

# annotationTools

**Alexandre Kuhn\***

\*[alexandre.m.kuhn@gmail.com](mailto:alexandre.m.kuhn@gmail.com)

**April 26, 2023**

## Contents

### 1 Introduction

---

*annotationTools* is an R package for the annotation of DNA microarray experiments and for the cross-platform and cross-species integration of gene expression profiles using plain text annotation and homology/orthology files [?]. Any flat annotation file can be used once it is loaded into R as a data.frame. Many functions in *annotationTools* are specialized look up tools working on data.frames and their use actually extend to any flat file database. Some functions are tailored to be used with Affymetrix annotation tables (i. e. HG-U133\_Plus\_2\_annot.csv for array 'HG-U133 Plus 2.0' for instance, available from <http://www.affymetrix.com>).

In this vignette, we provide a few practical examples on how to annotate microarray probes (Section ??) and how to retrieve orthologous genes and probe sets (in particular, how to match Affymetrix probe sets accross different species) using various sources of orthology information (namely NCBI's HomoloGene, see <http://www.ncbi.nlm.nih.gov/HomoloGene>, NCBI's 'Orthologs from Annotation pipeline' database and Affymetrix homology/orthology files) (Section ??). We also show how to build a mapping table of all the probe sets on a given microarray format and their orthologs on another format (Section ??). Finally, we demonstrate the use of such a table to perform cross-species analysis of gene expression regulation (Section ??).

### 2 Annotation

---

Assume that you want to annotate probe sets on Affymetrix array 'HG-U133 Plus 2.0'. The corresponding annotation file (HG-U133\_Plus\_2\_annot.csv) can be loaded into R with

```
> annotation_HGU133Plus2<-read.csv('HG-U133_Plus_2_annot.csv',  
+   colClasses='character',comment.char='#')
```

For demonstration purpose, a partial Affymetrix annotation file is provided with this package. We can load it with the following commands

```
> annotationFile<-system.file('extdata','HG-U133_Plus_2_annot_part.csv',  
+   package='annotationTools')  
> annotation_HGU133Plus2<-read.csv(annotationFile,colClasses='character',  
+   comment.char='#')
```

## annotationTools

The variable `myPS` contains two probe set IDs of interest

```
> myPS<-c('117_at','1007_s_at')
```

As an example, the gene symbols associated with these two probe sets can be retrieved from the annotation with the function `getGENESYMBOL`

```
> getGENESYMBOL(myPS,annotation_HGU133Plus2)

[[1]]
[1] "HSPA6"      "LOC652878"

[[2]]
[1] "DDR1"
```

Note that the output of `getGENESYMBOL` is a list. It contains two elements, one for each of the two elements in the input vector `myPS`. Note also that two gene symbols were found to be associated with the first probe set '117\_at' and that the first element in the output list thus is a vector of length 2 (containing gene symbols 'HSPA6' and 'LOC652878').

Further, you can for instance retrieve Gene Ontology (GO, <http://www.geneontology.org>) information, which is provided in the Affymetrix annotation file, with the function `getGENEONTOLOGY`

```
> getGENEONTOLOGY(myPS,annotation_HGU133Plus2)

[[1]]
[1] "6457 // protein folding // inferred from electronic annotation"
[2] "6986 // response to unfolded protein // traceable author statement"
[3] "6986 // response to unfolded protein // inferred from electronic annotation"

[[2]]
[1] "6468 // protein amino acid phosphorylation // inferred from electronic annotation"
[2] "7155 // cell adhesion // traceable author statement"
[3] "7169 // transmembrane receptor protein tyrosine kinase signaling pathway // inferred from electronic annotation"
[4] "7155 // cell adhesion // inferred from electronic annotation"
```

Again, the output list has two elements, one for each input probe set. Three and four gene ontology terms were found to be associated with the first and the second probe set respectively. Note that by default, `getGENEONTOLOGY` retrieves the 'biological process'–related GO annotation. To retrieve GO terms only and omit the rest (i. e. GO IDs and information on the GO annotation source), you can set the option `specifics` to 2

```
> getGENEONTOLOGY(myPS,annotation_HGU133Plus2,specifics=2)

[[1]]
[1] "protein folding"      "response to unfolded protein"
[3] "response to unfolded protein"

[[2]]
[1] "protein amino acid phosphorylation"
[2] "cell adhesion"
[3] "transmembrane receptor protein tyrosine kinase signaling pathway"
[4] "cell adhesion"
```

## annotationTools

Correspondingly, setting `specifics` to 1 (or 3) would result in retrieving GO IDs (respectively GO annotation source) only.

