binding_partners_final2.R

patrickjonker

2020-03-24

```r
##Description of the project

##I am rotating in a Cancer Biology lab that studies the protein Calreticulin (CALR).
##We see that mutant calreticulin drives a cancer phenotype in myeloid cells.
##We currently have proteomics data showing all of the proteins mutant calreticulin binds
##in the cell (which is a lot of proteins), but have done nothing with it. The code below, as well
##as code run prior (in python) attempts to take those data and overlap them with 15 datasets
##identifying genes in different pathways. This way, by the end of this exercise we will have
##novel information concerning which proteins mutant calreticulin is binding.
##The goals of this project include producing multiple graphs summarizing my findings, as well
##as a few summative tables indicating key genes (and their protein products) whose
##functions are potentially altered due to mutant calreticulin binding. A general note : most of
##the code below was run multiple times--once for each data set. I have tried to group these
##repeated bits of code clearly, so that you can read one line, get the picture, and move on.


library(tidyr)
library(tibble)
library(ggplot2)
library(Stack)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```r
## first find upr gene overlap data (below)
setwd("~/Documents/Graduate_School/Research/Elf_Research/Binding_Partners")
binding_partners <- read.delim("binding_partners.txt")
length(binding_partners)
```

```
## [1] 12
```

```r
View(binding_partners)
upr_proteins <- read.csv('upr_proteins.csv', header = FALSE)
view(upr_proteins)
v1 <- upr_proteins$V1
v2 <- binding_partners$gene
upr_overlay <- intersect(v1, v2)
##upr gene overlap data end



##NFkB signaling protein overlap
NFkB_raw <- read.csv("NFkB_proteins.csv", header = FALSE)

NFkB_proteins <- t(NFkB_raw)
view(NFkB_proteins)
NFkB_proteins <- as.data.frame(NFkB_proteins)
##change data from atomic list to recursive using as.data.frame function
v3 <- NFkB_proteins$V1
NFkB_overlap <- intersect(v2, v3)
view(NFkB_overlap)
##NFkB overlap data ends




##Amino Acid transporter overlap
AATransporter_raw <- read.csv("AATransporter_proteins.csv", header = FALSE)
view(AATransporter_raw)
AATransporter_proteins <- t(AATransporter_raw)
view(AATransporter_proteins)
AATransporter_proteins <- as.data.frame(AATransporter_proteins)
##change data from atomic list to recursive using as.data.frame function
v4 <- AATransporter_proteins$V1
AATransporter_overlap <- intersect(v2, v4)
view(AATransporter_overlap)
##Amino Acid transporter overlap end



##Antigen processing
Antigen_processing_raw <- read.csv("Antigen_processing_proteins.csv", header = FALSE)
view(Antigen_processing_raw)
Antigen_processing_proteins <- t(Antigen_processing_raw)
view(Antigen_processing_proteins)
Antigen_processing_proteins <- as.data.frame(Antigen_processing_proteins)
##change data from atomic list to recursive using as.data.frame function
v5 <- Antigen_processing_proteins$V1
Antigen_processing_overlap <- intersect(v2, v5)
view(Antigen_processing_overlap)
##antigen processing end
```

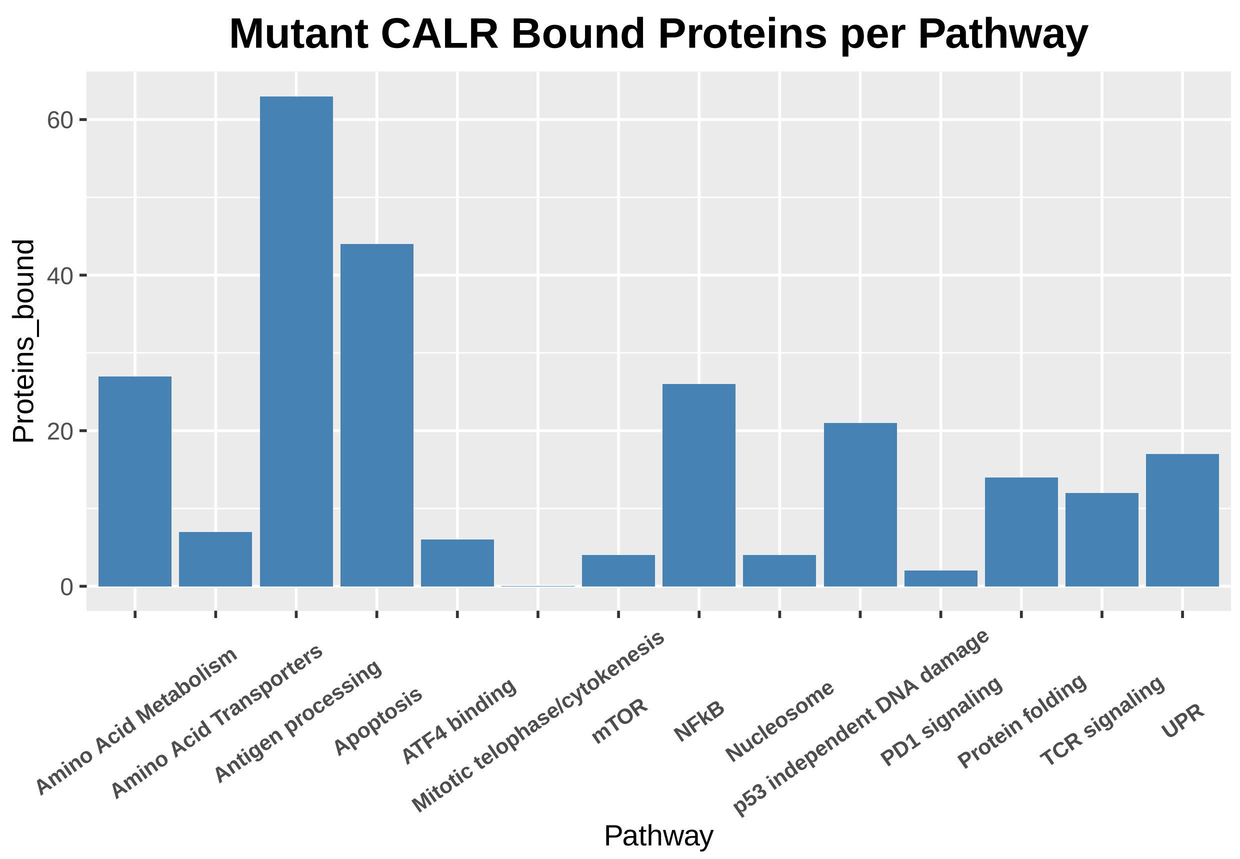
```r
##Apoptosis proteins start
Apoptosis_raw <- read.csv("Apoptosis_proteins.csv", header = FALSE)
view(Apoptosis_raw)
Apoptosis_proteins <- t(Apoptosis_raw)
view(Apoptosis_proteins)
Apoptosis_proteins <- as.data.frame(Apoptosis_proteins)
##change data from atomic list to recursive using as.data.frame function
v6 <- Apoptosis_proteins$V1
Apoptosis_overlap <- intersect(v2, v6)
view(Apoptosis_overlap)
##Apoptosis end




##
Protein_folding_raw <- read.csv("Protein_folding.csv", header = FALSE)
view(Protein_folding_raw)
Protein_folding <- t(Protein_folding_raw)
view(Protein_folding)
Protein_folding <- as.data.frame(Protein_folding)
##change data from atomic list to recursive using as.data.frame function
