinfo` function

# Example: Angiotensinogen

```
> b<-SNPinfo("372")
> b
SNPper SNP metadata:
     DBSNPID CHROMOSOME POSITION   ALLELES VALIDATED
[1,] "rs372" "chr13"    "31383542" "A/G"   "Y"
There are details on 4 populations
and 1 connections to gene features
> popDetails(b)
            PANEL     SIZE MAJOR.ALLELE MINOR.ALLELE   majorf    minorf
1 Yoruba-30-trios illumina            A            G 0.883333  0.116667
2        Japanese illumina            A            G 0.977273 0.0227273
3     Han_Chinese illumina            A            G 0.955556 0.0444444
4   CEPH-30-trios illumina            A            G 0.966667 0.0333333
```

# Example: finding chromosomes

We had a set 1524 SNPs, of which 409 did not have their chromosome listed.

I needed to know which SNPs were on the X chromosome, to estimate sex from DNA intensity and heterozygous X-chromosome loci, for QC.

```
> head(unknown)
[1] "UGT1A3-001449-0_B_R_1538822" "LIPC-002761-0_B_R_1538453"
[3] "CETP-001265-0_B_R_1538254"   "F8-165293-0_T_F_1538626"
[5] "CPB2-051208-0_B_F_1539402"   "VDRDIL-1355-0_T_F_1539404"
```

A hand-search would be easy but tedious, so we want an automated approach

# Example: finding chromosomes

First extract the gene names

```
genes <- sapply(unknown, function(snp) strsplit(snp,"-")[[1]][1])
ugenes <- unique(genes)
```

Now call SNPper

```
library(RSNPper)
chroms <- sapply(ugenes,
                  function(gene) geneInfo(gene,  useOldOutput=TRUE)["CHROM"]
```

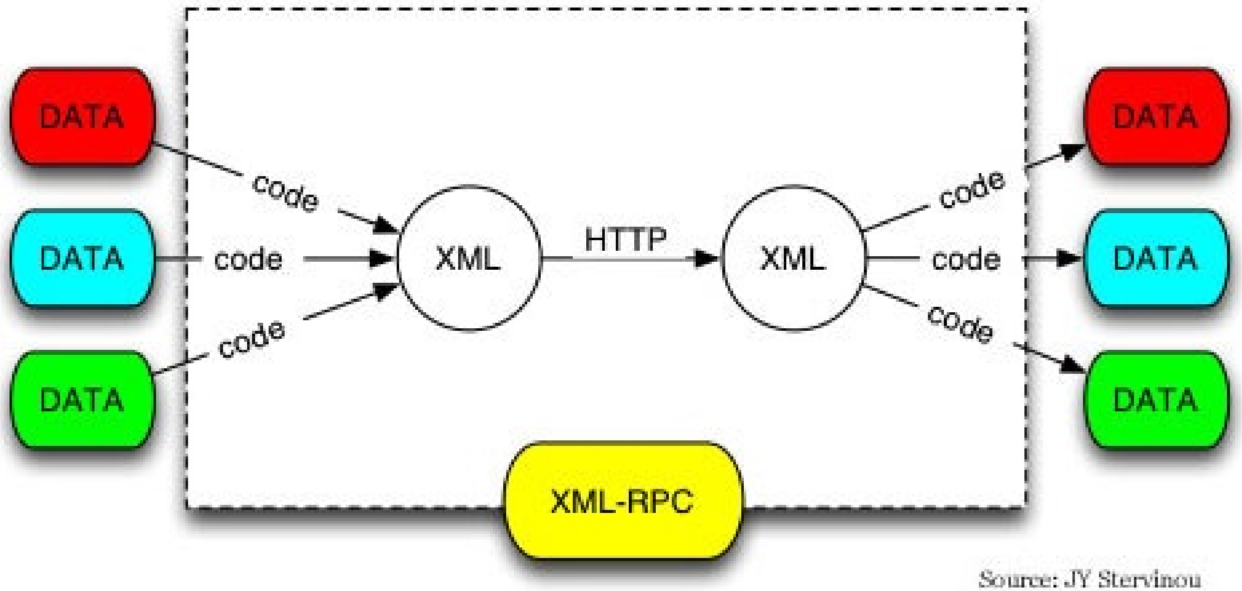
Works for all except one gene, where the name VDRDIL wasn't recognized

# Under the hood

SNPper responds to URLs like `http://snpper.chip.org/bio/rpcserv/dummy?cmd=geneinfo&name=CRP` with XML (structured text) format descriptions of the gene.

