RBPs and ER, Hypoxia and Unfolded Protein Response

Manasa Ramakrishna, Robert Harvey

26 June, 2019

Startup

We start by installing and loading the libraries required for our analysis. Additionally, tell R where you are running your program by setting your working directory as shown below using the variable 'wd'. We will use this later on. Also make your input and output directories (indir/outdir) as shown below.

```
suppressMessages(library(reshape2))
suppressMessages(library(ggplot2))
suppressMessages(library(ggsci))
suppressMessages(library(dplyr))
suppressMessages(library(MSnbase))
suppressMessages(library(ggbiplot))
suppressMessages(library(pRoloc))
suppressWarnings(library(mygene))
suppressWarnings(library(data.table))
suppressWarnings(library(patchwork))

# Setting working directories Note: Change the next
# line of card to point to your working directory
wd = "~/Documents/Work/TTT/15_UPR-Hypoxia-RHarvey/"
setwd(wd)
# getwd()

# Declaring input and output directories
indir = paste(wd, "Input", sep = "/")
outdir = paste(wd, "Output", sep = "/")
plotdir = paste(wd, "Plots", sep = "/")

# If output and plots directory exist, clear them
# out and start afresh
if (exists(outdir)) {
    system(paste0("rm -r", outdir))
}
if (exists(plotdir)) {
    system(paste0("rm -r", plotdir))
}

dir.create(outdir)
dir.create(plotdir)
```

Introduction

The main aims of this analysis are to identify and classify known RBPs into those that are involved in Hypoxia, Endoplasmic Reticulim and also in eliciting an Unfolded Protein Respose (UPR). To do this, I will be using two sets of proteins as known RBPs - (1) From Trendel et al that have used RBP data from RNA Interactome Capture in HeLa, HEK293 and MCF7 cells (2) List of RBPs from Queiroz et al that have used Trizol -based OOPS to target RBPs.

01. Reading in RNA Interactome Capture Data

I have obtained these data and stored them in text files in the "Input" folder and will bbe reading them and reformatting them for further analysis in this section.

```
# 1. List of known RBPs across cell lines in the
# XRNAX paper (Table S2)
xrnax = read.delim(paste(indir, "xrnax-genelist.txt",
    sep = "/"), sep = "\t", header = T)

# Check how many are common to the cell lines in
# the XRNAX paper
xrnax %>% dplyr::select(MCF7.RBP:ihRBP) %>% apply(2,
    table, useNA = "always")
```

```
##             MCF7.RBP HEK293.RBP HeLa.RBP ihRBP
## non-poly(A)      617        698      565   775
## poly(A)          590        659      674   978
## <NA>            1276       1126     1244   730
```

```
# Filter to only keep those that have been found by
# at least one cell line
xrnax.rbps = xrnax %>% dplyr::filter(!is.na(MCF7.RBP) |
    !is.na(HEK293.RBP) | !is.na(HeLa.RBP)) %>% dplyr::select(c(Uniprot.ID:Protein.name,
    MCF7.RBP:ihRBP))
# rownames(xrnax.rbps) = xrnax.rbps$Uniprot.ID
```

Note that in the protein lists, proteins have been classified as those that bind polyA RNA and those that bind non-polyA RNA. For now, i will leave all of them in but can change this later on.

1a. Reading in OOPS-based RBP data

```
# 2. List of RBPs from SILAC experiments using OOPS
# with 2 or more wash steps and CL/NC Ratio >1 i.e
# more than two-fold change in abundance (Table S1)
oops = read.delim(paste(indir, "oops-genelist.txt",
    sep = "/"), sep = "\t", header = T, row.names = NULL)
oops.rbps = oops %>% dplyr::filter(step > 1 & CL_NC_Ratio >=
    1) %>% dplyr::select(master_protein, step, RBP_glyco) %>%
    dplyr::filter(RBP_glyco != "Novel RBP, Glycoprotein") %>%
    dplyr::distinct(master_protein, .keep_all = TRUE)
rownames(oops.rbps) = oops.rbps$master_protein

# Merge both lists n = 2807
merged.rbp = union(xrnax$Uniprot.ID, oops$master_protein)
```

```
write.table(data.frame(merged.rbp), paste(outdir, "RBP-list-for-uniprot.txt",
    sep = "/"), sep = "\t", row.names = F, quote = F)
```

