Advanced data processing

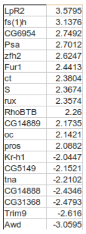
# 3

## Combining data from multiple sources
*Gene clustering example*

- R has powerful functions to combine heterogeneous data into a single data set
- Gene clustering example data:
  - five sets of differentially expressed genes from various experimental conditions
  - file with names of experimentally verified genes
- Gene clustering exercise:
  1. combine this dataset into a single table and cluster to see which conditions are similar
  2. repeat the clustering but only on a subset of experimentally verified genes

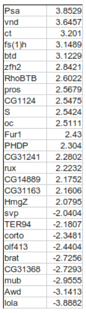
## Combining gene tables

- input files have two columns: gene names and fold change
- we want to combine all five tables into a single table, with 0 for missing values



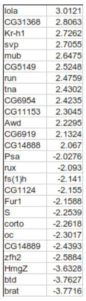
| | |
|---|---|
| LpR2 | 3.5795 |
| fs(1)h | 3.1376 |
| CG6954 | 2.7492 |
| Psa | 2.7012 |
| zfh2 | 2.6247 |
| Fur1 | 2.4413 |
| ct | 2.3804 |
| S | 2.3674 |
| rux | 2.3574 |
| RhoBTB | 2.26 |
| CG14889 | 2.1735 |
| oc | 2.1421 |
| pros | 2.0882 |
| Kr-h1 | -2.0447 |
| CG5149 | -2.1521 |
| tna | -2.2102 |
| CG14888 | -2.4346 |
| CG31368 | -2.4793 |
| Trim9 | -2.616 |
| Awd | -3.0595 |

+



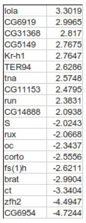
| | |
|---|---|
| Psa | 3.8529 |
| vnd | 3.6457 |
| ct | 3.201 |
| fs(1)h | 3.1489 |
| btd | 3.1229 |
| zfh2 | 2.8421 |
| RhoBTB | 2.6022 |
| pros | 2.5679 |
| CG1124 | 2.5475 |
| S | 2.5424 |
| oc | 2.5111 |
| Fur1 | 2.43 |
| PHDP | 2.304 |
| CG31241 | 2.2802 |
| rux | 2.2232 |
| CG14889 | 2.1752 |
| CG31163 | 2.1606 |
| HmgZ | 2.0795 |
| svp | -2.0404 |
| TER94 | -2.1807 |
| corto | -2.3481 |
| olf413 | -2.4404 |
| brat | -2.7256 |
| CG31368 | -2.7293 |
| mub | -2.9555 |
| Awd | -3.1413 |
| lola | -3.8882 |

+



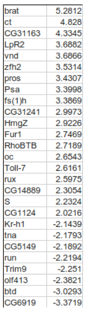
| | |
|---|---|
| lola | 3.0121 |
| CG31368 | 2.8063 |
| Kr-h1 | 2.7262 |
| svp | 2.7055 |
| mub | 2.6475 |
| CG5149 | 2.5248 |
| run | 2.4759 |
| tna | 2.4302 |
| CG6954 | 2.4235 |
| CG11153 | 2.3045 |
| Awd | 2.2295 |
| CG6919 | 2.1324 |
| CG14888 | 2.067 |
| Psa | -2.0276 |
| rux | -2.093 |
| fs(1)h | -2.141 |
| CG1124 | -2.155 |
| Fur1 | -2.1588 |
| S | -2.2539 |
| corto | -2.2618 |
| oc | -2.3017 |
| CG14889 | -2.4393 |
| zfh2 | -2.5884 |
| HmgZ | -3.6328 |
| btd | -3.7627 |
| brat | -3.7716 |

+



| | |
|---|---|
| lola | 3.3019 |
| CG6919 | 2.9965 |
| CG31368 | 2.817 |
| CG5149 | 2.7675 |
| Kr-h1 | 2.7647 |
| TER94 | 2.6286 |
| tna | 2.5748 |
| CG11153 | 2.4795 |
| run | 2.3831 |
| CG14888 | 2.0938 |
| S | -2.0243 |
| rux | -2.0668 |
| oc | -2.3437 |
| corto | -2.5556 |
| fs(1)h | -2.6211 |
| brat | -2.9904 |
| ct | -3.3404 |
| zfh2 | -4.4947 |
| CG6954 | -4.7244 |

