Analysing the first set of SILAC-based LOPIT data

```
#-----------------------------------------------------------------------------
# Author          : Manasa Ramakrishna, mr325@le.ac.uk
# Date started  : 1st June, 2017
# Last modified : 15th June, 2017
# Aim             : To take a look at first SILAC labelled LOPIT data on Trizol
# Depends       : On 'silacFunctions.R'. Make sure they are in the same directory
# Notes         : Works on data from Rayner's first experiments
#-----------------------------------------------------------------------------



# Invoking libraries
library(clusterProfiler)
library(ggplot2)
library(gplots)
library(limma)
library(org.Hs.eg.db)
library(outliers)
library(RColorBrewer)
library(reshape2)
library(stringr)

#Setting working directories
wd = "/Users/manasa/Documents/Work/TTT/02_Proteomics/01_First-SILAC-LOPIT/"
setwd(wd)
getwd()
```

```
## [1] "/Users/manasa/Documents/Work/TTT/02_Proteomics/01_First-SILAC-LOPIT"
```

```
indir = paste(wd,"Input",sep="/")
outdir = paste(wd,paste(Sys.Date(),"Output",sep = "_"),sep = "/")

if (exists(outdir)){
  print("Outdir exists")
}else{
  dir.create(outdir)
}

# Sourcing function file
source("silacFunctions.R")
```

Now that we have loaded all the packages we need for working with this data, let’s move on to the data.

```
# ------------------------------------------------
# Step 0: Read data
# Read in all the data required for analysis
# ------------------------------------------------

# File of contaminants - proteins to exclude from analysis as are things like keratin, alcohol dehydrog
contam = read.delim("Input/Common contaminant_all.csv",sep=",",header=T)

# Read in the data files that contain peptide level output from Proteome discoverer...
# Note: I have converted the excel files to text files and removed '(02)' from the filenames to make it
```

```r
# Modify the headers to be all lower case as well as remove unwanted spaces, symbols etc...to keep it s
# Columns of interest are "sequence", "modifications","master.protein.accessions","abundance.heavy","ab

infiles = grep("Trizol",list.files("Input/",full.names = T),value=T)
prot.data = NULL
for (i in infiles){
  in.dat = read.delim(i,sep="\t",comment.char="",as.is=T,header=F)
  in.dat$sample = strsplit(i,"//")[[1]][2]
  #print(i)
  prot.data = rbind(prot.data,in.dat)
}

colnames(prot.data) = prot.data[1,]
dim(prot.data)
```

```
## [1] 81595    29
```

```r
# Remove header lines as they differ in one of the columns (fraction number I think)
remove.head = which(prot.data[,1]=="Checked")
prot.data = prot.data[-(remove.head),]
dim(prot.data)
```

```
## [1] 81586    29
```

```r
# Change header names a little to make them neutral and remove space, special characters
colnames(prot.data) = tolower(colnames(prot.data))
colnames(prot.data) = gsub(" ",".",colnames(prot.data))
colnames(prot.data) = gsub("#","no",colnames(prot.data))
colnames(prot.data) = gsub("\\:\\.f2\\:","n",colnames(prot.data))
colnames(prot.data)[12] = "theoretical.mass"
colnames(prot.data)[13] = "light.sample"
colnames(prot.data)[14] = "heavy.sample"
colnames(prot.data)[15] = "abundance.ratio.heavy.to.light"
colnames(prot.data)[16] = "abundance.light"
colnames(prot.data)[17] = "abundance.heavy"
colnames(prot.data)[29] = "sample"

# Convert abundance values to numeric from character
prot.data$abundance.heavy = as.numeric(prot.data$abundance.heavy)
prot.data$abundance.light = as.numeric(prot.data$abundance.light)
prot.data$abundance.ratio.heavy.to.light = as.numeric(prot.data$abundance.ratio.heavy.to.light)

# Add rep, reagent and UV amount columns
prot.data$uv = sapply(strsplit(prot.data$sample,"_"),"[[",2)
prot.data$repl = gsub(".txt","",gsub("rep","",sapply(strsplit(prot.data$sample,"_"),"[[",3)))
prot.data$repl = paste(prot.data$uv,prot.data$rep,sep=".")
prot.data$reagent = sapply(strsplit(prot.data$sample,"_"),"[[",1)
head(prot.data)
```

```
##   checked confidence            sequence              modifications
## 2   FALSE       High          AGAHLQGGAK
## 3   FALSE       High        IMNTFSVVPSPK 1xLabel:13C(6)15N(2) [K12]
## 4   FALSE       High        IMNTFSVVPSPK
## 5   FALSE       High    NQVTQLKEQVPGFTPR
## 6   FALSE       High EXELQQTLQQEQSVLDQLR
## 7   FALSE       High      TTPSVVAFTADGER
##   qvality.pep qvality.q-value no.protein.groups no.proteins no.psms
## 2 5.61944E-06               0                 1           2      16
## 3  3.4751E-06               0                 4          11       8
## 4 2.42801E-06               0                 4          11       6
## 5 0.000142696               0                 1           2       4
## 6 1.97349E-07               0                 1           1       6
## 7 9.80731E-06               0                 1           3       4
##       master.protein.accessions no.missed.cleavages theoretical.mass
## 2                        P04406                   0      909.4900898
## 3 Q13509; P04350; P07437; P68371                   0      1327.716983
## 4 Q13509; P04350; P07437; P68371                   0      1319.702784
## 5                        F5H2F4                   1      1841.986822
## 6                        Q15149                   0      2313.168093
## 7                        P38646                   0      1450.717248
##   light.sample heavy.sample abundance.ratio.heavy.to.light abundance.light
## 2         High   Peak Found                           0.01         5200000
## 3    Not Found    Not Found                             NA              NA
## 4    Not Found    Not Found                             NA              NA
## 5    Not Found    Not Found                             NA              NA
## 6         High    Not Found                           0.01          656400
## 7         High    Not Found                             NA              NA
##   abundance.heavy      quan.info amanda.score.ms.amanda
## 2           43050         Unique            135.209857
## 3              NA No Quan Values           201.8662137
## 4              NA No Quan Values           176.2597659
## 5              NA No Quan Values           127.0712654
## 6              NA         Unique           152.1398947
## 7              NA No Quan Values           104.2831076
##   confidence.ms.amanda search.space.ms.amanda percolator.q-value.ms.amanda
## 2                 High                   1636                            0
## 3                 High                   2601                            0
## 4                 High                   2728                            0
## 5                 High                   3048                            0
## 6                 High                   3758                            0
## 7                 High                   2414                            0
##   percolator.pep.ms.amanda ions.score.mascot confidence.mascot
## 2               0.00001018             65.04              High
## 3              0.000002614             75.28              High
## 4               0.00001715             79.87              High
## 5               0.00005757             40.09              High
## 6                1.928E-07             44.95              High
## 7               0.00002452             48.27              High
##   search.space.mascot percolator.q-value.mascot percolator.pep.mascot
## 2                                             0             2.945E-07
## 3                                             0             1.695E-07
## 4                                             0             1.112E-07
## 5                                             0            0.00003356
## 6                                             0             1.211E-08
## 7                                             0             5.664E-07
##                  sample    uv    repl reagent
## 2 Trizol_150mJ_rep1.txt 150mJ 150mJ.1  Trizol
## 3 Trizol_150mJ_rep1.txt 150mJ 150mJ.1  Trizol
## 4 Trizol_150mJ_rep1.txt 150mJ 150mJ.1  Trizol
## 5 Trizol_150mJ_rep1.txt 150mJ 150mJ.1  Trizol
## 6 Trizol_150mJ_rep1.txt 150mJ 150mJ.1  Trizol
## 7 Trizol_150mJ_rep1.txt 150mJ 150mJ.1  Trizol
```

```
dim(prot.data)
```

```
## [1] 81586    32
```

prot.data has 32 columns and 81,586 rows - each row belonging to a peptide. We now go through a series of filtering steps to obtain a dataset we can use for downstream analyses.

```
# ---------------------------------------------------------------------------------
# Step 1 : Filter
# We perform 3 layers of filtering - unique proteins, contaminants,missing values
# ---------------------------------------------------------------------------------

# Step 1a : Filter only for those peptides that have a unique master protein. Done using column "quan.i
dim(prot.data)
```

```
## [1] 81586    32
```

```
peptide.stats = table(prot.data$sample,prot.data$quan.info)
peptide.stats
```

```
##
##                           No Quan Values Not Unique Unique
##   Trizol_150mJ_rep1.txt             5211        287   4088
##   Trizol_150mJ_rep2.txt             3869        248   3048
##   Trizol_150mJ_rep3.txt             4571        247   3483
##   Trizol_275mJ_rep1.txt             5266        289   3845
##   Trizol_275mJ_rep2.txt             4303        212   3091
##   Trizol_275mJ_rep3.txt             6578        468   6383
##   Trizol_400mJ_rep1.txt             5365        286   4129
##   Trizol_400mJ_rep2.txt             5203        265   3422
##   Trizol_400mJ_rep3.txt             4320        219   2890
```

```
filt.1a = prot.data[which(prot.data$quan.info == "Unique"),]
length(which(filt.1a$quan.info == "Unique"))
```

```
## [1] 34379
```

```
dim(filt.1a) #34279 are unique proteins, 47207 are non-unique or are missing values
```

```
## [1] 34379    32
```

```
# This table is very odd. Rayner had an explanation - "High" was equivalent to "Peak found"
# but also indicates which label "heavy" or "light" is higher in abundance
# However, there are peptides where it is "High" but the peptide values are NA. Hmmm....
table(light=filt.1a$light.sample,heavy=filt.1a$heavy.sample)
```

```
##             heavy
## light         High Not Found Peak Found
##   High           0     15810      11775
##   Not Found   1672         0          0
##   Peak Found  5122         0          0
```

```
# Step 1b : Filter out those proteins that are contaminants from the contaminants list and annotate mis
filt.1b = filt.1a[-which(filt.1a$master.protein.accessions %in% contam$Protein.Group.Accessions),]
num.contams = length(which(filt.1a$master.protein.accessions %in% contam$Protein.Group.Accessions))
```

```r
# Annotate which peptides are missing heavy, light or both, abundance values
filt.1b$missing.val = rowSums(is.na(filt.1b[,c("abundance.heavy", "abundance.light")])) > 0

dim(filt.1a) # 34379 in total
```

```
## [1] 34379    32
```

```r
dim(filt.1b) # 33657 filtered proteins
```

```
## [1] 33657    33
```

```r
print(num.contams) # 722 contaminant proteins
```

```
## [1] 722
```

```r
# Want to do some stats with missing values.
table(filt.1b$sample,filt.1b$missing.val) # More missing values in 150mJ_rep, 275mK_rep2 and 450mJ_rep3
```

```
##
##                          FALSE TRUE
##   Trizol_150mJ_rep1.txt  1914 2097
##   Trizol_150mJ_rep2.txt  1460 1523
##   Trizol_150mJ_rep3.txt  1667 1723
##   Trizol_275mJ_rep1.txt  1836 1915
##   Trizol_275mJ_rep2.txt  1358 1670
##   Trizol_275mJ_rep3.txt  3055 3175
##   Trizol_400mJ_rep1.txt  2024 2031
##   Trizol_400mJ_rep2.txt  1572 1797
##   Trizol_400mJ_rep3.txt  1428 1412
```

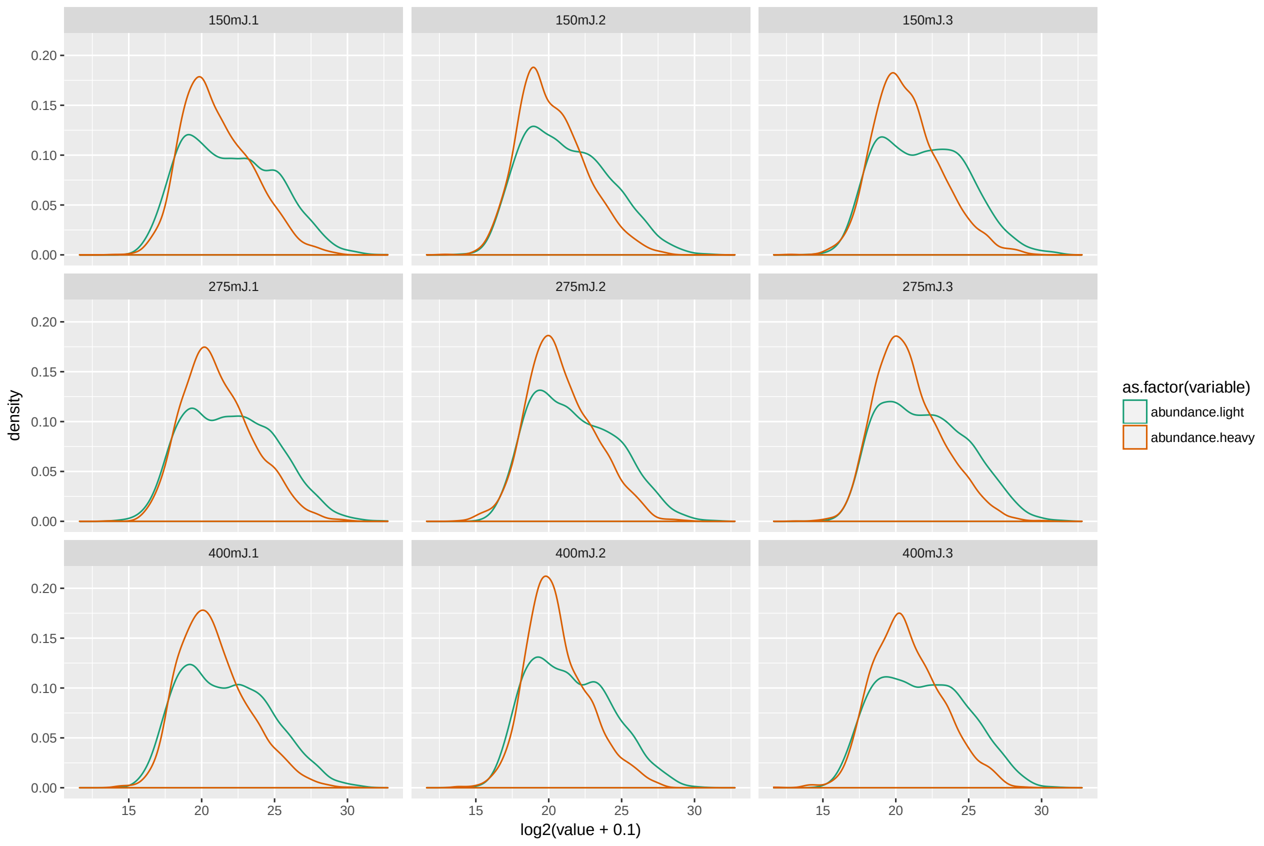
```r
round(table(filt.1b$sample,filt.1b$missing.val)/rowSums(table(filt.1b$sample,filt.1b$missing.val))*100,
```

```
##
##                          FALSE  TRUE
##   Trizol_150mJ_rep1.txt 47.72 52.28
##   Trizol_150mJ_rep2.txt 48.94 51.06
##   Trizol_150mJ_rep3.txt 49.17 50.83
##   Trizol_275mJ_rep1.txt 48.95 51.05
##   Trizol_275mJ_rep2.txt 44.85 55.15
##   Trizol_275mJ_rep3.txt 49.04 50.96
##   Trizol_400mJ_rep1.txt 49.91 50.09
##   Trizol_400mJ_rep2.txt 46.66 53.34
##   Trizol_400mJ_rep3.txt 50.28 49.72
```