The list of all functions available through *annotationTools* can be obtained with

```
> ls(grep('annotationTools', search()))

[1] "compactList"          "getANNOTATION"
[3] "getGENEID"            "getGENEONTOLOGY"
[5] "getGENESYMBOL"        "getGENETITLE"
[7] "getHOMOLOG"           "getMULTIANNOTATION"
[9] "getOrthologousProbesets" "getPROBESET"
[11] "listToCharacterVector" "ps2ps"
```

`getANNOTATION` and `getMULTIANNOTATION` are general functions that work similarly to the specific annotation functions (eg, `getGENESYMBOL`) but that accept arbitrary annotation files. Note that these two functions can also be used to retrieve any information in Affymetrix annotation files that is not handled by a specific function in *annotationTools*. Here is the information currently provided by Affymetrix in their annotation files

```
> colnames(annotation_HGU133Plus2)

[1] "Probe.Set.ID"          "GeneChip.Array"
[3] "Species.Scientific.Name" "Annotation.Date"
[5] "Sequence.Type"         "Sequence.Source"
[7] "Transcript.ID.Array.Design." "Target.Description"
[9] "Representative.Public.ID" "Archival.UniGene.Cluster"
[11] "UniGene.ID"            "Genome.Version"
[13] "Alignments"            "Gene.Title"
[15] "Gene.Symbol"           "Chromosomal.Location"
[17] "Unigene.Cluster.Type"  "Ensembl"
[19] "Entrez.Gene"           "SwissProt"
[21] "EC"                    "OMIM"
[23] "RefSeq.Protein.ID"     "RefSeq.Transcript.ID"
[25] "FlyBase"               "AGI"
[27] "WormBase"              "MGI.Name"
[29] "RGD.Name"              "SGD.accession.number"
[31] "Gene.Ontology.Biological.Process" "Gene.Ontology.Cellular.Component"
[33] "Gene.Ontology.Molecular.Function" "Pathway"
[35] "Protein.Families"      "Protein.Domains"
[37] "InterPro"              "Trans.Membrane"
[39] "QTL"                   "Annotation.Description"
[41] "Annotation.Transcript.Cluster" "Transcript.Assignments"
[43] "Annotation.Notes"
```

Note finally that if an annotation function does not return anything for one of the probe set IDs in input, it can be useful to trace back the reason for the failure by setting `diagnose=TRUE`. Additional output will then allow you to determine if the probe set ID was not found in the annotation file, if it was present in the annotation file but did not have any annotation associated with it, or if the probe set ID was simply absent from the input (i. e. empty character string or NA). Please refer to the help (type `?getGENESYMBOL` at the R prompt for instance) for detailed explanations on the output diagnosis option.

## 3 Find orthologs

### 3.1 Using HomoloGene

We will first show how to use HomoloGene to retrieve orthologs. The complete flat file version of HomoloGene can be downloaded from <http://www.ncbi.nlm.nih.gov/HomoloGene>. A partial version of the database is provided with this package as an example.

```
> homologeneFile<-system.file('extdata','homologene_part.data',
+                             package='annotationTools')
> homologene<-read.delim(homologeneFile,header=FALSE)
```

Given two human genes of interest (gene IDs 5982 and 93587 for instance), their mouse orthologs can be looked up in the previously loaded homology file with `getHOMOLOG`, specifying the appropriate species ID for *Mus musculus* (i. e. 10090, see <http://www.ncbi.nlm.nih.gov/Taxonomy>)

```
> myGenes<-c(5982,93587)
> getHOMOLOG(myGenes,10090,homologene)

[[1]]
[1] 19718

[[2]]
[1] 108943
```

As already explained in Section ??, all functions in *annotationTools* output a list. Each element in the output list corresponds to an element in the input vector.