RSNPper downloads the information in the same way that **`read.table()`** downloads data from a web page, and then uses the XML package to process the information.

# Under the hood



Source: JY Stervinou

```
> useSNPper("geneinfo&","name=CRP")
 <SNPPER-RPC SOURCE="*RPCSERV-NAME*" VERSION="$Revision: 1.38 $"
  GENOME="hg17" DBSNP="123">
   <GENEINFO>
     <GENE ID="1440">
       <GENEID>1440</GENEID>
       <NAME>CRP</NAME>
       <CHROM>chr1</CHROM>
       <STRAND>-</STRAND>
```

# Under the hood

```
     <PRODUCT>C-reactive protein, pentraxin-related</PRODUCT>
     <TRANSCRIPT>
       <START>156495525</START>
       <END>156497437</END>
     </TRANSCRIPT>
     <CODINGSEQ>
       <START>156496388</START>
       <END>156497348</END>
     </CODINGSEQ>
     <ACCESSION>
       <MRNAACC>NM_000567</MRNAACC>
       <PROTACC>NP_000558</PROTACC>
       <REFSEQACC START="NIL" END="NIL"></REFSEQACC>
     </ACCESSION>
     <LINKS>
       <LOCUSLINK>1401</LOCUSLINK>
       <OMIM>123260</OMIM>
       <UNIGENE>Hs.76452</UNIGENE>
       <SWISSPROT>P02741</SWISSPROT>
     </LINKS>
     <NSNPS>101</NSNPS>
   </GENE>
 </GENEINFO>
</SNPPER-RPC>
```

# BioMart

BioMart (`www.biomart.org`)is a query-oriented data management system developed jointly by the European Bioinformatics Institute (EBI) and Cold Spring Harbor Laboratory (CSHL).

`biomaRt` is an R interface to BioMart systems, in particular to Ensembl (`www.ensembl.org`). Ensembl is a joint project between EMBL - European Bioinformatics Institute (EBI) and the Wellcome Trust Sanger Institute (WTSI) to develop a software system which produces and maintains automatic annotation on selected eukaryotic genomes.

# BioMart

We begin by choosing which BioMart to use

```
> library(biomaRt)
Loading required package: RCurl
> listMarts()
                      biomart                                 version
1                     ensembl               ENSEMBL 49 GENES (SANGER)
2    compara_mart_homology_49            ENSEMBL 49 HOMOLOGY (SANGER)
3 compara_mart_pairwise_ga_49 ENSEMBL 49 PAIRWISE ALIGNMENTS (SANGER)
4 compara_mart_multiple_ga_49 ENSEMBL 49 MULTIPLE ALIGNMENTS (SANGER)
5                         snp          ENSEMBL 49 VARIATION  (SANGER)
6            genomic_features    ENSEMBL 49 GENOMIC FEATURES (SANGER)
7                        vega                       VEGA 30  (SANGER)
8                         msd                     MSD PROTOTYPE (EBI)
9                     uniprot                 UNIPROT PROTOTYPE (EBI)
...
> ens <- useMart("ensembl")
```

# BioMart

We then choose a database to use

```
> listDatasets(ens)
                      dataset                             description  version
1      oanatinus_gene_ensembl  Ornithorhynchus anatinus genes (OANA5)    OANA5
2     gaculeatus_gene_ensembl  Gasterosteus aculeatus genes (BROADS1)  BROADS1
3     cporcellus_gene_ensembl       Cavia porcellus genes (GUINEAPIG) GUINEAPIG
4      lafricana_gene_ensembl      Loxodonta africana genes (BROADE1)  BROADE1
...
13      hsapiens_gene_ensembl             Homo sapiens genes (NCBI36)   NCBI36
...
35    ogarnettii_gene_ensembl    Otolemur garnettii genes (BUSHBABY1) BUSHBABY1
36 dmelanogaster_gene_ensembl Drosophila melanogaster genes (BDGP5.4)  BDGP5.4
37     oprinceps_gene_ensembl          Ochotona princeps genes (PIKA)     PIKA
38     mmusculus_gene_ensembl            Mus musculus genes (NCBIM37)  NCBIM37
39   cfamiliaris_gene_ensembl        Canis familiaris genes (BROADD2)  BROADD2
> ens <- useDataset("hsapiens_gene_ensembl",mart=ens)
```