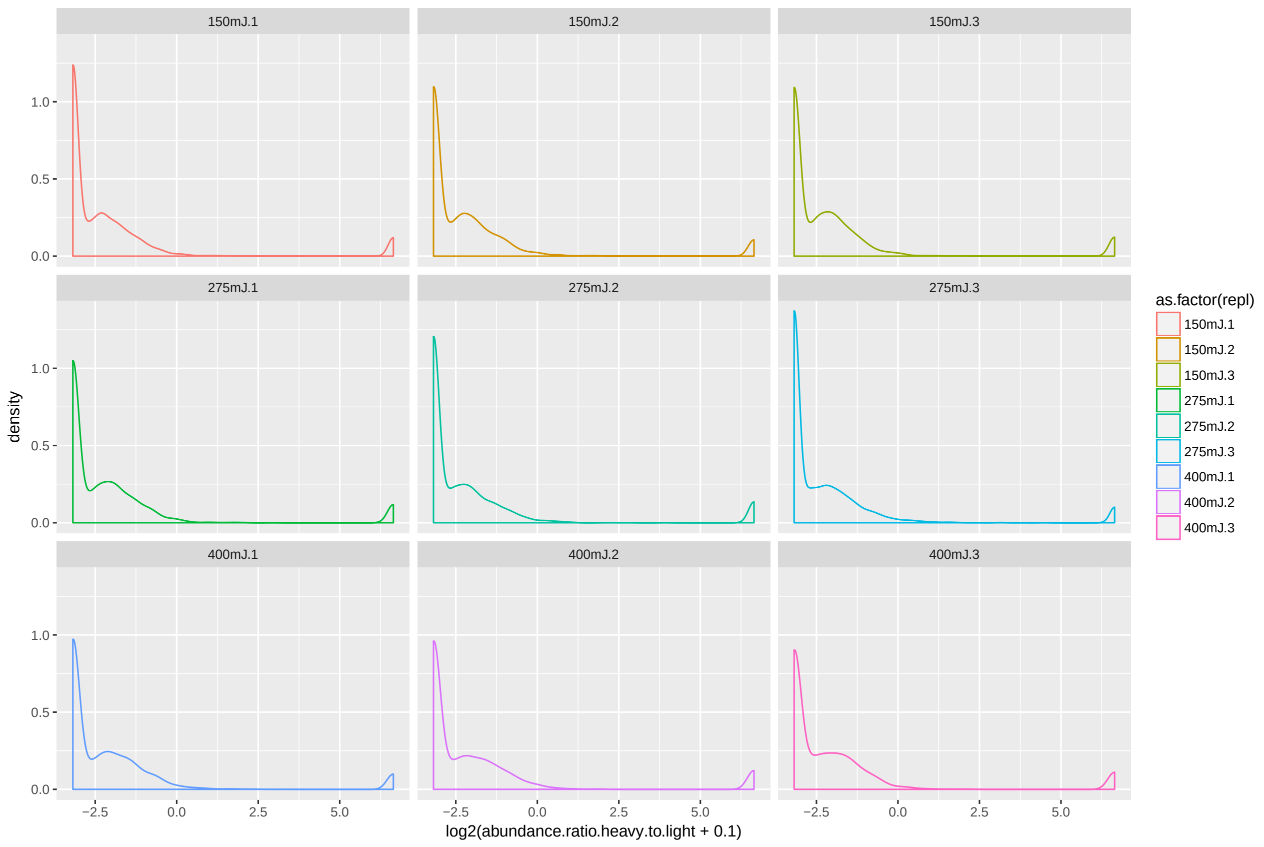
```r
# How many missing in heavy, how many missing in light
miss.l = table("_Missing light values_"=filt.1b$missing.val,filt.1b$light.sample)
miss.h = table("_Missing heavy values_"=filt.1b$missing.val,filt.1b$heavy.sample)
miss = cbind(miss.l,miss.h)
colnames(miss) = c("Light_High","Light_NotFound","Light_Found","Heavy_High","Heavy_NotFound","Heavy_Fou
rownames(miss) = c("notMissing","missing")
print(miss)
```

```
##            Light_High Light_NotFound Light_Found Heavy_High Heavy_NotFound
## notMissing      11258              0        5056       5056              0
## missing         15644           1658          41       1699          15215
##            Heavy_Found
## notMissing       11258
## missing             429
```

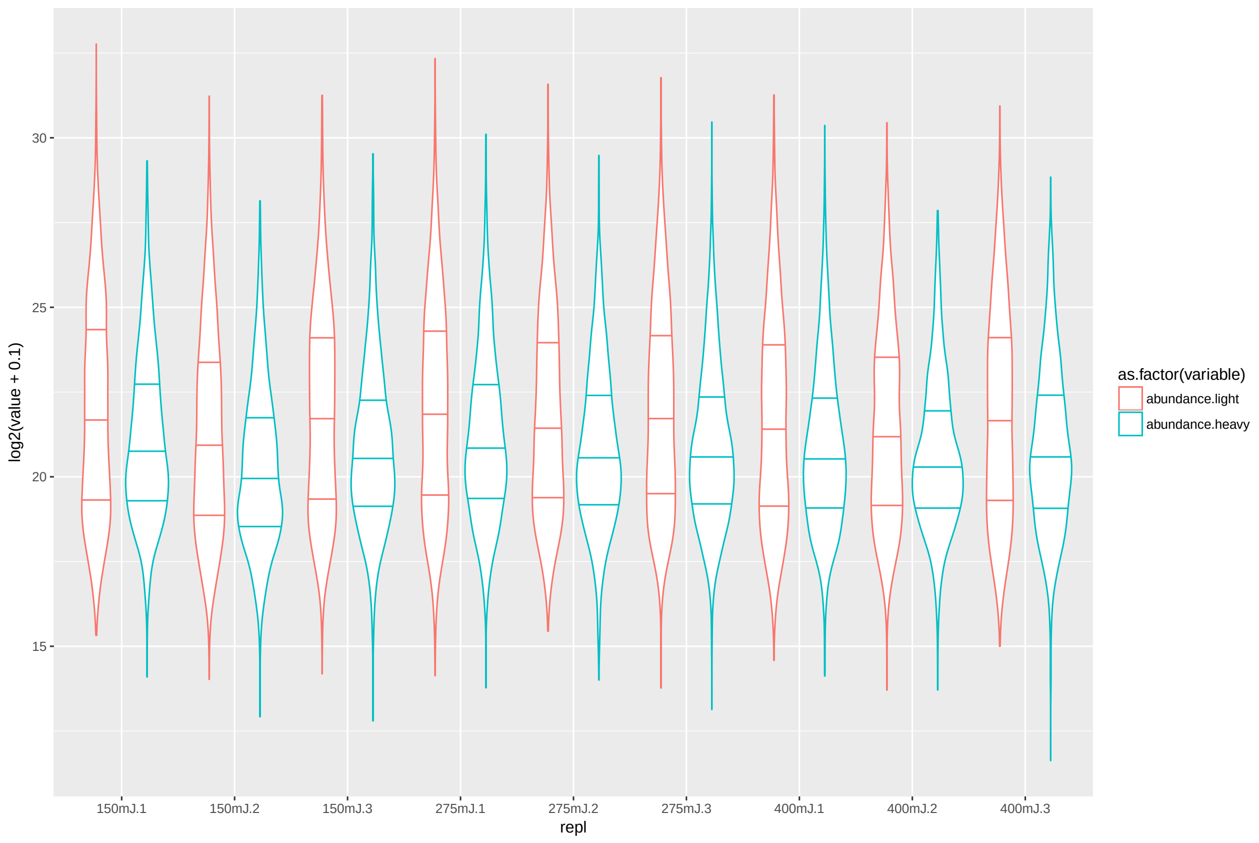
```
# Plot density plots
melt.1b = melt(filt.1b,id.vars = "repl", measure.vars = c("abundance.light", "abundance.heavy"))


ggplot(melt.1b,aes(x = log2(value+0.1))) + geom_density(aes(col = as.factor(variable))+facet_wrap(~repl
```



```
ggplot(filt.1b,aes(x = log2(abundance.ratio.heavy.to.light+0.1))) + geom_density(aes(col = as.factor(rep
```



```
ggplot(melt.1b,aes(x=repl,y = log2(value+0.1))) + geom_violin(aes(col = as.factor(variable)),draw_quant
```



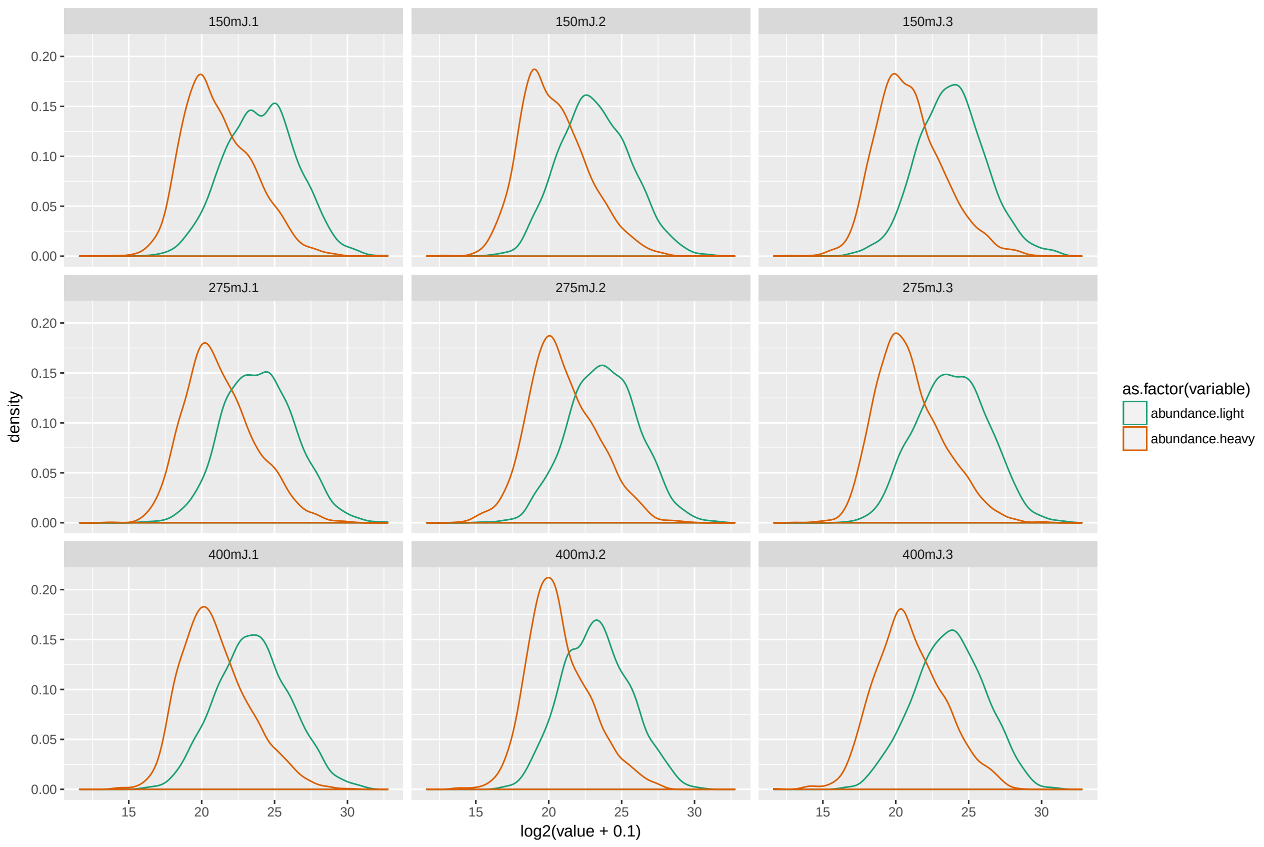
We have a column called “missing.val” to identify which peptides have either a heavy or light abundance value missing. TRUE means it is missing one or both. FALSE means both values are present. A lot more “missing” values in the “heavy/non-crosslinked samples than”light/cross-linked samples“.

The above plots - density plots and violin plots include both missing and non-missing values. Hence, in the heavy/light density plots, you see a huge overlap in the curve with a tiny portion of the “light” curve going beyond the heavy curve. This is where we expect the interesting RNA binding proteins to lie. The next step however is to filter out the missing values.