We can easily find orthologous probe sets on two different Affymetrix arrays by combining several functions in *annotationTools*. Assume that we are interested in probe set ID '1053\_at' (on human array 'HG-U133 Plus 2.0') and we would like to find orthologous probe sets on mouse array 'Mouse430 2.0' (i. e. probe sets associated with the mouse ortholog of the human gene probed by '1053\_at'): We first look up the human gene ID associated with probe set '1053\_at', then find the mouse orthologous gene ID, and finally retrieve the corresponding probe set IDs on the mouse array (using Affymetrix annotation file for array 'Mouse430 2.0')

```
> ps_human<- '1053_at'
> geneID_human<-getGENEID(ps_human,annotation_HGU133Plus2)
> geneID_mouse<-getHOMOLOG(geneID_human,10090,homologene)
> annotationFile<-system.file('extdata','Mouse430_2_annot_part.csv',
+                             package='annotationTools')
> annotation_Mouse4302<-read.csv(annotationFile,colClasses='character')
> geneID_mouse<-unlist(geneID_mouse)
> ps_mouse<-getPROBESET(geneID_mouse,annotation_Mouse4302)
> ps_mouse

[[1]]
[1] "1417503_at" "1457638_x_at" "1457669_x_at"
```

Note that `getHOMOLOG` can be tuned to other homology/orthology (flat file) databases. It can also be used to query with cluster IDs instead of gene IDs (setting the option `cluster=TRUE`). A cluster ID identifies a cluster of homologous/orthologous genes with a common identifier. Querying with a given cluster ID would result in retrieving all genes belonging to this cluster.

### 3.2 Using NCBI's 'Orthologs from Annotation pipeline' database

In 2014 NCBI introduced a new, dense source of orthologs for vertebrate species (see <https://www.ncbi.nlm.nih.gov/kis/info/how-are-orthologs-calculated/>). It is for instance used to provide links to orthologs in NCBI's Gene portal (<https://www.ncbi.nlm.nih.gov/gene>).

The complete flat file version of this database can be retrieved from [https://ftp.ncbi.nlm.nih.gov/gene/DATA/gene\\_orthologs.gz](https://ftp.ncbi.nlm.nih.gov/gene/DATA/gene_orthologs.gz). A partial version is provided with this package

```
> gene_orthologsFile<-system.file('extdata','gene_orthologs_part.data',
+   package='annotationTools')
> gene_orthologs<-read.delim(gene_orthologsFile,header=TRUE)
```

We can easily mine 'gene\_orthologs' by setting the `tableType` argument in `getHOMOLOG` to 'gene\_orthologs' (instead of the 'homologene' default)

```
> getHOMOLOG(myGenes,10090,gene_orthologs,tableType="gene_orthologs")

[[1]]
[1] 19718

[[2]]
[1] 108943
```

### 3.3 Using Affymetrix's ortholog tables

For each array format, Affymetrix provides a table listing homologous/orthologous probe sets on their other arrays (i. e. HG-U133\_Plus\_2\_ortholog.csv for array 'HG-U133 Plus 2.0' for instance, available from <http://www.affymetrix.com>). The `cluster=TRUE` option can in particular be used to mine these tables for orthologous probe sets on a particular array. We provide a partial Affymetrix homology/orthology file for array 'HG-U133 Plus 2.0' as an example

```
> affyOrthologFile<-system.file('extdata','HG-U133-Plus-2_ortholog_part.csv',
+   package='annotationTools')
> orthologs_HGU133Plus2<-read.csv(affyOrthologFile,colClasses='character')
```

Given the human probe set '1053\_at' (on array 'HG-U133 Plus 2.0'), we can for instance retrieve the orthologous probe sets proposed by Affymetrix for array 'Mouse 430 2.0' by specifying

```
> getHOMOLOG('1053_at','Mouse430_2',orthologs_HGU133Plus2,
+   cluster=TRUE,clusterCol=1,speciesCol=4,idCol=3)

[[1]]
[1] "1457669_X_AT" "1417503_AT" "1457638_X_AT"
```

Note that in this example, the retrieved probe sets exactly match those previously found using HomoloGene.