1b. Subset proteins of interest from RBP list

Here we are focussing on those RBPs that have a GO term relating them to UPR, Hypoxia or Endoplasmic Reticulim.

```
# Annotate OOPS/RIC RBPs with GO terms from Uniprot
go.rbp = read.delim(paste(indir, "human-rbp-uniprot.txt",
    sep = "/"), sep = "\t", stringsAsFactors = F, header = T)
colnames(go.rbp)[6] = "Gene.name"

# Hypoxia response
hypox = go.rbp[with(go.rbp, grepl("hypoxia", paste(Gene.ontology..biological.process.,
    Gene.ontology..cellular.component., Gene.ontology..molecular.function.))),
    ]

er = go.rbp[with(go.rbp, grepl("endoplasmic reticulum",
    paste(Gene.ontology..biological.process., Gene.ontology..cellular.component.,
        Gene.ontology..molecular.function.))), ]

upr = go.rbp[with(go.rbp, grepl("unfolded protein",
    paste(Gene.ontology..biological.process., Gene.ontology..cellular.component.,
        Gene.ontology..molecular.function.))), ]

colnames(upr)[6] = colnames(er)[6] = colnames(hypox)[6] = "Gene.name"

print(length(upr$Gene.name))
```

```
## [1] 91
```

```
print(length(hypox$Gene.name))
```

```
## [1] 63
```