v7 <- Protein_folding$V1
Protein_folding_overlap <- intersect(v2, v7)
view(Protein_folding_overlap)
##





##PD1 proteins
PD1_raw <- read.csv("PD1_proteins.csv", header = FALSE)
view(PD1_raw)
PD1_proteins <- t(PD1_raw)
view(PD1_proteins)
PD1_proteins <- as.data.frame(PD1_proteins)
##change data from atomic list to recursive using as.data.frame function
v8 <- PD1_proteins$V1
PD1_protein_overlap <- intersect(v2, v8)
view(PD1_protein_overlap)
##end





##p53 independent DNA damage/repair proteins
p53_raw <- read.csv("p53_indep_DNA_damage.csv", header = FALSE)
view(p53_raw)
p53_indep_DNA_damage <- t(p53_raw)
view(p53_indep_DNA_damage)
```

```
p53_indep_DNA_damage<- as.data.frame(p53_indep_DNA_damage)
##change data from atomic list to recursive using as.data.frame function
v9 <- p53_indep_DNA_damage$V1
p53_indep_overlap <- intersect(v2, v9)
view(p53_indep_overlap)
##end





##nucleosome proteins
nucleosome_raw <- read.csv("nucleosome_proteins.csv", header = FALSE)
view(nucleosome_raw)
nucleosome_proteins <- t(nucleosome_raw)
view(nucleosome_proteins)
nucleosome_proteins <- as.data.frame(nucleosome_proteins)
##change data from atomic list to recursive using as.data.frame function
v10 <- nucleosome_proteins$V1
nucleosome_proteins_overlap <- intersect(v2, v10)
view(nucleosome_proteins_overlap)
##nucleosome proteins end





##
mTOR_raw <- read.csv("mTOR_proteins.csv", header = FALSE)
view(mTOR_raw)
mTOR_proteins <- t(mTOR_raw)
view(mTOR_proteins)
mTOR_proteins <- as.data.frame(mTOR_proteins)
##change data from atomic list to recursive using as.data.frame function
v11 <- mTOR_proteins$V1
mTOR_proteins_overlap <- intersect(v2, v11)
view(mTOR_proteins_overlap)
##






##Mitotic telophase/cytokenesis proteins
Mitotic_raw <- read.csv("Mitotic_proteins.csv", header = FALSE)
view(Mitotic_raw)
Mitotic_proteins <- t(Mitotic_raw)
view(Mitotic_proteins)
Mitotic_proteins <- as.data.frame(Mitotic_proteins)
##change data from atomic list to recursive using as.data.frame function
v12 <- Mitotic_proteins$V1
Mitotic_proteins_overlap <- intersect(v2, v12)
```

```r
view(Mitotic_proteins_overlap)
##end






##Amino Acid Metabolism
AAMetabolism_raw <- read.csv("AAMetabolism.csv", header = FALSE)
view(AAMetabolism_raw)
AAMetabolism_proteins <- t(AAMetabolism_raw)
view(AAMetabolism_proteins)
AAMetabolism_proteins <- as.data.frame(AAMetabolism_proteins)
##change data from atomic list to recursive using as.data.frame function
v13 <- AAMetabolism_proteins$V1
AAMetabolism_proteins_overlap <- intersect(v2, v13)
view(AAMetabolism_proteins_overlap)
##End




##Glycolysis proteins
Glycolysis_raw <- read.csv("Glycolysis_proteins.csv", header = FALSE)
view(Glycolysis_raw)
Glycolysis_proteins <- t(Glycolysis_raw)
view(Glycolysis_proteins)
Glycolysis_proteins <- as.data.frame(Glycolysis_proteins)
##change data from atomic list to recursive using as.data.frame function
v14 <- Glycolysis_proteins$V1
Glycolysis_proteins_overlap <- intersect(v2, v14)
view(Glycolysis_proteins_overlap)
##End




##TCR signaling proteins
TCR_raw <- read.csv("TCR_signaling.csv", header = FALSE)
view(TCR_raw)
TCR_signaling_proteins <- t(TCR_raw)
view(TCR_signaling_proteins)
TCR_signaling_proteins <- as.data.frame(TCR_signaling_proteins)
##change data from atomic list to recursive using as.data.frame function
v15 <- TCR_signaling_proteins$V1
TCR_proteins_overlap <- intersect(v2, v15)
view(TCR_proteins_overlap)
##

##ATF4 binding proteins
ATF4_raw <- read.csv("ATF4_binding.csv", header = FALSE)
view(ATF4_raw)
```

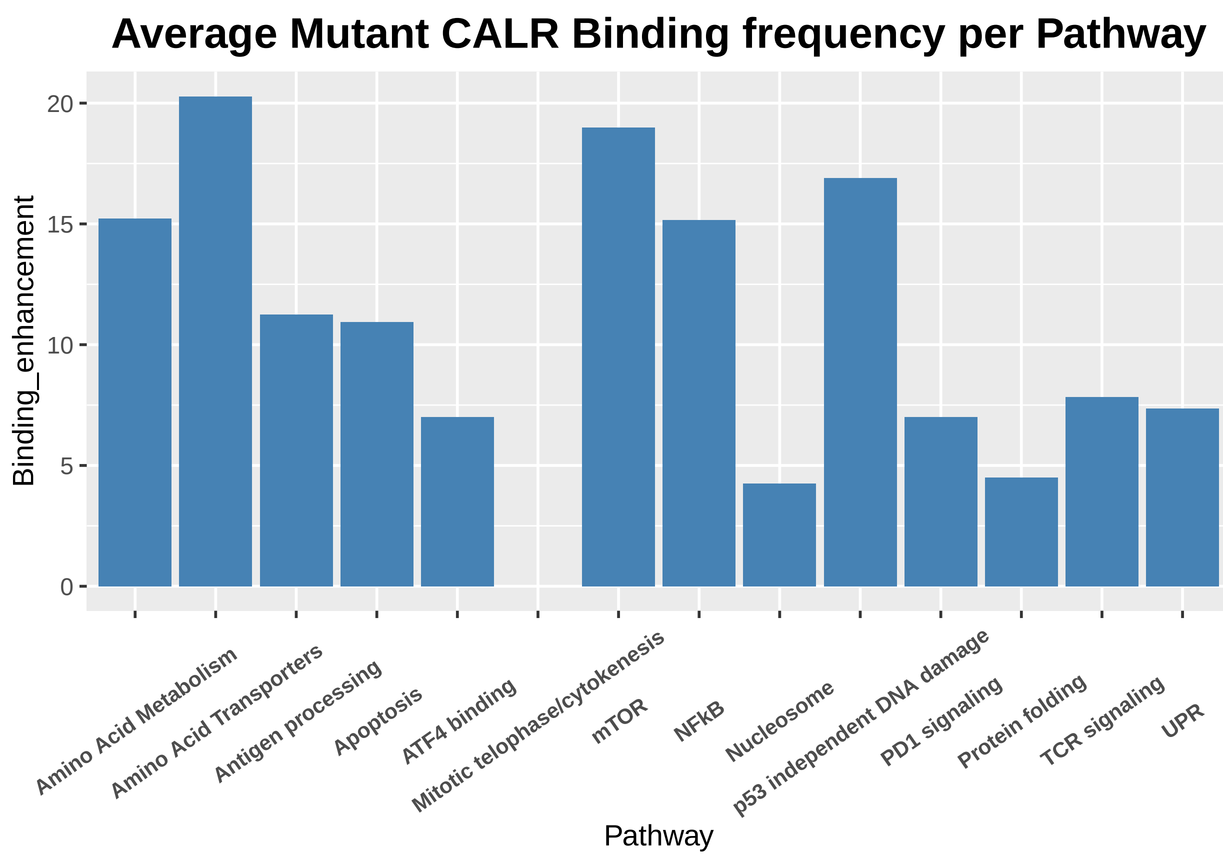
```r
ATF4_binding_proteins <- t(ATF4_raw)
view(ATF4_binding_proteins)
ATF4_binding_proteins <- as.data.frame(ATF4_binding_proteins)
##change data from atomic list to recursive using as.data.frame function
v16 <- ATF4_binding_proteins$V1
ATF4_binding_overlap <- intersect(v2, v16)
