ExprX-vignette

Ming-an Sun, Yejun Wang, Linlin Zou, Yanhua Li, Guoqiang Zhu, Todd Macfarlan

2020-09-10

1 Introduction

ExprX is an R package to streamline interspecies differential expression analysis. Taking TPM or FPKM/RPKM files for samples from different species as input, it provides functions to handle all the necessary steps, including data loading, ortholog matching, normalization, differential analysis and visualization.

ExprX is freely available at: https://github.com/mingansun/ExprX.

2 How to install ExprX

ExprX can be installed using the **install_git** function from devtools package. However, it depends on several other R packages, which should be installed first.

To install ExprX and its dependencies:

```
# install several dependent packages from Bioconductor using BiocManager
install.packages("BiocManager")
BiocManager::install(c("biomaRt", "edgeR", "RankProd"))

# install ExprX from GitHub using devtools
install.packages("devtools")
devtools::install_git("https://github.com/mingansun/ExprX")
```

To load the ExprX package:

```
library(ExprX)
#> Welcome to use ExprX!
```

For more details about how to install and use **ExprX**, please refer to the website: https://github.com/mingansun/ExprX

3 Apply ExprX for interspecies differential expression analysis

Using RNA-Seq data for human and mouse brain, this vignette demonstrates how to detect differentially expressed genes between species using ExprX. All the involved steps are described below.

3.1 Generate ExprX object by integrating interspecies expression data

By parsing the meta table (as a data frame or CSV file) which contains information about expression data files (usually contain TPM, FPKM or RPKM values) for different species, the function **make_ExprX_dataset** can read these data files to create an object which contains the expression levels of the replicates of different species. The created ExprX object can also contain additional data such as orthologue pairs, normalized expression etc, and will be used by most of the subsequent analysis.

To use **make_ExprX_dataset** to read CSV file with meta data for expression data files and compared species (ie. human and mouse) to create an ExprX object:

```
# meta table file
hs2mm.meta_file <- paste0(path.package("ExprX"), "/extdata/brain_metatable.csv")

# make ExprX object from meta table
hs2mm.data  <- make_ExprX_dataset(
  hs2mm.meta_file,
  data_dir = paste0(path.package("ExprX"), "/extdata")
  )
#> x is detected as a file name. Read to a data frame:
#> Species  FullName    AbbrName    IdColumn    ExprColumn  ExprType    RepIndex    File
#> human    Homo Sapiens    hsapiens    1   6   tpm 1   human_brain_1.genes.results
#> human    Homo Sapiens    hsapiens    1   6   tpm 2   human_brain_2.genes.results
#> human    Homo Sapiens    hsapiens    1   6   tpm 3   human_brain_3.genes.results
#> mouse    mmusculus   mmusculus   1   6   tpm 1   mouse_brain_1.genes.results
#> mouse    mmusculus   mmusculus   1   6   tpm 2   mouse_brain_2.genes.results
#> mouse    mmusculus   mmusculus   1   6   tpm 3   mouse_brain_3.genes.results
#>
#> Read gene IDs for each species ...
#> human: 1042 genes.
#> mouse: 1037 genes.
#>
#> Loading expression data for each species ...
#> human: 1042 genes from 3 files.
#> mouse: 1037 genes from 3 files.
```

3.2 Determine the 1-to-1 orthologs among compared species

The 1-to-1 orthologs among species are constructed based on the homolog annotations from ENSEMBL database. Thus, only species available in in ENSEMBL database (about 200 as checked on 2020-4-7) can be used for analysis.

To use the **list_species** function to get the information (eg. species name, abbreviation) for all the species supported by ExprX:

```
sp.lst <- list_species()
head(sp.lst)
#>         Dataset        Species                Version
#> 1   acalliptera Eastern happy             fAstCal1.2
#> 2 acarolinensis  Anole lizard              AnoCar2.0
#> 3  acchrysaetos  Golden eagle             bAquChr1.2
#> 4  acitrinellus Midas cichlid               Midas_v5
#> 5  amelanoleuca         Panda                ailMel1
#> 6    amexicanus Mexican tetra Astyanax_mexicanus-2.0
```