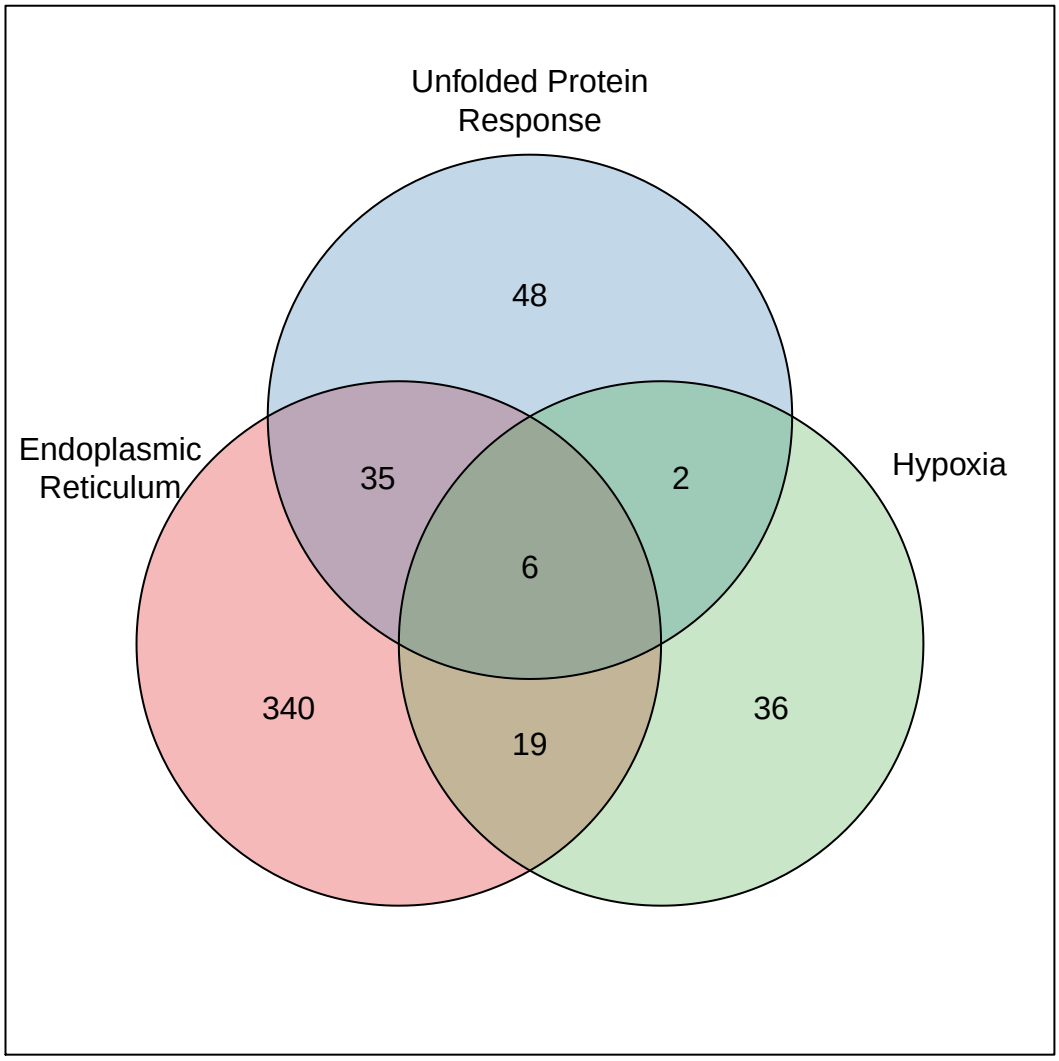
```
print(length(er$Gene.name))
```

```
## [1] 400
```

```
# Plot intersect
library(venn)
library(gplots)
library(RColorBrewer)
input.v = list(`Endoplasmic\nReticulum` = er$Gene.name,
    `Unfolded Protein\nResponse` = upr$Gene.name, Hypoxia = hypox$Gene.name)
pdf(paste(plotdir, "UPR-Hypoxia-ER-RBPs-Venn.pdf",
    sep = "/"))
v = venn::venn(input.v, zcolor = brewer.pal(n = 3,
    name = "Set1"), cexil = 1, cexsn = 1)
dev.off()
```

```
## pdf
##   2
```

```
venn::venn(input.v, zcolor = brewer.pal(n = 3, name = "Set1"),
    cexil = 1, cexsn = 1)
```



This brings us to the end of the first part of the analysis which was all about known RBPs and how many of those are involved in the Unfolded Protein Response.

02. Cancer Cell Fitness genes

The analysis now focuses on the paper that uses CRISPR screens to identify genes that are required for cancer cell fitness. Based on Supplementary table S2 in this paper, we want to find
a. The RBPs that are required for cancer cell fitness
b. Find out how many of these RBPs are involved in stress response pathways – UPR and hypoxia
c. And how many are RNA modifying enzymes?

2a. Cancer fitness genes

The data are presented per cell-line (columns) and per gene (rows) in supplementary table S2 of the paper. There are 7470 fitness genes across 326 cell lines. The table also comes with metadata classifying each cell

line into its tissue type. We need to extract this into a usable format before being able to comment on the occurence of fitness RBPs by cancer type.