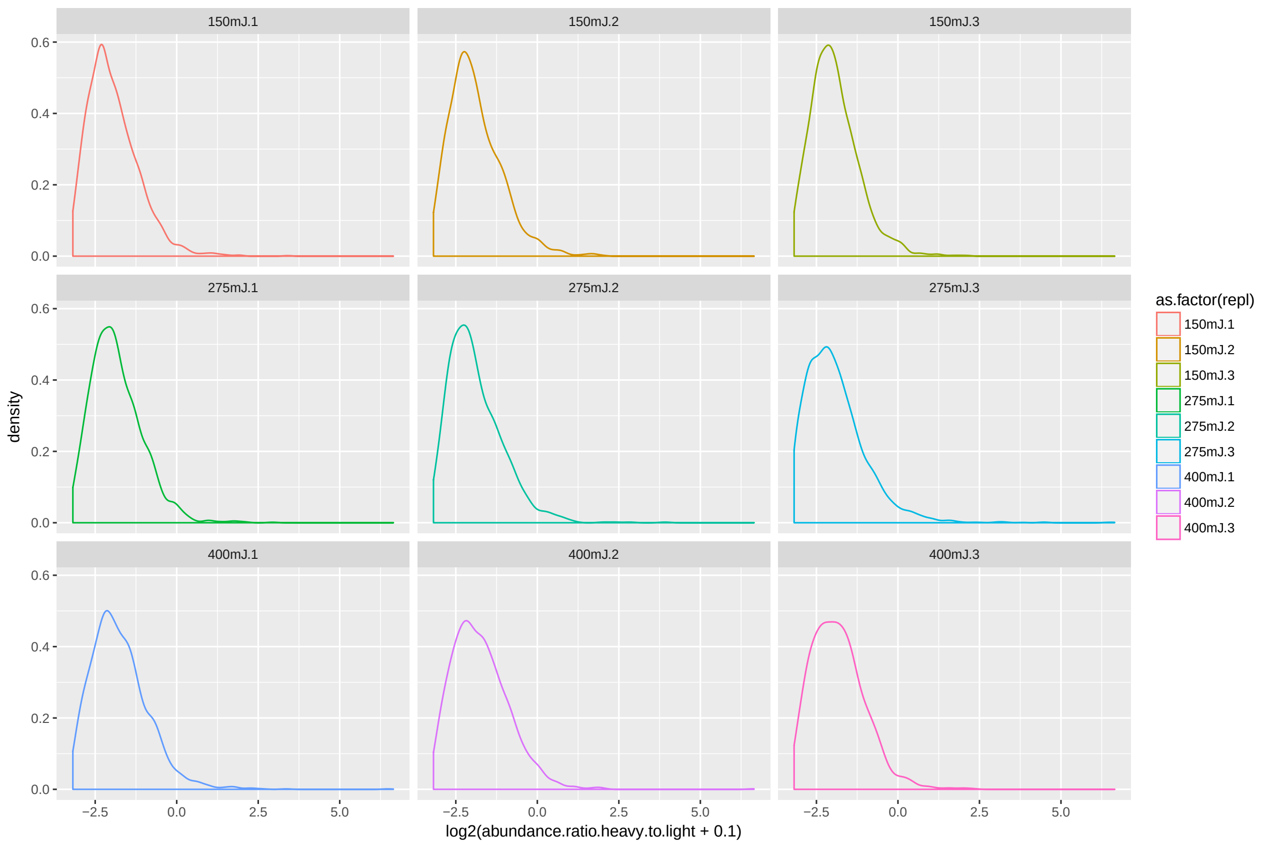
```
# Step 1c : Filter out those peptides which are missing either "high" or "low" abundance values. We wil
# as we do not know for sure whether these are a result of extremely low signal due to enrichment or ex
# There are 16314 peptides where we have quantification in both light and heavy abundance columns

filt.1c = filt.1b[which(rowSums(is.na(filt.1b[,c("abundance.heavy", "abundance.light")])) == 0),]
dim(filt.1c)
```

```
## [1] 16314    33
```

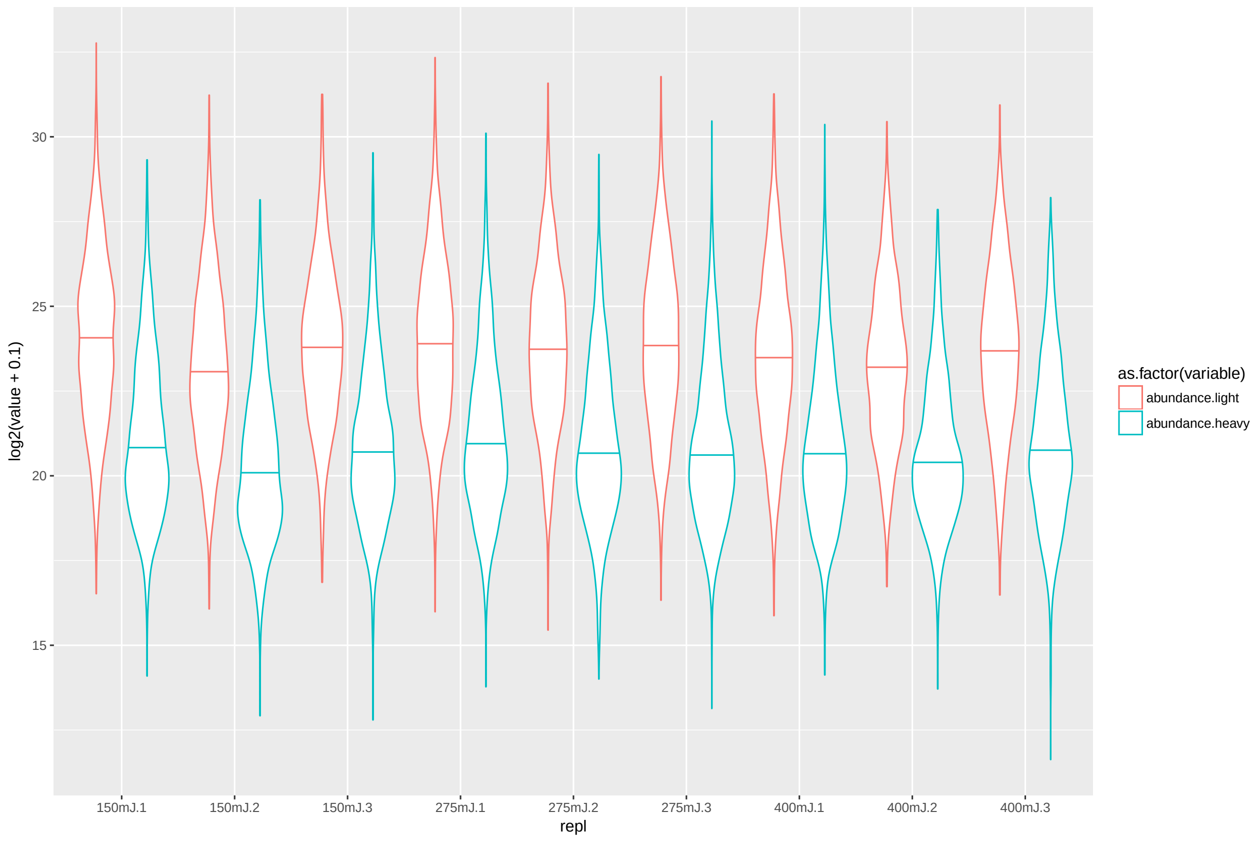
```
# Plot density plots
melt.1c = melt(filt.1c,id.vars = "repl", measure.vars = c("abundance.light", "abundance.heavy"))

ggplot(melt.1c,aes(x = log2(value+0.1))) + geom_density(aes(col = as.factor(variable)))+facet_wrap(~repl
```



```
ggplot(filt.1c,aes(x = log2(abundance.ratio.heavy.to.light+0.1))) + geom_density(aes(col = as.factor(re
```



```
ggplot(melt.1c,aes(x=repl,y = log2(value+0.1))) + geom_violin(aes(col = as.factor(variable)),draw_quant
```



Once we remove peptides where the "heavy" or "light" value is missing, then there is a clear shift in the curve of intensity values for the "light" labelled sample which is our cross-linked sample and we hope it contains true RNA binding proteins. The median abundance for light samples is visibily higher than in heavy samples.

Note : It is important to remember that in a true experimental setting, we will not have SILAC labelling so we won't have "heavy" and "light" values per peptide - all we will have is one abundance value. Rayner forced the mass spec to run as if it didn't know about the SILAC labelling to parially emulate later experiments. However, we won't be using the singleton data (heacvy only or light only) for the purposes of this initial analysis.

```
# -----------------------------
# Step 2 : Log-transform
# heavy = non-crosslinked
# light = crosslinked
# -----------------------------

# Log convert abundance values
filt.1c$heavy.log = log(filt.1c$abundance.heavy,2)
filt.1c$light.log = log(filt.1c$abundance.light,2)

# Generate an abundance ratio which for log transformed data is a subtraction
filt.1c$norm.abundance.ratio = filt.1c$light.log - filt.1c$heavy.log

# Data is checked
norm.data = filt.1c
dim(norm.data)
```

```
## [1] 16314    36
```

```
# Checking the counts of peaks with heavy and light values
table(light=norm.data$light.sample,heavy=norm.data$heavy.sample)
```

```
##             heavy
## light         High Peak Found
##   High           0      11258
##   Peak Found  5056          0
```

Once we have filtered the data to remove non-unique peptides and contaminants, we log transform ("normalise") the data for heavy and light abundances. In addition, we subtract the logged abundances light-heavy to yield logged abundance ratios.

When we re-draw the table of heavy and light sample counts, we don't have any "Not Found" values anymore. This was part of the exercise.

We have 16314 peptides that are present in both fractions. Analysing the difference in ratios between these two fractions are most likely to inform on whether or not they are enriched for RNA binding proteins.

```
# -------------------------------------------------------------
# Step 3 : Aggregating multiple peptides into a peptide group
# heavy = non-crosslinked
# light = crosslinked
# -------------------------------------------------------------

# Subset the data to include columns with useful metadata and abundance ratios
# Transform the dataframe using 'melt' so the values for heavy and light are in one column. Can use thi

subset.cols = norm.data[,c("master.protein.accessions","sequence","modifications","repl","uv","missing.
dim(subset.cols)
```

```
## [1] 16314     9
```

```
# Tried various methods of aggregation
# Using sequence and repeat columns to aggregate
# Using all columns but the abundance ratio columns to aggregate
# Using mean, median or max to aggregate
# Using ddply as an alternative to aggregate
# Note : Finally, settled on taking the mean of the logged values and using 'aggregate' function

# We have 15356 unique peptide groups across all samples
agg.mean = aggregate(cbind(light.log,heavy.log,norm.abundance.ratio)~sequence+repl,data=subset.cols,FUN
agg.pep.table = table(agg.mean$sequence,agg.mean$repl)
table(agg.mean$repl) # 275mJ, replicate 3 has a lot more peptide groups than other samples
```

```
##
## 150mJ.1 150mJ.2 150mJ.3 275mJ.1 275mJ.2 275mJ.3 400mJ.1 400mJ.2 400mJ.3
##    1802    1366    1590    1728    1275    2882    1894    1466    1353
```

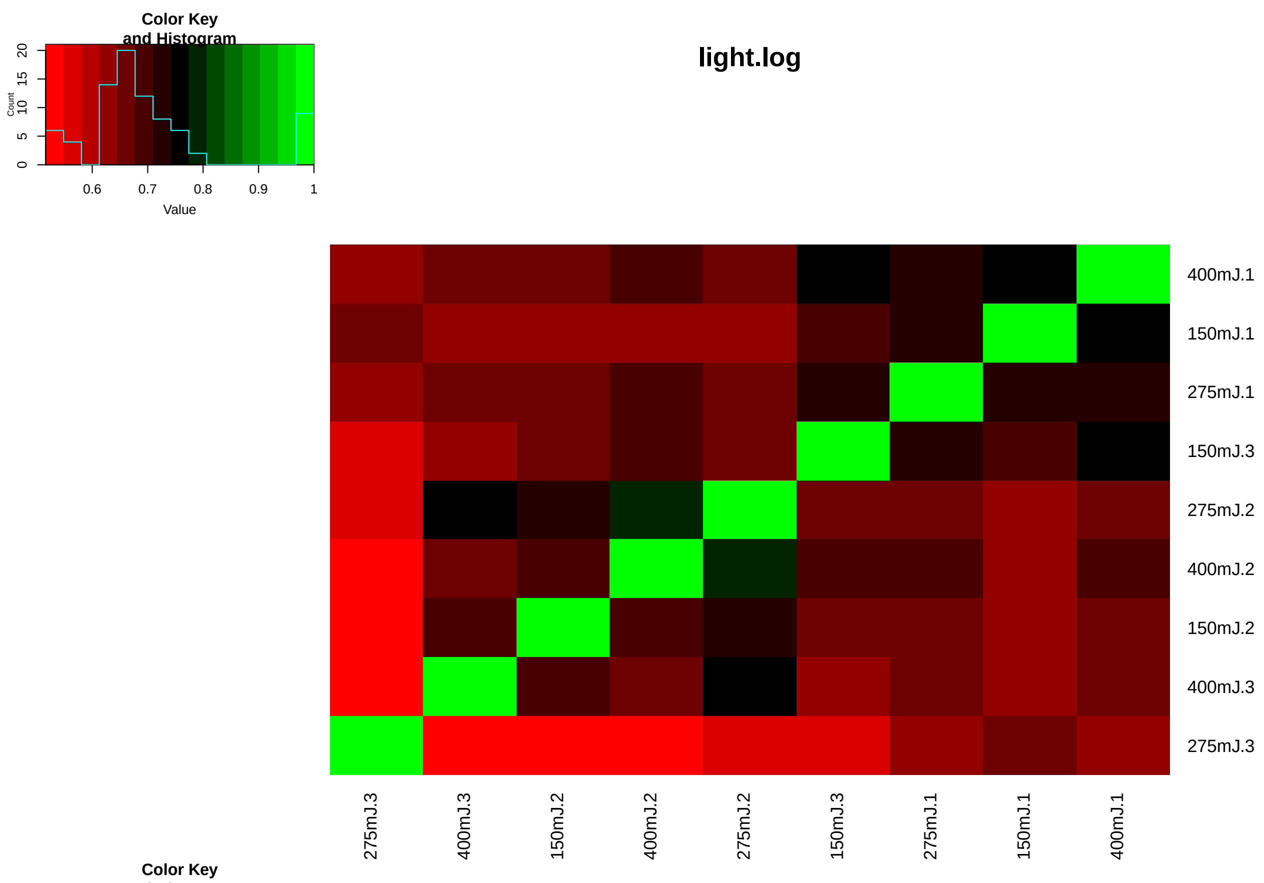
```
# Now that peptides have been aggregated into peptide groups, re-calculate the missing value table...
write.table(agg.pep.table, paste(outdir,"Aggregated-pepides-no-missing-values.txt",sep="/"),sep="\t",qu

# Will go with agg.mean for further analysis
agg = agg.mean

# I want to add protein annotations back to the aggregated data. Just want to make sure that one peptid
for(i in 1:nrow(agg)){
  agg$num.prot[i] = length(unique(subset.cols$master.protein.accessions[which(subset.cols$sequence == a
  agg$accessions[i] = paste(unique(subset.cols$master.protein.accessions[which(subset.cols$sequence == a
}

head(agg)
```

```
##                               sequence    repl light.log heavy.log
## 1                          AAAAAAALQAK 150mJ.1  27.91815  24.91056
## 2                            AAAETQSLR 150mJ.1  26.22839  22.15931
## 3                           AAAMANNLQK 150mJ.1  24.43873  20.25003
## 4 AAEAAPPTQEAQGETEPTEQAPDALEQAADTSR 150mJ.1  19.35638  19.07300
## 5                              AAGPISER 150mJ.1  22.39100  22.81245
## 6                     AAGPSLSHTSGGTQSK 150mJ.1  20.91634  17.96804
##   norm.abundance.ratio num.prot accessions
## 1            3.0075883        1     P36578
## 2            4.0690834        1     Q13523
## 3            4.1886989        1     Q14498
## 4            0.2833818        1     Q8N163
## 5           -0.4214498        1     Q15427
## 6            2.9483034        1     P27694
```