## 4 Build tables of orthologous probe sets

Here, we provide example code to map all probe sets on Affymetrix array 'HG-U133 Plus 2.0' to their orthologous probe sets on array 'Mouse430 2.0'. Any such mapping between Affymetrix GeneChips (and based on HomoloGene information) can be achieved in a straight-forward manner using the function `ps2ps`

```
> orthoTable<-ps2ps(annotation_HGU133Plus2,annotation_Mouse4302,homologene,10090)
```

However, we will now present the full code (encapsulated in `ps2ps`) as an example on how to combine various elementary annotation functions. We use HomoloGene to find the mouse orthologs of the human genes (alternatively, we could use NCBI's `gene_orthologs` file by setting the argument `tableType` in `ps2ps` to `'gene_orthologs'`). If a human probe set is annotated with several gene IDs, we retrieve the mouse orthologs corresponding to all of these genes. We therefore use the function `compactList` to obtain final lists of orthologous genes and orthologous probe sets of the same length as the original vector of human probe sets. Note that we assume in this example that the full annotation for 'HG-U133 Plus 2.0' has been downloaded from Affymetrix and has been saved in the file 'HG-U133\_Plus\_2\_annot.csv'.

```
> annotation_HGU133Plus2<-read.csv('HG-U133_Plus_2_annot.csv',
+   colClasses='character',comment.char='#')
> annotation_Mouse4302<-read.csv('Mouse430_2_annot.csv',
+   colClasses='character',comment.char='#')
> homologene<-read.delim('homologene.data',header=F)
> target_species<-10090
> ps_HGU133Plus2<-annotation_HGU133Plus2[,1]
> gid_HGU133Plus2<-getGENEID(ps_HGU133Plus2,annotation_HGU133Plus2)
> length_gid_HGU133Plus2<-sapply(gid_HGU133Plus2,function(x) {length(x)})
> gid_Mouse4302<-getHOMOLOG(unlist(gid_HGU133Plus2),target_species,homologene)
> length_gid_Mouse4302<-sapply(gid_Mouse4302,function(x) {length(x)})
> ps_Mouse4302<-getPROBESET(unlist(gid_Mouse4302),annotation_Mouse4302)
> ps_Mouse4302_1<-compactList(ps_Mouse4302,length_gid_Mouse4302)
> ps_Mouse4302_2<-compactList(ps_Mouse4302_1,length_gid_HGU133Plus2)
> gid_Mouse4302_1<-compactList(gid_Mouse4302,length_gid_HGU133Plus2)
```

We now remove duplicate (and absent) orthologous gene IDs and probe sets.

```
> ps_Mouse4302_2<-lapply(ps_Mouse4302_2,function(x) {unique(x)})
> ps_Mouse4302_2<-lapply(ps_Mouse4302_2,function(x) {
+   if (length(x)>1) na.omit(x) else x})
> gid_Mouse4302_1<-lapply(gid_Mouse4302_1,function(x) {unique(x)})
> gid_Mouse4302_1<-lapply(gid_Mouse4302_1,function(x) {
+   if (length(x)>1) na.omit(x) else x})
```

Finally, we can write a table listing orthologous probe sets between arrays 'HG-U133 Plus 2.0' and 'Mouse 430 2.0'.

```
> orthoTable<-cbind(ps_HGU133Plus2,listToCharacterVector(gid_HGU133Plus2,sep=' '),
+   listToCharacterVector(gid_Mouse4302_1,sep=' '),
+   listToCharacterVector(ps_Mouse4302_2,sep=' '))
> colnames(orthoTable)<-c('ps_HGU133Plus2','gid_HGU133Plus2',
+   'gid_Mouse4302','ps_Mouse4302')
```

```
> write.table(orthoTable, file='HGU133Plus2_Mouse4302.txt', sep='\t',
+             col.names=T, row.names=F)
```