Match and save 1-to-1 orthologs among species

The **ortholog_match** function invokes **biomaRt** package to retrieve homolog annotation from ENSEMBL database, then matches 1-to-1 orthologs by reciprocal comparison. This step usually takes a few minutes - depending on the network speed). To speed up, the obtained ortholog data can be stored on hard disk with **saveRDS** for later use.

```
# Match 1:1 orthologs between human and mouse
hs2mm.orth <- ortholog_match("human", "mouse")
#> Ensembl site unresponsive, trying useast mirror
#> Ensembl site unresponsive, trying uswest mirror
#> Get species name abbreviation to be used by biomaRt ...
#> human => hsapiens
#> mouse => mmusculus
#> Attempting web service request:
#> http://www.ensembl.org:80/biomart/martservice?
  type=version&requestid=biomaRt&mart=ENSEMBL_MART_ENSEMBL
#>     V1
#> 1 0.7
#> BioMartServer running BioMart version: 0.7
#> Mart virtual schema: default
#> Mart host: http://www.ensembl.org:80/biomart/martservice
#>
#> Connect to biomaRt for each species ...
#> human => hsapiens
#> mouse => mmusculus
#>
#> Determine the homolog table to be retrieved ...
#> human => mouse:  mmusculus_homolog_ensembl_gene
#> mouse => human:  hsapiens_homolog_ensembl_gene
#>
#> Retrieving homolog annotation for human ...
#>
#> Retrieving homolog annotation for mouse ...
#>
#> Number of genes with only 1 match in other species:
#> human:   18330
#> mouse:   18873
#>
#> Number of 1-to-1 ortholog pairs before filtering:    16758
#> Number of 1-to-1 ortholog pairs after  filtering:    16758

# Save the ortholog results on hard disk for later use
saveRDS(hs2mm.orth, "hs2mm.orth.rds")
```

Load previously saved ortholog matching results from hard disk

Below shows how to used **readRDS** to load ortholog matching result that is previously saved on hard disk. Alternatively, the ortholog matching result can be generated with **ortholog_match** as demonstrated above.

The ortholog matching result includes information for each involved species, such as GeneID, GeneName, Chrom, GeneType and so on.

```
# Load ortholog result with readRDS
hs2mm.orth <- readRDS(paste0(path.package("ExprX"), "/data/hs2mm.orth.rds"))

# View the structure of hs2mm.orth
str(hs2mm.orth)
#> List of 2
#>  $ human:'data.frame':   16536 obs. of  4 variables:
#>   ..$ GeneID  : chr [1:16536] "ENSG00000198695" "ENSG00000198712" "ENSG00000198727"
  "ENSG00000198763" ...
#>   ..$ GeneName: chr [1:16536] "MT-ND6" "MT-CO2" "MT-CYB" "MT-ND2" ...
#>   ..$ Chrom   : chr [1:16536] "MT" "MT" "MT" "MT" ...
#>   ..$ GeneType: chr [1:16536] "protein_coding" "protein_coding" "protein_coding"
  "protein_coding" ...
#>  $ mouse:'data.frame':   16536 obs. of  4 variables:
#>   ..$ GeneID  : chr [1:16536] "ENSMUSG00000064368" "ENSMUSG00000064354" "ENSMUSG00000064370"
  "ENSMUSG00000064345" ...
#>   ..$ GeneName: chr [1:16536] "mt-Nd6" "mt-Co2" "mt-Cytb" "mt-Nd2" ...
#>   ..$ Chrom   : chr [1:16536] "MT" "MT" "MT" "MT" ...
#>   ..$ GeneType: chr [1:16536] "protein_coding" "protein_coding" "protein_coding"
  "protein_coding" ...
#>  - attr(*, "Species")= chr [1:2] "human" "mouse"
#>  - attr(*, "SpeciesAbbr")= chr [1:2] "hsapiens" "mmusculus"
#>  - attr(*, "SpeciesFull")= chr [1:2] "Homo Sapiens" "mmusculus"
```

To summarize the ortholog results by genetype (eg. protein_coding, miRNA, lncRNA etc):

```
hs2mm.orth.genetype <- summarize_ortholog_gene(hs2mm.orth, group = "genetype")
head(hs2mm.orth.genetype)
#>           human mouse
#> IG_C_gene     3     3
#> IG_V_gene     2     2
#> TR_C_gene     1     1
#> TR_V_gene    16    16
#> lncRNA        1     0
#> miRNA       218   218
```

To summarize the ortholog matching results by chromosome:

```
hs2mm.orth.chrom <- summarize_ortholog_gene(hs2mm.orth, group = "chrom")
head(hs2mm.orth.chrom)
#>    human mouse
#> 1   1727  1050
#> 10   636   779
#> 11  1048  1342
#> 12   889   565
#> 13   286   546
#> 14   572   588
```