+



| | |
|---|---|
| brat | 5.2812 |
| ct | 4.828 |
| CG31163 | 4.3345 |
| LpR2 | 3.6882 |
| vnd | 3.6866 |
| zfh2 | 3.5314 |
| pros | 3.4307 |
| Psa | 3.3998 |
| fs(1)h | 3.3869 |
| CG31241 | 2.9973 |
| HmgZ | 2.9226 |
| Fur1 | 2.7469 |
| RhoBTB | 2.7189 |
| oc | 2.6543 |
| Toll-7 | 2.6161 |
| rux | 2.5975 |
| CG14889 | 2.3054 |
| S | 2.2324 |
| CG1124 | 2.0216 |
| Kr-h1 | -2.1439 |
| tna | -2.1793 |
| CG5149 | -2.1892 |
| run | -2.2194 |
| Trim9 | -2.251 |
| olf413 | -2.3821 |
| btd | -3.0293 |
| CG6919 | -3.3719 |

# Gene clustering
Script walkthrough 1

- To make the big table we first need to find out all the genes present in at least one of the files
- Make sure not to use factors in read.delim()

```
# start with en empty collection of genes
genes <- c()
for( fileNum in 1:5 ){
   # load in files 13_DiffGenes1.tsv ...
   t <- read.delim(paste("13_DiffGenes", fileNum, ".tsv", sep=""),
                 as.is=TRUE, header=FALSE)
   # label the input columns to help code readability
   names(t) <- c("gene", "expression")
   genes <- union(genes, t$gene)
}

# for tidiness order our genes by name
genes <- sort(genes)

genes # show all genes
```

when loading in character data use **as.is=T** to prevent it being converted to factors!

union() is a set operation, combines two vectors by eliminating duplicates. There are also intersect() and setdiff()

Example code:
13_geneClustering.R

# Gene clustering
Script walkthrough 2

- Using the complete list of genes, we can create the big table and fill in the values:

```
# make the destination table [rows = unique genes, cols = file numbers]
values <- matrix(0, nrow=length(genes), ncol=5)
rownames(values) <- genes # name the rows with the complete gene names

for(fileNum in 1:5){
  # read in the file again
     t <- read.delim(paste("13_DiffGenes", fileNum, ".tsv", sep=""),
            as.is=T, header=F)
     names(t) <- c("gene", "expression")

     # match the names of the genes to the rows in our big table
     index <- match(t$gene, rownames(values))
     # copy the expression levels
     values[index,fileNum] <- t$expression
}
```

match() returns the index of first argument in the second, i.e. index of input file genes in the big table

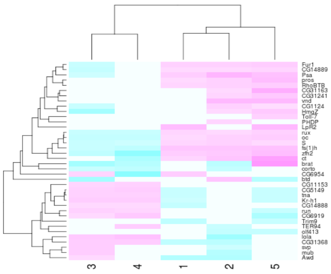
we use index to pick the rows in such way that they match the gene order in the input file

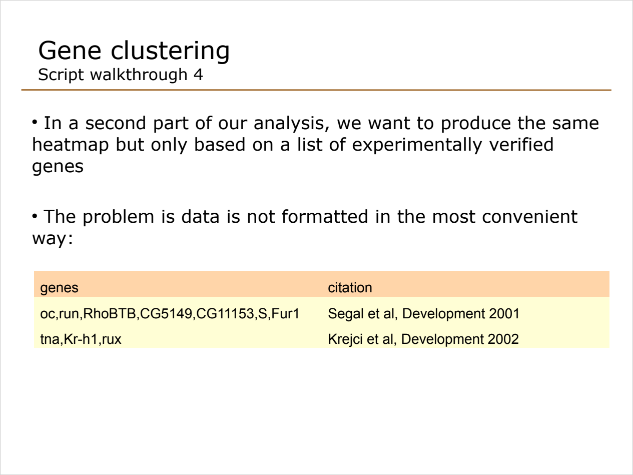
# Gene clustering
Script walkthrough 3

- Now we can do hierarchical clustering:

```
heatmap(values, scale="none", col = cm.colors(256))
```