# BioMart

The `getGene` function queries the database for gene information. It accepts many forms of gene identifier, eg Entrez, HUGO, Affy transcript

```
> getGene(id=1440, type="entrezgene", mart=ens)
  entrezgene hgnc_symbol
1       1440        CSF3

1 Granulocyte colony-stimulating factor precursor (G-CSF) (Pluripoietin) ...
  chromosome_name  band strand start_position end_position ensembl_gene_id
1              17 q21.1      1       35425140     35427592 ENSG00000108342

> getGene(id=c("AGT","AGTR1"), type="hgnc_symbol", mart=ens)
  hgnc_symbol hgnc_symbol
1         AGT         AGT
2       AGTR1       AGTR1

1 Angiotensinogen precursor (Serpin A8) [Contains: Angiotensin-1 ...
2   Type-1 angiotensin II receptor (AT1) (AT1AR) (AT1BR). ...
  chromosome_name  band strand start_position end_position ensembl_gene_id
1               1 q42.2     -1      228904897    228916564 ENSG00000135744
2               3   q24      1      149898355    149943478 ENSG00000144891
```

# BioMart

For non-human species we have been advised to used the more general getBM rather than getGene

```
fly <- useMart("ensembl", dataset="dmelanogaster_gene_ensembl")
g <- getBM(attributes=c("external_gene_id", "ensembl_gene_id", "chromosome_name",
 "start_position", "end_position"), filters="chromosome_name",
 values="4", mart=ens)
>  g[1:10,]
   external_gene_id ensembl_gene_id chromosome_name start_position end_position
1            ZNF595 ENSG00000197701               4          43227        78099
2            ZNF718 ENSG00000215383               4          43358       146491
3                   ENSG00000207643               4          55032        55124
4                   ENSG00000211553               4         120257       120351
5                   ENSG00000197135               4         122983       125183
6                   ENSG00000215382               4         149170       174241
7                   ENSG00000203599               4         160724       163527
8      Q49A33_HUMAN ENSG00000198155               4         196418       239769
9                   ENSG00000186777               4         254554       255716
10           ZNF141 ENSG00000131127               4         321622       359047
```

# BioMart

`getHomolog()` finds homologous genes in other species. For example, we can look up the mouse equivalents of a particular Affy transcript, or of the AGT gene.

```
> mouse = useMart("ensembl","mmusculus_gene_ensembl")
> homolog = getHomolog( id = "1939_at", to.type = "affy_mouse430_2",
     from.type = "affy_hg_u95av2", from.mart = ens, to.mart = mouse )
> homolog
       V1           V2
1 1939_at 1426538_a_at
2 1939_at 1427739_a_at
> homolog2 = getHomolog( id = "AGT", to.type = "affy_mouse430_2",
     from.type = "hgnc_symbol", from.mart = ens, to.mart = mouse )
> homolog2
   V1         V2
1 AGT 1423396_at
```

# BioMart

The `getSequence` function looks up DNA or protein sequences by chromosome position or gene identifiers

```
> agt<-getSequence(id="AGT",type="hgnc_symbol", seqType="peptide",mart=ens)
> agt

1 MRKRAPQSEMAPAGVSLRATILCLLAWAGLAAGDRVYIHPFHLVIHNESTCEQLAKANAGKPKDPTFIPAPIQAKTS
PVDEKALQDQLVLVAAKLDTEDKLRAAMVGMLANFLGFRIYGMHSELWGVVHGATVLSPTAVFGTLASLYLGALDHTAD
RLQAILGVPWKDKNCTSRLDAHKVLSALQAVQGLLVAQGRADSQAQLLLSTVVGVFTAPGLHLKQPFVQGLALYTPVVL
PRSLDFTELDVAAEKIDRFMQAVTGWKTGCSLMGASVDSTLAFNTYVHFQGKMKGFSLLAEPQEFWVDNSTSVSVPMLS
GMGTFQHWSDIQDNFSVTQVPFTESACLLLIQPHYASDLDKVEGLTFQQNSLNWMKKLSPRTIHLTMPQLVLQGSYDLQ
DLLAQAELPAILHTELNLQKLSNDRIRVGEVLNSIFFELEADEREPTESTQQLNKPEVLEVTLNRPFLFAVYDQSATAL
HFLGRVANPLSTA*
```