As already stated, the function `ps2ps` uses the above procedure and allows to easily map orthologous probe sets between any pair of Affymetrix microarrays. The code above can thus be replaced by the following call

```
> orthoTable<-ps2ps(annotation_HGU133Plus2, annotation_Mouse4302, homologene, 10090)
> write.table(orthoTable, file='HGU133Plus2_Mouse4302.txt', sep='\t',
+             col.names=T, row.names=F)
```

## 5 An example cross-species analysis of gene expression: Transcriptional dysregulation in Huntington's disease patients and in a genetic mouse model

Huntington's disease is a neurological disorder caused by a trinucleotide (CAG) expansion in the *HD* gene. One way of generating mouse models of HD is to expand the short CAG repeat of the mouse Huntington's disease gene homolog (*Hdh*) with CAG repeats within the length range found in HD patients.

Animal models of HD have allowed to show that mutant protein expression results in transcriptional dysregulation of many genes [?]. More recently, many mRNA changes have been detected in the brain of HD patients too [?]. How do these changes compare with those identified in mouse models [?]?

Here we will consider the CHL2 mouse model of HD [?] and investigate if top mRNA changes detected in striatal samples of these mutant mice parallel those measured in the corresponding brain region of HD patients. Thereby, we aim at assessing the faithfulness of the animal model with regard to transcriptional dysregulations. To perform this comparison, we need to find orthologous probe sets in the two microarray formats used in the aforementioned studies, namely MG-U74Av2 for the mouse and HG-U133A for humans. The corresponding table of orthologous probe sets (which thus maps probe sets from MG-U74Av2 to HG-U133A) has been generated using `ps2ps` (see Section ??) and we will now show how to use it to answer our question. We first show how to make use of the function `getOrthologousProbesets.R` to perform such a cross-species analysis. We then present a second, step-by-step solution that is not based on the use of `getOrthologousProbesets.R` and that is aimed at exposing the process in some more details.

Tables of differentially expressed genes in the CHL2 mouse model and in HD patients are available from the repository HDBase (<http://hdbase.org/cgi-bin/welcome.cgi>). Partial versions of these lists and of the ortholog mapping table, as well as a partial annotation for microarray HG-U133A are provided with this package as dataset `orthologs_example` (which contains `table_mouse`, `table_human`, `ortho` and `annot_HGU133A`). In a real application, they would need to be loaded individually by the analyst into R as `data.frame` objects.

```
> data(orthologs_example)
```

We start by selecting the top 8 (arbitrary) mouse probe sets from the (ordered) list of differentially expressed genes in CHL2 (`table_mouse`)

```
> selection<-1:8
> ps_mouse<-table_mouse$Probe.Set.ID[selection]
> table_mouse[selection,]
```

	Name	M	t	P.Value	X	Probe.Set.ID
1	92254_at	-0.688	-7.066242	0.000153	NA	92254_at
2	101631_at	0.777	6.017669	0.000429	NA	101631_at
3	93273_at	-0.532	-5.585021	0.000681	NA	93273_at
4	96497_s_at	-1.030	-5.445946	0.000795	NA	96497_s_at
5	99511_at	-1.020	-5.371624	0.000864	NA	99511_at
6	102711_at	-0.764	-5.209632	0.001039	NA	102711_at
7	100006_at	-0.421	-4.988125	0.001344	NA	100006_at
8	92555_at	0.464	4.939043	0.001424	NA	92555_at

	Title	Gene.Symbol
1	myosin Vb	Myo5b
2	SRY-box containing gene 11	Sox11
3	synuclein, alpha	Snca
4	myelin transcription factor 1-like	Myt1l
5	protein kinase C, beta	Prkcb
6	regulator of G-protein signaling 14	Rgs14
7	cadherin 11	Cdh11
8	transmembrane 4 superfamily member 6	Tm4sf6