Filter 1-to-1 orthologs to exclude specific groups of genes

In many cases, specific groups of genes (eg. pseudogenes or genes from sex chromosomes) are undesirable for gene expression comparison. The function **ortholog_filter** enables the filtering of ortholog pairs based on gene type, chromosome, or provided gene list, as demonstrated below.

```
# Filter orthologs by excluding genes from chromosomes X, Y and MT, and only keep
# protein_coding genes
hs2mm.orth.flt <- ortholog_filter(
  hs2mm.orth,
  genetype_include = "protein_coding",
  chrom_exclude = c("X", "Y", "MT")
)
#> Check data for human
#> genetype_include matches:    15849
#> chrom_exclude    matches:    595
#>
#> Check data for mouse
#> genetype_include matches:    15855
#> chrom_exclude    matches:    590
#>
#> Original gene number: 16536
#> Filtered gene number: 15275
```

To check the number of 1:1 orthologs before and after filtering:

```
# Before filtering
dim(hs2mm.orth[[1]])
#> [1] 16536     4

# After filtering
dim(hs2mm.orth.flt[[1]])
#> [1] 15275     4
```

3.3 Integrate the 1-to-1 ortholog matching result to ExprX object

Take the original ExprX object generated with **make_ExprX_dataset** and the ortholog matching data generated using **ortholog_match** as input, the function **ortholog_expression_merge** integrates together the expression data for all 1:1 orthologs among compared species. The integrated data will be appended to the original ExprX object and returned as an updated object. To be noted, for orthologs that don't have matched expression data, they will be excluded from the returned data.

```
# Merge data
hs2mm.data <- ortholog_expression_merge(
  expr_data = hs2mm.data, orth_data = hs2mm.orth.flt
)
#> Number of ortholog pairs absent from expr_data
#> human:   14233
#> mouse:   14238
#>
#> Original  ortholog number: 15275
#> Discarded ortholog number: 14375
#> Resulted  ortholog number: 900
```

3.4 Integrate the normalized expression data for 1-to-1 orthologs to ExprX object

Normalization of the expression data for 1:1 orthologs among species can be performed by using the **ortholog_expression_normalize** function. Different normalization approaches are supported, including TMM, TMMwsp, RLE, upperquartile and quantile. The normalized data matrix will be appended to the original ExprX object and returned as the updated ExprX object.

To normalize the expression data of 1:1 orthologs among samples using the TMM approach:

```
# Load required package
library(edgeR)
#> Loading required package: limma

# Perform data normalization
hs2mm.data <- ortholog_expression_normalize(
  expr_data = hs2mm.data, method = "TMM"
)
```

3.5 Perform interspecies differential expression analysis

Differential expression analysis of 1:1 orthologs between species can be performed using the **ortholog_expression_compare** function. Expression data of 1:1 orthologs after normalization are used for differential expression analysis. Statistics such as average expression level, log2foldChange and p-values are calculated and returned as a dataframe.

Below shows how to perform interspecies differential analysis for 1:1 orthologs using RankProd approach. To be noted, the demo dataset is only for less than 1000 genes, thus only a couple of genes are called as differentlly expressed. If use full dataset, the number of called differential genes can be as many as several hundreds.

```
# Load required package
library(RankProd)
#> Loading required package: Rmpfr
#> Loading required package: gmp
#>
#> Attaching package: 'gmp'
#> The following objects are masked from 'package:base':
#>
#>     %*%, apply, crossprod, matrix, tcrossprod
#> C code of R package 'Rmpfr': GMP using 64 bits per limb
#>
#> Attaching package: 'Rmpfr'
#> The following object is masked from 'package:gmp':
#>
#>     outer
#> The following objects are masked from 'package:stats':
#>
#>     dbinom, dgamma, dnorm, dpois, pnorm
#> The following objects are masked from 'package:base':
#>
#>     cbind, pmax, pmin, rbind

# Perform differential analysis and then sort by p-value
hs2mm.deg <- ortholog_expression_compare(
  hs2mm.data, method = "RankProd", p_adjust = "fdr"
  )
#> Rank Product analysis for unpaired case
#>
#>
#>  done
hs2mm.deg <- hs2mm.deg[order(hs2mm.deg$P_value),]

# Check what the differential analysis result looks like
head(hs2mm.deg)
#>        GeneID_human       GeneID_mouse GeneName_human GeneName_mouse
#> 346 ENSG00000089220 ENSMUSG00000047104          PEBP1           Pbp2
#> 233 ENSG00000131095 ENSMUSG00000020932           GFAP           Gfap
#> 198 ENSG00000132341 ENSMUSG00000028287            RAN  1700009N14Rik
#> 733 ENSG00000185559 ENSMUSG00000040856           DLK1           Dlk1
#> 533 ENSG00000129538 ENSMUSG00000035896         RNASE1         Rnase1
#> 222 ENSG00000171794 ENSMUSG00000051396           UTF1        Gm45902
#>     Expression_human Expression_mouse Expression_average Log2foldChange
#> 346     1.160654e+04        1.0046551          5803.7744      12.913280
#> 233     4.798557e+04      359.1007821         24172.3349       7.060076
#> 198     4.448541e+03        0.7653213          2224.6534      11.779775
#> 733     8.359417e-01     2378.4807928          1189.6584     -10.798271
#> 533     5.322110e+03        3.8217766          2662.9657      10.266294
#> 222     5.568115e-01      617.4629126           309.0099      -9.191658
#>           P_value
#> 346 1.504507e-05
#> 233 9.797977e-05
#> 198 1.016858e-04
#> 733 2.257631e-04
#> 533 4.537869e-04
#> 222 9.593216e-04
```