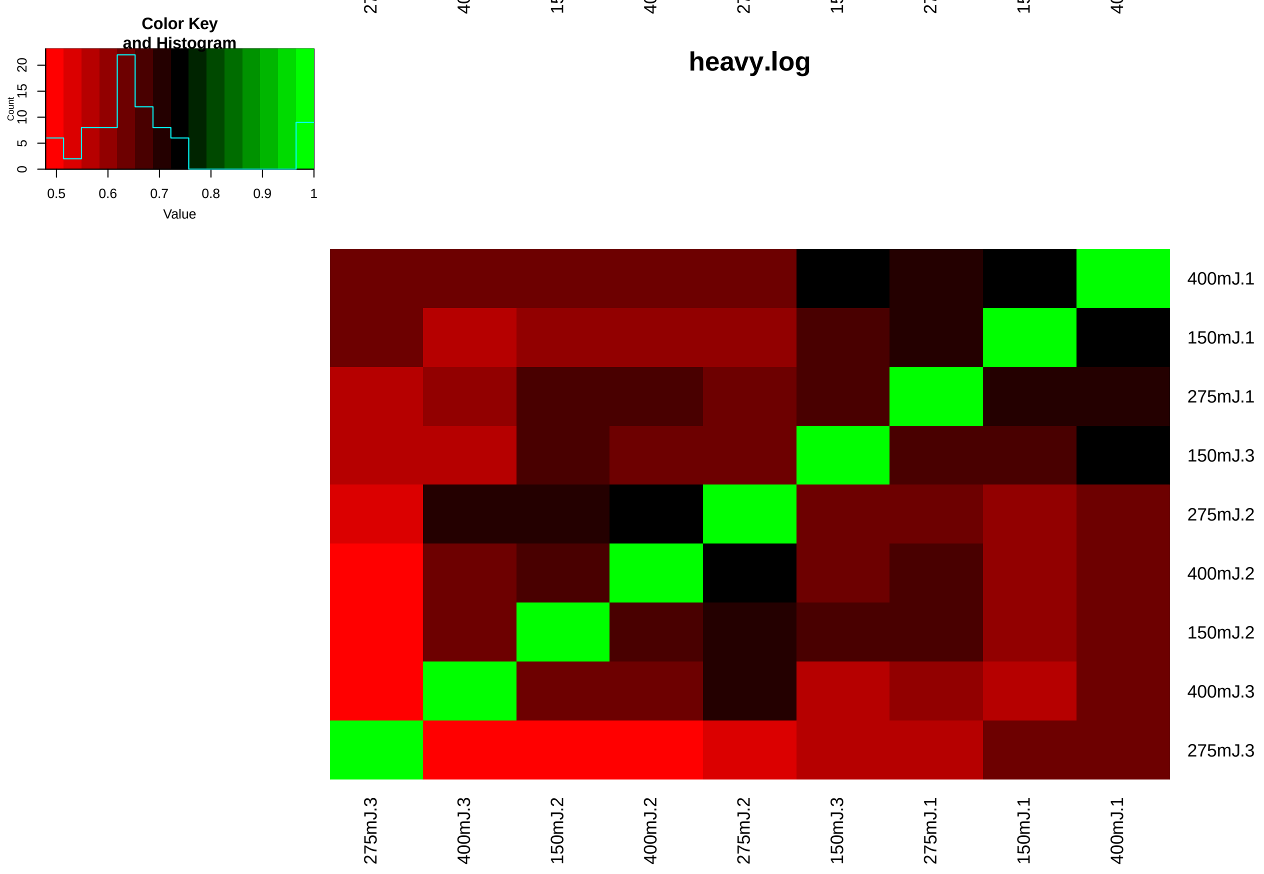
```
table(agg.mean$repl)
```

```
##
## 150mJ.1 150mJ.2 150mJ.3 275mJ.1 275mJ.2 275mJ.3 400mJ.1 400mJ.2 400mJ.3
##    1802    1366    1590    1728    1275    2882    1894    1466    1353
```

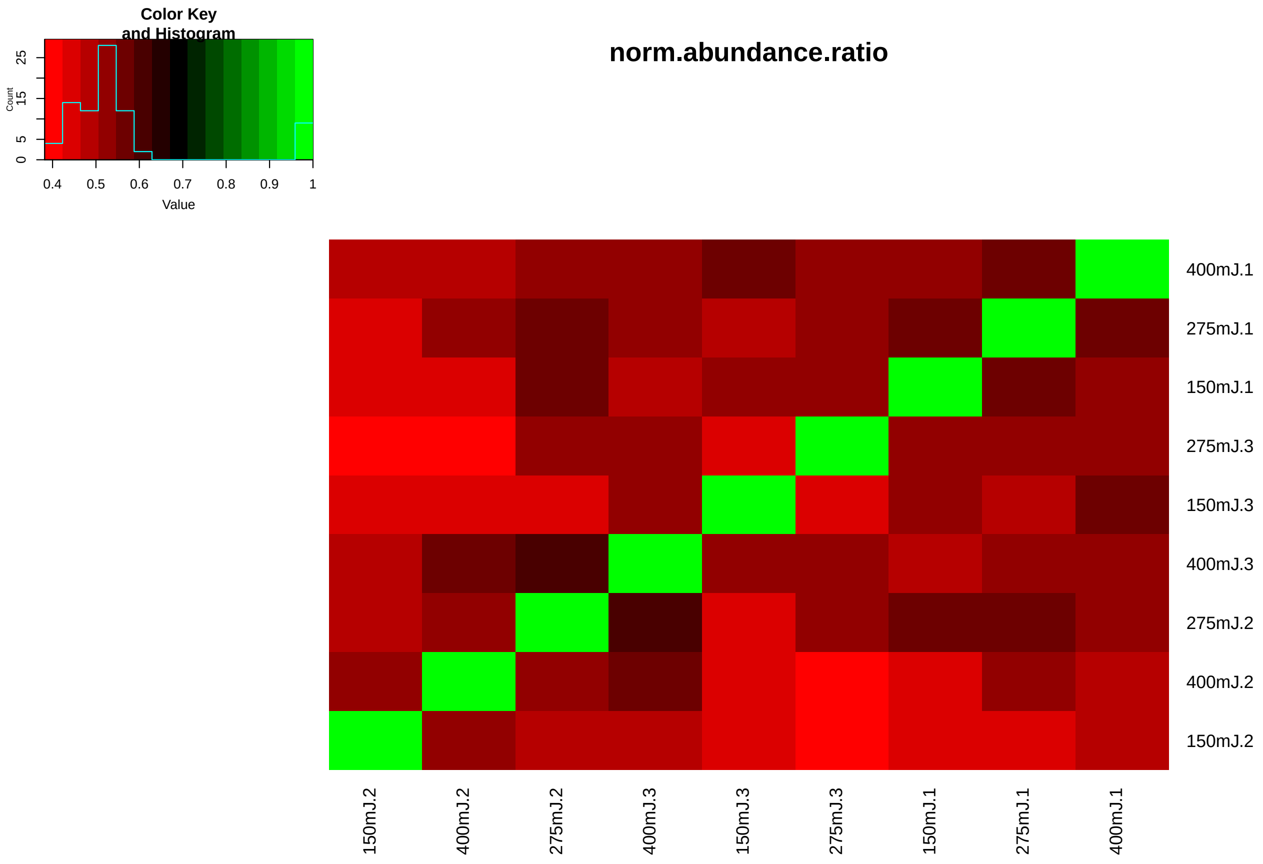
```
# Temporarily recast data into a matrix to calculate correlations

for (t in c("light.log","heavy.log","norm.abundance.ratio")){
  m = melt(agg,id.vars = c("sequence","repl"), measure.vars = t)
  m.cast = dcast(m, sequence~repl+variable, fun.aggregate = mean)
  cor.m = cor(m.cast[,2:ncol(m.cast)],use="pairwise.complete.obs")
  colnames(cor.m) = gsub(paste("_",t,sep=""),"",colnames(cor.m))
  rownames(cor.m) = gsub(paste("_",t,sep=""),"",rownames(cor.m))
  heatmap.2(cor.m,trace = "none", dendrogram="none",col="redgreen", main=t)
}
```



Color Key
and Histogram
Count
0
5
10
15
20
0.6
0.7
0.8
0.9
1
Value
light.log
400mJ.1
150mJ.1
275mJ.1
150mJ.3
275mJ.2
400mJ.2
150mJ.2
400mJ.3
275mJ.3
275mJ.3
400mJ.3
150mJ.2
400mJ.2
275mJ.2
150mJ.3
275mJ.1
150mJ.1
400mJ.1



Color Key
and Histogram
Count
0
5
10
15
20
0.5
0.6
0.7
0.8
0.9
1
Value
heavy.log
400mJ.1
150mJ.1
275mJ.1
150mJ.3
275mJ.2
400mJ.2
150mJ.2
400mJ.3
275mJ.3
275mJ.3
400mJ.3
150mJ.2
400mJ.2
275mJ.2
150mJ.3
275mJ.1
150mJ.1
400mJ.1



Looking at the correlations for 'heavy' and 'light' abundance values across all replicates, it looks like the correlation is more within experimental replicates i.e high for rep1 of 150mJ, 275mJ, 400mJ than between 150mJ.rep1 and 150mJ.rep2 and so on.

```
# -----------------------------------------------
# Step 4 : Aggregating multiple peptides into one protein
# heavy = non-crosslinked
# light = crosslinked
# -----------------------------------------------

# We have 1262 unique proteins across all samples
agg.prot = aggregate(cbind(light.log,heavy.log,norm.abundance.ratio)~accessions+repl,data=agg,FUN="mean
dim(agg.prot)
```

```
## [1] 5749    5
```

```
# Table of proteins vs samples - contingency to say which protein is present in which sample.
# Will help make overlaps
agg.prot.table = table(agg.prot$accessions,agg.prot$repl)
write.table(agg.prot.table, paste(outdir,"Aggregated-proteins-no-missing-values.txt",sep="/"),sep="\t",
table(agg.prot$repl)
```

```
##
## 150mJ.1 150mJ.2 150mJ.3 275mJ.1 275mJ.2 275mJ.3 400mJ.1 400mJ.2 400mJ.3
##     676     544     600     648     511     943     696     594     537
```