The human probe sets orthologous to the top 8 mouse probe sets and listed in `table_human` can be easily retrieved using `getOrthologousProbesets.R`

```
> orthops<-getOrthologousProbesets(ps_mouse,table_human,ortho)
> orthops[[1]]

[[1]]
[1] NA

[[2]]
[1] "204913_s_at" "204914_s_at" "204915_s_at"

[[3]]
[1] "204466_s_at" "204467_s_at" "207827_x_at" "211546_x_at" "215811_at"

[[4]]
[1] "210016_at" "216672_s_at"

[[5]]
[1] "207957_s_at" "209685_s_at"

[[6]]
[1] "204280_at" "211021_s_at" "38290_at"

[[7]]
[1] "207172_s_at" "207173_x_at"

[[8]]
[1] "209108_at" "209109_s_at"
```



Thus, no orthologous probe set has been found for the top 1 mouse probe set and each of the remaining mouse probe sets has at least 2 orthologous probe sets in `table_human`. `getOrthologousProbesets.R` can also be used to directly select among these multiple orthologous probe sets. Here is an example: out of the multiple orthologous probe sets found for each of the 8 mouse probe sets, we will select the one detecting the largest absolute log fold change (log fold changes are given in the second column of `table_human`). We first need to create a modified version of `table_human` containing the absolute log fold changes to pass it to `getOrthologousProbesets.R`

```
> table_human_absM<-data.frame(I(table_human[,1]),I(abs(table_human[,2])))
```

Now, we simply need to invoke `getOrthologousProbesets.R` and specify a selection function (here, 'max') that will be applied to the second column of `table_human_absM` (i. e. the absolute log fold changes) resulting in the selection of the corresponding probe set

```
> orthops_maxAbsM<-getOrthologousProbesets(ps_mouse,table_human_absM,
+      ortho,max,forceProbesetSelection=TRUE)
> orthops_maxAbsM[[1]]

[[1]]
[1] NA

[[2]]
[1] "204915_s_at"

[[3]]
[1] "204466_s_at"

[[4]]
[1] "210016_at"

[[5]]
[1] "207957_s_at"

[[6]]
[1] "204280_at"

[[7]]
[1] "207173_x_at"

[[8]]
[1] "209108_at"
```

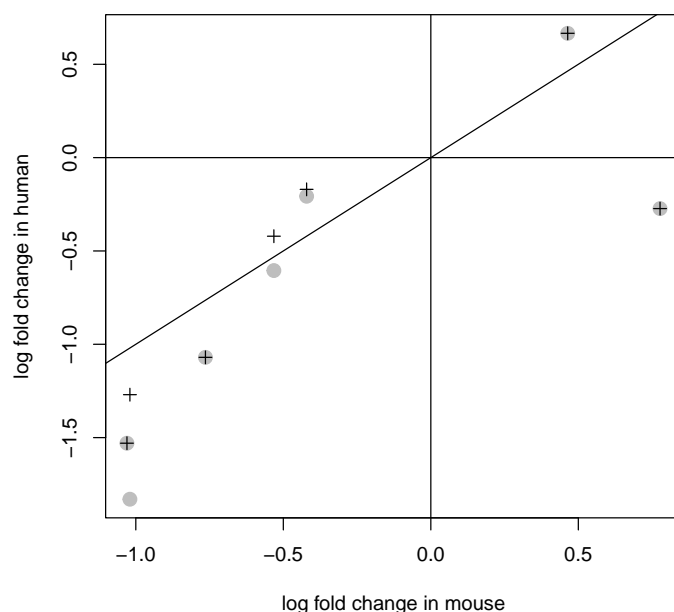
The original log fold changes associated with the selected orthologous probe sets can be found by looking them up in `table_human`

```
> orthops_maxAbsM_ind<-match(unlist(orthops_maxAbsM[[1]]),table_human[,1])
> table_human[orthops_maxAbsM_ind,2]

[1] NA -0.273 -0.605 -1.530 -1.830 -1.070 -0.207 0.666
```

Gray dots on Figure ?? represent these log fold changes against those measured by the corresponding mouse probe sets in the CHL2 model. To illustrate the importance of the probe set selection method in case of multiple orthologous probe sets, we also display the

log fold changes measured by orthologous human probe sets selected using a largest absolute t-statistic criterion, which correspond to the selection of the most significant probe sets (black crosses). The reported orthologous gene regulations are identical with both probe set selection methods for 4 probe sets out of 7.