To determine differential genes (based on cufoff of p-value and log2foldChange) which can be saved for downstream analysis:

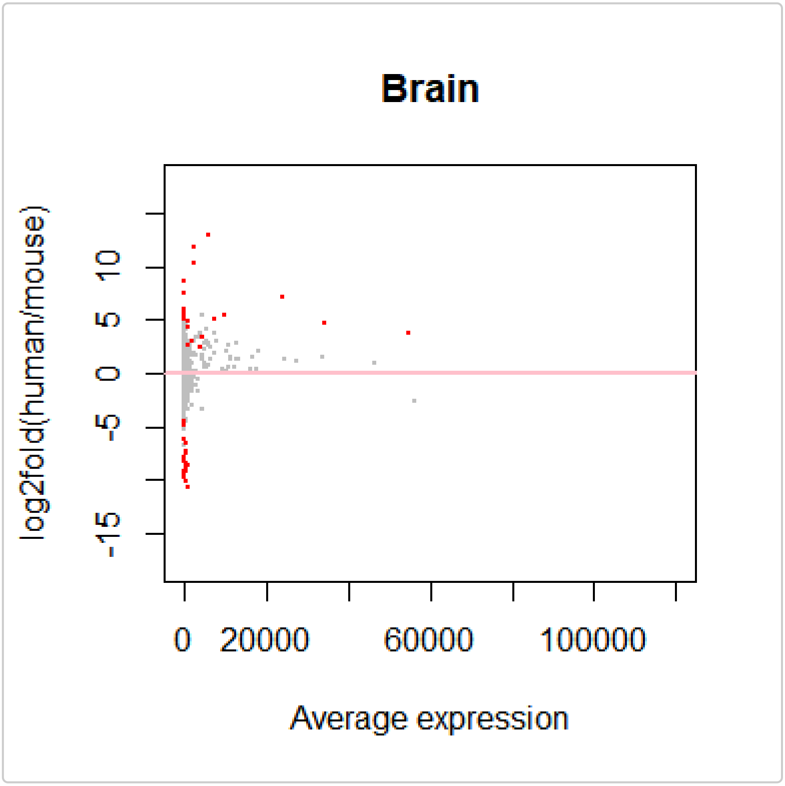
```
# Determine significant differential genes (human>mouse) based on p-values and log2foldChange
hs2mm.hsHigh <- subset(hs2mm.deg, subset = log2foldChange > 1  & P_value < 0.05)
head(hs2mm.hsHigh)
#>        GeneID_human       GeneID_mouse GeneName_human GeneName_mouse
#> 346 ENSG00000089220 ENSMUSG00000047104          PEBP1           Pbp2
#> 233 ENSG00000131095 ENSMUSG00000020932           GFAP           Gfap
#> 198 ENSG00000132341 ENSMUSG00000028287            RAN  1700009N14Rik
#> 533 ENSG00000129538 ENSMUSG00000035896         RNASE1         Rnase1
#> 504 ENSG00000145945 ENSMUSG00000038246         FAM50B         Fam50b
#> 190 ENSG00000204655 ENSMUSG00000076439            MOG            Mog
#>     Expression_human Expression_mouse Expression_average log2foldChange
#> 346       11606.5442        1.0046551          5803.7744      12.913280
#> 233       47985.5690      359.1007821         24172.3349       7.060076
#> 198        4448.5415        0.7653213          2224.6534      11.779775
#> 533        5322.1096        3.8217766          2662.9657      10.266294
#> 504         312.7094        0.2713377           156.4903       8.665549
#> 190        3020.8812      399.3283886          1710.1048       2.917755
#>           P_value
#> 346 1.504507e-05
#> 233 9.797977e-05
#> 198 1.016858e-04
#> 533 4.537869e-04
#> 504 2.639805e-03
#> 190 1.770427e-02

# Determine significant differential genes (human<mouse) based on p-values and log2foldChange
hs2mm.mmHigh <- subset(hs2mm.deg, subset = log2foldChange < -1 & P_value < 0.05)
head(hs2mm.mmHigh)
#>        GeneID_human       GeneID_mouse GeneName_human GeneName_mouse
#> 733 ENSG00000185559 ENSMUSG00000040856           DLK1           Dlk1
#> 222 ENSG00000171794 ENSMUSG00000051396           UTF1        Gm45902
#> 389 ENSG00000170236 ENSMUSG00000027364          USP50          Usp50
#> 618 ENSG00000008196 ENSMUSG00000025927         TFAP2B         Tfap2b
#> 385 ENSG00000151079 ENSMUSG00000038077          KCNA6          Kcna6
#> 507 ENSG00000170561 ENSMUSG00000001504           IRX2           Irx2
#>     Expression_human Expression_mouse Expression_average log2foldChange
#> 733       0.83594167        2378.4808          1189.6584     -10.798271
#> 222       0.55681154         617.4629           309.0099      -9.191658
#> 389       3.63377004        1604.9029           804.2683      -8.601262
#> 618       0.08851788         631.8555           315.9720     -10.069434
#> 385       0.00000000         296.3993           148.1996      -9.213830
#> 507       1.02556448         608.3116           304.6686      -8.640509
#>           P_value
#> 733 0.0002257631
#> 222 0.0009593216
#> 389 0.0010448787
#> 618 0.0011482998
#> 385 0.0021331634
#> 507 0.0021331634
```