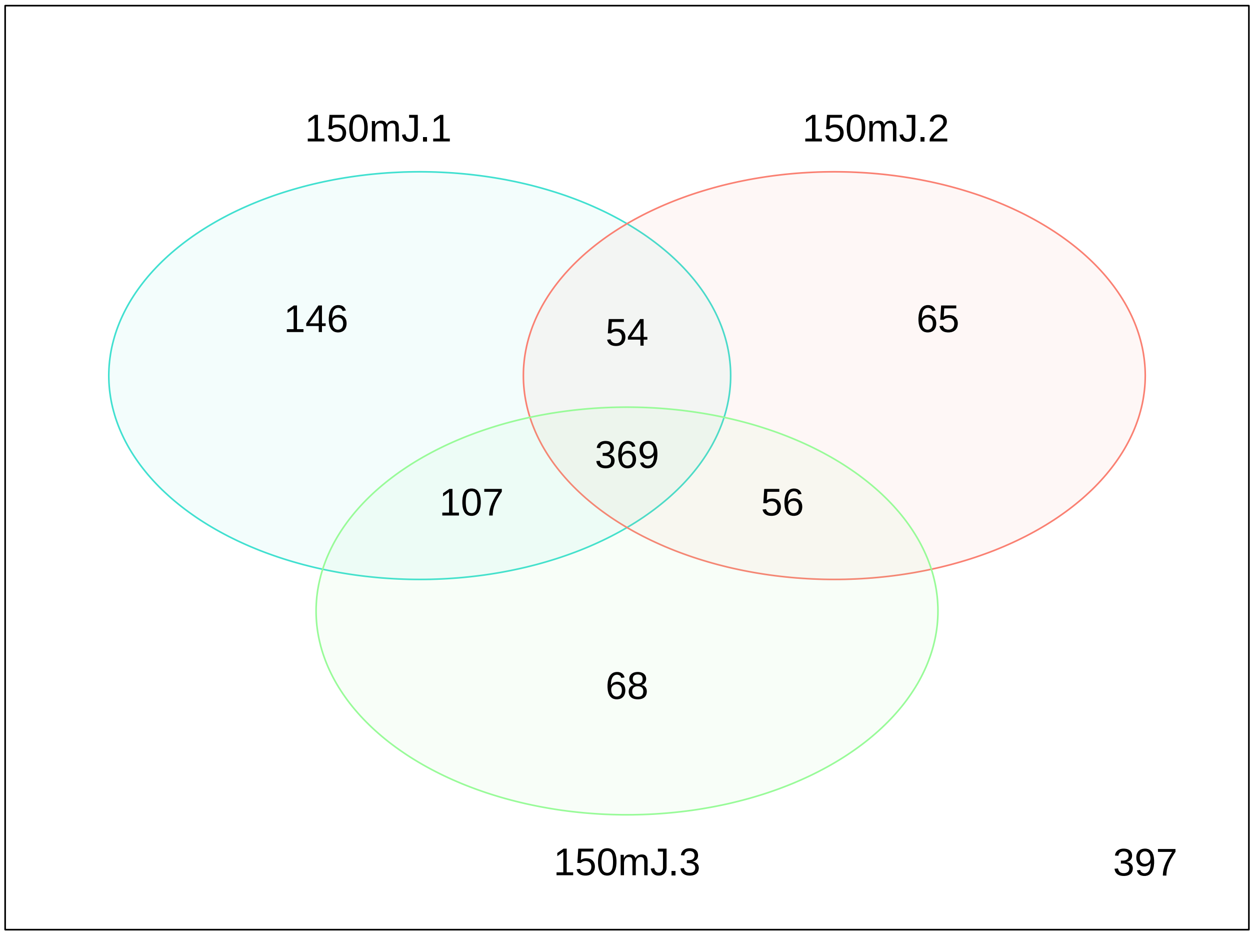
There seem to be on average, ~640 proteins per sample in this experiment. 275mJ, rep3 has an unusually high number at 943. The samples at 150mJ of UV exposure have ~605 proteins, 275mJ have on average 716 proteins and 400mJ have on average 610 proteins.

```
# -----------------------------------------------
# Step 5 : Looking for most commonly enriched proteins
# heavy = non-crosslinked
# light = crosslinked
# -----------------------------------------------

# First let us look at the intersects within and between replicates
prot.matrix = as.data.frame.matrix(agg.prot.table)
print(dim(prot.matrix))
```

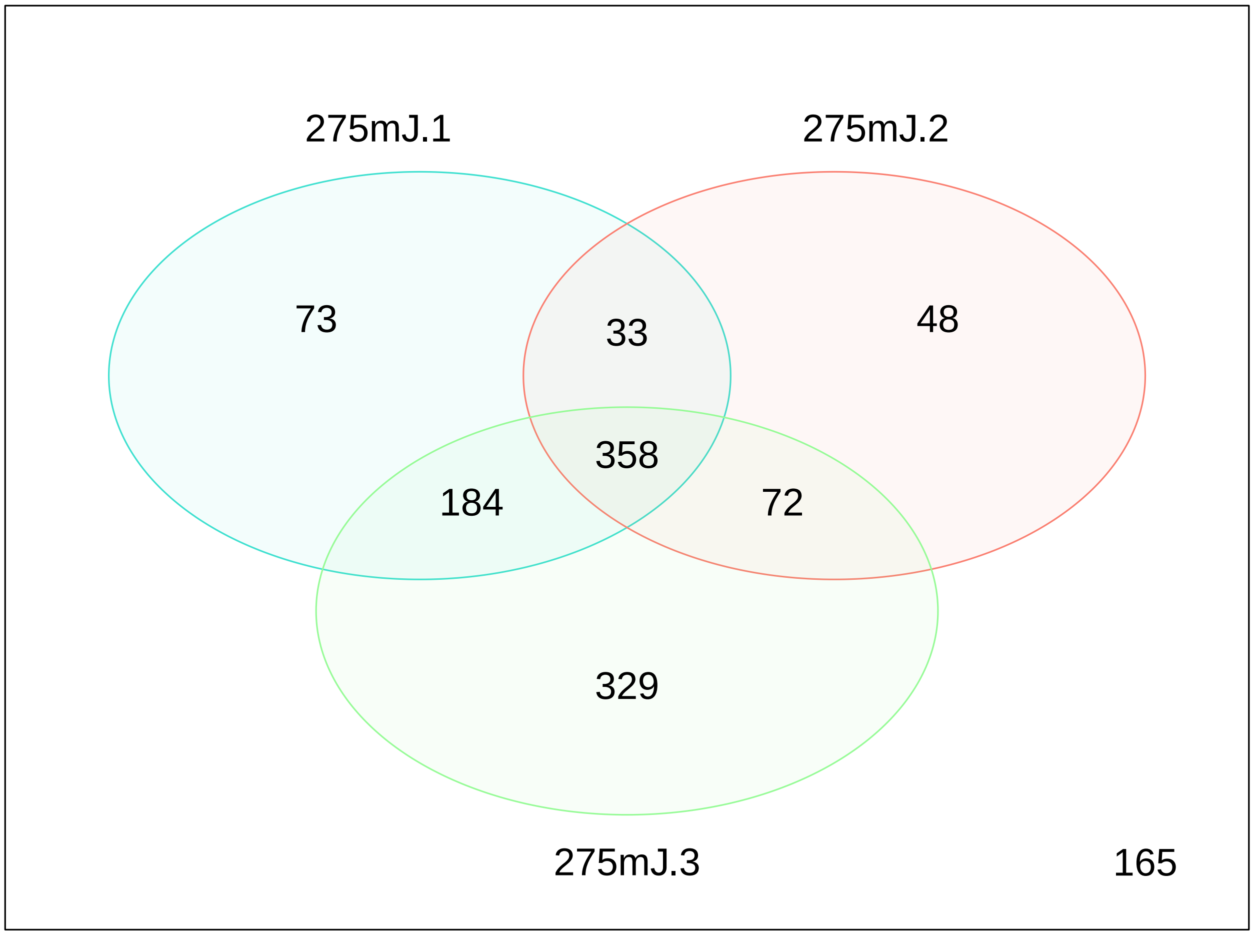
```
## [1] 1262    9
```

```
# Contains counts of overlap across 9 samples in various combinations
# Most intersections not very useful except that it tell us how many proteins overlap across all 9 samp
prot.venn = venn(prot.matrix,show.plot=F)
isect = attr(prot.venn,"intersections")

# table of intersections
isect.count = t(as.data.frame(lapply(isect,length)))
colnames(isect.count) = "Count"
write.table(isect.count, paste(outdir,"Count-of-protein-overlaps-across-various-samples.txt",sep="/"),s

# Looking at overlaps within each uv dose - the more useful intersection exercise
add.int = NULL

# Looping through each uv dosage triplicate - 1:3, 4:6, 7:9
# add.int contains all intersections for each triplicate
for(k in c(1,4,7)){
  print(k)
  prot.venn.tmp = venn(prot.matrix[,k:(k+2)],show.plot=F)
  vennDiagram(prot.matrix[,k:(k+2)],circle.col=c("turquoise", "salmon","palegreen"))
  add.int = c(add.int,attr(prot.venn.tmp,"intersections"))
}
```

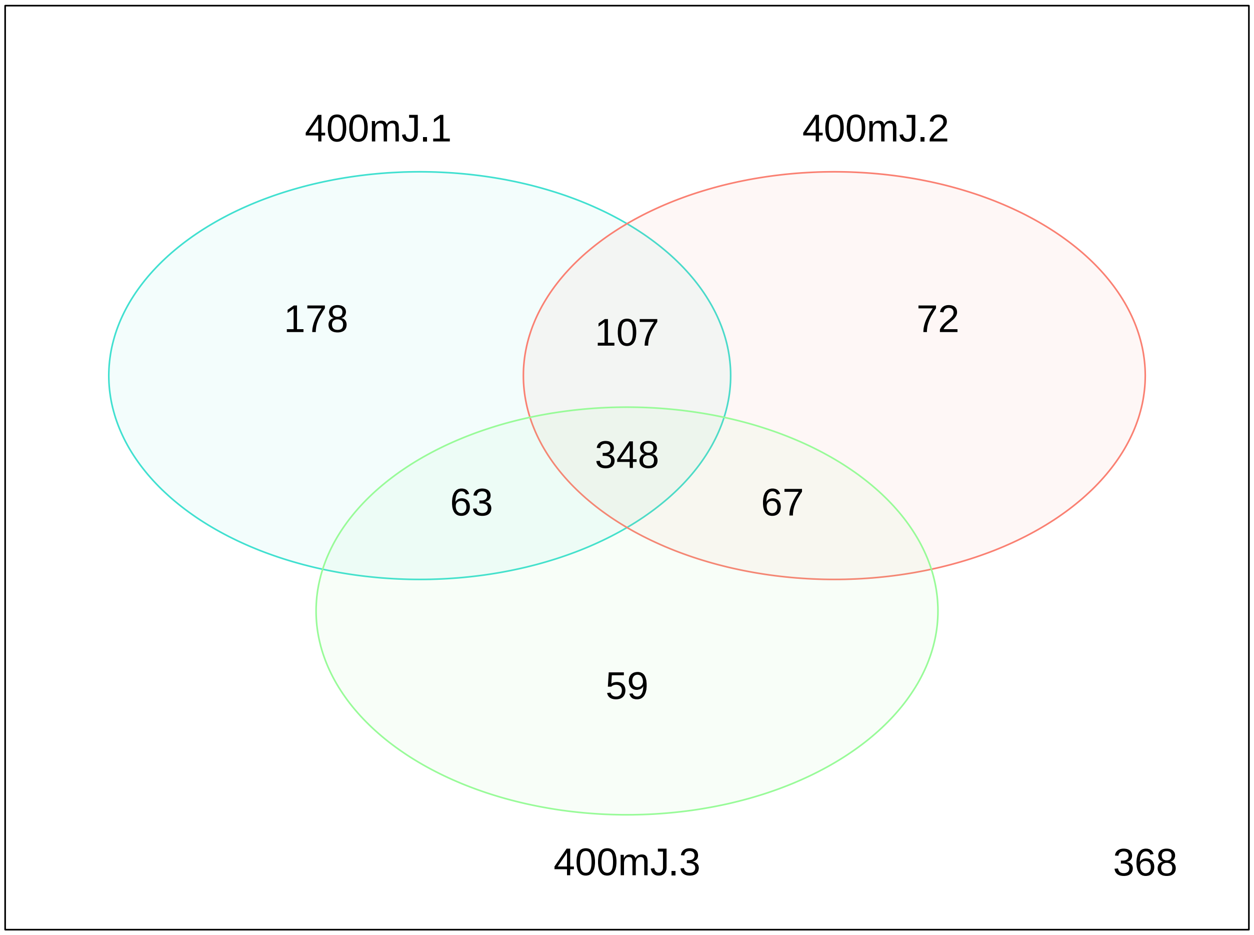
```
## [1] 1
```



```
## [1] 4
```



```
## [1] 7
```



```
write.table(t(as.data.frame(lapply(add.int,length))), paste(outdir,"Count-of-protein-overlaps-within-re
```

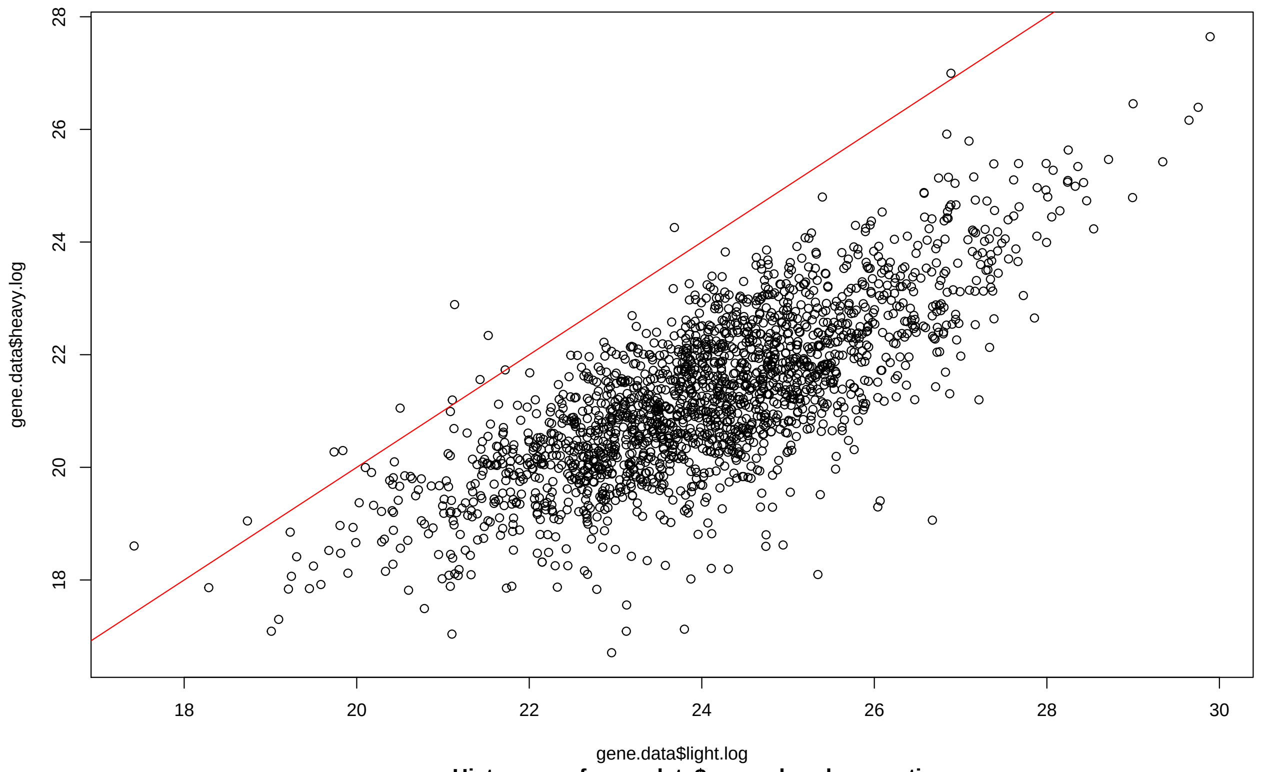
The venn diagrams show the overlap of proteins within each uv dosage across replicates. There are between 350 and 370 overlapping proteins within each UV dosage. Across all 9 replicates, there are 211 proteins that we can extract as shown below. The next step is to map these proteins to some functional annotations. We will map each interaction group separately to see what it yields. Will use 'clusterProfiler' to do this.