**Figure 1:** Gene expression regulation measured by the top 8 mouse probe sets in the CHL2 mouse model of Huntington's disease and their orthologous regulations in human patients. In case of multiple orthologous probe sets found in the human data, we selected the probe set detecting the largest absolute log fold change in gene expression (gray dots) or the largest absolute t-statistic, that is the smallest p-value (black crosses).

For demonstration purpose and to illustrate some of the internal steps involved in cross-species comparison, we now perform a similar analysis but without relying on `getOrthologousProbeSets.R`. Again, we start with the top 8 mouse probe sets (i. e. the 8 probesets showing the most robust differential gene expression in the CHL2 model) and we use the mapping table (stored in the variable `ortho`) to look up their orthologous probe sets

```
> ps_human<-ortho[match(ps_mouse,ortho[,1]),4]
> ps_human

[1] NA
[2] "204913_s_at,204914_s_at,204915_s_at"
[3] "204466_s_at,204467_s_at,207827_x_at,211546_x_at,215811_at"
[4] "210016_at,216672_s_at"
[5] "207957_s_at,209685_s_at"
[6] "204280_at,211021_s_at,38290_at"
[7] "207172_s_at,207173_x_at"
[8] "209108_at,209109_s_at"
```

Each mouse probe set can have between zero and many orthologous probe sets on the HG-U133A array (the top mouse probe set has none for instance). Let us split expressions containing multiple orthologous probe sets and retrieve their corresponding gene symbols

```
> ps_human<-lapply(ps_human,function(x){strsplit(x,',')[[1]]})
> length_ps_human<-sapply(ps_human,length)
> gs_human<-lapply(ps_human,function(x){
+   listToCharacterVector(getGENESYMBOL(x,annot_HGU133A))})
```

Warning: one or more empty probe sets in input

```
> gs_human
[[1]]
[1] NA

[[2]]
[1] "SOX11" "SOX11" "SOX11"

[[3]]
[1] "SNCA" "SNCA" "SNCA" "SNCA" "SNCA"

[[4]]
[1] "MYT1L" "MYT1L"

[[5]]
[1] "PRKCB1" "PRKCB1"

[[6]]
[1] "RGS14" "RGS14" "RGS14"

[[7]]
[1] "CDH11" "CDH11"

[[8]]
[1] "TSPAN6" "TSPAN6"
```

This suggests that we indeed identified orthologous probe sets correctly (compare with gene symbols of top mouse probe sets). Note that multiple orthologous human probe sets corresponding to a given mouse probe set all report expression of the same gene (which does not need to be always the case, a given mouse gene could be matched to several different orthologs in human). We can identify selected MG-U74Av2 probe sets with no orthologous probe sets on HG-U133A (which will be useful in the remaining)

```
> existing_ps_human<-!is.na(ps_human)
```

Finally, we can look up gene expression regulations (log fold changes) measured by the top mouse probe sets with at least one orthologous human probe set

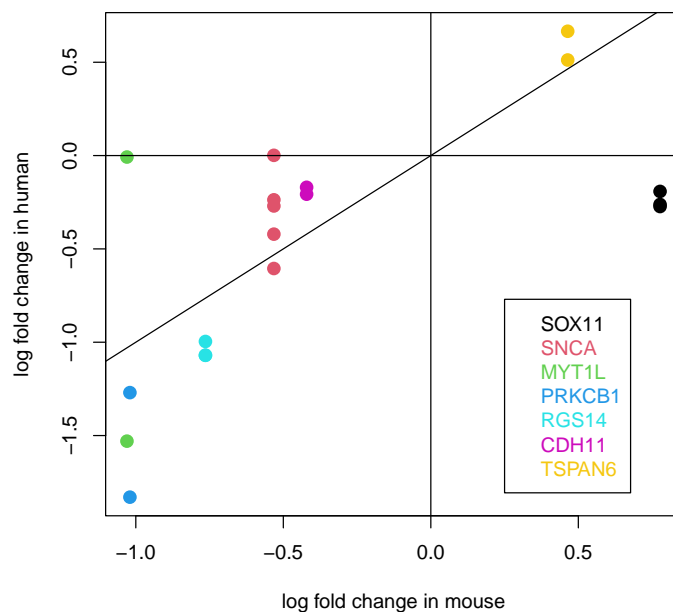
```
> LFC_mouse<-table_mouse$M[rep(match(ps_mouse[existing_ps_human],
+   table_mouse$Probe.Set.ID),length_ps_human[existing_ps_human])]
```