We extract metadata information into "metadat" and fitness information into the object "fitness" in the code below. These two dataframes will be used for downstream analysis.

```
# HT-29 was not classified in the Tissue column so
# did this manually Fitness dataframe
fitness <- read.delim(paste(indir, "Cancer-Fitness-S2.txt",
    sep = "/"), sep = "\t", header = F, stringsAsFactors = F)
dim(fitness)
```

```
## [1] 7474  326
```

```
colnames(fitness) = fitness[4, ]
rownames(fitness) = fitness$Gene.CellLine

# Metadata
metadat = data.frame(t(fitness[1:4, 2:ncol(fitness)]))
colnames(metadat) = c("Tissue", "Cancer.Type", "CMP.id",
    "Cell.line")

# Resize 'fitness' dataframe and only keep RBP rows
fitness = fitness[5:nrow(fitness), ]
dim(fitness)
```

```
## [1] 7470  326
```

```
# Reshape
all.fit = reshape2::melt(fitness, id = "Gene.CellLine",
    na.rm = T)
colnames(all.fit) = c("Gene", "Cell.line", "Fitness")
all.fit = left_join(all.fit, metadat)
head(all.fit)
```

```
##      Gene Cell.line Fitness   Tissue        Cancer.Type    CMP.id
## 1    A1BG     22RV1       0 Prostate Prostate Carcinoma SIDM00499
## 2    A1CF     22RV1       0 Prostate Prostate Carcinoma SIDM00499
## 3    AAAS     22RV1       0 Prostate Prostate Carcinoma SIDM00499
## 4    AACS     22RV1       0 Prostate Prostate Carcinoma SIDM00499
## 5 AADACL2     22RV1       0 Prostate Prostate Carcinoma SIDM00499
## 6   AADAT     22RV1       0 Prostate Prostate Carcinoma SIDM00499
```