Values from the matrix are colour-coded. Rows and columns are re-arranged according to similarity

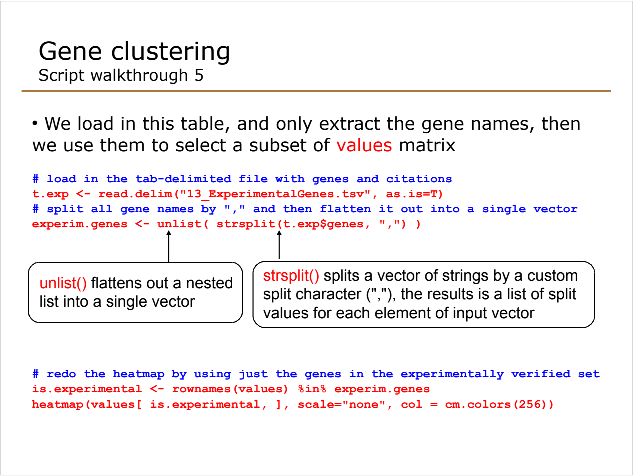




## Gene clustering
Script walkthrough 4

- In a second part of our analysis, we want to produce the same heatmap but only based on a list of experimentally verified genes
- The problem is data is not formatted in the most convenient way:

| genes | citation |
|---|---|
| oc,run,RhoBTB,CG5149,CG11153,S,Fur1 | Segal et al, Development 2001 |
| tna,Kr-h1,rux | Krejci et al, Development 2002 |



## Gene clustering
Script walkthrough 5

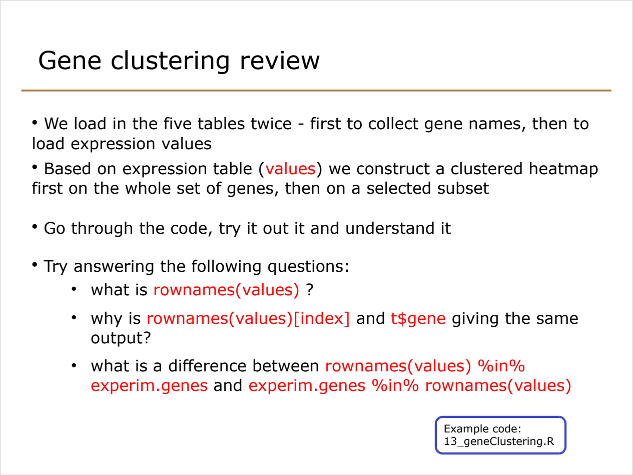
- We load in this table, and only extract the gene names, then we use them to select a subset of values matrix

```
# load in the tab-delimited file with genes and citations
t.exp <- read.delim("13_ExperimentalGenes.tsv", as.is=T)
# split all gene names by "," and then flatten it out into a single vector
experim.genes <- unlist( strsplit(t.exp$genes, ",") )
```

unlist() flattens out a nested list into a single vector

strsplit() splits a vector of strings by a custom split character (","), the results is a list of split values for each element of input vector

```
# redo the heatmap by using just the genes in the experimentally verified set
is.experimental <- rownames(values) %in% experim.genes
heatmap(values[ is.experimental, ], scale="none", col = cm.colors(256))
```



## Gene clustering review

- We load in the five tables twice - first to collect gene names, then to load expression values
- Based on expression table (values) we construct a clustered heatmap first on the whole set of genes, then on a selected subset
- Go through the code, try it out and understand it
- Try answering the following questions:
  - what is rownames(values) ?
  - why is rownames(values)[index] and t$gene giving the same output?
  - what is a difference between rownames(values) %in% experim.genes and experim.genes %in% rownames(values)

Example code:
13_geneClustering.R