and the regulations measured by their orthologous probe sets in humans (using the list of differentially expressed genes in HD patients, stored in `table_human`)

```
> LFC_human<-table_human$log2FC.HD.caudate.grade.0.2.vs.controls[
+   match(unlist(ps_human[existing_ps_human]),table_human$Probe.Set.ID)]
```

The selected mouse mRNA changes and the orthologous human mRNA changes can now be displayed as a scatterplot using the following code (see Figure ??)

```
> plot(LFC_mouse,LFC_human,col=rep(1:length(ps_human[existing_ps_human]),
+   length_ps_human[existing_ps_human]),pch=16,cex=1.5,
+   xlab='log fold change in mouse',ylab='log fold change in human')
> abline(h=0)
> abline(v=0)
> abline(0,1)
> legend(x=0.25,y=-0.77,legend=lapply(gs_human[existing_ps_human],
+   function(x){paste(unique(x),sep=',')}),
+   text.col=1:length(ps_human[existing_ps_human]))
```



**Figure 2:** Gene expression regulation measured by the top 8 mouse probe sets in the CHL2 mouse model of Huntington's disease and their orthologous regulations in human patients. Seven mouse probe sets out of eight could be matched to one or more orthologous probe sets on the human array. Multiple orthologous human probe sets (i. e. corresponding to a given mouse probe set) are identified by the same color. Their corresponding human gene symbols are indicated in the legend.

Note that we used a single color to identify multiple human orthologous probe sets corresponding to a given mouse probe set. We observe that human probe sets with identical annotation sometimes report regulations very consistently (e.g. the 3 probe sets for *RGS14*) but not always (e.g. the 5 probe sets for *SNCA*). An extreme case of inconsistency is provided by *MYT1L*, with a probe set measuring a log fold change of -1.5 and the other 0. Such a case might require checking both probe set sequences against their targeted transcript in order to make a decision on which probe set to take into account. Finally, we see that while some

orthologs seem to be regulated in the same manner in HD patients compared to the CHL2 mouse model (eg *PRKCB1* and *RGS14*), some others show opposite direction regulation or absence of regulation in HD patients compared to the mouse model (*SOX11*).

In conclusion, such a systematic comparison might improve our understanding of the pathogenic molecular mechanisms leading to disease in animal models and in humans. It might also be useful to assess how different animal models recapitulate transcriptional dysregulations detected in humans for instance. Finally, cross-species analysis of transcription profiles might allow to pinpoint interesting, conserved set of genes of particular relevance in a given pathology.

## 6 Session Information

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

```
R version 4.3.0 RC (2023-04-18 r84287 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows Server 2022 x64 (build 20348)

Matrix products: default

locale:
[1] LC_COLLATE=C
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] stats      graphics  grDevices  utils      datasets  methods   base

other attached packages:
[1] annotationTools_1.75.0

loaded via a namespace (and not attached):
 [1] BiocManager_1.30.20 compiler_4.3.0    fastmap_1.1.1
 [4] BiocStyle_2.29.0   cli_3.6.1        htmltools_0.5.5
 [7] tools_4.3.0        yaml_2.3.7       Biobase_2.61.0
[10] rmarkdown_2.21     knitr_1.42       BiocGenerics_0.47.0
[13] digest_0.6.31      xfun_0.39        rlang_1.1.0
[16] evaluate_0.20
```