2b. Cancer fitness and RBP

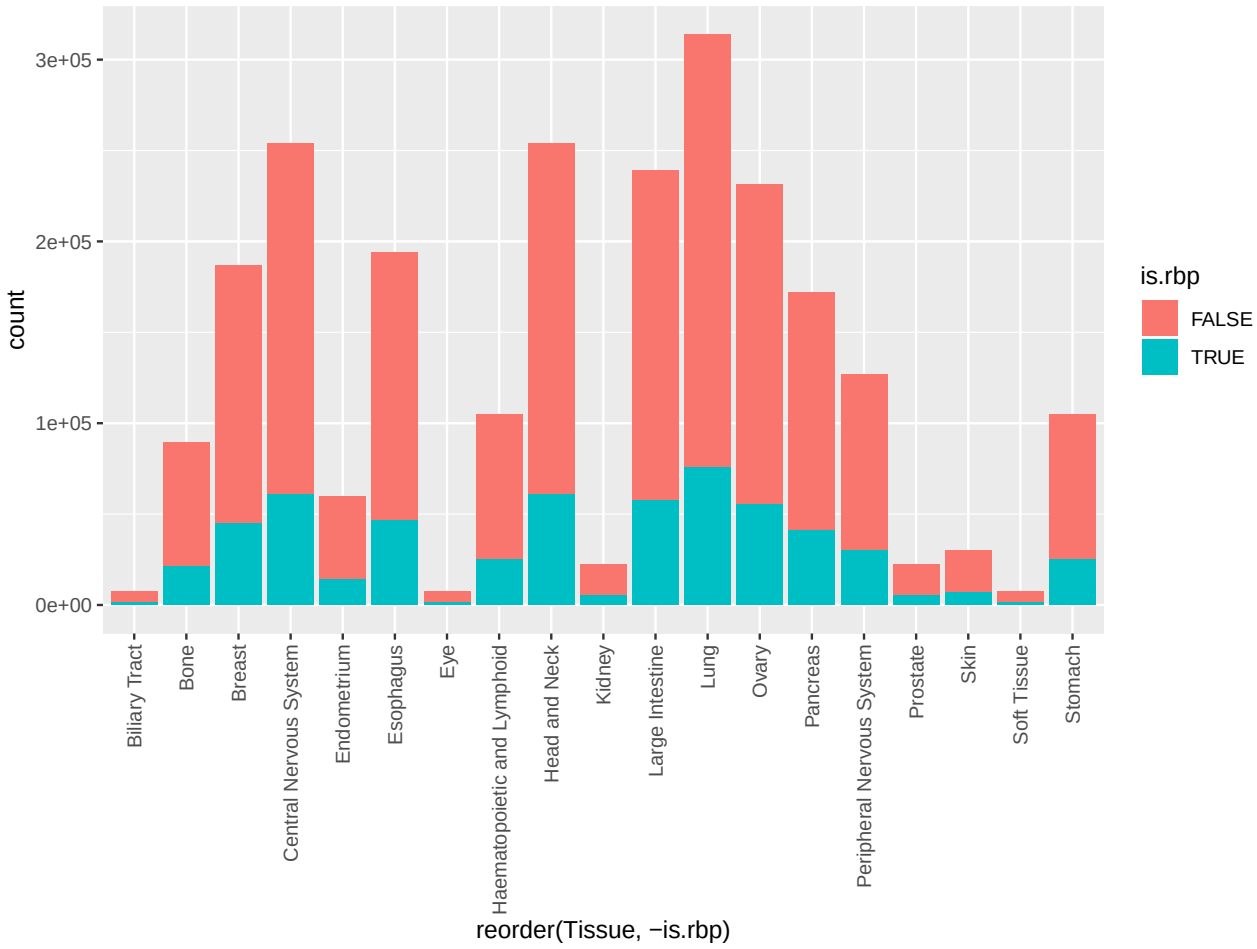
The most straightforward way of workng out how many fitness genes are RBPs is to merge our list from Section 01 above to the fitness list which is what we do below.

```
# Add RBP information
all.fit = left_join(all.fit, go.rbp, by = c(Gene = "Gene.name"))
all.fit$is.rbp = TRUE
all.fit$is.rbp[which(is.na(all.fit$Entry))] = FALSE
head(all.fit)
```

```
##      Gene Cell.line Fitness   Tissue        Cancer.Type    CMP.id  Entry
## 1    A1BG     22RV1       0 Prostate Prostate Carcinoma SIDM00499   <NA>
## 2    A1CF     22RV1       0 Prostate Prostate Carcinoma SIDM00499   <NA>
```

```
## 3    AAAS     22RV1       0 Prostate Prostate Carcinoma SIDM00499 Q9NRG9
## 4    AACS     22RV1       0 Prostate Prostate Carcinoma SIDM00499   <NA>
## 5 AADACL2     22RV1       0 Prostate Prostate Carcinoma SIDM00499   <NA>
## 6   AADAT     22RV1       0 Prostate Prostate Carcinoma SIDM00499   <NA>
##   Entry.name      Protein.names             Organism Length
## 1       <NA>               <NA>                 <NA>     NA
## 2       <NA>               <NA>                 <NA>     NA
## 3 AAAS_HUMAN Aladin (Adracalin) Homo sapiens (Human)    546
## 4       <NA>               <NA>                 <NA>     NA
## 5       <NA>               <NA>                 <NA>     NA
## 6       <NA>               <NA>                 <NA>     NA
##
## 1
## 2
## 3 centrosome [GO:0005813]; cytosol [GO:0005829]; membrane [GO:0016020]; nuclear envelope [GO:0005635]
## 4
## 5
## 6
##
## 1
## 2
## 3 fertilization [GO:0009566]; learning [GO:0007612]; mRNA export from nucleus [GO:0006406]; nucleocy
## 4
## 5
## 6
##   Gene.ontology..molecular.function. is.rbp
## 1                               <NA>  FALSE
## 2                               <NA>  FALSE
## 3                                      TRUE
## 4                               <NA>  FALSE
## 5                               <NA>  FALSE
## 6                               <NA>  FALSE
```

```r
# Plot distribution of RBPs by Tissue
ggplot(all.fit, aes(x = reorder(Tissue, -is.rbp), fill = is.rbp)) +
    geom_bar() + theme(axis.text.x = element_text(angle = 90,
    vjust = 0.5, hjust = 1))
```



03. Cancer fitness and RBPs

Our first task is to look at cancer fitness genes that are also known RBPs. We re-use the 'go.rbp' dataframe from the previous sections which consist of 2807 currently published RBPs (from interactome capture and OOPS) annotated with GO terms. Furthermore, we want to see if the number of RBPs involved in cancer fitness vary by cancer type or if there is a core set that is crucial to the survival of many cancer types.