view(ATF4_binding_overlap)
##

##Match common genes with values to get frequencies by finding their place in the
##data frame using the "match" function


#TCR
TCR_numbers <- match(TCR_proteins_overlap, binding_partners$gene, nomatch = FALSE)
TCR_frequency <- mean((binding_partners$frequency[TCR_numbers]))

#Amino acid transport
AATransport_numbers <- match(AATransporter_overlap, binding_partners$gene, nomatch = FALSE)
AATransport_frequency <- mean(binding_partners$frequency[AATransport_numbers])

#Amino acid metabolism
AAMetabolism_numbers <- match(AAMetabolism_proteins_overlap, binding_partners$gene, nomatch = FALSE)
AAMetabolism_frequency <- mean(binding_partners$frequency[AAMetabolism_numbers])

#Antigen processing
Antigen_processing_numbers <- match(Antigen_processing_overlap, binding_partners$gene, nomatch = FALSE)
Antigen_processing_frequency <- mean(binding_partners$frequency[Antigen_processing_numbers])

#Apoptosis
Apoptosis_numbers <- match(Apoptosis_overlap, binding_partners$gene, nomatch = FALSE)
Apoptosis_frequency <- mean(binding_partners$frequency[Apoptosis_numbers])

#ATF4
ATF4_numbers <- match(ATF4_binding_overlap, binding_partners$gene, nomatch = FALSE)
ATF4_frequency <- mean(binding_partners$frequency[ATF4_numbers])

#Glycolysis
Glycolysis_numbers <- match(Glycolysis_proteins_overlap, binding_partners$gene, nomatch = FALSE)
Glycolysis_frequency <- mean(binding_partners$frequency[Glycolysis_numbers])

#Mitotic proteins
Mitotic_numbers <- match(Mitotic_proteins_overlap, binding_partners$gene, nomatch = FALSE)
Mitotic_frequency <- mean(binding_partners$frequency[Mitotic_numbers])

#mTOR
mTOR_numbers <- match(mTOR_proteins_overlap, binding_partners$gene, nomatch = FALSE)
mTOR_frequency <- mean(binding_partners$frequency[mTOR_numbers])

#NFkB
NFkB_numbers <- match(NFkB_overlap, binding_partners$gene, nomatch = FALSE)
NFkB_frequency <- mean(binding_partners$frequency[NFkB_numbers])
```

```r
#nucleosome formation proteins
nucleosome_numbers <- match(nucleosome_proteins_overlap, binding_partners$gene, nomatch = FALSE)
nucleosome_frequency <- mean(binding_partners$frequency[nucleosome_numbers])

#p53 independent DNA Damage and repair
p53_numbers <- match(p53_indep_overlap, binding_partners$gene, nomatch = FALSE)
p53_frequency <- mean(binding_partners$frequency[p53_numbers])

#PD1 signaling proteins
PD1_numbers <- match(PD1_protein_overlap, binding_partners$gene, nomatch = FALSE)
PD1_frequency <- mean(binding_partners$frequency[PD1_numbers])

#protein folding proteins
Protein_folding_numbers <- match(Protein_folding_overlap, binding_partners$gene, nomatch = FALSE)
Protein_folding_frequency <- mean(binding_partners$frequency[Protein_folding_numbers])

#UPR proteins (unfolded protein response)
UPR_numbers <- match(upr_overlay, binding_partners$gene, nomatch = FALSE)
UPR_frequency <- mean(binding_partners$frequency[UPR_numbers])
#done calculating average frequency of each pathway analyzed




##Calculate the percent of each pathway bound by mutant CALR by dividing the number of
## proteins bound by the total number of proteins in the pathway

AAMetabolism_ratio <- length(AAMetabolism_proteins_overlap)/length(AAMetabolism_proteins$V1)
AATransport_ratio <- length(AATransporter_overlap)/length(AATransporter_proteins$V1)
Antigen_ratio <- length(Antigen_processing_overlap)/length(Antigen_processing_proteins$V1)
Apoptosis_ratio <- length(Apoptosis_overlap)/length(Apoptosis_proteins$V1)
ATF4_ratio <- length(ATF4_binding_overlap)/length(ATF4_binding_proteins$V1)
Mitotic_ratio <- length(Mitotic_proteins_overlap)/length(Mitotic_proteins$V1)
mTOR_ratio <- length(mTOR_proteins_overlap)/length(mTOR_proteins$V1)
NFkB_ratio <- length(NFkB_overlap)/length(NFkB_proteins$NFkB_proteins)
nucleosome_ratio <- length(nucleosome_proteins_overlap)/length(nucleosome_proteins$V1)