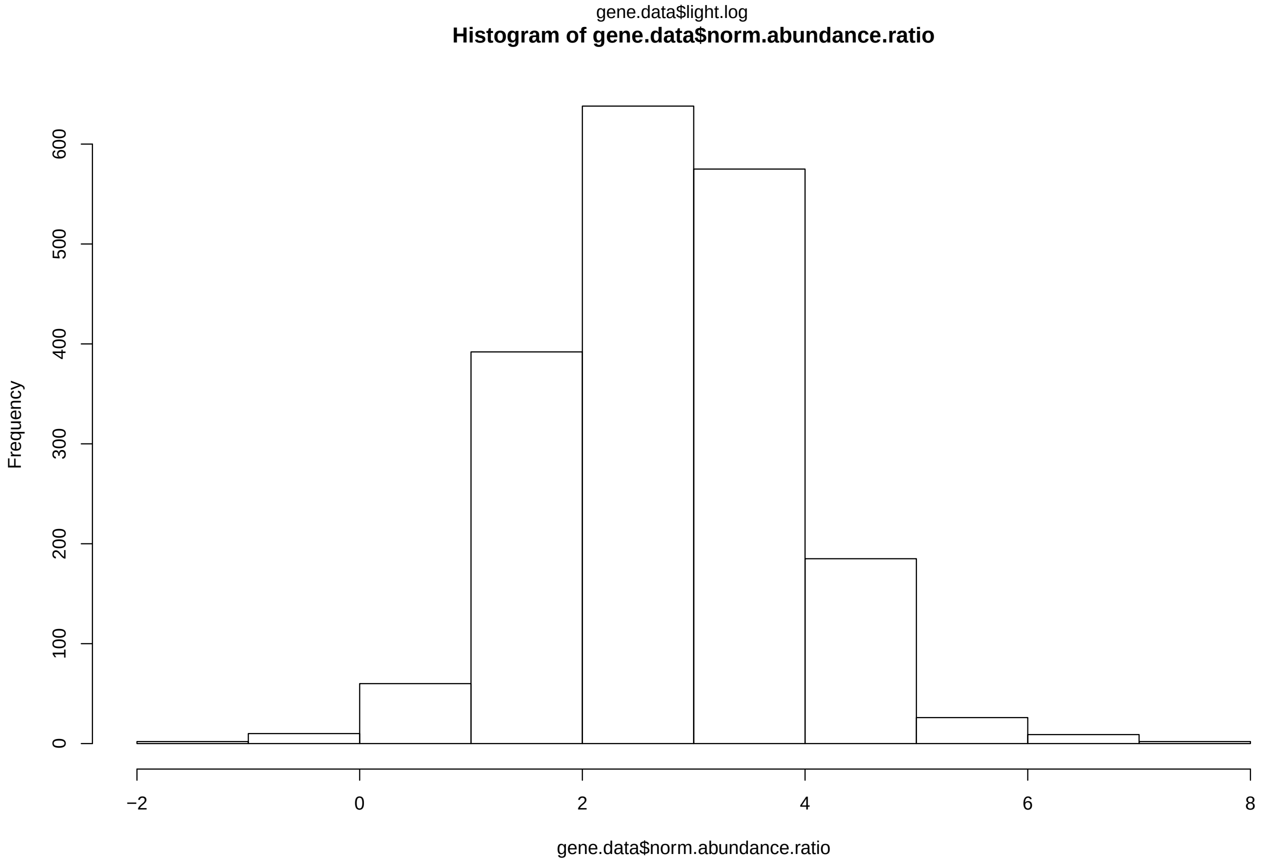
```
isect[280]
```

```
## $`150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3`
##   [1] "A0A024R4E5" "A0A087WUT6" "A0A087WVQ6" "A0A087X0X3" "A0A0A0MRV0"
##   [6] "A0A0C4DG17" "A0A0C4DG49" "A0A0D9SF53" "A0A0G2JNW7" "A0A0U1RRM4"
##  [11] "A8MXP9"     "E7EVA0"     "F5H2F4"     "F8W930"     "G8JLB6"
##  [16] "H3BLZ8"     "J3KPP4"     "J3KTA4"     "M0QYS1"     "O00203"
##  [21] "O00567"     "O15479"     "O43175"     "O43390"     "O43493"
##  [26] "O60506"     "O75369"     "O75400"     "O75533"     "O75534"
##  [31] "O76021"     "O95218"     "P02545"     "P02786"     "P04406"
##  [36] "P04792"     "P05023"     "P05556"     "P06748"     "P06756"
##  [41] "P07195"     "P07237"     "P07355"     "P07737"     "P07814"
##  [46] "P07900"     "P08238"     "P08621"     "P08670"     "P09429"
##  [51] "P09619"     "P09651"     "P0C7U0"     "P10809"     "P10909"
##  [56] "P11047"     "P11142"     "P11387"     "P11940"     "P13639"
##  [61] "P13667"     "P14618"     "P14625"     "P14866"     "P15880"
##  [66] "P16070"     "P16989"     "P18124"     "P18583"     "P18827"
##  [71] "P19338"     "P20700"     "P21399"     "P22314"     "P22626"
##  [76] "P23246"     "P23396"     "P23526"     "P26006"     "P26373"
##  [81] "P26641"     "P30101"     "P31942"     "P31948"     "P32004"
##  [86] "P35052"     "P35579"     "P35613"     "P35659"     "P36578"
##  [91] "P37802"     "P38646"     "P39023"     "P42704"     "P42892"
##  [96] "P43121"     "P46777"     "P46781"     "P46976"     "P47914"
## [101] "P48634"     "P49327"     "P49411"     "P49588"     "P49756"
## [106] "P50454"     "P50895"     "P50914"     "P50990"     "P51991"
## [111] "P52597"     "P53396"     "P55011"     "P55072"     "P55290"
## [116] "P55795"     "P60709"     "P61247"     "P61978"     "P62241"
## [121] "P62249"     "P62263"     "P62269"     "P62424"     "P62753"
## [126] "P62913"     "P62995"     "P67809"     "P78527"     "Q00839"
## [131] "Q01105"     "Q01130"     "Q01844"     "Q02878"     "Q05519"
## [136] "Q07065"     "Q07666"     "Q07954"     "Q08170"     "Q08211"
## [141] "Q08945"     "Q09666"     "Q12849"     "Q12906"     "Q13148"
## [146] "Q13151"     "Q13243"     "Q13263"     "Q13283"     "Q13641"
## [151] "Q13740"     "Q14103"     "Q14108"     "Q14152"     "Q14315"
## [156] "Q14444"     "Q14498"     "Q14690"     "Q15061"     "Q15084"
## [161] "Q15149"     "Q15233"     "Q15287"     "Q15717"     "Q15758"
## [166] "Q15904"     "Q16629"     "Q16658"     "Q5BKZ1"     "Q5T6F2"
## [171] "Q6PD62"     "Q6UVK1"     "Q7KZF4"     "Q7L2E3"     "Q7L4I2"
## [176] "Q7Z3B1"     "Q86SJ2"     "Q8N7H5"     "Q8NC51"     "Q8NE71"
## [181] "Q8WVC0"     "Q92541"     "Q92879"     "Q92945"     "Q96AE4"
## [186] "Q96I24"     "Q96KR1"     "Q96PK6"     "Q96T37"     "Q99700"
## [191] "Q99714"     "Q9BRL6"     "Q9BUQ8"     "Q9H307"     "Q9NR30"
## [196] "Q9NUM4"     "Q9NWH9"     "Q9NZB2"     "Q9P121"     "Q9UKM9"
## [201] "Q9UQ35"     "Q9UQ80"     "Q9Y2W1"     "Q9Y2X3"     "Q9Y383"
## [206] "Q9Y3Y2"     "Q9Y490"     "Q9Y4C8"     "Q9Y4L1"     "Q9Y520"
## [211] "X5DQS5"
```

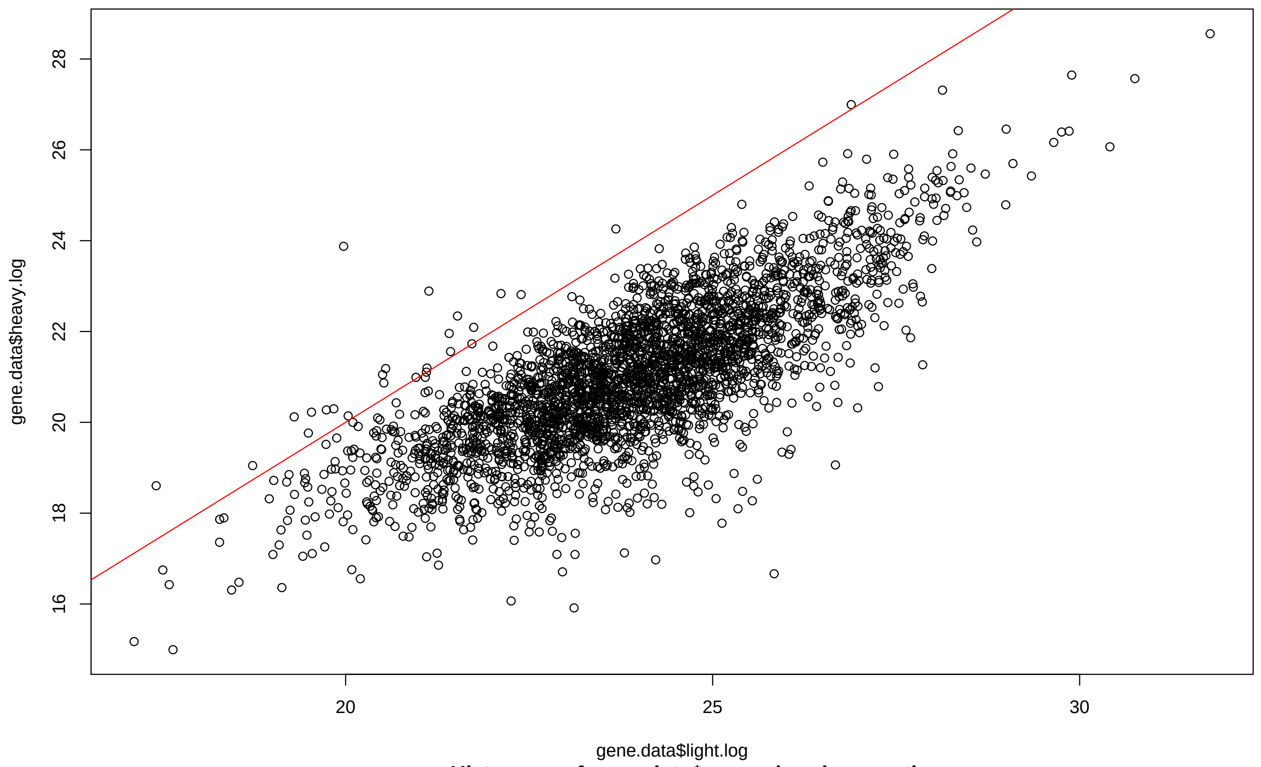
Displaying the `length(isect[280])` proteins that are enriched across all 9 replicate samples across 3 different UV dosages. We hope that this is the core set of RBPs we could use as a positive control later on in the project. Need to see what these proteins are and work out the rate of false positives.