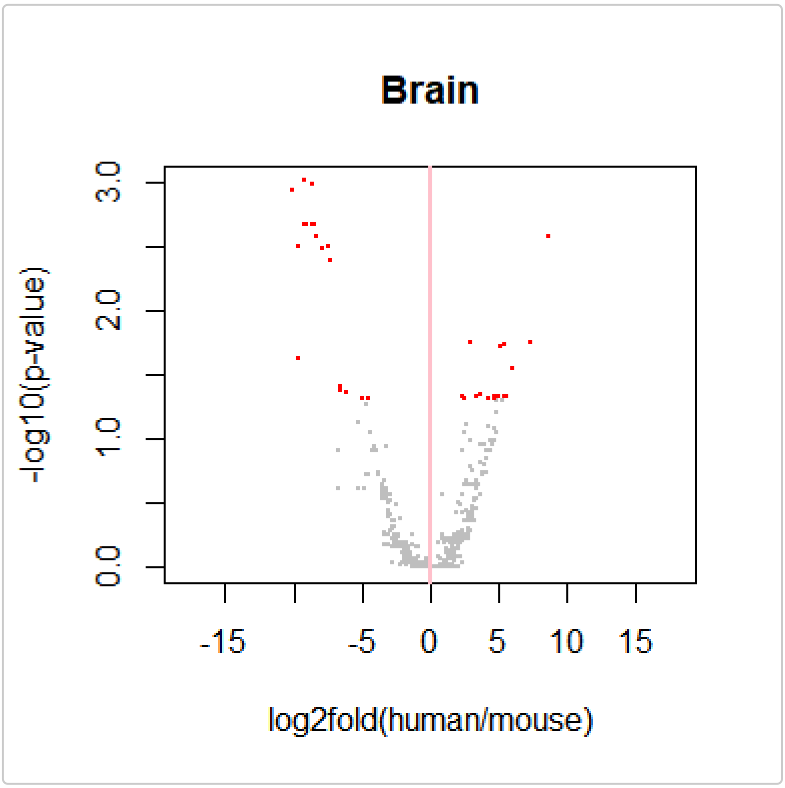
3.6 Visualize differential expression analysis result

The differential expression can be visualized as MA-plot or Volcano-plot, with differential genes highlighted by color. Below shows how to use the **ortholog_expression_plot** function to generate MA-plot and Volcano-plot, respectively.

```
# Generate MA-plot
ortholog_expression_plot(
  hs2mm.deg, "MA",
  main = "Brain", xlim = c(0,120000), ylim = c(-18,18)
  )
```



```
# Generate Volcano-plot
ortholog_expression_plot(
  hs2mm.deg, "volcano",
  main = "Brain", xlim = c(-18,18), ylim = c(0, 3)
  )
```



Brain
-log10(p-value)
0.0
1.0
2.0
3.0
-15
-5
0
5
10
15
log2fold(human/mouse)