p53_ratio <- length(p53_indep_overlap)/length(p53_indep_DNA_damage$V1)
PD1_ratio <- length(PD1_protein_overlap)/length(PD1_proteins$V1)
Protein_folding_ratio <- length(Protein_folding_overlap)/length(Protein_folding$V1)
TCR_ratio <- length(TCR_proteins_overlap)/length(TCR_signaling_proteins$V1)
UPR_ratio <- length(upr_overlay)/length(upr_proteins$V1)
##


##compile each data set into overlap database that contains number of proteins##
##bound as well as the average frequencies (fold binding enrichment compared to wild type)
##of those select genes##

overlap <- data.frame("Pathway", "Number of Genes", "Frequency", "Ratio",stringsAsFactors = FALSE)
overlap <- add_row(overlap, "X.Pathway." = "Amino Acid Metabolism",
                   "X.Number.of.Genes." = length(AAMetabolism_proteins_overlap),
                   "X.Frequency." = AAMetabolism_frequency, "X.Ratio." = AAMetabolism_ratio)
overlap <- add_row(overlap, "X.Pathway." = "Amino Acid Transporters",
                   "X.Number.of.Genes." = length(AATransporter_overlap),
                   "X.Frequency." = AATransport_frequency,"X.Ratio." = AATransport_ratio)
overlap <- add_row(overlap, "X.Pathway." = "Antigen processing",
                   "X.Number.of.Genes." = length(Antigen_processing_overlap),
                   "X.Frequency." = Antigen_processing_frequency, "X.Ratio." = Antigen_ratio)
overlap <- add_row(overlap, "X.Pathway." = "Apoptosis",
                   "X.Number.of.Genes." = length(Apoptosis_overlap),
                   "X.Frequency." = Apoptosis_frequency, "X.Ratio." = Apoptosis_ratio)
overlap <- add_row(overlap, "X.Pathway." = "ATF4 binding",
                   "X.Number.of.Genes." = length(ATF4_binding_overlap),
                   "X.Frequency." = ATF4_frequency, "X.Ratio." = ATF4_ratio)
overlap <- add_row(overlap, "X.Pathway." = "Mitotic telophase/cytokenesis",
                   "X.Number.of.Genes." = length(Mitotic_proteins_overlap),
                   "X.Frequency." = Mitotic_frequency, "X.Ratio." = Mitotic_ratio)
overlap <- add_row(overlap, "X.Pathway." = "mTOR",
                   "X.Number.of.Genes." = length(mTOR_proteins_overlap),
                   "X.Frequency." = mTOR_frequency, "X.Ratio." = mTOR_ratio)
overlap <- add_row(overlap, "X.Pathway." = "NFkB",
                   "X.Number.of.Genes." = length(NFkB_overlap),
                   "X.Frequency." = NFkB_frequency, "X.Ratio." = NFkB_ratio)
overlap <- add_row(overlap, "X.Pathway." = "Nucleosome",
                   "X.Number.of.Genes." = length(nucleosome_proteins_overlap),
                   "X.Frequency." = nucleosome_frequency, "X.Ratio." = nucleosome_ratio)
overlap <- add_row(overlap, "X.Pathway." = "p53 independent DNA damage",
                   "X.Number.of.Genes." = length(p53_indep_overlap),
                   "X.Frequency." = p53_frequency, "X.Ratio." = p53_ratio)
overlap <- add_row(overlap, "X.Pathway." = "PD1 signaling",
                   "X.Number.of.Genes." = length(PD1_protein_overlap),
                   "X.Frequency." = PD1_frequency, "X.Ratio." = PD1_ratio)
overlap <- add_row(overlap, "X.Pathway." = "Protein folding",
                   "X.Number.of.Genes." = length(Protein_folding_overlap),
                   "X.Frequency." = Protein_folding_frequency, "X.Ratio." = Protein_folding_ratio)
overlap <- add_row(overlap, "X.Pathway." = "TCR signaling",
                   "X.Number.of.Genes." = length(TCR_proteins_overlap),
                   "X.Frequency." = TCR_frequency, "X.Ratio." = TCR_ratio)
overlap <- add_row(overlap, "X.Pathway." = "UPR",
                   "X.Number.of.Genes." = length(upr_overlay),
                   "X.Frequency." = UPR_frequency,"X.Ratio." = UPR_ratio)
overlap <- overlap[c(-1),]


##code below makes bar graph for number of proteins bound for each pathway##
Proteins_bound <- as.numeric(overlap$X.Number.of.Genes.)