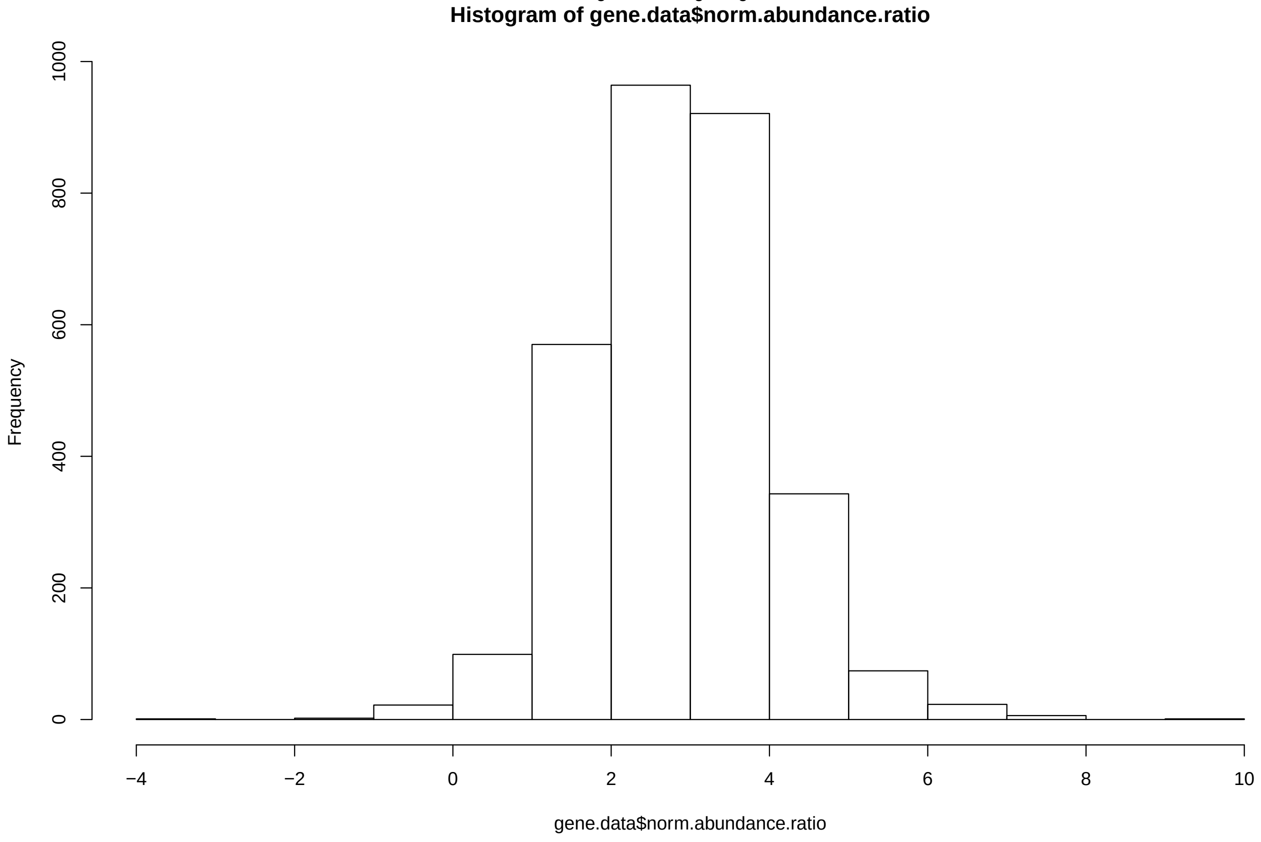
```
# -----------------------------------------------------------------------------------------------------------
# Step 6: Functional Enrichment
# Using KEGG pathways enrichment for the intersections of proteins within replicates
# Designate proteins as up(enriched) or down(not-enriched) in light:heavy relative to crosslink:non-cro
# Main interactions of interest are (1) across all 9 samples (n = 211) (2) Overlap within each triplica
# -----------------------------------------------------------------------------------------------------------

# A protein universe to use as background ??
prot.univ = bitr(unique(filt.1a$master.protein.accessions), fromType="UNIPROT", toType=c("ENTREZID"), O

# Calling the function 'enrichKEGG' on intersections of interest
across.9.kegg = enrichK(isect,280,agg.prot,0.05,outdir)
```



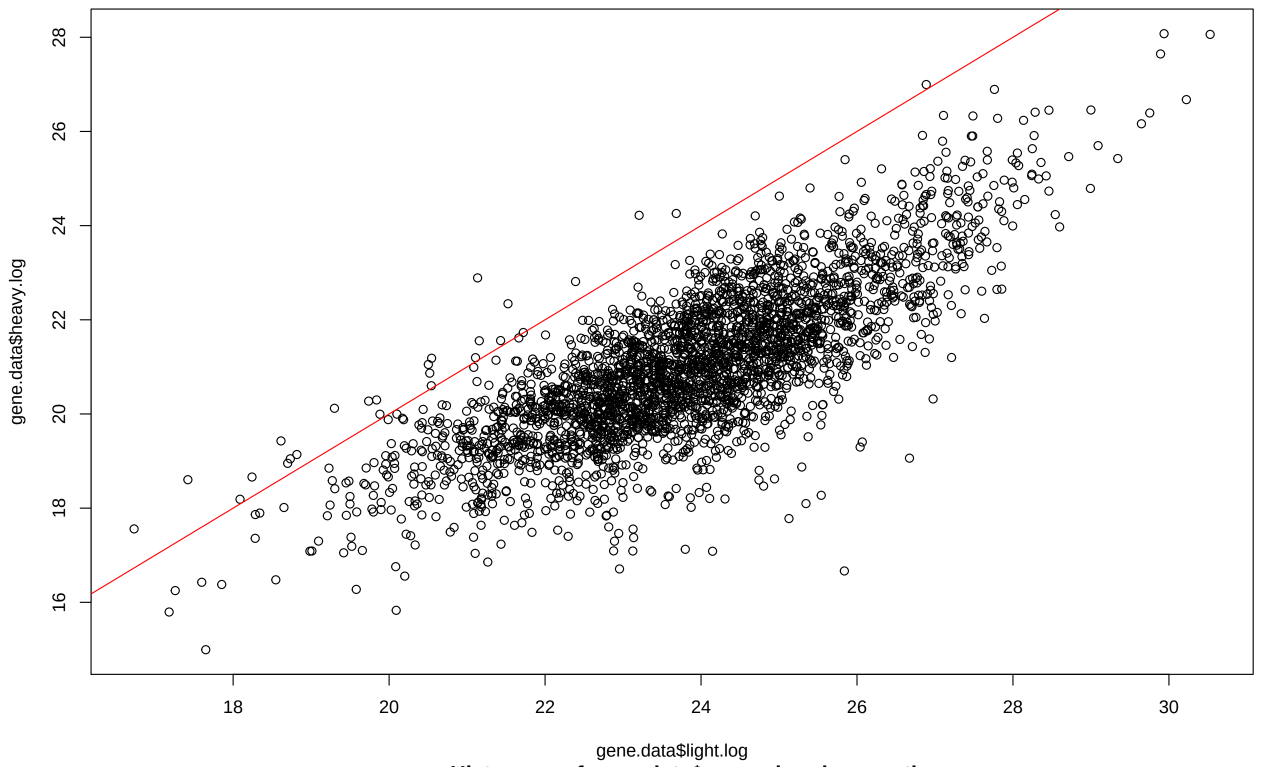


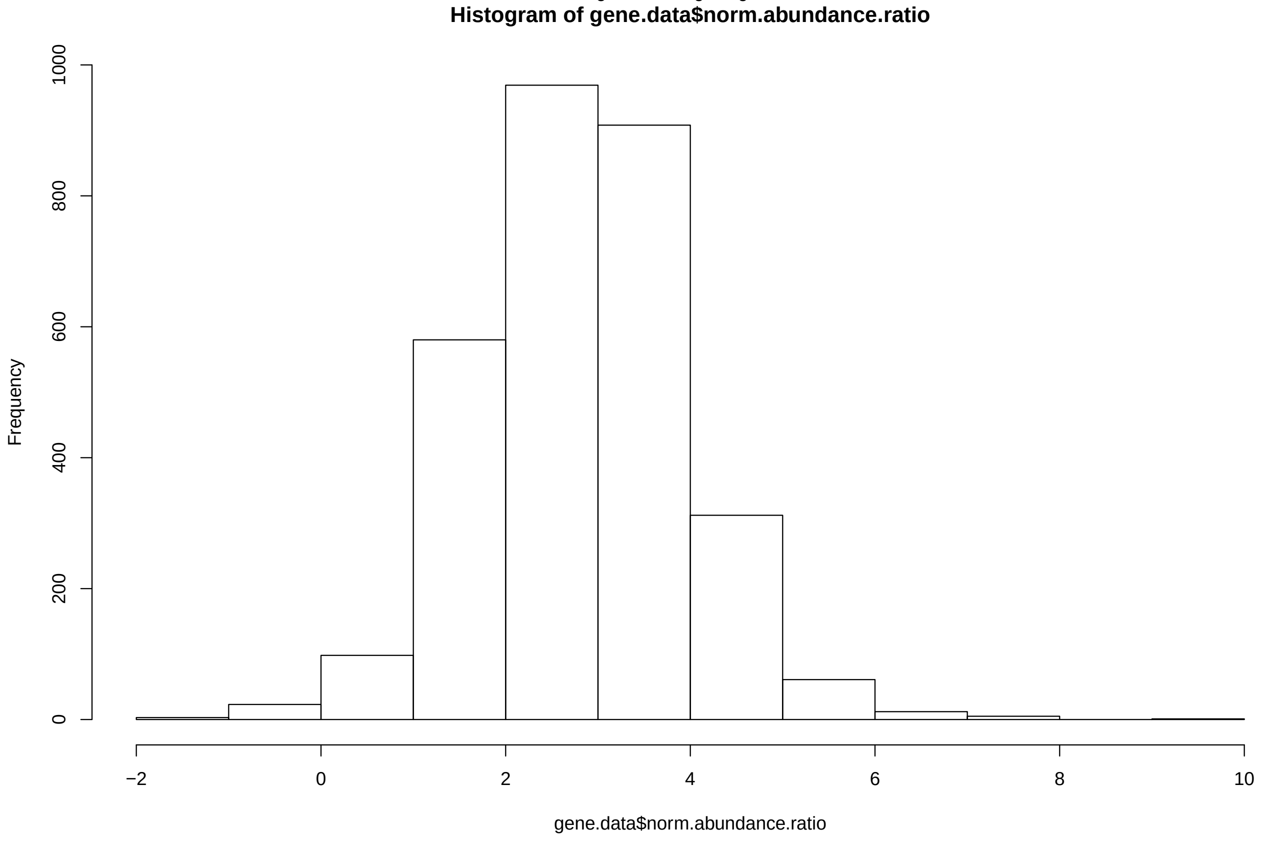
```
across.150mJ = enrichK(add.int,"150mJ.1:150mJ.2:150mJ.3",agg.prot,0.05,outdir)
```