3a. RBPs by cancer

We want to see how many fitness-RBPs there are per cancer type and also perhaps per tissue type.

```
# There are 2807 proteins that are RBPs in the
fit.rbp = fitness[go.rbp$Gene.name, ]
dim(fit.rbp)
```

```
## [1] 2807  326
```

```
# Reshape the fitness dataframe
library(tidyr)
fit.g = reshape2::melt(fit.rbp, id = "Gene.CellLine",
    na.rm = T)
colnames(fit.g) = c("Gene", "Cell.line", "Fitness")
```

```
# RBP stats by fitness and cell-line
cell.fit.rbp.tab = round(table(fit.g$Cell.line, fit.g$Fitness) *
    100/rowSums(table(fit.g$Cell.line, fit.g$Fitness)),
    2)

# RBP stats by fitness and cell-line
fit.g = left_join(fit.g, metadat)
tis.fit.rbp.tab = round(table(fit.g$Tissue, fit.g$Fitness) *
    100/rowSums(table(fit.g$Tissue, fit.g$Fitness)),
    2)
ctype.fit.rbp.tab = round(table(fit.g$Cancer.Type,
    fit.g$Fitness) * 100/rowSums(table(fit.g$Cancer.Type,
    fit.g$Fitness)), 2)
```

While looking at only those genes that are in the known.rbp list and in the Cancer Fitness study, we have an overlap of 2807. On a cancer type level, the range of these RBPs that are involved in core-fitness is between 25 and 45%. On a tissue level, the range of these RBPs that are involved in core-fitness is between 30 and 40%. At a cell-line level between 10 and 50% are core fitness genes.

We now want to turn the problem on its head and ask that if we were to just pull out the fitness genes, what proportion are RBPs at the tissue, cancer type and cell line level.

```
core.fit = reshape2::melt(fitness, id = "Gene.CellLine",
    na.rm = T)
colnames(core.fit) = c("Gene", "Cell.line", "Fitness")
core.fit = core.fit[which(core.fit$Fitness == 1), ]

# Add RBP information and metadata information to
# core.fit
core.fit = left_join(core.fit, metadat)
core.fit$is.rbp = FALSE
core.fit$is.rbp[which(core.fit$Gene %in% go.rbp$Gene.name)] = TRUE

# Ggplots
core.fit$Tissue.2 = core.fit$Tissue
core.fit$Tissue.2 = factor(core.fit$Tissue.2, levels = names(sort(table(core.fit$Tissue))))
head(core.fit)

# Calculate percentages
core.fit1 <- core.fit %>% dplyr::filter(Fitness ==
    1) %>% dplyr::group_by(Tissue.2, is.rbp) %>% dplyr::summarise(count = n()) %>%
    dplyr::mutate(perc = count/sum(count)) %>% data.frame()

core.fit2 <- core.fit %>% dplyr::filter(Fitness ==
    1) %>% dplyr::group_by(Cancer.Type, is.rbp) %>%
    dplyr::summarise(count = n()) %>% dplyr::mutate(perc = count/sum(count)) %>%
    data.frame()

core.fit3 <- core.fit %>% dplyr::filter(Fitness ==
    1) %>% dplyr::group_by(Cell.line) %>% dplyr::summarise(count = n()) %>%
    dplyr::mutate(perc = count/sum(count)) %>% data.frame()

table(core.fit$Cell.line, core.fit$is.rbp)
ggplot(core.fit, aes(Tissue.2, fill = is)) + geom_bar() +
    coord_flip() + labs(x = "", y = "Percentage")
```

```
ggplot(core.fit2, aes(factor(Tissue.2), y = perc *
    100, fill = is.rbp)) + geom_bar(stat = "identity") +
    coord_flip() + labs(x = "", y = "Percentage")
ggplot(core.fit3, aes(factor(Tissue.2), y = perc *
    100, fill = is.rbp)) + geom_bar(stat = "identity") +
    coord_flip() + labs(x = "", y = "Percentage")
```