Pathway <- overlap$X.Pathway.
figure1 <- ggplot(data = overlap, aes(x = Pathway, y = Proteins_bound)) +
  geom_col(fill = "steelblue") + theme(plot.title = element_text(family = "Helvetica",
                                                                 face = "bold",
                                                                 hjust = 0.5, size = 16),
                                       axis.text.x = element_text(angle = 35, vjust = 0.5, size = 8,
                                                                  face = "bold")) +
  ggtitle("Mutant CALR Bound Proteins per Pathway")
figure1
```

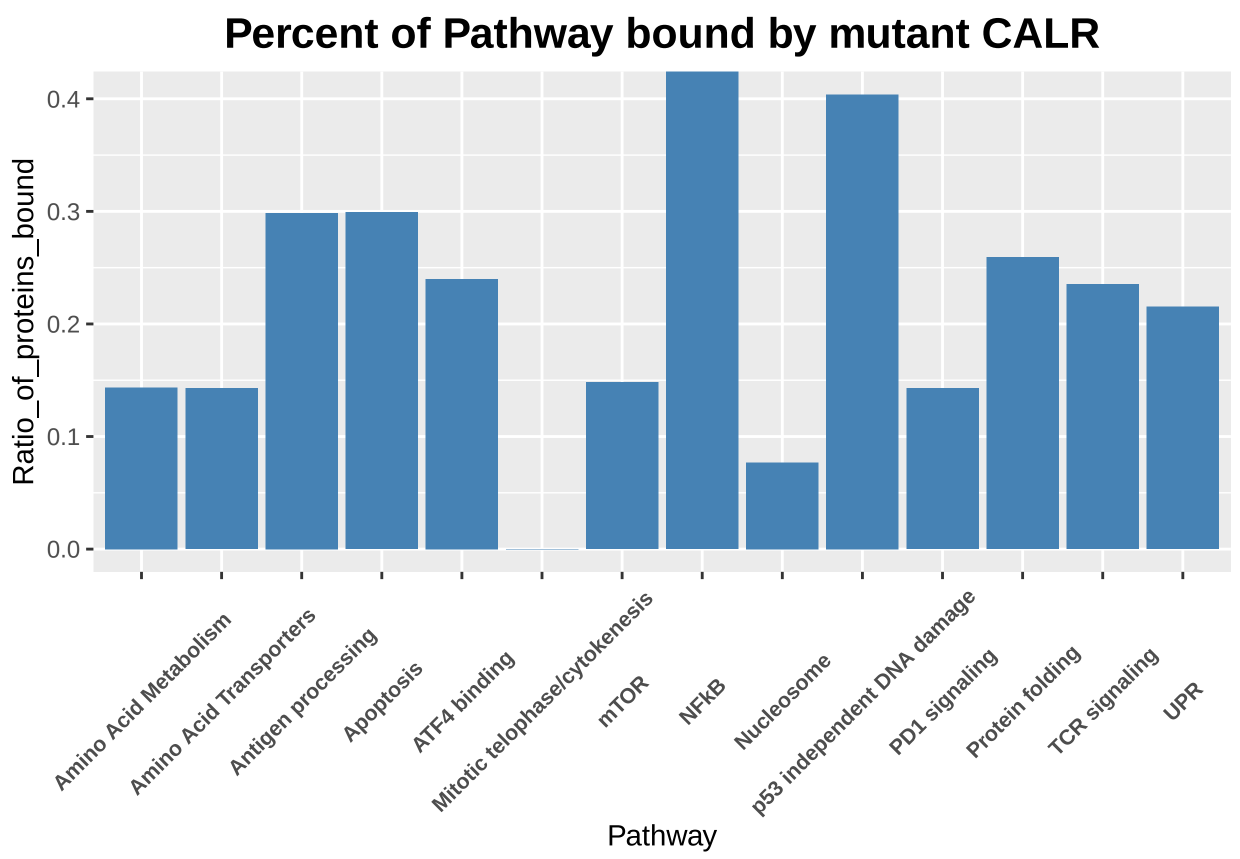


```
##bar graph for average binding enhancement (frequency) for each pathway##
Binding_enhancement <- as.numeric(overlap$X.Frequency.)
figure2 <- ggplot(data = overlap, aes(x = Pathway, y = Binding_enhancement)) +
  geom_col(fill = "steelblue") + theme(plot.title = element_text(family = "Helvetica", face = "bold",
                                                                  hjust = 0.5, size = 16),
                                       axis.text.x = element_text(angle = 35, vjust = 0.5, size = 8, fa
  ggtitle("Average Mutant CALR Binding frequency per Pathway")
figure2
```

```
## Warning: Removed 1 rows containing missing values (position_stack).
```



```r
##Bar graph showing ratio of proteins bound/ total proteins in pathway##
Ratio_of_proteins_bound <- as.numeric(overlap$X.Ratio.)