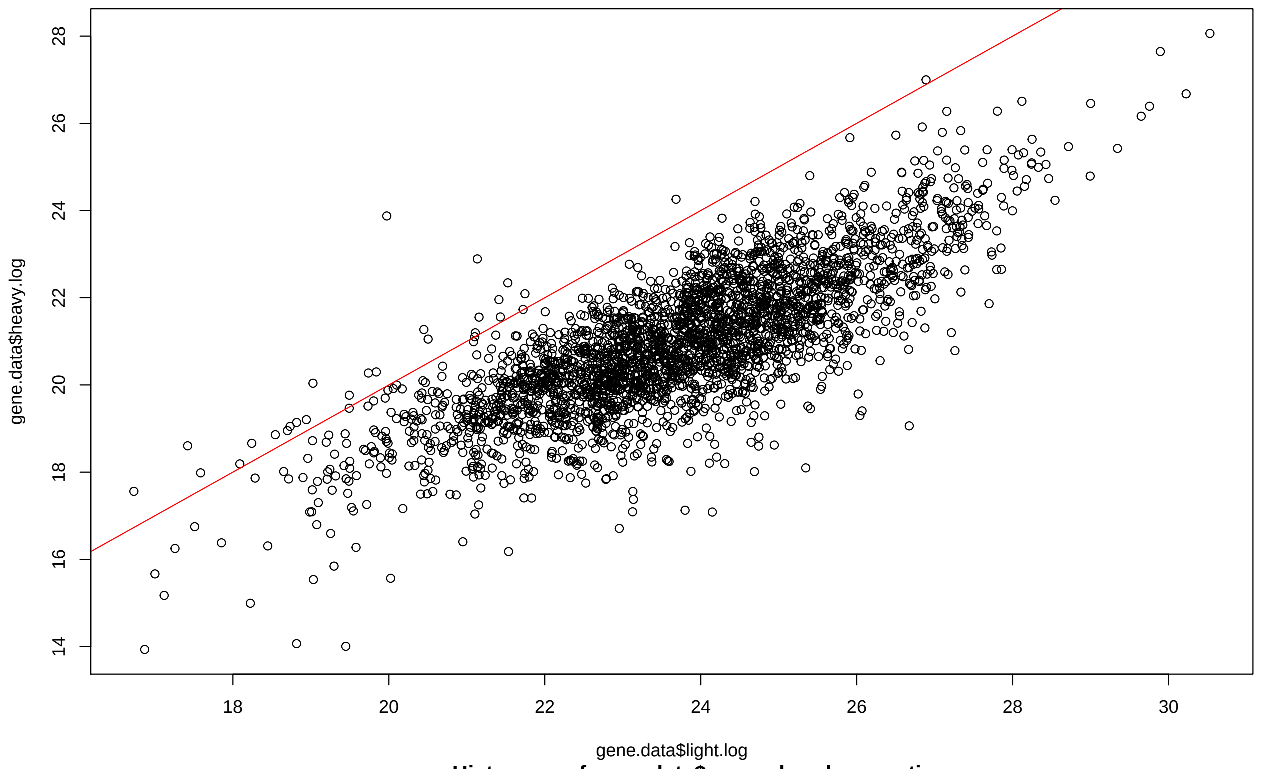


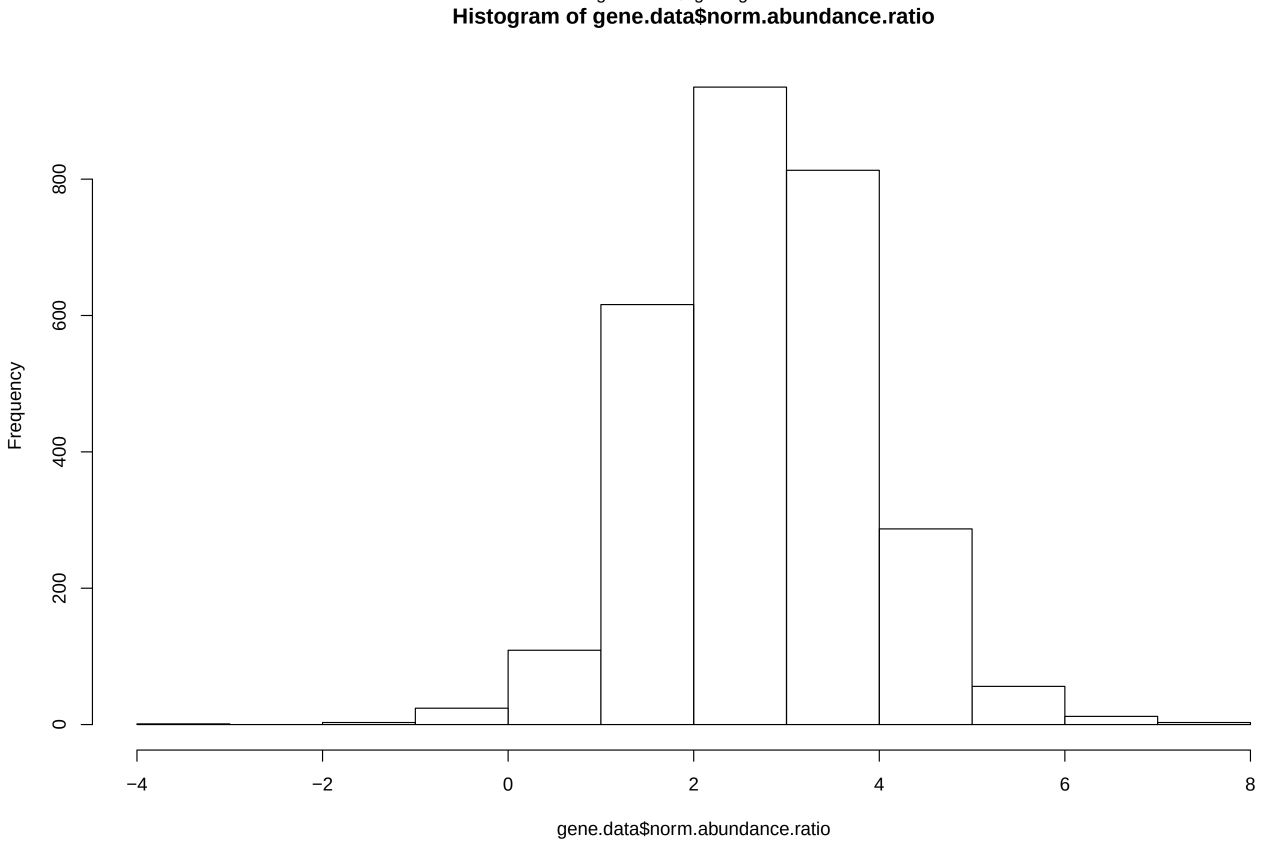
```
across.275mJ = enrichK(add.int,"275mJ.1:275mJ.2:275mJ.3",agg.prot,0.05,outdir)
```





```
across.400mJ = enrichK(add.int,"400mJ.1:400mJ.2:400mJ.3",agg.prot,0.05,outdir)
```





```
# Binding enriched KEGG pathway outcomes for all comparisons into one data frame for output
all.kegg = rbind(across.9.kegg,across.150mJ,across.275mJ,across.400mJ)
write.table(all.kegg, paste(outdir,"KEGG-enrichment-for-enriched-proteins.txt",sep="/"),sep="\t",quote=
```

Not sure what to define the protein “universe” as. Used all of the proteins in the aggregaed list but this is not sufficient to run the KEGG analysis (throws a “not suffiecient members in group” error. Need to read a bit more about the inner workings of enrichKEGG to see if this can be changed.

Meanwhile, the overlapping proteins across all samples are enriched for the terms “Ribosome”,“Spliceosome”,“Protein processing in ER”,“Cell adhesion molecules” etc... Rayner concerned about presence of proteoglycans as these could be unwanted members entering the interface. Experiments are underway to check this.

I have also done an enrichment for proteins that were common within triplicate and each UV dosage. Get very similar terms as before (which is expected) and a few extra. The 150mJ dosage has the most number of significant KEGG mappings of the three dosages. There are a few pathways that aren’t enriched in the crosslinked sample (Down) but majority are. There are instances where the term “splisosome” appears in both enriched and un-enriched categories but the genes that contribute to this KEGG term are different in the enriched and unenriched cases. Might be worth pursuing these genes that in the unenriched category - they are heterogeneous nuclear riboneucleo protein and small nuclear ribonucleoprotein, RNA helicase and splicing factor subunit.

The 275mJ dosage has a high number of histones which map to pathways such as Systemic lupus erythematosus, Viral carcinogenesis, ECM-receptor interaction and Alcoholism which are a bit odd. If you remember, 275mJ has on average more proteins per sample than the other two time points. Perhaps this isn’t the ideal UV dosage for the study.

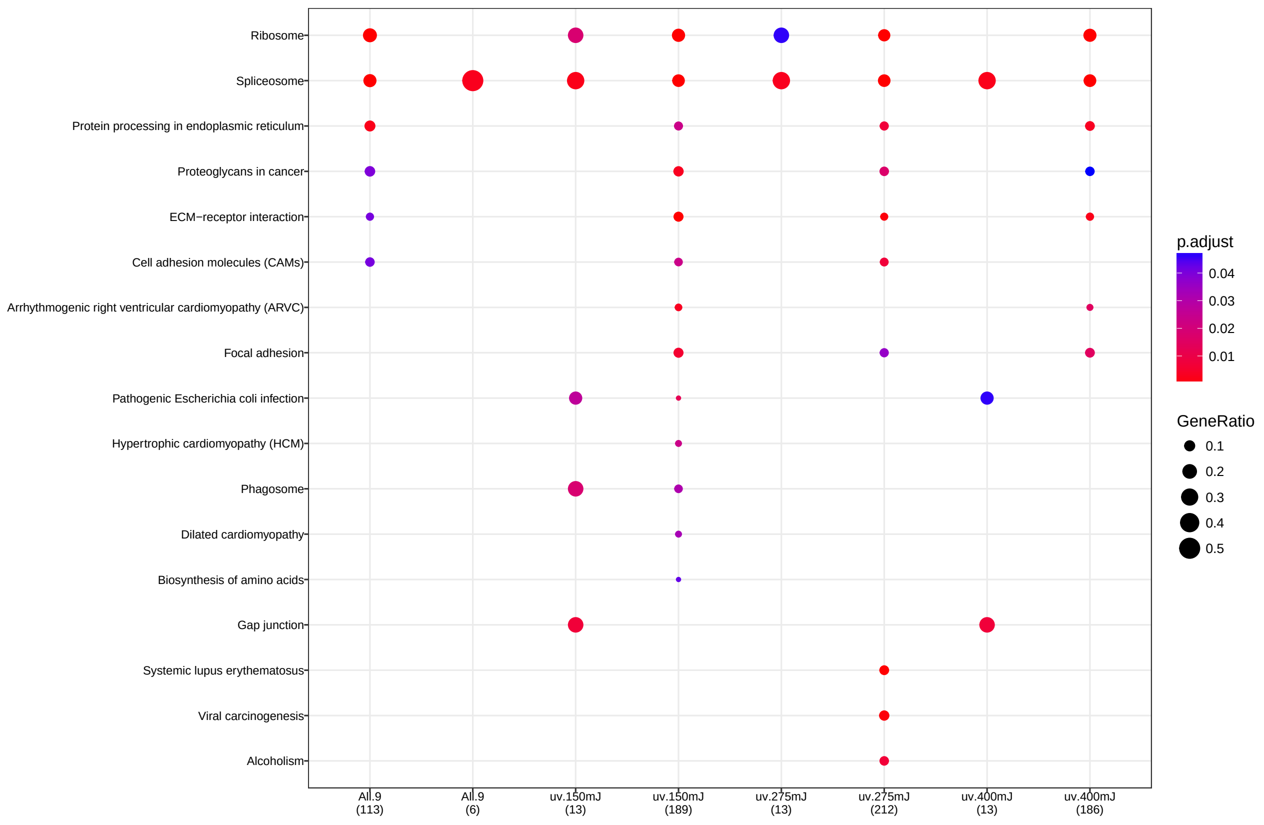
```r
#---------------------------------------------------------------------
# 07 : Plotting cluster membership
# Diagramatic representation of functional overlap/replicability
#---------------------------------------------------------------------
head(all.kegg)
```

```
##                ID                                 Description GeneRatio
## hsa03010 hsa03010                                    Ribosome    20/113
## hsa03040 hsa03040                                 Spliceosome    17/113
## hsa04141 hsa04141 Protein processing in endoplasmic reticulum    11/113
## hsa05205 hsa05205                     Proteoglycans in cancer    10/113
## hsa04512 hsa04512                    ECM-receptor interaction     6/113
## hsa04514 hsa04514              Cell adhesion molecules (CAMs)     8/113
##           BgRatio       pvalue     p.adjust       qvalue
## hsa03010 154/7251 1.482033e-13 1.956284e-11 1.747239e-11
## hsa03040 134/7251 1.733643e-11 1.144204e-09 1.021937e-09
## hsa04141 166/7251 5.021528e-05 2.209473e-03 1.973373e-03
## hsa05205 203/7251 1.196691e-03 3.949079e-02 3.527088e-02
## hsa04512  82/7251 1.678011e-03 4.046078e-02 3.613722e-02
## hsa04514 145/7251 1.839126e-03 4.046078e-02 3.613722e-02
##
## hsa03010 3921/6187/6129/6188/6137/6124/6122/6125/6203/6159/9045/6189/6202/6217/6208/6222/6130/6194/
## hsa03040          1655/55660/23451/6625/3178/3312/58517/220988/3190/6434/3192/6427/6429/6430/6432/10
## hsa04141                                          5034/3320/3326/3312/9601/7184/2923/7415/10970/101
## hsa05205                                                     1655/2317/3688/3685/960/6382/2817/60/
## hsa04512                                                                       3688/3685/3915/960/
## hsa04514                                                          5817/3688/3685/6382/3897/214/2571
##          Count dir
## hsa03010    20  Up
## hsa03040    17  Up
## hsa04141    11  Up
## hsa05205    10  Up
## hsa04512     6  Up
## hsa04514     8  Up
##                                                                        comparison
## hsa03010 150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3
## hsa03040 150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3
## hsa04141 150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3
## hsa05205 150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3
## hsa04512 150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3
## hsa04514 150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3
##
## hsa03010             A0A0C4DG17;P08865;A0A024R2P0;P15880;P18124;A8MUD9;P23396;A8K4C8;P26373;P36578;P3
## hsa03040 P17844;J3KTA4;O75400;Q05C41;B4DGZ4;O75533;P08621;A0A024QZD5;Q9UFS1;A0A024RB53;P09651;A0A024
## hsa04141                                                                     A0A024R8S5;P07237;P0790
## hsa05205
## hsa04512
## hsa04514
##
## hsa03010   RPSA;RPS2;RPL7;RPS3;RPL13;RPL4;RPL3;RPL5;RPS9;RPL29;RPL14;RPS3A;RPS8;RPS16;RPS14;RPS18;RP
## hsa03040 DDX5;PRPF40A;SF3B1;SNRNP70;HNRNPA1;HSPA8;RBM25;HNRNPA3;HNRNPK;TRA2B;HNRNPU;SRSF2;SRSF4;SRSF
## hsa04141                                         P4HB;HSP90AA1;HSP90AB1;HSPA8;PDIA4;HSP90B1;PDIA3;VC
## hsa05205                                                            DDX5;FLNB;ITGB1;ITGAV;CD44;SDC1;
## hsa04512                                                                              ITGB1;ITGAV;LA
## hsa04514                                                                   PVR;ITGB1;ITGAV;SDC1;L1CA
```

```r
# Need to create a 'compareClusterResult' object with the slots described below to be able to plot
# Normally, we'd feed in Entrez gene lists but at this stage we only have UniProt IDs.
# Too much of a pain to re-convert IDs, hence this hack.

# Results
# geneClusters
# fun (function)

# @Cluster
cluster = all.kegg$comparison
cluster = gsub("150mJ.1:150mJ.2:150mJ.3","uv.150mJ",cluster)
cluster = gsub("275mJ.1:275mJ.2:275mJ.3","uv.275mJ",cluster)
cluster = gsub("400mJ.1:400mJ.2:400mJ.3","uv.400mJ",cluster)
cluster = gsub("uv.150mJ:uv.275mJ:uv.400mJ","All.9",cluster)
cluster
```

```
##  [1] "All.9"    "All.9"    "All.9"    "All.9"    "All.9"    "All.9"
##  [7] "All.9"    "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ"
## [13] "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ"
## [19] "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ" "uv.150mJ"
## [25] "uv.150mJ" "uv.275mJ" "uv.275mJ" "uv.275mJ" "uv.275mJ" "uv.275mJ"
## [31] "uv.275mJ" "uv.275mJ" "uv.275mJ" "uv.275mJ" "uv.275mJ" "uv.275mJ"
## [37] "uv.275mJ" "uv.400mJ" "uv.400mJ" "uv.400mJ" "uv.400mJ" "uv.400mJ"
## [43] "uv.400mJ" "uv.400mJ" "uv.400mJ" "uv.400mJ" "uv.400mJ"
```

```r
all.kegg.compare = cbind(cluster,all.kegg[,1:10])
colnames(all.kegg.compare)[1] = "Cluster"

new.clusts = list(All.9=isect$`150mJ.1:150mJ.2:150mJ.3:275mJ.1:275mJ.2:275mJ.3:400mJ.1:400mJ.2:400mJ.3`

# Need to convert this to a cluster result object
clust.comp = new("compareClusterResult",compareClusterResult = all.kegg.compare,fun="enrichKEGG",geneClu
plot(clust.comp,type="dot", showCategory = 30,font.size=8)
```


Ribosome
Spliceosome
Protein processing in endoplasmic reticulum
Proteoglycans in cancer
ECM−receptor interaction
Cell adhesion molecules (CAMs)
Arrhythmogenic right ventricular cardiomyopathy (ARVC)
Focal adhesion
Pathogenic Escherichia coli infection
Hypertrophic cardiomyopathy (HCM)
Phagosome
Dilated cardiomyopathy
Biosynthesis of amino acids
Gap junction
Systemic lupus erythematosus
Viral carcinogenesis
Alcoholism
All.9 (113)
All.9 (6)
uv.150mJ (13)
uv.150mJ (189)
uv.275mJ (13)
uv.275mJ (212)
uv.400mJ (13)
uv.400mJ (186)
p.adjust
0.04
0.03
0.02
0.01
GeneRatio
0.1
0.2
0.3
0.4
0.5