# Gene Ontology

The GO project has developed three structured controlled vocabularies (ontologies) that describe gene products in terms of their associated biological processes, cellular components and molecular functions in a species-independent manner. (`http://www.geneontology.org/`)

Each gene will be in nested categories of increasing specificity, and may be in more than one set of categories (structure is a 'directed acyclic graph').

Several Bioconductor packages allow queries by GO labels (eg `biomaRt`) and others provide further analyses based on GO categories.

# Gene Ontology

Using `biomaRt`, we can find the genes in a particular category, eg MAP kinase activity

```
>mapk<-getGene(id="GO:0004707",type="go",ens)> mapk
 > names(mapk)
[1] "go"                "hgnc_symbol"       "description"       "chromosome_name"
[5] "band"              "strand"            "start_position"    "end_position"
[9] "ensembl_gene_id"
> mapk[,1:2]
           go hgnc_symbol
1  GO:0004707
2  GO:0004707      CDC2L1
3  GO:0004707      CDC2L2
4  GO:0004707       MAPK4
5  GO:0004707       MAPK1
6  GO:0004707       MAPK6
7  GO:0004707      CDC2L5
8  GO:0004707      MAPK12
9  GO:0004707      MAPK11
10 GO:0004707       MAPK8
11 GO:0004707       MAPK7
12 GO:0004707         NLK
13 GO:0004707       MAPK3
```

# Gene Ontology

```
14 GO:0004707      MAPK10
15 GO:0004707      MAPK15
16 GO:0004707      MAPK14
17 GO:0004707      MAPK13
18 GO:0004707       CRKRS
19 GO:0004707        CDK2
20 GO:0004707       MAPK9
```

and conversely find GO categories for a particular gene

```
agtgo <- getGO("AGT",type="hgnc_symbol",ens)
> names(agtgo)
[1] "hgnc_symbol"     "go"              "go_description"  "evidence_code"
[5] "ensembl_gene_id"
> agtgo$go_description[1:8]
[1] "integral to membrane"
[2] "extracellular region"
[3] "cell-cell signaling"
[4] "extracellular space"
[5] "kidney development"
[6] "cell surface receptor linked signal transduction"
[7] "soluble fraction"
[8] "serine-type endopeptidase inhibitor activity"
```

# Gene Ontology

`goTools::ontoCompare` takes lists of genes and looks up their GO categories. It can report categories for a single list of genes or compare categories for multiple lists.

The `GOstats` package does a more sophisticated test for whether a list of genes is overrepresented in certain GO categories

This example is taken from the `goTools` package:

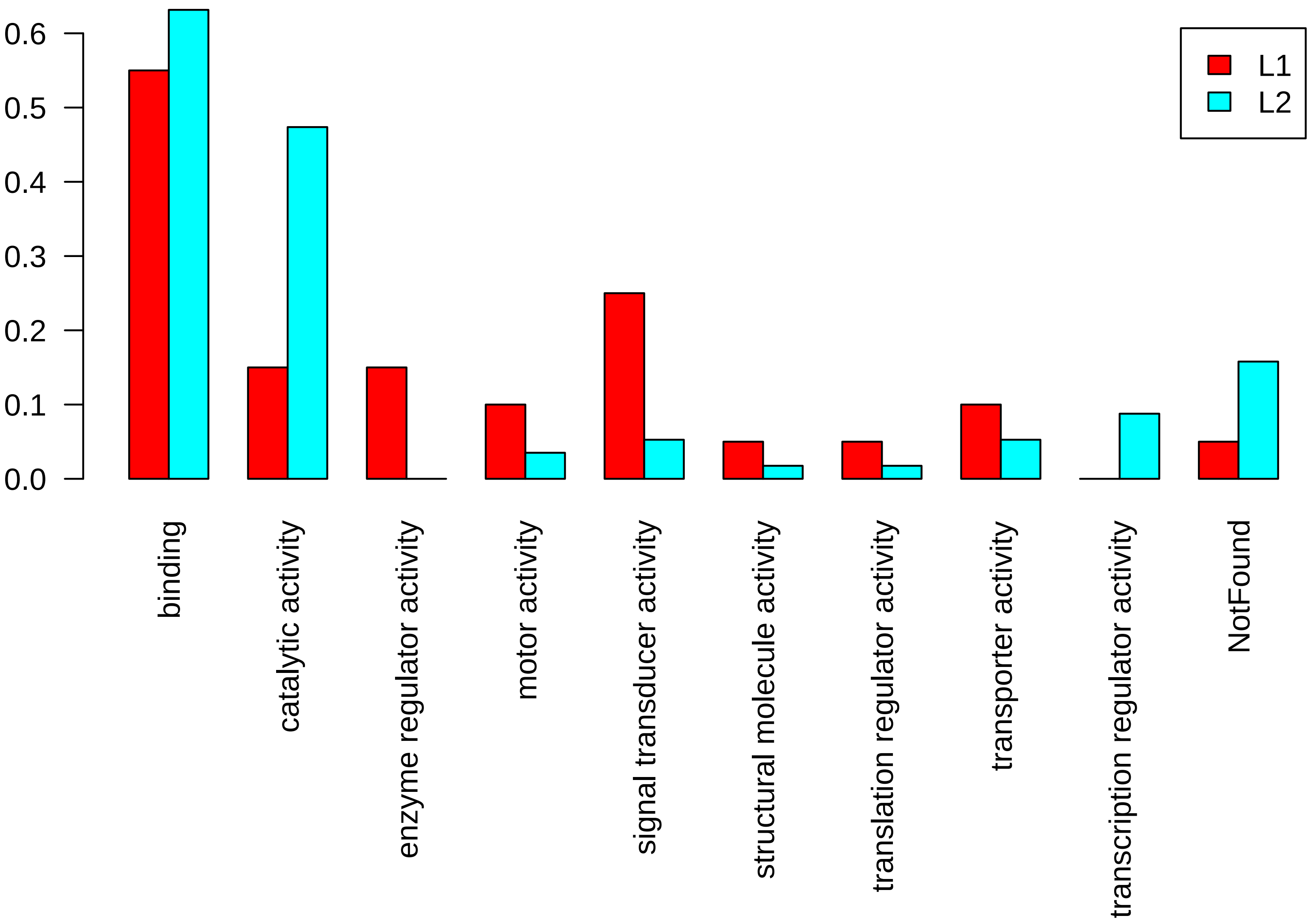
```
> library(goTools)
> data(ProbeID)
> str(operonlist)
List of 2
 $ L1: chr [1:30] "H200000481" "H200012124" "H200016088" "H200001913" ...
 $ L2: chr [1:85] "H200018146" "H200019124" "H200008091" "H200004721" ...

> ontoCompare(operonlist,probeType="operon",plot=TRUE,goType="MF")
   binding catalytic activity enzyme regulator activity
L1 0.55000            0.15000                      0.15
L2 0.63158            0.47368                      0.00
```

# Gene Ontology

```
   motor activity signal transducer activity
L1       0.100000                   0.250000
L2       0.035088                   0.052632
   structural molecule activity
L1                     0.050000
L2                     0.017544
   translation regulator activity transporter activity
L1                       0.050000             0.100000
L2                       0.017544             0.052632
   transcription regulator activity NotFound
L1                          0.00000  0.05000
L2                          0.08772  0.15789
```

# Gene Ontology


L1
L2
binding
catalytic activity
enzyme regulator activity
motor activity
signal transducer activity
structural molecule activity
translation regulator activity
transporter activity
transcription regulator activity
NotFound