figure3 <- ggplot(data = overlap, aes(x = Pathway, y = Ratio_of_proteins_bound)) +
  geom_col(fill = "steelblue") + theme(plot.title = element_text(family = "Helvetica", face = "bold",
                                                                  hjust = 0.5, size = 16),
                                       axis.text.x = element_text(angle = 45, vjust = 0.5,
                                                                   size = 8, face = "bold")) +
  ggtitle("Percent of Pathway bound by mutant CALR")
figure3
```

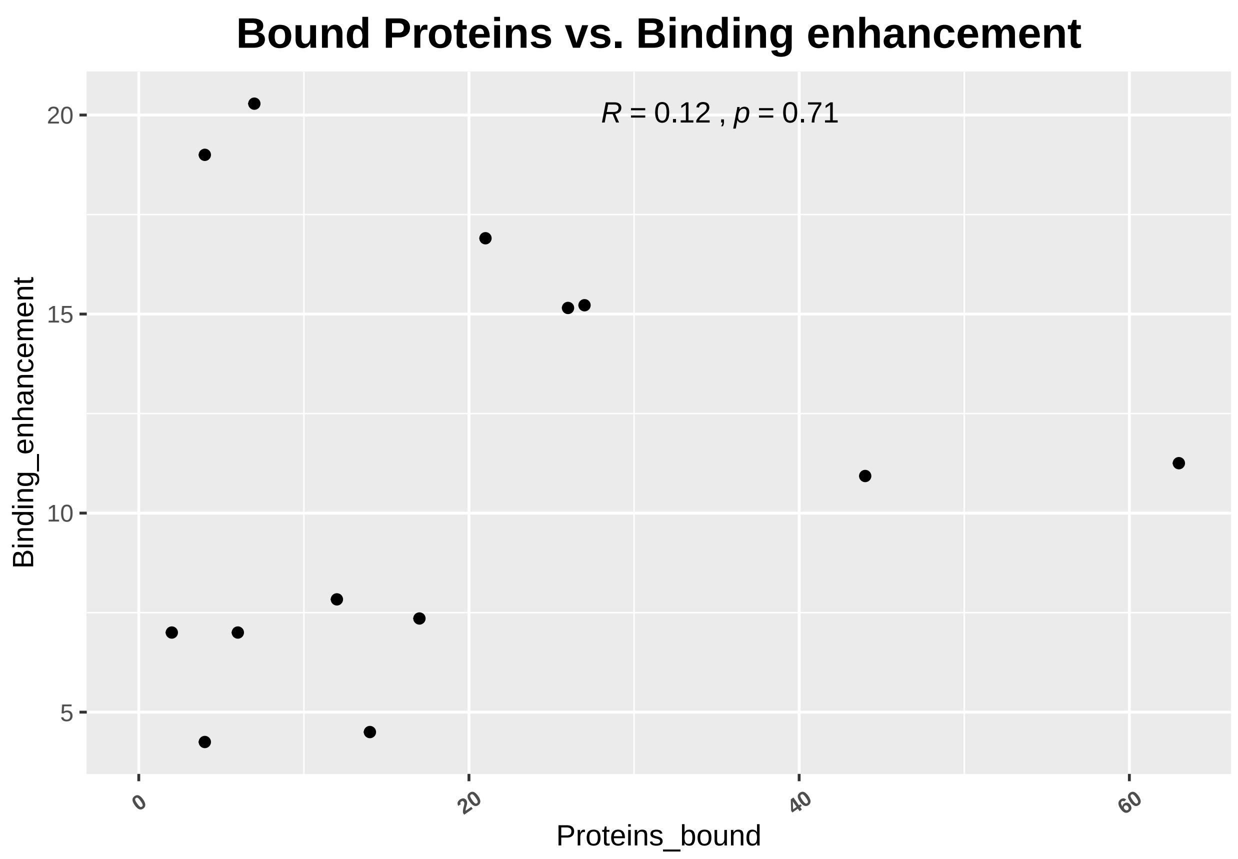


```r
##dot plot comparing frequency of binding to number of proteins bound

library(ggpubr)
```

```
## Loading required package: magrittr

##
## Attaching package: 'magrittr'

## The following object is masked from 'package:tidyr':
##
##     extract
```

```r
figure4 <- ggplot(data = overlap, aes(x = Proteins_bound, y = Binding_enhancement)) +
  geom_point() + theme(plot.title = element_text(family = "Helvetica",
                                        face = "bold", hjust = 0.5, size = 16),
                       axis.text.x = element_text(angle = 35, vjust = 0.5, size = 8, face = "bold")) +
  stat_cor(method = "pearson", label.x = 28, label.y = 20) +
  ggtitle("Bound Proteins vs. Binding enhancement")
figure4
```

```
## Warning: Removed 1 rows containing non-finite values (stat_cor).

## Warning: Removed 1 rows containing missing values (geom_point).
```



```
##this figure is just for fun, but proves there is very little correlation between the number
## of proteins bound and the strength with which they bind##


##compile all bound proteins from these pathways in order to identify proteins that appear in
##multiple pathways. First section pulls entire row of data from the original binding_partners
##dataset, while the second part stacks each pull to form a complete data frame
binding_partners1 <- binding_partners[,1:3]
glycolysis_1 <- binding_partners1[Glycolysis_numbers,]
glycolysis_1$"pathway"[1:4] = 'glycolysis'
AAMetabolism_1 <- binding_partners1[AAMetabolism_numbers,]
AAMetabolism_1$"pathway"[1:27] = "amino acid metabolism"
AATransport_1 <- binding_partners1[AATransport_numbers,]
AATransport_1$"pathway"[1:7] = "amino acid transporters"
Antigen_processing_1 <- binding_partners1[Antigen_processing_numbers,]
Antigen_processing_1$"pathway"[1:7] = "antigen processing"
Apoptosis_1 <- binding_partners1[Apoptosis_numbers,]
Apoptosis_1$"pathway"[1:44] = "Apoptosis"
ATF4_1 <- binding_partners1[ATF4_numbers,]
ATF4_1$"pathway"[1:6] = "ATF4 binding"
mTOR_1 <- binding_partners1[mTOR_numbers,]
mTOR_1$"pathway"[1:4] = "mTOR signaling"
NFkB_1 <- binding_partners1[NFkB_numbers,]
NFkB_1$"pathway"[1:26] = "NFkB signaling"
nucleosome_proteins_1 <- binding_partners1[nucleosome_numbers,]
nucleosome_proteins_1$"pathway"[1:4] = "Nucleosome formation"
```

```r
p53_indep_1 <- binding_partners1[p53_numbers,]
p53_indep_1$"pathway"[1:7] = "p53 independent DNA damage and repair"
PD1_1 <- binding_partners1[PD1_numbers,]
PD1_1$"pathway"[1:2] = "PD1 signaling"
protein_folding_1 <- binding_partners1[Protein_folding_numbers,]
protein_folding_1$"pathway"[1:14] = "protein folding proteins"
TCR_1 <- binding_partners1[TCR_numbers,]
TCR_1$"pathway"[1:12] = "TCR signaling"
UPR_1 <- binding_partners1[UPR_numbers,]
UPR_1$"pathway"[1:17] = "UPR numbers"

##compile each protein using STACK function
a <- Stack(AAMetabolism_1, AATransport_1)
b <- Stack(a, Antigen_processing_1)
c <- Stack(b, glycolysis_1)
d <- Stack(c, Apoptosis_1)
e <- Stack(d, ATF4_1)
f <- Stack(e, mTOR_1)
g <- Stack(f, NFkB_1)
h <- Stack(g, nucleosome_proteins_1)
i <- Stack(h, p53_indep_1)
j <- Stack(i, PD1_1)
k <- Stack(j, protein_folding_1)
l <- Stack(k, TCR_1)
Compiled_binding_1 <- Stack(l, UPR_1)

##put them in decreasing order based on frequency
compiled_binding_hfrq <- Compiled_binding_1[order(Compiled_binding_1$frequency, decreasing = TRUE),]

##isolate unique genes
compiled_binding_hfrq_numbers <- match(unique(compiled_binding_hfrq$gene),
                                       compiled_binding_hfrq$gene, nomatch = FALSE)
compiled_binding_hfrq <- compiled_binding_hfrq[compiled_binding_hfrq_numbers,]


duplicated_genes <- Compiled_binding_1$gene %>% duplicated()
##note: for each pair of duplicates, one is printed true and the other false
duplicated_genes <- as.data.frame(duplicated_genes)
Compiled_binding_1$duplicates[1:251] = duplicated_genes$duplicated_genes
Compiled_binding_1 <- as.data.frame(lapply(Compiled_binding_1, unlist))
Compiled_binding_1 <- Compiled_binding_1[order(Compiled_binding_1$duplicates),]
Compiled_duplicates <- Compiled_binding_1[152:251,]
##note: genes with duplicates remaining in this list have multiple duplicates

##print tables displaying diplicated genes, highest frequency at the top, lowest at the bottom
Compiled_duplicates_unique <- Compiled_duplicates[order(Compiled_duplicates$frequency, decreasing = TRU
view(Compiled_duplicates_unique)
unique_numbers <- match(unique(Compiled_duplicates_unique$gene),
                        Compiled_duplicates_unique$gene, nomatch = FALSE)

Compiled_duplicates_unique1 <- Compiled_duplicates_unique[unique_numbers,]
Compiled_duplicates_unique <- Compiled_duplicates_unique1[order(Compiled_duplicates_unique1$frequency,
                                                              decreasing = TRUE),]
```

```
##make list not including proteosome proteins bc proteosome just means its getting degraded##

Compiled_duplicates_unique_noproteosome <- filter(Compiled_duplicates_unique, family != "Proteasome")

##Have figure1,2,3,4 compiled_binding_hfrq (highest frequency genes in binding set), compiled_duplicate
##(list of highest scoring frequency duplicates),
##list of duplicates without proteasome##

library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
##FINAL FIGURES
##"figure1", "figure2", "figure3", "figure4"
##"Table1", "Table2", "Table3"
##Compile tables, remove row names##
##Table1 = highest frequency unique bound proteins
##Compile tables, remove row names##
##Table1 = highest frequency unique bound proteins

compiled_binding_hfrq_plrank <- compiled_binding_hfrq[1:20 ,1:4]
rownames(compiled_binding_hfrq_plrank) <- NULL
compiled_binding_hfrq_plrank <-
  add_column(compiled_binding_hfrq_plrank, 1:20, .before = "gene")
colnames(compiled_binding_hfrq_plrank) <- c("rank", "gene",
                                                        "family", "frequency",
                                                        "pathway")
Table1 <- kable(compiled_binding_hfrq_plrank,
      caption = "Top bound proteins") %>%
  kable_styling() %>% add_header_above(c("Top Bound Proteins" = 5),
                                       font_size = 18)

##Table2 = highest frequency proteins bound that have at least one duplicate
## + proteasome proteins
compiled_duplicates_unique_mindup_plrank <- Compiled_duplicates_unique[1:20 ,1:4]
row.names(compiled_duplicates_unique_mindup_plrank) <- NULL
```

Table 1: Top bound proteins

Top Bound Proteins				
rank	gene	family	frequency	pathway
1	EIF4B	RNA binding motif containing	47	mTOR signaling
2	SEC31A	WD repeat domain containing	35	UPR numbers
3	GLS	Ankyrin repeat domain containing	29	amino acid metabolism
4	ASB7	Ankyrin repeat domain containing	29	antigen processing
5	NFKB1	Ankyrin repeat domain containing	29	NFkB signaling
6	SPTAN1	EF-hand domain containing	28	Apoptosis
7	PPP3R1	EF-hand domain containing	28	Apoptosis
8	RPS27A	S ribosomal proteins	25	antigen processing
9	RPS6	S ribosomal proteins	25	mTOR signaling
10	SLC7A5	CD molecules	22	amino acid transporters
11	PTPRC	CD molecules	22	TCR signaling
12	RNF25	Ring finger proteins	21	antigen processing
13	DTX3L	Ring finger proteins	21	antigen processing
14	TRIM21	Ring finger proteins	21	antigen processing
15	RBCK1	Ring finger proteins	21	antigen processing
16	RBX1	Ring finger proteins	21	antigen processing
17	CBLB	Ring finger proteins	21	antigen processing
18	SLC25A10	Solute carriers	20	amino acid metabolism
19	SLC38A1	Solute carriers	20	amino acid transporters
20	SLC43A1	Solute carriers	20	amino acid transporters

```r
compiled_duplicates_unique_mindup_plrank <-
  add_column(compiled_duplicates_unique_mindup_plrank, 1:20, .before = "gene")
colnames(compiled_duplicates_unique_mindup_plrank) <- c("rank", "gene",
                                                          "family", "frequency",
                                                          "pathway")
Table2 <- kable(compiled_duplicates_unique_mindup_plrank,
      caption = "Top bound proteins in at least two pathways") %>%
  kable_styling() %>% add_header_above(c("Top Bound Proteins in 2 or more Pathways" = 5),
                                        font_size = 18)

##Table3 = highest frequency proteins bound that have at least one duplicate
## - proteasome proteins, because most bound to proteosome related proteins are likely
## just being degraded
compiled_duplicates_noproteosome_minusduplicates_plusrank <-
  Compiled_duplicates_unique_noproteosome[1:20,1:4]
compiled_duplicates_noproteosome_minusduplicates_plusrank <-
  add_column(compiled_duplicates_noproteosome_minusduplicates_plusrank, 1:20, .before = "gene")
colnames(compiled_duplicates_noproteosome_minusduplicates_plusrank) <- c("rank", "gene",
                                                          "family", "frequency",
                                                          "pathway")
Table3 <- kable(compiled_duplicates_noproteosome_minusduplicates_plusrank,
      caption = "Top bound proteins excluding proteasome bound proteins") %>%
  kable_styling() %>% add_header_above(c("Top Bound Proteins in 2 or more Pathways
                                          (Excluding Proteasome)" = 5), font_size = 18)

Table1
```

Table 2: Top bound proteins in at least two pathways

Top Bound Proteins in 2 or more Pathways				
rank	gene	family	frequency	pathway
1	NFKB1	Ankyrin repeat domain containing	29	TCR signaling
2	RPS27A	S ribosomal proteins	25	Apoptosis
3	SLC25A10	Solute carriers	20	amino acid transporters
4	PSME2	Proteasome	17	antigen processing
5	PSMD7	Proteasome	17	antigen processing
6	PSMA7	Proteasome	17	antigen processing
7	PSMB1	Proteasome	17	antigen processing
8	PSMB3	Proteasome	17	antigen processing
9	PSMD2	Proteasome	17	antigen processing
10	PSMB4	Proteasome	17	antigen processing
11	PSMB2	Proteasome	17	antigen processing
12	PSMA6	Proteasome	17	antigen processing
13	PSMB8	Proteasome	17	antigen processing
14	PSMB9	Proteasome	17	antigen processing
15	PSMA2	Proteasome	17	antigen processing
16	PSMB7	Proteasome	17	antigen processing
17	PSMD8	Proteasome	17	antigen processing
18	PSMB6	Proteasome	17	antigen processing
19	PSMB5	Proteasome	17	antigen processing
20	PSMD4	Proteasome	17	antigen processing

```
Table2
```

```
Table3
```

```
##FINAL FIGURES
##"figure1", "figure2", "figure3", "figure4"
##"Table1", "Table2", "Table3"
##Compile tables, remove row names##
##Table1 = Table shows the most enriched bound proteins (compared to wild type)
##Table2 = Table shows the most enriched bound proteins (compared to wild type) which
##appear in at least 2 different pathways that i analyzed
##Table3 = Table shows the most enriched bound proteins (compared to WT) which appear in at
##least two pathways and excludes all proteasome bound proteins since these are possibly ust
##being sent to the proteasome and degraded
##figure1 = Figure showing the number of affected genes and their protein products in each pathway
##Figure2 = Figure showing the average frequency score of each pathway
##Figure3 = Figure showing the percent of each pathway that mutant CALR is binding -- i.e. if
##the value on this graph is 100, then 100 percent of proteins in that pathway are being
##bound by mutant CALR
##figure4 = dot plot showing the correlation (more accurately the lack of correlation)
##between frequency of binding and number of proteins bound in each pathway
```

Table 3: Top bound proteins excluding proteasome bound proteins

Top Bound Proteins in 2 or more Pathways (Excluding Proteasome)				
rank	gene	family	frequency	pathway
1	NFKB1	Ankyrin repeat domain containing	29	TCR signaling
2	RPS27A	S ribosomal proteins	25	Apoptosis
3	SLC25A10	Solute carriers	20	amino acid transporters
4	PSMC4	AAA ATPases	15	antigen processing
5	PSMC5	AAA ATPases	15	antigen processing
6	PSMD9	PDZ domain containing	11	antigen processing
7	UBE2N	Ubiquitin conjugating enzymes E2	9	TCR signaling
8	CSK	SH2 domain containing	8	TCR signaling
9	EXOSC6	Exosome complex	7	UPR numbers
10	EXOSC3	Exosome complex	7	UPR numbers
11	DIS3	Exosome complex	7	UPR numbers
12	EXOSC7	Exosome complex	7	UPR numbers
13	EXOSC2	Exosome complex	7	UPR numbers
14	EXOSC4	Exosome complex	7	UPR numbers
15	CUL1	Cullins	6	NFkB signaling
16	CUL7	Cullins	6	UPR numbers
17	FBXO6	F-boxes other	2	protein folding proteins
18	MAP3K7	Mitogen-activated protein kinase kinase kinases	2	TCR signaling
19	LMNA	Lamins	2	UPR numbers
20	SKP1	SCF complex	1	NFkB signaling